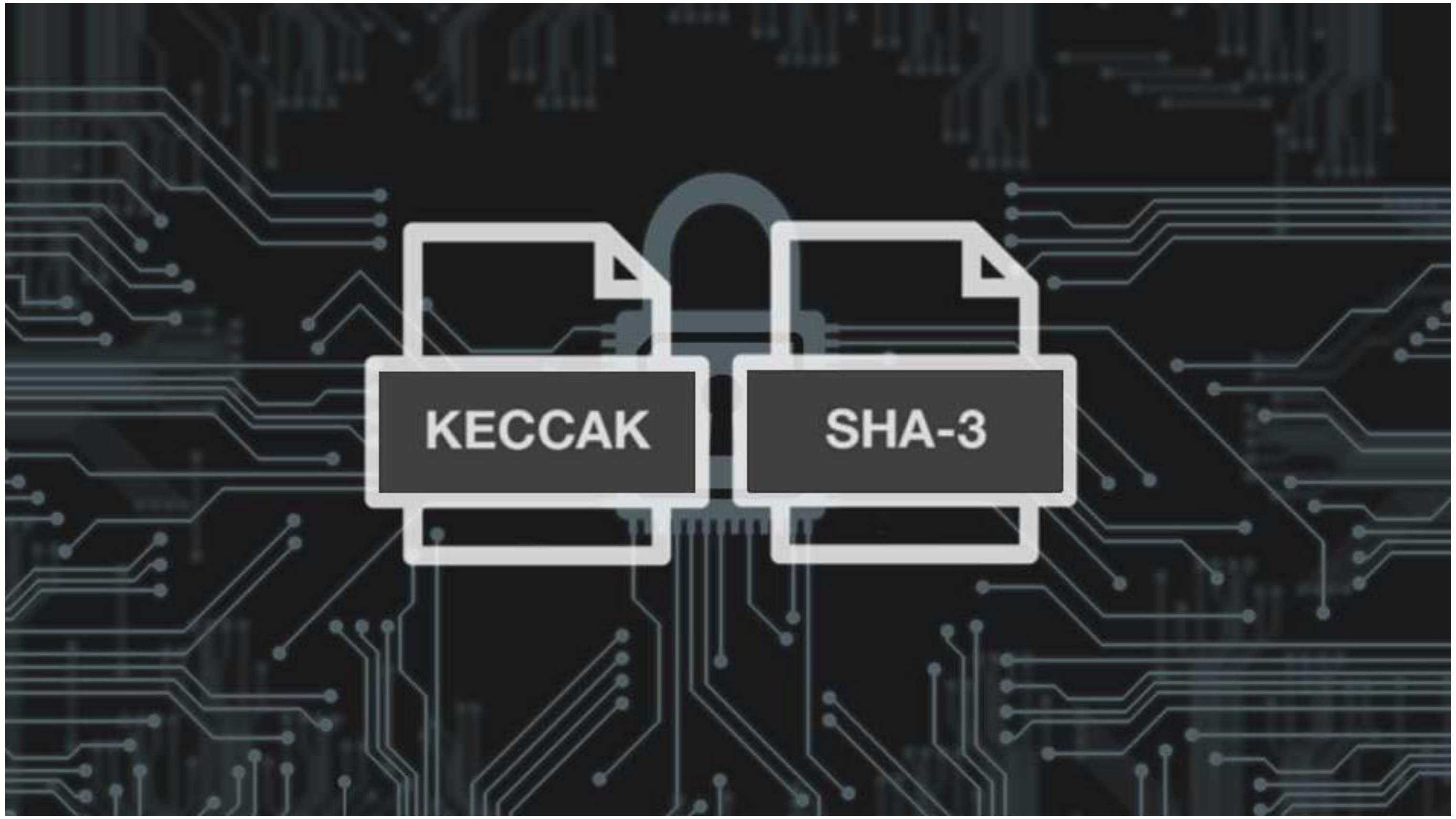
By Jonathan Duran

# SHA-3 (Keccak)

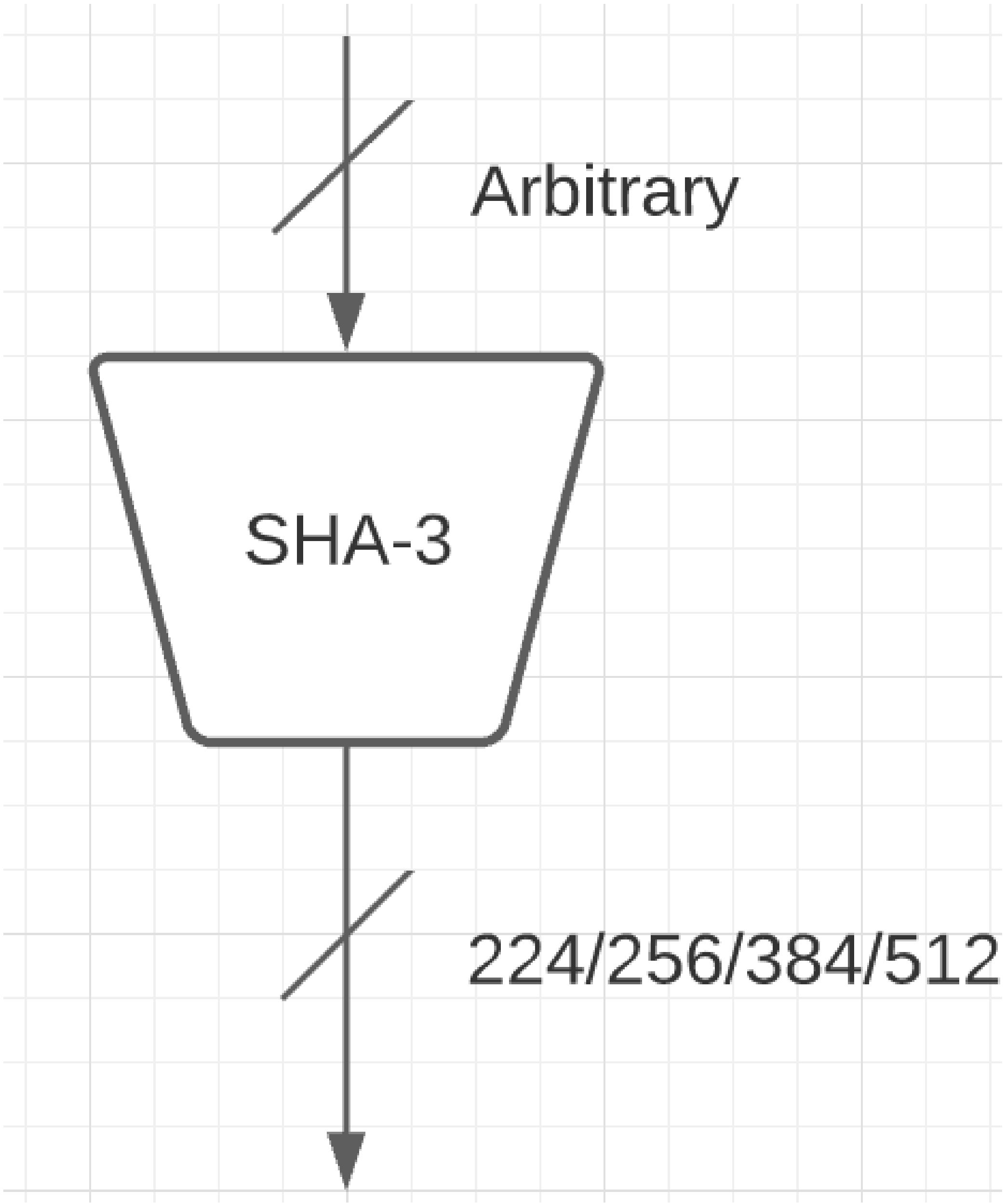


# SHA-3 Competition by NIST

Guido Bertoni,
Joan Daemen,
Michaël Peeters,
and Gilles Van Assche

- **Nov. 2007** — Call for algorithms
- **Oct. 2008** — 64 submissions
- **Dec. 2010** — 5 algorithm finalist's were chosen
- **2 Oct. 2012** — Keccak is selected as SHA-3

- Arbitrary Input length
- 4 Output lengths.



# Collision Resistance

| SHA-3 256 | SHA-3 384 | SHA-3 512 | SHA-3 224 |
|---|---|---|---|
| Collision resistance<br>:<br>128 | Collision resistance<br>:<br>192 | Collision resistance<br>:<br>256 | Collision resistance<br>:<br>112 |
| **AES-128** | **AES-192** | **AES-256** | **3DES: effective security 112** |

$$t \approx 2^{(n+1)/2}\sqrt{\ln\left(\frac{1}{1-\lambda}\right)}.$$

Collision $\approx 2^{n/2}$

# SHA-3 Parameters

- State Size 1600 bits
- 24 Rounds
- Block size & Capacity Dependent on the SHA-3 bit implementation.
- b = r + c

| Output | b (State) | r (block size) | c (capacity) |
|---|---|---|---|
| 224 | 1600 | 1152 | 448 |
| 256 | 1600 | 1088 | 512 |
| 384 | 1600 | 832 | 768 |
| 512 | 1600 | 576 | 1024 |

