Abstract

The study outlines a number of security requirements that is typical of a host of Web-based applications using a case study of a real life online Web-based customer support system. It subsequently proposes a security solution that employs a combination of Web server security measures and cryptographic techniques. The Web server security measures include the formulation and implementation of a policy for server physical security, configuration control, users’ access control and regular Web server log checks. Login passwords, in conjunction with public key cryptographic techniques and random nounces, are used to achieve user authentication, provide a safeguard against replay attacks, and prevent non-repudiatory usage of system by users. These techniques, together with the use of session keys, will allow data integrity and confidentiality of the customer support system to be enforced. Furthermore, a number of security guidelines have been observed in the implementation of the relevant software to ensure the further safeness of the system.

Keywords: Communication technology, Computer security, Data security, Web server security, Encryption

Introduction

The advent of the World-Wide-Web has seen an explosion of Web-based applications designed for all aspects of business computing. This diverse range of applications covers a simple change-of-address notification to complex artificial intelligence systems for risk assessment. With the move towards the era of global information technology, regionalisation and globalisation of businesses, and the ongoing enhancement to Web technology, the WWW is set to take off further in the future. It is predicted (International Data Corporation, 1997) that Internet commerce is likely to grow to over US$100 billion by the end of the year 2000. In addition, the number of devices assessing the WWW will grow from 12.6 million worldwide in 1995 to 233.3 million in 2000. During this period, the number of Web users is expected to jump from 16.1 million to 163 million.

Security is one of the most important design criteria in many Web-based applications. This paper examines a number of security issues in the delivery of a real-life industrial online
Web-based customer support system and proposes a number of solutions to tackle these issues. Through this, we hope to highlight typical security requirements in generic Web-based systems of a similar nature. We will give a brief description of the application and identify a number of essential security issues for the safe operation of the system. A number of techniques, including those to ensure Web server security, cryptographic means and implementation guidelines, are subsequently considered and employed for meeting the security requirements of the system.

**Case Study Background and Security Requirements**

A multinational corporation set up in Singapore (hereafter known as MNCS) manufactures and supplies insertion and surface mount machines for use in the electronics industry. A customer support (or service centre) was set up by the Customer Support Department to service its worldwide customer-base and provide installation, inspection and maintenance support for its customers. This is currently achieved through a 24-hour telephone-based support and on-site repair visits by service engineers.

This manually intensive system shall be replaced soon by an integrated online customer support environment over the Web. This environment is designed to provide an integrated platform to transform the customer service support into an effective and efficient activity to satisfy enhanced users’ requirements. The delivery of the system signals many new advantages including timely service, decrease in manpower requirements, easy and wide accessibility through Web, enhanced multimedia support, and cost savings among others (Liu, 1998).

![Diagram of Web-Based Integrated Customer Support Environment](image_url)

**Figure 1. Web-Based Integrated Customer Support Environment**
Figure 1 shows the system architecture of the Web-based integrated customer support prototype system, WetHotline (Liu, 1998). This system is designed to support four main functions:

- **Web-based database retrieval.** This allows remote customers to access and retrieve service records from the customer service (or knowledge) database through a Web-based retrieval tool. This database contains past customer service records which serve as records of problems as well as to provide the knowledge to tackle similar problems. These retrieved service records can then be subsequently used directly by customers to solve their problems.

- **Machine translation.** In order to exploit the expertise and share experiences gained by regional service engineers in Asia, an online translation of service records from English to other languages (such as Chinese and Japanese) is provided. This translation is achieved by simplifying and redefining the sentence structures used for documenting the service records so that a Multilingual Translation module can be used to carry the translation with the aid of translation look up tables. This will allow customers from within the region to gain access to one centralised service database to rectify simple problems encountered in the operation of their machines.

- **Intelligent fault diagnosis.** A CLIPS expert system based inference engine has been incorporated into the system based on a rule-based approach. The purpose of this inference engine is to support intelligent fault diagnosis to suggest possible remedial actions according to the fault-conditions reported. This facility is provided as an alternative to the Web-based database retrieval capability in cases where fault conditions are not available in the database.

- **Video-conferencing.** A low-bit rate video-conferencing system is incorporated into the system to support a different number of video-conferencing modes. This include the normal video-conferencing mode for interpersonal communication between customer and service engineer, a picture phone mode where enlarged still images are captured by the customer and viewed instantaneously by the service engineer, and an audio-priority mode where sound recordings (of a faulty machine or part) are captured at high resolution by the customer and transmitted to the service engineer who will be able to play it back and analyse the accompanying waveform. The provision of these various operating modes of sight and sound enhances MNCS’s capability to perform remote diagnosis support to distant customers, thereby reducing the overhead cost of sending service personnel abroad for an on-site repair unless there is an absolute need to do so.

