Cryptography

Module 18

Ethical Hacking and Countermeasures v9

Module 18: Cryptography

Exam 312-50

Market Survey 2014: The Year of Encryption

C|EH Certified Ethical Hacker

60% of those survey said that Edward Snowden revelations have made them more aware of data security

100% of those not interested in security systems admitted to regularly sharing sensitive/ confidential data with external third parties

Among the 60%, approximately **70%** have been directly influenced to look at new data security systems

Over **2/3** of people felt that government certification combined with ease of use would be deciding factors when selecting a data security solution

94% of people looking to invest in new systems are specifically examining secure (encryption) electronic data security systems

One in two people now perceive the Cloud to be less secure as result of Snowden

Only **17%** of those surveyed said their existing secure information sharing system was easy to use

One third of those surveyed were not that upcoming EU DPA reforms would impact the way they or their organization handles and protects data

http://www.egress.com

Case Study: Heartbleed

C|EH Certified Ethical Hacker

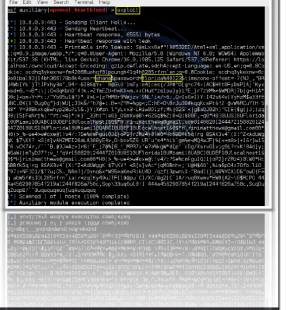
Heartbleed is a security flaw in the OpenSSL cryptographic software library, which allows data traversal over SSL/TLS in plain-text

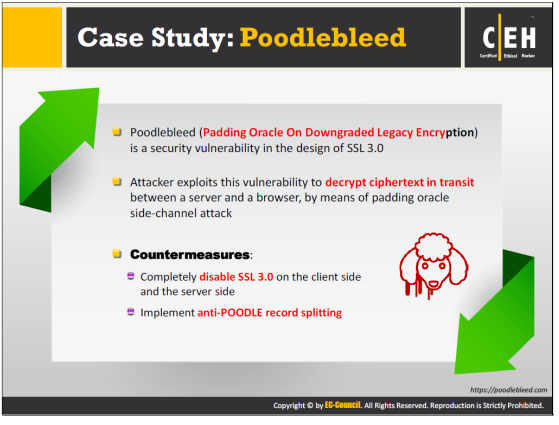
Heartbleed exploits a built-in feature of OpenSSL called heartbeat

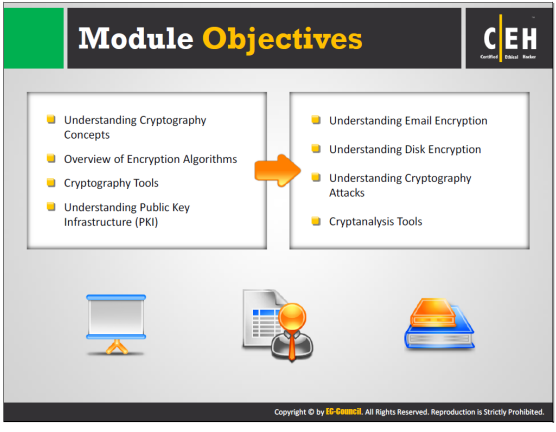
Attackers exploit this vulnerability to get information such as OpenSSL private keys, OpenSSL secondary keys, up to 64kb of memory from the affected server, usernames and passwords, etc.

Versions of OpenSSL affected by Heartbleed include 1.0.1 to 1.0.1f

Updating OpenSSL to version 1.0.1g or higher resolves the vulnerability

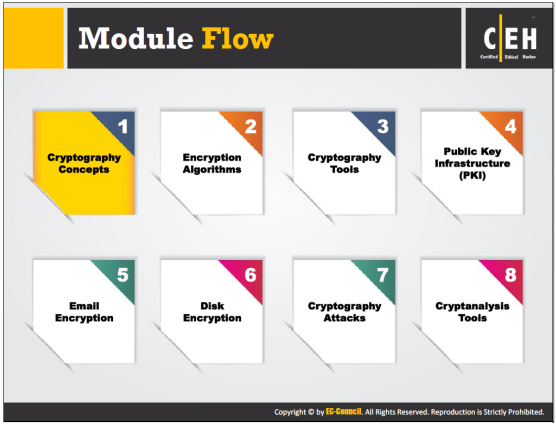




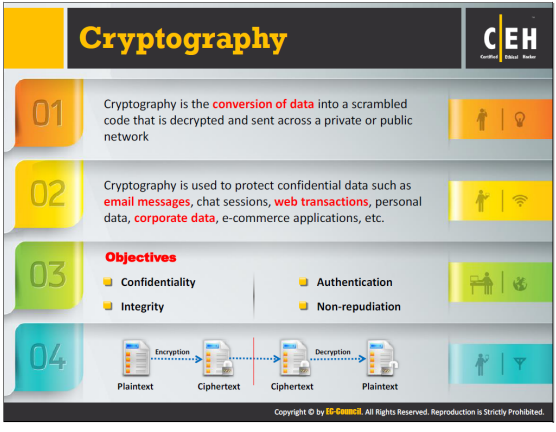


With the increasing adoption of Internet–World Wide Web use for business and personal communication, securing sensitive information such as credit-card and PINs, bank account numbers, and private messages is becoming increasingly more important and more difficult. Today's information-based organizations 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 helps 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.



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

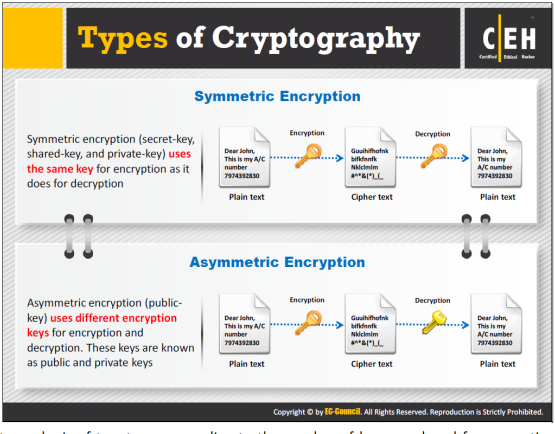
Cryptography is the practice of concealing information by converting plain text (readable format) into cipher text (unreadable format) using a key or encryption scheme. Cryptography protects confidential data such as email messages, chat sessions, web transactions, personal data, corporate data, e-commerce 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, MD5, SHA, DES, and AES, resulting in a cipher text (unreadable format) that, on reaching the destination, is decrypted into readable plain text.



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 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.

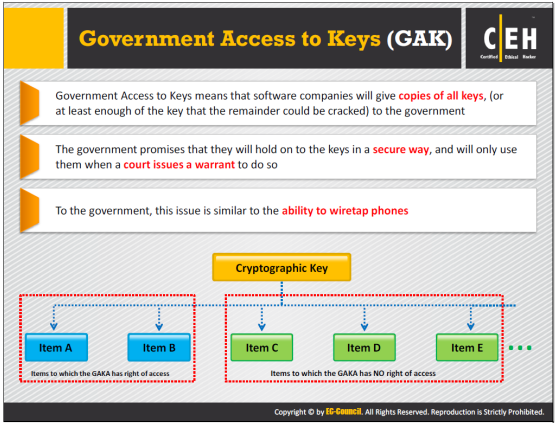
No one but the holder of the private key can decrypt a message composed with the corresponding public key. This increases the security of the information because all communications involve only public keys; the message sender never transmits or shares the private keys. The sender must link the public 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

Given below are the strengths and weaknesses of symmetric encryption and asymmetric encryption:

	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	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 be repudiated
Weaknesses	**Symmetric Encryption**	**Asymmetric Encryption**
	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 on determining the private key)
	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 MITM and brute-force attacks

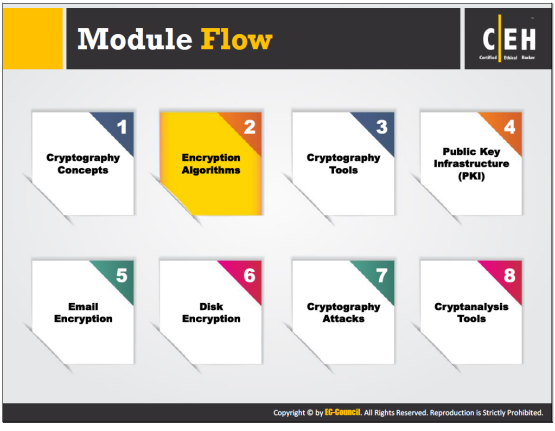
TABLE 18.1: Strengths and Weaknesses of Crypto Methods



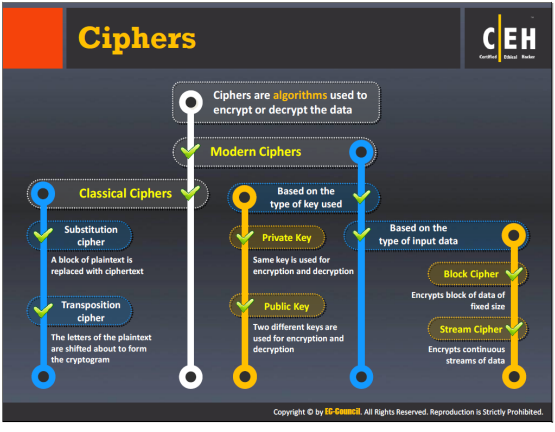
GAK refers to statutory obligation of individuals and organizations to disclose their cryptographic keys to government agencies. Law enforcement agencies around the world acquire and use these cryptographic keys to monitor suspicious communication, and collect evidence of cyber crimes in the interest of national security.

