

Module 20

Cryptography

The screenshot shows a page titled "Cryptography" with a sub-header "Module Objectives". On the left side, there is a graphic with the text "Module Objectives" and a small blue square icon. The main content area lists the following objectives:

- Understanding Cryptography Concepts
- Overview of Encryption Algorithms
- Cryptography Tools
- Understanding Public Key Infrastructure (PKI)
- Understanding Email Encryption
- Understanding Disk Encryption
- Understanding Cryptography Attacks
- Cryptanalysis Tools

At the bottom of the screenshot, there is a small copyright notice: "Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited."

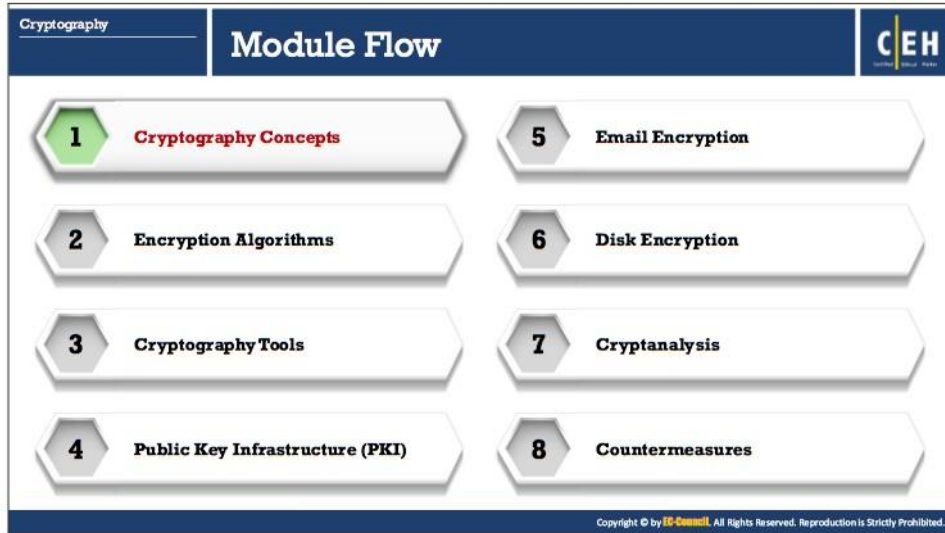
Module Objectives

With the increasing adoption of Internet–World Wide Web for business and personal communication, securing sensitive information such as credit-card and PINs, bank account numbers, and private messages is becoming increasingly important and yet more difficult to achieve. Today’s information-based organizations extensively use the Internet for e-commerce, market research, customer support, and a variety of other activities. Data security is critical to online business and privacy of communication.

Cryptography and cryptographic (“crypto”) systems help in securing data from interception and compromise during online transmissions. This module provides a comprehensive understanding of different crypto systems and algorithms, one-way hash functions, public-key Infrastructures (PKIs), and the different ways cryptography can help in ensuring privacy and security of online communication. It also covers various tools used to encrypt sensitive data.

At the end of this module, you will be able to:

- Describe cryptography concepts
- Understand different encryption algorithms
- Use different cryptography tools
- Describe about Public Key Infrastructure (PKI)
- Apply email encryption
- Apply disk encryption
- Describe about various cryptography attacks
- Use different cryptanalysis tools



Cryptography Concepts

Cryptography enables one to secure transactions, communications, and other processes performed in the electronic world. This section deals with cryptography and its associated concepts with which one should be familiar to understand the advanced topics, covered later in this module.

Cryptography
Cryptography Concepts

Cryptography

CEH

Types of Cryptography

- Cryptography is the **conversion of data** into a scrambled code that is encrypted and sent across a private or public network
- Cryptography is used to protect confidential data such as **email messages**, chat sessions, **web transactions**, personal data, **corporate data**, e-commerce applications, etc.

Objectives of Cryptography

- Confidentiality
- Integrity
- Authentication
- Nonrepudiation

Symmetric Encryption

Symmetric encryption (secret-key, shared-key, and private-key) **uses the same key** for encryption as it does for decryption

Asymmetric Encryption

Asymmetric encryption (public-key) **uses different encryption keys** for encryption and decryption. These keys are known as public and private keys

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Cryptography

“Cryptography” comes from the Greek words **kryptos**, meaning “concealed, hidden, veiled, secret, or mysterious,” and **graphia**, “writing”; thus, cryptography is “the art of secret writing.”

Cryptography is the practice of concealing information by converting plain text (readable format) into cipher text (unreadable format) using a key or encryption scheme. It is the process of conversion of data into a scrambled code that is encrypted and sent across a private or public network. Cryptography protects confidential data such as email messages, chat sessions, web transactions, personal data, corporate data, e-commerce applications, and many other kinds of communication. Encrypted messages can at times be decrypted by cryptanalysis (code breaking), even though modern encryption techniques are virtually unbreakable.

Objectives of Cryptography

- **Confidentiality** - Assurance that the information is accessible only to those authorized to have access.
- **Integrity** - The trustworthiness of data or resources in terms of preventing improper and unauthorized changes.
- **Authentication** - Refers to the characteristic of a communication, document, or any data that ensures the quality of being genuine.
- **Nonrepudiation** - Guarantee that the sender of a message cannot later deny having sent the message, and that the recipient cannot deny having received the message.

Cryptography Process:

Plain text (readable format) is encrypted by means of encryption algorithms such as RSA, DES, and AES, resulting in a cipher text (unreadable format) that, on reaching the destination, is decrypted into readable plain text.



FIGURE 20.1: Example of Cryptography

Types of Cryptography

Cryptography is of two types, according to the number of keys employed for encryption and decryption:

- **Symmetric Encryption**

Symmetric encryption requires that both the sender and the receiver of the message possess the same encryption key. The sender uses a key to encrypt the plaintext and sends the resultant cipher text to the recipient, who uses the same key (used for encryption) to decrypt the cipher text into plain text. Symmetric encryption is also known as secret key cryptography as it uses only one secret key to encrypt and decrypt the data. This kind of cryptography works well when you are communicating with up to only a few people.

Because the sender and receiver must share the key prior to sending any messages, this technique is of limited use for the Internet, where individuals who have not had prior contact frequently require a secure means of communication. The solution to this problem is asymmetric encryption (public-key cryptography).

- **Asymmetric Encryption**

The introduction of the concept of asymmetric encryption (also known as public-key cryptography) was to solve key-management problems. Asymmetric encryption involves both a public key and a private key. The public key is publicly available, but the sender keeps the private key as a secret.

An asymmetric key system is an encryption method using a key pair, one public key available to anyone, and one private key held only by the key owner, that helps to provide confidentiality, integrity, authentication, and nonrepudiation in data management.

Asymmetric encryption uses the following sequence to send a message:

1. An individual finds the public key of the person he or she wants to contact in a directory.
2. This public key is used to encrypt a message that is then sent to the intended recipient.
3. The receiver uses the private key to decrypt the message and reads it.

No one but the holder of the private key can decrypt a message composed with the corresponding public key. This increases the security of the information because all communications involve only public keys; the message sender never transmits or shares the private keys. The sender must link the public keys with the usernames in a secured method to ensure that individuals claiming to be the intended recipient do not intercept information. To meet the need for authentication, one can use digital signatures.

Strengths and Weaknesses of Crypto Methods

	Symmetric Encryption	Asymmetric Encryption
Strengths	Faster and easier to implement as same key is used to encrypt and decrypt data and also requires less processing power. Could be implemented in Application Specific Integrated Chip (ASIC).	Convenient to use as distribution of keys to encrypt the messages is not required
	Prevents widespread message security compromise as different secret key is used to communicate with different party	Enhanced security as one need not share or transmit private keys to anyone
	Key is not bound to the data being transferred on the link; therefore, even if data is intercepted it is not possible to decrypt it	Provides digital signatures that can't be repudiated
	Symmetric Encryption	Asymmetric Encryption
Weaknesses	Lack of secure channel to exchange secret key	Slow in processing and requires high processing power
	Difficult to manage and secure too many shared keys that are generated to communicate with different parties	Widespread message security compromise is possible (i.e., attacker can read his/her complete messages if private key is compromised)
	Provides no assurance about origin and authenticity of a message as same key is used by both sender and receiver	Messages received cannot be decrypted if the private key is lost
	Vulnerable to dictionary attacks and brute-force attacks	Vulnerable to Man-in-the-Middle and brute-force attacks

TABLE 20.1: Strengths and weaknesses of crypto methods

Cryptography
Cryptography Concepts

Government Access to Keys (GAK)

- GAK means that software companies will give **copies of all keys** (or at least enough of the key that the remainder could be cracked) to the government
- The government promises that they will hold on to the keys in a **secure manner** and will only use them when a **court issues a warrant** to do so
- To the government, this issue is similar to the **ability to wiretap phones**

```
graph TD; CK[Cryptographic Key] --> IA[Item A]; CK --> IB[Item B]; CK --> IC[Item C]; CK --> ID[Item D]; CK --> IE[Item E];
```

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Government Access to Keys (GAK)

Government Access to Keys (GAK) refers to statutory obligation of individuals and organizations to disclose their cryptographic keys to government agencies. It means that software companies will give copies of all keys, (or at least enough of the key that the remainder could be cracked) to the government. Law enforcement agencies around the world acquire and use these cryptographic keys to monitor suspicious communication, and collect evidence of cybercrimes in the interest of national security. The government promises that they will hold on to the keys in a secure way, and will only use them when a court issues a warrant to do so. To the government, this issue is similar to the ability to wiretap phones.

Government agencies often use key escrow to have uninterrupted access to keys. Key escrow is a key exchange arrangement in which essential cryptographic keys are stored with a third party in escrow. The third party can use or allow others to use the encryption keys under certain predefined circumstances. The third party, with reference to GAK, is generally a government agency that may use the encryption keys to decipher digital evidence using authorization or a warrant from a court of law. However, there is a growing concern about privacy and security of cryptographic keys and information. Government agencies are responsible for protecting the keys. These agencies generally use a single key to protect other keys, which is not a good idea, as revealing a single key could cause exposure of other keys.

These agencies are not aware just how confidential the information protected by the keys is, which makes it difficult to judge how much protection is required. In cases where seized keys also protect other information to which these agencies do not have the right to access, consequences of key revelation cannot be determined, because government agencies are not aware of the information the keys protect. In such cases, the key owner is liable for the consequences of key revelation. Before owner's hand over their keys to government agencies,

they need to be assured that government agencies will protect these keys according to a standard that is sufficient to protect their interests.

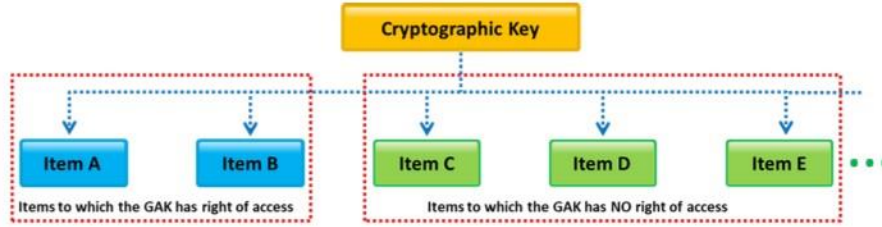
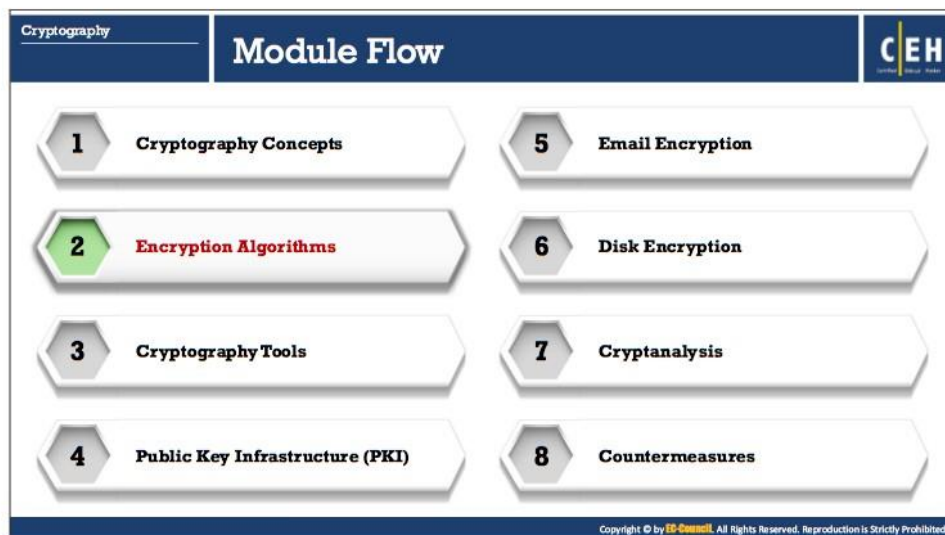
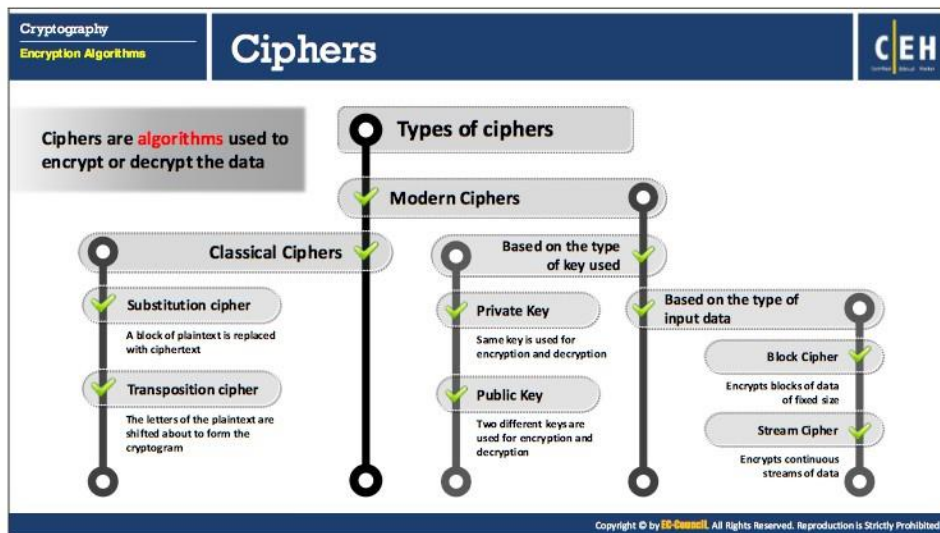


FIGURE 20.2: Illustration of GAK



Encryption Algorithms

Encryption is the process of converting readable plain text into an unreadable cipher text by applying a set of complex algorithms that transform the data into blocks or streams of random alphanumeric characters. This section deals with ciphers and various encryption algorithms such as DES, AES, RC4, RC5, RC6, DSA, RSA, MD5, and SHA.



Ciphers

In cryptography, a cipher is an algorithm (a series of well-defined steps) for performing encryption and decryption. Encipherment is the process of converting plain text into a cipher or code; the reverse process is called decipherment. A message is encrypted using a cipher is rendered unreadable, unless its recipient knows the secret key required to decrypt it. Communication technologies (e.g., Internet, cell phones) rely on ciphers to maintain both security and privacy. The cipher algorithms may be open sourced (the algorithmic process is in public domain while key is selected by a user and is private) or close sourced (the process is developed for use in specific domain like military and the algorithm itself is not in public domain). Further, the ciphers could be either free for public use or licensed.

Mainly, ciphers are of two types: classical and modern.

- **Classical Ciphers**

Classical ciphers are the most basic type of ciphers, which operate on alphabets (A-Z). Implementation of these ciphers is generally either by hand or with simple mechanical devices. Because these ciphers are easily deciphered, they are generally unreliable.

Types of classical ciphers:

- **Substitution cipher:** The user replaces units of plaintext with ciphertext, according to a regular system. Units may be single letters, pairs of letters, or combinations of them, and so forth. The recipient performs inverse substitution to decipher the text. Examples include Beale cipher, autokey cipher, Gronsfeld cipher, and Hill cipher.

For example, "HELLO WORLD" can be encrypted as "PSTER HGFST" (i.e., H=P, E=S, etc.).
- **Transposition cipher:** Here, rearranging letters in the plain text, according to a regular system produces the cipher text. For example, "CRYPTOGRAPHY" when encrypted becomes "AOYCRGPTYRHP." Examples include Rail Fence Cipher, Route cipher, and Myszkowski transposition.

- **Modern Ciphers**

Design of modern ciphers helps to withstand a wide range of attacks. Modern ciphers provide message secrecy, integrity, and authentication of the sender. The user can calculate the modern ciphers with the help of a one-way mathematical function that is capable of factoring large prime numbers.

Types of Modern ciphers:

- **Based on the type of key used**
 - **Symmetric key algorithms (Private-key cryptography):** Uses same key for encryption and decryption.
 - **Asymmetric key algorithms (Public-key cryptography):** Uses two different keys for encryption and decryption.
- **Based on the type of input data**
 - **Block ciphers:** Deterministic algorithm operating on block (group of bits) of fixed size with an unvarying transformation specified by a symmetric key. Most modern ciphers are block ciphers. These are widely used to encrypt bulk data. Examples include DES, AES, IDEA, etc. When the block size is less than that used by the ciphers, padding is employed to achieve fixed block size.
 - **Stream ciphers:** Symmetric key ciphers are plaintext digits combined with a key stream (pseudorandom cipher digit stream). Here, the user applies the key to each bit, one at a time. Examples include RC4, SEAL, etc.

The infographic is titled "Data Encryption Standard (DES)" and is part of a "Cryptography" series focusing on "Encryption Algorithms". It features the CEH logo in the top right corner. The content is organized into three horizontal sections, each with a text box on the left and an icon on the right:

- Section 1:** Text: "DES is designed to **encipher** and **decipher** blocks of data consisting of **64 bits** under control of a 56-bit key." Icon: A laptop and a mobile phone.
- Section 2:** Text: "DES is the **archetypal block cipher**—an algorithm that takes a fixed-length string of plaintext bits and transforms it into a ciphertext bitstring of the same length." Icon: A padlock with a keyhole.
- Section 3:** Text: "Due to the **inherent weakness** of DES with today's technologies, some organizations repeat the process thrice(3DES) for added strength, until they can afford to update their equipment to AES capabilities." Icon: A calendar icon.

At the bottom of the infographic, there is a small copyright notice: "Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited."

Data Encryption Standard (DES)

DES is a standard for data encryption that uses a secret key for both encryption and decryption (symmetric cryptosystem). DES uses a 64-bit secret key of which 56 bits are generated randomly and other 8 bits help in error detection. It uses the Data Encryption Algorithm (DEA), a secret key block-cipher employing a 56-bit key operating on 64-bit blocks. DES is the archetypal block cipher — an algorithm that takes a fixed-length string of plaintext bits and transforms it into a ciphertext bit string of the same length. DES's design allows users to implement it in hardware and use for single-user encryption, such as to store files on a hard disk in encrypted form.

DES provides 72 quadrillion or more possible encryption keys and chooses a random key for encryption of each message. Because of the inherent weakness of DES vis-à-vis today's technologies, some organizations use triple DES, in which they repeat the process three times (3DES) for added strength.


