SECURITY – ENCRYPTION TECHNIQUES, CASH PROTOCOLS, CASH SYSTEMS.

EXCERPTS FROM MSC THESIS
SUBMITTED FOR PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCES
IN
ENGINEERING MANAGEMENT

A STUDY ON THE INTERNET AS A COMMERCE MEDIUM

by
Rüçhan Ziya

Thesis Supervisor
Prof. Dr. M. Akif Eyler

İstanbul, 1997

Alıntılar için önsöz

1995-96 yıllarında hazırlanan ve 1997 yılında tamamlanan bu tez çalişmasından bugüne (Mart, 2002) Internet çok değişti. Gelişmeler bir anlamda 7 yıl once öngörülenlerle parallel ama farklı oldu. O zamanlar Türkiye'de Internet ile hemen hemen hiç ilgilenmeyen bankalar bugün Internet bayrağını taşıyan kuruluşlar oldular. (Ref: Rüçhan Ziya, A Study On The Internet, MSc. Thesis 1997)

En temel örnek bu kısaltmanın Garanti Bankası için hazırlanıyor olması.

İlk çalışmayı tekrar değerlendirdiğimde ogünden bugüne çok konuşulmuş olan "Internet nedir?" gibi konuları ve güncelliğini büyük ölçude yitirmiş "Gelecekte neler olabilir?" temalı kısımları atmayı tercih ettim.

Sadece, hala yararlı olabileceğini duşündüğüm, şifreleme ve elektronik para sistemlerinin genel prensiplerini inceleyen bölümleri bıraktım.

7-8 yıl öncesinde Internet'in nasıl göründüğünü merak ederek, veya başka nedenlerle (eğlence?) çalışmanın tamamına ulaşmak isteyenler Marmara Üniversitesi Kütüphanesinden veya http://www.geocities.com/rziya/commonthenet.pdf adresinden ulaşabilir veya rziya@yahoo.com elektronik posta adresinden isteyebilirler.

Rüçhan Ziya
İstanbul, 2002
rziya@yahoo.com

Table of Contents

List of Figures

4. Security - Encryption Techniques.

In this paper the term security is used as the reliability of communication between the consumer and the commercial Internet site. Due to the nature of the Internet, or generally the computer networks, a packet that contains data travels through different systems, such as routers, computers. It is possible that some third party other than the target destination can intercept the communication and examine the content of the packet. Doing so one can eavesdrop a communication session, or worse one can alter the contents of a network packet. As we'll be discussing later on, the communication between a consumer and a commercial site can contain critical information such as credit card information, or other forms of electronic payment. Access to such information presents a high security treat for both consumer and the vendor. By learning credit card details, the consumer faces the treat of being charged for purchases he did not realize. The third party can use the credit card information to purchase other items or can directly try to charge the credit card via the Banking system. Even though there are regulations and the transactions that a customer denies are thoroughly examined the situation is not different from having your credit card stolen except that you are not aware that it has been stolen as you are still possessing the card. A better analogy could be exactly replicating a card. Credit card do not have physical existence on the Internet. Anyone with the knowledge of a valid credit card number with associated name and expiry date can use the card as easily as the real owner.

It could be argued that the traffic on the Internet is really heavy and the packets containing sensitive information are not differentiated from the others so the probability of a security is very low. But such an argument does not make sense for a commercial medium. In order to solve the security issues there has been a number of initiatives and many of them are being used on the Internet today.

The answer to the security issues lies in the arena of mathematics. And the basis for establishing the authenticity or security of electronic data is the use of *digital signature*. There are complicated mathematical functions that can simulate every feature of manual signatures. These can only be produced by someone knowing a secret key, and they can be verified by anyone.

Digital signatures are often created from several different algorithms or equations. Commonly used basic algorithms are:

- **Private-Key Encryption** These algorithms scramble data that can only be understood by someone holding a single, private key which is used for both locking and unlocking data.

- **Public-key Encryption** These algorithms scramble data with two different keys. One is used for locking and the other is used for unlocking. These keys are not interchangeable. The one used for locking can not be used for unlocking. This system allows two parties to establish secure communication. Each participant publishes one of their keys (public key) and keeps the other secret (private key). Each end then uses the

public key of the recipient to encrypt messages which can only be decrypted with the corresponding private key.

In order to implement a digital signature using this method one can encrypt the signature line with his secret key which can be decrypted using the public key which has been made available. By decrypting the encrypted data using the public key one can assure that it has been signed by the owner of the public key.

The most common form of public-key encryption is RSA marketed by RSA Data Security.

- Secure Hash Functions Hash functions take a large file and reduce it to a relatively short number (128 to 512 bits) so that this short number can be used as a surrogate for the long file. These functions have two important features that make them usable in the encryption area. First, all of the possible short numbers are equally likely to emerge from the hash function which guarantees that all possible 2^n n bit numbers get used. Second, it should be impossible to recreate a file that generates any particular number.

These functions are used as surrogates to make sure that no one changes the file. After sending a file to another person two people could compare the hash values and make it sure that noone has changed the file. Files are generally digitally signed by computing the secure hash function of a file and then encrypting it with the secret key. So one could verify the signature by decrypting it using the public key and verifying the hash function of the file. This technique is often used because hash functions are faster to compute than public-key encryption. Some hash function algorithms are MD-5. and SHA (Secure Hash Algorithm).

- Signature-Only Public-Key Systems Even though standard public-key algorithms can be used to encrypt information, the governments tend to control the use of encryption. So there has been studies around public-key systems that can only be used to authenticate a signature.

Using these systems only someone who knows the right secret key can generate a digital signature and anyone can verify the authenticity of this signature. But the system theoretically can not be used to encrypt messages so governments can keep an eye on the content. But practically these systems can be modified to send encrypted messages.

Best known example of such algorithms is U.S. Government standard DSA (Digital Signature Algorithm).

- Blinded Digital Signatures These systems introduce the concept of signing, validating or authorizing data without knowing the content. Which is a practice that is not employed in the real world as nobody would sign a blank check. As we'll be discussing later these systems have use in anonymous cash systems.

- Secret Sharing In the real world there are examples of that kind of systems where banks have to keys for safe deposit boxes. These systems are used when you want *n* different parts or people to be available to open a secret key. The simplest systems split

data into *n* parts and require all *n* parts to be present for decryption. More complex systems might allow decryption when only *k (k<n)* parts are available.

These systems are used in digital cash systems to prevent double spending. A persons identity is split into two parts and if a person spend a digital bill twice you can reconstruct the identity using two pieces of information.

- Bit Commitment These protocols are used to lock up something in a way that can't be denied later, You might want to seal a prediction for the future so that people will be able to verify that you made it in the past. These algorithms are also used in digital cash systems.

- Zero Knowledge Proofs In all means of communication each side must prove that they are authentic. We do it by using our ID cards, telling our card numbers, etc. There are some problems when you start using electronic systems. You have to prove that you are authentic without revealing your ID number. These systems are neat solutions to prove that you know something without revealing information.

All of these algorithms is used in some form of digital money. Different systems use different algorithms. Generally the more privacy, anonymity a system offers more algorithms are used.

4.1. Private Key Encryption

These are the oldest and best-known forms of encryption that use private keys that are known to both the sender and the receiver. The cryptograms in newspapers are simple examples of encryption systems that use a scrambled alphabet as a key. Each "A" might be converted into a "P", each "B" becomes "M", etc. Modern computerized encryption systems use long collections of bits as the key and can be used to encrypt any digital data.

The most common private-key algorithm used today is still the DES (Digital Encryption Standard).

The basic structure of DES is now well known. Data is encrypted in 64-bit blocks with a 56 bit long key. The algorithm encrypts the data by repeating a basic scrambling in 16 rounds. In each round, the 64 bits are split into two 32-bit halves called the left and right halves. One half is scrambled by combining it with parts of the key and then passing it through a random function called *s-box*. Then the left half is used to modify the right half. [BS91].

DES gathers its strength as the basic scrambling step is repeated 16 times. The cipher gets most of its strength from the composition of the s-box which was part of the design that was classified. One pass through the s-box is simple to break, but 16 passes remains computationally infeasible.

The steps in DES are [BS91] :

1. First, 64 bits are passed through a function called the initial permutation. The order of the bits are rearranged. The first bit, for instance, is placed in the 58th bit slot. The second bit is placed in the 50th slot, and so forth. Figure 4.1 shows a complete permutation.

58	50	42	34	26	18	10	2	60	52	44	36	28
20	12	4	62	54	46	38	30	22	14	6	64	56
48	40	32	24	16	8	57	49	41	33	25	17	9
1	59	51	43	35	27	19	11	3	61	53	45	37
29	21	13	5	63	55	47	39	31	23	15	7	

Figure. 4.1 The Initial permutation for DES.

The bits with the same level of significance from each byte are lined up next to each other. The eighth bits from each byte are lined up in the first byte.

2. The 16 rounds begin. Let L_0 and R_0 stand for the left and right 32-bit halves formed by splitting up the 64 bits. Let L_i and R_i stand for the values of these halves after round i.

3. The values of L_i and R_i depend only on the values of L_{i-1} and R_{i-1}. L_i is set to R_{i-1}. R_i is computed as L_{i-1} $P(S(K_i$ $E(R_{i-1})))$. The function P is a 32-bit permutation, S is the s-box that takes 48 bits of input and scrambles it to produce 32 bits, and E is an expansion function that takes 32 bits and returns 48 bits. K_i is the key used in the *i*th round. The symbol represents the exclusive or function. The detailed steps of the process are:

(a) At the beginning of round *i*, the right half, R_{i-1} is passed through the expansion function *E.* This converts the 32-bits into a 48-bit set by duplicating some of the bits. Figure 4.2

32	1	2	3	4	5	4	5	6	7	8	9	8	9	10	11
12	13	12	13	14	15	16	17	16	17	18	19	20	21	20	21
22	23	24	25	24	25	26	27	28	29	28	29	30	31	32	1

Figure. 4.2 The Expansion Function used in each of DES's 16 rounds. This converts 32 bits of input to 48 bits of output by duplicating some bits. For instance, bit 32 of the input ends up as both the first bit of the output and the 47th bit. The first bit of the input ends up as the second and the 48th.

(b) The key for round *i, K_i* is XORed with the result of $E(R_{i-1})$. This key contains 48 bits of the 56 bits of key material.

(c) The result is passed through the s-box, S. In practice, the actual description of DES splits the s-box into eight parts, $S_1 \ldots S_8$. Each of these s-boxes accepts 6-bits as input and returns 4-bits. All eight of them combined convert the 48 bits of K_i $E(R_{i-1})$ into 32-bits. Each block of 4 bits is being scrambled in with two of its immediate neighbors. Figure 4.3 shows the first of the eight s-boxes.

000000	->	1110	000001	->	0100	000010	->	1101	000011	->	0001
000100	->	0010	000101	->	1110	000110	->	1011	000111	->	1000
001000	->	0011	001001	->	1010	001010	->	0110	001011	->	1100
001100	->	0101	001101	->	1001	001110	->	0000	001111	->	0111
010000	->	0000	010001	->	1111	010010	->	0111	010011	->	0100
010100	->	1110	010101	->	0010	010110	->	1101	010111	->	0001
011000	->	1010	011001	->	0110	011010	->	1100	011011	->	1011
011100	->	1001	011101	->	0101	011110	->	0011	011111	->	1000
100000	->	0100	100001	->	0001	100010	->	1110	100011	->	1000
100100	->	1101	100101	->	0110	100110	->	0010	100111	->	1011
101000	->	1111	101001	->	1100	101010	->	1000	101011	->	0111
101100	->	0011	101101	->	1010	101110	->	0101	101111	->	0000
110000	->	1111	110001	->	1100	110010	->	1000	110011	->	0010
110100	->	0100	110101	->	1001	110110	->	0001	110111	->	0111
111000	->	0101	111001	->	1011	111010	->	0011	111011	->	1110
111100	->	1010	111101	->	0000	111110	->	0110	111111	->	1101

Figure 4.3 The Table shows how the first DES s-box converts 6-bit values into 4-bit ones. The Function is nonlinear and difficult to approximate with linear functions.

(d) Some extra scrambling is added with another permutation *P* as shown in Figure 4.4

16	7	20	21	29	12	28	17	1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9	19	13	30	6	22	11	4	25

Figure 4.4. The P function used in each of the 16 rounds of DES. Bit 1 is moved to bit 16, bit 2 is moved to bit 7, and so forth.

(e) Finally the result of $P(S(K_i \quad E(R_{i-1})))$ is XORed L_{i-1}.

4. After all 16 rounds are completed, an inverse of the initial permutation is created. (Figure 4.1)

And the steps for defining the key scheduling algorithm are as follows.

There are 56 bits of key-material used in DES, but only 48 are used in each round. The steps for producing $K_1 \ldots K_{16}$ are:

1. Initially the key may contain 64 bits. The eighth bit is ignored as it is often zero in ASCII text. The rest of the bits are scrambled with a key transformation as in Figure 4.5. giving the result k_0.

57	49	41	33	25	17	9	1	58	50	42	34	26	18
10	2	59	51	43	35	27	19	11	3	60	52	44	36
63	55	47	39	31	23	15	7	62	54	46	38	30	22
14	6	61	53	45	37	29	21	13	5	28	20	12	4

Figure 4.5 The key transformation used to convert 64 bits of password into 56 bits of key material by both scrambling the data and ignoring the eighth bit.

