UNIT-3

MESSAGEAUTHENTICATION

Messageauthenticationisaproceduretoverifythatreceivedmessagescomefromthe alleged source and have not been altered. Message authentication may also verifysequencing and timeliness. It is intended against the attacks like content modification,sequence modification, timing modification and repudiation. For repudiation, concept

ofdigitalsignaturesisusedtocounterit.Therearethreeclassesbywhichdifferenttypesoffunction sthatmaybeusedtoproduce anauthenticator. Theyare:

Messageencryption–theciphertextservesasauthenticator

Message authentication code (MAC)–a public function of the message and a secretkey producing a fixed-length value to serve as authenticator. This does not provide adigitalsignature because AandBsharethesamekey.

Hash function–a public function mapping an arbitrary length message into a fixed-length hash value to serve as authenticator. This does not provide a digital signaturebecausethereis no key.

MESSAGEENCRYPTION:

Messageencryptionbyitselfcanprovideameasureofauthentication.Theanalysisdiffersforconv entionalandpublic-

keyencryptionschemes.Themessagemusthavecomefromthesenderitself,becausethecipherte xtcanbedecryptedusinghis(secretorpublic)key.Also, none of the bits in the message have been altered because an opponent does notknow how to manipulate the bits of the ciphertext to induce meaningful changes to theplaintext. Often one needs alternative authentication schemes than just encrypting themessage.

Sometimesoneneedstoavoidencryptionoffullmessagesduetolegalrequirements.

⬜⬜Encryptionandauthenticationmaybeseparatedinthesystemarchitecture.

Thedifferentwaysinwhichmessageencryptioncanprovideauthentication,confidentiality in both symmetric and asymmetric encryption techniques is explainedwiththetablebelow:

Confidentiality and Authentication Implications of Message Encryption

$A \rightarrow B: E_K[M]$
- Provides confidentiality
  - Only A and B share K
- Provides a degree of authentication
  - Could come only from A
  - Has not been altered in transit
  - Requires some formatting/redundancy
- Does not provide signature
  - Receiver could forge message
  - Sender could deny message

(a) Symmetric encryption

$A \rightarrow B: E_{KU_b}[M]$
- Provides confidentiality
  - Only B has KR_b to decrypt
- Provides no authentication
  - Any party could use KU_b to encrypt message and claim to be A

(b) Public-key encryption: confidentiality

$A \rightarrow B: E_{KR_a}[M]$
- Provides authentication and signature
  - Only A has KR_a to encrypt
  - Has not been altered in transit
  - Requires some formatting/redundancy
  - Any party can use KU_a to verify signature

(c) Public-key encryption: authentication and signature

$A \rightarrow B: E_{KU_b}\left[E_{KR_a}(M)\right]$
- Provides confidentiality because of KU_b
- Provides authentication and signature because of Kr_a

(d) Public-key encryption: confidentiality, authentication, and signature

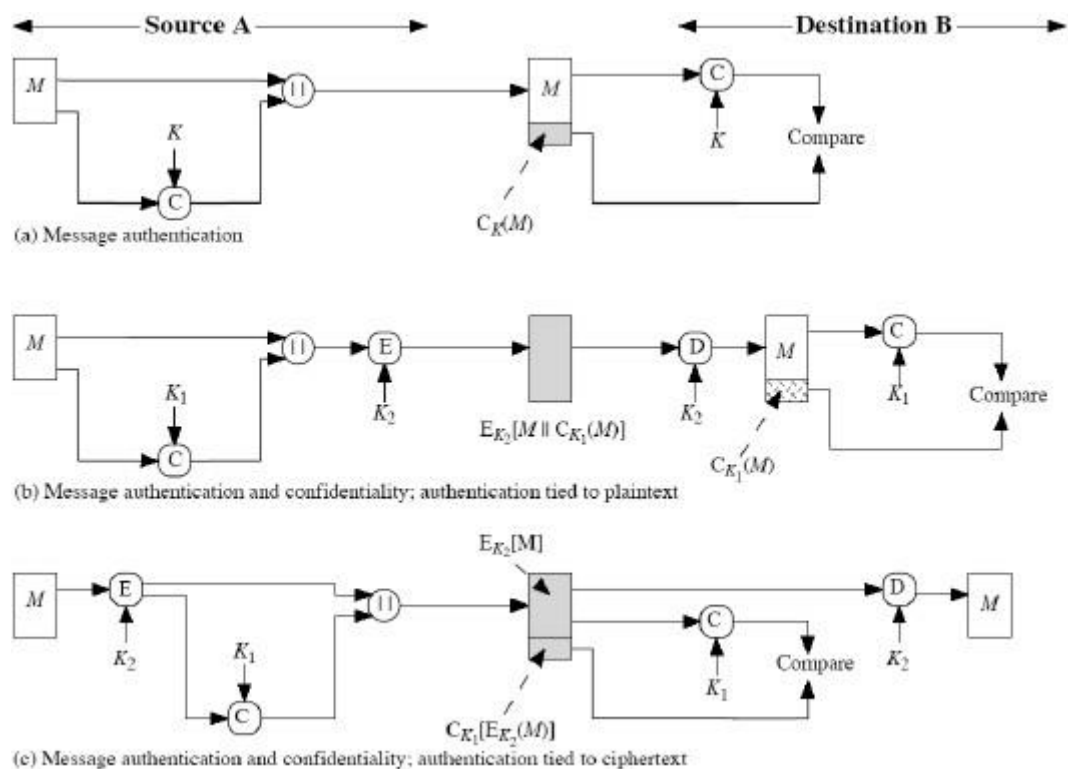## MESSAGE AUTHENTICATION CODE

Analternativeauthenticationtechniqueinvolvestheuseofasecretkeytogenerate a small fixed-size block of data, known as cryptographic checksum or MAC,whichisappendedtothemessage.Thistechniqueassumesthatboththecommunicatingpar ties say A and B share a common secret key K. When A has a message to send to B, itcalculatesMACasafunctionCofkeyandmessagegivenas: **MAC=Ck(M)** Themessage

andtheMACaretransmittedtotheintendedrecipient,whouponreceivingperformsthesame calculation on the received message, using the same secret key to generate a newMAC.ThereceivedMAC iscomparedtothecalculatedMACandonlyiftheymatch,then:

1. Thereceiverisassuredthatthemessagehasnotbeenaltered:Anyalternationsbeendonethe MAC'sdo notmatch.

2. The receiver is assured that the message is from the alleged sender: No one except thesenderhasthe secretkey andcouldprepareamessagewithaproperMAC.

3. Ifthemessageincludesasequencenumber,thenreceiverisassuredofpropersequenceasanatt acker cannotsuccessfullyalter thesequencenumber.

BasicusesofMessageAuthenticationCode(MAC)areshowninthefigure:

(a) Message authentication

(b) Message authentication and confidentiality; authentication tied to plaintext

(c) Message authentication and confidentiality; authentication tied to ciphertext

TherearethreedifferentsituationswhereuseofaMACisdesirable:

If a message is broadcast to several destinations in a network (such as a militarycontrol center), then it is cheaper and more reliable to have just one node responsible toevaluatetheauthenticity–messagewillbesentinplainwithanattachedauthenticator.

If onesidehasaheavyload,itcannotaffordtodecryptallmessages– itwilljustchecktheauthenticityofsome randomlyselectedmessages.

Authentication of computer programs in plaintext is very attractive service as theyneednotbedecryptedeverytimewastingofprocessorresources.Integrityoftheprogramca nalwaysbecheckedbyMAC.

MESSAGE AUTHENTICATION CODE BASED ON DES

The Data Authentication Algorithm, based on DES, has been one of the most widely usedMACsforanumberofyears.ThealgorithmisbothaFIPSpublication(FIPSPUB113)andanAN SIstandard(X9.17).But,securityweaknessesinthisalgorithmhavebeendiscoveredanditisbein greplacedbynewerandstrongeralgorithms.Thealgorithmcanbe defined as using the cipher block chaining (CBC) mode of operation of DES shownbelowwith aninitialization vectorofzero.
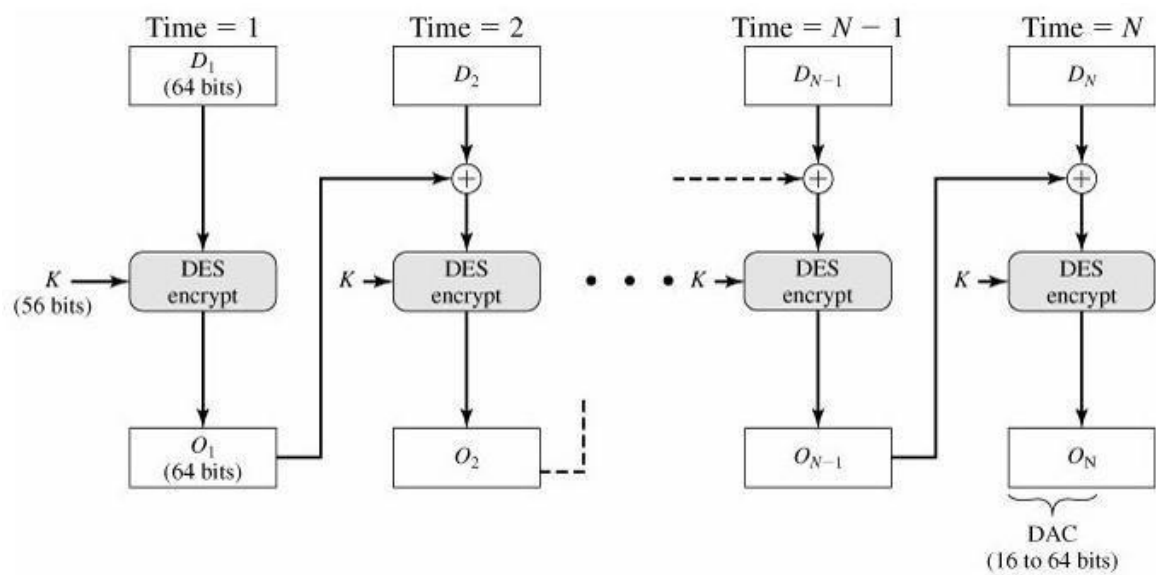
The data (e.g., message, record, file, or program) to be authenticated are grouped intocontiguous64-

bitblocks:D1,D2,...,DN.Ifnecessary,thefinalblockispaddedontherightwithzeroestoformafull6 4-bitblock.UsingtheDESencryptionalgorithm,E,andasecretkey,K,adataauthentication

$$O_1 = E(K, D_1)$$
$$O_2 = E(K, [D_2 \oplus O_1])$$
$$O_3 = (K, [D_3 \oplus O_2])$$

$\bullet$

$\bullet$

$\bullet$

$$O_N = E(K, [D_N \oplus O_{N1}])$$

code(DAC) is calculatedasfollows:

TheDACconsistsofeithertheentireblockONortheleftmostMbitsoftheblock,with16 ≤M ≤64
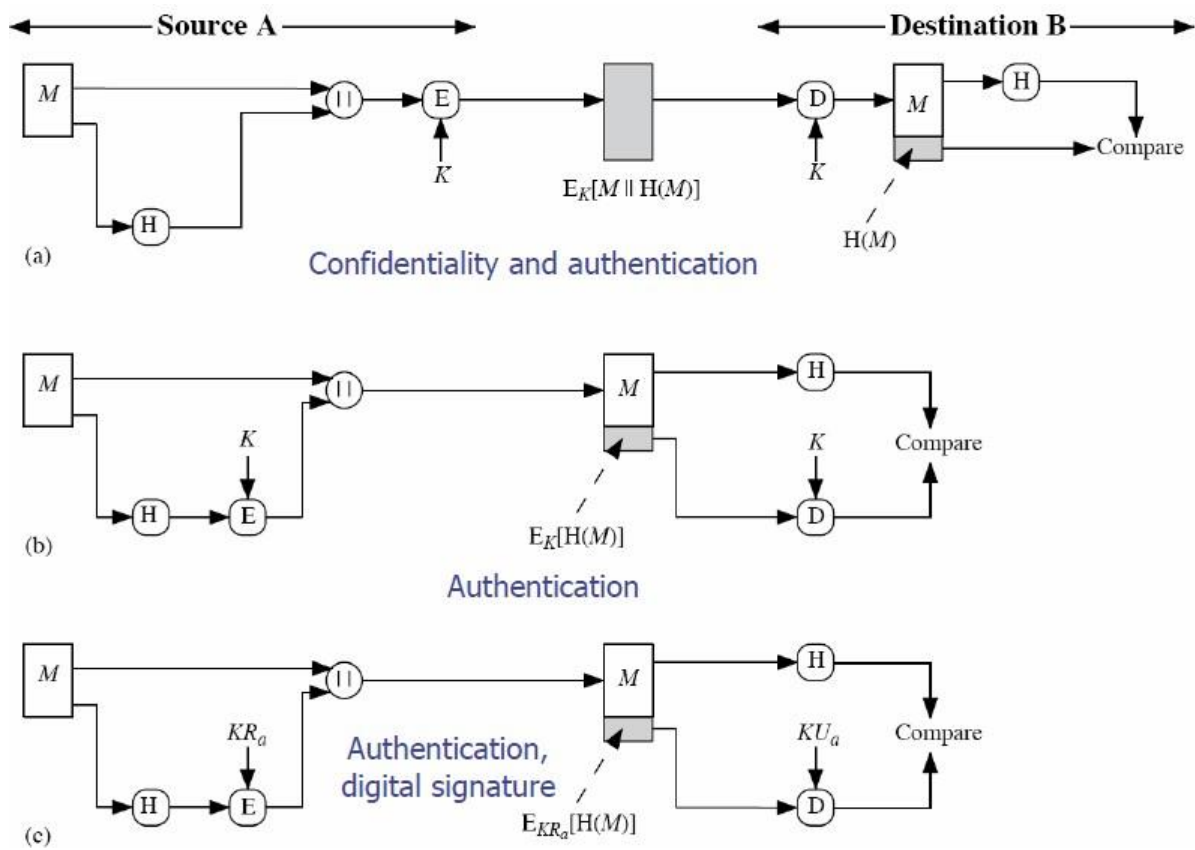
Use of MAC needs a shared secret key between the communicating parties and also MACdoes not provide digital signature. The following table summarizes the confidentialityandauthentication implicationsof theapproachesshownabove.

