

Course Announcements

- Assignments
 - Homework 10 has been posted, due Wednesday 4/29
 - Reading: *The Block Cipher Companion*, Section 6.1
 - (These are the final homework assignment + required reading for the course)
- Final exam
 - Take-home exam on May 5-6
 - Details on Piazza post 227
- Next week: guest lectures by Prof. Andy Sellars on crypto and the law

Lecture 24: Keccak, the new SHA-3 hash function

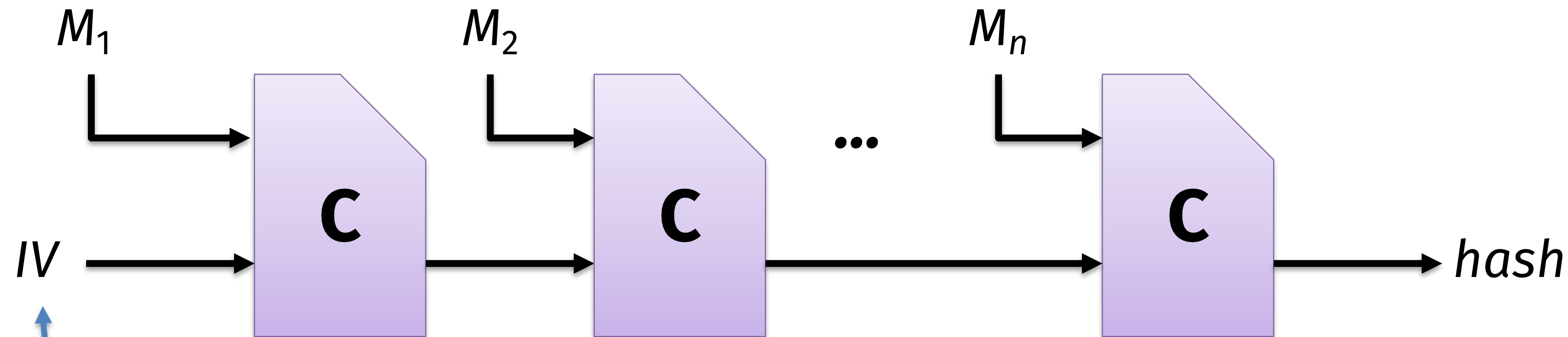
1. Concerns with Merkle-Damgård
2. Sponge functions and Keccak
3. Building cryptosystems from sponge functions
4. The Keccak-f permutation
5. Conclusion

1. Concerns with Merkle-Damgård

Reminder: Merkle-Damgård paradigm

Build a variable-length input hash function from two primitives:

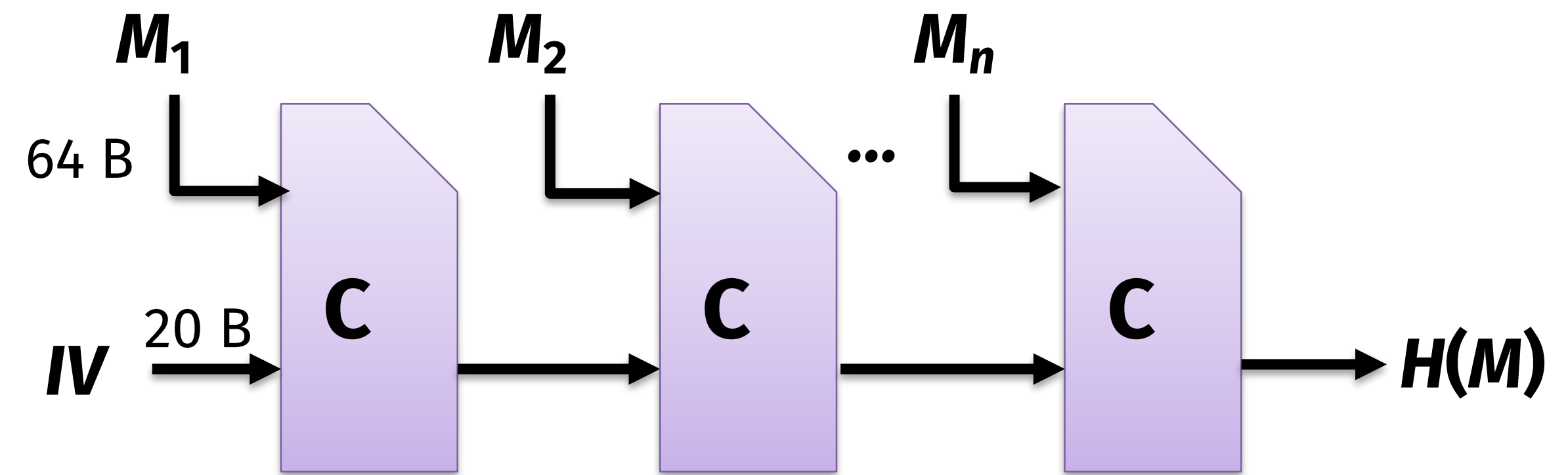
1. A fixed-length, compressing random-looking function
2. A mode of operation that iterates this function multiple times in a smart manner



IV for hash function is typically fixed in spec, not user configurable

Problems with SHA-1's compression function

- C has 160 bits of output \Rightarrow birthday bound yields collisions in (expected) 2^{80} steps
- Wang, Yin, Yu 2004: discover an algorithm that can find a SHA-1 collision in 2^{69} steps



JP Aumasson @veorq · 5 Jan 2016

2016 crypto predictions: like the last years,
no SHA-1 collision
no hisev vuln in libotr
also,
more postquantum
vulns in secure messengers



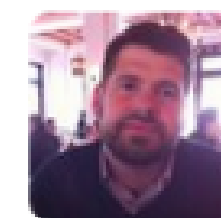
10



36



32



Matthew Green

@matthew_d_green

Follow

@veorq I will bet you on the SHA1 collision.
Let's say a dinner or something.

