Parallel FFT–Hashing

C.P. Schnorr
Fachbereich Mathematik/Informatik
Universität Frankfurt
Postfach 111932
60054 Frankfurt a.M.
e-mail: schnorr@informatik.uni-frankfurt.de

S. Vaudenay
Dép. Math. Inf.
ENS Paris
45 Rue d'Ulm
75230-05 Paris
e-mail: vaudenay@dmi.ens.fr

Abstract

We propose two families of scalable hash functions for collision–resistant hashing that are highly parallel and based on the generalized fast Fourier transform (FFT). FFT–hashing is based on *multipermutations*. This is a basic cryptographic primitive for perfect generation of diffusion and confusion which generalizes the boxes of the classic FFT. The slower FFT–hash functions iterate a compression function. For the faster FFT–hash functions all rounds are alike with the same number of message words entering each round.

1 Introduction and surview

We propose two families of hash functions for collision–resistant hashing. These hash functions are scalable, highly parallel and based on the generalized fast Fourier transform (FFT). In comparison to FFT–Hash II we discard the polynomial iteration over a finite field. This yields an extremely simple and highly parallel design which is entirely defined by the FFT–graph. The only free parameters are the order 2^k of the FFT, the number of rounds and the boxes which we require to be multipermutations.

Our first and slower family of hash functions iterates a compression function. For this hashing we have determined, by black box cryptanalysis, the minimal number of rounds that is necessary for collision–resistant hashing. Black box cryptanalysis shows that iterating a compression function is not the best method for FFT–hashing. Our second and faster family of hash functions uses multiple fan–in of message words. A message word enters four times within the same round. The same number of message words enters each round so that all rounds are alike.

Multipermutations are a new cryptographic primitive for perfect generation of diffusion and confusion. For an arbitrary set E we call a permutation $B : E^2 \to E^2$, $B(a,b) = (B_1(a,b),\ B_2(a,b))$ a *multipermutation* if for every $a, b \in E$ the mappings $B_i(a,*), B_i(*,b)$ for $i = 1,2$ are permutations on E. Our hashing uses a vector space $E = \mathbb{F}^m$ over the Galois field $\mathbb{F} = \{0,1\}$, e.g. $\mathbb{F}^8$, $\mathbb{F}^{16}$, $\mathbb{F}^{32}$. There is a large variety of multipermutations for these E, both linear and non–linear over $\mathbb{F}$. Non–linear multipermutations can be generated by composing linear ones with non–linear permutations on E. We can use for this composition the multiplication modulo $2^m + 1$.

We present in section 2 a family of compression functions based on FFT–networks. In section 3 we report on *black box attacks* on these compression functions. In section 4 we propose fast FFT hash algorithms that do not iterate a compression function. In section 5 we present multipermutations for the case that E is a linear space over the field $\mathbb{F}$. An example hash algorithm is proposed in section 6.

2 A family of compression functions.

Notation 1. We let $\mathbb{F}$ denote the Galois field of order 2 and let $E = \mathbb{F}^m$ be the linear space over the field $\mathbb{F}$ with dimension m, e.g. $m = 8, 16, 32, 64$. We call the elements of E *words.*
2. Let k be a fixed integer, $k > 0$ and let $i \in \{0, \ldots, 2^k - 1\}$. Let $i_j \in \{0,1\}$ for $j = 0, \ldots, k-1$ denote the j–th bit of $i = i_0 + 2i_1 + \ldots + 2^{k-1} i_{k-1}$. Let the integer $i(j)$ be obtained from i by negating the bit i_j. For our purposes it is convenient to define for $j \geq k$ that $i_j = i_{j(\bmod k)}$, $i(j) = i(j \bmod k)$.

The compression function $\mathbf{g_{k,s} : E^{2^k} \to E^{2^{k-1}}}$

`INPUT` $e_i \in E$ for $i = 0, \ldots, 2^k - 1$

(We call $H = [e_i \mid i_0 = 0]$ the hash input and $M = [e_i \mid i_0 = 1]$ the message input)

`FOR` $j = 0, \ldots, s$ `DO`

$(e_i, e_{i(j)}) := B_{i,j}(e_i, e_{i(j)})$ for all i, $0 \leq i < 2^k$ with $i_j = 0$, in parallel

`OUTPUT` $g_{k,s}(H, M) = [e_i \mid i_s = 0] \in E^{2^{k-1}}$.