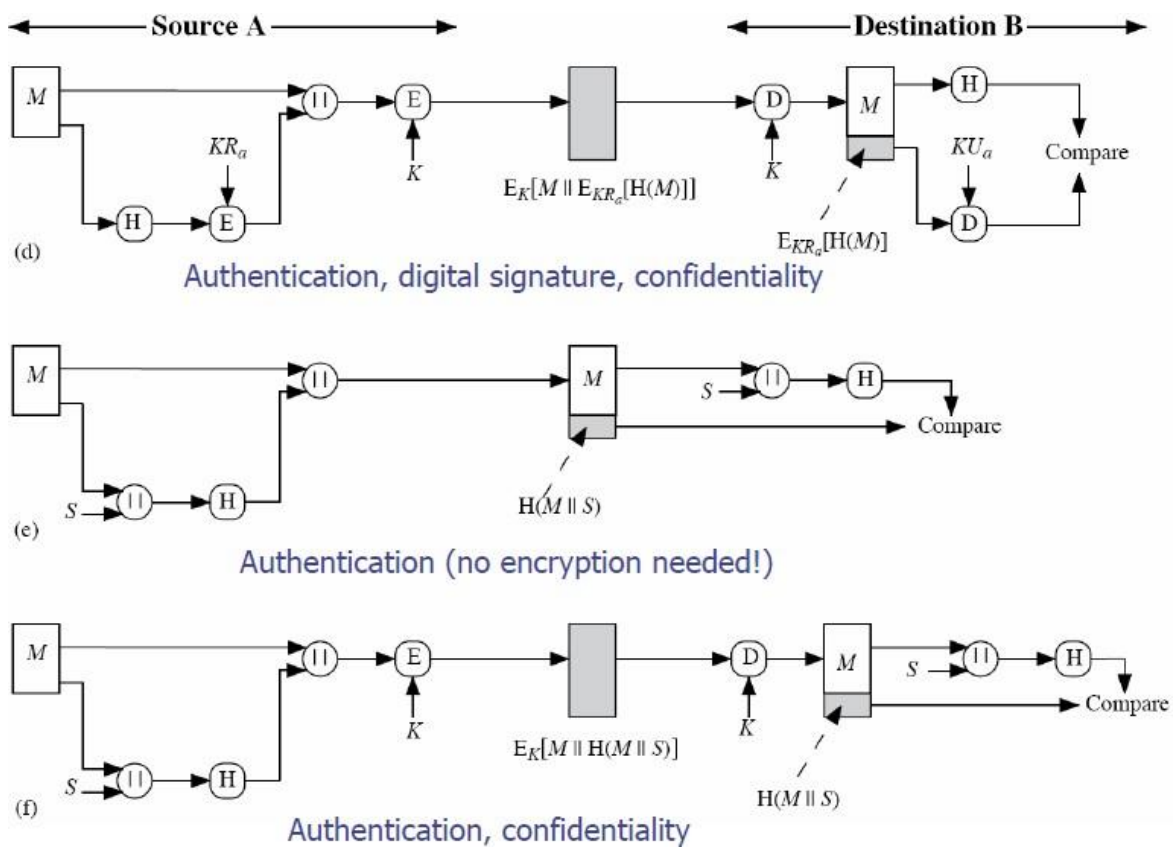$$A \rightarrow B: M \,\|\, C_K(M)$$
- Provides authentication
  — Only A and B share K

(a) Message authentication

$$A \rightarrow B: E_{K_2}\left[M \,\|\, C_{K_1}(M) \right]$$
- Provides authentication
  — Only A and B share K_1
- Provides confidentiality
  — Only A and B share K_2

(b) Message authentication and confidentiality: authentication tied to plaintext

$$A \rightarrow B: E_{K_2}[M] \,\|\, C_{K_1}\left(E_{K_2}[M] \right)$$
- Provides authentication
  — Using K_1
- Provides confidentiality
  — Using K_2

(c) Message authentication and confidentiality: authentication tied to ciphertext

HASHFUNCTION

A variation on the message authentication code is the one-way hash function. Aswith the message authentication code, the hash function accepts a variable-size messageMasinputandproducesafixed-

sizehashcodeH(M),sometimescalledamessagedigest,as output. The hash code is a function of all bits of the message and provides an error-detection capability: A change to any bit or bits in the message results in a change to thehash code. A variety of ways in which a hash code can be used to provide messageauthenticationis shownbelowandexplainedstepwise inthetable.

A → B: $E_K[M \parallel H(M)]$ •Provides confidentiality —Only A and B share K •Provides authentication —H(M) is cryptographically protected (a) Encrypt message plus hash code	A → B: $E_K[M \parallel E_{KR_a}[H(M)]]$ •Provides authentication and digital signature •Provides confidentiality —Only A and B share K (d) Encrypt result of (c) - shared secret key
A → B: $M \parallel E_K[H(M)]$ •Provides authentication —H(M) is cryptographically protected (b) Encrypt hash code - shared secret key	A → B: $M \parallel H(M \parallel S)$ •Provides authentication —Only A and B share S (e) Compute hash code of message plus secret value
A → B: $M \parallel E_{KR_a}[H(M)]$ •Provides authentication and digital signature —H(M) is cryptographically protected —Only A could create $E_{KR_a}[H(M)]$ (c) Encrypt hash code - sender's private key	A → B: $E_K[M \parallel H(M) \parallel S]$ •Provides authentication —Only A and B share S •Provides confidentiality —Only A and B share K (f) Encrypt result of (e)

(a) Confidentiality and authentication

(b) Authentication

(c) Authentication, digital signature

(d) Authentication, digital signature, confidentiality

(e) Authentication (no encryption needed!)

(f) Authentication, confidentiality

Incaseswhereconfidentialityisnotrequired,methodsbandchaveanadvantageoverthos ethatencrypttheentiremessageinthatlesscomputationisrequired.Growinginterest   for techniques that avoid encryption is due to reasons like, Encryption softwareis quite slow and may be covered by patents. Also encryption hardware costs are notnegligible and the algorithms   are   subject   to   U.S   export   control.   A   fixed-length   hash valuehisgeneratedbyafunctionHthattakesasinputamessageofarbitrarylength:**h=H(M).**

A sendsMandH(M)

B authenticatesthemessagebycomputingH(M)andcheckingthematch

Requirements for a hash function: The purpose of a hash function is to produce a"fingerprint"ofafile,message,orotherblockofdata.Tobeusedformessageauthentication,theh ash functionH musthavethefollowingproperties

H can be applied to a message of any

H sizeproducesfixed-lengthoutput

ComputationallyeasytocomputeH(M)foranygivenM

□□Computationallyinfeasibletofind M such that H(M)=h, for a given h, referred to as the *one-way property*

□□Computationallyinfeasibletofind M' such that H(M')=H(M), for a given M, referred to as *weak collision resistance.*

□□Computationallyinfeasibletofind M, M' with H(M)=H(M') (to resist to birthday attacks), referred to as *strong collision resistance.*

Examples of simple hash functions are:

- Bit-by-bit XOR of plaintext blocks: h=D1⊕D2⊕…⊕DN
- Rotated XOR – before each addition the hash value is rotated to the left with 1 bit
- Cipher block chaining technique without a secret key.

MD5 MESSAGE DIGEST ALGORITHM

The MD5 message-digest algorithm was developed by Ron Rivest at MIT and it remained as the most popular hash algorithm until recently. The algorithm takes as input, a message of arbitrary length and produces as output, a 128-bit message digest. The input is processed in 512-bit blocks. The processing consists of the following steps:

1.) *Append Padding bits*: The message is padded so that its length in bits is congruent to 448 modulo 512 i.e. the length of the padded message is 64 bits less than an integer multiple of 512 bits. Padding is always added, even if the message is already of the desired length. Padding consists of a single 1-bit followed by the necessary number of 0-bits.

2.) *Append length*: A 64-bit representation of the length in bits of the original message(before the padding) is appended to the result of step-1. If the length is larger than 264, the 64 least representative bits are taken.

3.) *Initialize MD buffer*: A 128-bit buffer is used to hold intermediate and final results of the hash function. The buffer can be represented as four 32-bit registers (A, B, C, D) and are initialized with A=0x01234567, B=0x89ABCDEF, C=0xFEDCBA98, D=0x76543210 i.e. 32-bit integers (hexadecimal values).
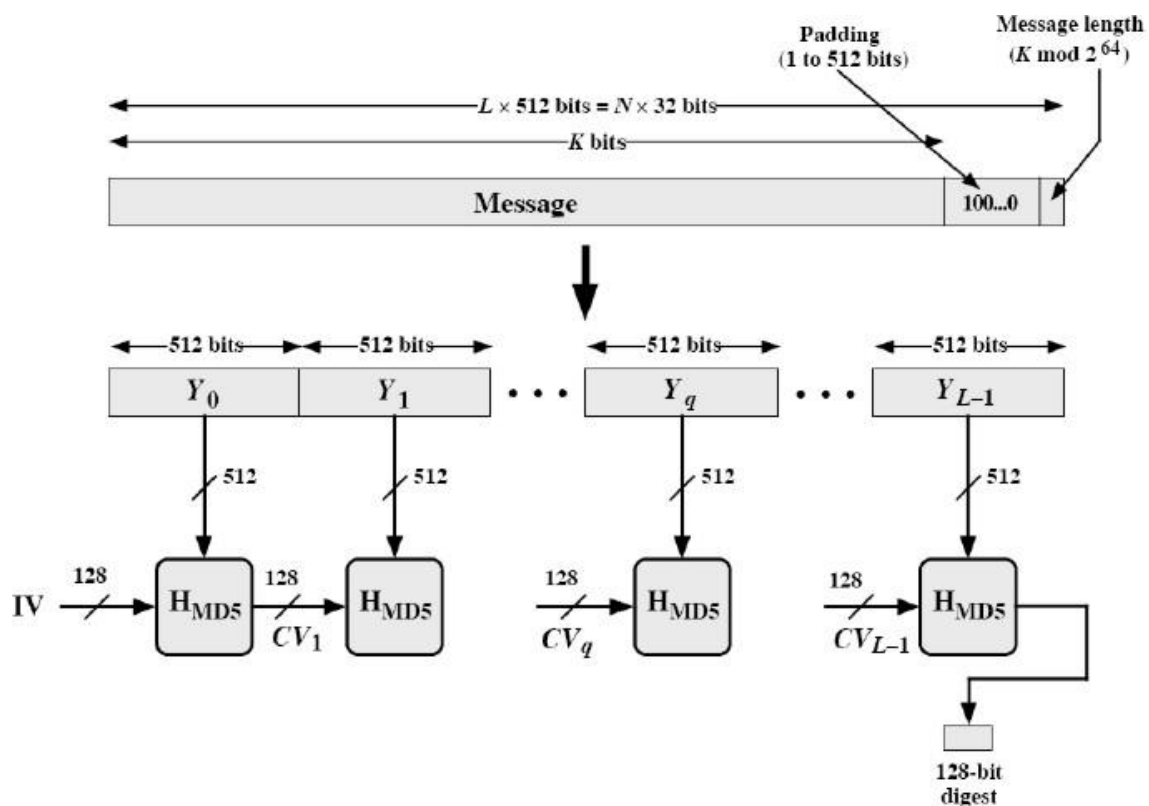
Message Digest Generation Using MD5

4.) *Process Message in 512-bit (16-word) blocks*: The heart of algorithm is the compression function that consists of four rounds of processing and this module is labeled HMD5 in the above figure and logic is illustrated in the following figure. The four rounds have a similar structure, but each uses a different primitive logical function, referred to as F, G,H and I in the specification. Each block takes as input the current 512-bit block being processed Yq and the 128-bit buffer value ABCD and updates the contents of the buffer. Each round also makes use of one-fourth of a 64- element table T*1....64+, constructed from the sine function. The ith element of T, denoted T[i], has the value equal to the integer part of 2^{32}*abs(sin(i)), where i is in radians. As the value of abs(sin(i)) is a value between 0 and 1, each element of T is an integer that can be represented in 32-bits and would eliminate any regularities in the input data. The output of fourth round is added to the input to the first round (CVq) to produce CVq+1. The addition is done independently for each of the four words in the buffer with each of the corresponding words in CVq, using addition modulo 2^{32}. This operation is shown in the figure below:

Note: addition (+) is mod 2^{32}
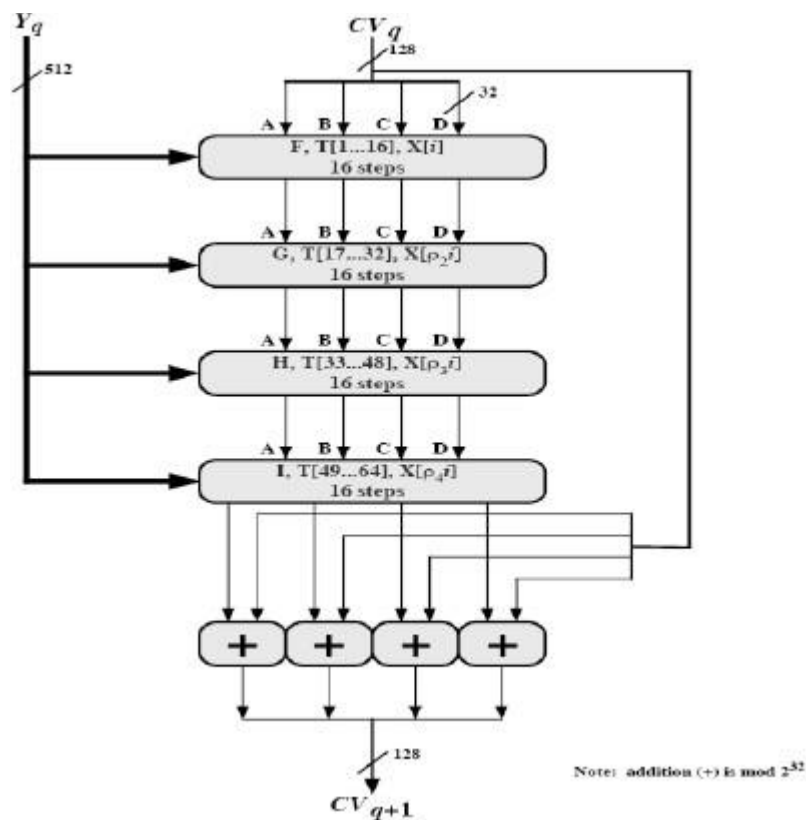
5.) *Output*: After all L 512-bit blocks have been processed, the output from the Lth stage is the 128-bit message digest. MD5 can be summarized as follows:

CV0=IV CV$_{q+1}$=SUM$_{32}$(CV$_q$,RF$_I$ Y$_q$ RF$_H$[Y$_q$,RF$_G$[Y$_q$,RF$_F$[Y$_q$,CV$_q$]]]]) MD=CV$_L$

Where,

IV = initial value of ABCD buffer, defined in step 3. Y$_q$

= the q$_{th}$ 512-bit block of the message

L = the number of blocks in the message

CV$_q$ = chaining variable processed with the q$_{th}$ block of the

message. RF$_x$ = round function using primitive logical function x.

