An Approximation of a Definitive Survey of Notes on the Future State of Hash Functions; Pre-re-visited Redux Encore

Saqib A. Kakvi

EUROCRYPT 2019 Rump Session

SHA-1 IS BROKEN!!!!

- Collisions have been found in SHA-1

SHA-1 IS BROKEN!!!!

- Collisions have been found in SHA-1
- PANIC!!!!

SHA-1 IS BROKEN!!!!

- Collisions have been found in SHA-1
- PANIC!!!!!

SHA-1 IS BROKEN!!!!

- Collisions have been found in SHA-1
- PANIC!!!!

SHA-1 IS BROKEN!!!!

- Collisions have been found in SHA-1
- PANIC!!!!!

SHA-1 IS BROKEN!!!!

- Collisions have been found in SHA-1
- PANIC!!!!!

SHA-1 IS BROKEN!!!!

- Collisions have been found in SHA-1
- ~~PANIC!!!!!~~

SHA-1 IS BROKEN!!!!

- Collisions have been found in SHA-1
- ~~PANIC!!!!!~~
- We have other hash functions

SHA-1 IS BROKEN!!!!

- Collisions have been found in SHA-1
- ~~PANIC!!!!!~~
- We have other hash functions
- SHA-2 and SHA-3 are still fine![1]

[1] I hope! I didn't bother to check but nobody will notice.

SHA-1 IS BROKEN!!!!

- Collisions have been found in SHA-1
- ~~PANIC!!!!!~~
- We have other hash functions
- SHA-2 and SHA-3 are still fine![1]
- But why should be stick to the SHA family?[2]

[1] I hope! I didn't bother to check but nobody will notice.
[2] My SHA-3 candidate was rejected due "lack of rigour"

The Solution

- Create a new family of hash functions[3].

[3] I'll just make my own hash function family.

The Solution

- Create a new family of hash functions[3].
- Encourage the usage of organic primitives.[4]

[3] I'll just make my own hash function family.

[4] Are sponges organic? This joke doesn't work if they are.

The Solution

- Create a new family of hash functions[3].
- Encourage the usage of organic primitives.[4]
- And range-free[5] primitives.

[3] I'll just make my own hash function family.

[4] Are sponges organic? This joke doesn't work if they are.

[5] It's like free range, but better! It makes my acronym work!

The Solution

- Create a new family of hash functions[3].
- Encourage the usage of organic primitives.[4]
- And range-free[5] primitives.
- Aim for wholesome primitives.[6][7]

[3] I'll just make my own hash function family.
[4] Are sponges organic? This joke doesn't work if they are.
[5] It's like free range, but better! It makes my acronym work!
[6] Again, for the acronym, but hide the details in footnotes!
[7] Surely nobody will notice.

The Solution

- Create a new family of hash functions[3].
- Encourage the usage of organic primitives.[4]
- And range-free[5] primitives.
- Aim for wholesome primitives.[6][7]
- I think we can all agree these are good things!

[3] I'll just make my own hash function family.
[4] Are sponges organic? This joke doesn't work if they are.
[5] It's like free range, but better! It makes my acronym work!
[6] Again, for the acronym, but hide the details in footnotes!
[7] Surely nobody will notice.

The Solution:
AN ALTERNATIVE HASH FUNCTION FAMILY!

The Solution: AN ALTERNATIVE HASH FUNCTION FAMILY!

W holesome

The Solution: AN ALTERNATIVE HASH FUNCTION FAMILY!

W holesome

O rganic

The Solution: AN ALTERNATIVE HASH FUNCTION FAMILY!

W holesome
O rganic
R ange-free

The Solution:
AN ALTERNATIVE HASH FUNCTION FAMILY!

W holesome

O rganic

R ange-free

T rusted

The Solution:
AN ALTERNATIVE HASH FUNCTION FAMILY!

W	holesome
O	rganic
R	ange-free
T	rusted
H	ash-function

The Solution: AN ALTERNATIVE HASH FUNCTION FAMILY!

W	holesome
O	rganic
R	ange-free
T	rusted
H	ash-function
-	of

The Solution: AN ALTERNATIVE HASH FUNCTION FAMILY!

