In the last chapter, we discussed the data integrity threats and the use of hashing technique to detect if any modification attacks have taken place on the data.

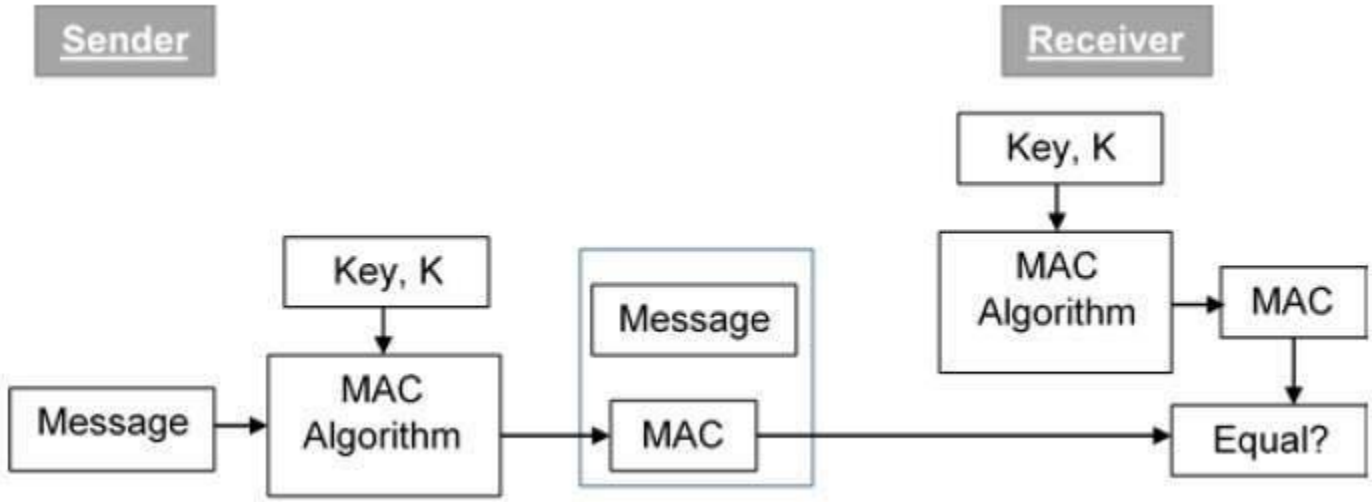
Another type of threat that exist for data is the lack of **message authentication**. In this threat, the user is not sure about the originator of the message. Message authentication can be provided using the cryptographic techniques that use secret keys as done in case of encryption.

Message Authentication Code (MAC)

MAC algorithm is a symmetric key cryptographic technique to provide message authentication. For establishing MAC process, the sender and receiver share a symmetric key K.

Essentially, a MAC is an encrypted checksum generated on the underlying message that is sent along with a message to ensure message authentication.

The process of using MAC for authentication is depicted in the following illustration −



Let us now try to understand the entire process in detail −

- The sender uses some publicly known MAC algorithm, inputs the message and the secret key K and produces a MAC value.
- Similar to hash, MAC function also compresses an arbitrary long input into a fixed length output. The major difference between hash and MAC is that MAC uses secret key during the compression.
- The sender forwards the message along with the MAC. Here, we assume that the message is sent in the clear, as we are concerned of providing message origin authentication, not confidentiality. If confidentiality is required then the message needs encryption.
- On receipt of the message and the MAC, the receiver feeds the received message and the shared secret key K into the MAC algorithm and re-computes the MAC value.

- The receiver now checks equality of freshly computed MAC with the MAC received from the sender. If they match, then the receiver accepts the message and assures himself that the message has been sent by the intended sender.
- If the computed MAC does not match the MAC sent by the sender, the receiver cannot determine whether it is the message that has been altered or it is the origin that has been falsified. As a bottom-line, a receiver safely assumes that the message is not the genuine.

Limitations of MAC

There are two major limitations of MAC, both due to its symmetric nature of operation −

- **Establishment of Shared Secret.**
  - It can provide message authentication among pre-decided legitimate users who have shared key.
  - This requires establishment of shared secret prior to use of MAC.
- **Inability to Provide Non-Repudiation**
  - Non-repudiation is the assurance that a message originator cannot deny any previously sent messages and commitments or actions.
  - MAC technique does not provide a non-repudiation service. If the sender and receiver get involved in a dispute over message origination, MACs cannot provide a proof that a message was indeed sent by the sender.
  - Though no third party can compute the MAC, still sender could deny having sent the message and claim that the receiver forged it, as it is impossible to determine which of the two parties computed the MAC.

Both these limitations can be overcome by using the public key based digital signatures discussed in following section.

2.4 Secure Hash Algorithm (SHA) 512

The SHA 512 algorithm is an algorithm that uses the one-way hash function created by Ron Rivest. This algorithm is the development of previous algorithms SHA 0, SHA 1, SHA 256 and SHA 384 algorithms. Journal of research Christian Angga [9], 2007, explains how the cryptographic algorithm of SHA 512 is receiving input in the form of messages of any size and generates message diggest which has 512-bit length.

