A STUDY ON CRYPTOGRAPHY AND DIGITAL SIGNATURE ISSUES AND CERTIFIED DIGITAL SIGNATURE

Amit Kumar Pathak[1], Dr. Kalpana Sharma[2], Dr. R K Sharma[3]
[1]Research Scholar, Deptt. of CSE, Bhagwant University, Ajmer, Rajasthan
[2]Assistant Prof., Deptt. of CSE, Bhagwant University, Ajmer, Rajasthan
[3]Associate Professor , Faculty of Engg. Agra College , Agra

Abstract

In this paper we are study on cryptography and digital signature issues and certified digital signature. Electronic commerce is taking place in an increasing manner via the internet, which is becoming the fastest growing communication channel this century. As has been stated, public key encryption techniques ensure financial institutions that their messages are secure and that other transacting parties with them are authenticated. Using this technology senders and receivers of electronic messages each possess two keys, one of which is never shared with anybody and the other of which is shared with everyone. Public key cryptosystems are one of the most essential parts of modern communication frameworks. Traditional public key cryptography requires each party which wants to send an encrypted message or a signed document should generate its own public key/private key pair. However the public of an entity needs to be authenticated by means of digital certificate which is provided by a recognized certification authority. However this method of attaching a certificate to authenticate a public key incurs unnecessary bandwidth and computation overhead for a wireless communication device that has a greater limitation in terms of computational power and speed of data transmission.

1.1 Introduction

Cybertrade, or electronic commerce, has taken place for many years in the form of electronic data interchange (EDI) although this has been restricted to a few sectors of industry such as banking, insurance, shipping and the motor trade. Where EDI is used it is confined to a closed user group where the parties have probably enjoyed a good working relationship and already trust each other. EDI is usually regulated by the use of an underlying interchange agreement setting out the contractual and commercial obligations between the parties and data transfer will most likely consist of the processing of purchasers orders and invoices and the acknowledgements of orders. In the case of banking, EDI arrangements exist in the form of SWIFT messaging and the CHAPS system and in the case of the insurance sector EDI networks such as LIMNET and the more recently established WIN. All of these types of closed networks still have the potential for fraud and loss but in more open networks where third parties become involved in the transaction then there is a greater need for security in order to avoid fraud, dispute or loss.

Electronic commerce is taking place in an increasing manner via the internet, which is becoming the fastest growing communication channel this century. The exponential growth of the internet is causing businesses to rush into cyberspace because they perceive the net as a Klondike of digital opportunity where riches and opportunities await early web site prospectors. In the foreseeable future, the internet is perceived as the replacement for our traditional method of communication. There are, however, increasing concerns within the banking, insurance and other financial institutions over the security surrounding electronic commerce, where often the parties have neither dealt with each other before nor is the internet secure for the purposes of confidentiality and message integrity. The main reason for electronic commerce security is to protect the integrity of an electronic message between the time it leaves the senders application system and the point it arrives on the receiving parties application system. This is known as "end to end security". It should be noted that this only covers the integrity of the message itself and does not deal with the integrity of the respective parties applications systems.

Apart from message integrity, security should extend to issues such as:-

• The repudiation by one party of receipt or dispatch of a message,

• the risk of hacking or masquerading,
• message modification or tampering, and
• message corruption or loss.

The security of messages and their authentication can be helped by the use of encryption techniques and digital signatures, all part of the science of cryptography.

1.1.1 Cryptography

The increasing use of open network communication systems, as opposed to closed networks, poses significant challenges to the implementation of global secure electronic commerce. Among the most significant problem in this area is in relation to information security. The use of the internet for electronic trade poses the challenge of "many to many" transactions.

A secure electronic commerce system requires the implementation of cryptography - the art or science of keeping a message secure. Cryptography can be used to:-

• hide information content:
• establish authenticity
• prevent undetected modification
• prevent repudiation, and
• prevent unauthorised use.

Cryptography can be used to protect the confidentiality of financial data or personal records whether it is in storage or in transit. Cryptography can also be used to verify the integrity of data by revealing whether it has been altered and identifying the person or device that sent it. Encryption techniques are critical to the development and use of national and global information and communication networks and technologies as well as the development of electronic commerce.

1.1.2 Encryption

Encryption is the technique by which clear text is scrambled into incomprehensible text and only decipherable by the holder of a secret code key. The method by which electronic messages are scrambled and unscrambled is through a mathematical formula or algorithm and a number of internationally recognised algorithms are used combining business data with small secret sections of data known as keys. The real value of an encryption system depends upon how difficult it is to unlock without the key and several supposedly secure systems have been cracked. Those that are deemed to be totally secure pose their own security problems for government and security agencies who resent the potential transmission of subversive messages in cyberspace. Two of the most widely used and accepted algorithms are DES (Data Encryption Standard) and RSA (named after its creators Rivest Shamir and Adelman) DES is a symmetric algorithm in that both parties know the same secret key whereas RSA is an asymmetric algorithm because different keys are required for encryption and decryption. Asymmetric algorithms usually have three code numbers in effect because apart from the private keys of sender and receiver, there is a common modulus and this means it is more secure because even with the possession of the common modulus it is hard to calculate either private key. As a result of the relatively secure encryption techniques that exist today, government agencies wish secret keys to be lodged with trusted third parties.

1.1.3 Digital Signatures

As has been stated, public key encryption techniques ensure financial institutions that their messages are secure and that other transacting parties with them are authenticated. Using this technology senders and receivers of electronic messages each possess two keys, one of which is never shared with anybody and the other of which is shared with everyone. These two keys correspond to each other so that whatever is encoded with one key can only be decoded by the other. During the encrypting process the sender of the message encodes the message with the recipient's public key, making it impossible for any party other than the one holding the private key to decrypt the message. Encryption, therefore, protects the message from all parties other than the recipient without the recipient having to divulge his private key to the sender. If the process above is reversed then public key cryptography also provides a highly dependable authentication mechanism which has come to be known as a digital signature. A digital signature is an electronic substitute for a manual signature which serves the same functions as a manual signature and more. It is an identifier created by a computer instead of a pen. In technical terms a digital signature is the sequence of bits that is created by running an electronic communication through a one way hash function and then encrypting the resulting message digest with the senders private key.

Digital signature is not a digitised image of a hand-written signature or a typed signature. A digital signature is unique for each document signed because it is derived from the document itself and any change to the document will produce a different digital signature.

A digital signature can serve the same purposes as a hand-written signature in that it may signify authorship, acknowledgement or assent but a digital signature also serves important information security purposes that hand-written signatures cannot. Assuming an asymmetric system is in place then in order to digitally sign a message the sender will run a computer program that creates a unique message digest or hash of the communication and the program then encrypts the resulting message digest using the sender's private key and the resulting encrypted message digest is the digital signature. The sender then attaches the digital signature to the communication and sends both to the intended recipient.

When the recipient gets the digitally signed communication in encrypted form the recipient's computer runs a program containing the same cryptographic algorithm and hash function that the sender used to create the digital signature and the program automatically decrypts the digital signature using the sender's public key. Therefore, if the program is able to decrypt the digital signature, the recipient knows that the communication came from the sender since only the sender's public key will decrypt the digital signature encrypted with the sender's private key.

