Hash And MAC

هي خوارزميات يتأكد المستلم ان الرسالة المرسلة إليه لم يتم التلاعب بهَ خلال فترة انتقالها إليه من المرسل أي التأكد من مصداقية وصول البيانات .اي يرسل رسالة وعند المستلم يتأكد ان الرسالة المستلمة هي نفسها الرسالة المرسلة ! إذن هي ليست خوارزميات تشفير إنما خوارزميات تأكد سلامة وصول بيانات لذالك ليس لها طرق فك تشفير تستخدم في كلمات مرور الحواسيب وفي ملفات التورنت للتأكد من وصول الملفات كاملة لأنه مثلا عند تنزيل أي ملف من الانترنت أي خلل في تنزيل ملف لا يعمل عند المستلم لذالك يجب ان يقوم الخادم باستخراج Hash من الملف وإرساله مع الملف والشخص الذي يحمل الملف عندما يكتمل تحميله يستخرج Hash للملف ويقارنه مع المرسل من قبل الخادم فإذا كان نفسه فالملف سليم وإلا الملف تالف .

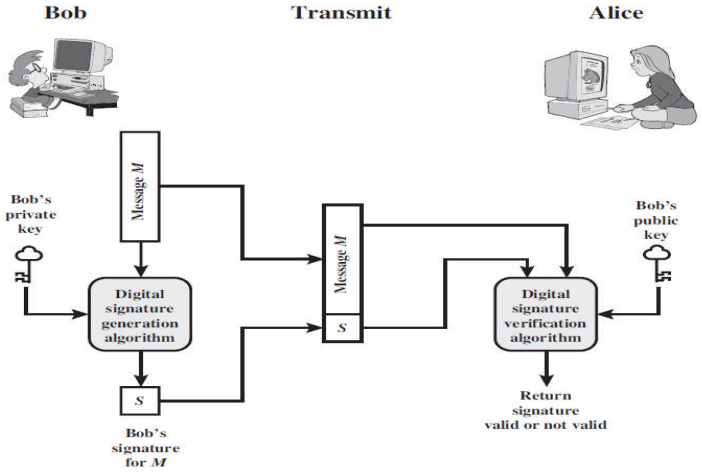


Figure 13.1 Generic Model of Digital Signature Process

تصل الحرب عند مرحلة انه لا يريد يقطع إرسال الرسالة أو انه حتى إذا قراها وفك شفرتها فلن يغير شيء من مسار العملية المطلوب تنفيذها فيحاول أن يغير مسار العملية المراد تنفيذها بتوصيل الرسالة للمستلم لكن بصيغة جديدة أو انه يعرف أن توصيل الرسالة مع إجراء بعض التغيرات عليها أفضل من قطعها فيعترض الرسالة ويغير عليها ويرسلها فلمستلم يستلم رسالة متغيرة وينفذها

فقد ينفذ العمليات التالية

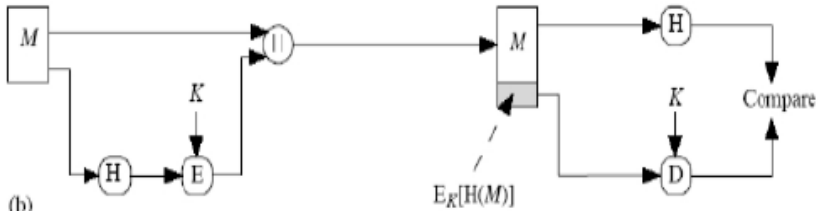
1. أما انه يعدل على الرسالة
2. او يؤثر على ترتيب البيانات أما يحذف او يضيف او يعيد ترتيب
3. او يؤخر او يعيد إرسال الرسالة

مثال : رسالة إلى محمد *(اذهب اقتل احمد)* معترض يعرف ان محمد سوف يقتل احمد وهو لا يستطيع ان يوقف مسار العملية أي لا يستطيع ان يمنعه من قتله لأنه مثلا في مكان بعيد عن القاتل لا يستطيع ان يصل إليه لذالك سوف يغير الرسالة ويجعلها *(اذهب اقتل صالح)* فلن يقتل احمد قتل صالح ونفذ ضربة المعترض ونجا الهدف ؟

- هذه العمليات التعديل على البيانات تحدث بسهولة في البث الجماعي Broadcast يعني سهولة شخص واحد يستلم رسالة ويعيد يبثها للجميع بصيغة ثانية فيغير العمل عليهم

تكون أساس عمل هذه الخوارزميات

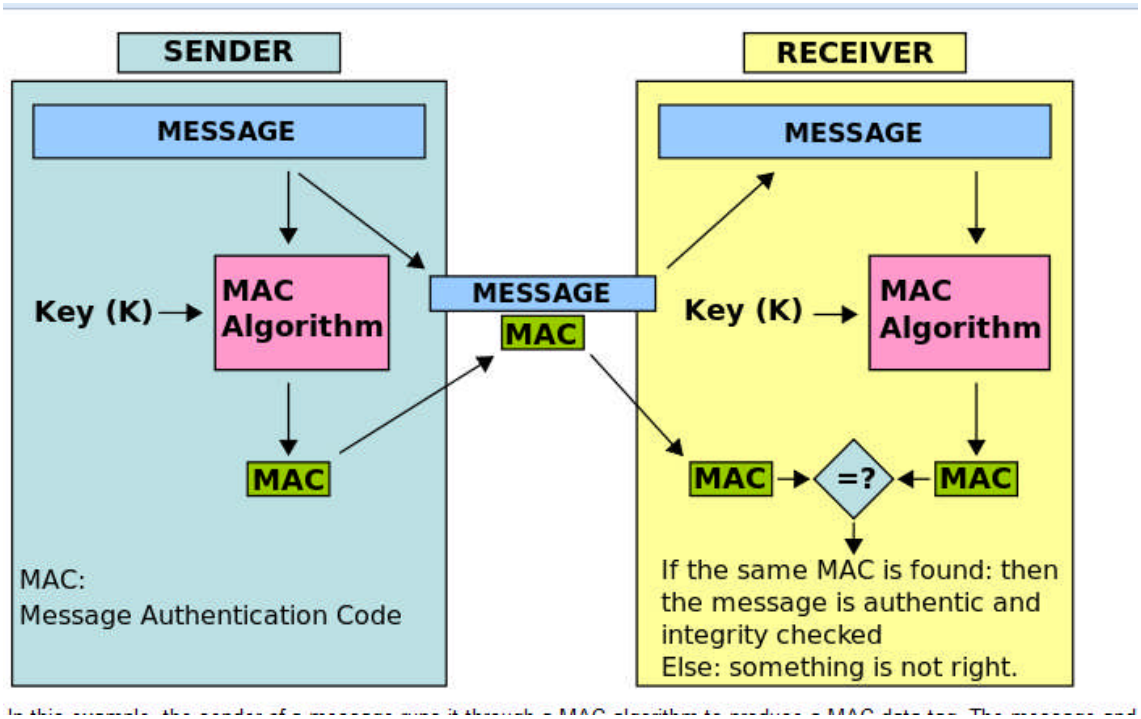
نستخدم كود معين (MAC) او (Hash) نستخرجه من الرسالة ونرسل الرسالة والكود الى المستلم فيقوم المستلم باستخراج نفس الكود من الرسالة المستلمة ويقارنه مع الكود المرفق مع الرسالة فإذا كان الكود متطابق يتأكد ان الرسالة لم يتم التلاعب به وخلافه يقوم بإهمال الرسالة ويعتبرها رسالة مزورة.



من ابسط طرق استخراج MAC لرسالة معينة نستخدم المعادلة التالية

$$Y=(A*M+B) \textbf{ mod } P$$

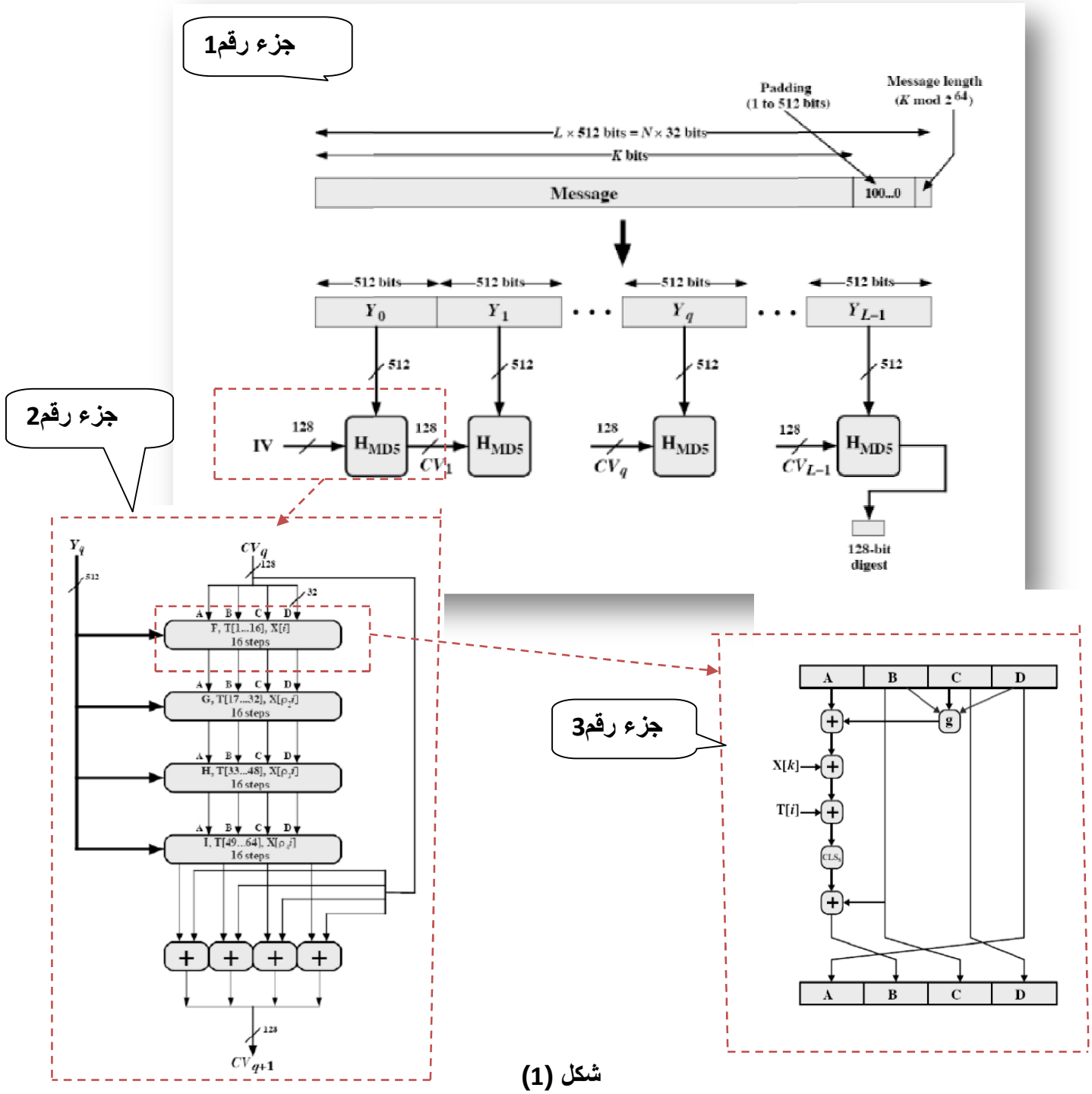
- **M :** هو حرف من أحرف الرسالة (ننفذ هذه المعادلة على كل حرف من أحرف الرسالة)
- **A,B :** هي أرقام عشوائية نحن نختارها بالنسبة للمرسل والمستلم نستخدم نفس الأرقام
- **P :** هو رقم أولي أيضا نحن نختاره



