INTERNATIONAL STANDARD

ISO/IEC 10118-4

First edition
1998-12-15

Information technology — Security techniques — Hash-functions —

Part 4:
Hash-functions using modular arithmetic

Technologies de l'information — Techniques de sécurité — Fonctions de brouillage —

Partie 4: Fonctions de hachage utilisant l'arithmétique modulaire

Reference number
ISO/IEC 10118-4:1998(E)

Contents

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

International Standard ISO/IEC 10118-4 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology,* Subcommittee SC27, *IT Security techniques.*

ISO/IEC 10118 consists of the following parts, under the general title *Information technology – Security techniques – Hash-functions:*

– Part 1: *General*

– Part 2: *Hash-functions using an n-bit block cipher algorithm*

– Part 3: *Dedicated hash-functions*

– Part 4: *Hash-functions using modular arithmetic*

Annexes A, B and C of this part of ISO/IEC 10118 are for information only.

Information technology — Security techniques — Hash functions —

Part 4:
Hash-functions using modular arithmetic

1 Scope

This part of ISO/IEC 10118 specifies two hash-functions which make use of modular arithmetic. These hash-functions, which are believed to be collision-resistant, compress messages of arbitrary but limited length to a hash-code whose length is determined by the length of the prime number used in the reduction-function defined in 7.3. Thus, the hash-code is easily scaled to the input length of any mechanism (e.g., signature algorithm, identification scheme).

The hash-functions specified in this part of ISO/IEC 10118, known as MASH-1 and MASH-2 (Modular Arithmetic Secure Hash) are particularly suitable for environments in which implementations of modular arithmetic of sufficient length are already available. The two hash-functions differ only in the exponent used in the round-function.

2 Normative reference

The following standard contains provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 10118. At the time of publication, the edition indicated was valid. All standards are subject to revision and parties to agreements based on this part of ISO/IEC 10118 are encouraged to investigate the possibility of applying the most recent edition of the standard indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO/IEC 10118-1: 1994, *Information technology – Security techniques – Hash-functions – Part 1: General.*

3 Terms and definitions

For the purposes of this part of ISO/IEC 10118, the following definitions apply.

3.1 From ISO/IEC 10118-1

– collision-resistant hash-function
– data string (data)
– hash-code
– hash-function
– initializing value
– padding.

3.2 Unique to this part of ISO/IEC 10118

3.2.1
block
a string of bits of length L_ϕ, which shall be an integer multiple of 16 (see also clause 6.1)

EXAMPLE The length of the output H_j of the round-function.

3.2.2
half-block
a string of bits of length $L_\phi/2$

EXAMPLE Half the length of the block H_j.

3.2.3
hash-function identifier
a byte identifying a specific hash-function

3.2.4
modulus
a parameter which is a positive integer and a product of two distinct prime numbers

3.2.5
reduction-function
a function *RED* that is applied to the block H_q of length L_ϕ to generate the hash-code H of length L_p

3.2.6
round-function
a function $\phi(\cdot,\cdot)$ that transforms two binary strings of length L_ϕ to a binary string of length L_ϕ

NOTE It is used iteratively as part of a hash-function, where it combines an 'expanded' data block of length L_ϕ with the previous output of length L_ϕ.

3.3 Conventions

3.3.1 Bit ordering
Bit ordering in this part of ISO/IEC 10118 is as described in clause 3 of ISO/IEC 10118-1.

3.3.2 Converting a number to a string
During computation of the round-function, integers need to be converted to strings of L bits. Where this is required, the string of bits shall be made equal to the binary representation of the integer, with the left-most bit of the string corresponding to the most significant bit of the binary representation. If the resulting string of bits has less than L bits, then the string shall be left-padded with the appropriate number of zeros to make it of length L.

3.3.3 Converting a string to a number
During computation of the round-function, strings of bits need to be converted into integers. Where this is required, the integer shall be made equal to the number having binary representation equal to the binary string, where the left-most bit of the string is considered as the most significant bit of the binary representation.

3.4 Hash-function identifier
Identifiers are defined for each of the two MASH hash-functions specified in this standard. The hash-function identifiers for the hash-functions specified in clause 8.1 and 8.2 are equal to 41 and 42 (hexadecimal) respectively. The range of values from 43 to 4f (hexadecimal) are reserved for future use as hash-function identifiers by this part of ISO/IEC 10118.

4 Symbols and abbreviated terms

Throughout this part of ISO/IEC 10118, the following symbols and abbreviations apply.

4.1 From ISO/IEC 10118-1

D	Data
H	Hash-code
IV	Initializing value
$X \oplus Y$	Exclusive-or of strings of bits X and Y

4.2 Unique to this part of ISO/IEC 10118

B_j The jth block derived from the data string D after the padding, splitting, and expansion process.

D_j The jth half-block derived from the data string D after the padding and the splitting process. D_{q+1} through D_{q+8} are additional data blocks computed in the reduction-function.

e The exponent used in the round-function.

E A constant block equal to four ones (in the left-most position) followed by $L_\phi - 4$ zeros.

H_j The output of the round-function in the jth round. H_j has length L_ϕ.

L_D The length of the input string D in bits.

L_ϕ The length of the output H_j of the round-function ϕ. It shall be an integer multiple of 16.

L_N The length of the modulus N used in the round-function.

L_p The length of the prime number p used in the reduction-function.

mod If Z_1 is an integer and Z_2 is a positive integer, then Z_1 mod Z_2 denotes the unique integer Z_3 which satisfies

a) $0 \le Z_3 < Z_2$, and

b) $Z_1 - Z_3$ is an integer multiple of Z_2.

N A composite integer, used as the modulus in the round-function.

NOTE For the determination of the value of N, see clause 5.

p A prime number used in the reduction-function.

NOTE For the determination of the value of p, see clause 5.

q The number of half-blocks in the data string D after the padding and splitting processes, also the number of blocks after the padding, splitting, and expansion process.

RED The reduction-function, that is applied as the last operation of the hashing procedure to reduce the block H_q of length L_ϕ to the hash-code H of length L_p.

Y_j The jth sub-string of length $L_\phi/4$ bits used in the reduction-function.

ϕ A round-function. If X and Y denote strings of L_ϕ bits, then $\phi(X,Y)$ denotes a string of L_ϕ bits obtained by applying ϕ to X and Y.

$\vee$ The bit-wise inclusive OR operation on strings of bits, i.e., if X and Y are strings of the same length, then $X \vee Y$ denotes the string obtained as the bit-wise inclusive OR of X and Y.

~ A symbol denoting the truncate operation. If X is a bitstring then $X\sim j$ denotes the bitstring obtained by taking the right-most j bits of X.

:= A symbol denoting the 'set equal to' operation. It is used in the procedural specification of the round-function and of the reduction-function, where it indicates that the block on the left side of the symbol shall be changed to equal the value of the expression on the right side of the symbol.

$X \,||\, Y$ Concatenation of bit-strings X and Y in the indicated order.

5 Requirements

5.1 To employ either of the hash-functions specified in this part of ISO/IEC 10118, two integers shall be selected: the modulus N used in the round-function and the prime p used in the reduction-function.

Both integers, N and p, are determined by the security requirements of the application for which these hash-functions are used.

