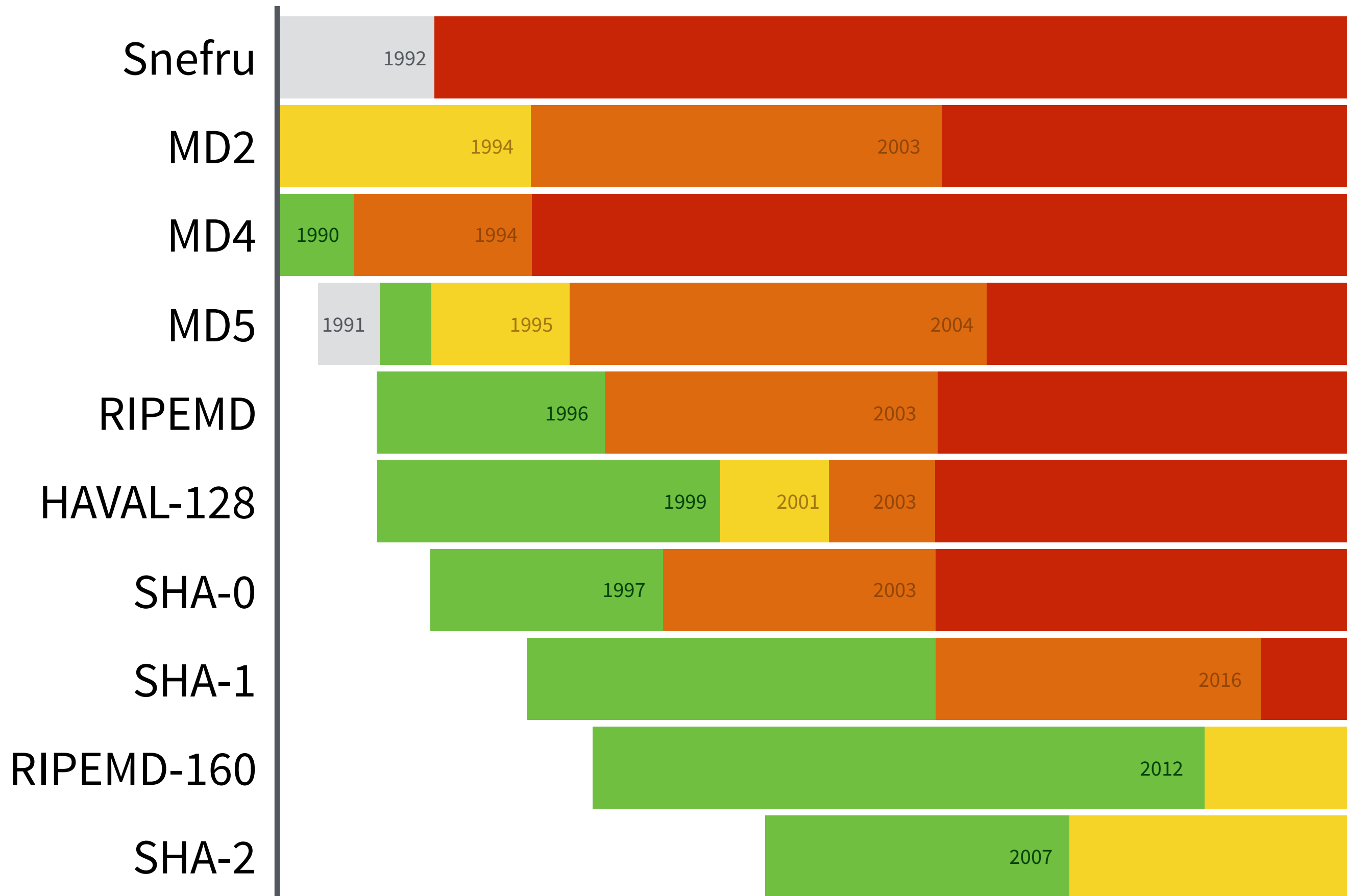


SHA-3 vs the world

David Wong





<http://valerieaurora.org/hash.html>

Computer Security Division

Computer Security Resource Center

[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.

Computer Security Division

Computer Security Resource Center

[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 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.

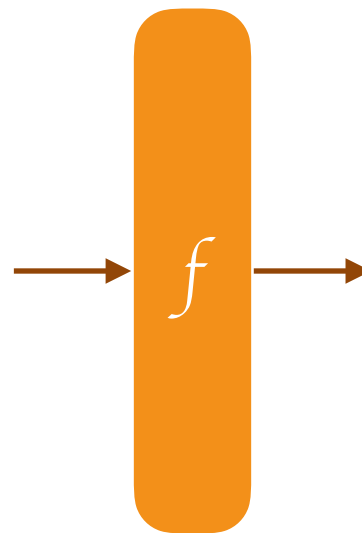
[History of Updates](#)

