Kompetisi Fungsi Hash NIST (SHA-3)

Bahan Kuliah IF4020 Kriptografi

Latar Belakang

- Seperti sejarah AES, *National Institute of Standards and Technology* (NIST) menyelenggarakan kompetisi terbuka untuk mengembangkan fungsi *hash* yang baru, bernama SHA-3

- SHA-3 menjadi komplementer SHA-1 dan SHA-2

- Kompetisi diumumkan pada tahun 2007 dan berakhir pada Oktober 2012 dengan memilih pemenang.

Proses Pemilihan

- Proses pemilihan terdiri dari 2 putaran dan final
- Jumlah *submission* adalah 64 rancangan fungsi *hash*.

- Putaran pertama (penyisihan): dipilih 51 *submission*
- Putaran kedua (semi final): dipilih 14 *submission*
- Babak final: 5 finalis

Finalis

1. BLAKE
2. Grøstl
3. JH
4. Keccak
5. Skein

BLAKE

- Designers: Jean-Philippe Aumasson, Luca Henzen, Willi Meier, Raphael C.-W. Phan

- Detail: Digest sizes 224, 256, 384 or 512 bits

- Rounds 14 or 16

Grøstl

- Designers: Praveen Gauravaram, Lars Knudsen, Krystian Matusiewicz, Florian Mendel, Christian Rechberger, Martin Schläffer, and Søren S. Thomsen

- Detail: Digest sizes 256 and 512

JH

- Designers: Hongjun Wu
- Detail: Digest sizes 224, 256, 384, 512

Keccak

- Designers: Guido Breton, Joan Daemen, Michaël Peeters and Gilles Van Assche.

- Detail: Digest sizes arbitrary

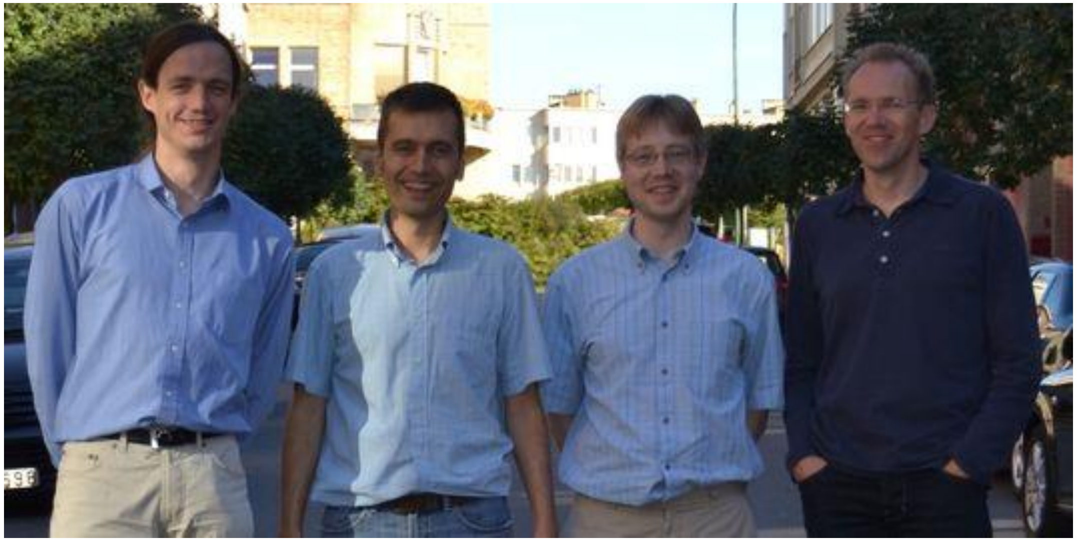
SKEIN

- Designers: Bruce Schneier, Stefan Lucks, Niels Ferguson, Doug Whiting, Mihir Bellare, Tadayoshi Kohno, Jon Callas and Jesse Walker.

