South African
Computer
Journal
Number 17
September 1996

Suid-Afrikaanse
Rekenaar-
tydskrif
Nommer 17
September 1996

Computer Science and Information Systems

Special Edition: Computer Security

Rekenaarwetenskap en Inligtingstelsels

The South African Computer Journal
An official publication of the Computer Society of South Africa and the South African Institute of Computer Scientists

Die Suid-Afrikaanse Rekenaartydskrif
'n Amptelike publikasie van die Rekenaarverenging van Suid-Afrika en die Suid-Afrikaanse Instituut vir Rekenaarwetenskaplikes

Editor
Professor Derrick G Kourie
Department of Computer Science
University of Pretoria
Hatfield 0083
Email: dkourie@dos-lan.cs.up.ac.za

Subeditor: Information Systems
Prof Lucas Introna
Department of Informatics
University of Pretoria
Hatfield 0083
Email: lintrona@econ.up.ac.za

Production Editor
Dr Riël Smit
Mosaic Software (Pty) Ltd
P.O.Box 23906
Claremont 7735
Email: gds@mosaic.co.za

World-Wide Web: http://www.mosaic.co.za/sacj/

Subscriptions

	Annual	Single copy
Southern Africa:	R50,00	R25,00
Elsewhere:	$30,00	$15,00

An additional $15 per year is charged for airmail outside Southern Africa

to be sent to:
Computer Society of South Africa
Box 1714 Halfway House 1685

Phone: +27 (11) 315-1319 Fax: +27 (11) 315-2276

Guest Editorial

Information Security – The Family Member Who Came Home

Basie von Solms
Rand Afrikaans University, Johannesburg, South Africa
basie@rkw.rau.ac.za

The claim that the information society is upon us and that the information superhighway is about to affect us all, is such a cliche that it hardly bears repeating - practically everyone is talking and writing about it. Although the assertion that information *security* is an essential and integral part of the information society might also seem cliched to some, it nevertheless does not get as much attention and exposure as it should. Information security has always been seen in the same light as taxes: nobody really wants it, but everybody (reluctantly) admits that we need it. Following an alternative analogy, it is like an unwanted family member: we realise that he is part of the family, but we really do not want to be bothered by him. In the last few years, the growth of the Internet and the explosive appearance of the World Wide Web (WWW) has brought information security into corporate boardrooms and private lounges. Suddenly everyone wants to get to know this unwanted family member a little better!

This issue of SACJ is dedicated to the unwanted family member and if, by reading this issue, all of us in the IT family are able to learn just a little more about this sibling, then producing the issue would have been worthwhile. In fact, we should welcome him back home as soon as possible.

A little background about this issue is in order. The International Federation for Information Processing (IFIP) is a consortium of about 60 member countries. It provides an umbrella for many international IT activities. Countries are represented by their national IT society, South Africa being represented by the Computer Society of South Africa (CSSA). IFIP has a total of 13 Technical Committees (TCs), each concentrating on a different aspect of IT. TC 11 deals with all aspects of information security. It organises an annual international conference – the so-called IFIP/Sec series, which is widely regarded as one of the major information security conferences.

IFIP/Sec 95, the 11th International Conference on Information Security, took place in Cape Town in May 1995 and IFIP/Sec 96, the 12th International Conference on Information Security, took place on the Greek island of Samos in May 1996. Complete proceedings of the two conferences, together containing more than 80 articles, have been published by the official IFIP publishers, Chapman and Hall. This issue of SACJ consists of a selection of four articles from the IFIP/Sec 95 Proceedings and two from the IFIP/Sec 96 Proceedings. It is intended not only to disseminate relevant information, but also to bring the information security interests of SAICSIT, CSSA, IFIP and TC 11 to the attention of readers. If, after reading this issue, you are interested in the remaining 74 articles, or in the activities of any of these bodies, please feel free to contact the guest editor directly.

The selected articles cover a diverse range of information security issues. Parker extends its theoretical and conceptual understanding, Hoffman addresses several of its non-technical but crucial aspects, Muftic concentrates on its role in open distributed systems, de Ru and Eloff link into the use of biometrics in information security, von Solms investigates the security protocols for the Internet and WWW, and Pangalos and Khair remind us that information security requirements extend even into the medical field.

The first four articles are from the IFIP/Sec 95 Proceedings. The first, by Donn Parker, suggests a framework for information security in order to avoid information anarchy. He argues that the traditional view of the role of information security – to protect the three elements of confidentiality, integrity and availability of information – is dangerously oversimplified. He includes three more elements of information into the equation: authenticity, utility and possession. The article provides a good platform for gaining a better understanding of what information security really ought to be about.

In the second article, Lance Hoffman addresses the important issues of escrow encryption and export controls – specifically, as he clearly points out, from a US standpoint. The article highlights the very important fact that cryptology is not merely a technical issue, but that the political overtones, civil liberties and administrative implications are also extremely relevant. Ignoring these non-technical issues, as many information security specialists tend to do, will have a negative impact on the field becoming a discipline in its own right. Since its first publication, some of the issues raised by the Hoffman article have been receiving attention. For example, the idea of international cryptology policies is being addressed by the European Organisation for Economic Cooperation and Development (OECD). Relevant recent articles on the escrow aspect can be found in [1].

In the next article, Sead Muftic concentrates on aspects of security in open distributed environments. He identifies a number of elements suitable for a secure system in such an environment. These are: smart cards, secure user workstations, integrated security clients, security servers and a global certification system for international networks. This last aspect is becoming more and more important as basically all secure protocols use public key encryption in some form or the other. Using these elements, he also describes a number of security enhanced applications – secure Internet e-mail and secure EDI. Muftic stresses the fact that all these elements are operational, implemented, and already in use.

de Ru and Eloff then discuss the reinforcement of password authentication using typing biometrics. Biometric methods are probably the best means of authentication, but many of these methods are technology-intensive and expensive. Their article tries to use typing characteristics as a cheap and user-transparent way to augment the traditional password.

