Hash Functions

ECE 650
Methods & Tools for Software Engineering (MTSE)
Fall 2018

Prof. Alireza Sharifi

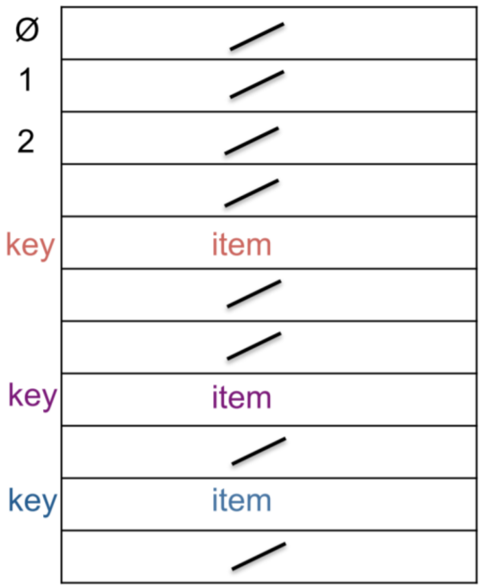
UNIVERSITY OF WATERLOO

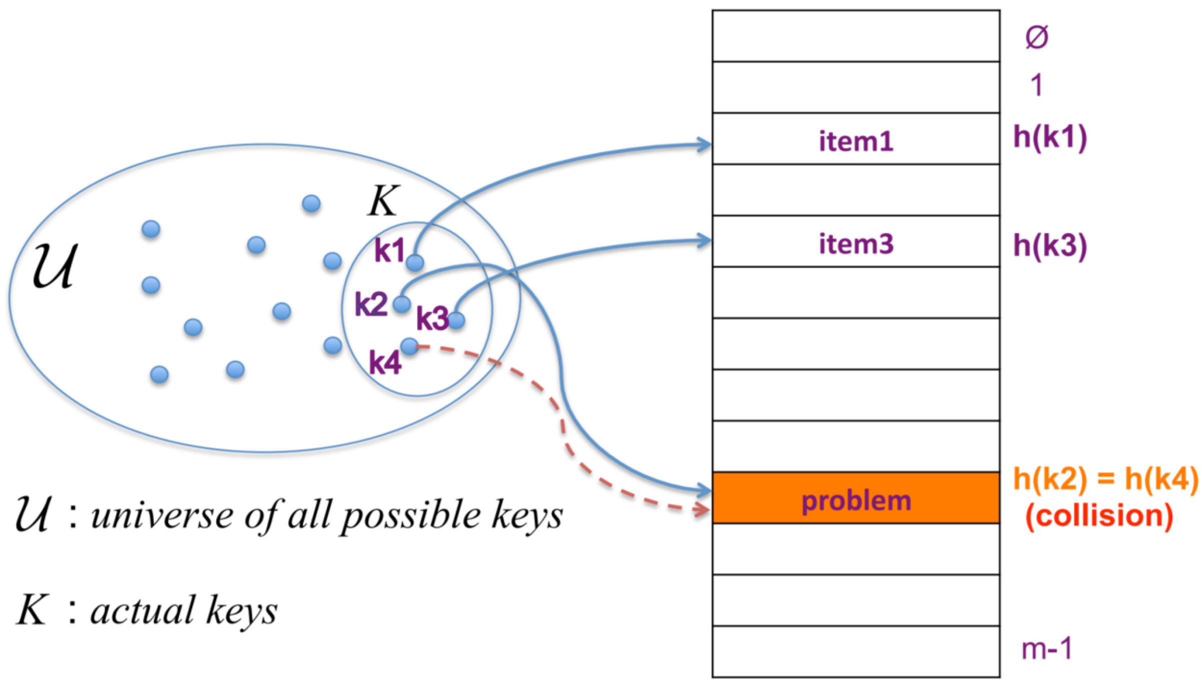
Hash Tables Operations

Operations to Support in Hash Tables:

- insert(item): add item to set
- delete(item): remove item from set
- search(key): return item with key if it exists

