SHA-3 vs the world

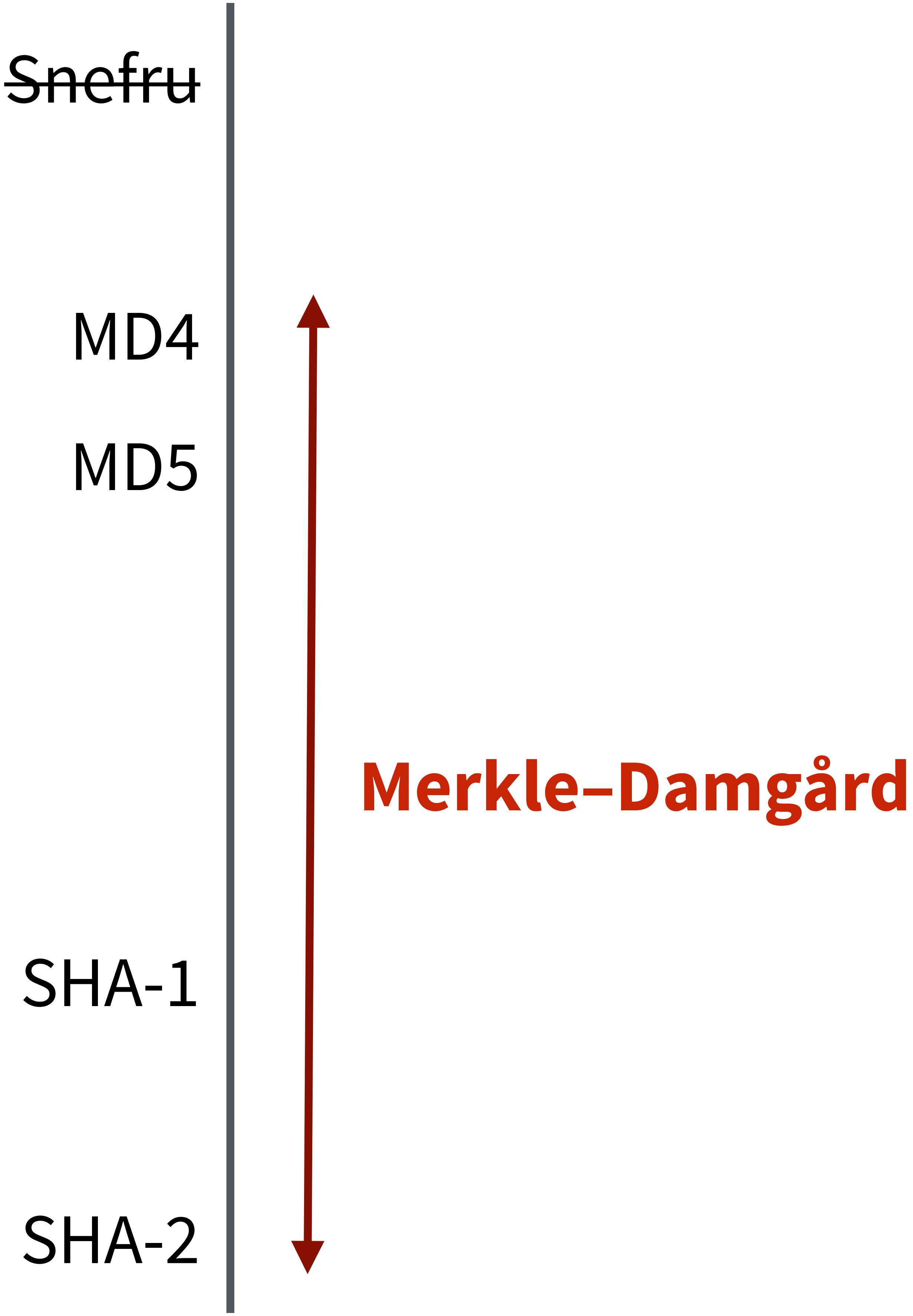
David Wong

Snefru

MD4

~~Snefru~~

MD4



~~Snefru~~

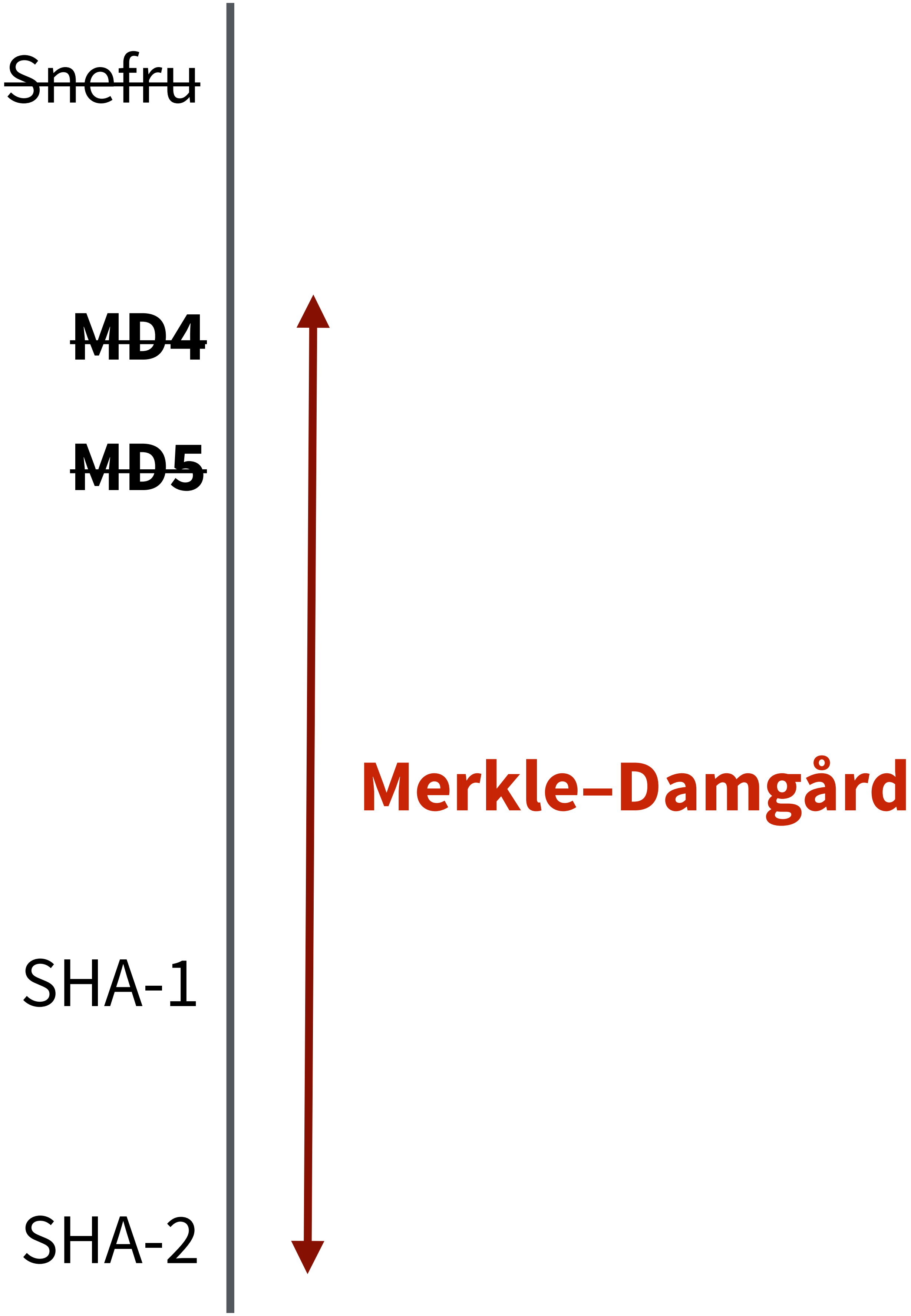
MD4

MD5

SHA-1

SHA-2

Merkle–Damgård



~~Snefru~~

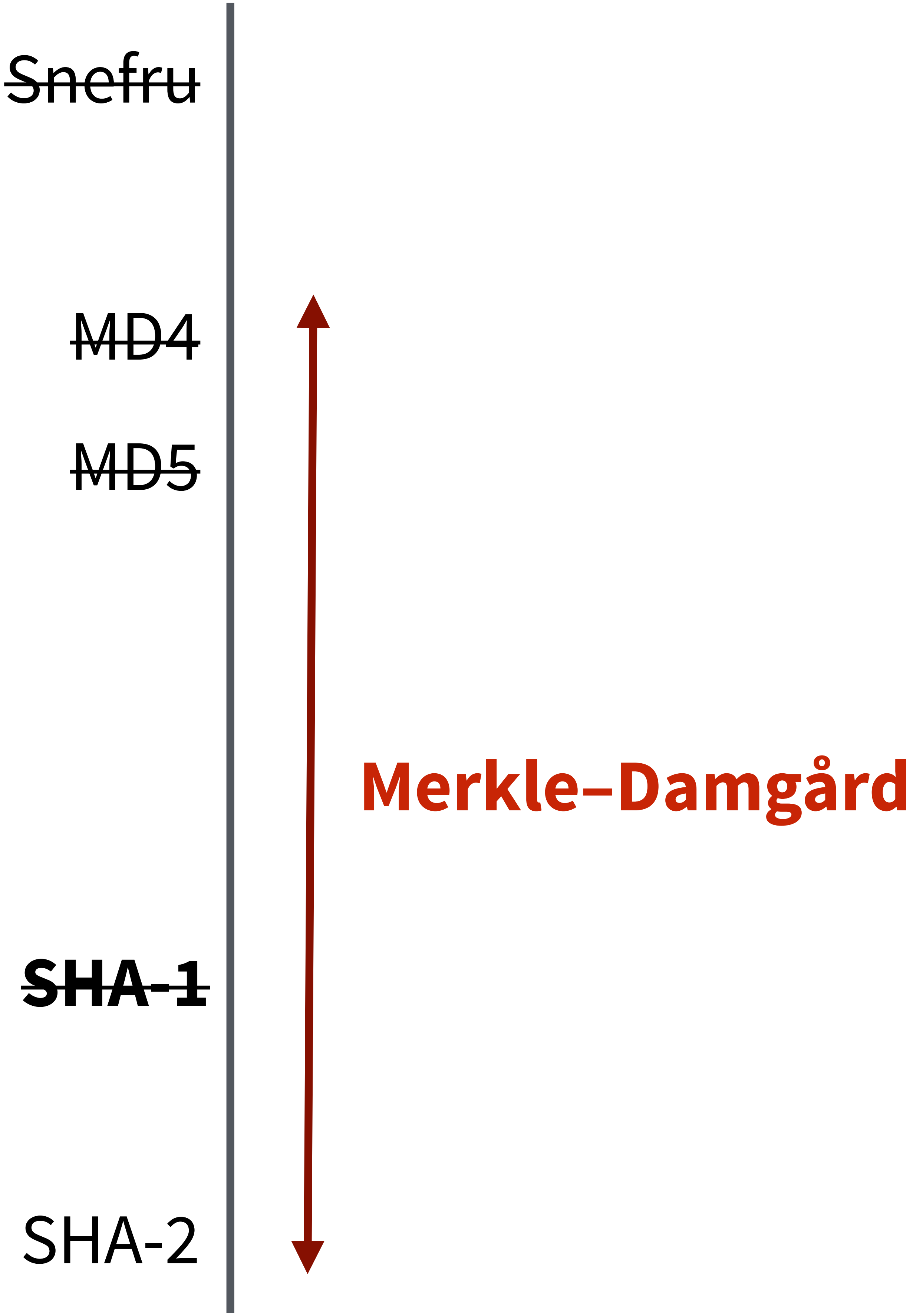
~~MD4~~

~~MD5~~

Merkle–Damgård

SHA-1

SHA-2



~~Snefru~~

~~MD4~~

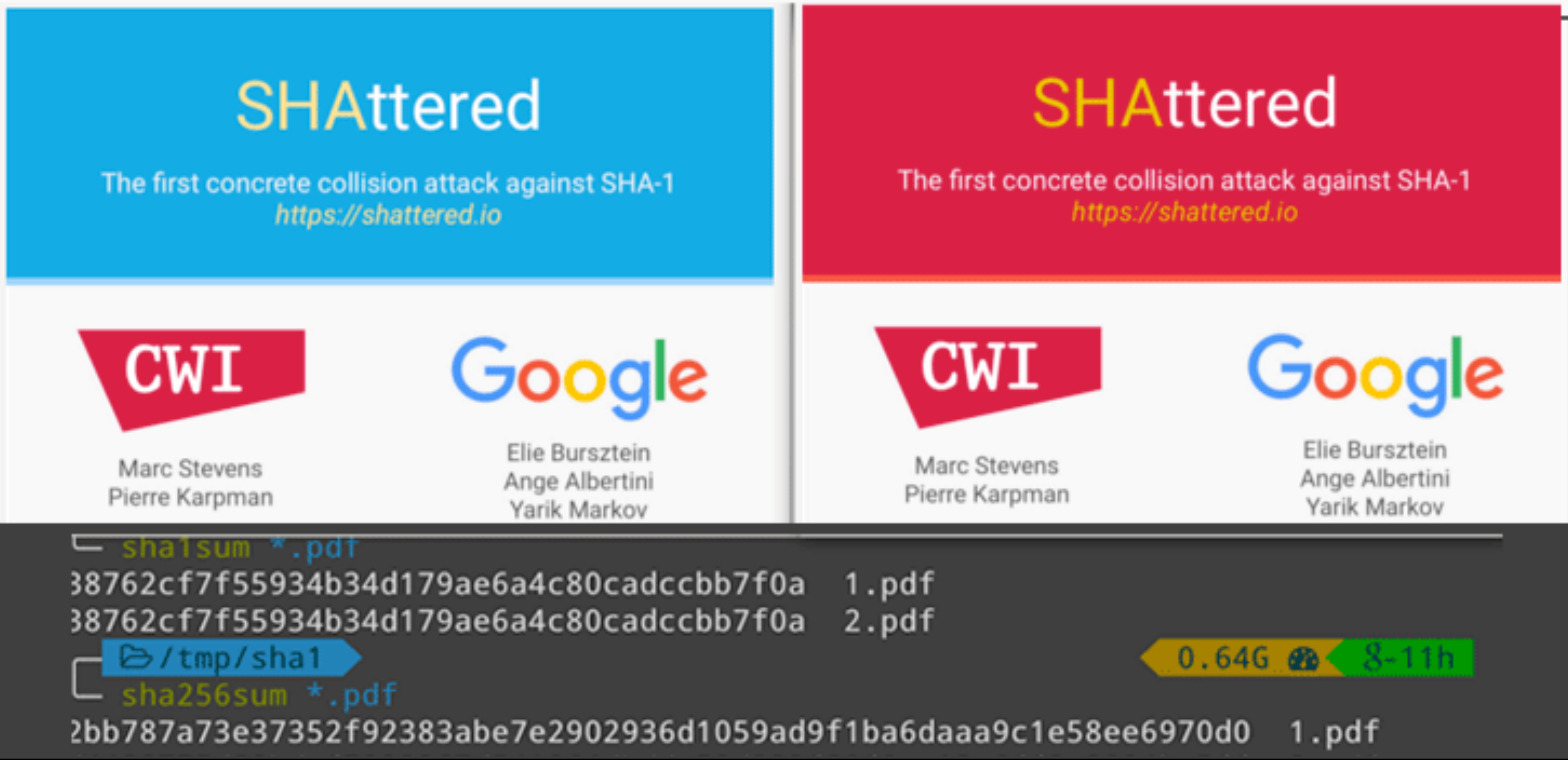
~~MD5~~

~~**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

Marc Stevens
Pierre Karpman

Elie Bursztein
Ange Albertini
Yarik Markov

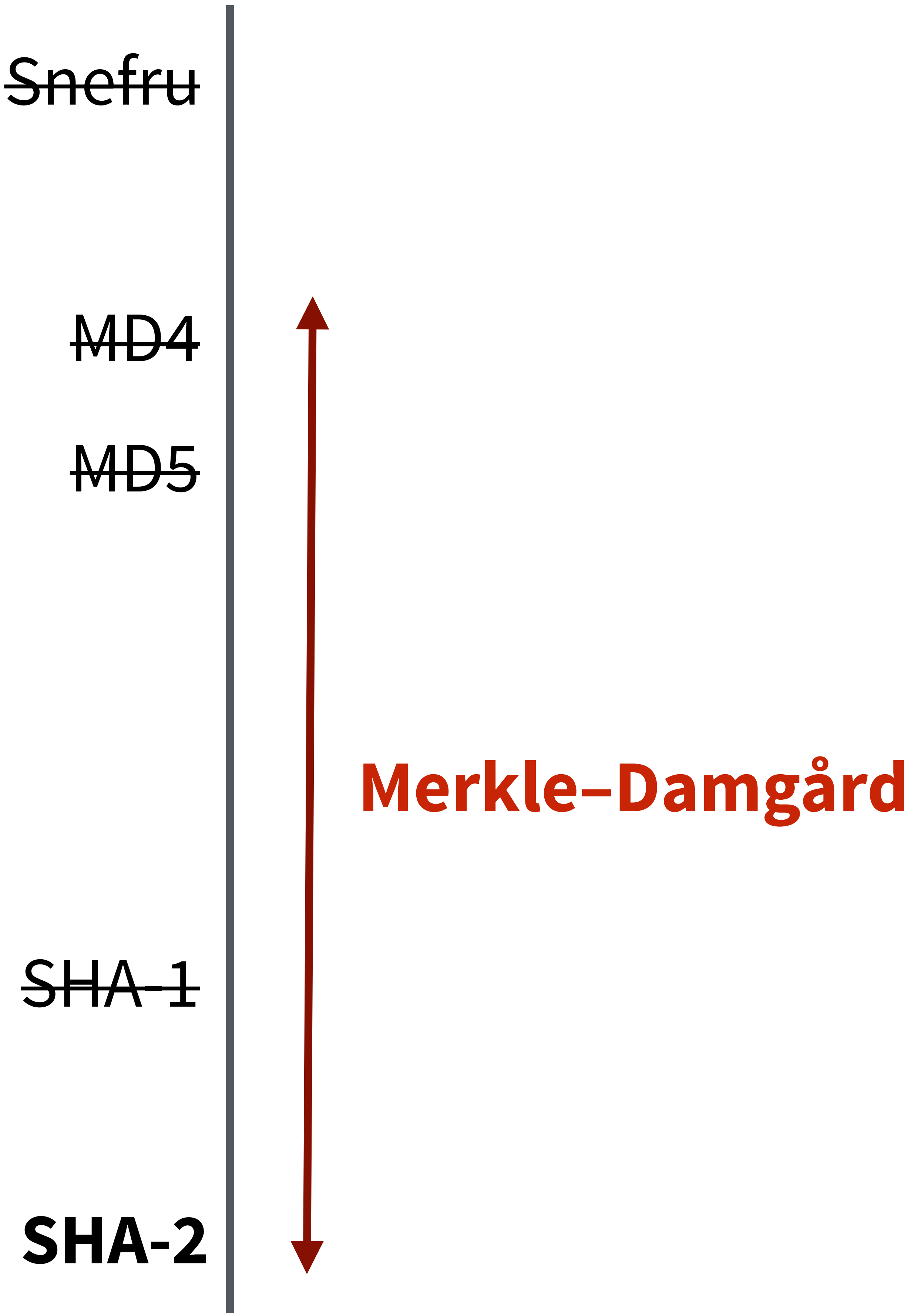
SHAttered

The first concrete collision attack against SHA-1
https://shattered.io

Marc Stevens
Pierre Karpman

Elie Bursztein
Ange Albertini
Yarik Markov

```
└─ sha1sum *.pdf
38762cf7f55934b34d179ae6a4c80cadccbb7f0a  1.pdf
38762cf7f55934b34d179ae6a4c80cadccbb7f0a  2.pdf
┌─ /tmp/sha1                                            0.64G   8-11h
└─ sha256sum *.pdf
2bb787a73e37352f92383abe7e2902936d1059ad9f1ba6daaa9c1e58ee6970d0  1.pdf
```



~~Snefru~~

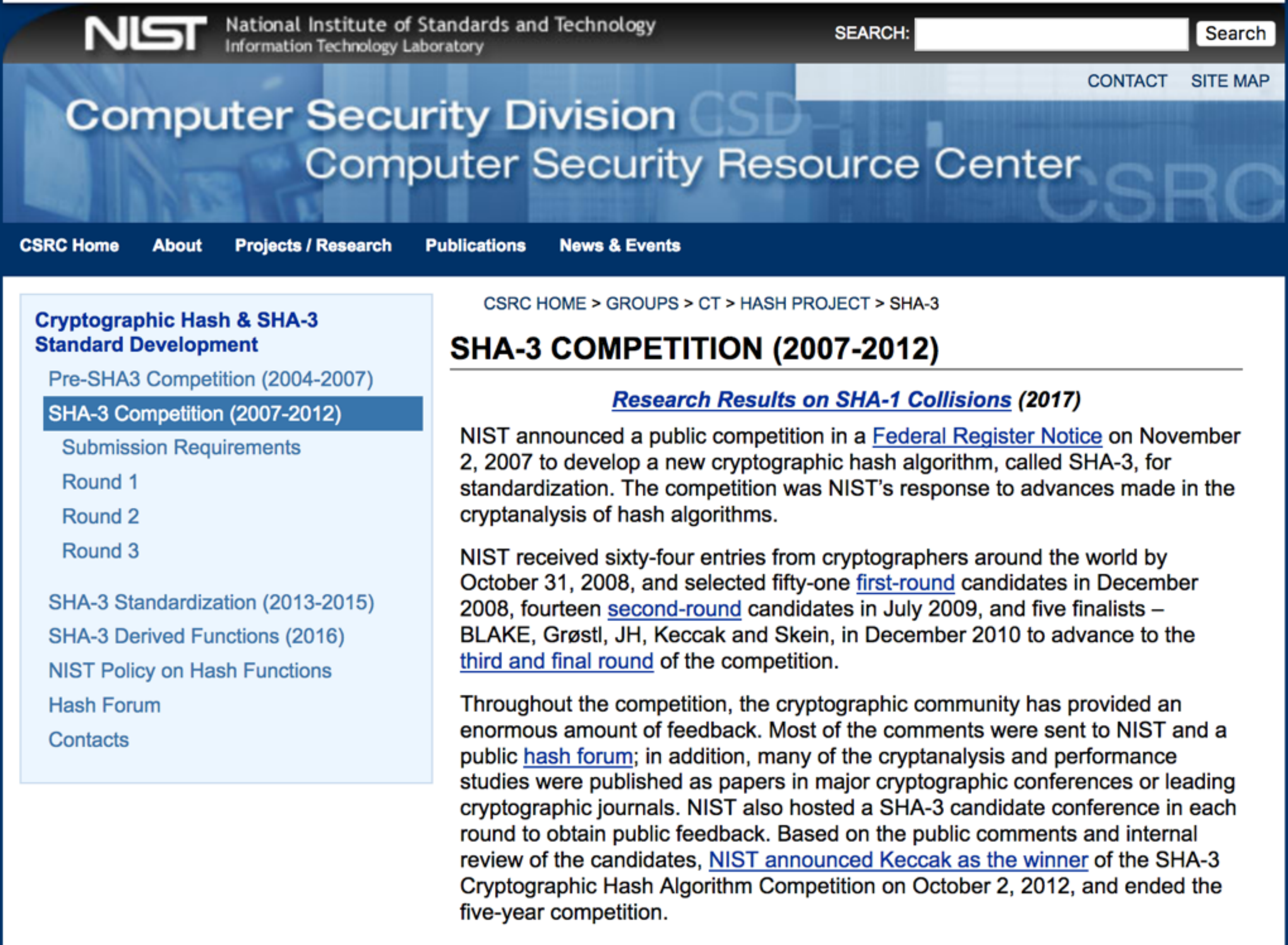
~~MD4~~

~~MD5~~

Merkle–Damgård

~~SHA-1~~

SHA-2


NIST
National Institute of Standards and Technology
Information Technology Laboratory
SEARCH:
Search
CONTACT
SITE MAP
Computer Security Division
CSD
Computer Security Resource Center
CSRC
CSRC Home
About
Projects / Research
Publications
News & Events
Cryptographic Hash & SHA-3 Standard Development
Pre-SHA3 Competition (2004-2007)
SHA-3 Competition (2007-2012)
Submission Requirements
Round 1
Round 2
Round 3
SHA-3 Standardization (2013-2015)
SHA-3 Derived Functions (2016)
NIST Policy on Hash Functions
Hash Forum
Contacts
CSRC HOME > GROUPS > CT > HASH PROJECT > SHA-3

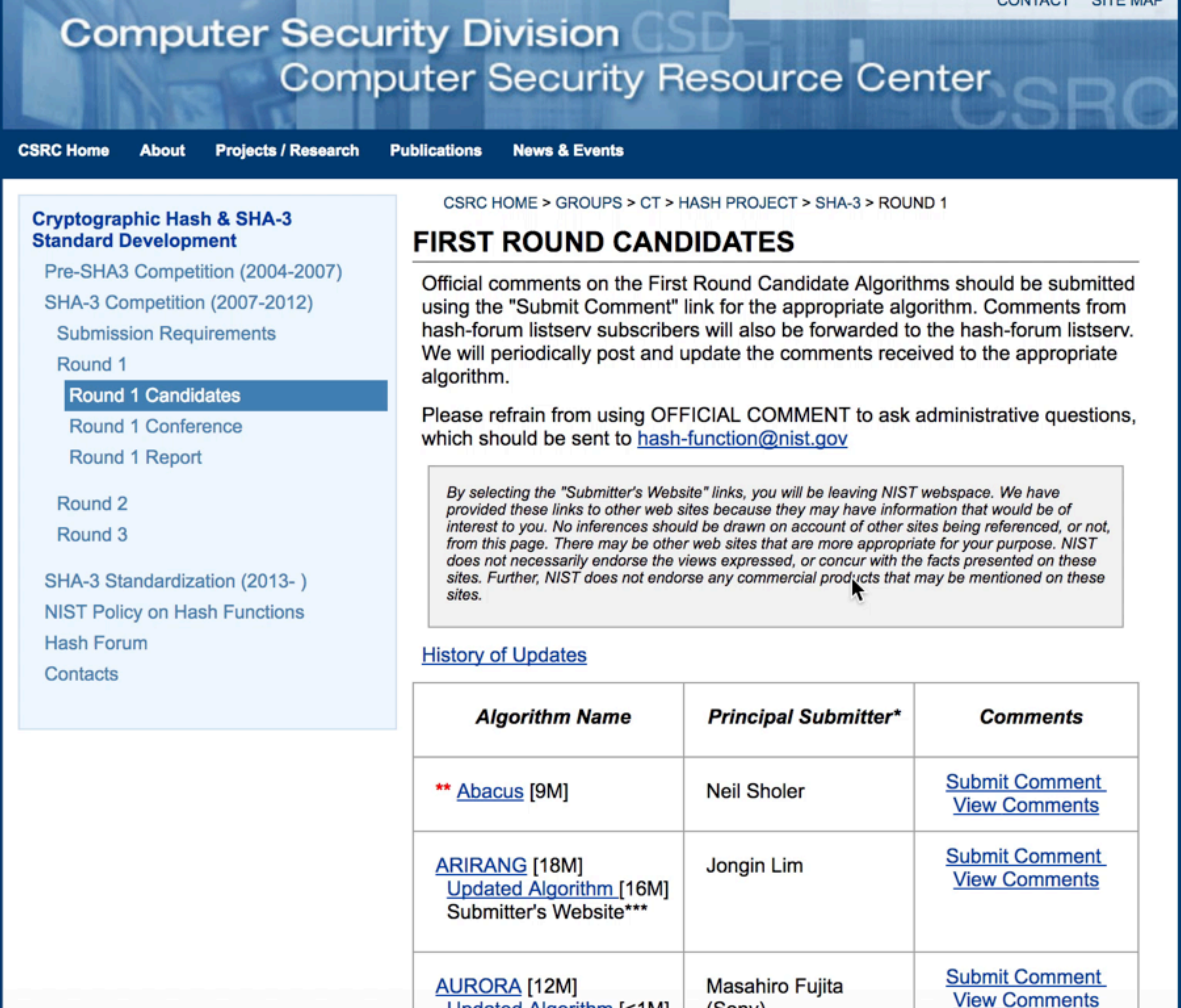
SHA-3 COMPETITION (2007-2012)

***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.



Cryptographic Hash & SHA-3 Standard Development

- Pre-SHA3 Competition (2004-2007)
- SHA-3 Competition (2007-2012)
  - Submission Requirements
  - Round 1
    - Round 1 Candidates
    - Round 1 Conference
    - Round 1 Report
  - Round 2
  - Round 3
- SHA-3 Standardization (2013-)
- NIST Policy on Hash Functions
- Hash Forum
- Contacts

CSRC HOME > GROUPS > CT > HASH PROJECT > SHA-3 > ROUND 1

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.