Security becomes an important design criterion in such a system as users interact and exchange important information with the system. Careful considerations of the following security requirements are necessary to meet MNCS’s operations. These are largely typical and generic security requirements that can be found in a myriad of Web-based applications:

- **Integrity of the invaluable service record database (with respect to MNCS’s operation) and the availability of customer support system.**

- **Access control:** only legitimate clients, service engineers and designated executives would be permitted to access the system. Access is also restricted to designated functions that commensurate with the capacity of the user concerned.
User authentication: in addition to access control, remote accesses over an open network will be subjected to a user authentication capability that includes the following desirable features:

i. Impossibility for a client to repudiate actual usage of the customer support system. This is to facilitate charging for services in future.

ii. Possibility for service engineers to access the customer support system from a client’s facility but without incurring unnecessary cost to the client. Such a feature may help MNCS to further reduce service turnaround time and travelling expenses of service engineers.

iii. Deterrence to espionage and/or sabotage activities carried out by business rivals, possibly with the help of ex-MNCS’s clients or ex-MNCS’s service personnel.

iv. Data origin authentication on the client’s end as well as ensure confidentiality and integrity of service information enroute on the Web. This is a necessary part of an effort to deter espionages and prevent sabotages to service quality by business rivals.

This seemingly complex set of requirements is necessary due to the security risks inherent in an open architecture like the Internet. The Internet provides a link to any computer that is connected to it to access an operational Web server. Malicious attacks on the customer support server, whether initiated by business rivals or not, may result in disastrous consequences to MNCS’s Customer Service Department.

Overview of Security Techniques

The document “World Wide Web Security FAQ” from the World Wide Web consortium (W3 Consortium, 1998) provides a detailed account on common security considerations required in the design of a Web-based system. These include the Web server configuration, CGI programming, and user access control, among others. ISO 7498-2 (ISO 7498-2, 1989) defines five main categories of security services, namely: authentication, access control, confidentiality, data integrity and non-repudiation. MNCS’s security requirements are expressed in terms of these security services. Schneier (Schneier, 1996) discusses a number of common cryptographic techniques that can be employed for achieving these security services along with possible software implementation in ‘C’. An account of popular cryptography algorithms currently used in the community is given in RSA’s web site (RSA Data Security Inc, 1998), Secur (Certicom Corp, 1998a), and a comparison on the complexity and performance of three widely accepted public key systems is available in Certicom’s web site (Certicom Corp, 1998a).

In designing the security solutions for the proposed requirements, it became clear that a variety of techniques must be employed collectively to address these overall security needs. These techniques can be broadly categorised into Web server security-related and software security-related techniques. The latter can be further classified into programming concerns and communication security, both of which involve cryptography extensively. In this paper, various techniques for meeting MNCS’s security requirements are examined. Whenever appropriate, possible off-the-shelf products that may be useful solution alternatives are identified and evaluated. It must be emphasised that our security model assumes that senior staff members and the system administrator(s) can be unconditionally trusted.

Web Server Security

Part of our effort to maintain integrity of the invaluable service record database, ensure availability of customer support system and provide the necessary access control to meet
MNCS’s requirement is to ensure that the Web server remains ‘safe’. Such a ‘safeness’ includes physical security as well as tight and carefully tailored access control over legitimate users of different categories. It must be pointed out that a ‘safe’ Web server is a necessary, though not sufficient, condition for upholding the validity and reliability of all the cryptographic measures to be described in subsequent sections. This section discusses a number of measures that address server security. These measures are very well understood, some of which even before the prolific use of the Internet. They are often discussed in a form or a set of rule-of-thumb using a do-don’t list (W3 Consortium 1998).

**Web Site Locality and Physical Security.**

The Web server computer should be protected by keeping the computer in a locked-away enclosure that is inaccessible to unauthorised users. Likewise, a computer lock, if it exists, can be used to secure the computer further. A security policy for controlled access to these keys should be in place to ensure proper key management.