The program then creates a second message digest of the communication and compares the decrypted message digest with the digest the recipient created. If the two messages digests match the recipient knows that the communication has not been altered or tampered with thereby verifying the integrity of the message. Because digital signatures are difficult to forge their use binds the signatory precluding a later repudiation of the document. Digital signature technology also forms the basis for formally legally binding contracts in the course of electronic commerce, since they provide electronically the same forensic effect that a traditional paper document and a hand written signature thereon provides.

1.2 Proposals in Europe and the US for Public Key Infrastructures

A hand written signature is unique to its creator and may be verified by such signature taking place (physically) before the party relying upon the signature or alternatively in the presence of witnesses, or a notary. In the case of a digital signature, this is merely a large strings of bits and is created in cyberspace and not in the presence of the party relying upon it. In an asymmetric crypto system or public key system there is a need to verify that the public key is not compromised.

If public key cryptography is to succeed as a method for secure electronic commerce on a global basis, then there is a need to establish a public key infrastructure (PKI) under which trusted third parties (TTP) or certification authorities (CA) would be established to hold keys in order to provide confirmation that the holder of a public key is who he or she purports to be. Without a TTP certifying that a given individual is in fact the holder of a public key, it is impossible for other transacting parties on the network to know for certain that the holder of the public key is not an impostor.

The nearest existing equivalent to an independent TTP is that of the Notary and whilst in Europe the Notary is a public official appointed by the government (or in the case of the United Kingdom, by the Archbishop of Canterbury on behalf of the Queen) and is a profession with strict rules, ethics and training, in the United States the Notary is a far less regulated and qualified profession. Indeed, almost anyone can be a Notary in the United States provided that they pay a annual fee, do not have previous convictions or bankruptcy proceedings against them, and in any event are only commissioned to practice as a Notary for a limited period of time.

The American Bar Association some two or three years ago, recognised the value in creating a higher level Notary for the purposes of providing the role of TTP in the PKI. The American Bar Association within the last two years has worked closely with the European notaries in order to create a higher level Notary known as a CyberNotary and has published its Digital Signature Guidelines in 1996 .

The CyberNotary would perform the role of TTP or CA and would be required to have both legal as well as technical ability in the area of authentication of public keys and the provision of digital certificates.

The European Notaries have maintained for some while that since they are public officials and already have years of experience in the authentication and certification of documents which then have proof of law, they are the logical body to provide the TTP services required as part of the PKI.

Notwithstanding the position of the European notaries a number of governments are now proposing a regulated PKI to provide for licensed TTP's. The United Kingdom

Government has recently issued a consultation paper on "The Licensing of TTP's for the Provision of Encryption Services", and the German Government has acknowledged the use of CA's within its draft Digital Signature Bill Ordinance within Article 3 of its draft Information and Communication Services Bill. The European Commission is actively promoting a number of electronic commerce thesiss which are all intended to anticipate the creation of an European wide PKI and indeed in European Commission's responses to the OECD's proposed "Guidelines on International Cryptography Policy" (now finalised and published March 1997), the Commission have advocated a European wide TTP system. In the United States the US administration announced a plan in October 1996 to enable

the use of stronger encryption products which envisioned a world-wide PKI using key escrow and key recovery encryption services.
Despite the US administrations previous concerns on the export of strong encryption software as evidenced by the Clipper chip policy, they now are making export and reexport of DES and RSA software available, provided that the countries of receipt are not those on the export ban list such as Cuba, Libya, Iran and Iraq.
In envisaging a world-wide PKI the US administration has recognised the need to use CA's to provide key escrow and key recovery services so that government official may obtain under proper legal authority and without co-operation or consent of the user plain text or encrypted data and communications and in particular the individuals private key.

1.3 Use of Trusted Third Parties for Key Escrow and/or Key Recovery

The US, UK, Germany and European Commission all recognise that PKI will require the establishment of a regulated TTP or CA system. In the UK Government's Public Consultation paper entitled "Licensing of Trusted Third Parties for the Provision of Encryption Services" a TTP has been described as "an entity trusted by other entities with respect of security related services and activities. A TTP would be used to offer value added services to users wishing to enhance the trust and business confidence in the services they receive and to facilitate secure communications between business trading partners. TTP's should have trust agreements arranged with other TTP's to form a network, thus allowing a user to communicate security with every user of every TTP with whom his TTP has an agreement." A CA as defined under US State law (Utah State) or a Certification Officer as defined in the draft German Signature Bill is a TTP or entity that ascertains the identity of a person called a subscriber and certifies that the public key of a public private key pair used to create digital signatures, belongs to that person. There is a difference between the role of the TTP as envisioned by the UK Government's recent paper and the CA as envisioned in the German Draft Signature Bill. The difference lies not in the name of the officer or entity within the PKI but in the particular role that it is performing.
The confusion that presently arises in much draft legislation is that on the one hand governments require legitimate access to encrypted messages and therefore require private keys to be placed in escrow to be released upon the execution of warrants, whereas for the purposes of verifying ownership of public keys in the case of digital signatures, it is not government, but the parties to the transactions, that require an independent third party to provide a certification or key escrow service for the purposes of holding public keys and being capable of providing a digital certificate authenticating the identity of the owner of the public key to satisfy the requirement of the recipient of the public key.
The US administrations present encryption policy stipulates a considerable number of obligations upon a key recovery agent if it is to be licensed to provide such services and the UK Government's recent proposals for TTP's suggest that they are considering similar levels of obligation. These obligations not only include the fact that the key recovery agent must hand over private keys to Government Officials with lawful warrants but that the key recovery agents should also be independent, of good character and standing, have suitable liability insurance and must accept a relatively high level of liability in monetary terms. It is understood that a number of potential TTP's in the UK have already commented on the UK Government's proposals by indicating that the levels of responsibility placed upon them are too onerous and inevitably will place undue burdens on competent but smaller TTP's, thereby placing TTP responsibility in the hands of only the largest potential entities. Furthermore, the present proposals in the United Kingdom do not sufficiently differentiate between the roles and responsibilities of a TTP in key recovery situations as against the role of the same entity in relation to the provision of digital certificates and public key authentication.

1.4 OECD/ ICC Proposals on Cryptograph Issues

In the BIAC/ICC "Joint Discussion Paper on International Cryptography Guidelines" submitted to OECD in 1996 in advance of the OECD "Guidelines on International Cryptography Policy" on the 27th March 1997 it was stated:- "International business is demanding seamless webs of communication networks whereby information can flow in a free and secure manner. Secure world-wide communications are critically important as intruders, criminals, competitors and other unauthorised parties find increasingly sophisticated tools to violate the privacy and security of communications. Business needs internationally accepted means for ensuring the confidentiality, integrity and availability of communications that permit the necessary compatibility of interoperability between different security techniques. Encryption is currently the most appropriate means to secure integrity. An internationally accepted and comprehensive security policy is essential and needed urgently for business to operate in a global marketplace".

The OECD has recently published its "Guidelines on International Cryptography Policy" in which the value of an international PKI is acknowledged but, some doubt is thrown upon such a mandatory system in the case of digital certification but is accepted in principle in relation to key recovery. The OECD Guidelines are precisely that and acknowledge that they can only have real value when they are implemented in national legislation of participating member countries.

