FFT–Hash II, Efficient Cryptographic Hashing

C.P. Schnorr

Fachbereich Mathematik / Informatik, Universität Frankfurt

Abstract. We propose an efficient algorithm that hashes messages of arbitrary bit length into an 128 bit hash value. The algorithm is designed to make the production of a pair of colliding messages computationally infeasible. The algorithm performs a discrete Fourier transform and a polynomial recursion over a finite field. Each hash value in $\{0,1\}^{128}$ occurs with frequency at most 2^{-120}. This hash function is an improved variant of the algorithm FFT–hash I presented in the rump session of CRYPTO'91.

1 The hash algorithm

Overview. We present a novel design for a cryptographic hash algorithm. It may serve as an alternative to the MD4 / MD5 algorithm of Rivest. These novel hash algorithms are not based on any encryption scheme. We need to have more cryptographic hash algorithms since improved methods of cryptoanalysis may exhibit more weaknesses in the proposed hash functions. Our design goal is to make it impossible to produce a collision using less than 2^{64} operations. Our algorithm can easily be implemented in software using addition and multiplication either modulo 2^{16} or modulo 2^8 . It uses the discrete Fourier transform in order to diffuse information in an ideal way. We also use a polynomial transformation of high degree over a finitefield. Such transformations generate local randomness, see Niederreiter, Schnorr (1992), this proceedings.

Padding the message. Let the message be given as a bit string $m_1 m_2 \dots m_t$ of t bits. The message must be padded so that its length in bits becomes a multiple of 128. We recommend to append to the message a single "1" bit followed by a suitable number of "0" bits followed by the binary representation of t. Let the padded message $M_1 || M_2 \dots || M_n$ consist of n blocks $M_1, \dots, M_n$ that each is 128 bits long.

The initial value. H_0 is given in hexadecimal as

$$H_0 = 0123\ 4567\ 89ab\ cdef\ fedc\ ba98\ 7654\ 3210 \quad \in \{0,1\}^{128} .$$

Algorithm for the hash function **h**

```
INPUT  $M = M_1||M_2 \cdots||M_n \in \{0,1\}^{n\cdot 128}$        (a padded message)
          $H_i := g(H_{i-1}, M_i)$    for $i = 1, \ldots, n$
OUTPUT $h(M) := H_n$
```

Requirements for function $g : \{0,1\}^{256} \to \{0,1\}^{128}$.

We wish to make the function h *collision-free*. This means that it is infeasible to find distinct messages $M_1 \,||\, M_2 \cdots ||\, M_n$, $M'_1 \,||\, M'_2 \cdots ||\, M'_{n'}$ such that $H_n = H'_{n'}$. Specifically the construction of two colliding messages should require about 2^{64} steps which is the time bound for the birthday attack.
To achieve this goal the following problems must be infeasible to solve for given $H, H' \in \{0,1\}^{128}$.

Problem 1 Find message blocks $M, M' \in \{0,1\}^{128}$ such that $g(H,M) = g(H',M')$.

Problem 2 Find $M \in \{0,1\}^{128}$ such that $g(H',M) = H$.

Description of the function $\mathbf{g} : \{0,1\}^{\mathbf{256}} \longrightarrow \{0,1\}^{\mathbf{128}}$