The last two articles are from the IFIP/Sec 96 Proceedings. von Solm's article gives an overview of two non-payment related and one payment related secure protocol for the Internet and the WWW. The non-payment related protocols are Secure Sockets Layer (SSL) and Secure Hypertext Transport Protocol (SHTTP). SSL is usable in any TCP/IP environment, while SHTTP is specifically for the WWW. Both make use of public key encryption. The payment related protocol, Secure Payment Protocol (SEPP) was superseded by the Secure Transaction Protocol (SET) early in 1996, but only after the article had already been submitted to IFIP/Sec 96. The presentation at the conference however, covered SET and not SEPP. An appendix is attached to this article, giving a brief overview of SET, as SEPP is no longer relevant. SET also uses public key encryption.

The last article by Pangalos and Khair introduces a methodology to improve the security of medical databases in relation to authentication. Though the methodology of the authors is in itself important and interesting, the article was selected for this issue to underline the importance and relevance of information security in medical IT applications – an area where security introduces new problems quite distinct from those traditionally encountered in the financial and other fields. Becuase of the differences, information security in medical applications still requires much research.

It is hoped that readers will find this issue of SACJ useful, and also that they will get involved with the activities of the bodies mentioned above.

References

1. *Communications of the ACM*, **39**(3):33–53, (March 1996).
2. http://www.visa.com.

Appendix A

Secure Electronic Protocol (SET) [2] is the secure payment protocol announced by MasterCard and Visa in February 1996. The two previous protocols, SEPP and STT, announced independantly by these two companies, were replaced by SET.

This Appendix gives a brief and oversimplified overview of the SET message flow between the customer and the merchant when a payment is made.

Only the most basic parts of messages are discussed, and many details are left out for the sake of simplicity.

The electronic purchasing process can basically be divided into two phases. Phase 1 is a browsing and negotiation phase in which the customer will decide what to buy. Goods and price will be negotiated and agreed upon between the customer and merchant. The phase will probably end with a completed order form, specifying the goods,with the associated price, the customer is about to buy.

At this point the customer decides to start Phase 2, the payment phase. It is at this point that the SET payment protocol is initiated.

Step 1:

A message is sent from the customer to the merchant, requesting, amongst other things, the public key certificates of the merchant and the merchant's acquirer.

Step 2:

The merchant sends these certificates back to the customer.

Step 3:

- The customer validates the certificates received from the merchant.
- The customer constructs the Order Information (OI).

  OI contains info about the goods and negotiated price.
- The customer generates a DES key K1 and encrypts OI under K1, giving K1(OI).
- The customer encrypts K1 using the public key of the merchant, giving MP(K1).
- The customer constructs the hash of the OI, giving H(OI).
- The customer constructs the Payment Information (PI). The PI contains info about the credit card nr, expiry date etc.
- The customer generates a DES key K2 and encrypts PI under K2, giving K2(PI).
- The customer encrypts K2 using the public key of the acquirer, giving AP(K2).
- The customer constructs the hash of the PI, giving H(PI).
- The customer constructs the hash of OI, the hash of PI and concatenates them, giving H(OI)||H(PI).
- The customer digitally signs this concatenation giving S(H(OI)||H(PI)). The customer performs

this operation to uniquely associate the specific OI with the specific PI.

Step 4:

The customer sends the message M1= [K1(OI), MP(K1), H(OI), K2(PI), AP(K2), H(PI), S(H(OI)||H(PI))] to the merchant.

Step 5:

The merchant receives M1= [K1(OI), MP(K1), H(OI), K2(PI), AP(K2), H(PI), S(H(OI)||H(PI))]

- The merchant retrieves K1 and then OI.

  Note that the merchant cannot retrieve PI in any way. The customer intended it this way, because he/she does not want the merchant to see his credit card info.

- The merchant creates H(OI), concatenates it with H(PI) received in M1, and compares it with the H(OI)||H(PI) received in M1.

  Note that the merchant can retrieve H(OI)||H(PI) from S(H(OI)||H(PI)) by using the customer's public key.

  If they are equal, then the merchant knows that the OI he/she has retrieved from M1 is the 'correct' OI intended by the customer to accompany the PI provided by the customer.

  Note that this process allows the merchant to associate a specific OI with a specific PI without knowing precisely what PI is.

Step 6:

The merchant now requests an authorisation from his acquirer, by sending the following message M2 to the acquirer: [H(OI), K2(PI), AP(K2), H(PI), S(H(OI)||H(PI))]

- The acquirer goes vica versa through the same procedure as the merchant, allowing the acquirer to associate the specific PI with the specific OI without knowing precisely what OI is.

  The customer does not want the acquirer to see what he/she is buying, but wants the acquirer to link the OI to the PI.

Step 7:

If necessary, the acquirer gets authorisation from the issuer, and send a digitally signed authorisation back to the merchant.

Step 8:

The merchant informs the customer that the transaction is authorised.

SACJ is produced with kind support from
Mosaic Software (Pty) Ltd.

Information Security on the Electronic Superhighway

S H von Solms
Rand Afrikaans University and IBM South Africa, PO Box 524, Aucklandpark, 2006, Johannesburg, South Africa
basie@rkw.rau.ac.za

Abstract

This paper discusses a number of security protocols for the Internet and World Wide Web. The first two, SSL and SHTTP, provide general authentication, confidentiality and integrity facilities, while SEPP is a payment protocol, specifically designed for use in electronic purchasing.
Keywords: *Security protocols, Internet, World Wide Web, SSL, SHTTP, SEPP*
Computing Review Categories: *D.4.6, K.6.5*

1 Introduction

Whichever paper you read or person you speak to, there is little doubt that the overwhelming consensus is that any company not planning to utilise the Internet/WWW infrastructure on a corporate level, is going to lose out.