Direct Access Table vs Universal Hashing

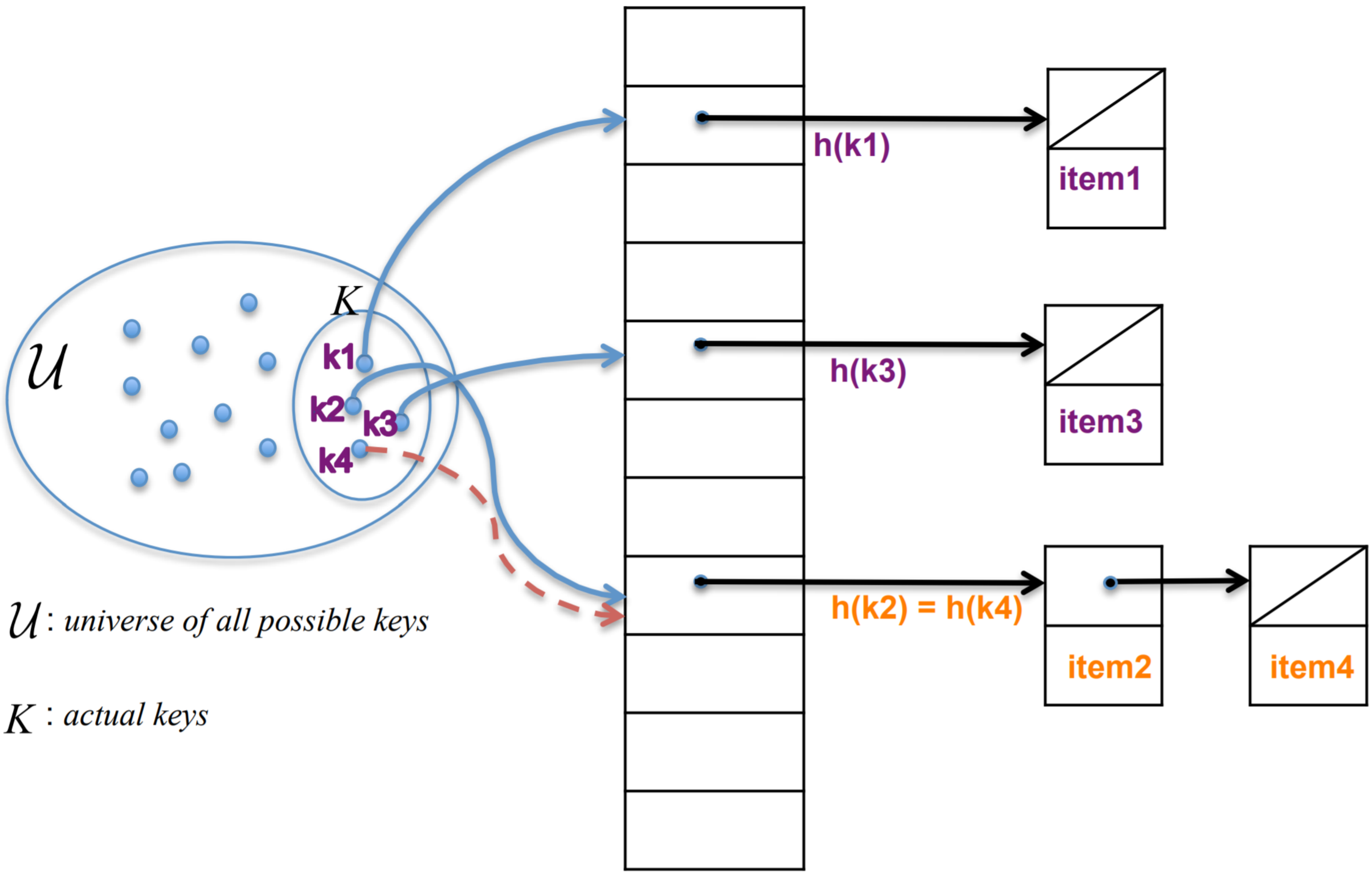




https://courses.csail.mit.edu

UNIVERSITY OF WATERLOO

Chaining



𝒰 : *universe of all possible keys*

K : *actual keys*

https://courses.csail.mit.edu

Hash Functions

There are six basic requirements for a hash function h:

1. It should accept any length message as input.

2. It should produce a small fixed sized output (~100 bits).

3. It should be easy and fast to compute *h* for any input.

4. The function *h* should be a one-way function (OWHF) hard or impossible to invert. That is given *h(m)* it should be very difficult to recover m.

5. It should be resistant to weak collisions. A weak collision occurs when given *u*, it is infeasible to find w such that $h(u)=h(w)$.

6. It should be resistant to strong collisions – it should be almost impossible to find two *meaningful messages* u and w such that $h(u) = h(w)$.

   Functions satisfying this condition are known as Collision Resistant Functions (CRFs).

Example:

A simple hash function can be constructed using the XOR operation by dividing the message into fixed sized blocks and XORing each block to produce the hash output.

For example, using 8 bit blocks the hash value for message "Go now" (given in its ASCII code) is:

G	0	1	0	0	0	1	1	1
o	0	1	1	0	1	1	1	1
n	0	1	1	0	1	1	1	0
0	0	1	1	0	1	1	1	1
w	0	1	1	1	0	1	1	1
5E	0	1	0	1	1	1	1	0

This is a very weak method.

One-Way Functions

Hashes come from two basic classes of one-way functions:

- Mathematical
  - Multiplication: $Z=X \bullet Y$
  - Modular Exponentiation: $Z = Y^X \pmod n$ (Chaum –van Heijst – Pfitzmann Hash)
- Ad-hoc (Symmetric cipher-like constructions)
  - Custom Hash functions (MD4, SHA-x, MD5, RIPEMD)

UNIVERSITY OF WATERLOO

The Birthday Problem

- In a room with N people, how large must N be so that the probability that two people celebrate their birthday on the same day and month is greater than 0.5?

Solution:

- For the first person that enters the room there are 365 distinct possible birthdays.
- When a second person enters the room there are 364 possible birthdays that do not match the first person's, so the probability of no match is 364/365
- When a third person enters the room there are 363 possible birthdays that do not match either of the other two so now the probability of no match is:

$$\frac{(364)(363)}{(365)(365)}$$

- When N people are in the room, this generalizes to a probability of no match given by:

$$\frac{(365-N+1)\cdots(362)(363)(364)}{(365)^{N-1}}$$

or a probability of a match of:

$$1-\frac{(365-N+1)\cdots(362)(363)(364)}{(365)^{N-1}}$$

- The value of N which makes the probability of a match greater than 0.5 is 23.

- The Birthday Problem can be restated in terms of an attack on a hash function as:

    - Given a hash function which produces a message digest of r bits, how many messages have to be examined so that the probability that two messages have the same hash value is greater than 0.5?

- Since the message digest is r bits long, the total number of messages is 2^r.

- Hence the number of messages required is about $2^{r/2}$.

Popular Practical Cryptographic Hashes

– MD4, MD5

– SHA-1, SHA-224, SHA-256, SHA-384, SHA-512

– Whirlpool

MD5

- MD5 accepts as input a message of any length and produces a 128-bit message digest as output.

- Given a message of length L bits, there are three steps required to set up the algorithm.

1. Padding

   - The first is to pad the message by adding extra bits to the end of the message.

   - Padding is a common feature of most hash functions and done correctly it can add to the security of algorithm.

- For MD5, the message is padded so that its bit length L is $L \equiv 448 \bmod 512$. This is 64 bits less than an integer multiple of 512 bits.

- Even if the original message is of the desired length, padding is added.

- The padding consists of a single 1 bit followed by enough 0 bits to create the required length.

- For example, if a message consisted of 704 bits, 256 bits would be added on to the end (a 1 followed by 255 0's) to expand the message to 960 bits (960 mod 512 = 448).

2. The original length of the message reduced mod 2^64 is added to the end of the expanded message as a 64 bit number.

   - In the example, the original size was 704 bits which in binary is 1011000000.

   - This is written as a 64 bit number (54 0's are appended to the beginning) and added to the end of the expanded message.

   - The result is a message with 1024 bits.

3. Third, the initial input to MD5 is placed in four 32-bit registers A, B, C, D which will later hold the intermediate and final results of the hash function.

