PUFacademy

Founded by PUFsecurity

Cryptographic Hash Algorithms .

Danny Chen

2022 November

Outline

1. SHA-2 Algorithm
2. SHA-3 Algorithm
3. HMAC Algorithm
4. KDF Algorithm
5. Summary

PUF academy

Outline

1. **SHA-2 Algorithm**
2. SHA-3 Algorithm
3. HMAC Algorithm
4. KDF Algorithm
5. Summary

PUF academy

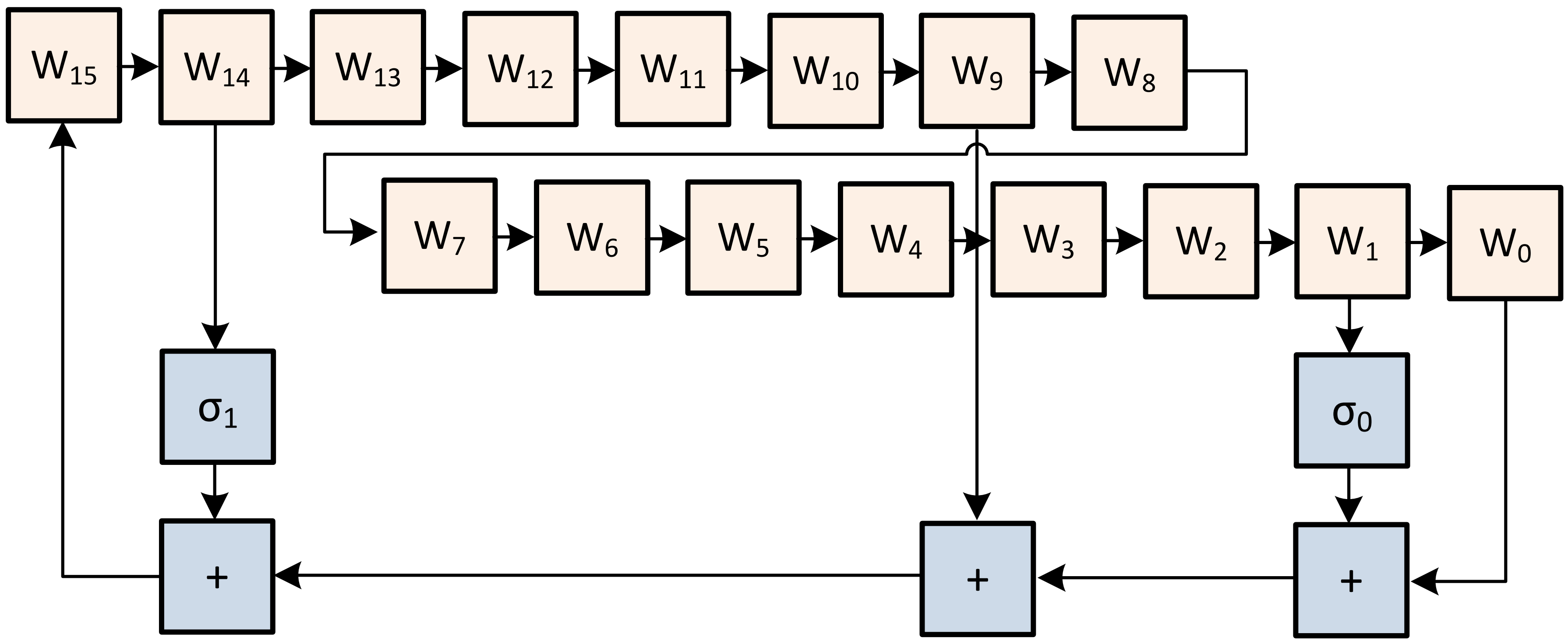
Secure Hash Algorithm

Name	Digest bits	Block bits	Collision	First Published
MD5	128	512	Yes	1992
SHA-0	160	512	Yes	1993
SHA-1	160	512	Yes	1995
RIPEMD	128	512	Yes	1996
SHA-2	224/256	512	No	2001
SHA-2	384/512	1024	No	2001
SHA-3	224/256/ 384/512/ arbitrary	1152/1088/ 832/576/ 1344	No	2015

SHA-2

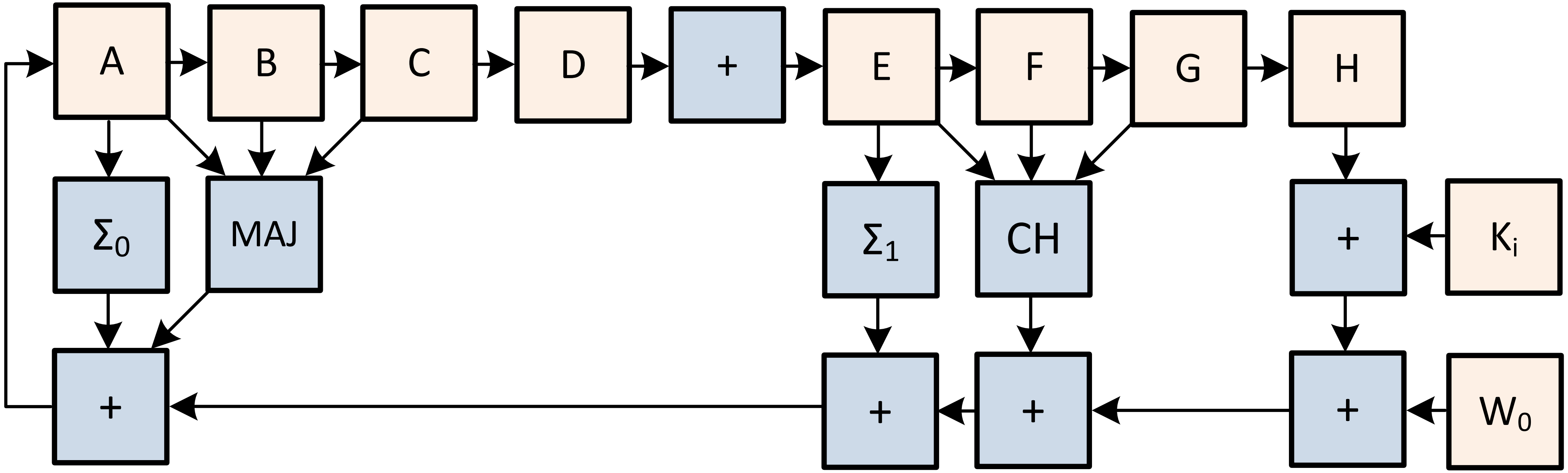
- SHA-2 was first published in 2001.
- The SHA-2 family consists of 6 hash functions with digests that are SHA224, SHA256, SHA384, SHA512, SHA512/224, SHA512/256