شكل إرسال رسالة مع MAC لها

خوارزمية MD5

هي إحدى خوارزميات Hash تكون من الرسالة المتغيرة التي نحن نريد إرسالها مفتاح ثابت طوله 128 bit (16 byte) كما في الشكل



شكل (1)

خطوات العمل مع خوارزمية MD5

1. في **شكل (1)** جزء رقم (1) تقسم البيانات على شكل Block كل واحد حجمه 512bit حيث L يمثل عدد Block كما في الشكل بالأسفل مقسم إلى ثلاثة مقاطع حيث

- الجزء الأول message يمثل بيانات الرسالة بعد تحويلها الى النظام الثنائي
- والجزء الثاني padding وضيفته إذا لم تكفي bits الخاص بالرسالة لي Block وتبقى بعض bits ضمن 512 bit داخل احد Block فارغة

تملئ bits فارغة بواحد (1) ويليه أصفار بقدر bits الفارغة ؟

في Block الأخير نستخدم فقط 448 bit ويبقى 64 bit الأخير لخزن طول الرسالة أي لخزن عدد bits الخاص بالرسالة داخل اخر 64 bit .

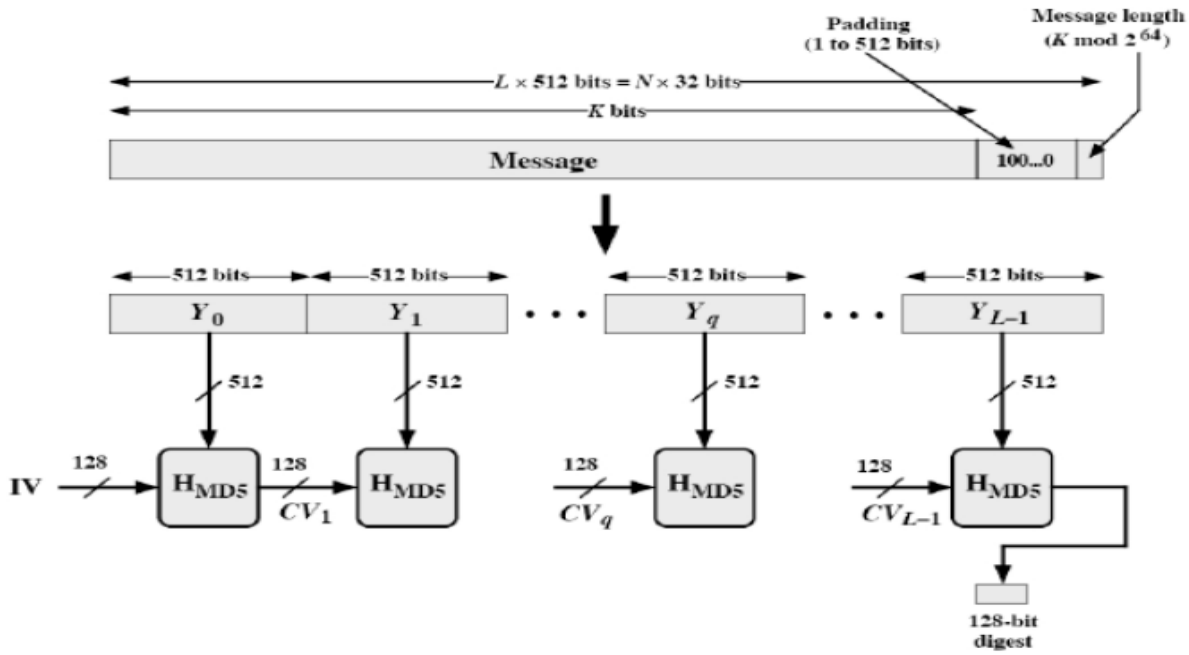
لحساب عدد bits الواجب إضافتها الى الرسالة نستخدم المعادلة التالية

A= عدد bits رسالة حقيقية / 512

B=الجزء ألكسري (A)*512

bits مضافة=448- B

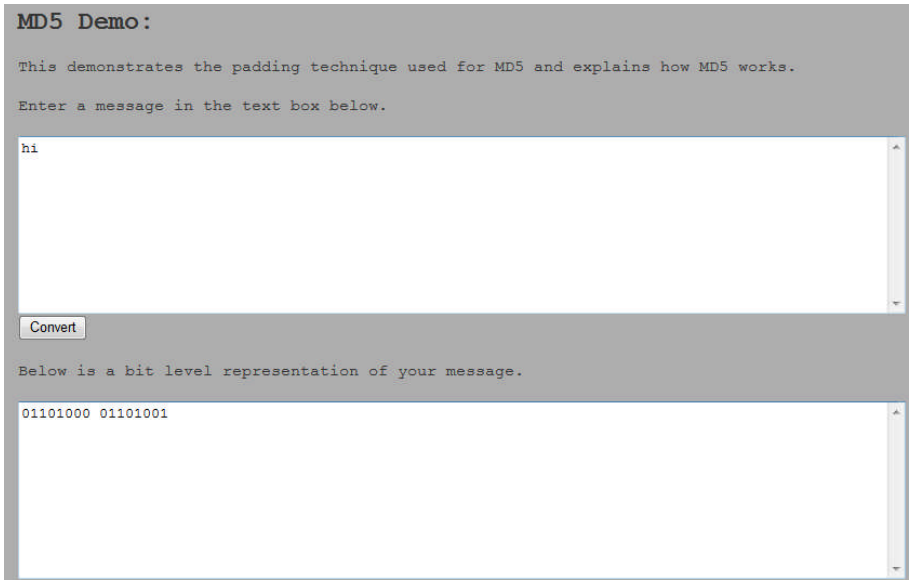
- الجزء الثالث message Length يمثل طول bits الرسالة الحقيقية ونحن هنا حجز لها 64 bit من اخر Block في الرسالة لنخزن فيه طول الرسالة .إذا كانت البيانات عبارة عن Block واحد تخزن طول البيانات في 64 bit الأخير أما إذا كانت أكثر من Block تخزن طول البيانات في أخر 64 bit في نهاية Block الأخير(ويعتبر هذا جزء من Block bit الخاص بالرسالة) .واستخدمنا هنا 64 bit لتمثيل طول الرسالة أي إننا نستطيع ان نشفر رسالة بخوارزمية MD5 طولها 2^64



شكل (2)

كما مبين في **الشكل (2)** الرسالة اذا كانت طولها أكثر من 512 bit تقسم الى Block كل واحد حجمه 512 bit بالأسماء التالية (y0,y1,yq,yl-1) وكل Block يدخل على HMD5 واحد .

مثال : إدخال رسالة محتواها (hi) لاحظ انه حولها الى النظام الثنائي بالصيغة التالية (01101000 01101001)

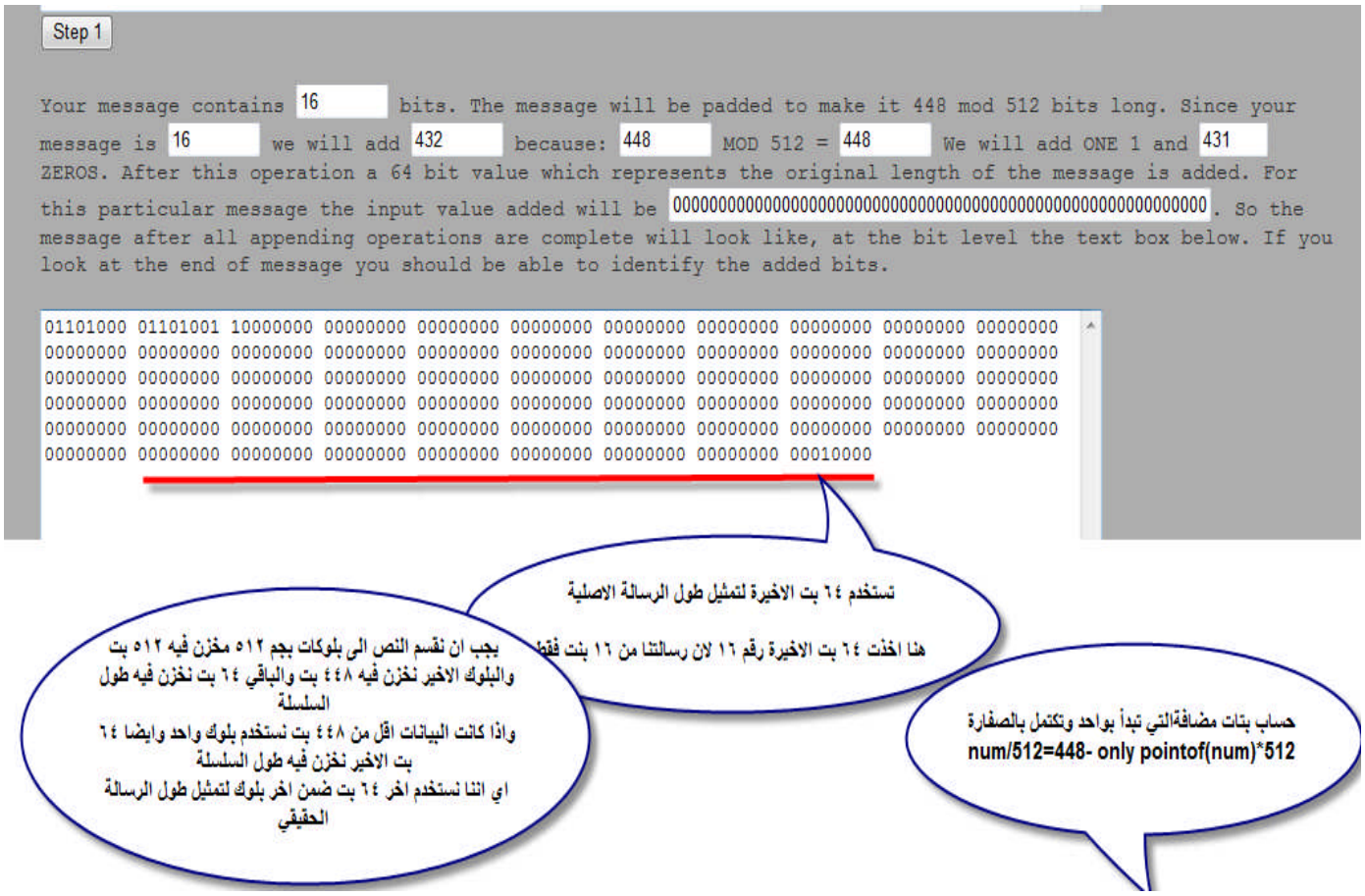



```
text =

hi


ascm =

   104   105


binasc =

1101000
1101001
```