History of Updates

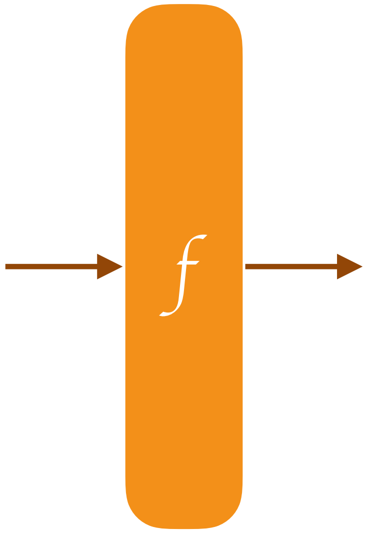
Algorithm Name	***Principal Submitter****	***Comments***
** Abacus [9M]	Neil Sholer	Submit Comment View Comments
ARIRANG [18M] Updated Algorithm [16M] Submitter's Website***	Jongin Lim	Submit Comment View Comments
AURORA [12M] Updated Algorithm [<1M]	Masahiro Fujita (Sony)	Submit Comment View Comments

Keccak

BLAKE, Grøstl, JH, Skein

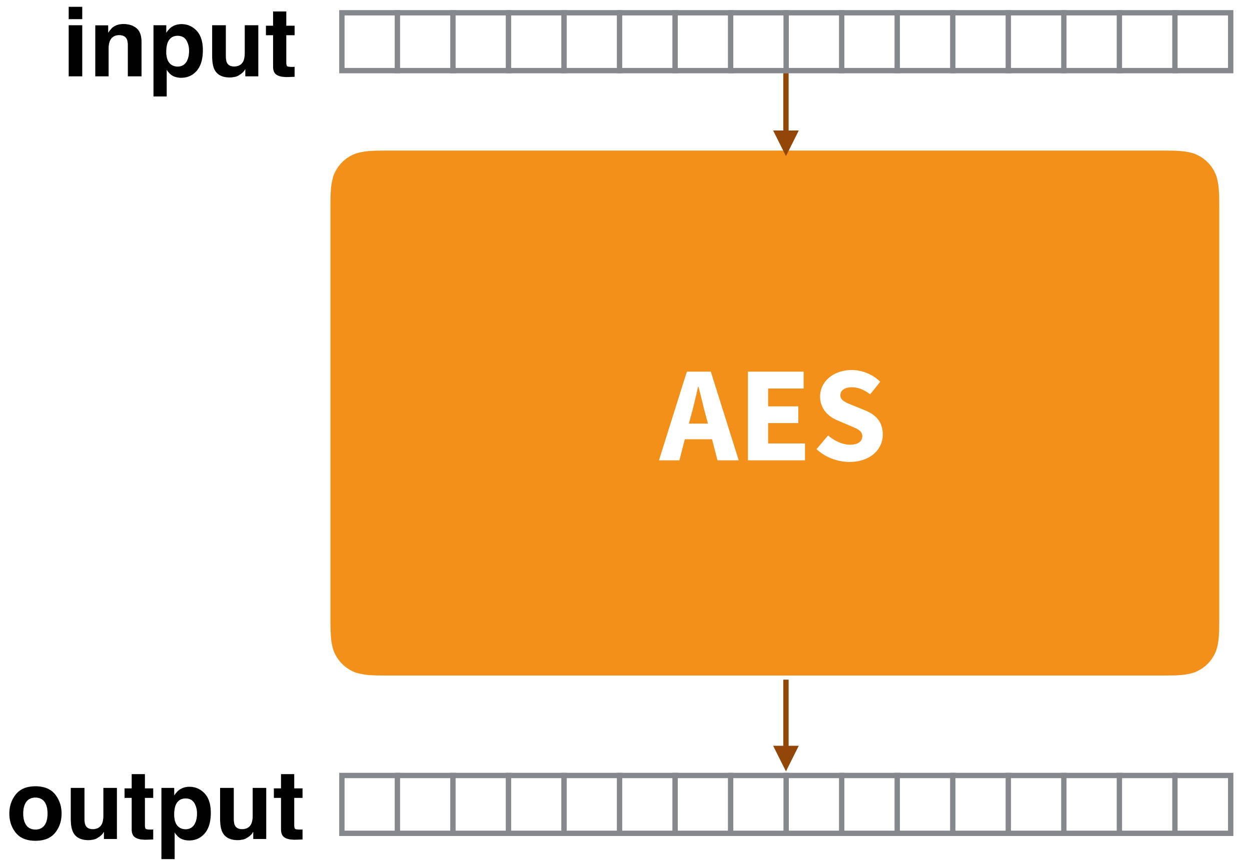
Outline

1. **SHA-3**
2. derived functions
3. derived protocols

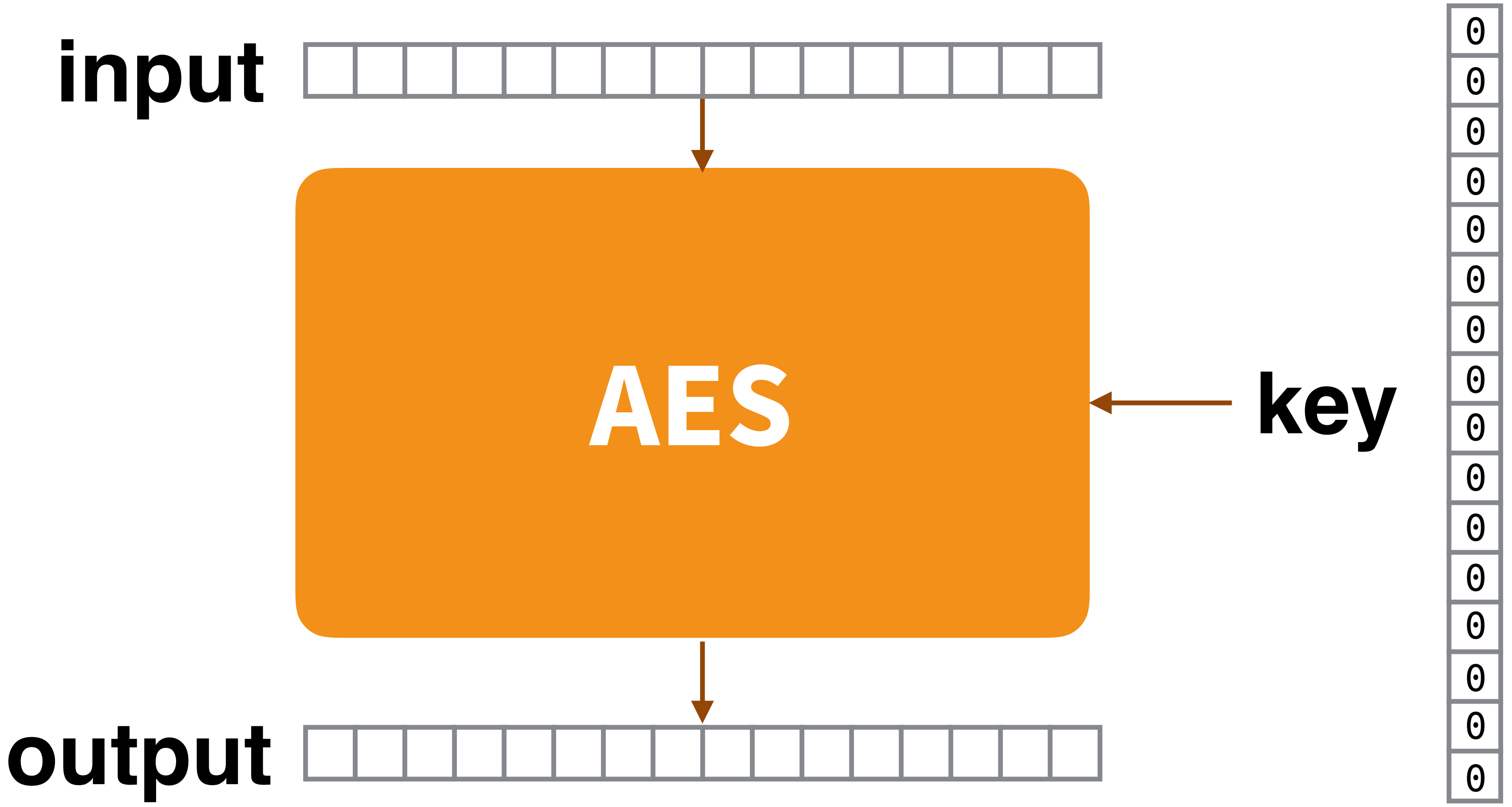


permutation-based cryptography

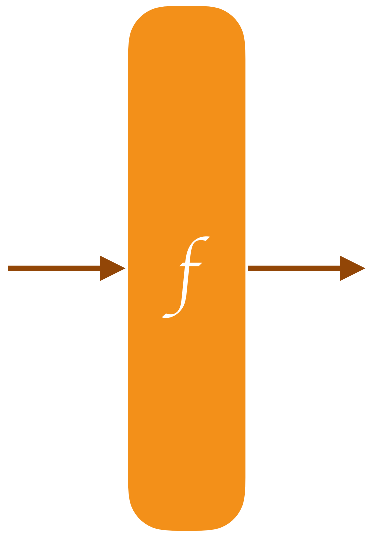
AES is a permutation



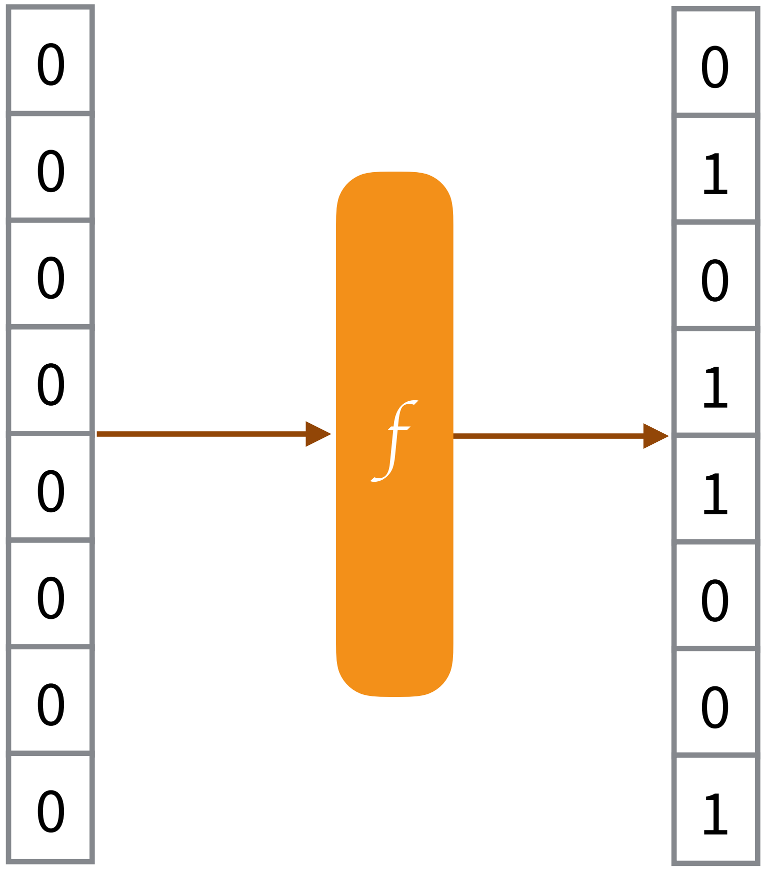
AES is a permutation



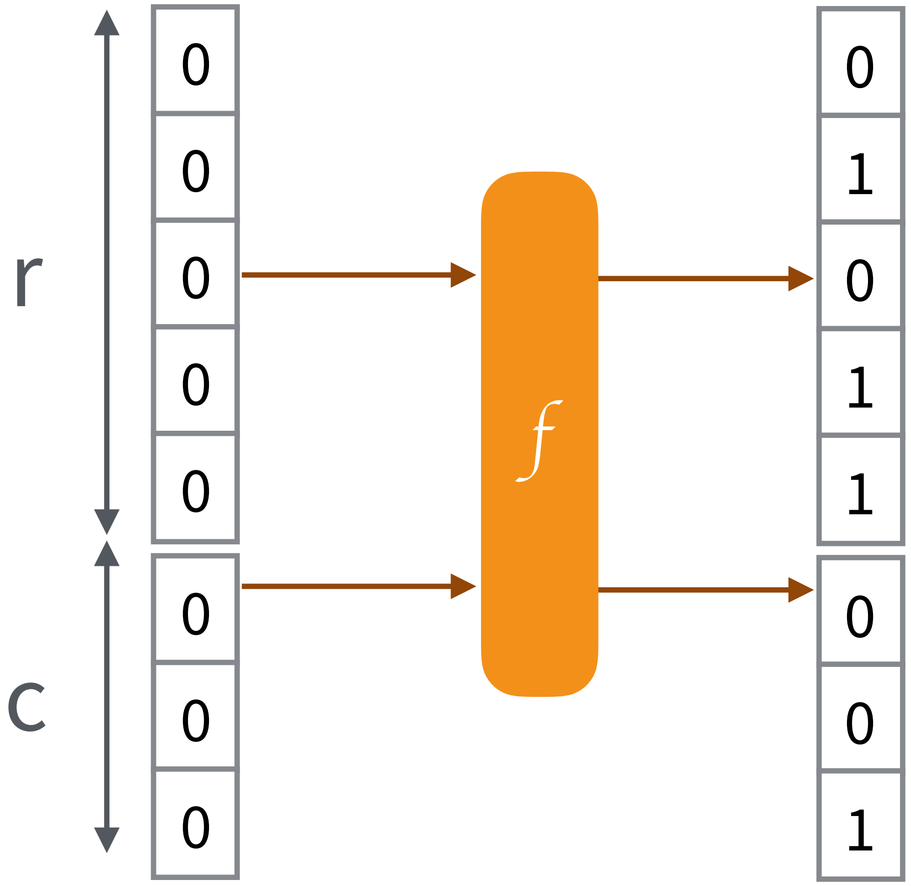
Sponge Construction



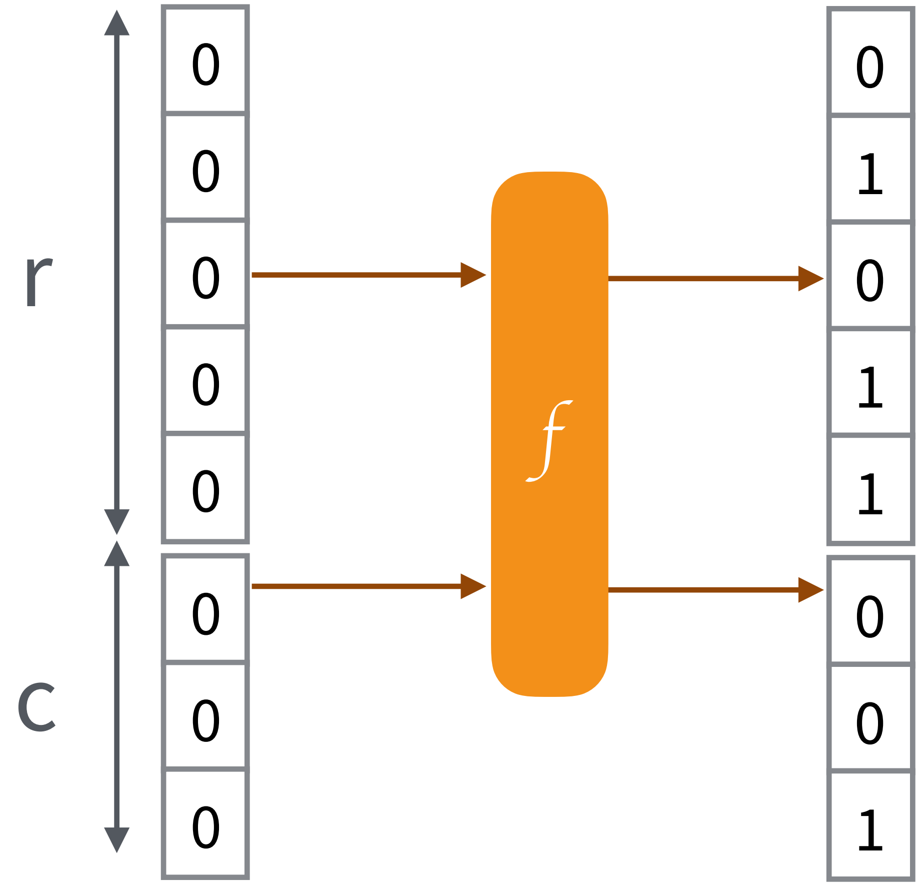
Sponge Construction

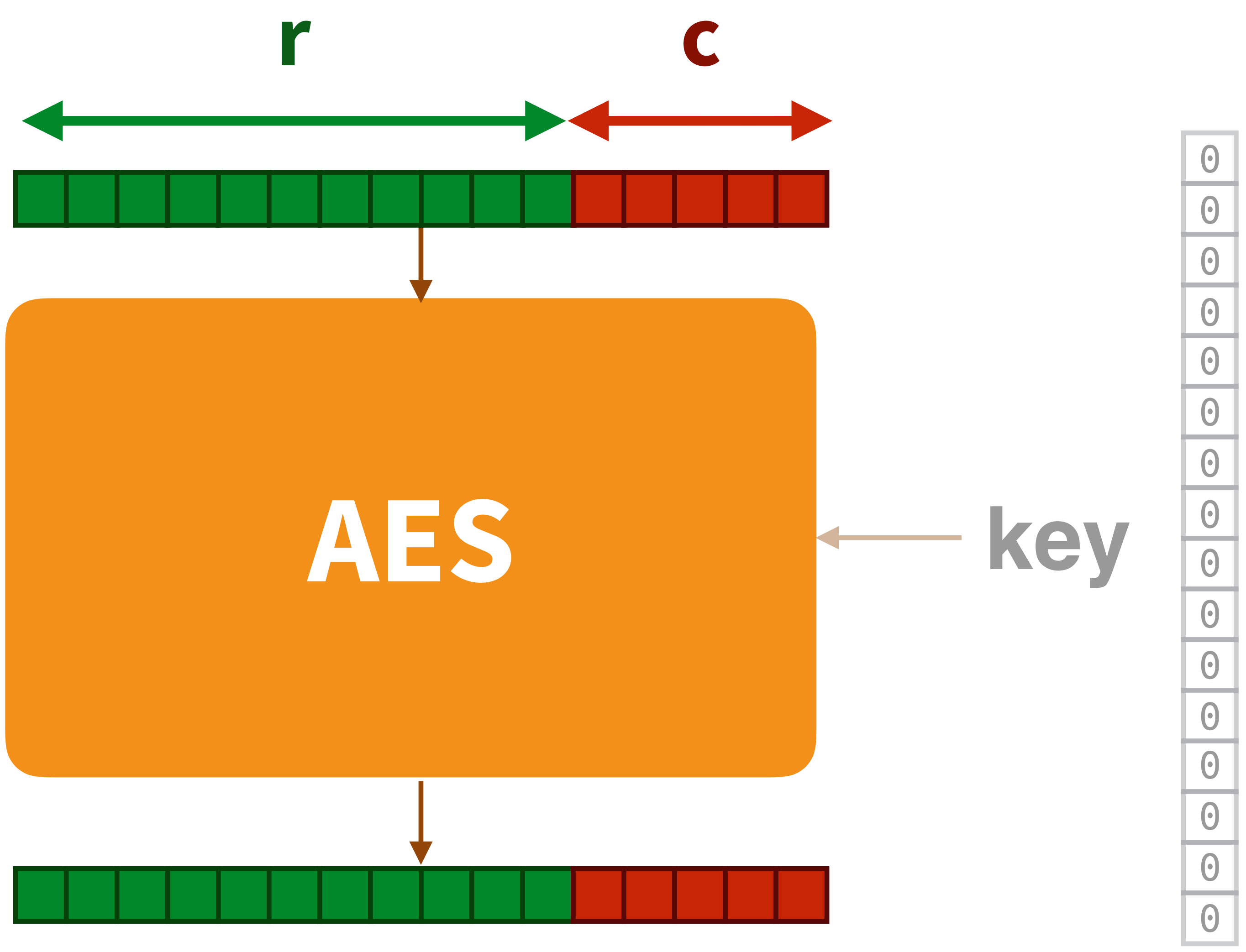


Sponge Construction

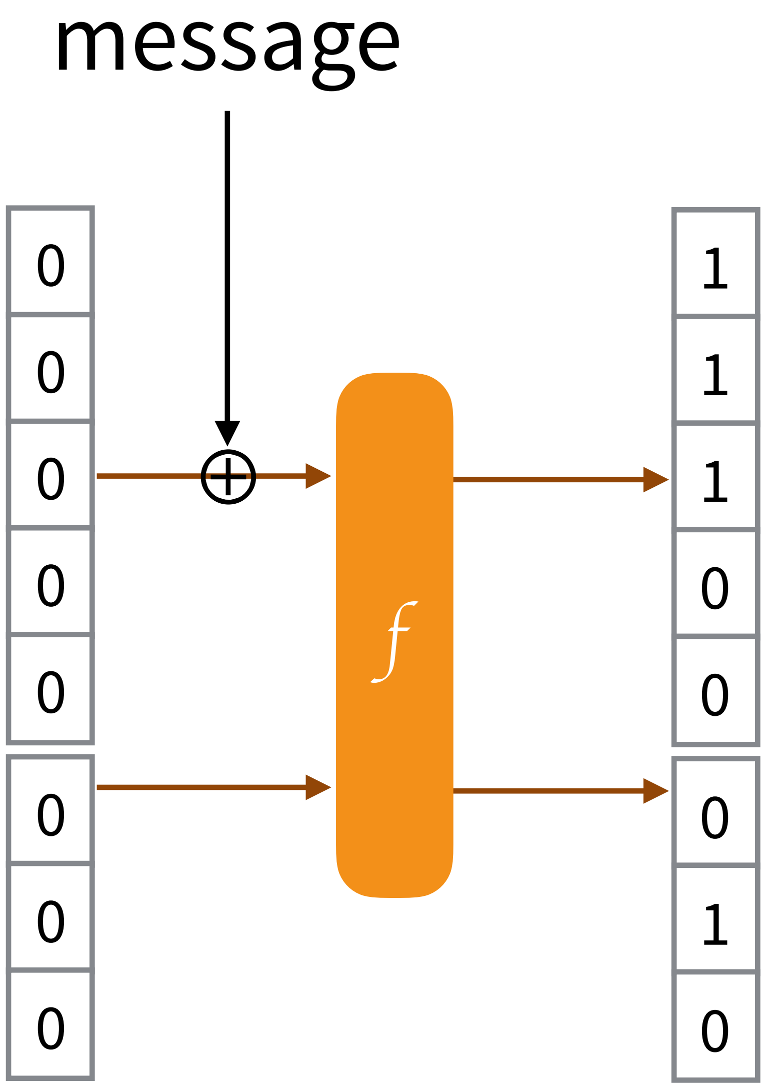


Sponge Construction

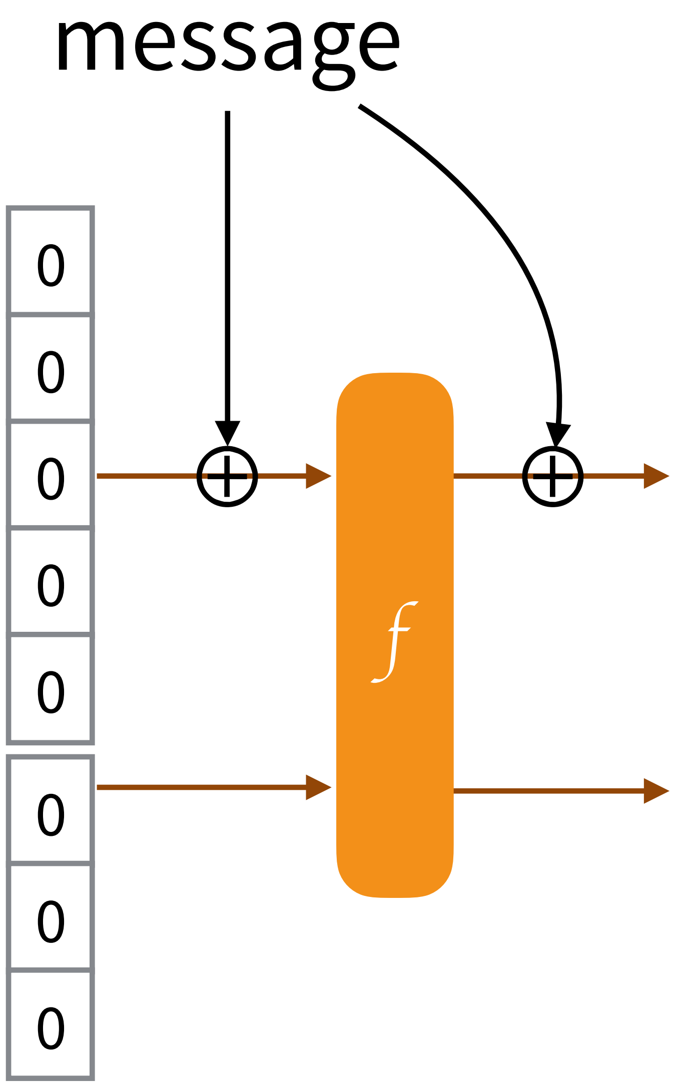




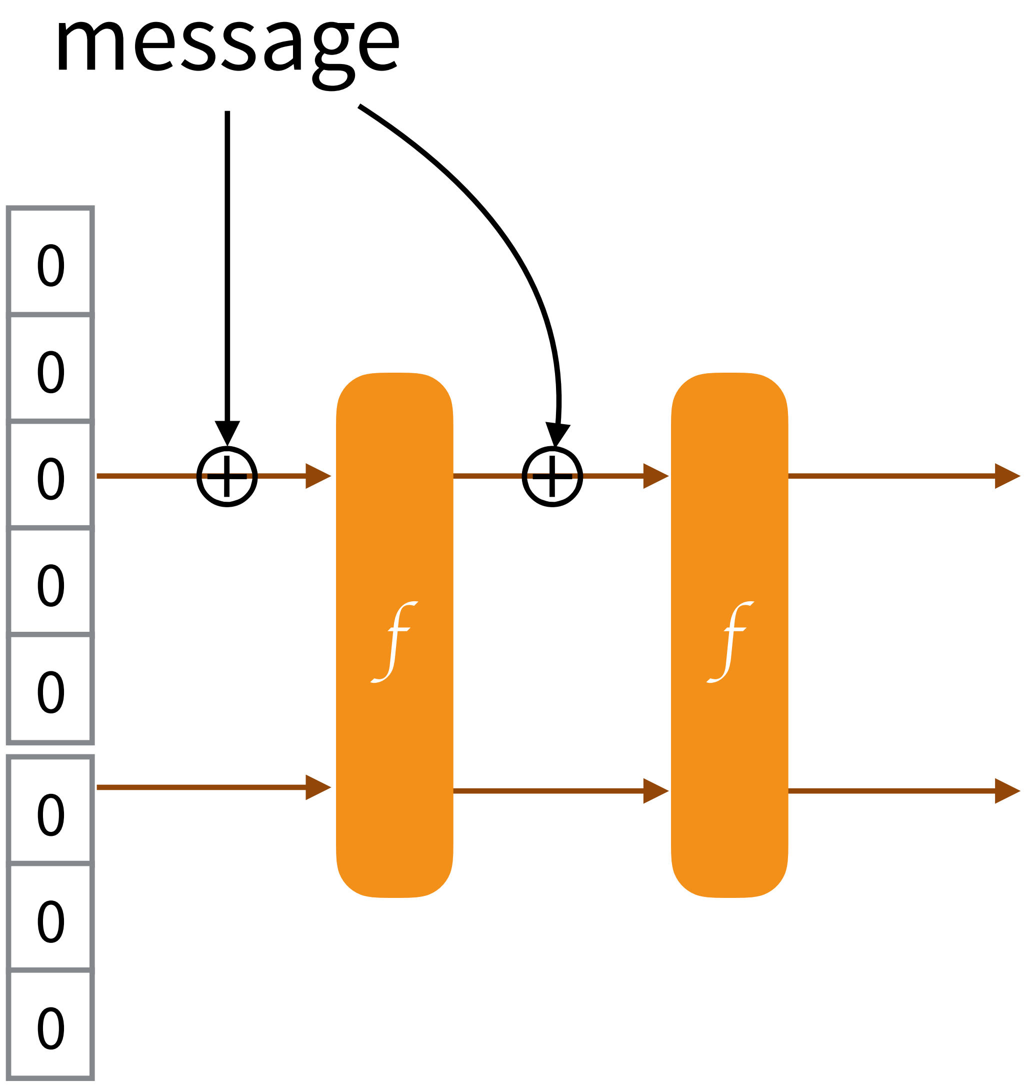
Sponge Construction



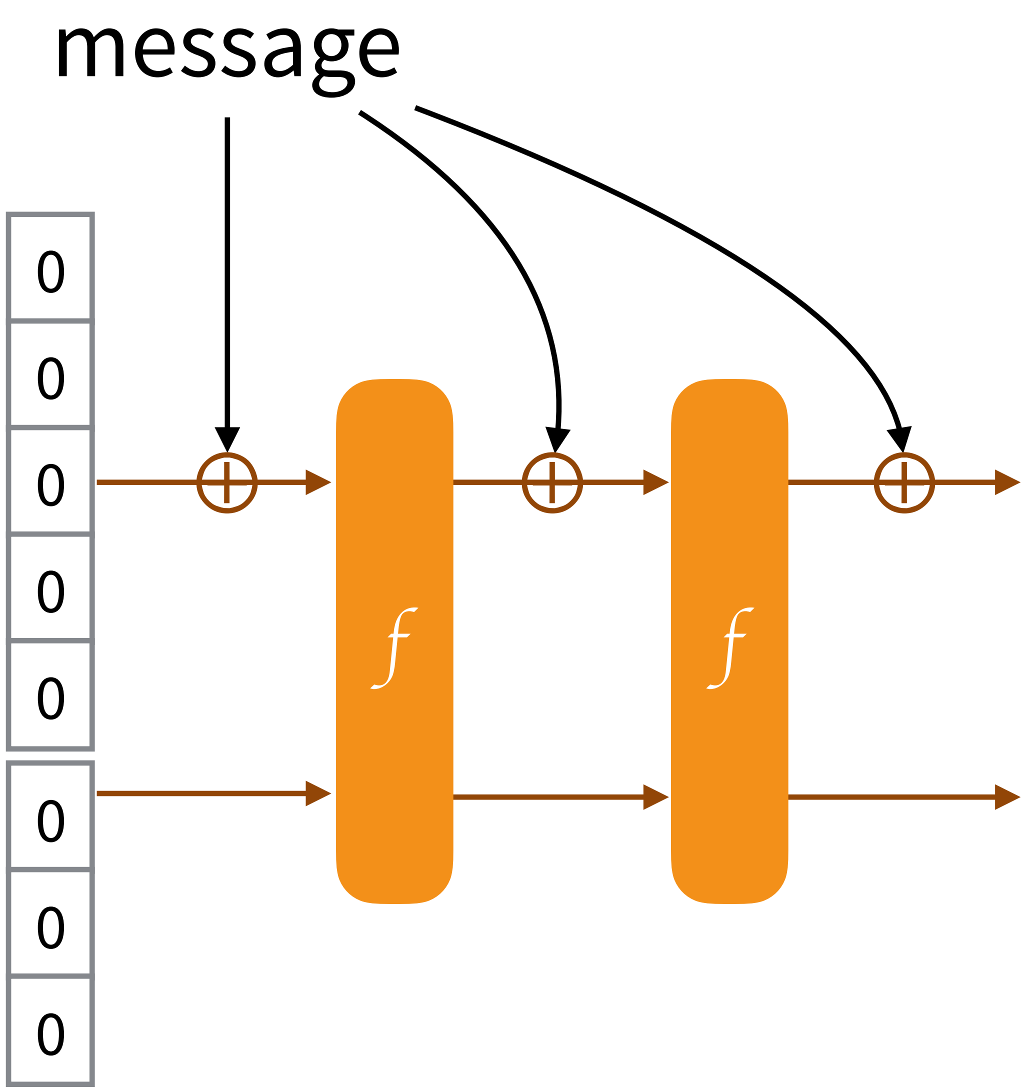
Sponge Construction



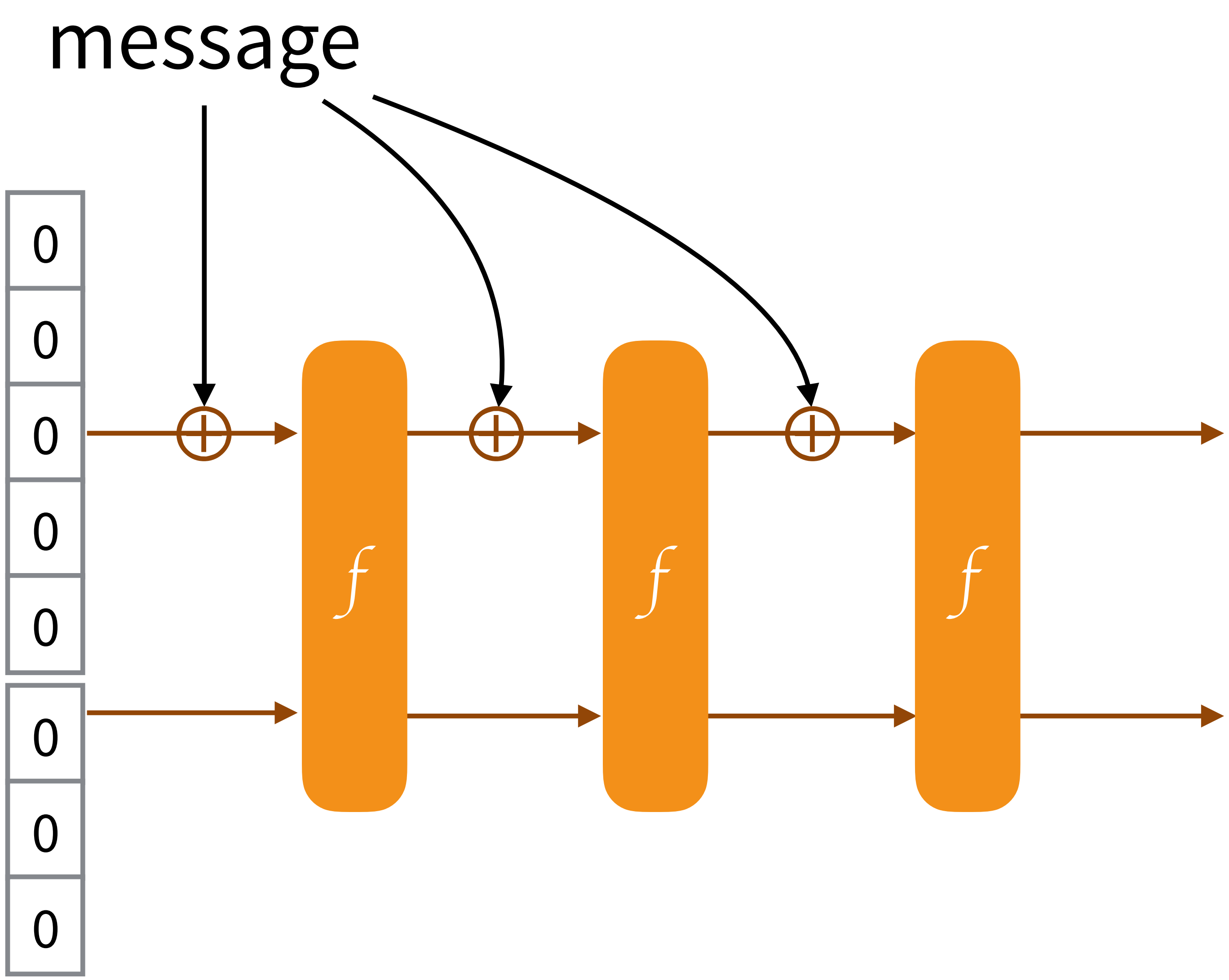
Sponge Construction



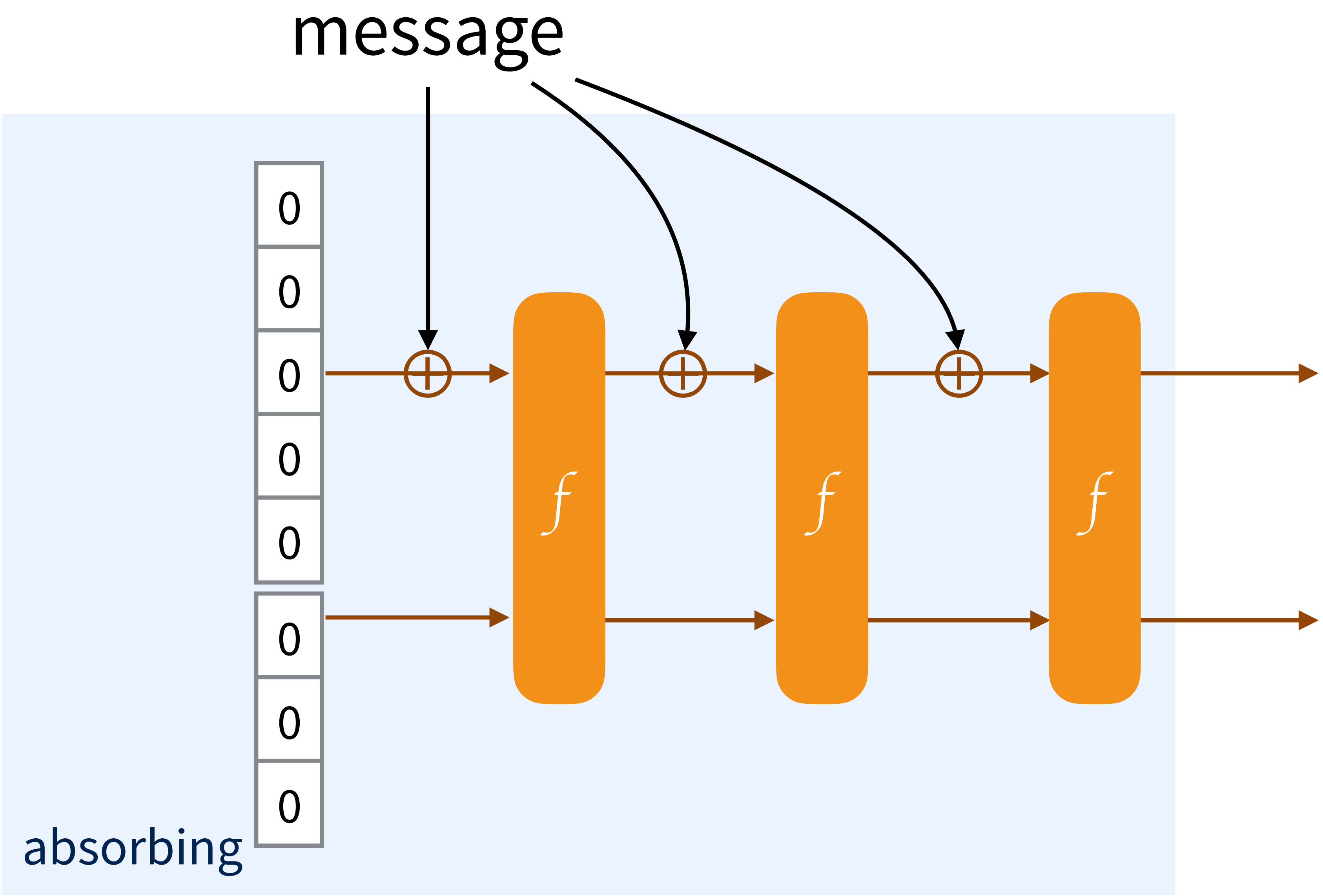
Sponge Construction



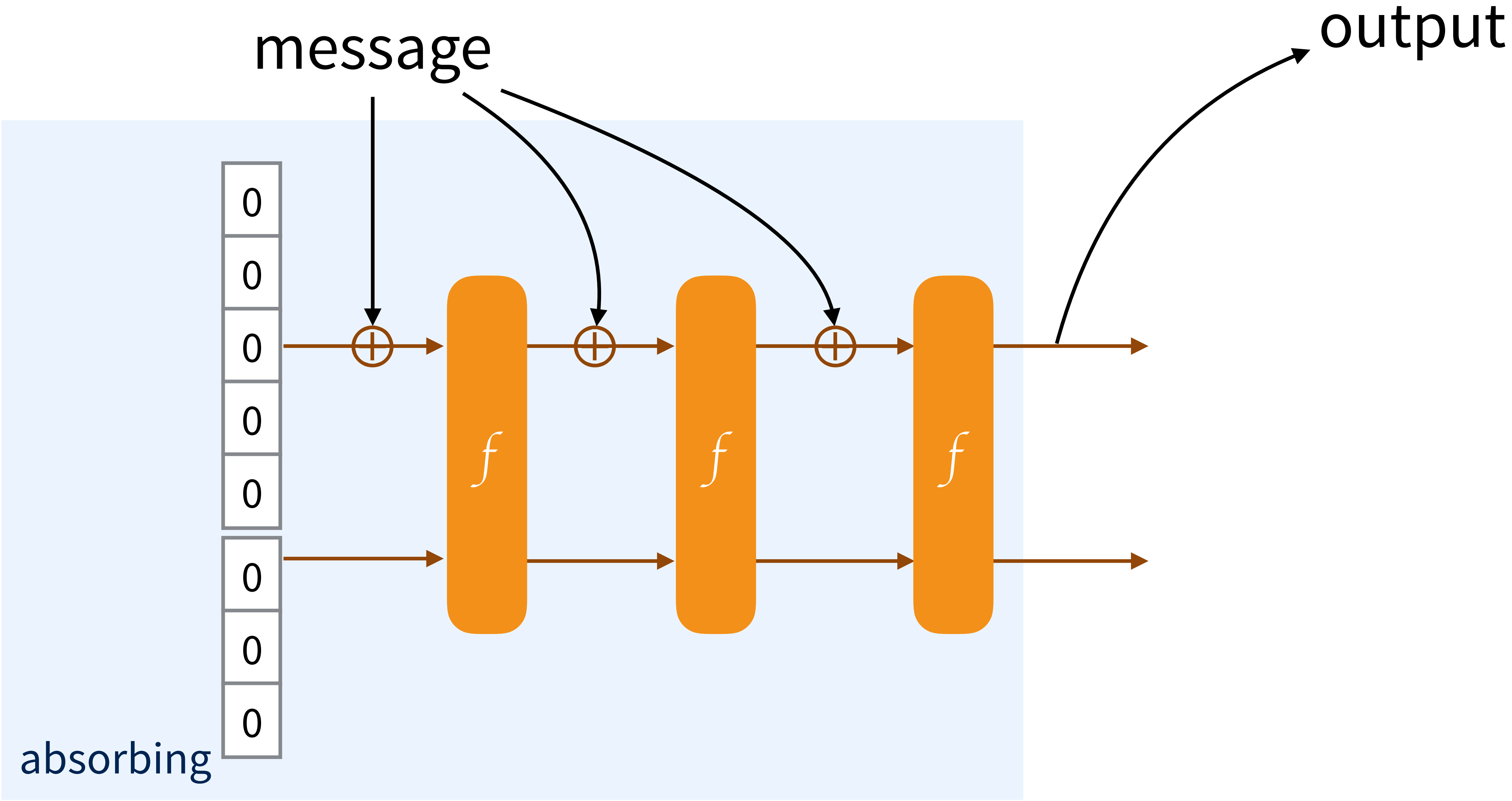