Triple Data Encryption Standard (3DES)

Eventually it became obvious that DES would no longer be secure. The U.S. federal government began a contest seeking a replacement cryptography algorithm. However, in the meantime, 3DES was created as an interim solution. Essentially, it does DES three times with three different keys. 3DES uses a "key bundle" which comprises three DES keys, K1, K2, and K3. Each key is standard 56-bit DES key. It will then apply the following process:

DES *encrypt* with K1, DES *decrypt* with K2, then DES *encrypt* with K3. There are three options for the keys. In the first option, all three keys are independent and different. In the second option, K1 and K3 are identical. In the third option, all three keys are the same. Therefore, you are literally applying the exact same DES algorithm three times with the same key. Option 1 is the most secure, with option 3 being the least secure.

Cryptography
Encryption Algorithms

Advanced Encryption Standard (AES)



- | AES is a **symmetric-key** algorithm that secures sensitive but unclassified material by the US government agencies

- | AES is an **iterated block cipher**, which works by repeating the same operation **multiple** times

- | It has a **128-bit** block size, with key sizes of 128, 192, and 256 bits, respectively, for AES-128, AES-192, and AES-256

AES Pseudocode

```
Cipher (byte in[4*Nb], byte out[4*Nb], word
w[Nb*(Nr+1)])
begin
  byte state[4, Nb]
  state = in
  AddRoundKey (state, w)
  for round = 1 step 1 to Nr-1
    SubBytes (state)
    ShiftRows (state)
    MixColumns (state)
    AddRoundKey (state, w+round*Nb)
  end for
  SubBytes (state)
  ShiftRows (state)
  AddRoundKey (state, w+Nr*Nb)
  out = state
end
```

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Advanced Encryption Standard (AES)

The Advanced Encryption Standard (AES) is a **National Institute of Standards and Technology (NIST)** specification for the encryption of electronic data. It also helps to encrypt digital information such as telecommunications, financial, and government data. US government agencies have been using it to secure sensitive but unclassified material.

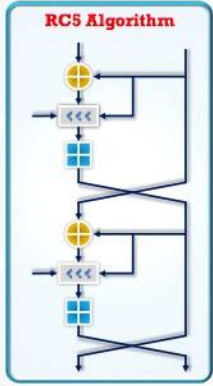
AES consists of a symmetric-key algorithm: both encryption and decryption are performed using the same key. It is an iterated block cipher that works by repeating the defined steps multiple times. It has a 128-bit block size, with key sizes of 128, 192, and 256 bits, respectively, for AES-128, AES-192, and AES-256. The design of AES makes its use efficient in both software and hardware. It works simultaneously at multiple network layers.

AES Pseudocode

Initially, the system copies the cipher input into the internal state and then adds an initial round key. The system transforms the state by iterating a round function in a number of cycles. Depending on the block size and key length, the number of cycles may vary. After completing rounding, the system copies the final state into the cipher output.

```
Cipher (byte in [4*Nb], byte out [4*Nb], word w[Nb*(Nr+1)])
begin
  byte state[4, Nb]
  state = in
  AddRoundKey (state, w)
  for round = 1 step 1 to Nr-1
    SubBytes (state)
```

```
        ShiftRows (state)
        MixColumns (state)
        AddRoundKey (state, w+round*Nb)
    end for
    SubBytes (state)
    ShiftRows (state)
    AddRoundKey (state, w+Nr*Nb)
    out = state
end
```

Cryptography Encryption Algorithms		RC4, RC5, and RC6 Algorithms	CEH
RC4	<ul style="list-style-type: none">A variable key size symmetric key stream cipher with byte-oriented operations and is based on the use of a random permutation		
RC5	<ul style="list-style-type: none">It is a parameterized algorithm with a variable block size, a variable key size, and a variable number of rounds. The key size is 128-bits		
RC6	<ul style="list-style-type: none">RC6 is a symmetric key block cipher derived from RC5 with two additional features:<ul style="list-style-type: none">Uses integer multiplicationUses four 4-bit working registers (RC5 uses two 2-bit registers)		

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

RC4, RC5, RC6 Algorithms

Discussed below are symmetric encryption algorithms, developed by RSA Security:

- **RC4**

RC4 is a variable key-size symmetric-key stream cipher with byte-oriented operations that depends on the use of a random permutation. According to some analyses, the period of the cipher is likely to be greater than 10,100. Each output byte uses eight to sixteen system operations, meaning the cipher has the ability to run fast when used in software. Products like RSA SecurPC use this algorithm for file encryption. RC4 enables safe communications such as traffic encryption (which secures Web sites) and for Web sites that use the SSL protocol.

- **RC5**

RC5 is a fast symmetric-key block cipher designed by Ronald Rivest for RSA Data Security (now RSA security). The algorithm is a parameterized algorithm with a variable block size, variable key size, and a variable number of rounds. The block sizes can be 32, 64, or 128 bits. The range of the rounds can vary from 0 to 255, and the size of the key can vary from 0 to 2,040 bits. This built in variability can offer flexibility at all levels of security. Routines used in RC5 are key expansion, encryption, and decryption.

In the key expansion routine, the secret key that a user provides is expanded to fill the key table (the size of which depends on the number of rounds). The RC5 uses key table for both encryption and decryption. The encryption routine has three fundamental operations: integer addition, bitwise XOR, and variable rotation. The intense use of data-dependent rotation, plus the combination of different operations, makes RC5 a secure encryption algorithm.

- **RC6**

RC6 is a symmetric key block cipher derived from RC5. It is a parameterized algorithm with a variable block size, key size, and number of rounds. Two features that differentiate the RC6 algorithm from RC5 are integer multiplication (which is used to increase the diffusion achieved in fewer rounds and increased speed of the cipher), and the use of four 4-bit working registers rather than two 2-bit registers. The RC6 algorithm uses four 4-bit registers in place of the two 2-bit registers because the block size of the AES is 128 bits.

Blowfish

Blowfish is a type of symmetric block cipher algorithm, designed to replace DES or IDEA algorithms. It uses a same secret key to encrypt and decrypt data. This algorithm splits the data into a block length of 64-bit size and produces a key ranging from length 32 bits to 448 bits. Due to the high speed and overall efficiency, blowfish is used in software ranging from categories such as password protection tools to e-commerce websites for securing payments.

It is a 16-round Feistel cipher working on 64-bit blocks. However, unlike DES, it can have varying key sizes ranging from 32 bits to 448 bits.

There are really two parts to this algorithm. The first part handles the expansion of the key. The second part actually encrypts the data.

The key expansion is handled in several steps. The first step is to break the original key into a set of subkeys. Specifically, a key of no more than 448 bits is separated into 4,168 bytes. There is a P-array and four 32-bit S-boxes. The P-array contains 18 32-bit subkeys, while each S-box contains 256 entries.

Key expansion is done as follows:

1. The first step in the key expansion phase is to initialize the **P-array** and **S-boxes**.
2. Then, XOR P-array with the key bits. For example, **P1 XOR** (first 32 bits of the key), **P2 XOR** (second 32 bits of the key).
3. Use the above method to encrypt the **all-zero string**.
4. This new output is now P1 and P2.
5. Encrypt the new P1 and P2 with the **modified subkeys**.
6. This new output is now P3 and P4.
7. Repeat **521 times** in order to calculate new subkeys for the P-array and the four S-boxes.

The round function splits the 32-bit input into four eight-bit quarters, and uses the quarters as input to the S-boxes. The outputs are added modulo **232** and **XORed** to produce the final **32-bit output**.

Cryptography
Encryption Algorithms

Twofish

CEH

- This algorithm was one of the five finalists to **replace DES** for the **US Government**, but it was not chosen 
- It uses a block size of 128 bits and key sizes up to 256 bits. It is a **Feistel cipher** 
- It was designed by **Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, and Niels Ferguson** 

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.


Twofish

Twofish algorithm was one of the five finalists to replace DES for the U.S. Government, but was not chosen. It was designed by Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, and Niels Ferguson.

TwoFish is a 128-bit block cipher. It is one of the most conceptually simple algorithm that uses a single key for both encryption and decryption for any length up to 256 bits. It is a Feistel cipher. It not only works fast for CPU or hardware in fact it is flexible with network based applications also. It even enables various levels of performance trade-off on parameters like encryption speed, hardware gate count, memory usage, etc. This technique of enabling different implementations improves the relative performance of the algorithm. Any user can optimize the performance based on the different key scheduling.

Cryptography
Encryption Algorithms

The DSA and Related Signature Schemes



Digital Signature Algorithm


FIPS 186-2 specifies the Digital Signature Algorithm (DSA) that may be used in the **generation and verification of digital signatures** for sensitive, unclassified applications

Digital Signature

The digital signature is **computed using a set of rules** (i.e., the DSA) **and a set of parameters** such that the identity of the signatory and integrity of the data can be verified

Each entity creates a public key and a corresponding private key

1. Select a prime number q such that $2^{159} < q < 2^{160}$
2. Choose t so that $0 \leq t \leq 8$
3. Select a prime number p such that $2^{511+64t} < p < 2^{512+64t}$ with the additional property that q divides $(p-1)$
4. Select a generator α of the unique cyclic group of order q in Z_p^*
5. To compute α , select an element g in Z_p^* and compute $g^{(p-1)/q} \bmod p$
6. If $\alpha = 1$, perform step five again with a different g
7. Select a random a such that $1 \leq a \leq q-1$
8. Compute $y = \alpha^a \bmod p$



Following are the public keys: p, q, α, y and a is the private key.

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

The DSA and Related Signature Schemes

The Digital Signature Algorithm (DSA) is a Federal Information Processing Standard for digital signatures. The NIST proposed the DSA for use in the Digital Signature Standard (DSS), adopted as FIPS 186. The DSA helps in the generation and verification of digital signatures for sensitive and unclassified applications. It creates a 320-bit digital signature but with 512–1024 bit security.

Digital signature is a mathematical scheme used for the authentication of digital messages. Computation of the digital signature uses a set of rules (i.e., the DSA) and a set of parameters, in that the user can verify the identity of the signatory and integrity of the data.

Processes involved in DSA:

- **Signature Generation Process:** The private key is used to know who has signed it.
- **Signature Verification Process:** The public key is used to verify whether the given digital signature is genuine.

DSA is a public-key crypto system as it involves the use of both private and public keys.

Benefits of DSA:

- Less chances of forgery than with a written signature
- Quick and easy method of business transactions
- Fake currency problem can be drastically reduced

Given below is the DSA Algorithm:

Each entity A does the following:

1. Select a prime number q such that $2^{159} < q < 2^{160}$
2. Choose t so that $0 \leq t \leq 8$, and select a prime number p where $2^{511+64t} < p < 2^{512+64t}$, with the property that q divides $(p-1)$
3. Select a generator α of the unique cyclic group of order q in \mathbb{Z}_p^* by choosing an element $g \in \mathbb{Z}_p^*$ and then computing $\alpha = g^{(p-1)/q} \bmod p$ until $\alpha \neq 1$
4. Select a random integer d such that $1 \leq d \leq q-1$
5. Compute $y = \alpha^d \bmod p$
6. A's public key is (p, q, α, y) ; A's private key is d .

To sign a message m , A does the following:

1. Select a random secret integer k , $0 < k < q$.
2. Compute $r = (\alpha^k \bmod p) \bmod q$
3. Compute $k^{-1} \bmod q$
4. Compute $s = k^{-1} \{h(m) + dr\} \bmod q$, where h is the Secure Hash Algorithm
5. A's signature for m is the pair (r, s)


To verify A's signature (r, s) on m , B should do the following:


1. Obtain A's authentic public key (p, q, α, y)
2. Verify that $0 < r < q$ and $0 < s < q$; if not, then reject the signature
3. Compute $w = s^{-1} \bmod q$ and $h(m)$
4. Compute $u_1 = w \cdot h(m) \bmod q$ and $u_2 = rw \bmod q$
5. Compute $v = (\alpha^{u_1} y^{u_2} \bmod p) \bmod q$
6. Accept the signature if and only if $v=r$


Cryptography
Encryption Algorithms

Rivest Shamir Adleman (RSA)

CEH

 RSA is an **Internet encryption and authentication system** that uses an algorithm developed by Ron Rivest, Adi Shamir, and Leonard Adleman

 It is widely used and is one of the **de-facto encryption standard**

 It uses **modular arithmetic** and **elementary number theories** to perform computations using two large prime numbers

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Cryptography
Encryption Algorithms

Rivest Shamir Adleman (RSA) (Cont'd)

CEH

The RSA Signature Scheme

Algorithm Key generation for the RSA signature scheme

SUMMARY: each entity creates an RSA public key and a corresponding private key. Each entity A should do the following:

1. Generate two large distinct random primes p and q , each roughly the same size.
2. Compute $n = pq$ and $\phi = (p-1)(q-1)$.
3. Select a random integer e , $1 < e < \phi$, such that $\text{gcd}(e, \phi) = 1$.
4. Use the extended Euclidean algorithm (Algorithm 2.107) to compute the unique integer d , $1 < d < \phi$, such that $ed \equiv 1 \pmod{\phi}$.
5. A's public key is (n, e) ; A's private key is d .

Algorithm RSA signature generation and verification

SUMMARY: entity A signs a message $m \in M$. Any entity B can verify A's signature and recover the message m from the signature.

1. **Signature generation.** Entity A should do the following:

- (a) Compute $dh = R(m)$, an integer in the range $[0, n-1]$.
- (b) Compute $s = dh^e \pmod{n}$.
- (c) A's signature for m is s .

2. **Verification.** To verify A's signature s and recover the message m , B should:

- (a) Obtain A's authentic public key (n, e) .
- (b) Compute $dh = s^e \pmod{n}$.
- (c) Verify that $dh \in M_A$; if not, reject the signature.
- (d) Recover $m = R^{-1}(dh)$.

Example of RSA Algorithm

$P = 61$ \Leftarrow first prime number (destroy this after computing E and D)

$Q = 53$ \Leftarrow second prime number (destroy this after computing E and D)

$PQ = 3233$ \Leftarrow modulus (give this to others)

$E = 17$ \Leftarrow public exponent (give this to others)

$D = 2753$ \Leftarrow private exponent (keep this secret!)

Your **public key** is (E, PQ) .
Your **private key** is D .

The encryption function is: $\text{encrypt}(T) = (T^E) \pmod{PQ}$
 $= (T^{17}) \pmod{3233}$

The decryption function is: $\text{decrypt}(C) = (C^D) \pmod{PQ}$
 $= (C^{2753}) \pmod{3233}$

To encrypt the plaintext value 123, do this:

$\text{encrypt}(123) = (123^{17}) \pmod{3233}$
 $= 337587917446653715596592958817679803 \pmod{3233}$
 $= 855$

To decrypt the cipher text value 855, do this:

$\text{decrypt}(855) = (855^{2753}) \pmod{3233}$
 $= 123$

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Rivest Shamir Adleman (RSA)

Ron Rivest, Adi Shamir, and Leonard Adleman formulated RSA, a public-key cryptosystem for Internet encryption and authentication. RSA uses modular arithmetic and elementary number theories to perform computations using two large prime numbers. The RSA system is widely used in a variety of products, platforms, and industries. It is one of the de-facto encryption standards. Companies such as Microsoft, Apple, Sun, and Novell build the RSA algorithms into

their operating systems. RSA can also be found on hardware secured telephones, Ethernet network cards, and smart cards.

RSA works as follows:

1. Two large prime numbers are taken (a and b), and their product is determined ($c = ab$, where " c " is called the modulus).
2. RSA chooses a number " e " that it is less than " c " and relatively prime to $(a-1)(b-1)$. Therefore, e and $(a-1)(b-1)$ have no common factors except 1.
3. Apart from this, RSA chooses a number " f " such that $(ef - 1)$ is divisible by $(a-1)(b-1)$.
4. The values " e " and " f " are the public and private exponents, respectively.
5. The public key is the pair (c, e) ; the private key is the pair (c, f) .
6. It is difficult to obtain the private key (c, f) from the public key (c, e) . However, if someone can factor " c " into " a " and " b ", then that person can decipher the private key (c, f) .

The security of the RSA system depends on the assumption that such factoring is difficult to carry out, making the cryptographic technique safe.

Following sequence is an example of how cryptography uses RSA algorithms in a practical interchange:

1. The sender of a message encrypts it using a randomly chosen DES symmetric key. DES (Data Encryption Standard) is a relatively insecure symmetric key system using 64-bit encryption (56 bits for key size, 8 bits for cyclic redundancy check) to encrypt data.
2. The sender will then look up the recipient's public key and use it to encrypt the DES key using the RSA system.
3. The sender transmits an RSA digital envelope, consisting of a DES-encrypted message and an RSA-encrypted DES key, to the recipient.
4. The recipient will decrypt the DES key and then use the DES key to decrypt the message itself.

This system combines the high speed of DES with the key management convenience of the RSA system.

The RSA Signature Scheme

Cryptography uses RSA for both public key encryption and for a digital signature (to sign a message and verify it). The RSA signature scheme is the first technique used to generate digital signatures. It is a deterministic digital signature scheme that provides message recovery from the signature itself, making it the most practical and versatile technique available.

RSA involves both a public key and a private key. The public key, as the name indicates, means any person can use it for encrypting messages. The messages that user encrypts with the public key require the private key for decryption.

Consider that John encrypts his document M using his private key S_A , thereby creating a signature $S_{\text{John}}(M)$. John sends M along with the signature $S_{\text{John}}(M)$ to Alice. Alice decrypts the document using Alice's public key, thereby verifying John's signature.

RSA Key Generation

The procedure for RSA key generation is common for all the RSA-based signature schemes. To generate an RSA key pair, i.e., both an RSA public key and corresponding private key, each entity A should do the following:

- Generate two large distinct primes p and q arbitrarily, each roughly the same bit length
- Compute $n = pq$ and $\phi = (p-1)(q-1)$
- Choose a random integer e , $1 < e < \phi$, such that $\text{gcd}(e, \phi) = 1$
GCD = Greatest Common Divisor
- Use the extended Euclidean algorithm in order to compute the unique integer d , $1 < d < \phi$, such that $ed \equiv 1 \pmod{\phi}$
- A 's public key is (n, e) ; A 's private key is d

Destroy p and q at the end of the key generation

RSA algorithm generates and verifies RSA signature in the following way:

Entity A signs a message $m \in M$. Any entity B can verify A 's signature and recover the message m from the signature.

1. Signature Generation

To sign a message m , entity A should do the following:

- Compute $\tilde{m} = R(m)$, an integer in the range $[0, n-1]$
- Compute $s = \tilde{m}^d \pmod{n}$
- A 's signature form is s

2. Signature Verification

To verify A 's signature s and recover the message m , B should do the following:

- Obtain A 's authentic public key (n, e)
- Compute $\tilde{m} = s^e \pmod{n}$
- Verify that $\tilde{m} \in M_R$; if not, reject the signature
- Recover $m = R^{-1}(\tilde{m})$

Example of RSA Algorithm

Given below is the math behind RSA public-key encryption:

1. Find P and Q , two large (e.g., 1024-bit) prime numbers.
2. Choose E such that E is greater than 1, E is less than PQ , and E and $(P-1)(Q-1)$ are relatively prime, which means they have no prime factors in common. E does not have

to be prime, but it must be odd. $(P-1)(Q-1)$ cannot be prime because it is an even number.