**Web Server Configuration**

The configuration of access rights to critical files in the Web server is very important. Only operating systems with at least a built-in password-based access control facility can be employed. Care must be taken to ensure that an ordinary local user cannot, whether intentionally or otherwise, change the Web server configuration file or the document tree in such a way that a security loophole is created. Hence file permissions in the server’s directories must be carefully tailored in order to ensure that unauthorised but legitimate local users cannot indirectly gain access privileges to install application software and access critical information (such as the encryption key). For instance, one may create a “www” group for which only trusted Web authors are added. The document root is then made writable only by members of this group. To increase security further, the server root where vital configuration files are kept is made writable only by the designated Web administrator.

**User Access Control**

User accesses to the Web server, either locally via the Intranet or remotely via the Internet, must be controlled carefully. Although eavesdropping by internal staff is possible, it is not a major concern in this study. The application trusts that all ‘insiders’ to carry out their respective duties without using their designated privilege against the company. The procedures adopted for local user control could be a common standalone LAN security policy. Control over remote users can be achieved through configuration set-ups of the Web server, use of firewalls or proxy servers.

Access control capability provided by Web servers to authorise accesses from other hosts as specified in terms of IP addresses and/or host names are employed in this regard. For example, Figure 2 shows Netscape’s Enterprise Server 3.0’s interface for access control definition. Internet Protocol (IP) address restriction can be made much safer by running the Web server behind a firewall machine that is capable of detecting and rejecting attempts at spoofing IP addresses (W3 Consortium, 1998). The firewall machine intercept packets originated from the outside world that attempt to disguise as messages from trusted machines on the internal network.
Care must be exercised in cases when a browser is fetching documents through a proxy server. In this instance, the Web server will only recognise the IP address of the proxy in such a setup, but unable to identify the real user. This means that if the proxy is in a trusted domain, anyone can use that proxy to access the Web site. Unless a particular proxy can be trusted to do its own restriction, it becomes imperative not to add the IP address of a proxy (or a domain containing a proxy server) to the list of authorised addresses.

Likewise, it is necessary for the system administration to be aware of “DNS (Domain Name Server) spoofing” (W3 Consortium, 1998) when using restriction by host/domain name. In an attack of this kind, the Web server is temporarily fooled into thinking that a trusted host name belongs to an alien IP address. To lessen the risk, the server should be configured to do an extra DNS lookup for each client. After translating the IP address of the incoming request to a host name, the server uses the DNS to translate from the host name back to the IP address. Permission to access is granted only if the two addresses are identical.

**Password Control**

Good password policies and control should be implemented for the system. Users with login privileges should be made to choose good passwords. A poorly chosen password is likely to be guessed and used by unauthorised users. The critically important administrator’s password must be carefully controlled. A policy for regular changes to password must be enforced. We recommend the system administrator to minimally adopt the following guidelines for defining passwords:

- At least 7 alpha-numeric characters in length, preferably case sensitive and not pure numbers since the latter by itself is considered as a bad password;
- Avoid using names (or login names), user’s particulars such as middle names, birthdays, respective office telephone numbers, favourite pet names, and so on, as the password;
- Avoid using existing words from a dictionary as the password. A password should be a non-existent word from the dictionary and phonetically hard to pronounce;
- A regular password change should be enforced (such as once in every three months). An even better practice is to change the password in a ‘sporadic’ fashion within these regular intervals;
These guidelines are essential to hinder hackers who employ password guessing programs such as AntiCrack (AntiCrack, 1998) to break in by means of brute force.

**Turn off Unused Services**

Unused services should be turned off whenever possible. For example, if there is no need to run a File Transfer Protocol (FTP) service on the Web server host, the FTP software should either be removed or disabled. This should also be the case for other Internet services such as tftp, Gopher, NFS, and anything else that may be present but not necessary for the server’s proper operation. Regular checks on servers that may be lurking should be carried out.

**Check System and Web Logs Regularly**

Both system and Web logs should be checked regularly for suspicious activity. Normal checks include the identification of accesses involving system commands or extremely long lines in URL requests. The former may indicate an attempt to trick a CGI script into invoking a system command; the latter may be an attempt to overrun a program’s input buffer. Particular attention should be focused on repeated attempts to access password protected documents. These could be symptomatic of someone trying to guess a password (W3 Consortium, 1998). If this really happens, deny further accesses from these IP addresses or domain names by re-configuring the Web server, the firewall machine or the proxy server.

**Keep the Newest Service Pack**

Newest update packs of system software regularly distributed by vendors should be carefully analysed with respect to security implications. This may reduce the bugs in the system and provide more security but may also enhance other performance aspects at the expense (albeit unintentionally and indirectly) of security needs.

