Secure Hash Algorithm 3

Courtney Cook

Division of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA

17 November 2018
Senior Seminar

Why should you care?

Why should you care?

Almost everything you do online uses a hash function

Why should you care?

Almost everything you do online uses a hash function

- Passwords

Why should you care?

Almost everything you do online uses a hash function

- Passwords
- Site certificates

Why should you care?

Almost everything you do online uses a hash function

- Passwords
- Site certificates
- Message authentication

Hash Algorithms

"Any function that can be used to map data of arbitrary size to data of a fixed size"

Hash Algorithms

"Any function that can be used to map data of arbitrary size to data of a fixed size"

SHA-256[My] ->

8ed6791bdf3d61a1e6edcbb253979b0a6bef7f3d99dda0fb49cffe96923514b6

SHA-256[My name is Courtney] ->

543ab313f11d6316f84438e074964058613ffa595f1494f81eeafad23364b7cb

SHA-256, www.movable-type.co.uk

Hash Algorithms

"Any function that can be used to map data of arbitrary size to data of a fixed size"

SHA-256[My] ->

8ed6791bdf3d61a1e6edcbb253979b0a6bef7f3d99dda0fb49cffe96923514b6

SHA-256[My name is Courtney] ->

543ab313f11d6316f84438e074964058613ffa595f1494f81eeafad23364b7cb

SHA-256[My name is Courtnet] ->

602e1fad697d322020a89b03339458cdcfabfef70a6173ae1daf4feafebe4a76

SHA-256, www.movable-type.co.uk

Hash Algorithms

"Any function that can be used to map data of arbitrary size to data of a fixed size"

SHA-256[My] ->

8ed6791bdf3d61a1e6edcbb253979b0a6bef7f3d99dda0fb49cffe96923514b6

SHA-256[My name is Courtney] ->

543ab313f11d6316f84438e074964058613ffa595f1494f81eeafad23364b7cb

SHA-256[My name is Courtnet] ->

602e1fad697d322020a89b03339458cdcfabfef70a6173ae1daf4feafebe4a76

HASH[Pay me $30] = HASH[You owe me $50] $\leftarrow$ collision!

SHA-256, www.movable-type.co.uk

Outline

1. Background
   - History
   - Operators
   - Notation
   - Liner Feedback Shift Registers
   - Padding
2. Base Construction
3. Inner Workings
4. Conclusion

Outline

1. Background
   - History
   - Operators
   - Notation
   - Liner Feedback Shift Registers
   - Padding
2. Base Construction
3. Inner Workings
4. Conclusion

Secure Hash Algorithm Family

Secure Hash Algorithm Family

SHA1 - 1995, NSA. 160 bits. Somewhat vulnerable to collisions and length-extension attacks

Secure Hash Algorithm Family

SHA1 - 1995, NSA. 160 bits. Somewhat vulnerable to collisions and length-extension attacks

SHA2 - 2001, NSA. 224, 256, 384 or 512 bits.

Secure Hash Algorithm Family

SHA1 - 1995, NSA. 160 bits. Somewhat vulnerable to collisions and length-extension attacks

SHA2 - 2001, NSA. 224, 256, 384 or 512 bits.

SHA3 - 2015, chosen via competition.

Bitwise Operators

Bitwise Operators

XOR ($\oplus$) : Adding bits modulo 2

$$
\begin{array}{r}
0011 \\
\oplus 0101 \\
= 0110
\end{array}
$$

Bitwise Operators

XOR ($\oplus$) : Adding bits modulo 2

$$\begin{array}{r} 0011 \\ \oplus 0101 \\ = 0110 \end{array}$$

AND ($\wedge$) : Multiplying bits modulo 2

$$\begin{array}{r} 0011 \\ \wedge 0101 \\ = 0001 \end{array}$$

Bitwise Operators

XOR ($\oplus$) : Adding bits modulo 2

$$\begin{array}{r} 0011 \\ \oplus 0101 \\ = 0110 \end{array}$$

AND ($\wedge$) : Multiplying bits modulo 2

$$\begin{array}{r} 0011 \\ \wedge 0101 \\ = 0001 \end{array}$$

NOT ($\neg$) : Bit flipping

$$\begin{array}{r} \neg(0011) \\ = 1100 \end{array}$$

Notation

Notation

Working in set of all strings of 0s and 1s
($\varepsilon, 0, 1, 0010101, 100001,$ *etc*)