5.1.1 The modulus N shall be chosen so that factoring it is computationally infeasible.

5.1.2 The modulus N shall be generated in a way that the factors remain secret. This can be accomplished by a trusted third party or by a secure multiparty computation.

NOTE 1 Generating a modulus N with the property that its factors remain secret can be accomplished by using a trusted third party, trusted hardware, and/or a secure multiparty computation. Examples can be found in Boneh [1], Cocks [2], and Frankel [3].

NOTE 2 If the factors of the modulus are kept secret, and if the size of the prime p is sufficiently large, then the best known algorithm to find a collision takes approximately $2^{Lp/2}$ evaluations of the round-function, and the best known algorithm to find a (2nd) pre-image requires approximately 2^{Lp} evaluations of the round-function. Thus in these circumstances MASH-1 and MASH-2 are believed to be collision-free hash-functions.

5.1.3 The reduction-function prime p shall not be a factor of the modulus N of the round-function.

5.1.4 The length L_p of the prime p shall be at most half of the length of the modulus N, $L_p \leq L_\phi/2$.

5.1.5 The three high order bits of prime p shall consist of ones.

5.2 To employ one of the hash-functions, MASH-1 or MASH-2, the user has to select one of the two exponents e used in the round-function ϕ.

5.3 MASH-1 and MASH-2 can be applied to all data strings D containing at most $2^{L\phi/2}-1$ bits.

6 Variables and values needed for the hash operation

6.1 The length of the hash-code and of the modulus

The length of the modulus N and the length of the blocks H_j are related in the following manner:

$$L_\phi+1 \leq L_N \leq L_\phi+16$$

The length L_ϕ of the block H_q shall be an integer multiple of 16.

NOTE 1 If the length L_ϕ is chosen, then the length L_N is constrained by the inequalities above. If the length L_N is chosen, then the length L_ϕ will be the largest multiple of 16 less than L_N.

NOTE 2 Knowledge of N is sufficient to determine L_N, and consequently L_ϕ.

6.2 The modulus of the round-function

The modulus N used in the round-function is a composite integer generated as a product of two prime numbers of about the same length such that it is computationally infeasible to factorize N.

NOTE 1 In addition to the infeasibility of the factorization of the modulus, the security of the MASH hash-functions is based in part on the difficulty of extracting modular roots.

NOTE 2 The choice of a specific modulus N of appropriate length is outside the scope of this part of ISO/IEC 10118.

6.3 Initializing value

The initializing value *IV* is defined to be the string of L_ϕ binary zeros.

6.4 Exponent

For MASH-1 the value of the exponent *e* in the round-function equals 2. For MASH-2 the value of the exponent *e* in the round-function equals 257.

6.5 Reduction-function prime number

The reduction-function specified in 7.3 requires a prime *p.* The length L_p of prime *p* is determined by the security requirements, and by the input length of any mechanism using the hash-code. The length L_p shall be at most half of the length of the modulus *N,* $L_p \leq L_\phi/2$.

NOTE 1 The choice of a specific prime *p* of appropriate length is outside the scope of this part of ISO/IEC 10118.

NOTE 2 To avoid unbalanced results by the reduction modulo *p*, the prime number *p* shall be selected with the three high order bits equal to ones.

7 Hashing procedure

The hash-code *H* of the data string *D* shall be calculated using the following steps (see Figure 1):

7.1 Preparation of the data string

The data string *D* is transformed into a sequence of blocks for input to the round-function ϕ. The preparation consists of padding, splitting, and expanding as detailed in the following sub-clauses.

7.1.1 Padding the data string

If the length L_D of the data string *D* is not an integer multiple of $L_\phi/2$, *D* is right-padded with binary zeros according to padding method 1 described in ISO/IEC 10118-1, Appendix B.

7.1.2 Appending the length

An additional half-block is right-appended to the data string. It contains the binary representation of the length L_D of the original (unpadded) data string *D,* left-padded with binary zeros (see 3.3.2).

NOTE – If the data block *D* is empty, only the length block is input to the hashing procedure.

7.1.3 Splitting the data string

The resulting string is divided into a sequence of *q* half-blocks $D_1, D_2, \ldots, D_{q-1}, D_q$.

7.1.4 Expansion

Every half-block D_j, $j = 1,2,\ldots,q$, is now doubled in length from $L_\phi/2$ bits to L_ϕ bits. This is achieved by dividing D_j into half-bytes and preceding each half-byte of D_j with a half-byte consisting of four ones (1111), for $j=1,2,\ldots,q$. The result of this process when applied to half-block D_j is denoted as B_j, $j =1,2,\ldots,q$.

7.2 Application of the round-function

The round-function ϕ on which the MASH hash-functions are based takes as input two blocks, H_{j-1} and B_j, both of

length L_ϕ . It returns a block H_j of length L_ϕ. It is defined as follows:

$$\phi(B_j,H_{j-1}) = ((((H_{j-1} \oplus B_j) \vee E)^e \bmod N) \sim L_\phi) \oplus H_{j-1}$$

This round-function is applied sequentially to the data blocks B_j as follows:

$$H_0 := IV$$

$$H_j := \phi(B_j,H_{j-1}) \quad j = 1,2, \ldots ,q$$

7.3 The Reduction-function

The reduction-function *RED* consists of eight applications of the round-function with a data input derived from H_q. The hash-code H shall be calculated using the following four steps: splitting the block H_q, extending the data string, processing the additional data blocks, and reducing the block H_{q+8}.

7.3.1 Splitting the block H_q

The block H_q is divided into four strings of length $L_\phi/4$ bits, denoted with H_{q1}, H_{q2} , H_{q3}, H_{q4}, i.e.,

$$H_q := H_{q1} \,||\, H_{q2} \,||\, H_{q3} \,||\, H_{q4}$$

7.3.2 Extending the data string

Define $Y_0 := H_{q3}$, $Y_1 := H_{q1}$, $Y_2 := H_{q4}$, and $Y_3 := H_{q2}$. Then for $i = 4$ to 15 let

$$Y_i := Y_{i-1} \oplus Y_{i-4}$$

Define then eight additional data half-blocks D_{q+1} through D_{q+8} as follows: for $i = 1$ to 8 let

$$D_{q+i} := Y_{2i-2} \,||\, Y_{2i-1}$$

7.3.3 Processing the half-blocks

The eight half-blocks D_{q+1} through D_{q+8} are processed as in 7.1.4 and 7.2, but with IV equal to H_q, yielding the result H_{q+8}.

7.3.4 Reduction

The hash-code H of length L_p is computed as follows:

$$H := H_{q+8} \bmod p$$

8 Hash-functions

The two hash-functions specified in this part of ISO/IEC 10118 differ in the value of the exponent e used in the round-function ϕ.

8.1 MASH-1

For MASH-1, the round-function ϕ as specified in clause 7 becomes:

$$\phi(B_j,H_{j-1}) = ((((H_{j-1} \oplus B_j) \vee E)^2 \bmod N) \sim L_\phi) \oplus H_{j-1}$$

The ISO/IEC hash-function identifier for the hash-function MASH-1 is equal to 41 (hexadecimal).