- Detail: Digest sizes arbitrary
- Rounds: 72 (256 & 512 block size), 80 (1024 block size)

dan pemenangnya adalah…

Keccak



Guido Breton, Joan Daemen, Michaël Peeters and Gilles Van Assche.

Keccak terpilih sebagai SHA-3

Kriteria Penilaian oleh NIST

NIST noted some factors that figured into its selection as it announced the finalists:[11]

- Performance: "A couple of algorithms were wounded or eliminated by very large [hardware gate] area requirement – it seemed that the area they required precluded their use in too much of the potential application space.“

- Security: "We preferred to be conservative about security, and in some cases did not select algorithms with exceptional performance, largely because something about them made us 'nervous,' even though we knew of no clear attack against the full algorithm.“

- Analysis: "NIST eliminated several algorithms because of the extent of their second-round tweaks or because of a relative lack of reported cryptanalysis – either tended to create the suspicion that the design might not yet be fully tested and mature.“

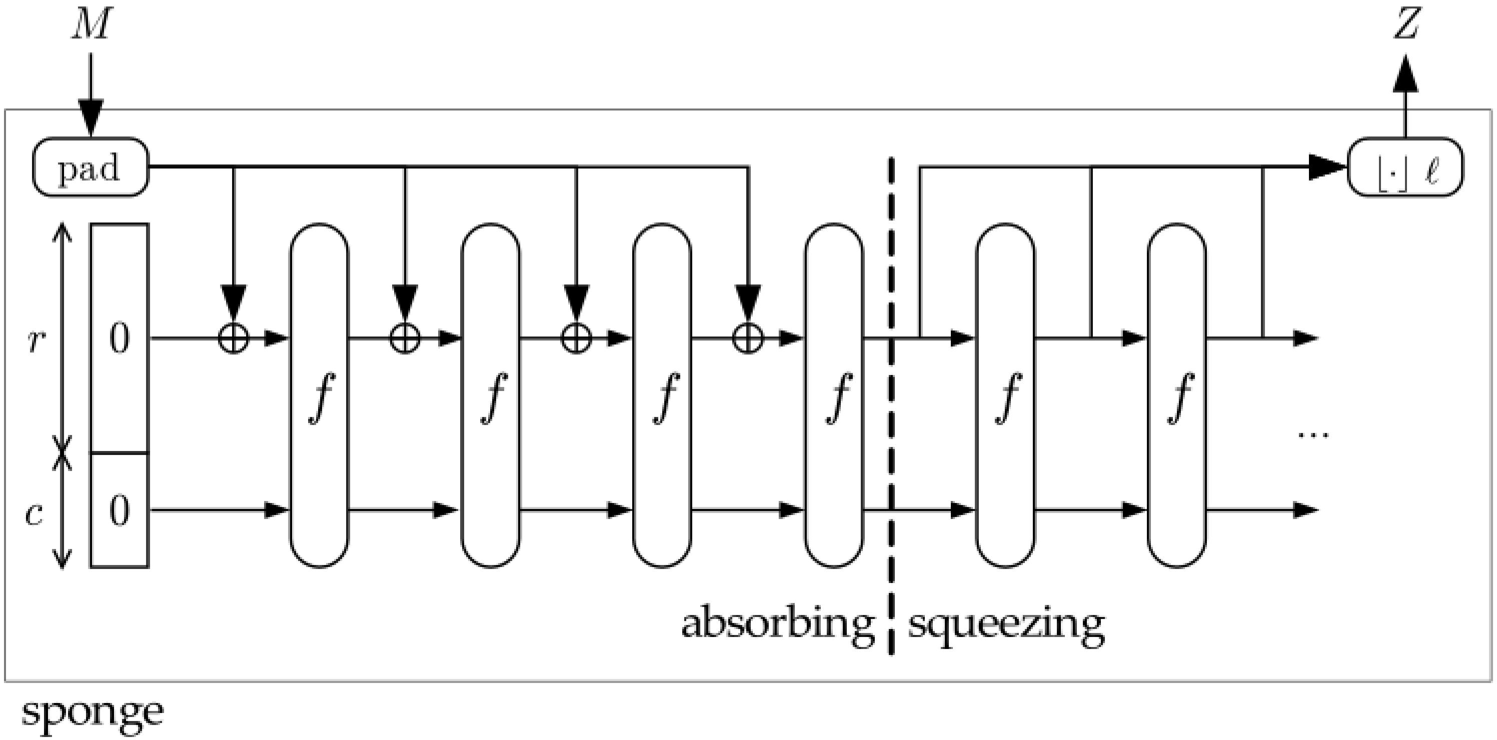
- Diversity: The finalists included hashes based on different modes of operation, including the HAIFA and sponge function constructions, and with different internal structures, including ones based on AES, bitslicing, and alternating XOR with addition.

