# Spectral Hash - sHash
# A SHA-3 Candidate

Gökay Saldamlı
Cevahir Demirkıran
Megan Maguire
Carl Minden
Jacob Topper
Alex Troesch
Cody Walker
Çetin Kaya Koç

**University of California Santa Barbara**

**CS@UCSB - Nov 12, 2008**

## Our Motivation

- Current hash algorithms are weakened: MD5 & SHA-1
- NIST has a repertoire of newer algorithms: SHA-224, SHA-256, SHA-384, and SHA-512 since August 2002
- In response to recent advances in the cryptanalysis of hash functions, NIST has opened a public competition to develop a new cryptographic hash algorithm: SHA-3
- The deadline for submission was October 31, 2008

# Our Team

- We have submitted a new hash algorithm *Spectral Hash* (sHash) which is based on the properties of the Discrete Fourier Transform and nonlinear transformations via data dependent permutations

- This is a collaborative work between
  - Gökay Saldamlı (my Ph.D. student from OSU, 2006)
  - Cevahir Demirkıran (a Ph.D. student from Barcelona, Spain)
  - Megan Maguire, Carl Minden, Jacob Topper, Alex Troesch, Cody Walker (students from UCSB)
  - myself

# Submissions

- There seem to be about 45 submissions, however, NIST has not yet published the full list of submissions

- You can follow the excitement here:

  `http://csrc.nist.gov/groups/ST/hash/sha-3/index.html`

  `http://ehash.iaik.tugraz.at/wiki/The_SHA-3_Zoo`
  `http://en.wikipedia.org/wiki/SHA-3`

# sHash Building Blocks - Finite Fields

- Our hash function uses the elements of the fields $GF(2^4)$ and $GF(17)$
- The field $GF(2^4)$ is generated by the irreducible polynomial $p(x) = x^4 + x^3 + x^2 + x + 1$
- The arithmetic of the $GF(17)$ is simply mod 17 arithmetic

## sHash Building Blocks - DFTs

- The DFTs are performed in $GF(17)$
- We use 4-point DFTs and 8-point DFTs

$$\mathbf{X}_i = DFT_d(\mathbf{x}) := \sum_{j=0}^{d-1} \mathbf{x}_j \omega_d^{ij} \bmod 17,$$

