SHA-3 vs the world

David Wong
| Snefru | MD4 |
Snefru

MD4

MD5

SHA-1

SHA-2

Merkle–Damgård
<table>
<thead>
<tr>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Snefru</td>
</tr>
<tr>
<td>MD4</td>
</tr>
<tr>
<td>MD5</td>
</tr>
<tr>
<td>SHA-1</td>
</tr>
<tr>
<td>SHA-2</td>
</tr>
</tbody>
</table>

**Merkle–Damgård**
Snefru

MD4

MD5

SHA-1

SHA-2

Merkle–Damgård
Collision Attack: Two Different Documents, But Same SHA-1 Hash Fingerprint

**SHAttered**

The first concrete collision attack against SHA-1
[https://shattered.io](https://shattered.io)

**CWI**
Marc Stevens
Pierre Karpman

**Google**

Elie Bursztein
Ange Albertini
Yarik Markov

**SHAttered**

The first concrete collision attack against SHA-1
[https://shattered.io](https://shattered.io)

**CWI**
Marc Stevens
Pierre Karpman

**Google**

Elie Bursztein
Ange Albertini
Yarik Markov

```
sha1sum *.pdf
38762cf7f55934b34d179ae6a4c80cadccbb7f0a  1.pdf
38762cf7f55934b34d179ae6a4c80cadccbb7f0a  2.pdf
```

```
sha256sum *.pdf
2bb787a73e37352f92383abe7e2902936d1059ad9f1ba6daaa9c1e58ee6970d0  1.pdf
```
SHA-3 COMPETITION (2007-2012)

Research Results on SHA-1 Collisions (2017)

NIST announced a public competition in a Federal Register Notice on November 2, 2007 to develop a new cryptographic hash algorithm, called SHA-3, for standardization. The competition was NIST’s response to advances made in the cryptanalysis of hash algorithms.

NIST received sixty-four entries from cryptographers around the world by October 31, 2008, and selected fifty-one first-round candidates in December 2008, fourteen second-round candidates in July 2009, and five finalists – BLAKE, Grøstl, JH, Keccak and Skein, in December 2010 to advance to the third and final round of the competition.

Throughout the competition, the cryptographic community has provided an enormous amount of feedback. Most of the comments were sent to NIST and a public hash forum; in addition, many of the cryptanalysis and performance studies were published as papers in major cryptographic conferences or leading cryptographic journals. NIST also hosted a SHA-3 candidate conference in each round to obtain public feedback. Based on the public comments and internal review of the candidates, NIST announced Keccak as the winner of the SHA-3 Cryptographic Hash Algorithm Competition on October 2, 2012, and ended the five-year competition.
FIRST ROUND CANDIDATES

Official comments on the First Round Candidate Algorithms should be submitted using the "Submit Comment" link for the appropriate algorithm. Comments from hash-forum listserv subscribers will also be forwarded to the hash-forum listserv. We will periodically post and update the comments received to the appropriate algorithm.

Please refrain from using OFFICIAL COMMENT to ask administrative questions, which should be sent to hash-function@nist.gov

By selecting the "Submitter's Website" links, you will be leaving NIST webspace. We have provided these links to other web sites because they may have information that would be of interest to you. No inferences should be drawn on account of other sites being referenced, or not, from this page. There may be other web sites that are more appropriate for your purpose. NIST does not necessarily endorse the views expressed, or concur with the facts presented on these sites. Further, NIST does not endorse any commercial products that may be mentioned on these sites.

<table>
<thead>
<tr>
<th>Algorithm Name</th>
<th>Principal Submitter*</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>** Abacus [9M]</td>
<td>Neil Sholer</td>
<td>Submit Comment View Comments</td>
</tr>
<tr>
<td>ARIRANG [18M]</td>
<td>Jongin Lim</td>
<td>Submit Comment View Comments</td>
</tr>
<tr>
<td>Updated Algorithm [16M]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Submitter's Website***</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AURORA [12M]</td>
<td>Masahiro Fujita</td>
<td>Submit Comment View Comments</td>
</tr>
<tr>
<td>Updated Algorithm [41M]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Keccak

BLAKE, Grøstl, JH, Skein
Outline

1. SHA-3
2. derived functions
3. derived protocols
permutation-based cryptography
AES is a permutation
AES is a permutation

input

AES

key

output
Sponge Construction
Sponge Construction
Sponge Construction

\[
\begin{array}{cccccc}
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]
Sponge Construction

 AES

 key
Sponge Construction

message

0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0

\[ f \]

1 1 1 0 0 0 0
1 1 1 0 0 0 0
0 0 0 1 1 0 0
Sponge Construction

message

\[ f \]