Its predecessor is SHA1, and MD5 which is a renewal of MD4, the linkage, and development of the hash algorithm, indicating that the algorithm has proven to have been found to be a collision vulnerability. Currently, the National Institute of Standards and Technology (NIST) has made SHA 224, SHA 256, SHA 384, and SHA 512 as the new standard hash function. In Table 2 the resume parameters show some hash functions.

SHA 512 hash function performs the same hash operation as SHA 2 operation in general [10]. SHA 512 hash function is a function that generates message diggest 512-bit size and 1024 bit block length. How the cryptographic algorithm works SHA 512 is to accept input in the form of a message with any length or size and will generate a message digest that has a fixed length of 512 bits as shown in Figure 3.

The workings of making message diggest with SHA 512 algorithm are as follows:

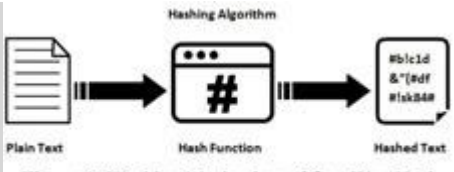


Figure 2. Working Mechanism of One Way Hash Function

1. The addition of bits

The first process is to add a message with a number of bit wedges such that the message length (in bits) is congruent with 890 mod 1024. The thing to remember is that the 1024 number appears because of the SHA 512 algorithm processes messages in blocks of 1024 sizes. If there is a message with a 24-bit length, then the message will still be added with the bundle bits. The message will be added with 896- (24 + 1) = 871 bits. So the length of the wedge bits is between 1 and 896. Then one more thing to note is that the bit bits consist of a bit 1 followed by the remaining bit 0.

2. Adding Long Message Redemption Value Then the next process is the message added again with 128 bits stating the length of the original message. If the message length is greater than 2128 then the length is taken in modulo 2128. In other words, if initially, the message length is equal to K bit, then 128 bit adds K modulo 2128, so after the second process is done then the message length now is 1024 bits.

3. Initialize Hash Value

In the SHA 512 algorithm, the H hash value (0) consists of 8 words with 64 bits in the hexadecimal notation as in Table 3.

3 METHODOLOGY

Table 2. Comparison of Multiple Hash Functions

Algorithm	The Size of the Message Digest (bit)	Message Block Size	Collision
MD2	128	128	Yes
MD4	128	512	Almost
MD5	128	512	Yes
RIPEMD	128	512	Yes
RIPEMD-128/256	128/256	512	No
RIPEMD-160/320	160/320	512	No
SHA-0	160	512	Yes
SHA-1	160	512	There is a Disability
SHA-256/224	256/224	512	No
SHA-512/384	512/384	1024	No
WHIRPOOL	512	512	No

This section will explain the systematic way used to solve the research problem and also the steps undertaken in the testing and analysis of this research. The stages consist of literary studies is to analyze the system used to determine the current conditions, needs, advantages, and disadvantages of these programs. This stage is done by reading several books, previous research journals, papers or articles that are appropriate or relevant as well as collecting resources from the internet both journals, websites, proceedings and source code that can be used in this research.

Needs analysis and system vulnerabilities are carried out to analyze the vulnerabilities and needs of the system used. The analysis is focused on the web-based application login system encryption function which aims to find out the advantages and disadvantages of the encryption method currently used when replaced using the latest algorithm method.

Needs analysis and design for improvement are to describe and display an overview of the encryption process when the login is done. The description carried out is by showing a flowchart and conceptual diagram so that the work process in which password encryption is carried out until the login activity occurs can be delivered and understood more clearly.

Mitigation performed with the implementation of the latest hash function algorithm calling the method, code change for patching and test results from implementation. Testing in this study was conducted to show a comparison between the use of MD5 encryption method and SHA 512 encryption method. Testing was done by Penetration Testing and User Acceptance Test. Penetration Testing is done by Brute Force testing while User Acceptance Test is done by filling out a questionnaire that is used as one of the recommendations to improve data security in web-based applications. These stages are described in Figure 4.

Digital signatures are the public-key primitives of message authentication. In the physical world, it is common to use handwritten signatures on handwritten or typed messages. They are used to bind signatory to the message.