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

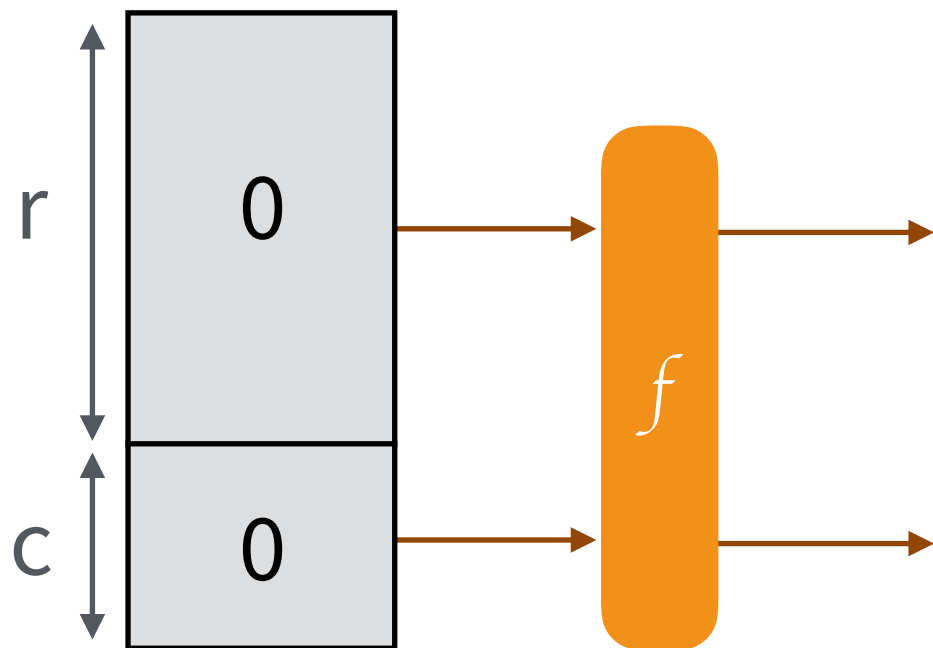
BLAKE, Grøstl, JH, Skein

Sponge Construction

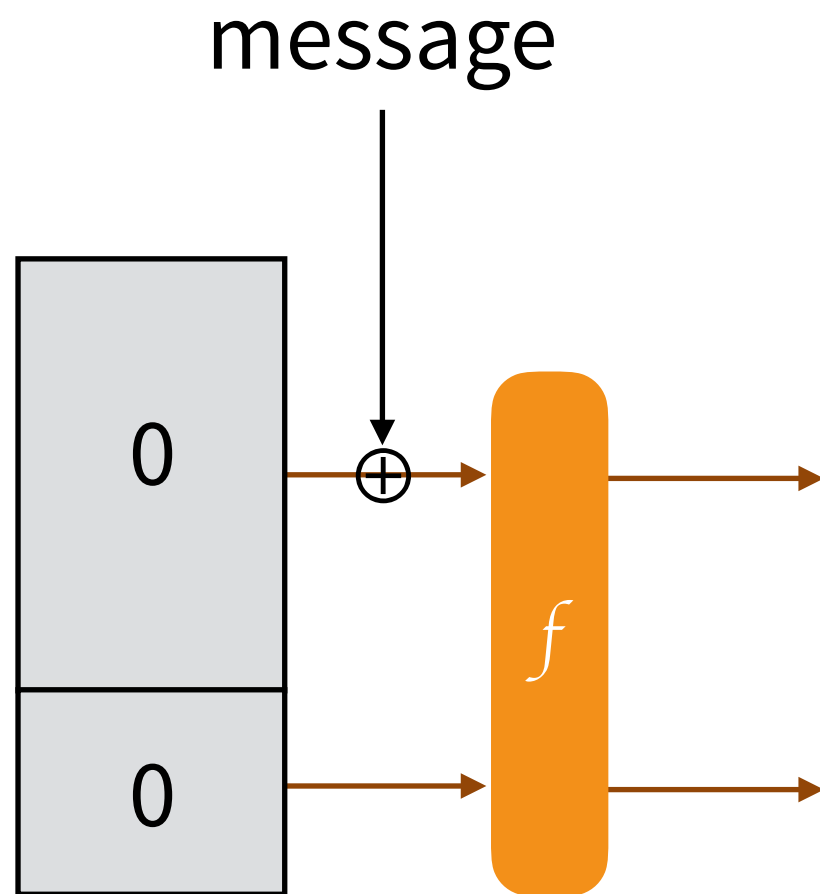


permutation-based cryptography

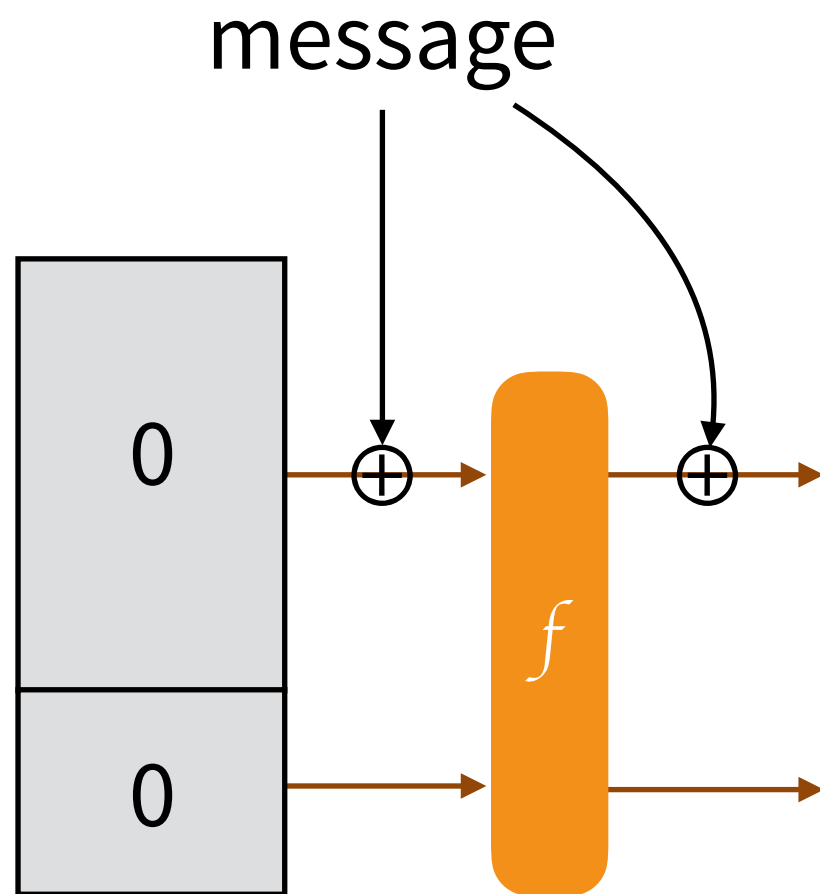
Sponge Construction



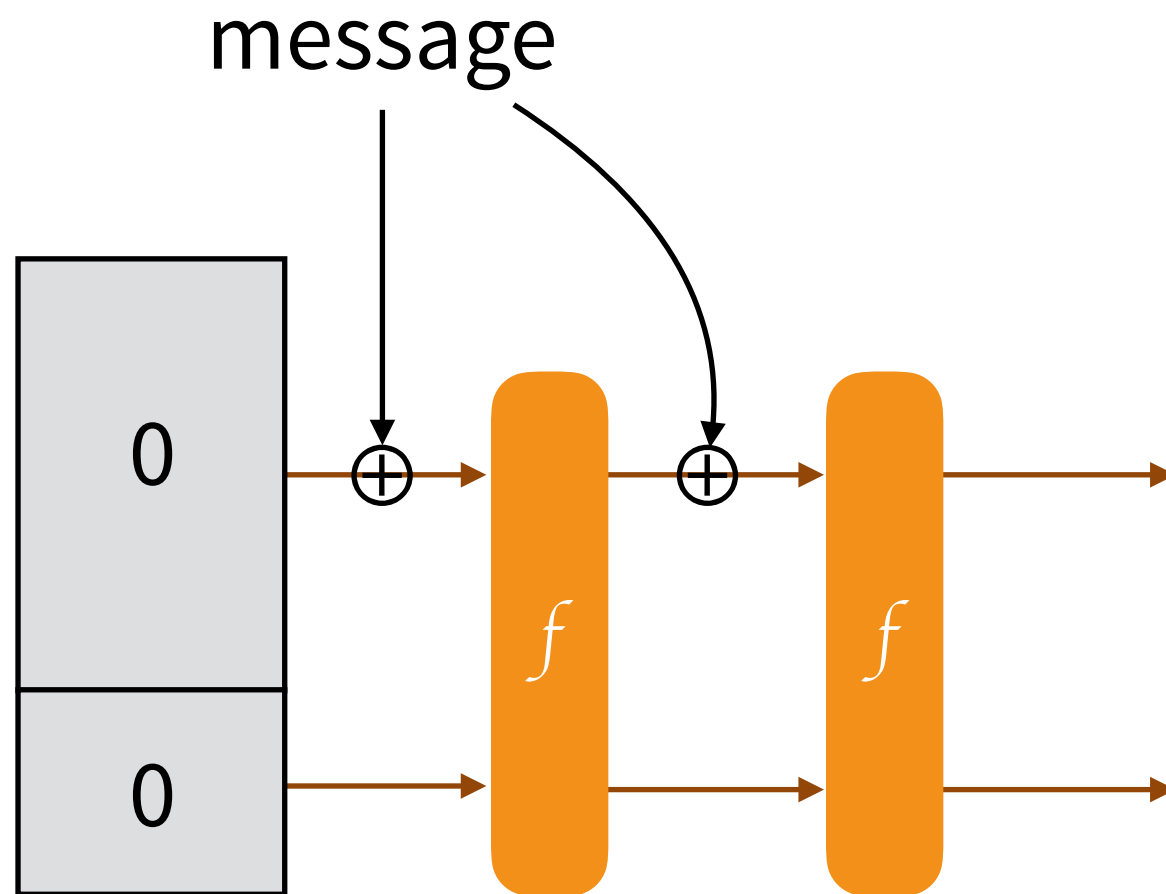
Sponge Construction



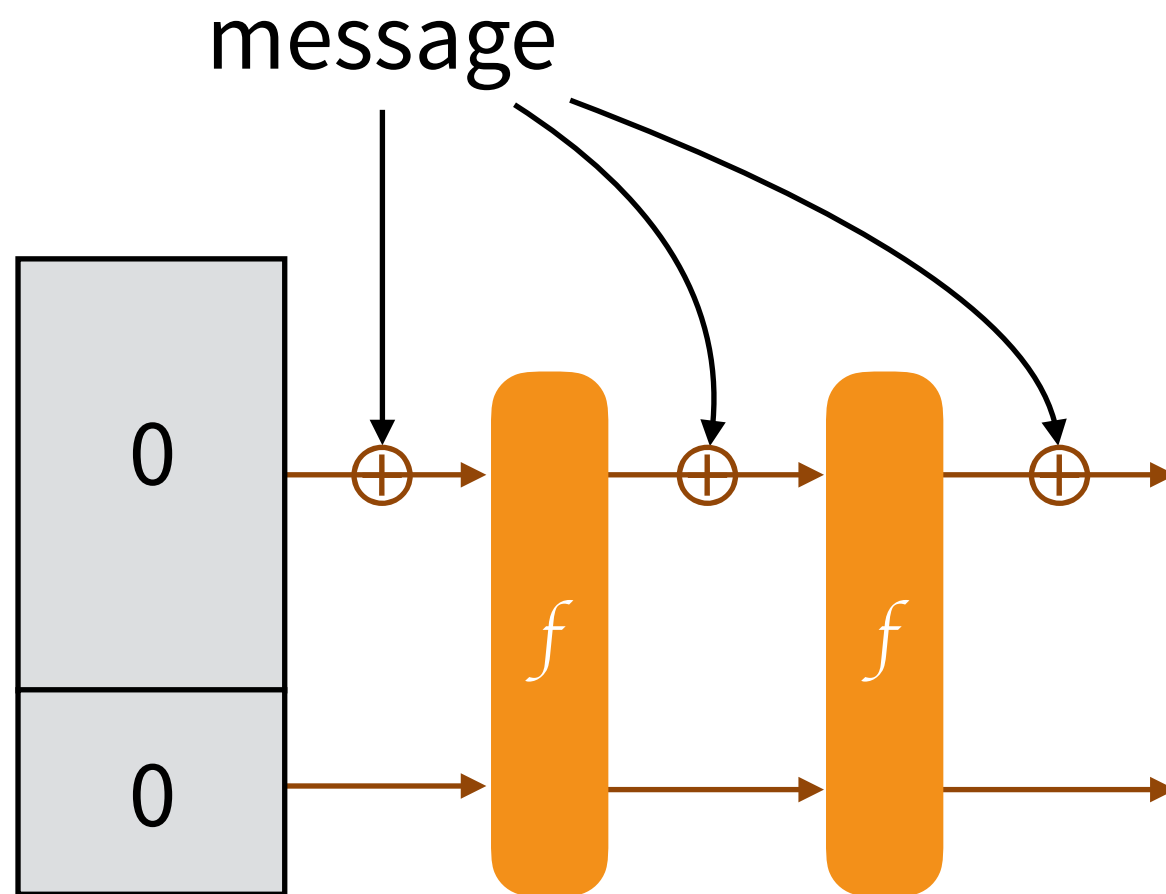
Sponge Construction



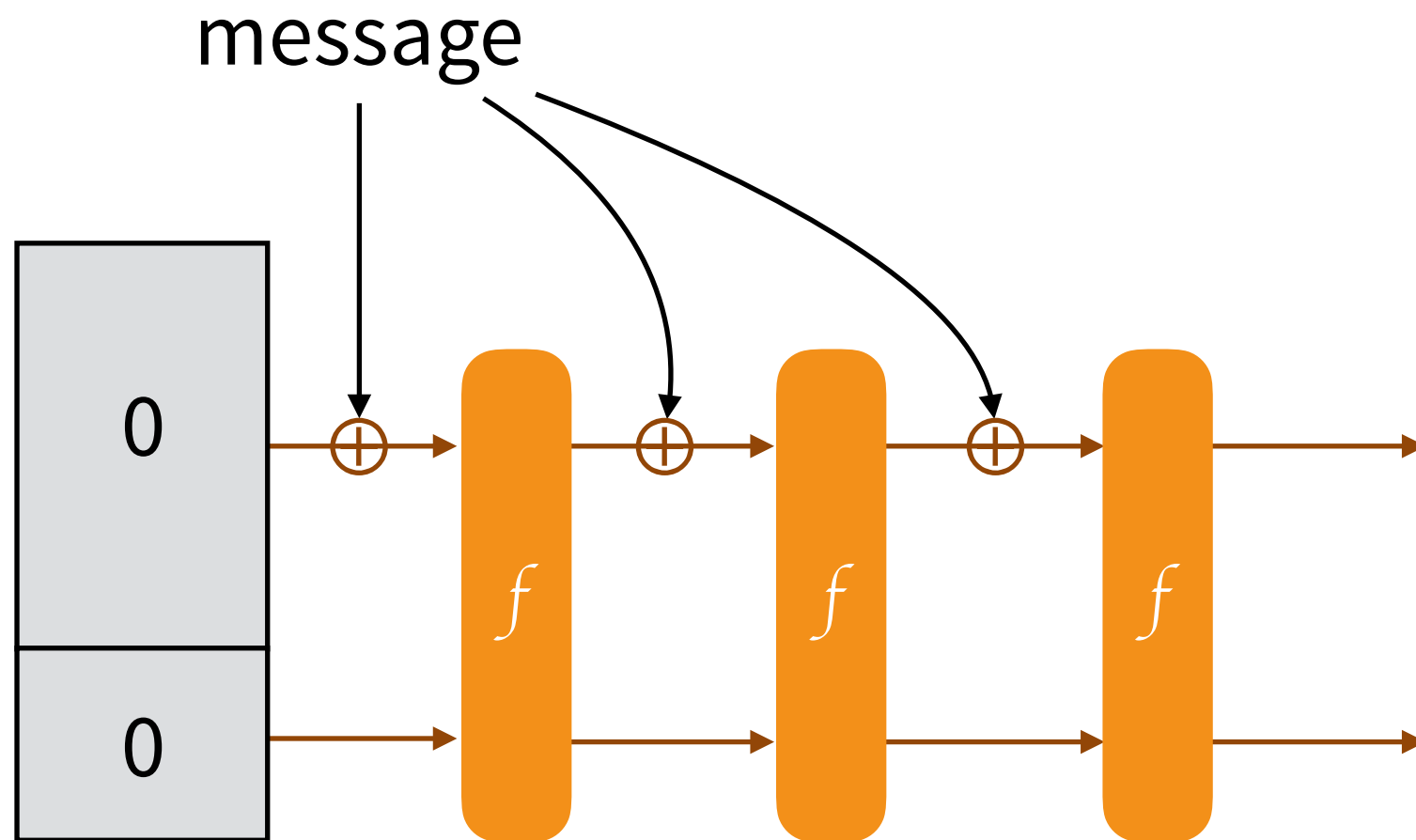
Sponge Construction



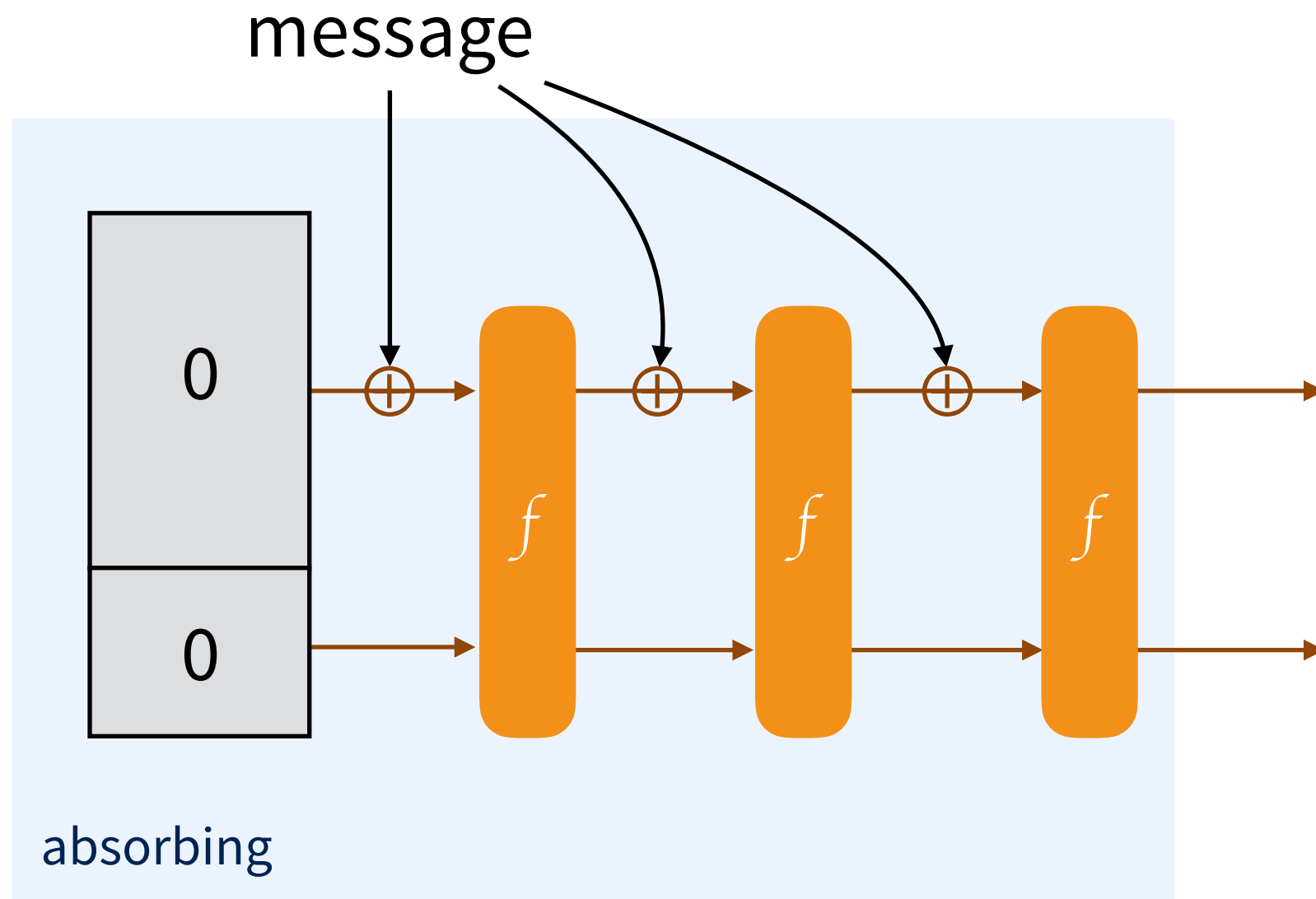
Sponge Construction



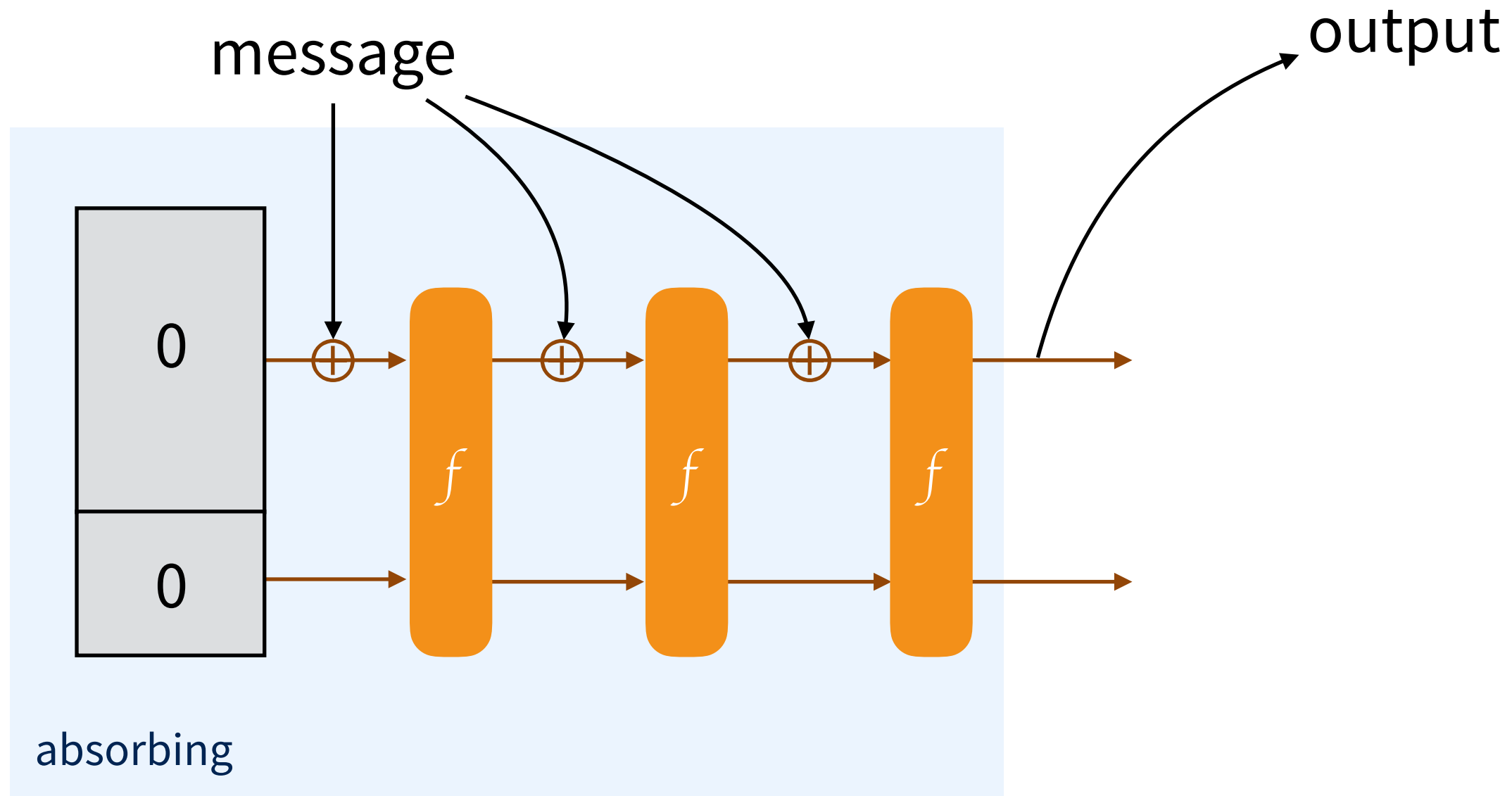
Sponge Construction



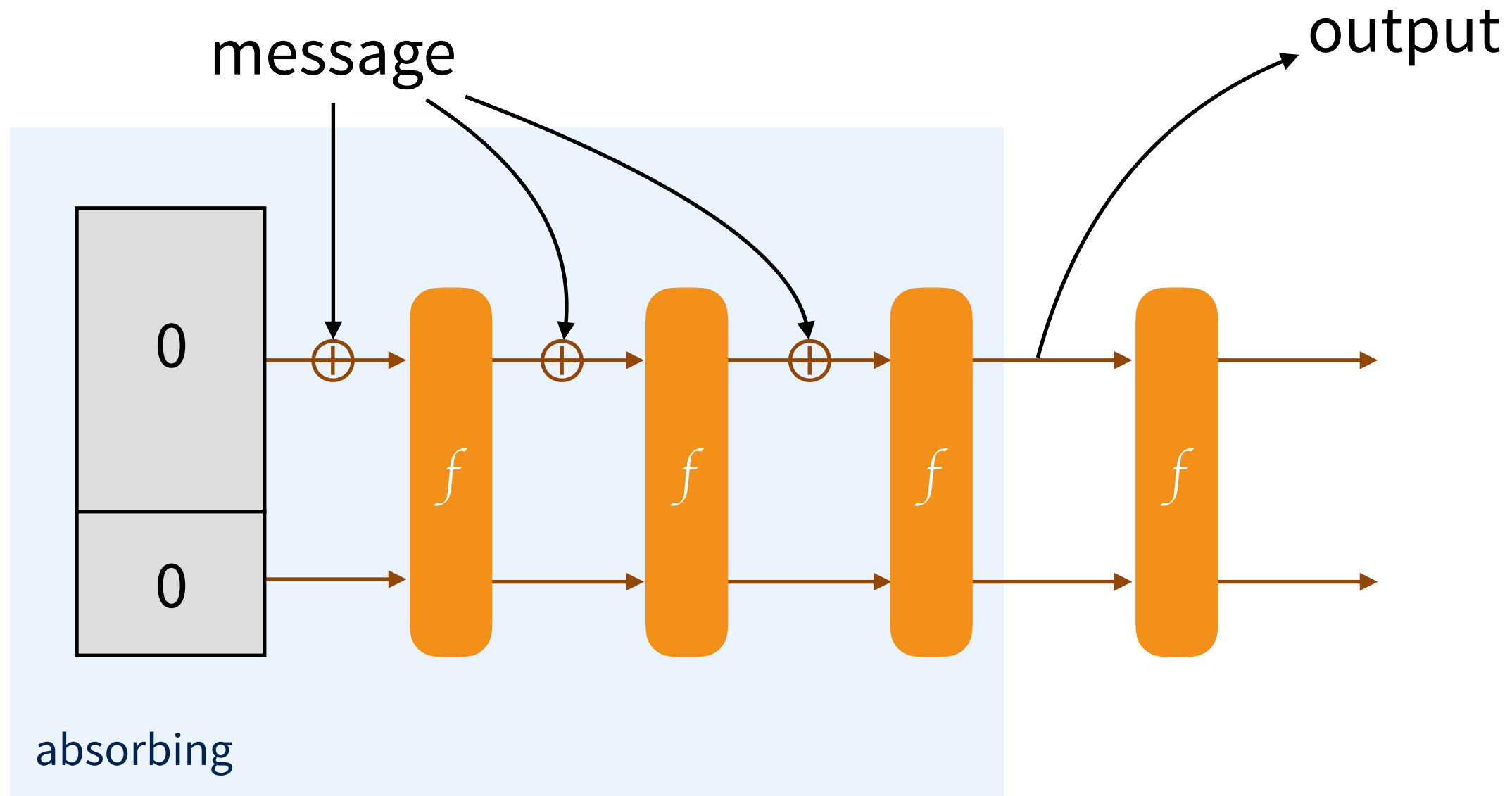
Sponge Construction



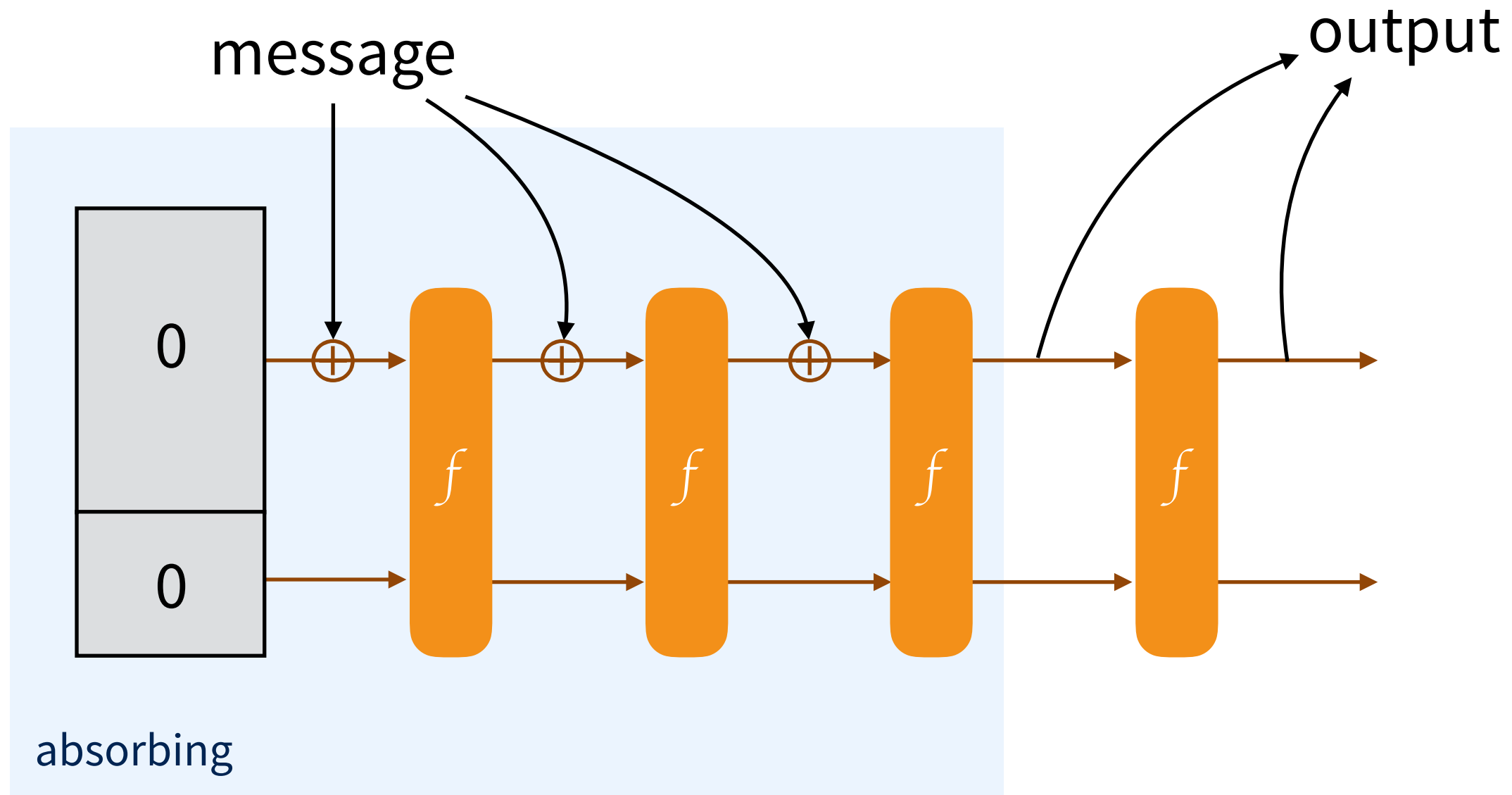
Sponge Construction



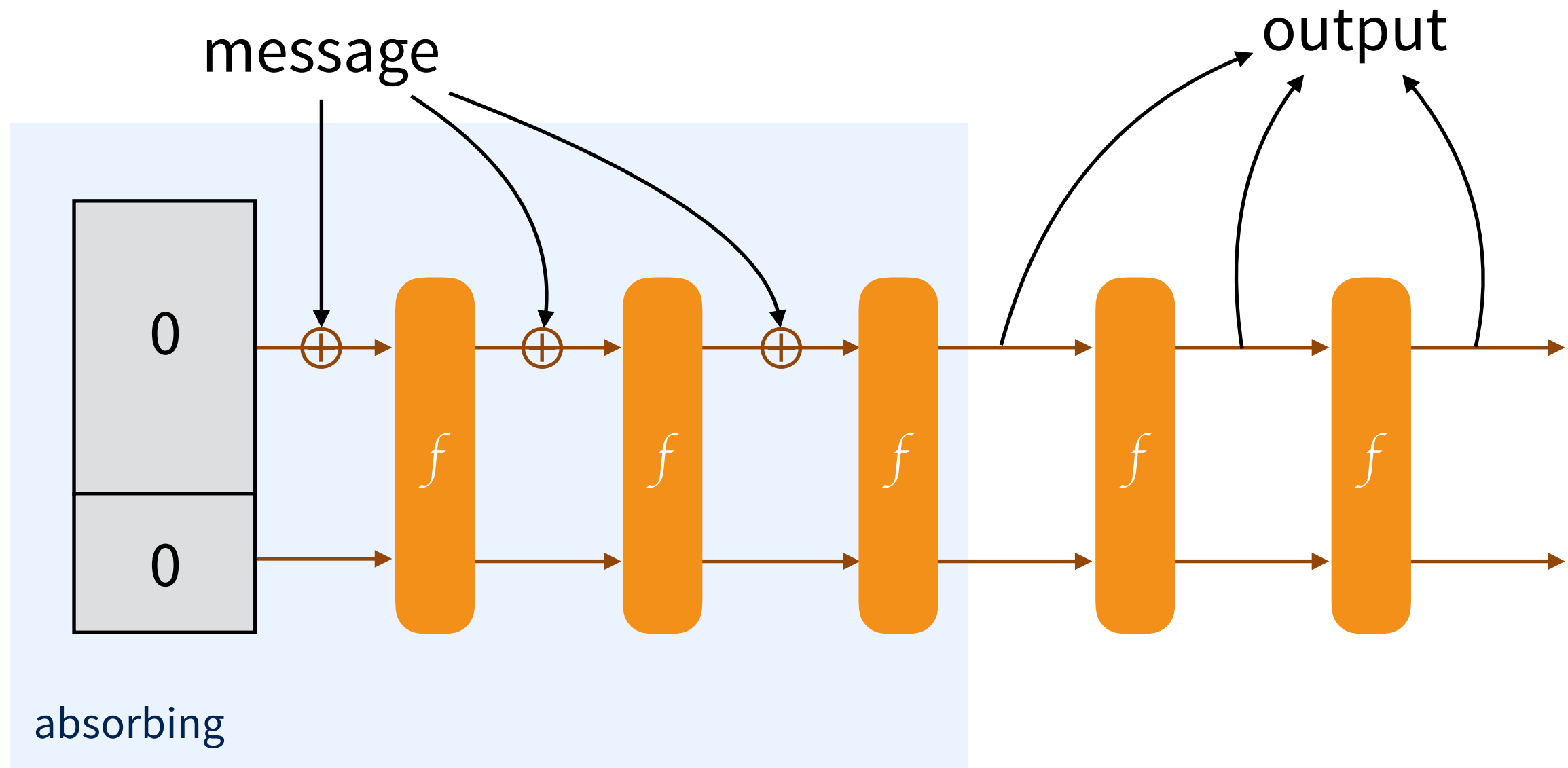
Sponge Construction



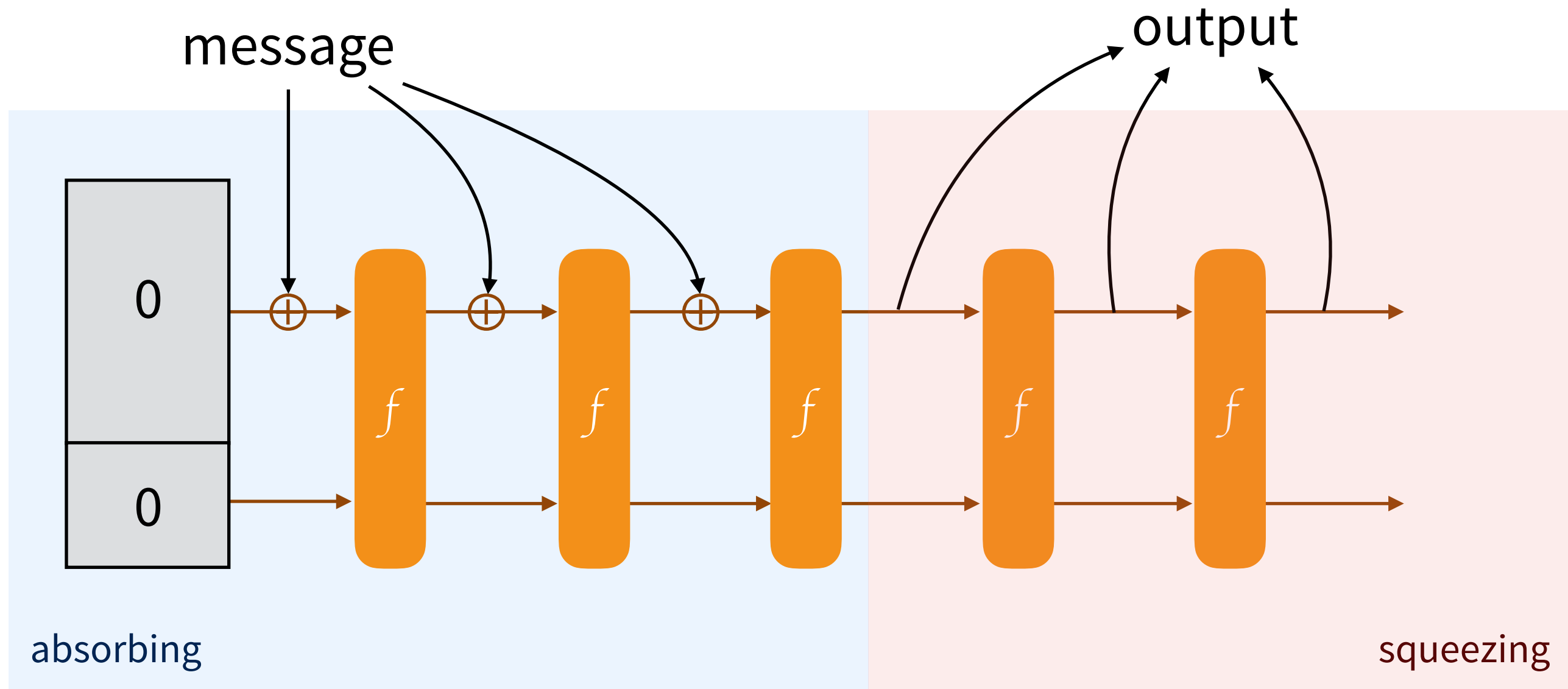
Sponge Construction



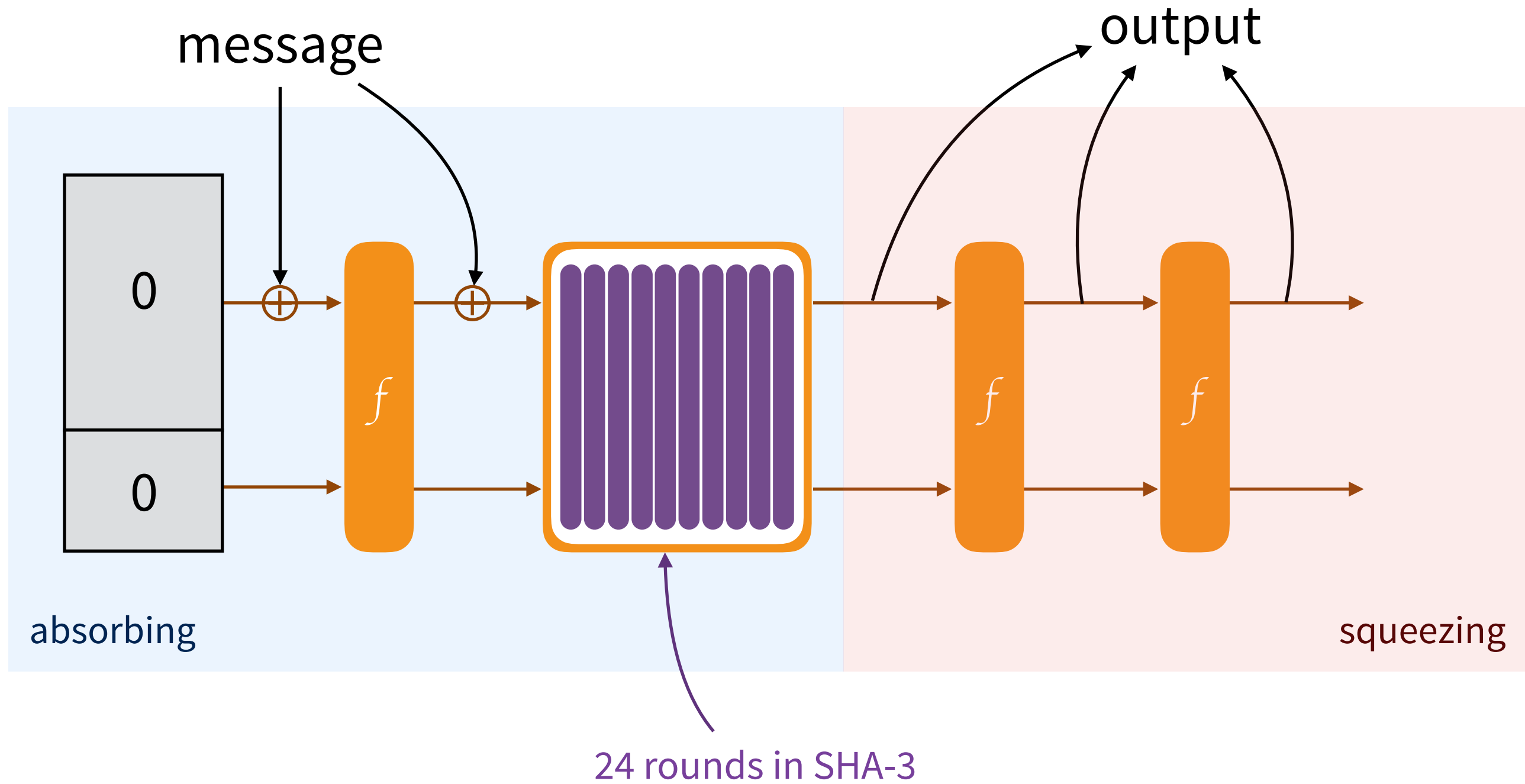
Sponge Construction



Sponge Construction



Sponge Construction