SHA-2 Expansion



$$\sigma_0^{\{256\}}(x) = ROTR^{7}(x) \oplus ROTR^{18}(x) \oplus SHR^{3}(x)$$
$$\sigma_1^{\{256\}}(x) = ROTR^{17}(x) \oplus ROTR^{19}(x) \oplus SHR^{10}(x)$$

SHA-2 Compression

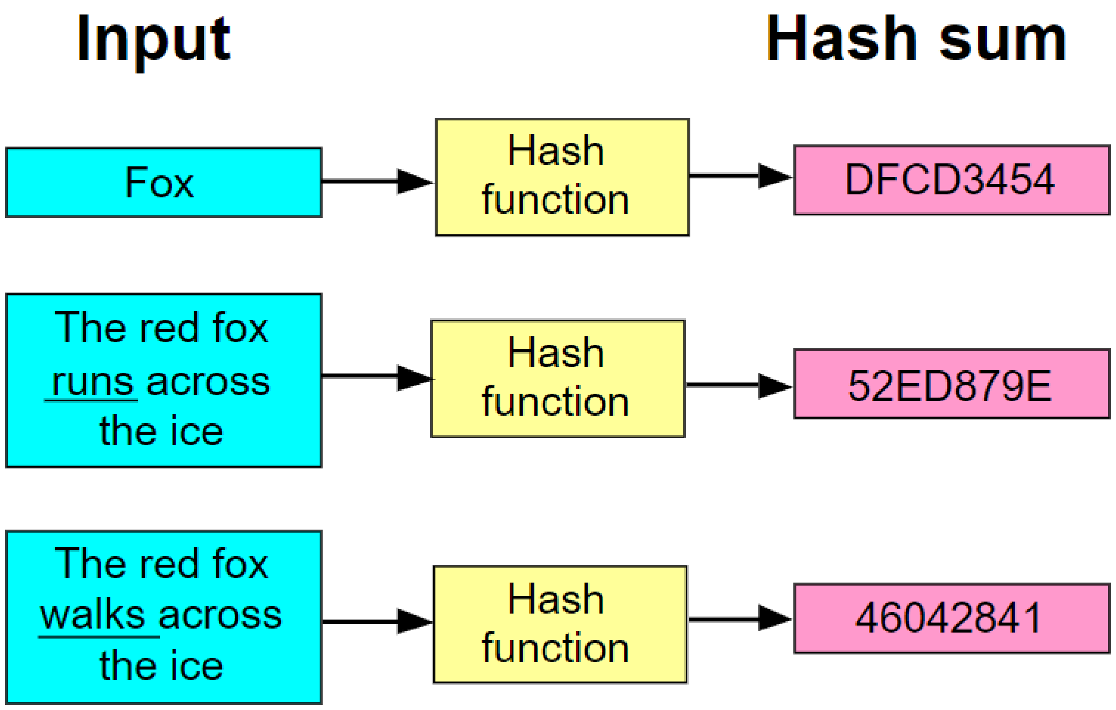


$$Ch(x,y,z) = (x \wedge y) \oplus (\neg x \wedge z)$$
$$Maj(x,y,z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z)$$

$$\Sigma_0^{\{256\}}(x) = ROTR^2(x) \oplus ROTR^{13}(x) \oplus ROTR^{22}(x)$$
$$\Sigma_1^{\{256\}}(x) = ROTR^6(x) \oplus ROTR^{11}(x) \oplus ROTR^{25}(x)$$

Avalanche effect

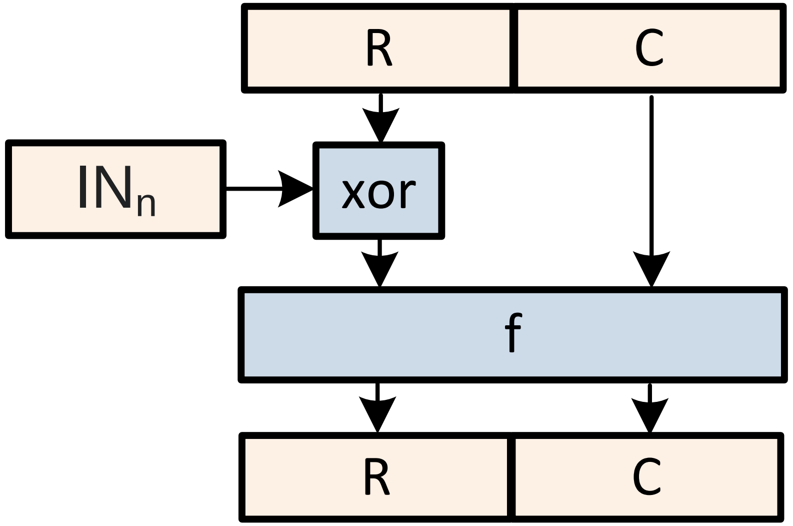
- If the input changes slightly, the output will change significantly.