Let p be the prime $p = 65537 = 2^{16}+1$. We represent elements in $\mathbb{Z}_p = \mathbb{Z}/p\mathbb{Z}$ by the residues in the interval $[0, 2^{16}]$. We associate with a double byte $(x_1, \ldots, x_{16}) \in \{0,1\}^{16}$ the integer $\sum_{i=1}^{16} x_i 2^{i-1} \in \mathbb{Z}_p$. Identification of a bit string in $\{0,1\}^{16}$ with its corresponding integer yields natural inclusions $\{0,1\}^{16} \subset \mathbb{Z}_p$, $\{0,1\}^{16n} \subset \mathbb{Z}_p^n$ for all n. Conversely we associate with $a \in [0, 2^{16}]$ the integer a', $a' = a (\bmod 2^{16})$ in $[0, 2^{16}-1]$, corresponding to a double byte in $\{0,1\}^{16}$. Here $a = 2^{16}$ and $a = 0$ both yield $a' = 0$.
Due to the inclusion $\{0,1\}^{256} \subset \mathbb{Z}_p^{16}$ it is sufficient to compute a function $g : \mathbb{Z}_p^{16} \longrightarrow \{0,1\}^{128}$. We will use the discrete Fourier transform $FT_8 : \mathbb{Z}_p^8 \longrightarrow \mathbb{Z}_p^8$ with the primitive root 2^4 of order 8. We have

$$FT_8(a_0, \ldots, a_7) = (b_0, \ldots, b_7) \qquad \text{with}$$

$$b_i = \sum_{j=0}^{7} 2^{4ij} a_j \ (\bmod p) \qquad \text{for } i = 0, \ldots, 7 \,.$$

Algorithm for $\mathbf{g} : \mathbb{Z}_{\mathbf{p}}^{\mathbf{16}} \longrightarrow \{0,1\}^{\mathbf{8\cdot 16}}$

```
INPUT $(e_0, \ldots, e_{15}) \in \{0,1\}^{16\cdot 16} \subset \mathbb{Z}_p^{16}$
    1. FOR  $i = 0, \ldots, 15$  DO  $e_i := e_i + e_{i-1}^* e_{i-2}^* + e_{i-3} + 2^i \pmod{p}$
       where $e^* = e$ if $e \neq 0$ and $e^* = 1$ if $e = 0$ .
       (The indices $i, i-1, i-2, i-3$ are taken modulo 16)
    2. $(e_0, e_2, \ldots, e_{14}) := \mathrm{FT}_8(e_0, e_2, \ldots, e_{14})$
       $(e_1, e_3, \ldots, e_{15}) := \mathrm{FT}_8(e_1, e_3, \ldots, e_{15})$
    3. FOR  $i = 0, \ldots, 15$  DO  $e_i := e_i + e_{i-1}^* e_{i-2}^* + e_{i-3} + 2^i \pmod{p}$
OUTPUT $e_i \pmod{2^{16}}$  for  $i = 8, \ldots, 15$
```

Comparison to the algorithm FFT-Hash I presented in the rump session of CRYPTO'91.
The previous version of the above algorithm g performed instead of Steps 1 – 3 the following steps:

```
REPEAT  Steps 1,2  twice
    1. $(e_0, e_2, \ldots, e_{14}) := \mathrm{FT}_8(e_0, e_2, \ldots, e_{14})$
    2. FOR  $i = 0, \ldots, 15$  DO  $e_i := e_i + e_{i-1} e_{i-2} + e_{e_{i-3}} + 2^i \pmod{p}$
END  loop
```

This version has been broken by BOSSELAER and DAEMEN and independently by M. GIRAULT. A weakness of this algorithm is that the information contained in an input element e_j with $8 \leq j \leq 15$ must in some cases not be diffused to all final elements $e_0, \ldots, e_{15}$. E.g. if j is odd then e_j has no impact on the Fourier transform in Step 1. Moreover if $e_{j-1} = e_{j+1} = 0$ holds just before the execution of $e_j := e_j + e_{j-1}e_{j-2} + e_{e_{j-3}} + 2^j \pmod{p}$ in Step 2 then the first round of Steps 1 and 2 will diffuse e_j only to those e_i for which $e_{i-3} = j$ holds during the execution of $e_i := e_i + e_{i-1}e_{i-2} + e_{e_{i-3}} + 2^i \pmod{p}$. A critical point is that the transformation is almost linear in e_i if e_{i-1} and e_{i+1} are 0. The new function g does not suffer from this weakness.