نلاحظ ان البيانات حجمها 16bit لذالك لكي يجعلها Block يجب ان تصل عددها الى 448bit لذالك سيضيف 1bit قيمته واحد و431bit قيمته صفر الى نهاية السلسلة

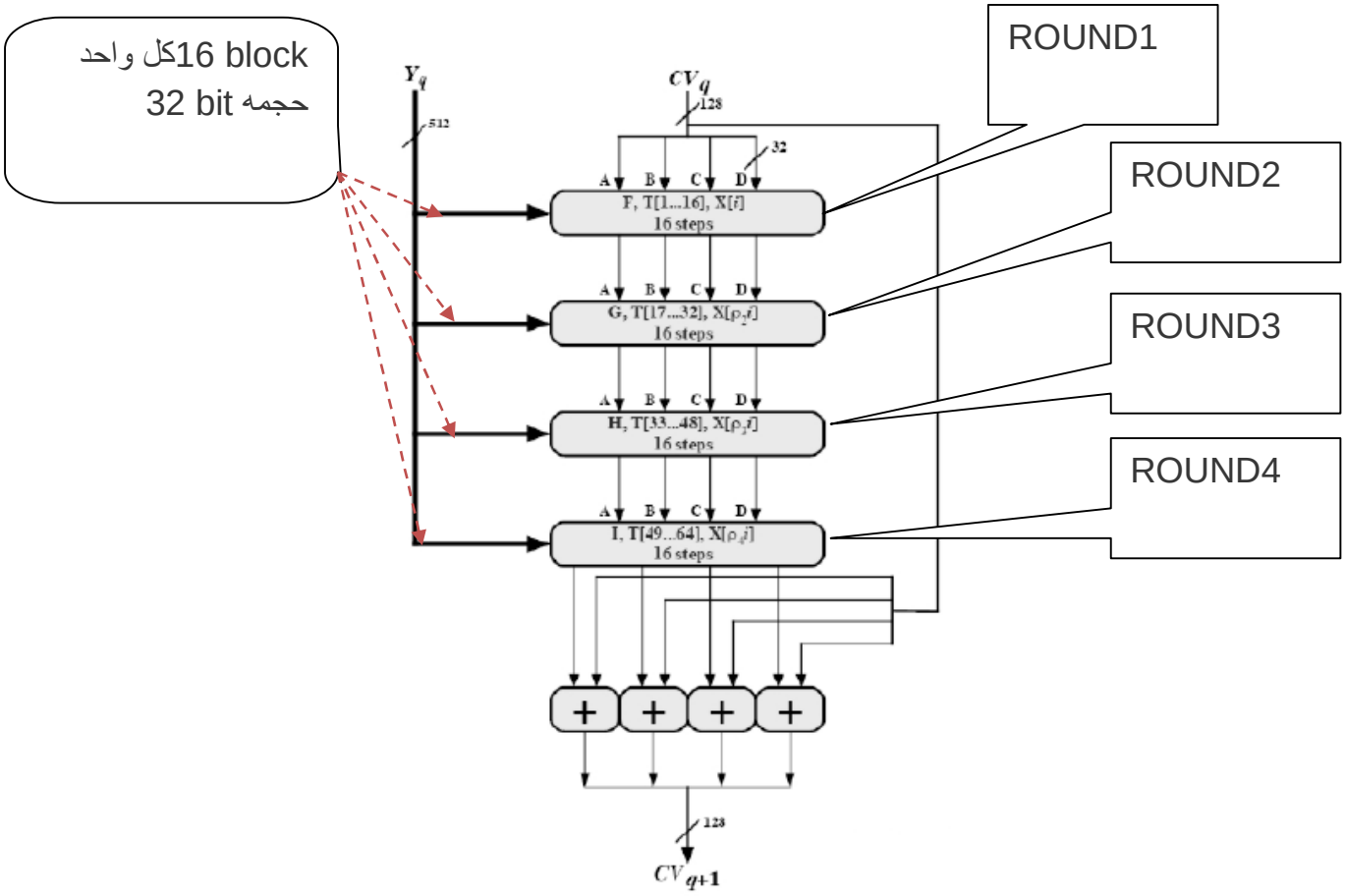
```
01101000 01101001 10000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00010000
```

و64bit البقية تمثل طول الرسالة الحقيقية بصيغة النظام الثنائي لاحظها تحتها خط (وهي طولها 16 bit)
(0000000000000000000000000000000000000000000000000000000000010000)
حتى يصبح طول البيانات الكلي 512bit.



2. في **شكل (1)** جزء رقم (2) او **شكل(3)** وهي مجموعة عمليات تجري ضمن كل HMD5 هذا الجزء مقسم إلى أربع ROUND كل واحدة منها تنفذ عملياتها 16 مرة سوف نشرح على جزء واحد و لمرة واحدة والبقية نفسها تكرر لكن على البيانات المحدثة الجديدة حيث

Yq: يمثل 512 bit الخاص ب Block بيانات رسالة معينة حيث يقسم الى 16 block كل واحد بحجم 32 bit كلها تمرر على جميع ROUND الأربعة وهي ثابتة لا تتغير خلال 16 مرة تنفيذ داخل كل ROUND ضمن HMD5 الواحد لأنها تمثل بيانات رسالة معينة لكنها تتغير في HMD5 التالي لأنها بيانات block أخر ضمن الرسالة والبيانات الناتجة لكل من **(A,B,C,D)** من هذا HMD5 تنتقل عبر CV الى CV1 التالي شاهد **شكل (1)**



شكل (3)

ناتج الجمع هنا هو mod 2^32

(A,B,C,D) هي أربعة REGISTER حجمها الكلي 128 bit كل واحد حجمه 32 bit مخزن فيها قيم أولية وتتحدث ضمن ROUND الواحد 16 مرة وبعدها الناتج ROUND ينتقل الى ROUND التالي وعندما ينتهي من الأربعة ينتقل الى HMD5 التالي. هذه القيم الأولية بالنظام السادس عشري

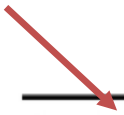
A = 67452301
B = EFCDAB89
C = 98BADCFE
D = 10325476

T: هي قيم بالرادين تستخدم لتعقيد التشفير تحضر وفق المعادلة التالية

i=1:64; x= abs(sin(i))* 2^32

أي نكون مصفوفة T تبدأ من 1 الى64 وفق المعادلة أعلاه . حيث تحول هذه البيانات الى نظام Hex حيث كل ROUND يأخذ جزء منها
توضيح :-

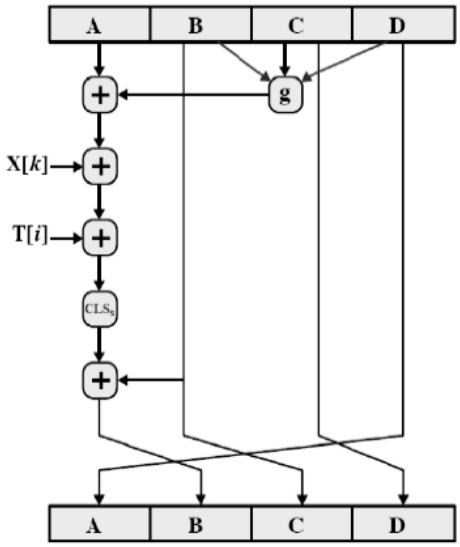
- ROUND1 الأول يأخذ من T[1—16]
- ROUND2 الثاني يأخذ من T[17—32]
- ROUND3 الثالث يأخذ من T[33—48]
- ROUND4 الرابع يأخذ من T[49—64]



(b) Table T, constructed from the sine function

T[1] = D76AA478	T[17] = F61E2562	T[33] = FFFA3942	T[49] = F4292244
T[2] = E8C7B756	T[18] = C040B340	T[34] = 8771F681	T[50] = 432AFF97
T[3] = 242070DB	T[19] = 265E5A51	T[35] = 699D6122	T[51] = AB9423A7
T[4] = C1BDCEEE	T[20] = E9B6C7AA	T[36] = FDE5380C	T[52] = FC93A039
T[5] = F57COFAF	T[21] = D62F105D	T[37] = A4BEEA44	T[53] = 655B59C3
T[6] = 4787C62A	T[22] = 02441453	T[38] = 4BDECFA9	T[54] = 8F0CCC92
T[7] = A8304613	T[23] = D8A1E681	T[39] = F6BB4B60	T[55] = FFEFF47D
T[8] = FD469501	T[24] = E7D3FBC8	T[40] = BEBFBC70	T[56] = 85845DD1
T[9] = 698098D8	T[25] = 21E1CDE6	T[41] = 289B7EC6	T[57] = 6FA87E4F
T[10] = 8B44F7AF	T[26] = C33707D6	T[42] = EAA127FA	T[58] = FE2CE6E0
T[11] = FFFF5BB1	T[27] = F4D50D87	T[43] = D4EF3085	T[59] = A3014314
T[12] = 895CD7BE	T[28] = 455A14ED	T[44] = 04881D05	T[60] = 4E0811A1
T[13] = 6B901122	T[29] = A9E3E905	T[45] = D9D4D039	T[61] = F7537E82
T[14] = FD987193	T[30] = FCEFA3F8	T[46] = E6DB99E5	T[62] = BD3AF235
T[15] = A679438E	T[31] = 676F02D9	T[47] = 1FA27CF8	T[63] = 2AD7D2BB
T[16] = 49B40821	T[32] = 8D2A4C8A	T[48] = C4AC5665	T[64] = EB86D391

3. **شكل (4)** هي العمليات التي ستجري 16 مرة ضمن كل ROUND لاحظ **شكل (1)**



شكل (4)

CLS: هي circular left shift أي تدوير مع s bit
مثال : 00110 لو أزحناها بمقدار 2 من اليسار إلى اليمين ستصبح النتيجة 11000

g: هي معادلة الدوال (F,G,H,I) وهي عند كل ROUND لها معادلة مختلفة
مثال: عند ROUND1 الخاص F تكون (g=F(b,c,d)=(b^c)V(!b^d.
وهي دوال Boolean تحضر وفق المعادلات التالية

Round	Primitive Function g	g(b, c, d)
1	F(b, c, d)	(b ∧ c) v (!b ∧ d)
2	G(b, c, d)	(b ∧ d) v (c ∧ !d)
3	H(b, c, d)	b ⊕ c ⊕ d
4	I(b, c, d)	c ⊕ (b v !d)

توضيح معنى العمليات

AND =Λ
V = OR
NOT = !
Xor=(.:.)