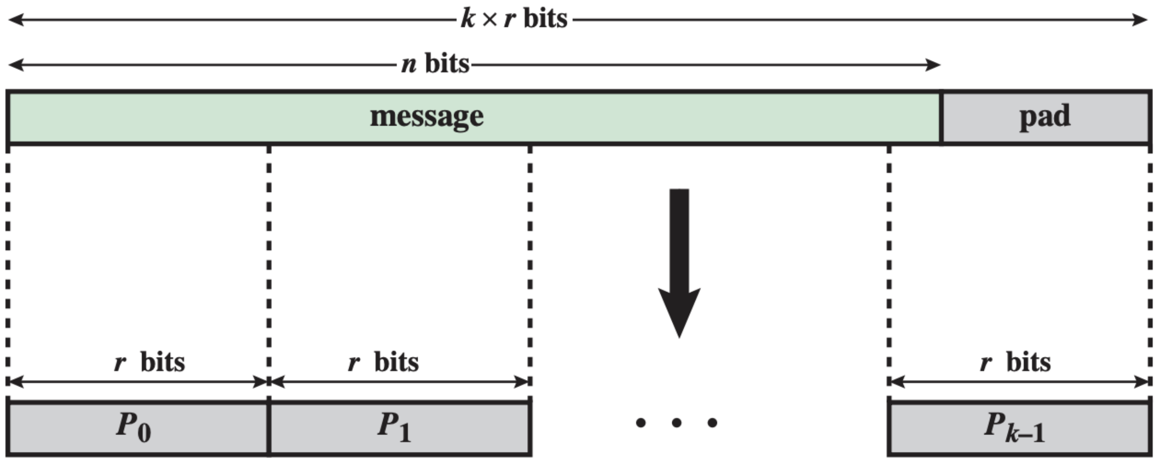
# High Level View of Keccak

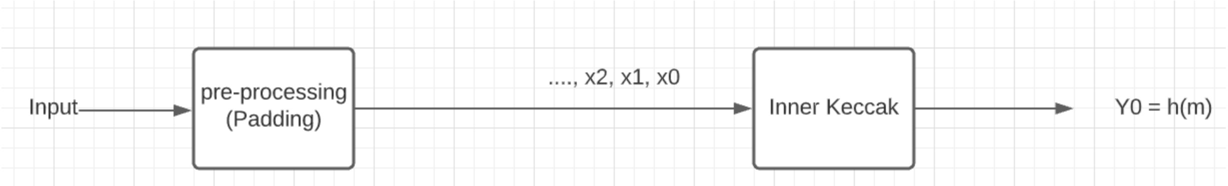
Sponge construction

- 1. Absorbing Phase: Input $x_i$ is read-in & processed. Where $x_i$ the is block size.
- 2. Squeezing Phase: Output is produced