0
0
0
0
0
0
0
0
0
0
0
Sponge Construction

message

\[ f \]
Sponge Construction

message

0 0 0 0 0 0 0

\( f \)

\( f \)
Sponge Construction

message

0
0
0
0
0
0
0
0
0
0
0
0
Sponge Construction

message

absorbing
Sponge Construction

Message $\oplus f \oplus f \oplus f \rightarrow$ Output

Absorbing
Sponge Construction

message

output

absorbing
Sponge Construction

message

output

absorbing
Sponge Construction

message

output

absorbing
Third-party cryptanalysis

This page lists all the third-party cryptanalysis results that we know of on Keccak, including FIPS 202 and SP 800-185 instances, KangarooTwelve and the authenticated encryption schemes Ketje and Keyak. We may have forgotten some results, so if you think your result is relevant and should be on this page, please do not hesitate to contact us.

The results are divided into the following categories:

- analysis of the Keccak (covering also KangarooTwelve, FIPS 202 and SP 800-185 instances) in the context of (unkeyed) hashing;
- analysis that is more specifically targeting keyed modes of use of Keccak, including the Ketje and Keyak authenticated encryption schemes;
- analysis on the (reduced-round) Keccak-f permutations that does not extend to any of the aforementioned cryptographic functions.

In each category, the most recent results come first.

Analysis of unkeyed modes

First, the Crunchy Crypto Collision and Pre-image Contest contains third-party cryptanalysis results with practical complexities.

---

K. Qiao, L. Song, M. Liu and J. Guo, New Collision Attacks on Round-Reduced Keccak, Eurocrypt 2017

In this paper, Kexin Qiao, Ling Song, Meicheng Liu and Jian Guo develop a hybrid method combining algebraic and differential techniques to mount collision attacks on Keccak. They can find collisions on various instances of Keccak with the permutation Keccak-f[1600] or Keccak-f[800] reduced to 5 rounds. This includes the 5-round collision challenges in the Crunchy Contest. In the meanwhile, they refined their attack and produced a 6-round collision that took $2^{50}$ evaluations of reduced-round Keccak-f[1600].

---

D. Saha, S. Kuila and D. R. Chowdhury, SymSum: Symmetric-Sum Distinguishers Against Round Reduced Keccak

---
Keccak

Guido Bertoni, Joan Daemen, Michaël Peeters and Gilles Van Assche
SHA-3 competition

2007

2012
FIPS PUB 202

FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION

SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions

CATEGORY: COMPUTER SECURITY       SUBCATEGORY: CRYPTOGRAPHY

Information Technology Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899-8900

This publication is available free of charge from:

SHA-3 competition

2007

2012

SHA-3 standard (FIPS 202) → 2015
### Keccak Code Package

- **186 commits**
- **1 branch**
- **0 releases**
- **15 contributors**

**Branch:** master  
**New pull request**

**The Keccak Team Converted Vladimir Sedach’s AVX2 implementation from C++ to C**

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build</td>
<td>Added grouping of source packages</td>
<td>a year ago</td>
</tr>
<tr>
<td>CAESAR</td>
<td>Reorganized support for Ketje</td>
<td>4 months ago</td>
</tr>
<tr>
<td>Common</td>
<td>Use C89 comments rather than C++ comment style</td>
<td>2 years ago</td>
</tr>
<tr>
<td>Constructions</td>
<td>Fixed various minor syntax issues</td>
<td>3 months ago</td>
</tr>
<tr>
<td>KeccakSum</td>
<td>Improved the granularity of the targets</td>
<td>4 months ago</td>
</tr>
<tr>
<td>Ketje</td>
<td>Fixed ARM assembly syntax, see issue #35 (thanks bitwiseshiftleft and...</td>
<td>a month ago</td>
</tr>
<tr>
<td>Keyak</td>
<td>Reorganized Keyak</td>
<td>4 months ago</td>
</tr>
<tr>
<td>Modes</td>
<td>Fixed various minor syntax issues</td>
<td>3 months ago</td>
</tr>
<tr>
<td>PISnP</td>
<td>Fixed ARM assembly syntax, see issue #35 (thanks bitwiseshiftleft and...</td>
<td>a month ago</td>
</tr>
<tr>
<td>SUPERCOP</td>
<td>Added the generation of packages for SUPERCOP</td>
<td>3 months ago</td>
</tr>
</tbody>
</table>
Where is SHA-3 being used?
Outline

1. SHA-3
2. derived functions
3. derived protocols
FIPS PUB 202

FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION

SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions

CATEGORY: COMPUTER SECURITY         SUBCATEGORY: CRYPTOGRAPHY

Information Technology Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899-8900

This publication is available free of charge from:

SHAKE is a XOF
NIST Special Publication 800-185

SHA-3 Derived Functions:
cSHAKE, KMAC, TupleHash and ParallelHash

John Kelsey
Shu-jen Chang
Ray Perlner

This publication is available free of charge from:
https://doi.org/10.6028/NIST.SP.800-185
SHA-3 competition

2007

SHA-3 standard (FIPS 202) → 2015

SP 800-185 → 2016
KMAC

TupleHash

ParallelHash
KMAC

message || SHA-256(message)

TupleHash

ParallelHash
KMAC

message || SHA-256(key||message)

TupleHash

ParallelHash
KMAC

message || more || SHA-256(key||message||more)

TupleHash

ParallelHash
KMAC

message || SHAKE(key || message)

TupleHash

ParallelHash
KMAC

message || SHAKE(key || message)

TupleHash

my RSA public key = (e, N)

ParallelHash
KMAC

message || SHAKE(key || message)

TupleHash

my RSA public key = (e, N)
fingerprint = SHA-256(e || N)

ParallelHash
KMAC

message || SHAKE(key || message)

TupleHash

fingerprint1 = SHA-256(101011000000000010001...)

ParallelHash
KMAC

message || SHAKE(key || message)

TupleHash

fingerprint1 = SHA-256(1010110000000010001000000000010001...)

ParallelHash

fingerprint2 = SHA-256(1010110000000010001000000000010001...)

KMAC

message || SHAKE(key || message)

TupleHash

SHAKE(len(e) || e || len(N) || N)

ParallelHash
Sponge Construction

message

output

absorbing

game

Sponge Construction

output

squeezing
Sponge Construction

message

output

absorbing

squeezing
Sponge Construction

message

output

absorbing

squeezing
Sponge Construction

message

output

absorbing

message

output

squeezing

\[ f \]
KMAC

message || SHAKE(key || message)

TupleHash

SHAKE(len(e) || e || len(N) || N)

ParallelHash

SHAKE(SHAKE(b1) || SHAKE(b2) || SHAKE(b3) || …)
SHA-3 competition

2007

2012

SHA-3 / SHAKE → 2015

TupleHash / ParallelHash / KMAC → 2016
CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness

Timeline

- M-14, 2013.01.15: Competition announced at the Early Symmetric Crypto workshop in Mondorf-les-Bains; also announced online.
- M0, 2014.03.15: Deadline for first-round submissions.
- M2, 2014.05.15: Deadline for first-round software.
- M17, 2015.08.29: Deadline for second-round tweaks.
- M27, 2016.06.30: Deadline for Verilog/VHDL.
- M29, 2016.08.15: Announcement of third-round candidates.
- M31, 2016.10.15: Deadline for third-round software.
- TBA: Deadline for third-round Verilog/VHDL.
- TBA: Announcement of finalists.
- TBA: Deadline for finalist tweaks.
- TBA: Deadline for finalist software.
- TBA: Deadline for finalist Verilog/VHDL.

Version: This is version 2016.08.15 of the caesar.html web page.
SHA-3 competition

2007

2012

SHA-3 / SHAKE → 2015

TupleHash / ParallelHash / KMAC → 2016

KangarooTwelve & MarsupilamiFourteen
SHA-3 competition

2007

SHA-3 / SHAKE

TupleHash / ParallelHash / KMAC

KangarooTwelve & MarsupilamiFourteen

2012

2015

2016
**Keccak Code Package**

- **172 commits**
- **1 branch**
- **0 releases**
- **15 contributors**

Branch: master | New pull request

<table>
<thead>
<tr>
<th>Directory</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Build</strong></td>
<td>Added grouping of source packages</td>
<td>11 months ago</td>
</tr>
<tr>
<td><strong>CAESAR</strong></td>
<td>Updated to Ketje v2</td>
<td>5 months ago</td>
</tr>
<tr>
<td><strong>Common</strong></td>
<td>Use C89 comments rather than C++ comment style</td>
<td>a year ago</td>
</tr>
<tr>
<td><strong>Constructions</strong></td>
<td>Added KangarooTwelve optimized implementation</td>
<td>10 months ago</td>
</tr>
<tr>
<td><strong>KeccakSum</strong></td>
<td>Fixed possible printf format string vulnerability</td>
<td>4 months ago</td>
</tr>
<tr>
<td><strong>Ketje</strong></td>
<td>uxth needs two parameters</td>
<td>3 months ago</td>
</tr>
<tr>
<td><strong>Modes</strong></td>
<td>Added back missing headers in KangarooTwelve.c</td>
<td>14 days ago</td>
</tr>
<tr>
<td><strong>PISnP</strong></td>
<td>Added more AVX-512 implementations</td>
<td>5 months ago</td>
</tr>
<tr>
<td><strong>SnP</strong></td>
<td>uxth needs two parameters</td>
<td>3 months ago</td>
</tr>
</tbody>
</table>

Latest commit 83f4063 14 days ago

Outline

1. SHA-3
2. derived functions
3. derived protocols
Sponge Construction

message

output

absorbing

squeezing
Duplex Construction

input | output | input | output | input | output

init

0 0 0 0 0 0 0 0

duplexing | duplexing | duplexing
Keyed-mode

key

init

duplexing
Keyed-mode

key

duplexing

0 0 0 0 0 0

init

f

secret part

0 0 0 0 0

leak
Encryption?