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



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.



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 messages 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.

Types of ciphers:

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.

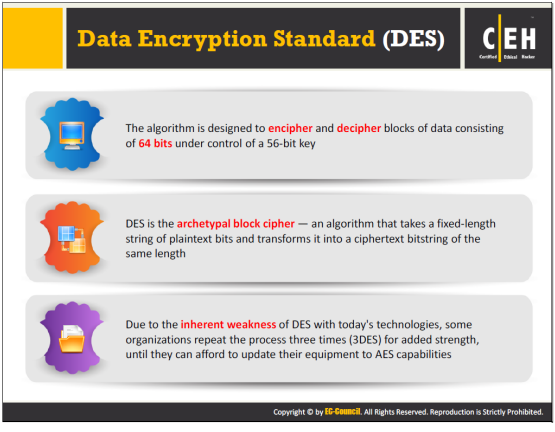
Types of Modern ciphers:

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.

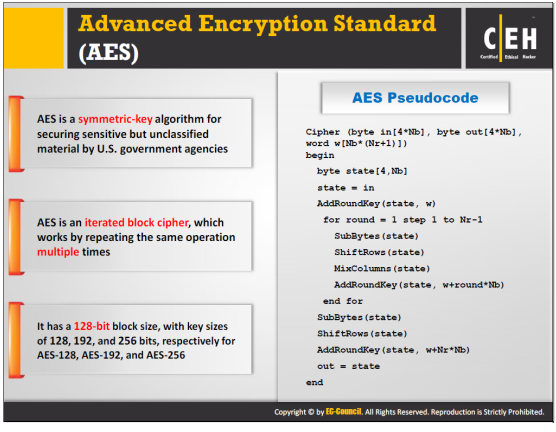
Based on the type of input data

- **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.
- **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.



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'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 choses 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.



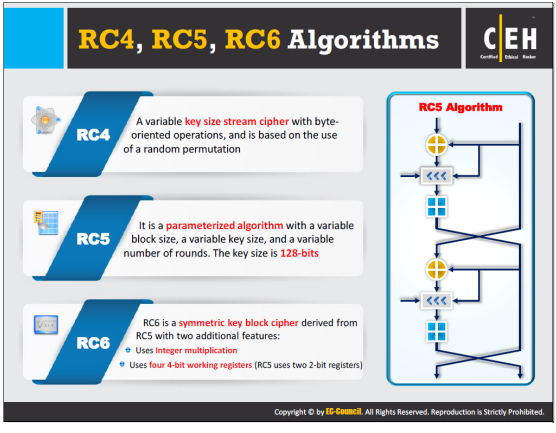
The Advanced Encryption Standard (AES) is a **National Institute of Standards and Technology** (**NIST**) specification for the encryption of electronic data. It also helps to encrypt digital information such as telecommunications, financial, and government data. US government agencies have been using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



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**

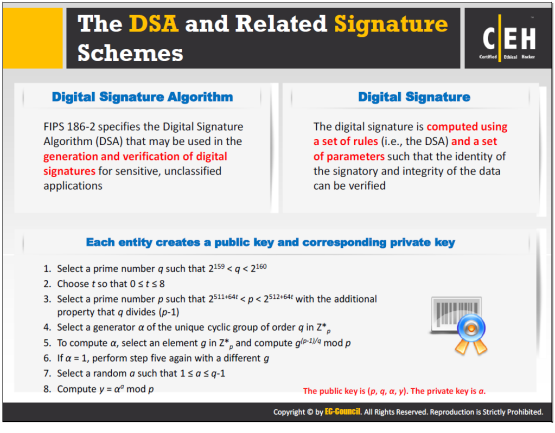
  RC5 is a fast symmetric-key block cipher designed by Ronald Rivest for RSA Data Security (now RSA security). The algorithm is a parameterized algorithm with a variable block size, key size, and number of rounds. The block sizes can be 32, 64, or 128 bits. The range of the rounds can vary from 0 to 255, 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 similar to the RC5 algorithm in that 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.



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:

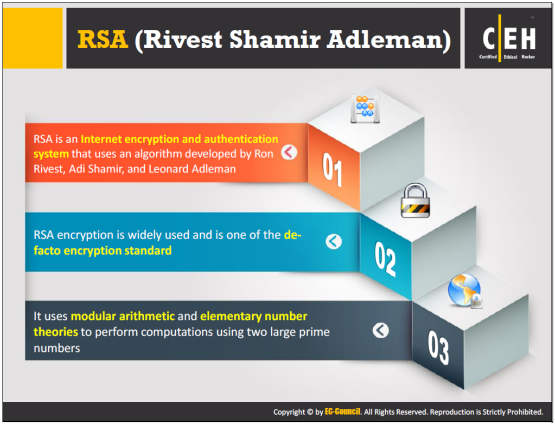
1. Select a prime number q such that $2^{159} < q < 2^{160}$
2. Choose t so that $0 \leq t \leq 8$, and select a prime number p where $2^{511+64t} < p < 2^{512+64t}$, with the property that q divides $(p-1)$
3. Select a generator α of the unique cyclic group of order q in Z^*_p by choosing an element $g \in Z^*_p$ 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 $u_1 = w.h(m) \bmod q$ and $u_2 = rw \bmod q$
5. Compute $v = (\alpha^{u1} y^{u2} \bmod p) \bmod q$
6. Accept the signature if and only if $v=r$



Ron Rivest, Adi Shamir, and Leonard Adleman formulated RSA, a public-key cryptosystem for 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. 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:

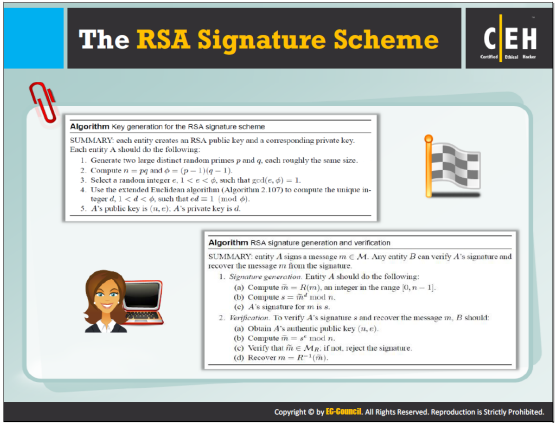
1. Two large prime numbers are taken (a and b), and their product is determined (c = ab, where "c" is called the modulus).
2. RSA chooses a number "e" that it is less than "c" and relatively prime to (a-1)(b-1). Therefore, e and (a-1)(b-1) have no common factors except 1.
3. Apart from this, RSA chooses 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 is 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.



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 $S_{John}(M)$. John sends M along with the signature $S_{John}(M)$ 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 ϕ = (p-1)(q-1)
- Choose a random integer e, $1 < e < \phi$, such that $\gcd(e, \phi) = 1$

  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.

Signature Generation

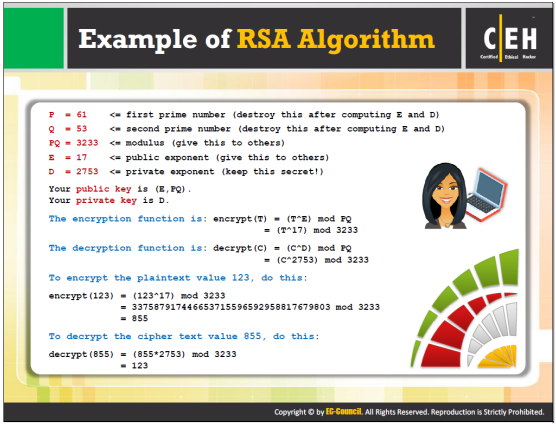
To sign a message m, entity A should do the following:

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

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 $\tilde{m} = s^e \bmod n$
- Verify that $\tilde{m} \in M_R$; if not, reject the signature
- Recover $m = R^{-1}(\tilde{m})$



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 (mod (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) mod 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) mod 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:

encrypt(T) = (T^E) mod PQ

= (T^17) mod 3233

The decryption function is:

decrypt(C) = (C^D) mod PQ

= (C^2753) mod 3233

To encrypt the plaintext value 123, do this:

encrypt(123) = (123^17) mod 3233

= 337587917446653715596592958817679803 mod 3233

= 855

To decrypt the cipher text value 855, do this:

decrypt(855) = (855*2753) mod 3233

= 123

One way to compute the value of **855^2753 mod 3233** is like this:

Consider these powers of 855:

- 855^1 = 855 (mod 3233)
- 855^2 = 367 (mod 3233)
- 855^4 = 367^2 (mod 3233) = 2136 (mod 3233)
- 855^8 = 2136^2 (mod 3233) = 733 (mod 3233)
- 855^16 = 733^2 (mod 3233) = 611 (mod 3233)
- 855^32 = 611^2 (mod 3233) = 1526 (mod 3233)
- 855^64 = 1526^2 (mod 3233) = 916 (mod 3233)
- 855^128 = 916^2 (mod 3233) = 1709 (mod 3233)

- 855^256 = 1709^2 (mod 3233) = 1282 (mod 3233)
- 855^512 = 1282^2 (mod 3233) = 1160 (mod 3233)
- 855^1024 = 1160^2 (mod 3233) = 672 (mod 3233)
- 855^2048 = 672^2 (mod 3233) = 2197 (mod 3233)

Given the above, we know this:

855^2753 (mod 3233)

= 855^(1 + 64 + 128 + 512 + 2048) (mod 3233)

= 855^1 * 855^64 * 855^128 * 855^512 * 855^2048 (mod 3233)

= 855 * 916 * 1709 * 1160 * 2197 (mod 3233)

= 794 * 1709 * 1160 * 2197 (mod 3233)

= 2319 * 1160 * 2197 (mod 3233)

= 184 * 2197 (mod 3233)

= 123 (mod 3233)

= 123

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.