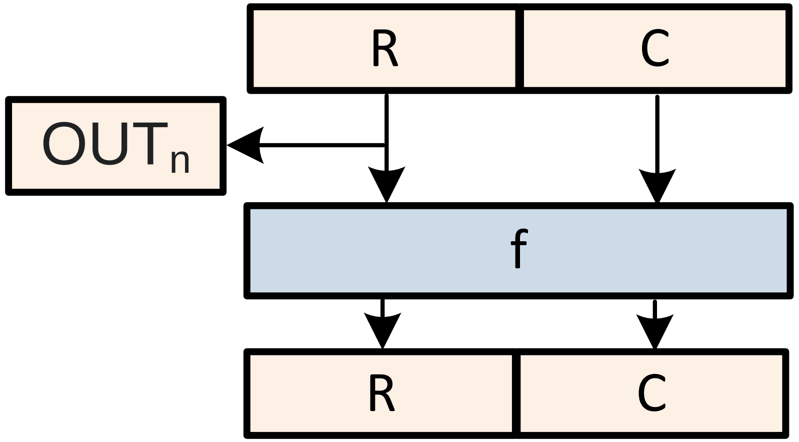


(ref. https://zh.wikipedia.org/wiki/%E6%95%A3%E5%88%97%E5%87%BD%E6%95%B8)

Sponge Function

- State: containing b bits
  - R: a positive number that is less than width b
  - C: The capacity, a positive number b-R





Keccak

- Keccak (/ˈkɛtʃæk/ or /ˈkɛtʃɑːk/) is a family of sponge functions
- Pad function: pad10*1 = *msg(with suffix) || 1 || 0 ^{j} || 1* is a positive multiple of block size

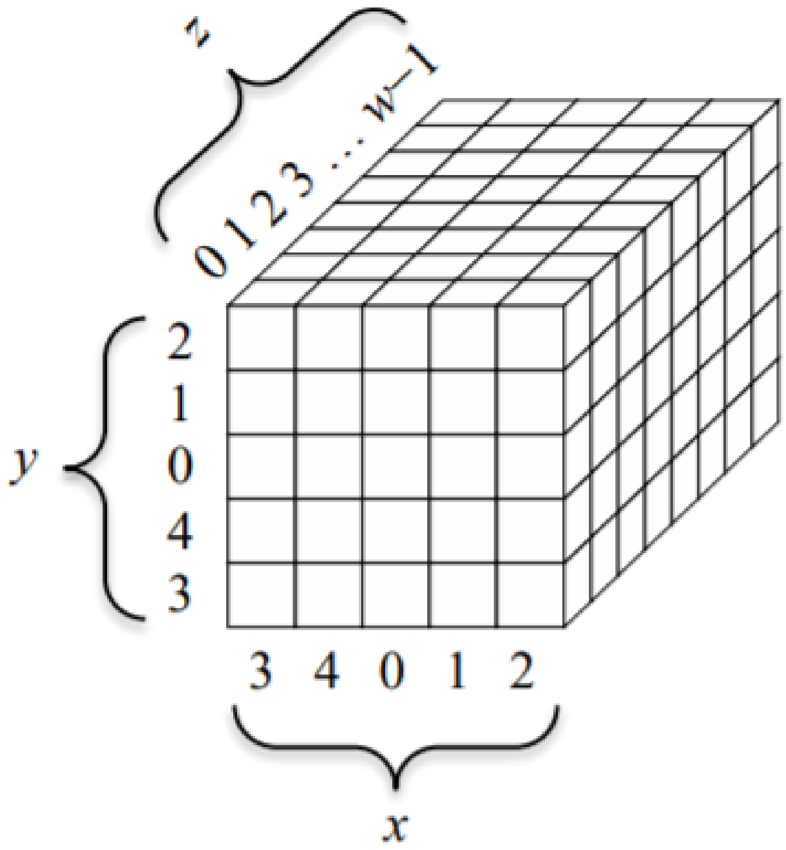
Input:
positive integer x;
non-negative integer m.

Output:
string P such that $m + \text{len}(P)$ is a positive multiple of x.

Steps:
1. Let $j = (-m - 2) \bmod x$.
2. Return $P = 1 \,||\, 0^j \,||\, 1$.

3D Array

- It is defined for word size, $w = 2^{\ell}$ bits(main method uses 64-bit, $\ell = 6$)
- Convert string into the state array A
- For i_r from 1 to $12 + 2\ell$ rounds of five steps:
  - θ (theta)
  - ρ (rho)
  - π (pi)
  - χ (chi)
  - ι (iota)
- Convert the state array A into a string