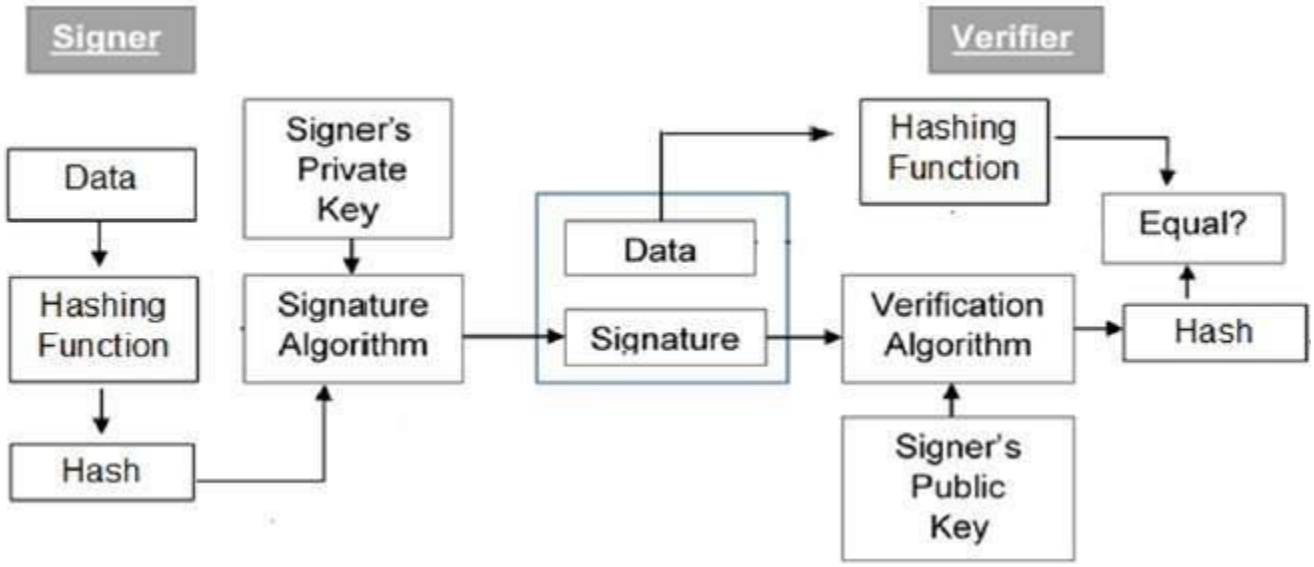
Similarly, a digital signature is a technique that binds a person/entity to the digital data. This binding can be independently verified by receiver as well as any third party.

Digital signature is a cryptographic value that is calculated from the data and a secret key known only by the signer.

In real world, the receiver of message needs assurance that the message belongs to the sender and he should not be able to repudiate the origination of that message. This requirement is very crucial in business applications, since likelihood of a dispute over exchanged data is very high.

Model of Digital Signature

As mentioned earlier, the digital signature scheme is based on public key cryptography. The model of digital signature scheme is depicted in the following illustration −



The following points explain the entire process in detail −

- Each person adopting this scheme has a public-private key pair.
- Generally, the key pairs used for encryption/decryption and signing/verifying are different. The private key used for signing is referred to as the signature key and the public key as the verification key.
- Signer feeds data to the hash function and generates hash of data.
- Hash value and signature key are then fed to the signature algorithm which produces the digital signature on given hash. Signature is appended to the data and then both are sent to the verifier.
- Verifier feeds the digital signature and the verification key into the verification algorithm. The verification algorithm gives some value as output.
- Verifier also runs same hash function on received data to generate hash value.

- For verification, this hash value and output of verification algorithm are compared. Based on the comparison result, verifier decides whether the digital signature is valid.
- Since digital signature is created by 'private' key of signer and no one else can have this key; the signer cannot repudiate signing the data in future.

It should be noticed that instead of signing data directly by signing algorithm, usually a hash of data is created. Since the hash of data is a unique representation of data, it is sufficient to sign the hash in place of data. The most important reason of using hash instead of data directly for signing is efficiency of the scheme.

Let us assume RSA is used as the signing algorithm. As discussed in public key encryption chapter, the encryption/signing process using RSA involves modular exponentiation.

Signing large data through modular exponentiation is computationally expensive and time consuming. The hash of the data is a relatively small digest of the data, hence **signing a hash is more efficient than signing the entire data**.

Importance of Digital Signature

Out of all cryptographic primitives, the digital signature using public key cryptography is considered as very important and useful tool to achieve information security.

Apart from ability to provide non-repudiation of message, the digital signature also provides message authentication and data integrity. Let us briefly see how this is achieved by the digital signature −

- **Message authentication** − When the verifier validates the digital signature using public key of a sender, he is assured that signature has been created only by sender who possess the corresponding secret private key and no one else.
- **Data Integrity** − In case an attacker has access to the data and modifies it, the digital signature verification at receiver end fails. The hash of modified data and the output provided by the verification algorithm will not match. Hence, receiver can safely deny the message assuming that data integrity has been breached.
- **Non-repudiation** − Since it is assumed that only the signer has the knowledge of the signature key, he can only create unique signature on a given data. Thus the receiver can present data and the digital signature to a third party as evidence if any dispute arises in the future.

By adding public-key encryption to digital signature scheme, we can create a cryptosystem that can provide the four essential elements of security namely − Privacy, Authentication, Integrity, and Non-repudiation.