These same papers and people, however, explicitly warns about the information security risks of doing precisely what they suggest!

There are different routes a company can follow to expose itself to the Internet, but in general, there seems to be an identifiable pattern. Usually the following major phases can be identified, not necessarily in the order suggested below :

Phase 1: Allow employees to retrieve information from the Net and WWW. Clients in the company therefore have access to WWW servers, but the company provides no servers accessible by clients outside the company.

Phase 2: Start using the Net and WWW to provide information about company products and prices – including confidential information on a selective basis to specific customers. The company now provide a server facility which can be accessed by clients outside the company. This is often the beginning of Internet use for marketing

Phase 3: Start using the Net and WWW for electronic purchasing in a 'closed' environment, where the customer must register before the time, and transactions are only performed with registered customers.

Phase 4: 'Open' electronic purchasing, where transactions can be performed with any customer presenting a credit card number.

This paper is very much tutorial in nature, and intends to address some of the risks involved in these phases, providing some solutions, currently available. By no means can all possible risks and exposures be covered, and we will mainly concentrate on security solutions relevant to Phases 2 and 4, discussing a number of protocols which seem to be significant players in this area.

We will discuss each of the phases separately.

2 Phase 1

This is basically a 'one-way' traffic, in that information is 'brought into' the company, but no information is provided to outsiders, so no information flows out of the company's systems to the outside world. Usually at this stage, some type of firewall is installed. Firewalls come in many different flavours, and we will not discuss the concept of firewalls any further, apart from saying that a firewall is only as good as the way it is managed.

3 Phase 2

At this stage, the environment starts opening up further, in that company data is now being moved from a company server to the outside world. The risks start to increase because outside contamination of the company's resources, as well as the possibility of providing confidential data to unauthorised outsiders, become possible.

In the next 4 sections, we will discuss solutions relevant to Phase 2, as defined above.

At this point in time, the following facilities are starting to become essential :

F.1 The facility to authenticate the parties involved in a transaction, i.e. the user (client) is sure he is talking to a legal system (server), and vica versa.

F.2 The facility to protect and secure data sent between the client and server, i.e. to maintain confidentiality.

F.3 The facility to prevent replay of transactions.

F.4 The facility to ensure that data is not changed during transmission, i.e. to maintain integrity.

F.5 The facility to digitally sign messages, i.e. to enforce non-repudiability.

Now identification and authentication of the parties involved in a transaction become essential, as well as the confidentiality and integrity of the data during transmission. Solutions will therefore be more complex and difficult, but must always as transparent to the user as possible.

Two widely accepted technologies which are quite useful at this point in time, and provides some of the facilities

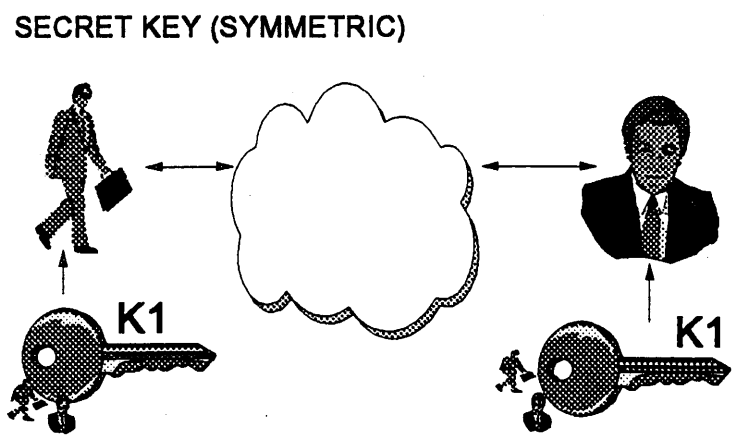

Figure 1. Symmetric encryption.

mentioned above, are Secure Sockets Layer (SSL) [3], and Secure Hypertext Transport Protocol (SHTTP) [2].

Because both technologies make extensive use of digital signatures, a brief overview of such signatures are in order. A more detailed discussion can be found in many documents on encryption technology, e.g. [5].

4 Digital Signatures

In the traditional symmetric approach to encryption, two parties involved in exchanging secure messages, secure such messages by encrypting the content of the message using a shared algorithm and a shared key. This shared key is used for encrypting and decrypting the message. Figure 1 illustrates this.

In the asymmetric or public key approach to encryption, each party involved in exchanging secure messages, has a unique pair of related keys. One of the keys in the pair is called the party's public key, and the other his private key. The public key is publicised as wide as possible, for e.g. in directories accessible to everyone. The private key is kept private, and is only known to the specific owner (party). Anything encrypted under the public key of a specific owner, can only be decrypted by the related (corresponding) private key of the key pair, and vica versa. Figure 2 illustrates this.

The result of encrypting a message, say 'Hello', with the private key of a specific person, say P, is called the digital signature of P for message 'Hello'.

Figure 3 illustrates this, where '&2%77(' is the digital signature of person P for the message 'Hello'.

Public/private keys are issued by a trusted Certification Management Authority. Because it is essential to ensure that a specific public key really belongs to a specific person, say P, P also has a certificate, which is an electronic 'vouching statement', provided by the Certification Management Authority who issued the pair of keys to P, that the relevant public key is authentic, and really belongs to P.

5 Secure Sockets Layer (SSL)

The Secure Sockets Layer (SSL) secures the transmission channel between the client and server handling the transaction. Security is implemented 'below' the application level, and the application has no control over what services to apply to a specific document. All document (transactions) are secured generically in precisely the same way. This is referred to as channel-based security, and can provide the services F.1 to F.4 above. SSL therefore secures the channel of communication between to parties as private and authenticated, and do not 'see' individual documents.