The ICC has been actively involved in the area of cryptography issues for many years since clearly cryptography and electronic commerce itself impacts upon international trade and in 1995 set up a specific thesis to look at the whole issue of electronic commerce (Thesis E-100). Thesis E-100 consisted of six working parties, namely:-

- Open account trading
- Electronic credit
- Electronic transport documents
- Legal and regulatory
- ETERMS
- Digital authentication

In April 1997 Thesis E-100 changed its name to the Electronic Commerce Thesis and reduced the number of working parties to:-

- Electronic trade practices
- Information security
- ETERMS

The objectives of the Electronic Commerce are the development of international rules and guidelines for electronic commerce and the development of business services in the field of electronic commerce. Particularly, the Electronic Commerce Thesis expects shortly to publish its Uniform International Authentication and Certification Practices (UIACP) in order to establish a general framework for digital authentication and certification of digital messages based upon existing authentication and certification law and practice in different legal systems. In so doing the UIACP will provide a detailed explanation of authentication and certification principles, particularly as they relate to information system security issues and public key cryptographic techniques.

The underlying policies articulated and promoted in the UIACP are:-

- To enhance the ability of the international business community to execute secure digital transactions.
- To establish legal principles that promotes trustworthy and reliable digital authentication and certification practices.
- To encourage the development of trustworthy digital authentication and certification systems.
- To protect users of the digital information infrastructure from fraud and errors.
- To balance digital authentication and certification technologies with existing policies, laws, customs and practices.
- To define and clarify the duties of participants in the emerging digital authentication and certification system.
- To foster global awareness of developments and digital authentication and certification technology and their relationship to secure electronic commerce.

In addition, UIACP draws upon and extends existing international law treatment of digital authentication in particular that articulated in the United National Model Law on Electronic Commerce (UNCITRAL Model Law). In addition to UIACP the ETERMS Working Group within the Electronic Commerce Thesis has developed an electronic repository for the voluntary deposit of standard terms and messages and contracts used within electronic commerce and it is anticipated that in due course the ETERMS repository may be made publicly available through the ICC Web site currently under development.

The following major points are covered:

1.) A description of the Lamport-Diffie one time signature.
2.) An improved version of the Lamport-Diffie one time signature.
3.) A method of converting anyone time signature into a convenient signature system.

1.5 The Lamport-Diffie One Time Signature

The Lamport-Diffie one time signature [6] is based on the concept of a one way function [7], [38], If y = F(x) is the result of applying the one way function F to input x, then the key observation is: The person who computed y = F(x) is the only person who knows x. If y is publicly revealed, only the origi.nator of y knows x, and can choose to reveal or conceal x at his whim. This is best clarified by an example. ~uppose a person A has some stock, which he can sell at any time. A might wish to sell the stock on short notice, which means that A would like to tell his broker over the phone. The broker, E, does not wish to sell with only a phone call as authorization. To solve this problem, *A* computes y = F(x) and gives y to E. They agree that when *A* wants to sell his stock he will reveal x to B. (This agreement could be formal ized as a wrHten contract [17] which includes the value of y and a description of F but

not the value of x will then be able to prove that A wanted to sell his stock, because B will be able to exhibit x, and demonstrate that F(x) = y. If *A* later denies having ordered E to sell the stock, B can show the contract and x to a judge as proof that A, contrary to his statement, did order the stock sold. Both F and y are given in the original (written) contract, so the judge can compute F(x) and verify that it equals y. The only person who knew x was A, and the only way E could have learned x would be if A had revealed x. Therefore, *A* must have revealed x; an action which by prior agreement meant that A wanted to sell his stock. This example illustrates a signature system which "signs" a single bit of information. Either A sold the stock, or he did not. If A wanted to tell his broker to sell 10 shares of stock, then A must be able to sign a several bit message. In the general Lamport-Diffie scheme, if *A* wanted to sign a message m whose size was s bits, then he would precompute F(x1) = Y1' F(x2) = Y2' F(x3) = Y3'··· F(xs) = Ys· A and B would agree on the vector Y = Y1' Y2 ••• Ys. If the jth bit of m was a 1, A would reveal x.. If the jth bit of m was a 0, A would not J reveal In essence, each bit of m would be individually signed. Arbitrary messages can be signed, one bit at a time. TN practice, long messages (greater than 100 bits) can be mapped into short messages (100 bits) by a one way function and only the short message signed. We can therefore assume, without loss of generality, that all messages are a fixed length, e.g., 100 bits.

The method as described thus far suffers from the defect that B can alter m by changing bits that are 1's into a's. B simply denies he ever received x., (in spite of the fact he J did). However, O's cannot be changed to, 'so Lamport and Diffie overcame this problem by signing a new message m', which is exactly twice as long as m and is computed by concatenating m with the bitwise complement of m. That is, each bit mj in the original message is represented by two bits, m. and the compleJ ment of mj in the new message m'. Clearly, one or the other bit must be a O. To alter the message, B would have to turn an o into a “something he cannot do. It should now be clear why this method is a "one time" signature: Each Y = y" Y 2' ••• Y 2 s can only be used to sign one message. If more than one message is to be signed, then new values Y" Y2, Y3, ••• are needed, a new Yi for each message.

One time signature are practical between a single pair of users who are willing to exchange the large amount of data necessary but they are not practical for most applications without further refinements. If each y. is '00 bits long and a1 100 bit one way hash function of each message is signed, each Yi must be 20,000 bits. If 1000 messages are to be signed before new public authentication data is needed, over 20,000,000 bi ts or 2.5 megabytes must be stored as public informat.ion. Even if this is not overly burdensome when only two users, A and B, are involved in the signature system, if B had to keep 2.5 megabytes of data for 1000 other users, B would have to store 2.5 gigabytes of data. While possible, this hardly seems economical. With further increases in the number of users, or in the number of messages each user wants to be able to sign, the system becomes completely unwieldy. How to eliminate the huge storage requirements is a major subject of this chapter.

1.6 Hash Functions （One Way）

There are many instances in which a large data field (e.g. 10, 000 bits) needs to be authenticated, but only a small data field (e.g. 100 bits) can be stored or authenticated. It is often required that it be infeasible to compute other large data fields with the same image under the hash function, giving rise to the need for a ~ way hash function.

Intuitively, a one way hash function F is one which is easy to compute but difficult to invert and can map arbitrarily large data fields onto much smaller ones. If y = F(x), then given x and F, it is easy to compute y, but given y and F it is effectively impossible to compute x. More precisely:

1) F can be applied to any argument of any size. F applied to more than one argument (e.g. F(x1 ,x2)) is equi val ent to F appl ied to the concatenation of the arguments, i.e. F(x1 ,x2).

2) F always produces a fixed size output, which, for the sake of concreteness, we take to be 100 bits.

3) Given F and x it is easy to compute F(x).

4) Given F and F(x), it is computationally infeasible to determine x.

5) Given F and x, it is computationally infeasible to find an x' *i* x such that F(x) = F(x'). The major use of one way functions is for authentication.