Counter measures against the Bosselaer–Daemen/Girault attack
Each of the measures 1, 2 and 3, when taken separately, protects against the above attack, measure 4 slows the attack down.

1. Replace the term $e_{e_{i-3}}$ in FFT–Hash I by e_{i-3}
2. Replace the FT_8 on the elements e_i with even index i by a full Fourier transform FT_{16} on all 16 elements
3. Eliminate zero–multipliers in the polynomial recursion, e.g. replace the recursion in Step 2 of FFT–Hash I by

$$e_i := e_i + e^*_{i-1}e^*_{i-2} + e_{e_{i-3}} + 2^i \ (\text{mod} p) \ ,$$

where $e^* = e$ if $e \neq 0$ and $e^* = 1$ if $e = 0$.

4. Invert the order of Step 1 and Step 2 in FFT–Hash I.

In FFT–Hash II we have essentially combined all these four counter measures. Step 2 of this algorithm performs a full Fourier transform except for the stage that mixes even and odd elements. We think that this stage is not needed because this mixing is done in Steps 1 and 3.

The function g is easy to invert. This is because Steps 1 – 3 can easily be inverted. We can complement the output elements $e'_8, \ldots, e'_{15}$ by any elements $e_0, \ldots, e_7$. We obtain a corresponding input for g by inverting Steps 1 – 3 on $e_0, \ldots, e_7$, $e'_8, \ldots, e'_{15}$. If this inversion yields elements in the interval $[0, 2^{16} - 1]$, we have found an inverse image of $(e'_8, \ldots, e'_{15})$ for g. Since g is easy to invert it is not collision–free. The fast inversion algorithm for function g does not necessarily constitute a weakness for the hash function h since it does not produce inverse images having the prefix $H_0 \in \{0,1\}^{128}$.

Design principles for the function g. The polynomial recursion in Steps 1 and 3 is at both ends of the procedure. It forms a shell for the linear transformation in Step 2. Each of Steps 1 and 3 generates a polynomial transformation so that e_i^{new} is a polynomial of degree at least F_{i+3} in $e_0^{\text{old}}, \ldots, e_{15}^{old}$ where F_{i+3} is the $i+3$–th Fibbonacci number. Thus the degree of e_{15}^{new} is at least $F_8 = 2584$. We believe that the problems 1 and 2 are intractable in worst case for this polynomial transformation. That is it seems difficult to enforce a particular pattern for the second half of the output of this transformation by manipulating only the first half of the input. The function g is a polynomial transformation with the property that all output coordinates are polynomials of degree at least 229976 in the input coordinates.

We use the Fourier transform to mix in a perfect way the information contained in the numbers $e_0, \ldots, e_{15}$. It is of interest that the function g transforms message bits and hash bits in a similar way.

The cost for evaluating g.
Evaluating the Fourier transform FT_8 requires $3 \cdot 8$ additions and $3 \cdot 8$ shifts modulo p. The two evaluations of FT_8 cost 48 additions modulo p and 48 shifts modulo p. Steps 1 and 3 require 64 additions and 32 multiplications modulo p. We neglect the costs for adding the bit 2^i and for setting up the FOR-loop. We obtain the following total costs

$$\begin{array}{rl} 112 & \text{additions} \mod p \\ 32 & \text{multiplications} \mod p \\ \underline{48} & \text{shifts} \mod p \\ 192 & \text{operations} \mod p\,. \end{array}$$