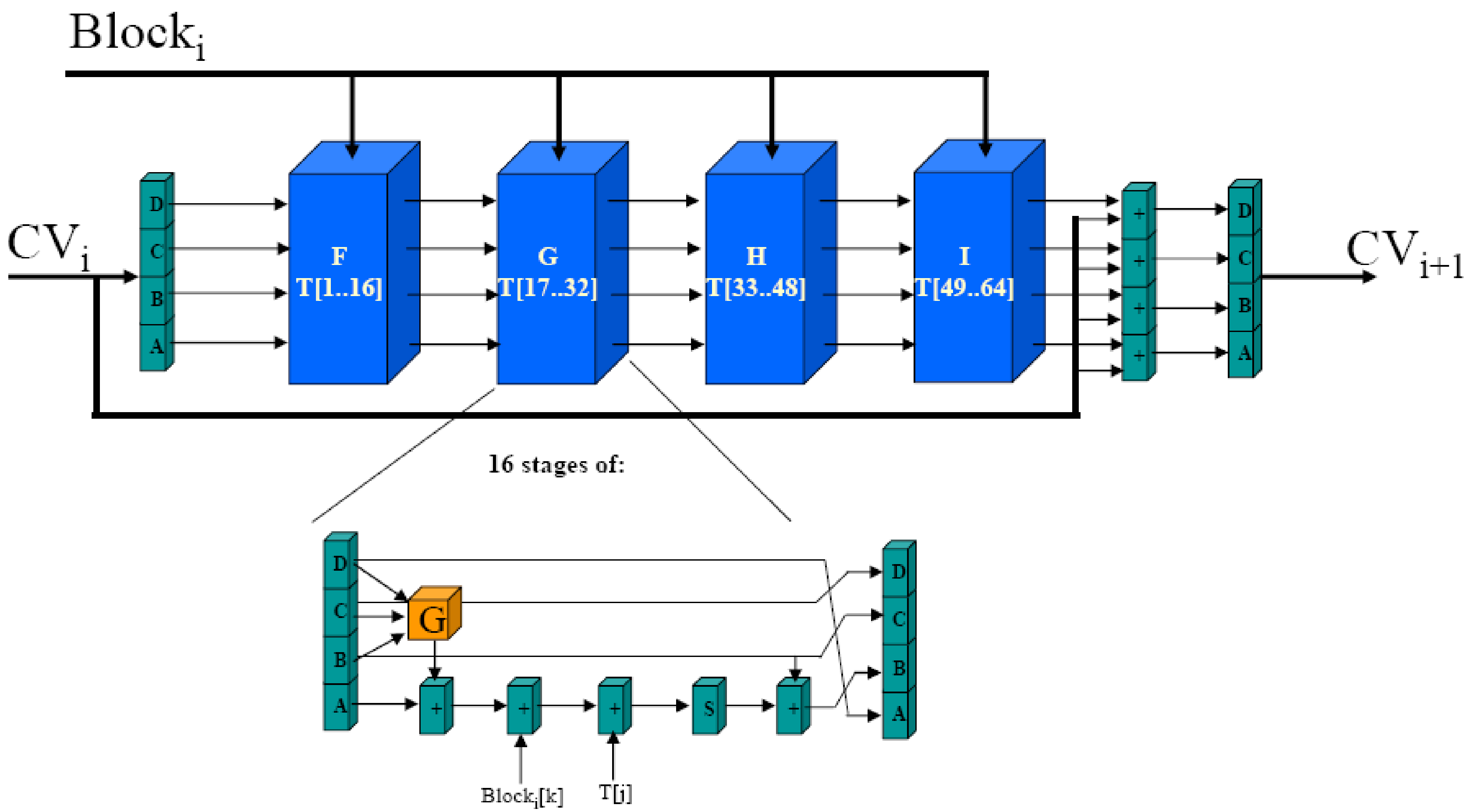
   - The initial values (in hex) are:

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

MD5 Operation

- Once these set-up steps are completed, MD5 will process each 512-bit block in four rounds.

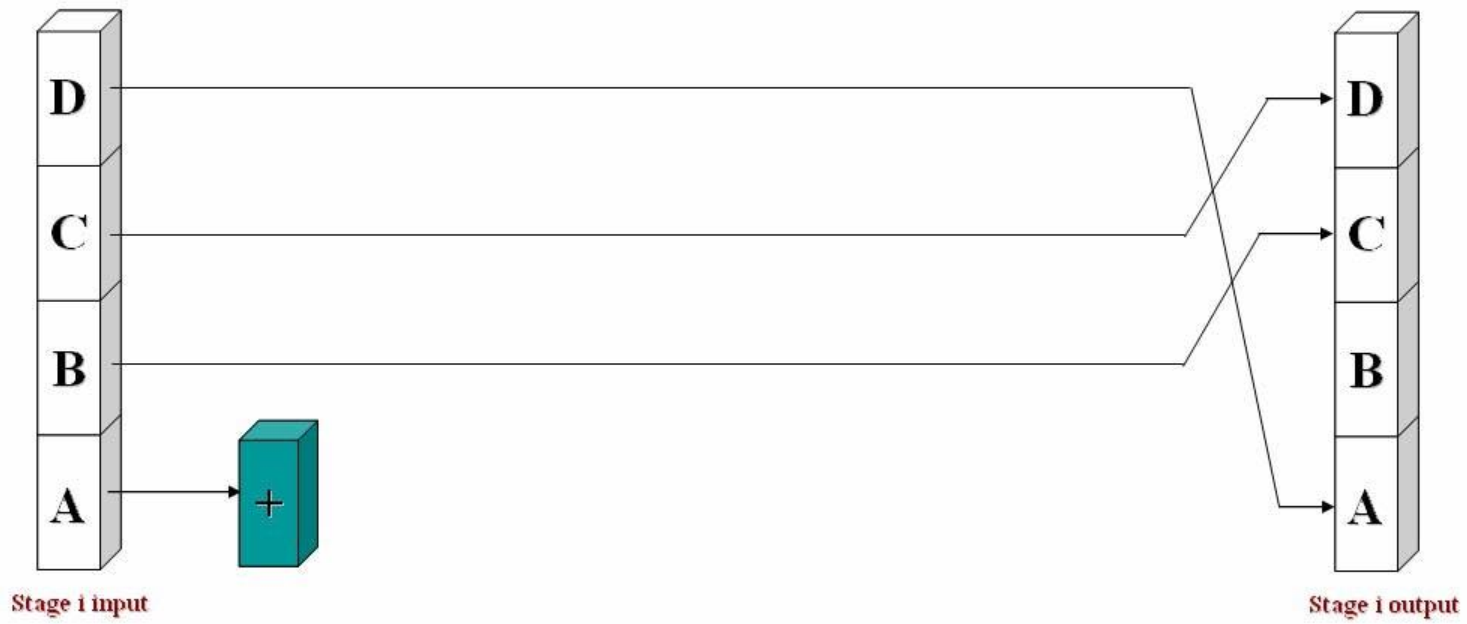
- Each round consists of 16 stages which implement a round specific function (F, G, H, or I), 32-bit addition to part of the message block, 32-bit addition to a built-in value found in the array T, a shift operation, and a final addition and permutation operation.