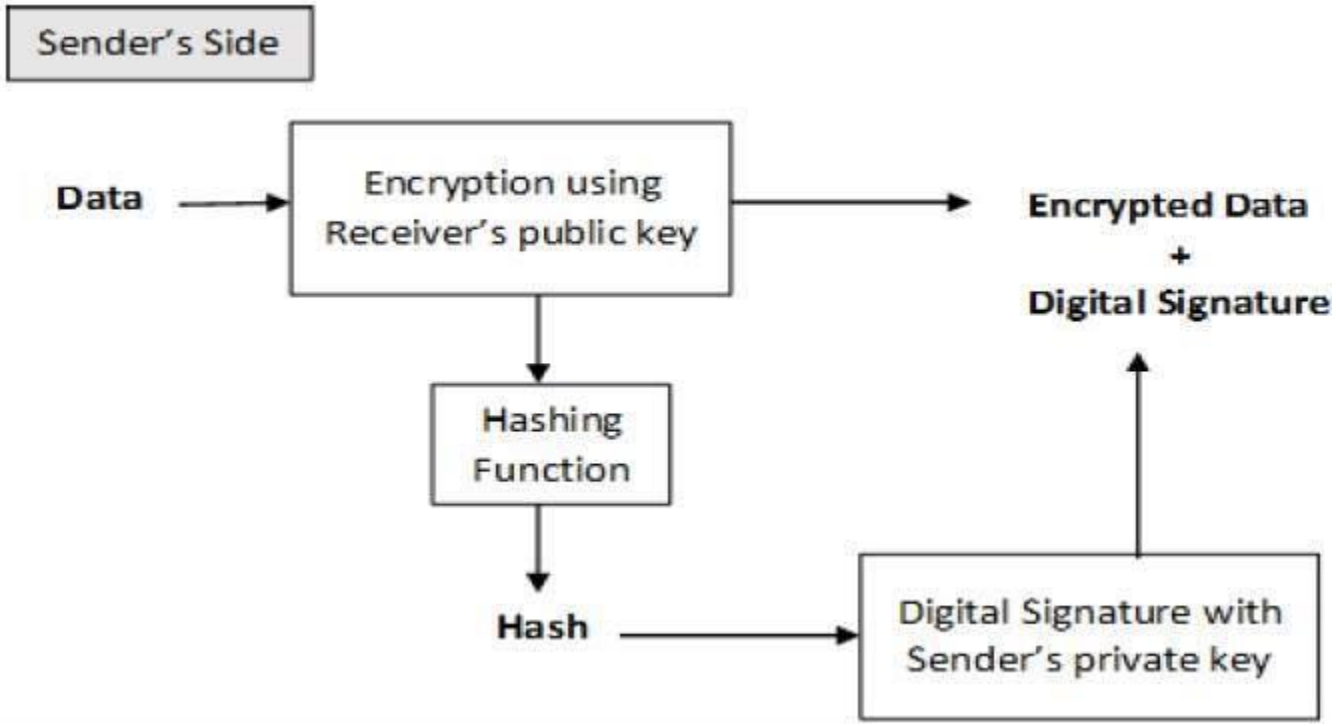
Encryption with Digital Signature

In many digital communications, it is desirable to exchange an encrypted messages than plaintext to achieve confidentiality. In public key encryption scheme, a public (encryption) key of sender is available in open domain, and hence anyone can spoof his identity and send any encrypted message to the receiver.

This makes it essential for users employing PKC for encryption to seek digital signatures along with encrypted data to be assured of message authentication and non-repudiation.

This can archived by combining digital signatures with encryption scheme. Let us briefly discuss how to achieve this requirement. There are **two possibilities, sign-then-encrypt** and **encrypt-then-sign**.

However, the crypto system based on sign-then-encrypt can be exploited by receiver to spoof identity of sender and sent that data to third party. Hence, this method is not preferred. The process of encrypt-then-sign is more reliable and widely adopted. This is depicted in the following illustration −



The receiver after receiving the encrypted data and signature on it, first verifies the signature using sender's public key. After ensuring the validity of the signature, he then retrieves the data through decryption using his private key.

Data is prone to various attacks. One of these attacks includes message authentication. This threat arises when the user does not have any information about the originator of the message. Message authentication can be achieved using cryptographic methods which further make use of keys.

Authentication Requirements:

- **Revelation:** It means releasing the content of the message to someone who does not have an appropriate cryptographic key.
- **Analysis of Traffic:** Determination of the pattern of traffic through the duration of connection and frequency of connections between different parties.
- **Deception:** Adding out of context messages from a fraudulent source into a communication network. This will lead to mistrust between the parties communicating and may also cause loss of critical data.
- **Modification in the Content:** Changing the content of a message. This includes inserting new information or deleting/changing the existing one.
- **Modification in the sequence:** Changing the order of messages between parties. This includes insertion, deletion, and reordering of messages.
- **Modification in the Timings:** This includes replay and delay of messages sent between different parties. This way session tracking is also disrupted.
- **Source Refusal:** When the source denies being the originator of a message.
- **Destination refusal:** When the receiver of the message denies the reception.