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

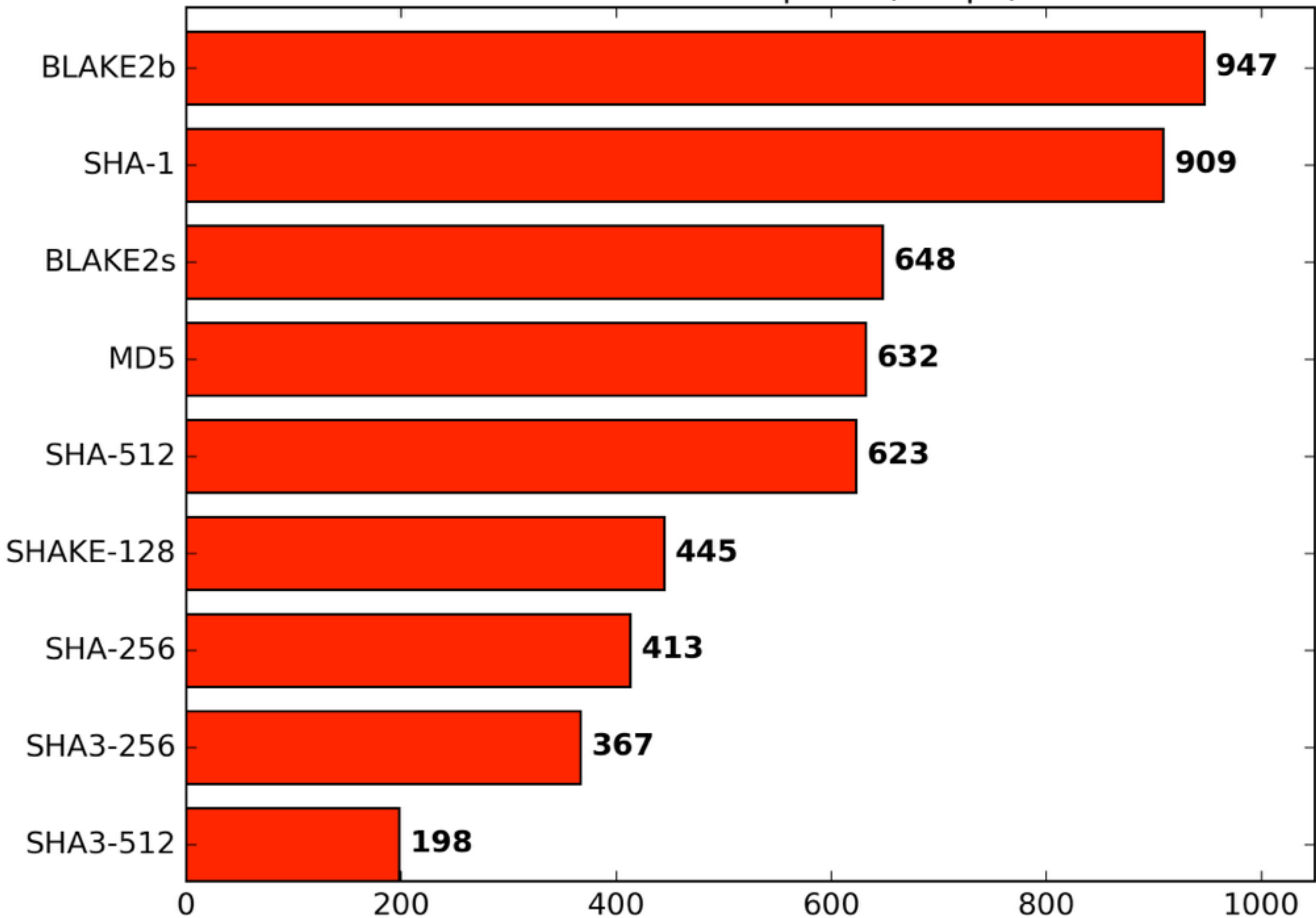
2007

SHA-3 competition

BLAKE2 →

2012

Hash functions speed (MiBps)



- **WolfSSL**: WolfSSL includes BLAKE2b
- **OpenSSL**: OpenSSL includes BLAKE2b and BLAKE2s
- **Wireguard**: The Wireguard VPN uses BLAKE2s for hashing and as a MAC
- **Botan**: The Botan library includes BLAKE2b
- **Crypto++**: The Crypto++ library includes BLAKE2s and BLAKE2b
- **Noise**: The Noise protocol (now used in WhatsApp) uses BLAKE2s and BLAKE2b
- **Cifra Extrema**: Cifra Extrema products use several versions of BLAKE2 in its servers and satellite modules
- **Bouncy Castle**: Includes BLAKE2b-160, BLAKE2b-256, BLAKE2b-384, and BLAKE2b-512
- **Peerio**: BLAKE2s is used to generate IDs and for integrity checks
- **8th**: BLAKE2s is the default hash in the 8th cross-platform development system
- **librsync**: BLAKE2b is the default hash in this popular remote delta-compression library
- **checksum**: BLAKE2s is one of the three hash functions supported with MD5 and SHA-1
- Password hashing schemes:
 - **Argon2** (by Biryukov, Dinu, Khovratovich; PHC winner)
 - **Catena** (by Forler, Lucks, Wenzel; PHC candidate)
 - **Lanarea** (by Mubarak; PHC candidate)
 - **Lyra and Lyra2** (by Simplicio Jr., Barreto, Almeida, Andrade; PHC candidate)
 - **Neoscrypt** (by Doering)
 - **RIG** (by Chang, Jati, Mishra, Sanadhya; PHC candidate)
 - **TwoCats** (by Cox; PHC candidate)
 - **Yarn** (by Capun; PHC candidate)
- Crypto tools by **catid**:
 - **Cymric** ("portable secure random number generator")
 - **Snowshoe** ("portable, secure, fast elliptic curve math library")
 - **Tabby** ("strong, fast, and portable cryptographic signatures and handshakes")
- **Sodium**: BLAKE2b is the default hash function of this cryptography library based on NaCl
- **Accumulus**: BLAKE2s is used for producing unique keys of the data stored
- **Archivarius 3000**: BLAKE2s is used for deduplication in this desktop search system