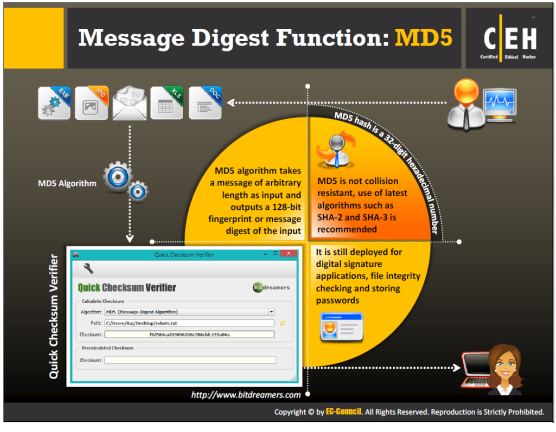
Message digest functions are also called one-way hash functions because they produce values that are almost impossible to invert, resistant to attack, mostly unique, and widely distributed. Message-digest algorithms themselves do not participate in encryption and decryption operations. They enable creation 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 one-way bash functions because they are not reversible.



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

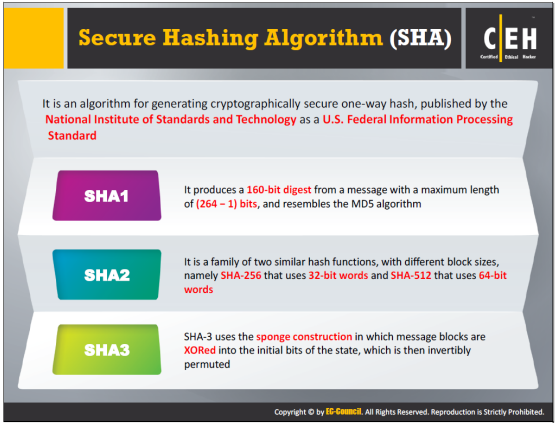
  2db1ff7a70245309e9f2165c6c34999d

Even minimally different texts produce radically different MD5 codes.

Quick Checksum Verifier

Source: *http://www.bitdreamers.com*

Checksum Verifier generates and checks file integrity by secure time-proven algorithms like MD5 and SHA-1. One can create checksums (the digital fingerprints) of files and verify their integrity in the future.



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. That is 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.

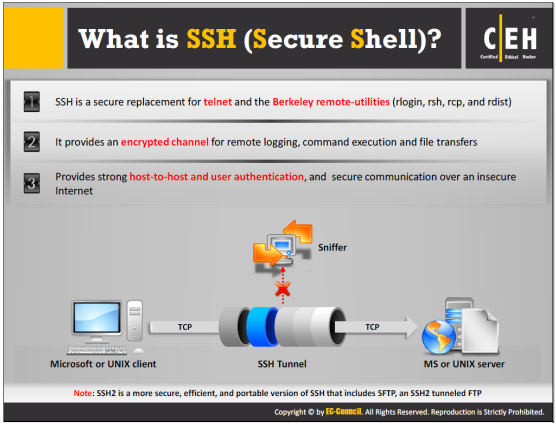
- **SHA-3**

  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(theoritical attack[3] 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						128
	Sha-384	384	512 (8*64)	1024	2^{128}-1	80	Add mod 2^{64}, and, or, xor, shr, rot.	192
	Sha-512	512						256
	Sha-512/224	224						112
	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 18.2: Comparison between SHA-0, SHA-1 and SHA-2 Functions



Secure Shell (SSH) is a program that provides an encrypted channel for remote logging, command execution, and file transfers. It offers strong host-to-host and user authentication, besides securing encrypted communications over an insecure Internet. SSH is a secure replacement for telnet and the Berkeley remote utilities (rlogin, rsh, rcp, and rdist). It uses RSA (public key cryptography) to establish connection and to authenticate. Encryption algorithms it uses include DES, Blowfish, and IDEA (by default). SSH guards network against attacks such as IP spoofing, IP source routing, and DNS spoofing.

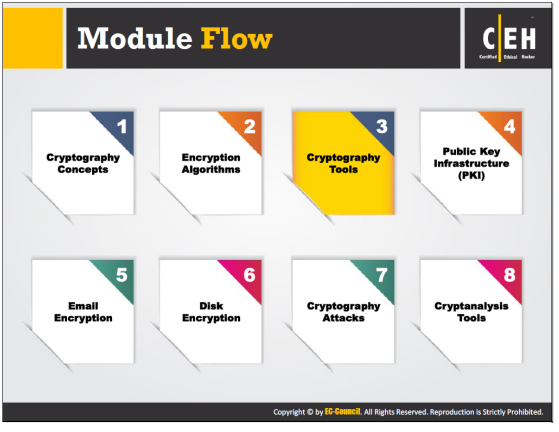
SSH and SSH-2 are completely different protocols and are not compatible with each other. SSH encrypts the user's server and host keys to authenticate, while SSH-2 only uses the host keys. SSH-2 is a more secure, efficient, and portable version of SSH that includes SFTP, an SSH-2 tunneled FTP. Additionally, SSH is vulnerable to attacks due to the presence of structural weaknesses.

SSH authenticates with the help of one or more of the following mechanisms:

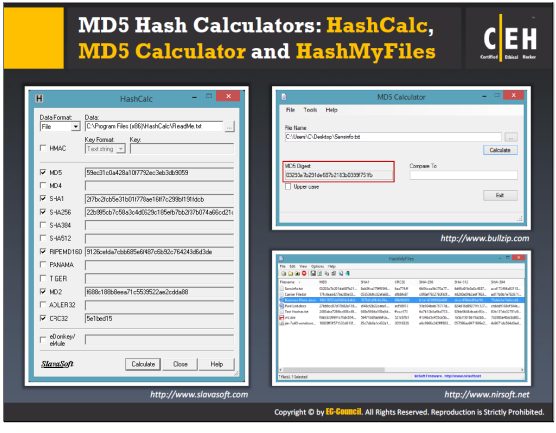
- Password (the /etc/passwd or /etc/shadow in UNIX)
- User public key (RSA or DSA, depending on the release)
- Kerberos (for SSH)
- Host-based authentication (.rhosts or /etc/hosts equivalent in SSH, or public key in SSH-2)

Secure Shell protects against:

- A remote host sending out packets that pretend to come from another trusted host (IP spoofing)
- A host pretending that an IP packet comes from another trusted host (IP source routing)
- An attacker forging name-server records (DNS spoofing)
- Intermediate hosts capturing passwords and other data
- People who control the intermediate hosts exploiting captured data
- Attackers listening to X authentication data and spoofing connections to the X11 server



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.



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

HashCalc

Source: *http://www.slavasoft.com*

HashCalc utility allows to compute message digests, checksums, and HMACs for files, as well as for text and hex strings. It offers different types of hash and checksum algorithms (MD2, MD4, MD5, SHA-1, SHA-2 (256, 384, and 512), RIPEMD-160, PANAMA, TIGER, ADLER32, CRC32) for calculations.

MD5 Calculator

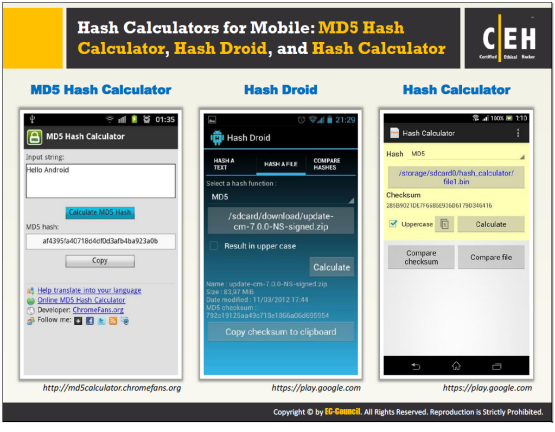
Source: *http://www.bullzip.com*

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 equals 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: *http://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.



MD5 Hash Calculator

Source: http://md5calculator.chromefans.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

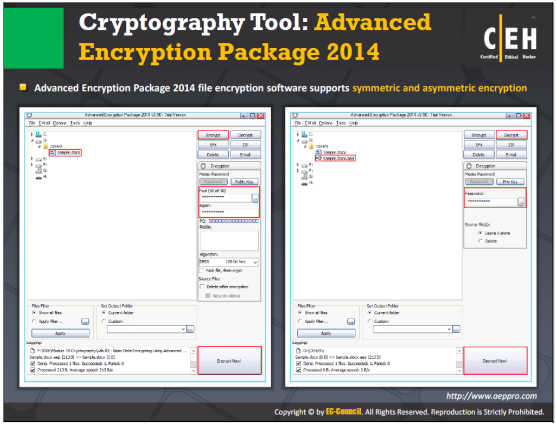
The Hash Droid utility helps to calculate a hash from a given text or from a file stored on the device. In this application, the available hash functions are: Adler-32, CRC-32, Haval-128, MD2, MD4, MD5, RIPEMD-128, RIPEMD-160, SHA-1, SHA-256, SHA-384, SHA-512, Tiger and Whirlpool. It allows copying the 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.