Sponge Construction



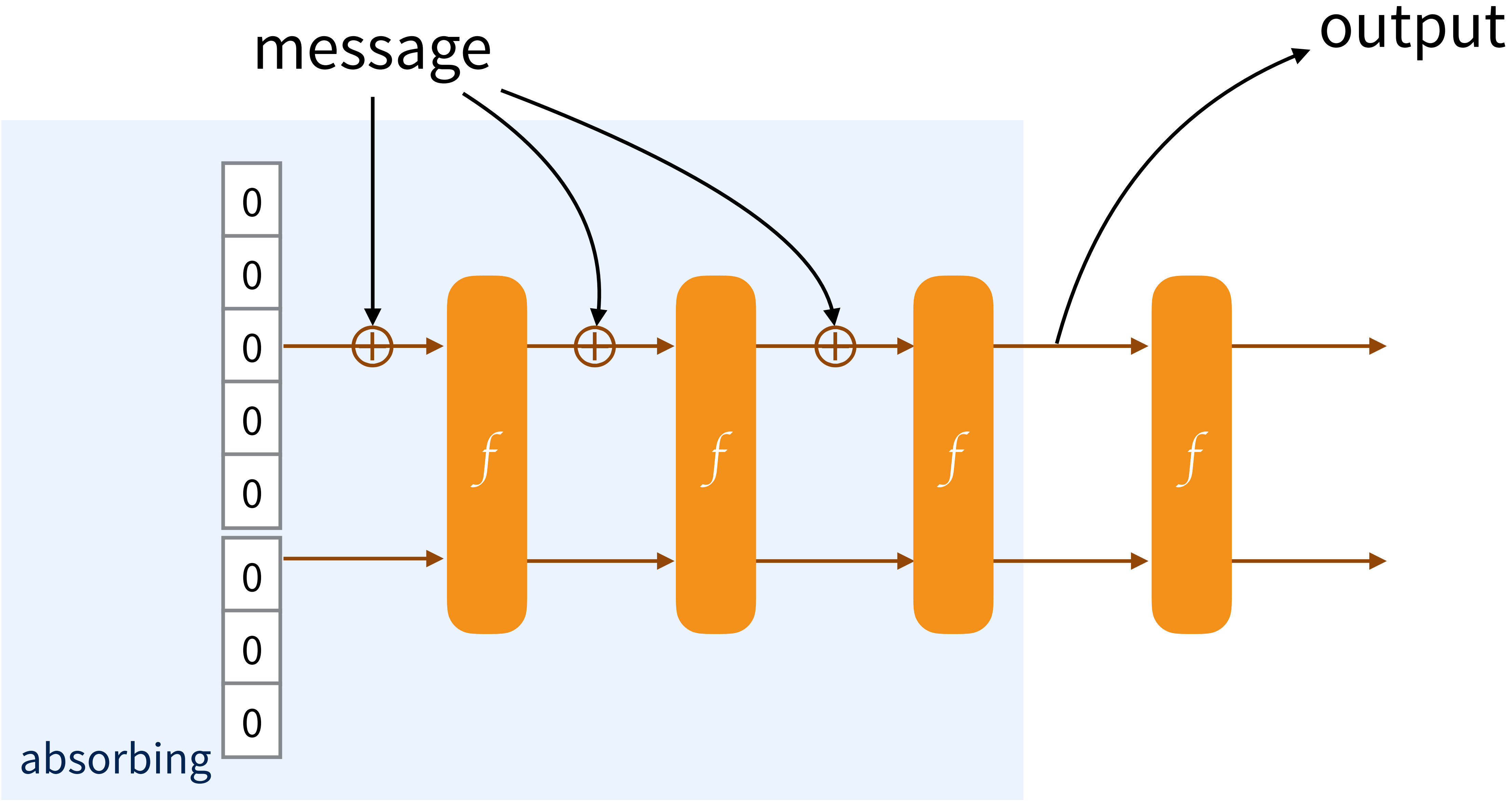
Sponge Construction



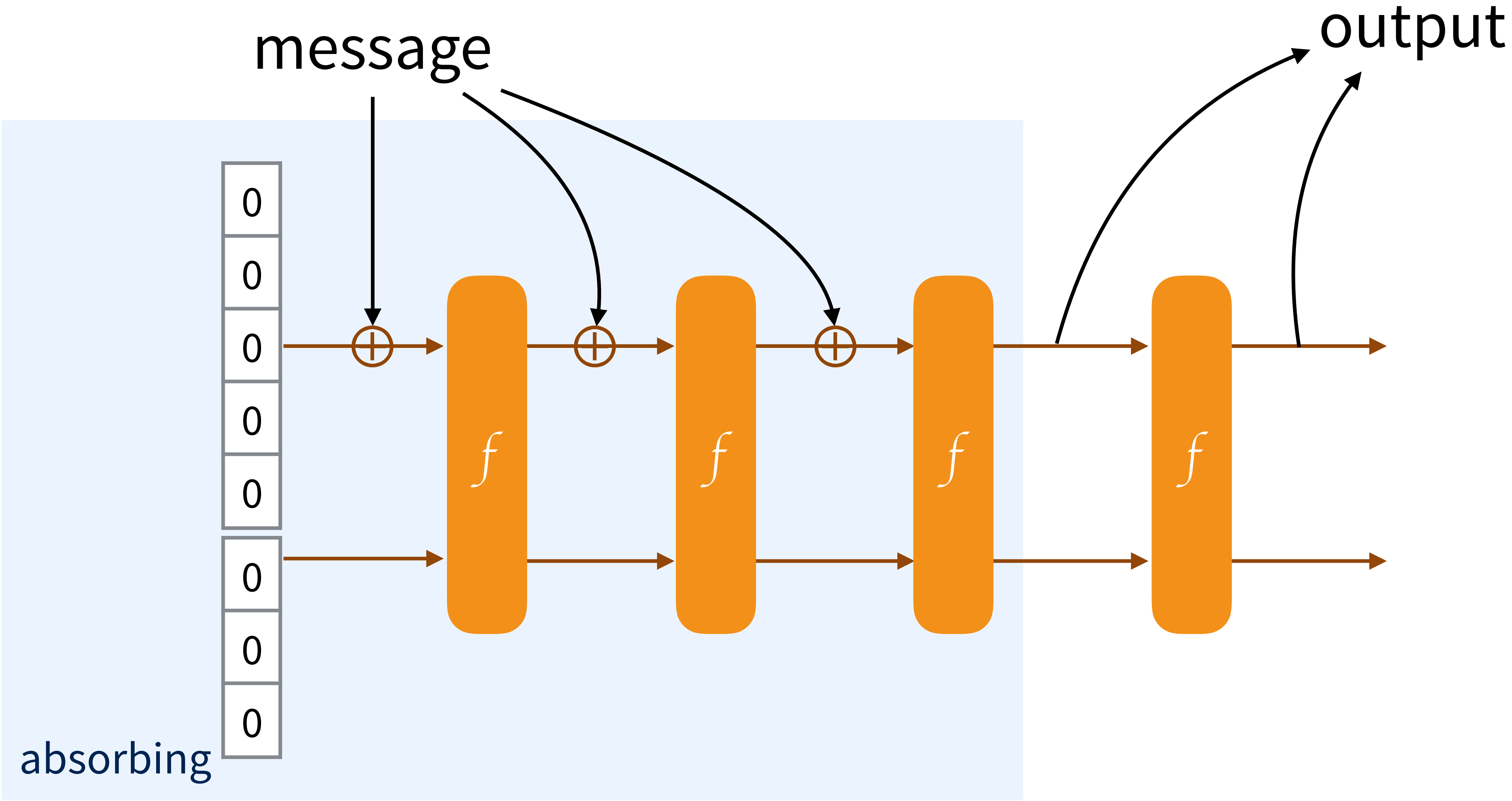
Sponge Construction



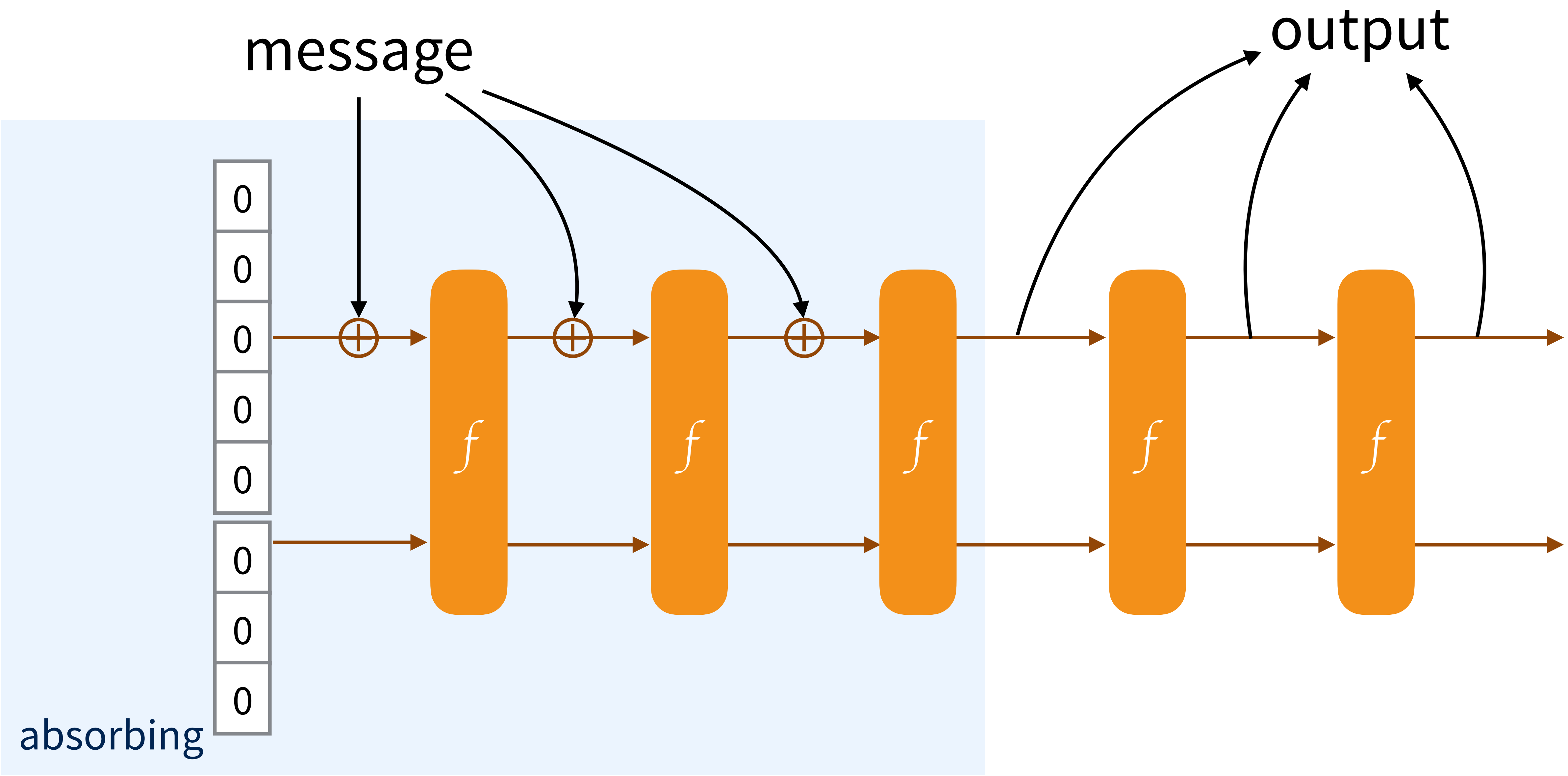
Sponge Construction



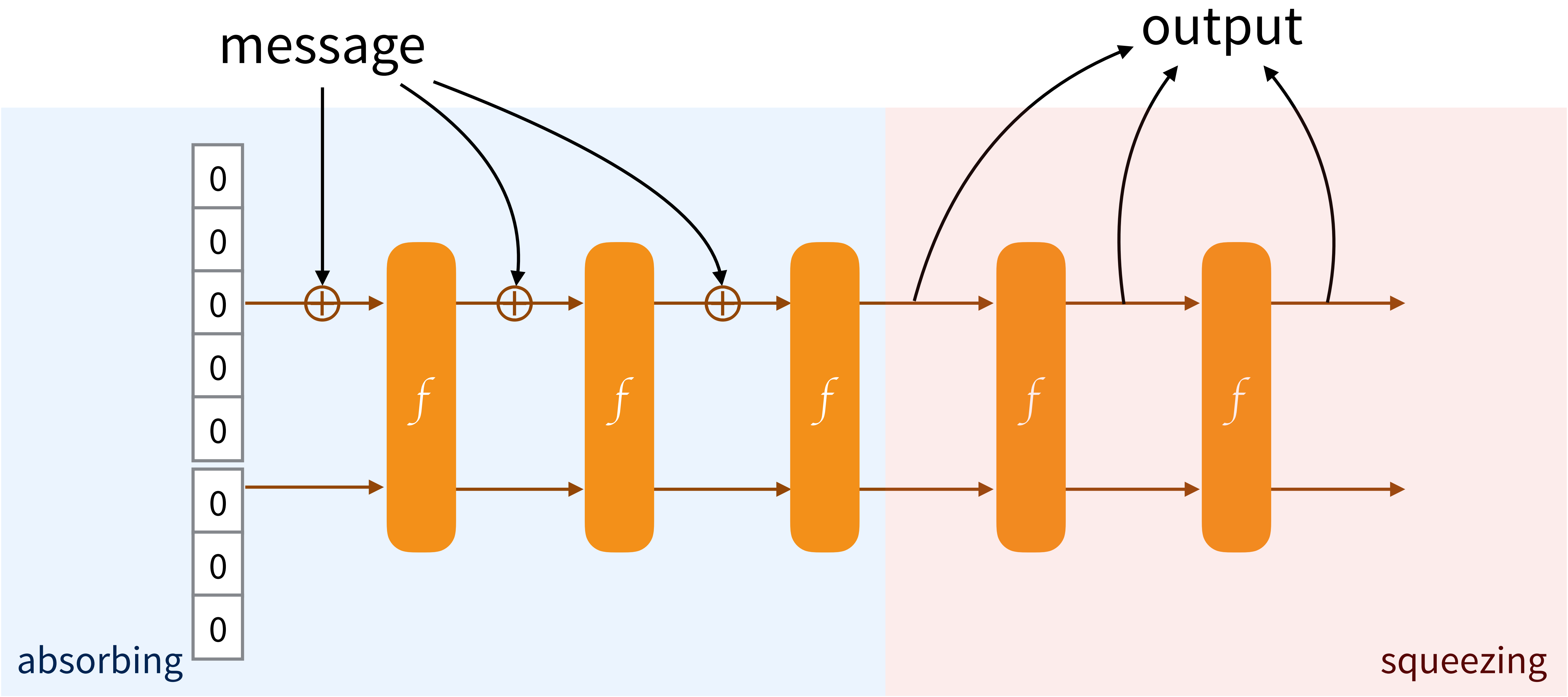
Sponge Construction

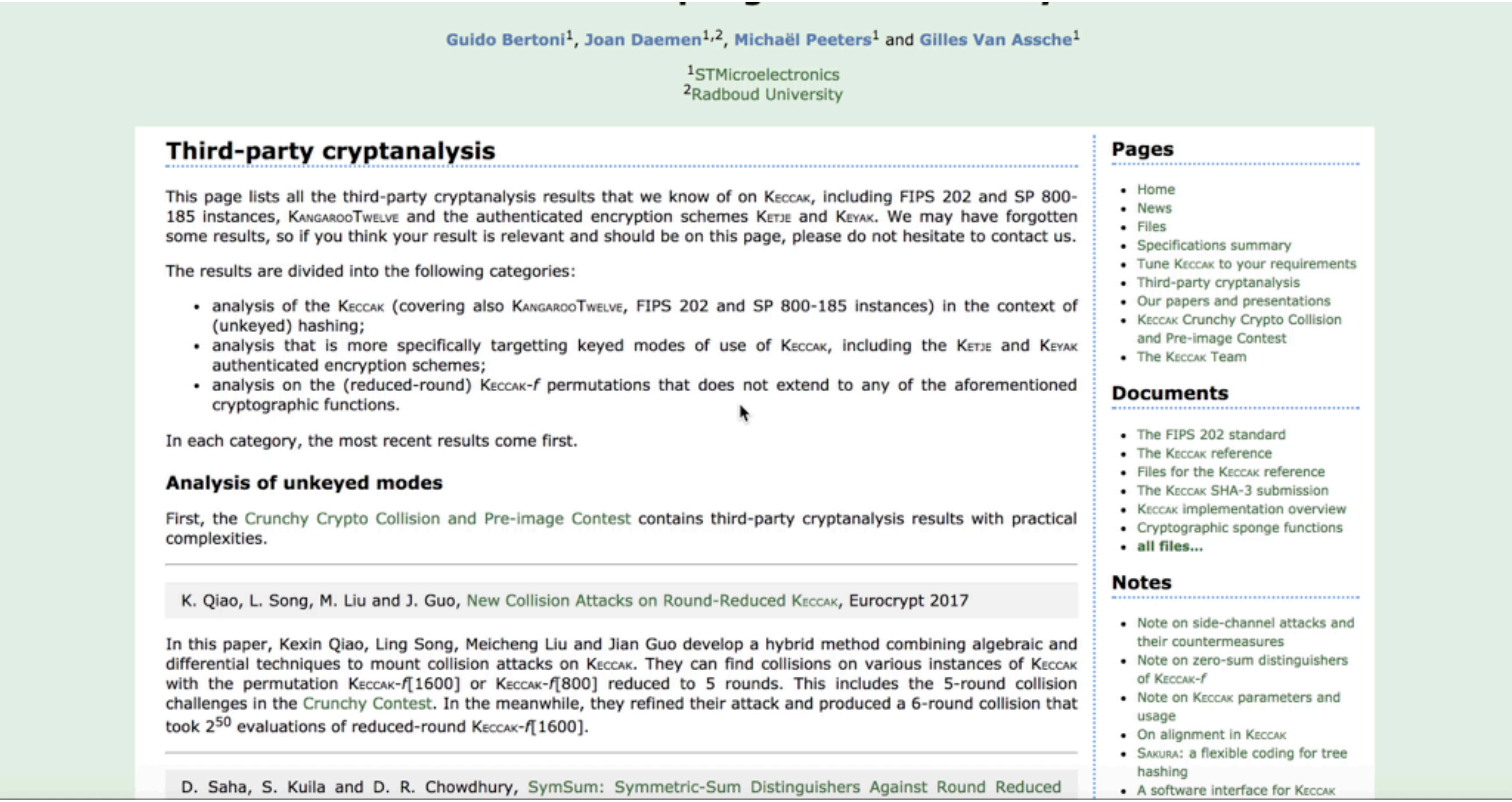


Sponge Construction



Sponge Construction





Guido Bertoni[1], Joan Daemen[1,2], Michaël Peeters[1] and Gilles Van Assche[1]

[1]STMicroelectronics
[2]Radboud University

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 targetting 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

Pages

- Home
- News
- Files
- Specifications summary
- Tune Keccak to your requirements
- Third-party cryptanalysis
- Our papers and presentations
- Keccak Crunchy Crypto Collision and Pre-image Contest
- The Keccak Team

Documents

- The FIPS 202 standard
- The Keccak reference
- Files for the Keccak reference
- The Keccak SHA-3 submission
- Keccak implementation overview
- Cryptographic sponge functions
- **all files...**

Notes

- Note on side-channel attacks and their countermeasures
- Note on zero-sum distinguishers of Keccak-*f*
- Note on Keccak parameters and usage
- On alignment in Keccak
- Sakura: a flexible coding for tree hashing
- A software interface for Keccak

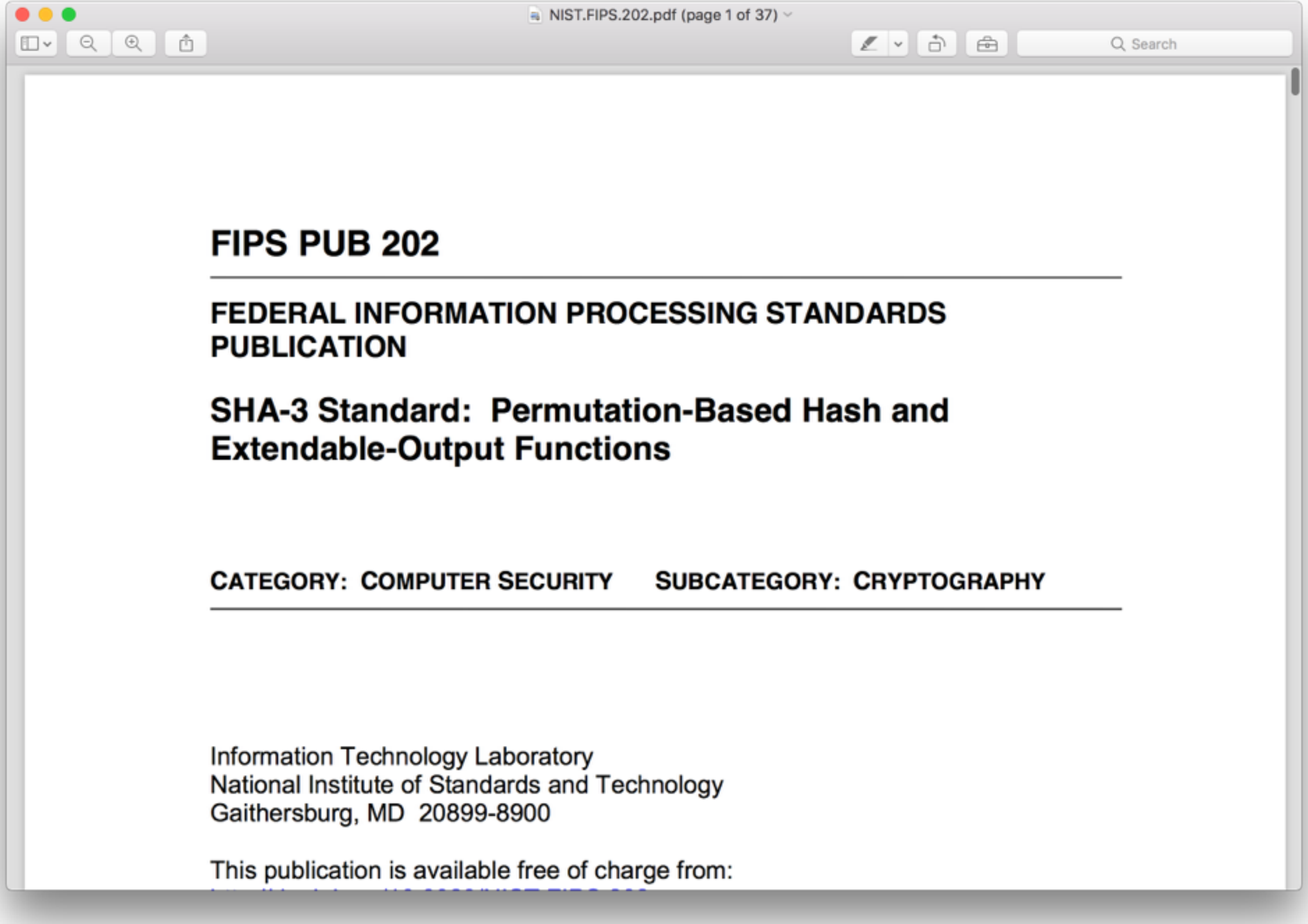