If a value y can be authenticat.ed, we can authenticate x by computing F(x) = y and authenticating y. No other input x' can be found (al though they probably ex 1st) which will generate y. A 100 bit Y can authenticate an arbitrarily large x. This property is crucial for the convenient authentication of large amounts of information. Although a 100 bit y is plausible, selection of the size in a real system involves tradeoffs between the reduced cost and improved efficiency of a smaller size, and the improved security of a larger size.

Because y is used to aut.henticate the corresponding x, it would be intolerable if someone could comput.e an x' such that y = F(x) = F(x'). The fraudulent x' could be substituted for the legitimate x and would be authenticated by the same information. If y is 100 bits long, an interloper must try about 2100 different values of x' before getting a value such that F(x') = y. In an actual system, F will be applied to many different values of x, producing many different values of y. As a consequence, trying fewer than 2100 different values of x will probably yield an x' such that F(x')

= y for some already authenticated y. To take a concrete example, assume F has been applied to 2^{40} different values of x, and produced 240 corresponding values of y, each of which has been authenticated. If the y's are 100 bits, then a random search over 2^{60} values of x would probably yield an x' such that y = F(x) = F(x') for some value of y. While this search is still difficult, it is easier than searching over 2^{100} different values of x. This demonstrates that y might have to be longer than expected in a heavily used system. Forcing an opponent to search over all 2^{100} different values of x would be more desirable. This can usually be done by using many different functions, F1, F2,…… The effect of using many different one way functions is to prevent analysis of F by exhaustive techniques, because each value of x is authenticated witl'1 a distinct Fi This will significantly increase security, yet requires only minor changes in implementation.

Functions such as F can be defined in terms of conventional cryptographic functions. Assume we have a conventional encryption function C(key,plaintext) which has a 200 bit key size and encrypts 100 bit blocks of plaintext into 100 bit blocks of ciphertext. (It is a common misconception that the key can be no larger than the plaintext blocksize, but as an example the DES can be regarded as having a 768 bit key and a 64 bit block size).

We first define F 0' which is simpler than F and wMch satisfies properties 2, 3, 4, and 5; but whose input x is *res*tricted to be 200 bits. We define

$F_O(X) = y = C(x,o)$

F_O accepts a 200 bit input x and produces a 100 bit output y, as desired. Furthermore, given y, the problem of finding an x' such that F(x') = y is equivalent to finding a key x' such that

y = C(x' ,*Q).* If C is a good encryption function, this is computationally infeasible.

If the input x to F is fewer than 200 bits, then we Can "pad" x by adding 0' s until it is exactly 200 bits, and then define $F = F_o$. If the input is more than 200 bits, we will break it into 100 bit pieces. Assume that

$\underline{x} = x_1, x_2 \ldots\ldots\ldots\ldots x_k$

and that each x1 is 100 bits long. Then F is defined in terms of repeated applications of F_O. F_0 is first applied to x1 and x2 to obtain y1 $= F_0$ (x1,x2) then y2= F_0 (y1,x3), y3= F_0 (y2,x4), y4= F_0 (y3,x5),……yi= F_0 (y_{i-1}, x_{i+1}) ……$y_{k-1} = (y_{k-2}, x_k)$. F(x) is defined to be Y_{k-1}; the final Y in the series. If x is not an exact multiple of 100 bits, then it is padded with 0's, as above.

It is obvious that F can accept arbitrarily large values for x. Although complexity theory has not progressed to the point where it is possible to prove that it will be computationally infeasible to find any vector x' not equal to x such that F(x) = F(x'), a pausibility argument cen be made inductively that this is the case. As a basis, when k = 2, the property holds because F(x) = F 0 (x1, ,x2), and the property holds for F_O by assumption. We establish the case for k = 3 by contradiction.

We assume that F(x1,x2 ,x3) = F(x1',x2 ',x3 ') and that Xi *is not equal to* Xi' for some i in (1,2,3) We first note that F(x1,x2 ,x3) = F_O(Y1, X3) by definition. If either y1 is not equal to y2 or

X3 *is not equal to X3'* and F_O(y1, x3) = F_O(y1',x3 '), then we have violated assumption 5 made about F_O. If y1 = y,1 and x3 = x3' then either x1 *is not equal to* x1' or x2 *is not equal to* x2 '. By definition F_O(x1 ,x2) = y1 and F_O(x1 ',x2 ') = y1', so F_O(x1, x2) = F_O(x1 ',x2 ') and again we contradict assumption 5 *made* about F_O. This line of logic can be extended to the cases k = 3. 4, 5, This argument cannot be made fully rigorous until the properties of F_0 are *made* rigorous. This must await further advances in complexity theory.

1.7 An Improved One Time Signature

This section explains how to reduce the size of signed messages in the Lamport-Diffie method by almost a factor of 2. As previously mentioned the Lamport-Diffie method solves the problem that 1 's in the original message can be altered to O's by doubling the length of the message, and signing each bit and its complement independently. In this way, changing a 1 to a 0 in the new message, m', would result in an incorrectly formatted message, which would be rejected represents a solution to the problem:

In essence, this Create a coding scheme in which accidental or intentional conversion of *1's* to O's will produce an illegal codeword.

An al ternati ve coding method which accompl ishes the same result is to append a count of the number of 0 bits in m before signing. The new message, m', would be only log2 s bits longer than the original s bit message, m. If any *1's* in m' were changed to O's (O's cannot be changed to 1's), it would falsify the count of O's. Notice that while it is possible to reduce the count by changing 1 's to O's in the count field, and while it is possible to increase the number of O's by changing 1 's to *O's* in the message, these two "errors" cannot be made to compensate for each other. A small example is in order. Assume that our messages are 8 bits long, and that our count is 10g2 8 = 3 bits long. If our message m is

m = 11010110

Then m' would be

m' = 11010110,011
(Where a comma is used to clarify the division of m' into m and its 0 count)
If the codeword 11010110,011 were changed to 01010110,011 by changing the first 1 to a 0, then the count 011 would have to be changed to 100 because we now have 40's, not 3. Put this requires changing a 0 to a 1, something we cannot do. If the codeword were changed to 11010110,010 by altering the 0 count then the message would have to be changed so that it had only 20's instead of 3. Again, this ch8nge is illegal because it requires changing O's to 1 'so This improved method is easy to implement and cuts the size of the signed message almost in half, although this is still too large for most applications; e.g., it reduces 2.5 gigabytes to 1.25 gigabytes.