The choice of the boxes $\mathbf{B_{i,j}}$: We require that the boxes $B_{i,j}$ perform multipermutations on E^2. We call a permutation $B : E^2 \to E^2$, $B(a,b) = (B_1(a,b),\ B_2(a,b))$, a *multipermutation* if for every fixed $a, b \in E$ the mappings $B_i(a,*)$, $B_i(*,b)$ for $i = 1,2$ are permutations on E. We call the component mappings $B_i : E^2 \to E$ $i = 1,2$ a *bipermutation* because they act as a permutation on both inputs. A permutation $B : E^2 \to E^2$ is a multipermutation iff both component mappings B_1, B_2 are bipermutations.

It is important that the message inputs e_{2i+1} and hash inputs e_{2i} are mixed by the boxes $B_{i,0}$ of the first round $j = 0$. The hash outputs are from distinct boxes $B_{i,s}$ of the last round $j = s$.

It may be of interest that $g_{k,s}$ transforms the uniform distribution on E^{2^k} into the uniform distribution on $E^{2^{k-1}}$. This is because the boxes $B_{i,j}$ perform permutations on E^2.

We can represent the algorithm $g_{k,s}$ by a network. It consists of $s+1$ layers $j = 0, \ldots, s$. Layer j has 2^{k-1} vertices $B_{i,j}$ for $i = 0, \ldots, 2^k-1$ with $i_j = 0$. The edges of the $g_{k,s}$–network correspond to the inputs/outputs $e_i, e_{i(j)}$ of $B_{i,j}$. More precisely we let $e_{i,j}, e_{i(j),j}$ denote the inputs of $B_{i,j}$ and $e_{i,j+1}, e_{i(j),j+1}$ the outputs of $B_{i,j}$. The hash input is $H = [e_{i,0} \mid i_0 = 0]$, the hash output is $g_{k,s}(H, M) = [e_{i,s+1} \mid i_s = 0]$ and we have for $j = 0, \ldots, s$:

$$(e_{i,j+1}, e_{i(j),j+1}) = B_{i,j}(e_{i,j}, e_{i(j),j}) \quad \text{for} \quad 0 \leq i < 2^k \text{ with } i_j = 0.$$

By iterating the compression function $g_{k,s}$ we can transform arbitrary binary messages into a hash value in $E^{2^{k-1}}$ that is $m2^{k-1}$ bits long. We require that a given message, consisting of t bits, is padded so that its bit length becomes a multiple of $m2^{k-1}$. We recommend to append to the message a single "1" followed by a suitable number of "0" bits followed by the binary representation of t. So the padded message $M = M_1 M_2 \cdots M_n$ consists of n blocks $M_1, \ldots, M_n \in E^{2^{k-1}}$, $n = \lceil (t + 1 + \lceil \log_2(t+1) \rceil)/m2^{k-1} \rceil$.

The hash function $\mathbf{h_{k,s}}$

`INPUT` $M = M_1 \cdots M_n \in E^{n \cdot 2^{k-1}}$ (the padded message)

Fix an initial value $H_0 \in E^{2^{k-1}}$

$H_i := g_{k,s}(H_{i-1}, M_i)$ for $i = 1, \ldots, n$

`OUTPUT` $h_{k,s}(M) := H_n$ (the hash value of M)

The goal of the design is to make it infeasible to construct a collision for $h_{k,s}$, i.e. to produce distinct messages with the same hash value. To be infeasible for current technology the production of collisions should at least require about 2^{64} steps. We call $h_{k,s}$ *collision–resistant* if producing a collision is infeasible.

3 Black box attacks on the compression function.

We consider the following problems:

inverting $g_{k,s}$: Given random $H, H' \in E^{2^{k-1}}$ find $M \in E^{2^{k-1}}$ satisfying $g_{k,s}(H, M) = H'$.