2. The 48 key bits for each round are selected using the function *KS* in Figure 4.6.

14	17	11	24	1	5	3	28	15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	2	41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56	34	53	46	42	50	36	29	32

Figure 4.6. The Key selection function. Here the data must be interpreted in a different way than the other functions. Here the 14th bit of input is selected to be the first bit of the output. Bit 17 comes second. Bits like 9, 18, 22, 25, 35, 38, 43, and 54 are left out.

3. Each K_i will be produced by computing $KS(k_i)$. Each k_i is produced by splitting the 56 bits of k_{i-1} into two 28 bit halves and rotating each half by s_i bits, where s_i is given in Figure 4.7

1 1 2 2 2 2 2 2 1 2 2 2 2 2 2 1

Figure 4.7. The 16 values of s_i used in the key scheduling algorithm.

The 16 rounds of DES is summarized in Figure 4.8.

$$L_k=R_{k-1} \qquad R_k=L_{k-1}+f(R_{k-1},K_k) \qquad k=1,2,\ldots 16$$

Figure 4.8. How 16 rounds of DES encrypts the two 32-bit halves of the message L_0 and R_0 are assembled using the initial permutation. At the end, they are disassembled using the inverse of this permutation. The function f is $P(S(K_i \quad E(R_{i-1})))$.

Decryption of DES is not different from encryption. The algorithm was designed to allow the same steps to be used in decryption. The only difference is that keys must be used in reverse order.

Many other private-key systems use a structure similar to DES. These include IDEA, Blowfish, Shipjack, Khufu.

DES is rapidly losing its dominance. the 56-bit key is widely considered to be too short and some attacks have succeeded in finding weaknesses. Many people are using triple DES which uses three rounds of DES and three different keys or other variants.

4.2. Public-Key Encryption

Public-key encryption systems have two keys. One locks the data and the other unlocks it. The important feature is that the key that locks the data can't be used to unlock it. Only the other key can do this. This allows the owner of a system to publish one key and keep the other secret. If someone wants to send a message that can't be read by anyone else he uses the public key of the recipient to encrypt the message so only the recipient can open it using his private key. And the owner of a pair of keys can generate digital signatures by using the private key for encryption, hence letting anyone to verify it by using the published private key.

The most popular form of public-key encryption is RSA system developed by Ron Rivest, Adi Shamir, and Len Adleman. There are other public-key systems also but they are not widely used. This is not a good situation as sudden developments in mathematics which leads to easy breaking of RSA might leave current systems without choice.

RSA

The RSA system relies upon mathematics of modulo arithmetic. Both encryption and decryption are completed by raising numbers to a power modulo, a number that is the product of two large primes. The two primes are kept secret. The system can be broken if the two primes are recovered by factoring which is a process that has been proved to be extremely difficult. [RSA97]

To encode a message using RSA, a user needs to create a public and a private key which are chosen as follows:

1. First two large primes (in the range of 200-1000 bits), p and q are chosen. The numbers are randomly selected and then primal tests are applied.

2. The primes are multiplied together to yield $n=pq$. n is often 512 or 1024 bits long in practice.

3. The secret key e is chosen. The greatest common divisor of e and (p-1)(q-1) must be 1.

4. The public key d is the modular inverse of e mod (p-1)(q-1)

5. The factors p and q are discarded. That leaves public key pair n and d, secret key pair n and e.

1. Let p=13, q=17
2. n = pq; n = 221
3. Let e = 5
4. d = 77 (ed $\text{mod}_{(p-1)(q-1)}$ = 385 mod 192 = 1 mod 192
5. Let m = 5; Encryption: $5^{77} \text{mod } 221$ =>31
6. Decryption: $31^5 \text{ mod } 221 = 5$.

Figure 4.9. An application of the RSA algorithm.

For encryption: A message is converted into a number less than n that is m. This can be realized with various methods of which the simplest is concatenating bytes. If the message is longer, encryption must be done in blocks. m is encrypted by computing $m^d \ mod \ n$. This message is decrypted by computing $(m^d \ mod \ n)^e \ mod \ n$.

The system relies on the fact that

$$\begin{aligned}(m^d \ mod \ n)^e \ mod \ n &= (m^d)^e \ mod \ n \\ &= m^{de} \ mod \ n \\ &= m \ mod \ n.\end{aligned}$$

There are a few possible interpretations of “breaking RSA.” The most damaging would be for an attacker to discover the private key corresponding to a given public key; this would enable the attacker both to read all messages encrypted with the public key and to forge signatures. The obvious way to do this attack is to factor the public modulus, n, into its two prime factors, p and q. From p, q, and e, the public key, the attacker can

easily get *d*, the private key. The hard part is factoring *n*; the security of RSA depends on factoring integers being difficult. In fact, the task of recovering the private key is equivalent to the task of factoring the modulus: you can use *d* to factor *n*, as well as use the factorization of *n* to find *d* .

Another way to break RSA is to find a technique to compute *e*-th roots mod *n*. Since $c=m^e \bmod n$, the *e*-th root of *c* mod *n* is the message *m*. This attack would allow someone to recover encrypted messages and forge signatures even without knowing the private key. This attack is not known to be equivalent to factoring. No general methods are currently known that attempt to break RSA in this way. However, in special cases where multiple related messages are encrypted with the same small exponent, it may be possible to recover the messages.

The attacks just mentioned are the only ways to break RSA in such a way as to be able to recover all messages encrypted under a given key. There are other methods, however, that aim to recover single messages; success would not enable the attacker to recover other messages encrypted with the same key. Some people also studied whether part of the message can be recovered from an encrypted message [ACG84].

The simplest single-message attack is the *guessed plaintext attack*. An attacker sees a ciphertext, guesses that the message might be "Attack at dawn," and encrypts this guess with the public key of the recipient; by comparison with the actual ciphertext, the attacker knows whether or not the guess was correct. This attack can be thwarted by appending some random bits to the message. Another single-message attack can occur if someone sends the same message *m* to three others, who each have public exponent *e* = 3. An attacker who knows this and sees the three messages will be able to recover the message *m*; this attack and ways to prevent it. [Has88]. This attack can also be defeated by padding the message before each encryption with some random bits. There are also some *chosen ciphertext attacks* (or *chosen message attacks* for signature forgery, in which the attacker creates some ciphertext and gets to see the corresponding plaintext, perhaps by tricking a legitimate user into decrypting a fake message. [Dav82, DO86, KR95]

There are also attacks that aim not at RSA itself but at a given insecure implementation of RSA. For example, if someone stores a private key insecurely, an attacker may discover it. One cannot emphasize strongly enough that to be truly secure RSA requires a secure implementation; mathematical security measures, such as choosing a long key size, are not enough. In practice, most successful attacks will likely be aimed at insecure implementations and at the key management stages of an RSA system.

Current estimates for breaking RSA encryption is about $1,000,000 in cost and eight months of effort for a 512-bit key. And the key sizes can be increased as the hardware gets faster. RSA is highly strong today, noone would be willing to spend millions to break a transaction which could only worth $1.00 or even less.

4.3. Blinded Digital Signatures

Noone would like signing a blank sheet of paper, but many digital cash schemes require banks to do this. This is due to the anonymity. The customer wants banks to attach their

signature without knowing the bits. These systems have been invented by David Chaum [Cha83, Cha85, Cha88a, Cha88b]

Imagine a bank uses the RSA system to sign its bank checks. Ordinarily, t would use some hash function, *H(m)*, to compute the hash of a bundle of bits representing the note *m*, and hen it would compute the $H(m)^{\wedge d} \bmod n$ using its secret key *d* and *n*. The problem is that the bank could keep a record of m and match it to your name. When a merchant returned with the digital note *m* the bank would know where you have spent your money.

You can get the bank sign something without knowing what it is by using a *blinding factor*. That is, you ask the bank to sign $H(m)k^e \bmod n$ where k is a random number ($1 \le k \le n$) and *e* is the published key of the bank. You send the bank the value:

$r = H(m)k^e \bmod n$

The bank signs it by computing:

$r^d = (H(m)k^e)^d \bmod n$

The bank could try to keep a record of this transaction, but you can eliminate this by removing the blinding factor.

$t = r^d k^{-1} \bmod n = m^d \bmod n$

This solution works as $(k^e)^d \bmod n = k^{de} \bmod n = k$

Naturally this is not the whole solution. The bank would like to know whether it is signing a $100 bill or $1,000,000 bill. The solution will be presented in other algorithms.

4.4. Hash Algorithms

The main purpose of a hash algorithm is to come up with a relatively short number that could be used as a surrogate for a large file. These are quite useful as you can then use a signature algorithm to sign the surrogate instead of the whole file, saving processor time.

The cryptographically secure hash functions make it infeasible for someone to tamper with the file in any way without the tampering changing the final hash value. There are also other hash functions the most common of is the *checksum*. The procedure is used many file transfer protocols to determine whether an error occurred. Someone sending a file would add up all of the bytes of data and append this sum to the end of the file. Since there are 8 bits in the checksum byte, there are 256 different possible values and a 1/256 chance that a random error will leave the checksum unchanged. A 32-bit checksum has $1/2^{32}$ chance of an error going undetected. [Pre93]

The checksums may be quite effective against random error, but they fall quickly to malicious attack. Consider a simple file that is just the value of a money transfer. $100095. If the checksum is just computed on the digits modulo 10, then the checksum is: 1+0+0+0+9+5 = 15 mod 10 = 5.

Which can be manipulated by simply rearranging the digits. A cryptographically secure hash function is very similar to encryption. In fact, a simple hash function, can be built out of a secure encryption function, f(k,B), that takes a key, k, and a block of data B. As a simple case let k,B and the output of f have the same number of bits. If the file to be hashed consists of the blocks of data, $B_1,B_2,\ldots,B_i$ then a hash value could be computed as:

$f(B_i, f(B_{i-1},\ldots f(B_2,f(B_1,R))\ldots))$.

The final answer serves as the hash value for the entire file. The value R is a random vector that is part of the hash value standard [Win84]

It is very hard to tamper with a hash value of this function. If you want to change the first block of data that hash value will remain unchanged if you can find another value such that $f(\hat{B}_1,R) = f(B_1,R)$. But f needs to encrypt and decrypt successfully, which means that there should be one and only one encrypted block of data for each unencrypted block. This means that there is no other $\hat{B}_1$.

One solution is to find a $\hat{B}_2$ that balances the changes made in $\hat{B}_1$ so that $f(B_2,f(B_1,R)) = f(\hat{B}_2,f(\hat{B}_1,R))$. That means you are looking for a particular key $\hat{B}_2$ that encrypts $f(\hat{B}_1,R)$ into $f(B_2,f(B_1,R))$. Finding the key that converts one block into another is breaking the encryption with a known plaintext attack. Most good encryption systems, including DES resist this kind of attack. Breaking the hash function would be equivalent to breaking the encryption.

MD-4, MD-5, and the SHA

Many hash functions are based on the same process as the chain of encryption functions, although they are optimized to make the process more efficient. The best known hash functions are MD-4 and MD-5 created by Ron Rivest [Kal92, Riv91, Riv92]. The Structure of MD-4 was borrowed and modified by the National Institute of Standards and Technology in U.S. to create the Secure Hash Algorithms that is used with the Digital Signature Standard. [Rob94, NIS92].

MD-5 processes data in 512-bit blocks and produces a 128-bit hash value. This is more efficient than using encryption functions like DES which have a block size and a key size that are close to each other. The last block of a file is often not 512 bits, so padding is added in the form of a single bit 1, a flexible amount of 0 bits, and then a 64-bit number representing the number of bits in the file. The extra 0-bits are added until the last block is 512 bits long. The 512-bit block is broken up into sixteen 32-bit blocks., $M_0,\ldots,M_{15}$.

There are four 32-bit variables A,B,C and D, that are permuted in four major rounds by each of the sixteen blocks. When this is completed for all 512 bit block in the file, then the four values A,B,C and D, are appended to create the 128-bit hash value.

The four values are permuted by four different mixing functions. Round one consists of using the first function, called *FF*, to mix A,B,C, and D with the sixteen different values of M_0 through M_{15}. Round two has the same structure but it uses a different function *GG* . Round three uses *HH* and four uses *II*.

These scrambling procedures can be summarized as:

```
FF(a,b,c,d,j,s,t) = a:= a+(F(b,c,d)+Mj+t) << s
GG(a,b,c,d,j,s,t) = a:= a+(G(b,c,d)+Mj+t) << s
HH(a,b,c,d,j,s,t) = a:= a+(H(b,c,d)+Mj+t) << s
II(a,b,c,d,j,s,t) = a:= a+(I(b,c,d)+Mj+t) << s
```

The "<<" stands for left shift.

The basic scrambling functions, F, G, H, and I, are:

F(X,Y,Z) = (X Y) (NOT(X) AND Z)
G(X,Y,Z) = (X Z) (Y (NOT(Z)))
H(X,Y,Z) = X ⊙ Y ⊙ Z
I(X,Y,Z) = Y ⊙ (X (NOT(Z))

Here, stands for bitwise AND, stands for a bitwise OR, and ⊙ stands for bitwise XOR. Three of the basic functions that serve as this foundation are nonlinear. When they are used repeatedly they scramble data.