1.8 Tree Authentication

A new protocol would eliminate the large storage requirements and the need for prior arrangements. If A transmitted Y1 to B just before signing a message, then B would not previously have had to get and keep copies of the Y. from A. Unfortunately, such a protocol would not work because anyone could claim to be A, send a false Y., and tr ick B into thinking he had reI ceived a properly authorized signature when he had received nothing of the kind. B must somehow be able to confirm that he was sent the correct Y. and not a forgery. The problem is to authenticate A's Y The simplest (but unsatisfactory) method is, as suggested above, to keep a copy of A's Yi • in this section, we describe a method called "tree authentication" which can be used to authenticate any Y. of any user quickly and easily, but which requires minimal storage. Problem Definition: Given a vector of data items! = Y1,.Y2, ••• Yn design an algorithm which can quickly authenticate a randomly chosen Y. but which has modest memory requirements, i.e., does not have a table of Y1, Y2, •.• Yn •
We authenticate the Y. by "divide and conquer". As illustrated in figure 7.1, define the function H (I, j , y) by
1. H(i, i , y) = F(yi)
2. H(i, j, y) = F{H (I,k,y), H(k+1, j, y)}
where k = (i+j)/2 for the <i ,j> pairs of figure 4.1 and its extensions to larger trees described below.
H(i,j,y) is a function of Yi., Y_{i+1}. ... Yj. and can be used to authenticate all of them. H(1,n,y) can be used to authenticate Y_1 through Yn and is only 100 bits, so it can be conveniently stored. We restrict n to powers of 2 to simplify the explanation. "
This method lets us selectively authenticate any "leaf," Y i' that we wish. To see this, we use an example where n = 8. To authenticate Y5 , we proceed in the manner illustrated in figure 4.2:
1.) H(1,8,y) is already known and authenticated.
2.) H(1,8,y)= F(H(1,4,y) H(5,8,y)). Send H(1,4,y) and H(5,8,y) and let rhe reciver compute H(1,8,y)= F(H(1,4,y) H(5,8,y)) to confirm that they are correct
3.) The receiver has authenticated H(5, 8,y). Send H(5,6,y) and H(7,8,y) and let the receiver compute

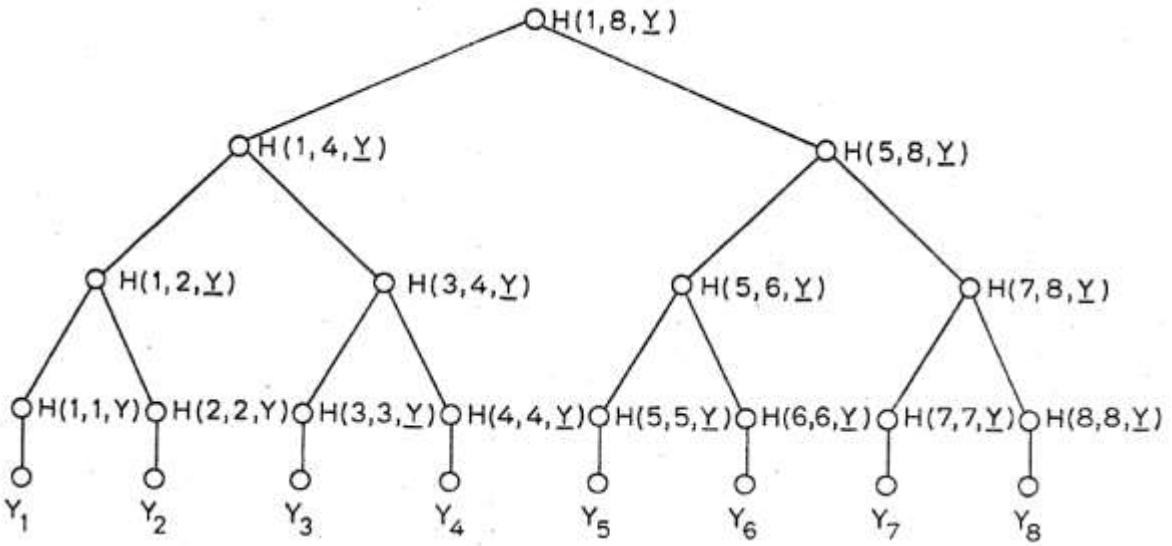

Figure4. 1: An Authentication Tree with N = 8.

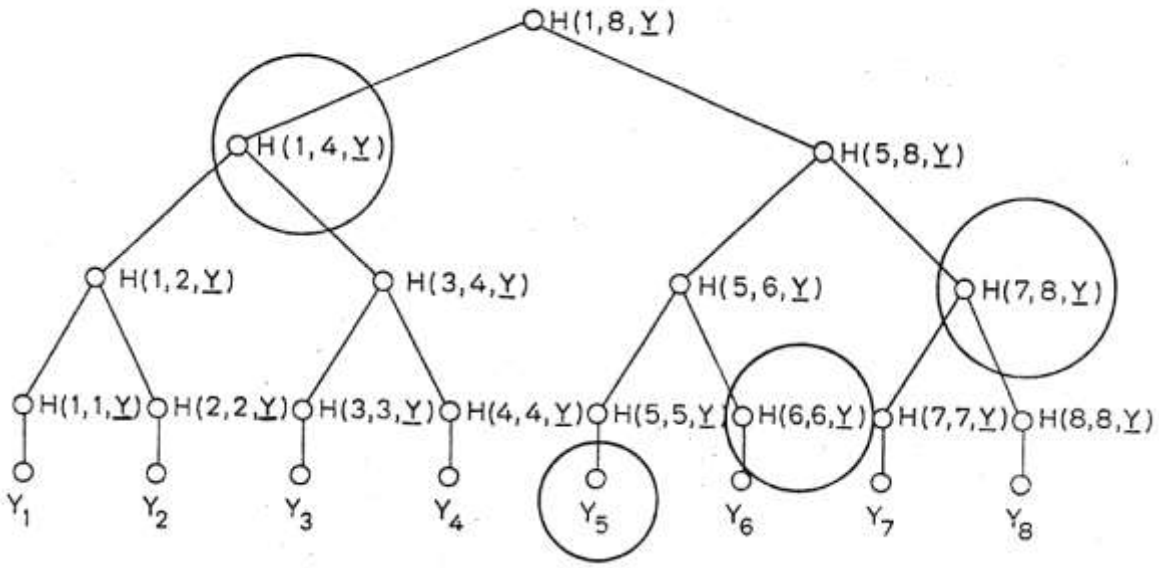

Figure1. 2: Circled Entries Show the Authentication Path for Y5

H(5,8,y) = F(H(5,6,y), H(7,8,y)) to confirm that they are correct.

4.) The receiver has authenticated H(5,6,y). Send H(5,5,y) and H(6,6,y) and let the receiver compute H(5,6,y) = F(H(5,5,y), H(6,6.y) to confirm that they are correct.

5.) The receiver has authenticated H(5,5,y). Send Y5 and let the receiver compute H(5,5,y) = F(Y5) to confirm that it is correct.