The Secure Sockets Layer (SSL) Protocol is a secure protocol that provides privacy over the Internet. The protocol allows client/server applications to communicate in a way that cannot be eavesdropped. Each server always possesses a private/public key pair, and servers are always authenticated through their secret/public key pairs. Clients can (optionally) possess private/public key pairs, and if they do, they can be authenticated through such key pairs.

SSL requires a reliable transport protocol, like TCP/IP, for data transmission and reception. The SSL Protocol is application protocol independent, meaning that a higher level application protocol, like HTTP, FTP, TELNET, can layer on top of the SSL protocol transparently. This is illustrated in Figure 4.

In an intended client/server WWW communication, the SSL Protocol will, through its Handshake (sub)Protocol, let the client authenticate the server, decide on an encryption algorithm supported by both the client and server, and establish a session key for the specific session, before the application protocol sends or receives its first byte of data.

Clients always initiates a connection with a server, by selecting (dereferencing) a Uniform Resource Locator (URL), which is an 'address' specifying the location

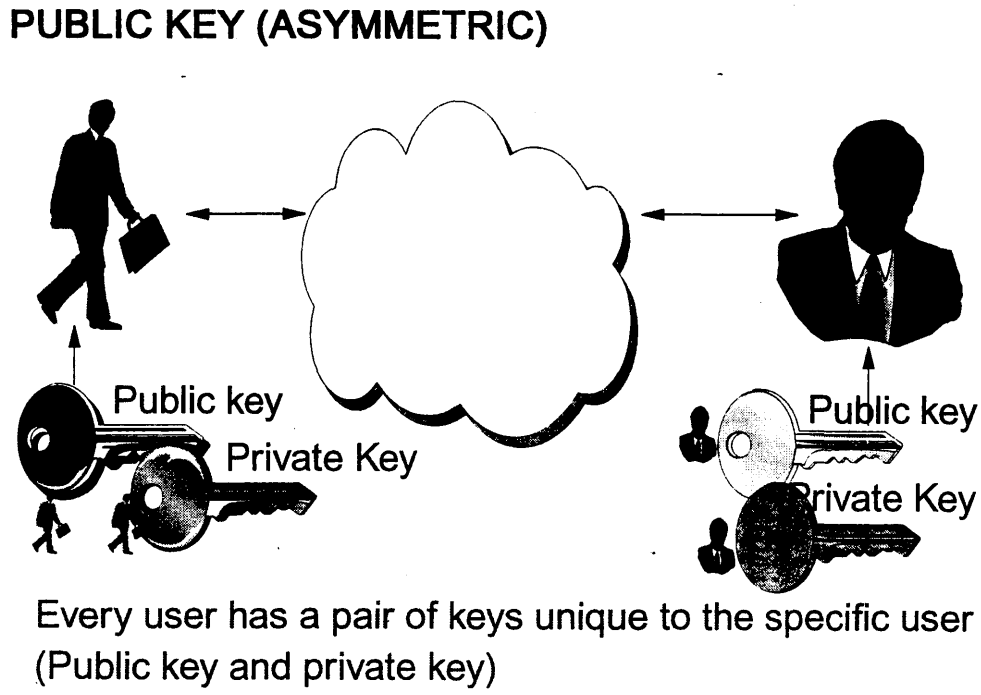

Figure 2. Asymmetric or public key encryption.

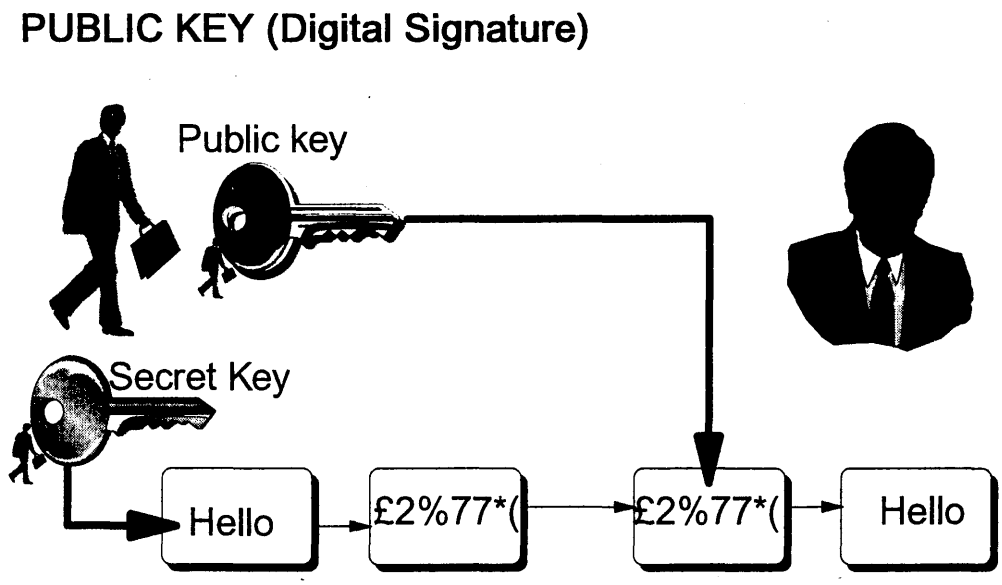

Figure 3. A digital signature.