MD5 Rounds and Stages

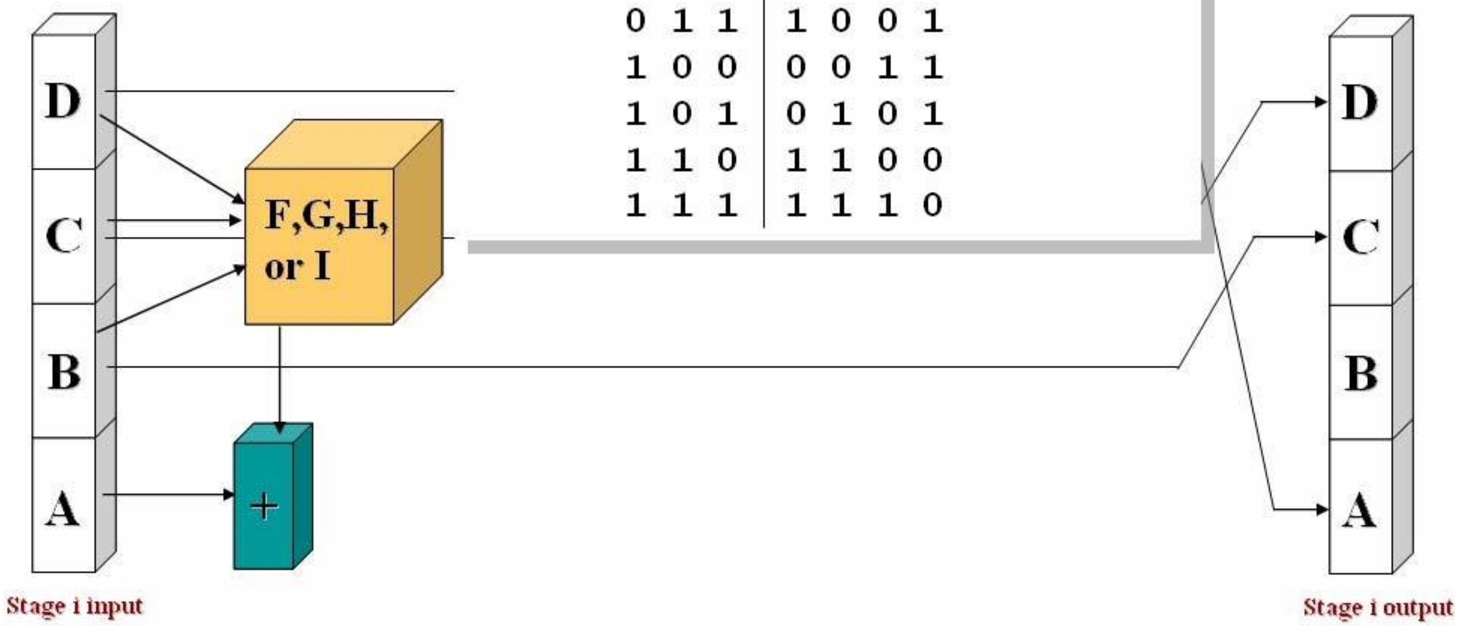
UNIVERSITY OF WATERLOO

Five 32-bit registers

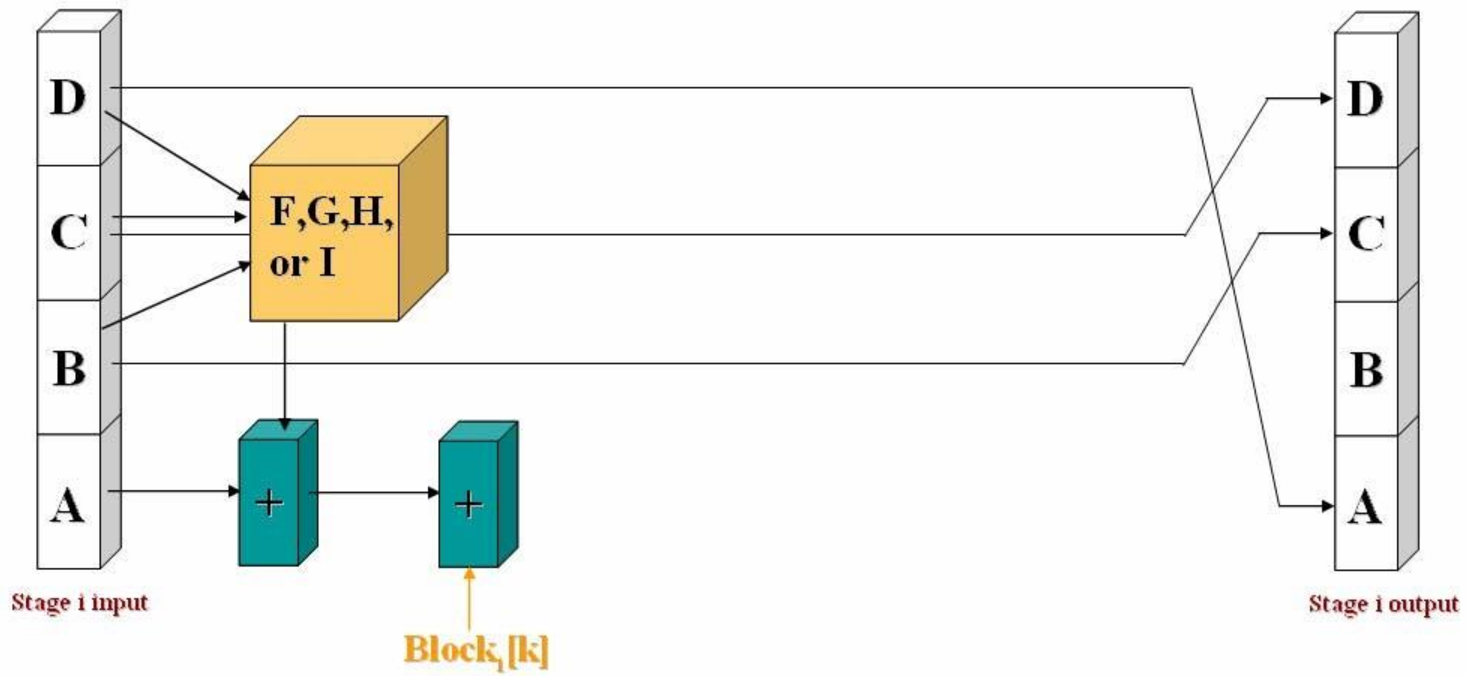

D
C
B
A
+
Stage i input
D
C
B
A
Stage i output

The round specific functions accept three 32-bit words as input and produces a 32-bit output based on the 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