**Software Security**

Authentication, confidentiality and data integrity can be addressed by well-studied cryptographic techniques (Schneier, 1996; Needham, 1978). In using such techniques, it is recognised that information enroute on the Internet can pass through numerous computers before it reaches its destination. A malicious user of any of the intermediary computers can monitor the Internet traffic enroutes, eavesdrop, intercept, modify or replace the data along the way. Cryptographic techniques can be used to protect this data. There are two kinds of key-based encryption systems: symmetric and asymmetric systems (Schneier, 1996).

Symmetric encryption uses the same secret key to encrypt and decrypt a message. Two users wishing to communicate in confidence must agree in advance, via a secure channel, to maintain a common secret key and each entity must trust the other in not divulging the key to a third party. In addition to the key distribution and management problem (Needham, 1978), the fact that multiple parties knowing the key implies that authentication by means of knowledge of the key becomes very difficult (AntiCrack, 1998). Non-repudiation issues are almost impossible to address when any one in the group could edit or replace messages in transit from the source to the intended recipient. Examples of symmetric encryption systems include DES and IDEA among others.

Asymmetric encryption separates the functions of encryption and decryption by having a complementary pair of keys; usually referred to as the private key and public key. The private key is normally kept secret while the public key does not need to be kept secret. This two-key approach simplifies key management by minimising the number of keys to be stored and allows the distribution of public keys via unprotected open networks. The difficulties in meeting authentication and non-repudiation needs in symmetric systems as described earlier...
vanish in the asymmetric system. However, asymmetric systems are generally much less efficient in terms of computation.

In an asymmetric encryption system, if Alice wishes to securely transmit messages to Bob, Alice must first obtain the certified public key from Bob. This is carried out in the open. Alice will then encrypt the message using Bob’s public key. Since the complimentary private key is necessary for decrypting the message, it can only be decrypted and read by Bob, the only one in possession of the private key required. Even the originator of the message cannot decrypt it once encryption has taken place.

Different asymmetric cryptographic algorithms based upon different mathematical problem exist. RSA is an example of the Integer Factorisation problem (RSA Data Security Inc, 1998; Certicom Corp, 1998a). Examples of Discrete Logarithm problem (Schneier, 1996) include the ElGamal’s system and its variations such as that employed in the DSA (Digital Signature Algorithm). A DSA equivalent, ECDSA, has been proposed based on the Elliptic Curve Discrete Logarithm problem (RSA Data Security Inc, 1998). Studies on the advantages of the various cryptosystems have been reported by Schneier (Schneier, 1996). For example, RSA is easy to implement, ElGamal works well for encryption, and DSA is great for digital signatures.

An analysis of the security requirements reveals that a simple cryptographic protocol turns out to be sufficient for the application concerned. The protocol is carefully designed by following the design guideline suggested in (Abadi, 1994). The proposed solution did not include the use of commercial products that provide similar cryptographic services due to cost reasons and the fact that some of these commercial products are over-kills. We will briefly mention possible commercial candidates in the following before describing our proposed protocol.

**Commercial Cryptographic Products**

Many vendors have addressed security issues pertaining to the Web applications. Examples include Netscape Security Solutions (Netscape Communications Inc, 1998), Microsoft Security Advisor Program (Microsoft Corp, 1998), and Web Based Documentation - Security Issues (Orasis Corp, 1998). Besides these solutions for normal Web site security, there are also other commercial products that provide encryption services to data transferred over the Internet. SSL (Secure Socket Layer) and SHTTP (Secure HTTP) are two popular schemes. SSL is a low level encryption scheme used to encrypt transactions in higher-level protocols such as HTTP, NNTP and FTP. The SSL protocol includes provisions for server authentication, encryption of data in transit, and optional client authentication. SSL is currently available on several different common browsers. It is also available on server software from major vendors. However, usage of SSL requires every legitimate user of the customer support system to register with a third party vendor at a cost as well as subject to the 40-bit restriction imposed by the government of the USA. This additional cost is unnecessary as all legitimate users of the customer support system are known to the system administrator of the server.

SHTTP is a higher level protocol that only works with the HTTP protocol although it is potentially more extensible than SSL. Currently SHTTP is implemented for the Open Marketplace Server marketed by Open Market, Inc on the server side, and Secure HTTP Mosaic by Enterprise Integration Technologies on the client side. The main drawback of using these off-the-shelf security products for Web applications is that it requires the correct combination of compatible browser and server to operate. From the point of view of customer support, the utilisation of such a solution is unfeasible since it does not only increase costs but also decreases usability since compatible browsers become a prerequisite in using the system. Furthermore, the control by the vendor also means that MNC
would be unable to verify and assess security measures that are being employed. In addition, the subscription cost for the key-pair certificates must either be borne by MNC or its clients.