Variations of the function g *preserving the invertability.* The provable properties of Section 2 rely on the invertability of the Steps 1 – 3 in the algorithm for g. It is interesting to study a class of functions g with this invertability. We can replace Steps 1 and 3 of the algorithm for g by the following step

$$e_i := e_i + f(e_0, \ldots, e_{i-1}, e_{i+1}, \ldots, e_{15}, i) \bmod p \text{ for } i = 0, \ldots, 15$$

where f is an arbitrary function. For any such function f the Steps 1,2,3 in the algorithm for g remain invertible. It may be possible that there are better choices for the function f enhancing security and efficiency than our choice $f(e_0, \ldots, e_{i-1}, e_{i+1}, \ldots, e_{15}, i) = \bar{e}_{i-1}\bar{e}_{i-2} + e_{i-3} + 2^i$. Alternatively we can also use the recursion

$$e_i := e_i \oplus f(e_0, \ldots, e_{i-1}, e_{i+1}, \ldots, e_{15}, i) \bmod p \text{ for } i = 0, \ldots, 15$$

where f is an arbitrary function and $\oplus : \mathbb{Z}_p^2 \to \mathbb{Z}_p$, $(x, y) \mapsto x \oplus y$ is the bitwise addition which is defined as follows. Let $x, y \in \mathbb{Z}_p$ have binary representations $(x_0, \ldots, x_{16})$ and $(y_0, \ldots, y_{16})$, i.e. $x = \sum_i 2^i x_i$, $y = \sum_i 2^i y_i$. Then we put $x \oplus y = \sum_{i=0}^{16} 2^i \ (x_i + y_i \bmod 2) \bmod p$. It is of interest to note that the function $\oplus$ is *associative*. The associativity of $\oplus$ implies that Steps 1 and 3 in the algorithm for g remain invertible.

Transforming g *into a function that is most likely collision-free.* That there is a simple heuristic that transforms g into a function $\bar{g}$ that is most likely collision-free. We obtain $\bar{g}$ from g by inserting a Step 4 into the computation that adds the input elements e_i^{inp} to e_i for $i = 8, \ldots, 15$, i.e.

$$\text{Step 4.} \qquad e_i := e_i + e_i^{\text{inp}} (\text{mod} p) \text{ for } i = 8, \ldots, 15.$$

2 Provable properties of the function g

We show in the next proposition that no information is lost during the computation of g . Moreover this computation cannot create any redundancies.

Proposition 1. *Steps 1 - 3 act as bijective transformations on the configurations* $(e_0, \ldots, e_{15}) \in \mathbb{Z}_p^{16}$.

Proof . The Fourier transform $FT_8 : \mathbb{Z}_p^8 \to \mathbb{Z}_p^8$ is bijective. Its inverse is given by

$$FT_8^{-1}(b_0, \ldots, b_7) = (a_0, \ldots, a_7)$$

$$a_i = 8^{-1} \sum_{j=0}^{7} 2^{-4ij} b_j \pmod p \qquad \text{for } i = 0, \ldots, 7 .$$

The inverse is correct since 2^4 is a primitive root of order 8 in $\mathbb{Z}_p$ i.e. $2^{4 \cdot 8} = 1 \bmod p$, $2^{4 \cdot 4} = -1 \bmod p$.
Moreover a single step

$$e_i^{\text{new}} := e_i + e_{i-1}^* e_{i-2}^* + e_{i-3} + 2^i \pmod p$$

can be inverted as

$$e_i := e_i^{\text{new}} - e_{i-1}^* e_{i-2}^* - e_{i-3} - 2^i \pmod p$$

□

By Proposition 1 no collision of messages is possible within the evaluation of the function g, i.e. distinct messages yield distinct configurations in the same step. If the input $(e_0, \ldots, e_{15})$ for g is uniformly distributed over $\mathbb{Z}_p^{16}$ then the final configuration $(e_0, \ldots, e_{15})$ in the program for g is also uniformly distributed over $\mathbb{Z}_p^{16}$.

Next we study the probability distribution of $g(H, M)$ when (H, M) ranges uniformly over $\{0,1\}^{256}$. We first prove an upper bound for the probability of any output value for g. Moreover we show that the distribution of $g(H, M)$ is close to the uniform distribution on $\{0,1\}^{128}$. The following theorem is interesting since it excludes the construction of collisions for g corresponding to most likely outputs. Note that it is impossible to verify the statement of the theorem by empirical testing.