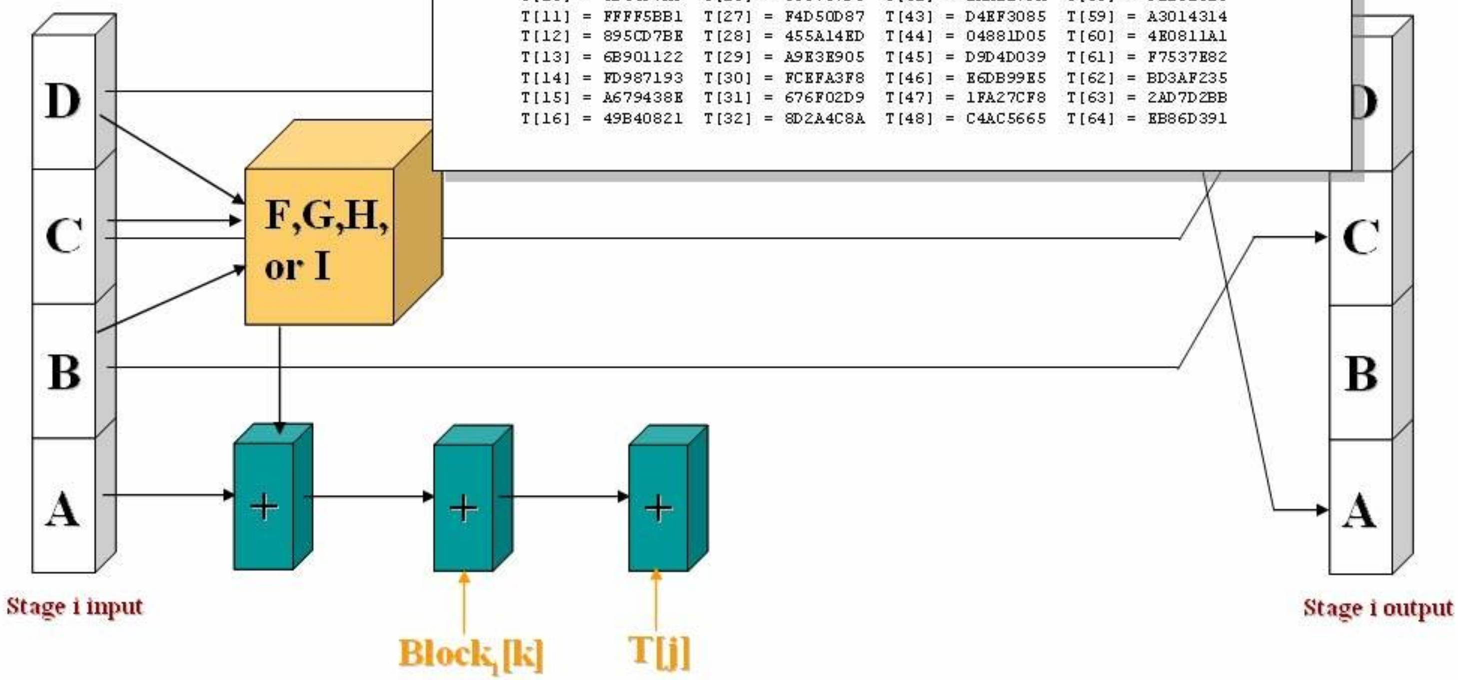
The 512-bit input is divided into sixteen 32-bit blocks. In the first round the blocks are used in order (k= 1 to 16). In the second round, k = (1 + 5j) mod 16 where j is the current stage. In the third round, k = (5 + 3j) mod 16 and in the fourth round k= 7j mod 16.



UNIVERSITY OF WATERLOO

The constant elements from the array T are derived from the sin function

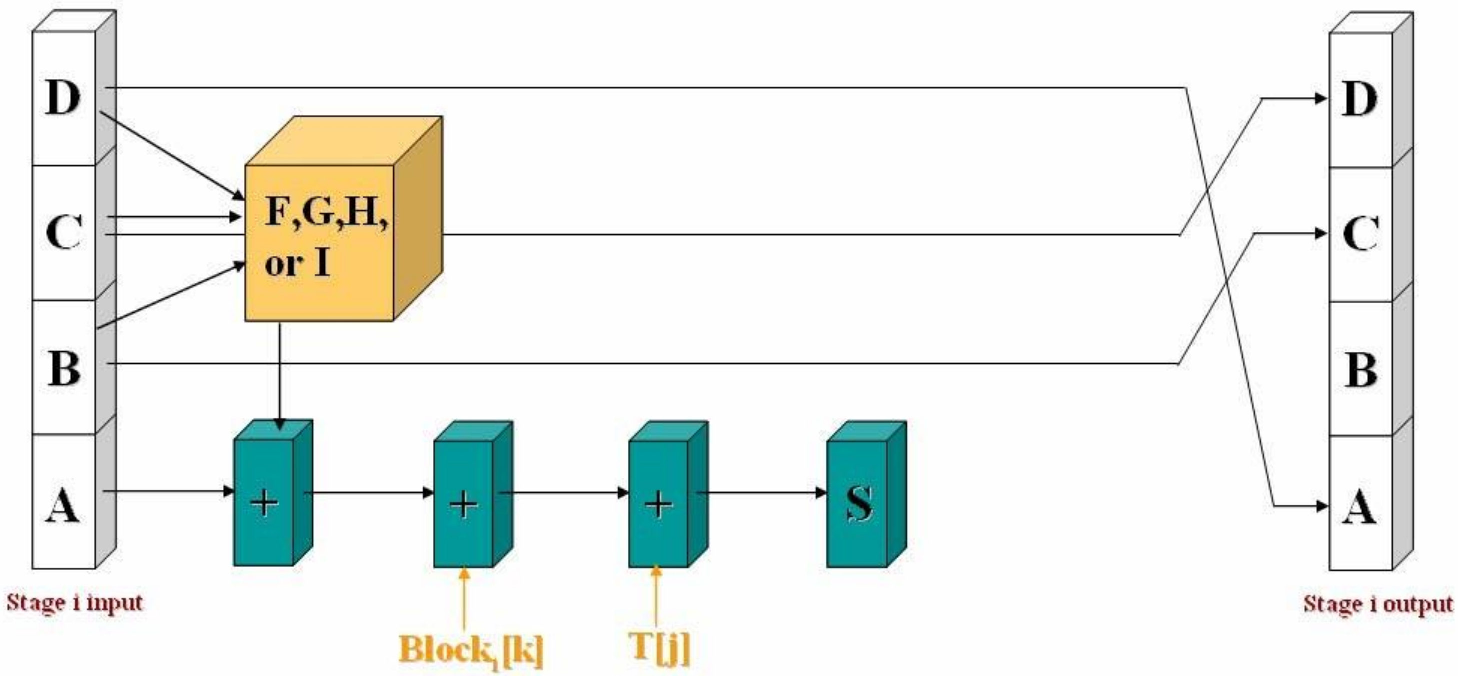
T[1] = D76AA478	T[17] = F61E2562	T[33] = FFFA3942	T[49] = F4292244
T[2] = E8C7B756	T[18] = C040B340	T[34] = 8771F681	T[50] = 432AFF97
T[3] = 242070DB	T[19] = 265E5A51	T[35] = 699D6122	T[51] = AB9423A7
T[4] = C1BDCEEE	T[20] = E9B6C7AA	T[36] = FDE5380C	T[52] = FC93A039
T[5] = F57C0FAF	T[21] = D62F105D	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