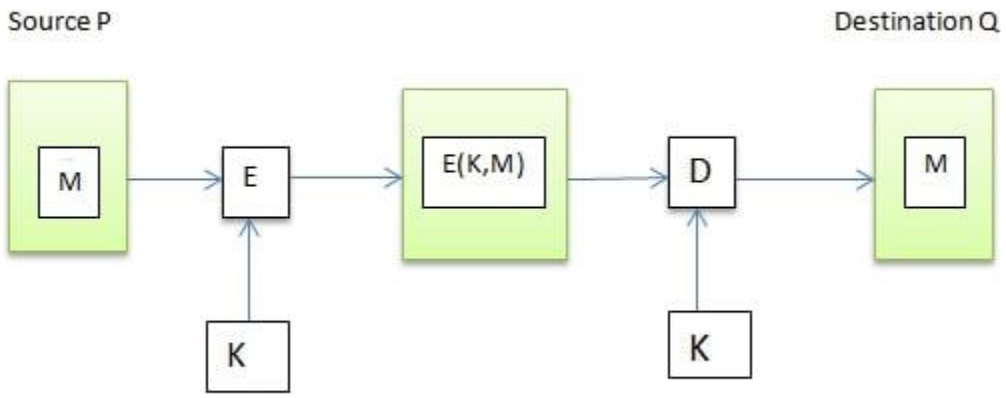
Message Authentication Functions:

All message authentication and digital signature mechanisms are based on two functionality levels:

- **Lower level:** At this level, there is a need for a function that produces an authenticator, which is the value that will further help in the authentication of a message.
- **Higher-level:** The lower level function is used here in order to help receivers verify the authenticity of messages.

These message authentication functions are divided into three classes:

- **Message encryption:** While sending data over the internet, there is always a risk of a Man in the middle(MITM) attack. A possible solution for this is to use message encryption. In message encryption, the data is first converted to a ciphertext and then sent any further. Message encryption can be done in two ways:
- **Symmetric Encryption:** Say we have to send the message M from a source P to destination Q. This message M can be encrypted using a secret key K that both P and Q share. Without this key K, no other person can get the plain text from the ciphertext. This maintains confidentiality. Further, Q can be sure that P has sent the message. This is because other than Q, P is the only party who possesses the key K and thus the ciphertext can be decrypted only by Q and no one else. This maintains authenticity. At a very basic level, symmetric encryption looks like this:



.

MACS BASED ON BLOCK CIPHERS: DAA AND CMAC

In this section, we look at two MACs that are based on the use of a block cipher mode of operation. We begin with an older algorithm, the Data Authentication Algorithm (DAA), which is now obsolete. Then we examine CMAC, which is designed to overcome the deficiencies of DAA.

Data Authentication Algorithm

The **Data Authentication Algorithm** (DAA), based on DES, has been one of the most widely used MACs for a number of years. The algorithm is both a FIPS publication (FIPS PUB 113) and an ANSI standard (X9.17). However, as we discuss subsequently, security weaknesses in this algorithm have been discovered, and it is being replaced by newer and stronger algorithms.

The algorithm can be defined as using the cipher block chaining (CBC) mode of operation of DES (Figure 6.4) with an initialization vector of zero. The data (e.g., message, record, file, or program) to be authenticated are grouped into contiguous 64-bit blocks: $D_1, D_2, \ldots, D_N$. If necessary, the final block is padded on the right with zeroes to form a full 64-bit block. Using the DES encryption algorithm E and a secret key K, a data authentication code (DAC) is calculated as follows (Figure 12.7).

$$
\begin{aligned}
O_1 &= \mathrm{E}(K, D) \\
O_2 &= \mathrm{E}(K, [D_2 \oplus O_1]) \\
O_3 &= \mathrm{E}(K, [D_3 \oplus O_2]) \\
&\bullet \\
&\bullet \\
&\bullet \\
O_N &= \mathrm{E}(K, [D_N \oplus O_{N-1}])
\end{aligned}
$$

ElGamal Encryption Algorithm

S

SamanvayaPanda

- Read
- Discuss

ElGamal encryption is a public-key cryptosystem. It uses asymmetric key encryption for communicating between two parties and encrypting the message. This cryptosystem is based on the difficulty of finding **discrete logarithm** in a cyclic group that is even if we know g^a and g^k, it is extremely difficult to compute g^{ak}.

Idea of ElGamal cryptosystem:

Suppose Alice wants to communicate with Bob.