$g_{k,s}$ *collisions* : Given random $H, H' \in E^{2^{k-1}}$ find $M, M' \in E^{2^{k-1}}$ satisfying $g_{k,s}(H, M) = g_{k,s}(H', M')$

Black box cryptanalysis has been introduced in [SV 93]. It assumes that the $B_{i,j}$ are black box multipermutations given by oracles. For $B_{i,j}(a, b) = (u, v)$ any two words out of a, b, u, v determine the other two. The box $B_{i,j}$ has *degree of freedom* 2, i.e. any two input/output edges of $B_{i,j}$ determine all edges of $B_{i,j}$.

A *black box attack* is a sequence of two type of steps

- Guess an unknown edge $e_{i,j}$, i.e. try all possible values in E.
- Evaluate a vertex (box) $B_{i,j}$, i.e. determine all its edges from two known ones.

The *complexity* of a step is the number of edge assignments to be tried for this step. Initially the step complexity is 1; it increases by a factor 2^m when guessing a new edge in $\mathbb{F}^m$; it decreases by a factor 2^m when evaluating a vertex with three known edges. The *complexity of the attack* is the maximal step complexity over all steps. This complexity depends in a crucial way on the choice of guessed edges and the order in which the boxes are evaluated. The *time bound* for the attack is the product of the complexity and the time for evaluating all vertices of the $g_{k,s}$–network. We take the second factor to be 1.

In some attacks we pick random values for certain edges and we hold these values fixed. E.g. when H, H' are given the edges corresponding to H, H' are fixed. We assume that all fixed edges are uniformly at random. Then the complexity of the attack is the average complexity with respect to the fixed random edges.

To invert $g_{k,s}$ we have to evaluate for the given random H, H' all boxes of the $g_{k,s}$–network. To produce a collision for $g_{k,s}$ we fix some output words of $g_{k,s}$ at random, we evaluate with these random values all boxes of the network and we apply the birthday paradox to the output words of $g_{k,s}$ that have not been fixed. Here are the complexities of the *best known* black box attacks according to [SV 93]:

	$g_{3,2}$	$g_{3,3}$	$g_{3,4}$	$g_{4,3}$	$g_{4,4}$	$g_{4,5}$	$g_{4,6}$	$g_{5,4}$	$g_{5,5}$	$g_{5,6}$	$g_{5,7}$	$g_{5,8}$
inversion	2^{2m}	2^{3m}	2^{4m}	2^{4m}	2^{5m}	2^{6m}	2^{8m}	2^{8m}	2^{9m}	2^{10m}	2^{12m}	2^{16m}
collision	2^{m}	2^{2m}	2^{2m}	2^{2m}	2^{3m}	2^{4m}	2^{4m}	2^{4m}	2^{5m}	2^{6m}	2^{8m}	2^{8m}

We let $I(k,s)$ and $C(k,s)$ denote the minimal complexity for the inversion of $g_{k,s}$ and for producing $g_{k,s}$–collisions where the minimum is taken over all black box attacks. The best known black box attacks are based on the following inequalities reflecting a simple divide and conquer argument:

$$
\begin{aligned}
&I(1,0) = 1, \quad I(k,k-1) \leq 2^{m2^{k-2}} && \text{for } k \geq 2 \\
&I(k,k-1+t) \leq I(k,k-1) \cdot I(t+1,t) && \text{for } 0 \leq t \leq k-1 \text{ and } k \geq 2\,. \\
&C(k,k-1+t) \leq 2^{m2^{k-3}} \cdot I(t+1,t) && \text{for } 0 \leq t \leq k-2 \text{ and } k \geq 3\,.
\end{aligned}
$$

We conjecture that these inequalities are actually equalities. If this holds true they determine the minimal complexities $I(k,s)$ and $C(k,s)$ for all $k \geq 2$, $s \geq k-1$ and the above tables give the minimal complexities for arbitrary black box attacks. The restriction $t \leq k-1$ (resp. $t \leq k-2$) is due to the inequalities $I(k,s) \leq 2^{m2^{k-1}}$, $C(k,s) \leq 2^{m2^{k-2}}$ since the length of the hash input/output is $2^{m2^{k-1}}$. Under our conjecture the inversion and the collision production have maximal complexity for $s = 2k-2$ and $s = 2k-3$, i.e. $I(k,2k-2) = 2^{m2^{k-1}}$ and $C(k,2k-3) = 2^{m2^{k-2}}$.