X[k] : هي بيانات رسالة لكل block الذي كان يحتوي على 512 bit قسمة الى 16 block كل واحد بحجم 32 bit

مثال : لنكمل المثال السابق علىROUND1

بيانات الرسالة (hi) التي أصبحت 512 bit one Block تحول الى 16 block كل واحد بحجم 32 bit

x[1]= 01101000 01101001 10000000 00000000
x[2]= 00000000 00000000 00000000 00000000
x[3]= 00000000 00000000 00000000 00000000
x[4]=00000000 00000000 00000000 00000000
x[5]= 00000000 00000000 00000000 00000000
x[6]= 00000000 00000000 00000000 00000000
x[7]= 00000000 00000000 00000000 00000000
x[8]= 00000000 00000000 00000000 00000000
x[9]= 00000000 00000000 00000000 00000000
x[10]=00000000 00000000 00000000 00000000
x[11]=00000000 00000000 00000000 00000000
x[12]=00000000 00000000 00000000 00000000
x[13]=00000000 00000000 00000000 00000000
x[14]=00000000 00000000 00000000 00000000
x[15]=00000000 00000000 00000000 00000000
x[16]=00000000 00000000 00000000 00010000

عندما i=k= 1 اذن

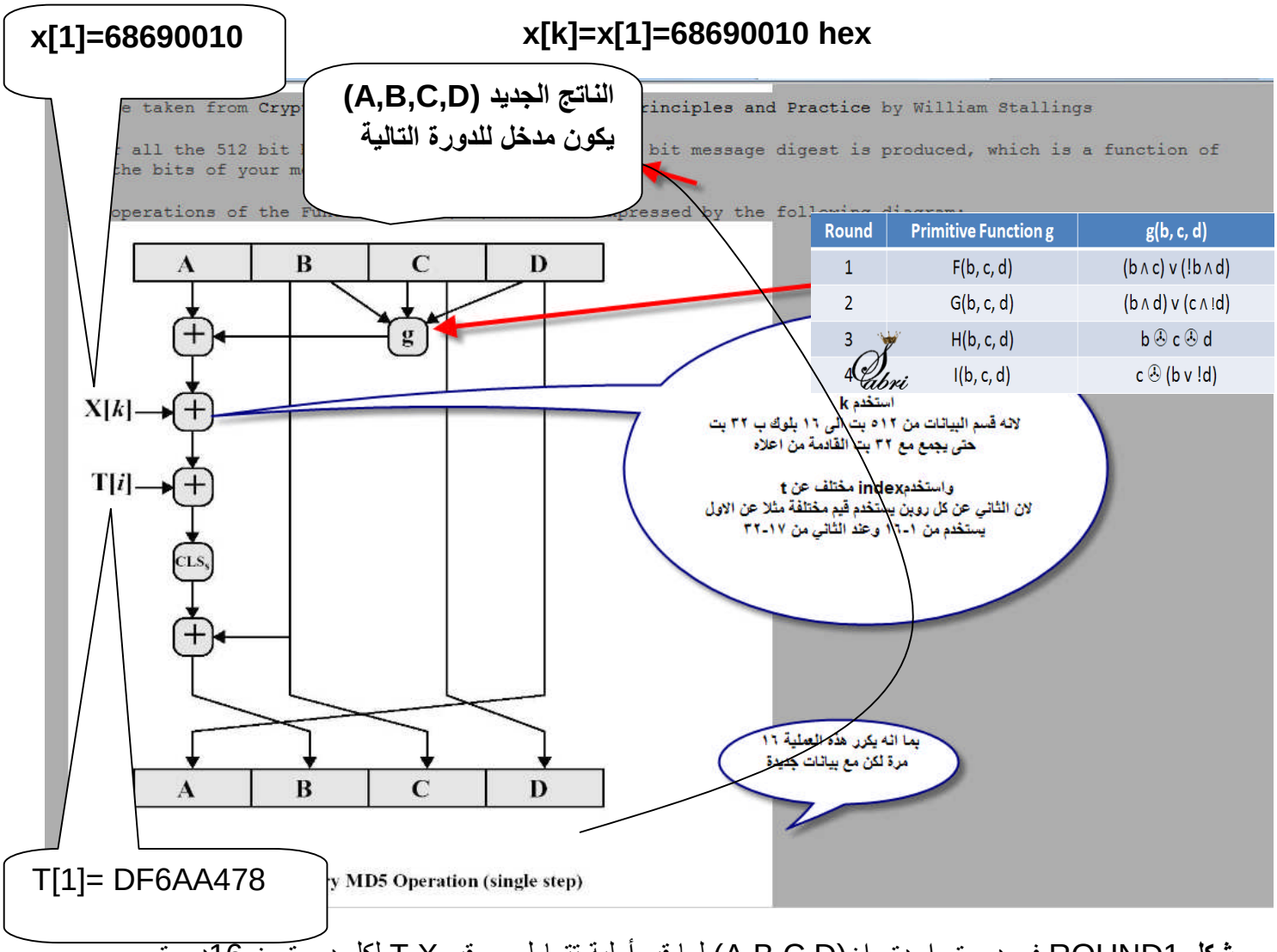
T[i]=T[1]= DF6AA478

استخرجنا هذه القيمة من جدول T

x[k]=x[1]= 01101000 01101001 00000000 00010000

x[1]=68690010

x[k]=x[1]=68690010 hex



شكل ROUND1 في دورة واحدة وان(A,B,C,D) لها قيم أولية تتعامل مع قيم T,X لكل دورة من 16دورة والبيانات الناتجة في(A,B,C,D) لكل دورة تستخدم كمدخلات للدورة التالية ضمن نفس ROUND وبعد ان تنتهي 16 دورة لكل ROUND الناتج لكل من (A,B,C,D) يعتبر مدخلات لل ROUND الثاني وإذا انتهى من ROUND4 الرابع الناتج يعتبر كمدخل لل HMD5 الجديد الخاص بمعالجة Block ثاني من البيانات والناتج النهائي بعد كل Blockيعتبر

هوا المفتاح الناتج

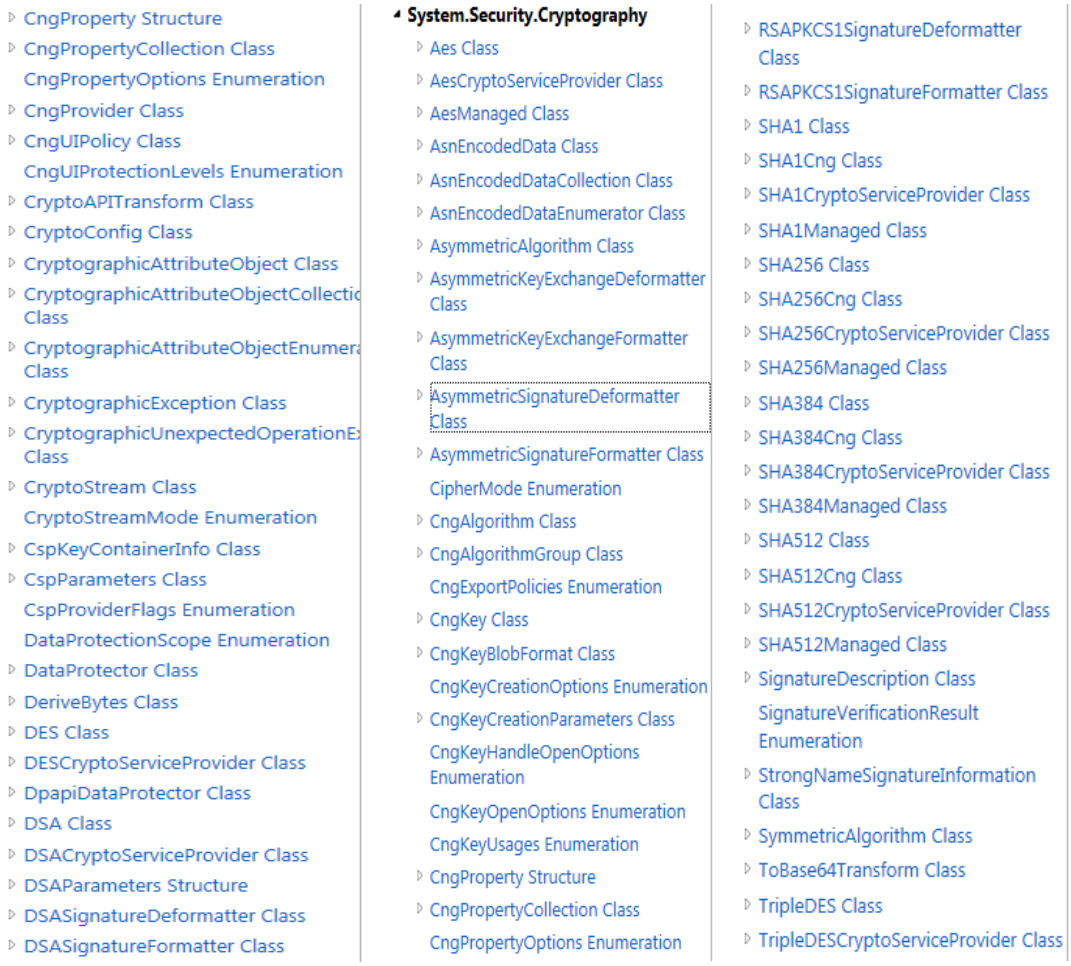
التشفير في بيئة VB.NET

في بيئة VB.NET جميع خوارزميات التشفير تقع ضمن هذه المكتبة . لذالك بكل سهولة نكون كائن جديد من نوع خوارزمية تشفير معينة ونجد التشفير لأي بيانات نريدها