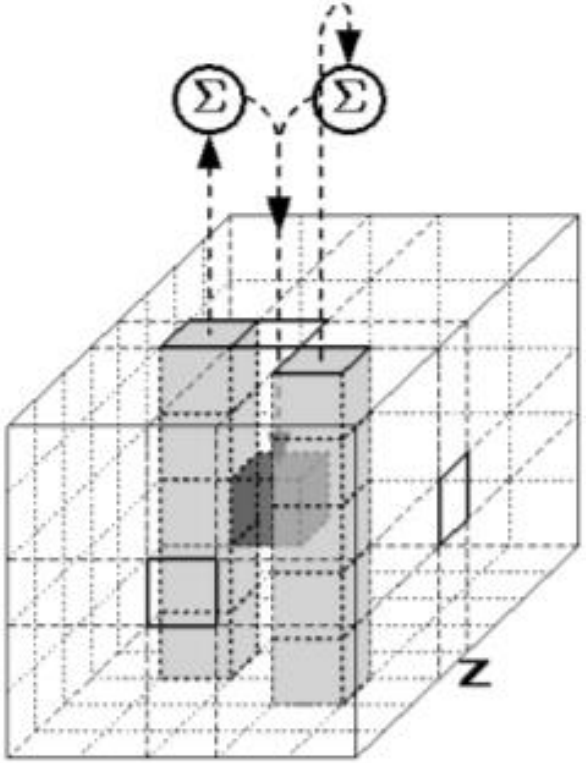
3D Array

- Converting a string of b(1600=5*5*64) bits into the state array

$$
\begin{aligned}
S = \; & \mathbf{A}[0,0,0] \,\|\, \mathbf{A}[0,0,1] \,\|\, \mathbf{A}[0,0,2] \,\|\, \ldots \,\|\, \mathbf{A}[0,0,62] \,\|\, \mathbf{A}[0,0,63] \\
\| \, & \mathbf{A}[1,0,0] \,\|\, \mathbf{A}[1,0,1] \,\|\, \mathbf{A}[1,0,2] \,\|\, \ldots \,\|\, \mathbf{A}[1,0,62] \,\|\, \mathbf{A}[1,0,63] \\
\| \, & \mathbf{A}[2,0,0] \,\|\, \mathbf{A}[2,0,1] \,\|\, \mathbf{A}[2,0,2] \,\|\, \ldots \,\|\, \mathbf{A}[2,0,62] \,\|\, \mathbf{A}[2,0,63] \\
\| \, & \mathbf{A}[3,0,0] \,\|\, \mathbf{A}[3,0,1] \,\|\, \mathbf{A}[3,0,2] \,\|\, \ldots \,\|\, \mathbf{A}[3,0,62] \,\|\, \mathbf{A}[3,0,63] \\
\| \, & \mathbf{A}[4,0,0] \,\|\, \mathbf{A}[4,0,1] \,\|\, \mathbf{A}[4,0,2] \,\|\, \ldots \,\|\, \mathbf{A}[4,0,62] \,\|\, \mathbf{A}[4,0,63] \\[1em]
\| \, & \mathbf{A}[0,1,0] \,\|\, \mathbf{A}[0,1,1] \,\|\, \mathbf{A}[0,1,2] \,\|\, \ldots \,\|\, \mathbf{A}[0,1,62] \,\|\, \mathbf{A}[0,1,63] \\
\| \, & \mathbf{A}[1,1,0] \,\|\, \mathbf{A}[1,1,1] \,\|\, \mathbf{A}[1,1,2] \,\|\, \ldots \,\|\, \mathbf{A}[1,1,62] \,\|\, \mathbf{A}[1,1,63] \\
\| \, & \mathbf{A}[2,1,0] \,\|\, \mathbf{A}[2,1,1] \,\|\, \mathbf{A}[2,1,2] \,\|\, \ldots \,\|\, \mathbf{A}[2,1,62] \,\|\, \mathbf{A}[2,1,63] \\
\| \, & \mathbf{A}[3,1,0] \,\|\, \mathbf{A}[3,1,1] \,\|\, \mathbf{A}[3,1,2] \,\|\, \ldots \,\|\, \mathbf{A}[3,1,62] \,\|\, \mathbf{A}[3,1,63] \\
\| \, & \mathbf{A}[4,1,0] \,\|\, \mathbf{A}[4,1,1] \,\|\, \mathbf{A}[4,1,2] \,\|\, \ldots \,\|\, \mathbf{A}[4,1,62] \,\|\, \mathbf{A}[4,1,63] \\[1em]
& \qquad\qquad\qquad \vdots \\[1em]
\| \, & \mathbf{A}[0,4,0] \,\|\, \mathbf{A}[0,4,1] \,\|\, \mathbf{A}[0,4,2] \,\|\, \ldots \,\|\, \mathbf{A}[0,4,62] \,\|\, \mathbf{A}[0,4,63] \\
\| \, & \mathbf{A}[1,4,0] \,\|\, \mathbf{A}[1,4,1] \,\|\, \mathbf{A}[1,4,2] \,\|\, \ldots \,\|\, \mathbf{A}[1,4,62] \,\|\, \mathbf{A}[1,4,63] \\
\| \, & \mathbf{A}[2,4,0] \,\|\, \mathbf{A}[2,4,1] \,\|\, \mathbf{A}[2,4,2] \,\|\, \ldots \,\|\, \mathbf{A}[2,4,62] \,\|\, \mathbf{A}[2,4,63] \\
\| \, & \mathbf{A}[3,4,0] \,\|\, \mathbf{A}[3,4,1] \,\|\, \mathbf{A}[3,4,2] \,\|\, \ldots \,\|\, \mathbf{A}[3,4,62] \,\|\, \mathbf{A}[3,4,63] \\
\| \, & \mathbf{A}[4,4,0] \,\|\, \mathbf{A}[4,4,1] \,\|\, \mathbf{A}[4,4,2] \,\|\, \ldots \,\|\, \mathbf{A}[4,4,62] \,\|\, \mathbf{A}[4,4,63]
\end{aligned}
$$

N_{th} word	x=3	x=4	x=0	x=1	x=2
y=2	13	14	10	11	12
y=1	8	9	5	6	7
y=0	3	4	0	1	2
y=4	18	19	15	16	17
y=3	23	24	20	21	22

θ (theta) step



θ step

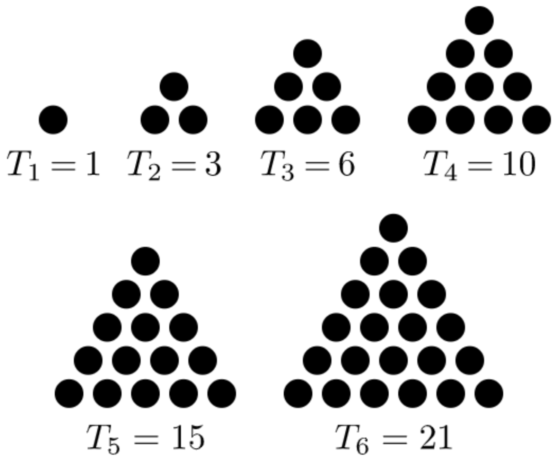
$$C[x] = A[x, 0] \oplus A[x, 1] \oplus A[x, 2] \oplus A[x, 3] \oplus A[x, 4], \quad \forall x \text{ in } 0 \ldots 4$$

$$D[x] = C[x - 1] \oplus ROT(C[x + 1], 1), \quad \forall x \text{ in } 0 \ldots 4$$

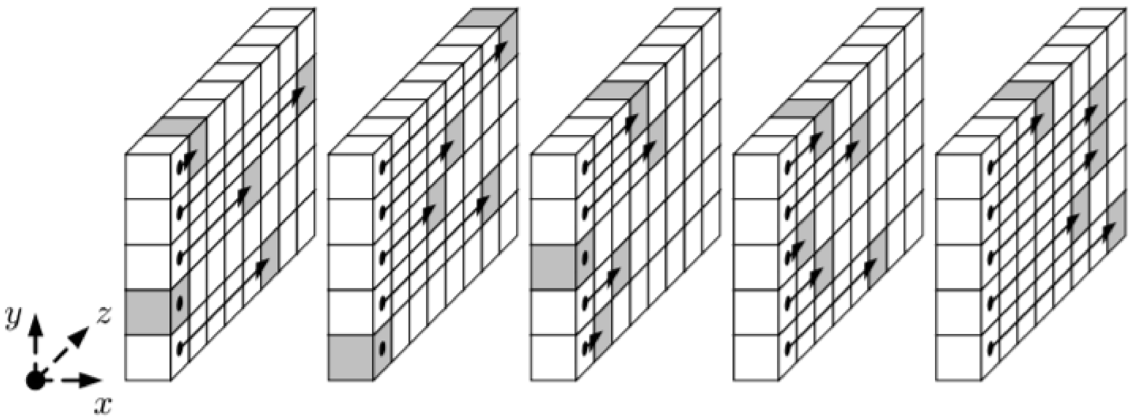
$$A[x, y] = A[x, y] \oplus D[x], \quad \forall (x, y) \text{ in } (0 \ldots 4, 0 \ldots 4)$$