MD = final message digest value

SUM$_{32}$ = Addition modulo 2^{32} performed separately.

MD5 Compression Function:

Each round consists of a sequence of 16 steps operating on the buffer ABCD. Each step is of the form, **a=b+((a+g(b,c,d)+X[k]+T[i])<<<s)**

where a, b, c, d refer to the four words of the buffer but used in varying permutations. After 16 steps, each word is updated 4 times. g(b,c,d) is a different nonlinear function in each round (F,G,H,I). Elementary MD5 operation of a single step is shown below.
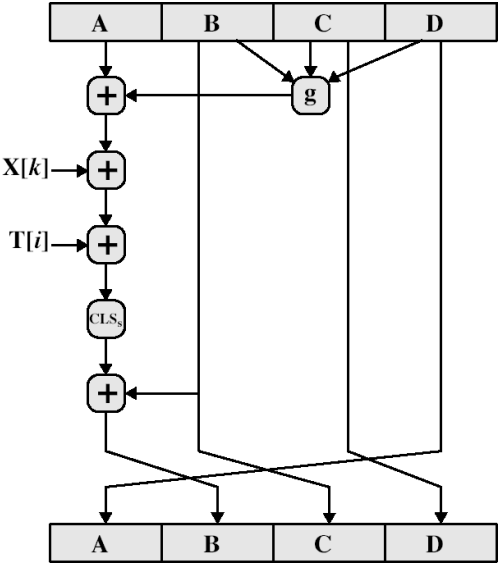
The primitive function of the F, G, H, I is given as:

<table>
<tr><th>Round</th><th>Primitive function g</th><th>g(b, c, d)</th></tr>
<tr><td>1</td><td>F(b, c, d)</td><td>(b ∧ c) ∨ (b' ∧ d)</td></tr>
<tr><td>2</td><td>G(b, c, d)</td><td>(b ∧ d) ∨ (c ∧ d')</td></tr>
<tr><td>3</td><td>H(b, c, d)</td><td>b ⊕ c ⊕ d</td></tr>
<tr><td>4</td><td>I(b c, d)</td><td>c ⊕ (b ∨ d')</td></tr>
</table>

Truth table

b	c	d	F	G	H	I
0	0	0	0	0	0	1
0	0	1	1	0	1	0
0	1	0	0	1	1	0
0	1	1	1	0	0	1
1	0	0	0	0	1	1
1	0	1	0	1	0	1
1	1	0	1	1	0	0
1	1	1	1	1	1	0

Where the logical operators (AND, OR, NOT, XOR) are represented by the symbols (∧, ∨, **~, (+))**.

Each round mixes the buffer input with the next "word" of the message in a complex, non-linear manner. A different non-linear function is used in each of the 4 rounds (but the same function for all 16 steps in a round). The 4 buffer words (a,b,c,d) are rotated from step to step so all are used and updated. g is one of the primitive functions F,G,H,I for the 4 rounds respectively. X[k] is the kth 32-bit word in the current message block. T[i] is the ith entry in the matrix of constants T. The addition of varying constants T and the use of different shifts helps ensure it is extremely difficult to compute collisions. The array of 32-bit words X[0..15] holds the value of current 512-bit input block being processed. Within a round, each of the 16 words of X[i] is used exactly once, during one step. The order in which these words is used varies from round to round. In the first round, the

wordsareusedintheiroriginalorder.Forrounds2through4,thefollowingpermutationsare used

p2(i) = (1 + 5i) mod

16p3(i) = (5 + 3i) mod

16p4(I)= 7imod 16

MD4

PrecursortoMD5

Design goals of MD4 (which are carried over to

MD5)Security

Speed

Simplicity and

compactnessFavorlittle-
   endianarchitecture

Main differences between MD5 and

A MD4fourthroundhasbeenadded.

Eachstepnowhasauniqueadditiveconstant.

The function g in round 2 was changed from (bc v bd v cd) to (bd v cd') to make g lesssymmetric.

Eachstepnowaddsintheresultofthepreviousstep.Thispromotesafaster"avalancheeff ect".

Theorderinwhichinputwordsareaccessedinrounds2and3ischanged,tomakethesepattern s lesslikeeachother.

Theshiftamountsineachroundhavebeenapproximatelyoptimized,toyieldafaster"avalancheeffect. "Theshifts in differentrounds aredistinct.

SECUREHASHALGORITHM

The secure hash algorithm (SHA) was developed by the National Institute of Standardsand Technology (NIST). SHA-1 is the best established of the existing SHA hash functions,andisemployedinseveralwidelyusedsecurityapplicationsandprotocols.Thealgorit hm takes as input a message with a maximum length of less than 264 bits andproducesas outputa160-bit messagedigest.
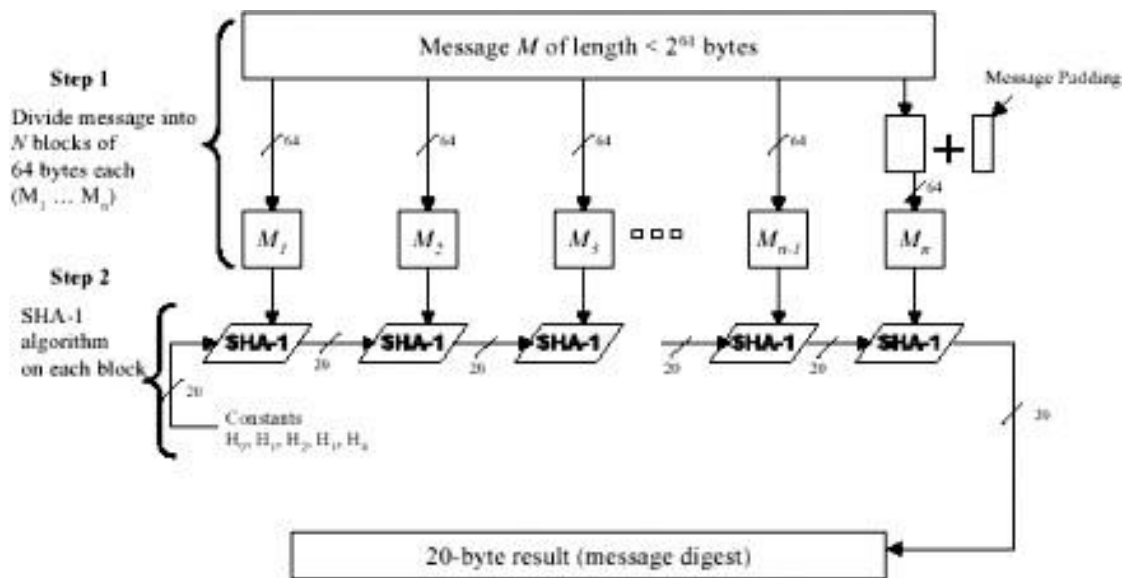
The input is processed in 512-bit blocks. The overall processing of a message follows thestructureofMD5withblocklengthof512bitsandahashlengthandchainingvariablelengthof160bits. The processingconsistsof followingsteps:

1.) **Append Padding Bits:** The message is padded so that length is congruent to 448 modulo512;paddingalwaysadded –onebit1followedbythenecessarynumberof0bits.

2.) **Append Length:** a block of 64 bits containing the length of the original message is added.3.)**InitializeMDbuffer**:A160-

bitbufferisusedtoholdintermediateandfinalresultsonthehash function. This is formed by 32-bit registers A,B,C,D,E. Initial values: A=0x67452301,B=0xEFCDAB89,C=0x98BADCFE,D=0x10325476,E=C3D2E1F0.Storesinbig-endianformat

i.e.themostsignificantbitinlowaddress.

4.) **Process message in blocks 512-bit (16-word) blocks**: The processing of a single 512-bitblock is shown above. It consists of four rounds of processing of 20 steps each. These fourrounds have similar structure, but uses a different primitive logical function, which we referto as f1, f2, f3 and f4. Each round takes as input the current 512-bit block being processedand the 160-bit buffer value ABCDE and updates the contents of the buffer. Each round alsomakes use of four distinct additive constants Kt. The output of the fourth round i.e. eightiethstepisadded tothe inputto the firstroundto produce CV$_{q+1}$.

5.) *Output:* After all L 512-bit blocks have been processed, the output from the Lth stageisthe 160-bit messagedigest.
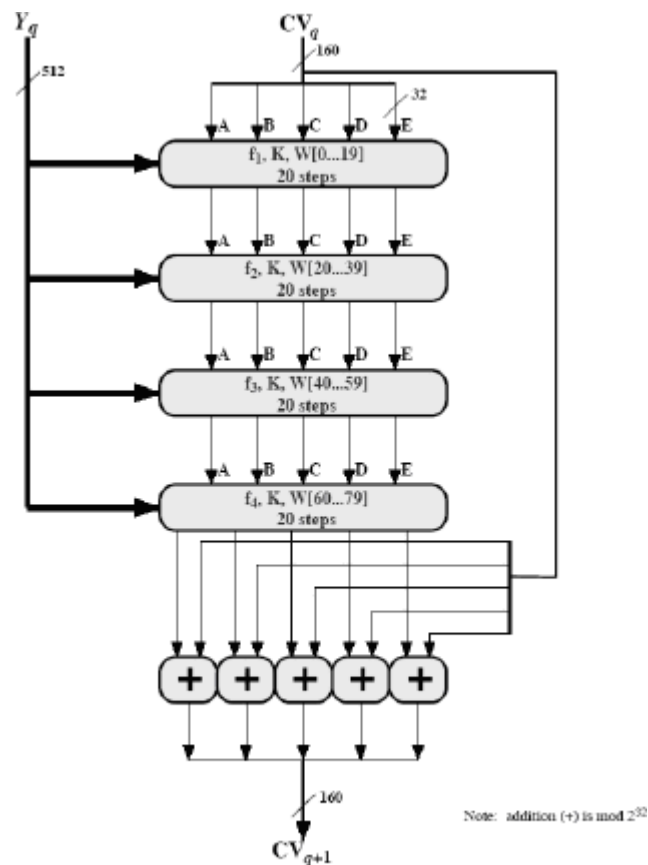
SHA-1 Processing of a Single 512-bit Block
(SHA-1 Compression Function)

ThebehaviorofSHA-1isasfollows:**CV0=IVCVq+1=SUM32(CVq,ABCDEq)MD=CVL**Where, IV = initial value of ABCDE buffer ABCDEq = output of last round of processing ofqth message block L = number of blocks in the message SUM32 = Addition modulo 232MD=finalmessage digest value.

SHA-1CompressionFunction:

Eachroundhas20stepswhichreplacesthe5bufferwords.Thelogicpresentineachoneofthe80rounds presentisgivenas**(A,B,C,D,E)<-(E+f(t,B,C,D)+S5(A)+Wt+Kt),A,S30(B),C,D**Where,A,B,C,D,E=thefivewordsofthebuffert=stepnumber;0<t

< 79 $f(t,B,C,D)$ = primitive logical function for step t S_k = circular left shift of the 32-bitargument by k bits Wt = a 32-bit word derived from current 512-bit input block. Kt = anadditiveconstant;four distinctvaluesare used+=moduloaddition.