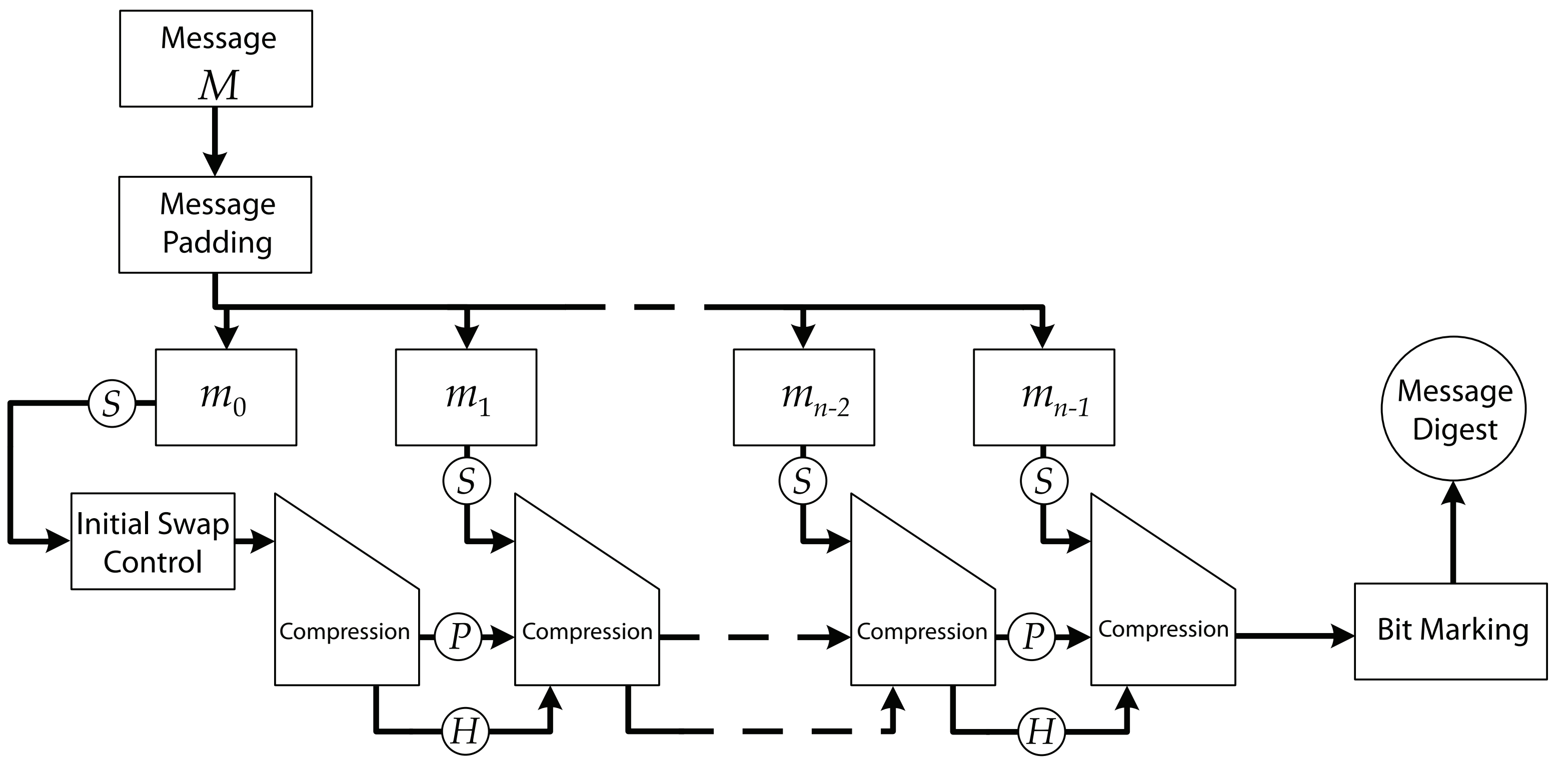
where $i = 0, 1, 2, \ldots d-1$, and $d$ is either $4$ or $8$.

- $\omega$ is the $d$-th root of unity in $GF(17)$
- For the 4-point DFTs ($d = 4$), we have $\omega_4 = 4$
- For the 8-point DFTs ($d = 8$), we have $\omega_8 = 2$

## sHash Building Blocks - Nonlinearity

- We employ the inverse map in $GF(2^4)$ which has good nonlinearity
- We use a nonlinear system of equations by selecting variables from a permutation table generated using data dependent permutations
- The general structure of sHash is an augmented Merkle-Damgard scheme