The entire hashing process is:

1. The file is broken into 512-bit blocks and padded.
2. The four variables A,B, C and D are set to 67452301, efcdab89, 98adcfe, and 10325476, respectively.
3. Each 512-bit block is processed in turn with these four rounds:
(a) A copy of *A*, *B*, *C* and *D* is made. Call them $\overline{A}$, $\overline{B}$, $\overline{C}$, and $\overline{D}$.
(b) In round one, the function *FF* is used to operate on $\overline{A}$, $\overline{B}$, $\overline{C}$, and $\overline{D}$ sixteen times. In each of these instances, a different part of the 512-bit block, M_i, is used along with a different constant and shift value *s*.
(c) In round two, the function *GG* is used 16 times in the same manner.
(d) In round three, the function *HH* is used 16 times in the same manner.
(e) In round four, the function *II* is used 16 times in the same manner.
(f) Finally, the values of $\overline{A}$, $\overline{B}$, $\overline{C}$, and $\overline{D}$ are added back into *A*, *B*, *C* and *D* .
4. *A*, *B*, *C* and *D* are concatenated to produce the hash value.

The values of t were chosen using a sine function. The values of s are chosen to maximize diffusion.

SHA, the Secure Hash Algorithm

The SHA is based upon MD-4, which is a shorter version of MD-5. The function produces, though, a 160-bit hash value. Which is ideal as DSS uses a 160-bit modulus. [NIS93].

The major similarities and differences between the MD-5 and SHA are:

• The SHA also processes information in 512-bit blocks, The padding is accomplished in the same manner.

• There are five variables *A, B, C, D,* and *E* , not four, that are used to accumulate the final hash value.

• There are four rounds, but in each round the functions are applied 20 times instead of 16. There are still sixteen 32-bit values in the 512-bit block being processed, but some are reused. So each of the five values *A, B, C, D,* and *E* are modified four times.

• The scrambling functions used are:

```
F(X,Y,Z) = (X  Y)  (NOT(X) AND Z)
G(X,Y,Z) = X⊙ Y ⊙ Z
H(X,Y,Z) = (X  Z)  (X  Z)  (Y  Z)
I(X,Y,Z) = X⊙ Y ⊙ Z
```

In this case the same function is used in both the second and the fourth rounds. stands for bitwise AND, stands for bitwise OR, $\odot$ stands for bitwise XOR.

• In MD-5 the same sixteen 32-bit blocks are used. In the SHA, new versions are created for each of the 80 different rounds using an error correcting code-like scheme. If the sixteen blocks are $M_0,\ldots M_{15}$, then the 80 modified blocks are $\overline{M}_0,\ldots,\overline{M}_{79}$. For i between 0 and 15 $\overline{M}_i = M_i$. For i between 16 and 79, $\overline{M}_i = M_{i-3} \odot M_{i-8} \odot M_{i-14} \odot M_{i-16}$

• The functions *FF,GG,HH,II* are much more complicated. The first, *FF(A,B,C,D,E),* consists of these six steps:

```
1. t = (A<<5)+F(B,C,D)+E+M̄i+5A827999
2. E = D
3. D = C
4. C = B<<30
5. B = A
6. A = t
```

• The function *GG* is the same, but it uses *G* instead of *F* and 6ED9EBA1 for the additive constant. The function *HH* uses *H* and 8F1BBCDC and the function *II* uses *I* and CA62C1D1.

SHA is regarded as the best hash function available but involvement of NSA raises various questions such as availability of backdoors.

4.5. Bit commitment

The bit-commitment protocol was developed to prevent people from changing answers. For instance if you wanted to prove that you knew something before it was made public. You could encrypt your predictions give them to another party. When the time to verify your predictions has come you could give them the key. So that they could verify it. Which may seem logical. But what if there were two keys, k_1, k_2. that give different answers when used in decryption? You could easily wait for the real answer and give the key according to outcome..

Bit commitment protocols are designed to prevent this. The simplest form is the use of a long, prearranged value, V,. V is supplied to you and you encrypt your prediction M, is encrypted by concatenating V and M leading $f(VM,k)$

4.6. Secret Sharing

There are many occasions when splitting a secret among different parties is desirable. Only if *n* parts are rejoined can the secret be constructed. [Bla79, Sha79].

The simplest way to split up a secret is with the XOR function. If the secret will have *k* bits and it will be split between n parts, then create *n-1* random *k*-bit numbers, $s_1,s_2 \ldots s_{n-1}$. If the secret is *S* then, set $s_n=s_1 \odot s_2 \odot \ldots \odot s_{n-1} \odot S$. "$\odot$" is a bitwise XOR function. The numbers $s_1,s_2,\ldots,s_n$ are the n parts of the secret *S*.

This system for splitting the secret does not give a holder of one part any information about the entire secret. Even is some one gathers n-1 keys, the task of finding missing part is no easier than finding the secret. A simple but ineffective method could be is simply dividing the secret S into k/n bit parts. The problem with this approach is each person knows some of the secret's bits. If the secret is the key is a 56-bit DES key and was shared among seven people. Even if only two people collaborate there would be only 40-unknown bits which is very short for a brute-force attack and can easily be revealed.

4.7. Kerberos

One of the big problems in cryptology is key management. If you want to hold a conversation with someone that you've never met before, it is hard to set up a key that both of you can use to create a secure channel. Even a greater problem is knowing that you're really talking to the right person instead of someone who is simply intercepting the messages and pretending to be the right person. Public-key cryptology offers one solution to the problem, and many digital cash systems rely upon public-key pairs that are certified by a central authority. The SSL low-level encryption standard used for secure HTML connections is one example.

The same key management problem can be attacked with private keys. *Kerberos* is one of the most popular models to emerge in common use. It was first developed at MIT where it kept many workstations on campus secure. This model is also used occasionally in digital cash systems. [Ker97].

In a Kerberos secured network, one machine known as the Kerberos server is responsible for keeping a list of everyone's password and everyone's secret key. This server, which is kept in a secure location could use this secret information to establish connections between two machines.

Let's say that user A wants to establish a secure connection with user B with the help of the Kerberos server KS.

1. User A petitions KS for a ticket that would create a connection with B. Both KS and A know A's password but they don't want this to travel across the network. So A's computer keeps it in local memory. The request merely asks for a connection to B.

2. KS receives the request and creates a new random key, K_{AB}, that A and B will use to communicate. Than it takes this key and encrypts it into two different packets. The first uses the hash of A's password as the key. The second packet uses the hash of B's password as the key. KS sends both of these to A.

3. A receives the two packets. A can decrypt the packet encrypted with the hash of its password, but it can't decrypt the other packet. A's password never traveled over the network so no network eavesdropper could have gotten it. Now A knows K_{AB}.

4. A sends the second packet to B over the network. Only B could decrypt this packet because only B and KS know B's secret password. When B decrypts it B knows K_{AB}. The connection is established.

That is the basic mechanism by which a Kerberos server can establish secret links between any of its clients. One of the important features of the system is that it also authenticates two clients to each other. A knows that it could only be talking to B and B knows that it is A on the other end of the line. Or more correctly, the connection joins a person who knows A's password with a person who knows B's.

There are many details that are built into a working system. The tickets, for instance, come with expiration times. All of the clocks in the network are synchronized and a ticket may only be good for a few minutes. This prevents the reuse of a key again.

Many Kerberos implementations also maintain multiple ticket servers. A client that wants access to a particular data server must first ask the Kerberos server for a secure link to a ticket server. Then, it asks this ticket server for a secure link to a data server using the same basic protocol each time.

4.8. Zero-Knowledge Proofs

The classic failure of many security systems arise when the attacker learns the password, by eavesdropping, by pure guesswork etc. You have to use your password to access your priviliges, bank account etc. During the communication process you reveal your password and if this conversation is trapped by others they can learn your password.

Zero-knowledge systems form the framework for communication between two parties such as even the whole conversation is trapped the listener can not learn anything yet you are able to prove that you are authentic, you know the password. The process is widely known as *challenge and response*. One side comes with a question and the other side answers it.

A Discrete Log Zero-Knowledge System [CEG88]
In this case, you choose a large prime number p, as well as three large numbers A,B, and x such that $A^x = B \bmod p$. In this case $A^x \bmod p$ can be used as a digital signature if x is kept secret. Finding of x is considered computationally infeasible given A,B,p.

Assume that you want to prove that you have created a particular signature. In this case challenge and response protocols can be dangerous. If the challenge is "Prove that you are authentic by signing this random string V" you can not just sign and give away $V^x \bmod p$. You would have proved yourself but V could be blank check. You can not apply your signature on anything just like you would not in real world.

An approach could be:
1. You know x. A,B and p are public. You want to prove that you know x such that $A^x = B \bmod p$ without revealing x.
2. You choose a random number, $r<p$ and send $h=A^r$.
3. The other end send you a random bit, b.
4. You send back $s = r+bx \bmod p-1$.
5. The other end computes $A^s \bmod p$ which should equal $hB_b \bmod p$. If it is not you do not know x.

It obvious that there is a 50% of chance that you don't know x but guessing *b* correctly. As there is a random bit involved. But if the procedure is repeated n times producing correct values for h and s. The probability is $1/2^n$.

5. Cash protocols

5.1. Digital Checks

Much of the money used in business transactions flows through paper checks. They are simple to use and popular as they provide a proof of transaction. A check not only shows the intention of a party to pay a certain amount but also the acceptance of payment by the other party.

Electronic transactions can imitate the paper check systems. There are various electronic payment systems such as EFT, but they lack the flexibility of checks. Banks offer payment systems that allow you to transfer money from account to account, or to pay your periodic payments such as credit cards, utility bills. But to use these for your daily transactions you have to know the recipients account number.

A solution could be offering digital checks secured by digital signatures. Such a check would consist of a block of data like this:

$Signed_{owner}$(Bank Name, Owner's Name, Amount, Destination Name)

The phrase $Signed_{owner}$ means that the entire block would be signed by the owner's digital signature. The block of data containing the bank's name could also contain an electronic address where the draft could be presented for payment. The owner's name would contain the account number. when the recipient gets the check, he would take it to the bank, which would verify the signature and transfer the amount into his account. the system mimics the normal checks.

The system could offer proof that the check was cashed by returning the entire block signed with recipient's digital signature. The bank receiving the check could add its signature and the bank on which the check was drawn would add its signature before returning the check to the owner. The entire chain would look like this:

$Signed_{Owner'sBank}(Signed_{Recip'sBank}(Signed_{Recip}(Signed_{Owner}(details))))$

A digital check can be strengthened by locking the check so that only the intended recipient can spend it. This can be done by encrypting the message which can be done both by private-key systems and public-key systems. A public-key system allows more flexibility by letting exchange of messages between parties that have not met before. You could use the publicly available public key of a recipient to encrypt part or all of the message, which would also reduce information leakage. The recipient then would use his private key to open this.

5.2. Digital Cashier's Checks

Cashier's checks are a common way to guarantee larger sums of money. The bank produces a special check and places a hold on the money promised by the check. People trust the cashier's checks more than an ordinary bank draft as the bank has guaranteed that the funds are available.

Digital cashier's checks can be even more secure. Digital signatures can only be compromised if someone learns the signer's secret key. It is entirely possible that someone could produce a fairly official looking piece of paper with a bank's name on it without knowing what the bank's real checks looked like.

A simple digital cashier's check might look like a regular check with the bank's signature instead of the owner's:

$Signed_{Bank}$(Bank Name, Owner's Name, Amount, Destination Name)

Which can probably be generated by the owner sending a regular check to the bank and asking the bank to guarantee it. The bank could strip off the owner's signature or leave it. This check then would travel the same way as a regular digital cash.

Figure 5.1 shows an interpretation of a digital cashier's check. First Virtual Bank's signature is like a seal guaranteeing the contents of the details inside the box.

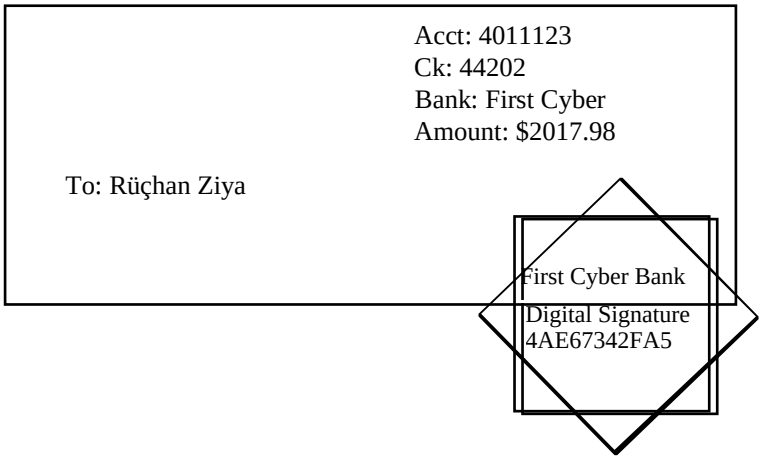

Figure 5.1. A basic digital cashier's check. The bank creates a digital signature for the note by hashing the recipient, the account number, the check number and the amount, and then encrypting this value with its private key.

An enhanced model could hide extra information from others. which is illustrated in figure 5.2.