Keccak

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

2007

SHA-3 competition

2012



FIPS PUB 202

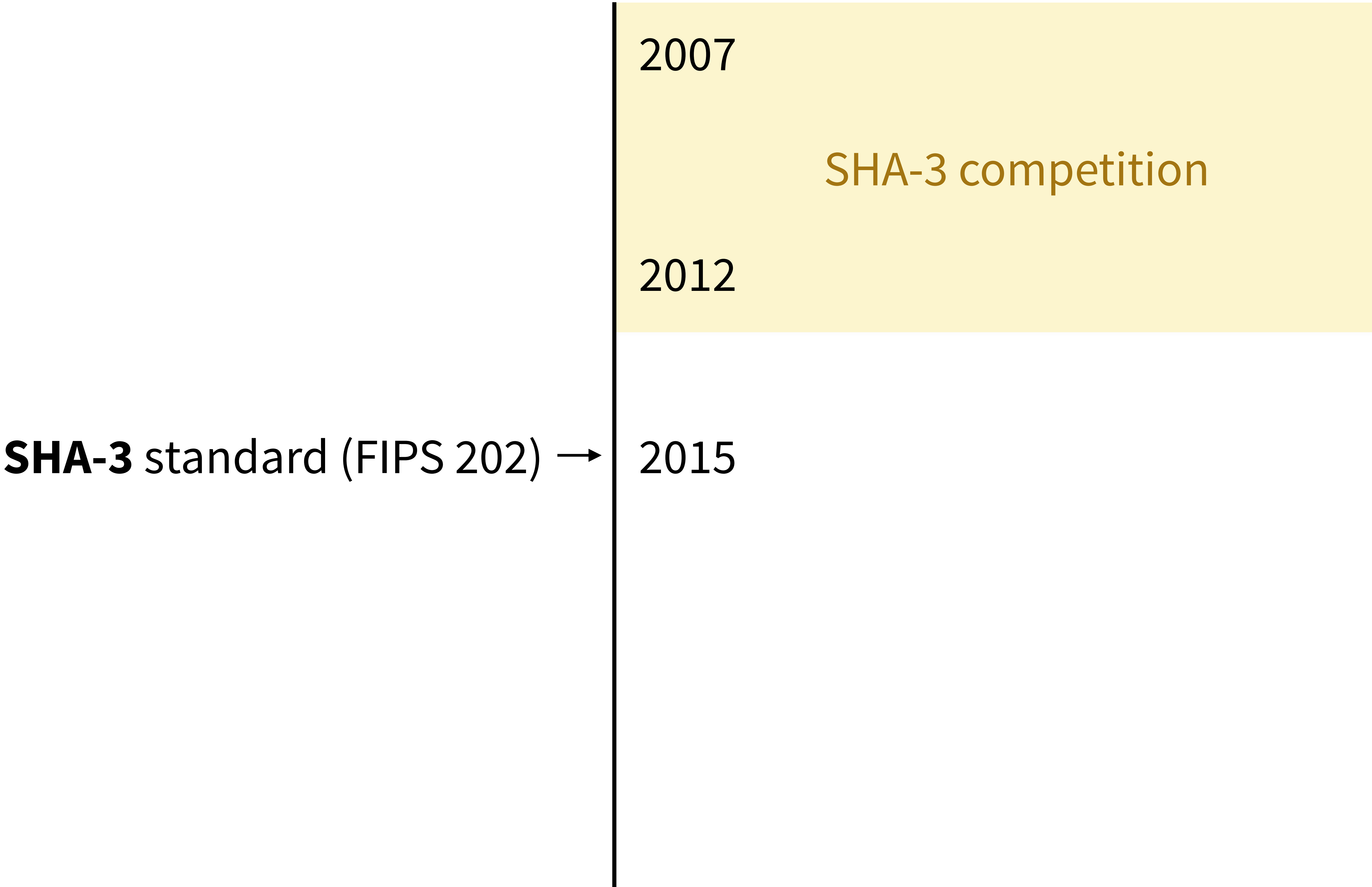
FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION

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:

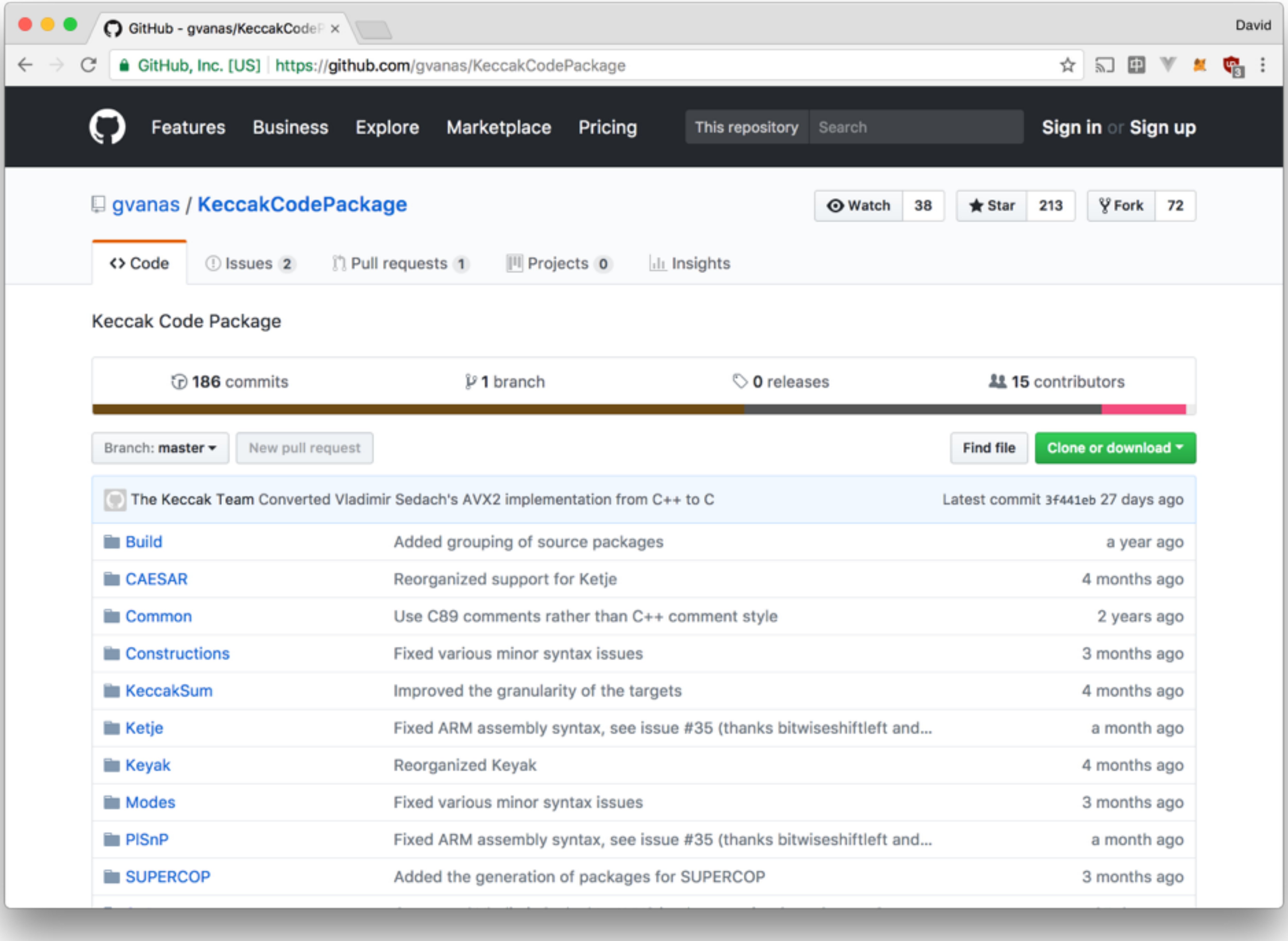


2007

SHA-3 competition

2012

SHA-3 standard (FIPS 202) → 2015

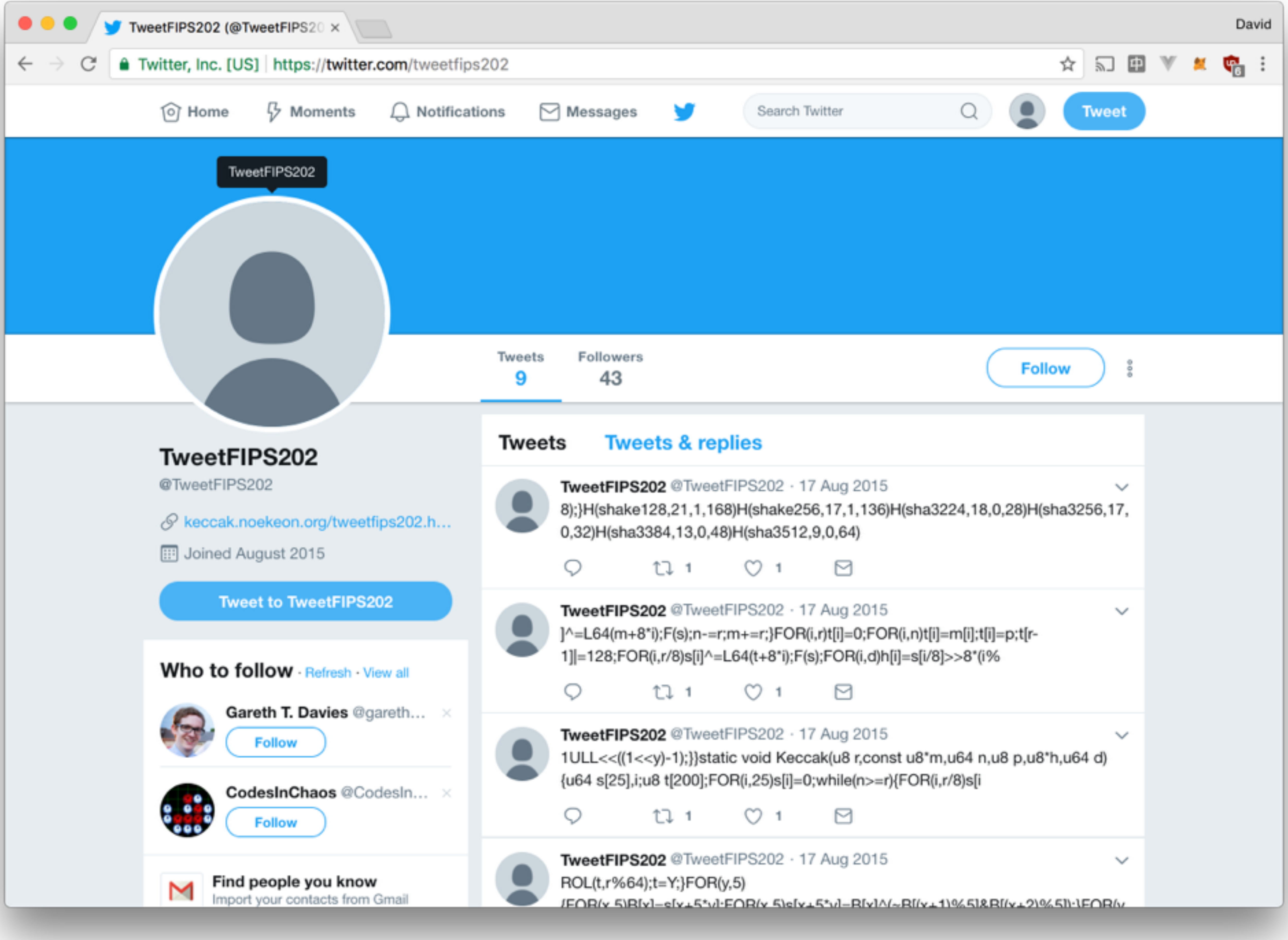


Keccak Code Package

186 commits | 1 branch | 0 releases | 15 contributors

The Keccak Team Converted Vladimir Sedach's AVX2 implementation from C++ to C — Latest commit 3f441eb 27 days ago

Build	Added grouping of source packages	a year ago
CAESAR	Reorganized support for Ketje	4 months ago
Common	Use C89 comments rather than C++ comment style	2 years ago
Constructions	Fixed various minor syntax issues	3 months ago
KeccakSum	Improved the granularity of the targets	4 months ago
Ketje	Fixed ARM assembly syntax, see issue #35 (thanks bitwiseshiftleft and…	a month ago
Keyak	Reorganized Keyak	4 months ago
Modes	Fixed various minor syntax issues	3 months ago
PlSnP	Fixed ARM assembly syntax, see issue #35 (thanks bitwiseshiftleft and…	a month ago
SUPERCOP	Added the generation of packages for SUPERCOP	3 months ago



TweetFIPS202

@TweetFIPS202

keccak.noekeon.org/tweetfips202.h...