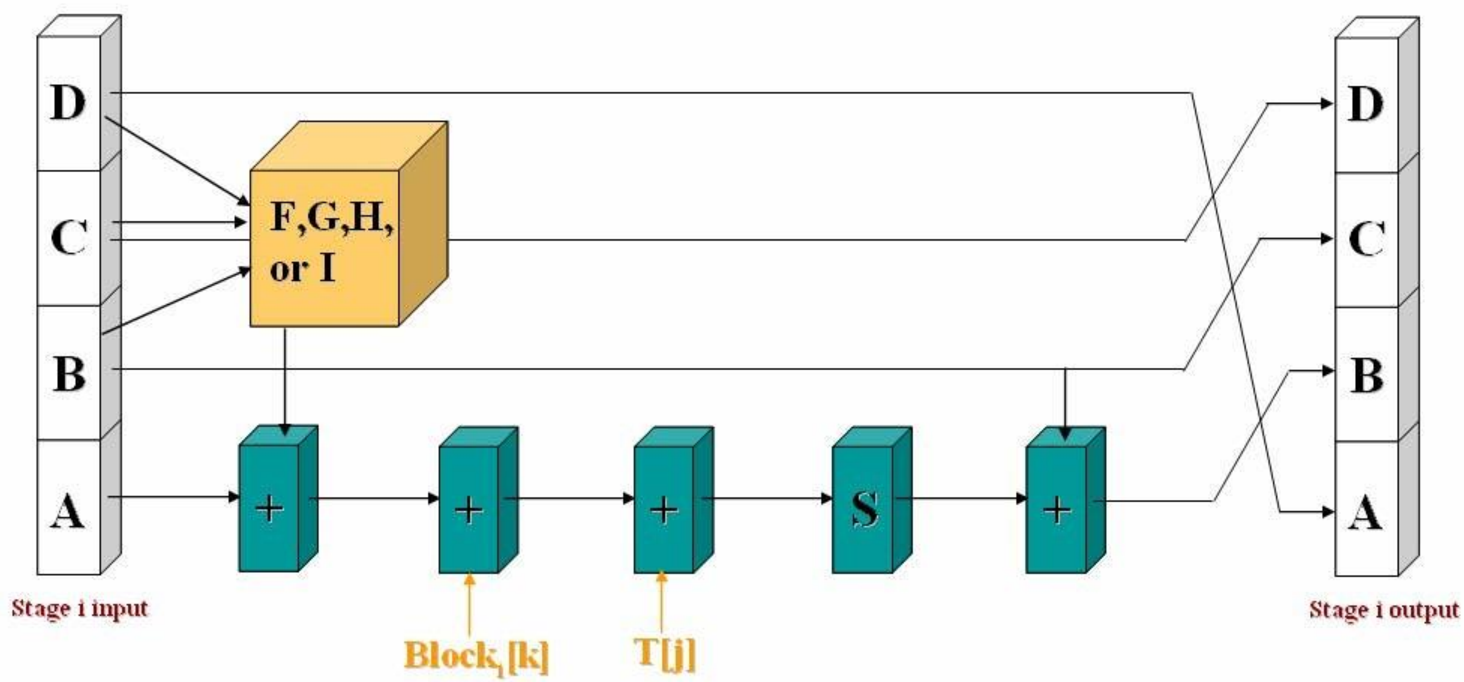
UNIVERSITY OF WATERLOO

The value is rotated to the left by a variable number of bits defined by the schedule:

	Stage															
Round	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	7	12	17	22	7	12	17	22	7	12	17	22	7	12	17	22
2	5	9	14	20	5	9	14	20	5	9	14	20	5	9	14	20
3	4	8	16	23	4	8	16	23	4	8	16	23	4	8	16	23
4	6	10	15	21	6	10	15	21	6	10	15	21	6	10	15	21

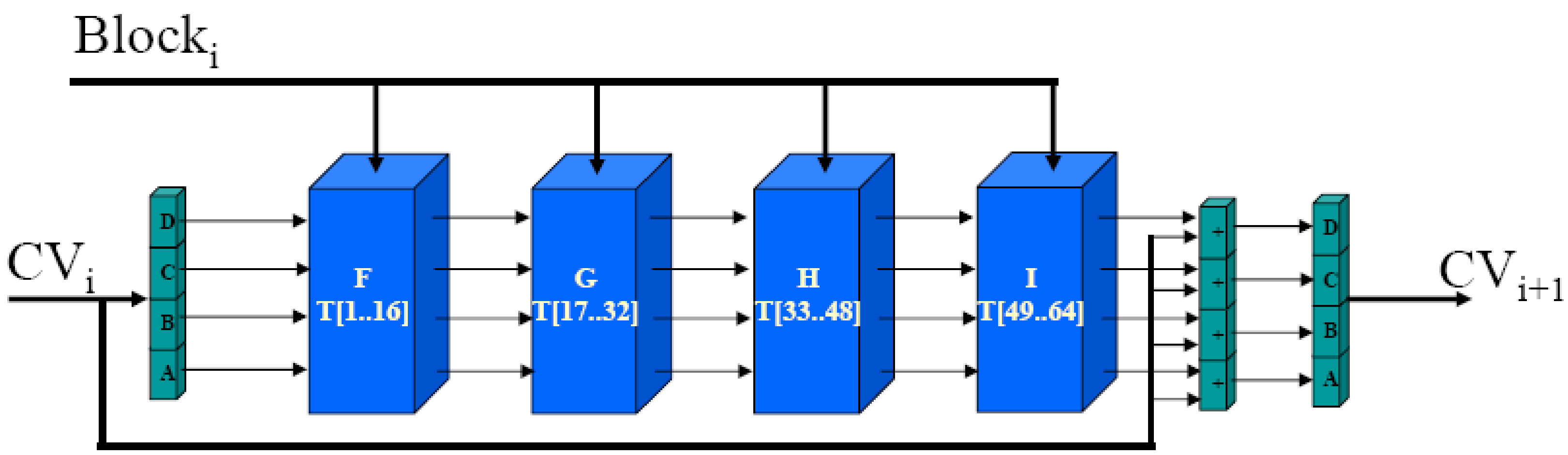


UNIVERSITY OF WATERLOO


D
C
B
A
F,G,H,
or I
+
+
+
S
+
D
C
B
A
Stage i input
Stage i output
Block$_i$[k]
T[j]

Final Operation

- After all four rounds, the original contents of ABCD is added to the new contents of ABCD to produce the output for the i^{th} message block.
- This output serves as an input to begin processing the $(i+1)^{th}$ message block.
- The 128-bit contents of ABCD which remain after the final message block is processed form the hash value.



UNIVERSITY OF WATERLOO

Secure Hash Algorithm (SHA)

- SHA was originally designed by NIST & NSA in 1993
- It was revised in 1995 as SHA-1

- SHA is the US standard for use with DSA signature
  - Scheme standard is FIPS 180-1 1995, and
  - also Internet RFC3174
- Note: the algorithm is SHA, the standard is SHS

- SHA is based on design of MD4 with key differences
- It produces 160-bit hash values

Since SHA-1 was modeled after MD4 a precursor to MD5 so it has many of MD5's features:

- It accepts a message of any size and produces a 160-bit digest
- It works on blocks of 512 bits divided up into 32-bit words
- It runs in four rounds with 20 steps per round
- The message is padded by the method used in MD5