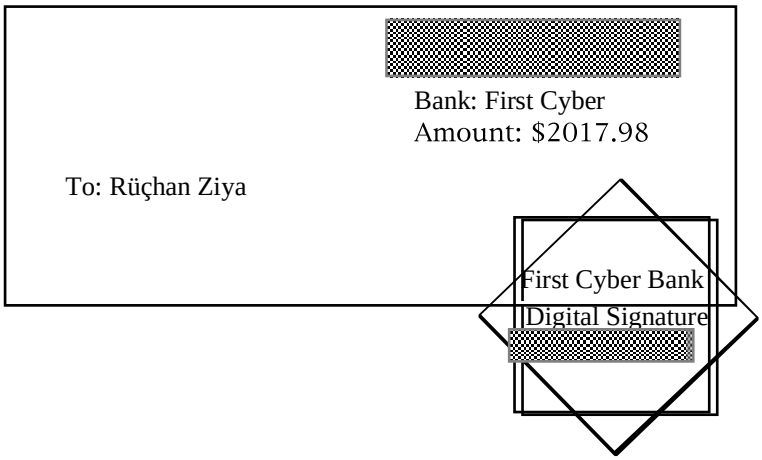

Figure 5.2. An enhanced digital cashier's check. The account and the check number are encrypted with a secret key by the bank. It's signature is encrypted with the public key of the recipient so that only the recipient can cash it.

The structure of this system is definitely not anonymous. The bank must keep a record of all checks that are issued and maintain a block on the accounts. When checks arrive, the transactions are recorded and every bank along the path can know the identity of both halves of the recipient. This information can be quite valuable as the banks and credit card companies already know.

Digital Cashier's checks could be made more anonymous. The bank could transfer the money to another bank account when a customer requests a check and write the draft against this account. This would hide the identity of the sender from the recipient. But the bank would still know both parties as it would create the check.

It is not possible to make the digital cashier's check system more anonymous. Even if the sender has asked the bank to prepare a check to "cash" giving right to anyone to cash the check. The recipient's bank would have the identity of the party who has cashed the check.

If a bank chose to issue cashier's checks made out to "cash" from a central account, then it would be minting a type of digital currency. The bits that made up this check could be passed around several times among different parties before it was presented for a deposit at a particular bank. This system is unlikely to succeed as anyone along the chain could keep a copy of the check and then race to their bank to cash it.

5.3. Simple Anonymous Cash:

drafting a check against an anonymous account owned by the bank is not enough to ensure anonymity because the bank can see who presents the check for payment. A true anonymous system can be built which can prevent the bank from discovering any identity information about a bill. The anonymity is preserved through the blinded digital signatures. and a cut-and choose protocol similar to zero knowledge proofs. Combining these two procedures allows a customer to create a bank certified check without letting the bank to know what it is signing.

1. A customer who wants a unit of anonymous cash creates k sample units and presents them to the bank. This unit would contain the name of the bank, the value of the unit, and the currency.

2. Each unit is given a random serial number.

3. The data is put in standard format m_1= (bank, amount, serial number, currency) … m_k= (bank, amount, serial number, currency).

4. The customer blinds the k units with random blinding factors $\{b_1 \ldots b_k\}$. And returns $m_1 b_1^e \ldots m_k b_k^e$ using the banks public key e and the associated modulus..

5. The blinding factors $\{b_1 \ldots b_k\}$ prevent the bank to check for contents.

6. The bank requests the blinding factors for k-1 units but the unit number i.

7. The customer gives the blinding factors except the b_i. Figure 5.3

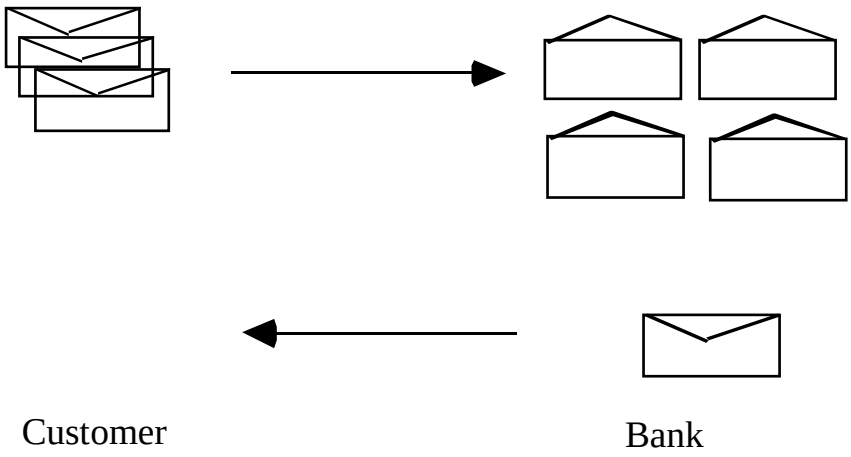

Figure 5.3. A Customer who wants to create anonymous cash must create *i* sealed bills, shown here as envelopes. The bank opens up *i-1* "envelopes" to check for fraud and returns the last one.

8. The bank unblinds k-1 units to check for their validity.

9. If all checked units are correct. The bank signs the unit i with its private key. $(m_i b_i^e)^d = m_i^d b_i$.

10. The customer unblinds the note by multiplying with b_1^{-1}.

The chances of fraud is depended on value of k. But even if k is 2 then the chance of being caught is 1 to 1. And if the punishment is serious enough the fraud should be inexistant.

There are some shortcomings of the system. How would a bank know the who is the cheater if a bill is spent more than once.

5.4. Traceable Anonymous Cash

The simple anonymous cash may be useful, but it still leaves room for fraud. If double spending occurs, the bank can not determine which of the two was the guilty party. A more robust system would allow the bank to catch the guilty one. This system can be created by using the secret-sharing techniques. If a unit is spent twice, then the information in both instances can be used to reveal the cheater.

Two cryptographic algorithms play a part in catching the cheater. The first, secret-sharing is only used in its simplest form. The identity is split into two halves, *Id* and *Id'* so that $Id \odot Id'$ reveals the identity. The second is the bit commitment scheme that prevents someone to present a false key to unlock an identity half. Each of the halves of the identity of the person requesting the unit of cash will be sealed with a bit-commitment protocol so that the half can not be denied later.

The steps are:

1. The customer requesting a unit of cash creates k sample units as before containing the bank's name, the amount of the draft, the underlying currency, and a unique serial number for the bill.

2. The customer also takes j copies of his identity and splits them in half using the simple XOR secret sharing system: $\{Id_1 \ldots Id_j\}$ and $\{Id_1' \ldots Id_j'\}$. If anyone discovers both Id_i and Id_i' he would know the identity of the customer creating the bill.

3. Each of the 2j identity halves in each of the k units is sealed up with its own bit committed encryption key, $Key_{(a,b)}$. For Instance, in unit m, the two halves of the identity Id_i and Id_i', are encrypted as $f(Id_i G, Key_{(2i,m)})$ and $f(Id_i' G, Key_{(2i+1,m)})$. The string G appended to the identity is a unique string that might be the name of the bank in ASCII. It is added to prevent someone from providing the wrong key later.

4. The *k* units are blinded using blinding factors $\{b_1 \ldots b_k\}$ and presented to the bank.

5. The bank asks to inspect the contents of k-1 units. That is customer supplies the keys and blinding factors for all units but one. The bank unblinds these bill and decrypts the identity halves. If each pair properly reveals the customer and other information such as serial numbers is fine. The bank is satisfied.

6. The bank signs the secret bill and sends it back to the customer.

7. The customer unblinds the unit and uses.

These bills are significantly larger than simple anonymous cash because they contain extra j copies of the identity bound within them.

For a bank to catch double spenders, the people trading the bills must engage in a significantly more complicated process. Simple anonymous cash could be traded by copying it. This traceable variety must be traded with the protocol below, which is designed to reveal the identity to catch the cheater:

1. When a customer arrives at a store to spend coins, the shopkeeper flips a coin j times and gives the results to the customer.

2. For each time the coin comes heads in bit i, the customer reveals Id_i by producing $Key_{(2i,b)}$. For the tails, the customer reveals Id_i' by producing $Key_{(2i+1,b)}$. The customer, in the end, has revealed j different halves of the transaction but the identity can not be revealed as none of them is from a matching pair.

3. The shopkeeper can check that the customer has presented the correct keys because the customer sealed them using a bit-commitment protocol. The special string G should appear at the end of each message.

4. The shopkeeper can check the validity of the bill by looking at the digital signature and checking to see that it is a valid bank signature.

5. The shopkeeper forwards the note to the bank including the bits holding the flips and the individual keys.

This spending protocol forces the customer to reveal half of his identity at j different times. If two notes arrive with the same serial number, then this information will reveal the cheater. If the customer is the cheater who spends the bill twice, then there will be two different sets of identity halves. There is a good chance that one half in one note will match the other half in the other note and the identity will be revealed. The chance that the customer would be asked to reveal the same halves in both of the transactions is $1/2^n$.

If the identities don't reveal the customer than the cheater is the shopkeeper. The shopkeeper doesn't know the values of different keys so can't decrypt a different set of pairs.

The digital cash produced by this system can't be spent multiple times in a chain. It must be returned to the bank after each transaction because of the identity revealing protocol. This effectively reduces the amount of anonymity. The bank knows who whitdraws money and who deposits it, but they don't know how they trade it in between. The trades, however, are only one step away from the bank.

5.5. HTTP and Cash

The World Wide Web (WWW) is currently the most popular way for people to publish information over the Internet. It is quite natural that people want to extend the system to send money across the Internet.

In the simplest sense, anyone can use HTML forms to exchange credit card numbers. Some web pages already use this method but it is insecure. It is clear that it is not feasible in the long run. Someone could build a program that watch the traffic for credit card numbers.

Encryption is the obvious solution. And there are two basic approaches S-HTML and SSL. These technologies not only offer a way of transferring credit card numbers securely but they allow the secure exchange of other forms of data.

S-HTTP [RS95]

S-HTTP (Secure Hypertext Transfer Protocol) is a simple extension to HTTP This standard allows the traffic to and from the server to be either encrypted, signed or authenticated. At the beginning of a session client and server choose the right combination. Any combination of nine different options is possible. The standard is not tied to a particular algorithm, negotiation of a mutually compatible set of algorithms is possible.

SSL [Hic95]

The Secure Sockets Layer (SSL) is a basic encryption system developed by Netscape. The software is designed to exist transparently above TCP/IP. Any TCP/IP application can initiate a TCP/IP connection using SSL. Unlike S-HTTP which is integrated into HTML and requires the negotiations to go through headers SSL is a low level standard on may coexist with S-HTTP.

SSL establishes a secure socket level connection. First the client and the server agree on a cipher and a key. And optionally authenticate the client. In details:

CLIENT-HELLO This is the first message that a client sends asking to start a connection with the server. It contains three different types of information: the types of ciphers the client knows to handle, any session ID that might be left over from a broken connection and some random data as challenge to the server.

SERVER-HELLO The server responds with two different types of messages. If the server recognizes the old session ID, it confirms this so the connection can begin again and a new set of keys is selected.

If this is a new connection or the ID is not recognized, the server sends back a certificate with its public key that has been authenticated by a certificate authority, the list of ciphers supported, and a random connection number. The challenge is not returned yet.

CLIENT-MASTER-KEY The client finishes the key selection. The choice for cipher is sent back. The key is transferred in three blocks. One block contains the extra arguments if the cipher requires. The other two blocks contain the key. The first portion contains the clear portion of the key. If a key is n bits long, the first n-k bits are shipped clear. The other block contains the k bits encrypted with the servers public key. This is due to export regulations.

CLIENT-FINISH Client indicates that it is finished with the authentication by sending the session ID encrypted with the current key.

SERVER-VERIFY When the server receives the master key, the secret portion is decrypted . And it learns the master key. The challenge data is encrypted using the key and send it back. The client can verify the existence of a secure link by decrypting the challenge.

REQUEST-CERTIFICATE This is an optional step for requesting authentication of the client. Includes challenge data.

CLIENT-CERTIFICATE If the client is asked for a proof of identity. Client sends a certificate and the encrypted challenge data.

SERVER-FINISH When the server is satisfied it sends the session ID encrypted with the master key.

When the master key is created, the protocol requires generation of two new keys which depend on the cipher used.

JEPI and UPP

In the real world when you walk in a store you can see what kinds of payment options they accept, which credit cards are accepted. In an electronic commerce as there will be different payment option this negotiation must be realized.

The UPP (Universal Payment Protocol) was developed by CommerceNet and the World Wide Web Consortium as part of the Joint Electronic Payments Initiative (JEPI). The standard allows a client and server to find a payment method acceptable to both. UPP is a subset of the Protocol Extension Protocol (PEP) which allows to see it the other end speaks a particular protocol.

6. Cash Systems

At this time of writing, there are many different companies trying to position their system as the dominant way of transferring money across the Internet. The proposed

systems a wide variety to questions like "How much anonymity is good?", "How much flexibility is ideal?" and "How much security is necessary?".

Digital money systems can be examined according to different criteria.