3. Compute D such that $(DE - 1)$ is evenly divisible by $(P-1)(Q-1)$. Mathematicians write this as $DE = 1 \pmod{(P-1)(Q-1)}$, and they call D the multiplicative inverse of E . This is easy to do—simply find an integer X which causes $D = (X(P-1)(Q-1) + 1)/E$ to be an integer, then use that value of D .
4. The encryption function is $C = (T^E) \pmod{PQ}$, where C is the ciphertext (a positive integer), T is the plaintext (a positive integer), and $^$ indicates exponentiation. During the encryption of the message, T must be less than the modulus, PQ .
5. The decryption function is $T = (C^D) \pmod{PQ}$, where C is the ciphertext (a positive integer), T is the plaintext (a positive integer), and $^$ indicates exponentiation.

Your public key is the pair (PQ, E) . Your private key is the number D (reveal it to no one). The product PQ is the modulus. E is the public exponent. D is the secret exponent.

You can publish your public key freely, because there are no known easy methods of calculating D , P , or Q given only (PQ, E) (your public key).

Given below is an example of the RSA algorithm:

$P = 61$ ← first prime number (destroy this after computing E and D)

$Q = 53$ ← second prime number (destroy this after computing E and D)

$PQ = 3233$ ← modulus (give this to others)

$E = 17$ ← public exponent (give this to others)

$D = 2753$ ← private exponent (keep this secret)

Your **public** key is **(E, PQ)**

Your **private** key is **D**

The encryption function is:

$\text{encrypt}(T) = (T^E) \pmod{PQ}$

$= (T^{17}) \pmod{3233}$

The decryption function is:

$\text{decrypt}(C) = (C^D) \pmod{PQ}$

$= (C^{2753}) \pmod{3233}$

To encrypt the plaintext value 123, do this:

$\text{encrypt}(123) = (123^{17}) \pmod{3233}$

$= 337587917446653715596592958817679803 \pmod{3233}$

$= 855$

To decrypt the cipher text value 855, do this:

$$\begin{aligned} \text{decrypt}(855) &= (855 * 2753) \bmod 3233 \\ &= 123 \end{aligned}$$

One way to compute the value of **$855^{2753} \bmod 3233$** is like this:

Consider these powers of 855:

- $855^1 = 855 \pmod{3233}$
- $855^2 = 367 \pmod{3233}$
- $855^4 = 367^2 \pmod{3233} = 2136 \pmod{3233}$
- $855^8 = 2136^2 \pmod{3233} = 733 \pmod{3233}$
- $855^{16} = 733^2 \pmod{3233} = 611 \pmod{3233}$
- $855^{32} = 611^2 \pmod{3233} = 1526 \pmod{3233}$
- $855^{64} = 1526^2 \pmod{3233} = 916 \pmod{3233}$
- $855^{128} = 916^2 \pmod{3233} = 1709 \pmod{3233}$
- $855^{256} = 1709^2 \pmod{3233} = 1282 \pmod{3233}$
- $855^{512} = 1282^2 \pmod{3233} = 1160 \pmod{3233}$
- $855^{1024} = 1160^2 \pmod{3233} = 672 \pmod{3233}$
- $855^{2048} = 672^2 \pmod{3233} = 2197 \pmod{3233}$

Given the above, we know this:

$$\begin{aligned} &855^{2753} \pmod{3233} \\ &= 855^{(1 + 64 + 128 + 512 + 2048)} \pmod{3233} \\ &= 855^1 * 855^{64} * 855^{128} * 855^{512} * 855^{2048} \pmod{3233} \\ &= 855 * 916 * 1709 * 1160 * 2197 \pmod{3233} \\ &= 794 * 1709 * 1160 * 2197 \pmod{3233} \\ &= 2319 * 1160 * 2197 \pmod{3233} \\ &= 184 * 2197 \pmod{3233} \\ &= 123 \pmod{3233} \\ &= 123 \end{aligned}$$

Cryptography
Encryption Algorithms

Diffie-Hellman

CEH
Certified Ethical Hacker

- A cryptographic protocol that allows two parties to establish a **shared key** over an **insecure channel**
- Developed and published by **Whitfield Diffie** and **Martin Hellman** in **1976**
- Actually was independently developed a few years earlier by **Malcolm J. Williamson** of the **British Intelligence Service**, but it was classified

Diffie-Hellman Algorithm

- The system has two parameters called **p** and **g**
 - Parameter **p** is a **prime number**
 - Parameter **g** (usually called a generator) is an integer less than **p** , with the following property: for every number **n** between 1 and **$p-1$** inclusive, there is a power **k** of **g** such that **$n = g^k \bmod p$**
- Many cryptography textbooks use the fictitious characters "**alice**" and "**bob**" to illustrate cryptography, and we will do that here as well:
 - Alice generates a random private value **a** and Bob generates a random private value **b** . Both **a** and **b** are drawn from the **set of integers**
 - They derive their public values using parameters **p** and **g** and their private values. Alice's public value is **$g^a \bmod p$** and Bob's public value is **$g^b \bmod p$**
 - They exchange their public values
 - Alice computes **$g^{ab} = (g^b)^a \bmod p$** , and Bob computes **$g^{ba} = (g^a)^b \bmod p$**
 - Since **$g^{ab} = g^{ba} = k$** , Alice and Bob now have a shared secret key **k**

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Diffie-Hellman

A cryptographic protocol that allows two parties to establish a shared key over an insecure channel. It was developed and published by Whitfield Diffie and Martin Hellman in 1976. Actually, it was independently developed a few years earlier by Malcolm J. Williamson of the British Intelligence Service, but it was classified.

Diffie-Hellman Algorithm

The system has two parameters called **p** and **g**

- Parameter **p** is a prime number and
- Parameter **g** (usually called a generator) is an integer less than **p** , with the following property: for every number **n** between 1 and **$p-1$** inclusive, there is a power **k** of **g** such that **$n = g^k \bmod p$**

Many cryptography textbooks use the fictitious characters 'alice' and 'bob' to illustrate cryptography and we will do that here as well:

- Alice generates a random private value **a** , and Bob generates a random private value **b** . Both **a** and **b** are drawn from the **set of integers**
- They derive their public values using parameters **p** and **g** and their private values. Alice's public value is **$g^a \bmod p$** , and Bob's public value is **$g^b \bmod p$** .
- They exchange their public values
- Alice computes **$g^{ab} = (g^b)^a \bmod p$** , and Bob computes **$g^{ba} = (g^a)^b \bmod p$**
- Since **$g^{ab} = g^{ba} = k$** , Alice and Bob now have a shared secret key **k**

The diagram illustrates the process of a message digest function. On the left, a document icon contains the text 'abcd', 'efgh', 'ijklm', and 'nop'. An arrow labeled 'Message Digest Function' points to a gear icon, which then points to a yellow box containing the hash value 'e1 40 92 aF 94 8b9 38 56 95 84 e5b 8d8d30 7a'. Below the diagram, three bullet points describe the properties of hash functions: 1) They calculate a unique fixed-size bit string representation of any arbitrary block of information. 2) If any given bit of the function's input is changed, every output bit has a 50 percent chance of changing. 3) It is computationally infeasible to have two files with the same message digest value. A note at the bottom states: 'Note: Message digests are also called one-way hash functions because they cannot be reversed.' The slide is titled 'Message Digest (One-Way Hash) Functions' and includes the CEH logo.

Message Digest (One-way Hash) Functions

Hash functions calculate a unique fixed-size bit string representation called a message digest of any arbitrary block of information. Message digest functions distill the information contained in a file (small or large) into a single fixed-length number, typically between 128 and 256 bits. If any given bit of the function's input is changed, every output bit has a 50% chance of changing. Given an input file and its corresponding message digest, it should be nearly impossible to find another file with the same message digest value, as it is computationally infeasible to have two files with the same message digest value.

Message digest functions are also called one-way hash functions because they produce values that are almost impossible to invert, resistant to attack, mostly unique, and widely distributed. Message-digest algorithms themselves do not participate in encryption and decryption operations. They enable creation of digital signatures and message authentication codes (MACs), and the derivation of encryption keys from passphrases.

The main role of a cryptographic hash function is to provide integrity in document management. Cryptographic hash functions are an integral part of digital signatures. They are relatively faster than digital-signature algorithms; hence, their characteristic feature is to calculate the signature of the document's hash value, which is smaller than the document. In addition, digests help to hide the contents or source of the document.


Widely used message-digest functions include the following algorithms:

- MD5
- SHA

Note: Message digests are also called one-way hash functions because they cannot be reversed.

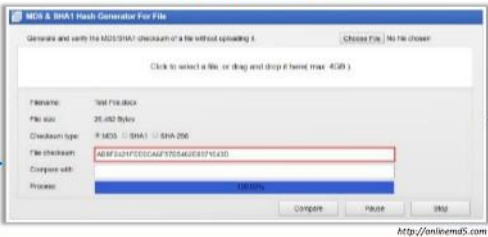
Cryptography
Encryption Algorithms

Message Digest Function: MD5



- MD5 algorithm takes a message of **arbitrary length** as the input and then outputs a **128-bit fingerprint** or message digest of the input
- MD5 is not collision resistant; use of latest algorithms such as **SHA-2** and **SHA-3** is recommended
- It is still deployed for digital signature applications, file integrity checking, and storing passwords

MD5 & SHA1 Hash Generator and Verifier



<http://online-md5.com>

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Message Digest Function: MD5

MD2, MD4, and MD5 are **message-digest algorithms** used in digital signature applications to compress document securely before the system signs it with a private key. The algorithms can be of variable length, but the resulting message digest is always 128 bits.

The structures of all three algorithms appear similar, although the design of MD2 is reasonably different from MD4 and MD5. MD2 supports 8-bit machines, while MD4 and MD5 support 32-bit machines. The algorithm pads the message with extra bits to ensure that the length of the bits is divisible by 512. The extra bits may include a 64-bit binary message.

Attacks on versions of MD4 have become increasingly successful. Research has shown how attacker launches collision attacks for the full version of MD4 under a minute on a typical PC. MD5 is slightly more secure, but is slower than MD4. However, both the message-digest size and padding requirements remain the same.

MD5 algorithm is a widely used cryptographic hash function that takes a message of arbitrary length as input and outputs a 128-bit (16-byte) fingerprint or message digest of the input. MD5 algorithm comes into use in a wide variety of cryptographic applications and is useful for digital signature applications, file integrity checking, and storing passwords. On the other hand, MD5 is not collision resistant; therefore, it is better to use the latest algorithms, such as SHA-2 and SHA-3.

To calculate the effectiveness of hash functions, check the output produced when the algorithm randomizes an arbitrary input message.

The following are examples of minimally different message digests:

- echo "There is CHF1500 in the blue bo" | md5sum
e41a323bdf20eadafd3f0e4f72055d36

- echo "There is CHF1500 in the blue box" | md5sum
7a0da864a41fd0200ae0ae97afd3279d
- echo "There is CHF1500 in the blue box." | md5sum
2db1ff7a70245309e9f2165c6c34999d

Even minimally different texts produce radically different MD5 codes.

- **Onlinemd5**

Source: <http://onlinemd5.com>

Onlinemd5 generates and checks file integrity by secure time-proven algorithms like MD5, SHA-1 and SHA-256. One can create checksums (the digital fingerprints) of files and verify their integrity in the future using this online tool.

Cryptography
Encryption Algorithms

Message Digest Function: Secure Hashing Algorithm (SHA)

CEH

It is an algorithm to generate cryptographically secure one-way hash, published by the **National Institute of Standards and Technology** as a **US Federal Information Processing Standard**

SHA1 It produces a **160-bit digest** from a message with a maximum length of **(264 - 1) bits**, and it resembles the MD5 algorithm

SHA2 It is a family of two similar hash functions with different block sizes, namely, **SHA-256** that uses **32-bit words** and **SHA-512** that uses **64-bit words**

SHA3 SHA-3 uses the **sponge construction** in which message blocks are **XORed** into the initial bits of the state, which is then invertibly permuted

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Message Digest Function: Secure Hashing Algorithm (SHA)

The NIST has developed the Secure Hash Algorithm (SHA), specified in the **Secure Hash Standard (SHS)** and published as a federal information-processing standard (FIPS PUB 180). It generates a cryptographically secure one-way hash. Rivest developed the SHA, which is similar to the message-digest algorithm family of hash functions. It is slightly slower than MD5, but its larger message digest makes it more secure against brute-force collision and inversion attacks.

SHA encryption is a series of five different cryptographic functions, and it currently has three generations: SHA-1, SHA-2, and SHA-3.

- **SHA-0:** A retronym applied to the original version of 160-bit hash function published in the year 1993 under the name SHA, which was withdrawn from the trade due to undisclosed **"significant flaw"** in it and was replaced with slightly revised version SHA-1.
- **SHA-1:** It is a 160-bit hash function that resembles the former MD5 algorithm developed by Ron Rivest. It produces a 160-bit digest from a message with a maximum length of (264 - 1) bits. It was designed by the National Security Agency (NSA) to be part of the Digital Signature Algorithm (DSA). It is most commonly used in security protocols such as PGP, TLS, SSH, and SSL. As of 2010, SHA-1 is no longer approved for cryptographic use because of cryptographic weaknesses.
- **SHA-2:** SHA2 is a family of two similar hash functions, with different block sizes, namely, SHA-256, which uses 32-bit words, and SHA-512, which uses 64-bit words. Truncated versions of each standard are SHA-224 and SHA-384.
- **SHA-3:** SHA-3 uses the sponge construction in which message blocks are XORed into the initial bits of the state, which the algorithm then invertibly permutes. It supports the

same hash lengths as SHA-2 and differs in its internal structure considerably from rest of the SHA family.

Comparison of SHA functions (SHA-0, SHA-1, SHA-2, and SHA-3).





Algorithm and variant		Output size (bits)	Internal state size (bits)	Block Size (bits)	Maximum message size (bits)	Rounds	Operations	Security (bits)
MD5 (as reference)		128	128 (4*32)	512	$2^{64}-1$	64	Add mod 2^{32} , and, or, xor, rot	<64 (collisions found)
SHA-0		160	160 (5*32)	512	$2^{64}-1$	80	Add mod 2^{32} , and, or, xor, rot	<80 (collisions found)
SHA-1		160	160 (5*32)	512	$2^{64}-1$	80	Add mod 2^{32} , and, or, xor, rot	<80 (theoretical-attack in 2^{61})
SHA-2	SHA-224	224	256 (8*32)	512	$2^{64}-1$	64	Add mod 2^{32} , and, or, xor, shr, rot	112
	SHA-256	256						
	Sha-384	384	512 (8*64)	1024	$2^{128}-1$	80	Add mod 2^{64} , and, or, xor, shr, rot.	192
	Sha-512	512						
Sha-512/224	224							
	Sha-512/256	256						128
SHA-3	Sha3-224	224	1600 (5*5*64)	1152	∞	24	and, xor, not, rot	112
	Sha3-256	256		1088				128
	Sha3-384	384		832				192
	Sha3-512	512		576				256
	Shake128	d(arbitrary)		1344				Min(d/2,128)
	shake256	d(arbitrary)		1088				Min(d/2,256)

TABLE 20.2: Comparison between SHA-0, SHA-1 and SHA-2 functions

Cryptography
Encryption Algorithms

RIPEMD-160

CEH

- **RACE Integrity Primitives Evaluation Message Digest (RIPEMD)** is a **160-bit hash algorithm** developed by Hans Dobbertin, Antoon Bosselaers, and Bart Preneel 
- There exist 128, 256, and 320-bit versions of this algorithm, called **RIPEMD-128**, **RIPEMD-256**, and **RIPEMD-320**, respectively 
- The compression function consists of **80 stages made up of 5 blocks** that execute **16 times each** 
- This process **repeats twice** by combining the results at the bottom using **modulo 32 addition** 

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

RIPEMD-160

RACE Integrity Primitives Evaluation Message Digest (RIPEMD) is a 160-bit hash algorithm developed by Hans Dobbertin, Antoon Bosselaers, and Bart Preneel. There exist 128, 256 and 320-bit versions of this algorithm, called RIPEMD-128, RIPEMD-256, and RIPEMD-320, respectively. These all replace the original RIPEMD, which was found to have collision issue. It does not follow any standard security policies or guidelines.

RIPEMD-160 is a more secure version of the RIPEMED algorithm. In this algorithm, the compression function consists of 80 stages made up of 5 blocks that execute 16 times each. This process repeats twice by combining the results at the bottom using modulo 32 addition.

Cryptography
Encryption Algorithms

HMAC

CEH

- 1 HMAC is a type of **message authentication code** (MAC) that makes use of **cryptographic key** with a combination of a cryptographic hash function
- 2 It is widely used to verify the **integrity of the data** and **authentication** of a message
- 3 This algorithm includes an embedded hash function such as **SHA-1** or **MD5**
- 4 The strength of HMAC depends on the **embedded hash function**, key size, and the size of the hash output
- 5 As HMAC executes the underlying hash function twice, it protects from various **length extension attacks**

Copyright © by **EC-Council**. All Rights Reserved. Reproduction is Strictly Prohibited.

HMAC

Hash based message authentication code (HMAC) is a type of message authentication code (MAC) that uses a cryptographic key along with a cryptographic hash function. It is widely used to verify the integrity of the data and authentication of a message. This algorithm includes an embedded hash function such as SHA-1 or MD5. The strength of the HMAC depends on the embedded hash function, key size, and the size of the hash output.

HMAC includes two stages for computing the hash. The input key is processed to produce two keys namely inner key and outer key. The first stage of the algorithm inputs inner key and message to produce an internal hash. The second stage of the algorithm inputs the output from the first stage and outer key and produces the final HMAC code.

As the HMAC executes the underlying hash function twice, it protects from various length extension attacks. The size of the key and output depends on the embedded hash function. For example, 128 or 160-bits in the case of MD5 or SHA-1, respectively.