Revised Secure Hash Standard

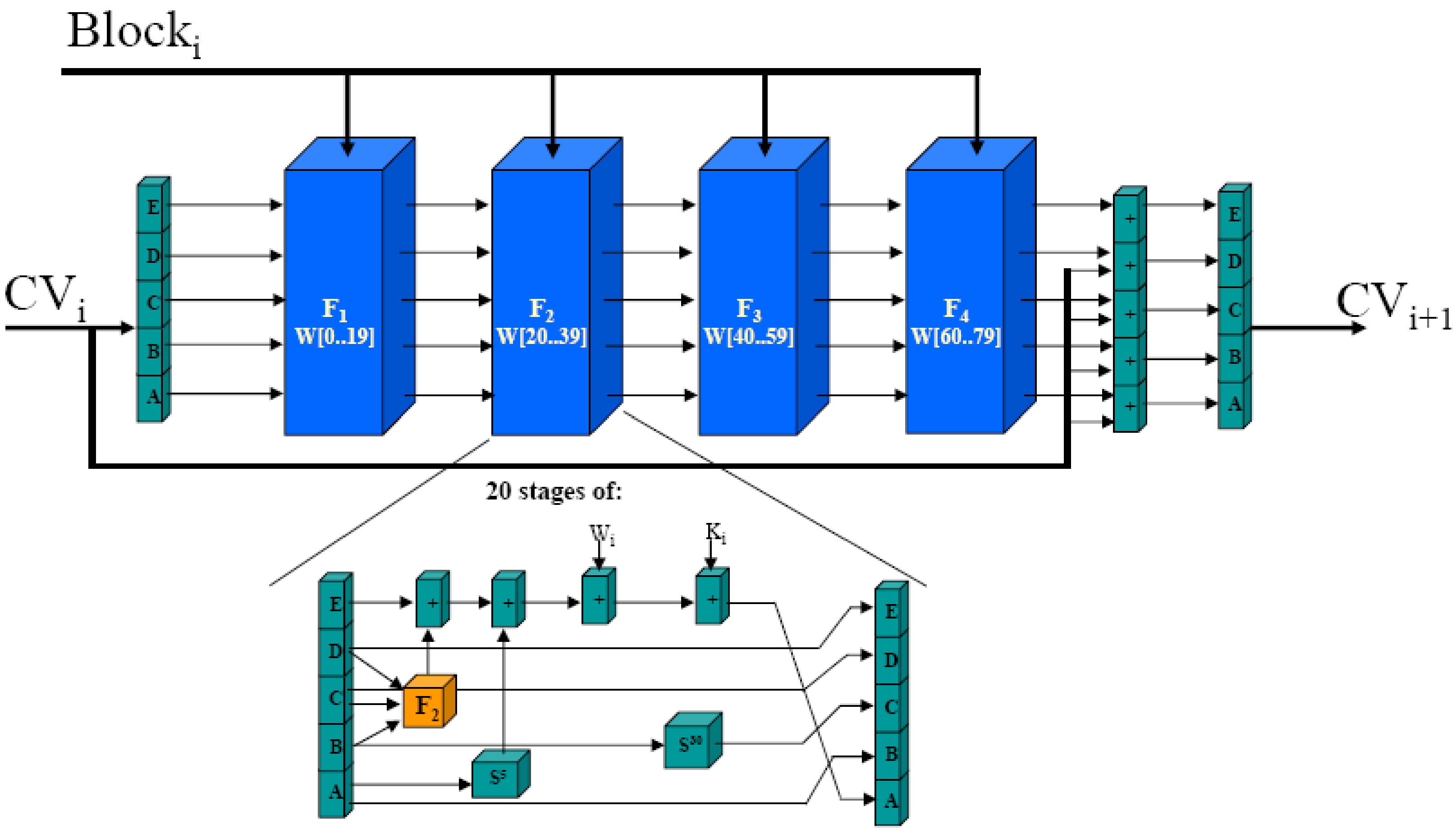
- NIST issued revision FIPS 180-2 in 2002
- Revision adds 3 additional versions of SHA: SHA-256, SHA-384, SHA-512
- They were designed for compatibility with increased security provided by the AES cipher
- Their structure is similar to SHA-1
- Security levels are rather higher

Comparison of SHA Parameters

	SHA-1	SHA-256	SHA-384	SHA-512
Message digest size	160	256	384	512
Message size	$<2^{64}$	$<2^{64}$	$<2^{128}$	$<2^{128}$
Block size	512	512	1024	1024
Word size	32	32	64	64
Number of steps	80	64	80	80
Security	80	128	192	256

- All sizes are measured in bits.
- Security refers to the fact that a birthday attack on a message digest of size n produces a collision with the work factor of approximately $2^{n/2}$.

SHA-1



Block$_i$
CV$_i$
E
D
C
B
A
F$_1$
W[0..19]
F$_2$
W[20..39]
F$_3$
W[40..59]
F$_4$
W[60..79]
+
CV$_{i+1}$
20 stages of:
W$_i$
K$_i$
F$_2$
S^5
S^{30}

SHA-1 Operation

- The initial input to SHA-1 is placed in five 32-bit registers A, B, C, D and E which will later hold the intermediate and final results of the hash function.

- The initial values (in hex)

  are: A = 67452301

  B = EFCDAB89
  C = 98BADCFE
  D = 10325476
  E = C3D2E1F0

UNIVERSITY OF WATERLOO

- The message is divided up into blocks of 512 bits consisting of sixteen 32-bit words.

- The 16 words in the block are expanded into 80 words by a process of mixing and shifting defined by:

$$\text{Initial Block: } M_0, M_1, \ldots, M_{15}$$

$$W[0] = M_0, W[1] = M_1, \ldots, W[15] = M_{15}$$

$$W[t] = W_{t-16} \text{ XOR } W_{t-14} \text{ XOR } W_{t-8} \text{ XOR } W_{t-3}$$

for *t* from 15 to 80

- Rather than use 64 constants like MD5, only four constant values are added into the various stages.

- They are defined by the array K where:

K[t] = 5A827999	for t from 0 to 19
K[t] = 6ED9EBA1	for t from 20 to 39
K[t] = 8F1BBCDC	for t from 40 to 59
K[t] = CA62C1D6	for t from 60 to 79

SHA-1 and Pseudo-Random Numbers

- SHA-1 can be used to generate pseudo-random numbers as well as to hash messages
- SHA-1 is modified slightly into a function G(t,c).
  - The first parameter, t, specifies the initial values of the five registers A to E.
  - The second parameter, c, serves as the message to be hashed by SHA-1.
  - The following algorithm will generate m random numbers based on a secret key, XKEY, and a set of optional seed values $XSEED_j$ (j =0 to m-1).

1. Select a secret key value XKEY, the number of random numbers to generate, m, the maximum number of bits in each random number, b, and (if desired) a set of m seed values XSEEDj and a 160 bit random prime q.