Notation

Working in set of all strings of 0s and 1s
$(\varepsilon, 0, 1, 0010101, 100001, \textit{etc})$

Concatenation of strings A and B is A||B.
(0100||101 = 0100101)

Notation

Working in set of all strings of 0s and 1s
($\varepsilon, 0, 1, 0010101, 100001, \textit{etc}$)

Concatenation of strings A and B is A||B.
(0100||101 = 0100101)

Truncating M to its first ℓ bits is $\lfloor M \rfloor_\ell$.
($\lfloor 10110100 \rfloor_4$ = 1011)

Notation

Working in set of all strings of 0s and 1s
($\varepsilon, 0, 1, 0010101, 100001,$ *etc*)

Concatenation of strings A and B is A||B.
(0100||101 = 0100101)

Truncating M to its first ℓ bits is $\lfloor M \rfloor_\ell$.
($\lfloor 10110100 \rfloor_4$ = 1011)

A series of n 0s or 1s will be written 0^n or 1^n
So 111100 can be written $1^4 0^2$
If the number of bits is unknown, we'll use 0^* or 1^*

LFSRs

LFSRs

Liner Feedback Shift Registers are given as polynomial

LFSRs

Liner Feedback Shift Registers are given as polynomial

Input is a linear function of previous state

LFSRs

Liner Feedback Shift Registers are given as polynomial

Input is a linear function of previous state

$x^3 + x^2 + 1$ means:

LFSRs

Liner Feedback Shift Registers are given as polynomial

Input is a linear function of previous state

$x^3 + x^2 + 1$ means:
- internal state is 3 bits long

LFSRs

Liner Feedback Shift Registers are given as polynomial

Input is a linear function of previous state

$x^3 + x^2 + 1$ means:
- internal state is 3 bits long
- get next bits by XORing bits at spaces 0 and 2

LFSRs

Liner Feedback Shift Registers are given as polynomial

Input is a linear function of previous state

$x^3 + x^2 + 1$ means:
- internal state is 3 bits long
- get next bits by XORing bits at spaces 0 and 2

LFSR	Output
100	-

LFSRs

Liner Feedback Shift Registers are given as polynomial

Input is a linear function of previous state

$x^3 + x^2 + 1$ means:
- internal state is 3 bits long
- get next bits by XORing bits at spaces 0 and 2

LFSR	Output
100	-
110	0

LFSRs

Liner Feedback Shift Registers are given as polynomial

Input is a linear function of previous state

$x^3 + x^2 + 1$ means:
- internal state is 3 bits long
- get next bits by XORing bits at spaces 0 and 2

LFSR	Output
100	-
110	0
111	0

LFSRs

Liner Feedback Shift Registers are given as polynomial

Input is a linear function of previous state

$x^3 + x^2 + 1$ means:
- internal state is 3 bits long
- get next bits by XORing bits at spaces 0 and 2

LFSR	Output
100	-
110	0
111	0
011	1

Padding

Hash functions work with a specific block size

Padding

Hash functions work with a specific block size

Protect against extension attacks

Padding

Hash functions work with a specific block size

Protect against extension attacks

Multi-rate padding : input = $M||10^*1$

Padding

Hash functions work with a specific block size

Protect against extension attacks

Multi-rate padding : input = $M||10^*1$

The number of 0s depends on how many are needed to complete a block.

Padding

Hash functions work with a specific block size

Protect against extension attacks

Multi-rate padding : input = M||10*1

The number of 0s depends on how many are needed to complete a block.