4 Fast parallel FFT–hashing

In view of the known attacks we are going to improve the hashing $h_{k,s}$. Here are our main points:

- Instead of discarding for every iteration of $g_{k,s}$ half of the hash words e_i we combine them with the next message words. Only after processing the entire message we apply the compression function $g_{k,s}$ discarding half the hash words.
- Each message word enters repeatedly in the same round. The same number of message words enters each round.

Using the circular rotation R on $\{0, 1, \ldots, 2^k - 1\}$ defined as $R(i_0 + 2i_1 + \ldots + 2^{k-1} i_{k-1}) = (i_{k-1} + 2i_0 + \ldots + 2^{k-1} i_{k-2})$ we can simplify the recursion for $g_{k,s}$ to

`FOR` $j = 0, \ldots, s$ `DO`

`FOR` $i = 0, 1, \ldots, 2^{k-1} - 1$ `DO` in parallel $(e_{R^j(2i)}, e_{R^j(2i+1)}) := B^*_{i,j}(e_{R^j(2i)}, e_{R^j(2i+1)})$

The boxes $B^*_{i,j}$, $0 \le i < 2^{k-1}$, are a corresponding permutation of the boxes $B_{i,j}$, $0 \le i < 2^k$ with $i_j = 0$. R^j is the circular rotation by j positions. We can further rewrite this to

1. `FOR` $i = 0, \ldots, 2^k - 1$ `DO` in parallel $e_{R(i)} := e_i$
2. `FOR` $j = 0, \ldots, s$ `DO`

   `FOR` $i = 0, \ldots, 2^{k-1} - 1$ `DO` in parallel $(e_{2i}, e_{2i+1}) := B^*_{i,j}(e_{R(2i)}, e_{R(2i+1)})$
3. `FOR` $i = 0, \ldots, 2^k - 1$ `DO` in parallel $e_{R^s(i)} := e_i$

We use this recursion for the fast parallel FFT hashing. We again write $B_{i,j}$ for $B^*_{i,j}$. Let the padded message M consist of n words $m_i \in E$ for $i = 0, \ldots, n-1$ and let $m_i = 0$ for $i \ge n$.

Fast parallel FFT hashing $\mathbf{h_k}$

`INPUT` $M = m_0 m_1 \ldots m_{n-1} \in E^n$ (the padded message)

Fix initial values $e_i \in E$ for $i = 0, \ldots, 2^k - 1$

`FOR` $j = 0, \ldots, \lceil n/\ell \rceil + s - 1$ `DO`

$e_{R(i)} := e_{R(i)}\, op_i\, m_{\ell j + (i \bmod \ell)}$ for $i = 0, \ldots, 4\ell - 1$

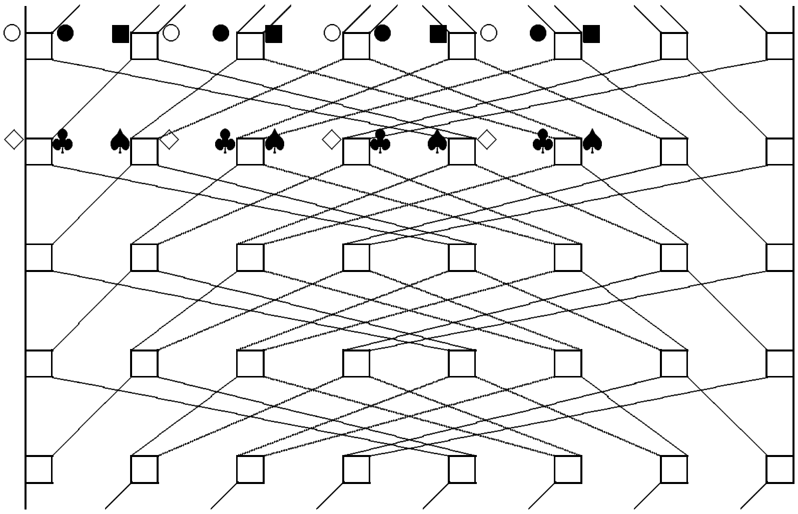
`FOR` $i = 0, \ldots, 2^{k-1} - 1$ `DO` in parallel $(e_{2i}, e_{2i+1}) := B_{i,j}(e_{R(2i)}, e_{R(2i+1)})$