8.2 MASH-2

For MASH-2, the round-function ϕ as specified in clause 7 becomes:

$$\phi(B_j, H_{j-1}) = ((((H_{j-1} \oplus B_j) \vee E)^{257} \bmod N) \sim L_\phi) \oplus H_{j-1}$$

The ISO/IEC hash-function identifier for the hash-function MASH-2 is equal to 42 (hexadecimal).

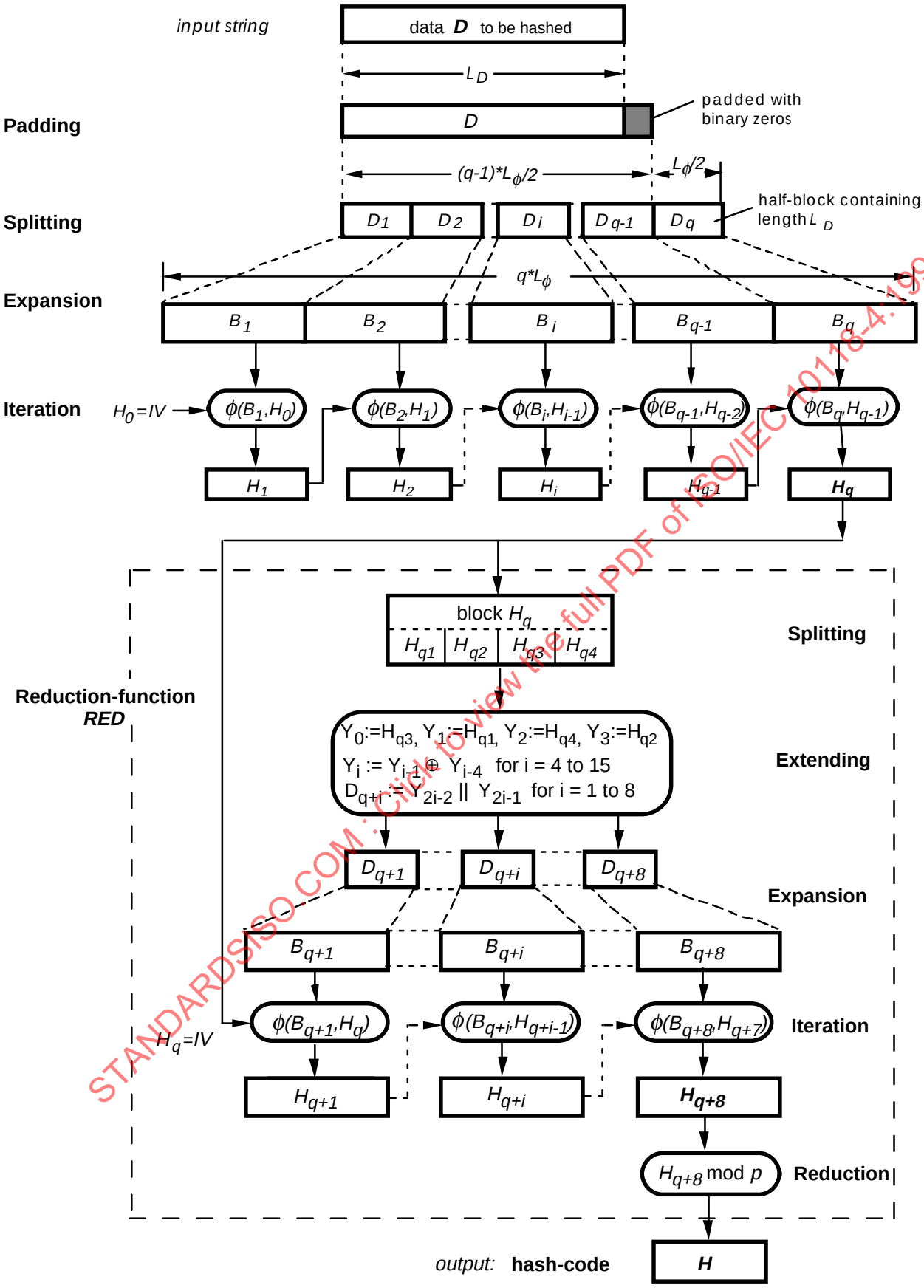

Figure 1 — The MASH hash-function

Annex A
(informative)

Examples

A.1 The hashing procedure

In practice, the data string may be supplied as a stream. Its overall length L_D may be unknown before the end of the stream is reached. As the stream passes through, the hash calculation is performed. For convenience of the description, a constant $k = L_\phi/2$ is introduced. Only the following four registers are required for the variables: *i, A, B, C.*

i bit counter for the length L_D .

A buffer to hold the hash-code.

B accumulator to hold the intermediate results.

C holding one half-block of data.

The hash-code *H* of the data *D* is calculated in the following steps:

A.1.1 Step 1 (Initialisation)

The buffer *A* is set to zero: $A := 0$ (initializing value $IV = H_0 = 0$)
The accumulator *B* is set to zero: $B := 0$
The bit counter is set to zero: $i := 0$ (counts the number of effective data bits hashed)

A.1.2 Step 2a (Reading a data block)

If at least k bits of the data are remaining, read *k* bits of the data *D* into *C.* If *k'* (with $0 < k' < k$) bits of the data are remaining, read *k'* bits of the data *D* into *C* and fill up *C* with *k-k'* binary zeros on the right. *k* (respectively *k'*) is added to *i.* If no more data bits are available $(k' = 0)$, go to Step 3a.

Step 2b (Expansion)
Each byte of *C* is split into halves and each half is preceded with four binary ones. The result is put into the accumulator *B.*

Step 2c (Combination with previous hash-value)
Calculate: $B := B \oplus A$ (combination)

Step 2d
$B := B \vee E$ (the four highest valued bits are set to 1)

Step 2e (Exponentiation)
Calculate: $B := B^e \bmod N$ where $e=2$ (squaring) with MASH-1, and $e=257$ with MASH-2.

Step 2f (Truncation)
(the leftmost bits of C which exceed the length L_ϕ are erased)

Step 2g (Feed-forward with previous hash-value)
Calculate: $A := B \oplus A$

Step 2h Return to Step 2a.

A.1.3 Step 3a (Reading the length counter)

$C := i$ (the content of the length counter is put into C).
The integer is converted into a string as specified in 3.3.2.

Step 3b (Hash with length counter)

Perform steps 2b, 2c, 2d, 2e, 2f and 2g, then go to Step 4.

A.1.4 Step 4 (Output the result)

The block H_q is contained in the buffer A as the rightmost L_ϕ bits.

A.1.5 The reduction procedure

The following registers are required for the variables: $A, B, C, C_0, C_1, C_2, C_3, i$.

A holding the block H_q to be reduced.

B buffer used in the round-function.

C accumulator of length $L_\phi/2$, to hold a half-block.

C_0, C_1, C_2, C_3 four buffers of length $L_\phi/4$.

i a counter.

The hash-code H is calculated in the following steps.