0110011 with a block size of 4 becomes

0110|011

0110|0111|0001

Outline

Sponge Construction

Transformation f, padding, and the bitrate r (and capacity c)
r+c=length of state s

Sponge Construction

Transformation f, padding, and the bitrate r (and capacity c)
r+c=length of state s

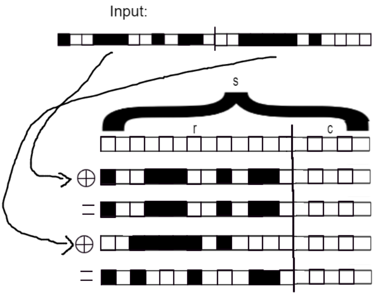
Variable input and output length, but fixed-length transformation

Sponge Construction

Transformation *f*, padding, and the bitrate r (and capacity c)
r+c=length of state s

Variable input and output length, but fixed-length transformation

Absorbing: XOR r-length blocks into *s*, interleaved with the transformation



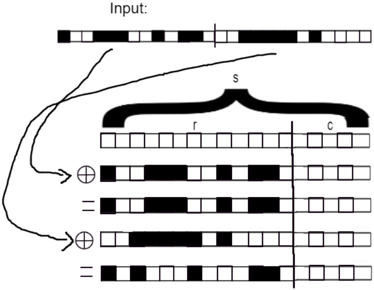
Sponge Construction

Transformation f, padding, and the bitrate r (and capacity c)
r+c=length of state s

Variable input and output length, but fixed-length transformation

Absorbing: XOR r-length blocks into s, interleaved with the transformation

Squeezing: output r-length blocks, interleaving with transformation f



Example

Small Example

Example

Small Example

Input: 10010100
$f(x)$: Circular Left Shift by 1 (Shift left by 1 bit)
Block length: 4
Capacity: 2
Output length: 12

Example

Small Example

Input: 10010100
$f(x)$: Circular Left Shift by 1 (Shift left by 1 bit)
Block length: 4
Capacity: 2
Output length: 12

Padding:

1001|0100|1001

Example

1001|0100|1001

Absorb:

$$
\begin{aligned}
s_0^a &= 000000 \\
s_1^a &= s_0^a \oplus 1001||00
\end{aligned}
$$

$$
\begin{aligned}
& 000000 \\
\oplus & 100100 \\
= & 100100
\end{aligned}
$$

$$
\begin{aligned}
s_2^a &= f(s_1^a) \\
&= f(100100) \\
&= 001001
\end{aligned}
$$

Example

1001|<u>0100</u>|1001

Absorb:

$$
\begin{aligned}
s_2^a =& 001001 \\
s_3^a =& s_2^a \oplus 0100||00 \\
\\
& \ 001001 \\
& \oplus 010000 \\
& = 011001 \\
s_4^a =& f(s_3^a) \\
=& f(011001) \\
=& 110010
\end{aligned}
$$

Example

1001|0100|<u>1001</u>

Absorb:

$$\begin{aligned}
s_4^a =& 110010 \\
s_5^a =& s_4^a \oplus 1001||00 \\
\\
& \;110010 \\
\oplus& 100100 \\
=& 010110 \\
s_6^a =& f(s_5^a) \\
=& f(010110) \\
=& 101100
\end{aligned}$$

Example

1001|0100|1001

Absorb:

$$
\begin{aligned}
s_4^a &= 110010 \\
s_5^a &= s_4^a \oplus 1001||00 \\
\\
&110010 \\
&\oplus 100100 \\
&= 010110 \\
s_6^a &= f(s_5^a) \\
&= f(010110) \\
&= 101100
\end{aligned}
$$