6.) The receiver has authenticated Y5 •

Uslng this method, only log2 n transmissions are required, each of about 200 bits. Close examination of the algorithm will reveal that half the transmissions are redundant. For example, H(5,6,y) can be computed from H(5,5,y) end H(6,6,y), so there is really no need to send H(5,E,y). Similarly, H(5,8,y) can be computed from H(5,6,y) and H(7,8,y), so H(5,e,y) need never be transmitted, either. (The recel.ver must compute these quantities anyway for proper authentication.) Therefore, to authenticate Y5 required only that we have previously authenticated H(1,8,y), and that we transmit Y5 , H(6,6,y), H(7,8,y), and H(1,4,y). That is, we require 100 log2 n bits of information to authenticate an arbitrary Yi. The method is called tree authentication because the computation of H(1,n,y) forms a binary tree of recursive calls. Authenticating a particular leaf Y. in the tree requires only those values of H() starting from the leaf and progressing to the root, i.e., from H(i,i,,y) to H(1,n,,y). H(1,n,,y) will be referred to as the root of the authentication tree, or R. The information near the path from R to H(i, i ,1) required to authenticate Yi will be called the authentication path for Yi. A "proof" that the authentication path actually authenticates the chosen leaf is similar to the "proof" that F defined correctly authenticates its input. Again, more rigorous proofs must await advances in complexity theory. Although H() produces a 100 bit output, unless additional

precautions are taken, only 260 or 270 operations would suffice to break the system. To force the cryptanal ysis to use 2100 operations it is necessary to make each application of F unique, i.e., to use a family Of one way functions F1, F2 , ••• each one of which is used only once. The use of tree authentication is now fa irly clear. A transmits Y. to B. A then transmits the authentication path for Y. B knows R, the root of the authentication tree, by prior arrangement. P can then authenticate Y., and can accept the ith signed message from A as genuine. The prior arrangements include the computation of R by A. If A wishes to be able to sign 1,000,000 messages, this precomputation will require about an hour, assuming a single encryption takes 10 microseconds. (Fairchild is now (1979) producing a 4-chip set which costs about $100 and which encrypts faster than this.) The time required for the pre-computation is linear in n, so if A desires to be able to sign 1,000,000,000 messages, his pre-computation will be about 1000 hours.

The major distinction between this method and digital signatures generated using publ ic key cryptosystems is the requirement that R be changed periodically because only n messages can be signed. With a publ ic key cryptosystem, it is possible to sign an almost indefinite number of messages, and for a user to retain DA for his I ifetime if he so desires. In practice, this restriction does not appear to be significant. If the jth user has a distinct

authentication tree with root R j' then tree authentication can be used to authenticate R j just as easily as it can be used to authenticate Yi • It is not necessary for each user to remember all the R. in order to J authenticate them. A central clearinghouse could accept the Rj from all u users, and compute H(1 ,u,R). This single 100 bit quantity could then be distrihuted and would serve to authenticate all the R., which would in turn be used to authenticate J the Yi • In practice, A would remember R A and the authentication path for RA and send them to P along with Yi and the authentication path for Yi • (A different method of authentication would be for the clearinghouse to digitally sign "letters of reference" for new users of the system using a one time signature. Tree authentication and authentication using one time sig- natures can be intermixed to produce systems with all the flexibility of public key based systems.

1.9 The Path Regeneration Algorithm

A must know the authentication path for Yi before transmitting it to B. Unfortunately this requires the computation of H(i ,j ,y) for many different values of i and j. In the example, it was necessary to compute H(6,6, y), H(7.8,y), and H(1,4,y) and send them to B along with Y5 This is simple for the small tree used in our example, but computing H(4194304,8388608, y) just prior to sending it would be an intolerable burden. One obvious solution would be to precompute H(1,n,y) and to save all the intermediate computations: i.e., precompute all authentication paths. This would certainly allow the quick regeneration of the authentication path for Yi , but would require a large memory.

A more satisfactory solution is to note that we wish to authenticate Y1, Y2 , Y3 , Y4 , ••• in that order. Most of the computations used in reconstructing the authentication path for Yi can be used in computing the authentication path for Y_{i+1}. Only the incremental computations need be performed, and these are quite modest. In addition although the Yi must appear to be random, they can actually be generated (safely) in a pseudo-random fashion from a small truly random seed. It is not necessary to keep the Yi in memory, but only the small truly random seed used to generate them.

The result of these observations is an algorithm which can recomputed each Yi and its authentication path quickly and with modest memory requirements. Before describing it we review the problem:

Problem Definition: Sequentially generate the authentication paths for Y1, Y2, Y3, ... Yn with modest time and space bounds.

The simplest way to understand how an algorithm can efficiently generate all authentication paths is to generate all the authentication paths for a small example.

An example of all authentication paths for n = 8 is:

Leaf	**Authentication Path**
y1	H(1,8,y) H(5,8,y) H(3,4,y) H(2,2,y)
y2	H(1,8,y) H(5,8,y) H(3,4,y) H(1,1,y)
y3	H(1,8,y) H(5,8,y) H(1,2,y) H(4,4,y)
y4	H(1,8,y) H(5,8,y) H(1,2,y) H(3,3,y)
y5	H(1,8,y) H(1,4,y) H(7,8,y) H(6,6,y)
y6	H(1,8,y) H(1,4,y) H(7,8,y) H(5,5,y)
y7	H(1,8,y) H(1,4,y) H(5, 6,y) H(8,8,y)
y8	H(1,8,y) H(1,4,y) H(5,6,y) H(7,7,y)

Table 4.1

If we had to separately compute each entry in table 4.1, then it would be impossible to efficient]y generate the authentication paths. Fortunately, there is a great deal of duplication. If we eliminate all duplicate entries, then table 1 becomes table 4.2:

Leaf	Authentication Path
y1	H(1,8,y) H(5,8,y) H(3,4,y) H(2,2,y)
y2	H(1,1,y)
y3	H(1,2,y) H(4,4,y)
y4	H(3,3,y)
y5	H(1,4,y) H(7,8,y) H(6,6,y)
y6	H(5,5,y)
y7	H(5,6,y) H(8,8,y)

y8	H(7,7,y)

TABLE 1.2

Clearly we can generate all authentication paths by separately computing each of the 2 n-1 entries in table 4.2, but this would require too much memory, and it is not clear what the execution time would be. We first consider the execution time, the memory requirement will be considered later. Because all computations must eventually be defined in terms of the underlying encryption function C(key ,plaintext), it seems appropriate to define execution time requirements in terms of the number of applications of C. One application of C counts as one "unit" of computation. We shall call this "unit" the "et," (pronounced eetee) which stands for "encryption time." Computing F requires a number of ets proportional to the length of its input. In particular, if the input is composed of 100 k bits, then F requires k-1 ets.

First, we must determine the cost of computing the individual entries. The algorithm for computing H(i,j,y) from y does a tree traversal of the subtree whose leaves are Yi, Yi+1' Yi+2 ••• Yj. At each non-leaf node in this traversal it does 1 et of computation (one application of F to a 200-bit argument). A tree with j-i+1 leaves has j-i non-leaf nodes, i.e.,j-i nodes internal to the tree. For example, a tree with 8 leaves has 4 + 3 + 2 + 1 = 10 internal nodes. Because there are j-i non-leaf nodes, computing H(i ,j ,y) requires j-i ets, excluding the leaves. The computations required to regenerate a leaf (using a truly random seed in a pseudo random number generator) will be fixed and finite. Let r be the (fixed) number of ets required to regenerate a leaf. There are (j-i+1) leaves, so the overall cost of computing H(i,j,y) is (j-i) + (j-i+1) • r ets. In practice r will be a few hundred, so we can approximate this by (j-i+1) • r ets. We can now approximate the cost of computing each entry in table 2. There are n entries which require about r ets, n/2 entries which require about 2 r ets, *n/4* entries which require about 4 r ets, and *n/8* entr ies which require about 8 r ets. This means that the total cost of computing all entries in a single column is about 8 r ets. There are 4 columns, so the total computational effort is about 4·8 r = 32 r ets. In general, the computational effort required to compute table 2 will be n • (1 + log2 n) • r ets. This is because computing all the entries in each column will require n • r ets, and there are 1+ log2 n columns. This result impl ies that an algorithm which sequentially generated the authentication paths would require an average of about