VB.NET CODE

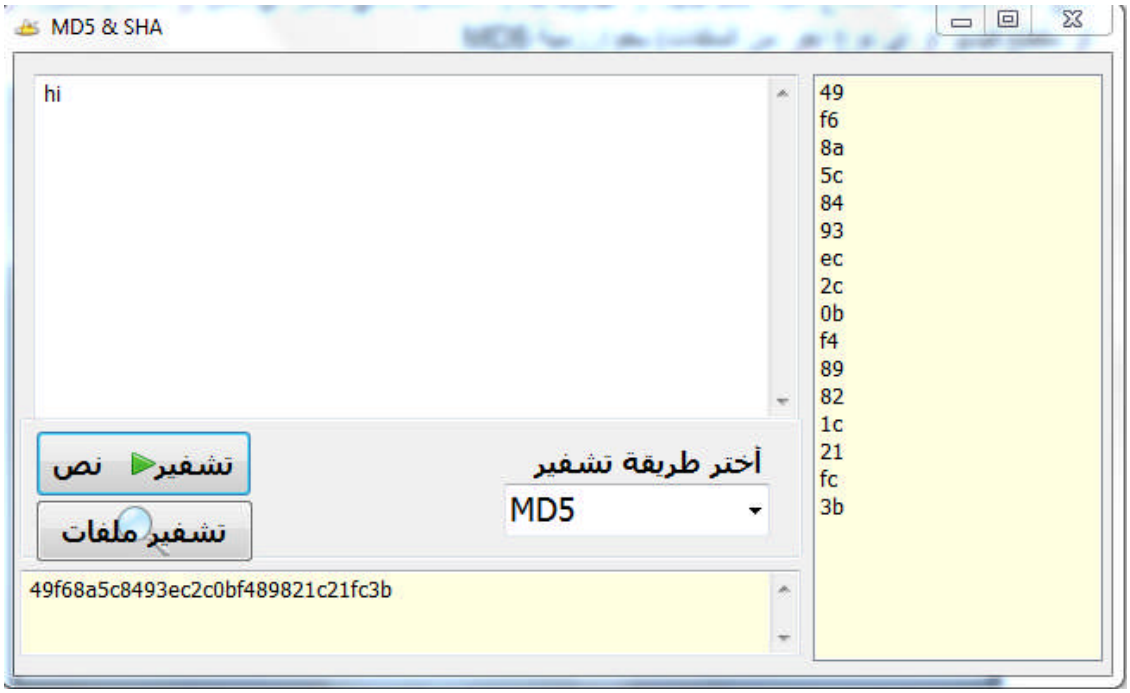
```
Imports System.Security.Cryptography
```

هذه جزء من خوارزميات الموجودة ضمن هذه المكتبة



خوارزمية (Message-Digest Algorithm) MD5 hash algorithm

تنتج هذه الخوارزمية مفتاح طوله 128 bit (16 byte) .المثال التالي يشفر أي نص او ملف (صورة او صوت او مقطع فيديو او أي نوع اخر من الملفات) بخوارزمية MD5



بداية نبني دالة تستقبل النص او الملف المراد تشفيره بصيغة Byte وتعيد لنا MD5 Hash الخاص بهذا النص بكل سهولة حيث نعرف داخل هذه الدالة متغير من نوع MD5 جديد ونكونه وبعده نعمل ComputeHash للبيانات المستلمة لحساب Hash لها وبعدها نحولها اللى بينات من Byte الى صيغة HEX

VB.NET CODE

```
Public Function GetMD5Data(ByVal data As Byte()) As String
'create new instance of md5
Dim md5a As MD5 = MD5.Create()

'convert the input text to array of bytes
Dim hashData As Byte() = md5a.ComputeHash(data)

'create new instance of StringBuilder to save hashed data
Dim returnValue As New StringBuilder()

'loop for each byte and add it to StringBuilder
For i As Integer = 0 To hashData.Length - 1
returnValue.Append(hashData(i).ToString("x2"))
Next
' return hexadecimal string
Return returnValue.ToString()

End Function
```

في زر (FIND MD5) نحول النص الذي نكتبه الى Byte ونرسله الى الدالة لكي تعيد لنا Hashله لنستعرض نتيجتها بالنص الثاني ونعرض البيانات ايضاً داخل ListBox

VB.NET CODE

```
TextBox2.Text = GetMD5Data(Encoding.[Default].GetBytes(TextBox1.Text))
        ListBox1.Items.Clear()
        For i = 0 To Len(TextBox2.Text) - 2 Step 2
            ListBox1.Items.Add(TextBox2.Text(i) & TextBox2.Text(i + 1))
        Next
```

وفي زر (تشفير ملفات) نتصف ملفات من الحاسبة وبعدها نحولها الى Byte ونرسلها الى الدالة لكي تعيد لها Hash

VB.NET CODE

```
   'read file and find it md5
Dim op As New OpenFileDialog
op.ShowDialog()
Dim fData As Byte() = File.ReadAllBytes(op.FileName)
TextBox2.Text = GetMD5Data(fData)

ListBox1.Items.Clear()
For i = 0 To Len(TextBox2.Text) - 2 Step 2
ListBox1.Items.Add(TextBox2.Text(i) & TextBox2.Text(i + 1))
Next
```

نستخدم الدالة التالية عند المستلم للتأكد ان النص المستلم هو نفسه المرسل وذالك بمقارنة المفتاح المرسل مع المفتاح الناتج من الرسالة المستلمة

VB.NET CODE

```
Function ValidateMD5Data(ByVal inputData As Byte(), ByVal
storedHashData As String) As Boolean
'hash input text and save it string variable
Dim getHashInputData As String = GetMD5Data(inputData)

If String.Equals(getHashInputData, storedHashData) Then
Return True
Else
Return False
End If
End Function
```

خوارزمية Secure Hash Algorithm(SHA)

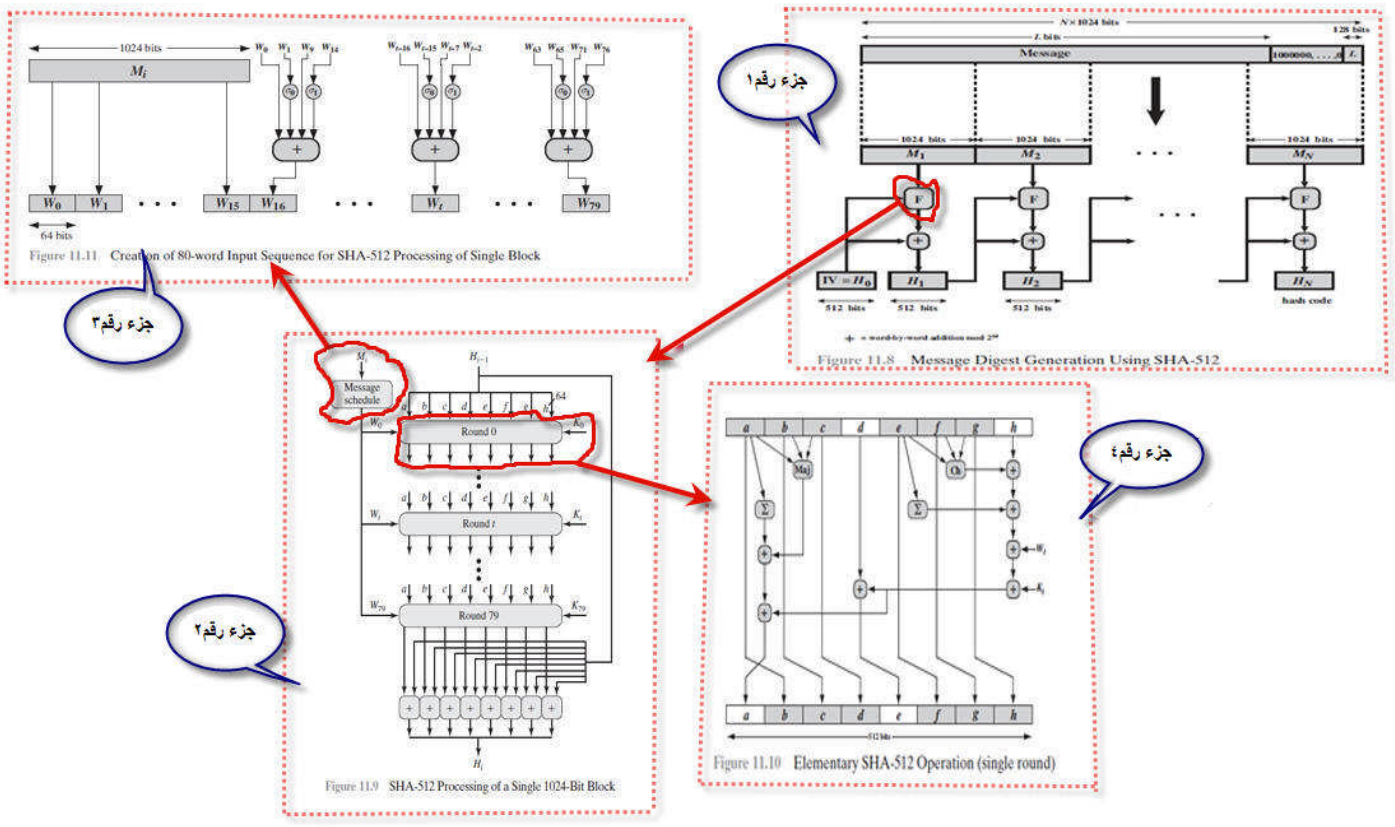
هي إحدى خوارزميات Hash تقوم بإيجاد سلسلة ثابتة من أي نص أو ملف توجد عدة أنواع منها
SHA1: تنتج هذه الخوارزمية مفتاح طوله 160 bit (20 byte)
SHA512: تنتج هذه الخوارزمية مفتاح طوله 512 bit (64 byte) من أي رسالة او ملف مهما كان طوله
وعدد من الأنواع SHA الأخرى كما نرها **بالشكل (4)** بالأسفل وخاصية كل نوع

شكل (4)

	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512
Message Digest Size	160	224	256	384	512
Message Size	$< 2^{64}$	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Block Size	512	512	512	1024	1024
Word Size	32	32	32	64	64
Number of Steps	80	64	64	80	80

Note: All sizes are measured in bits.