Squeeze:

$$
\begin{aligned}
Z &= \varepsilon \\
s_0^s &= 101100
\end{aligned}
$$

Example

1001|0100|1001

Absorb:

$$s_4^a = 110010$$
$$s_5^a = s_4^a \oplus 1001||00$$

$$\begin{aligned} & 110010 \\ \oplus & 100100 \\ = & 010110 \end{aligned}$$

$$\begin{aligned} s_6^a &= f(s_5^a) \\ &= f(010110) \\ &= 101100 \end{aligned}$$

Squeeze:

$$Z = \varepsilon$$
$$s_0^s = 101100$$

$$\begin{aligned} Z &= \lfloor s_0^s \rfloor_4 \\ &= 1011 \end{aligned}$$

Example

1001|0100|1001

Absorb:

$$
\begin{aligned}
s_4^a &= 110010 \\
s_5^a &= s_4^a \oplus 1001||00
\end{aligned}
$$

$$
\begin{aligned}
& 110010 \\
\oplus & 100100 \\
= & 010110
\end{aligned}
$$

$$
\begin{aligned}
s_6^a &= f(s_5^a) \\
&= f(010110) \\
&= 101100
\end{aligned}
$$

Squeeze:

$$
\begin{aligned}
Z &= \varepsilon \\
s_0^s &= 101100
\end{aligned}
$$

$$
\begin{aligned}
Z &= \lfloor s_0^s \rfloor_4 \\
&= 1011 \\
s_1^s &= f(s_0^s) \\
&= f(101100) \\
&= 011001
\end{aligned}
$$

Example

1001|0100|1001

Absorb:

$$s_4^a = 110010$$
$$s_5^a = s_4^a \oplus 1001||00$$

$$\begin{aligned} & 110010 \\ \oplus & 100100 \\ = & 010110 \end{aligned}$$

$$\begin{aligned} s_6^a &= f(s_5^a) \\ &= f(010110) \\ &= 101100 \end{aligned}$$

Squeeze:

$$Z = 1011$$
$$s_1^s = 011001$$

$$\begin{aligned} Z &= Z || \lfloor s_1^s \rfloor_4 \\ &= 1011|0110 \end{aligned}$$

Example

1001|0100|1001

Absorb:

$$s_4^a = 110010$$
$$s_5^a = s_4^a \oplus 1001||00$$

$$\begin{aligned} & 110010 \\ \oplus & 100100 \\ = & 010110 \end{aligned}$$

$$\begin{aligned} s_6^a &= f(s_5^a) \\ &= f(010110) \\ &= 101100 \end{aligned}$$

Squeeze:

$$Z = 1011$$
$$s_1^s = 011001$$

$$\begin{aligned} Z &= Z||\lfloor s_1^s \rfloor_4 \\ &= 1011|0110 \\ s_2^s &= f(s_1^s) \\ &= f(011001) \\ &= 110010 \end{aligned}$$

Example

1001|0100|1001

Absorb:

$$s_4^a = 110010$$
$$s_5^a = s_4^a \oplus 1001||00$$

$$\begin{aligned} & 110010 \\ \oplus & 100100 \\ = & 010110 \end{aligned}$$

$$s_6^a = f(s_5^a)$$
$$= f(010110)$$
$$= 101100$$

Squeeze:

$$Z = 1011|0110$$
$$s_2^s = 110010$$

$$Z = Z|| \lfloor s_2^s \rfloor_4$$
$$= 1011|0110|1100$$

Example

1001|0100|1001

Absorb:

$$s_4^a = 110010$$
$$s_5^a = s_4^a \oplus 1001||00$$

$$\begin{aligned} & 110010 \\ \oplus & 100100 \\ = & 010110 \end{aligned}$$

$$s_6^a = f(s_5^a)$$
$$= f(010110)$$
$$= 101100$$

Squeeze:

$$Z = 1011|0110$$
$$s_2^s = 110010$$

$$Z = Z||\lfloor s_2^s \rfloor_4$$
$$= 1011|0110|1100$$

Output: 101101101100

Outline

Keccak

Keccak

Keccak

There are 7 different versions of Keccak, labelled 0-6 (ℓ)