2. Initialize t to 67452301 EFCDAB89 98BADCFE 10324576 C3D2E1F0

3. For j = 0 to m-1 do

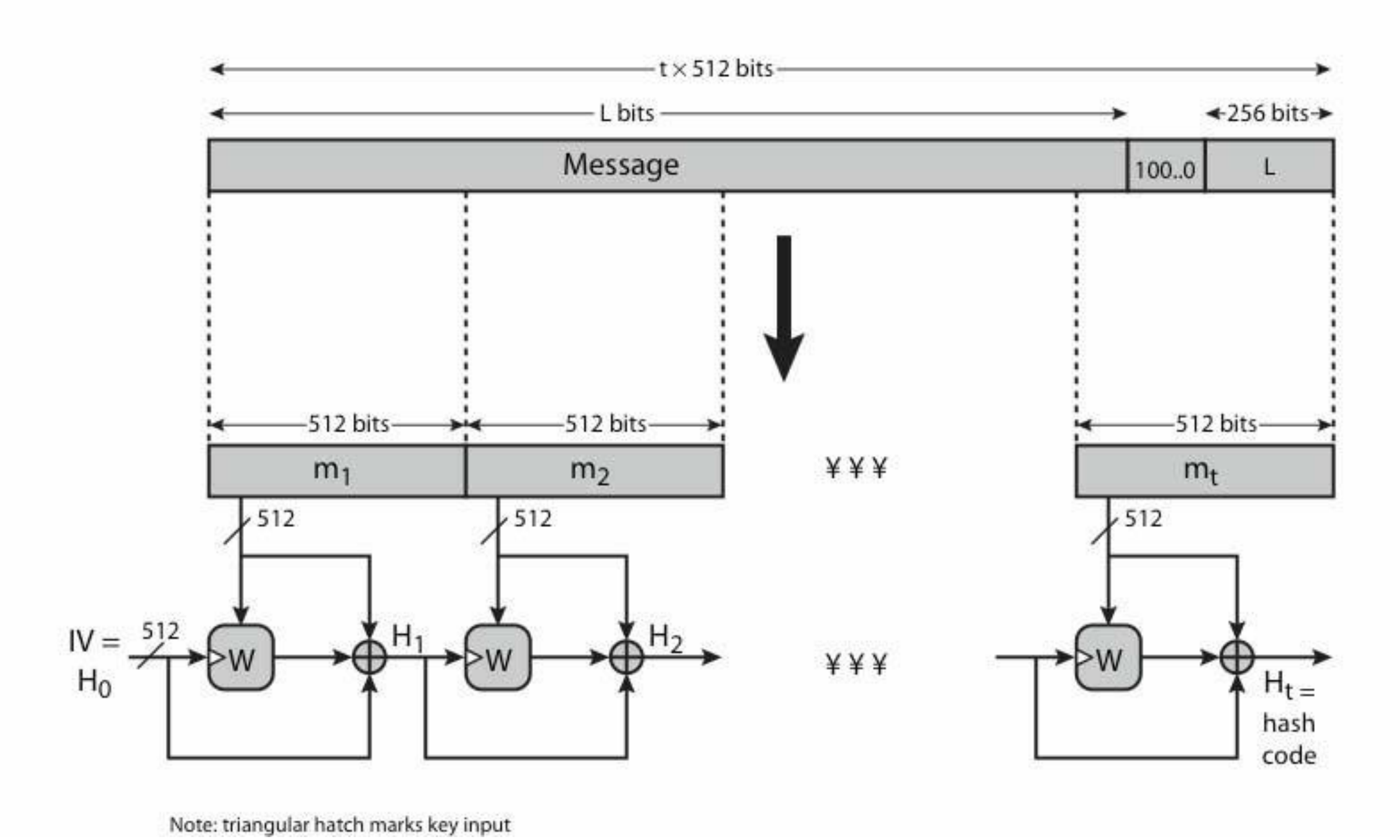
   a. XVAL = (XKEY + XSEEDj) mod 2b
   b. x_j = G(t,XVAL) mod q
   c. XKEY = (1 + XKEY + x_j) mod 2b

The x_j's are the random numbers.

Whirlpool Hash Algorithm

- Whirlpool hash function was endorsed by European NESSIE project
- It uses modified AES internals as compression function addressing concerns on use of block ciphers seen previously
- Whirlpool has performance comparable to dedicated algorithms like SHA

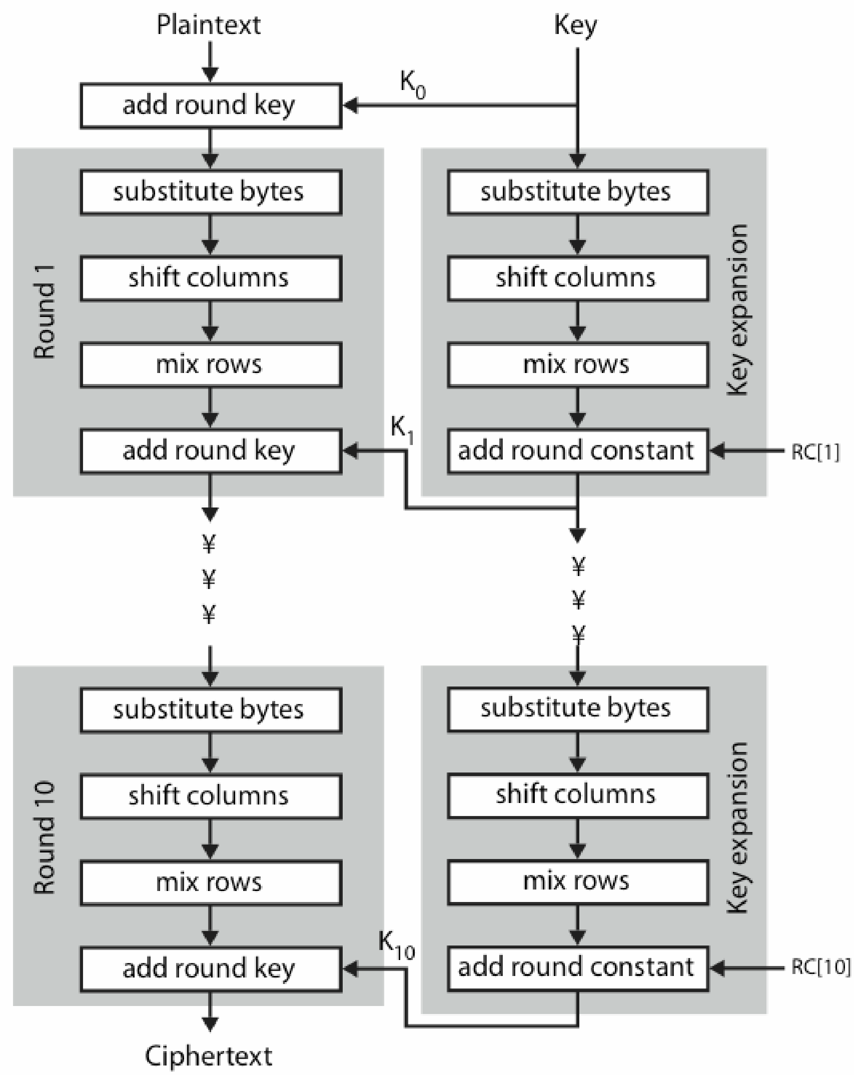
Whirlpool Overview



UNIVERSITY OF WATERLOO

Whirlpool Block Cipher W

- Designed specifically for hash function use
- With security and efficiency of AES but with 512-bit block size and as such-hash
- Similar structure and functions as AES but
  - input is mapped row-wise
  - has 10 rounds
  - a different primitive polynomial for $GF(2^8)$
  - uses different S-box design & values



UNIVERSITY OF WATERLOO