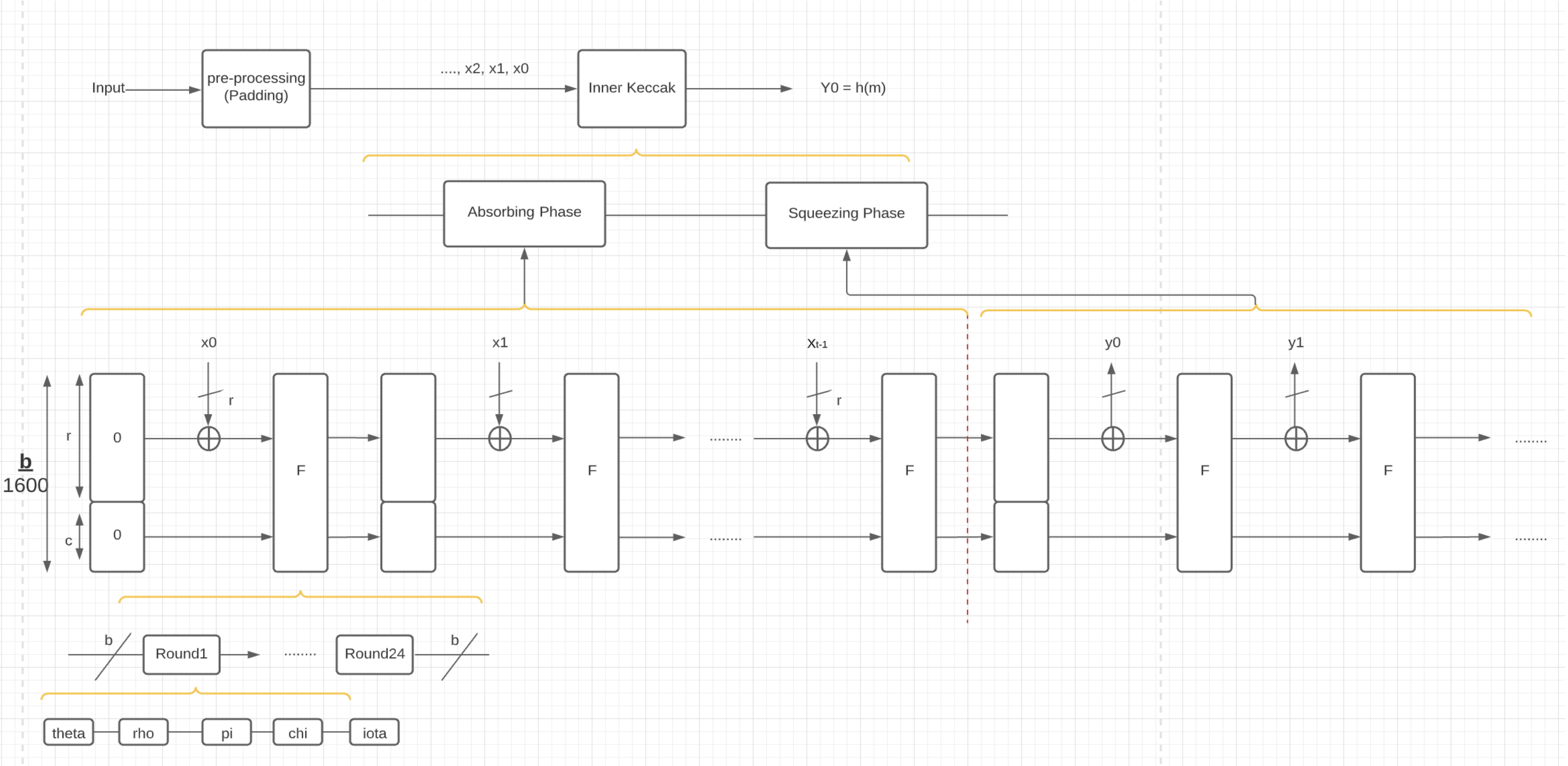
# High Level

Padding to make sure input will conform to block size



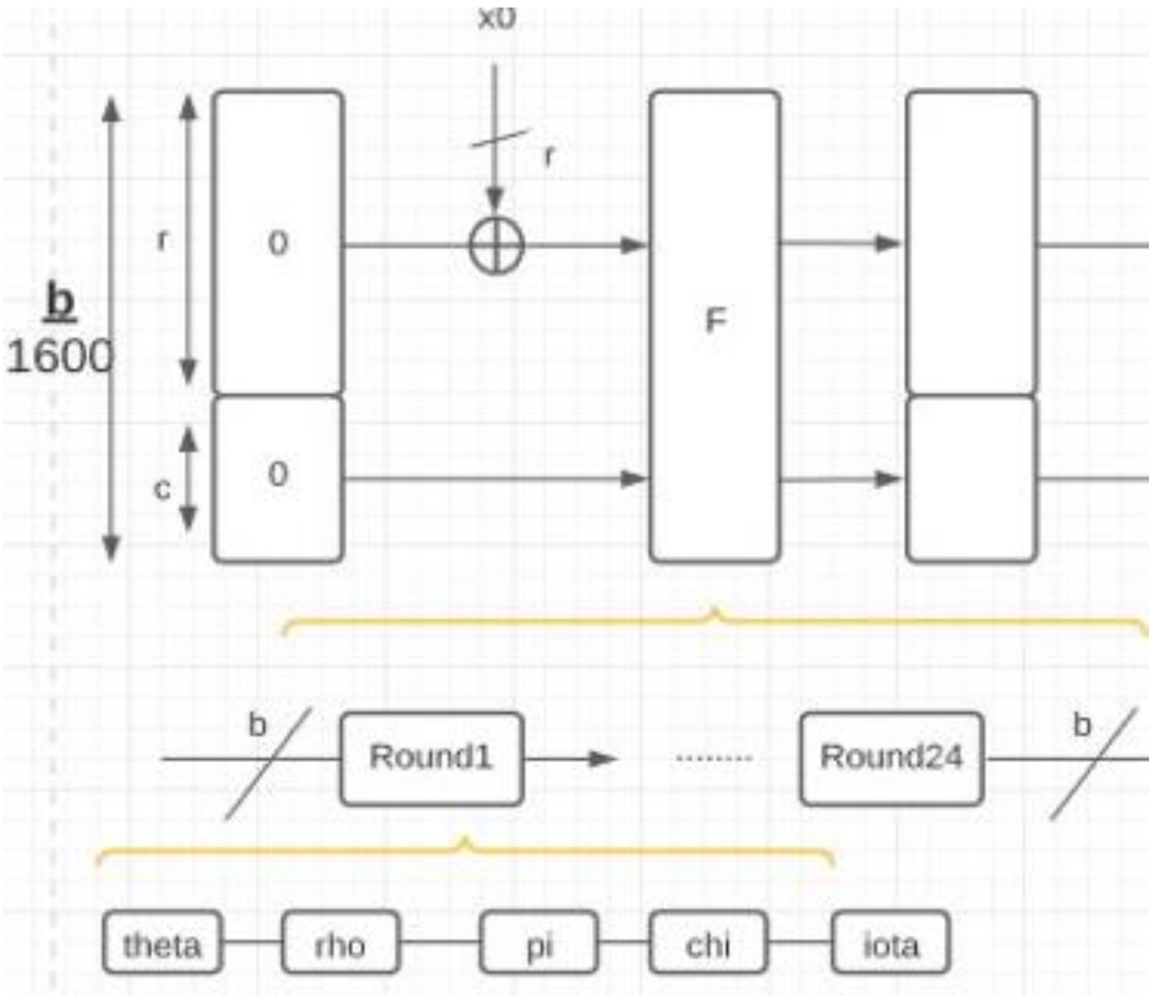