A.1.5.1 Step 4a (Initialisation)

The counter is set to 8: $i := 8$ (the number of half-blocks to be processed)

A.1.5.2 Step 4b (Splitting of the block H_q)

The block H_q contained in buffer A is divided into four parts $H_{q1}, H_{q2}, H_{q3}, H_{q4}$ each of length $L_\phi/4$ (see Figure 1) and saved:

$C_0 := H_{q3}$
$C_1 := H_{q1}$
$C_2 := H_{q4}$
$C_3 := H_{q2}$

A.1.5.3 Step 4c (Extension and iteration)

Calculate: $C := C_0 \| C_1$ (concatenate)
and perform Steps 2b through 2g (apply the round-function ϕ) (Round number q+1)
Calculate: i := i-1. (reduce counter)

Step 4d (Extension and iteration)

Calculate: $C := C_2 \| C_3$ (concatenate)
and perform Steps 2b through 2g (apply the round-function ϕ) (Round number q+2)
Calculate: i := i-1 (reduce counter).

A.1.5.4 Step 4e (Combination, extension and iteration)

Calculate: $C_0 := C_0 \oplus C_3$ (combination)
$C_1 := C_1 \oplus C_0$

Calculate: $C := C_0 \| C_1$ (concatenate)
and perform Steps 2b through 2g (apply the round-function ϕ) (Round number q+3, q+5, q+7)
Calculate: i := i-1 (reduce counter).

Step 4f (Combination of split blocks)
Calculate: $C_2 := C_2 \oplus C_1$ (combination)
$C_3 := C_3 \oplus C_2$
Calculate: $C := C_2 \| C_3$ (concatenate)
and perform Steps 2b through 2g (apply the round-function ϕ) (Round number q+4, q+6, q+8)
Calculate: i := i-1 If i does not equal zero, go to Step 4e.

A.1.6 Step 5 (Final reduction)

Calculate: $A := A \bmod p$

A.1.7 Step 6 (Output the result)

The hash-code H is contained in the buffer A as the rightmost L_p bits.

A.2 Example hash calculations

In the following example N is a composite number, 12 bits larger than L_ϕ. L_ϕ is a multiple of 16 strictly below L_N. The length of the modulus N is selected just to demonstrate the hashing procedure step by step, and does not necessarily meet the security requirements. To distinguish between decimal and hexadecimal numbers the subscript d or h is added respectively .

The following data are used:

Length of the modulus N,	L_N =	$10c_h$ =	268_d
Length of the block H_j,	L_ϕ =	100_h =	256_d
Length of the prime p,	L_p =	80_h =	128_d
Length of the hash-code H,	L_p =	80_h =	128_d

Modulus N = $\texttt{eec19b75218d08ad5516deceec1cf4af3824a95691ab41806865e09ffd51fbfa54d}_h$
E = $\texttt{f0000000000000000000000000000000000000000000000000000000000000000}_h$
Modulus p = $\texttt{ceced8f8b6b854189f8d6b39b75c1329}_h$

Data string D = `'Now is the time for all '`
Data string D encoded according to ISO 646 in hexadecimal representation:
$\texttt{4e6f77206973207468652074696d6520666f7220616c6c20}_h$
Length of the data string D, L_D = $\texttt{c0}_h$ = 192_d

NOTE — ISO 646 code is commonly called ASCII code because it is equivalent to ASCII 7-bit code. The 7-bit ASCII code is extended to eight bits by inserting a leading zero bit.

Every step is recorded to show the changing values of the variables i, A, B, C and C_0, C_1, C_2, C_3 during the course of the calculation. The first column refers to the steps described in Annex A.1.

A.2.1 Example hash calculation using MASH-1

Initialisation step: A := 0
B := 0
i := 0

First round:

(2a) C = $4e6f772069732074686520746d6520_h$ (the first 16_d bytes of the data string D)
i = 80_h

(2b) B = $f4fef6fff7f7f2f0f6f9f7f3f2f0f7f4f6f8f6f5f2f0f7f4f6f9f6fdf6f5f2f0_h$
(2c) B = $f4fef6fff7f7f2f0f6f9f7f3f2f0f7f4f6f8f6f5f2f0f7f4f6f9f6fdf6f5f2f0_h$
(2d) B = $f4fef6fff7f7f2f0f6f9f7f3f2f0f7f4f6f8f6f5f2f0f7f4f6f9f6fdf6f5f2f0_h$
(2e) B = $00000d87836ef6fbfe7f050ffa9c69d6c5ebab11b7f42df62f56904a924f169bf41b39f1_h$
(2f) A = $836ef6fbfe7f050ffa9c69d6c5ebab11b7f42df62f56904a924f169bf41b39f1_h$
(2g) A = $836ef6fbfe7f050ffa9c69d6c5ebab11b7f42df62f56904a924f169bf41b39f1_h$

Second round:

(2a) C = $666f7220616c6c20_h$ (the last 8_d bytes of the data string D)
(2a) C = $666f7220616c6c2000000000000000000_h$ (… padded with binary zeros)
i = $c0_h$

(2b) B = $f6f6f6fff7f2f2f0f6f1f6fcf6fcf2f0f0f0f0f0f0f0f0f0f0f0f0f0f0f0f0_h$
(2c) B = $75980004098df7ff0c6d9f2a331759e14704dd06dfa660ba62bfe66b04ebc901_h$
(2d) B = $f5980004098df7ff0c6d9f2a331759e14704dd06dfa660ba62bfe66b04ebc901_h$
(2e) B = $000004ebdaecc33c684979d2695461839008637c449e153c4d57ef76036361b335bc941f_h$
(2f) A = $daecc33c684979d2695461839008637c449e153c4d57ef76036361b335bc941f_h$
(2g) A = $598235c796367cdd93c8085555e3c86df36a38ca62017f3c912c7728c1a7adee_h$

Round no. q (length counter)

(3a) C = $00000000000000000000000000000c0_h$ (the last 8_d bytes of the data string D)
(2b) B = $f0f0f0f0f0f0f0f0f0f0f0f0f0f0f0f0f0f0f0f0f0f0f0f0f0f0fcf0_h$
(2c) B = $a972c53766c68c2d6338f8a5a513389d039ac83a92f18fcc61dc87d83157511e_h$
(2d) B = $f972c53766c68c2d6338f8a5a513389d039ac83a92f18fcc61dc87d83157511e_h$
(2e) B = $0000025b1e0a335d8cd76a299d717e7a2bf111d5ea3b3ff6cb56afae0b84cf3ee98c818a_h$
(2f) A = $1e0a335d8cd76a299d717e7a2bf111d5ea3b3ff6cb56afae0b84cf3ee98c818a_h$
(2g) A = $4788069a1ae116f40eb9762f7e12d9b81951073ca957d0929aa8b816282b2c64_h$
H_q = $4788069a1ae116f40eb9762f7e12d9b81951073ca957d0929aa8b816282b2c64_h$

Reduction procedure RED

Splitting

(4a) H_{q1} = $4788069a1ae116f4_h$
H_{q2} = $0eb9762f7e12d9b8_h$
H_{q3} = $1951073ca957d092_h$
H_{q4} = $9aa8b816282b2c64_h$

(4b) C_0:= H_{q3} = $1951073ca957d092_h$
C_1:= H_{q1} = $4788069a1ae116f4_h$
C_2:= H_{q4} = $9aa8b816282b2c64_h$
C_3:= H_{q2} = $0eb9762f7e12d9b8_h$