Theorem 2. *If the pair* (H, M) *is uniformly distributed over* $\{0,1\}^{256}$ *then we have for all* $a \in \{0,1\}^{128}$ *that prob* $[g(H, M) = a] \leq 2^{-120} e^{-2^{-13}} \approx 2^{-120}$.

Proof. Let $\text{out} : \mathbb{Z}_p^{16} \to \{0,1\}^{128}$ be the function that associates with a final configuration $(e_0, \ldots, e_{15}) \in \mathbb{Z}_p^{16}$ the output $(e'_8, \ldots, e'_{15}) \in \{0,1\}^{128}$ of g. For every $a = (e'_8, \ldots, e'_{15})$ we have

$$\# \, \text{out}^{-1}(a) \le (2^{16}+1)^8 2^t$$

where $t = \#\{i | 8 \le i \le 15, e'_i = 0\}$. Since pairwise distinct inputs (H, M) for g are mapped into pairwise distinct final configurations we see that

$$\begin{aligned} &\text{prob}\,[g(H,M) = a] \le (2^{16}+1)^8 2^t 2^{-256} \\ &= 2^{-128+t}(1+2^{-16})^8 \approx 2^{-128+t} e^{-2^{-13}} \approx 2^{-128+t} , \end{aligned}$$

where the probability space is the set of all $(H, M) \in \{0,1\}^{256}$. □

Let X, Y be probability distributions over some finite set S. We measure the distance of X and Y by the 1-norm

$$\|X - Y\|_1 := \sum_{s \in S} \left|\text{prob}_x(s) - \text{prob}_y(s)\right| .$$

We have that $0 \le \|X - Y\|_1 \le 2$ and $\|X - Y\|_1 = 0$ if and only if $X = Y$. The norm satisfies the triangular inequality:

$$\|X - Y\|_1 \le \|X - Z\|_1 + \|Z - Y\|_1 . \tag{1}$$

The norm cannot increase by the application of a function f:

$$\|f(X) - f(Y)\|_1 \le \|X - Y\|_1 . \tag{2}$$

Here $f(X)$ is the probability distribution on Image (f) given as

$$\text{prob}_{f(X)}(t) = \sum_{f(s)=t} \text{prob}_X(s) .$$

Let U_S denote the uniform distribution on the set S. If $S' \subset S$ we have

$$\|U_{S'} - U_S\| = 2\left(1 - \frac{\#S'}{\#S}\right) . \tag{3}$$

Theorem 3. $\|g\left(U_{\{0,1\}^{256}}\right) - U_{\{0,1\}^{128}}\|_1 \le 10^{-3}$, *i.e. if* (H, M) *ranges uniformly over* $\{0,1\}^{256}$ *the 1-norm distance of* $g(H, M)$ *from the uniform distribution on* $\{0,1\}^{128}$ *is at most* 10^{-3}.

Proof. Let $\bar{g} : \mathbb{Z}_p^{16} \longrightarrow \mathbb{Z}_p^{16}$ be the bijective function that associates to an initial configuration the corresponding final configuration in the computation of g. We have that

$$\bar{g}\left(U_{\mathbb{Z}_p^{16}}\right) = U_{\mathbb{Z}_p^{16}} \ . \tag{4}$$

We see from (2) and (3) that

$$\begin{aligned} &\left\|\bar{g}\left(U_{\mathbb{Z}_p^{16}}\right) - \bar{g}\left(U_{\{0,1\}^{256}}\right)\right\|_1 \leq \left\|U_{\mathbb{Z}_p^{16}} - U_{\{0,1\}^{256}}\right\|_1 \\ &\leq 2\left(1 - \left(\frac{2^{16}}{2^{16}+1}\right)^{16}\right) \approx 2\left(1 - \left(1 - \frac{1}{2^{16}+1}\right)^{16}\right) \\ &\approx 2\left(1 - e^{-16/2^{16}}\right) \approx 4.88 \cdot 10^{-4} \ . \end{aligned} \tag{5}$$