# High-Level Model

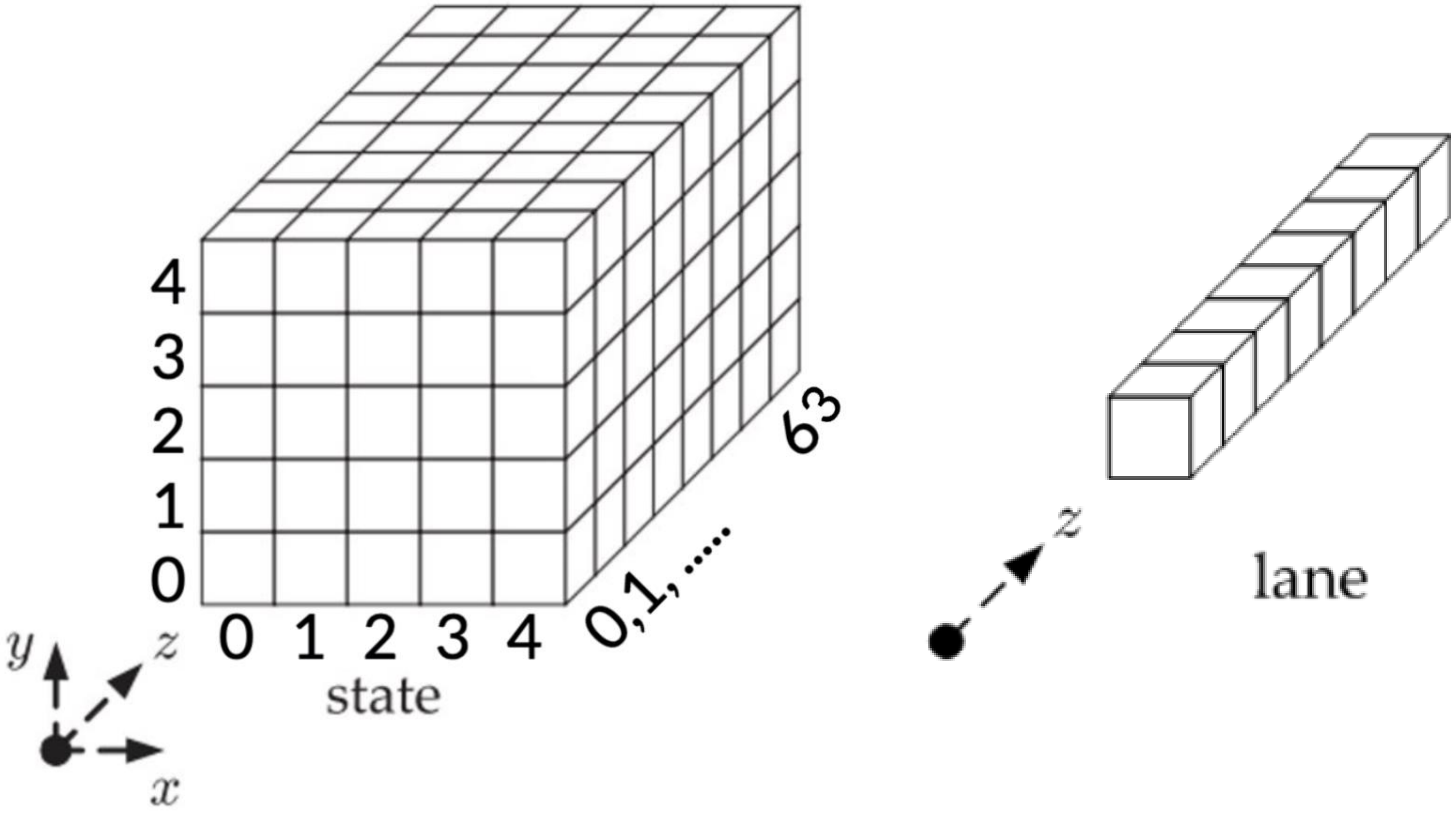


# F-Function

- 24 Rounds
- 5 subfunctions
- Theta (θ)
- Rho (ρ)
- Pi (π)
- Chi (χ)
- Iota (ι)