Extension

(4c) C = $C_0 || C_1$ = $1951073ca957d0924788069a1ae116f4_h$

Round no. q+1

(2b) B = $f1f9f5f1f0f7f3fcfaf9f5f7fdf0f9f2f4f7f8f8f0f6f9faf1fafef1f1f6fff4_h$
(2c) B = $b671f36bea16e508f44083d883e2204aeda6ffc459a129686b5246e7d9ddd390_h$
(2d) B = $f671f36bea16e508f44083d883e2204aeda6ffc459a129686b5246e7d9ddd390_h$
(2e) B = $000004beda0f78385d1a959a02687ab92ca6051b91e092f60b72105a2c3fa4059d239311_h$
(2f) A = $da0f78385d1a959a02687ab92ca6051b91e092f60b72105a2c3fa4059d239311_h$
(2g) A = $9d877ea247fb836e0cd10c9652b4dca388b195caa225c0c8b6971c13b508bf75_h$

Extension

(4d) C = $C_2 || C_3$ = $9aa8b816282b2c640eb9762f7e12d9b8_h$

Round no. q+2

(2b) B = f9fafaf8fbf8f1f6f2f8f2fbf2fcf6f4f0fefbf9f7f6f2fff7fef1f2fdf9fbf8$_h$
(2c) B = 647d845abc037298fe29fe6da0482a57784f6e3355d332374169ede148f1448d$_h$
(2d) B = f47d845abc037298fe29fe6da0482a57784f6e3355d332374169ede148f1448d$_h$
(2e) B = 00000741c0abe2c27ac0bf2384d377ab7b07f3374c402f87ec26baafabd6f2e360f7000b$_h$
(2f) A = c0abe2c27ac0bf2384d377ab7b07f3374c402f87ec26baafabd6f2e360f7000b$_h$
(2g) A = 5d2c9c603d3b3c4d88027b3d29b32f94c4f1ba4d4e037a671d41eef0d5ffbf7e$_h$

Round no. q+3

(4e) $C_0 = C_0 \oplus C_3$ = 17e87113d745092a$_h$
(4e) $C_1 = C_1 \oplus C_0$ = 50607789cda41fde$_h$
(4e) $C = C_0 \| C_1$ = 17e87113d745092a50607789cda41fde$_h$

(2b) B = f1f7fef8f7f1f1f3fdf7f4f5f0f9f2faf5f0f6f0f7f7f8f9fcfdfaf4f1fffdfe$_h$
(2c) B = acdb6298cacacdbe75f58fc8d94add6e31014cbdb9f4829ee1bc140424004280$_h$
(2d) B = fcdb6298cacacdbe75f58fc8d94add6e31014cbdb9f4829ee1bc140424004280$_h$
(2e) B = 0000085f0e394019c43439b100f3865b24c54196d0a6612bc8298ee325583501f5fe139b$_h$
(2f) A = 0e394019c43439b100f3865b24c54196d0a6612bc8298ee325583501f5fe139b$_h$
(2g) A = 5315dc79f90f05fc88f1fd660d766e021457db66862af4843819dbf12001ace5$_h$

Round no. q+4

(4f) $C_2 = C_2 \oplus C_1$ = cac8cf9fe58f33ba$_h$
(4f) $C_3 = C_3 \oplus C_2$ = c471b9b09b9dea02$_h$
(4f) $C = C_2 \| C_3$ = cac8cf9fe58f33bac471b9b09b9dea02$_h$

(2b) B = fcfafcf8fcfff9fffef5f8fff3f3fbfafcf4f7f1fbf9fbf0f9fbf9fdfefaf0f2$_h$
(2c) B = afef208105f0fc0376040599fe8595f8e8a32c977dd30f74c1e2220cdefb5c17$_h$
(2d) B = ffef208105f0fc0376040599fe8595f8e8a32c977dd30f74c1e2220cdefb5c17$_h$
(2e) B = 000009a0a067af1cb50e126496355f4f8033d62330dd0c151703d398534f4a1e89e28d4a$_h$
(2f) A = a067af1cb50e126496355f4f8033d62330dd0c151703d398534f4a1e89e28d4a$_h$
(2g) A = f37273654c0117981ec4a2298d45b821248ad7739129271c6b5691efa9e321af$_h$

Round no. q+5

(4e) $C_0 = C_0 \oplus C_3$ = d399c8a34cd8e328$_h$
(4e) $C_1 = C_1 \oplus C_0$ = 83f9bf2a817cfcf6$_h$
(4e) $C = C_0 \| C_1$ = d399c8a34cd8e32883f9bf2a817cfcf6$_h$

(2b) B = fdf3f9f9fcf8faf3f4fcfdf8fef3f2f8f8f3fff9fbfff2faf8f1f7fcfffcfff6$_h$
(2c) B = 0e818a9cb0f9ed6bea385fd173b64ad9dc79288a6ad6d5e693a76613561fde59$_h$
(2d) B = fe818a9cb0f9ed6bea385fd173b64ad9dc79288a6ad6d5e693a76613561fde59$_h$
(2e) B = 0000013eb89f8f9208f8dd6af54caf0c5783bcb901c9ed5dc70ef255eb64da4fbafe19a2$_h$
(2f) A = b89f8f9208f8dd6af54caf0c5783bcb901c9ed5dc70ef255eb64da4fbafe19a2$_h$
(2g) A = 4bedfcf744f9caf2eb880d25dac6049825433a2e5627d54980324ba0131d380d$_h$

Round no. q+6

(4f) $C_2 = C_2 \oplus C_1$ = 493170b564f3cf4c$_h$
(4f) $C_3 = C_3 \oplus C_2$ = 8d40c905ff6e254e$_h$
(4f) $C = C_2 \| C_3$ = 493170b564f3cf4c8d40c905ff6e254e$_h$

(2b) B = f4f9f3f1f7f0fbf5f6f4fff3fcfff4fcf8fdf4f0fcf9f0f5fffff6fef2f5f4fe$_h$
(2c) B = bf140f06b30931071d7cf2d62639f064ddbecedeaade25bc7fcdbd5ee1e8ccf3$_h$
(2d) B = ff140f06b30931071d7cf2d62639f064ddbecedeaade25bc7fcdbd5ee1e8ccf3$_h$
(2e) B = 000002cd631f7f2804f90933a27240f80ea77e9668d319c6157c23f7b11684f2d7080ace$_h$
(2f) A = 631f7f2804f90933a27240f80ea77e9668d319c6157c23f7b11684f2d7080ace$_h$
(2g) A = 28f283df4000c3c149fa4dddd4617a0e4d9023e8435bf6be3124cf52c41532c3$_h$

Round no. q+7

(4e) $C_0 = C_0 \oplus C_3 = 5ed901a6b3b6c666_h$
(4e) $C_1 = C_1 \oplus C_0 = dd20be8c32ca3a90_h$
(4e) $C = C_0 \| C_1 = 5ed901a6b3b6c666dd20be8c32ca3a90_h$

(2b) B = f5fefdf9f0f1faf6fbf3fbf6fcf6f6f6fdfdf2f0fbfef8fcf3f2fcfaf3faf9f0$_h$
(2c) B = dd0c7e26b0f13937b209b62b28978cf8b06dd118b8a50e42c2d633a837efcb33$_h$
(2d) B = fd0c7e26b0f13937b209b62b28978cf8b06dd118b8a50e42c2d633a837efcb33$_h$
(2e) B = 00000b5c780401af81fcafb61d0af544bb9533b074260c5b9536bc8cc8d89c021c8cc326$_h$
(2f) A = 780401af81fcafb61d0af544bb9533b074260c5b9536bc8cc8d89c021c8cc326$_h$
(2g) A = 50f68270c1fc6c7754f0b8996ff449be39b62fb3d66d4a32f9fc5350d899f1e5$_h$