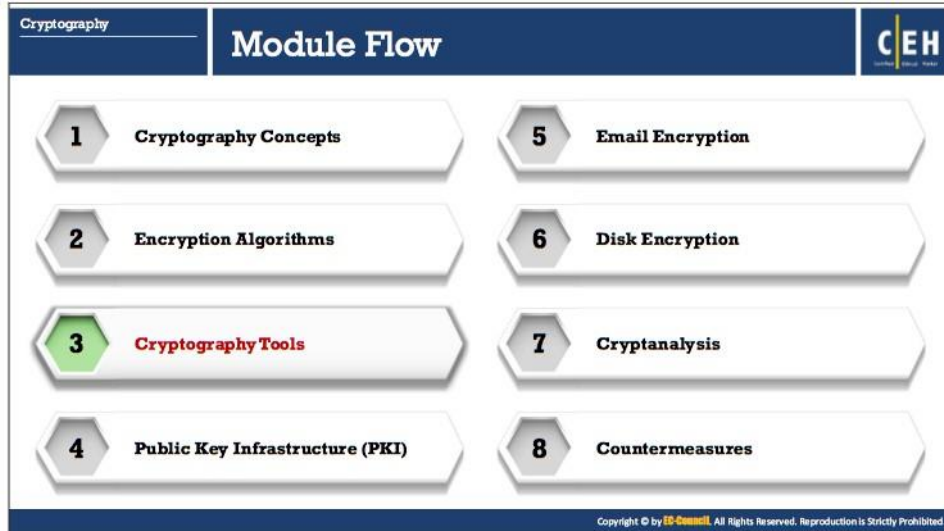
CHAP

Challenge-Handshake Authentication Protocol (CHAP) is an authentication mechanism used by Point to Point protocol (PPP) servers in order to authenticate or validate the identity of remote clients or network hosts. It is more secure and effective as compared to Password Authentication Procedure (PAP) as it regularly verifies the identity of the client using three-way handshake and provides protection against replay attacks.

EAP

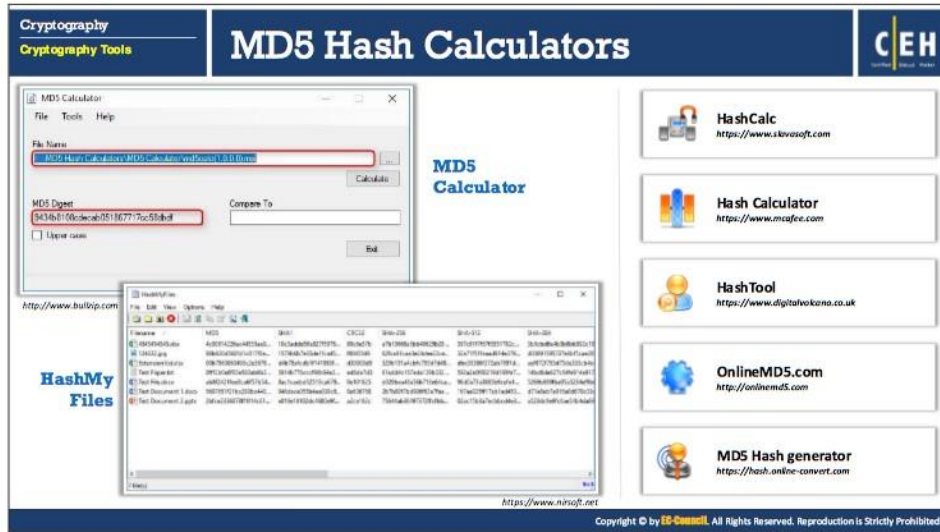
Extensible Authentication Protocol (EAP) is an authentication protocol that was originally designed for Point-to-Point connections. It is used as an alternative to CHAP and PAP authentication protocols as it is more secure and supports different authentication mechanisms

such as passwords, smart tokens, OTPs (one-time passwords), Secure ID card, digital certificates and public key encryption mechanism. After the selection of EAP authentication mechanism, a session is established and messages are exchanged between the client and the authenticating server. The session consists of requests and responses for authentication information. The length and details of the authentication session are determined by the EAP authentication mechanism used.



Cryptography Tools

This section deals with various cryptography tools that you can use to encrypt sensitive data to protect it from unauthorized access by any party other than the person for whom it is intended.



MD5 Hash Calculators

Discussed below are MD5 hash calculators that use different hash algorithms to convert plain text into its equivalent hash value.

- **MD5 Calculator**

Source: <http://www.bullzip.com>

MD5 Calculator allows to calculate the MD5 hash value of the selected file. Right click the file and choose "MD5 Calculator," the program will calculate the MD5 hash. The MD5 Digest field contains the calculated value. To compare this MD5 digest to another, one can paste the other value into the Compare To field. Obviously, an equal sign ("=") appears between the two values if they are equal; otherwise, the less than ("<") or greater than (>) sign will tell you that the values are different.

- **HashMyFiles**

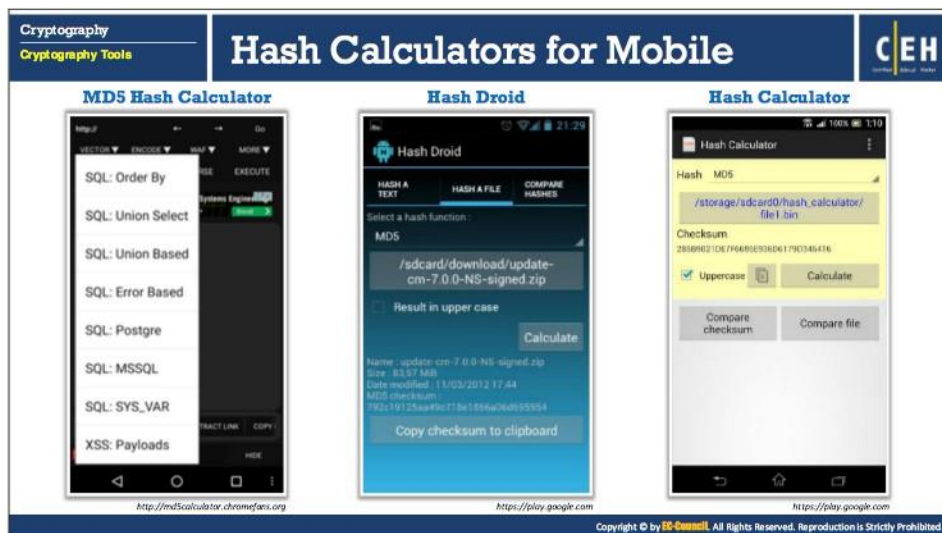
Source: <https://www.nirsoft.net>

HashMyFiles is small utility that allows to calculate the MD5 and SHA1 hashes of one or more files in the system. It allows to copy the MD5/SHA1 hashes list into the clipboard or save them into text/html/xml file. One can launch HashMyFiles from the context menu of Windows Explorer and display MD5/SHA1 hashes of the selected file or folder.

Some of the additional MD5 hash calculators are listed below:

- HashCalc (<https://www.slavasoft.com>)
- Hash Calculator (<https://www.mcafee.com>)
- HashTool (<https://www.digitalvolcano.co.uk>)

- OnlineMD5.com (<http://onlinemd5.com>)
- MD5 Hash generator (<https://hash.online-convert.com>)
- MD5 HASH CALCULATOR (<http://www.md5calc.com>)
- MD5 hash calculator (<http://www.xorbin.com>)
- md5 Hash Generator (<https://www.miraclesalad.com>)
- MD5 Hash Generator (<https://hash.online-convert.com>)
- MD5 Hash Generator (<https://www.md5hashgenerator.com>)
- MD5 hash calculator (<https://abunchofutils.com>)
- HTML5 File Hash Online Calculator (<https://md5file.com>)



Hash Calculators for Mobile

Discussed below are some of the hash calculators for mobile.

- **MD5 Hash Calculator**

Source: <http://md5calculator.chrome fans.org>

The MD5 Hash Calculator for Android is used to generate the MD5 hash of a string in security. It is useful for encoding passwords, credit-card numbers, and other sensitive data into databases (MySQL, MSSQL, Postgress, or others).

- **Hash Droid**

Source: <https://play.google.com>

The Hash Droid utility helps to calculate a hash from a given text or from a file stored on the device. In this application, the available hash functions are: Adler-32, CRC-32, Haval-128, MD2, MD4, MD5, RIPEMD-128, RIPEMD-160, SHA-1, SHA-256, SHA-384, SHA-512, Tiger and Whirlpool. It allows copying the calculated hash to the clipboard to reuse it elsewhere. Hash Droid often facilitates to check an Android ROM before flashing it.

- **Hash a Text tab** – enables to calculate the hash of a given string.
- **Hash a File tab** – helps to compute the hash of a file located on the internal or external memory of the device. It displays the size of the file and the last date modified.
- **Compare Hashes tab** – helps to compare the calculated hash with another given hash but more generally, one can compare any hashes by just pasting them.

- **Hash Calculator**

Source: <https://play.google.com>

Hash Calculator allows users to calculate MD5, SHA1 or CRC32 checksum of files. It allows to compare a checksum of a file with known checksum in order to check the file integrity. It also compares files with their checksums.

Some of the additional MD5 hash calculators are listed below:

- Hash Calc (<https://play.google.com>)
- Hashr - Checksum & Hash Digest Calculator (<https://play.google.com>)
- HashStamp MD5 & SHA1 Checker (<https://play.google.com>)
- Hash Tools (<https://play.google.com>)
- HashCalc (<https://play.google.com>)
- Hash Generator (<https://play.google.com>)
- Checksum MD5 & SHA1 (<https://play.google.com>)
- Hash Generator - Checksum Calculator (<https://play.google.com>)



Cryptography Tools: Advanced Encryption Package 2017 and BCTextEncoder

You can use various cryptographic tools for encrypting and decrypting your information, files, and so on. These tools implement different types of available encryption algorithms.

- **Advanced Encryption Package 2017**

Source: <https://www.aepro.com>

Advanced Encryption Package 2017 is file encryption software for Windows used for secure file transfer, batch file encryption, and encrypted backups.

Features:

- Supports file and/or text encryption
- Uses symmetric and asymmetric algorithms and supports for public-private RSA key pair (one key for encryption and another for decryption), as well as for binary symmetric (randomly generated) files
- Performs secure file deletion
- Supports USB flash drives to store encryption and decryption keys
- Creates encrypted self-extracting file to send it as email attachment
- Command line support to fully automate encryption and decryption tasks

- **BCTextEncoder**

Source: <https://www.jetico.com>

BCTextEncoder utility software simplifies the encoding and decoding of text data. It compresses, encrypts, and converts plain text data to text format, which the user can

then copy to the clipboard or save as a text file. It uses public key encryption methods as well as password-based encryption. It uses strong and approved symmetric and public key algorithms for data encryption.

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Cryptography Tools

Some of the additional cryptography tools are listed below:

- AxCrypt (<https://www.axcrypt.net>)
- Folder Lock (<http://www.newsoftwares.net>)
- CryptoExpert 8 (<https://www.cryptoexpert.com>)
- CertainSafe (<https://certainsafe.com>)
- VeraCrypt (<https://veracrypt.codeplex.com>)
- Cryptainer LE Free Encryption Software (<http://www.cipherix.com>)
- CryptoForge (<https://www.cryptoforge.com>)
- winAES (<https://winaes.com>)
- EncryptOnClick (<https://www.2brightsparks.com>)
- GNU Privacy Guard (<https://www.gnupg.org>)
- Steganos Safe 19 (<https://www.steganos.com>)
- Secure IT (<http://www.cypherix.com>)
- AES Crypt (<https://www.aescrypt.com>)
- Steganos LockNote (<https://sourceforge.net>)
- Autokrypt (<http://www.hiteksoftware.com>)



Cryptography Tools for Mobile

Discussed below are some cryptographic tools for mobile devices:

- **Secret Space Encryptor**

Source: <https://paranoiaworks.mobi>

Secret Space Encryptor is an integrated solution of password manager, message (text) encryption, and file encryption.

Features:

- Stores and manages all passwords, PINs, or notes in one secure place with the Password Vault
- Keeps messages, notes, and other texts safe from unintended readers with the Message Encryptor
- Securely encrypts private and confidential files or whole folders with the File Encryptor. Wiping (secure delete) feature is included
- Uses encryption algorithms: AES (Rijndael) 256bit, RC6 256bit, Serpent 256bit, Blowfish 256bit/448bit, Twofish 256bit, GOST 256bit
- Has encryption engine based on the Bouncy Castle Crypto Library (Java) and Crypto++ (C++)

- **Decrypto**

Source: <https://play.google.com>

Decrypto is an application that provides a range of encryption and decryption tools. Over 25 traditional ciphers are implemented such as the Caesar code, ASCII code, or the

code of Vigenère. This tool allows to test all the different types of ciphers and display the possible results based on a dictionary. A frequency analysis tool for an encrypted text is also available. One can know exactly the number of occurrences of each letter in the encrypted text.

- **SealNote Secure Encrypted Note**

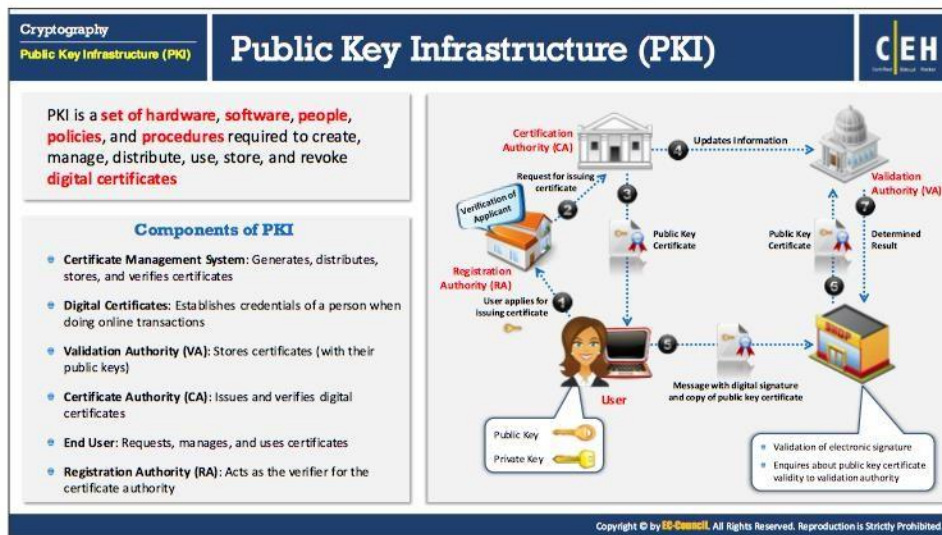
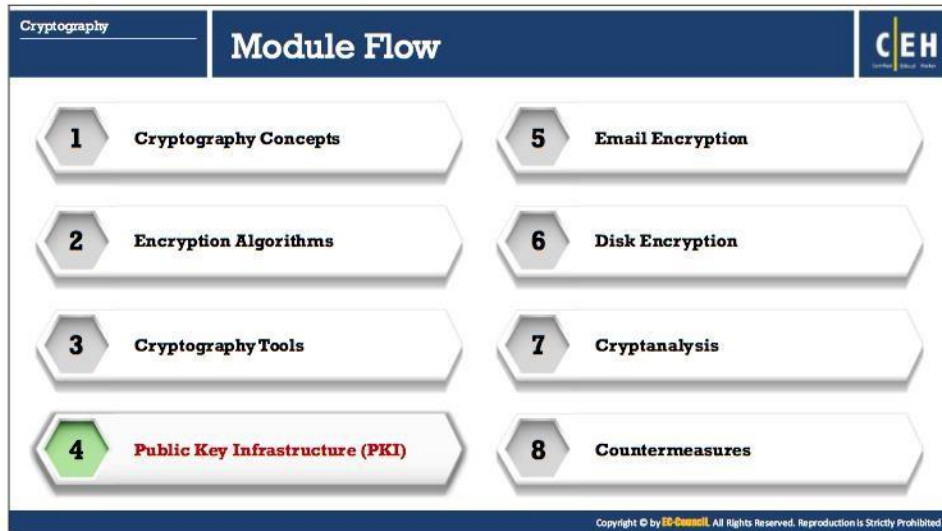
Source: <https://play.google.com>

SealNote is simple, safe and easy to use notepad application that puts security first. Your notes are password protected using industry standard 256-bit AES encryption. Keep sensitive information always available without compromising security!

- Password protection (256-bit AES encryption)
- Organize notes using colors codes and tags
- Three different styles to list your notes and please your eyes
- Password expires after configurable timeout
- Protect content from screenshots, window switcher, and other non-secure displays
- Create encrypted data file for backup/restore

Some of the additional cryptography tools for mobiles are listed below:

- Encrypt Decrypt (<https://play.google.com>)
- Crypten : Encryption (<https://play.google.com>)
- Cipher Sender (<https://play.google.com>)



Public Key Infrastructure (PKI)

This section deals with Public Key Infrastructure (PKI) and the role of each components of PKI, certification authorities: Comodo, IdenTrust, Symantec, and GoDaddy, etc. and the signed certificate (CA) vs. the self-signed certificate.

Public Key Infrastructure (PKI) is a security architecture developed to increase the confidentiality of information exchanged over the insecure Internet. It includes hardware, software, people, policies, and procedures required to create, manage, distribute, use, store, and revoke digital certificates. In cryptography, the PKI helps to bind public keys with corresponding user identities by means of a Certificate Authority (CA).

Components of PKI

- **Certificate Management System:** Generates, distributes, stores, and verifies certificates
- **Digital Certificates:** Establishes credentials of a person when doing online transactions
- **Validation Authority (VA):** Stores certificates (with their public keys)
- **Certificate Authority (CA):** Issues and verifies digital certificates
- **End User:** Requests, manages, and uses certificates
- **Registration Authority (RA):** Acts as the verifier for the certificate authority

PKI is a comprehensive system that allows the use of public-key encryption and digital signature services across a wide variety of applications. PKI authentication depends on digital certificates (also known as public-key certificates) that certificate authorities sign and provide. Digital certificate is a digitally signed statement with a public key and the subject (user, company, or system) name in it.

PKI uses public-key cryptography, which is widely used on the Internet to encrypt messages or authenticate message senders. In public-key cryptography, a Certification Authority generates a public and private key with the same algorithm simultaneously by (CA). The private key is held only by the subject (user, company, or system) mentioned in the certificate, while the public key is made publicly available in a directory that all parties can access. The subject keeps the private key secret and uses it to decrypt the text encrypted by someone else using the corresponding public key (available in a public directory). This way, others encrypt messages for the user with the user's public key, and the user decrypts it with his/her private key.

Given below are the steps involved in PKI process:

1. The subject (user, company, or system) intended to exchange information securely, applies for a certificate with the Registration Authority (RA).
2. The Registration Authority (RA) receives request from the subject, verifies subject's identity, and requests the Certification Authority (CA) to issue a public key certificate to the user.

3. The Certification Authority (CA) issues the public key certificate binding subject's identity with subject's public key and thereafter sends the updated information to the Validation Authority (VA).
4. When a user makes a transaction, the user duly signs the message digitally in the public key certificate and sends the message to the client.
5. The client verifies the authenticity of the user by inquiring about the user's public key certificate validity with the VA.
6. The VA compares the public key certificate of the user with that of the updated information provided by the CA and determines the result (whether valid or not).



Certification Authorities

Certification authorities are trusted entities that issue digital certificates. The digital certificate certifies the possession of the public key by the subject (user, company, or system) specified in the certificate. This aids others to trust signatures or statements made by the private key that is associated with the certified public key.

Discussed below are some popular certification authorities:

- **Comodo**

Source: <https://www.comodo.com>

Comodo offers a range of PKI digital certificates with strong SSL encryption available 128/256 with SGC (Server-Gated Cryptography). It ensures standards of confidentiality, system reliability, and pertinent business practices as judged through qualified independent audits. It offers a PKI (public-key infrastructure) management solution such as Comodo Certificate Manager and Comodo EPKI Manager.