A   B   C   D   E

f_t

S^5

$+$ W_t

S^{30} $+$ K_t

A   B   C   D   E

SHA shares much in common with MD4/5, but with 20 instead of 16 steps in each of the 4 rounds. Note the 4 constants are based on sqrt(2,3,5,10). Note also that instead of just splitting the input block into 32-bit words and using them directly, SHA-1 shuffles and mixes them using rotates & XOR's to form a more complex input, and greatly increases the difficulty of finding collisions. A sequence of logical functions f_0, f_1,..., f_{79} is used in the SHA-1. Each f_t, 0<=t<=79, operates on three 32-bit words B, C, D and produces a 32-bit word as output. f_t(B,C,D) is defined as follows: for words B,C,D, **f_t(B,C,D)=(B AND C) OR ((NOT B) AND D) (0<=t<=19) f_t(B,C,D)=B XOR C XOR D (20<=t<=39) f_t(B,C,D)=(B AND C) OR (B AND D) OR (C AND D) (40<=t<=59) f_t(B,C,D)=B XOR C XOR D (60 <=t<=79).**

WHIRLPOOL HASH FUNCTION

• Created by Vincent Rijmen and Paulo S.L.M. Barreto

• Hashes messages of plaintext length 2^{256}

• Result is a 512 bit message

• Three versions have been released – WHIRLPOOL-0 – WHIRLPOOL-T – WHIRLPOOL

> designed specifically for hash function use

> with security and efficiency of AES

> but with 512-bit block size and hence hash

> similar structure & functions as AES but

- inputismappedrowwise
- has10rounds
- adifferentprimitivepolynomialforGF(2^8)
- usesdifferentS-boxdesign& values

• "W"isa512-bitblockcipher

• "m"istheplaintext,splitinto512bitblocks

• "H"istheblocksformedfromthehashes

WHIRLPOOL OVERVIEW

Note: triangular hatch marks key input
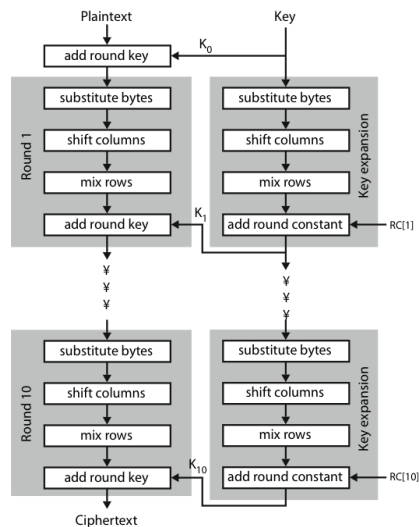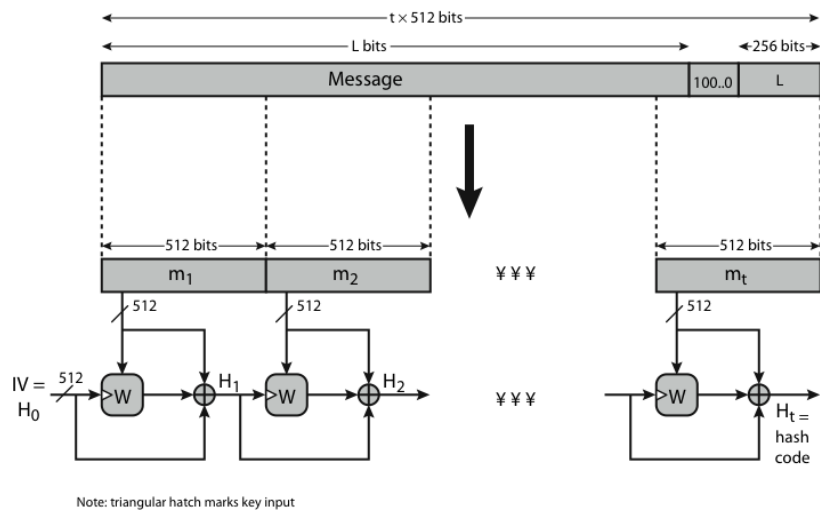
• TheblockcipherWis thecoreelementoftheWhirlpoolhashfunction

• Itiscomprisedof 4steps.

      – AddRoundKey

      – ShiftColumns

      – MixRows

      – Substitutebytes

Add Round Key

• DuringtheAddRoundKeystep,themessageisXOR'dwiththekey

• Ifthisisthefirstmessageblockbeingrunthrough,thekeyisablockofallzeros

• Ifthisisanyblockexceptthefirst,thekeyisthedigestofthepreviousblock

ShiftColumns

• Starting from left to right, each column gets rotated vertically a number of bytes

equaltowhich number columnitis,fromtop tobottom–

Ex:

        • [0,0][0,1][0,2][0,0][2,1][1,2]

        • [1,0][1,1][1,2]------>[1,0][0,1][2,2]

        • [2,0][2,1][2,2][2,0][1,1][0,2]

MixRows

• Each row gets shifted horizontally by the number of row it is. Similar to the

shiftcolumnfunction,butrotatedlefttoright–

Ex:

        • [0,0][0,1][0,2][0,0][0,1][0,2]

        • [1,0][1,1][1,2]------>[1,2][1,0][1,2]

        • [2,0][2,1][2,2][2,1][2,2][0,2]

Substitutebytes

• Eachbyteinthemessageispassedthrougha setofs-boxes

• Theoutputofthisisthensettobethekeyforthenextround

HMAC

InterestindevelopingaMAC,derivedfromacryptographichashcodehasbeenincreasingmainlyb
ecausehashfunctionsaregenerallyfasterandarealsonotlimitedbyexport restrictions unlike
block ciphers. Additional reason also would be that the librarycode for cryptographic hash
functions is widely available. The original proposal is forincorporation of a secret key into
an existing hash algorithm and the approach
thatreceivedmostsupportisHMAC.HMACisspecifiedasInternetstandardRFC2104.It

makes use of the hash function on the given message. Any of MD5, SHA-1, RIPEMD-160 canbe used.

HMACDesignObjectives

⬚⬚Touse,withoutmodifications,availablehashfunctions

⬚⬚To allow for easy replaceability of the embedded hash

⬚⬚functionTopreservetheoriginalperformanceofthehashfunction

⬚⬚Touseandhandlekeysina simpleway

⬚⬚TohaveawellunderstoodcryptographicanalysisofthestrengthoftheMACbasedonreasonableassumptions on theembeddedhash function

The first two objectives are very important for the acceptability of HMAC. HMAC treatsthe hash function as a "black box", which has two benefits. First is that an existingimplementation of the hash function can be used for implementing HMAC making thebulk of HMAC code readily available without modification. Second is that if ever anexistinghashfunctionistobereplaced,theexistinghashfunctionmoduleisremovedandnew module is dropped in. The last design objective provides the main advantage ofHMAC over other proposed hash-based schemes. HMAC can be proven secure providedthattheembeddedhashfunctionhassomereasonablecryptographicstrengths.

StepsinvolvedinHMACalgorithm:

1. AppendzeroestotheleftendofKtocreateab-bitstringK_+(ex:IfKisoflength160-bitsandb =512,then Kwill beappendedwith 44zerobytes).

2. XOR(bitwiseexclusive-OR)K_+ withipadtoproducetheb-bitblockS_i.

3. AppendM toS_i.

4. NowapplyHtothestreamgeneratedinstep-3

5. XORK_+withopadtoproduce theb-bitblockS_0.

6. Appendthehashresult fromstep-4toS_0.

7. ApplyHtothestreamgeneratedinstep-6andoutputtheresult.

HMACAlgorithm

- **Define the following terms**

  H   = embedded hash function

  M   = message input to HMAC

  Y_i = i^{th} block of M, $0 \leq i \leq L-1$

  L   = number of blocks in M

  b   = number of bits in a block

  n   = length of hash code produced by embedded hash function

  K   = secret key; if key length is greater than b, the key is input to the hash function to produce an n-bit key; recommended length $\geq$ n

  K^+ = K padded with 0's on the left so that the result is b bits in length

  ipad = 00110110 repeated b/8 times

  opad = 01011100 repeated b/8 times

- **Then HMAC can be expressed as**

$$HMAC_K = H[\ (K^+ \oplus opad)\ ||\ H[K^+ \oplus ipad)\ ||\ M]\ ]$$

HMACStructure
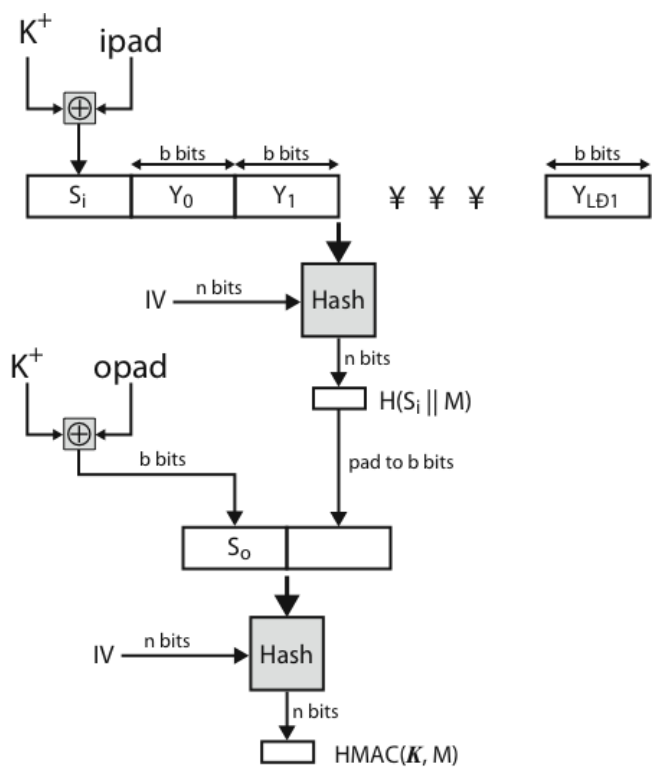

The XOR with ipad results in flipping one-half of the bits of K. Similarly, XOR with opadresults in flipping one-half of the bits of K, but different set of bits. By passing S_i and S_0throughthecompressionfunctionofthehashalgorithm,wehavepseudorandomlygeneratedt wokeys fromK.

HMACshouldexecuteinapproximatelythesametimeastheembeddedhashfunctionforlong messages. HMAC adds three executions of the hash compression function (for S_0, S_i,andtheblockproduced fromtheinner hash)

A more efficient implementation is possible. Two quantities are precomputed.$f(IV,(K^+ \oplus ipad)$

$f(IV,(K^+ \oplus opad)$

where f is the compression function for the hash function which takes as arguments achainingvariableofnbitsandablockofb-bitsandproducesachainingvariableofnbits.
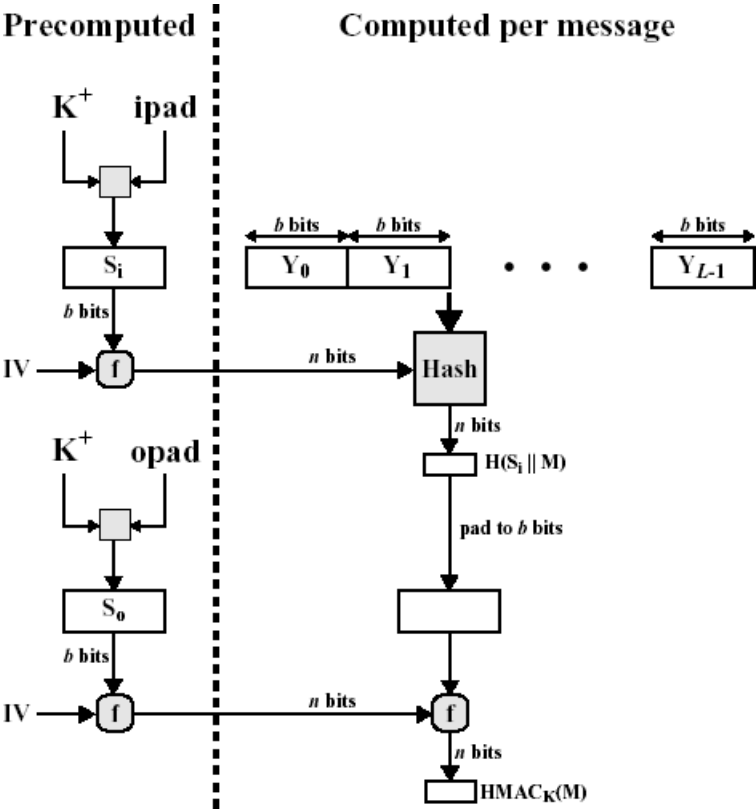
As shown in the above figure, the values are needed to be computed initially and everytimeakeychanges.Theprecomputedquantitiessubstitutefortheinitialvalue(IV)inthehashfunction.Withthisimplementation,onlyoneadditionalinstanceofthecompressionfunctionisaddedtotheprocessingnormallyproducedbythehashfunction.Thisimplementation   is worthwhile if most of the messages for which a MAC is computed areshort.

SecurityofHMAC:

The appeal of HMAC is that its designers have been able to prove an exact relationshipbetweenthestrengthoftheembeddedhashfunctionandthestrengthofHMAC.The

securityofaMACfunctionisgenerallyexpressedintermsoftheprobabilityofsuccessfulforgery with a given amount of time spent by the forger and a given number of message-MAC pairs created with the same key. Have two classes of attacks on the embedded hashfunction:

1. TheattackerisabletocomputeanoutputofthecompressionfunctionevenwithanIVthatis random, secretandunknowntotheattacker.

2. The attacker finds collisions in the hash function even when the IV is random andsecret.

Theseattacksarelikelytobecausedbybruteforceattackonkeyusedwhichhasworkoforder $2n$; or a birthday attack which requires work of order $2_{(n/2)}$ - but which requirestheattackertoobserve2nblocksofmessagesusingthesamekey-

veryunlikely.SoevenMD5is still secure foruse inHMACgiven theseconstraints.

CMAC

In cryptography, **CMAC** (Cipher-based Message Authentication Code)[1] is a block cipher-basedmessageauthenticationcodealgorithm.Itmaybeusedtoprovideassuranceoftheauthenti cityand,hence,theintegrityofbinarydata.ThismodeofoperationfixessecuritydeficienciesofCB C-MAC(CBC-MACissecure onlyforfixed-lengthmessages).