`OUTPUT` $h_k(M) = [e_{2i} \mid i = 0, \ldots, 2^{k-1} - 1]$

Specific proposals for the binary operations $op_i : E^2 \to E$ and the multipermutations $B_{i,j} : E^2 \to E^2$ are given in section 6. We let each rounds $j = 0, \ldots, \lceil n/\ell \rceil - 1$ process $\ell \le 2^{k-2}$ consecutive message words $m_{\ell j + i}$ for $i = 0, \ldots, \ell - 1$. Each message word enters four times. After the last message word

m_{n-1} is entered there follow $s+1$ layers of multipermutations $B_{i,j}$ for $j = \lceil n/\ell \rceil - 1, \ldots, \lceil n/\ell \rceil + s - 1$ corresponding to the compression function $g_{k,s}$. The number $s+1$ of the final layers can well be smaller than $2k$ and even $2k-2$, the number of layers which maximize the complexities $I(k,s)$ and $C(k,s)$. This is because the multiple duplication of the message words seriously complicates the attacks.

Figure of the network for h_4 with $n = 6,\ \ell = 3,\ s = 3$. The values $k = 4, \ell = 3$ are as in the example algorithm in section 6.



message words ○ ● ■ ◇ ♣ ♠ marking
the edges to which they are mixed via op_i

PARALLEL COSTS (depth) of $\mathbf{h_k}$: Processing ℓ message words with $m \cdot \ell$ bits requires a single operation op_i and a single box $B_{i,j}$ plus s boxes on termination.

5 Examples of multipermutations

We introduce multipermutations on E^2 that are based on the operations $\oplus$ (bitwise XOR), $\wedge$ (bitwise AND), $+$ (addition modulo 2^m), $\cdot$ (multiplication modulo 2^m), $*$ (multiplication in $\mathbb{Z}^*_{2^m+1}$) and $\mathbf{R}$ the circular rotation to the right on E.

Notation Let $\oplus$ denote the bitwise XOR on $E = \mathbb{F}^m$, i.e. $\oplus$ is the vector addition on the linear space E over the field $\mathbb{F}$. For $b, c \in E$ let $b \wedge c$ denote the bitwise AND . Let $\mathbf{R}^\ell : E \to E$ denote the circular rotation by ℓ positions to the right.

Theorem 1 [SV 93] *For $c \in E,\ \ell \in \mathbb{Z}$ the mapping $L_c : E^2 \to E^2$, $L_c(a,b) = (a \oplus b,\ a \oplus (b \wedge c) \oplus \mathbf{R}^\ell(b))$, is a multipermutation if and only if the iterates of $\mathbf{R}^\ell$ on c take for each bit position both values* $0, 1$.

Remarks. 1. We see that if L_c is a multipermutation then $c \notin \{0^m, 1^m\}$ since otherwise the bits of c are constant and so are the bits of $\mathbf{R}^{\ell \cdot n}(c)$ for all n.

2. If $gcd(\ell, m) = 1$, i.e. if ℓ is odd, then L_c is a multipermutation if and only if $c \notin \{0^m, 1^m\}$. This is because, for odd ℓ, the iterates of $\mathbf{R}^\ell$ carry every bit of c to all positions.