ρ (rho) step

- Bit rotate each the 25 words by different triangle number
- A′[x,y,z] = A[x, y, (z–(t+1)(t+2)/2) mod w]



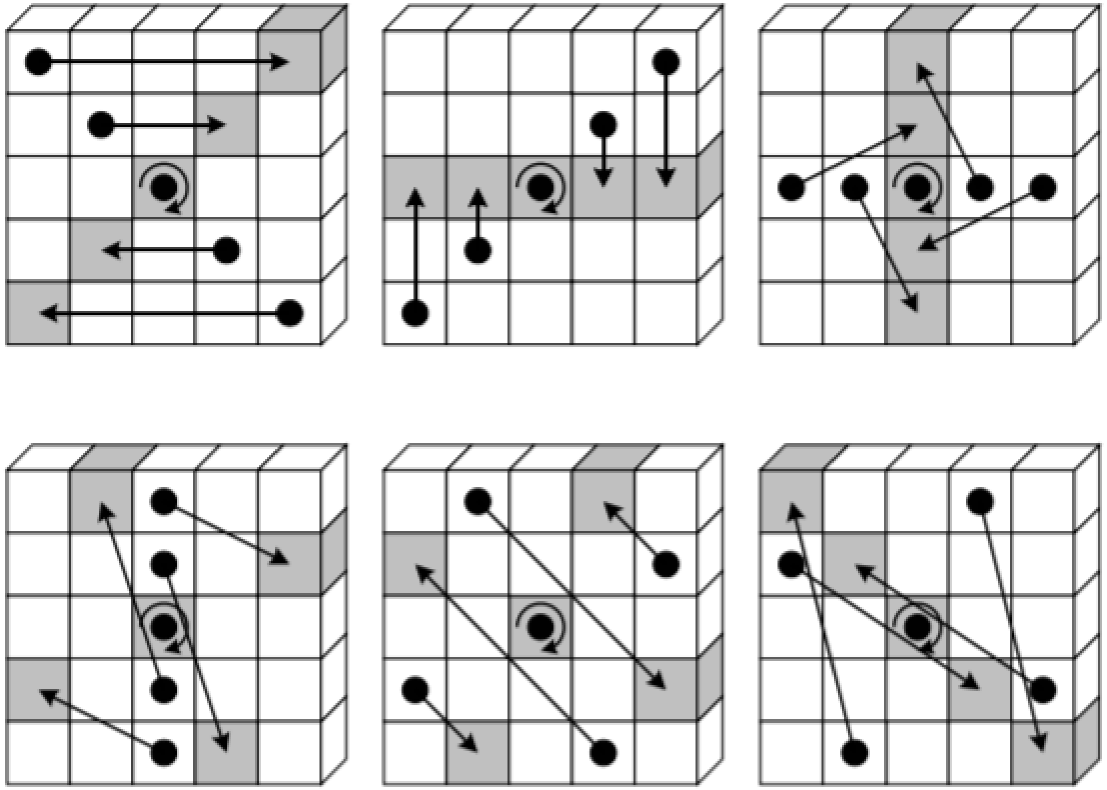
	x=3	x=4	x=0	x=1	x=2
y=2	153	231	3	10	171
y=1	55	276	36	300	6
y=0	28	91	0	1	190
y=4	120	78	210	66	253
y=3	21	136	105	45	15



π (pi) step

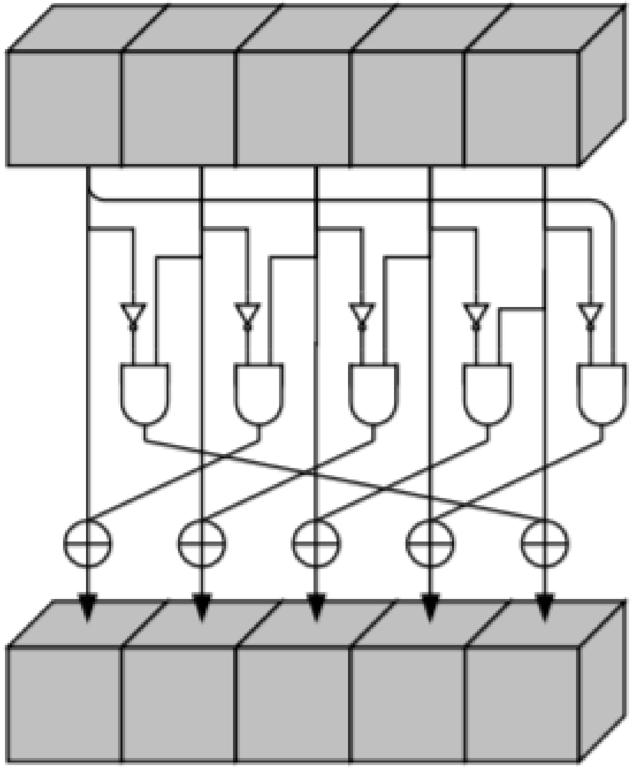
- Permutation the 25 words
- A′[x,y,z] = A[(x + 3y) mod 5, x, z]

	x=3	x=4	x=0	x=1	x=2
y=2	17	21	2	4	18
y=1	10	23	8	24	3
y=0	7	13	X	1	19
y=4	15	12	20	11	22
y=3	6	16	14	9	5