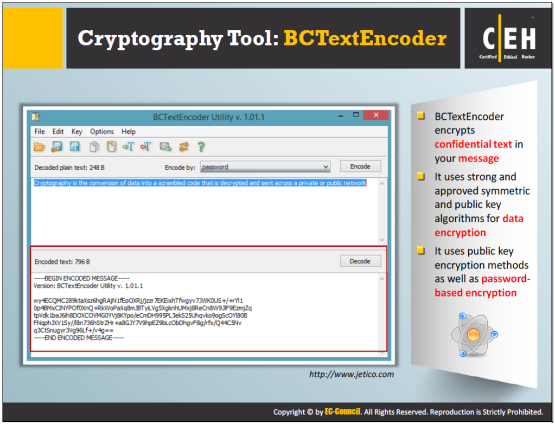


Advanced Encryption Package 2014 is file encryption software for Windows used for secure file transfer, batch file encryption, and encrypted backups.

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

Source: http://www.aeppro.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 and password-based encryption methods. BCTextEncoder uses symmetric and public-key algorithms to encrypt data.

Source: http://www.jetico.com

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:

AutoKrypt

Source: *http://www.hiteksoftware.com*

AutoKrypt is data encryption software designed for automation. It automatically encrypts or decrypts files and folders on a schedule.

Features:

- Encryption methods include: OpenPGP password encryption/decryption, public/private key encryption/decryption, key sign/verify, key encrypt and sign/decrypt and verify.
- Create profiles for each encryption method. Then use these profiles in Encryption tasks.
- The encryption and decryption tasks can automatically encrypt/decrypt according to filename or file timestamp.

Cryptainer LE Free Encryption Software

Source: *http://www.cypherix.com*

Cryptainer LE Free Encryption Software helps to protect confidential data on Windows PC, desktop, laptop, hard disk or removable drive such as USB flash drive, memory stick.

- Creates an encrypted disk drive (volume) to store any type of data

- Uses 448 bit strong encryption
- Cryptainer Mobile edition encrypts any data on any media (USB Drive, CD ROM, Flash Disk)
- Send secure emails

Steganos LockNote

Source: https://www.steganos.com

Steganos LockNote allows users to encrypt confidential data on laptops, PCs, USB sticks, CDs and DVDs. One can use it to hide serial numbers, passwords, phone numbers, and everyday notes in a safe place. This tool encrypts Information using a password and AES 256-bit encryption technology.

AxCrypt

Source: http://www.axantum.com

AxCrypt is the file encryption software for Windows. It integrates with Windows to compress, encrypt, decrypt, store, send, and work with individual files.

CryptoForge

Source: http://www.cryptoforge.com

CryptoForge is file encryption software for personal and professional data security. It allows protecting the privacy of sensitive files, folders, or emailing messages. After encrypting the information, one can store it on insecure media or transmit it on an insecure network—like the Internet—and still keep it secret. Later, it decrypts the information into its original form.

NCrypt XL

Source: http://www.littlelite.net

NCrypt XL is a tool to encrypt and scramble MS Excel spreadsheets, using standard algorithms (DES, AES). It carries out the encryption on every cell of the sheet and retains user formats.

ccrypt

Source: http://ccrypt.sourceforge.net

ccrypt is a utility for encrypting and decrypting files and streams. It has the Rijndael block cipher as the base, a version of which is part of the Advanced Encryption Standard. The algorithm provided by ccrypt is not symmetric (i.e., one must specify whether to encrypt or decrypt).

WinAES

Source: http://fatlyz.com

WinAES is a cryptography toolbox designed to enable one to encrypt and decrypt files.

Features:

- AES/Camellia Encryption/Decryption – Uses 256-bit encryption key, CBC mode with randomized IV, provides maximum security.

- File Hash Calculation – supports hash algorithms, including MD5, SHA-1, SHA-2(256bits), SHA-2(512bits), Keccak (One of SHA-3 Finalists).
- Secure File Eraser – ensures that there is no possibility of recovering an erased file.
- Random Password Generator – generates passwords 9999 characters long.
- Base64 Encoder/Decoder – encodes ANSI or Unicode text with Base64 encoder.

EncryptOnClick

Source: http://www.2brightsparks.com

EncryptOnClick helps to encrypt and protect sensitive files.

Features:

- Secure encryption and decryption method is used (256-bit AES encryption)
- Files are both compressed & encrypted, which results in a smaller file
- Password protected
- Encrypt single files or all files in a folder
- Unicode enabled so filenames in any language can be encrypted
- Encrypt, decrypt, compress, and uncompress files which can also be opened and decrypted using third party programs like WinZip 9 (provided the correct password is used)

GNU Privacy Guard

Source: http://www.gnupg.org

GnuPG is a command-line tool that allows to encrypt and sign the data and communication. It is a crypto engine, which one can use directly from a command prompt, from shell scripts or by other programs. Therefore, it is a backend for other applications.

Features:

- Does not use any patented algorithms
- Full OpenPGP implementation
- Decrypts and verifies PGP 5, 6 and 7 messages
- Supports ElGamal, DSA, RSA, AES, 3DES, Blowfish, Twofish, CAST5, MD5, SHA-1, RIPE-MD-160 and TIGER
- Supports key and signature expiration dates

Discussed below are some cryptographic tools for mobile devices:

Secret Space Encryptor

Source: *http://www.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++)

CryptoSymm

Source: *https://play.google.com*

CryptoSymm is an application that can encrypt text messages and files to easily share them. It works based on the use of symmetric key algorithms. Only the person who knows the key and the encryption method can decrypt it.

Functions:

- Encrypt/decrypt texts
- Encrypt/decrypt files
- Share the results directly from the app

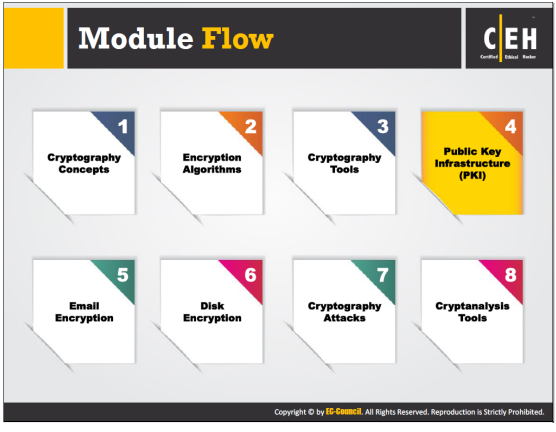
Cipher Sender

Source: *https://play.google.com*

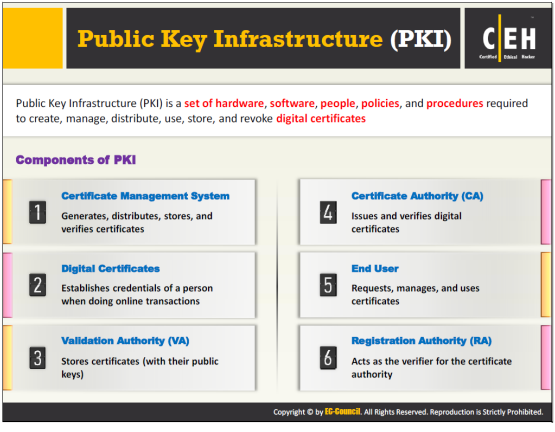
Cipher Sender allows to send and receive coded messages. It is a cross-platform, so one can send messages on iPhones.

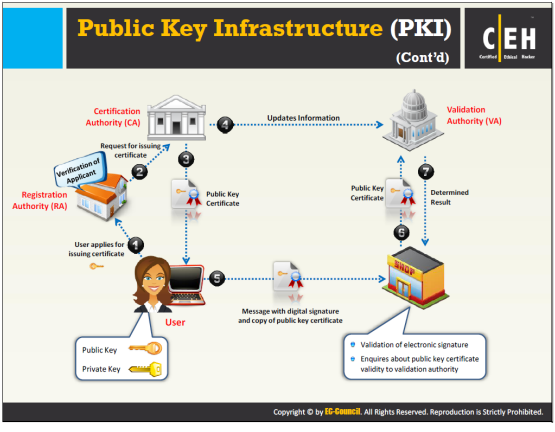
This app includes ciphers and codes given below:

- Caesar Shift Cipher
- Morse Code
- Rail Fence Cipher
- Rot13
- Monoalphabetic Substitution Cipher
- Vigenere Cipher
- Letter-Number Code
- Atbash Cipher
- Keyboard Code



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





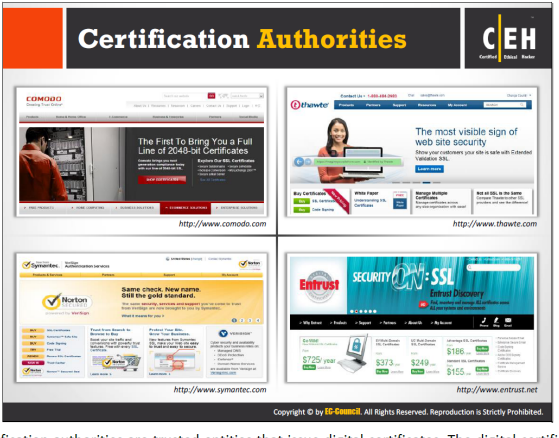
Public Key Infrastructure (PKI) is a security architecture developed to increase the confidentiality of information exchanged over the insecure Internet. It includes hardware, software, people, policies, and procedures required to create, manage, distribute, use, store, and revoke digital certificates.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

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 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: http://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 solutions such as Comodo Certificate Manager and Comodo EPKI Manager.