Digital Certificates offered by Comodo:

- Extended validation (EV)-SSL, Multi-domain EV SSL and Wildcard SSL
- Unified Communications (UC)
- Intel Pro Series
- General Purpose SSL and Secure Email - S/MIME
- Client Authentication and Code Signing

- **IdenTrust**

Source: <https://www.identrust.com>

IdenTrust is a trusted third party that provides certification authority services for many sectors like banks, corporate, government, and healthcare. It provides the following solutions:

- Digital Signing and Sealing
- Comply with NIST SP 800-171
- Global Identity Network
- Managed PKI Hosting Services
- eSignature Solutions / Electronic Document Signing
- Individual Level Signing of Payment Files
- IdenTrust Community Bank PKI [PDF] and Secure Email

- **Symantec**

Source: <https://www.websecurity.symantec.com>

Symantec Corporation (NASDAQ: SYMC) provides solutions that allow companies and consumers to engage in communications and commerce online with confidence.

SSL/TLS Certificates Offered:

- Secure Site, Secure Site with EV, Secure Site Pro, Secure Site Pro with EV, and Secure Site Wildcard
- Complete Website Security

- **GoDaddy**

Source: <https://in.godaddy.com>

GoDaddy SSL Certificates offer a complete range of certificates that comply with CA/Browser Forum guidelines.

Features:

- SHA-2 hash algorithm and 2048-bit encryption
- Protects unlimited servers
- Boost Google search ranking
- Displays a Security Seal on client's site
- Compatible with all major browsers

Some of the available Certification Authorities are listed below:

- GlobalSign (<https://www.globalsign.com>)
- DigiCert (<https://www.digicert.com>)

- Certum (<http://www.certum.eu>)
- Entrust (<https://www.entrustdatacard.com>)
- StartCom (<https://www.startcomca.com>)
- thwate (<https://www.thawte.com>)
- Verisign (<https://www.verisign.com>)
- RapidSSL (<https://www.rapidssl.com>)
- Escrow (<https://www.escrow.com>)



Signed Certificate (CA) Vs. Self-Signed Certificate

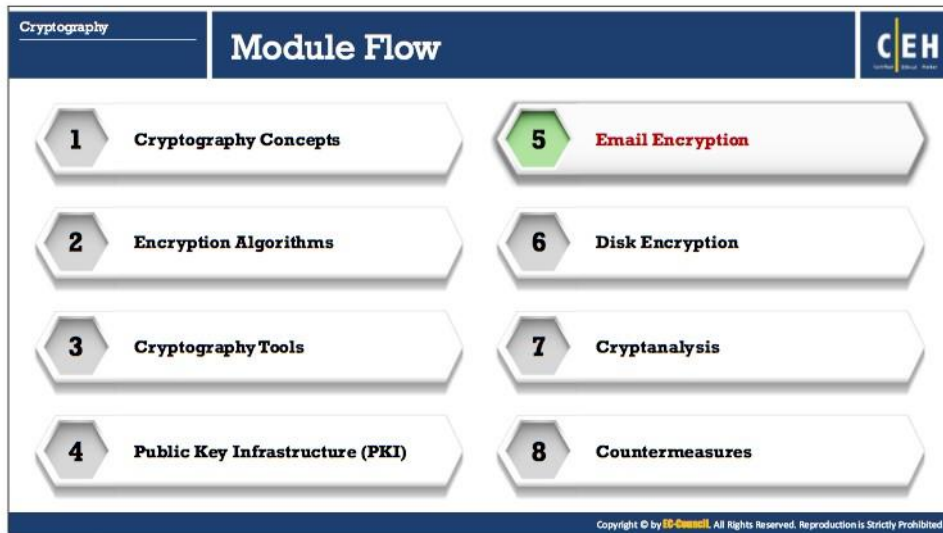
- **Signed Certificates**

Certification authorities (CAs) sign and issue signed certificates. These certificates contain a public key and the identity of the owner. The corresponding private key is kept secret by the CA. By issuing the certificate, the CA confirms or validates that the public key contained in the certificate belongs to the person, company, server, or other entity mentioned in the certificate. CA verifies an application's credentials, thus users and relying parties trust the information in the CA's certificates. The CA takes accountability for saying, "Yes, this person is who they state they are, and we, the CA, certify that." Some of the popular CAs include Comodo, IdenTrust, Symantec, and GoDaddy.

The first diagram in the above slide illustrates the process of using a signed certificate. A user approaches a CA and purchases a digital certificate. The CA issues a certificate to the user and updates the VA with the same information. The user signs a document with the public key and sends it to the receiver. The receiver verifies the certificate with VA, who verifies the authenticity but does not share the private key.

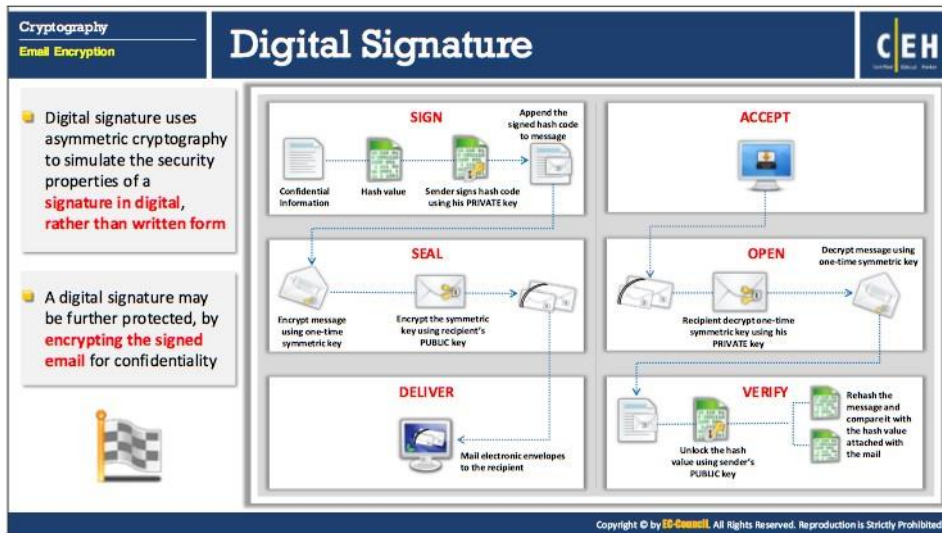
- **Self-Signed Certificates**

A self-signed certificate is an identity certificate signed by the same entity whose identity it certifies. Self-signed certificates are widely used for testing purposes. In self-signed certificates, a user creates a pair of public and private keys using a certificate creation tool such as Adobe Reader, Java's keytool, Apple's Keychain, etc. and signs the document with the public key. The receiver requests the sender for the private key to verify the certificate. However, the certificate verification rarely occurs due to necessity of disclosing the private key. This makes self-signed certificates useful only in a self-controlled testing environment.



Email Encryption

Currently, most businesses use email as the primary source of communication, as it is simple and easy to communicate or share information. Emails can contain sensitive information about the organization, such as projects, upcoming news, and financial data, which, when accessed by the wrong person can cause huge losses to the organization. One can protect emails containing sensitive information by encrypting them. This section deals with email security mechanisms—digital signature, SSL, TLS, cryptographic toolkits, and PGP.

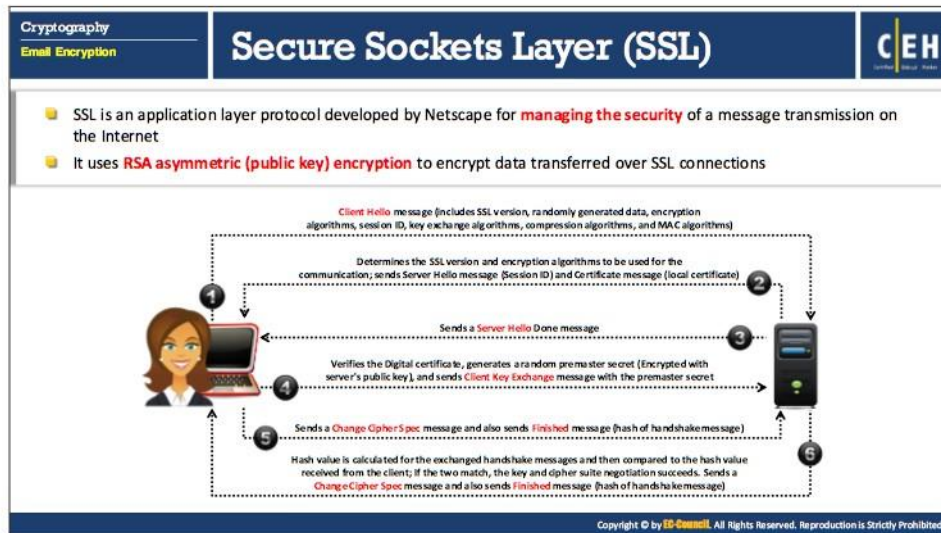


Digital Signature

Digital signature uses asymmetric cryptography to simulate the security properties of a signature in digital, rather than written form. A digital signature is a cryptographic means of authentication. Public-key cryptography uses asymmetric encryption and helps the user to create a digital signature. The two types of keys in public key cryptography are the private key (only signer knows this key and uses it to create digital signature) and the public key (more widely known and a relying party uses it to verify the digital signature).

A hash function is an algorithm that helps user to create and verify a digital signature. This algorithm creates a digital representation, also known as the message fingerprint. This fingerprint has a hash value that is much smaller than the message, but one that is unique to it. If the attacker changes the message, the hash function will automatically produce a different hash value.

To verify the digital signature, one needs the hash value of the original message and the hash function used to create the digital signature. With the help of both the public key and the new result, the verifier checks to see if the digital signature was created with the related private key, and whether the new hash value is the same as the original. A digital signature may be further protected by encrypting the signed email for confidentiality.



Secure Sockets Layer (SSL)

The Secure Sockets Layer (SSL) is an application layer protocol developed by Netscape for managing the security of a message transmission on the Internet. It is a protocol used to provide a secure authentication mechanism between two communicating applications, such as a client and a server. The SSL requires a reliable transport protocol, such as TCP, for data transmission and reception. It uses RSA asymmetric (public key) encryption to encrypt data transferred over SSL connections.

Any application-layer protocol that is higher than SSL, such as HTTP, FTP, and telnet, can form a transparent layer over the SSL. SSL acts as an arbitrator between the encryption algorithm and session key; it also verifies the destination server prior to the transmission and reception of data. The SSL encrypts the complete data of the application protocol to ensure security.

The SSL protocol also offers “**channelsecurity**” with three basic properties:

- **Private channel** – All the messages are encrypted after a simple handshake is used to define a secret key.
- **Authenticated channel** – The server endpoint of the conversation is always encrypted, whereas the client endpoint is optionally authenticated.
- **Reliable channel** – message transfer has an integrity check.

SSL uses both asymmetric and symmetric authentication mechanisms. Public-key encryption verifies the identities of the server, the client, or both. Once authentication has taken place, the client and server can create symmetric keys allowing them to communicate and transfer data rapidly. An SSL session is responsible for carrying out the SSL handshake protocol to organize the states of the server and clients, thus ensuring the consistency of the protocol.

SSL Handshake Protocol Flow

The SSL handshake protocol works on top of the SSL record layer. The processes executed in the three-way handshake protocol are as follows:

1. The client sends a hello message to the server, which the server must respond to with a hello message, or the connection will fail due to the occurrence of a fatal error. The attributes established due to the server, and client hello are protocol version, session ID, cipher suite, and compression method.
2. After the connection is established, the server sends a certificate to the client for authentication. In addition, server might send a server-key exchange message. On authentication of server, it may ask the client for the certificate (if appropriate to the cipher suite selected).
3. The server sends a "hello done" message to inform the client that the handshake phase is complete and waits for the client's response.
4. If the client receives a certificate-request message, the client must respond to the message by sending a certificate message or "no certificate" alert. The server sends the client key-exchange message. The content of the message depends on the public-key algorithm between the server hello and client hello. If the certificate sent by the client has signing ability, a digitally signed certificate verifies the message, and the client transmits it.
5. The client transmits the changed cipher-spec message and copies the pending cipher spec into the current cipher spec. The client sends a message to initiate the completion of the message under the new algorithm, keys, and secrets.
6. In response, the server replies by sending its own changed cipher-spec message, transfers the pending cipher spec to the current cipher spec, and initiates the completion of the message under the new cipher spec. At this point, the handshake is complete, and the server starts to exchange the application-layer data.

The resumption of a previous session or the replication of an existing session proceeds as follows:

- The client initiates the communication by sending a hello message with the session ID of the session that is to be resumed.
- If the server finds a match, it re-establishes the session under the specified session state with the same session ID.
- At this point, both the server and the client exchange the changed spec messages and proceed directly to the finished messages.
- After re-establishment, the server and client exchange data at the application layer.
- If the session ID does not exist, the server creates a new session ID. The SSL client and server then carry out a complete handshake.

Cryptography
Email Encryption

Transport Layer Security (TLS)

CEH

- TLS is a protocol to establish a secure connection between a client and a server and ensure privacy and integrity of information during transmission
- It uses the RSA algorithm with 1024 and 2048 bit strengths

TLS Handshake Protocol

It allows the client and server to authenticate each other, select encryption algorithm, and exchange symmetric key prior to data exchange

TLS Record Protocol

It provides secured connections with an encryption method such as DES

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Transport Layer Security (TLS)

Transport Layer Security (TLS) is a protocol used to establish a secure connection between a client and a server and ensure privacy and integrity of information during transmission. It uses symmetric key for bulk encryption, asymmetric key for authentication and key exchange, and message authentication codes for message integrity. It uses the RSA algorithm with 1024- and 2048-bit strengths. With the help of TLS, one can reduce security risks such as message tampering, message forgery, and message interception. An advantage of TLS is that it is application-protocol independent. Higher-level protocols can layer on top of the TLS protocol transparently.

TLS protocol consists of two layers; TLS Record Protocol and TLS Handshake Protocol.

1. TLS Record Protocol

The TLS Record Protocol is a layered protocol. It provides secured connections with an encryption method such as Data Encryption Standard (DES). It secures application data using the keys generated during the handshake and verifies its integrity and origin. The TLS Record Protocol provides connection security that has two basic properties:

- The connection is private:** Uses symmetric cryptography for data encryption (e.g., DES and RSA). The protocol generates unique keys for symmetric encryption for each connection, depending on a secret negotiated by another protocol (such as the TLS Handshake Protocol). One can use the Record Protocol without encryption.
- The connection is reliable:** It provides a message integrity check at the time of message transport using a keyed MAC. Secure hash functions (e.g., SHA, MD5) help to perform MAC computations.

TLS Record Protocol manages the following:

- Fragments outgoing data into manageable blocks and reassembles incoming data
- Optionally compresses outgoing data and decompresses incoming data
- Applies Message Authentication Code (MAC) to the outgoing data and uses MAC to verify the incoming data
- Encrypts outgoing data and decrypts incoming data

The record protocol sends the outgoing encrypted data to TCP layer for transport.

2. TLS Handshake Protocol

TLS Handshake Protocol allows the client and server to authenticate each other and to select an encryption algorithm and cryptographic keys prior to data exchange by the application protocol.

It provides connection security that has three basic properties:

- The peer's identity can be authenticated using asymmetric cryptography. This can be made optional but mostly required for at least one of the peers.
- The negotiation of a shared secret is secure.
- The negotiation is reliable.

The TLS handshake protocol operates on top of the TLS record layer and is responsible to produce cryptographic parameters of the session state. At the start of communication, TLS client and server agree on a protocol version, select cryptographic algorithms, optionally authenticate each other, and use asymmetric cryptography techniques to create shared secrets.


Given below are the steps involved in TLS handshake Protocol:

- Initially, the client sends a "Client hello" message, accompanied by the client's random value and supported cipher suites to the server.
- The server responds to the client by sending a "Server hello" message accompanied by the server's random value.
- The server sends its certificate to the client to authenticate and may request client's certificate. The server sends the "Server hello done" message.
- The client sends its certificate to the server, if requested.
- The client generates a random Pre-Master Secret and encrypts it with the Server's public key; it then sends the encrypted Pre-Master Secret to the server.
- The server receives the Pre-Master Secret. Thereafter, the client and server each create the Master Secret and session Keys based on the Pre-Master Secret.
- The client sends "Change cipher spec" notification to the server to indicate that it will start using the new session keys for hashing and encrypting messages. The client also sends "Clientfinished" message.

- The server receives "Changecipher spec" from the client and switches its record layer security state to symmetric encryption using the session keys. The server sends "Serverfinished" message to the client.
- Now, the client and server can exchange application data over the secure channel they have established and all the messages being exchanged between the client and server are encrypted using a session key.

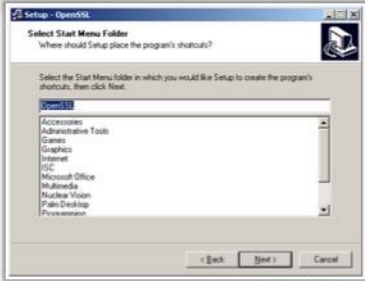
Cryptography
Email Encryption

Cryptography Toolkits: OpenSSL and Keyczar



OpenSSL

- OpenSSL is an open source cryptography toolkit implementing the **Secure Sockets Layer (SSL v2/v3)** and **Transport Layer Security (TLS v1)** network protocols and related cryptography standards required by them



<https://www.openssl.org>

Keyczar

- Keyczar is an open source cryptographic toolkit designed to make it easier and safer for developers to use **cryptography in their applications**
- It **supports authentication and encryption** with both symmetric and asymmetric keys

Features:

- Key rotation and versioning
- Safe default algorithms, modes, and key lengths
- Automated generation of initialization vectors and ciphertext signatures
- Java, Python, and C++ implementations
- International support in Java

<https://github.com>

Copyright © by **EC-Council**. All Rights Reserved. Reproduction is Strictly Prohibited.

Cryptography Toolkits

Cryptography toolkits include cryptographic primitives, algorithms, and schemes used to provide security for various applications. Discussed below are few of the cryptography toolkits:

- **OpenSSL**

Source: <https://www.openssl.org>

OpenSSL is an open source cryptography toolkit implementing the Secure Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS v1) network protocols and related cryptography standards required by them. The openssl program is a command line tool for using the various cryptography functions of OpenSSL's crypto library from the shell.

OpenSSL can be used for:

- Creation and management of private keys, public keys, and parameters
- Public key cryptographic operations
- Creation of X.509 certificates, CSRs, and CRLs
- Calculation of Message Digests
- Encryption and Decryption with Ciphers
- SSL/TLS Client and Server Tests
- Handling of S/MIME signed or encrypted mail
- Time Stamp requests, generation, and verification

- **Keyczar**

Source: <https://github.com>

Keyczar is an open source cryptographic toolkit designed to make it easier and safer for developers to use cryptography in their applications. It supports authentication and encryption with both symmetric and asymmetric keys.