# Augmented Merkle-Damgard Scheme



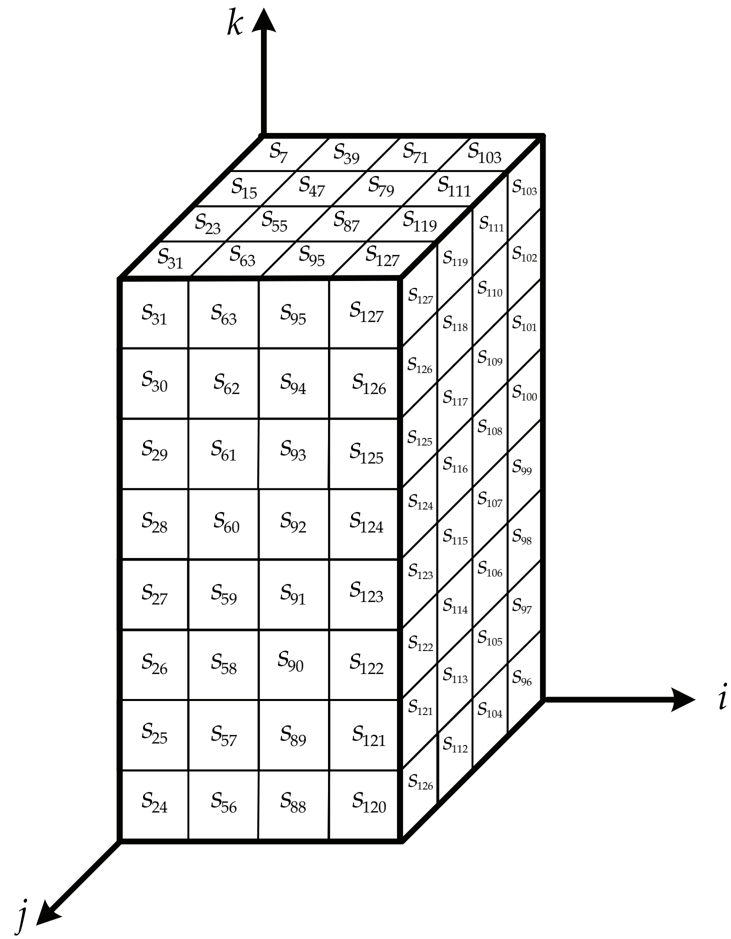
# 512-bit Message Block $m_i$

- s-prism: Break the message into 128 4-bit blocks represented as a $4 \times 4 \times 8$ prism

- p-prism: Create a permutation of 7-bit numbers $\{0, 1, \ldots, 127\}$ represented as a $4 \times 4 \times 8$ prism

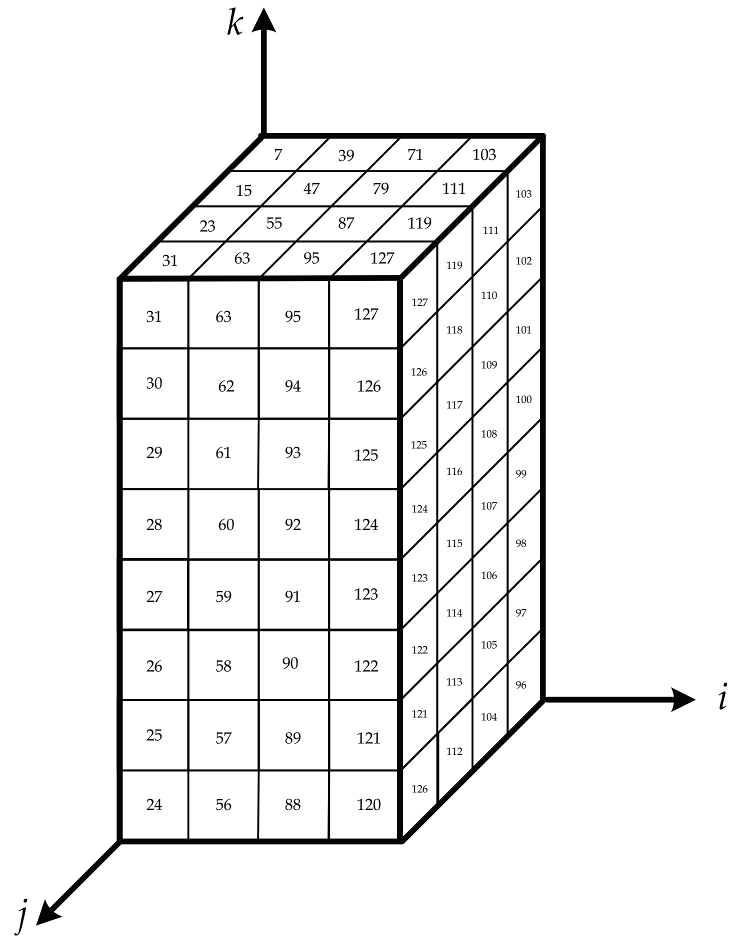
- permutations are determined by message bits and previous rounds

# S-Prism



$k$
$i$
$j$
$S_{7}$ $S_{39}$ $S_{71}$ $S_{103}$
$S_{15}$ $S_{47}$ $S_{79}$ $S_{111}$
$S_{23}$ $S_{55}$ $S_{87}$ $S_{119}$
$S_{31}$ $S_{63}$ $S_{95}$ $S_{127}$
$S_{31}$ $S_{63}$ $S_{95}$ $S_{127}$
$S_{30}$ $S_{62}$ $S_{94}$ $S_{126}$
$S_{29}$ $S_{61}$ $S_{93}$ $S_{125}$
$S_{28}$ $S_{60}$ $S_{92}$ $S_{124}$
$S_{27}$ $S_{59}$ $S_{91}$ $S_{123}$
$S_{26}$ $S_{58}$ $S_{90}$ $S_{122}$
$S_{25}$ $S_{57}$ $S_{89}$ $S_{121}$
$S_{24}$ $S_{56}$ $S_{88}$ $S_{120}$
$S_{127}$ $S_{126}$ $S_{125}$ $S_{124}$ $S_{123}$ $S_{122}$ $S_{121}$ $S_{126}$
$S_{119}$ $S_{118}$ $S_{117}$ $S_{116}$ $S_{115}$ $S_{114}$ $S_{113}$ $S_{112}$
$S_{111}$ $S_{110}$ $S_{109}$ $S_{108}$ $S_{107}$ $S_{106}$ $S_{105}$ $S_{104}$
$S_{103}$ $S_{102}$ $S_{101}$ $S_{100}$ $S_{99}$ $S_{98}$ $S_{97}$ $S_{96}$