Digital Certificates offered by Comodo:

- Extended validation (EV)-SSL
- Multi-domain EV SSL
- Wildcard SSL
- Unified Communications (UC)
- Intel Pro Series

- General Purpose SSL
- Secure Email - S/MIME
- Client Authentication
- Code Signing

thawte

Source: *http://www.thawte.com*

thawte is a global Certification Authority. It offers SSL and code signing digital certificates to secure servers, provide data encryption, authenticate users, protect privacy and assure online identifies through stringent authentication and verification processes.

SSL Certificates Offered:

- SSL Web Server Certificates with EV
- SGC SuperCerts
- SSL Web Server Certificates
- SSL123 Certificates
- SAN/UC Capable Certificates
- Wildcard SSL Certificates
- SSL for the Enterprise

Norton Secured Seal

Source: *http://www.symantec.com*

VeriSign Authentication Services, now part of Symantec Corporation (NASDAQ: SYMC), provides solutions that allow companies and consumers to engage in communications and commerce online with confidence.

SSL Certificates Offered:

- Secure Site Pro with EV SSL Certificates
- Secure Site with EV SSL Certificates
- Secure Site Pro SSL Certificates
- Secure Site Wildcard SSL Certificates
- Secure Site SSL Certificates
- Managed PKI for SSL
- Symantec Certificate Intelligence Center

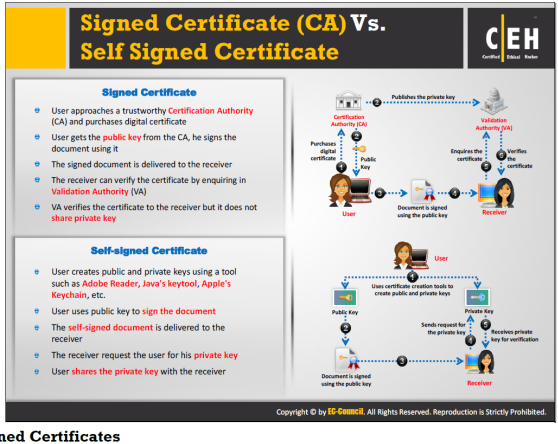
Entrust

Source: *http://www.entrust.net*

Entrust provides identity-based security solutions that empower enterprises, consumers, citizens and Web sites. Entrust's solutions include strong authentication, fraud detection, digital certificates, SSL, and PKI.

SSL Certificates Offered:

- Standard SSL Certificate
- Advantage SSL Certificate
- EV Multi-Domain SSL Certificate
- Wildcard SSL Certificate
- UC Multi-Domain SSL Certificate
- Private SSL Certificate



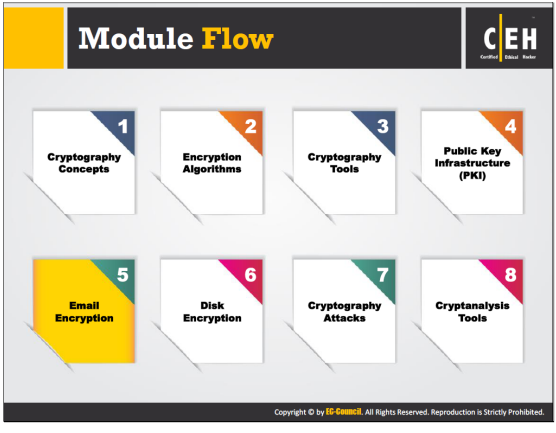
Signed Certificates

Certification authorities (CAs) signs and issues signed certificates. These certificates contain a public key and the identity of the owner. The corresponding private key is not made publicly available, instead kept as secret by the authorized user. 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 VeriSign, Symantec, Thawte, and RapidSSL.

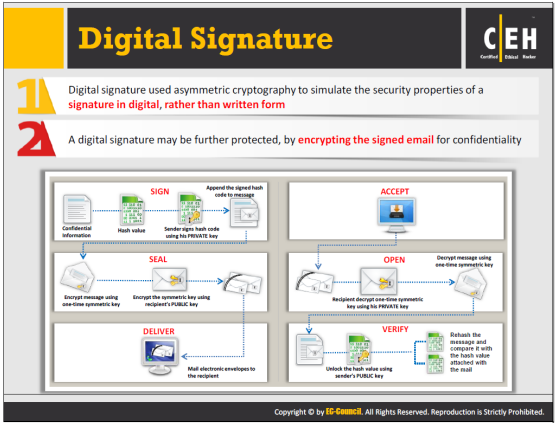
In the diagram above, user approaches a trustworthy CA and purchases a digital certificate. The CA issues a certificate (has public key and the identity of the user) to the user, and updates the VA with the same information. The user signs a document with the public key and sends it to the receiver. Thereafter, the receiver verifies the certificate by enquiring to the VA, who verifies the certificate to the receiver 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. In general, self-signed certificates are widely used for testing servers. In the diagram above, user creates public and private keys using certificate creation tools. User signs the document with public key and sends it to the receiver. The receiver sends a request to the user for his/her private key. The user shares the private key with the receiver, through which the receiver verifies if the document is from the user who he/she intends to be.



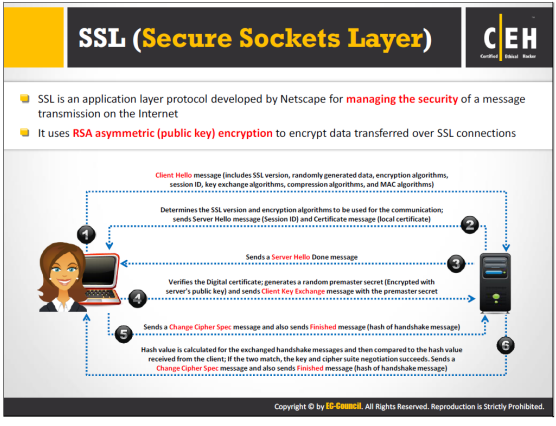
Currently, most businesses use email as the primary source of communication, as it is simple and easy to communicate or share information. Emails can contain sensitive information about the organization, such as projects, upcoming news, and financial data, which, when accessed by the wrong person can 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.



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.



The Secure Sockets Layer (SSL) 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.

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 "**channel security**" 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.

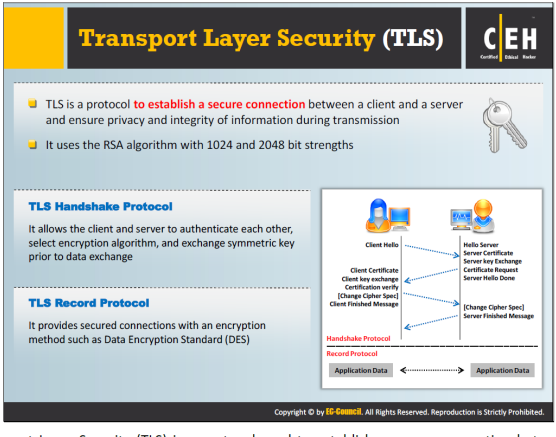
SSL Handshake Protocol Flow

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.



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:

1. TLS Record Protocol
2. TLS Handshake Protocol

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.

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.

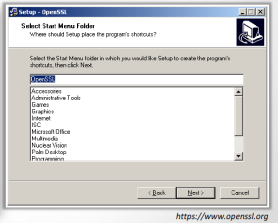
- The server receives the Pre-Master Secret. Thereafter, the client and server each create the Master Secret and session Keys based on the Pre-Master Secret.
- The client sends “Change cipher spec” notification to the server to indicate that it will start using the new session keys for hashing and encrypting messages. The client also sends “Client finished” message.
- The server receives "Change cipher spec" from the client and switches its record layer security state to symmetric encryption using the session keys. The server sends "Server finished" 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 Toolkit: OpenSSL