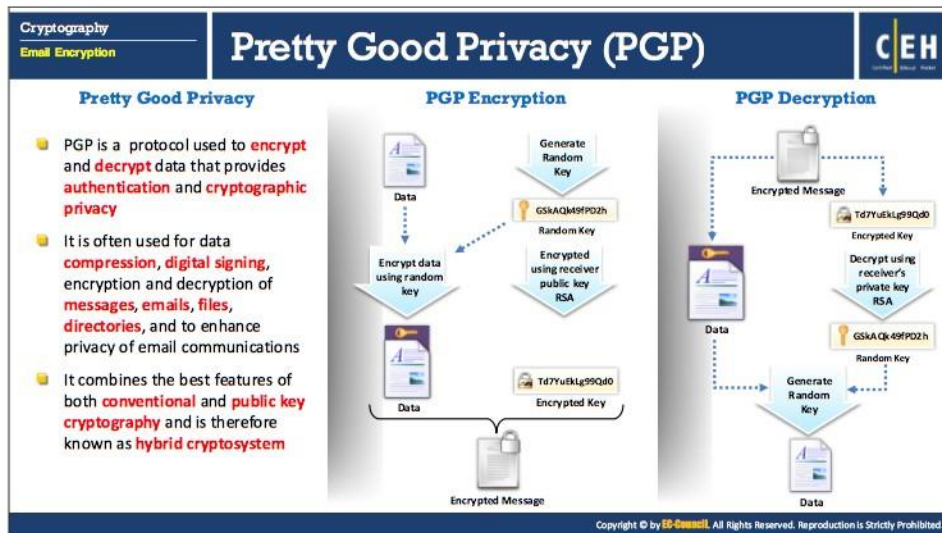
Features:

- Key rotation and versioning
- Safe default algorithms, modes, and key lengths
- Automated generation of initialization vectors and ciphertext signatures
- Java, Python, and C++ implementations
- International support in Java

An illustrative use case:

Suppose an application needs to encrypt a URL parameter value with a symmetric key. Normally, a developer would need to decide which algorithm to use, the key length to use, the mode of operation, how to handle initialization vectors, how to rotate keys, and how to sign ciphertexts. Keyczar simplifies these choices allowing a Java developer to use an existing keyset of it by making a call to the following:

```
Crypter crypter = new Crypter("/path/to/your/keys");  
String ciphertext = crypter.encrypt("Secret message");
```



Pretty Good Privacy (PGP)

PGP (Pretty Good Privacy) is a protocol used to encrypt and decrypt data that provides authentication and cryptographic privacy. It is often used for data compression, digital signing, encryption and decryption of messages, emails, files, directories, and to enhance privacy of email communications. The algorithm used for message encryption is RSA for key transport and IDEA for bulk-message encryption. PGP uses RSA for computing digital signatures and MD5 for computing message digests.

PGP combines the best features of both conventional (about 1,000 times faster than public-key encryption) and public-key cryptography (solution to key distribution and data transmission issues) and is therefore known as hybrid cryptosystem.

PGP is used for:

- Encrypting a message or file prior to transmission so that only the recipient can decrypt and read it
- Clear signing of the plaintext message to ensure the authenticity of the sender
- Encrypting stored computer files so that no one other than the person who encrypted them can decrypt them
- Deleting files, rather than just removing them from the directory or folder
- Data compression for storage or transmission

How PGP Works?

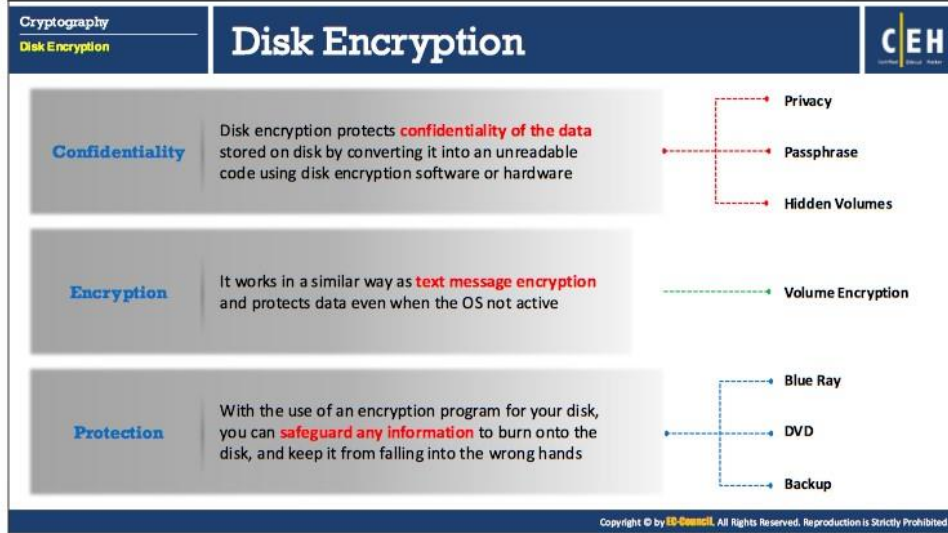
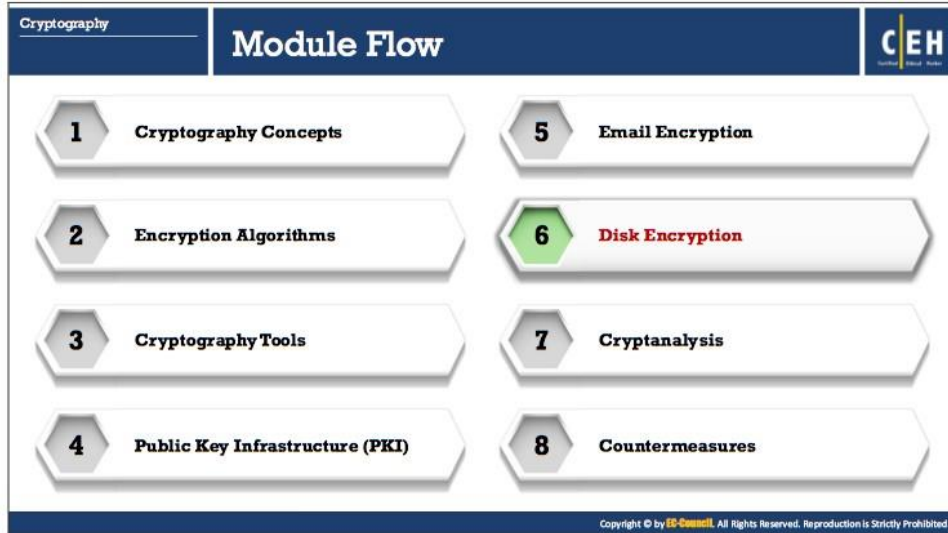
▪ PGP Encryption

- When a user encrypts data with PGP, PGP first compresses the data.
Compressing data reduces patterns in the plaintext that could be exploited by most of the cryptanalysis techniques to crack the cipher, thus prominently increasing resistance to cryptanalysis.
- PGP then creates a random key (GSkAQk49fPD2h) that is a one-time-only secret key.
- PGP uses the random key generated to encrypt the plain text resulting in cipher text.
- Once data is encrypted, random key is encrypted with the recipient's public key.
- Public key-encrypted random key (Td7YuEkLg99Qd0) is sent along with the cipher text to the recipient.

▪ PGP Decryption

- Decryption works in the reverse.
- Recipient's copy of PGP uses his or her private key, instead of the public key to recover the temporary random key.
- PGP then uses the recovered random key to decrypt the conventionally-encrypted cipher text.

Note: Each step of PGP encryption process (hashing, data compression, symmetric key cryptography, and public key cryptography) uses one of the several supported algorithms.



Disk Encryption

Disk encryption encrypts every bit of data stored on a disk or a disk volume, thus preventing illegal access to data storage. This section deals with disk encryption concepts and various disk encryption tools.

Disk encryption is a technology, which protects the confidentiality of the data stored on disk by converting it into an unreadable code using disk encryption software or hardware, thus

preventing unauthorized users from accessing it. Disk encryption provides confidentiality and privacy using passphrases and hidden volumes.

Disk encryption works in a manner similar to text-message encryption and protects data even when the OS is not active. By using an encryption program for the user's disk (Blue Ray, DVD, Backup), the user can safeguard any or all information burned onto the disk and save it from falling into the wrong hands. Disk-encryption software scrambles the information burned on the disk into an illegible code. It is only after decryption of the disk information that one can read and use it.

Disk encryption is useful when the user needs to send sensitive information through the mail. In addition, disk encryption can prevent the real-time exchange of information from compromising threats. When the users exchange encrypted information, it minimizes the chances of compromising the information. The only way an attacker could access the information is by decrypting the message. Furthermore, encryption software installed on a user's system ensures the security of the system. Install encryption software on any systems that hold valuable information or the ones exposed to unlimited data transfer.



Disk Encryption Tools

The common goal of disk encryption tools is encrypting a disk partition to provide confidentiality to the information stored on it. Discussed below are some of the disk encryption tools.

- **VeraCrypt**

Source: <https://veracrypt.codeplex.com>

VeraCrypt is a software for establishing and maintaining an on-the-fly-encrypted volume (data storage device). On-the-fly encryption means that data is automatically encrypted right before it is saved and decrypted right after it is loaded, without any user intervention. No data stored on an encrypted volume can be read (decrypted) without using the correct password/keyfile(s) or correct encryption keys. The entire file system is encrypted (e.g., file names, folder names, free space, metadata, etc.).

Files can be copied to and from a mounted VeraCrypt volume just like they are copied to/from any normal disk (for example, by simple drag-and-drop operations). Files are automatically being decrypted on the fly (in memory/RAM) while they are being read or copied from an encrypted VeraCrypt volume. Similarly, files that are being written or copied to the VeraCrypt volume are automatically being encrypted on the fly (right before they are written to the disk) in RAM.

- **Symantec Drive Encryption**

Source: <https://www.symantec.com>

Symantec Drive Encryption (formerly PGP Whole Disk Encryption) provides organizations with complete, transparent drive encryption for all data (user files, swap

files, system files, hidden files, etc.) on laptops, desktops, and removable media. It protects data from unauthorized access, providing strong security for intellectual property, customer, and partner data. Symantec Encryption Management Server can centrally manage the protected systems, as it simplifies deployment, policy creation, distribution, and reporting.

- **BitLocker Drive Encryption**

Source: <https://docs.microsoft.com>

BitLocker Drive Encryption is a data protection feature that integrates with the operating system and addresses the threats of data theft or exposure from lost, stolen, or inappropriately decommissioned computers. BitLocker provides the most protection when used with a Trusted Platform Module (TPM) version 1.2 or later. The TPM is a hardware component installed in many newer computers by the computer manufacturers. It works with BitLocker to help protect user data and to ensure that a computer has not been tampered with while the system was offline.



FIGURE 20.3: Screenshot of BitLocker Drive Encryption

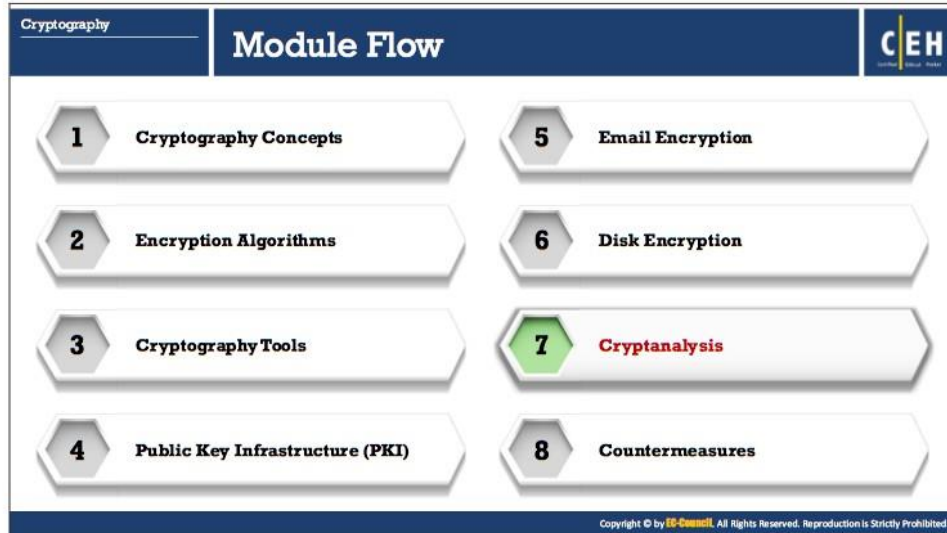
The infographic is titled "Disk Encryption Tools" and is part of a "Cryptography" section. It lists 15 different disk encryption tools arranged in a 5x3 grid. Each tool is represented by a small icon, the product name, and its website URL. The tools listed are: Gillsoft Full Disk Encryption, Full Disk Encryption Software, PocketCrypt, Endpoint Full Disk Encryption, SafeGuard Encryption, DriveCrypt Plus Pack, Dell Data Protection | Encryption, Alertsec, Rohos Disk Encryption, AxCrypt, DriveCrypt, east-tec SafeBit, Folder Lock, ShareCrypt, and Cryptainer LE. A copyright notice for EC-Council is at the bottom of the infographic.

Tool Name	Website URL
Gillsoft Full Disk Encryption	http://www.gilisoft.in
Full Disk Encryption Software	https://www.winmagic.com
PocketCrypt	http://www.securstar.com
Endpoint Full Disk Encryption	https://www.checkpoint.com
SafeGuard Encryption	https://www.sophos.com
DriveCrypt Plus Pack	http://www.securstar.com
Dell Data Protection Encryption	http://www.dell.com
Alertsec	https://www.alertsec.com
Rohos Disk Encryption	http://www.rohos.com
AxCrypt	https://www.axcrypt.net
DriveCrypt	http://www.securstar.com
east-tec SafeBit	https://www.east-tec.com
Folder Lock	http://www.newsoftwares.net
ShareCrypt	http://www.securstar.com
Cryptainer LE	http://www.cypherix.com

Disk Encryption Tools

Some of the additional disk encryption tools are listed below:

- Gillsoft Full Disk Encryption (<http://www.gilisoft.in>)
- Endpoint Full Disk Encryption (<https://www.checkpoint.com>)
- Dell Data Protection | Encryption (<http://www.dell.com>)
- AxCrypt (<https://www.axcrypt.net>)
- Folder Lock (<http://www.newsoftwares.net>)
- Full Disk Encryption Software (<https://www.winmagic.com>)
- SafeGuard Encryption (<https://www.sophos.com>)
- Alertsec (<https://www.alertsec.com>)
- DriveCrypt (<http://www.securstar.com>)
- ShareCrypt (<http://www.securstar.com>)
- PocketCrypt (<http://www.securstar.com>)
- DriveCrypt Plus Pack (<http://www.securstar.com>)
- Rohos Disk Encryption (<http://www.rohos.com>)
- east-tec SafeBit (<https://www.east-tec.com>)
- Cryptainer LE (<http://www.cypherix.com>)
- DiskCryptor (<https://diskcryptor.net>)



Cryptanalysis

Attackers may implement various cryptography attacks in order to evade security of a cryptographic system by exploiting vulnerabilities in a code, cipher, cryptographic protocol, or key management scheme. This process is known as cryptanalysis.

Cryptanalysis is the study of ciphers, cipher text, or cryptosystems with the ability to identify vulnerabilities in them that allows to extract plaintext from the ciphertext even if the cryptographic key or algorithm used to encrypt the plaintext is unknown.

This section deals with various cryptography attacks that an attacker uses to compromise cryptographic system and various cryptanalysis techniques and tools that help in breaching cryptographic security.

The infographic is titled "Cryptanalysis Methods" and is divided into three columns. The left column, "Linear Cryptanalysis", lists: it is commonly used on block ciphers; it is a known plaintext attack using a linear approximation; given enough pairs of plaintext and corresponding ciphertext, bits of information about the key can be obtained; and for example, with the 56 bit DES key brute force could take up to 256 attempts. The middle column, "Differential Cryptanalysis", lists: it is a form of cryptanalysis applicable to symmetric key algorithms; it is the examination of differences in an input and how that affects the resultant difference in the output; it originally worked only with chosen plaintext; and it can also work with known plaintext and ciphertext only. The right column, "Integral Cryptanalysis", lists: this attack is useful against block ciphers based on substitution-permutation networks; integral analysis, for block size b, holds b-k bits constant and runs the other k through all 2k possibilities; and for k=1, this is just differential cryptanalysis, but with k>1 it is a new technique. A copyright notice for EC-Council is at the bottom.

Cryptanalysis Methods

Linear Cryptanalysis

Linear cryptanalysis is based on finding affine approximations to the action of a cipher. It is commonly used on block ciphers. This technique was invented by Mitsuru Matsui. It is a known plaintext attack and uses a linear approximation to describe the behavior of the block cipher. Given enough pairs of plaintext and corresponding ciphertext, bits of information about the key can be obtained. Obviously, the more pairs of plaintext and ciphertext one has, the greater the chance of success.

Remember cryptanalysis is an attempt to crack cryptography. For example, with the 56-bit data encryption standard (DES) **key brute force could take up to 256 attempts**. Linear cryptanalysis will take 2^{43} known plaintexts. This is better than brute force, but still impractical for most situations. The math may be a bit complex for beginning cryptographers, but let's look at the basics of it.

With this method, a linear equation expresses the equality of two expressions which consist of binary variables XORed. For example, in the following equation, XORs sum of the first and third plaintext bits, and the first ciphertext bit is equal to the second bit of the key:

$$P_1 \oplus P_3 \oplus C_1 = K_2$$

You can use this method to slowly re-create the key that was used.

After doing this for each bit, you will have an equation of the form:

$$P_{i1} \oplus P_{i2} \oplus \dots \oplus C_{j1} \oplus C_{j2} \oplus \dots = K_{k1} \oplus K_{k2} \oplus \dots$$

You can then use **Matsui's Algorithm 2**, using known plaintext-ciphertext pairs, to guess at the values of the key bits involved in the approximation. For each set of values of the key bits on the right-hand side (referred to as a partial key), count how many times the approximation holds true over all the known plaintext-ciphertext pairs; call this count T . The partial key whose T has the greatest absolute difference from half the number of plaintext-ciphertext pairs is designated as the most likely set of values for those key bits.

- **Differential Cryptanalysis**

Differential cryptanalysis is a form of cryptanalysis applicable to symmetric key algorithms. This was invented by Eli Biham and Adi Shamir. Essentially, it is the examination of differences in an input and how that affects the resultant difference in the output. It originally worked only with chosen plaintext. It could also work with known **plaintext** and **ciphertext**.

- **Integral Cryptanalysis**

Integral cryptanalysis was first described by Lars Knudsen. This attack is particularly useful against block ciphers based on substitution-permutation networks as an extension of differential cryptanalysis. Differential analysis looks at pairs of inputs that differ in only one-bit position, with all other bits being identical. Integral analysis, for block size b , holds $b-k$ bits constant and runs the other k through all 2^k possibilities. For $k = 1$, this is just differential cryptanalysis, but with $k > 1$, it is a new technique.