Round no. q+8

(4f) $C_2 = C_2 \oplus C_1 = 9411ce395639f5dc_h$
(4f) $C_3 = C_3 \oplus C_2 = 1951073ca957d092_h$
(4f) $C = C_2 \| C_3 = 9411ce395639f5dc1951073ca957d092_h$

(2b) B = f9f4f1f1fcfef3f9f5f6f3f9fff5fdfcf1f9f5f1f0f7f3fcfaf9f5f7fdf0f9f2$_h$
(2c) B = a90273813d029f8ea1064b609001b442c84fda42269ab9ce0305a6a725690817$_h$
(2d) B = f90273813d029f8ea1064b609001b442c84fda42269ab9ce0305a6a725690817$_h$
(2e) B = 00000d67969376fb27ef9563bd5bd2b08156dd01904bc7ac090552607e69252ba9265ed$_h$
(2f) A = 7969376fb27ef9563bd5bd2b08156dd01904bc7ac090552607e69252ba9265ed$_h$
(2g) A = 299fb51f738295216f2505b267e1246e20b293c916fd1f14fe1ac102620b9408$_h$

H_{q+8} = 299fb51f738295216f2505b267e1246e20b293c916fd1f14fe1ac102620b9408$_h$

Reduction mod *p*

(5) A = 497a26f552e76e98ea6eaae0e54b6d7d$_h$
(6) **H** = 497a26f552e76e98ea6eaae0e54b6d7d$_h$ **Hash-code calculated by MASH-1**

A.2.2 Example hash calculation using MASH-2

Initialisation step: A := 0
B := 0
i := 0

First round:

(2a) C = 4e6f772069732074686520746d6520$_h$ (the first 16_d bytes of the data string D)
i = 80$_h$

(2b) B = f4fef6fff7f7f2f0f6f9f7f3f2f0f7f4f6f8f6f5f2f0f7f4f6f9f6fdf6f5f2f0$_h$
(2c) B = f4fef6fff7f7f2f0f6f9f7f3f2f0f7f4f6f8f6f5f2f0f7f4f6f9f6fdf6f5f2f0$_h$
(2d) B = f4fef6fff7f7f2f0f6f9f7f3f2f0f7f4f6f8f6f5f2f0f7f4f6f9f6fdf6f5f2f0$_h$
(2e) B = 0000083871d3c375922382b7996401171c44350ffc63f85a11aebbf07e182108ba9d1f3f$_h$
(2f) A = 71d3c375922382b7996401171c44350ffc63f85a11aebbf07e182108ba9d1f3f$_h$
(2g) A = 71d3c375922382b7996401171c44350ffc63f85a11aebbf07e182108ba9d1f3f$_h$

Second round:

(2a) C = 666f7220616c6c20$_h$ (the last 8_d bytes of the data string D)
(2a) C = 666f7220616c6c2000000000000000000$_h$ (… padded with binary zeros)
i = c0$_h$

(2b) B = f6f6f6fff7f2f2f0f6f1f6fcf6fcf2f0f0f0f0f0f0f0f0f0f0f0f0f0f0f0f0$_h$
(2c) B = 8725358a65d170476f95f7ebeab8c7ff0c9308aae15e4b008ee8d1f84a6defcf$_h$
(2d) B = f725358a65d170476f95f7ebeab8c7ff0c9308aae15e4b008ee8d1f84a6defcf$_h$
(2e) B = 0000064adfe7bc5d78f2c00b3c11cc020a4c7e798de85ff6bc98c78061e803f40e5b2b7a$_h$

(2f) A = dfe7bc5d78f2c00b3c11cc020a4c7e798de85ff6bc98c78061e803f40e5b2b7a$_h$
(2g) A = ae347f28ead142bca575cd1516084b76718ba7acad367c701ff022fcb4c63445$_h$

Round no. q (length counter)

(3a) C = 00000000000000000000000000000c0$_h$ (the last 8$_d$ bytes of the data string D)
(2b) B = f0f0f0f0f0f0f0f0f0f0f0f0f0f0f0f0f0f0f0f0f0f0f0f0f0f0f0f0fcf0$_h$
(2c) B = 5ec48fd81a21b24c55853de5e6f8bb86817b575c5dc68c80ef00d20c4436c8b5$_h$
(2d) B = fec48fd81a21b24c55853de5e6f8bb86817b575c5dc68c80ef00d20c4436c8b5$_h$
(2e) B = 00000441d23756b05a6bd8830708cf278486582fdc49fb478c4966b20b042d12dce7fb69$_h$
(2f) A = d23756b05a6bd8830708cf278486582fdc49fb478c4966b20b042d12dce7fb69$_h$
(2g) A = 7c032998b0ba9a3fa27d0232928e1359adc25ceb217f1ac214f40fee6821cf2c$_h$

H_q = 7c032998b0ba9a3fa27d0232928e1359adc25ceb217f1ac214f40fee6821cf2c$_h$

Reduction procedure RED

Splitting

(4a) H_{q1} = 7c032998b0ba9a3f$_h$
H_{q2} = a27d0232928e1359$_h$
H_{q3} = adc25ceb217f1ac2$_h$
H_{q4} = 14f40fee6821cf2c$_h$

(4b) C_0:= H_{q3} = adc25ceb217f1ac2$_h$
C_1:= H_{q1} = 7c032998b0ba9a3f$_h$
C_2:= H_{q4} = 14f40fee6821cf2c$_h$
C_3:= H_{q2} = a27d0232928e1359$_h$

Extension

(4c) C = $C_0 \| C_1$ = adc25ceb217f1ac27c032998b0ba9a3f$_h$

Round no. q+1

(2b) B = fafdfcf2f5fcfefbf2f1f7fff1fafcf2f7fcf0f3f2f9f9f8fbf0fbfaf9faf3ff$_h$
(2c) B = 86fed56a454664c4508cf5cd6374efab5a3eac18d386e33aef04f41491db3cd3$_h$
(2d) B = f6fed56a454664c4508cf5cd6374efab5a3eac18d386e33aef04f41491db3cd3$_h$
(2e) B = 000006324c63d04f673f195bd0d7d79212c0b1a6d1037d07c20d31f946c07a954b480ecd$_h$
(2f) A = 4c63d04f673f195bd0d7d79212c0b1a6d1037d07c20d31f946c07a954b480ecd$_h$
(2g) A = 3060f9d7d785836472aad5a0804ea2ff7cc121ece3722b3b5234757b2369c1e1$_h$