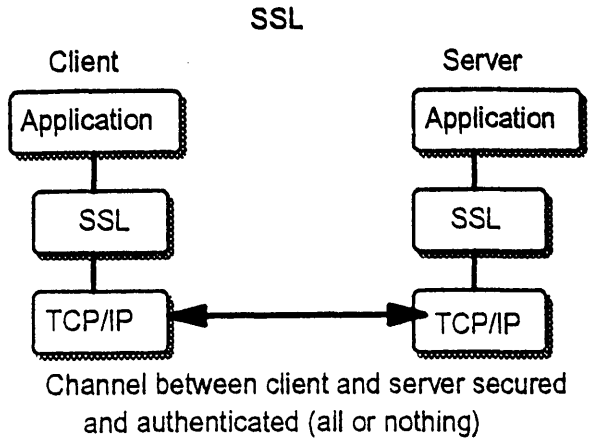

Figure 4. SSL in relation to other levels.

of a Web page. The client must know whether the server supports SSL or not. This is indicated to the client by the form of the URL. If it starts with http://, the client knows the server does not require (or support) SSL, and issues an ordinary Connect() to establish the connection. If the URL starts with https://, the client know the server supports and requires the use of SSL, and therefore the client issues a SSL_Connect(). SSL_Connect() establishes an 'ordinary' connection between the two parties, and when that is successfully completed, performs the Handshake Protocol described below. When this handshaking is finished, the first application data starts to flow.

Suppose a client dereferences a https:// URL, indicating a secure channel between the client and server is required. The browser issues a SSL_Connect(), and the two parties now perform a handshake protocol, in which the server is always authenticated to the client, and optionally the client authenticated to the server. If the handshaking process is successful, a session key for encrypting data during the session, is established between the two parties, and the application data, secured by the established session key, can start flowing.

The server is always in possession of a public/secret key pair. The client may be in possession of a public/secret key pair, but not necessarily. In the discussion below, we just indicate those data which are relevant for our explanation. All message do include other data, which is not directly relevant to our explanation.

The handshaking protocol discussed in the next section, creates a secure 'path' between the client and the server, and in the process ensures that the server is authentic, i.e. that the server is not masquerading as somebody else. Part of the process is also to establish a session key between the client and the server which will be used to encrypt any data exchanged between the two.

We consider three cases :

- No session had yet between established between the client and the server
- A session had been established, but had been terminated, and are to be re-used. This is primarily done to eliminate unnecessary overhead
- The server wants to authenticate the client during an established session.

The SSL Handshake Protocol

Creating a session between the client and server:

CS.1 The client sends a *client-hello* message to the server. This message contains a random challenge generated by the client, as well as the cipher-specs (encryption facilities) supported by the client.

CS.2 The server responds with a *server-hello* message. This message contains a connection-id, which is byte string randomly generated by the server, as well as the server's public key certificate and the server's cipher-specs.

CS.3 The client now generates a master key, using a cipher chosen from the server's cipher-spec. The master key is generated in two parts. One part of the master key is encrypted using the server's public key, retrieved from the public key certificate sent by the server. The client sends this encrypted part as well as the clear part of the master key to the server, using a *client-master-key* message.

From the master key, the challenge sent to the server in step CS.1, as well as the connection-id received from the server in step CS.2, the client generates two private keys to be used for symmetric encryption during the to be established session with the server (the session keys). One of these keys is called the client-write key, and will be used to encrypt data to be sent to the server. The other key is called the client-read key, and will be used to decrypt data received from the server.

The master key as well as the session key are stored in the client's cache.

CS.4 The server receives the two parts of the master key, decrypts the encrypted part using its secret key, and forms the master key. Using the same procedure as the client, the server now generates two keys, which

will be the same as the two generated by the client, because the same parameters are used. The one key is called the server-write key, which will be the same as the client-read key, and the other the server-read key, which will be the same as the client-write key. These two keys as well as the master key are cached.

The server now waits for a *client-finish* message from the client.

CS.5 The client encrypts the connection-id sent by the server in step CS.2, using its client-write key. It sends this to the server in a *client-finish* message.

CS.6 The server now encrypts the challenge sent to it by the client in step CS.1, using the server-write key, and sends it to the client using a *server-verify* message.

CS.7 The client decrypts the received data using the client-read key, and compares the content with the original challenge sent.

Note that if they match, the client can be satisfied that the server is authentic – so the client has authenticated the server.

CS.8 The server now generates a session-id for the to be established session, place the session id in its cache, and send this new session-id, encrypted under the server-write key, to the client using a *server-finished* message.

CS.9 On receipt the client retrieves the session-id, and stores it in its cache.

Once both client and server had sent a finish message, the SSL handshake protocol is done, and the application protocol can begin to operate, using the established session key for bulk encryption.

Using an established session between a client and server

In this case, the server had already been verified by the client, but the server wants to re-authenticate the client. It may be that a session, with a session-id and session keys had been established earlier, used for some application, finished, and wants to be reused by the client.

UE.1 The client sends a *client-hello* message to the server, containing a challenge, and the previously agreed upon session-id.

UE.2 The server returns a *server-hello* message, containing a connection-id as described in CS.2 above, as well as the session-id-hit flag, if the session-id provided by the client, is known to the server. The contents is encrypted using the server-write key. If the session-id is not known to the server, we revert back to step CS.2 above.

UE.3 The client encrypts the connection-id sent by the server in step UE.2, using its client-write key. It sends this to the server in a *client-finish* message.

UE.4 The server now encrypts the challenge sent to it by the client in step UE.1, using the server-write key, and sends it to the client using a *server-verify* message.

UE.5 The client decrypts the received data using the client-read key, and compares the content with the original challenge sent.

Note that if they match, the client can be satisfied that the server is authentic – so the client has (again) authenticated the server.

UE.6 The server encrypts the existing session-id again, using the server-write key, and sends it to the client using a *server-finish* message.

Once both client and server had sent a finish message, the SSL handshake protocol is done, and the application protocol can begin to operate, using the established session key for bulk encryption.

Using an established session between a client and server and requiring the client to sign a challenge if in possession of a public/private key pair

At any time during the re-establishment of a session, as described above, the server can request the client to further authenticate itself using the client's secret key, if it is in possession of such a key.

UC.1 After UE.3 above, the server can issue a *request-certficate* message. This message contains some challenge data, and requests the client to provide its certificate to the server.

UC.2 The client signs the challenge data received from the server with its private key, and sends the signed version, as well as a copy of its own certificate, to the server, using a *client-certificate* message.