سوف نشرح بالتفصيل طريقة عمل SHA512 المبينة بالشكل (5)



شكل (5)

في **الشكل(5)** يبين لك طريقة عمل SHA512 وقمت بتقسيمها الى أربعة اجزاء لكل يسهل فهمها وسوف اشرح كل جزء بالتفصيل

جزء رقم (1)

نعرف ان SHA512 تستقبل أي طول من البيانات وتجد له HASH طوله 512 bit في **شكل (6)**
يقسم الرسالة الى Block كل واحد حجمه 1024 bit واخر 128 bit في اخير Block يكون محجوز لطول البيانات الحقيقية أي تستطيع هذه الخوارزمية ان تجد HASH لبيانات أقصى حد لطولها 2^128 والبتات Bit التي تبقى فارغة بين اخر 128 bit والبيانات الحقيقية للرسالة بعد تحويلها الى binary نعمل لها padding أي ندخل رقم واحد ويتبعه عدد من الاصفار حتى نملئ bits الفارغة. وان اخر Blockيتقبل فقط 896 bit لان كما قلنا اخر 128 bit في اخر Block تكون محجوزة لطول الرسالة الحقيقية بصيغة نظام ثنائي

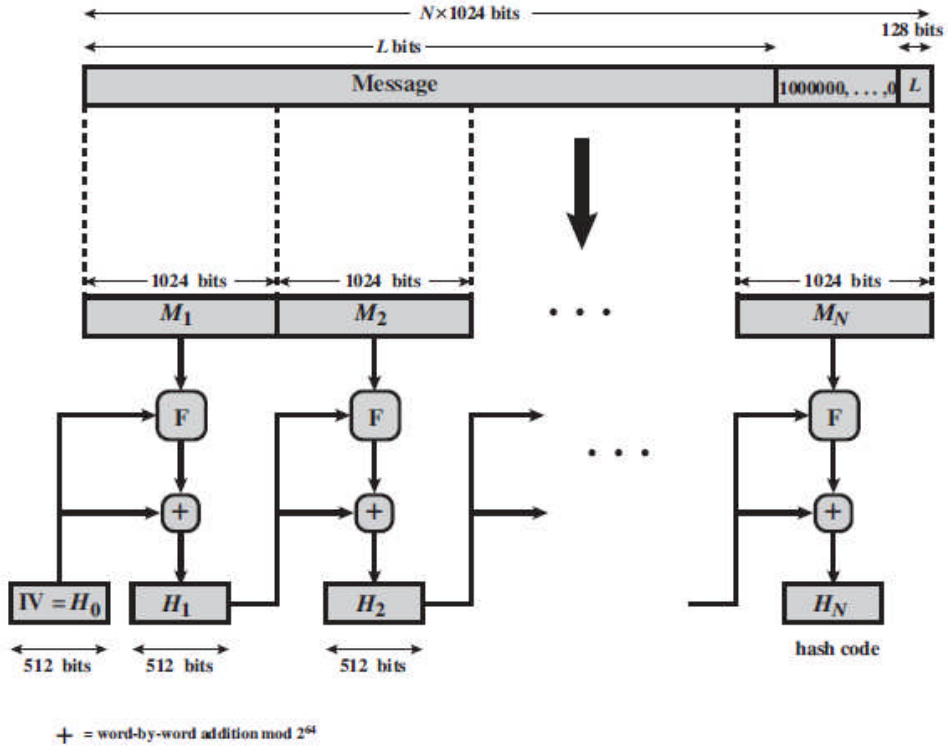


Figure 11.8 Message Digest Generation Using SHA-512

شكل (6)

حيث iv=H0 يمثل Registers الثمانية (A,B,C,D,E,F,G,H) كل واحد حجمه 64 bit مجموعهم هو 512 bit الذي سيمثل في النهاية Hashالخاص بالرسالة .وهذه القيم الأولية المخزنة داخل Registers

```
a = 6A09E667F3BCC908   e = 510E527FADE682D1
b = BB67AE8584CAA73B   f = 9B05688C2B3E6C1F
c = 3C6EF372FE94F82B   g = 1F83D9ABFB41BD6B
d = A54FF53A5F1D36F1   h = 5BE0CD19137E2179
```

الجزء رقم (2)

الذي يمثل **الجزء رقم2** في **شكل (5)** هو حرف (F) في **الجزء رقم1** .حيث يتكرر نفس عملياته مع كل Block لذالك سوف نشرح على Block واحد والبقية نفس الشيء. يمثل **شكل (7)** العمليات التي ستجري على كل Block من الرسالة الحقيقية لينتج مفتاح طولها 512 bit مخزن في (A,B,C,D,E,F,G,H) ويعتبر كمدخل للعمليات على Block التالي اذا كانت البيانات اكثر من Block وكما مبين في **الشكل(7)** او تعتبر النتيجة النهائية اذا كانت البيانات عبارة عن Block واحد . وان كل (F) مقسم الى 80 Round كل واحد تنفذ العمليات في داخله مره واحد (**جزء رقم4** في **شكل(5)** يمثل العمليات التي ستجري ضمن كل Round)

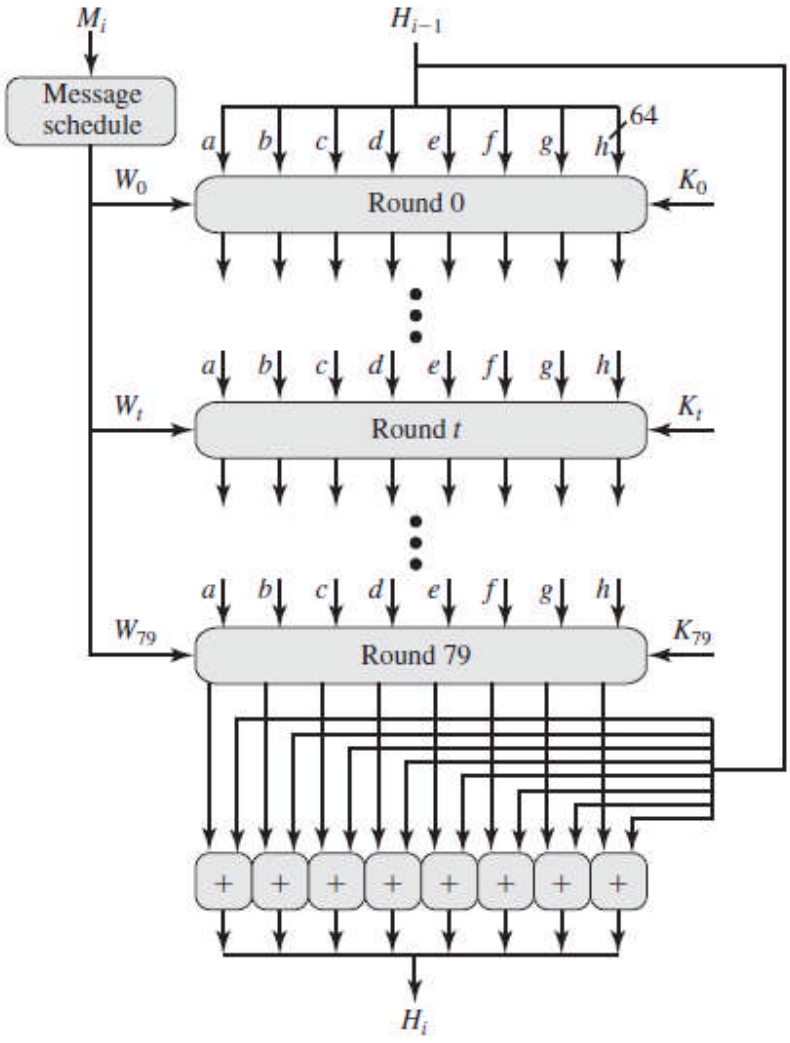


Figure 11.9 SHA-512 Processing of a Single 1024-Bit Block

شكل (7)

في **شكل (7)** كل Round من 80 يدخل عليه قيمة K معين بين (K0-K79) وهي قيم ثابتة مكون من 64 bit تأخذ من الجدول التالي

K0

428a2f98d728ae22	7137449123ef65cd	b5c0fbcfec4d3b2f	e9b5dba58189dbbc
3956c25bf348b538	59f111f1b605d019	923f82a4af194f9b	ab1c5ed5da6d8118
d807aa98a3030242	12835b0145706fbe	243185be4ee4b28c	550c7dc3d5ffb4e2
72be5d74f27b896f	80deb1fe3b1696b1	9bdc06a725c71235	c19bf174cf692694
e49b69c19ef14ad2	efbe4786384f25e3	0fc19dc68b8cd5b5	240ca1cc77ac9c65
2de92c6f592b0275	4a7484aa6ea6e483	5cb0a9dcbd41fbd4	76f988da831153b5
983e5152ee66dfab	a831c66d2db43210	b00327c898fb213f	bf597fc7beef0ee4
c6e00bf33da88fc2	d5a79147930aa725	06ca6351e003826f	142929670a0e6e70
27b70a8546d22ffc	2e1b21385c26c926	4d2c6dfc5ac42aed	53380d139d95b3df
650a73548baf63de	766a0abb3c77b2a8	81c2c92e47edaee6	92722c851482353b
a2bfe8a14cf10364	a81a664bbc423001	c24b8b70d0f89791	c76c51a30654be30
d192e819d6ef5218	d69906245565a910	f40e35855771202a	106aa07032bbd1b8
19a4c116b8d2d0c8	1e376c085141ab53	2748774cdf8eeb99	34b0bcb5e19b48a8
391c0cb3c5c95a63	4ed8aa4ae3418acb	5b9cca4f7763e373	682e6ff3d6b2b8a3
748f82ee5defb2fc	78a5636f43172f60	84c87814a1f0ab72	8cc702081a6439ec
90befffa23631e28	a4506cebde82bde9	bef9a3f7b2c67915	c67178f2e372532b
ca273eceea26619c	d186b8c721c0c207	eada7dd6cde0eb1e	f57d4f7fee6ed178
06f067aa72176fba	0a637dc5a2c898a6	113f9804bef90dae	1b710b35131c471b
28db77f523047d84	32caab7b40c72493	3c9ebe0a15c9bebc	431d67c49c100d4c
4cc5d4becb3e42b6	597f299cfc657e2a	5fcb6fab3ad6faec	6c44198c4a475817

الجزء رقم (3)