On- or Off-line

Digital money systems that need the help of a distant computer throughout the transaction are said to be "on-line" cash systems. The transaction is monitored by this third party and it blesses it if everything is correct. Addition of a third party takes communication resources, time, and cost so many people are actively exploring "off-line" systems. These are the ones that would allow people to meet on the street and transfer cash to each other even in the midst of a computer blackout. Clearly "off-line" cash is better because it is more flexible and also because communications cost money. If it costs, for instance, $.02 to clear an online cash transaction people would not want to use it in $.01 transactions and would also hesitate to use it in small transactions.

One problem is there is no true off-line cash that can be traded ad infinitum without a third party acting as a referee. Paper money or gold coins can be traded for years without being deposited in a bank, but this is because they can't be counterfeited. Digital notes are easy to copy. The cryptographic algorithms can catch the counterfeiters, but only when money is processed through a bank.

This means that even off-line cash is still in a sense on-line. It just means that interaction with the third party, the bank can take place at a more leisurely pace. The people might realize transactions on the street and walk away. Any fraud would be detected later when the digital cash is deposited in the bank.

Encryption and Security

Security is one of the major components of a digital monetary system. And encryption is a way of establishing security. The established systems use a wide range of encryption from none to highly complex. Encryption also has another dimension which has to be paid attention which is the governments tend to have regulation an encryption systems and especially for export. Even though the situation is improving and U.S. Government has authorized the export of 128-bit encryption in May 1997. Governments would never like criminals exchanging information that can not be intercepted or broken.

Certificates and Repudiation

Certificates are one way that people can add security to public-key systems by arranging for a central, trusted authority to verify and vouch for the public key of someone. This means that a store can start a transaction with a random customer and be certain of his identity when the customer presents a valid certificate.

The main advantage is that certificates prevent someone from denying a transaction later. Naturally, the certificates can be compromised if someone gets a copy of the private key matching the public key endorsed by the certificate. then such a person

could masquerade as you. In many cases the private key will be bound in a smart card and protected by PIN.

It is difficult to assess the need for repudiation. The banks need to solve this problem already with ATM machines, stolen checks, and credit cards. Although public-key certificates can be mathematically very convincing, they are not much secure than ATM cards. People will need to store the corresponding private key in a file or a smart card and unlock it with a PIN. These smart cards would be moderately more secure than an ATM card because it would not be possible to simply create one from an account statement and knowledge of the PIN number. It is possible to forge an ATM card if you know the account number, which can be gained from the statement, and the PIN number, which can be picked up by looking over someone's shoulder. finding the private key that matches the public one bound into a certificate is a hard problem that is currently intractable.

Anonymity

Anonymity is one of the most debated features of digital cash. There are two main views on the issue one sees it as dangerous because it will protect criminals. The other view sees it as an essential component that we should use to preserve freedom. The systems supply a wide variety of anonymity. Many systems do not offer any anonymity where as some protect the identity of the customer from the merchant but reveal it to the band and some keep everyone in the dark.

Digitized 'e-cash' Systems

A number of companies are developing payment systems which permit direct payments to be made anonymously. Payment takes the form of encoded messages representing the encrypted equivalent of digitized money. The aim is to be able to effect payment directly without requiring the use of intermediaries. A number of trials are presently under way to test the concepts which are involved.

Besides the system examined here there are many other systems that have been proposed namely: NetCash, NetCheque, CheckFree, OpenMarket, CAFE, Millicent, MicroMint, PayWord, MagicMoney. These systems generally resemble the others presented here but they may offer less anonymity, more security. Or the difference lies in the arena of trust. That is a system should trust the customer or merchant. Shall I charge first and send the goods or shall I sent the goods first and if customer is satisfied I can charge. The systems have trade-offs between anonymity, security, flexibility.

6.1. iKP [IKPP97]

IBM recently published a proposal for securely transferring money over a network. The system is designed to work with existing bank systems. The most important feature of the system is it provides a complete cryptographic protection and audit trail. Embedded security systems like SSL provide secure channels but how the channel is used is up to the programmer.

There are three different levels of the protocol, that has different levels of sophistication. The lowest level, 1KP only requires that a central authority publish a certificate guaranteeing its public key. There is no need for general public-key certificates to be issued to everyone. Which means that it can be adopted quickly due to the lack of need for a large-scale public-key infrastructure. The second, 2KP, requires each merchant to publish a public-key that is certified. the third level 3KP requires customers to have certified keys.

Even though iKP protocol is unlikely to be used the SET standard is highly close to iKP.

1KP

There are three entities in the *i*KP model: C, the customer, M, the merchant, and A, the acquirer or the bank.

1. When the customer chooses an item, the merchant makes an offer containing the cost, the currency for the transaction, the date and the merchant's ID number. The merchant also provides the public key and the certificate for the bank that will process the transaction.

2. The customer checks the certificate and adds his credit card number, the expiry date and PIN to the bundle. And encrypts it with the banks public key which is sent to the merchant who can't read the contents.

3. The merchant adds the secure hash of transaction details he knows and sends them to the bank.

4. The bank compares secure hash values of transaction details. If they are equal the credit card is charge. And a signed transaction approval or denial is sent to the merchant.

5. The merchant can check whether the transaction is approved or not and hand the goods.

1KP provides several forms of security for parties involved. The customer is totally anonymous to the merchant. The merchant has the authorization of transaction. The bank can be sure of mutual approval of transaction.

2KP

The second level offers greater accountability. Each merchant needs to have a certified public-key pair. The difference is the merchants adds his signature to the packet containing the customer's encrypted credit card info and transaction details. Now the merchant can not deny the transaction. The customer can also check the merchant. And if the merchant supplies a signed receipt the customer can keep this as a proof.

3KP

Now the customer who also has a public-key pair signs the encrypted details and the transaction details. And these can be used as proofs that the customer has actually spent money.

6.2. DigiCash [EC97]

One of the leading firms which is developing an 'e-cash' system is DigiCash, which has previously been involved in developing smart card technologies. DigiCash is a Dutch company based in Amsterdam created by David Chaum to build the software to use his cryptographic inventions. It has been running trials since November, 1994, involving the transmission of what is effectively electronic money using more than thirty sellers linked to the Internet. Test sites include the Encyclopedia Britannica. Payments consist of uniquely coded digital tokens which are established in such a way as to prevent duplication or fraud. Under the DigiCash scheme, customers would use local currency to buy an equivalent amount of digital cash from a bank. Instructions would then be sent from the bank's to the DigiCash user's personal computer, enabling payment instructions to be sent directly to sellers of goods and services on the Internet.

The structure of the DigiCash system includes both account based money and token based money. Each person maintains a central bank account with DigiCash that is filled with their larger denominations. Each person also gets a DigiCash wallet that can be filled with token based digital coins subtracted from an account. These coins are bundles of bits created with the basic digital cash algorithm. The bank automatically dispenses the coins in sizes that grow exponentially. This approach increases the number of times that someone must break a bill. If a merchant must break a coin, it must interrupt the transaction to go to the bank for the right change.

The current system only provides anonymity for the buyer. When the digital coins are whit drawn from the bank, they're created with blinded signatures and this prevents the bank from matching a serial number with a user. The merchants however do not have any anonymity because they must return the coins immediately upon receipt. The incoming amount is credited to their DigiCash account. It is argued that this form of cash is worthless to the lawless as records are kept. A drug dealer would have to deposit them in his account if he accepts DigiCash.

In theory, people could subvert the system by exchanging disks with cash stored on them. At the most extreme people would trade laptops, full of cash. So people could exchange coins without the involvement of DigiCash computers. The system has no cryptographic protection against double payment. DigiCash keeps a record of serial numbers.

The key to the DigiCash system is anonymity. A person who spends one of its electronic tokens does not need to reveal his or her identity to the buyer nor to any third party except when there is an attempt at fraud, i.e., when the same piece of digital money is presented twice for payment. At this point it is possible to unravel the digital token to reveal the entity to which it was originally issued. DigiCash is attempting to license its e-cash system to banks and other financial institutions. One of its key attractions is that it avoid the time and expense associated with becoming an approved credit card accepting merchant. Anyone will be able to set up a business and receive e-

cash once the system is operating fully. However, DigiCash will require the direct involvement of a bank for its system of digital cash issuance and this may yet prove to be a significant obstacle to the realization of the scheme. A bank is integral to the scheme, since it is required to hold collateral and to provide ultimate settlement of e-cash to more directly convertible currencies.

Ecash defines three different parts in clearing, realizing transactions The client, the merchant and the issuer. The issuer is normally a bank who has issued the Ecash. In order to be able to accept a specific type of Ecash merchants make agreements with issuers so that they can clear the transactions on line. The clients also deal with the issuer so that they own some Ecash to be used in transactions. As the issuers are normally banks they can offer ecash in exchange for real money. The clients can transfer their money from their account to their ecash accounts and then transfer the money to their computers using their ecash client software. This money is analogous to the money in our pockets or wallets. When a client wants to shop in a ecash accepting merchant the merchant checks for validity of the ecash that the client is offering by contacting the issuer. The issuer can only identify his signature on the ecash he can't identify the client which supplies the anonymity of the client just like the real money.

DigiCash is based on pre-defined valued electronic notes like the real world money.

The system has been in operation for a while with the Mark Twain Bank as the first issuer, with Deutche Bank in line. Mark Twain Bank has been issuing DigiCash for about a year now. And there are about 100 merchants that accept digi-cash.

6.3. Payment Clearing Systems

A number of companies are attempting to overcome the security issues involved in handling payments on the Internet by establishing electronic clearing systems. Essentially the service on offer involves a system of secure messages which permit buyer and seller to communicate with each other, while also permitting instructions to make payment to be sent via the message/payment clearer, frequently using existing proprietary networks.

One other approach is the introduction of a third party: a company that collects and approves payments from one client to another. After a certain period of time, one credit card transaction for the total accumulated amount is completed. There are, however, other factors to consider when using third party (or credit card) payments. For one, there is always a possibility that a payment is refused because the spending limit has been reached. For another, all payment details of a person are gathered in one centralized system: where they buy, when they buy and sometimes what they buy is stored. The collection of this data tells much about the person involved and this can conflict with the individual's right to privacy.

6.4. First Virtual [FV96]

First Virtual Holdings is one of the first companies to offer a digital money transfer system created for the Internet. First Virtual has developed a system for linking credit

card, banks and processing agents with the Internet. It has developed a closed loop payment system which involves First Virtual's providing a mailbox from which instructions to make the payment and to credit the seller's account are made. The system depends on the "off-line" network provided by EDS which is used to transfer credit card/bank account information, with First Virtual effectively acting as a message clearing house. In effect the buyer sends a message to First Virtual which passes this on to EDS. EDS in turn acts under instruction from First Virtual to pass on the account details to the seller. When the transaction is confirmed, First Virtual sends a message to the buyer to confirm that the transaction should still go ahead, at which point payment is effected.

The First Virtual system has been in operation since October, 1994, and is seen to have the advantage that it does not require encrypted messages as do other credit card-based systems. First Virtual checks with the buyer that a particular transaction is to go ahead before arranging the appropriate account debit. First USA Merchant Services have been contracted to provide clearing, settlement, and authorization for the credit card transactions. First Virtual is significant in that its payment system was created almost entirely independently of the banking system. The President and Chief Executive of First Virtual is quoted as stating "There was no traditional banking mechanism set up to deal with the Internet". First Virtual's initial aim is to permit businesses to receive income for services (such as publications) over the Internet where traditional payment methods would be uneconomic.

A First Virtual transaction includes several steps:

1. The buyer opens up a First Virtual account. This can be done by sending electronic mail to `apply@card.com`, telnetting to `card.com` or connecting to the WWW page `http://www.fv.com/html/setup.html`. The buyer needs to be ready to provide a Visa or MasterCard charge number because all bills will be sent to the buyer as a charge against this card. The substeps are:
(a) The potential buyer fills out a form including his name, address, e-mail address, and requested passcode. The e-mail address is used for confirmation messages. The passcode is used as he account name and password.
(b) If the application is processed correctly, First Virtual's automatic software will send a confirmation note to the e-mail address contained in the application. This note will include a temporary account number and a telephone number.
(c) The potential buyer calls First Virtual's computer and inputs the temporary account number and the credit card number.
(d) First Virtual charges $2.00 as a new account fee.

2. A potential seller must first get a buyer's account. This account can be converted into two different types of seller's accounts. The first, known as a *pioneer seller*, can be opened when the potential seller mails a paper check to First Virtual for $10.00. The check is cashed as a new seller's account fee and the bank account number is recorded. When First Virtual wants to deliver money it will send it electronically to this checking account. The pioneer seller account is intended for small, relatively unorganized businesses. There is no credit check, which forces First Virtual to withhold payment for at least 91 days until the purchaser actually pays the bill. Which is a defense against credit card fraud by dishonest merchants. Larger or more organized businesses can

receive money within four days if they open an express seller account which costs $350 for a credit check.

3. When the seller attracts a buyer, they negotiate the price and then the buyer gives a copy of his account ID to the seller.

4. The seller sends a transfer request to First Virtual through a variety of ways. The simplest is an e-mail message that looks like:

```
To:transfer@card.com
From: seller's e-mail account
Subject:
BUYER: buyer's account code
SELLER: seller's account code
AMOUNT: numerical amount
CURRENCY: currency ID
DESCRIPTION: A description of the description for future
identification.
```

The First Virtual receives this transaction request and begin processing.