- **WolfSSL**: WolfSSL includes BLAKE2b
- **OpenSSL**: OpenSSL includes BLAKE2b and BLAKE2s
- **Wireguard**: The Wireguard VPN uses BLAKE2s for hashing and as a MAC
- **Botan**: The Botan library includes BLAKE2b
- **Crypto++**: The Crypto++ library includes BLAKE2s and BLAKE2b
- **Noise**: The Noise protocol (now used in WhatsApp) uses BLAKE2s and BLAKE2b
- **Cifra Extrema**: Cifra Extrema products use several versions of BLAKE2 in its servers and satellite modules
- **Bouncy Castle**: Includes BLAKE2b-160, BLAKE2b-256, BLAKE2b-384, and BLAKE2b-512
- **Peerio**: BLAKE2s is used to generated IDs and for integrity checks
- **8th**: BLAKE2s is the default hash in the 8th cross-platform development system
- **librsync**: BLAKE2b is the default hash un this popular remote delta-compression library
- **checksum**: BLAKE2s is one of the three hash functions supported with MD5 and SHA-1
- Password hashing schemes:
 - **Argon2** (by Biryukov, Dinu, Khovratovich; PHC winner)
 - **Catena** (by Forler, Lucks, Wenzel; PHC candidate)
 - **Lanarea** (by Mubarak; PHC candidate)
 - **Lyra and Lyra2** (by Simplicio Jr., Barreto, Almeida, Andrade; PHC candidate)
 - **Neoscript** (by Doering)
 - **RIG** (by Chang, Jati, Mishra, Sanadhya; PHC candidate)
 - **TwoCats** (by Cox; PHC candidate)
 - **Yarn** (by Capun; PHC candidate)
- Crypto tools by **catid**:
 - **Cymric** ("portable secure random number generator")
 - **Snowshoe** ("portable, secure, fast elliptic curve math library")
 - **Tabby** ("strong, fast, and portable cryptographic signatures and handshakes")
- **Sodium**: BLAKE2b is the default hash function of this cryptography library based on NaCl
- **Accumulus**: BLAKE2s is used for producing unique keys of the data stored
- **Archivarius 3000**: BLAKE2s is used for deduplication in this desktop search system

- **WolfSSL**: WolfSSL includes BLAKE2b
- **OpenSSL**: OpenSSL includes BLAKE2b and BLAKE2s
- **Wireguard**: The Wireguard VPN uses BLAKE2s for hashing and as a MAC
- **Botan**: The Botan library includes BLAKE2b
- **Crypto++**: The Crypto++ library includes BLAKE2s and BLAKE2b
- **Noise**: The Noise protocol (now used in WhatsApp) uses BLAKE2s and BLAKE2b
- **Cifra Extrema**: Cifra Extrema products use several versions of BLAKE2 in its servers and satellite modules
- **Bouncy Castle**: Includes BLAKE2b-160, BLAKE2b-256, BLAKE2b-384, and BLAKE2b-512
- **Peerio**: BLAKE2s is used to generated IDs and for integrity checks
- **8th**: BLAKE2s is the default hash in the 8th cross-platform development system
- **librsync**: BLAKE2b is the default hash un this popular remote delta-compression library
- **checksum**: BLAKE2s is one of the three hash functions supported with MD5 and SHA-1
- Password hashing schemes:
 - **Argon2** (by Biryukov, Dinu, Khovratovich; PHC winner)
 - **Catena** (by Forler, Lucks, Wenzel; PHC candidate)
 - **Lanarea** (by Mubarak; PHC candidate)
 - **Lyra and Lyra2** (by Simplicio Jr., Barreto, Almeida, Andrade; PHC candidate)
 - **Neoscript** (by Doering)
 - **RIG** (by Chang, Jati, Mishra, Sanadhya; PHC candidate)
 - **TwoCats** (by Cox; PHC candidate)
 - **Yarn** (by Capun; PHC candidate)
- Crypto tools by **catid**:
 - **Cymric** ("portable secure random number generator")
 - **Snowshoe** ("portable, secure, fast elliptic curve math library")
 - **Tabby** ("strong, fast, and portable cryptographic signatures and handshakes")
- **Sodium**: BLAKE2b is the default hash function of this cryptography library based on NaCl
- **Accumulus**: BLAKE2s is used for producing unique keys of the data stored
- **Archivarius 3000**: BLAKE2s is used for deduplication in this desktop search system

- **WolfSSL**: WolfSSL includes BLAKE2b
- **OpenSSL**: OpenSSL includes BLAKE2b and BLAKE2s
- **Wireguard**: The Wireguard VPN uses BLAKE2s for hashing and as a MAC
- **Botan**: The Botan library includes BLAKE2b
- **Crypto++**: The Crypto++ library includes BLAKE2s and BLAKE2b
- **Noise**: The Noise protocol (now used in WhatsApp) uses BLAKE2s and BLAKE2b
- **Cifra Extrema**: Cifra Extrema products use several versions of BLAKE2 in its servers and satellite modules
- **Bouncy Castle**: Includes BLAKE2b-160, BLAKE2b-256, BLAKE2b-384, and BLAKE2b-512
- **Peerio**: BLAKE2s is used to generate IDs and for integrity checks
- **8th**: BLAKE2s is the default hash in the 8th cross-platform development system
- **librsync**: BLAKE2b is the default hash in this popular remote delta-compression library
- **checksum**: BLAKE2s is one of the three hash functions supported with MD5 and SHA-1
- Password hashing schemes:
 - **Argon2** (by Biryukov, Dinu, Khovratovich; PHC winner)
 - **Catena** (by Forler, Lucks, Wenzel; PHC candidate)
 - **Lanarea** (by Mubarak; PHC candidate)
 - **Lyra and Lyra2** (by Simplicio Jr., Barreto, Almeida, Andrade; PHC candidate)
 - **Neoscrypt** (by Doering)
 - **RIG** (by Chang, Jati, Mishra, Sanadhya; PHC candidate)
 - **TwoCats** (by Cox; PHC candidate)
 - **Yarn** (by Capun; PHC candidate)
- Crypto tools by **catid**:
 - **Cymric** ("portable secure random number generator")
 - **Snowshoe** ("portable, secure, fast elliptic curve math library")
 - **Tabby** ("strong, fast, and portable cryptographic signatures and handshakes")
- **Sodium**: BLAKE2b is the default hash function of this cryptography library based on NaCl
- **Accumulus**: BLAKE2s is used for producing unique keys of the data stored
- **Archivarius 3000**: BLAKE2s is used for deduplication in this desktop search system

- **WolfSSL**: WolfSSL includes BLAKE2b
- **OpenSSL**: OpenSSL includes BLAKE2b and BLAKE2s
- **Wireguard**: The Wireguard VPN uses BLAKE2s for hashing and as a MAC
- **Botan**: The Botan library includes BLAKE2b
- **Crypto++**: The Crypto++ library includes BLAKE2s and BLAKE2b
- **Noise**: The Noise protocol (now used in WhatsApp) uses BLAKE2s and BLAKE2b
- **Cifra Extrema**: Cifra Extrema products use several versions of BLAKE2 in its servers and satellite modules
- **Bouncy Castle**: Includes BLAKE2b-160, BLAKE2b-256, BLAKE2b-384, and BLAKE2b-512
- **Peerio**: BLAKE2s is used to generate IDs and for integrity checks
- **8th**: BLAKE2s is the default hash in the 8th cross-platform development system
- **librsync**: BLAKE2b is the default hash in this popular remote delta-compression library
- **checksum**: BLAKE2s is one of the three hash functions supported with MD5 and SHA-1
- Password hashing schemes:
 - **Argon2** (by Biryukov, Dinu, Khovratovich; PHC winner)
 - **Catena** (by Forler, Lucks, Wenzel; PHC candidate)
 - **Lanarea** (by Mubarak; PHC candidate)
 - **Lyra and Lyra2** (by Simplicio Jr., Barreto, Almeida, Andrade; PHC candidate)
 - **Neoscrypt** (by Doering)
 - **RIG** (by Chang, Jati, Mishra, Sanadhya; PHC candidate)
 - **TwoCats** (by Cox; PHC candidate)
 - **Yarn** (by Capun; PHC candidate)
- Crypto tools by **catid**:
 - **Cymric** ("portable secure random number generator")
 - **Snowshoe** ("portable, secure, fast elliptic curve math library")
 - **Tabby** ("strong, fast, and portable cryptographic signatures and handshakes")
- **Sodium**: BLAKE2b is the default hash function of this cryptography library based on NaCl
- **Accumulus**: BLAKE2s is used for producing unique keys of the data stored
- **Archivarius 3000**: BLAKE2s is used for deduplication in this desktop search system

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:
<http://dx.doi.org/10.6028/NIST.FIPS.202>

August 2015



2007

SHA-3 competition

BLAKE2 → 2012

SHA-3 standard (FIPS 202) → 2015

gvanas / KeccakCodePackage

Watch 35 Unstar 174 Fork 60

Code Issues 1 Pull requests 1 Projects 0 Wiki Insights

Keccak Code Package

172 commits 1 branch 0 releases 15 contributors

Branch: master New pull request Create new file Upload files Find file Clone or download

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
PISnP	Added more AVX-512 implementations	5 months ago
SnP	uxth needs two parameters	3 months ago

github.com/gvanas/KeccakCodePackage



TweetFIPS202

@TweetFIPS202

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

Joined August 2015

[Tweet to TweetFIPS202](#)

12 Followers you know



Tweets
9

Followers
43

Tweets

Tweets & replies



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)



1



1



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%



1



1



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]



1



1



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,7)if((R=(R<<1)^(113*(R>>7)))&2)*s^=



2



1

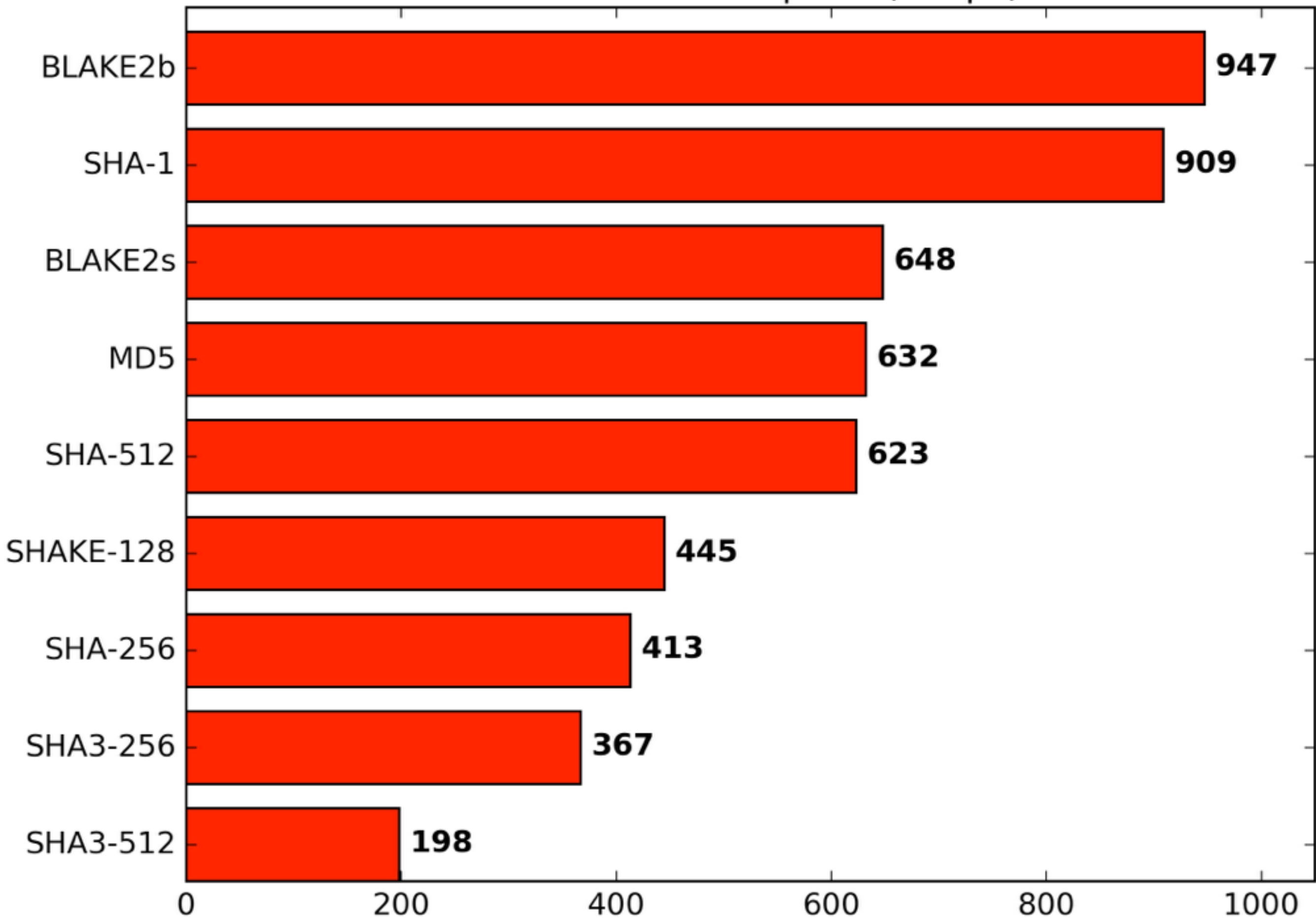


TweetFIPS202 @TweetFIPS202 · 17 Aug 2015

];}FOR(x,5){t=B[(x+4)%5]^ROL(B[(x+1)%5],1);FOR(y,5)s[x+5*y]^=t;t=s[1];y=r=0;x=1;FOR(j,24){r+=j+1;Y=2*x+3*y;x=y;y=Y%5;Y=s[x+5*y];s[x+5*y]=

[Translate from Spanish](#)

Hash functions speed (MiBps)



Bit Security

- bit security of AES-128?
- bit security of AES-256?
- bit security against pre-image attacks of SHA-256?
- bit security against pre-image attacks of SHA-512?
- bit security against pre-image attacks of SHA-3-512?



Where is **SHA-3** being used?