Proof. L_c is a multipermutation if and only if both mappings

$$b \mapsto (b \wedge c) \oplus \mathbf{R}^\ell(b) , \qquad b \mapsto b \oplus (b \wedge c) \oplus \mathbf{R}^\ell(b)$$

are permutations of E. Now the claim follows from Lemma 2 with $d = c$ and $d = \bar{c}$, the bitwise negation of c. For the second mapping we use that $b \oplus (b \wedge c) = b \wedge \bar{c}$. $\square$

Lemma 2 [SV 93] *For $d \in E$ the linear mapping $f_d : E \to E$, $f_d(b) = (b \wedge d) \oplus \mathbf{R}^\ell(b)$, is a permutation of E if and only if the iterates $\mathbf{R}^{\ell \cdot n}(d)$ take for each bit position the 0–bit for some n.*

Let $d = (d_0, \ldots, d_{m-1}) \in E = \{0,1\}^m$ and for $i \notin \{0, \ldots, m-1\}$ let $d_i = d_{i(\mathrm{mod} m)}$. The claim means that f_d is a permutation if and only if for every i there is some n with $d_{i+\ell n} = 0$.

Further multipermutations can be constructed by composition. The composition of a multipermutation $P : E^2 \to E^2$ with arbitrary permutations $\sigma_1, \sigma_2 : E \to E$ yields new multipermutations

$$P(\sigma_1(a), \sigma_2(b)), \quad (\sigma_1 P_1(a,b), \sigma_2 P_2(a,b)) .$$

Also the inverse of a multipermutation is again a multipermutation.

We finally point to several useful permutations on E. We identify an integer $b \in \{0, \ldots, 2^m - 1\}$, $b = \Sigma_i b_i 2^i$ with its binary representation $(b_0, b_1, \ldots, b_{m-1}) \in E$.

Lemma 3 *Let $c \in \{0, \ldots, 2^m - 1\}$ be odd. Then $\sigma_c(a) = a \cdot c \bmod 2^m$ defines a permutation σ_c on E.*

Further permutations on E can be derived from the binary operation $*$ on $E = \{0, \ldots, 2^m - 1\}$:

$$a * b \;=\; (a'b' \bmod 2^m + 1) \bmod 2^m ,$$

where $a' = 2^m$ if $a = 0$ and $a' = a$ otherwise. Lai and Massey (1990) use the operation $*$ in the case $m = 16$. If $2^m + 1$ is prime e.g. for $m = 8, 16$ the operation $*$ is invertible. Then $(E, *)$ is a cyclic group of order 2^m with neutral element 1. The group $(E, *)$ is isomorphic to $\mathbb{Z}^*_{2^m+1}$, the multiplicative group of residues modulo $2^m + 1$. We have an isomorphism $\varphi : (E, *) \to \mathbb{Z}^*_{2^m+1}$, $\varphi(a) = a'$.

Lemma 4 *If $2^m + 1$ is prime then every $c \in E$ defines a permutation $a \mapsto a * c$ on E.*

6 An example hash algorithm $\mathbf{h_4}$

We propose particular boxes $B_{i,j}$ and operations op_i for the hash function h_4 of section 4. These propositions are preliminary and subject to further studies. Let $E = \mathbb{F}^{16}$, $m = 16$, $k = 4$, $\ell = 3$, $s = 5$. The choice $s = 5$ maximizes the collision complexity $C(4, s)$. This may be an overkill due to the multiple duplication of message words. One of the final 6 FFT–layers comes with the output.

INPUT $M = m_0 m_1 \cdots m_{n-1} \in E^n$ (the padded message)

FOR $i = 0, \ldots, 15$ DO $e_i := c_i$ (c_i is defined below)

FOR $j = 0, \ldots, \lceil n/3 \rceil + 3$ DO

FOR $i = 0, \ldots, 11$ DO

$$e_{R(i)} := \begin{cases} e_{R(i)} + m_{3j+(i \bmod 3)} & \text{for even } i \\ e_{R(i)} * m_{3j+(i \bmod 3)} & \text{for odd } i \end{cases}$$

FOR $i = 0, \ldots, 7$ DO in parallel

$$e_{2i} := e_{R(2i)} \oplus e_{R(2i+1)}\ ,\quad e_{2i+1} := e_{R(2i)} \oplus (e_{R(2i+1)} \wedge c) \oplus \mathbf{R}^{2i+1}(e_{R(2i+1)})$$