Keccak

There are 7 different versions of Keccak, labelled 0-6 (ℓ)

Consists of n_r rounds of R, where R = $\iota \circ \chi \circ \rho \circ \pi \circ \theta$.

n_r = 12+2ℓ

Keccak

There are 7 different versions of Keccak, labelled 0-6 (ℓ)

Consists of n_r rounds of R, where R = $\iota \circ \chi \circ \rho \circ \pi \circ \theta$.

n_r = 12+2ℓ

Works on state α = [5][5][w]

$w = 2^\ell$

Keccak

There are 7 different versions of Keccak, labelled 0-6 (ℓ)

Consists of n_r rounds of R, where R = $\iota \circ \chi \circ \rho \circ \pi \circ \theta$.

n_r = 12+2ℓ

Works on state α = [5][5][w]

$w = 2^\ell$

$$s[w(5y + x) + z] = \alpha[x][y][z]$$

Keccak

There are 7 different versions of Keccak, labelled 0-6 (ℓ)

Consists of n_r rounds of R, where
R = $\iota \circ \chi \circ \rho \circ \pi \circ \theta$.
n_r = 12+2ℓ

Works on state α = [5][5][w]
$w = 2^{\ell}$

$s[w(5y + x) + z] = \alpha[x][y][z]$
Keccak-0:
α[1][1][1] $\rightarrow$s[$2^0(5 * 1 + 1) + 1$] = s[7]

Keccak

There are 7 different versions of Keccak, labelled 0-6 (ℓ)

Consists of n_r rounds of R, where R = $\iota \circ \chi \circ \rho \circ \pi \circ \theta$.

n_r = 12+2ℓ

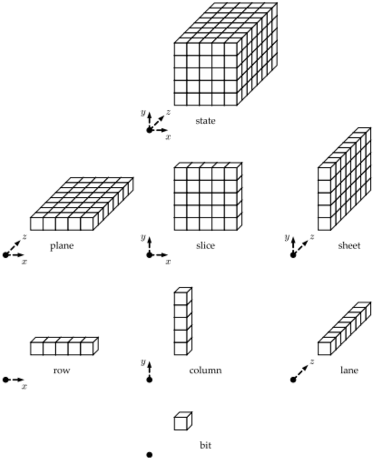
Works on state α = [5][5][w]

$w = 2^{\ell}$

$s[w(5y + x) + z] = \alpha[x][y][z]$

Keccak-0:

α[1][1][1] $\rightarrow$s[$2^0(5 * 1 + 1) + 1$] = s[7]



The state of Keccak [Keccak Reference]

R $= \iota \circ \chi \circ \rho \circ \pi \circ \underline{\theta}$.

Theta

```
FOR x = 0 to 4
    C[x][z] := α[x][0][z]
    FOR y = 1 to 4
        C[x][z] := C[x][z] ⊕ α[x][y][z]
    END FOR
END FOR
FOR x = 0 to 4
    D[x][z] := C[x-1][z] ⊕ C[x+1][z-1]
    FOR y = 0 to 4
        α[x][y][z] := α[x][y][z] ⊕ D[x][z]
    END FOR
END FOR
```

$R = \iota \circ \chi \circ \rho \circ \pi \circ \underline{\theta}$.

Theta

```
FOR x = 0 to 4
    C[x][z] := α[x][0][z]
    FOR y = 1 to 4
        C[x][z] := C[x][z] ⊕ α[x][y][z]
    END FOR
END FOR
FOR x = 0 to 4
    D[x][z] := C[x-1][z] ⊕ C[x+1][z-1]
    FOR y = 0 to 4
        α[x][y][z] := α[x][y][z] ⊕ D[x][z]
    END FOR
END FOR
```

The state is collapsed down into 2 dimensional plane in array C

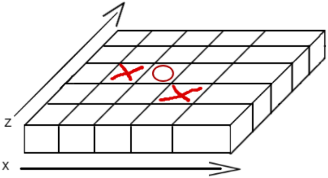
R = $\iota \circ \chi \circ \rho \circ \pi \circ \underline{\theta}$.