1. Bob generates public and private keys:
   - Bob chooses a very large number **q** and a cyclic group $\mathbf{F}_q$.
   - From the cyclic group $\mathbf{F}_q$, he choose any element **g** and an element **a** such that gcd(a, q) = 1.
   - Then he computes $h = g^a$.
   - Bob publishes **F**, $\mathbf{h = g^a}$, **q**, and **g** as his public key and retains **a** as private key.
2. Alice encrypts data using Bob's public key :
   - Alice selects an element **k** from cyclic group **F** such that gcd(k, q) = 1.
   - Then she computes $p = g^k$ and $s = h^k = g^{ak}$.
   - She multiples s with M.
   - Then she sends $(p, M*s) = (g^k, M*s)$.
3. Bob decrypts the message :
   - Bob calculates $s' = p^a = g^{ak}$.
   - He divides M*s by s′ to obtain M as s = s′.

Schnorr Digital Signature

C

ciberexplosion

- Read
- Discuss

In cryptography, a **Schnorr signature** is a digital signature produced by the Schnorr signature algorithm that was described by Claus Schnorr. It is a digital signature scheme known for its simplicity, is efficient and generates short signatures. It is one of the protocols used to implement "Proof Of Knowledge".In cryptography, a proof of knowledge is an interactive proof in which the prover succeeds in 'convincing' a verifier that the prover knows something 'X'. For a machine to know 'X' is defined in terms of computation. A machine knows 'X' if this 'X' can be computed. The Verifier either accepts or rejects the proof. The signature proof is supposed to convince the Verifier that they are communicating with a user who knows the private key corresponding to the public key. In other words, the Verifier should be convinced that they are communicating with the Prover without knowing the private key. **Schnorr Digital Signature to implement Zero Knowledge Proof :** Let's take an example of two friends Sachin and Sanchita. Sanchita has announced to the world that she has a public key and can accept and receive information through it. Sachin thinks that Sanchita is lying. Sanchita wants to prove her honesty without showing her private keys. Here is where Schnorr's protocol will help us.

Consider the following parameters:

p, q, a, s, v, r, x, y

where,

"p" is any prime number

"q" is factor of p-1

"a" such that a^q = 1 mod p

The above three variables are global and public which means anyone can see these three variables at a given scenario. We will have two keys.

"s" is the secret key or the private key (0<s<q).

"v" is the public key = a^-s mod q.

The public key "v" will be global and public knowledge along with p, q and a. However only Sanchita will have the knowledge of the private key "s". Now Sanchita signs wants to sends an encrypted message "M". She will follow the following steps to use Schnorr's signature:-

1. She will first choose a random number "r" such that 0<r<q.
2. She will now compute a value X such that: X= a^r mod p.
3. Now that she has computed the value of X, she is going concatenate this with the original message (same as string concatenation). So, she is going to concatenate M and X to get M||X. and she is going to store the hash of this value in e.

e = H(M||X) where H() is the hash function

1. She is going to get a value “y” such that:

```
y = (r + s*e) mod q
```

Now that all the computations are over, she is going to send the following to Sachin.

1. The message “M”.
2. The signatures e and y.

Along with this, Sachin has the following public piece of information:-

1. Sanchita’s public key “v”.
2. The prime number that Sanchita choose “p”.
3. “q” which is the factor of “p-1” which Sanchita choose.
4. “a” such that a^q = 1 mod p, chosen by Sanchita.

Now, Sachin will have to compute X’ such that:

```
X’ = a^y * v^e mod p
```

We know that v = a^-s, let’s substitute that in the equation above and we get:

```
X’ = a^y * a^-se = a ^ (y-s*e)
```

Now we also know that,

```
y = r + s*e
```

```
Which means:
```

```
r = y-s*e
```

Let’s substitute this value in the equation above:

```
We get: X’ = a^r
```

As we have already seen above:

```
X= a^r
```

So technically:

```
X = X’
```

But Sachin doesn’t know the value of “X” because he never received that value. All that he received are the following: The message M, the signatures (e and y) and the host of public variables (public key “v”, p, q, and a). So he is going to solve for e by doing the following:

```
e = H ( M||X’)
```

Note that earlier we solved for e by doing:

```
H(M||X))
```

So, by that logic, if the two values of e come up to be the same then that means

X = X’

This follows all three Properties of Zero Knowledge Proof :

1. **Completeness –** Sachin was convinced of Sanchita’s honesty because at the end X = X’.
2. **Soundness –** The plan was sound because Sanchita only had one way to prove her honesty and that was through her private key.
3. **Zero Knowledge –** Sachin never got to know about Sanchita’s private key.