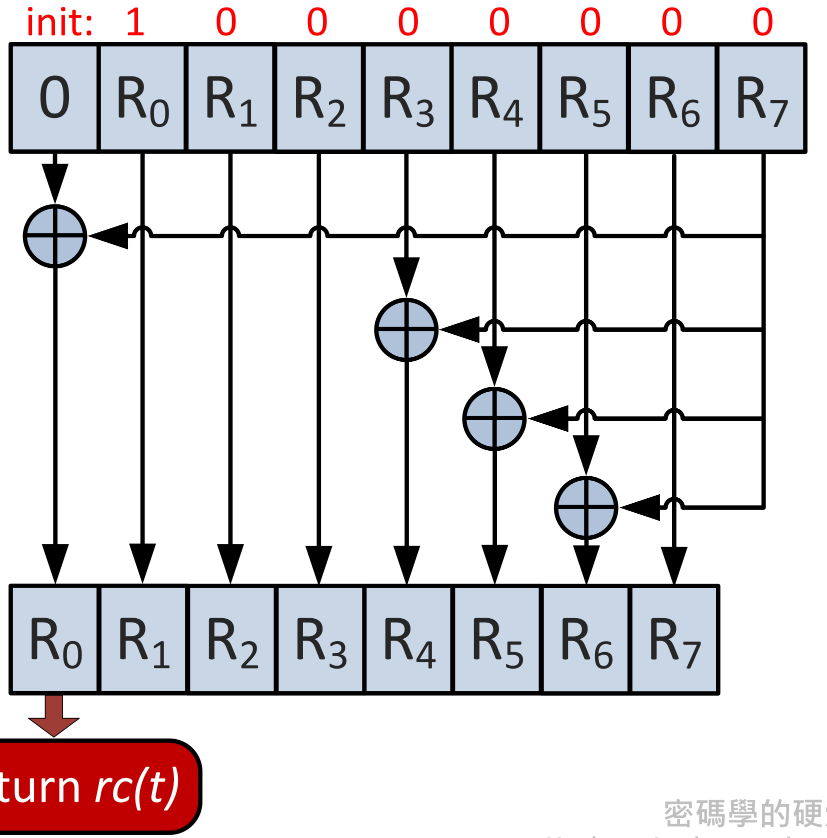
χ (chi) step

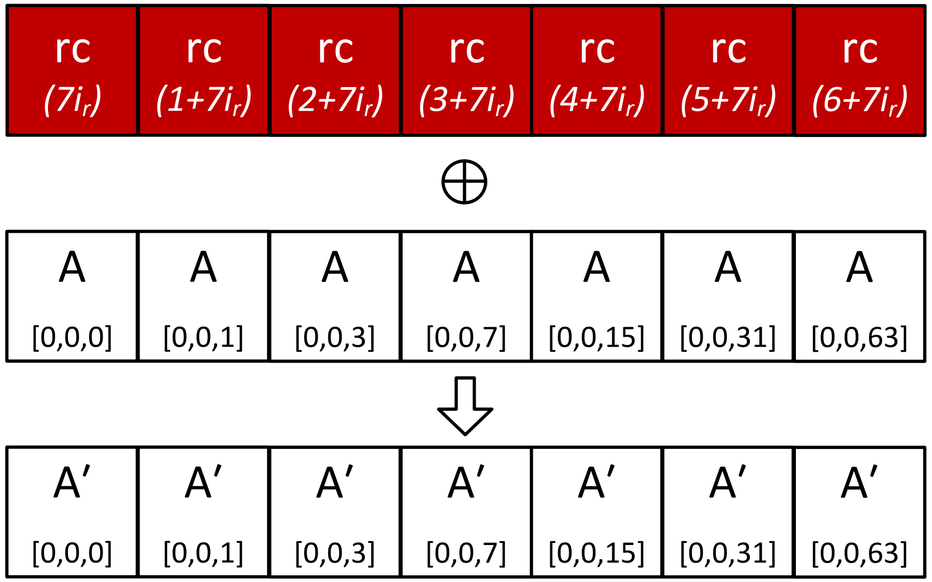
- Bitwise combine along rows
- $A'[x,y,z] = A[x, y,z] \oplus ((A[(x+1) \bmod 5, y, z] \oplus 1) \cdot A[(x+2) \bmod 5, y, z])$



ι (iota) step

- Bitwise combine along rows
- A′[x,y,z] = A[x, y,z] ⊕ ((A[(x+1) mod 5, y, z] ⊕ 1) · A[(x+2) mod 5, y, z])





Outline

1. SHA-2 Algorithm
2. **SHA-3 Algorithm**
3. HMAC Algorithm
4. KDF Algorithm
5. Summary

SHA-3

- In 2006, NIST started to organize NIST hash function competition to create a new hash standard
- In 2008, the Keccak algorithm was accepted as one of the 51 candidates. It advanced to the last round in 2010
- In 2012, Keccak was permitted to improve its performance
- In 2015, NIST announced that SHA-3 had become a hashing standard

$$
\begin{aligned}
&\text{SHA3-224}(M) = \text{KECCAK}[448]\,(M \,\|\, 01, 224);\\
&\text{SHA3-256}(M) = \text{KECCAK}[512]\,(M \,\|\, 01, 256);\\
&\text{SHA3-384}(M) = \text{KECCAK}[768]\,(M \,\|\, 01, 384);\\
&\text{SHA3-512}(M) = \text{KECCAK}[1024]\,(M \,\|\, 01, 512).\\
&\text{SHAKE128}(M, d) = \text{KECCAK}[256]\,(M \,\|\, 1111, d),\\
&\text{SHAKE256}(M, d) = \text{KECCAK}[512]\,(M \,\|\, 1111, d).
\end{aligned}
$$