Extension

(4d) C = $C_2 \| C_3$ = 14f40fee6821cf2ca27d0232928e1359$_h$

Round no. q+2

(2b) B = f1f4fff4f0fffefef6f8f2f1fcfff2fcfaf2f7fdf0f2f3f2f9f2f8fef1f3f5f9$_h$
(2c) B = c1940623277a7d9a845227517cb150038633d6111380d8c9abc68d85d29a3418$_h$
(2d) B = f1940623277a7d9a845227517cb150038633d6111380d8c9abc68d85d29a3418$_h$
(2e) B = 00000c2b4825db0a9091297c530abd55dfff5b93e28f2c0c65cc50299a52bf7602d872be$_h$
(2f) A = 4825db0a9091297c530abd55dfff5b93e28f2c0c65cc50299a52bf7602d872be$_h$
(2g) A = 784522dd4714aa1821a068f55fb1f96c9e4e0de086be7b12c866ca0d21b1b35f$_h$

Round no. q+3

(4e) C_0 = $C_0 \oplus C_3$ = 0fbf5ed9b3f1099b$_h$
(4e) C_1 = $C_1 \oplus C_0$ = 73bc7741034b93a4$_h$
(4e) C = $C_0 \| C_1$ = 0fbf5ed9b3f1099b73bc7741034b93a4$_h$

(2b) B = f0fffbfff5fefdf9fbf3fff1f0f9f9fbf7f3fbfcf7f7f4f1f0f3f4fbf9f3faf4$_h$
(2c) B = 88bad922b2ea57e1da539704af48009769bdf61c71498fe338953ef6d84249ab$_h$
(2d) B = f8bad922b2ea57e1da539704af48009769bdf61c71498fe338953ef6d84249ab$_h$
(2e) B = 000002595aaa22afa6266de454fa9395ae40ff1dc667735f1e6e5f603e5c1ea79c94d8e9$_h$

(2f) A = $5aaa22afa6266de454fa9395ae40ff1dc667735f1e6e5f603e5c1ea79c94d8e9_h$
(2g) A = $22ef0072e132c7fc755afb60f1f1067158297ebf98d02472f63ad4aabd256bb6_h$

Round no. q+4

(4f) $C_2 = C_2 \oplus C_1$ = $674878af6b6a5c88_h$
(4f) $C_3 = C_3 \oplus C_2$ = $c5357a9df9e44fd1_h$
(4f) $C = C_2 \| C_3$ = $674878af6b6a5c88c5357a9df9e44fd1_h$

(2b) B = $f6f7f4f8f7f8fafff6fbf6faf5fcf8f8fcf5f3f5f7faf9fdfff9fef4f4fffdf1_h$
(2c) B = $d418f48a16ca3d0383a10d9a040dfe89a4dc8d4a6f2add8f09c32a5e49da9647_h$
(2d) B = $f418f48a16ca3d0383a10d9a040dfe89a4dc8d4a6f2add8f09c32a5e49da9647_h$
(2e) B = $000006444434f5a752c3635d88ea144e3d28e4613eb15035e1b9c47c8ec14e50b95623eb_h$
(2f) A = $4434f5a752c3635d88ea144e3d28e4613eb15035e1b9c47c8ec14e50b95623eb_h$
(2g) A = $66dbf5d5b3f1a4a1fdb0ef2eccd9e21066982e8a7969e00e78fb9afa0473485d_h$

Round no. q+5

(4e) $C_0 = C_0 \oplus C_3$ = $ca8a24444a15464a_h$
(4e) $C_1 = C_1 \oplus C_0$ = $b9365305495ed5ee_h$
(4e) $C = C_0 \| C_1$ = $ca8a24444a15464ab9365305495ed5ee_h$

(2b) B = $fcfaf8faf2f4f4f4f4faf1f5f4f6f4fafbf9f3f6f5f3f0f5f4f9f5fefdf5fefe_h$
(2c) B = $9a210d2f41055055094a1edb382f16ea9d61dd7c8c9a10fb8c026f04f986b6a3_h$
(2d) B = $fa210d2f41055055094a1edb382f16ea9d61dd7c8c9a10fb8c026f04f986b6a3_h$
(2e) B = $00000b472e2404e982562be4723c0f7818292619525274b24a8b64f89ea41d1d89a764a0_h$
(2f) A = $2e2404e982562be4723c0f7818292619525274b24a8b64f89ea41d1d89a764a0_h$
(2g) A = $48fff13c31a78f458f8ce056d4f0c40934ca5a3833e284f6e65f87e78dd42cfd_h$

Round no. q+6

(4f) $C_2 = C_2 \oplus C_1$ = $de7e2baa22348966_h$
(4f) $C_3 = C_3 \oplus C_2$ = $1b4b5137dbd0c6b7_h$
(4f) $C = C_2 \| C_3$ = $de7e2baa223489661b4b5137dbd0c6b7_h$

(2b) B = $fdfef7fef2fbfafaf2f2f3f4f8f9f6f6f1fbf4fbf5f1f3f7fdfbfdf0fcf6fbf7_h$
(2c) B = $b50106c2c35c75bf7d7e13a22c0932ffc531aec3c61377011ba47a177122d70a_h$
(2d) B = $f50106c2c35c75bf7d7e13a22c0932ffc531aec3c61377011ba47a177122d70a_h$
(2e) B = $000002cac6f3b0a9d6b41c106b1ef6465385af56258886f78d80f016914d005f66f51db2_h$
(2f) A = $c6f3b0a9d6b41c106b1ef6465385af56258886f78d80f016914d005f66f51db2_h$
(2g) A = $8e0c4195e7139355e492161087756b5f1142dccfbe6274e0771287b8eb21314f_h$

Round no. q+7

(4e) $C_0 = C_0 \oplus C_3$ = $d1c1757391c580fd_h$
(4e) $C_1 = C_1 \oplus C_0$ = $68f72676d89b5513_h$
(4e) $C = C_0 \| C_1$ = $d1c1757391c580fd68f72676d89b5513_h$

(2b) B = $fdf1fcf1f7f5f7f3f9f1fcf5f8f0fffdf6f8fff7f2f6f7f6fdf8f9fbf5f5f1f3_h$
(2c) B = $73fdbd6410e664a61d63eae57f8594a2e7ba23384c9483168aea7e431ed4c0bc_h$
(2d) B = $f3fdbd6410e664a61d63eae57f8594a2e7ba23384c9483168aea7e431ed4c0bc_h$
(2e) B = $0000013ef76551c6e3b17dfb5ecb837566eec9c7fb39f16c39351b9fd4bdacb352eaf01d_h$
(2f) A = $f76551c6e3b17dfb5ecb837566eec9c7fb39f16c39351b9fd4bdacb352eaf01d_h$
(2g) A = $7969105304a2eeaeba599565e19ba298ea7b2da387576f7fa3af2b0bb9cbc152_h$

Round no. q+8

(4f) $C_2 = C_2 \oplus C_1$ = $b6890ddcfaafdc75_h$
(4f) $C_3 = C_3 \oplus C_2$ = $adc25ceb217f1ac2_h$
(4f) $C = C_2 \| C_3$ = $b6890ddcfaafdc75adc25ceb217f1ac2_h$

(2b) B = fbf6f8f9f0fdfdfcfffafafffdfcf7f5fafdfcf2f5fcfefbf2f1f7fff1fafcf2$_h$
(2c) B = 829fe8aaf45f135245a36f9a1c67556d1086d15172ab9184515edcf448313da0$_h$
(2d) B = f29fe8aaf45f135245a36f9a1c67556d1086d15172ab9184515edcf448313da0$_h$
(2e) B = 00000a8bf084975e7ac8e096b464172ed9a5cbffeb32c5512faec0b3702f427608497abd$_h$
(2f) A = f084975e7ac8e096b464172ed9a5cbffeb32c5512faec0b3702f427608497abd$_h$
(2g) A = 89ed870d7e6a0e380e3d824b383e69670149e8f2a8f9afccd380697db182bbef$_h$