Let $\text{out} : \mathbb{Z}_p^{16} \longrightarrow \{0,1\}^{128}$ be the function that associates with a final configuration $(e_0, \ldots, e_{15}) \in \mathbb{Z}_p^{16}$ the output $(e'_8, \ldots, e'_{15}) \in \{0,1\}^{128}$. We have that $\text{out}\left(U_{\{0,1\}^{256}}\right) = U_{\{0,1\}^{128}}$. We finally get

$$\begin{aligned} &\left\|g\left(U_{\{0,1\}^{256}}\right) - U_{\{0,1\}^{128}}\right\|_1 \\ &= \left\|\text{out}\ \bar{g}\left(U_{\{0,1\}^{256}}\right) - \text{out}\left(U_{\{0,1\}^{256}}\right)\right\|_1 \\ &\overset{(2)}{\leq} \left\|\bar{g}\left(U_{\{0,1\}^{256}}\right) - U_{\{0,1\}^{256}}\right\|_1 \\ &\overset{(1),(4)}{\leq} \left\|\bar{g}\left(U_{\{0,1\}^{256}}\right) - \bar{g}\left(U_{\mathbb{Z}_p^{16}}\right)\right\|_1 + \left\|U_{\mathbb{Z}_p^{16}} - U_{\{0,1\}^{256}}\right\|_1 \\ &\overset{(5)}{\leq} 2 \cdot 4.88 \cdot 10^{-4} < 10^{-3} \ . \end{aligned}$$

□

We next consider the discrete Fourier transform FT_8. If a_j ranges uniformly over $\mathbb{Z}_p$ and the a_k for $k \neq j$ are all fixed then each component b_i of $(b_0, \ldots, b_7) = FT_8(a_0, \ldots, a_7)$ ranges uniformly over $\mathbb{Z}_p$ for $i = 0, \ldots, 7$. This holds because no coefficient 2^{4ij} of FT_8 is zero in $\mathbb{Z}_p$. We next consider pairs of components.

Lemma 4. *Let* $0 \leq i,\ \bar{\imath}, j, \bar{\jmath} \leq 7$ *and* $(j - \bar{\jmath})(i - \bar{\imath}) \not\equiv 0 (\bmod 8)$. *If* $(a_j, a_{\bar{\jmath}})$ *ranges uniformly over* $\mathbb{Z}_p^2$ *and the* a_k *for* $k \notin \{j, \bar{\jmath}\}$ *are all fixed then the pair* $(b_i, b_{\bar{\imath}})$ *from* $(b_0, \ldots, b_7) = FT_8(a_0, \ldots, a_7)$ *ranges uniformly over* $\mathbb{Z}_p^2$.

Proof. Consider the matrix $[16^{ij}]_{0\le i,j\le 15}$ representing the linear transformation FT_8. It is sufficient to prove that the 2×2 matrix

$$\begin{bmatrix} 16^{ij}, & 16^{i\bar{j}} \\ 16^{\bar{i}j}, & 16^{\bar{i}\bar{j}} \end{bmatrix}$$

corresponding to rows $i,\bar{i}$ and columns $j,\bar{j}$ of this matrix is regular. The determinant of this matrix is

$$16^{ij+\bar{i}\bar{j}} - 16^{i\bar{j}+\bar{i}j} = 16^{ij+\bar{i}\bar{j}}\left(1-16^{(\bar{i}-i)(j-\bar{j})}\right) .$$