The Proposed Protocol

In the proposed solution, we employ asymmetric encryption based techniques to satisfy the authentication and non-repudiation security needs, and use symmetric encryption to meet confidentiality needs. An off-the-shelf public domain software package (Network Associate, 1998) which provides RSA asymmetric key was employed for the development of the prototype customer support system in order to reduce development time. We consider RSA as being a well scrutinised algorithm with no known efficient general method of breaking (Stanford University, 1998), and that a 512 bit key is sufficiently secure for our application according to existing literature (Stanford University, 1998; Network Associates, 1998; Certicom Corp, 1998b). The software package used provides the functionality for message encryption, digital signatures and data compression. It also handles exceptions and avoids usage of parameters that may result in a weak system.

Figure 3 shows the simple cryptographic protocol designed for the customer support system. The protocol focuses mainly on user authentication and server authentication. Figure 4 shows the software architecture for implementing the protocol. In this system, besides the Web server security considerations discussed previously, ActiveX and CGI programs are used to embed cryptographic operations in the application layer. Ideally, these cryptographic
functions should be developed in-house in order to have full control and assurance of the source code although the employment of properly licensed source-code packages may be functionally acceptable.

The process of normal customer access is outlined as follow:

- When a user wishes to access the customer support server via a Web browser, the Web server will first send a plain-text 64-bit random number \( n_s \) (as the nonce\(^1\)) to the Web client and start the ActiveX program in the client side. The ActiveX program should be pre-installed in the Web client side; otherwise some alternative form of server authentication by the client must be put into place if downloading is used instead. This can be achieved, for example, by accompanying the program with a digital signature produced using the server’s private key (Schneier, 1996).

- The ActiveX program in the Web client side will require a name and password to be supplied by the user. In the current prototype, these are then combined (e.g. Exclusive-ORed or padded) with \( n_s \) before being encrypted using the server’s public key and transmitted to the Web server for authentication. If deemed necessary, the client can also generate a random session key \( k_s \) for symmetric encryption (i.e. DES for our application) of subsequent data and include it in the encrypted reply to the Web server. For an even more robust solution, we recommend the use of a smart card to store the password and the server public key. This smart card must be physically present during each access to the server.

---

\(^1\) A ‘nonce’ is an entity to ensure freshness, usually used for prevention of attacks by replay and message linkage.
• The *CGI* program in the Web server will decrypt and authenticate the user prior to permitting access to the various customer support functions. If the nonce, user’s name and password are all correct for the session, the user will be allowed to use the customer support system continuously until the session is terminated by either the user or system.

• If deemed necessary, all subsequent data that is transmitted across the network between client station and Web server in the session can be protected by padding with $n_s$ before symmetrically encrypted using the key $k_s$. This ensures confidentiality and integrity of the data enroute on the Internet.

The nonce is used mainly to confirm ‘freshness’ in the current communication session in order to prevent replay attacks. This nonce will be deleted and deemed useless as soon as the session ends. The Web server will deny any access request without a correct nonce. In this protocol, actual user authentication is carried out by means of password authentication. Confidentiality and integrity of enroute data are achieved using $n_s$ in combination with $k_s$ as a symmetric encryption key, both of which vary from one session to another. The ability to retrieve $k_s$ provides the server the confirmation of data origin authentication. In the event of interruption in communication, accesses by the same user’s IP address or domain name should be detected and denied. A new session must therefore be initiated. This, together with the use of the nonce, the server public key and the correct password will prevent the user from repudiating actual usage. Charges to usage of the customer support system can then be achieved by employing system tools available on the Windows NT server software installed on the customer support server.

Our implementation chooses to compile both the *ActiveX* and *CGI* programs (written in Visual C++) into binary code before deployment in the system. Hence all encryption operations are embedded in the program, but the cryptographic algorithm can still be changed when necessary. It must be emphasised that since the server’s public key is being embedded in the binary code of the *ActiveX* program, it is possible for a malicious attacker who has access to the client machine to replace the program on the client machine with one that contains his own public key. He can then trick the client to release the password by impersonating the server. Our model assumes that all relevant staff members who have access to the client machine are trustworthy in this regard and that the machine is maintained by the same group of people. Otherwise, the server’s public key must be placed on a removable media or device (such as a smart card). The current model permits changes to the public and private key pair on the Web server side by a recompilation of the client program with the new Web server’s public key. The new executable program, accompanied by a digital signature produced using the original private key, can then be conveyed to the client over the Internet.