$$r \bullet \log_2 n \qquad (4.1)$$

ets per path, where r is a constant representing the number of ets required to regenerate a leaf. This is quite reasonable. Although the time required to generate each authentication path is small, we must also insure that the space required is small and that the computational burden is smoothly distributed as a function of time. We can do this by again looking at table 4.2. As we sequentially generate the authentication paths, we will sequentially go through the entries in a column. This implies that at any point in time there are only two entries in a column of any interest to us: the entry needed in the current authentication path, and the entry immediately following it. We must know the entry in the current authentication path, for without it, we could not generate that path. At some point, we will need the next entry in the column to generate the next authentication path. Because it might require a great deal of effort to compute the next entry (e.g. to compute H(1,4,y)) we need to compute it incrementally, and to begin computing it well in advance of the time we will actually require it to generate an authentication path.

As an example, H(5,8,y) is required in the authentication paths for Y1, Y2 , Y3, and Y4 while H(1,4,y) is required in the paths for Y5, Y6 , Y7, and Y8" The values of H( ) for the first authentication path must be precomputed with some del ay (discussed below). Once this precomputation is complete, the succeeding values of H() required in succeeding authentication paths must be incrementally computed. As we generate the first 4 authentication paths, we must be continuously computing H(1,4,y) even though it is not needed until we reach Y5. If we waited until time 5 to start computing it, it would take about 4 r ets to compute and entail some delay. By computing H during times 1 through 4, a processor capable of only r ets/unit time is needed. In general, if the tree is of depth k it will take 2^{k-1} • r ets to compute the second element in the second column, but there are 2^{k-1} time units in which to compute it, again requiring a processor capable of only r ets/unit time. Similarly, we must start computing the second element in the third column, H(1, 2,y). when we generate the first authentication path. It takes about 2 r ets to compute this element (2^{k-2} in general) but there are 2 time units (2k -2 in general) in which to do this. So the processor for computing entries in the third column also needs to operate at only r ets/unit time. It is seen that it takes 2^{k-i-1} • r ets to compute in the next entry ith the column and that there are 2^{k-i-1} time units in which to do this. Thus. only one processor is needed per column (log2 n in all) and each processor need operate at only r ets/unit time.

If we assume a convenient block size (of 100 bits) and if we ignore constant factors. Then the memory required by this method can be computed. We can first determine the memory required by the computations in each column, and then sum over all log2 n columns. We must have one block to store the current entry in the column. We must al so have enough memory to compute the next entry in the column. The memory required while computing H(i,j,y) is 1 + log2 (j-i+1) blocks. This assumes a straightforward recursive algorithm whose maximum stack depth will be 1 + log2 (j-i+1). The memory required to recomputed a leaf (to recompute H(i,j,y)) is ignored because it is small (a few

blocks) constant and the same memory can be shared by all the columns. Representing the memory requirements of H() in a new table in the same format as table 4.2 gives table 4.3:

leaf	**memory required to compute entries in authentication path (in blocks)**			
y1	4	3	2	1
y2				1
y3			2	1
y4				1
y5		3	2	1
y6				1
y7			2	1
y8				1

TABLE 1.3

Table 3 shows the memory required to compute each entry in table 4.2. Clearly the memory required to compute H(i,i,y) is 1. The memory required to compute H(1,2,y) = 1 + the memory required to compute H(1,1 ,y) since we first compute B(1,1 ,y) and must remember it to compute H(1,2,y). Similarly, to compute H(1,2t ,y) requires one more memory location than was needed for H(1,2 t-1 ,y). The memory required for each column will be about the memory required during the computation of a single entry in the column because once an entry is computed, the memory is available to compute the next entry and the old entry is discarded after use. This means the total memory required will be about: 3 + 2 + 1 = 6 blocks. (This assumes we do not recomputed H(1,8,y))·

For n in general, there are log2 n columns and each column requires, on an average, (log2 *n)/2* blocks so the total memory required will be on the order of:

2 (log2 n) *12* blocks This means that the memory required when n = 220 (1,048,576) is about *20'20/2* = 200 blocks. For 100 bit blocks, this means 20 kilobits, or 2.5 kilobytes. Other overhead might amount to 2 or 3 kilobytes, giving an algorithm which requires 5 or 6 kilobytes of memory, in total. Readers interested in implementing this technique can use the following program, written in a Pascal-like language with two multiprocessing primitives added:

1.) While <condition> wait

2.) Fork <statement>

In addition, the function "MakeY(i)" will regenerate the value of Yi from the truly random seed.

```
Declare flag: array[O •• log2(n)-1] of integer;
AP: array[0 •• log2(n)-1] of block;
(* AP -- Authentication Path *)
Procedure Gen(i);
Begin
For j:= 1 to n step 2 i+1 Do
Begin
Emit(i,H(j+2i ,j+2 i +1-1));
Emit(i,H(j,j+2 i -));
End;
End;
Procedure Emit(i,value);
Begin
While flag[i] is not equal to 0 wait;
AP[i]:= value;
Flag[i]:= 2 i ;
End;
Procedure H(a,b);
Begin
If a = b Return( F(makeY(a)))
Else
Return(F(H(a,(a+b-1)/2),H((a+b+1)/2,b));
(* Note that F should be parameterized by the user's name and by a and b. If this is not done, Y must be made larger
to assure security)
End;
(* The main program *)
Begin
For i := 0 to log2(n)-1 Do
```

```
Begin
flag[i]:= 0;
Fork Gen(i);
End;
For j:= 1 to n Do
Begin
Print("Authentication Path ", j, " is:");
For k := 0 to 10g2(n)-1 Do
Begin
While flag[k] = 0 wait;
Print(AP[k]);
flag[k]:= flag[k]-1;
End;
End;
End;
```

The general structure of this program is simple: the main routine forks off log2 n processes to deal with the log2 n columns. Then it prints each authentication path by sequentially printing an output from each process. The major omission in this program is the rate at which each process does its computations. It should be clear from the previous discussion that each process has adequate time to compute its next output. There are three major ways of improving this algorithm. First, each process is completely independent of the other processes. However, separate processes often require the same intermediate values of H(), and could compute these values once and share the result. Second, values of H() are discarded after use, and must be recomputed later when needed. While saving all values of H() takes too much memory, saving some values can reduce the computation time and also reduce memory requirements. The reduction in memory is because of the savings in memory when the saved value is not recomputed. Recomputing a value requires memory for the computation, while saving the value requires only a single block. Finally, the memory requirements can be reduced by carefully scheduling the processes. While it is true that each process requires about log2 n blocks of memory, this is a maximum requirement, not a typical requirement. By speeding up the execution of a process when it is using a lot of memory, and then slowing it down when it is using little memory, the average memory requirement of a process (measured in blockseconds) can be greatly reduced. By scheduling the processes so that the peak memory requirements of one process coincide with the minimum memory requirements of other processes, the total memory required can be reduced.