Cryptography		Code Breaking Methodologies		CEH	
Cryptanalysis					
		■ One can measure the strength of an encryption algorithm using various code-breaking techniques			
Brute-Force		Cryptography keys are discovered by trying every possible combination			
Frequency Analysis		<ul style="list-style-type: none">It is the study of the frequency of letters or groups of letters in a ciphertextIt works on the fact that, in any given stretch of written language, certain letters and combinations of letters occur with varying frequencies			
Trickery and Deceit		It involves the use of social engineering techniques to extract cryptography keys			
One-Time Pad		A one-time pad contains many non-repeating groups of letters or number keys, which are chosen randomly			

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Code Breaking Methodologies

One can measure the strength of an encryption algorithm using various code-breaking techniques, some of which are as follows:

- **Brute Force**

Code-breakers or cryptanalysts work to recover the plaintext of a message without knowing the required key in advance. They may first try to recover the key, or they may go after the message itself. A common cryptanalytic technique is a brute-force attack, or exhaustive search, in which the keys are determined by trying every possible combination of characters.

The efficiency of a brute-force attack depends on the hardware configuration. Use of faster processors means testing more keys per second. Cryptanalysts carried out a successful brute-force attack on a DES encryption method that effectively made DES obsolete.

- **Frequency Analysis**

Frequency analysis is the study of the frequency of letters or groups of letters in a ciphertext. Frequency analysis of letters and words is another method used to attack ciphers. It works on the fact that, in any given stretch of written language, certain letters and combinations of letters occur with varying frequencies. This technique examines the number of times that a particular symbol comes up in a ciphertext. For example, the letter “e” is a common letter in the English language. If the letter “k” appears commonly in a ciphertext, it can be reasonably concluded that “k” in the encrypted language is equivalent to “e” in English.

Encrypted source code is more vulnerable to these types of attacks because words like **"#define," "struct," "else,"** and **"return"** are repeated frequently in code. Sophisticated cryptosystems are required to maintain the security of messages against frequency analysis.

- **Trickery and Deceit**

Trickery and Deceit requires a high level of mathematical and cryptographic skills. It involves the use of social engineering techniques to extract cryptography keys.

Example: It is fairly easy to decrypt an entire message if the user knows some of its content.

An attacker can use social engineering techniques to trick or bribe someone to encrypt and send a known message, which, when intercepted, could then be easily decrypted using standard cryptanalysis techniques.

- **One-Time Pad**

One can crack any cipher if provided with sufficient time and resources. However, there is an exception called a one-time pad, which the users assume to be unbreakable even with infinite resources.

A one-time pad contains mostly a non-repeating set of letters or numbers, which the system chooses randomly. The user writes them on small sheets of paper and then pastes them together in a pad.

Example of One-time pad usage:

Sender encrypts only one plaintext character using each key letter on the pad, and the receiver decrypts each letter of the ciphertext using an identical pad. Once the letter uses a page, he or she tears it off the pad and securely discards it, thus it got the name One-time Pad.

Drawback:

The key length is same as that of the message, thus making it impossible to encrypt and send large messages.

Cryptography Cryptanalysis	Cryptography Attacks	CEH
<p>🔹 Cryptography attacks are based on the assumption that the cryptanalyst has access to the encrypted information</p>		
Ciphertext-only Attack	Attacker has access to the cipher text; goal of this attack to recover encryption key from the ciphertext	
Adaptive Chosen-plaintext Attack	Attacker makes a series of interactive queries , choosing subsequent plaintexts based on the information from the previous encryptions	
Chosen-plaintext Attack	Attacker defines his own plaintext , feeds it into the cipher, and analyzes the resulting ciphertext	
Related-key Attack	Attacker can obtain ciphertexts encrypted under two different keys , and this attack is useful if the attacker can obtain the plaintext and matching cipher text	
Dictionary Attack	Attacker constructs a dictionary of plaintext along with its corresponding ciphertext that he/she has learnt for a certain period of time	
Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.		

Cryptography Cryptanalysis	Cryptography Attacks (Cont'd)	CEH
Known-plaintext Attack	Attacker has knowledge of some part of the plain text ; using this information, the key used to generate ciphertext is deduced so as to decipher other messages	
Chosen-ciphertext Attack	Attacker obtains the plaintexts corresponding to an arbitrary set of ciphertexts of his own choosing	
Rubber Hose Attack	Extraction of cryptographic secrets (e.g. the password to an encrypted file) from a person by coercion or torture	
Chosen-key Attack	Attacker usually breaks an n bit key cipher into 2^{n/2} number of operations	
Timing Attack	It is based on repeatedly measuring the exact execution times of modular exponentiation operations	
Man-in-the-middle Attack	Attacker performs this attack on the public key cryptosystems where key exchange is required before communication takes place	
Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.		

Cryptography Attacks

Attackers conduct cryptography attacks assuming that the cryptanalyst has access to the encrypted information. Cryptography attack or cryptanalysis involves study of various principles and methods of decrypting the ciphertext back to the plaintext without knowledge of the key.

The following are various types of cryptography attacks:

- **Ciphertext-only Attack**

Ciphertext-only is less effective but much more likely for the attacker. The attacker only has access to a collection of cipher texts. This is much more likely than known plaintext but also the most difficult. The attack is completely successful if the corresponding plaintexts can be deduced, or even better, the key. The ability to obtain any information at all about the underlying plaintext is still considered a success. So what does the attacker do with the ciphertexts he has accumulated? You can analyze them for patterns, trying to find something that would give you a hint as to the key that was used to crack them. Often, the result of this attack is just a partial break and not a complete break.

- **Adaptive Chosen-plaintext Attack**

In this type of attack, an attacker has a complete access to the plaintext message including its encryption, and he/she can also modify the content of the message by making series of interactive queries, choosing subsequent plaintext blocks based on the information from the previous encryption queries and functions. To perform this attack, an attacker needs to interact with the encryption device.

- **Chosen-plaintext Attack**

Chosen plaintext attack is a very effective type of cryptanalysis attack. In this attack, the attacker obtains the ciphertexts corresponding to a set of plaintexts of his own choosing. This can allow the attacker to attempt to derive the key used and thus decrypt other messages encrypted with that key. Basically, since the attacker knows the plaintext and the resultant ciphertext, he has a lot of insight into the key used. This technique can be difficult but is not impossible.

- **Related-Key Attack**

The related-key attack is similar to the chosen plaintext attack, except the attacker can obtain ciphertexts encrypted under two different keys. This is actually a very useful attack if you can obtain the plaintext and matching ciphertext. The attack requires that the differing keys be closely related, for example, in a wireless environment where subsequent keys might be derived from previous keys. Then, while the keys are different, they are close. Much like the cipher text-only attack, this one is most likely only going to yield a partial break.

- **Dictionary Attack**

In this attack, the attacker constructs a dictionary of plaintext along with its corresponding ciphertext that he/she has analyzed and obtained for a certain period of time. After building the dictionary, if an attacker obtains the ciphertext, he/she uses the already built dictionary to find the corresponding plaintext. Attackers use this technique to decrypt keys, passwords, passphrases, and ciphertext.

- **Known-plaintext Attack**

In this attack, the only information available to the attacker is some plaintext blocks along with corresponding ciphertext and algorithm used to encrypt and decrypt the text. Using this information, the key used to generate ciphertext that is deduced so as to decipher other messages. This attack works on block ciphers and is an example of linear cryptanalysis. The known plaintext blocks are generated using a series of intelligent guesses and logic and not through accessing the plaintext over a channel.

- **Chosen-ciphertext Attack**

Attacker obtains the plaintexts corresponding to an arbitrary set of ciphertexts of his own choosing. Using this information, the attacker tries to recover the key used to encrypt the plaintext. To perform this attack, the attacker must have access to communication channel between the sender and the receiver.

There are two variants of this attack:

- Lunchtime or Midnight Attack: In this attack, the attacker can have access to the system for only a limited amount of time or can access only few plaintext-ciphertext pairs.
- Adaptive Chosen-ciphertext Attack: In this attack, the attacker selects a series of ciphertexts and then observes the resulting plaintext blocks.

- **Rubber Hose Attack**

Attackers extract cryptographic secrets (e.g. the password to an encrypted file) from a person by coercion or torture. Generally people under pressure cannot maintain security, and they will reveal secret or hidden information. Attackers torture concerned person to reveal secret keys or passwords used to encrypt the information.

- **Chosen-key Attack**

In this type of attack, an attacker not only breaks a ciphertext but also breaks into a bigger system, which is dependent on that ciphertext. Attacker usually breaks an n bit key cipher into $2^{n/2}$ number of operations. Once, an attacker breaks the cipher, he gets access to the system, he can control the whole system, access the confidential data, and can perform further attacks.

- **Timing Attack**

It is based on repeatedly measuring the exact execution times of modular exponentiation operations. Attacker tries to break the ciphertext by analyzing the time taken to execute the encryption and decryption algorithm for various inputs. In a computer, the time taken to execute a logical operation may vary based on the input given. An attacker by giving varying inputs tries to extract the plaintext.

- **Main-in-the-Middle Attack**

This attack is performed against a cryptographic protocol. Here, an attacker intercepts the communication between a client and server and negotiates the cryptographic parameters. Using this attack, an attacker can decrypt the encrypted content and obtain confidential information such as system passwords. An attacker can also inject commands that can modify the data in transit. Attacker usually performs MITM attack on the public key cryptosystems where key exchange is required before communication takes place.

Attacker performs this attack on the public key cryptosystems where key exchange is required before communication takes place.

Cryptography Cryptanalysis		Brute-Force Attack			CEH
Attack Scheme	Defeating a cryptographic scheme by trying a large number of possible keys until the correct encryption key is discovered				
Brute-Force Attack	Brute-force attack is a high resource and time intensive process , however, more certain to achieve results				
Success Factors	Success of brute-force attack depends on length of the key, time constraint, and system security mechanisms				
Power/Cost	40 bits (5 char)	56 bit (7 char)	64 bit (8 char)	128 bit (16 char)	
\$ 2K (1 PC. Can be achieved by an individual)	1.4 min	73 days	50 years	10^20 years	
\$ 100K (this can be achieved by a company)	2 sec	35 hours	1 year	10^19 years	
\$ 1M (Achieved by a huge organization or a state)	0.2 sec	3.5 hours	37 days	10^18 years	
Estimate Time for Successful Brute-force Attack					
Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.					

Brute-Force Attack

It is very difficult to crack cryptographic systems, as they have no practical weaknesses to exploit; however, it is not impossible. Cryptographic systems use cryptographic algorithms to encrypt a message. These cryptographic algorithms use a key to encrypt or decrypt messages. In cryptography, this key is the important parameter that specifies the transformation of plain text to ciphertext and vice versa. If you are able to guess or find the key used for decryption, then you can decrypt the messages and read it in clear text; 128-bit keys are common and considered strong. From security perspectives to avoid guessing of the key, the cryptographic systems use randomly generated keys. This makes you put a lot of effort in guessing the key. However, you still have a choice to determine the key used for encryption or decryption.

You can attempt to decrypt a message using all possible keys, until you discover the key used for encryption. This method of discovering a key is called a brute-force attack. However, to do so requires a huge amount of processing power. It is a high resource and time intensive process. For any non-flawed protocol, the average time needed to find the key in a brute-force attack depends on the length of the key. If its key length is short, then it will take less time to find the key; if longer, it will take more time. A brute-force attack will be successful if and only if the attacker has enough time to discover the key. However, the time required is relative to the length of the key.

The difficulty of a brute-force attack depends on various issues, such as:

- Length of the key
- The numbers of possible values each component of the key can have
- The time it takes to attempt each key
- If there is any mechanism, which locks the attacker out after a certain number of failed attempts

For example, if a system could brute force a DES 56-bit key in one second, then for an AES 128-bit key it takes approximately 149 trillion years to brute force. To perform a brute-force attack, the attacker needs double the time for every additional bit of key length; the reason behind it is that the number of keys double with increase of a bit.

A brute-force attack is, however, more certain to achieve results.

Power/Cost	40 bits (5 char)	56 bit (7 char)	64 bit (8 char)	128 bit (16 char)
\$ 2K (1 PC. Can be achieved by an individual)	1.4 min	73 days	50 years	10 ²⁰ years
\$ 100K (this can be achieved by a company)	2 sec	35 hours	1 year	10 ¹⁹ years
\$ 1M (Achieved by a huge organization or a state)	0.2 sec	3.5 hours	37 days	10 ¹⁸ years

TABLE 20.3: Estimate time for successful brute-force attack

Cryptography
Cryptanalysis


Birthday Attack

CEH

- A birthday attack is a name used to refer to a class of brute-force attacks against cryptographic hashes that makes the brute forcing easier
 - Birthday paradox:** The probability that two or more people in a group of 23 share the same birthday is greater than half

Birthday Paradox

- How many people do you need to have a high likelihood that **2 share the same birth day** (i.e., same day and month, but not the same year)
- There are **365 days** in a year, so you might think at least half of that or 182 people, but it is actually only **23!**
- The basic idea is this: How **many people** would you need to have in a room to have a **strong likelihood** that two would have the **same birthday** (month and day, but not the same year)
- Obviously, if you put **367 people** in a room, at least 2 of them must have the same birthday, since there are only 365 days in a year, plus one more in a leap year
- The paradox is not asking how many people you need to **guarantee a match**, just how many you need to have a strong probability
- Even with 23 people in the room, you have a **50 percent chance** that 2 will have the same birthday



Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Birthday Attack

A birthday attack is a name used to refer to a class of brute-force attacks against cryptographic hashes that makes the brute forcing easier. The birthday attack depends on birthday paradox. Birthday paradox is the probability that two or more people in a group of 23 share the same birthday is greater than $\frac{1}{2}$.

Birthday Paradox

For example, how many people do you need to have a high likelihood that 2 share the same birthday (i.e. same day and month, not same year). There are 365 days a year, so you might think at least $\frac{1}{2}$ that or 182 people, but it is actually only 23!

The basic idea is this: How many people would you need to have in a room to have a strong likelihood that two would have the same birthday (month and day, but not year). Obviously, if you put 367 people in a room, at least 2 of them must have the same birthday, since there are only 365 days in a year, plus one more in a leap year. The paradox is not asking how many people you need to guarantee a match; just how many you need to have a strong probability. Even 23 people in the room, you have a 50 percent chance that 2 will have the same birthday.

Cryptography
Cryptanalysis

Birthday Paradox: Probability

CEH
Certified Ethical Hacker

- 01** The probability that the first person does not share a birthday with any previous person is **100 percent**, because there are no previous people in the set. This can be written as **365/365**
- 02** The second person has only one preceding person, and the **odds that the second person has a birthday** different from the first are **364/365**
- 03** The third person might share a birthday with two preceding people, so the odds of having a birthday from either of the two preceding people are **363/365**
- 04** Because each of these are independent, we can compute the probability as follows:
 $365/365 * 364/365 * 363/365 * 362/365 \dots * 342/365$
(342 is the probability of the 23rd person shares a birthday with a preceding person)
- 05** When we convert these to decimal values, it yields (truncating at the third decimal point):
 $1 * 0.997 * 0.994 * 0.991 * 0.989 * 0.986 * \dots * 0.936 = 0.49$, or **49 percent**
- 06** This **49 percent is the probability** that **23 people** will not have any **birthdays** in common; thus there is a **51 percent** (better than even odds) chance that **2** of the **23** will have a **birthday** in common

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Birthday Paradox: Probability

The probability that the first person does not share a birthday with any previous person is 100 percent, because there are no previous people in the set. That can be written as 365/365. The second person has only one preceding person, and the odds that the second person has a birthday different from the first are 364/365. The third person might share a birthday with two preceding people, so the odds of having a birthday from either of the two preceding people are 363/365. Because each of these are independent, we can compute the probability as follows: $365/365 * 364/365 * 363/365 * 362/365 \dots * 342/365$ (342 is the probability of the 23rd person shares a birthday with a preceding person). When we convert these to decimal values, it yields (truncating at the third decimal point): $1 * 0.997 * 0.994 * 0.991 * 0.989 * 0.986 * \dots * 0.936 = 0.49$ or 49 percent. This 49 percent is the probability that 23 people will not have any birthdays in common; thus, there is a 51 percent (better than even odds) chance that 2 of the 23 will have a birthday in common.

The idea of the Birthday Attack is to attempt to find a collision for a given hash. Now assume the hash is MD5, with a 128-bit output. You would have to try 2^{128} possible hashes to guarantee a collision. That is a very large number. In decimal notation that is 3.4028236692093846346337460743177e+38

Now from the birthday paradox we know that we actually need about $1.174\sqrt{2^{128}}$ or 21656477542535013597.184. Still it is a very large number but many orders of magnitude smaller.

Cryptography
Cryptanalysis

Meet-in-the-Middle Attack on Digital Signature Schemes

CEH

- The attack works by **encrypting from one end** and **decrypting from the other end**, thus meeting in the middle
- It can be used for **forging signatures** even on digital signatures that use multiple-encryption schemes

Copyright © by **EC-Council**. All Rights Reserved. Reproduction is Strictly Prohibited.

Meet-in-the-Middle Attack on Digital Signature Schemes


A meet-in-the-middle attack is the best attack method for cryptographic algorithms using multiple keys for encryption. This attack reduces the number of brute force permutations needed to decode text encrypted by more than one key and conducted mainly for forging signatures on mixed type digital signatures. A meet-in-the-middle attack uses space-time trade-off; it is a birthday attack, because it exploits the mathematics behind the birthday paradox. It takes less time than an exhaustive attack. It is called a meet-in-the-middle attack, because it works by encrypting from one end and decrypting from the other end, thus meeting “in the middle.”

In the meet-in-the-middle attack, the attacker uses a known plaintext message. The attacker has access to both the plaintext as well as the respective encrypted text. It is used by attackers for forging signatures even on digital signatures that use multiple-encryption scheme.

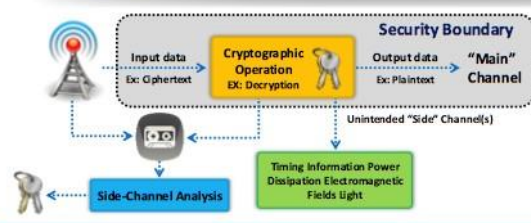
Consider an example where the plain text is “John” and the resulting double DES encrypted message is “AvBr.” To recover both the keys (i.e., key1 and key2) used for encryption, the attacker performs a brute-force attack on key1 using all 2^{56} **different single DES** possible keys to encrypt the plaintext of “John” and saves each key and the resulting intermediate ciphertext in a table. The attacker conducts brute force on key2, decrypting “AvBr” up to 2^{56} **times**. The attack is successful when the second brute-force attack gives the same result as that of the intermediate ciphertext present in the ciphertext table after the first brute-force attack. Once the attacker finds a match, he/she can determine both keys and complete the attack. At most, this attack takes 2^{56+} or a maximum of 2^{57} **total operations**. This enables the attacker to gain access to the data easily when compared with the Double DES.