Joined August 2015

TweetFIPS202 @TweetFIPS202 · 17 Aug 2015

```
8);}H(shake128,21,1,168)H(shake256,17,1,136)H(sha3224,18,0,28)H(sha3256,17,
0,32)H(sha3384,13,0,48)H(sha3512,9,0,64)
```

TweetFIPS202 @TweetFIPS202 · 17 Aug 2015

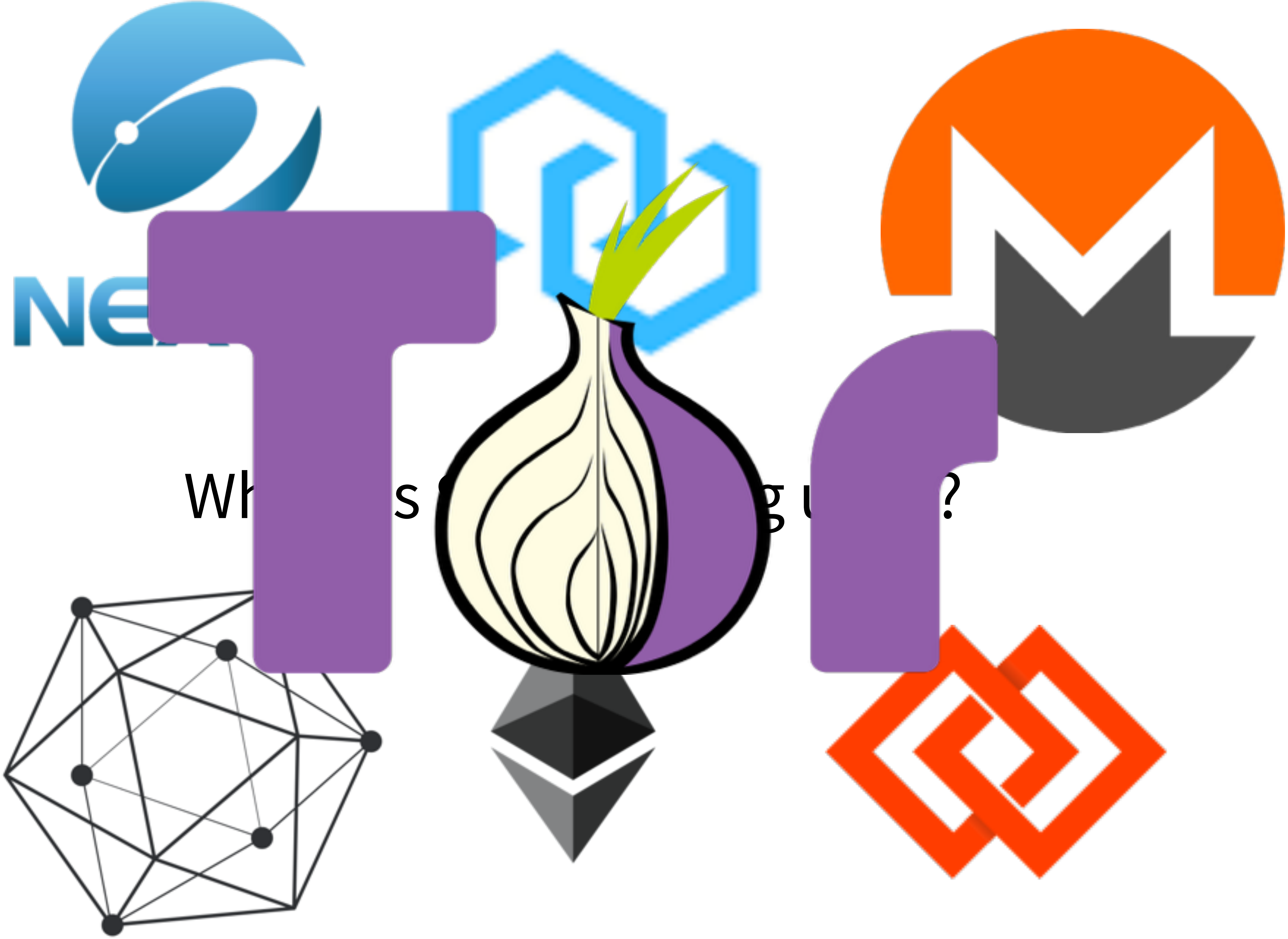
```
]^=L64(m+8*i);F(s);n-=r;m+=r;}FOR(i,r)t[i]=0;FOR(i,n)t[i]=m[i];t[i]=p;t[r-
1]|=128;FOR(i,r/8)s[i]^=L64(t+8*i);F(s);FOR(i,d)h[i]=s[i/8]>>8*(i%
```

TweetFIPS202 @TweetFIPS202 · 17 Aug 2015

```
1ULL<<((1<<y)-1);}}static void Keccak(u8 r,const u8*m,u64 n,u8 p,u8*h,u64 d)
{u64 s[25],i;u8 t[200];FOR(i,25)s[i]=0;while(n>=r){FOR(i,r/8)s[i
```

TweetFIPS202 @TweetFIPS202 · 17 Aug 2015

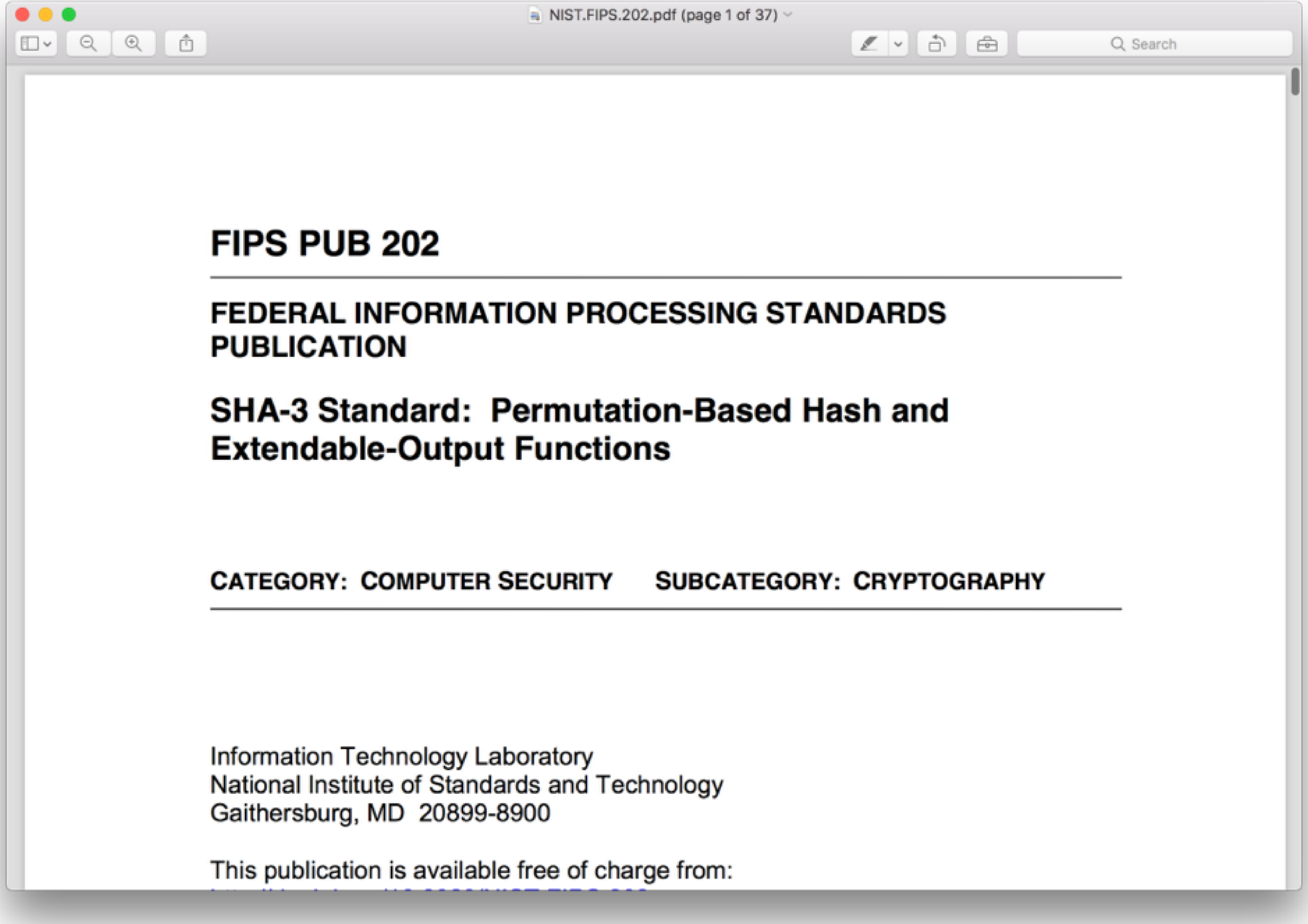
```
ROL(t,r%64);t=Y;}FOR(y,5)
{FOR(x,5)B[x]=s[x+5*y];FOR(x,5)s[x+5*y]=B[x]^(~B[(x+1)%5]&B[(x+2)%5]);}FOR(y
```



Wh s g u ?

Outline

1. SHA-3
2. **derived functions**
3. derived protocols



FIPS PUB 202

FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION

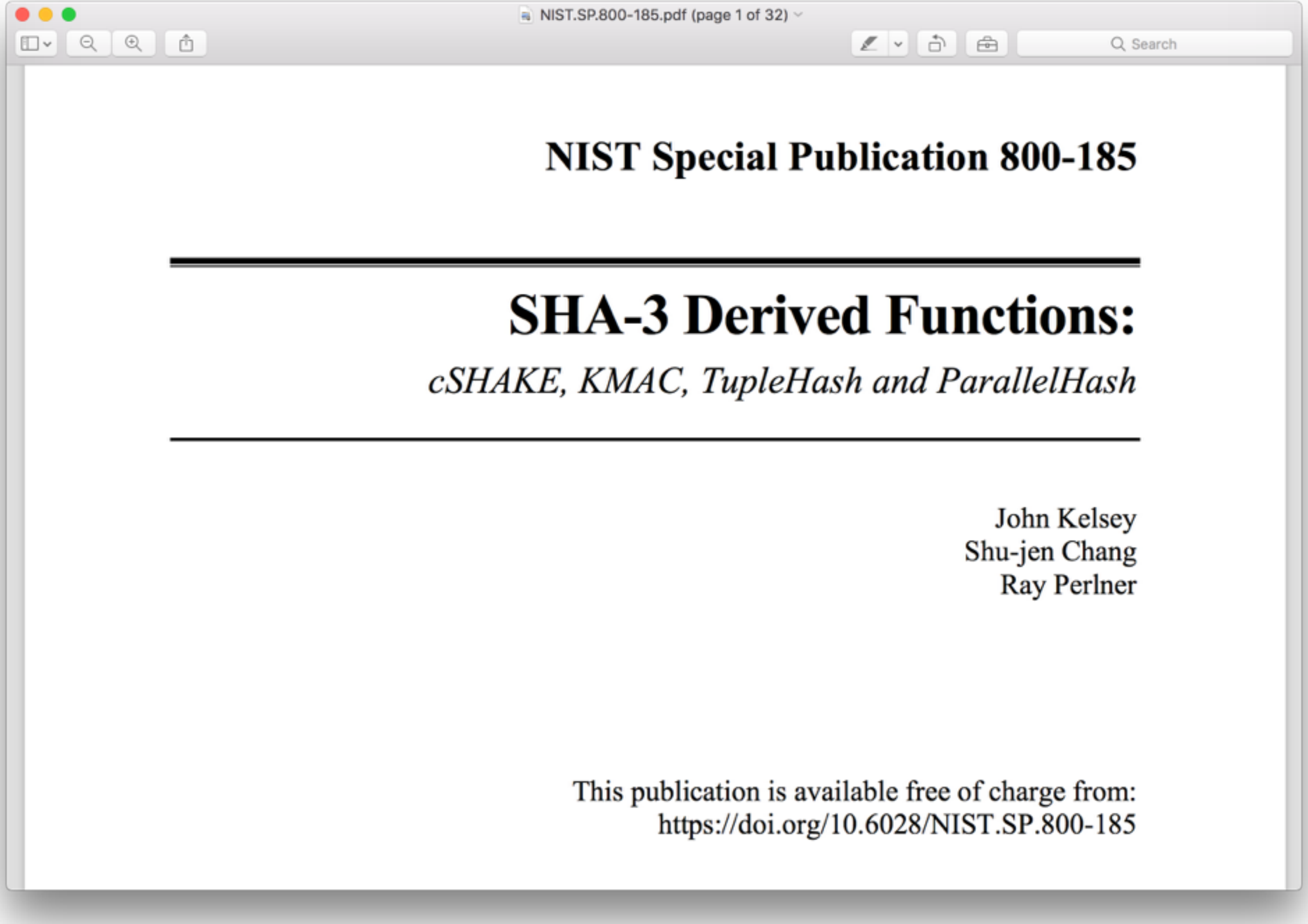
SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions

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

2007

SHA-3 competition

2012

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

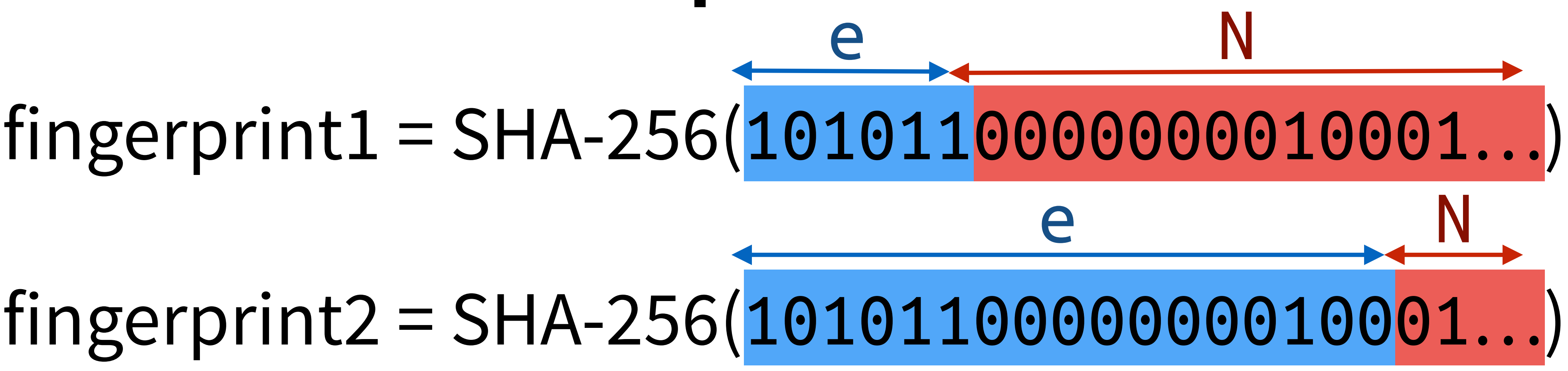
$$\text{fingerprint1} = \text{SHA-256}(\overbrace{\texttt{101011}}^{e}\overbrace{\texttt{0000000010001...}}^{N})$$

ParallelHash

KMAC

message || SHAKE(key || message)

TupleHash



fingerprint1 = SHA-256(10101100000000100001...)

fingerprint2 = SHA-256(10101100000000100001...)

ParallelHash

KMAC

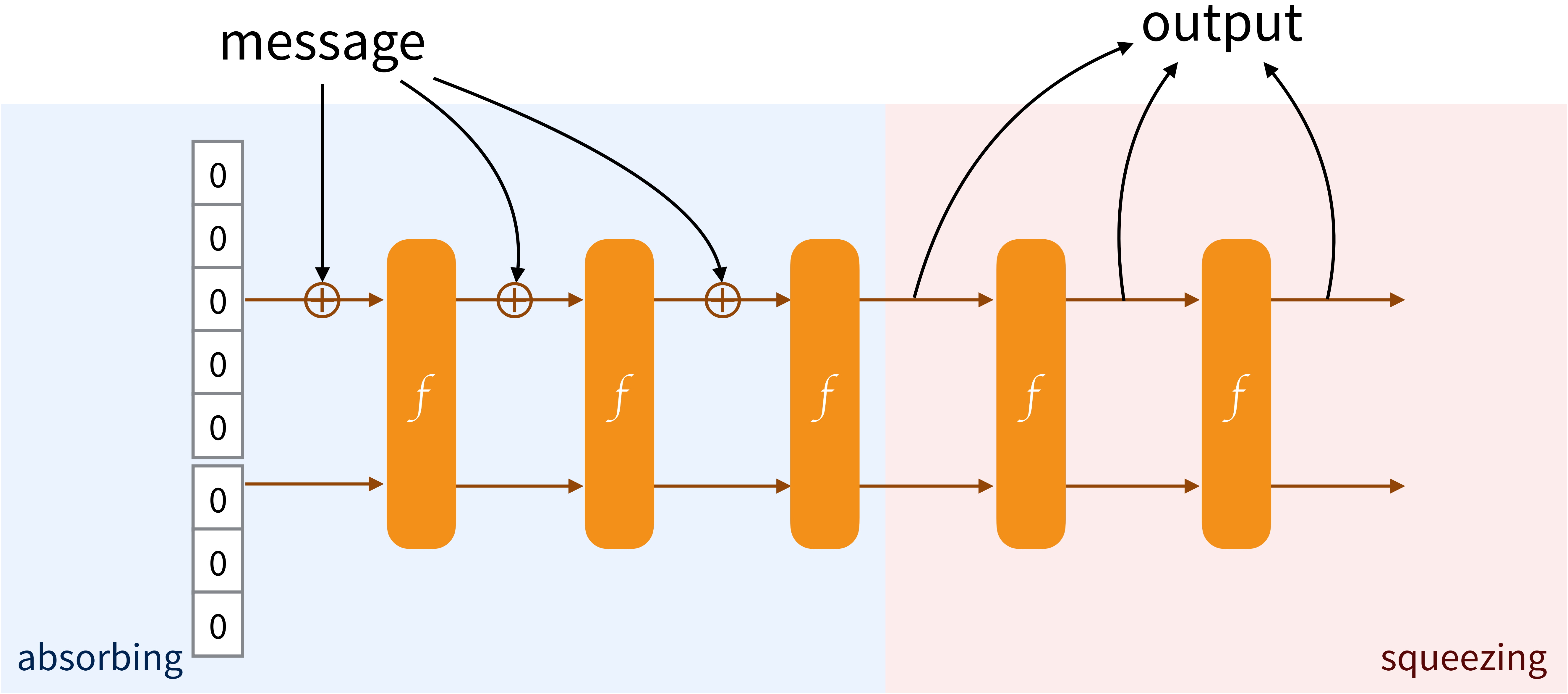
message || SHAKE(key || message)

TupleHash

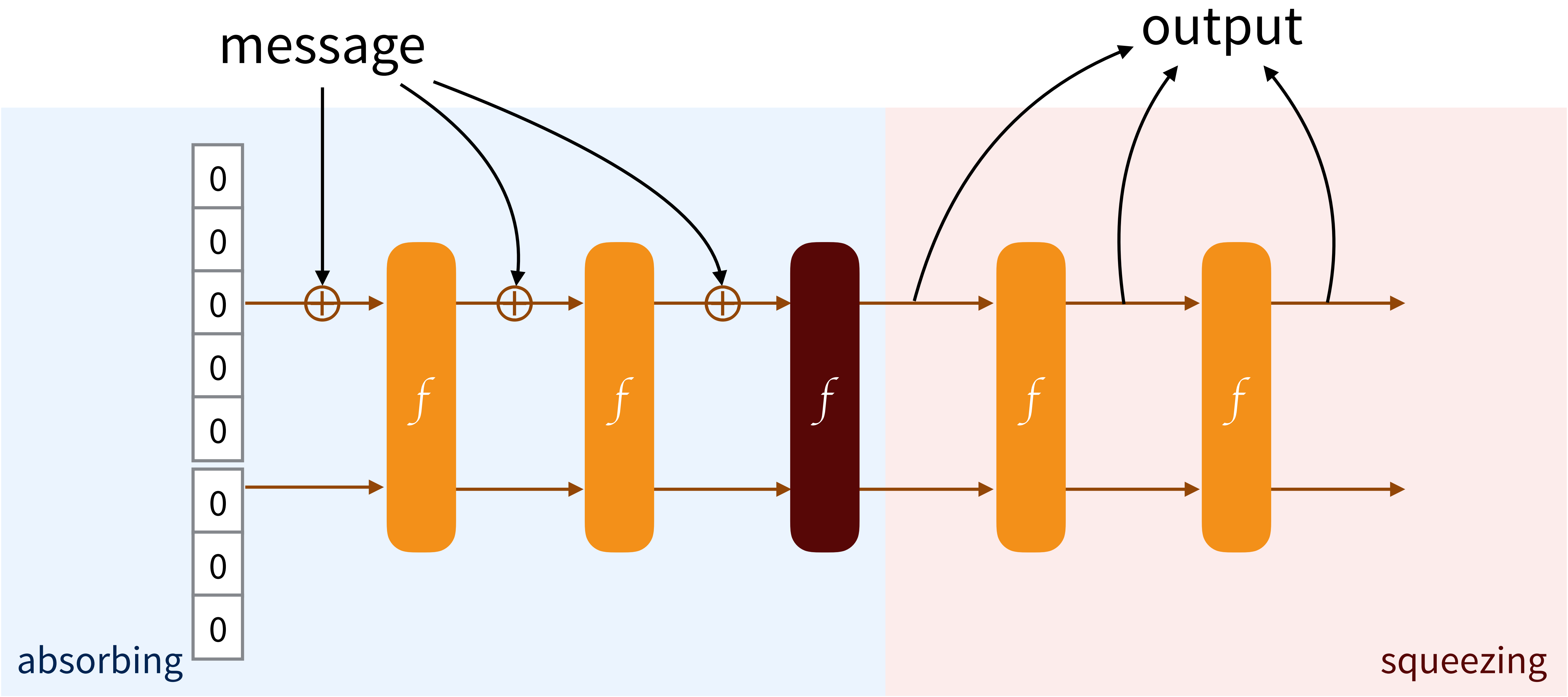
SHAKE(len(e) || e || len(N) || N)