**Other Considerations**

Strong cryptographic algorithm is powerful when it is used correctly but it is not a panacea (Needham, 1978). Focusing on the cryptographic algorithms while ignoring other aspects of security is like defending your house with strong walls, strong doors, strong window frames but without protection over the chimney. Attackers can often find simple loopholes to get around the algorithms if the system is not designed carefully. We discuss a number of further pitfalls that may be exploited by smart attackers.

**Attacks against Implementations**

While encryption is very difficult to be attacked, *CGI* scripts are a major source of security voids. *CGI* scripts must be written with just as much care as the server itself. Normally, compiled languages such as C/C++ can give further deterrent to potential attackers except for the most determined. Much more effort and technical expertise would be required to carry out
a reverse engineering process to identify the functionality of the software. It is also much more laborious to detect implementation bugs that can induce a security void.

On the other hand, CGI scripts can also be written using scripting languages (e.g. C shell script) instead of compiled languages. In this case, it is very difficult to write a CGI script of arbitrary complexity that completely avoids dangerous constructions. For example, it is extremely easy to send data to system commands and capture their output in a scripting language, yet invocation of system commands from within a script is a major potential source of security problems. For this reason, using a scripting language for writing CGI scripts is not recommended unless it is a very trivial and straightforward function.

In the prototype system, all the C/C++ programs are written to perform only designated functions in the most simple and direct way. These programs are compiled and stored in the cgi-bin or wincgi-bin directories with carefully tailored access control. In addition, they are designed and implemented carefully to avoid a set of common unsafe mistakes. In particular, these programs will:

- Make assumptions about the size of user inputs and allocate fixed memory for these inputs. When there is a need to receive user’s inputs (such as the password in some CGI programs), the program would use previously fixed and defined memory for storing these inputs so that a memory overflow exception will be activated if these limits are breached.

- Prevent passing unchecked remote user inputs to a shell command. All the CGI programs in the system do not use or start any shell commands by itself or through user inputs. This prevents the potential serious consequences that can arise if commands are allowed to execute freely.

- Ensure erasure of logs, password and other critical information from the memory after usage to avoid unintended covert channels. All CGI programs will reset dynamically requested memory blocks to a known form before releasing it so that no useful information is left in memory.

- Avoid giving any explicit clue which may lead to reverse engineering. This system combines many separate CGI programs into one and controls the program by using different parameters. This minimises the possibility that an individual CGI program is giving sufficient explicit clues for tracing.

- Avoid giving explicit warning messages in a program for detecting the password. This is mainly to hinder attackers who intend to employ a bypass-password-testing attack to circumvent the password authentication process.

**Conclusion**

By using a case study of an online Web-based customer support system, we have presented a set of security requirements that is typical of many Web-based applications. These security requirements are met through a proposed solution that employs a combination of Web server security measures and cryptographic techniques. This is supplemented by carefully observing a number of security guidelines during implementation of the relevant software. It is believed that security can be further enhanced in the system by employing smart card technology although a further and detailed study is required.

Recommendations for policy regarding physical security of server, configuration control, access control over users and client machines as well as regular checks by system administrators have been discussed. Login passwords in combination with public key
cryptographic techniques and random nounces are used to achieve two way authentication and safeguard against replay attacks. Furthermore, the use of this authentication process prevents customers from repudiating usage of the system and facilitates proper charging via the login audit log mechanism. At the same time, this also permits service engineers to access the customer support system from a customer’s location without incurring unnecessary cost on the customer. Together with the use of the session key, integrity and confidentiality of the customer support system can be enforced. Risk of business espionage and/or sabotage can be reduced if not eliminated as a result. Access controls over legitimate users are achieved via the control mechanisms inherent to the operating system of the server. The implementation structure permits change of the encryption algorithm (e.g. longer keys) to enhance security when necessary. We adopted the philosophy that if the information is worth less than what it would cost to protect, then it is probably safe. Nevertheless, we were careful in the choice of the economically feasible encryption scheme.

Although the approach (along with the guidelines proposed earlier in the paper) adopted in the current work will not make the Web-based customer support system perfectly secure, it nonetheless provides the security features which in our opinion, satisfies the security requirements specified by MNCS.

References