The core of the CMAC algorithm is a variation of CBC-MACthatBlackandRogawayproposedandanalyzedunderthenameXCBC[2]andsubmitted to NIST.[3]The XCBC algorithm efficiently addresses the security deficienciesof CBC-MAC, but requires three keys. Iwata and Kurosawa proposed an improvement ofXCBCandnamedtheresultingalgorithmOne-KeyCBC-MAC(OMAC)intheirpapers.[4][5]TheylatersubmittedOMAC1[6],arefinementofOMAC,andadditi onalsecurityanalysis.[7]TheOMACalgorithmreducestheamountofkeymaterialrequiredforXCB C.CMACis equivalent toOMAC1.
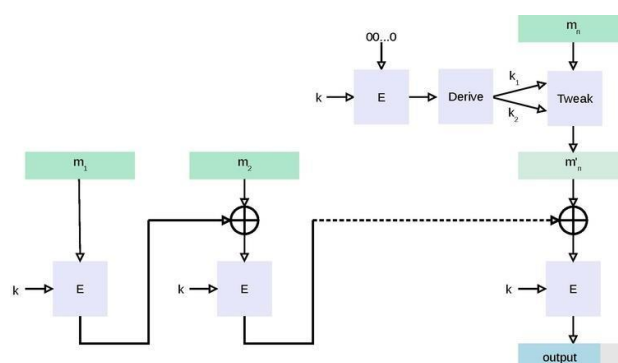
To generate an ℓ-bit CMAC tag (t) of a message (m) using a b-bit block cipher (E) and a secret key (k), one first generates two b-bit sub-keys (k_1 and k_2) using the following algorithm (this is equivalent to multiplication by x and x^2 in a [finite field](#) GF(2^b)). Let $\ll$ denote the standard left-shift operator and $\oplus$ denote [exclusive or](#):

1. Calculate a temporary value $k_0 = E_k(0)$.
2. If $msb(k_0) = 0$, then $k_1 = k_0 \ll 1$, else $k_1 = (k_0 \ll 1) \oplus C$; where C is a certain constant that depends only on b. (Specifically, C is the non-leading coefficients of the lexicographically first irreducible degree-b binary polynomial with the minimal number of ones.)

3. If $msb(k_1) = 0$, then $k_2 = k_1 \ll 1$, else $k_2 = (k_1 \ll 1) \oplus C$.

4. Return keys (k_1, k_2) for the MAC generation process.

As a small example, suppose $b = 4$, $C = 0011_2$, and $k_0 = E_k(0) = 0101_2$. Then $k_1 = 1010_2$ and $k_2 = 0100 \oplus 0011 = 0111_2$.

The CMAC tag generation process is as follows:

1. Divide message into b-bit blocks $m = m_1 \| ... \| m_{n-1} \| m_n$ where $m_1, ..., m_{n-1}$ are complete blocks. (The empty message is treated as 1 incomplete block.)
2. If m_n is a complete block then $m_n' = k_1 \oplus m_n$ else $m_n' = k_2 \oplus (m_n \| 10...0_2)$.

3. Let $c_0 = 00...0_2$.
4. For $i = 1, ..., n\text{-}1$, calculate $c_i = E_k(c_{i-1} \oplus m_i)$.
5. $c_n = E_k(c_{n-1} \oplus m_n')$
6. Output $t = msb_\ell(c_n)$.

The verification process is as follows:

1. Use the above algorithm to generate the tag.
2. Check that the generated tag is equal to the received tag.

DIGITAL SIGNATURE

The most important development from the work on public-key cryptography is the digital signature. Message authentication protects two parties who exchange messages from any third party. However, it does not protect the two parties against each

other.Adigitalsignatureisanalogoustothehandwrittensignature,andprovidesasetofsecurity capabilities that would be difficult to implement in any other way. It must havethefollowingproperties:•Itmustverifytheauthorandthedateandtimeofthesignature

• Itmusttoauthenticatethecontentsatthetimeofthesignature•Itmustbeverifiablebythirdparti es,toresolvedisputesThus,thedigitalsignaturefunctionincludestheauthenticationfunction.Av arietyofapproacheshasbeenproposedforthedigitalsignature function. These approaches fall into two categories: direct and arbitrated.**DirectDigitalSignature**

DirectDigitalSignaturesinvolvethedirectapplicationofpublic-keyalgorithmsinvolvingonly the communicating parties. A digital signature may be formed by encrypting theentiremessagewiththesender'sprivatekey,orbyencryptingahashcodeofthemessagewith the sender's private key. Confidentiality can be provided by further encrypting theentire message plus signature using either public or private key schemes. It is importanttoperformthesignaturefunctionfirstandthenanouterconfidentialityfunction,sincei ncase of dispute, some third party must view the message and its signature. But theseapproachesaredependentonthesecurityofthesender'sprivate-key.Willhaveproblemsif it is lost/stolen and signatures forged. Need time-stamps and timely key revocation.**ArbitratedDigitalSignature**

The problems associated with direct digital signatures can be addressed by using anarbiter, in a variety of possible arrangements. The arbiter plays a sensitive and crucialrole in this sort of scheme, and all parties must have a great deal of trust that thearbitration mechanism is working properly. These schemes can be implemented witheither private or public-key algorithms, and the arbiter may or may not see the actualmessagecontents.**UsingConventional encryption**

$$X \quad : M \ || \ E(\ Kxa,[IDx|| \ H \ (M)])$$
$$A \quad Y:E(Kay,[IDx|| \ M||E(Kxa,[IDx||H(M))])|| \ T])$$

It isassumedthatthesenderXandthearbiterAshareasecretkeyKxaandthatAandYsharesecr etkeyKay.XconstructsamessageMandcomputesitshashvalueH(m).ThenX transmits the message plus a signature to A. the signature consists of an identifier IDxofXplusthehashvalue,allencryptedusingKxa.

A decrypts the signature and checks the hash value to validate the message. Then Atransmits a message to Y, encrypted with Kay. The message includes IDx, the originalmessagefromX, thesignature, andatimestamp.

🞂🞂Arbiterseesmessage

🞂🞂Problem : the arbiter could form an alliance with sender to deny a signed message, orwiththereceiver toforge the sender's signature.

UsingPublicKeyEncryption

$$X \rightarrow A \quad :IDx||E(PRx,[IDx||E(PUy,E(PRx,M))])$$

$$A \rightarrow Y: E(PRa,[IDx||E(PUy, E (PRx,M))|| T])$$

🞂🞂XdoubleencryptsamessageMfirstwithX'sprivatekey,PRx,andthenwithY'spublickey, PUy. This is a signed, secret version of the message. This signed message, togetherwith X's identifier , is encrypted again with PRx and, together with IDx, is sent to A. Theinner,doubleencryptedmessageissecurefromthearbiter(andeveryoneelseexceptY)

🞂🞂A can decrypt the outer encryption to assure that the message must have come fromX (because only X has PRx). Then A transmits a message to Y, encrypted with PRa. ThemessageincludesIDx, thedoubleencryptedmessage,andatimestamp.

🞂🞂Arbiterdoesnotseemessage

DigitalSignatureStandard(DSS)

The National Institute of Standards and Technology (NIST) has published FederalInformation Processing Standard FIPS 186, known as the Digital Signature Standard(DSS).TheDSSmakesuseoftheSecureHashAlgorithm(SHA)andpresentsanewdigitals ignature technique, the Digital Signature Algorithm (DSA). The DSS uses an algorithmthat is designed to provide only the digital signature function and cannot be used forencryptionorkey exchange,unlike RSA.

The RSA approach is shown below. The message to be signed is input to a hashfunctionthatproducesasecurehashcodeoffixedlength.Thishashcodeisthenencryptedusi ngthesender'sprivatekeytoformthesignature.Boththemessageandthesignatureare then transmitted.
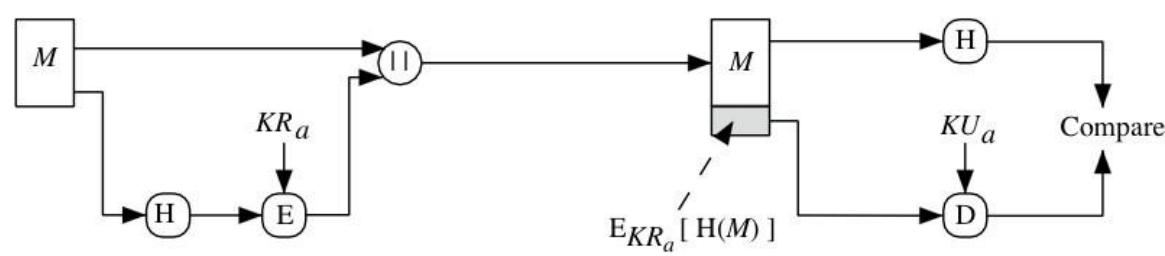
The recipient takes the message and produces a hash code. The recipient alsodecrypts the signature using the sender's public key. If the calculated hash code matchesthe decrypted signature, the signature is accepted as valid. Because only the senderknowstheprivate key,onlythesendercouldhaveproducedavalidsignature.

TheDSSapproachalsomakesuseofahashfunction.Thehashcodeisprovidedasinputtoasi gnaturefunctionalongwitharandomnumberkgeneratedforthisparticularsignature. The signature function also depends on the sender's private key (PRa) and aset of parameters known to a group of communicating principals. We can consider thisset to constitute a global public key (PUG).The result is a signature consisting of twocomponents, labeleds andr.

(b) DSS approach
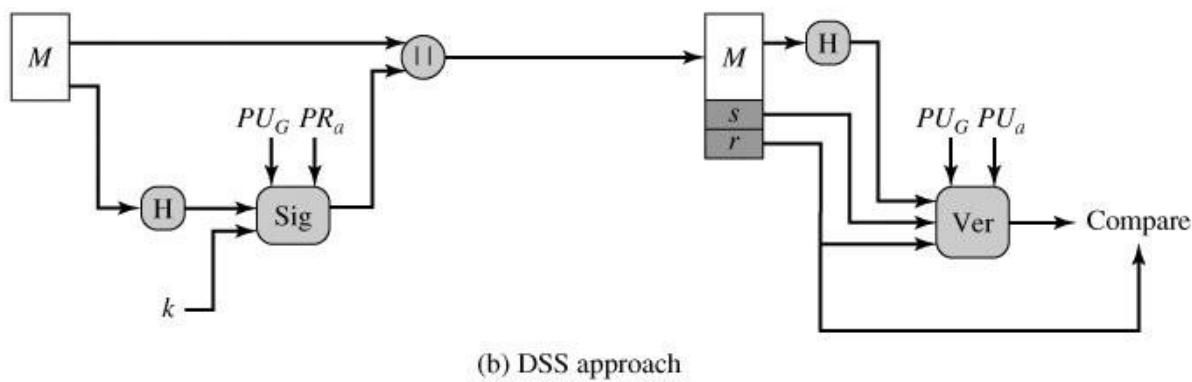
Atthereceivingend,thehashcodeoftheincomingmessageisgenerated.Thisplusthesigna tureisinputtoaverificationfunction.Theverificationfunctionalsodependsonthe global public key as well as the sender's public key (PUa), which is paired with thesender's private key. The output of the verification function is a value that is equal to thesignature component r if the signature is valid. The signature function is such that onlythesender,withknowledgeoftheprivatekey,couldhaveproducedthevalidsignature.

KNAPSACKALGORITHM

Public-Key cryptography was invented in the 1970s by Whitfield Diffie, Martin HellmanandRalph Merkle.

Public-keycryptographyneedstwokeys.Onekeytellsyouhowtoencrypt(orcode)amessage and this is "public" so anyone can use it. The other key allows you to decode(or decrypt) the message. This decryption code is kept secret (or private) so only thepersonwhoknowsthekeycandecryptthemessage.Itisalsopossiblefortheperson

withtheprivatekeytoencryptamessagewiththeprivatekey,thenanyoneholdingthepublic key can decrypt the message, although this seems to be of little use if you aretryingtokeep something secret!

The First General Public-Key Algorithm used what we call the Knapsack Algorithm.Althoughwenowknowthatthisalgorithmisnotsecurewecanuseittolookathowthe setypes ofencryptionmechanismswork.

Theknapsackalgorithmworkslikethis:

Imagine you have a set of different weights which you can use to make any total weightthatyouneedbyaddingcombinations ofanyofthese weightstogether.

Letuslookatanexample:

Imagine you had a set of weights 1, 6, 8, 15 and 24. To pack a knapsack weighing 30, youcould use weights 1, 6, 8 and 15. It would not be possible to pack a knapsack that weighs17butthismightnot matter.

You might represent the weight 30 by the binary code 11110 (one 1, one 6, one 8, one15andno24).

Example:

Plaintext	10011	11010	01011	00000
Knapsack	1681524	1681524	1681524	1681524
Ciphertext	1+15+24=40	1+6+15=22	6+15+24=45	0=0

Whattotalweightsisitpossibletomake?

So, if someone sends you the code 38 this can only have come from the plain text 01101.When the Knapsack Algorithm is used in public key cryptography, the idea is to createtwo different knapsack problems. One is easy to solve, the other not. Using the easyknapsack, the hard knapsack is derived from it. The hard knapsack becomes the publickey.Theeasyknapsackistheprivatekey.Thepublickeycanbeusedtoencryptmessages,but cannotbe usedtodecryptmessages. Theprivate keydecryptsthemessages.

TheSuperincreasingKnapsackProblem

An easy knapsack problem is one in which the weights are in a superincreasingsequence. A superincreasing sequence is one in which the next term of the sequence isgreaterthanthesumofallprecedingterms.Forexample,theset{1,2,4,9,20,38}is superincreasing,buttheset{1,2,3,9,10,24}isnotbecause10<1+2+3+9.

It is easy to solve a superincreasing knapsack. Simply take the total weight of theknapsack and compare it with the largest weight in the sequence. If the total weight isless than the number, then it is not in the knapsack. If the total weight is greater thenthenumber,itisintheknapsack.Subtractthenumberfromthetotal,andcomparewiththe next highest number. Keep working this way until the total reaches zero. If the totaldoesn'treach zero, then thereisno solution.