ParallelHash

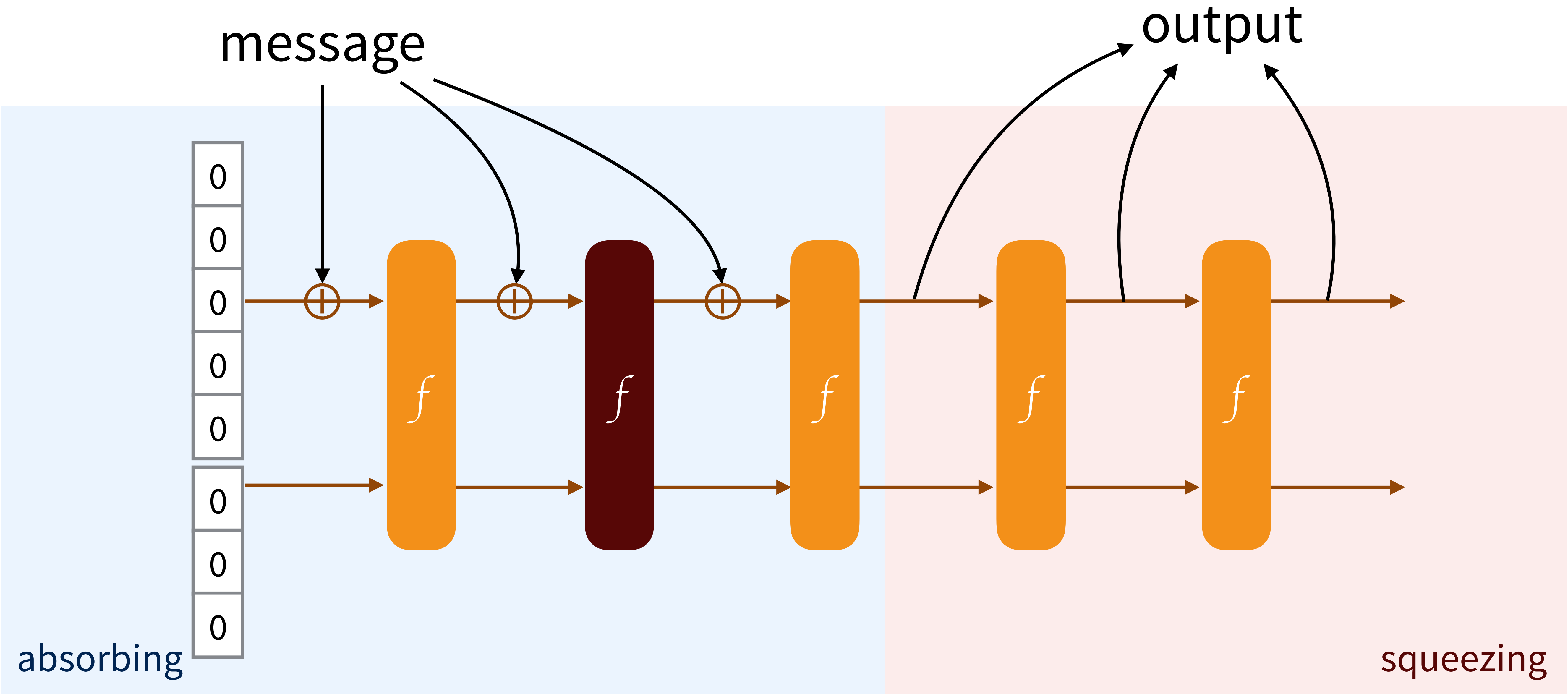
Sponge Construction



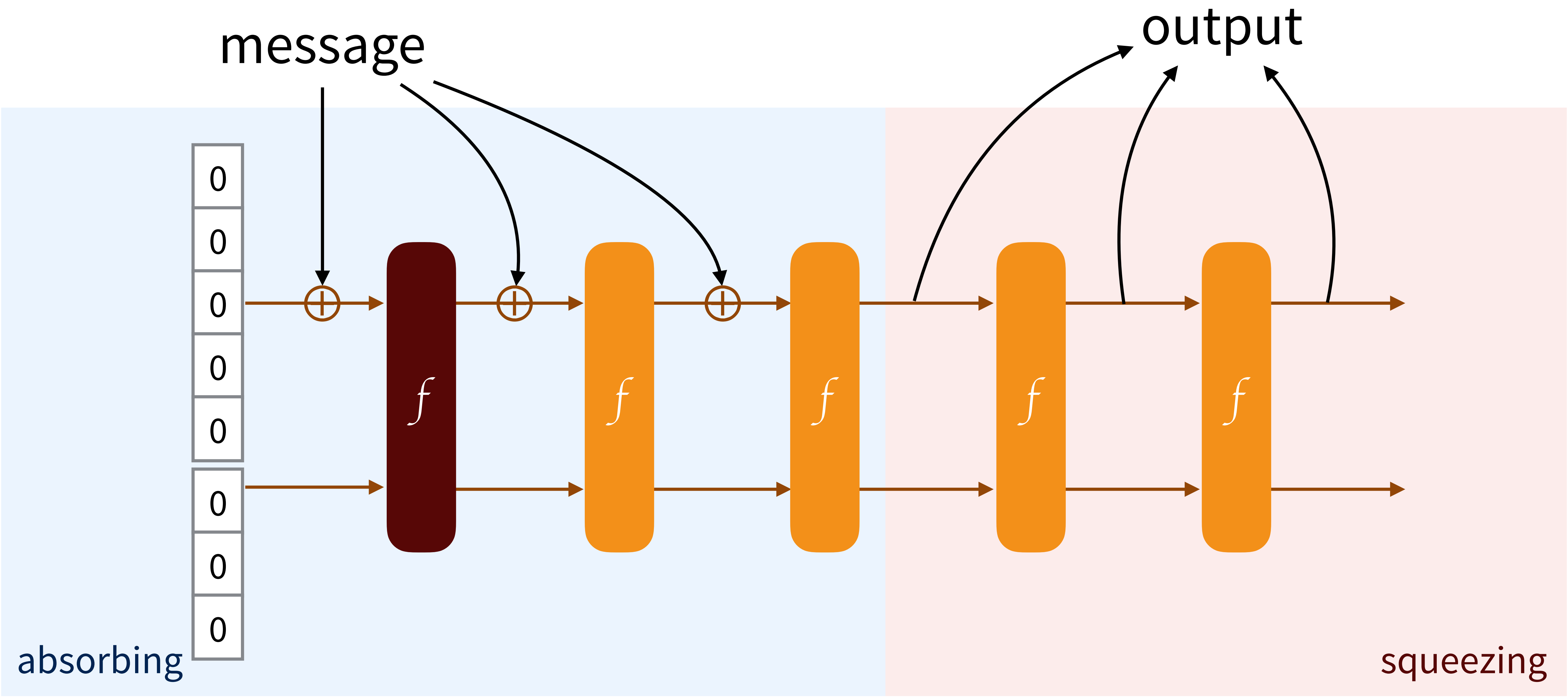
Sponge Construction



Sponge Construction



Sponge Construction



KMAC

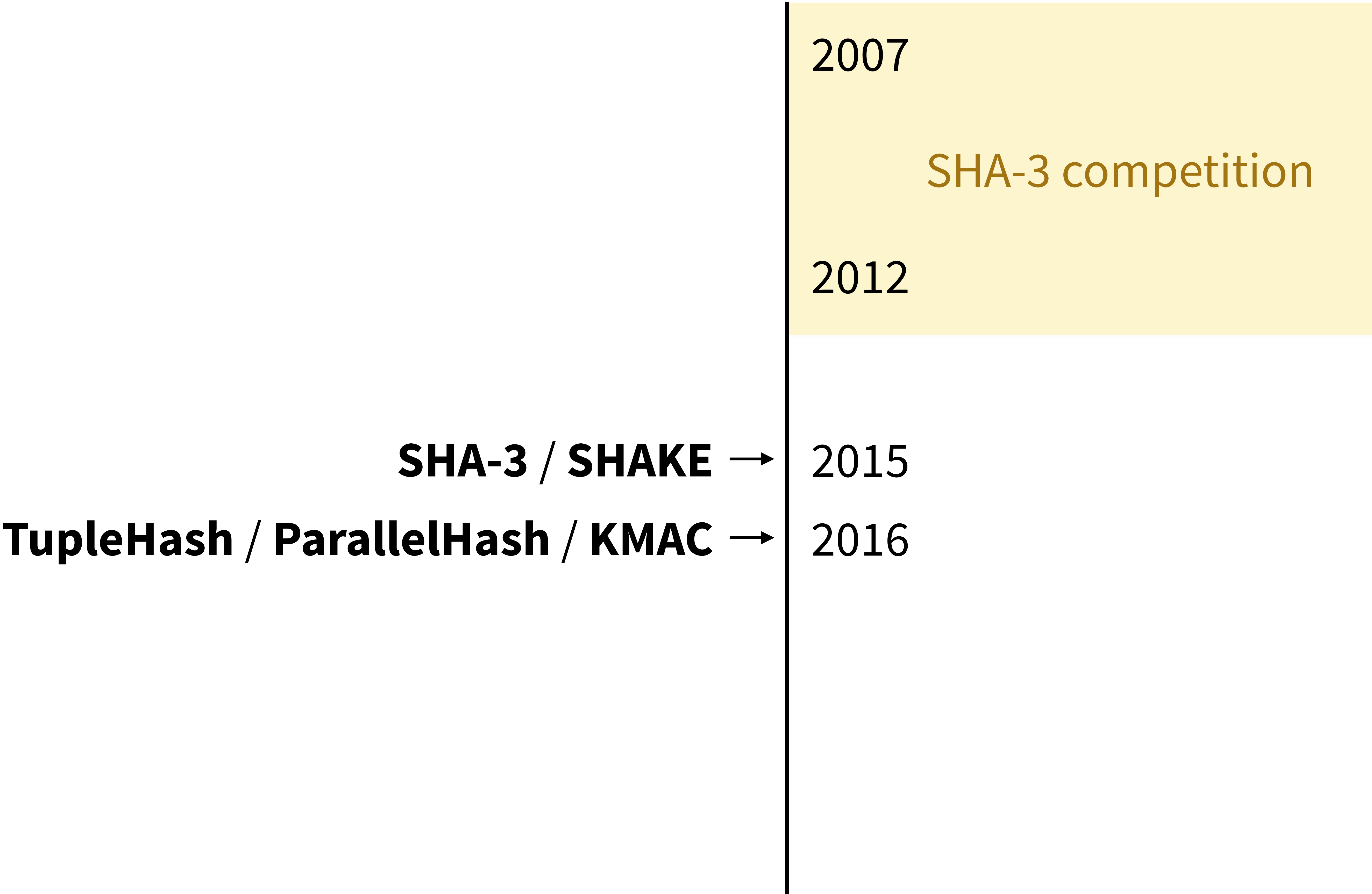
message || SHAKE(key || message)

TupleHash

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

ParallelHash

SHAKE(SHAKE(b1) || SHAKE(b2) || SHAKE(b3) || …)



2007

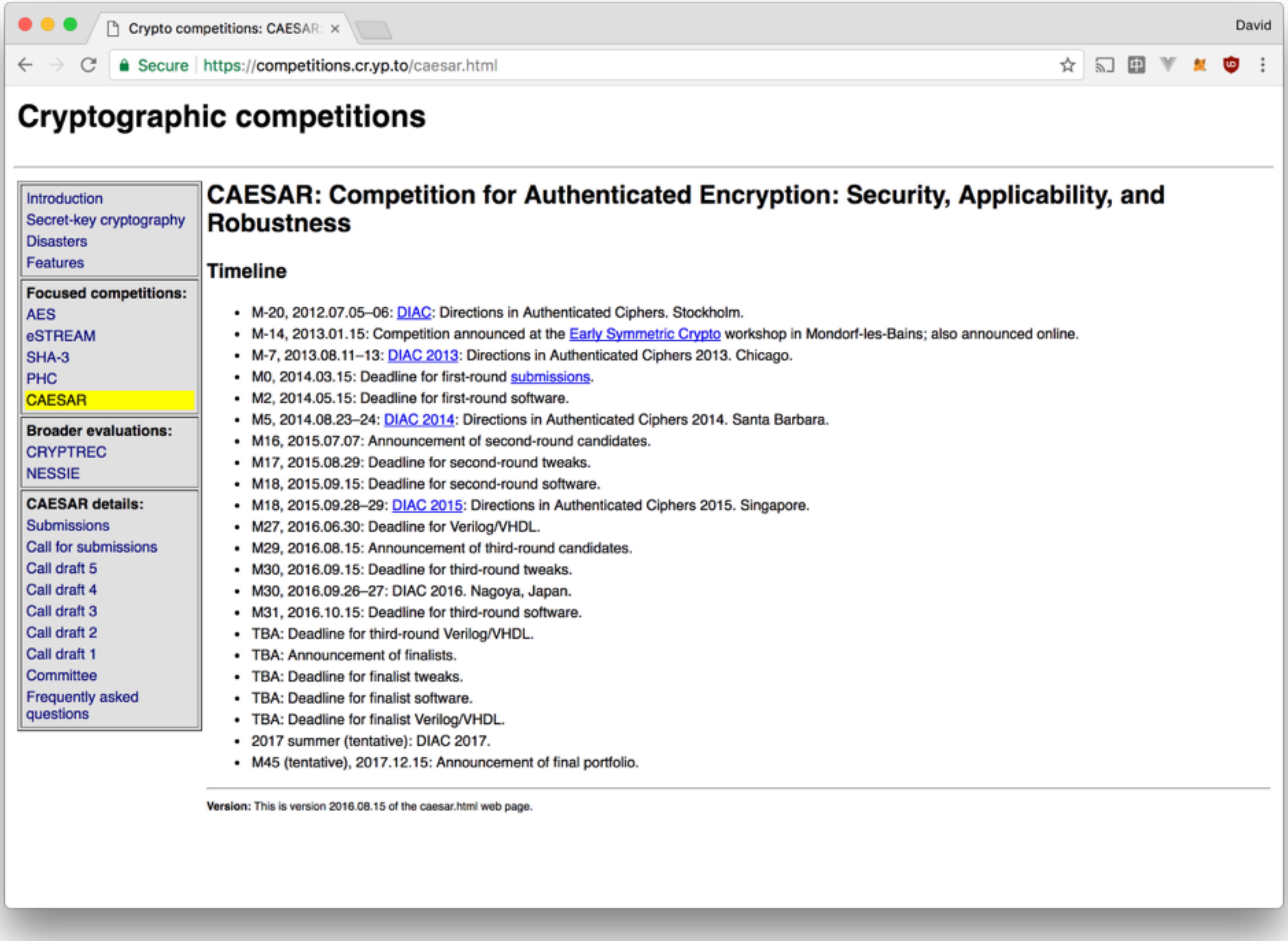
SHA-3 competition

2012

SHA-3 / SHAKE → 2015

TupleHash / ParallelHash / KMAC → 2016

Keyak and **Ketje**



Crypto competitions: CAESAR ×
David
Secure https://competitions.cr.yp.to/caesar.html

Cryptographic competitions

Introduction
Secret-key cryptography
Disasters
Features

Focused competitions:
AES
eSTREAM
SHA-3
PHC
CAESAR

Broader evaluations:
CRYPTREC
NESSIE

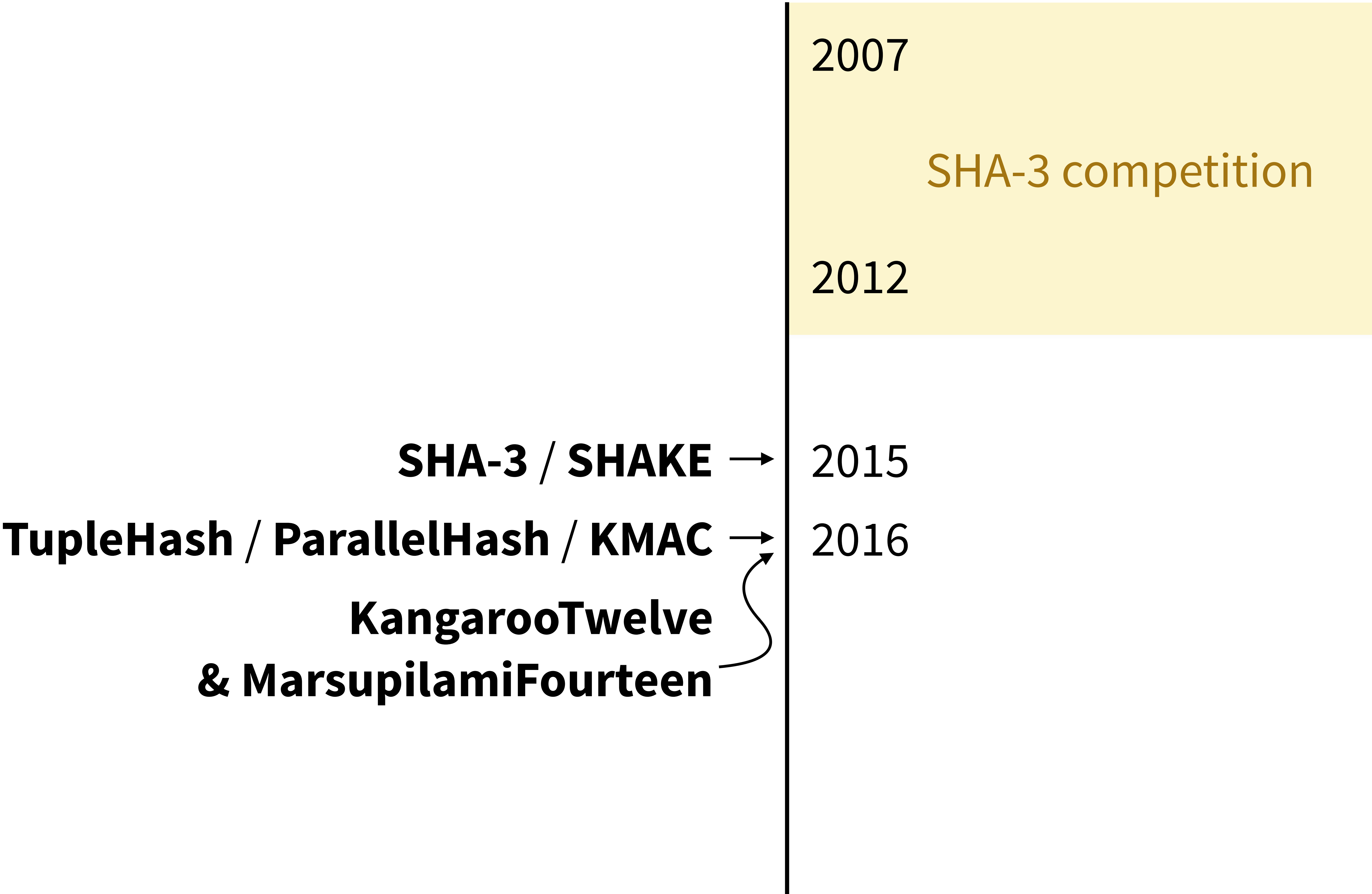
CAESAR details:
Submissions
Call for submissions
Call draft 5
Call draft 4
Call draft 3
Call draft 2
Call draft 1
Committee
Frequently asked questions

CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness

Timeline

- M-20, 2012.07.05–06: DIAC: Directions in Authenticated Ciphers. Stockholm.
- M-14, 2013.01.15: Competition announced at the Early Symmetric Crypto workshop in Mondorf-les-Bains; also announced online.
- M-7, 2013.08.11–13: DIAC 2013: Directions in Authenticated Ciphers 2013. Chicago.
- M0, 2014.03.15: Deadline for first-round submissions.
- M2, 2014.05.15: Deadline for first-round software.
- M5, 2014.08.23–24: DIAC 2014: Directions in Authenticated Ciphers 2014. Santa Barbara.
- M16, 2015.07.07: Announcement of second-round candidates.
- M17, 2015.08.29: Deadline for second-round tweaks.
- M18, 2015.09.15: Deadline for second-round software.
- M18, 2015.09.28–29: DIAC 2015: Directions in Authenticated Ciphers 2015. Singapore.
- M27, 2016.06.30: Deadline for Verilog/VHDL.
- M29, 2016.08.15: Announcement of third-round candidates.
- M30, 2016.09.15: Deadline for third-round tweaks.
- M30, 2016.09.26–27: DIAC 2016. Nagoya, Japan.
- 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.
- 2017 summer (tentative): DIAC 2017.
- M45 (tentative), 2017.12.15: Announcement of final portfolio.

Version: This is version 2016.08.15 of the caesar.html web page.



2007

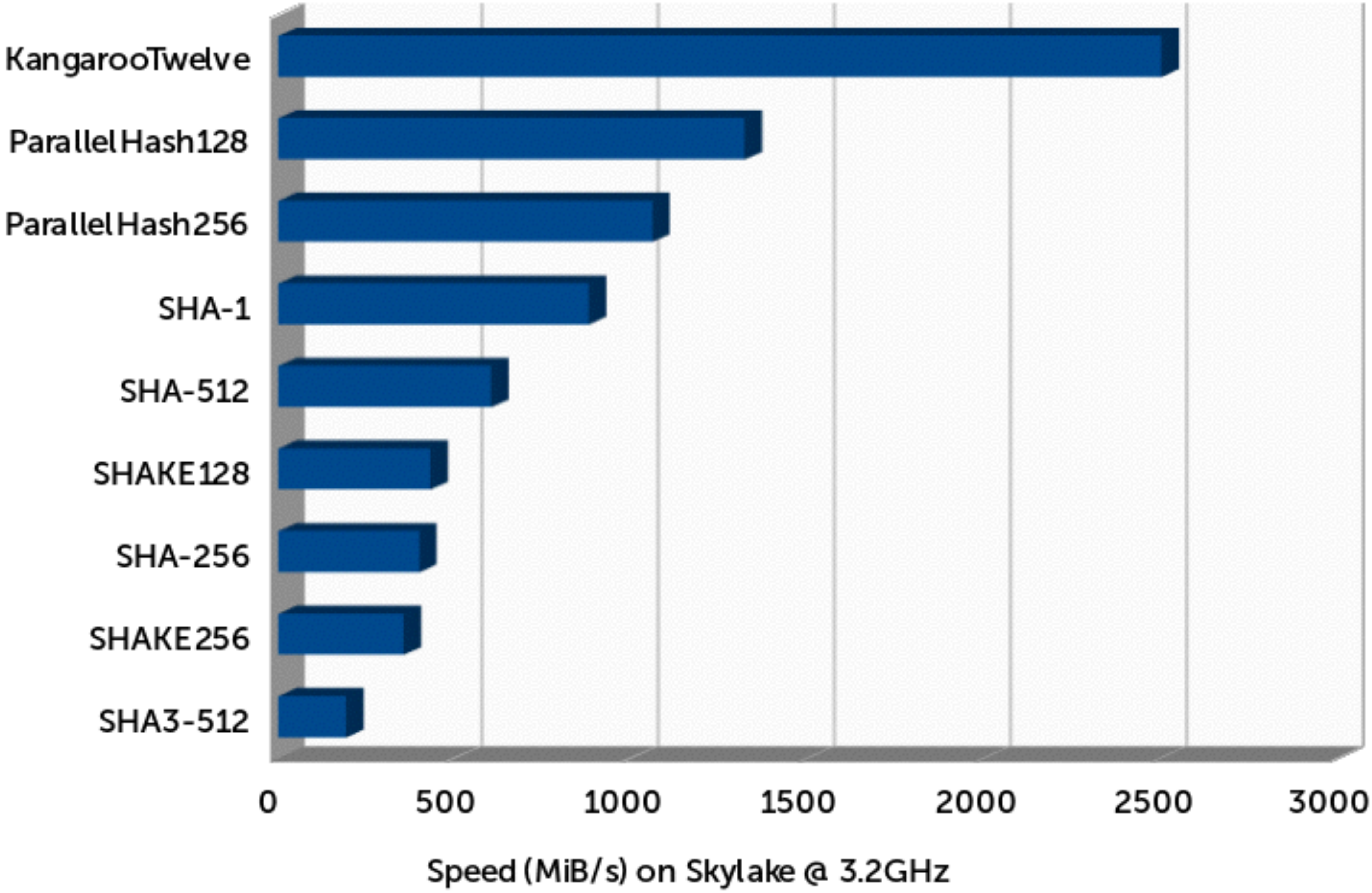
SHA-3 competition

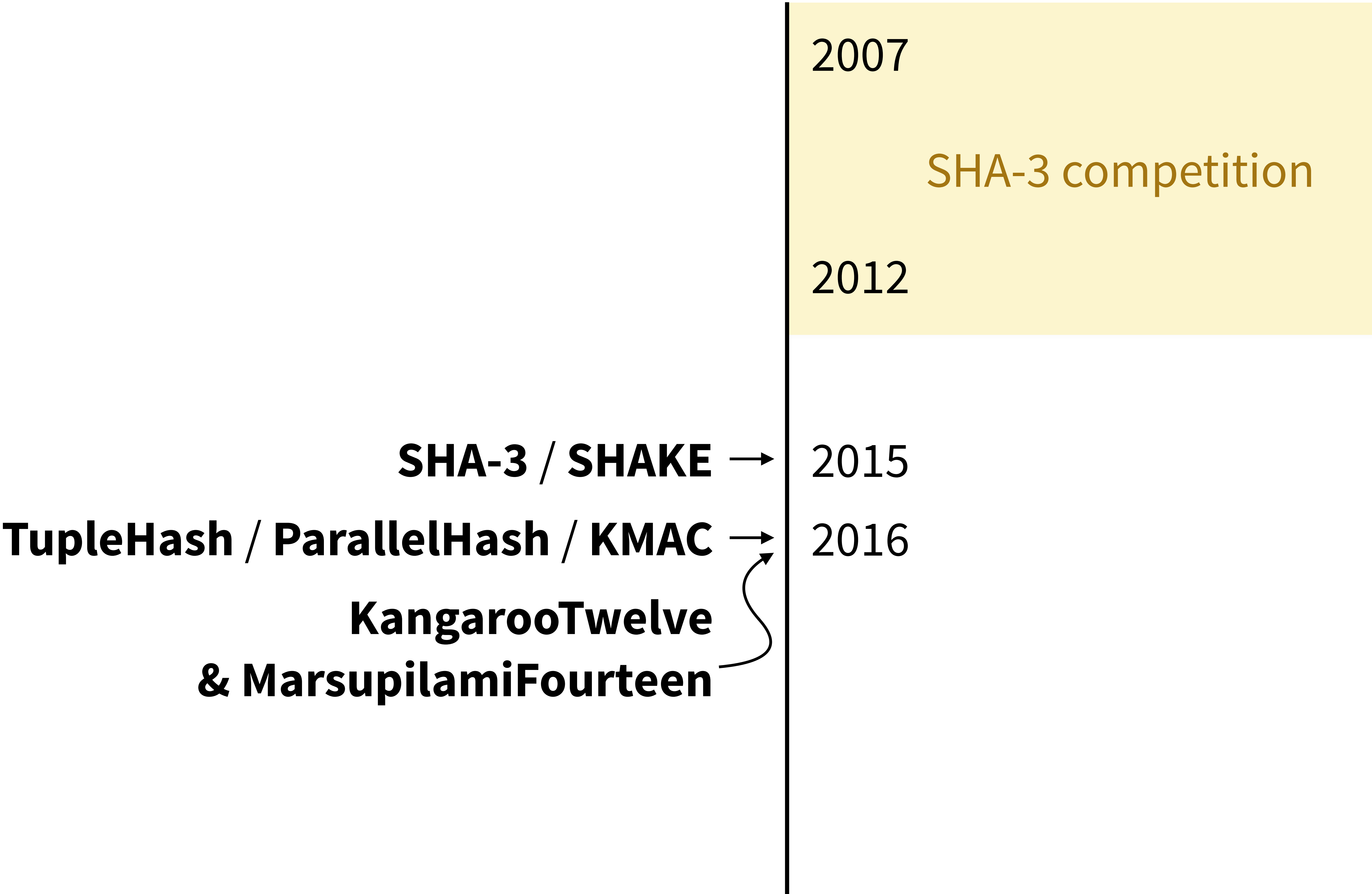
2012

SHA-3 / SHAKE → 2015

TupleHash / ParallelHash / KMAC → 2016

KangarooTwelve
& MarsupilamiFourteen


KangarooTwelve
ParallelHash128
ParallelHash256
SHA-1
SHA-512
SHAKE128
SHA-256
SHAKE256
SHA3-512
0
500
1000
1500
2000
2500
3000
Speed (MiB/s) on Skylake @ 3.2GHz



2007

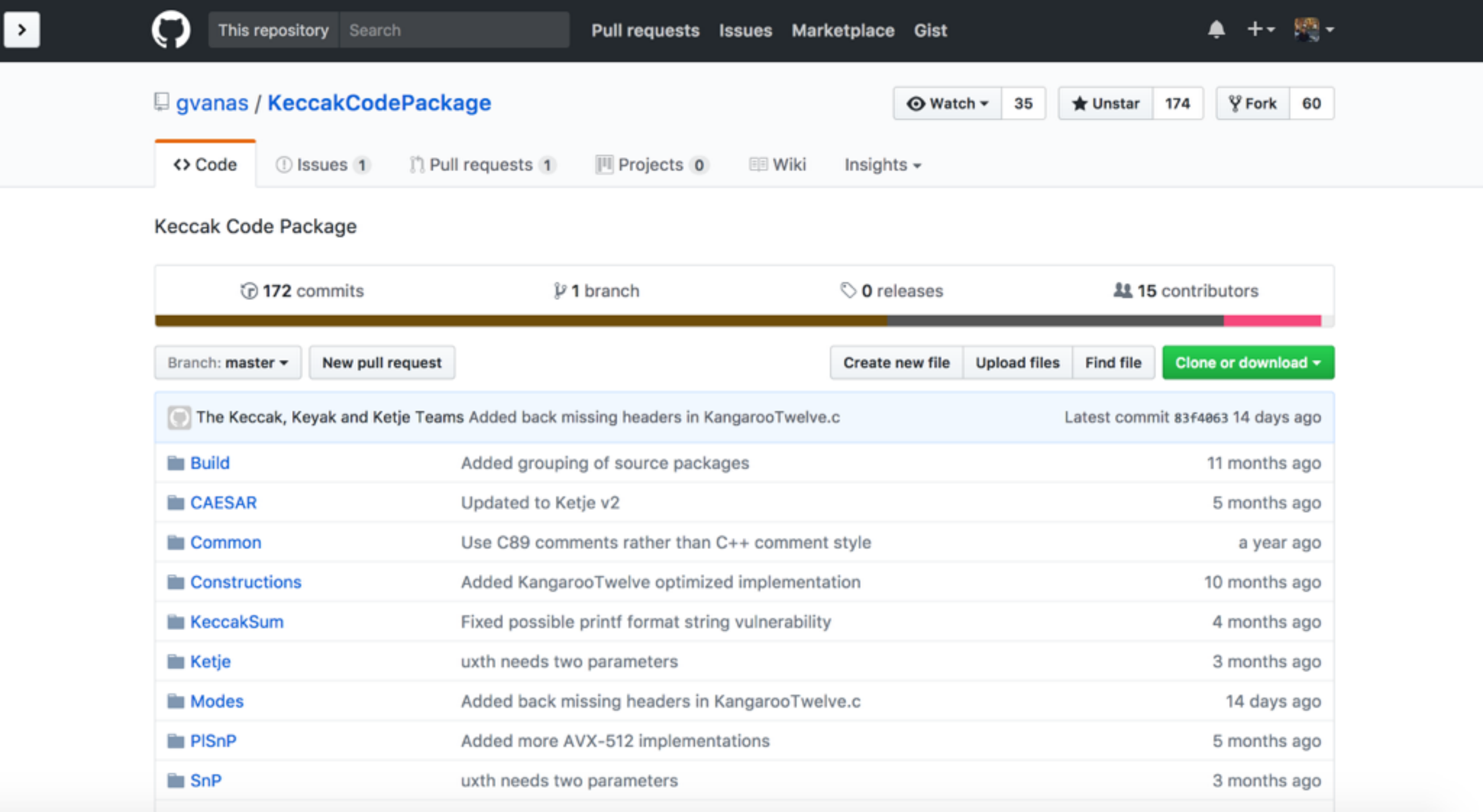
SHA-3 competition

2012

SHA-3 / SHAKE → 2015

TupleHash / ParallelHash / KMAC → 2016

KangarooTwelve
& MarsupilamiFourteen



gvanas / KeccakCodePackage

Keccak Code Package

172 commits · 1 branch · 0 releases · 15 contributors

The Keccak, Keyak and Ketje Teams Added back missing headers in KangarooTwelve.c — Latest commit 83f4063 14 days ago