5. First Virtual sends a request for a confirmation to the purchaser's e-mail account. The purchaser can answer with three different responses:

YES: All is well. The buyer authorizes First Virtual to bill the credit card on file for the amount.
NO: The buyer is refusing to pay. This is a significant event and First Virtual keeps record of this. If a buyer does this too often, First Virtual might terminate the account. First Virtual will make this because it does not want people to take advantage of sellers by refusing payment.
FRAUD: The buyer never authorized the transaction and First Virtual should investigate.

6. If the seller requests it, First Virtual sends along a transfer notification or payment authorization which is digitally signed.

7. When it is clear that the buyer has actually paid the credit card company, First Virtual deposits the correct amount in the checking account of the seller after subtracting the fees. The cost is $.29 plus %2 of the total amount.

One feature missing from this system is content encryption. The signing of transfer notification is recent development and the company may add additional layers in the future. Even though the credit card numbers never travel across the Internet the account numbers do. The security is based on the assumption that the e-mail account is secure and customers can deny transaction they have not realized.

This system has advantages as encryption is not used the system does not have to obey encryption export regulations. And the customer does not need to have complicated software. All a customer needs is transferring the account number which can be done by phone, e-mail or fax.

First Virtual relies on its partners such as banks and credit card companies if fraud occurs these institutions will also be involved to stop, investigate and make sure that it does not repeat.

Shortcomings

The simplest attack on the First Virtual is as follows:

1. Attacker gains access to the electronic mail of someone's account.
2. Attacker opens a bank account with a fake name.
3. Attacker creates a seller's account and registers it with the bank account.
4. Attacker learns the buyer's account code by reading mail.
5. Attacker starts a transaction with the buyer and his seller account.
6. Attacker confirms the First Virtual confirmation message using buyer's account.
7. When the money is deposited, the attacker withdraws it and disappears.

6.5. CyberCash [CC96]

This is the electronic clearance technology developed by CyberCash Inc.

CyberCash Inc. is a small start-up company set up in August, 1994, in Reston, Virginia. Its founder is William Melton, who was responsible for the creation of Verifone Inc., currently the leading supplier in the United States of POS (point of sale) credit card authorization systems. CyberCash is developing an on-line payment service and on December 12, 1994 announced that it had signed an agreement with Wells Fargo to run a pilot service. Initially the system will provide a secure means of providing credit card details and thereby effecting payment electronically. One of the features of the systems being developed by CyberCash is that it will be 'browser independent.' This stems from the breakthrough which Verifone made in supplying authorization terminals which can handle all the principal credit card services. William Melton is quoted as saying "The Internet is going to happen, with or without the bankers. But the bankers, the bright ones, are going to make this an opportunity".

The basic credit card transactions are very similar in nature to the iKP system. Merchants sign up with CyberCash to clear their credit card transactions and list the bank that clears the transaction.

The customers set up accounts by obtaining copies of software from CyberCash and filling out a form. The customer includes the credit card number that they intend to use. CyberCash creates an individualizes file that includes the card numbers and digital signatures assigned by CyberCash.

1. Once a price has been negotiated with the merchant, the customer is sent an on-line invoice detailing the purchase information together with a statement confirming the total charges

2. The customer's software presents the user with the preprocessed payment options. Which include or may include in the future credit cards, debit cards, bank accounts. The customer adds credit card number or debit card information, including personal

identification number (PIN) where appropriate. This information is then sent to the merchant in encrypted form together with the original invoice.

3. The merchant adds identification information and forwards all the information to the CyberCash server.

4. CyberCash checks whether they agree on transaction details. CyberCash then initiates a standard credit card or debit authorization request to the merchant's bank or designated merchant acquirer (processing center).

5. After the authorization request has been processed, CyberCash forwards a response to the merchant including a second packet encrypted with customers public-key. who then completes the transaction. Involvement on the part of CyberCash is completely automated and runs off its Internet file server.

6. The merchant may return the customer's receipt and deliver the goods.

In addition to facilitating debit or credit card payments, CyberCash will also provide independent electronic payment services.

Accounts are established directly with CyberCash and maintained on the basis of an account holder's CyberCash key and not on direct user identity. CyberCash accounts are non-interest-bearing holding accounts for cash which the account holder intends to transfer or has received through CyberCash. There are neither float nor checks, only signed receipts that can be sent to receivers to indicate that a transfer has occurred. The only way to place cash into or remove cash from a CyberCash account will be through a demand deposit account in a bank. Consequently, any funds in CyberCash accounts remain within the participating banks. CyberCash accounts are particularly suitable for electronic cash payments that are too small to be processed cost effectively as discrete credit card or debit card payments. This service is expected to permit the processing of the large volume of small payments which are expected to arise from a projected explosion in entrepreneurial electronic information publishing and commerce.

The technique used in CyberCoin is based on realizing electronic transactions between bank and credit card accounts. Unlike E-Cash there is no concept of coins and no actual transfer of money to users computer. The encryption technology used is RSA encryption.

6.6. SET [SET97]

Visa and MasterCard produced SET to be their standard for processing credit card transactions that travel over networks. The system includes a large amount of strong cryptology to authenticate transactions and ensure that the system remains secure. A large portion of the cryptography is based on public-key systems, and SET makes heavy demands upon a certification authority for support.

The SET system is similar to other credit card front-ends like CyberCash or *i*KP.

While using credit cards in the conventional way. The customer negotiates with a merchant to buy a product. When the price is set, the merchant calls the acquirer to request authorization for the customer. The aquirer is most likely a bank, that processes transactions with the issuer and asks for the amount to be set aside to cover the transaction. If the credit is available, the merchant receives authorization and delivers the goods.

Later the merchant posts the transaction by officially requesting repayment. This may occur several days or even months afterwards because the merchant may not deliver the goods immediately after authorization. During this time the amount *n* can not be used because it has been set aside even though the charge has not been officially made. Hotels can serve as an example here as they ask for large authorizations to cover extra charges.

when the charge is posted the acquirer informs the issuer of the new amount. The credit is consumed, the charge is made to the customer and the issuer gives the acquirer the funds. The merchants receives the payment after a certain amount of time which may differ from contract to contract.

SET is aimed to support all these operations over the Internet through a mixture of digital signatures and encryption.

The wide variety of people and companies in the SET universe sign their transactions with digital signatures. Such a wide spread use of digital signatures requires a solid infrastructure for maintaining certificates guaranteeing public-keys. These certificates form a chain of responsibility that vouches for the public part of every key pair so people know whether to trust it or not. A certificate is like a letter of credit as someone it states that someone is willing to stand behind this signature.

The SET standard relies on a particularly strong chain of *certificate authorities* (CA's), that is, an organization that offers to back up a person's signature and vouch for its authenticity. A customer or card holder using the SET system will have his own public-key pair and the public part will be packaged in a certificate that includes the digital signature of the issuer. This is generated by the *cardholder certificate authority* (CCA), which works in cooperation with the issuer. Any merchant can examine the certificate and see that the issuer is going to stand behind the charges as the public-key was signed by the CCA.

How could you be sure of the CCA's signature? It might come with another certificate by a higher authority. And this can go on forever. As this chain has to stop eventually. At the top is the *root certificate authority* or Root CA. This Root CA may guarantee the signatures of many groups. Beneath the Root CA is the *Brand CA*, which is set up by each brand (e.g. Visa, MasterCard or AMEX). The SET also imagines that there may be optional geopolitical CA that would guarantee local signatures.

If a customer begins to sign off on a charge, the merchant must check at least four digital signatures and certificates. The customer's signature is backed up by signature of CCA, which is backed by Brand CA, which in turn is backed by Root CA. A geopolitical CA could also exist in the chain.

But this check need not to be done for every transaction. Many customers will have their cards issued by the same bank hence backed by the same CCA.

There is also a similar organization for the merchants, and acquirers. The merchants have their digital signature pairs that are backed up by the *merchant certificate authority* (MCA). A customer might want to check out the signature of a merchant on a receipt ant MCA stands behind. There is also the *acquirer payment gateway certificate authority* (PCA), that backs up the acquirer signatures.

Certificates in essence are like the plastic cards today. If a CCA issues a certificate it is willing to stand behind. But if a customer abuses limits credit card issuers might want to draw the card back. The solution in SET is the revoked certificates. Which is very much alike the denial lists that were being and still distributed which contain the card numbers that will be denied by the issuer.

But this is not a smart solution and is very time consuming. The easiest solution is giving each certificate an expiry date minimizing the size of the list.. SET includes a technique known as thumbprint, which is used to surrogate the revoked certificates list which is the hash of the entire list. Someone who wants to check a certificate must return to the certificate authority to get a list of the revoked certificates.

The thumbprint allows a merchant to maintain a list of revoked certificates. When a new customer offers a card, the merchant can check its local list to see if the certificate is there. If not then the merchant can ask CCA for the thumbprint and compare it with hash value of the local list. If the hash values match he can be sure that his list is up-to-date.

The certificates act as a match between a cardholder and the digital signature. In many ways, they are just like the plastic cards in use today. These cards contain the expiration date, the account number, the person's name and even a photograph. The SET certificates could contain all of this information, but they don't.

In fact, no personal information is to be gotten from a SET certificate. The name is not included and the account number is blinded from the view. So the merchant may receive a signature and make sure that it was generated by the private key matching the certificate, but he can not extract the name if the customer or the account number. Which prevents fraud by dishonest merchants.

The blinding algorithm uses a *keyed hash algorithm* that acts like a public-key signature. The card holder and the CCA agree on a shared secret by exchanging random values. This acts as a key to the hash of the actual credit card number when it is appended to the credit card data before the hash function is computed. The result is unique value that acts like a pseudonym for the customer. It uniquely identifies the customers, but it won't reveal their actual identities or their account numbers.

The certificates for merchants and the acquirer clearing houses contain extra information. Their name and identity are left clear as there is little value in keeping them anonymous. In fact, their whole authority derives from their stability and reputation. The certificates can also contain data about transaction limitations.

SET imagines that certificates will not be used in the beginning because of the large overhead involved in distributing the keys and certificates to all card holders. So merchants will be able to present the charges without customer's signatures or certificates. Eventually, the acquirers will require the merchants to provide this.

The possibilities in the electronic environment may offer new opportunities in the arena such as transaction limits. Which may force the card holder not to spend below or above a certain amount for a transaction. Which may prove to be useful.

SET involves three parties: the card holder, the merchant, the acquirer. At the highest level, the cardholder asks to purchase something, the merchant asks the acquirer for enough credit, the acquirer responds. If all is fine the merchant delivers the goods and asks the acquirer to finalize the payment. The transaction in a bit more detailed is as follows:

PInitReq - **The Initialization Request** This is an optional message that the card holder's machine may generate to start the transaction. Its main job is to introduce the card holder, select the card brand and synchronize the certificates. The cardholder will make a list of certificates it holds and send the hash to the merchant.

PInitRes - **The Merchant's Response** The merchant decodes the message and starts a record for the upcoming transaction. The most important phase is to check through the list of certificates to make sure that the card holder has the latest certificates.

PReq - **The Payment Request** When ready to pay, the cardholder sends this message containing the data about the card and the amount. The details of the order are transmitted "out of band" that is SET protocol does not care about this.

This message is broken into two parts, The order information (OI) is for the merchant and the payment information (PI) is for the acquirer. The merchant does not touch the PI and is not able to read it.

The OI's main job is to carry a hash value of order data, This prevents disagreeing with the purpose of the transaction. When the merchant receives OI, it is compared against the hash value of his order details. The merchant will also send this to the acquirer who can be sure that all parties agree on the transaction by comparing it to the hash value which is also contained in PI.

The PI contains information for the acquirer, including the customer's description of the transaction. Merchant's ID number, the amount, scrambled version of customer's account number, and hash of the order data. There is also a slot for a random key for establishing a secure link between customer and the acquirer.

The OI and PI are mixed together with a "dual signature". PI is encrypted with the public key of the acquirer. Before this is done, the OI and the PI are hashed together in a fairly complicated procedure to ensure that neither can be changed without affecting the final signature, The result is signed by the cardholder.

If the cardholder has no certificate, then the result is simply hashed together and the PI information is encrypted with the public key of the acquirer.

PRes - **The Merchant's Response** The Merchant acknowledges the cardholder's request with this message. Before sending it, the merchant must verify some of the information in PReq. That is, the hash of the order data must correspond to the merchant's idea of what is being ordered. Also, the merchant may check to see that the challenge values produced during the optional PINit phase are copied over correctly. This value is copied over again and again in this transaction to prevent replay attacks.

The merchant can also check to make sure whether the correct signature is in place. Eventually, the acquirer will require all customers to have certificates and refuse to process transactions without certificate-based signatures. This requirement is noted in the acquirer's certificate, which the merchant already has on hand. A quick check is all the merchant needs to make sure that the acquirer will accept the transaction. This is one advantage of distributing information through the certificates.

At this point, the SET protocol leaves the merchant several options. The Pres message is issued only when the merchant is willing to sign off on the transaction. This usually will be after the merchant receives authorization from the acquirer but not necessarily. The merchant may choose to announce that the transaction is complete before authorization to save time. If the amount is small enough and the loss is small, the risk often is worth taking.

The merchant also can delay sending the PRes message until after capture is over; that is, until after the merchant and the acquirer have completed the transaction and the acquirer is sending the money directly to the merchant. This adds an additional delay, but the merchant may choose to do this.