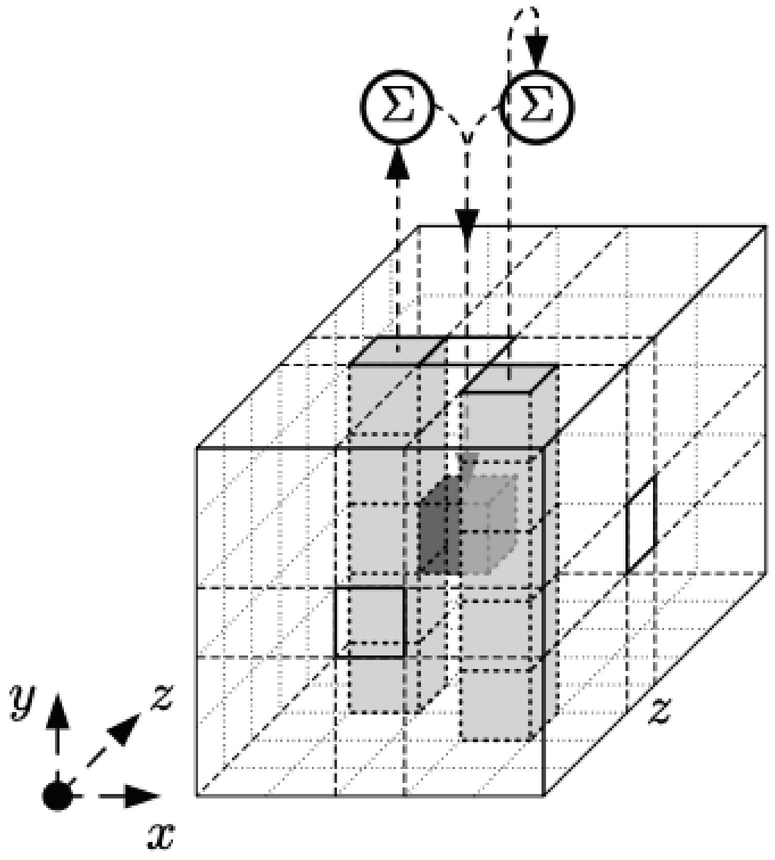


# State 3-D Array



# Theta Step

$$
\begin{aligned}
C[x] &= A[x,0] \oplus A[x,1] \oplus A[x,2] \oplus A[x,3] \oplus A[x,4] && ,\ x = 0,1,2,3,4 \\
D[x] &= C[x-1] \oplus \mathrm{rot}(C[x+1],1) && ,\ x = 0,1,2,3,4 \\
A[x,y] &= A[x,y] \oplus D[x] && ,\ x,y = 0,1,2,3,4
\end{aligned}
$$