Cryptography
Cryptanalysis

Side Channel Attack



- Side channel attack is a **physical attack** performed on a cryptographic device/cryptosystem to gain sensitive information
- Cryptography is generally part of the hardware or software that runs on physical devices such as semi-conductors (includes resistor, transistor, and so on)
- These physical devices are affected by various **environmental factors** that include power consumption, electro-magnetic field, light emission, timing and delay, and sound
- In side channel attack, an attacker **monitors these channels (environmental factors)** and tries to acquire the information useful for cryptanalysis



- Assume that an encrypted data is to be decrypted and displayed as a plain text, inside a **trusted zone**
- At the time of decryption in a cryptosystem, **physical environmental factors** such as timing, power dissipation, etc., acting on the components of a computer are recorded by an attacker
- The attacker analyzes this information in an attempt to **gain useful information** for cryptanalysis

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Side Channel Attack

A side channel attack is a physical attack performed on a cryptographic device/cryptosystem to gain sensitive information. Cryptography is generally part of the hardware or software that runs on physical devices such as semi-conductors (includes resistor, transistor, and so on) those interact with and affect various environmental factors that include:

- **Power Consumption**
Reveals operations that take place and parameters involved. It is applicable only to hardware cryptosystems. Power consumption analysis is of two types:
 - **Simple Power Analysis (SPA):** Provides information regarding the instruction being executed at a certain time and the values of input and output
 - **Differential Power Analysis (DPA):** It does not require the knowledge of algorithm implementation details, exploits statistical methods in the analysis process
- **Electromagnetic Field**
Computer components often generate electromagnetic radiations. By measuring the variations of the electromagnetic field over the chip surface, an attacker could predict their correlation to the underlying computation and data may deduce some amount of valuable information about this computation and data.
- **Light Emission**
Kuhn found that the average luminosity of a Cathode Ray Tube (CRT) diffuse reflection of a wall is enough to reconstruct the signal displayed on the CRT. Thus, an attacker can gather ample information by reading the signals that a trusted computing platform's optical output channels emit.

According to Loughry and Umphress, one can deduce the data a computer is processing based on optical radiation emitted from its LED (light-emitting diode) status indicators.

- **Timing and Delay**

The systems often compute the cryptographic algorithms without time consistency due to performance optimizations. If such computation involves secret data, then the variations in time can leak the information. Here the attacker analyzes the time taken by the cryptographic device to process each message in order to discover the secret parameters.

- **Sound**

Acoustic attacks exploit the sound produced during a computation. These acoustic emissions are from keyboards and computing components (e.g., CPU, memory)






In a side channel attack, an attacker monitors these channels (environmental factors) and tries to acquire the information useful for cryptanalysis. The information collected in this process is termed as side channel information. Side channel attacks do not relate with traditional/theoretical form of attacks like brute force attack. The concept of the side channel attack depends on the way systems implement cryptographic algorithms, rather than the algorithm itself.

Side-channel-attack mitigation techniques include:

- Use Differential Power Analysis (DPA) proof protocols with delimited side-channel leakage characteristics and update keys before leakage accumulation is significant
- Use Fixed-time algorithms (i.e., no data-dependent delays)
- Mask and blind algorithms using random nonces
- Implement differential matching techniques to minimize net data-dependent leakage from logic-level transitions
- Pre-charge registers and busses to remove leakage signatures from predictable data transitions
- Add amplitude or temporal noise to reduce the attacker's signal-to-noise ratio

Side Channel Attack – Scenario

Assume that an encrypted data is to be decrypted and displayed a plain text inside a trusted zone. At the time of decryption in a cryptosystem, physical environmental factors such as timing, power dissipation etc., acting on the components of a computer, are recorded by an attacker. The attacker analyzes this information in an attempt to gain useful information for cryptanalysis.

Cryptography		Hash Collision Attack		CEH
Cryptanalysis				
<input type="radio"/>	Hash collision attack is performed by finding two different input messages that result into same hash output			
<input type="radio"/>	This allows an attacker to perform cryptanalysis by exploiting the digital signature to decode the encoded data			
<input type="radio"/>	SHA-1 algorithm converts input message into constant length of unstructured strings of numbers and alphabets, which act as a finger print for the sent file			
<input type="radio"/>	Attacker is able to forge victim's digital signature of message a1 on the incorrect message a2			
<input type="radio"/>	Once the attacker is able to detect any collisions in the hash, then he/she tries to identify more collisions by concatenating data to the matching messages			

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Hash Collision Attack

Hash collision attack is performed by finding two different input messages that result into same hash output. For example, in a hash collision attack the values “**hash(a1) = hash(a2)**”, where a1 and a2 are some random messages. Since, the algorithm itself randomly selects these messages, attackers have no role on the content of these messages. This allows the attacker to perform cryptanalysis by exploiting the digital signature to decode the encoded data.

One of the most popular hash function is SHA-1, which is widely used as a digital signature algorithm. SHA-1 algorithm converts input message into constant length of unstructured strings of numbers and alphabets, which act as a finger print for sent file. Therefore, the attacker tries to identify similar hashed output to get the digital signatures of the victim. This allows the attacker to forge victim's digital signature of message a1 on the message a2.

Once the attacker is able to detect any collisions in the hash then he/she tries to identify more collisions by concatenating data to the matching messages.

This is a personal copy of devinwong.

Cryptography
Cryptanalysis

DUHK Attack

CEH

- 1 DUHK (Don't Use Hard-Coded Keys) is a **cryptographic vulnerability** that allows an attacker to **obtain encryption keys** used to secure VPNs and web sessions
- 2 This attack mainly affects any hardware/software using ANSI X9.31 **Random Number Generator (RNG)**
- 3 The **pseudorandom number generators (PRNGs)** generate random sequences of bits based on the initial secret value called a seed and the current state
- 4 Both the factors are the key issues of DUHK attack as any attacker could combine ANSI X9.31 with the hard-coded seed key **to decrypt the encrypted data** sent or received by that device
- 5 Using this attack, attackers identify encryption keys and **steal confidential information** such as critical business data, user credentials, credit card details, etc.

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

DUHK Attack

DUHK (Don't Use Hard-Coded Keys) is a cryptographic vulnerability that allows attackers to obtain encryption keys used to secure VPNs and web sessions. This attack mainly affects any hardware/software using ANSI X9.31 Random Number Generator (RNG). The Pseudorandom number generators (PRNGs) generate random sequences of bits based on the initial secret value called a seed and the current state. The PRNG algorithm generates cryptographic keys that are used to establish a secure communication channel over VPN network. In some cases, the seed key is hardcoded into the implementation. Both the factors are the key issues of DUHK attack as any attacker could combine ANSI X9.31 with the hard coded seed key to decrypt the encrypted data sent or received by that device.

Man-in-the middle attackers use DUHK attack to learn the seed value, observe the current session, and obtain current state value. Using this attack, attackers identify encryption keys and steal confidential information such as critical business data, user credentials, credit card details, etc.

Cryptography
Cryptanalysis

Rainbow Table Attack

CEH

- A rainbow table attack is a type of cryptography attack where an **attacker uses a rainbow table for reversing cryptographic hash functions**
- A rainbow table is a **precomputed table** which **contains word lists** like dictionary files and brute force lists and their hash values
- It uses the **cryptanalytic time-memory trade-off technique** to crack the cryptography, which requires less time than some other techniques
- An attacker computes the hash for a list of possible passwords and compares it to the precomputed hash table (rainbow table). If attackers find a match, **they can crack the password**

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Rainbow Table Attack

A rainbow table attack is a type of cryptography attack where an attacker uses a rainbow table for reversing cryptographic hash functions. A rainbow table attack uses the cryptanalytic time-memory trade-off technique, which requires less time than some other techniques. It uses already-calculated information stored in memory to crack the cryptography. In the rainbow table attack, the attacker creates a table of all the possible passwords and their respective hash values, known as a rainbow table, in advance.

A rainbow table is a pre-computed table, which contains word lists like dictionary files and brute force lists and their hash values. It is a lookup table specially used in recovering a plaintext password from a ciphertext. The attacker uses this table to look for the password and tries to recover it from password hashes.

An attacker computes the hash for a list of possible passwords and compares it to the pre-computed hash table (rainbow table). If attackers find a match, they can crack the password. An attacker captures the hash of a passwords and compare it with the pre-computed hash table. If a match is found, then the password is cracked. It is easy to recover passwords by comparing captured password hashes to the pre-computed tables.

Cryptology
Cryptanalysis

Cryptanalysis Tools

CEH

CrypTool

- Cryptool is a free e-learning program in the area of **cryptology** and **cryptanalysis**
- It consists of e-learning software (CT1, CT2, JCT, and CTO)

CryptoBench
<http://www.oddoris.org>

Cryptool
<http://cryptool.net>

Ganzúa
<http://ganzua.sourceforge.net>

EverCrack
<http://evercrack.sourceforge.net>

AlphaPeeler
<http://alp-peeler.sourceforge.net>

Copyright © by **EC-Council**. All Rights Reserved. Reproduction is Strictly Prohibited.

Cryptanalysis Tools

Attackers use cryptanalysis tools to analyze and break the ciphers. Discussed below are some of the cryptanalysis tools.

- **CrypTool**

Source: <https://www.cryptool.org>

The Cryptool project develops e-learning programs in the area of cryptography and cryptanalysis. It consists of e-learning software (CT1, CT2, JCT, and CTO).

CrypTool 1 (CT1) – It is written in C++ and is a Windows program.

Features:

- Supports classic and modern cryptographic algorithms (encryption and decryption, key generation, secure passwords, authentication, secure protocols, etc.)
- Visualization of several algorithms (Caesar, Enigma, RSA, Diffie-Hellman, digital signatures, AES, etc.)
- Cryptanalysis of several algorithms (Vigenère, RSA, AES, etc.)
- Cryptanalytical measurement methods (entropy, n-grams, autocorrelation, etc.)
- Related auxiliary methods (primality tests, factorization, base64 encoding, etc.)

CrypTool 2 (CT2) – Supports visual programming GUI and execution of cascades of cryptographic procedures. It runs under Windows.

JCryptTool (JCT) –Allows comprehensive cryptographic experimentation on Linux, MAC OS X, and Windows. It also allows users to develop and extend its platform in various ways with their own crypto plug-ins.

CrypTool-Online (CTO) –Runs in a browser and provides a variety of encryption methods and analysis tools.

Some of the additional cryptanalysis tools are listed below:

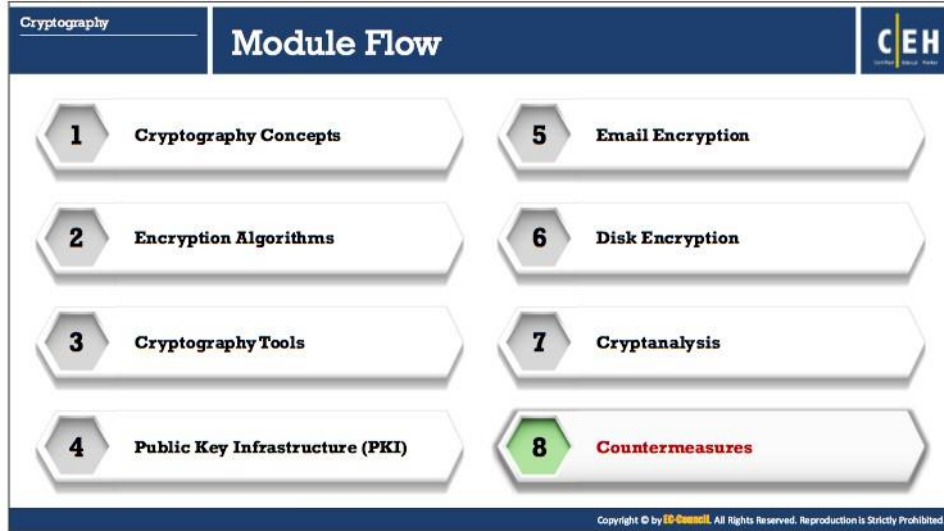
- CryptoBench (<http://www.addario.org>)
- Cryptol (<https://cryptol.net>)
- Ganzúa (<http://ganzua.sourceforge.net>)
- EverCrack (<http://evercrack.sourceforge.net>)
- AlphaPeeler (<http://alphapeeler.sourceforge.net>)
- Mediggo (<https://www.darknet.org.uk>)
- SubCypher (<https://sourceforge.net>)

The screenshot shows a webpage titled "Online MD5 Decryption Tools" with a dark blue header. The header includes "Cryptography" and "Cryptanalysis" on the left, and "CEH" on the right. The main content is a grid of 15 tool cards, each with an icon, a name, and a URL. The tools listed are: MD5 Decoder (https://www.dcode.fr), HashKiller.co.uk (https://hashkiller.co.uk), MD5/Sha1 hash cracker (https://crackstation.net), MD5 Decrypt (http://www.md5decrypt.org), Md5.My-Addr.com (http://md5.my-addr.com), Md5() Encrypt & Decrypt (http://md5decrypt.net), MD5 Decrypter (http://www.md5online.org), cmd5.org (http://www.cmd5.org), MD5Decryption (http://md5decryption.com), MD5Decrypter (https://www.md5decrypter.com), CrackStation (https://crackstation.net), md5hashing (https://md5hashing.net), OnlineHashCrack.com (https://www.onlinehashcrack.com), md5this (http://www.md5this.com), and Online Reverse Hash Lookup (http://reverse-hash-lookup.online-domain-tools.com). A copyright notice at the bottom reads: "Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited."

Online MD5 Decryption Tools

Below are some online MD5 decryption tools that can be used to decrypt the MD5 hash value, thus discovering the original message:

- MD5 Decoder (<https://www.dcode.fr>)
- MD5 Decrypt (<http://www.md5decrypt.org>)
- MD5 Decrypter (<http://www.md5online.org>)
- MD5Decrypter (<https://www.md5decrypter.com>)
- OnlineHashCrack.com (<https://www.onlinehashcrack.com>)
- HashKiller.co.uk (<https://hashkiller.co.uk>)
- Md5.My-Addr.com (<http://md5.my-addr.com>)
- cmd5.org (<http://www.cmd5.org>)
- CrackStation (<https://crackstation.net>)
- md5this (<http://www.md5this.com>)
- MD5/Sha1 hash cracker (<http://crackhash.com>)
- Md5() Encrypt & Decrypt (<http://md5decrypt.net>)
- MD5Decryption (<http://md5decryption.com>)
- md5hashing (<https://md5hashing.net>)
- Online Reverse Hash Lookup (<http://reverse-hash-lookup.online-domain-tools.com>)



Countermeasures

Attackers use various cryptanalysis methods and techniques to break the cryptosystems and steal confidential information that is being transmitted in the network. This section discusses some of the countermeasures to be followed to protect from such type of attacks.

The infographic is titled "How to Defend Against Cryptographic Attacks" and is part of the "Cryptography Countermeasures" series. It lists 12 numbered countermeasures:

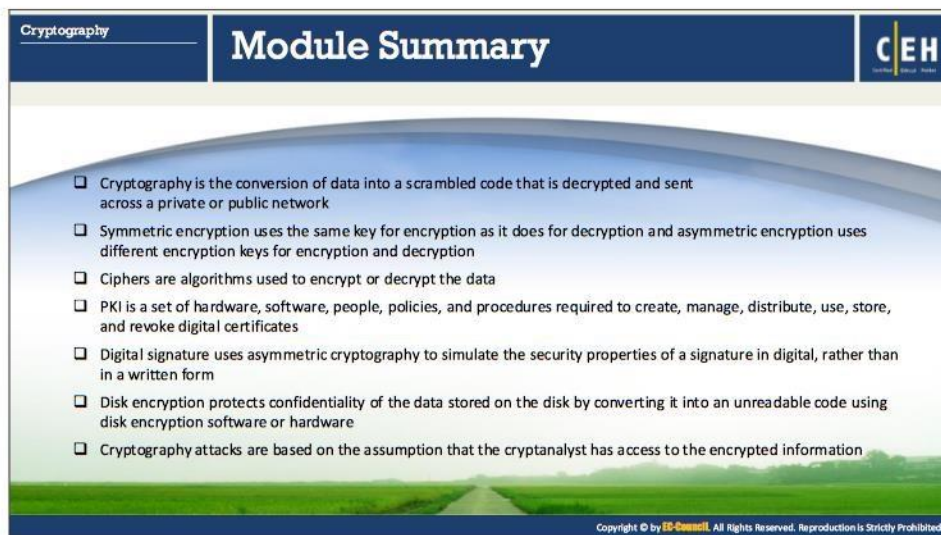
- 1 Access of **cryptographic keys** should be given to the application or to the user directly
- 2 **Intrusion detection system** should be deployed to monitor exchanging and access of keys
- 3 Passphrases and passwords must be used to **encrypt the key**, if stored in disk
- 4 Keys should not be present inside the **source code** or **binaries**
- 5 For certificate signing, **transfer of private keys** should not be allowed
- 6 For symmetric algorithms, key size of **168 bits** or **256 bits** should be preferred for a secure system, especially in case of large transactions
- 7 **Message authentication** must be implemented for encryption of symmetric-key protocols
- 8 For asymmetric algorithms, key size of **1536 bits** and **2048 bits** should be considered for secure and highly protected application
- 9 In case of hash algorithm, key size of **168** or **256 bit** should be considered
- 10 Only recommended tools or products should be preferred rather than creating self-engineered **crypto** algorithms or functions
- 11 Put a limit on **number of operations** per key
- 12 The output of the hash function should have **larger bit length**, which makes it hard to decrypt

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

How to Defend Against Cryptographic Attacks

Follow the countermeasures given below to prevent cryptographic attacks:

- Access of cryptographic keys should be given to the application or to the user directly.
- Intrusion detection system should be deployed to monitor exchanging and access of keys.
- Passphrases and passwords must be used to encrypt the key, if stored in disk.
- Keys should not be present inside the source code or binaries.
- For certificate signing, transfer of private keys should not be allowed.
- For symmetric algorithms, key size of 168 bits or 256 bits should be preferred for a secure system, especially in case of large transactions.
- Message authentication must be implemented for encryption of symmetric-key protocols.
- For asymmetric algorithms, key size of 1536 bits and 2048 bits should be considered for secure and highly protected application.
- In case of hash algorithm, key size of 168 or 256-bit should be considered.
- Only recommended tools or products should be preferred rather than creating self-engineered crypto algorithms or functions.
- Put a limit on number of operations per key.
- The output of the hash function should have larger bit length that makes it hard to decrypt.



The slide features a dark blue header with 'Cryptography' on the left and 'Module Summary' in the center. On the right is the CEH logo. The main content area has a light blue background with a list of seven bullet points. At the bottom, there is a green landscape image and a small copyright notice.

- Cryptography is the conversion of data into a scrambled code that is decrypted and sent across a private or public network
- Symmetric encryption uses the same key for encryption as it does for decryption and asymmetric encryption uses different encryption keys for encryption and decryption
- Ciphers are algorithms used to encrypt or decrypt the data
- PKI is a set of hardware, software, people, policies, and procedures required to create, manage, distribute, use, store, and revoke digital certificates
- Digital signature uses asymmetric cryptography to simulate the security properties of a signature in digital, rather than in a written form
- Disk encryption protects confidentiality of the data stored on the disk by converting it into an unreadable code using disk encryption software or hardware
- Cryptography attacks are based on the assumption that the cryptanalyst has access to the encrypted information

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Module Summary

This module familiarized you with various topics related with cryptography, such as its concepts, encryption algorithms, tools, public-key infrastructures (PKI), email and disk encryption, cryptographic attacks, and cryptanalysis tools.