The PRes includes all of the available information. At the very least it includes the transaction ID, and a code indicating the status of the transaction. If either the authorization or the capture is finished, then the result codes are added to the end. This may include a separate message from the acquirer back to the cardholder.

The entire message is signed by the merchant.

InqReq - **A Status Request** The cardholder may request information on the status of the transaction any time after sending the Preq message. This message is quite short and it only contains a new challenge number and a copy of the transaction ID. The random challenge numbers ensure that the merchant will be able to distinguish between multiple InqReq messages.

This message also may be sent after the merchant has approved the transaction. The cardholder may want to find out additional information about the authentication and capture of data. These might not have been present in the merchant's first approval message if the merchant chose to send approval before everything was processed.

If the card holder has a certificate, then he will sign this message.

InqRes - **The Merchant Responds** The merchant responds by sending the current version of the PRes in the same format, which includes the latest value of the challenge number, the transaction ID, and a code indicating the status. If there are either

authentication or capture responses from the acquirer, then they are included in this message.

The response code encapsulates all of the various responses from the merchant. These include the finality of the transaction and any potential errors.

This is signed by the merchant.

AuthReq - **The Merchant Requests Authorization** When the merchant requests authorization for a payment, this message carries the data. It is normally sent by the merchant before responding to the customer but it may be sent afterward if the merchant chooses to absorb the risk.

This message has two jobs: to synchronize certificates with the acquirer and to pass along the transaction data. The merchant has a copy of both the customer's and the acquirer's certificate. This message includes a thumbprint of both of these certificates to ensure that they are current. A merchant who doesn't have these values must pause and exchange PcertReq messages to get all of the right certificates.

If all of these are available, then the merchant must send along the PI information from the PReq message. This is the customer's card data and the customer's version of the transaction. The merchant includes its hash of the order data so the acquirer can confirm later that both are in agreement concerning the terms of the transaction.

The merchant may also include additional data about the customer's address that may be used to verify the cardholder. Merchant's often refuse to ship to anything other than the billing address on the card. If the merchant is suspicious about the customer, an additional flag can be sent to alert the acquirer to take special care to check for the signs of credit card fraud.

The final information depends on the details of the transaction. If the merchant wants only to authorize a card for payment, then the message includes the authorized amount. This may be larger than the initial charge if the merchant wants to both authorize and capture the money from the transaction, this can be accomplished in a single transaction by setting the appropriate flags.

If the merchant chooses to separate the authorization and capture phases, then the merchant later must send a CapReq message requesting capture.

AuthRes - **The Authorization Response** The acquirer checks out the request for a transaction and sends this response back if everything is in order and the money is available. Otherwise, it sends an error message.

The acquirer checks the copy of the PI from the cardholder's PReq has a valid signature, if it is required. Then it makes sure that both the card holder and the merchant agree on the hash value of the order data.

The customer's account number is bound up in a scrambled field (PAN) that must be decoded. After it is in the clear, the account is checked and the appropriate authorization and capture is done. This account number may be returned to the merchant in the

AuthRes message if the merchant is set to receive it. Many Merchants keep spending records on their customers. This data traditionally was provided to them. If the merchant and the acquirer agree to it, then the merchant gets this information and the transaction is not anonymous.

If the merchant's version of the certificates is wrong, then the acquirer would include new copies in this message as well as new thumbprints.

The message also contains proper response codes to the authentication and capture requests. These are sent on to the cardholder when the merchant receives the AuthRes and generates the Pres message.

CapReq - **A Separate Capture Request** When the merchant chooses to separate authentication from capture, the merchant must follow up with this message at a later date. This message can close out multiple transactions.

The data identifies the transaction and is returned with the authorization so the acquirer can correlate them.

CapRes - **The Capture Response** When the capture requests are processed, the acquirer sends this message and begins the transfer of the money.

AuthRevReq - **Reducing the Authorized Amount** This is a message used to reduce the authorization for a transaction. Contains the transaction ID, challenge number, authorization limits. And is signed by the merchant.

AuthRevRes - **The Acquirer Approves** When the authorization is reduced, the acquirer responds with a signed message.

CapRevReq - **Capture Reversal** This message is almost the same as the CreditReq. They return money from the merchant to acquirer for the cancellation of a previous transaction.

CapRevRes - **Capture Reversal Response** This message signs the approval of the credit.

CredRevReq - **Credit Reversal** This is sent if something goes wrong with CapRevReq or CreditReq.

CredRevRes - **Credit Reversal Response** The response to CredRevReq.

PCertReq - **Request for More Certificates** When the merchant determines that it lacks the proper certificates. This message is sent requesting the new certificates.

PCertRes - **The Certificates** The certificates and the thumbprint arrive in this message.

In summary the three parties involved, the card holder, the merchant and the acquirer hold certificates issued by different certificate authorities. These certificates are backed by a chain of signatures, that ends with Root CA.

Each party keeps a list of certificates to verify signatures. A card holder, might hold the certificates of several merchants, the merchant's CA, the Brand CA and the Root CA. When two parties begin to exchange messages, they also exchange the list of certificates they hold and a thumbprint, which is a hash of certificates. This allows them to make sure that they hold the same certificates.

The card holder encrypts the account number and a hash of payment information in the PI. The merchant adds it s own version of payment information and passes it. The merchant can't see the cardholder's data. The acquirer compares the two versions of transaction and processes if two sides agree.

The SET standard is being backed by great players in the Arena such as VISA and MasterCard and various governments are backing the initiative. In June 1997 the first transaction using SET has been realized. But the widespread use, and universal acceptance is not a reality yet.

6.7. NetBill [NB97]

Information Networking Institute of Carnegie Mellon University is sponsoring 'NetBill,' which is a scheme which would broker transactions through a third-party financial institution, similar to the system of debit cards which is already in existence. The system requires both the customer and merchant to maintain accounts with the third party acting as the broker. When a customer wants to make a purchase of goods or services, a message is sent using the NetBill software to the third party financial institution instructing a transfer of the relevant amounts to the seller's account. Similarly, the Information Sciences Institute at the University of Southern California is developing both a cash/cheque model and a debit card model based on a similar structure. Selection can then be made according to whether the buyer is or is not seeking anonymity. ISI is working with a several banks including Bank of America and Citibank. [FT94b]

6.8. Credit Card-Based Systems

The major credit companies have been the only established financial institutions actively investing in future payment systems on the Internet. Credit card based payment systems have some limitations already identified earlier in the paper. Nevertheless, telephone-based credit card payments currently account for the majority of Internet commercial transactions. A secure and efficient method to transmit credit card details will clearly be welcomed as a major step forward. A number of initiatives are underway. Both of the major credit card companies have linked up with software houses to develop the encryption software systems which will be required.

On November 9, 1994, Visa and Microsoft announced that they were jointly developing software based on a security architecture which will enable customers to make purchases using coded credit card, debit card and charge card numbers on-line. The intention is to add future optional upgrades to the PC-based Windows operating system which will permit card details to be encrypted so that payment will be secure. The

intention is to make the software available as part of future optional upgrades to the Windows operating system, which will also incorporate a user-friendly interface to the Internet as standard. Visa International is controlled by its members, the banks that issue VISA credit cards. Voting rights are in proportion to the volume of business transacted. In common with other credit card-based systems, Visa's Internet payment system will be limited to merchants who have been approved by the organization to accept their credit cards.

In January, 1995, MasterCard announced that it will start providing banking services on the Internet. MasterCard will base their services on the Netscape computer software, which is designed to encrypt transactions and keep them secure. Netscape has already successfully developed a suite of programs for accessing the Internet which are sold commercially. Netscape is also working with Bank of America and with First Data Corp as well as MasterCard International.

6.9. Smart Cards [SC97a, SC97b]

Some observers believe that the only way in which payments over the Internet can be made secure is to physically separate authentication from the process which provides the communication links between buyer and seller.

The major smart card systems will place the value "on the card". That is, the card will be considered the repository for the money. If it is lost or stolen the, the money may be gone, although the central companies may have the ability to replace it like traveler's checks.

The main difference between the smart card systems and the other systems is the amount of central control. The smart cards act like branch offices of a business. They have the ability to transfer money in and out without getting permission from the central office. For instance, two people can meet on the street and move money between their Mondex smart cards without discussing the matter with a central computer. Systems like DigiCash must get central approval before any money moves.

The most obvious advantage in the lack of central control is cost. The network infrastructure required to process all of the payments is extremely high, which will most likely be amortized over the costs of the cards in forms of fees or interests.

A number of companies are therefore examining the use of smart card readers linked to a personal computer. A smart card used to store money in the same way as a phone card would make it possible to separate authentication from the payment process. Smart cards technology would permit them to be charged up with cash at an ATM or separately using a proprietary bank network. The value in the smart card could then be transferred securely and anonymously over the Internet.

6.10. Citibank's Transaction Cards.

Citibank has announced that it is experimenting with a smart card based digital cash. The system is quite similar to digital cashier's checks. The packets that contain the information about the bank and the amount are signed by the Bank's signature.

The smart card maintains a secure file system that catalogs the cash on the hand and apply a secure digital signature to back up each transaction. The current version of the system can support multiple currencies, credit lines. Each can be exchanged and two smart cards can buy and sell foreign currencies between themselves.

The transactions in the system are straightforward. Each note consists of the name of the bank, a serial number, the amount, and a certificate used to guarantee the note. This package of bits is signed with an RSA signature produced by the key pairs in the certificate. The initial note is produced by a special sealed module kept secure by the Bank.

When money is transferred from one smart card to another, the spending smart card wraps a new layer of bits at the end of the note. This data includes, the amount being transferred, the smart card's identification number, the time and the smart card's certificate. The smart card then signs the entire package, effectively guaranteeing that it obtained the note legitimately and is now offering it legitimately. The notes can travel through several smart cards before returning to the bank. Through each step, each smart card in the chain adds its own signature building a chain of custody for the note and the value it contains. This chain can be used to reconstruct the flow of money through the economy to either stop money laundering or trace fraud.

The notes can also be split in parts. If your smart card contains a $10.00 note and you want to spend $0.50 the smart card will create a new note worth $0.50. This note will contain the old $10.00 note at its core and the outside wrap of the data will specify that only $0.50 of its value is being transferred to this new note. The smart card may then split off other fractions from the same $10.00, making sure that all of them add up to $10.00.

Each note can be split into parts by each smart card that holds it in the chain. So each note can spawn a wide variety of notes with fractional values. And when these bills return to the bank they can be checked whether they add up to the value of the original bill.

The smart card receiving the money must validate it to make sure it is correct. By the time a note arrives at a particular card, it might have passed through ten other smart cards. Each card added its own signature and a certificate used to validate the signature. The receiver checks each signature in the chain to make sure it is valid and matches the certificate. This is a fairly secure chain of custody. In this system the accepting smart card will trust a bill without checking with the central server.

When two cards are used for a transaction the following steps are followed:

Authentication The two "trusted agents" authenticate each other by exchanging digital signatures backed by certificates. That is,

(a) The first produces a random number, r_1, signs it and sends it to the second.

(b) The second checks the signature, signs r_1 and returns it. Then it generates a second random number, r_2, signs it, and sends it on.

(c) The first checks the signature on r_1, and r_2. Then it signs r_2 and returns it.

(d) Now both cards have established that they both know the secret keys to the public-key pair backed up by certificates.

Secure Channel The two random numbers can be combined to make a key that will be used to encrypt all further communication.

Exchange of Data If the transaction is being used to sell data or software over the network, the seller encrypts the data and sends it.

Cash is Tendered The purchaser's smart card produces a collection of notes that add up to the correct amount and sends these to the reader's smart card.

Cash Verification The seller's card examines the digital signatures in notes received. If they check out, then the seller's card send a receipt R_1, acknowledging that the notes arrived safely.

Purchaser Accepts the Transaction When the purchaser's card receives the acknowledgment . the notes are marked as spent and another receipt (R_2) releasing the cash is issued.

Seller Accepts Money When the second receipt is received the notes are marked as owned.

Seller Releases Data The seller supplies the key to the data.

Any disputes about the transaction can be solved later. The bank can investigate the status of transaction that has ended with frozen cash.

These cards also introduce another concept credit-lines along with digital cash. Which is something in-between digital cash and credit cards. These cards can also contain notes backed with credits from other institutions. But unlike normal cash these can not be transferred from card to card. They have to be returned to the issuer after being used. Which is for abiding by the regulation as otherwise it would be equivalent of issuing money.

The system is quite decentralized and allows off-line transactions. Which lowers the costs. The system relies on the integrity of the smart cards to prevent double spending. Double spending is technically possible but only if one is able to break into the smart card and duplicate what it does.

The design of the system also includes several layers of protection. The smart cards lack the ability to mint money. They can only spend notes that were produced by the central

money creator. Every transaction must be guaranteed by a digital signature. If double spending occurs, the bank can determine which card is double spending. This may not identify the cheating person immediately, but it can produce a good trail of where the notes were spent.

The decentralized nature also means that it is impossible for the bank to shut down a rogue smart card. If someone found a way to work around the tamper-resistant packaging of a smart card and turn it to do his bidding, then the bank could not shutdown this person. Even the bank would be able to identify the card from the multiple spent bills, they would not be able to stop the card so that it would not appear as legitimate to the other cards. The solution is periodically changing certificates, which would stop the identified rogue cards.