W	holesome
O	rganic
R	ange-free
T	rusted
H	ash-function
-	of
L	arge &

The Solution:
AN ALTERNATIVE HASH FUNCTION FAMILY!

W	holesome
O	rganic
R	ange-free
T	rusted
H	ash-function
-	of
L	arge &
E	xceptionally

The Solution:
AN ALTERNATIVE HASH FUNCTION FAMILY!

W	holesome
O	rganic
R	ange-free
T	rusted
H	ash-function
-	of
L	arge &
E	xceptionally
S	trong

The Solution:
AN ALTERNATIVE HASH FUNCTION FAMILY!

W holesome
O rganic
R ange-free
T rusted
H ash-function
- of
L arge &
E xceptionally
S trong
S ecurity

Syntax and Security

- Block sizes are fixed at 1 million bits for technical reasons.

Syntax and Security

- Block sizes are fixed at 1 million bits for technical reasons.
- Hash inputs are padded up to the next million bits.

Syntax and Security

- Block sizes are fixed at 1 million bits for technical reasons.
- Hash inputs are padded up to the next million bits.
- Security is defined as:

$$\frac{|input|}{|output|}$$

Syntax and Security

- Block sizes are fixed at 1 million bits for technical reasons.
- Hash inputs are padded up to the next million bits.
- Security is defined as:

$$\frac{|input|}{|output|}$$

- Security is measured on parts per million ($\mathfrak{ppm}$).

Syntax and Security

- Block sizes are fixed at 1 million bits for technical reasons.
- Hash inputs are padded up to the next million bits.
- Security is defined as:

$$\frac{|input|}{|output|}$$

- Security is measured on parts per million ($\mathfrak{ppm}$).
- For high security we need a low $\mathfrak{ppm}$, but not too low.

Syntax and Security

- Block sizes are fixed at 1 million bits for technical reasons.
- Hash inputs are padded up to the next million bits.
- Security is defined as:

$$\frac{|input|}{|output|}$$

- Security is measured on parts per million ($\mathfrak{ppm}$).
- For high security we need a low $\mathfrak{ppm}$, but not too low.
- Optimal $\mathfrak{ppm} = 47$. $21 \leq \mathfrak{ppm} \leq 999$ is secure.[8]

[8] Detailed explanation & formulae are in the Full Version.

First candidate: **WORTH-LESS-1** ($\lll_1$)

- We first measure the Hamming weight of the block ω.

First candidate: **WORTH-LESS-1** ($\ll_1$)

- We first measure the Hamming weight of the block ω.
- We compute:

$$\texttt{WMI} = \frac{\omega}{1,000,000}.$$

First candidate: **WORTH-LESS-1** ($\lll_1$)

- We first measure the Hamming weight of the block ω.
- We compute:

$$\mathtt{WMI} = \frac{\omega}{1,000,000}.$$

- The output sets are sets of divine numbers $\mathbb{D}_{\mathtt{WMI}}$ chosen using a ~~magical~~ known method from ~~fiction~~ literature.

First candidate: **WORTH-LESS-1** ($\lll_1$)

- We first measure the Hamming weight of the block ω.
- We compute:
$$\texttt{WMI} = \frac{\omega}{1,000,000}.$$
- The output sets are sets of divine numbers $\mathbb{D}_{\texttt{WMI}}$ chosen using a ~~magical~~ known method from ~~fiction~~ literature.
- We then pick the n^{th} element of the set where:
$$n = \frac{1}{\omega + e^{i}} \mod |\mathbb{D}_{\texttt{WMI}}|.$$

First candidate: **WORTH-LESS-1** ($\ll_1$)

- We first measure the Hamming weight of the block ω.
- We compute:

$$\texttt{WMI} = \frac{\omega}{1,000,000}.$$

- The output sets are sets of divine numbers $\mathbb{D}_{\texttt{WMI}}$ chosen using a ~~magical~~ known method from ~~fiction~~ literature.
- We then pick the nth element of the set where:

$$n = \frac{1}{\omega + e^i} \mod |\mathbb{D}_{\texttt{WMI}}|.$$

- The hash of that block is then $\mathbb{D}_{\texttt{WMI}}[n] \times \pi$.