UC.3 The server authenticates the client by decrypting the signed challenge data by using the client's public key retrieved from the client's certificate.

The SSL Record Protocol

All SSL data, also those during the handshaking process, is sent in terms of SSL records. All data records are automatically provided with a sequence number, for synchronisation between the client and the server.

To protect the integrity of data, all records sent are automatically provided with a Message Authentication Code (MAC).

Many WWW browsers are now SSL aware, meaning that WWW traffic can be transparently secured by SSL.

6 Secure Hypertext Transport Protocol (SHTTP)

If the WWW is now being introduced as a marketing vehicle, and more control over selective delivery of information is needed, SHTTP is a possible approach to take.

The Secure Hypertext Protocol (SHTTP) can secure individual documents (transactions) in different ways, because security is implemented on the application level. The application therefore has full control over what security services it wants to apply to a specific document. This is referred to as *document-based security*, and can provide the services F.1 to F.5 above. SHTTP therefore marks individual documents as private, signed etc.

As we can see from the discussion above, SSL in very much channel-oriented, and only protects the channel. Everything on the channel is protected, and there is no control on application level to decide what to encrypt and what not.

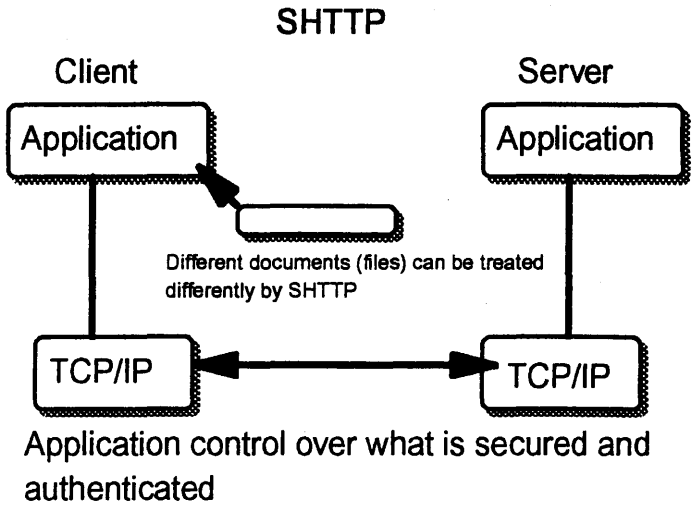

Figure 5. The Secure Hypertext Transport Protocol

SHTTP addresses much more selective application control than SSL.

The Secure Hypertext Transport Protocol (SHTTP) is an extension of the Hypertext Transport Protocol (HTTP). HTTP is the protocol used for communicating via the WWW, and the SHTTP extension provides extensive security features to secure messages sent via the WWW.

SHTTP makes a very wide range of security mechanisms available to HTTP servers and clients, and even allows clients and servers to negotiate precisely which mechanisms they want to use for secure communication. The security services provided by SHTTP are:

- confidentiality
- authenticity
- integrity
- non-repudiability
- replay-securing

These services are called *privacy enhancements* in SHTTP.

We will not discuss all these mechanisms in detail, but rather concentrate on a functional and highlevel description of the features of SHTTP.

SHTTP operates on the application level – that means that there is application control over what security features must be used, and what not. (This is opposed to SSL where the security features are implemented 'below' and transparent to the application level.)

Using SHTTP, every HTML page (document, hyperlink) on a SHTTP-aware server, can be handled security-wise, in a different way.

The application designer must therefore, as part of the design and implementation, specify the security options to be implemented for every document (hyperlink). Figure 5 illustrates the ideas.

SHTTP makes uses of public key encryption. Every server will have at least one public/private key pair. If there are different application owners on the server, the server may have a public/private key pair for each (See DNs below). Although clients may have a public/private key pair, and SHTTP can leverage the possession of such a key pair, client key pairs are not mandatory.

SHTTP is considered to be rather complex – one reason is the many options available within the protocol. We will try to steer away from all these specific options, and rather try to give an overview.

The easiest will be to try to explain the operation of SHTTP is by using a simple example. Let us start with a HTML reference (hyperlink), 'enhanced' with some SHTTP features to secure the referred page.

An example of a SHTTP 'transaction'

The Request

The client is confronted by a HTML page. On the HTML page several hyperlinks may appear, referencing documents belonging possibly to different owners.

Hyperlinks using SHTTP will use a protocol indication of shttp:// instead of http://, indicating to the client that the server is SHTTP-aware for this specific reference. With every shttp:// hyperlink, two anchor properties are associated :

- The CRYPTOPTS block, containing information about algorithms used, as well as privacy-enhancement information. These privacy-enhancements specify the actions required by the server when the specific hyperlink is dereferenced, i.e. the page referred to by this hyperlink is requested. Suppose in our example, the server requires that the request must be encrypted.
- The distinguished name (DN) of the principal on the server requiring these privacy enhancements. There may be different 'document owners' on the server, and each will have its own DN.

The HTML page will also contain the public key certificates of all the DN's associated with shttp:// hyperlinks on that

page.

The client now creates an HTTP request (GET) to retrieve the page, and, because of the privacy requirement of encrypt specified above, the client generates a DES key, encrypts his request with the DES key, and encrypts the DES key with the relevant DN's public key (found in the HTML page).

Several SHTTP headers are now added to the (encrypted) request, specifying information about the algorithms used by the client, the format of the request etc.

This SHTTP encapsulated message is now sent to the server.

Client actions as far as security enhancements are concerned, are therefore governed by the anchor security-enhancements specified in the anchor properties of the shttp:// hyperlink selected.

Server actions

The server uses the information in the SHTTP headers, and the relevant private key to retrieve the DES key supplied by the client, and using that key, retrieves the request itself. The server now retrieves the document specified in the request.