The system also requires that the money "age". That is, each note comes with an expiration date. After that, the notes must be exchanged for new notes from the bank. This forces the notes to flow through the bank every n days and allows the bank to identify any fraud or double spending at that time. This can be done behind the scenes and in the background whenever the customer interacts with the bank via ATM or Internet or smart phones.

6.11 Mondex [MDX97]

National Westminster Bank is one of several organizations developing smart card technology to create what is referred to as an 'electronic purse'. While the magnetic strip on a credit card can only hold one or two lines of information, smart cards can store several pages of text. This permits credit card-sized smart cards to be used to transfer cash amounts which can then be 'spent' using special terminals. The initial applications will involve replacing small cash payments made to retailers (e.g., for newspapers, confectionery) or to pay for services like public telephones or public transport.

NatWest's longer range vision is for Mondex to be used on a more global basis to buy goods from suppliers on the Internet. Payments would involve having a special smart card reading device linked to the PC and software that recognized the card reader. Although this may prove a cumbersome alternative compared with more direct payment methods, NatWest and other smart card developers believe there would be a key benefit from the inherent security which is built into smart cards. Smart cards could also be more readily integrated into the existing networks of ATMs, integrating with what is likely to remain the predominant form of payment transaction, namely cash.

6.12 Electronic data interchange (EDI) [EDI96]

Money is not the only element needed for realizing transactions. And electronic commerce needs more than that before exchanging money.

Electronic data interchange (EDI) involves the exchange of structured business documents, such as orders and invoices, directly between computers. Financial EDI extends this process to the payment and settlement process performed by banks. At present EDI is principally used for inter-company communication, removing the need

for paper-based transmission of orders and remittance information. Although EDI standards were developed in the early 1980s, actual implementation has been very modest to date [FT94f]. Out of several million individual businesses which are registered in the United States, currently only 44,000 companies are using EDI to exchange business data electronically. Furthermore, of this group, only 10% are also using financial EDI (4,400), with the rest ending their electronic links with a paper-based payment trail to their respective banks. [FRB94]

The experience in the United States is mirrored in Europe and Japan. In Europe, regular users of EDI have only recently increased to 40,000 companies, compared to 20,000 in 1990, and it is now acknowledged that in order for electronic commerce to happen, a whole series of changes both technical and organizational have to be implemented. Actual implementation rates vary significantly from country to country and are further constrained by the fact that currently only 50% of European EDI traffic is based on the international United Nations-based EDIFACT standard [FT94e]. Successful EDI implementations remain largely the preserve of major retailers such as Sears Roebuck in the US and Marks & Spencer in the UK. Other examples include General Motors, which in 1993 collected US dollars 12 billion from its US auto dealers using an ACH based financial EDI system. In Europe, EDI service providers like IBM, BT, AT&T as well as INS/General Electric Information Services (Geis) are gearing up for substantial future growth. Low take-up rates to date have been explained in terms of high start-up costs, lack of familiarity, as well as overall low take-up rates, which frequently require businesses to maintain dual paper-based and electronic remittance systems in parallel.

One of the main reasons that EDI did not take off immediately was the lack of global networks and the complexity of the X.12 standard that served as the basis for implementations was a bit complex for daily use.

In June 1997, various players in the arena of digital commerce including IBM, Sun, VISA have announced that they are working on a new standard that would serve as an infrastructure for business communication. The new standard is called "OBI" and at the time of this writing not enough details are supplied to the public.

These high level standards are the most important barriers in front of the electronic commerce. The theoretical and practical infrastructure for establishing secure communications and secure financial transactions is already present. Establishment of mid-layer standards that may work one layer above SSL, that can cover the negotiation of kind of encryption, cash system that will be used for a transaction described universally will be the ultimate solution for now. In the future when more complex systems can find easier application, such as when public phone are modified to accept smart cards the transactions will become totally different. But now, and even then, there will be a need to cover different standards to enable them to work in their current form. [Ley93, Emm90]

9. Appendix

Internet Sources

http://www.fv.com/ The home page of the First Virtual system.

http://www.commerce.net/ The home page of CommerceNet, a non-profit consortium that is creating protocols for secure Web usage.

http://www.digicash.com/ The home page of Digicash.

http://www.zurich.ibm.ch/Technology/Security/sirene/projects/cate/index.html CAFE (Conditional Access For Europe) is a European coalition of companies and universities developing portable electronic wallets.

http://www.cybercash.com/ The home page for CyberCash.

http://www.ini.cmu.edu:80/netbill/ The NetBill is a project of the information Networking Institute at Carnegie-Mellon University.

http://nii-server.isi.edu/info/NetCheque/ The home page of the NetCheque system developed at the Information Sciences Institute and the University of Southern California.

http://www.openmarket.com/about/technical/payment/ Open Market page describing its payment system.

http://www.u-net.com/gmlets/ The home page of LET system that runs Manchester Money.

http://www.netchex.com/ The home page of NetChex.

http://www.cts.com/ The home page for CTSNET's marketplace.

http://www.amazon.com/ The home page for online bookstore that accepts various payment options.

http://www.cdnow.com/ The home page for online music store.

10. References

[ACG84] W. Alexi, B. Chor, O. Goldreich, and C.P. Schnorr. RSA and Rabin functions: Certain parts are as hard as the whole. *SIAM Journal of Computing*, October 1984.

[APACS94] Association for Payment Clearing Services (APACS) (1994). Annual report, p. 28.

[BIS94] BIS Strategic Decisions (1994). Projected growth of the U.S. home shopping industry. Interactive Consumer Media & Electronics.

[BLA79] G.R. Blakley. Safeguarding cryptographic keys. AFIPS Conference Proceedings,49. 1979

[BS91] E. Biham, A.Shamir. Differential cryptanalysis of DES like crypto-systems. Advances in Cryptology - CRYPTO 90 Proceedings, Springer - Verlag, 1991

[BT95] Banking Technology (1995, March 25). The next generation - Internet.

[CC96] CyberCash System, 1996 <http://www.cybercash.com/cybercash/about/index.html>

[CEG88] D.Chaum, J.H.Everste, J.V.Graff Demonstrating possession of a discrete algorithm without revealing it - Advances in Cryptology - EUROCRYPT 87, Berlin 1988, p 127-141

[Cha83] D. Chaum. Blind signatures for untraceable payments. In *Advances in Cryptology: Proceedings of Crypto 82.* Plenum Press, 1983

[Cha85] D.Chaum. Security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM*, October 1985.

[Cha88a] D. Chaum. Blind signature systems. U.S. Patent #4,759,063, July 1988.

[Cha88b] D.Chaum. Blinding for unanticipated signatures. In *Advances in Cryptology - EUROCRYPT '87 Proceedings.* Springer-Verlag, 1988

[Cha92] D.Chaum. Achieving electronic privacy. *Scientific American,* August 1992

[Cho94] Chown, J. F. (1994). A history of money from AD 800. London : Routledge, pp 3-41.

[Cro94] Cronin, M. J. (1994). Doing business on the Internet - How the electronic highway is transforming American companies. New York : Van Nostrand Reinhold.

[Dav82] G. Davida. *Chosen signature cryptanalysis of the RSA public key cryptosystem*. Technical Report TR-CS-82-2, Department of EECS, University of Wisconsin, Milwaukee, 1982.

[DO86] Y. Desmedt and A.M. Odlyzko. A chosen text attack on the RSA cryptosystem and some discrete logarithm schemes. In *Advances in Cryptology — Crypto '85*, pages 516–522, Springer-Verlag, 1986.

[EC97] E-Cash, Digicash Inc. 1997 <http://www.digicash.com/ ecash/ecash-home.html>

[EDI96] The EDI Standards <http://www.premenos.com/standards/>

[Emm90] Margaret A. Emmelhainz. *Electronic Data Interchange: A Total Management Guide.* Van Nostrand Reinhold, NewYork, 1990

[FRB94]) Federal Reserve Bulletin (1994) ,Vol 80 No. 4.

[FT94a] Financial Times (1994, September 13). International Capital Markets.

[FT94b] Financial Times (1994, October 13). World Trade News: Taking the paper out of trade - The quest for more efficient commerce.

[FT94c] Financial Times (1994, November 15). Survey of Computers in Finance .

[FT94d] Financial Times (1994, June 6). Media Futures: Doing cyber business -

[FT94e] Financial Times (1994, October 26). Survey of Technology in the Office

[FT94f]] Financial Times (1994, June 28). Survey of Computer Networking

[FT94g] Financial Times (1994, November 29). Survey of Global Custody

[FT94h] Financial Times (1994, November 15). Survey of Computers in Finance

[FT94i] Financial Times (1994, September 13). International Capital Markets

[FT94j] Financial Times (1994, October 9). What to do when chaps can't cope: Reform of UK Bank settlements procedures aims to reduce risk to the financial system.

[FT94k] Financial Times (1994, October 13). World Trade News: Taking the paper out of trade - The quest for more efficient commerce.

[FT94l] Financial Times (1994, November 29). Survey of Global Custody : Getting the right message across.

[FT94m] Financial Times (1994, November 29). Survey of Global Custody: Oil for the world's investment machinery.

[FT94n] Financial Times (1994, August 8). Economics Notebook: Payments and settlements.

[FT95] Financial Times (1995, November15). Survey of Computers in Finance.

[FV96] First Virtual System, <http://www.fv.com/demo/>

[Has88] J. Hastad. Solving simultaneous modular equations of low degree. *SIAM Journal of Computing*, 17: 336–241, 1988.

[Hic95] K.E.B. Hickman. The SSL Protocol. December 1995. <http://www.netscape.com/newsref/std/ssl.html>

[IKPP97] Internet Keyed Payment Protocols , 1997 <http://www.zurich.ibm.com/Technology/Security/extern/ecommerce/iKP.html>

[Kal92] B.S.Kaliski. The MD-2 message digest algorithm. Technical report, RSA laboratories. Inc, April 1992

[Kee95] L. Keeler. . CyberMarketing. AMACOM, 1995.

[Ker97] Kerberos: The Network Authantication Protocol. http://web.mit.edu/kerberos/www/

[KR95] B.S. Kaliski Jr. and M.J.B. Robshaw. The secure use of RSA. *CryptoBytes*, 1(3): 7–13, 1995.

[KWY94] Knudson, S. E, Walton II, J. K, & Young, F. M. (1994). Business-to-business payments and the role of financial electronic data interchange. Federal Reserve Bulletin Vol 80, No. 4, pp 269-278.

[Ley93] Valerie A. Leyland. Electronic Data Interchange: A Management View. Prentice-Hall, New York, 1993.

[Lin94] Lindsey, I. (1994). Credit cards - The authoritative guide to credit and payment cards. London : Rushmere Wynne.

[Mat97] M. Mathiesen. Marketing on the Internet. Maximum Press, 1997.

[MDX97] Mondex Technology, 1997 <http:// www.mondex.com/ mondex/cgi-bin/printpage.pl?english+global&technology.html>

[NB97] About Net Bill, <http://www.netbill.com/netbill/about.html>

[NIS92] NIST. Proposed Federal Information Processing Standard for Secure Hash Standard. *Federal Register,* 57(21), January 1992

[NIS93] National Institute of Standards and Technology. FIPS Publication 180: Secure Hash Standard (SHS). May 1993.

[Pre93] B. Preeneel. Analysis and Design of Cryptographic Hash Functions. Ph.D Thesis, Katholieke University, Leuven, 1993

[Riv91] R.Rivest. The MD-4 message digest algorithm. In *Advances in Cryptology-CRYPTO '90 Proceedings*. Springer-Verlag, 1991

[Riv92] R.Rivest. The MD-5 message digest algorithm. Technical Report RFC 1321, RSA Data Security, Inc. April 1992

[Rob90] Roberts, J. $1000 billion a day - Inside the foreign exchange markets. London : Harper Collins, 1990.

[Rob94] M.J.B. RobShaw. MD2, MD4, MD5, SHA and other hash functions. Technical Report TR-101, RSA Laboratories, July 1994

[RS95] E. Rescorla, A. Schiffman. The Secure HyperText Transfer Protocol. Internet-Draft, July 1995.

[RSA97] Cryptology Frequently Asked Questions. <http://www.rsa.com/rsalabs/LABSFAQ.PDF>

[SC97a] SmartCards <http://ctl77.nectec.or.th/~nopporn/smartcard/whatsmartcard.html>

[SC97b] SmartCard Technology <http://www.smartcard.co.uk/technology.html>

[SET97] Secure Electronic Transactions, May 1997 <http://www.mastercard.com/set/set.htm>

[Sha79] A. Shamir. How to Share a secret. Communications of the ACM, 22, 1979

[TE91] The Economist (1991, June 29). Plastic community: European payment cards.

[TE92a] The Economist (1992, November 7). European payment systems

[TE92b] The Economist (1992, October 17). Hand over the money: Central bankers want to bring payment systems closer

[TE93] The Economist (1993, September 11). The future surveyed: The challenge of global money

[TE94b] The Economist (1994, November 26). Electronic money:

[TE94] The Economist (1994, March 26). Bankers Trust's 2020 vision

[Win84] R.S. Winternitz. Producing one way hash functions built from DES. Advances in Cryptology: Proceedings of Crypto 83.