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:
<http://dx.doi.org/10.6028/NIST.FIPS.202>

August 2015





SHAKE is a XOF

The KECCAK sponge function family

Guido Bertoni¹, Joan Daemen^{1,2}, Michaël Peeters¹ and Gilles Van Assche¹

¹STMicroelectronics

²Radboud University

Tune KECCAK to your requirements

The capacity parameter and chosen output length in KECCAK can be freely chosen. Their combination determines the attainable security and the capacity has an impact on performance. This page gives you the optimal capacity and output length values, given the classical hash function criteria.

Required collision resistance: 2^x	$x =$	0 (don't care)	⬇
Required (second) preimage resistance: 2^y	$y =$	128	⬇

The optimal choice of parameters is:

KECCAK[$r=1344, c=256$] with a least **128 bits** of output.

Security claim

In line with our [hermetic sponge strategy](#), we make a [flat sponge claim](#) with $c=256$ bits of capacity: for any output length, we claim this KECCAK sponge function resists any attack up to 2^{128} operations (each of complexity equivalent to one call to KECCAK- f), unless easier on a random oracle. For 128 bits of output specifically, this translates into the following claimed security level:

	Claimed security level
Collision resistance	2^{64}
(Second) preimage resistance	2^{128}

Speed

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\$](#)

keccak.noekeon.org/tune.html

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>

C O M P U T E R S E C U R I T Y

2007

SHA-3 competition

BLAKE2 → 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

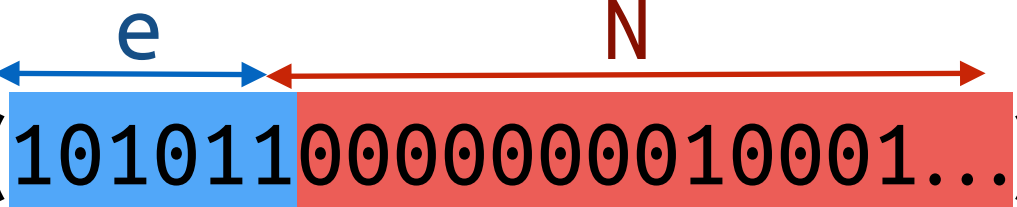
my RSA public key = (e, N)
fingerprint = SHA-256(e || N)

ParallelHash

KMAC

message || SHAKE(key || message)

TupleHash

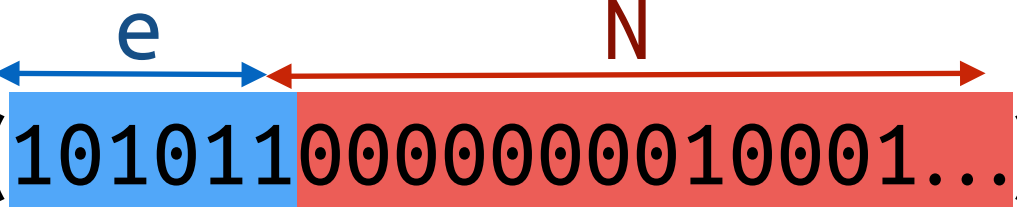
fingerprint1 = SHA-256()

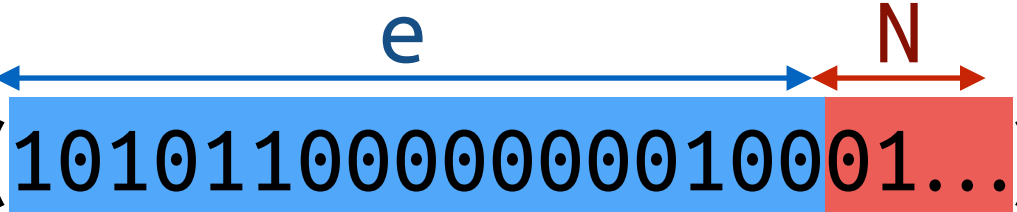
ParallelHash

KMAC

message || SHAKE(key || message)

TupleHash

fingerprint1 = SHA-256()

fingerprint2 = SHA-256()

ParallelHash

KMAC

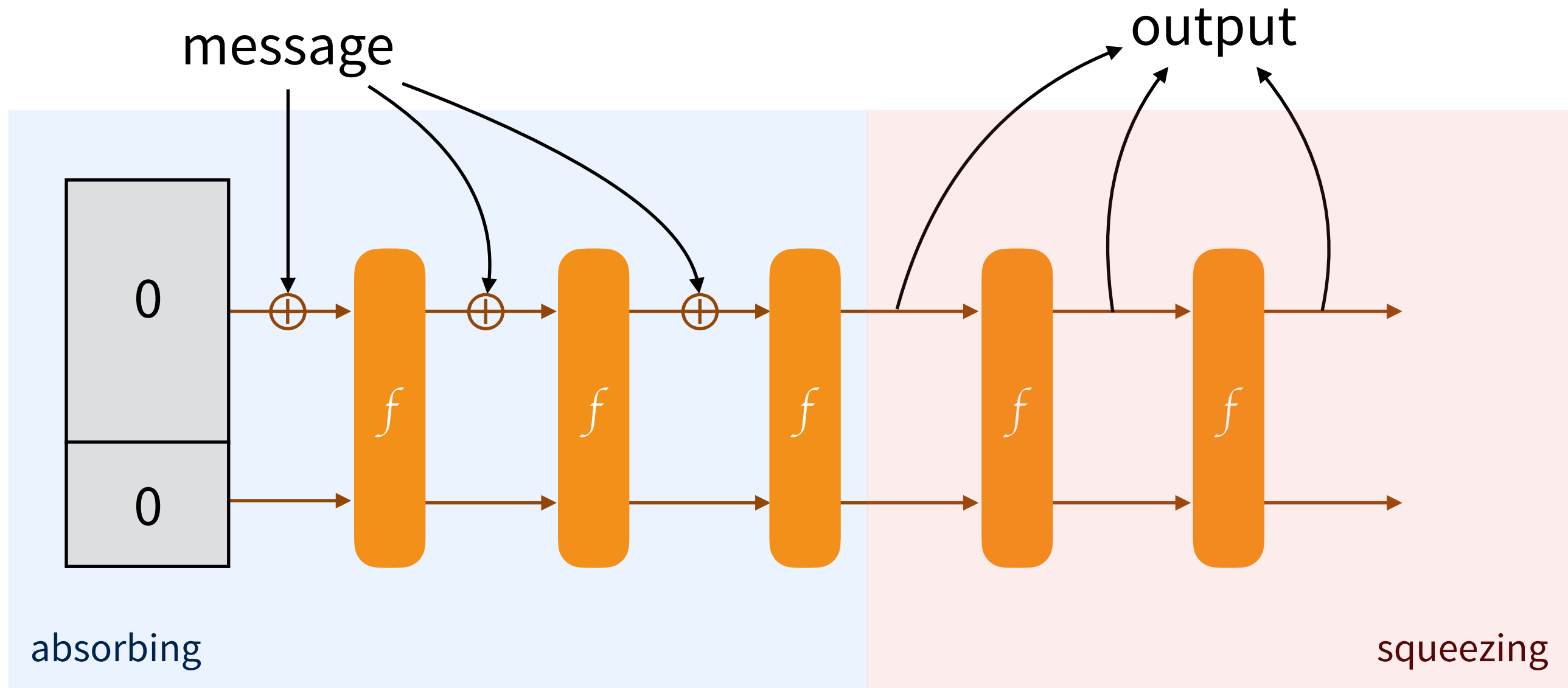
message || SHAKE(key || message)

TupleHash

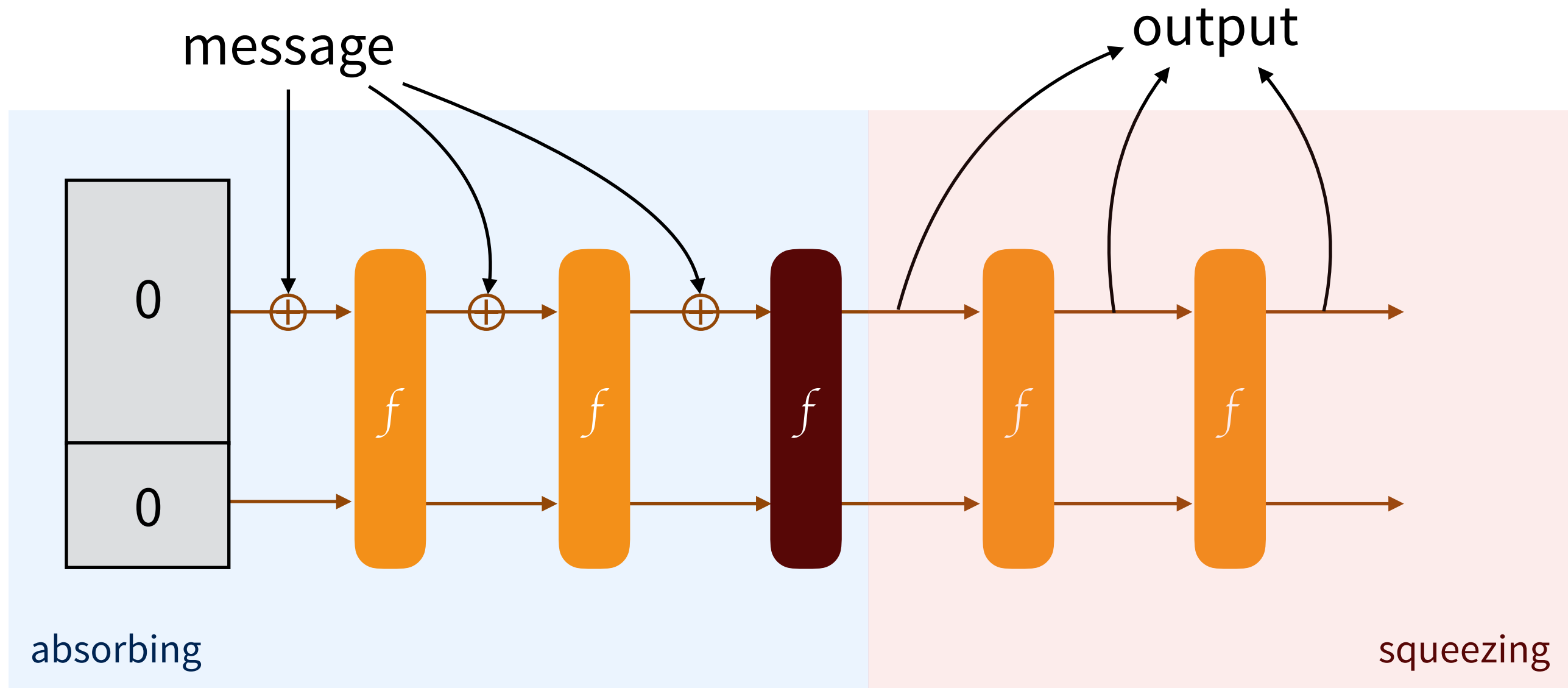
SHAKE(len(e) || e || len(N) || N)