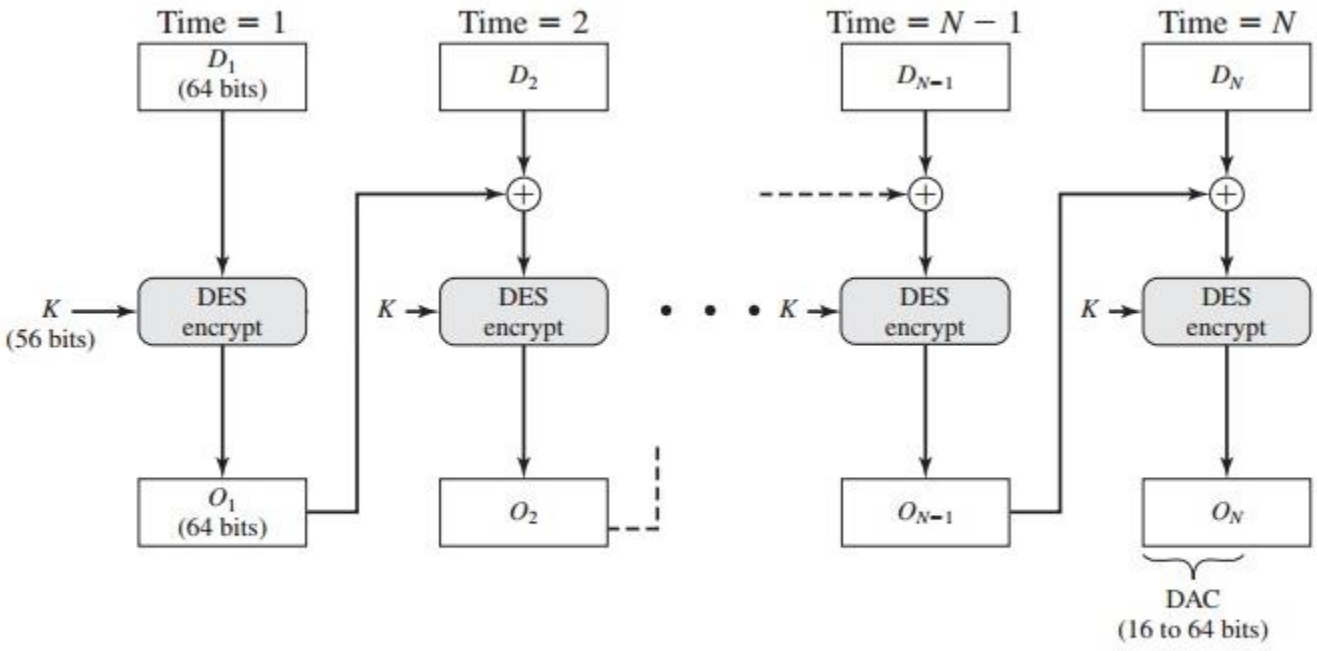


Figure 12.7 Data Authentication Algorithm (FIPS PUB 113)

The DAC consists of either the entire block O_N or the leftmost M bits of the block, with 16 <= M <= 64.

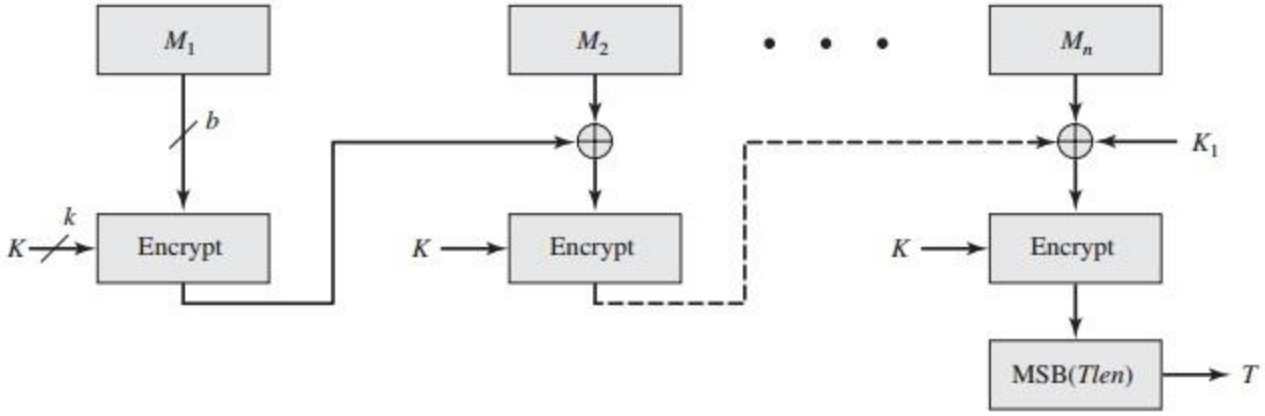
Cipher-Based Message Authentication Code (CMAC)

As was mentioned, DAA has been widely adopted in government and industry. [BELL00] demonstrated that this MAC is secure under a reasonable set of security criteria, with the following restriction. Only messages of one fixed length of *mn* bits