Outline

1. SHA-2 Algorithm
2. SHA-3 Algorithm
3. **HMAC Algorithm**
4. KDF Algorithm
5. Summary

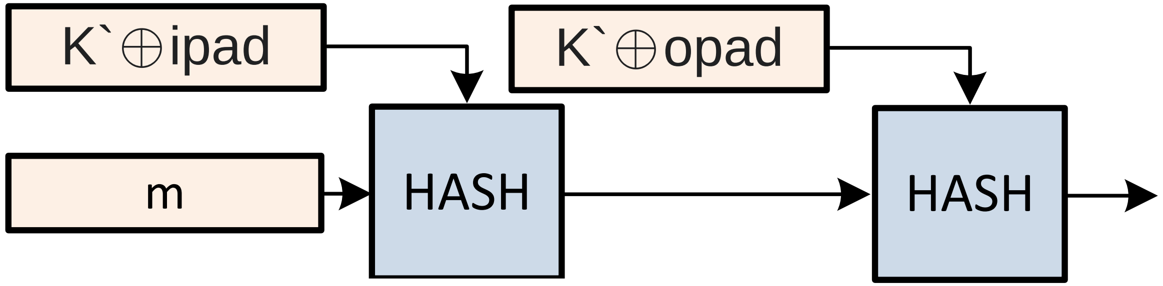
PUF academy

HMAC

- In 1996, HMAC construction was first published

$$\mathrm{HMAC}(K,m) = \mathrm{H}\Big(\big(K' \oplus opad\big) \parallel \mathrm{H}\big(\big(K' \oplus ipad\big) \parallel m\big)\Big)$$

$$K' = \begin{cases} \mathrm{H}(K) & K \text{ is larger than block size} \\ K & \text{otherwise} \end{cases}$$



H is a cryptographic hash function

m is the message to be authenticated

K is the secret key

K' is a block-sized key derived from the secret key, *K*; either by padding to the right with 0s up to the block size,
or by hashing down to less than or equal to the
block size first and then padding to the right with zeros

|| denotes concatenation

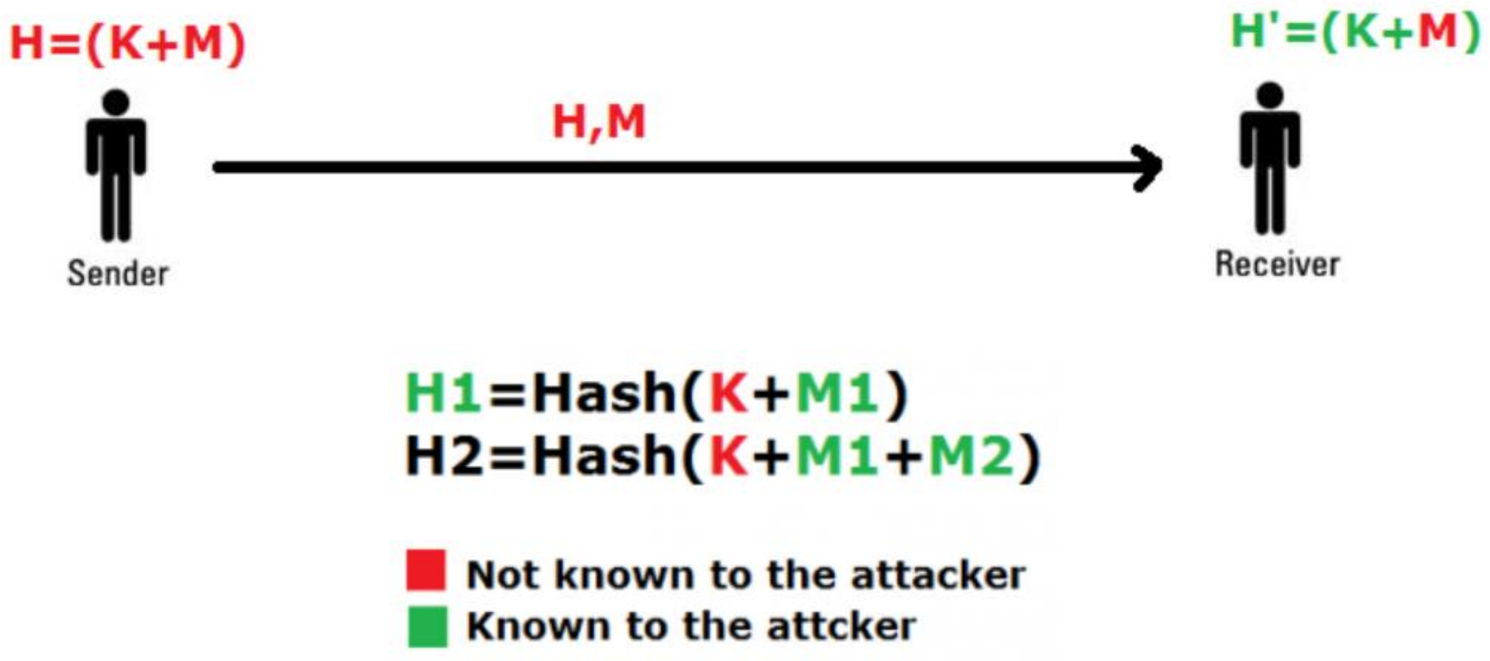
⊕ denotes bitwise exclusive or (XOR)

opad is the block-sized outer padding, consisting of repeated bytes valued 0x5c

ipad is the block-sized inner padding, consisting of repeated bytes valued 0x36

Length Extension Attack

- Attacker can use *hash(msg1)* and length of *msg1* to calculate *hash(msg1||msg2)* for an attacker-controlled *msg2*



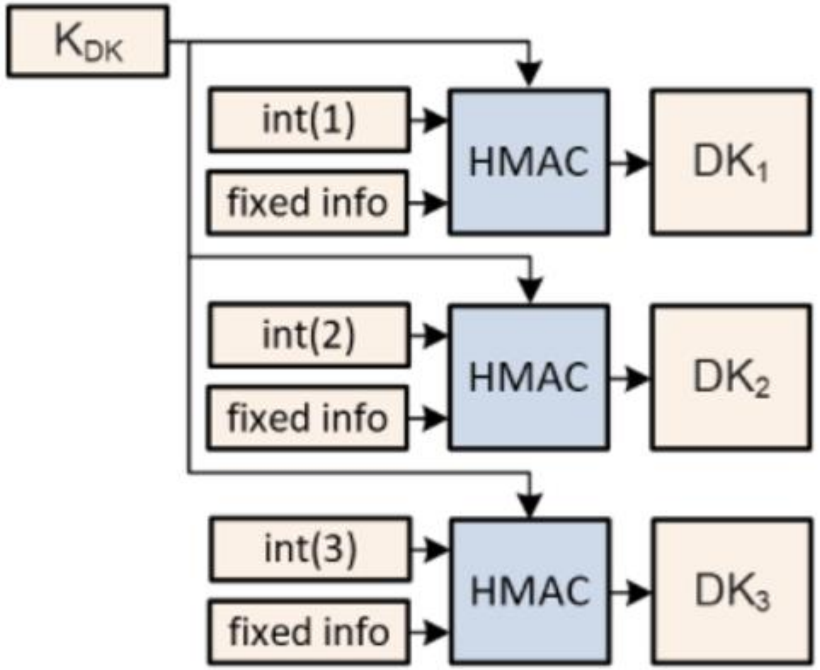
(ref. https://www.roguesecurity.in/2017/05/14/length-extension-attack-and-how-it-can-be-exploited/)