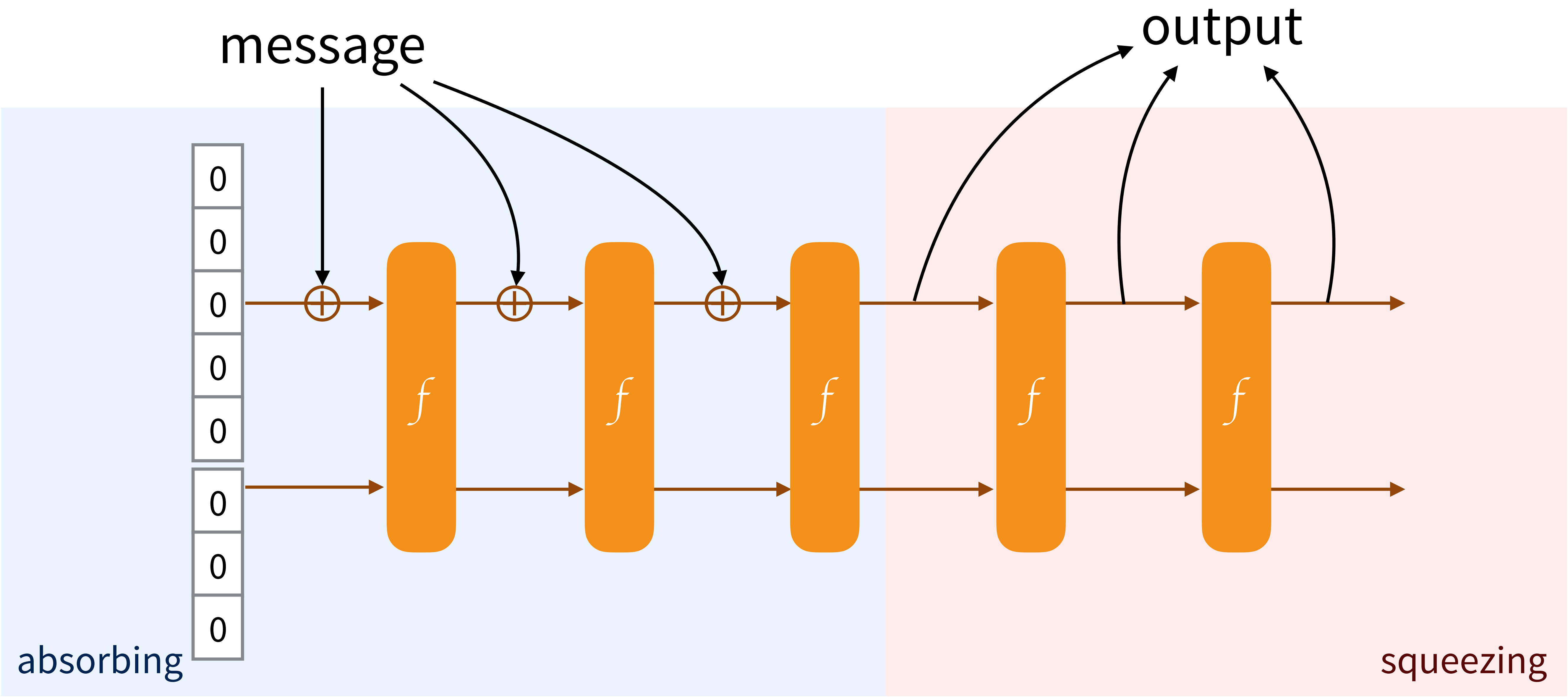
Build	Added grouping of source packages	11 months ago
CAESAR	Updated to Ketje v2	5 months ago
Common	Use C89 comments rather than C++ comment style	a year ago
Constructions	Added KangarooTwelve optimized implementation	10 months ago
KeccakSum	Fixed possible printf format string vulnerability	4 months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

github.com/gvanas/KeccakCodePackage

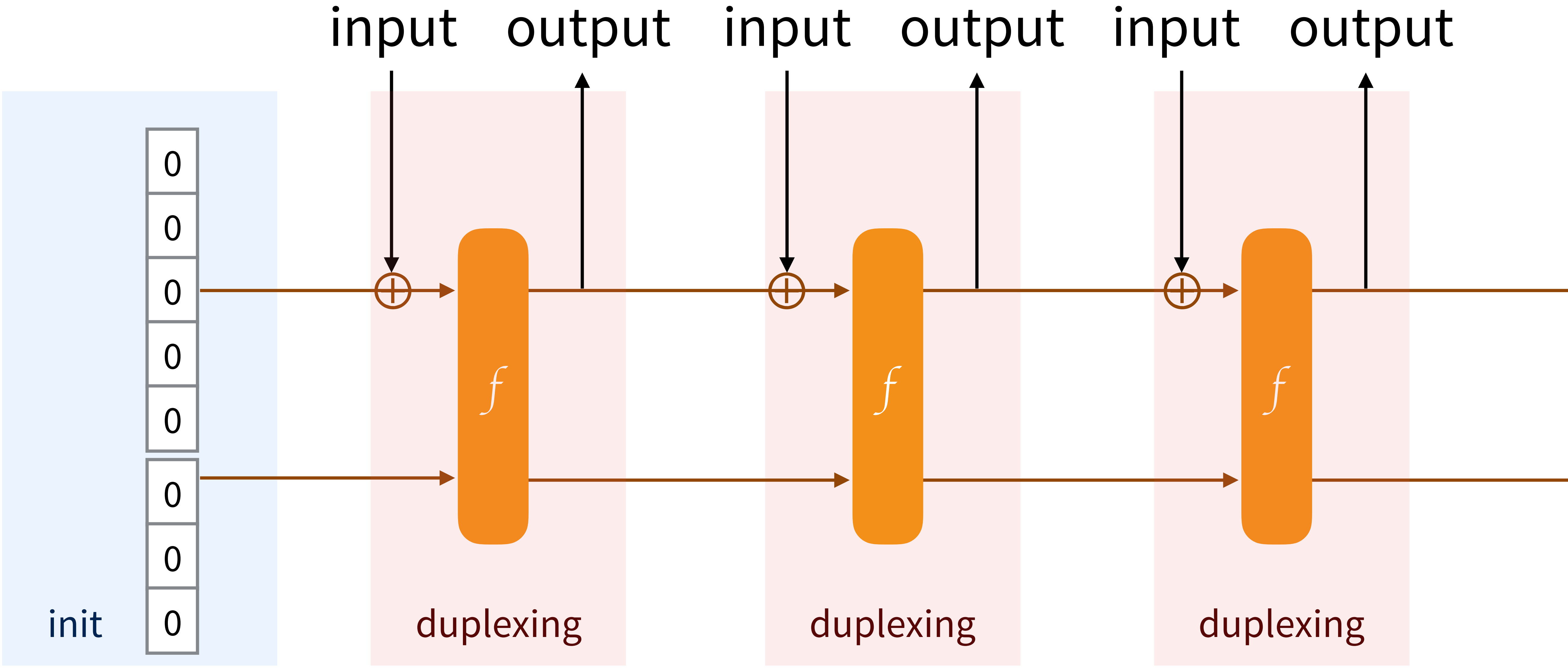
Outline

1. SHA-3
2. derived functions
3. **derived protocols**

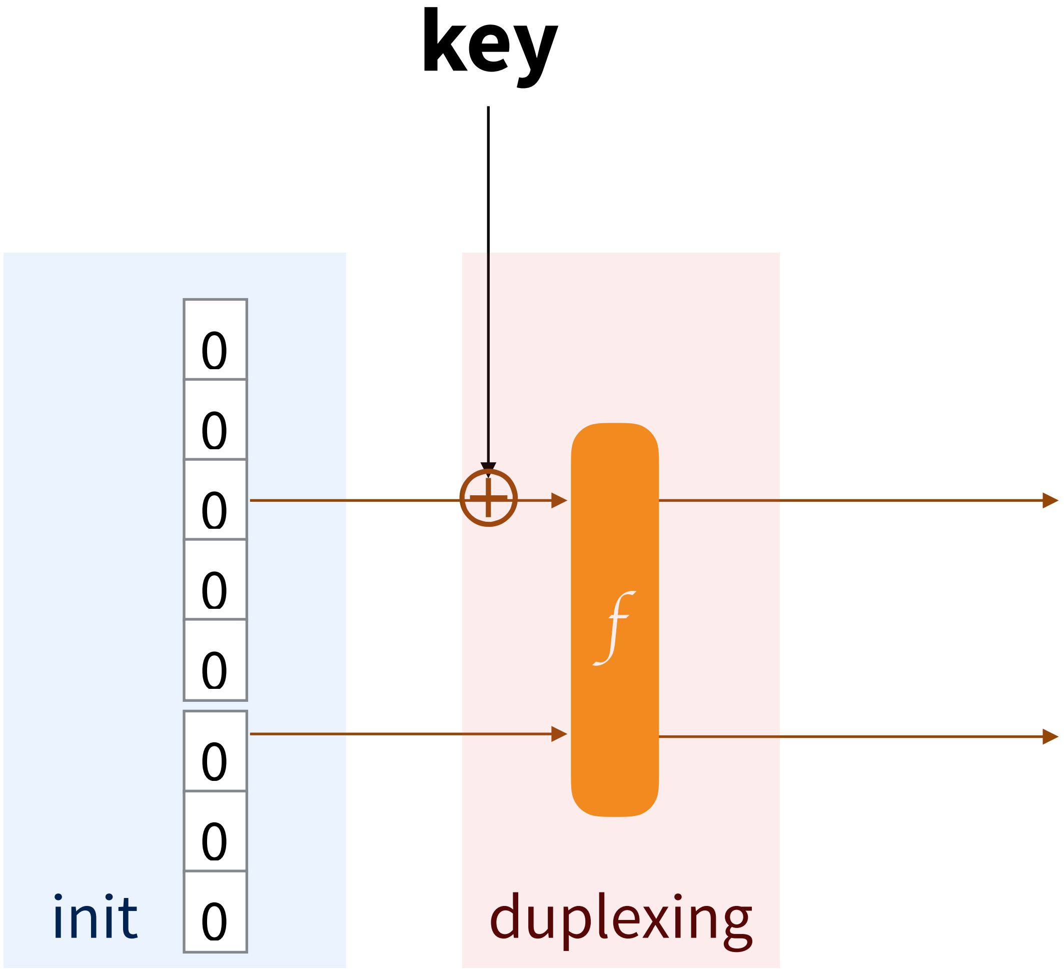
Sponge Construction



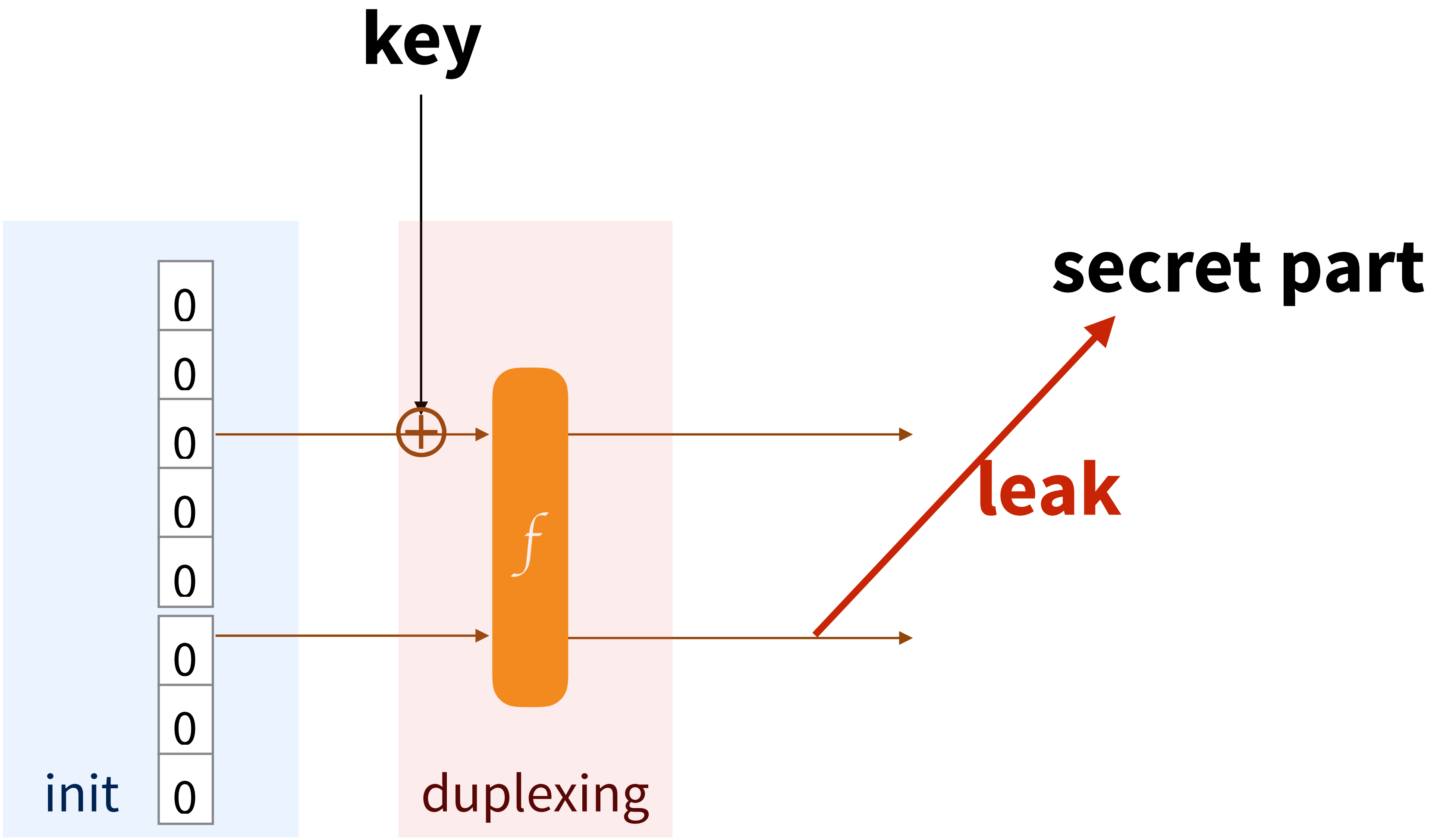
Duplex Construction



Keyed-mode



Keyed-mode


key

0

0

0

0

0

0

0

0

init

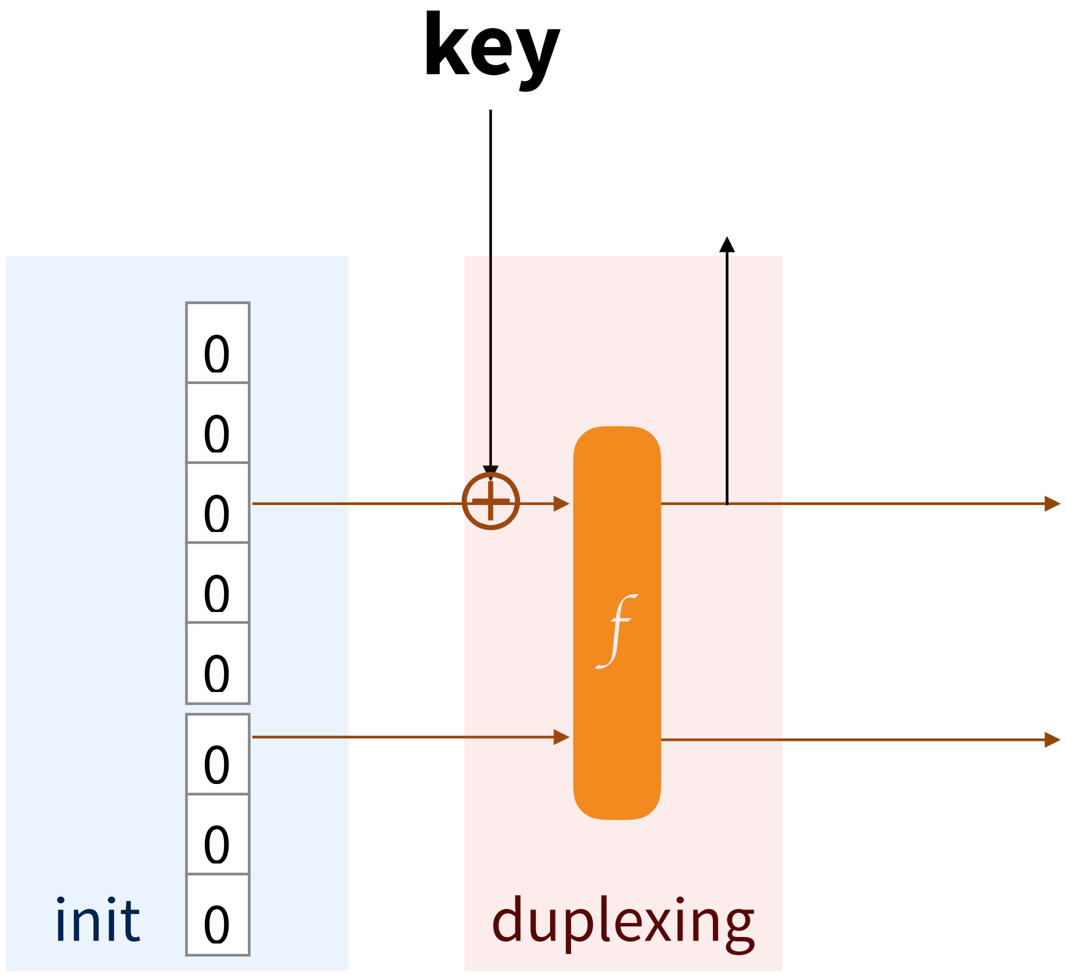
f

duplexing

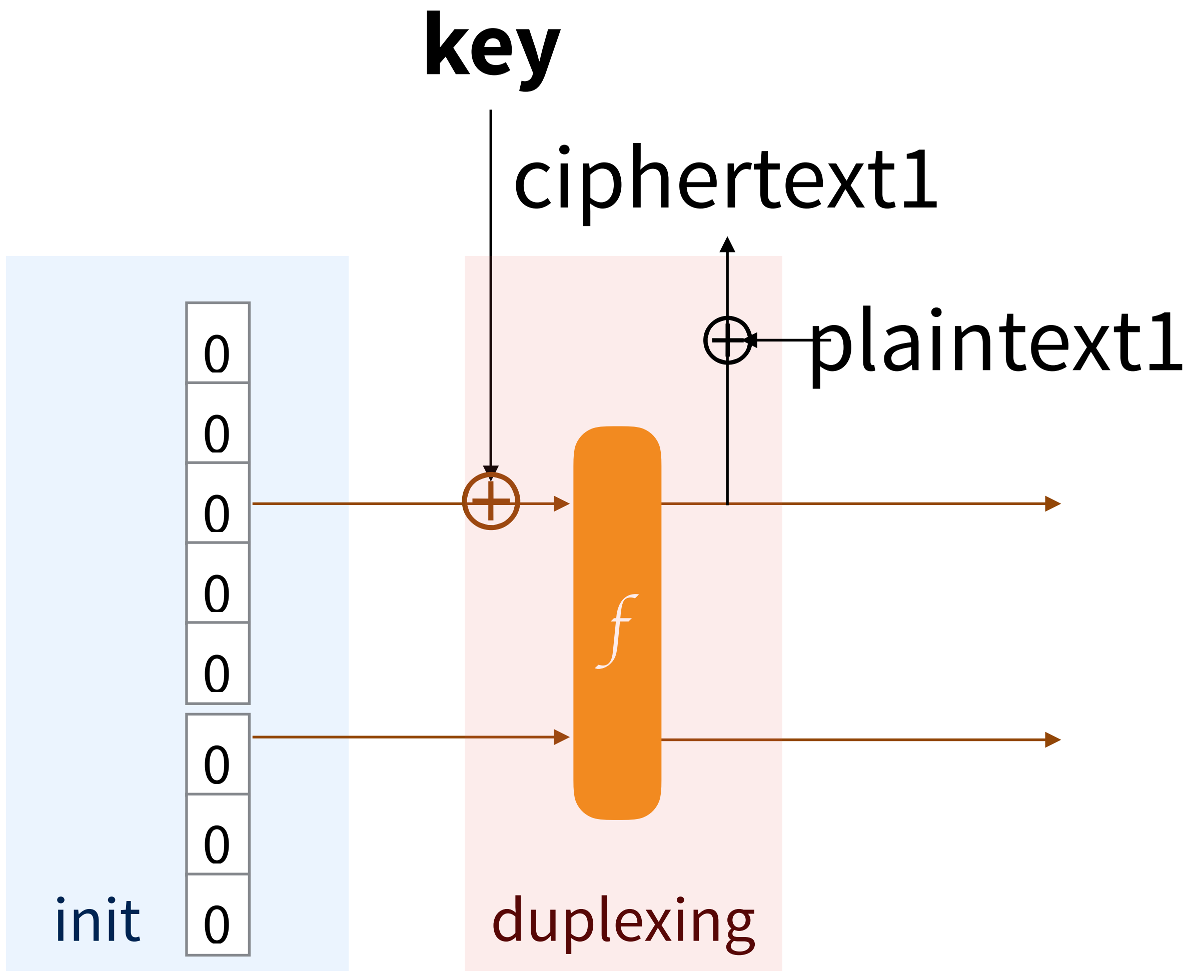
secret part

leak

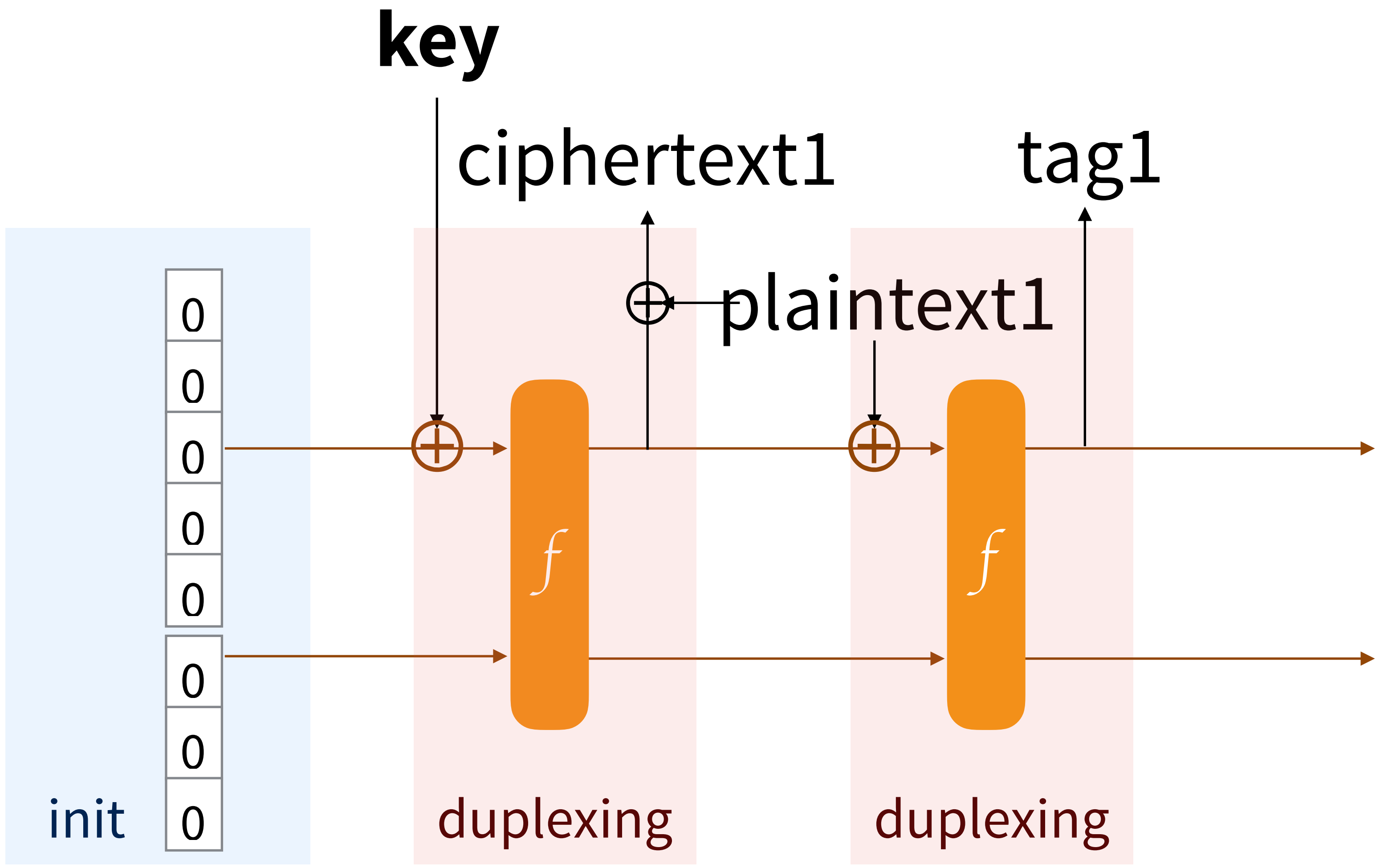
Encryption?