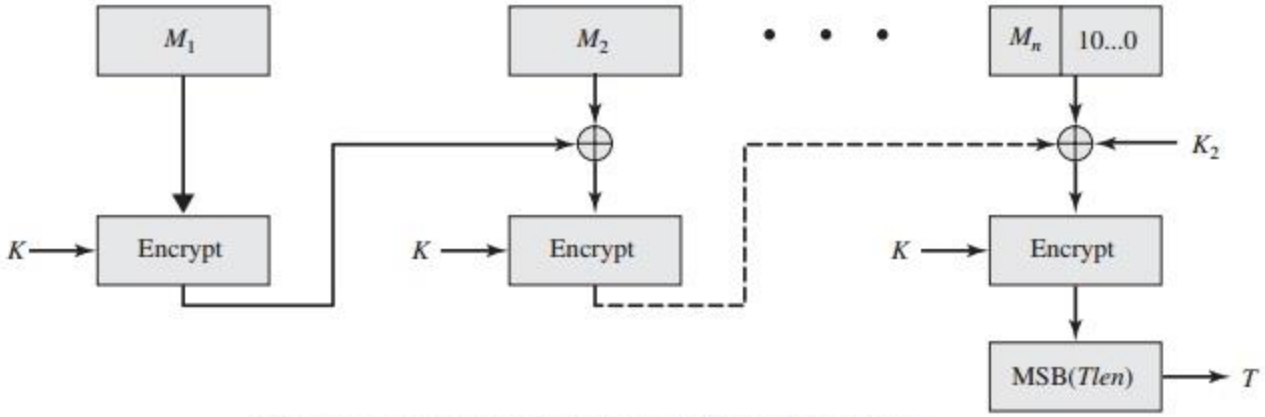
are processed, where n is the cipher block size and m is a fixed positive integer. As a simple example, notice that given the CBC MAC of a one-block message X, say $T = \text{MAC}(K, X)$, the adversary immediately knows the CBC MAC for the two- block message $X \| (X \{ T)$ since this is once again T.

Black and Rogaway [BLAC00] demonstrated that this limitation could be overcome using three keys: one key of length K to be used at each step of the cipher block chaining and two keys of length n, where k is the key length and n is the cipher block length. This proposed construction was refined by Iwata and Kurosawa so that the two n-bit keys could be derived from the encryption key, rather than being provided separately [IWAT03]. This refinement, adopted by NIST, is the **Cipher-based Message Authentication Code** (CMAC) mode of oper- ation for use with AES and triple DES. It is specified in NIST Special Publication 800-38B.

First, let us define the operation of CMAC when the message is an integer multiple n of the cipher block length b. For AES, $b = 128$, and for triple DES, $b = 64$. The message is divided into n blocks ($M_1, M_2, \ldots, M_n$). The algorithm makes use of a k-bit encryption key K and an n-bit constant, K_1. For AES, the key size k is 128, 192, or 256 bits; for triple DES, the key size is 112 or 168 bits. CMAC is calculated as follows (Figure 12.8).



(a) Message length is integer multiple of block size



(b) Message length is not integer multiple of block size

Figure 12.8 Cipher-Based Message Authentication Code (CMAC)

$$
\begin{aligned}
C_1 &= \mathrm{E}(K, M_1) \\
C_2 &= \mathrm{E}(K, [M_2 \oplus C_1]) \\
C_3 &= \mathrm{E}(K, [M_3 \oplus C_2]) \\
&\bullet \\
&\bullet \\
&\bullet \\
C_n &= \mathrm{E}(K, [M_n \oplus C_{n-1} \oplus K_1]) \\
T &= \mathrm{MSB}_{Tlen}(C_n)
\end{aligned}
$$

where

T = message authetication code, also referred to as the tag
$Tlen$ = bit length of T
$\mathrm{MSB}_s(X)$ = the s leftmost bits of the bit string X

If the message is not an integer multiple of the cipher block length, then the final block is padded to the right (least significant bits) with a 1 and as many 0s as necessary so that the final block is also of length b. The CMAC operation then proceeds as before, except that a different n-bit key K_2 is used instead of K_1.

The two n-bit keys are derived from the k-bit encryption key as follows.

$$
\begin{aligned}
L &= \mathrm{E}(K, 0^n) \\
K_1 &= L \bullet x \\
K_2 &= L \bullet x^2 = (L \bullet x) \bullet x
\end{aligned}
$$

where multiplication ($\bullet$) is done in the finite field GF(2_n) and x and x_2 are first- and

second-order polynomials that are elements of GF(2_n). Thus, the binary representation of x consists of n - 2 zeros followed by 10; the binary representation of x_2 consists of n - 3 zeros followed by 100. The finite field is defined with respect to an irreducible polynomial that is lexicographically first among all such polynomials with the minimum possible number of nonzero terms. For the two approved block sizes, the polynomials are $x_{64} + x_4 + x_3 + x + 1$ and $x_{128} + x_7 + x_2 + x + 1$.

To generate K_1 and K_2, the block cipher is applied to the block that consists entirely of 0 bits. The first subkey is derived from the resulting ciphertext by a left shift of one bit and, conditionally, by XORing a constant that depends on the block size. The second subkey is derived in the same manner from the

first subkey. This property of finite
fields of the form GF(2^n) was explained in the discussion of MixColumns in Chapter 5.