So,forexample,ifyouhaveaknapsackthatweighs23thathasbeenmadefromtheweightsofthes uperincreasingseries{1,2,4,9,20,38}thenitdoesnotcontainthe
weight38(as38> 23)
butitdoescontaintheweight20;leaving3;
which does not contain the weight 9 still leaving
3;whichdoesnotcontaintheweight4stillleaving3;
which contains the weight 2, leaving 1; which contains the weight
1.Thebinarycode is therefore 110010.

It is much harder to decrypt a non-superincreasing knapsack problem. Give a friend anon-superincreasingknapsackandatotalandseewhythisisthecase.
One algorithm that uses a superincreasing knapsack for the private (easy) key and anon-superincreasing knapsack for the public key was created by Merkle and HellmanThey did this by taking a superincreasing knapsack problem and converting it into anon-superincreasingonethatcouldbemadepublic, usingmodulusarithmetic.

MakingthePublicKey

To produce a normal knapsack sequence, take a superincreasing sequence; e.g. {1, 2, 4,10, 20, 40}. Multiply all the values by a number, n, modulo m. The modulus should be anumbergreaterthanthesumofallthenumbersinthesequence,forexample,110.The

multiplier should have no factors in common with the modulus. So let's choose 31.

Thenormalknapsacksequence wouldbe:

1×31mod(110)=31
2×31mod(110)=62
4×31mod(110)=14
10×31mod(110)=90
20×31mod(110)=70
40×31mod(110)=30

Sothepublickeyis:{31,62,14,90,70,30}and
theprivatekeyis{1,2,4,10,20.40}.

Let's try to send a message that is in binary
code:100100111100101110
The knapsack contains six weights so we need to split the message into groups of
six:100100
111100
101110
Thiscorrespondstothreesetsofweightswithtotalsasfollows100100
=31+90=121
111100 =31+62+14+90=197
101110=31+14+90+70=205
Sothecodedmessageis121197205.

Nowthereceiverhasto decodethemessage...
The person decoding must know the two numbers 110 and 31 (the modulus and
themultiplier).Let'scallthemodulus"m"andthenumberyoumultiplyby"n".
We need $n-1$, which is a multiplicative inverse of n mod m, i.e. $n(n-1) = 1$ mod

mInthis case Ihave calculated $n-1$tobe 71.

All you then have to do is multiply each of the codes 71 mod 110 to find the total in theknapsackwhichcontains {1,2,4, 10,20,40}andhence todecodethe message. Thecodedmessageis121197205:

121×71mod(110)=11 =100100
197×71mod(110) =17=111100
205×71mod(110)=35 =101110

Thedecodedmessageis:
100100111100101110.
JustasIthought!

Simple and short knapsack codes are far too easy to break to be of any real use. For aknapsack code to be reasonably secure it would need well over 200 terms each of length200bits.

AUTHENTICATION

APPLICATIONSKERBEROS

Kerberos is an authentication service developed as part of Project Athena at MIT. Itaddressesthethreatsposedinanopendistributedenvironmentinwhichusersatworkstations wish to access services on servers distributed throughout the network.Someofthese threatsare:

- Ausermaygainaccesstoaparticularworkstationandpretendtobeanotheruseroperating fromthat workstation.

- Ausermayalterthenetworkaddressofaworkstationsothattherequestssentfromthealte redworkstationappeartocomefromtheimpersonatedworkstation.

- Ausermayeavesdroponexchangesanduseareplayattacktogainentrancetoaserverorto disruptoperations.

TwoversionsofKerberosareincurrentuse:Version-4andVersion-
5.Thefirstpublishedreporton Kerberos listedthefollowingrequirements:

Secure: A network eavesdropper should not be able to obtain the necessary information to impersonate a user. More generally, Kerberos should be strong enough that a potential opponent does not find it to be the weak link.

Reliable: For all services that rely on Kerberos for access control, lack of availability of the Kerberos service means lack of availability of the supported services. Hence, Kerberos should be highly reliable and should employ a distributed server architecture, with one system able to backup another.

Transparent: Ideally, the user should not be aware that authentication is taking place, beyond the requirement to enter a password.

Scalable: The system should be capable of supporting large numbers of clients and servers. This suggests a modular, distributed architecture

Two versions of Kerberos are in common use: Version 4 is most widely used version. Version 5 corrects some of the security deficiencies of Version 4. Version 5 has been issued as a draft Internet Standard (RFC1510)

KERBEROS VERSION 4

1.) **SIMPLE DIALOGUE:**

MORE SECURE DIALOGUE

Once per service session

$$TicketV = EK_v[IDc, ADc, IDv, Ts2, Lifetime2]$$
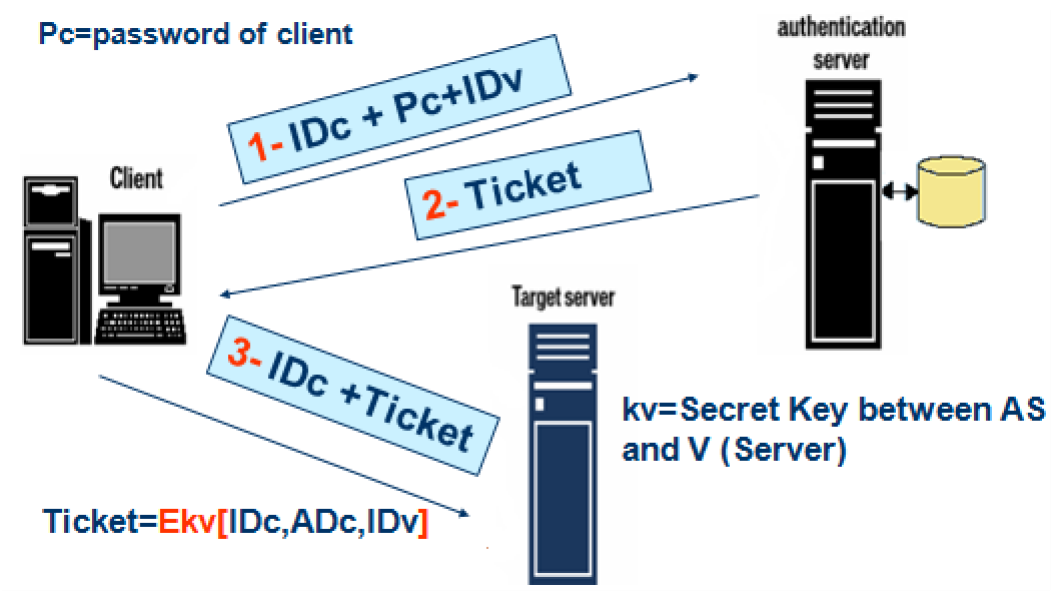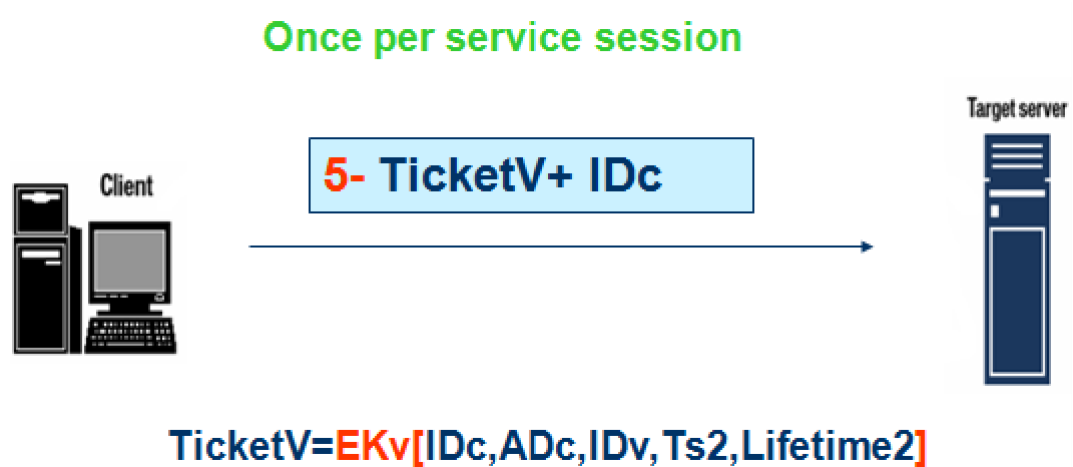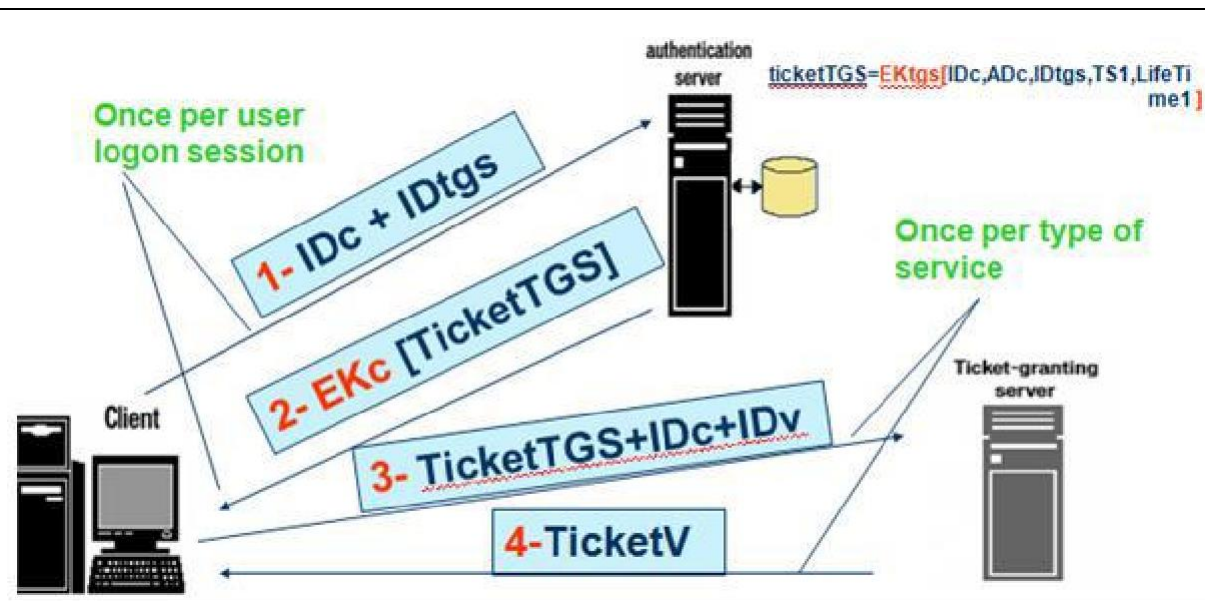
The Version 4 Authentication Dialogue The full Kerberos v4 authentication dialogue isshownhere divided into3phases.

(1) C → AS $ID_c \parallel ID_{tgs} \parallel TS_1$

(2) AS → C $E(K_c, [K_{c,tgs} \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2 \parallel Ticket_{tgs}])$

$Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \parallel ID_C \parallel AD_C \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2])$

(a) Authentication Service Exchange to obtain ticket-granting ticket

(3) C → TGS $ID_v \parallel Ticket_{tgs} \parallel Authenticator_c$

(4) TGS → C $E(K_{c,tgs}, [K_{c,v} \parallel ID_v \parallel TS_4 \parallel Ticket_v])$

$Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \parallel ID_C \parallel AD_C \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2])$

$Ticket_v = E(K_v, [K_{c,v} \parallel ID_C \parallel AD_C \parallel ID_v \parallel TS_4 \parallel Lifetime_4])$

$Authenticator_c = E(K_{c,tgs}, [ID_C \parallel AD_C \parallel TS_3])$

(b) Ticket-Granting Service Exchange to obtain service-granting ticket

(5) C → V $Ticket_v \parallel Authenticator_c$

(6) V → C $E(K_{c,v}, [TS_5 + 1])$ (for mutual authentication)

$Ticket_v = E(K_v, [K_{c,v} \parallel ID_C \parallel AD_C \parallel ID_v \parallel TS_4 \parallel Lifetime_4])$

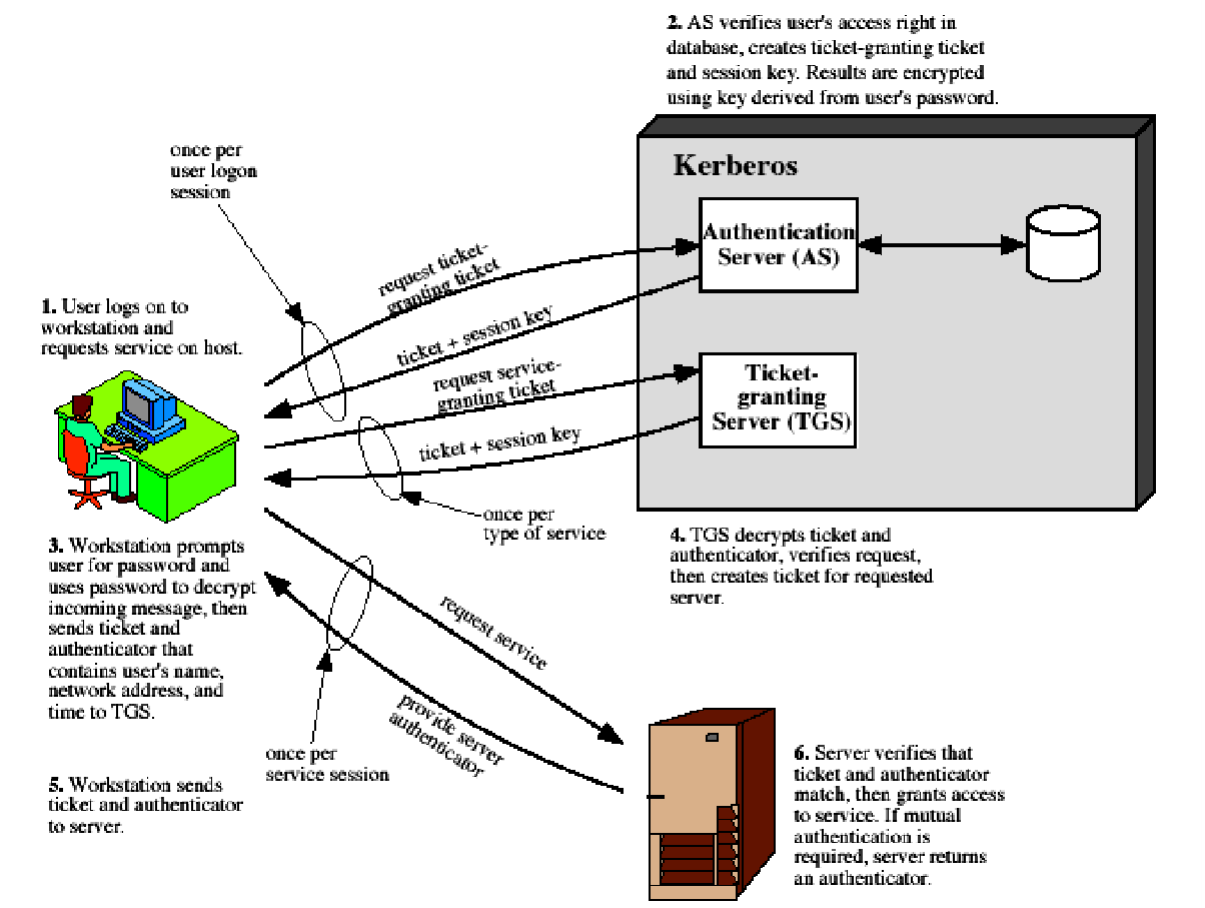$Authenticator_c = E(K_{c,v}, [ID_C \parallel AD_C \parallel TS_5])$

(c) Client/Server Authentication Exchange to obtain service

There is a problem of captured ticket-granting tickets and the need to determinethat the ticket presenter is the same as the client for whom the ticket was issued. Anefficientwayofdoingthisis tousea sessionencryptionkey tosecureinformation.