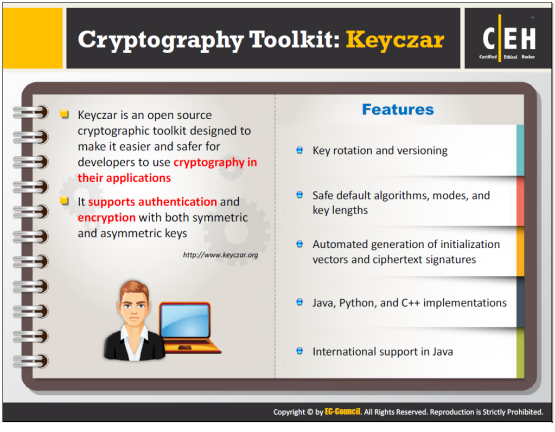
- OpenSSL is an open source cryptography toolkit implementing the **Secure Sockets Layer (SSL v2/v3)** and **Transport Layer Security (TLS v1)** network protocols and related cryptography standards required by them
- The openssl program is a command line tool for using the various **cryptography functions**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



https://www.openssl.org

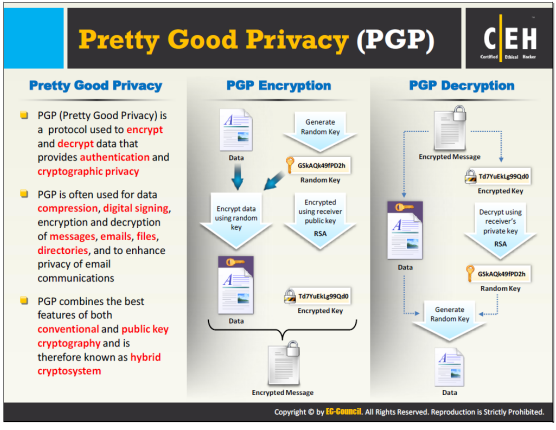


An illustrative use case:

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");
```

Source: *http://www.keyczar.org*



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

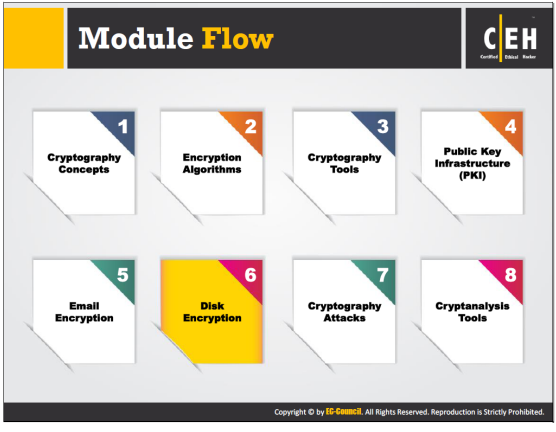
- When a user encrypts data with PGP, PGP first compresses the data.

  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.

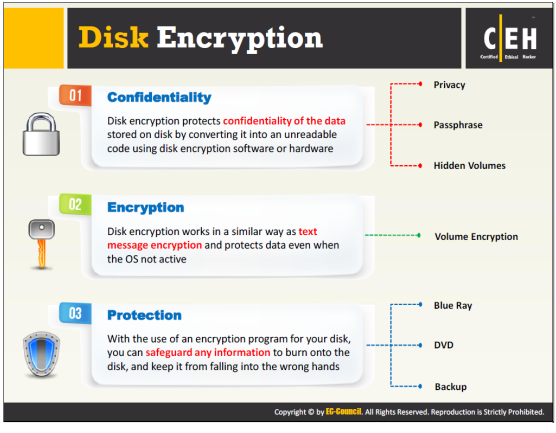
PGP Decryption

- 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 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.

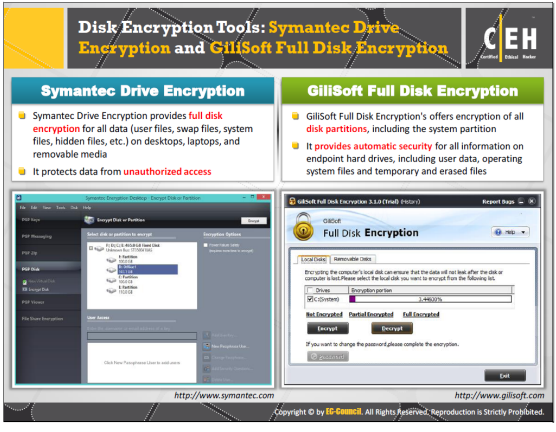


Disk encryption is a technology, which protects the confidentiality of the data stored on disk by converting it into an unreadable code using disk encryption software or hardware, thus preventing unauthorized users from accessing it.

Disk encryption works in a manner similar to text-message encryption. By using an encryption program for the user's disk, the user can safeguard any and 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.



Discussed below are tools used to encrypt data on disk, preventing unauthorized user access:

Symantec Drive Encryption

Source: http://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.

GiliSoft Full Disk Encryption

Source: http://www.gilisoft.com

GiliSoft Full Disk Encryption offers encryption of all disk partitions, including the system partition. Through password protecting a disk, disk partition, or operating system launch, the program disables any unauthorized reading/writing activity on the disk or PC, restricts access and launch of specific disks and files. It provides automatic security for all information on endpoint hard drives, including user data, operating system files, and temporary and erased files. For maximum data protection, multi-factor pre-boot authentication ensures user identity, while encryption prevents data loss from theft.

Discussed below are additional disk encryption tools with a common goal: encrypting a disk partition.

DriveCrypt

Source: http://www.securstar.com

DRIVECRYPT protects all proprietary data on notebooks and desktop computers. It allows users to effectively protect business plans, client lists, product specifications, confidential corporate memos, stock information, and others.

Features:

- Disk Partition and file volume encryption
- Hide data into music files (Steganography)
- Encrypts Pen-Drives and USB disks on a container or partition level
- Password Security
- Administrator Password Control (keyfiles)
- Secure Disk Deletion (Disk Wiping)
- Encrypted Volume Resizing (DriveCrypt Standard edition)

ShareCrypt

Source: http://www.securstar.com

ShareCrypt stores files encrypted on local disk, network shared folders, and file syncing cloud services. It provides user-based access rights to folders and their content. It allows to protect sensitive data ensuring that only authorized user can access the data and lock out any unauthorized party.

PocketCrypt

Source: http://www.securstar.com

PocketCrypt allows user to implement pocket PC encryption and security. It is a file encryption program that provides a virtually encrypted storage area for MS Windows Mobile 2003/2005, Win CE operating systems. The tool creates a container in the memory of the Windows Mobile device and PocketCrypt Windows encryption software protects it. This Pocket PC encryption software creates a "virtual directory" through which the data is accessed. AES algorithm automatically encrypts any data written to the "virtual directory."

Features:

- AES 256-bit Disk Encryption
- SHA 256-bit for password encryption
- Manages up to 400 Mb of encrypted data
- Pcc Viewer allows the user to manage all the encrypted data of the pocket pc also through a desktop

Rohos Disk Encryption

Source: http://www.rohos.com

Rohos Disk creates hidden and protected partitions on the computer or USB flash drive, and password protects/locks access to Internet applications.

Features:

- Strong disk encryption (uses AES - 256 bit)
- Hides encrypted disk into a media container such as AVI, MKV, MP3, OGG, and WMA
- Integrated File-shredder - Any file or folder can be moved into encrypted Rohos Disk with shredding afterwards

R-Crypto

Source: http://www.r-tt.com

R-Crypto is disk encryption software that protects confidential information and personal data on a desktop, notebook, or a removable data storage device against unauthorized access. To protect the data R-Crypto creates encrypted virtual disks (virtual data storage devices). It encrypts the data on virtual disks using the cryptographic infrastructure of the Microsoft Windows operating system.

east-tec SafeBit

Source: http://www.east-tec.com

east-tec SafeBit is disk encryption software that protects data against potential unauthorized access and information leaks.

Features:

- On-the-fly encryption - data is automatically encrypted right before it is saved and decrypted right after it is loaded, without any user intervention
- Uses the strong 256-bit AES encryption algorithm to scramble and make the data unreadable
- Safeguard data on the PC, removable devices or in the cloud
- Allows to save passwords when creating an encrypted safe on personal removable disk device, such as USB key stick, memory card, and use them later as a key to open safes
- Protection against KeyLoggers, viruses, Trojans, and hackers

DiskCryptor

Source: http://diskcryptor.net

DiskCryptor is an encryption solution that offers encryption of all disk partitions, including the system partition.

Features:

- Support for encryption algorithms such as AES, Twofish, Serpent, and combinations thereof
- Creates encrypted CD and DVD disks
- Supports encryption of external USB storage devices
- Encryption of system and bootable partitions with pre-boot authentication
- Automatic mounting of disk partitions and external storage devices

alertsec

Source: http://www.alertsec.com

Alertsec Endpoint Encrypt ensures encryption of all the data. It provides data security for both PCs and laptops through strong encryption, and addresses the internal threat of unauthorized copying of enterprise data to personal storage devices and removable media through a combination of Port Control and Media Encryption.

Included Software Applications:

- **Full Disk Encryption** - With pre-boot authentication, it ensures that only authorized users can access data on protected computers.
- **Media Encryption** - Provides encryption of removable storage media such as USB sticks, external hard drives, CDs and DVDs.

- **Port Control** - Enables control of the activity on all computer ports and includes centralized logging of port activity. Access to endpoint ports such as USB, FireWire, Bluetooth, Wi-Fi, etc. can be centrally managed.
- **Compliance Check** - All endpoints are scanned for compliance with pre-defined security policies.

Cryptainer LE

Source: *http://www.cypherix.com*

Cryptainer LE - Free Encryption Software protects confidential data on Windows PC, desktop, laptop, hard disk or removable drive such as USB flash drive, memory stick, and so on.

- Creates an encrypted disk drive (volume) to store any type of data
- Uses 448 bit encryption
- Sends secure emails
- Cryptainer Mobile edition encrypts any data on any media (e.g., USB Drive, CD ROM, Flash Disk)

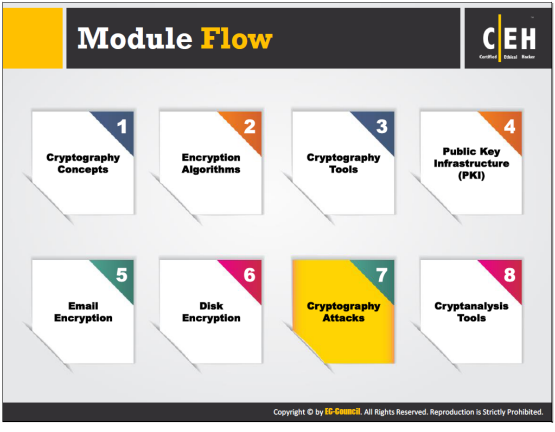
DriveCrypt Plus Pack

Source: *http://www.securstar.com*

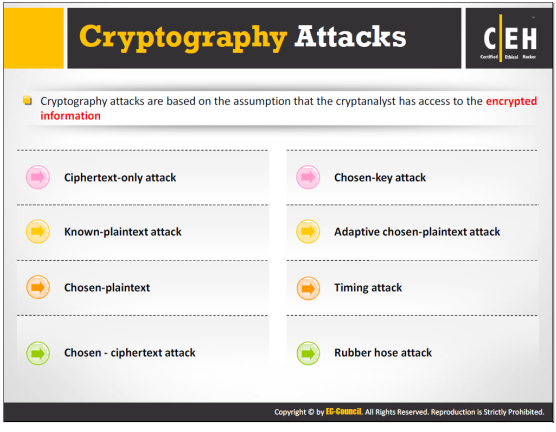
DriveCrypt Plus Pack (DCPP) allows to secure disk(s) (including removable media) with an encryption algorithm (AES-256) at the sector level, ensuring that only authorized users may access it.

Features:

- Boot protection
- Full or partial hard disk encryption
- No size limitation for encrypted disks
- Allows steganography to hide data into pictures
- Encrypts almost any kind of media (e.g., hard disks, floppy disks, ZIP, JAZ)
- Facility to validate the integrity of the encryption method



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. This section deals with various cryptography attacks that an attacker uses to compromise cryptographic system.



Attacker 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 cipher text back to the plain text without knowledge of the key.

Cryptography Attacks (Cont'd)

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

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

Cryptography Attacks (Cont'd)

Chosen-ciphertext Attack

Attacker obtains the plaintexts corresponding to an arbitrary set of ciphertexts of his own choosing