ParallelHash

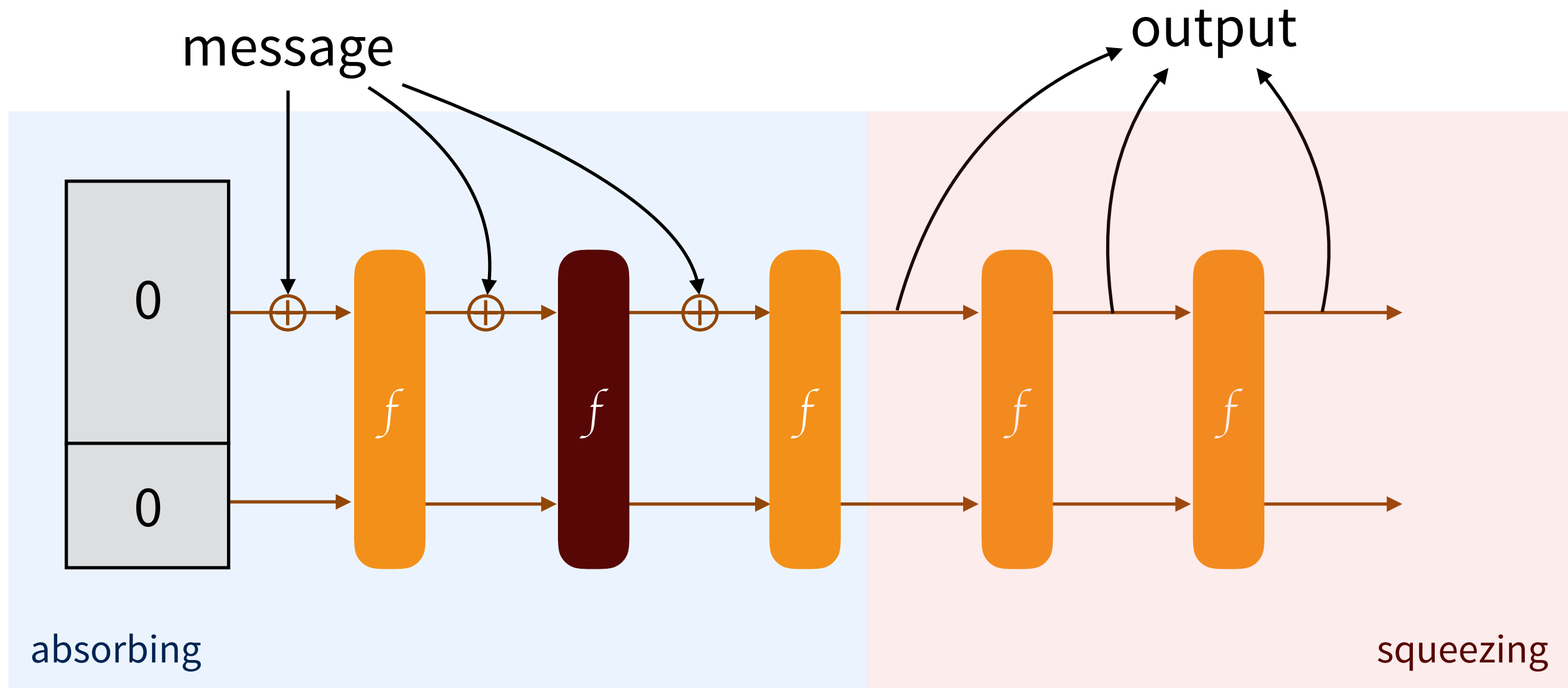
Sponge Construction



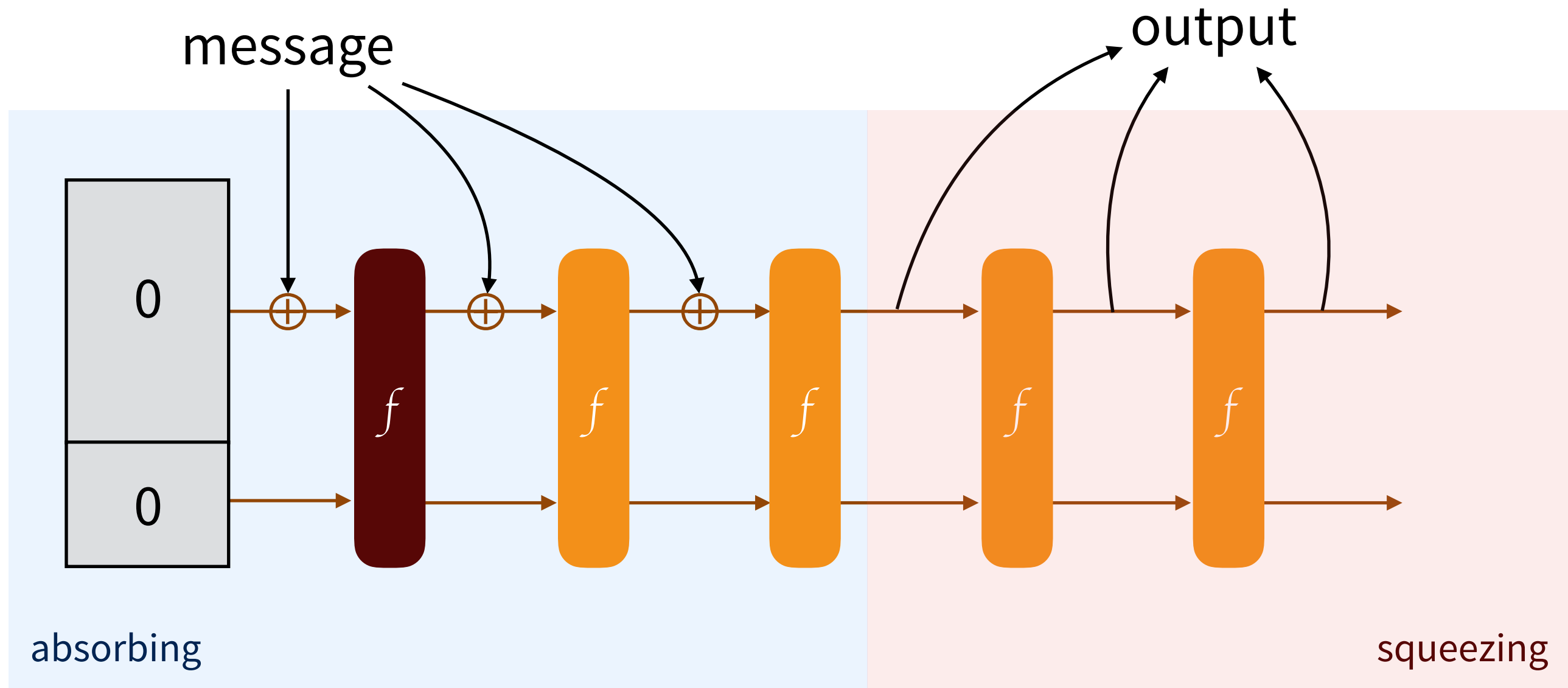
Sponge Construction



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

BLAKE2 → 2012

SHA-3 / SHAKE → 2015

TupleHash / ParallelHash / KMAC → 2016

Keyak and Ketje

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

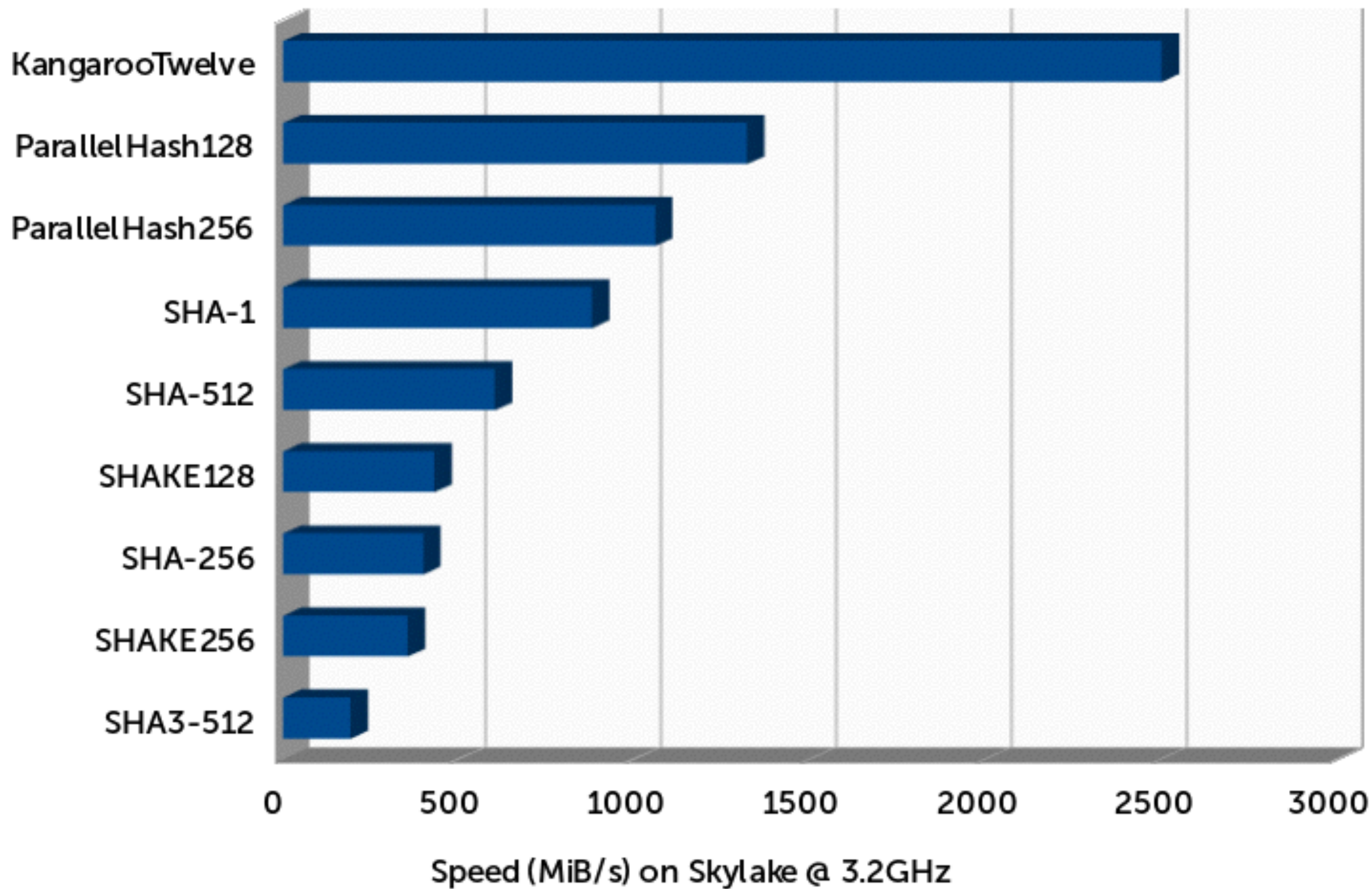
BLAKE2 → 2012

SHA-3 / SHAKE → 2015

TupleHash / ParallelHash / KMAC → 2016

**KangarooTwelve
& MarsupilamiFourteen**





2007

SHA-3 competition

BLAKE2 → 2012

SHA-3 / SHAKE → 2015

TupleHash / ParallelHash / KMAC → 2016

**KangarooTwelve
& MarsupilamiFourteen**



gvanas / KeccakCodePackage

Watch 35 Unstar 174 Fork 60

Code Issues 1 Pull requests 1 Projects 0 Wiki Insights

Keccak Code Package

172 commits 1 branch 0 releases 15 contributors

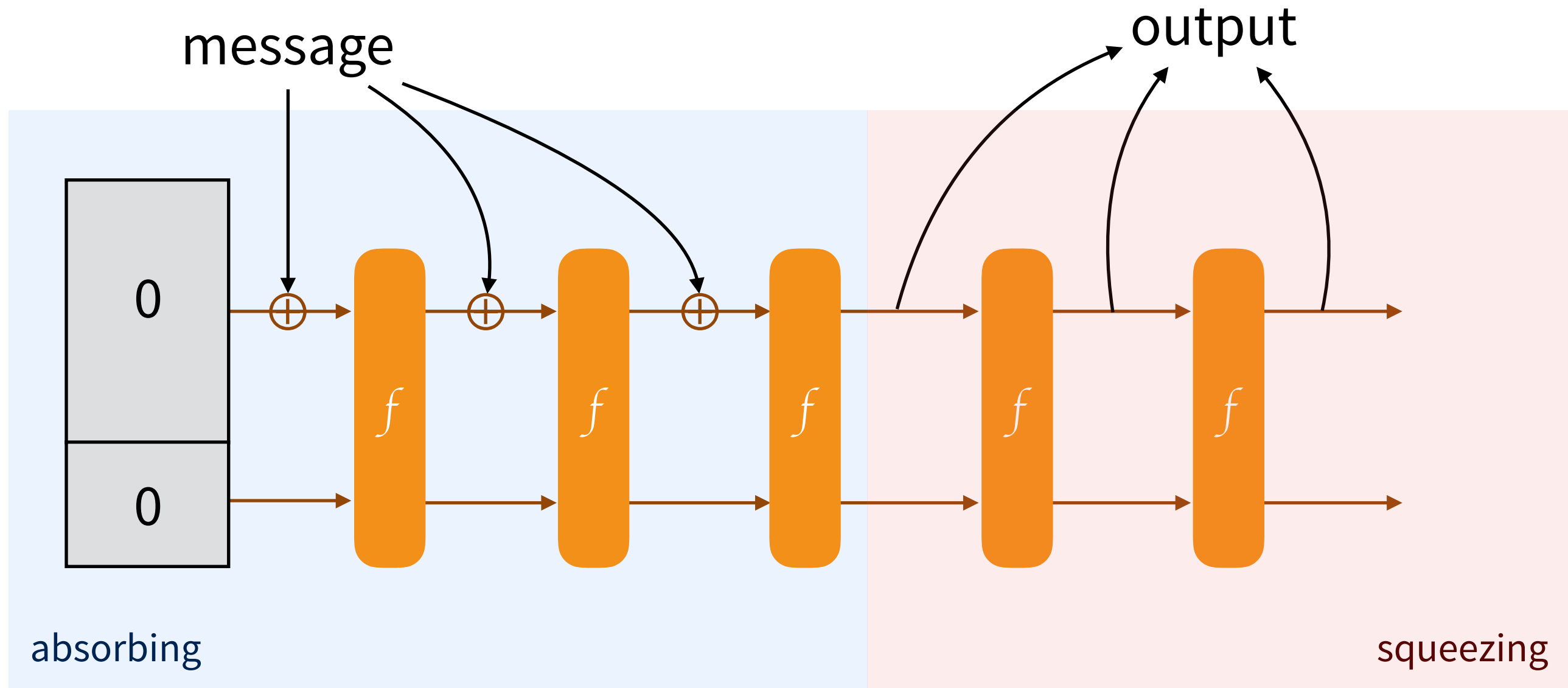
Branch: master New pull request Create new file Upload files Find file Clone or download

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
PISnP	Added more AVX-512 implementations	5 months ago
SnP	uxth needs two parameters	3 months ago