Theta

```
FOR x = 0 to 4
    C[x][z] := α[x][0][z]
    FOR y = 1 to 4
        C[x][z] := C[x][z] ⊕ α[x][y][z]
    END FOR
END FOR
FOR x = 0 to 4
    D[x][z] := C[x-1][z] ⊕ C[x+1][z-1]
    FOR y = 0 to 4
        α[x][y][z] := α[x][y][z] ⊕ D[x][z]
    END FOR
END FOR
```

The state is collapsed down into 2 dimensional plane in array C

D's entries are XOR of previous and diagonal next entries of C



$\mathsf{R} = \iota \circ \chi \circ \rho \circ \pi \circ \underline{\theta}.$

Theta

```
FOR x = 0 to 4
    C[x][z] := α[x][0][z]
    FOR y = 1 to 4
        C[x][z] := C[x][z] ⊕ α[x][y][z]
    END FOR
END FOR
FOR x = 0 to 4
    D[x][z] := C[x-1][z] ⊕ C[x+1][z-1]
    FOR y = 0 to 4
        α[x][y][z] := α[x][y][z] ⊕ D[x][z]
    END FOR
END FOR
```

The state is collapsed down into 2 dimensional plane in array C

D's entries are XOR of previous and diagonal next entries of C

α XORed with corresponding D entry

$R = \iota \circ \chi \circ \rho \circ \pi \circ \underline{\theta}$.

Theta

```
FOR x = 0 to 4
    C[x][z] := α[x][0][z]
    FOR y = 1 to 4
        C[x][z] := C[x][z] ⊕ α[x][y][z]
    END FOR
END FOR
FOR x = 0 to 4
    D[x][z] := C[x-1][z] ⊕ C[x+1][z-1]
    FOR y = 0 to 4
        α[x][y][z] := α[x][y][z] ⊕ D[x][z]
    END FOR
END FOR
```

The state is collapsed down into 2 dimensional plane in array C

D's entries are XOR of previous and diagonal next entries of C

α XORed with corresponding D entry

For massive diffusion

R = $\iota \circ \chi \circ \rho \circ \underline{\pi} \circ \theta$.

Pi

FOR x = 0 to 4

FOR y = 0 to 4

$$\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

α[a][b][z] := α[x][y][z]

END FOR

END FOR

$R = \iota \circ \chi \circ \rho \circ \underline{\pi} \circ \theta.$

Pi

FOR x = 0 to 4

FOR y = 0 to 4

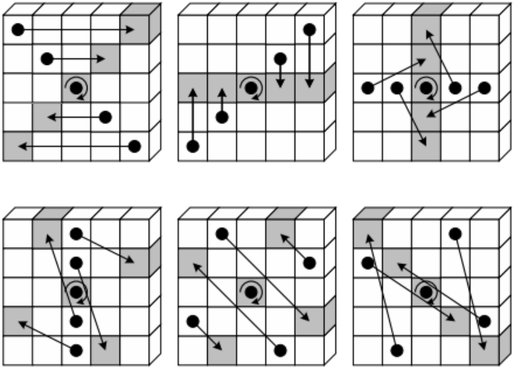
$$\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

α[a][b][z] := α[x][y][z]

END FOR

END FOR

Lanes shifted



Pi [Keccak Reference]

$R = \iota \circ \chi \circ \rho \circ \underline{\pi} \circ \theta.$

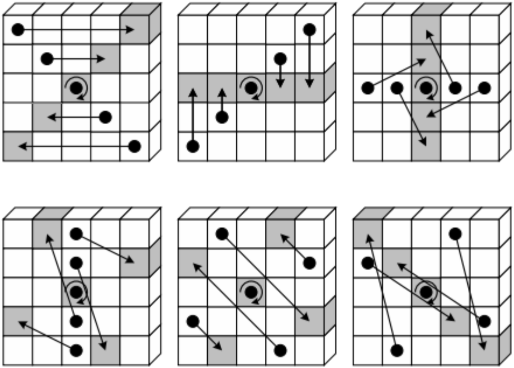
Pi

FOR x = 0 to 4
    FOR y = 0 to 4

$$\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

α[a][b][z] := α[x][y][z]
    END FOR
END FOR

Lanes shifted



Pi [Keccak Reference]