This determinant is zero modulo p iff $(i-\bar{i})(j-\bar{j}) = 0 (\bmod 8)$. □

We see from Lemma 4 that independent pairs of input components $(a_j, a_{\bar{j}})$ for FT_8 yield independent pairs of output components $(b_i, b_{\bar{i}})$ for most $(i,\bar{i},j,\bar{j})$, in particular for all $i,\bar{i}$ with $i-\bar{i}$ odd. The condition that $i-\bar{i}$ is odd seems to be sufficient for our purpose since any two output coordinates $e_k^{\text{new}}, e_{\bar{k}}^{\text{new}}$ with $k \ne \bar{k}$ of Step 2

$$e_k^{\text{new}} := e_k + e_{k-1}^* e_{k-2}^* + e_{k-3} + 2^k \ (\bmod p)$$

depend on two distinct output components $e_{2i}, e_{2\bar{i}}$ of

$$(e_0, e_2, \ldots, e_{14}) = FT_8(\bar{e}_0, \bar{e}_2, \ldots, \bar{e}_{14})$$

where $i-\bar{i}$ is odd. If e.g. k and $\bar{k}$ are even then $e_k^{\text{new}}, e_{\bar{k}}^{\text{new}}$ depend on $e_{2i}, e_{2\bar{i}}$ with $i = k/2$ and with $\bar{i} = \bar{k}/2$ and $\bar{i} = \bar{k}/2+1$.

Lemma 4 can be extended from pairs $(i,\bar{i})$, $(j,\bar{j})$ to triples, quadruples of components and so on. Since independence of components is unlikely to cancel out later on in the computation of g this shows that the function g has highly desirable statistical properties.

Appendix : A program for h

```
INPUT $M_1, M_2, \ldots, M_n \in \{0,1\}^{128}$ with $M_j = M_{j,0} \| M_{j,1} \cdots \| M_{j,7}$ and
      $M_{j,i} \in \{0,1\}^{16}$.
      (Let $i = i_0 2^2 + i_1 2 + i_2 \in [0,7]$ with $i_j \in \{0,1\}$ and let
      $\text{rev}(i) = i_2 2^2 + i_1 2 + i_0$ be the number $i$ in reversed notation. Let
      $i(j,0)$ ($i(j,1)$, resp.) be obtained from $i$ by setting $i_j := 0$ ($i_j := 1$,
      resp.). We abbreviate $e_i = e_i^{(0)}$ and $e'_i = e_i(\text{mod} 2^{16})$. Lower in-
      dices $i$ of $e_i^{(j)}$ are taken modulo 16.)
INITIATION $(e_8, e_9, \ldots, e_{15}) := 0123456789abcdeffedcba9876543210 \in (\mathbb{Z}_p)^{16}$
      (in hexadecimal notation)
      FOR $k = 1, \ldots, n$ DO
          FOR $i = 0, \ldots, 7$ DO $[e_i := e'_{i+8}, e_{i+8} := M_{k,i}]$
          FOR $i = 0, \ldots, 15$ DO
              $e_i := e_i + e^*_{i-1} e^*_{i-2} + e_{i-3} + 2^i \bmod (2^{16}+1)$
              where $e^* = e$ if $e \neq 0$ and $e^* = 1$ if $e = 0$
          FOR $j = 0, 1, 2$ FOR $i = 0, \ldots, 7$ DO
              $e_{2i}^{(j+1)} := e_{2 \cdot i(j,0)}^{(j)} + 16^{\text{rev}(i) 2^{2-j}} e_{2 \cdot i(j,1)}^{(j)} \bmod (2^{16}+1)$
      $e_{2i+1}^{(j+1)} := e_{2 \cdot i(j,0)+1}^{(j)} + 16^{\text{rev}(i) 2^{2-j}} e_{2 \cdot i(j,1)+1}^{(j)} \bmod (2^{16}+1)$
          FOR $i = 0, \ldots, 15$ DO $[e_{2i} := e_{2 \cdot \text{rev}(i)}^{(3)}, e_{2i+1} := e_{2 \cdot \text{rev}(i)+1}^{(3)}]$
          FOR $i = 0, \ldots, 15$ DO
              $e_i := e_i + e^*_{i-1} e^*_{i-2} + e_{i-3} + 2^i \bmod (2^{16}+1)$
      END for $k$
   OUTPUT $e'_8, \ldots, e'_{15}$.
```