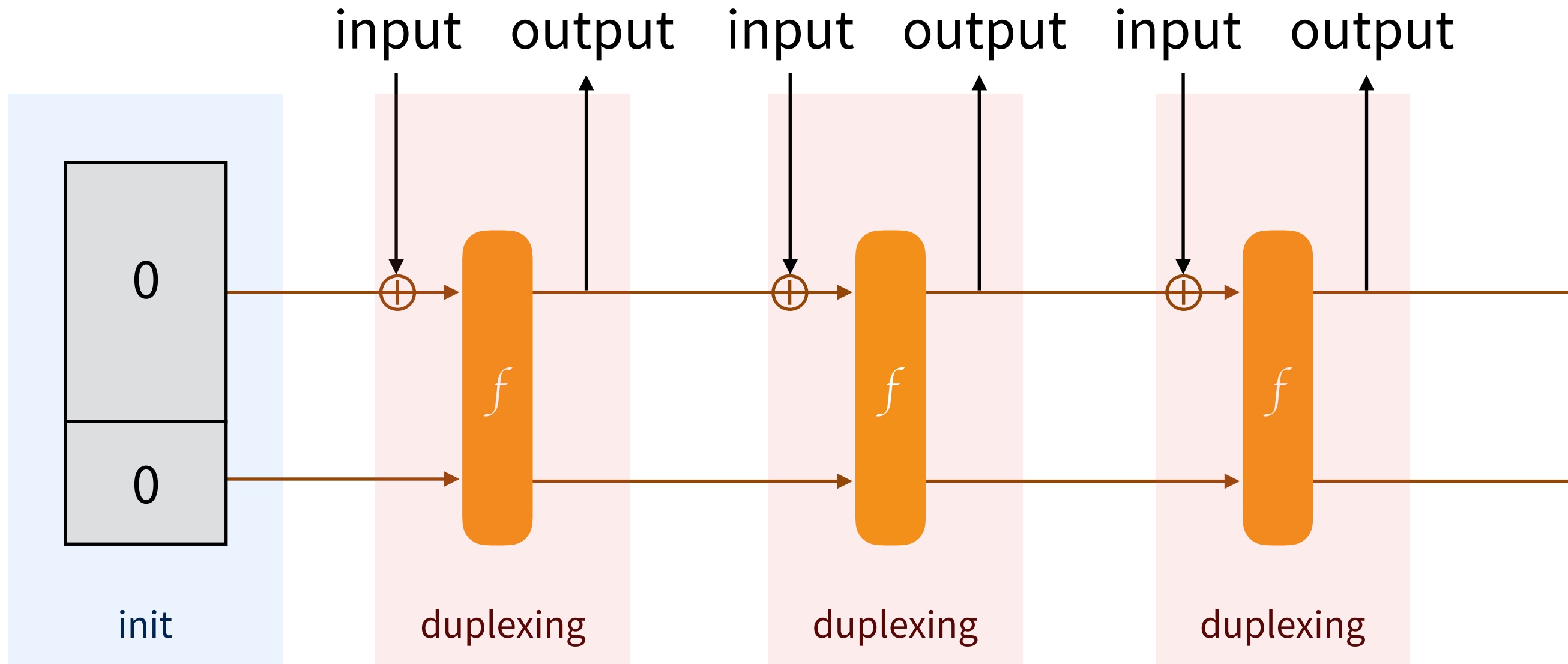
github.com/gvanas/KeccakCodePackage

Part II: Strobe

Sponge Construction



Duplex Construction



Symmetric Protocol

```
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

internal operations

default

state = input \oplus state

cbefore

state = input

cafter

output, state = input \oplus state

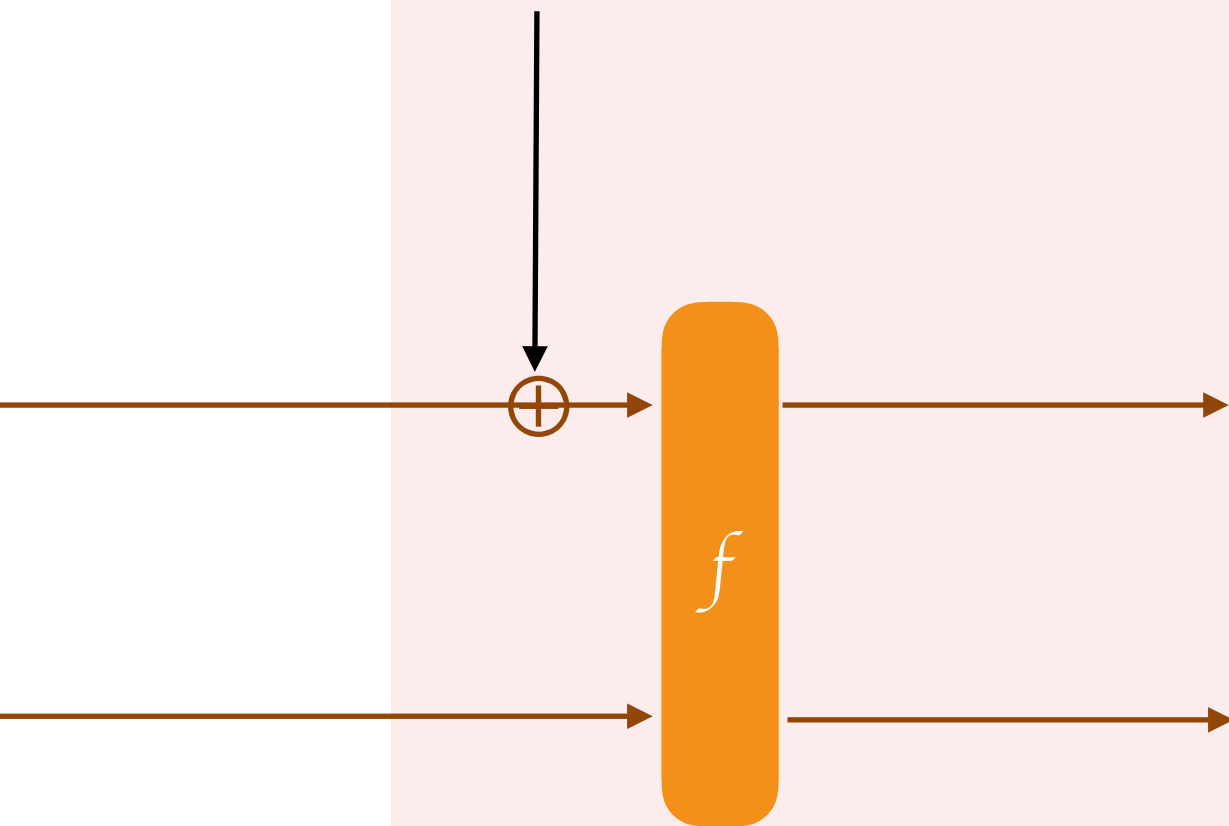
forceF

the permutation is ran before the operation

operation = KEY

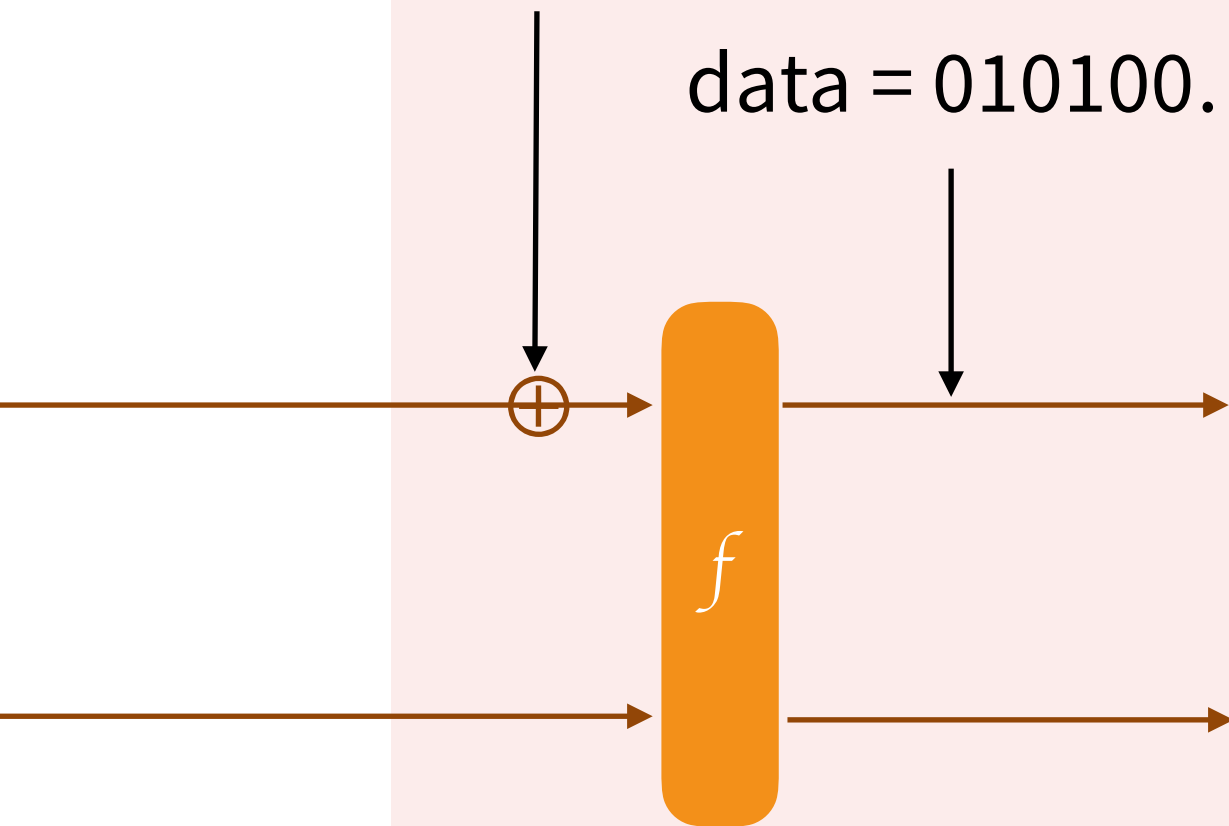


operation = KEY



operation = KEY

data = 010100...



operation = KEY

data = 010100...

\oplus

f

operation = send_ENC

data = hello

ciphertext

\oplus

f

\oplus

operation = AD

data = 010100...



operation = AD

data = 010100...



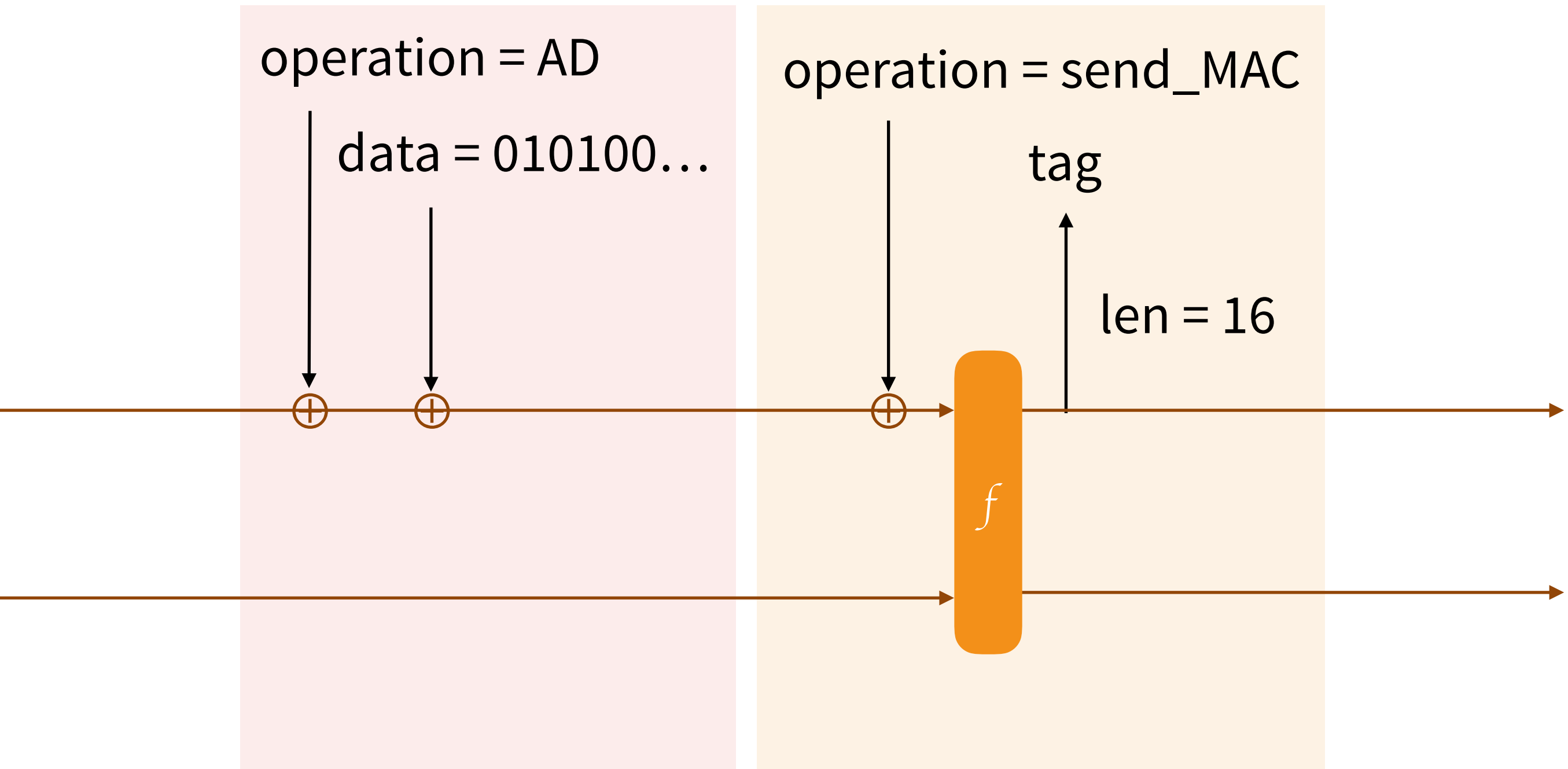
operation = send_MAC



tag

len = 16

f



operation = send_CLR

data = 010100...



operation = send_CLR

data = 010100...



operation = RATCHET

data = 00000



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:]
```

STROBE protocol framework

[overview](#)[specification](#)[example protocols](#)[code](#)[papers](#)

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

Part III: Disco?

```
115 // ... the patterns
116 for _, pattern := range patterns {
117     pattern = strings.Trim(pattern, " ")
118
119     if pattern == "e" {
120         h.e = GenerateKeypair()
121         *messageBuffer = append(*messageBuffer, h.e.publicKey[:]...)
122         h.strobeState.Send_CLR(false, h.e.publicKey[:])
123     } else if pattern == "s" {
124         *messageBuffer = append(*messageBuffer, h.strobeState.Send_AEAD(h.s.publicKey[:], []byte{}))
125     } else if pattern == "ee" {
126         h.strobeState.Send_AEAD(h.e.publicKey[:], []byte{})
127     }
128 }
```

Noise + Strobe = **Disco**

github.com/mimoo/NoiseGo/disco/specification.md

Where to find me

 cryptologie.net

 twitter.com/lyon01_david