For long-term diffusion

$\mathrm{R} = \iota \circ \chi \circ \underline{\rho} \circ \pi \circ \theta.$

Rho

$$\begin{bmatrix} x \\ y \end{bmatrix} := \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

FOR t = 0 to 23

α[x][y]: = ROT(α[x][y],(t+1)(t+2)/2)

$$\begin{bmatrix} x \\ y \end{bmatrix} := \begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

END FOR

$\mathrm{R} = \iota \circ \chi \circ \underline{\rho} \circ \pi \circ \theta.$

Rho

$\begin{bmatrix} x \\ y \end{bmatrix} := \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

FOR t = 0 to 23

α[x][y]: = ROT(α[x][y],(t+1)(t+2)/2)

$\begin{bmatrix} x \\ y \end{bmatrix} := \begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$

END FOR

Every lane is rotated by a function of t:

$$\frac{(t+1)(t+2)}{2}$$

$R = \iota \circ \chi \circ \underline{\rho} \circ \pi \circ \theta.$

Rho

$\begin{bmatrix} x \\ y \end{bmatrix} := \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

FOR t = 0 to 23

α[x][y]: = ROT(α[x][y],(t+1)(t+2)/2)

$\begin{bmatrix} x \\ y \end{bmatrix} := \begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$

END FOR

Every lane is rotated by a function of t:

$$\frac{(t+1)(t+2)}{2}$$

Order the lanes are rotated = lane shift in π

$R = \iota \circ \chi \circ \underline{\rho} \circ \pi \circ \theta.$

Rho

$\begin{bmatrix} x \\ y \end{bmatrix} := \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

FOR t = 0 to 23

α[x][y]: = ROT(α[x][y],(t+1)(t+2)/2)

$\begin{bmatrix} x \\ y \end{bmatrix} := \begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$

END FOR

Every lane is rotated by a function of t:

$$\frac{(t+1)(t+2)}{2}$$

Order the lanes are rotated = lane shift in π

So when t = 2, rotate lane at α[2][3] by 6

$R = \iota \circ \chi \circ \underline{\rho} \circ \pi \circ \theta.$

Rho

$\begin{bmatrix} x \\ y \end{bmatrix} := \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

FOR t = 0 to 23

α[x][y]: = ROT(α[x][y],(t+1)(t+2)/2)

$\begin{bmatrix} x \\ y \end{bmatrix} := \begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$

END FOR

Every lane is rotated by a function of t:

$$\frac{(t+1)(t+2)}{2}$$

Order the lanes are rotated = lane shift in π

So when t = 2, rotate lane at α[2][3] by 6

For inter-slice dispersion

$R = \iota \circ \underline{\chi} \circ \pi \circ \rho \circ \theta.$

Chi

FOR x = 0 to 4
FOR y = 0 to 4
α[x][y] := α[x][y]
$\oplus$ ($\neg\alpha$[x+1][y] $\wedge$ α[x+2][y])
END FOR
END FOR

$R = \iota \circ \underline{\chi} \circ \pi \circ \rho \circ \theta.$

Chi

FOR x = 0 to 4
  FOR y = 0 to 4
    α[x][y] := α[x][y]
      $\oplus$ ($\neg\alpha$[x+1][y] $\wedge$ α[x+2][y])
  END FOR
END FOR