كما نلاحظ ان في **شكل (7)** كل Round يدخل عليه قيمة من بيانات Block طولها 64 bit وان طول Block الواحد هو 1024 bit لذالك يقسم (1024) الى 16 Block كل واحد حجمه 64 bitممثلا (W0-W15) ويدخل على اول 16 Round

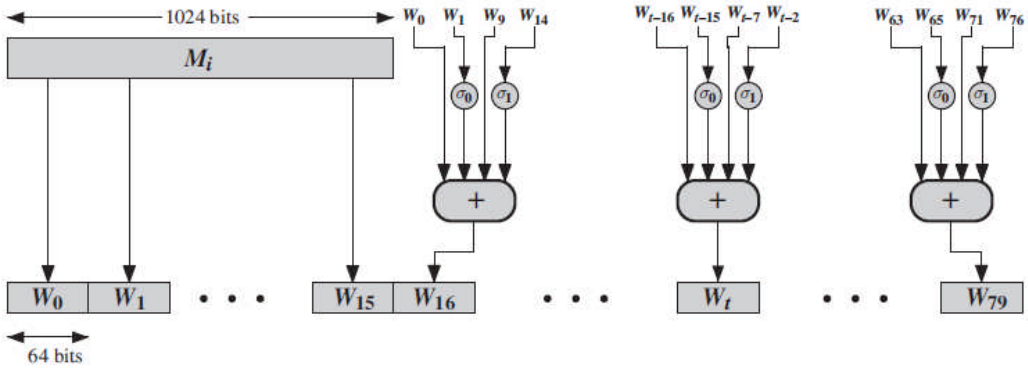


Figure 11.11 Creation of 80-word Input Sequence for SHA-512 Processing of Single Block

شكل (8)

وبقية Round تأخذ Wt وفق المعادلة التالية

$$W_t = \sigma_1^{512}(W_{t-2}) + W_{t-7} + \sigma_0^{512}(W_{t-15}) + W_{t-16}$$

where

$$\sigma_0^{512}(x) = \text{ROTR}^1(x) \oplus \text{ROTR}^8(x) \oplus \text{SHR}^7(x)$$
$$\sigma_1^{512}(x) = \text{ROTR}^{19}(x) \oplus \text{ROTR}^{61}(x) \oplus \text{SHR}^6(x)$$
$$\text{ROTR}^n(x) = \text{circular right shift (rotation) of the 64-bit argument } x \text{ by } n \text{ bits}$$
$$\text{SHR}^n(x) = \text{left shift of the 64-bit argument } x \text{ by } n \text{ bits with padding by zeros on the right}$$
$$+ = \text{addition modulo } 2^{64}$$

جزء رقم (4):

جزء رقم4 في **شكل(5)** يمثل العمليات التي ستجري ضمن كل Round ليتم من خلا لها التحديث على قيم Registers التي ستعتبر القيم المحدثة كمدخل Round التالي وفي اخر Round تعتبر Hash الخاص بالرسالة اذا كانت الرسالة من Block واحد وخلافه تعتبر مدخل لبيانات Registers في Block التالي

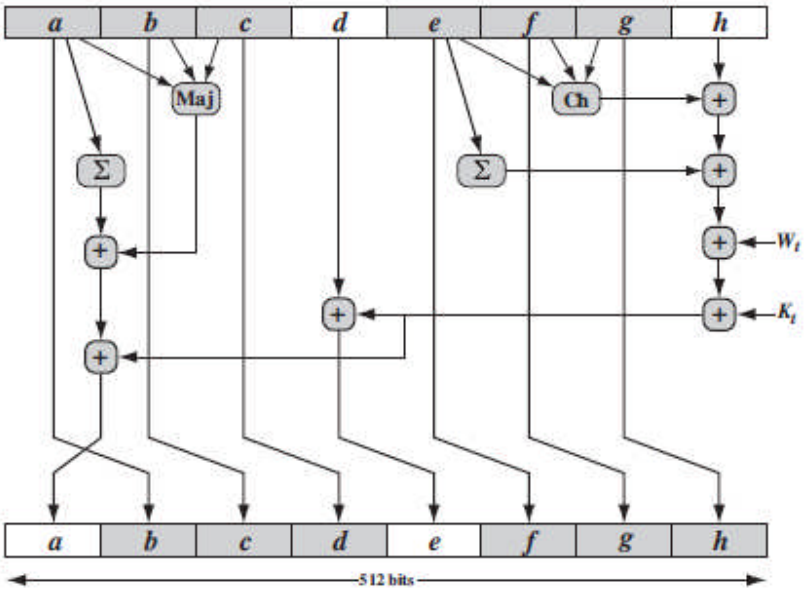


Figure 11.10 Elementary SHA-512 Operation (single round)

شكل (9)

Wt: يمثل 64 bit من بيانات Block تم شرح طريقة تحضيرها لكل Round سابقا

Kt : وهي قيم ثابتة مكون من 64 bit تأخذ من الجدول تم شرحها سابقا

$$T_1 = h + \text{Ch}(e, f, g) + \left(\sum_1^{512} e\right) + W_t + K_t$$
$$T_2 = \left(\sum_0^{512} a\right) + \text{Maj}(a, b, c)$$
$$h = g$$
$$g = f$$
$$f = e$$
$$e = d + T_1$$
$$d = c$$
$$c = b$$
$$b = a$$
$$a = T_1 + T_2$$

where

t = step number; $0 \le t \le 79$

$\text{Ch}(e, f, g) = (e \text{ AND } f) \oplus (\text{NOT } e \text{ AND } g)$
the conditional function: If e then f else g

$\text{Maj}(a, b, c) = (a \text{ AND } b) \oplus (a \text{ AND } c) \oplus (b \text{ AND } c)$
the function is true only of the majority (two or three) of the arguments are true

$\left(\sum_0^{512} a\right) = \text{ROTR}^{28}(a) \oplus \text{ROTR}^{34}(a) \oplus \text{ROTR}^{39}(a)$

$\left(\sum_1^{512} e\right) = \text{ROTR}^{14}(e) \oplus \text{ROTR}^{18}(e) \oplus \text{ROTR}^{41}(e)$

$\text{ROTR}^n(x)$ = circular right shift (rotation) of the 64-bit argument x by n bits

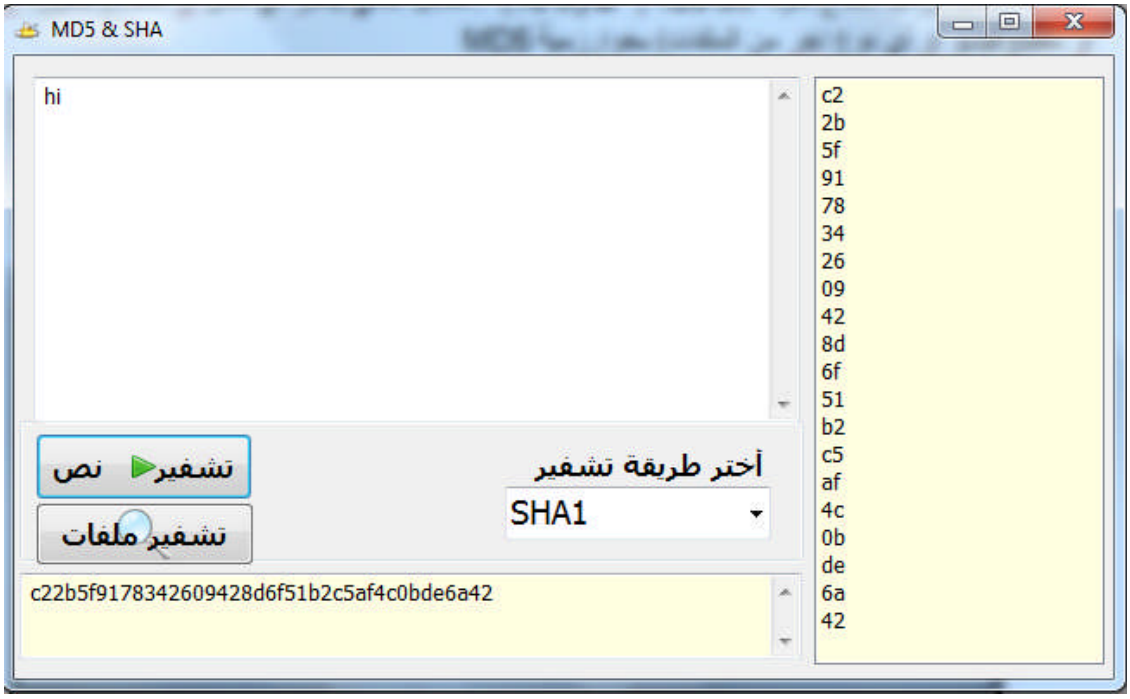
W_t = a 64-bit word derived from the current 512-bit input block

K_t = a 64-bit additive constant

$+$ = addition modulo 2^{64}

تكوين خوارزمية (SHA) بلغة VB.NET

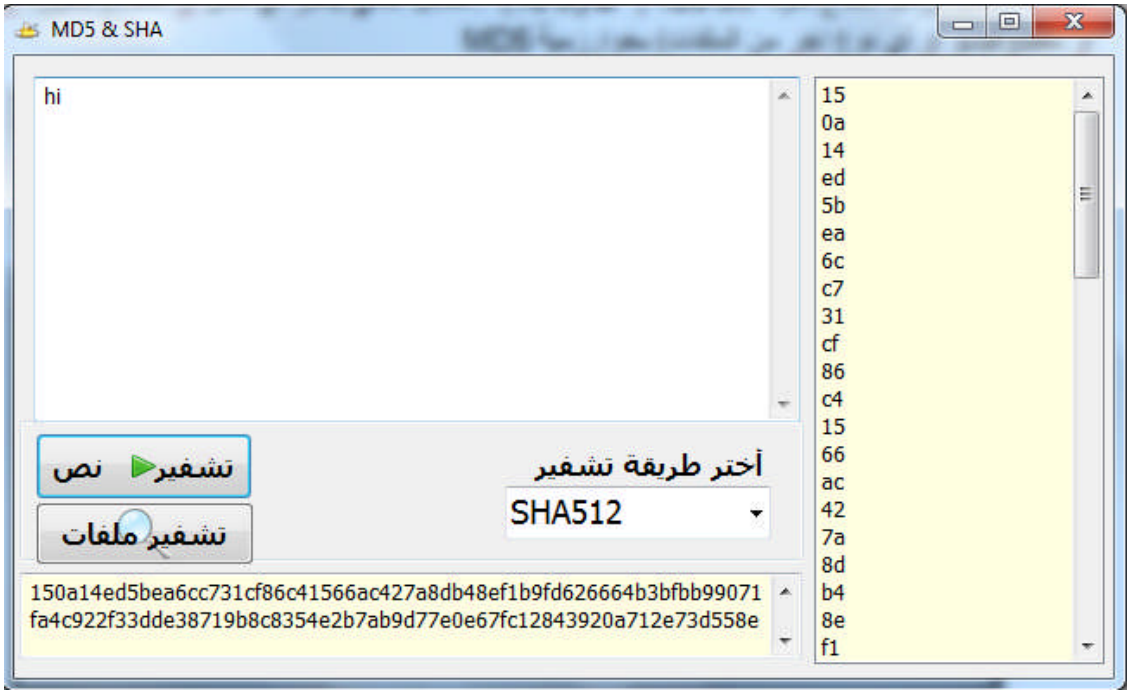
SHA1: تنتج هذه الخوارزمية مفتاح طوله 160 bit (20 byte)