Sekilas Keccak

- Nama 'Keccak' berasal dari 'Kecak', tarian Bali.

- Keccak berbeda dari finalis SHA3 lainnya dalam hal menggunakan konstruksi 'spons' (*sponge construction*). Jika desain lainnya bergantung pada 'fungsi kompresi, Keccak menggunakan fungsi non-kompresi untuk menyerap dan kemudian 'memeras' digest.

- Desain Keccak berbeda dari pendekatan yang ada. NIST merasa bahwa dalam kasus ini, yang berbeda adalah lebih baik.



Konstruksi spons

- First, the input string is padded with a reversible padding rule and cut into blocks of *r* bits. Then the *b* bits of the state are initialized to zero and the sponge construction proceeds in two phases:

  (a) In the absorbing phase, the *r*-bit input blocks are XORed into the first *r* bits of the state, interleaved with applications of the function *f*. When all input blocks are processed, the sponge construction switches to the squeezing phase.

  (b) In the squeezing phase, the first r bits of the state are returned as output blocks, interleaved with applications of the function f. The number of output blocks is chosen at will by the user.

- The last c bits of the state are never directly affected by the input blocks and are never output during the squeezing phase.

Sumber: http://sponge.noekeon.org/

```
Keccak[r,c](M) {
 Initialization and padding
 S[x,y] = 0,                              forall (x,y) in (0…4,0…4)
 P = M || 0x01 || 0x00 || … || 0x00
 P = P xor (0x00 || … || 0x00 || 0x80)

 Absorbing phase
 forall block Pi in P
   S[x,y] = S[x,y] xor Pi[x+5*y],        forall (x,y) such that x+5*y < r/w
   S = Keccak-f[r+c](S)

 Squeezing phase
 Z = empty string
 while output is requested
   Z = Z || S[x,y],                      forall (x,y) such that x+5*y < r/w
   S = Keccak-f[r+c](S)

 return Z
}
```

```
Keccak-f[b](A) {
  forall i in 0…nr-1
    A = Round[b](A, RC[i])
  return A
}

Round[b](A,RC) {
  θ step
  C[x] = A[x,0] xor A[x,1] xor A[x,2] xor A[x,3] xor A[x,4],   forall x in 0…4
  D[x] = C[x-1] xor rot(C[x+1],1),                             forall x in 0…4
  A[x,y] = A[x,y] xor D[x],                           forall (x,y) in (0…4,0…4)

  ρ and π steps
  B[y,2*x+3*y] = rot(A[x,y], r[x,y]),                forall (x,y) in (0…4,0…4)

  χ step
  A[x,y] = B[x,y] xor ((not B[x+1,y]) and B[x+2,y]), forall (x,y) in (0…4,0…4)

  ι step
  A[0,0] = A[0,0] xor RC

  return A
}
```

- Spesifikasi Keccak (termasuk *source code*) dapat dilihat di: http://keccak.noekeon.org/specs_summary.html