Rubber Hose Attack

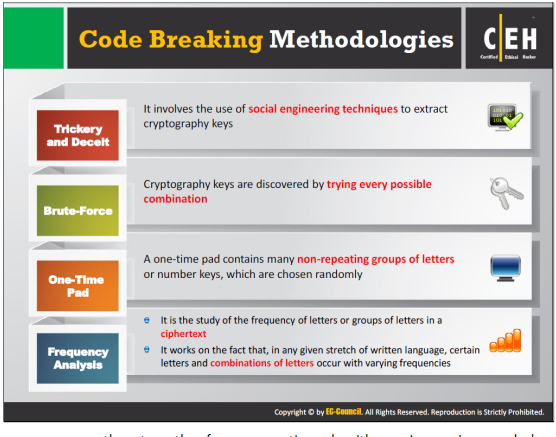
Extraction of cryptographic secrets (e.g. the password to an encrypted file) from a person by coercion or torture

Chosen-key Attack

A generalization of the chosen-text attack

Timing Attack

It is based on repeatedly measuring the exact execution times of modular exponentiation operations



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 of letters and words is another method used to attack ciphers. 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. But there is an exception called a one-time pad, which the users assume to be unbreakable even with infinite resources.

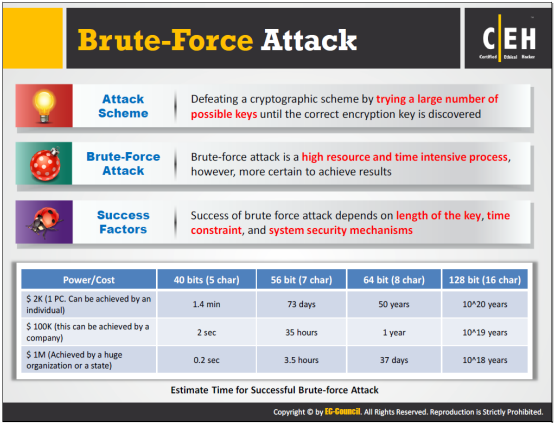
A one-time pad contains mostly a non-repeating set of letters or numbers, which the system chooses randomly. The user writes them on small sheets of paper and then pastes them together in a pad.

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.



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. 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.

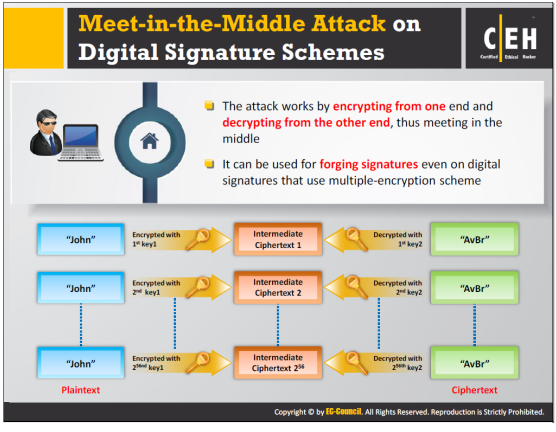
The difficulty of a brute-force attack depends on various issues, such as:

- 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.

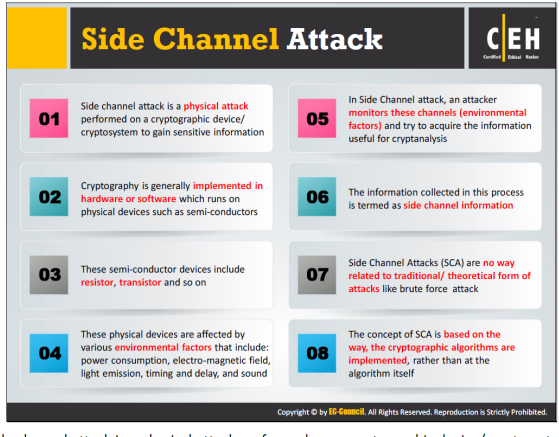
A brute-force attack is, however, more certain to achieve results.



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.

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.



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.

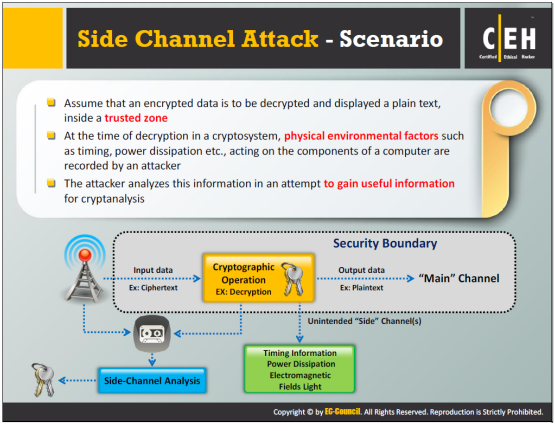
Sound:

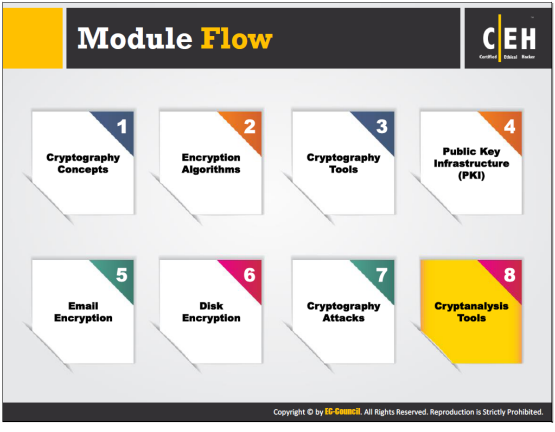
Acoustic attacks exploit the sound produced during a computation. These acoustic emissions are from keyboards and computing components (e.g., CPU, memory)

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.

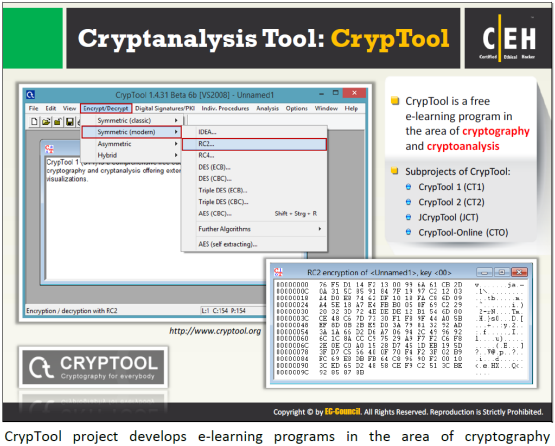
Side-channel-attack mitigation techniques include:

- 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





Cryptanalysis is the study of ciphers, cipher text, or cryptosystems with the ability to identify vulnerabilities in them that allows to extract plain text from the cipher text even if the cryptographic key or algorithm used to encrypt the plain text is unknown. This section deals with various cryptanalysis tools help in breaching cryptographic security.



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.

JCrypTool (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.

Source: *http://www.cryptool.org*

Below are some additional cryptanalysis tools:

CryptoBench

Source: *http://www.addario.org*

CryptoBench provides a source of strong cryptographic transformations to help in the cryptanalysis process of common cryptographic schemes.

Jipher

Source: *http://cipher.org.uk*

Jipher is a cryptanalysis tool that helps attackers to attack old ciphers. An additional functionality is present to analyze cookies.

Ganzúa

Source: *http://ganzua.sourceforge.net*

Ganzúa is a cryptanalysis tool for classical ciphers (mono-alphabetic and polyalphabetic). It lets the user define almost completely arbitrary cipher and plain alphabets, allowing for the proper cryptanalysis of cryptograms obtained from non-English texts.

Monoalphabetic cryptanalysis features:

- Alphabet-wide substitution tools for the Caesar shift cipher and other mono-alphabetic ciphers

- Obtain and display the relative frequencies of the characters, bigrams, and trigrams of cryptograms

Polyalphabetic cryptanalysis features:

- Alphabet-wide substitution tools for the Vigenère or Alberti ciphers
- Obtain and display the relative frequencies of the characters ciphered using each alphabet
- Perform the Kasiski Test on cryptograms

Crank

Source: *http://crank.sourceforge.net*

Crank is short for "**CRyptANalysis toolKit.**" Its overall purpose is to provide an environment for solving classical (pen-and-paper) ciphers, providing as much automation as possible. Classical ciphers include common schemes such as mono-alphabetic substitutions, where it maps each letter of the alphabet to another (usually different) letter consistently through the text.

EverCrack

Source: *http://evercrack.sourceforge.net*

EverCrack is an Open-Source (GPL) Cryptanalysis Engine. It performs cryptanalysis on mono-alphabetic substitution and transposition ciphers.

AlphaPeeler

Source: *http://alphapeeler.sourceforge.net*

AlphaPeeler is a tool for learning cryptology. It includes frequency analysis, mono-alphabetic substitution, Caesar, transposition, Vigenere, and Playfair cipher. Some of its professional features are DES, Gzip, MD5, SHA1, SHA256, RIPEMD-16, RSA, and secret share files.

Draft Crypto Analyzer

Source: *http://www.literatecode.com*

Draft Crypto Analyzer (DRACA) is a tool to perform preliminary detection and analysis of crypto algorithms within executable. It gives a rough idea of what kind of algorithms to look at without spending time on decompilation and code analysis. It supports CRC32, RC5, RC6, RC2, TEA, MD5, Ripemd-160, Tiger, Skipjack, DES, Blowfish, Twofish, Safer, MARS, CAST-256, AES (Rijndael), and SHA-1 algorithms.

Linear Hull Cryptanalysis of PRESENT

Source: *http://www.ecrypt.eu.org*

This tool computes linear hulls for the original PRESENT cipher. It confirms and even improves on the predicted bias (and the corresponding attack complexities) of conventional linear relations based on a single linear trail.

Mediggo

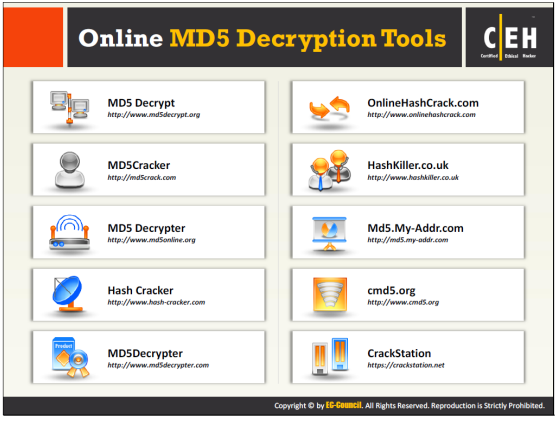
Source: *http://code.google.com*

This library implements generic cryptanalysis techniques to detect weak or insecure cryptosystems or learn and practice with cryptanalysis. This library is open source and written in C programming language.

SubCypher

Source: *http://www.esclepiusllc.com*

SubCypher is a utility to learn about cryptanalysis. It assists in breaking simple substitution cypher encryption on plain text.



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

MD5 Decrypt

Source: *http://www.md5decrypt.org*

MD5 Decrypt allows user to encrypt or decrypt any md5 hash.

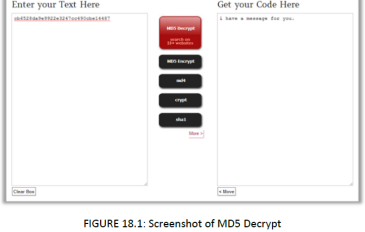


FIGURE 18.1: Screenshot of MD5 Decrypt

MD5Cracker

Source: *http://md5crack.com*

MD5Cracker uses Google and Rainbow tables to crack and decrypt MD5 hashes.

FIGURE 18.2: Screenshot of MD5 Cracker

MD5 Decrypter

Source: *http://www.md5online.org*

MD5 Decrypt allows the user to input hash value and decrypts it to its respective readable value.

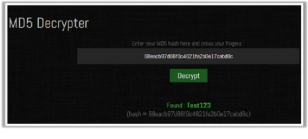


FIGURE 18.3: Screenshot of MD5 Decrypter

Hash Cracker

Source: *http://www.hash-cracker.com*

Hash Cracker allows to crack strings hashed with MD5, SHA1 and NTLM algorithms.

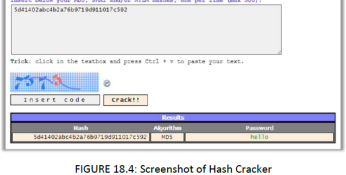


FIGURE 18.4: Screenshot of Hash Cracker

MD5Decrypter

Source: *http://www.md5decrypter.com*

MD5Decrypter allows to input an MD5 hash and search for its decrypted state in its database.

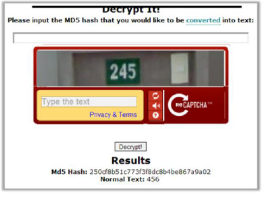


FIGURE 18.5: Screenshot of MD5Decrypter

OnlineHashCrack.com

Source: *http://www.onlinehashcrack.com*

It allows the user to enter hash value (supports LM/NTLM/MD5/MD4/SHA1/MYSQL/OSX) and notifies via email when cracked.

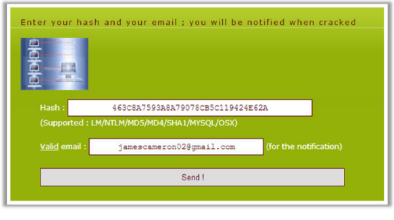


FIGURE 18.6: Screenshot of OnlineHashCrack.com

HashKiller.co.uk

Source: *http://www.hashkiller.co.uk*

It is a MD5 cracker/decryption tool. It allows to input an MD5 hash and search for its decrypted state in its database.

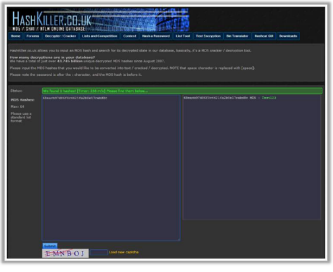


FIGURE 18.7: Screenshot of HashKiller.co.uk

Md5.My-Addr.com

Source: *http://md5.my-addr.com*

It is an online md5 decrypter tool that decrypts md5 string (not case sensitive) into its respective readable value.

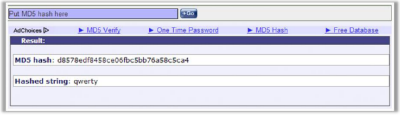


FIGURE 18.8: Screenshot of Md5.My-Addr.com

cmd5.org

Source: *http://www.cmd5.org*

It is an online MD5 decryptor tool that decrypts MD5 hash to its respective readable value.

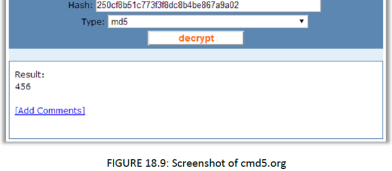


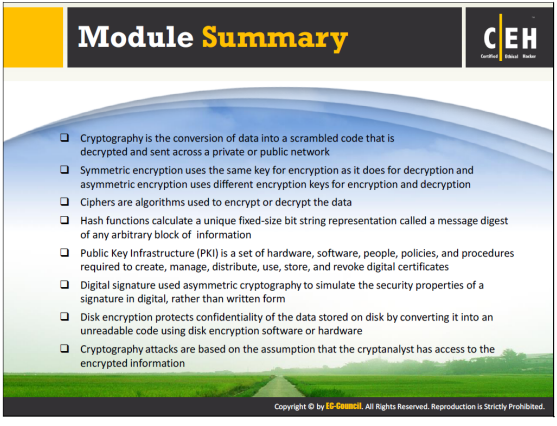
FIGURE 18.9: Screenshot of cmd5.org

CrackStation

Source: *https://crackstation.net*

CrackStation uses pre-computed lookup tables to crack password hashes. It works only for "**unsalted**" hashes. It supports LM, NTLM, md2, md4, md5, md5(md5), md5-half, sha1, sha1(sha1_bin()), sha224, sha256, sha384, sha512, ripeMD160, whirlpool, and MySQL 4.1+ algorithms.

FIGURE 18.10: Screenshot of CrackStation



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.