H_{q+8} = 89ed870d7e6a0e380e3d824b383e69670149e8f2a8f9afccd380697db182bbef$_h$

Reduction mod *p*

(5) A = 8ad87c2de674c2e82de5769806e1bb28$_h$

(6) **H =** 8ad87c2de674c2e82de5769806e1bb28$_h$ **Hash-code calculated by MASH-2**

A.3 Sample test messages and their hash-codes

In the calculation of the hash-code for the set of test messages (A.3.1 to A.3.9) the following integers are used:

1) the composite modulus

N = 8b251fa16f8b7a3c8a1ec50da421de6bfdfc4db6cf4452d0df98ad327b9f6feca66422e1434938ffe3576
d7b7a76b8c94a90dc9d0cc576bd6f9a128f4af1d907e3b4dbb67f52683e7992a4cf031f885533d21d07c7
e14811adbd600fb78d62a789f$_h$

2) the prime number

p = cde7e6f6e432331d896a7b02d031a09d7b2c77a1$_h$

The length of the modulus N used in the round-function is: $L_N = 780_d$ = 30C$_h$

The length of the output of the round-function is: $L_\phi = 768_d$ = 300$_h$

The length of the prime p used in the reduction-function is: $L_p = 160_d$ = A0$_h$

The length of the hash-code is: $L_p = 160_d$ = A0$_h$

The contents of the fields:

length of text contains the number of bits of the message text. It is given in decimal first (with subscript d) followed by the equivalent hexadecimal representation (with subscript h).

text contains the message to be hashed delimited by single quotation marks.

text in ASCII contains the message in ASCII code. The blanks are inserted for readability.

H_q contains the output of the round-function in hexadecimal representation after q applications.

H_{q+8} contains the output of the round-function in hexadecimal representation after $q+8$ applications.

H contains the hash-code in hexadecimal representation after the modulo p reduction.

A.3.1 Example 1

length of text $0_d = 0_h$
text ''
text in ASCII (empty string)

MASH-1 H_q a69e5191337162387c3a2701bc8a49ad385d648194fa1e188ca2d4f5f149fdf89cea0e9
c1324fd96270a1fee9a77e00925a014432896dd6a54232947f222b28b786259c45de549
6b166e4955cd0c84f2539c49fb61f6fc66763915dd006e8b8c

H_{q+8} c42af288cfd830875401c6e622a417058076b4869183cf5d79721a33c5d297937f90506
891cbd0e0d307d8515ec3dd5d64bd6257e90ba06acf53c14c47cb007e54ae429dfa0329
fff22cd1fdb2702598258d081d4f0d25a612084f906c2f5627

H b7231fa49ce0249adef0ce0a1429796e8f19ad8b

MASH-2	H_q	499bcae76a9b15f8061fc0fc2da49d7d8f1ca7436a70c36d564ac35875cdcf310f3a8e8 791e358774669341bd07f7917cd35928f9ee60a88f7c53cecb2d5e82f9778049b24243b 896a3e66448801ab6f0b975018a7732a87baa3c1ac07e95eaf
	H_{q+8}	416000041591700f319a8e2bbc2bab56e88cd39dd24194da16e81b5b6db15d9ca22849f 7ad7ae6d83987d92c1cdfdc65775c610740b935d1258f568af3038a79d8a7b97000aec7 63b960c27a63245d91d3c58f21bd386fb9b8600f4765d94efe
	H	bcce0b6f5646e7eb0cfb1ed6dbe787016ce4b18e

A.3.2 Example 2

length of text	$8_d = 8_h$
text	'a'
text in ASCII	61

MASH-1	H_q	843d8fbd095cb55514b0017843541ea3003c03b6b9753501ad28a28f8e3d5a125461f8a 6db9fc6bdb3475916915bb50dd71a833647710db0b71c4696570b37de678d185c99e285 e496be7c5415130c2ee51108e4fe38f0e80ec4cfed7bb3c9f2
	H_{q+8}	b7cd378165eb4ed4ca34f1372643d57d395810e696959eb44f3514be6413c9131817ac8 27a8f501961ac3cb0a51e2ecb32c990443f2eac32890b81ecf5874eabccc97ef292870a b945c00eb0555a4ede7ab9e23cca450bfdaff46921e2a40d25
	H	c7c5e8262a0067a3f0479fdc5c814e90e636610b
MASH-2	H_q	4851adcd1809c0b66018a4f62a0008ea08e25fa5a6a8f970d0f09e3be81276301d68e5d a78314ecae818a32589132a22caacdbe5f4e06afe7273632ef218255f4bd3c58d3fc3fd 4357411c21047e77017c0358401a013f251638f3f4764ef5a4
	H_{q+8}	597292939e469da0fa542227c5e8c6a4eade4497433cc3ca8db32e473c23287af7dfcfd eb7e1fe83723697772acc19ef5206d681bdab2b6fa3253fec58980284dda2fc077fbd04 92e125c204af1f4625f439968115619e517ab05b74ee60eb0c
	H	c8ae72da006544d524a6ca09c9d7702628bfb683

A.3.3 Example 3

length of text	$24_d = 18_h$
text	'abc'
text in ASCII	61 62 63

MASH-1	H_q	329f3abba8d9ffde27478023bcfba35bf1abaa31fdf7e32a9a314fb43769a3809bb654a 94838382b544702263e6ff3bd8a21fc38f5ff5f06859d11aab1d49af96912fed50ed775 5ed3d1da0776ede4985ae1fd56dbe8cff52e37eb7b724ec694
	H_{q+8}	5910c9e2d3fa19bda8c4143aaeeeac65f7acdf8d067b43891ee2d4626c0fc570188083e e86d8e35fe89fded34e2b3ee75dfc5229bcce85b250b312339355b74c62e9429021b5a9 65e028ea3f5477687ae5d4858b46a480976b7e741dbba0c8d0
	H	9e76dd088a4e88ea72bb9ce3727f34bd27c75912
MASH-2	H_q	3bbd7e23260830c085a6282c5ef2f2d35d7e78a74d0f3de4964bbea9a74780b43662d9a 666daba357abef2c3186129e3202f80a015de4ddc6811db472cf36420e0ccb17a9a937e 1a89647eb7f84105c023617b9144ba2faf8113070acb36edf6
	H_{q+8}	344d4e0f7627fbb5affdaa62610d096ce6951f9a17f191e56a25ba5d3dcc4b08a361ffe 917b8eef53d0e0a39193fd43274b34f931b96e469518d5e7d60b1d794fe78c1740b0c42 c7273e2b7b1fa7d8092d7a0c590957eb76caf7868339e6aa6b
	H	18086f7c5c96e1bdc659c1c7b5c957f3dfbae7c8

A.3.4 Example 4

length of text	$112_d = 70_h$
text	'message digest'
text in ASCII	6D 65 73 73 61 67 65 20 64 69 67 65 73 74