Message (1) includes a timestamp, so that the AS knows that the message is timely.Message(2)includesseveralelementsoftheticketinaformaccessibletoC.ThisenablesC to confirm that this ticket is for the TGS and to learn its expiration time. Note that theticket does not prove anyone's identity but is a way to distribute keys securely. It is theauthenticatorthatprovestheclient'sidentity.Becausetheauthenticatorcanbeusedonlyonce and has a short lifetime, the threat of an opponent stealing both the ticket and theauthenticator for presentation later is countered. C then sends the TGS a message thatincludes the ticket plus the ID of the requested service (message 3). The reply from theTGS, in message (4), follows the form of message (2). C now has a reusable service-granting ticket for V. When C presents this ticket, as shown in message (5), it also sendsanauthenticator.

Theservercandecrypttheticket,recoverthesessionkey,anddecrypttheauthenticator.Ifmutual authenticationisrequired,theservercanreplyasshowninmessage(6).

OverviewofKerberos

1. User logs on to workstation and requests service on host.

once per user logon session

request ticket-granting ticket

ticket + session key

request service-granting ticket

ticket + session key

once per type of service

2. AS verifies user's access right in database, creates ticket-granting ticket and session key. Results are encrypted using key derived from user's password.

Kerberos

Authentication Server (AS)

Ticket-granting Server (TGS)

3. Workstation prompts user for password and uses password to decrypt incoming message, then sends ticket and authenticator that contains user's name, network address, and time to TGS.

5. Workstation sends ticket and authenticator to server.

once per service session

request service

provide server authenticator

4. TGS decrypts ticket and authenticator, verifies request, then creates ticket for requested server.

6. Server verifies that ticket and authenticator match, then grants access to service. If mutual authentication is required, server returns an authenticator.

Kerberos Realms A full-service Kerberos environment consisting of a Kerberos server,a number of clients, and a number of application servers is referred to as a Kerberosrealm.AKerberosrealmisasetofmanagednodesthatsharethesameKerberosdatabase , and are part of the same administrative domain. If have multiple realms, theirKerberosservers mustshare keysandtrust eachother.

Thefollowingfigureshowstheauthenticationmessageswhereserviceisbeingrequestedfrom another domain. The ticket presented to the remote server indicates the realm inwhich the user was originally authenticated. The server chooses whether to honor theremote request. One problem presented by the foregoing approach is that it does notscalewelltomanyrealms,as eachpairofrealms needtoshare akey.

Request for Service in another realm:
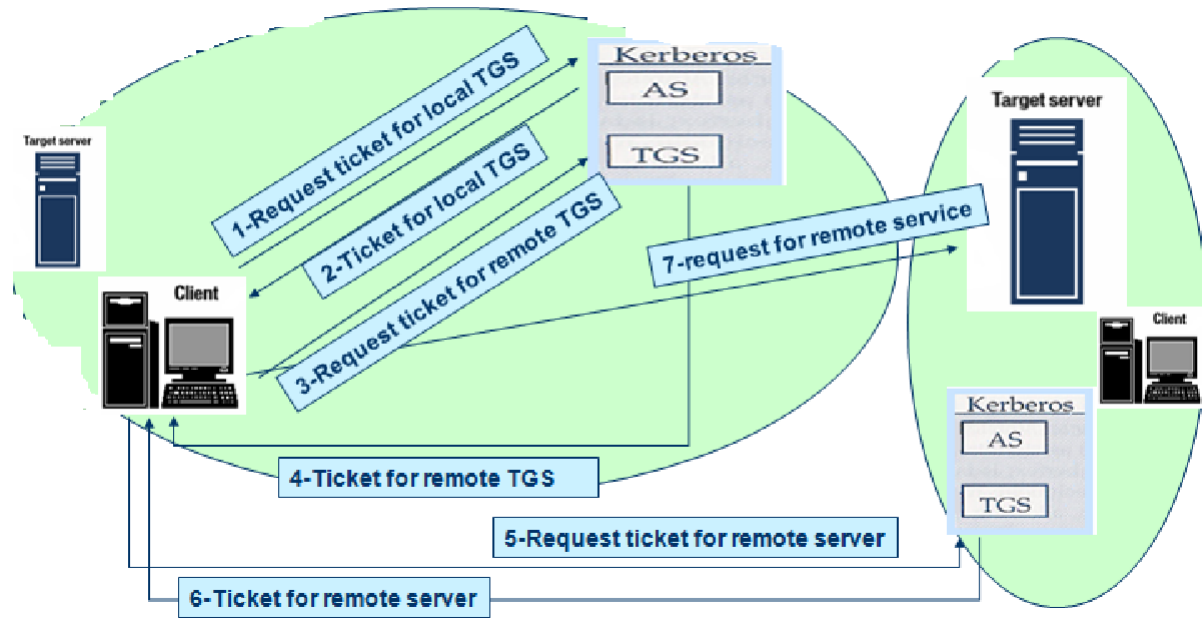
The limitations of Kerberos version-4 are categorised into two

types:EnvironmentalshortcomingsofVersion4:

– Encryptionsystemdependence:DES

– Internetprotocoldependence

– Ticketlifetime

– Authenticationforwarding

– Inter-

realmauthenticationTechnicaldeficien

  ciesofVersion4:

– Doubleencryption

– SessionKeys

– Passwordattack

KERBEROSVERSION5

Kerberos Version 5 is specified in RFC 1510 and provides a number of improvementsover version 4 in the areas of environmental shortcomings and technical deficiencies. Itincludessome newelements such as:

Realm: Indicates realm of the

userOptions

Times

– From:thedesiredstarttimefortheticket

– Till:therequestedexpirationtime

– Rtime:requestedrenew-tilltime

⬚⬚Nonce:Arandomvaluetoassurethe responseisfresh

ThebasicKerberosversion5authenticationdialogueisshownhereFirst,considerthe **authenticationserviceexchange.**

(1) C → AS Options ‖ ID_c ‖ $Realm_c$ ‖ ID_{tgs} ‖ Times ‖ $Nonce_1$

(2) AS → C $Realm_c$ ‖ ID_C ‖ $Ticket_{tgs}$ ‖ $E(K_c, [K_{c,tgs}$ ‖ Times ‖ $Nonce_1$ ‖ $Realm_{tgs}$ ‖ $ID_{tgs}])$

$Ticket_{tgs} = E(K_{tgs}, [Flags$ ‖ $K_{c,tgs}$ ‖ $Realm_c$ ‖ ID_C ‖ AD_C ‖ Times])

(a) Authentication Service Exchange to obtain ticket-granting ticket

(3) C → TGS Options ‖ ID_v ‖ Times ‖ ‖ $Nonce_2$ ‖ $Ticket_{tgs}$ ‖ $Authenticator_c$

(4) TGS → C $Realm_c$ ‖ ID_C ‖ $Ticket_v$ ‖ $E(K_{c,tgs}, [K_{c,v}$ ‖ Times ‖ $Nonce_2$ ‖ $Realm_v$ ‖ $ID_v])$

$Ticket_{tgs} = E(K_{tgs}, [Flags$ ‖ $K_{c,tgs}$ ‖ $Realm_c$ ‖ ID_C ‖ AD_C ‖ Times])

$Ticket_v = E(K_v, [Flags$ ‖ $K_{c,v}$ ‖ $Realm_c$ ‖ ID_C ‖ AD_C ‖ Times])

$Authenticator_c = E(K_{c,tgs}, [ID_C$ ‖ $Realm_c$ ‖ $TS_1])$

(b) Ticket-Granting Service Exchange to obtain service-granting ticket

(5) C → V Options ‖ $Ticket_v$ ‖ $Authenticator_c$

(6) V → C $E_{K_{c,v}}$ [TS_2 ‖ Subkey ‖ Seq#]

$Ticket_v = E(K_v, [Flags$ ‖ $K_{c,v}$ ‖ $Realm_c$ ‖ ID_C ‖ AD_C ‖ Times])

$Authenticator_c = E(K_{c,v}, [ID_C$ ‖ $Realm_c$ ‖ TS_2 ‖ Subkey ‖ Seq#])$

(c) Client/Server Authentication Exchange to obtain service

Message (1) is a client request for a ticket-granting ticket. Message (2) returns a ticket-granting ticket, identifying information for the client, and a block encrypted using theencryption key based on the user's password. This block includes the session key to beusedbetweentheclientandtheTGS.Nowcomparethe**ticket-**

grantingserviceexchangeforversions4and5.Seethatmessage(3)forbothversionsincludesan authenticator, a ticket, and the name of the requested service. In addition, version 5includesrequestedtimesandoptionsfortheticketandanonce,allwithfunctionssimilartothose ofmessage(1).Theauthenticatoritselfisessentiallythesameastheoneusedinversion 4. Message (4) has the same structure as message (2), returning a ticket plusinformation needed by the client, the latter encrypted with the session key now sharedby the client and the TGS. Finally, for the client/server authentication exchange, severalnewfeaturesappearinversion5,suchasarequestformutualauthentication.Ifrequired,th eserverrespondswithmessage(6)thatincludesthetimestampfromthe

authenticator.Theflagsfieldincludedinticketsinversion5supportsexpandedfunctionalitycompared to that available in version 4.

Advantages of Kerberos:

User's passwords are never sent across the network, encrypted or in plain

textSecretkeysare*only*passedacross thenetworkinencryptedform

Client and server systems mutually

It authenticatelimitsthedurationoftheirusers'authenticat

ion.

Authenticationsare**reusable**and**durable**

Kerberoshasbeenscrutinizedbymanyofthetopprogrammers,cryptologistsandsecurityexperts in theindustry

X.509 AUTHENTICATION SERVICE

ITU-TrecommendationX.509ispartoftheX.500seriesofrecommendationsthatdefineadirectoryservice.Thedirectoryis,ineffect,aserverordistributedsetofserversthatmaintainsadatabaseofinformationaboutusers.Theinformationincludesamappingfromusernametonetworkaddress,aswellasotherattributesandinformationabouttheusers.X.509isbasedontheuseofpublic-keycryptographyanddigitalsignatures. The heart of the X.509 scheme is the public-key certificate associated witheachuser.Theseusercertificatesareassumedtobecreatedbysometrustedcertificationauthority (CA) and placed in the directory by the CA or by the user. The directory serveritself is not responsible for the creation of public keys or for the certification function; itmerelyprovidesan easilyaccessiblelocation foruserstoobtaincertificates.