FOR $i = 0, \ldots, 15$ DO $e_i := e_i * c_i$

OUTPUT $h_4(M) := [e_{R(2i)} * e_{R(2i+1)} \mid i = 0, \ldots, 7]$

Here $+$ is the addition modulo 2^{16} on $E \cong \{0, \ldots, 2^{16} - 1\}$, $*$ is the multiplication in $E \cong \mathbb{Z}^*_{2^{16}+1}$ defined in section 5, $\oplus$ is the bitwise XOR, $\wedge$ is the bitwise logical AND, $\mathbf{R}$ is the circular right shift on E, R is the circular shift on $\{0, 1, \ldots, 15\} \cong \{0, 1\}^4$ so that $R(i) = 2i$ for $i \leq 7$ and $R(i) = 1 + 2(i - 8)$ for $i > 7$.

The constants. Let $c = 0000000011111111$. We define $c_0, \ldots, c_{15}$ in hexadecimal notation $(c_0, c_1, c_2, c_3) := (ef01, 2345, 6789, abcd)$, $(c_4, c_5, c_6, c_7) := (dcba,\ 9876,\ 5432,\ 10fe)$, $c_{8+i} := \bar{c}_i$ for $i = 0, \ldots, 7$ where $\bar{c}_i$ is the bitwise logical negation of c_i.

The example algorithm uses multipermutations $B_{i,j}$ that are non–linear both in $E = \mathbb{F}^{16}$, in $E \cong \mathbb{Z}/2^{16}\mathbb{Z}$ and in $E \cong \mathbb{Z}^*_{2^{16}+1}$. We have $B_{i,j}(a, b) = (\ (a \oplus b) * c_{2i},\ \ (a \oplus (b \wedge c) \oplus \mathbf{R}^{2i+1}(b)) * c_{2i+1}\ )$ for $j \leq \lceil n/3 \rceil + 3$. The final round $j = \lceil n/3 \rceil + 4$ is special $B_{i,j}(a, b) = (a * b,\ \text{undef})$.

The sequential costs The algorithm processes 48 message bits (three words) per round performing the following operations $22\ *,\quad 24\ \oplus,\quad 8\ \wedge,\quad 6\ +,\quad 8\ \mathbf{R}^{2i+1}$. In total there are 68 operations on E per round. There are 4 additional rounds per message and 8 additional $*$ operations in the output. The number of operations on E per message bit 68/48 is smaller than for FFT–Hash II which requires 192 operations per 128 message bits.

Parallelization The degree of parallelization is 16, i.e. 16 parallel processors yield a speed up factor 16. The potential of possible speed–up's in this design is twofold:

- If we operate on 32 words, i.e. $k = 5$, instead on 16 words the degree of parallelization is 32.
- If we choose for E the set of bit strings of length 32 with a suitable operation $*$ we obtain a speed–up factor 2.

It is important to analyse carefully the number of message words that can safely enter per round as we increase k . It is open whether the processing rate of *three* message words per round is the maximal rate for which the example algorithm is secure.

Acknowledgement We thank H. Hörner for producing the figure for the h_k–network. J. Massey proposed the term bipermutation.

References

[LM 91] Lai, X. and Massey, J.L.: A proposal of a new block encryption standard. Advances in Cryptology. Eurocrypt'90. Proceedings LNCS 473, pp. 389–404, Springer Verlag, Berlin, 1991.

[S 92] Schnorr, C.P.: FFT–Hash II, efficient cryptographic hashing. Proceedings EUROCRYPT'92. Springer LNCS 658 (1992), pp. 45–54.

[V 93] Vaudenay, S.: FFT–Hash II is not yet Collision–free. Advances in Cryptology, Proceedings of Crypto'92, Springer LNCS 740, (1993) pp. 587–593.

[SV 93] Schnorr, C.P. and Vaudenay, S.: Black Box Cryptanalysis of Hash Networks based on Multipermutations. Technical Report, Universität Frankfurt – ENS Paris, December 1993. Submitted to Eurocrypt'94.