Key

init 0 0 0 0 0 0 0

duplexing $f$

Encryption?
Encryption

duplexing

key

cipher_{text1} ⊕ plaintext_{text1}
Authenticated Encryption

key

\[\text{ciphertext1} \oplus f \oplus \text{plaintext1} \rightarrow \text{tag1}\]

duplexing

duplexing
Sessions

key

ciphertext1 ⊕ plaintext1

ciphertext2 ⊕ plaintext2

tag1 ⊕ plaintext1

tag2 ⊕ plaintext2

init

0 0 0 0
0 0 0 0
0 0 0 0

f

duplexing
myProtocol = Strobe_init("myWebsite.com")
myProtocol.KEY(sharedSecret)
buffer += myProtocol.send_ENC("GET /")
buffer += myProtocol.send_MAC(len=16)

// send the buffer

// receive a ciphertext
message = myProtocol.recv_ENC(ciphertext[:-16])
ok = myProtocol.recv_MAC(ciphertext[-16:])
if !ok {
    // reset the connection
}
<table>
<thead>
<tr>
<th>Operation</th>
<th>Flags</th>
</tr>
</thead>
<tbody>
<tr>
<td>AD</td>
<td>A</td>
</tr>
<tr>
<td>KEY</td>
<td>A C</td>
</tr>
<tr>
<td>PRF</td>
<td>I A C</td>
</tr>
<tr>
<td>send_CLR</td>
<td>A T</td>
</tr>
<tr>
<td>recv_CLR</td>
<td>I A T</td>
</tr>
<tr>
<td>send_ENC</td>
<td>A C T</td>
</tr>
<tr>
<td>recv_ENC</td>
<td>I A C T</td>
</tr>
<tr>
<td>send_MAC</td>
<td>C T</td>
</tr>
<tr>
<td>recv_MAC</td>
<td>I C T</td>
</tr>
<tr>
<td>RATCHET</td>
<td>C</td>
</tr>
</tbody>
</table>
myHash = Strobe_init("hash")
myHash.AD("something to be hashed")
hash = myHash.PRF(outputLen=16)
KDF = Strobe_init("deriving keys")
KDF.KEY(keyExchangeOutput)
keys = KDF.PRF(outputLen=32)
key1 = keys[:16]
key2 = keys[16:]
operation = AD
data = 010100…
data = 010100...
operation = AD

len = 16
tag

operation = send_MAC
operation = KEY
data = 010100…
operation = KEY
data = 010100...

operation = send_ENC
data = hello
ciphertext
operation = KEY
data = 010100...

operation = send_ENC
data = hello
ciphertext

operation = send_MAC
tag
len = 16
Version and changelog

This is version 1.0.2 of the STROBE specification. The software is in alpha.

- January 24, 2017: version 1.0.2. Fix the length of $S$ in the cSHAKE domain separation string. Hopefully the last change for this silly reason.
- January 6, 2017: version 1.0.1. Adjust, hopefully, to the final version of the NIST cSHAKE standard. The difference is how the empty personalization string is encoded, and in the order of the $N$ and $S$ strings. The draft was ambiguous, but $N$ followed $S$ and the empty string was probably best interpreted as [0]. The final version changed it to [1,0] with $N$ preceding $S$. I'm still not sure I got it right because there are no test vectors.
- January 3, 2017: version 1.0.0.

Goals

The Internet of Things (IoT) promises ubiquitous, cheap, connected devices. Unfortunately, most of these devices are hastily developed and will never receive code updates. Part of the IoT’s security problem is cryptographic, but established cryptographic solutions seem too heavy or too inflexible to adapt to new use cases.

STROBE is a new framework for cryptographic protocols. It can also be used for regular encryption. Its goals are to make cryptographic protocols much simpler to develop, deploy and
Outline

1. SHA-3
2. derived functions
3. derived protocols
4. Disco?
Noise + Strobe = Disco

www.discocrypto.com
I write about crypto at www.cryptologie.net

I tweet my mind on twitter.com/lyon01_david

and I work here