NOT of lane in next x spot
AND lane two x spots over

$R = \iota \circ \underline{\chi} \circ \pi \circ \rho \circ \theta.$

Chi

FOR x = 0 to 4
  FOR y = 0 to 4
    α[x][y] := α[x][y]
      $\oplus$ ($\neg\alpha$[x+1][y] $\wedge$ α[x+2][y])
  END FOR
END FOR

NOT of lane in next x spot
AND lane two x spots over
XOR with original lane

$R = \iota \circ \underline{\chi} \circ \pi \circ \rho \circ \theta.$

Chi

FOR x = 0 to 4
  FOR y = 0 to 4
    α[x][y] := α[x][y]
      $\oplus$ ($\neg\alpha$[x+1][y] $\wedge$ α[x+2][y])
  END FOR
END FOR

NOT of lane in next x spot
AND lane two x spots over
XOR with original lane

Non-linear

$\mathrm{R} = \underline{\iota} \circ \chi \circ \pi \circ \rho \circ \theta.$

Iota

$\alpha[0][0] := \alpha[0][0] \oplus \mathrm{RC}_{i_r}$

$\mathrm{R} = \iota \circ \chi \circ \pi \circ \rho \circ \theta.$

Iota

$\alpha[0][0] := \alpha[0][0] \oplus \mathrm{RC}_{i_r}$

RC_{i_r} determined by a Linear Feedback Shift Register

$\mathrm{R} = \underline{\iota} \circ \chi \circ \pi \circ \rho \circ \theta$.

Iota

$\alpha[0][0] := \alpha[0][0] \oplus \mathrm{RC}_{i_r}$

RC_{i_r} determined by a Linear Feedback Shift Register

Changes from round to round
Number of non-zero bits is ℓ+1

$R = \iota \circ \chi \circ \pi \circ \rho \circ \theta.$

Iota

$\alpha[0][0] := \alpha[0][0] \oplus \mathrm{RC}_{i_r}$

RC_{i_r} determined by a Linear Feedback Shift Register

Changes from round to round
Number of non-zero bits is ℓ+1
(Meaning if ℓ=4, there are 16 bits, of which 5 are 1s)

R = $\underline{\iota} \circ \chi \circ \pi \circ \rho \circ \theta$.

Iota

α[0][0] := α[0][0] $\oplus$ RC_{i_r}

RC_{i_r} determined by a Linear Feedback Shift Register

Changes from round to round
Number of non-zero bits is ℓ+1
(Meaning if ℓ=4, there are 16 bits, of which 5 are 1s)

LFSR output is XORed with lane at α[0][0]

R = $\underline{\iota} \circ \chi \circ \pi \circ \rho \circ \theta$.

Iota

$\alpha[0][0] := \alpha[0][0] \oplus \mathrm{RC}_{i_r}$

RC_{i_r} determined by a Linear Feedback Shift Register

Changes from round to round
Number of non-zero bits is ℓ+1
(Meaning if ℓ=4, there are 16 bits, of which 5 are 1s)

LFSR output is XORed with lane at $\alpha[0][0]$

To disrupt symmetry

Outline

Conclusion

A user of the Secure Hash Algorithm 3 can decide what trade-offs they want to make (speed vs security)

Conclusion

A user of the Secure Hash Algorithm 3 can decide what trade-offs they want to make (speed vs security)

Because in SHA3 the capacity only interacts with the transformation, it is not as vulnerable to length-extension attacks as previous SHAs are.

Conclusion

A user of the Secure Hash Algorithm 3 can decide what trade-offs they want to make (speed vs security)

Because in SHA3 the capacity only interacts with the transformation, it is not as vulnerable to length-extension attacks as previous SHAs are.

Collisions found for SHA1, can be applied to SHA2.

Thank You

Elena Machkasova, advisor

References

G. Bertoni, J. Daemen, M. Peeters, G. Van Assche.
Cryptographic sponge functions.
SHA-3 competition (round 3), 2011.

G. Bertoni, J. Daemen, M. Peeters, G. Van Assche.
The Making of Keccak.
Cryptologia, 2014.

Questions?