# Rho Step

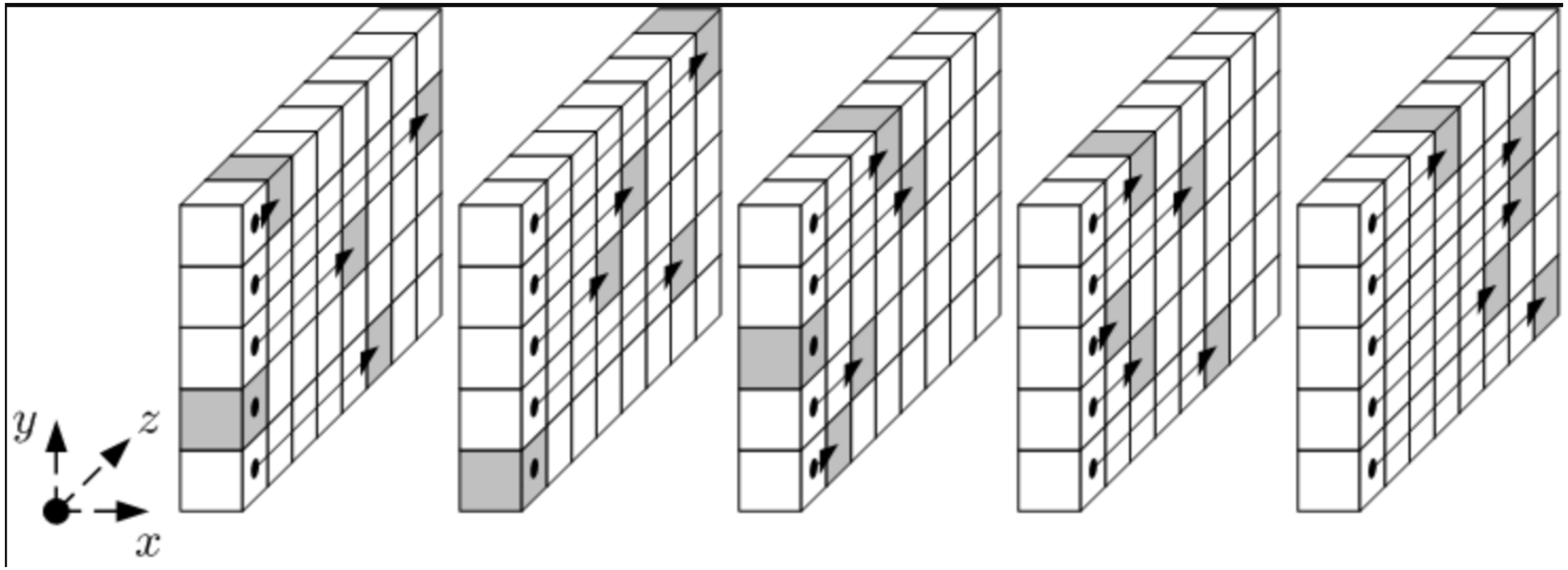
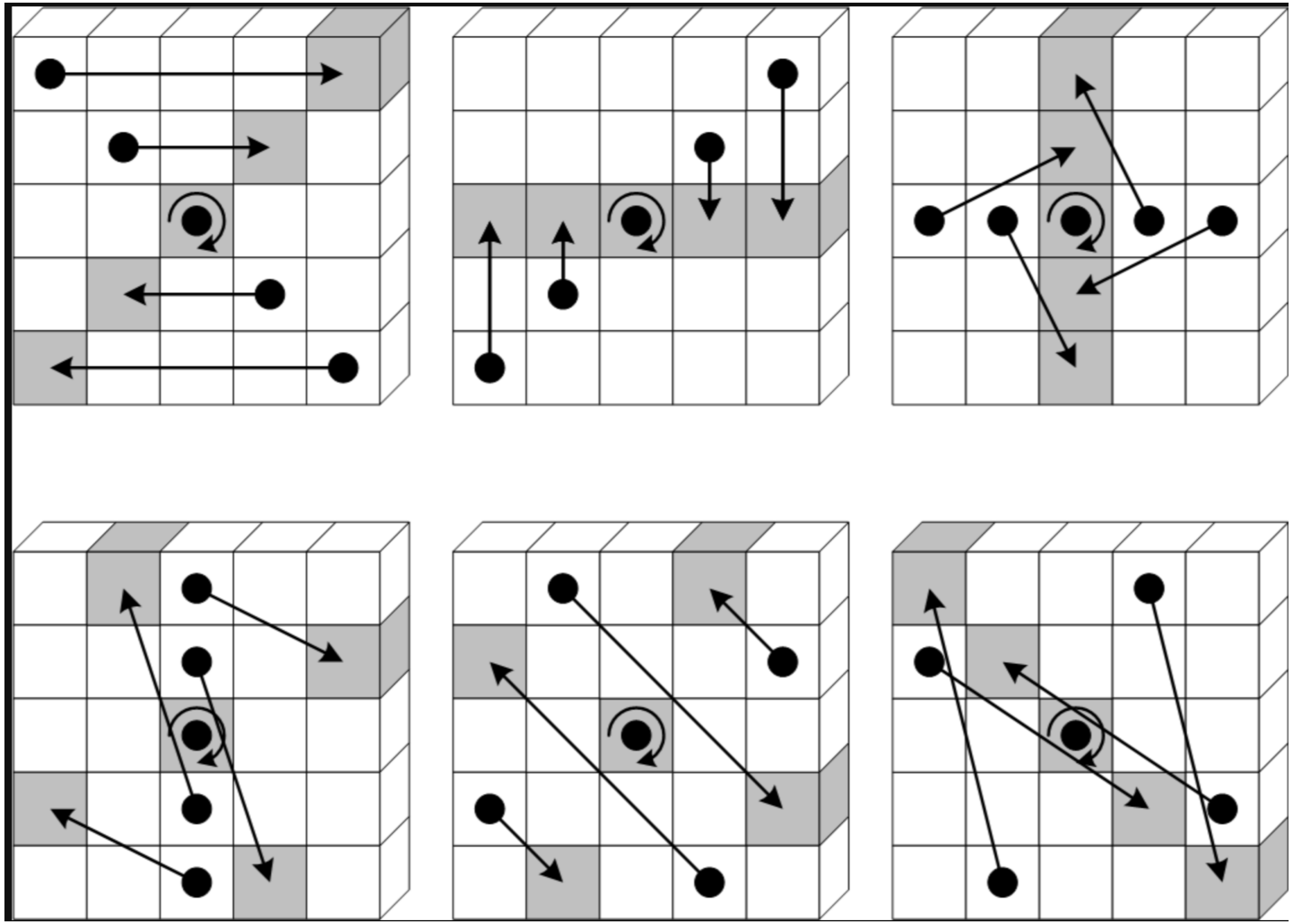