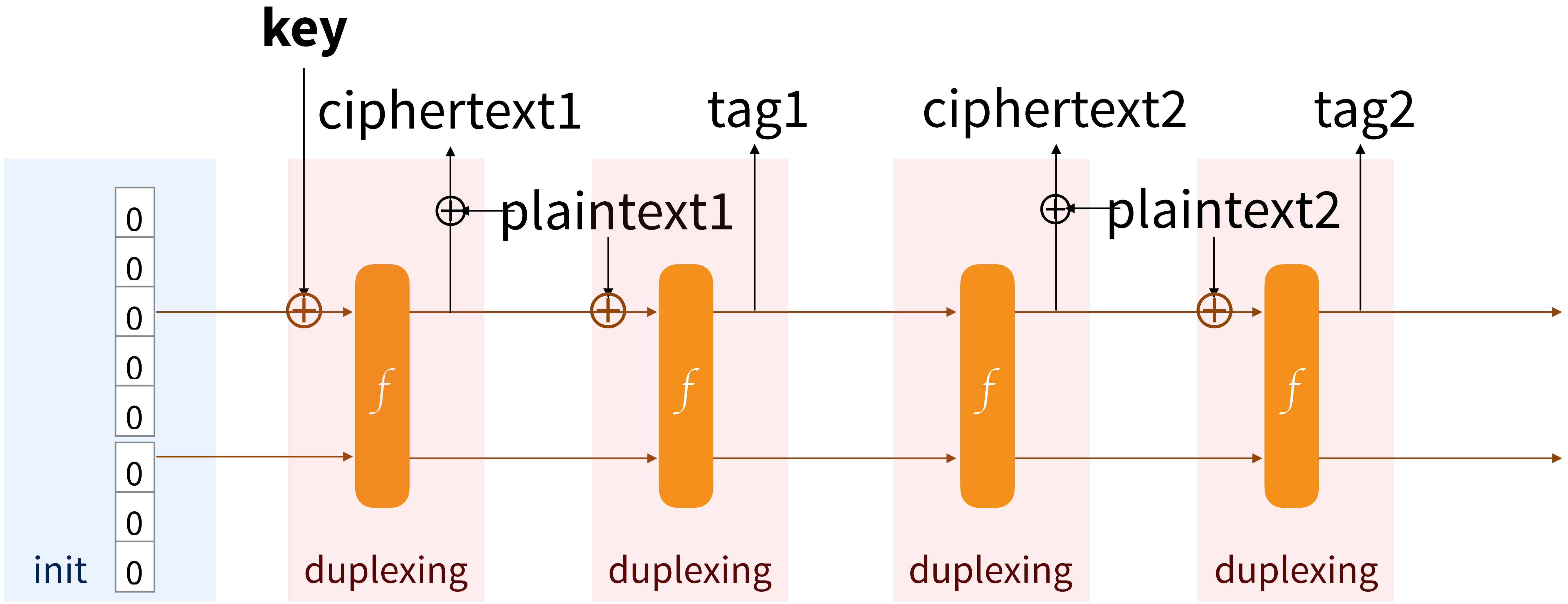
Encryption



Authenticated Encryption

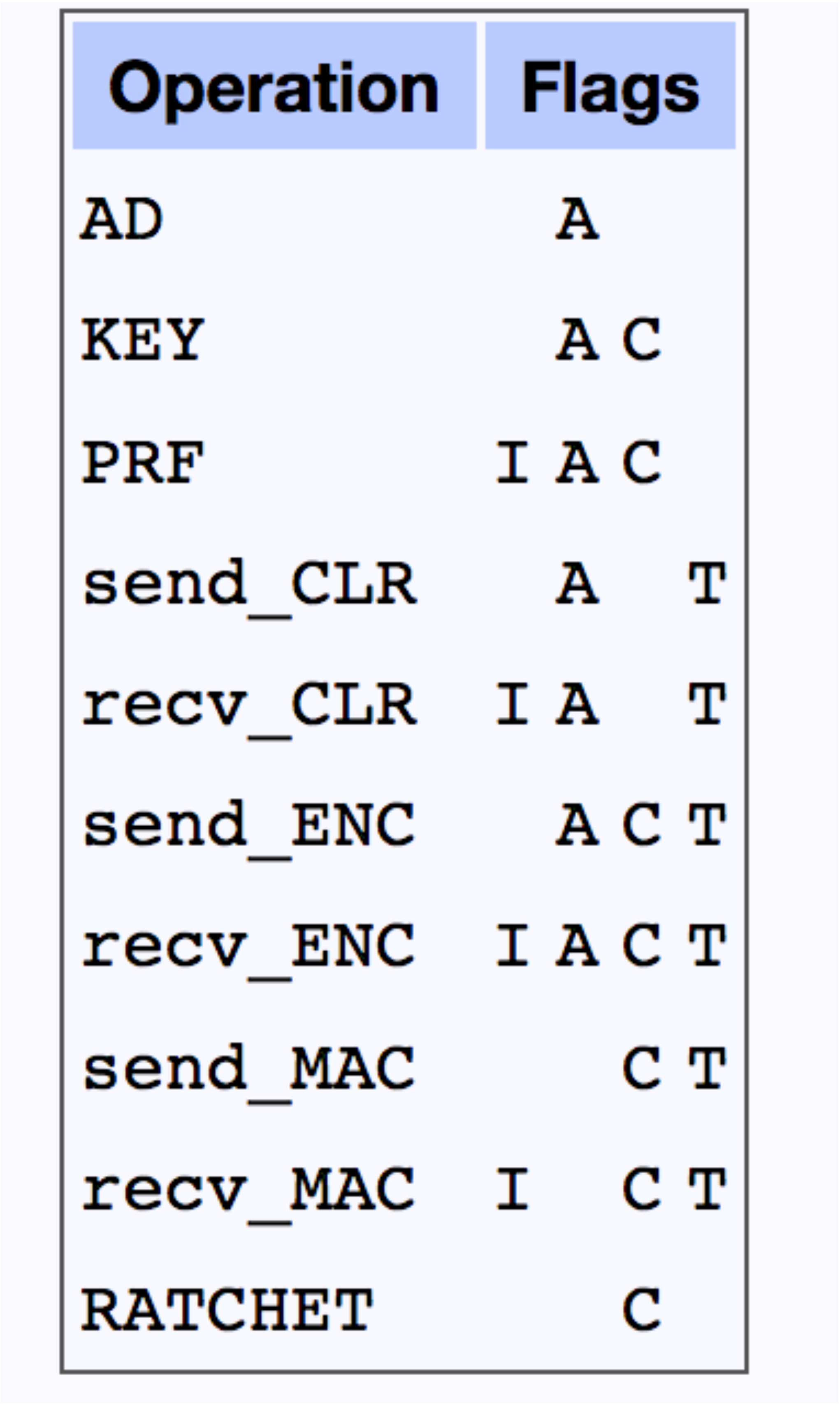


Sessions


key
ciphertext1
tag1
ciphertext2
tag2
plaintext1
plaintext2
0
0
0
0
0
0
0
0
init
f
f
f
f
duplexing
duplexing
duplexing
duplexing

Strobe

```
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
}
```



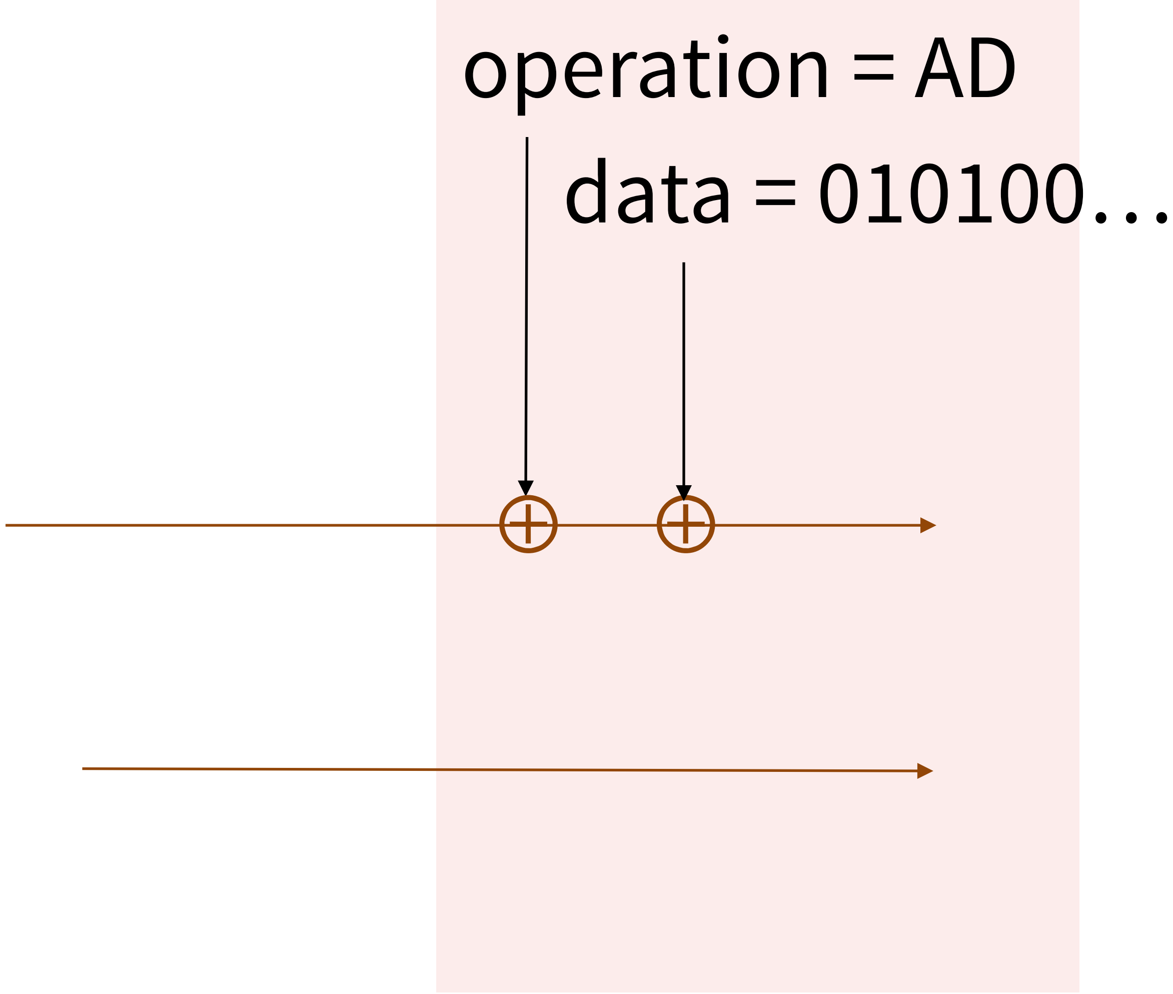
Operation	Flags
AD	A
KEY	A C
PRF	I A C
send_CLR	A T
recv_CLR	I A T
send_ENC	A C T
recv_ENC	I A C T
send_MAC	C T
recv_MAC	I C T
RATCHET	C

Hash Function

```
myHash = Strobe_init(“hash”)
myHash.AD(“something to be hashed”)
hash = myHash.PRF(outputLen=16)
```

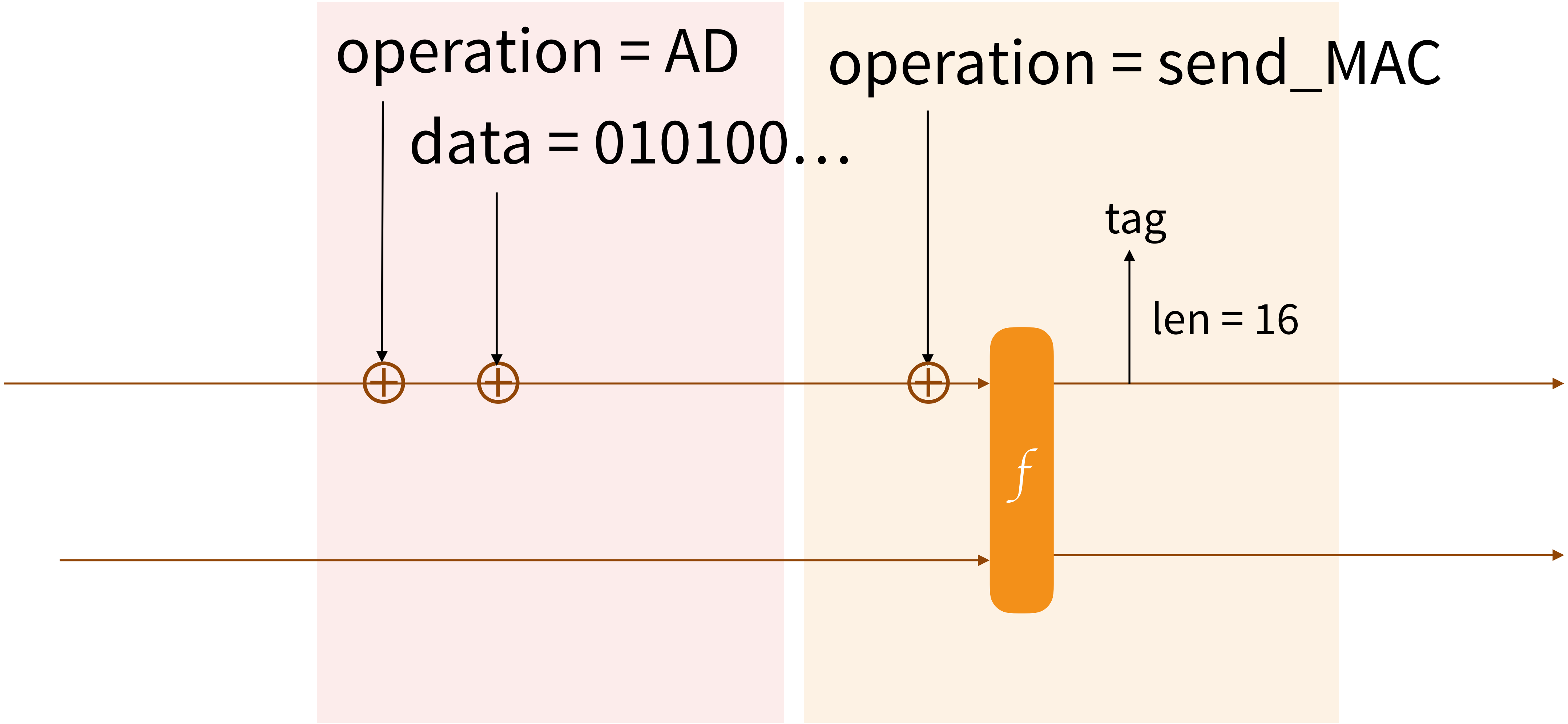
Key Derivation Function

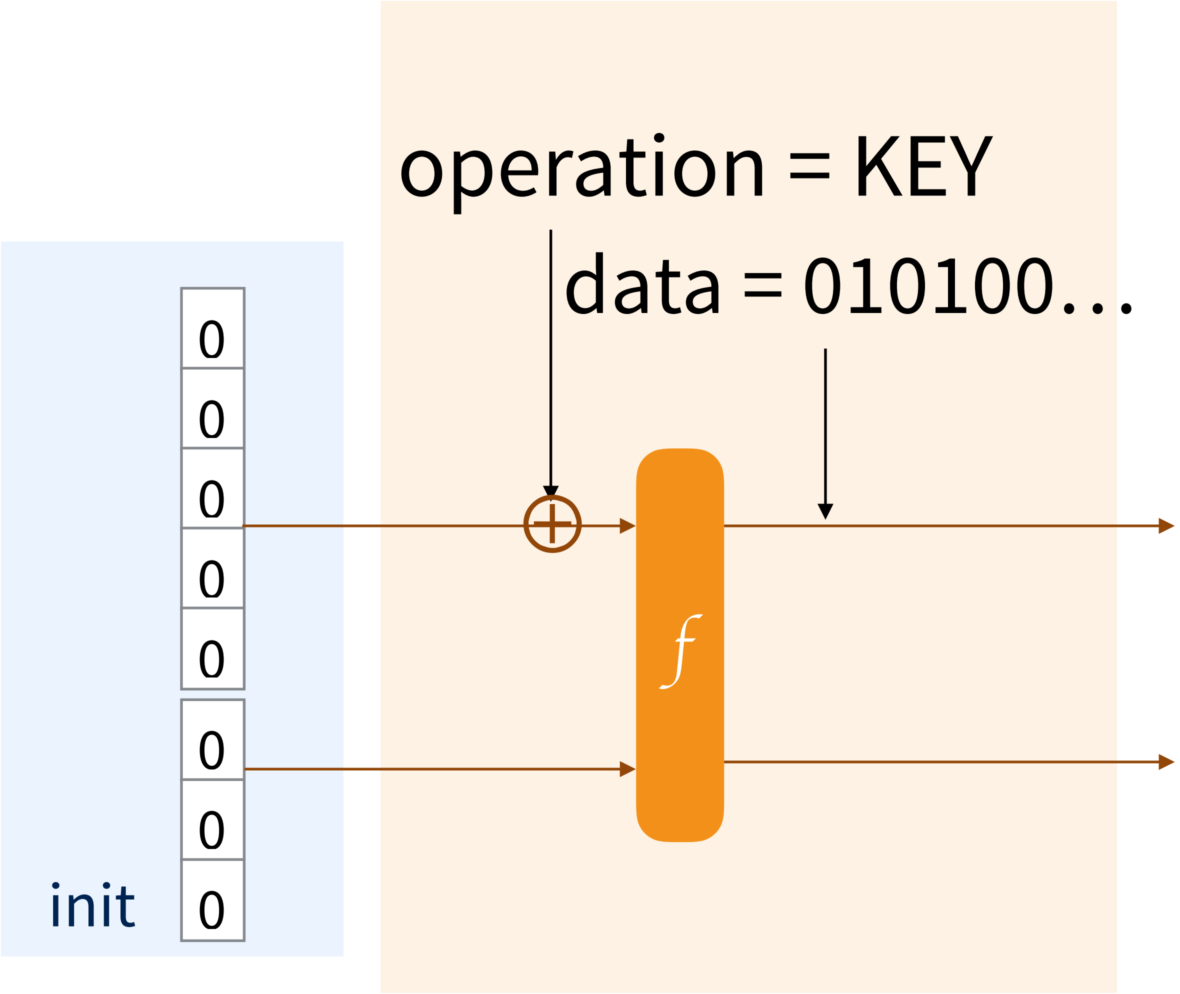
```
KDF = Strobe_init(“deriving keys”)
KDF.KEY(keyExchangeOutput)
keys = KDF.PRF(outputLen=32)
key1 = keys[:16]
key2 = keys[16:]
```



operation = AD

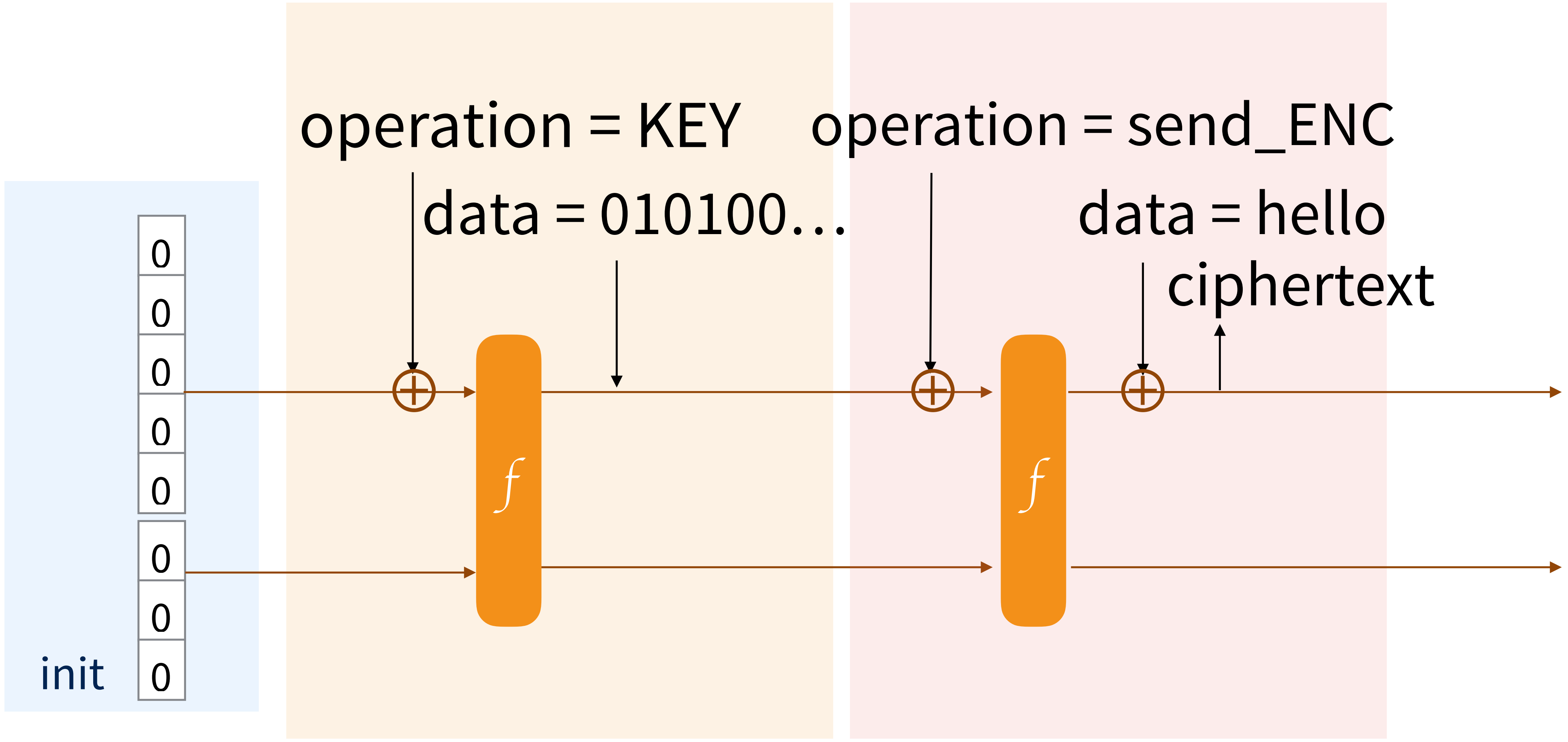
data = 010100…


operation = AD
data = 010100…
operation = send_MAC
tag
len = 16
f



operation = KEY

data = 010100…



operation = KEY

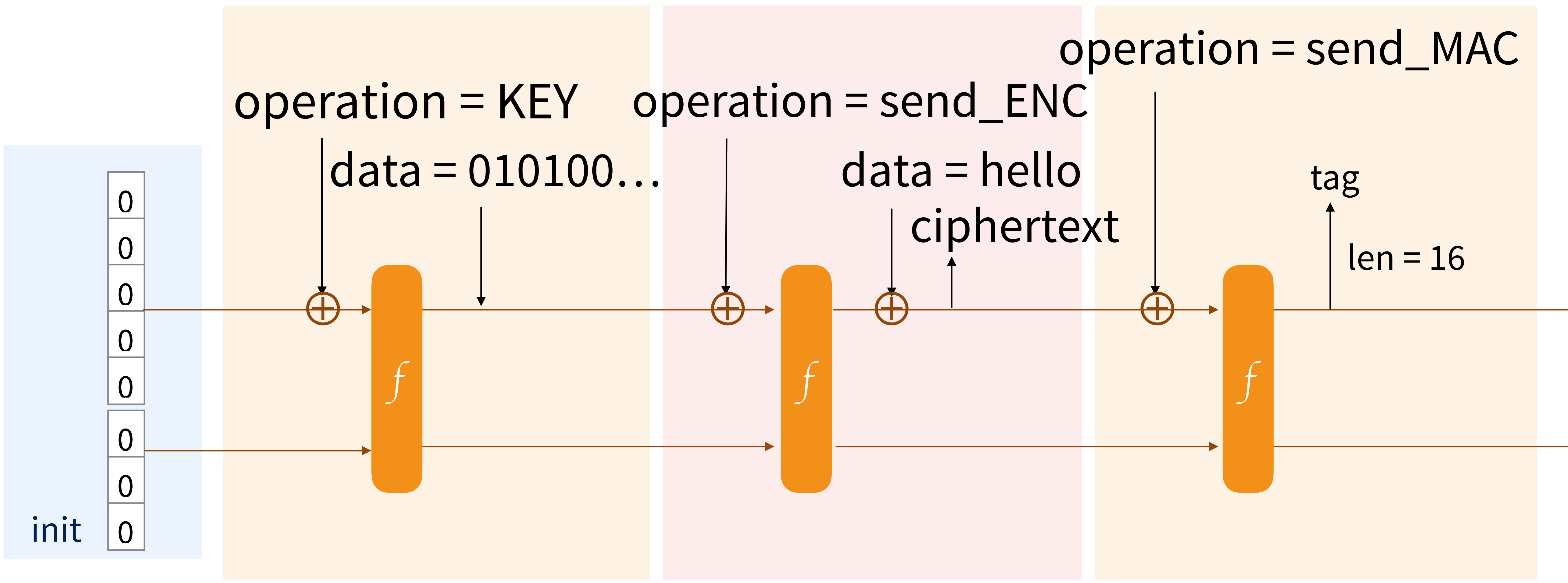
data = 010100…

operation = send_ENC

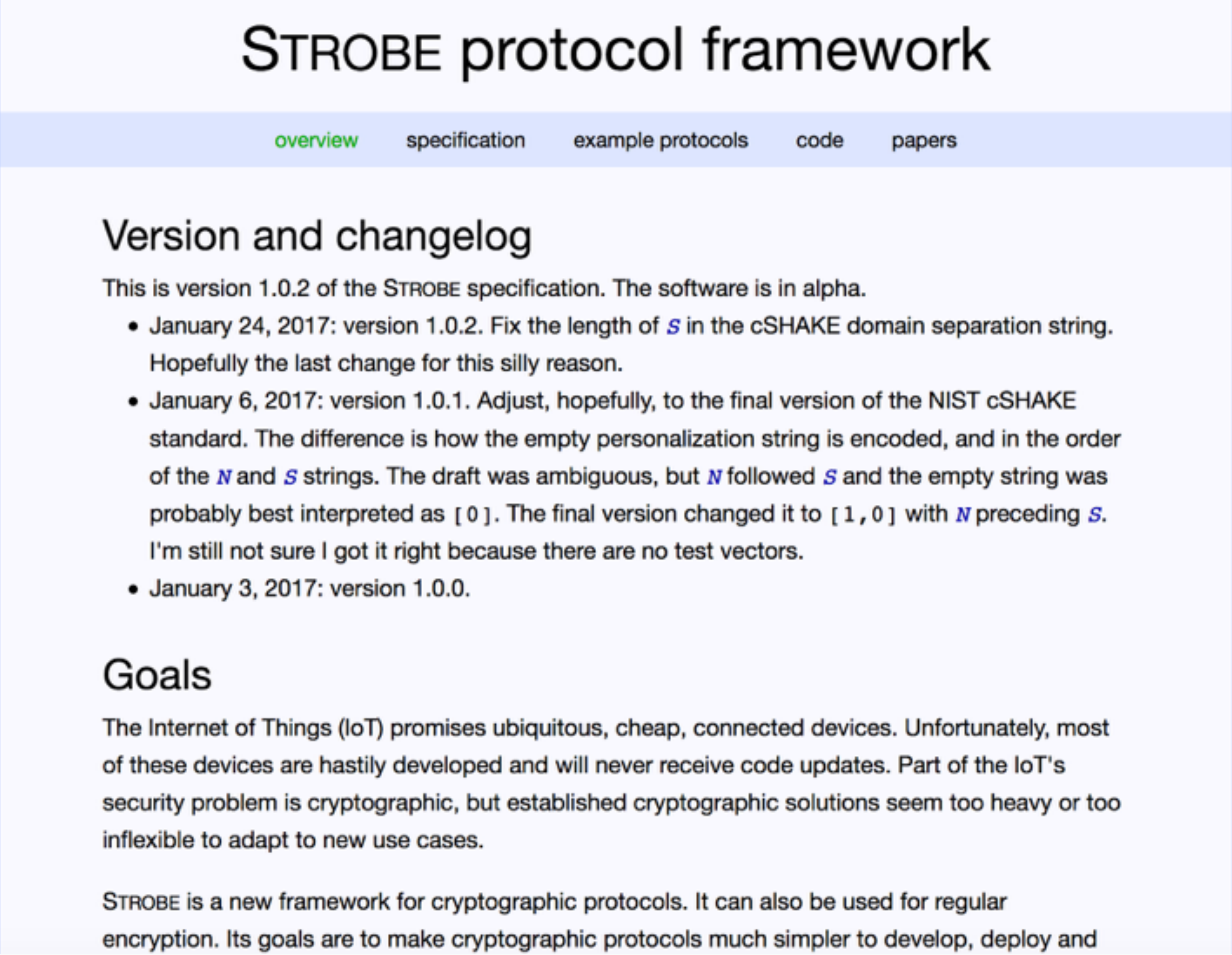
data = hello

ciphertext

init



init
0
0
0
0
0
0
0
0
operation = KEY
data = 010100…
operation = send_ENC
data = hello
ciphertext
operation = send_MAC
tag
len = 16
f
f
f



STROBE protocol framework

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

strobe.sourceforge.io

Outline

1. SHA-3
2. derived functions
3. derived protocols
4. **Disco?**

```
    the patterns                                    [0][2:], ",")
for _, pattern := range patterns {
	pattern = strings.Trim(pattern, " ")
	if pattern == "e" {
		h.e = GenerateKeypair()
		*messageBuffer = append(*messageBuffer, h.e.publicKey[:]...)
		h.strobeState.Send_CLR(false, h.e.publicKey[:])
	} else if pattern == "s" {
		*messageBuffer = append(*messageBuffer, h.strobeState.Send_AEAD(h.s.publicKey[:], []byte{})...)
	} else if pattern == "ee" {
		h.strobeSt
```

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

nccgroup