The general format of a certificate is shown above, which includes the followingelements:

version1,2, or3

serial number (unique within CA) identifying

certificatesignaturealgorithmidentifier

issuerX.500name(CA)
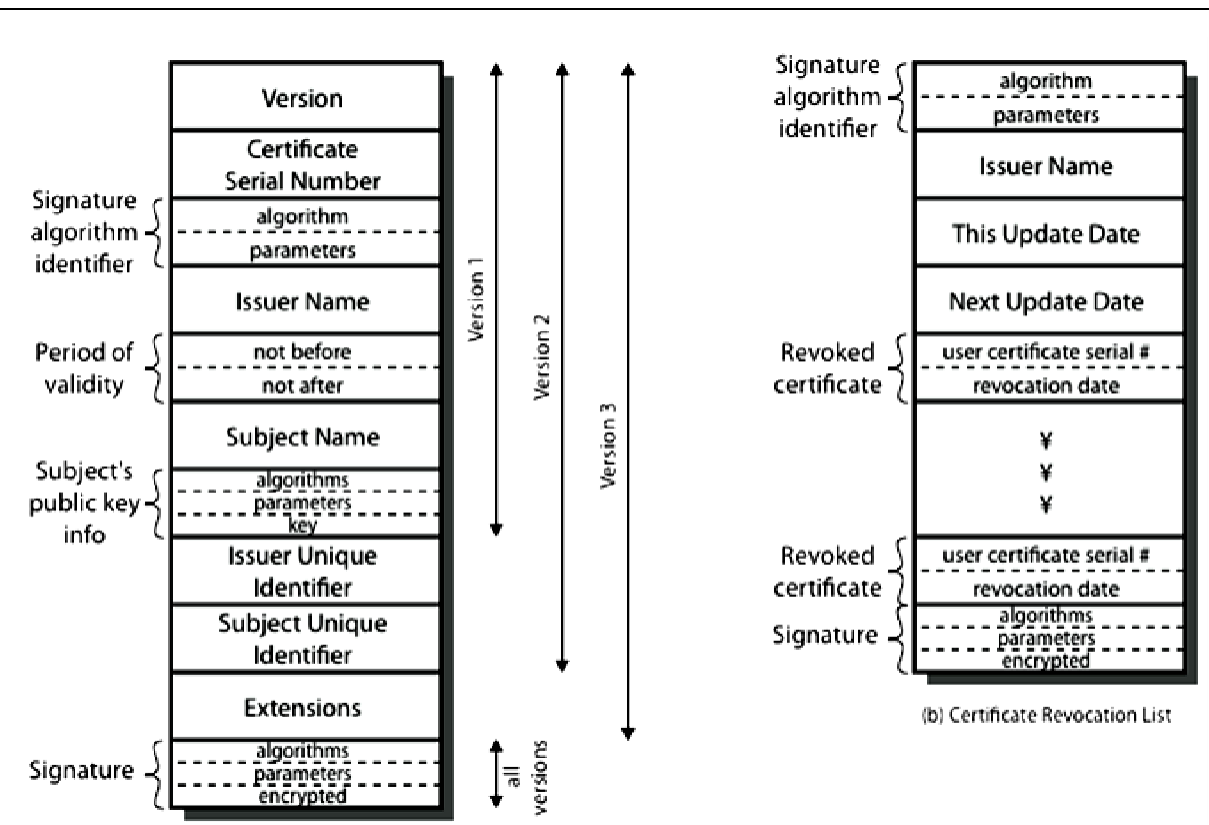
period of validity (from - to

dates)subjectX.500name(nameofowner

)

subject public-key info (algorithm, parameters,

key)issueruniqueidentifier(v2+)

(b) Certificate Revocation List

 subject unique identifier

(v2+)extensionfields (v3)

signature(ofhashof allfieldsincertificate)

Thestandardusesthefollowingnotationtodefineacertificate:

CA<<A>>=CA {V, SN,AI,CA,TA, A,Ap}

WhereY<<X>>=thecertificateofuserXissuedby certificationauthorityY

Y{I}==thesigningofI

byY.ItconsistsofIwithanencryptedhashcodeappendedUsercertificates

generatedbyaCAhavethe followingcharacteristics:

AnyuserwithCA'spublickeycanverifythe userpublickeythatwascertifiedNo party

other than the CA can modify the certificate without being

detectedbecausetheycannotbeforged,certificatescanbeplacedinapublicdirectory

Scenario: Obtaining a User Certificate If both users share a common CA then they

areassumedtoknowitspublickey.OtherwiseCA'smustformahierarchyandusecertificates

linking members of hierarchy to validate other CA's. Each CA has

certificatesforclients(forward)andparent(backward).Eachclienttrustsparentscertificates.It

enablesverificationofanycertificatefromoneCAbyusersofallotherCAsinhierarchy.Ahasobtain edacertificatefromtheCAX1.BhasobtainedacertificatefromtheCAX2.AcanreadtheB'scertificat ebutcannotverifyit.Inordertosolvetheproblem,theSolution:**X1<<X2>X2<<B>>.**Aobtainthec ertificateofX2signedbyX1from₂directory.

obtainX2'

₂publickey.Agoesbacktodirectoryandobtainthecertificateof BsignedbyX2.obtainB'spublickey securely.ThedirectoryentryforeachCAincludestwotypesofcertificates:Forwardcertificates:C ertificatesofXgeneratedbyotherCAsReverse certificates:CertificatesgeneratedbyXthatarethecertificatesofotherCAs

X.509CAHierarchy

AacquiresBcertificateusingchain:
X<<W>>W<<V>>V<<Y>>Y<<Z>>Z<<B>>
BacquiresAcertificateusingchain:
Z<<Y>>Y<<V>>V<<W>>W<<X>>X<<A>>
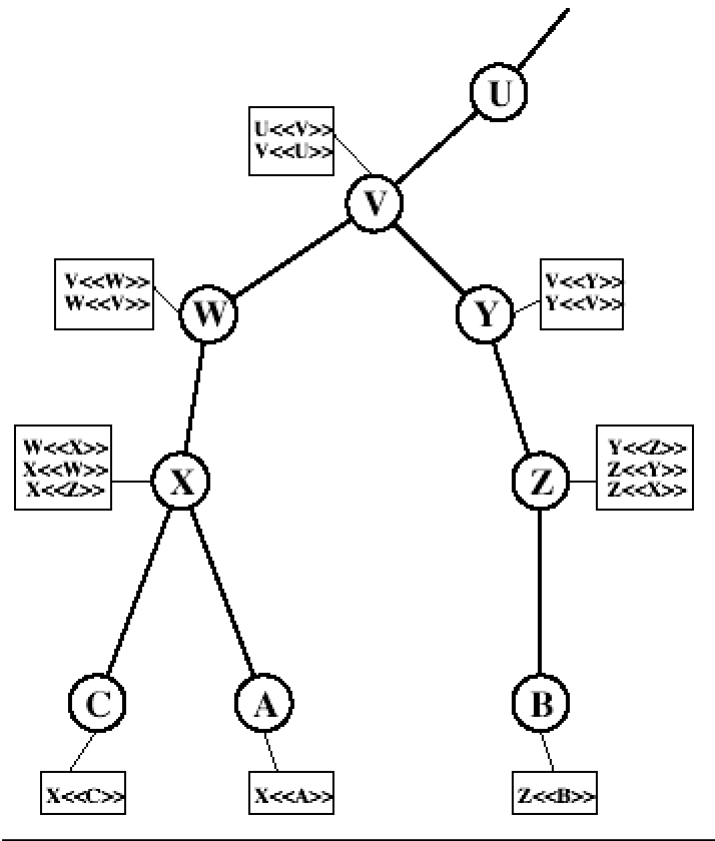
Revocation of Certificates Typically, a new certificate is issued just before theexpirationoftheoldone.Inaddition,itmaybedesirableonoccasiontorevokeacertificatebefor e itexpires,forone ofthefollowingreasons:

The user's private key is assumed to be compromised. The user is no longer certified by this CA.

The CA's certificate is assumed to be compromised.

Each CA must maintain a list consisting of all revoked but not expired certificates issued by that CA, including both those issued to users and to other CAs. These lists should also be posted on the directory. Each **certificate revocation list (CRL) posted** to the directory is signed by the issuer and includes the issuer's name, the date the list was created, the date the next CRL is scheduled to be issued, and an entry for each revoked certificate. Each entry consists of the serial number of a certificate and revocation date for that certificate. Because serial numbers are unique within a CA, the serial number is sufficient to identify the certificate.

AUTHENTICATION PROCEDURES

X.509 also includes three alternative authentication procedures that are intended for use across a variety of applications. All these procedures make use of public-key signatures. It is assumed that the two parties know each other's public key, either by obtaining each other's certificates from the directory or because the certificate is included in the initial message from each side. 1. One-Way Authentication: One way authentication involves a single transfer of information from one user (A) to another (B), and establishes the details shown above. Note that only the identity of the initiating entity is verified in this process, not that of the responding entity. At a minimum, the message includes a timestamp, a nonce, and the identity of B and is signed with A's private key. The message may also include information to be conveyed, such as a session key for B.

- 1 message (A->B) used to establish
  - the identity of A and that message is from A
  - message was intended for B
  - integrity & originality of message

A ——— 1-A {ta,ra,B,sgnData,KUb[Kab]} ———> B

Ta-timestamp rA=nonce B =identity
sgnData=signed with A's private key

Two-Way Authentication: Two-way authentication thus permits both parties in a communication to verify the identity of the other, thus additionally establishing the above details. The reply message includes the nonce from A, to validate the reply. It also includes a timestamp and nonce generated by B, and possible additional information for A.

- 2 messages (A->B, B->A) which also establishes in addition:
  - the identity of B and that reply is from B
  - that reply is intended for A
  - integrity & originality of reply

1-A {ta,ra,B,sgnData,KUb[Kab]}

2-B {tb,rb,A,sgnData,KUa[Kab]}

Three-WayAuthentication:Three-WayAuthenticationincludesafinalmessagefromAtoB,whichcontainsasignedcopyofthenonce, sothattimestampsneednotbechecked,forusewhen synchronized clocks arenotavailable.

- 3 messages (A->B, B->A, A->B) which enables above authentication without synchronized clocks

1- A {ta,ra,B,sgnData,KUb[Kab]}

2 -B {tb,rb,A,sgnData,KUa[Kab]}

3- A{rb}
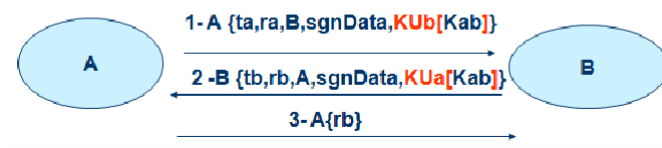
BIOMETRICAUTHENTICATION

Biometric authentication is a type of system that relies on the unique biologicalcharacteristicsofindividualstoverifyidentityforsecureaccesstoelectronicsystems.

Biometricverificationisconsideredasubsetof biometric authentication.The biometric technologies involved are based on the ways in which individuals can beuniquelyidentifiedthroughoneormoredistinguishingbiologicaltraits,suchasfingerprints, hand geometry, earlobe geometry, retina and iris patterns, voice waves,keystroke dynamics, DNA and signatures. Biometric authentication is the application ofthatproofofidentityaspartofaprocessvalidatingauserforaccess to asystem.Biometrictechnologiesareusedtosecureawiderangeofelectroniccommunications,in cludingenterprisesecurity,onlinecommerceandbanking--evenjustloggingin toa computerorsmartphone.

Biometric authentication systems compare the current biometric data capture to stored,confirmedauthenticdatainadatabase.Ifbothsamplesofthebiometricdatamatch,

authentication is confirmed and access is granted. The process is sometimes part ofamultifactor authentication system. For example, a smartphone user might log on withhis personal identification number (PIN) and then provide an iris scan to complete theauthenticationprocess.

Types of biometric authentication technologies: Retina scans produce an image of the blood vessel pattern in the light-sensitive surfaceliningthe individual'sinner eye.

Iris recognition is used to identify individuals based on unique patterns within the ring-shapedregion surroundingthepupiloftheeye.

Fingerscanning, the digital version of the ink-and-paper fingerprinting process, workswithdetailsinthepatternofraisedareasandbranchesinahumanfingerimage.

FingerveinIDisbasedontheuniquevascularpatterninanindividual'sfinger.

Facial recognition systems work with numeric codes called faceprints, which identify 80nodalpoints on a humanface.

Voice identification systems rely on characteristics created by the shape of the speaker'smouthandthroat,ratherthan more variableconditions.

Once seen mostly in spy movies (where it might be used to protect access to a top-secretmilitarylab,forexample),biometricauthenticationisbecomingrelativelycommonplace.
In addition to the security provided by hard-to-fake individual biologicaltraits, the acceptance of biometric verification has also been driven by convenience: Onecan'teasilyforget orlose ones biometrics.

Thehistoryofbiometricverification:
The oldest known use of biometric verification is fingerprinting. Thumbprints made onclaysealswereusedas ameansofuniqueidentificationasfarbackasancientChina.
Modern biometric verification has become almost instantaneous, and is increasinglyaccuratewiththeadventofcomputerizeddatabasesandthedigitizationof analogdata.

The market for biometrics products is still too fractured to name specific top providers.The physical characteristics of the biometrics products available today vary from themundane, such as fingerprinting, to the esoteric, like typing speeds andelectrophysiologicalsignals.

Until recently, biometrics was typically used at a physical security level – protectingfacilities at military bases or impenetrable bank vaults, for example. But, because single-factorauthentication methodsareeasytobreak,companieshavestartedlookingto two-factorsolutions,likebiometrics.

However, the following five fundamental barriers may limit the growth of biometricauthentication:

1. Biometrics can be complicated and costly to deploy. All biometric deploymentsrequireinstallationoftheirownhardwareandapplicationservers.

2. The market is still fractured. Should you buy a fingerprint reader, a voicerecognition system or an iris scanner? Since each product differs greatly in itsapproach and installation, it is difficult to compare them during a typical companybidprocess.

3. Biometric data is like any other data. It sits on servers, which are bait forhackers ifnot properly hardened and secured. Therefore, when reviewing any biometricproduct, make sure it transmits data securely, meaningencrypted, from thebiometric reader back to the authenticating server. And, make sure theauthenticatingserverhasbeenhardened, patchedandprotected.

4. Biometric readers are prone to errors. Fingerprints can smudge, faces and voicescan be changed and all of them can be misread, blocking a legitimate user, orpermittingaccesstoanunauthorizedormalicioususer.

5. Difficulties with user acceptance. Properly trained employees may be willing to usebiometrics devices, but customers, like those logging on to your Web site, may bemore reluctant to use – or worse, forced to purchase – a device that's difficult to useormakes doingbusiness, suchasbanking,on yoursite,ahassle insteadofa

convenience. And both your employees and customers may be squeamish aboutexposingtheireyestodeviceslikeirisscanners,eveniftheyappearharmless.

Despite these issues, biometrics is slowly gaining acceptance for two-factorauthentication purposes. The products are getting better, lighter and easier to use. Errorratesaregoingdown,andfingerprintreadersinstalledontokensandlaptopsaregetting smaller and less intrusive. And, like the rest of the security product industry,vendors will eventually merge and consolidate, uniting a fractured market, which willmakeiteasier tochoose aproductthat suits yourbusiness needs.