**Table 1.4** The rotation constants (aka rotation offsets)

| | x = 3 | x = 4 | x = 0 | x = 1 | x = 2 |
|---|---|---|---|---|---|
| y=2 | 25 | 39 | 3 | 10 | 43 |
| y=1 | 55 | 20 | 36 | 44 | 6 |
| y=0 | 28 | 27 | 0 | 1 | 62 |
| y=4 | 56 | 14 | 18 | 2 | 61 |
| y=3 | 21 | 8 | 41 | 45 | 15 |

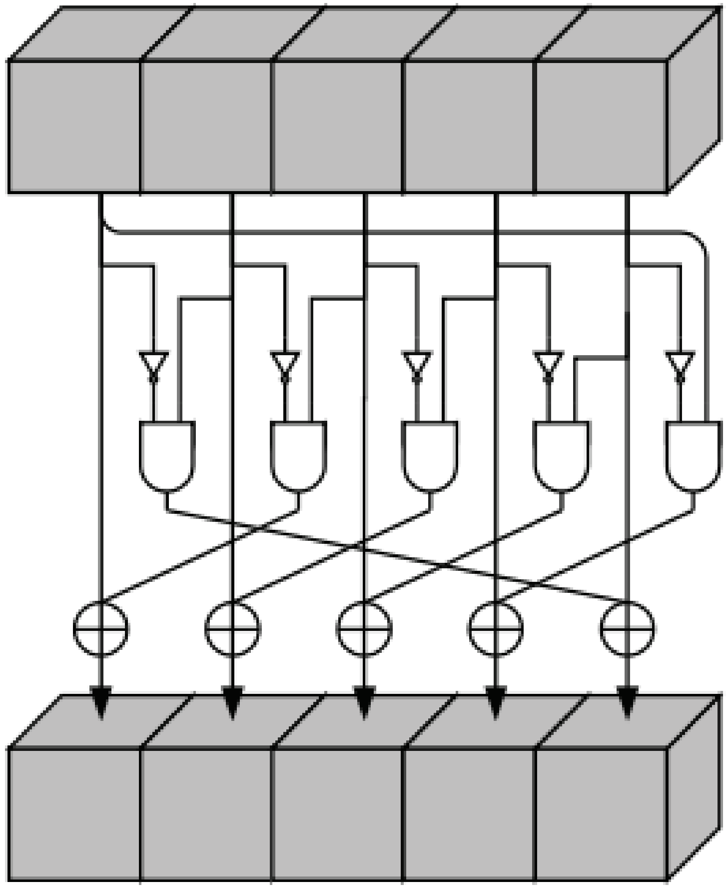
# Pi Step



$$B[y, 2x+3y] = \mathrm{rot}(A[x,y], r[x,y]) \quad , \quad x, y = 0,1,2,3,4$$

# Chi Step

B¯[i, j] denotes the bitwise complement of the lane at address [i, j], and ∧ is the bitwise Boolean AND operation of the two operands.



$$A[x,y] = B[x,y] \oplus ((\bar{B}[x+1,y]) \wedge B[x+2,y]) \quad , \quad x,y = 0,1,2,3,4$$

# Iota Step

$$A[0,0] = A[0,0] \oplus RC[i]$$

**Table 1.5** The round constants RC[i], where each constant is 64 bits long and given in hexadecimal notation

| | | | |
|---|---|---|---|
| RC[ 0] | = 0x0000000000000001 | RC[12] | = 0x000000008000808B |
| RC[ 1] | = 0x0000000000008082 | RC[13] | = 0x800000000000008B |
| RC[ 2] | = 0x800000000000808A | RC[14] | = 0x8000000000008089 |
| RC[ 3] | = 0x8000000080008000 | RC[15] | = 0x8000000000008003 |
| RC[ 4] | = 0x000000000000808B | RC[16] | = 0x8000000000008002 |
| RC[ 5] | = 0x0000000080000001 | RC[17] | = 0x8000000000000080 |
| RC[ 6] | = 0x8000000080008081 | RC[18] | = 0x000000000000800A |
| RC[ 7] | = 0x8000000000008009 | RC[19] | = 0x800000008000000A |
| RC[ 8] | = 0x000000000000008A | RC[20] | = 0x8000000080008081 |
| RC[ 9] | = 0x0000000000000088 | RC[21] | = 0x8000000000008080 |
| RC[10] | = 0x0000000080008009 | RC[22] | = 0x0000000080000001 |
| RC[11] | = 0x000000008000000A | RC[23] | = 0x8000000080008008 |

# Resources Used


- Understanding Cryptography: A Textbook for Students and Practitioners, Christof Paar, Jan Pelz, Springer; 1st Edition, July 8, 2010.
- https://www.crypto-textbook.com/download/Understanding-Cryptography-Keccak.pdf
- https://keccak.team/figures.html