All three approaches deserve more careful study because the potential savings in time and space might be large. Even without such improvements the technique is completely practical. Before the time requirements of the algorithm can be fully analyzed, a description of MakeY is needed: i.e., we must determine r in equation (5.1). If we assume that the improved version of the Lamport-Diffie algorithm is used, then MakeY must generate pseudo-random Xi vectors, from which Yi vectors can then be generated. If the one way hashed messages are all 100 bits long, then the Xi vectors will have 100 + log2 100 = 107 elements. The Xi vectors can be generated using a conventional cipher, C(key,plaintext). A single 200 bit secret key is required as the "seed" of the pseudo-random process which generates the Xi vect.ors. The output of C is always 100 bits, and the input must be 100 bits or fewer, (if fewer, O's are appended). We can now define Xi,j as

Xi,j = C (seed ke y , < i , j>) i,j

where "seedkey" is the 200 bit secret and truly random key used as the "seed" of this somewhat unconventional pseudo-random number generator. The subscript i is in the range 1 to n, while the subscript j is in the range 1 to 107. There are n possible messages, each 100 bits in length. Each Xi is a vector Xi,1, Xi,2, ••• Xi,107.

Determining any xi ,j knowing some of the other Xi ,j's is equivalent to the problem of cryptanalyzing C under a known plaintext attack. If C is a good encryption function, it will not be possible to determine any of the Xi,j. without already knowing the key. The secret vectors Xi are therefore safe.We know that Yi,j= F(Xi,j), and that H(i,i,Y) = F(Y1) = F(Yi,1, Yi,2 Yi,3 .•• Yi,107). The cost of computing F(Y i) is 106 ets, because Yi is 107, 100 bits long. The total effort to compute H(i ,i ,y) is the effort to generate the elements of the Xi vector, plus the effort to compute F(Xi , 1), F(xi , 2), ••• F(x1. ,n), plus the effort to compute F(Yi). This is 107 ets to compute the Xi vector, 107 ets to compute the Yi vector, and 106 ets to compute F(Y.) = H(i,i,Y). This is a total of 320 ets to regenerate each leaf in the authentication tree.

Using equation (4. 1), we know that the cost per authentication path is 320 log2 nets. For n = 2 20 ,this is 6400 ets. To generate authentication paths at the rate of one per second implies 1 et is about 160 microseconds. While easily done in hardware, this speed is difficult to attain in software on current computers. Reducing the number of ets per authentication path is a worthwhile goal. This can effectively be done by reducing either the cost of computing

H(i,i,y), or by reducing the number of times that H(i,i,y) has to be computed. As mentioned earlier, keeping previously computed values of H() rather than discarding them and sharing commonly used values of H() among the 10g2 n processes reduces the cost of computing each authentication path. In fact, a reduction from over 6000 ets to about 1300 ets (for n = 220) can be attained. (To put this in perspective, MakeY requires 320 ets and must be executed at least once per authentication path. Therefore, 320 ets is the absolute minimum that can be attained without modifying MakeY.) This means the path regeneration algorithm can run in reasonable time (a few seconds) even when the underlying encryption function, C, is implemented in software.

Reference

[1]. A. JOUX, A new index calculus algorithm with complexity L(1/4+o(1)) in very small characteristic, iacr e-archive 2013.

[2]. XML Encryption Syntax and Processing Version 1.1, W3C Candidate Recommendation 2011: http://www.w3.org/TR/2011/CR-xmlenc-core1- 20110303/.

[3]. EMV 4.3, "Integrated Circuit Card Specifications for Payment Systems: Book 1 Application Independent ICC to Terminal Interface Requirements, Book 2 Security and Key Management, Book 3 Application Specification, Book 4 Cardholder, Attendant and Acquirer Interface Requirements", Version 4.3, EMVCo, 2011.

[4]. A. BOGDANOV, M. KHOVRATOVICH and C. RECHBERGER, "Biclique Cryptanalysis of the Full AES", http://eprint.iacr.org/2011/449.

[5]. J. W. BOS, M. E. KAIHARA, T. KLEINJUNG, A. K. LENSTRA and P. L. MONTGOMERY, "On the Security of 1024-bit RSA and 160-bit Elliptic Curve Cryptography", version 1, July 14, 2009, https://documents.epfl.ch/users/l/le/lenstra/public/papers/ecdl.pdf.

[6]. G. BERTONI, J. DAEMEN, M. PEETERS and G. VAN ASSCHE, "The Keccak Sponge Family Function", June 2010, http://keccak.noekeon.org/Keccak-main2.1.pdf.

[7]. G. BERTONI, J. DAEMEN, M. PEETERS and G. VAN ASSCHE, "The Keccak SHA3 submission, January 14 2011, http://keccak.noekeon.org/Keccak-submission3.pdf.

[8]. G. BERTONI, J. DAEMEN, M. PEETERS and G. VAN ASSCHE, "The Keccak Reference", January 14 2011, http://keccak.noekeon.org/Keccak-reference3.0.pdf.

[9]. Gahan A V , Geetha D Devanagavi, 2019 "A Empirical Study of Security Issues In Encryption Techniques", International Journal of Applied Engineering Research ISSN 0973-4562 Volume 14, Number 5 (2019) pp. 1049-1061 © Research Europe Publications. http://www.ripublication.com

[10]. P. Ananth and V. Vaikuntanathan. Optimal bounded-collusion secure functional encryption. IACR Cryptology ePrint Archive, 2019:314, 2019.

[11]. 38. S. Kim and D. J. Wu. Watermarking PRFs from lattices: Stronger security via extractable PRFs. In CRYPTO, 2019.

[12]. Fateme Siar, Saeid Alirezazadeh, Fateme Jalali " A Novel Steganography Approach Based on Ant Colony Optimization", 2018 6th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS-2018), Vol. 3, Issue. 1, pp.95 – 99, 2018

[13]. Seyed Hossein Kamali, Reza Shakerian, Maysam Hedayati " A new modified version of AES based algorithm for image encryption" 2018 International Conference on Electronics and Information Engineering (ICEIE 2018), Vol. 4, Issue. 3, pp.161 – 165, 2018

[14]. S Usha, Sathish Kumar, K Boopathybagan " A Secure Triple level encryption method using Cryptography and Steganography " 2017 International Conference on Computer Science and Network Technology (ICCSNT2017), Vol. 2, Issue. 1, pp.74 – 79, 2017

[15]. May H.Abood " An Efficient Image Cryptography using Hash LSB Steganography with RC4 and pixel shuffling Encryption algorithms", 2017 Annual Conference on New Trends in Information & Communications Technology Applications- (NTICT'2017), Vol. 3, Issue. 2, pp.32 – 35, 2017

[16]. Shilpa Goyal, Maninder Singh Nehra " Texture based video steganography technique using block-wise encryption ", 2017 13th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS), Vol. 2, Issue. 1, pp.155 – 160, 2017.

[17]. Abdullah Al Hasib, Abul Ahsan Md. Mahmudul Haque " A Comparative study of the Performance and security issues of AES and RSA Cryptography " Third 2008 International Conference on Convergence and Hybrid Information Technology, Vol. 5, Issue. 2, pp.91 – 95, 2008


IJARIIE