VB.NET CODE

```
Function GetSHA1Data(ByVal data As Byte()) As String
'create new instance of md5
Dim sha As SHA1 = SHA1.Create()

'convert the input text to array of bytes
Dim hashData As Byte() = sha.ComputeHash(data)

'create new instance of StringBuilder to save hashed data
Dim returns As New StringBuilder()

'loop for each byte and add it to StringBuilder
For i As Integer = 0 To hashData.Length - 1
returns.Append(hashData(i).ToString("x2"))
Next
' return hexadecimal string
Return returns.ToString()
End Function
```

SHA512: تنتج هذه الخوارزمية مفتاح طوله 512 bit (64 byte)

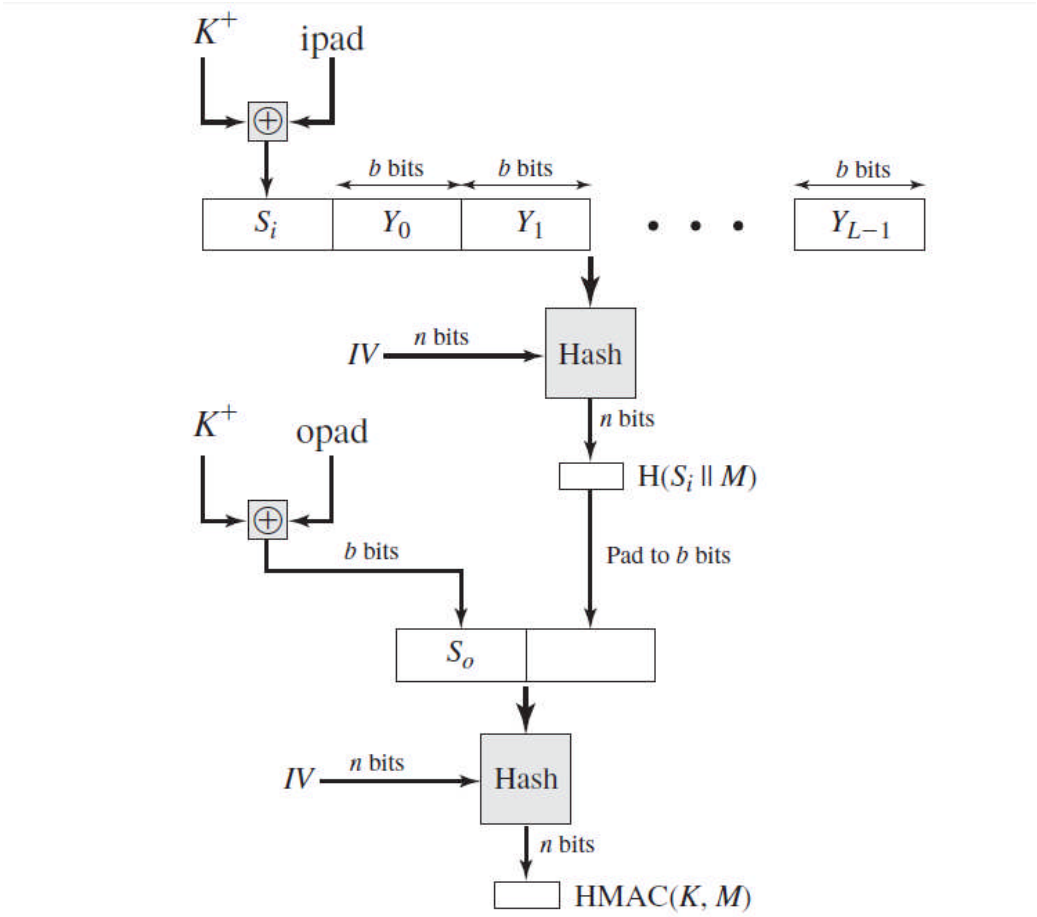


VB.NET CODE

```
Function GetSHA512Data(ByVal data As Byte()) As String
'create new instance of md5
Dim sha As SHA512 = SHA512.Create()

'convert the input text to array of bytes
Dim hashData As Byte() = sha.ComputeHash(data)

'create new instance of StringBuilder to save hashed data
Dim returns As New StringBuilder()

'loop for each byte and add it to StringBuilder
For i As Integer = 0 To hashData.Length - 1
returns.Append(hashData(i).ToString("x2"))
Next
' return hexadecimal string
Return returns.ToString()
End Function
```

خوارزمية MAC Based ON Hash Function (HMAC)

هي إحدى خوارزميات MAC حيث تقوم بإيجاد Hash ثابت من أي نص أو ملف متغير .
حيث يعتمد على أحدى خوارزميات Hash مثلا (MD5,SHA512) بالإضافة إلى Secret Key نحن ندخله شاهد **شكل (10) .** ويختلف MAC عن HASH أن الأول يستخدم Secret Key لإيجاد Hash ثابت من أي رسالة أما الثاني لا يستخدم Secret Key تقوم بإيجاد سلسلة ثابتة من أي نص أو ملف .



شكل (10)

H ,Hash: إحدى خوارزميات Hash التي سنستخدمها هنا مثلا (SHA512,MD5)
[Y0---Yn]: هي بيانات الرسالة الحقيقية Mi التي نريد إيجاد HMAC لها تقسم الى Block حسب خوارزمية Hash المستخدمة مثلا تقسم إلى Block حجمه 512 bit اذا كنا نستخدم MD5 او تقسم الى Block حجمه 1024bit اذا كنا نستخدم SHA512
b bits: يمثل عدد Bit ضمن كلBlock من بيانات الرسالة الحقيقية

K+:هو Secret Key الذي نحن ندخله على رسالة لأننا نستخدم هنا MAC وهذه الخوارزمية تحتاج الى Secret Key.

اذا كان عدد Bits الخاص Secret Key اقل من عدد b bits الخاص بكل Block نضيف أصفار إلى الجهة اليسرى من Secret Key الى ان يصبح عدد bits الخاص به مساوي لعدد bits الخاص بكل Block .

مثلا اذا كنا نستخدم في HMAC خوارزمية MD5 هذه الخوارزمية تقسم البيانات الى Blockحجمه 512 bits وكان Secret Key من 400 bits نحتاج الى إضافة 122 صفر في الجهة اليسرى من Secret Key

اذا كان عدد Bits الخاص Secret Key اكبر من عدد b bits الخاص بكل Block نعمل Hash على Secret Key لكي نقلل عدد bits الخاص به

مثلا: اذا كنا نستخدم في HMAC خوارزمية MD5 هذه الخوارزمية تقسم البيانات الى Blockحجمه 512 bits وكان Secret Key من 1024 bits نعمل hash على Secret Key لكي نقلل Bits الخاص به الى 512 bits هنا سنستخدم خوارزمية SHA512 لأنها تجد Hash طوله 512 bits من Block طوله 1024 bits

ipad: هو رقم معين مثلا نستخدم هنا (36 hex)

opad: هو رقم معين مثلا نستخدم هنا (5C hex)

Si :هي عملية Xor بين ipad و +K وبعدها يلحق الناتج على شكل Block الى Blocks الخاص بالرسالة الحقيقية لينفذ خوارزمية Hash عليها. وطول البيانات في آخر Block يمثل طول بيانات الرسالة الحقيقية بالإضافة إلى طول Si

iV: هي القيم الأولية لبيانات Registers المستخدم في خوارزمية Hash

H(Si||Mi): تنفيذ احدى خوارزميات Hash على بيانات الرسالة الحقيقية مضافة اليها Si

S0: هي عملية Xor بين opad و +K

n bits: هو Hash الناتج من احدى خوارزميات Hash التي استخدمناها على الرسالة تقوم بإلحاقها ببيانات S0 ونطبق عليها احدى خوارزميات Hash التي استخدمناها في البداية لينتج لنا HMAC

تختصر خوارزمية HMAC على شكل المعادلة التالية :

$$\mathrm{HMAC}(K, M) = \mathrm{H}[(K^+ \oplus \mathrm{opad}) \| \mathrm{H}[(K^+ \oplus \mathrm{ipad}) \| M]]$$

نستخدم (ipad,opad) للحصول على مفتاحين مختلفين من مفتاح واحد

برمجة الخوارزمية HMAC بلغة VB.NET

نكون دالة تستقبل الرسالة و لمفتاح Secret Key بصيغة Byte ونحدد خوارزمية Hash التي سنستخدمها داخل HMAC وتعيد لنا مفتاح ثابت من أي رسالة سنرسلها لها

VB.NET CODE

```
Function GetHMACData(ByVal data As Byte(), ByVal key As Byte()) As
String
'create new instance of md5
Dim sha As HMAC = HMAC.Create()
'حددنا مفتاح الخاص بهذه الخوارزمية
sha.Key = key
'  SHA512حددنا هنا HMAC داخل  التي سنستخدمها Hashنحدد خوارزمية '
sha.HashName = "SHA512"
'convert the input text to array of bytes
Dim hashData As Byte() = sha.ComputeHash(data)

'create new instance of StringBuilder to save hashed data
Dim returns As New StringBuilder()

'loop for each byte and add it to StringBuilder
For i As Integer = 0 To hashData.Length - 1
returns.Append(hashData(i).ToString("x2"))
Next
' return hexadecimal string
Return returns.ToString()
End Function
```

عند استدعاء الدالة نرسل لها الرسالة والمفتاح

VB.NET CODE

```
 TextBox2.Text =
GetHMACData(Encoding.[Default].GetBytes(TextBox1.Text),
Encoding.[Default].GetBytes("1234"))
```