# P-Prism



# Compression Function

- In the beginning of each round, the s-prism holds new message chunk, and the p-prism holds the permutation as updated by the previous round
- Compression function applies:
  - Affine transformation
  - Discrete Fourier transform
  - Nonlinear transformation

# Affine Transform

The following affine transform is applied to each entry of the s-prism:

$$S_{(i,j,k)} := \alpha(S_{(i,j,k)})^{-1} \oplus \gamma,$$

$$\alpha = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix} \text{ and } \gamma = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

# Discrete Fourier Transform

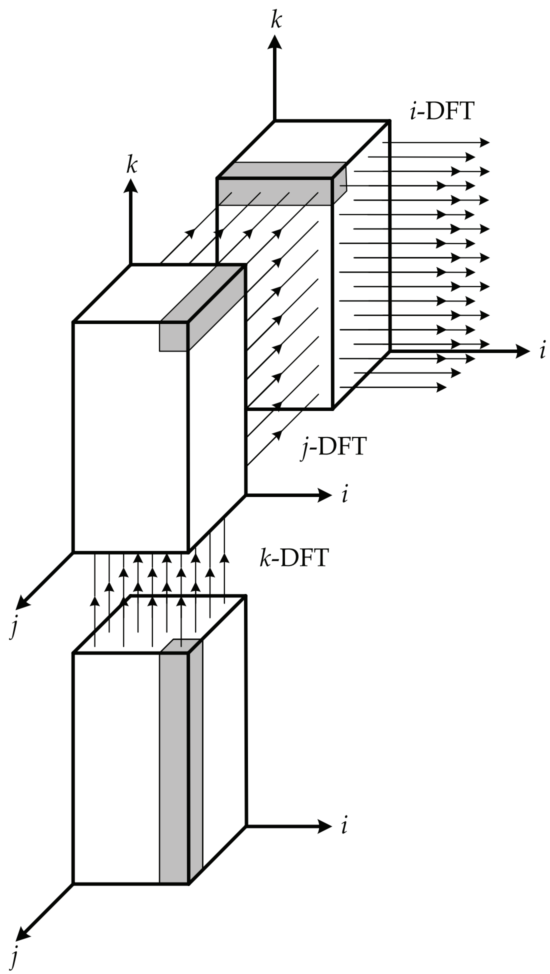
- After the affine transforms, we apply the 3-dimensional DFT to the s-prism.

- The DFT is defined over the prime field $GF(17)$, permitting transforms of length 8 and 4 for the principle roots of unity $\omega_8 = 2$ and $\omega_4 = 4$

- In the first iteration of the row-column method (i.e. DFT through the $k$-axis) one has to compute 16 different 1-dimensional 8-point DFTs

- Through the $i$ and $j$ axes, we need to calculate 32 different 4-point DFTs for each axis

# 3-D Discrete Fourier Transform



## Nonlinear Transformation

- At this step of the compression function, we collect and combine the data from the s-prism and p-prism to set up a nonlinear transformation that acts on the s-prism
- The nonlinear transformation is specifically designed to resist pre-image attacks and related weaknesses

$$S_{(i,j,k)} := (S'_{(i,j,k)} \oplus Pl_{(i,j,k)})^{-1} \oplus (S'_{P_{(i,j,k)}} \oplus Ph_{(i,j,k)})^{-1} \oplus H_{(i,j,k)},$$

for all $i, j = 0, 1, 2, 3$ and $k = 0, 1, \ldots, 7$.

# Rubic Rotations