The security actions of the server, ie what must the server now do to the information to be delivered to the client, depend on whether the hyperlink chosen by the client, specifies a static file, or the execution of a Common Gateway Interface (CGI) program.

Every static file (page), has a local security configuration file ('dot file'), in which the server actions on delivering the results to the client, are specified.

Every CGI program contains a Privacy-Enhancements header in which the actions to be taken by the server on delivering the results to the client are defined.

Suppose the client chose a static page in our example above. The server checks the local configuration file of the page, and if required to encrypt, does so with the session key provided by the client, and again encapsulates the response with the necessary SHTTP headers.

The response is sent to the client.

Client actions

Using the information in the SHTTP headers, and the DES key generated above, the client decrypts the HTML document requested.

Suppose in the HTML page retrieved, there is a hyperlink to another HTML page on the server. The privacy enhancements for this request may now require the client to digitally sign the request before sending it to the server. The client will now sign the request with its own private key, if it has one, otherwise it is stuck. Suppose the client can sign it. Using the SHTTP different headers, information about the algorithms used for signature, and the client's public key, is added to the message, and sent off to the server.

In this way every page/request can be treated in a different and individual way, enforcing the security needed for that specific page.

7 Phase 3

At this stage, the company starts using the Net and WWW for electronic purchasing in a 'closed' environment, where the customer must register before the time, and transactions are only performed with registered customers.

Because payment comes into play in this phase, banking institutions and credit card companies are essential parts of the environment.

Payment in this phase is usually handled by existing procedures.

Potential customers must register with the company, and electronic purchasing is only possible by such pre-registered customers. Such customers signs an agreement with the company for payment by debit order, or by debiting the customer's credit card, the number which is kept on the customer's record at the company. Identification and authentication of the customer is done via a password shared between the customer and the company.

8 Phase 4

This stage brings 'open' electronic purchasing, where transactions can be performed with any customer presenting a credit card number.

In this environment, no pre-knowledge about the customer exists at the company, and the company wants to allow this totally unknown customer to do electronic purchasing. The company, however also wants to be sure that the potential cutomer is the real owner of the presented credit card number, and that the card issuing authority will authorize the transaction .

As the security risks of this environment is much greater than any other, much more sophisticated protocols are needed.

Several such secure payment protocols exist. Two major players in this arena are STT, the Secure Transaction Technology Protocol [4], developed in collaboration between Microsoft and Visa. Another is SEPP, the Secure Electronic Payment Protocol [1], jointly developed by IBM, Netscape, Mastercard and a number of other cooperators.

We will review SEPP in this section. To make the presentation as simple as possible, the protocol is simplified to a big extent, and only the really important aspects are discussed.

The following players are active in this scenario:

- The Card holder (potential customer)
- The Merchant
- The Acquirer bank
- The Issuer Bank and
- The Certificate Management System.

Figure 6 illustrates the relation between these parties.

We will now look at the flow of messages to complete a transaction, with specific reference to the security aspects.

Assume that the Card holder has completed the Negotiation phase, i.e. he has done his browsing, and has decided, possibly after negotiation with the Merchant, that he want

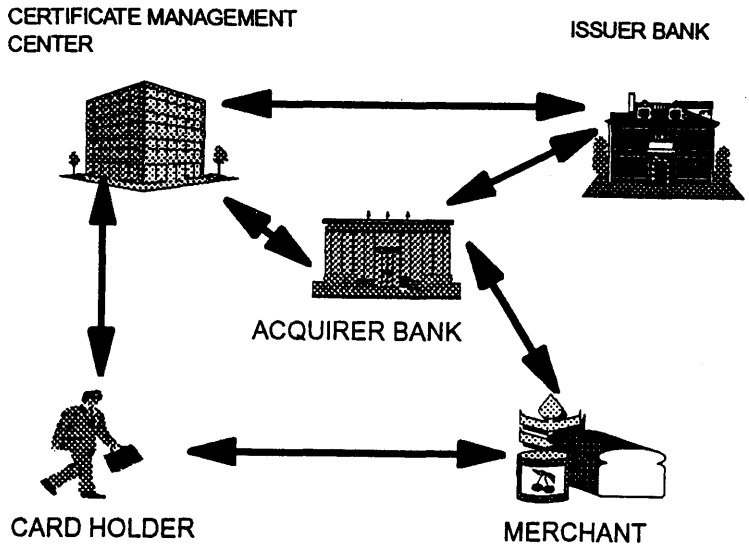

Figure 6. The SEPP Environment

to move to the Purchase/Payment phase. SEPP address this Purchase/Payment phase.

Message 1

The first message, the *initiate message*, is sent from the Card holder to the Merchant. This message contains inter-alia a customer-id and the brand-id of the to be presented credit card.

Message 2

The Merchant now creates a message containing information about the goods and price decided on, and digitally signs it with his private key. This commits the Merchant to the agreed goods and price, which he cannot now deny at a later stage. This is sent back to the Card holder in the *invoice message.*

Message 3

The Card holder now checks the contents of the invoice message by retrieving the contents using the Merchant's public key. If successful, this authenticates the Merchant, and commits the Merchant to the specific goods and price.

The Card holder now creates the *purchase_order_ request message*, consisting of

1. information about the agreed goods and price, his credit card number and expiry date, all encrypted with the public key of the Acquirer. Let us call this M1. (This means that the Merchant cannot retrieve the credit card number and expiry date, and cannot change the information about the goods and price.)
2. the goods and price decided on, digitally signed by the Card holder with his private key. (This means that the Card holder cannot later deny placing the order, for the agreed goods at the agreed price.)

The purchase_order_request message is sent to the Merchant.

Message 4

The Merchant now creates the *auth_request message*, which is sent to the Acquirer to authorize the transaction. This message contains :

1. M1 as described above
2. Information about the agreed goods and price, digitally signed by the Merchant. Let us call this M2.