First candidate: **WORTH-LESS-1** ($\lll_1$)

- We first measure the Hamming weight of the block ω.
- We compute:

$$\texttt{WMI} = \frac{\omega}{1,000,000}.$$

- The output sets are sets of divine numbers $\mathbb{D}_{\texttt{WMI}}$ chosen using a ~~magical~~ known method from ~~fiction~~ literature.
- We then pick the n^{th} element of the set where:

$$n = \frac{1}{\omega + e^{i}} \mod |\mathbb{D}_{\texttt{WMI}}|.$$

- The hash of that block is then $\mathbb{D}_{\texttt{WMI}}[n] \times \pi$.
- The final hash is the concatenation of all the block has values.

$$\lll_1 (m) = ||_{j=1}^{\lceil |m|/1{,}000{,}000 \rceil} \mathbb{D}_{\texttt{WMI}_j}[n_j] \times \pi_j.$$

First candidate: **WORTH-LESS-1** ($\lll_1$)

- Security of $\lll_1$ is between 25$\mathfrak{ppm}$ and 47$\mathfrak{ppm}$.

First candidate: **WORTH-LESS-1** ($\lll_1$)

- Security of $\lll_1$ is between 25$\mathfrak{ppm}$ and 47$\mathfrak{ppm}$.
- This depends on the input string and choice of divine numbers.

First candidate: **WORTH-LESS-1** ($\lll_1$)

- Security of $\lll_1$ is between $25\mathfrak{ppm}$ and $47\mathfrak{ppm}$.
- This depends on the input string and choice of divine numbers.
- This helps to confuse attackers!

First candidate: **WORTH-LESS-1** ($\lll_1$)

- Security of $\lll_1$ is between 25$\mathfrak{ppm}$ and 47$\mathfrak{ppm}$.
- This depends on the input string and choice of divine numbers.
- This helps to confuse attackers!
- Also weeds out lazy developers!

First candidate: **WORTH-LESS-1** ($\lll_1$)

- Security of $\lll_1$ is between 25$\mathfrak{ppm}$ and 47$\mathfrak{ppm}$.
- This depends on the input string and choice of divine numbers.
- This helps to confuse attackers!
- Also weeds out lazy developers!
- Only committed devs will implement multiple check algorithms.

First candidate: **WORTH-LESS-1** ($\lll_1$)

- Security of $\lll_1$ is between $25\mathfrak{ppm}$ and $47\mathfrak{ppm}$.
- This depends on the input string and choice of divine numbers.
- This helps to confuse attackers!
- Also weeds out lazy developers!
- Only committed devs will implement multiple check algorithms.
- THERE ARE NO COLLISIONS!!! GUARANTEED!!![9]

[9]Not an actual guarantee. Terms and conditions apply.

Conclusions

- Newer, better, cooler, shinier hash function family!

Conclusions

- Newer, better, cooler, shinier hash function family!
- First candidate, which is super secure!

Conclusions

- Newer, better, cooler, shinier hash function family!
- First candidate, which is super secure!
- More candidates welcome!

Conclusions

- Newer, better, cooler, shinier hash function family!
- First candidate, which is super secure!
- More candidates welcome!
- Full version available at

Conclusions

- Newer, better, cooler, shinier hash function family!
- First candidate, which is super secure!
- More candidates welcome!
- Full version available at `www.NotAFakeWebsite.com/CreditCardBorrower.js.exe.pdf`

Conclusions

- Newer, better, cooler, shinier hash function family!
- First candidate, which is super secure!
- More candidates welcome!
- Full version available at `www.NotAFakeWebsite.com/CreditCardBorrower.js.exe.pdf`

Conclusions

- Newer, better, cooler, shinier hash function family!
- First candidate, which is super secure!
- More candidates welcome!
- Full version available at
  `www.NotAFakeWebsite.com/CreditCardBorrower.js.exe.pdf`

THANK YOU!