Outline

1. SHA-2 Algorithm
2. SHA-3 Algorithm
3. HMAC Algorithm
4. **KDF Algorithm**
5. Summary

Key Derivation Function

- In cryptography, keys are often indispensable
- KDF needs to prepare a master key or password or share secret and use the pseudo random function to derive one or more keys.



HKDF .

- To extract
- To expand

$$HKDF\_extract\,(S, SS) = HMAC(S, SS)$$
$$HKDF\_expand\,(K_{DK}, FI) = HMAC(K_{DK}, FI)$$

where

S is the salt which may be select or non-select

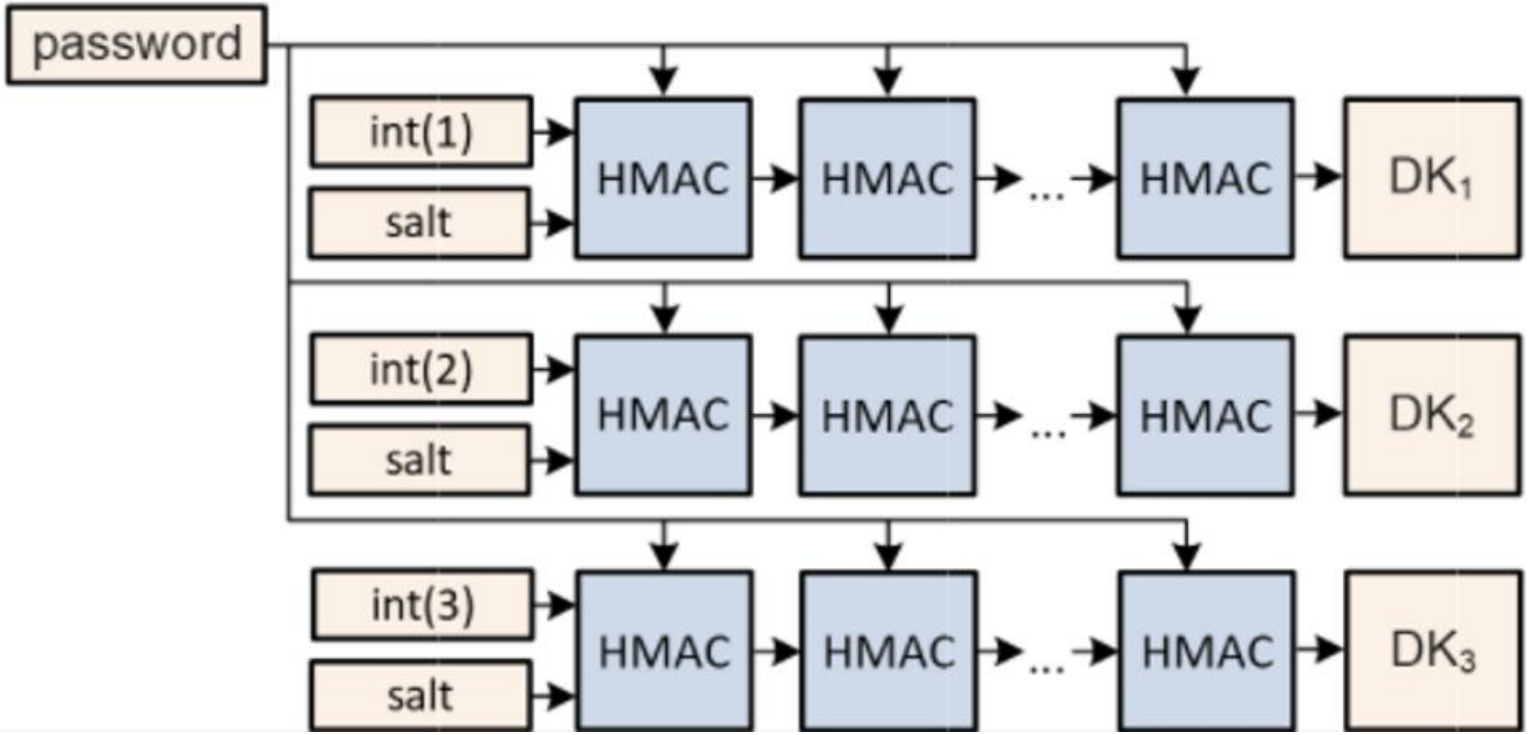
SS is the shared secret that is computed during an approved key-establishment scheme

K_{DK} is the key-derivation key

FI is the Fixed information whose value does not change during the execution

PBKDF

- Users' passwords are often not long enough or random enough due to memory limitations
- The recommended minimum number of iterations was larger than 1000



Outline

1. SHA-2 Algorithm
2. SHA-3 Algorithm
3. HMAC Algorithm
4. KDF Algorithm
5. **Summary**

Summary

Algorithm		Output size bits	State size bits	Input size bits	Rounds	Input bits/ round
SHA2	SHA-224 SHA-256	224 256	256+256	512	64	8
	SHA-384 SHA-512 SHA-512/224 SHA-512/256	384 512 224 256	512+512	1024	80	12.8
SHA3	SHA3-224 SHA3-256 SHA3-384 SHA3-512	224 256 384 512	1600	1152 1088 832 576	24	48 45.3 34.7 24
	SHAKE128 SHAKE256	d (Arbitrarily)		1344 1088		56 45.3

Thank you!

More educational materials? Feel free to follow us!

PUFacademy