The Acquirer now retrieves the contents of M1, and determines what the Card holder claims he is buying, and at what price. The Acquirer also retrieves the contents of M2, and determines what the Merchant claims he is selling, and at what price.

If needed, the Acquirer now requests an authorisation from the Issuer.

The Acquirer receives an authorisation code back.

Message 5

The Acquirer now creates the *auth_response message* consisting of the authorisation code, digitally signed by the Acquirer.

This message is sent to the Merchant.

Message 6

The Merchant now sends the purchase_order_response message back to the Card holder. This message consists of the authorisation code, digitally signed by the Acquirer. At this stage the transaction is basically completed.

Many other messages are specified to acquire certificates, check certificates etc., but the 6 messages above describe the situation when on-line authorisation is done.

9 Summary

In this paper we tried to give a high level overview of some important security protocols for the Internet and WWW. By no means are these the only, or necessarily the most

important. They are however all good protocols, enjoying wide support at the moment.

References

1. http://mastercard.com/Sepp/sepptoc.htm.
2. http://www.commerce.net/information/standards/drafts/shttp.txt.
3. http://www.netscape.com/info/SSL.html.
4. http://www.visa.com/visa-stt/stt_home.html.
5. Phleeger. *Security in Computing*. Prentice Hall, 1989.

Notes for Contributors

The prime purpose of the journal is to publish original research papers in the fields of Computer Science and Information Systems, as well as shorter technical research notes. However, non-refereed review and exploratory articles of interest to the journal's readers will be considered for publication under sections marked as Communications or Viewpoints. While English is the preferred language of the journal, papers in Afrikaans will also be accepted. Typed manuscripts for review should be submitted in triplicate to the editor.

Form of Manuscript

Manuscripts for *review* should be prepared according to the following guidelines.

- Use wide margins and $1\frac{1}{2}$ or double spacing.
- The first page should include:
  - title (as brief as possible);
  - author's initials and surname;
  - author's affiliation and address;
  - an abstract of less than 200 words;
  - an appropriate keyword list;
  - a list of relevant Computing Review Categories.
- Tables and figures should be numbered and titled.
- References should be listed at the end of the text in alphabetic order of the (first) author's surname, and should be cited in the text in square brackets [1–3]. References should take the form shown at the end of these notes.

Manuscripts accepted for publication should comply with the above guidelines (except for the spacing requirements), and may be provided in one of the following formats (listed in order of preference):

1. As (a) LaTeX file(s), either on a diskette, or via e-mail/ftp – a LaTeX style file is available from the production editor;
2. As an ASCII file accompanied by a hard-copy showing formatting intentions:
   - Tables and figures should be original line drawings/printouts, (not photocopies) on separate sheets of paper, clearly numbered on the back and ready for cutting and pasting. Figure titles should appear in the text where the figures are to be placed.
   - Mathematical and other symbols may be either handwritten or typed. Greek letters and unusual symbols should be identified in the margin, if they are not clear in the text.

   Contact the production editor for markup instructions.
3. In exceptional cases camera-ready format may be accepted – a detailed page specification is available from the production editor;

Authors of accepted papers will be required to sign a copyright transfer form.

Charges

Charges per final page will be levied on papers accepted for publication. They will be scaled to reflect typesetting, reproduction and other costs. Currently, the minimum rate is R30-00 per final page for LaTeX or camera-ready contributions that require no further attention. The maximum is R120-00 per page (charges include VAT).

These charges may be waived upon request of the author and at the discretion of the editor.

Proofs

Proofs of accepted papers in category 2 above may be sent to the author to ensure that typesetting is correct, and not for addition of new material or major amendments to the text. Corrected proofs should be returned to the production editor within three days.

Note that, in the case of camera-ready submissions, it is the author's responsibility to ensure that such submissions are error-free. Camera-ready submissions will only be accepted if they are in strict accordance with the detailed guidelines.

Letters and Communications

Letters to the editor are welcomed. They should be signed, and should be limited to less than about 500 words.

Announcements and communications of interest to the readership will be considered for publication in a separate section of the journal. Communications may also reflect minor research contributions. However, such communications will not be refereed and will not be deemed as fully-fledged publications for state subsidy purposes.

Book reviews

Contributions in this regard will be welcomed. Views and opinions expressed in such reviews should, however, be regarded as those of the reviewer alone.

Advertisement

Placement of advertisements at R1000-00 per full page per issue and R500-00 per half page per issue will be considered. These charges exclude specialized production costs which will be borne by the advertiser. Enquiries should be directed to the editor.

References

1. E Ashcroft and Z Manna. 'The translation of 'goto' programs to 'while' programs'. In *Proceedings of IFIP Congress 71*, pp. 250–255, Amsterdam, (1972). North-Holland.
2. C Bohm and G Jacopini. 'Flow diagrams, turing machines and languages with only two formation rules'. *Communications of the ACM*, **9**:366–371, (1966).
3. S Ginsburg. *Mathematical theory of context free languages*. McGraw Hill, New York, 1966.

South African Computer Journal

Number 17, September 1996
ISSN 1015-7999

Suid-Afrikaanse Rekenaar-tydskrif

Nommer 17, September 1996
ISSN 1015-7999

Contents

GUEST EDITORIAL

Information Security – The Family Member Who Came Home
SH von Solms 1

SPECIAL EDITION: COMPUTER SECURITY

A New Framework for Information Security to Avoid Information Anarchy
DB Parker 4

Encryption Policy for the Global Information Infrastructure
LJ Hoffman 10

Functional and Operational Security System for Open Distributed Environments
S Muftic 17

Reinforcing Password Authentication with Typing Biometrics
WG de Ru and JHP Eloff 26

Information Security on the Electronic Superhighway
SH von Solms 36

Design of Secure Medical Database Systems
G Pangalos and M Khair 45

Drukkers
Printers