2017: SHA-1 SHAttered!

The first collision for full SHA-1

Marc Stevens¹, Elie Bursztein², Pierre Karpman¹, Ange Albertini², Yarik Markov²

¹ CWI Amsterdam

² Google Research

info@shattered.io

<https://shattered.io>

Table 1: Colliding message blocks for SHA-1.																				
CV_0	4e	a9	62	69	7c	87	6e	26	74	d1	07	f0	fe	c6	79	84	14	f5	bf	45
$M_1^{(1)}$			<u>7f</u>	46	dc	<u>93</u>	<u>a6</u>	b6	7e	<u>01</u>	<u>3b</u>	02	9a	<u>aa</u>	<u>1d</u>	b2	56	<u>0b</u>		
			<u>45</u>	ca	67	<u>d6</u>	<u>88</u>	c7	f8	<u>4b</u>	<u>8c</u>	4c	79	<u>1f</u>	<u>e0</u>	2b	3d	<u>f6</u>		
			<u>14</u>	f8	6d	<u>b1</u>	<u>69</u>	09	01	<u>c5</u>	<u>6b</u>	45	c1	<u>53</u>	<u>0a</u>	fe	df	<u>b7</u>		
			<u>60</u>	38	e9	<u>72</u>	<u>72</u>	2f	e7	<u>ad</u>	72	8f	0e	<u>49</u>	<u>04</u>	e0	46	<u>c2</u>		
$CV_1^{(1)}$	8d	64	<u>d6</u>	<u>17</u>	ff	ed	<u>53</u>	<u>52</u>	eb	c8	59	15	5e	c7	eb	<u>34</u>	<u>f3</u>	8a	5a	7b
$M_2^{(1)}$			<u>30</u>	57	0f	<u>e9</u>	<u>d4</u>	13	98	<u>ab</u>	<u>e1</u>	2e	f5	<u>bc</u>	<u>94</u>	2b	e3	<u>35</u>		
			<u>42</u>	a4	80	<u>2d</u>	<u>98</u>	b5	d7	<u>0f</u>	<u>2a</u>	33	2e	<u>c3</u>	<u>7f</u>	ac	35	<u>14</u>		
			<u>e7</u>	4d	dc	<u>0f</u>	<u>2c</u>	c1	a8	<u>74</u>	<u>cd</u>	0c	78	<u>30</u>	<u>5a</u>	21	56	<u>64</u>		
			<u>61</u>	30	97	<u>89</u>	<u>60</u>	6b	d0	<u>bf</u>	3f	98	cd	<u>a8</u>	<u>04</u>	46	29	<u>a1</u>		
CV_2	1e	ac	b2	5e	d5	97	0d	10	f1	73	69	63	57	71	bc	3a	17	b4	8a	c5
CV_0	4e	a9	62	69	7c	87	6e	26	74	d1	07	f0	fe	c6	79	84	14	f5	bf	45
$M_1^{(2)}$			<u>73</u>	46	dc	<u>91</u>	<u>66</u>	b6	7e	<u>11</u>	<u>8f</u>	02	9a	<u>b6</u>	<u>21</u>	b2	56	<u>0f</u>		
			<u>f9</u>	ca	67	<u>cc</u>	<u>a8</u>	c7	f8	<u>5b</u>	<u>a8</u>	4c	79	<u>03</u>	<u>0c</u>	2b	3d	<u>e2</u>		
			<u>18</u>	f8	6d	<u>b3</u>	<u>a9</u>	09	01	<u>d5</u>	<u>df</u>	45	c1	<u>4f</u>	<u>26</u>	fe	df	<u>b3</u>		
			<u>dc</u>	38	e9	<u>6a</u>	<u>c2</u>	2f	e7	<u>bd</u>	72	8f	0e	<u>45</u>	<u>bc</u>	e0	46	<u>d2</u>		
$CV_1^{(2)}$	8d	64	<u>c8</u>	<u>21</u>	ff	ed	<u>52</u>	<u>e2</u>	eb	c8	59	15	5e	c7	eb	<u>36</u>	<u>73</u>	8a	5a	7b
$M_2^{(2)}$			<u>3c</u>	57	0f	<u>eb</u>	<u>14</u>	13	98	<u>bb</u>	<u>55</u>	2e	f5	<u>a0</u>	<u>a8</u>	2b	e3	<u>31</u>		
			<u>fe</u>	a4	80	<u>37</u>	<u>b8</u>	b5	d7	<u>1f</u>	<u>0e</u>	33	2e	<u>df</u>	<u>93</u>	ac	35	<u>00</u>		
			<u>eb</u>	4d	dc	<u>0d</u>	<u>ec</u>	c1	a8	<u>64</u>	<u>79</u>	0c	78	<u>2c</u>	<u>76</u>	21	56	<u>60</u>		
			<u>dd</u>	30	97	<u>91</u>	<u>d0</u>	6b	d0	<u>af</u>	3f	98	cd	<u>a4</u>	<u>bc</u>	46	29	<u>b1</u>		
CV_2	1e	ac	b2	5e	d5	97	0d	10	f1	73	69	63	57	71	bc	3a	17	b4	8a	c5

- Effort to find the collision
 - 6500 CPU-years
 - 110 GPU-years
- # hashes: 9,223,372,036,854,775,808
 - $\approx 9 \text{ quintillion} = 9 \cdot (10^3)^6$
 - $\approx 2^3 \cdot 2^{60} = 2^{63}$
- (Actual effort, from paper: $2^{63.1}$)
- “The SHAttered attack is 100,000 faster than the brute force attack that relies on the birthday paradox”

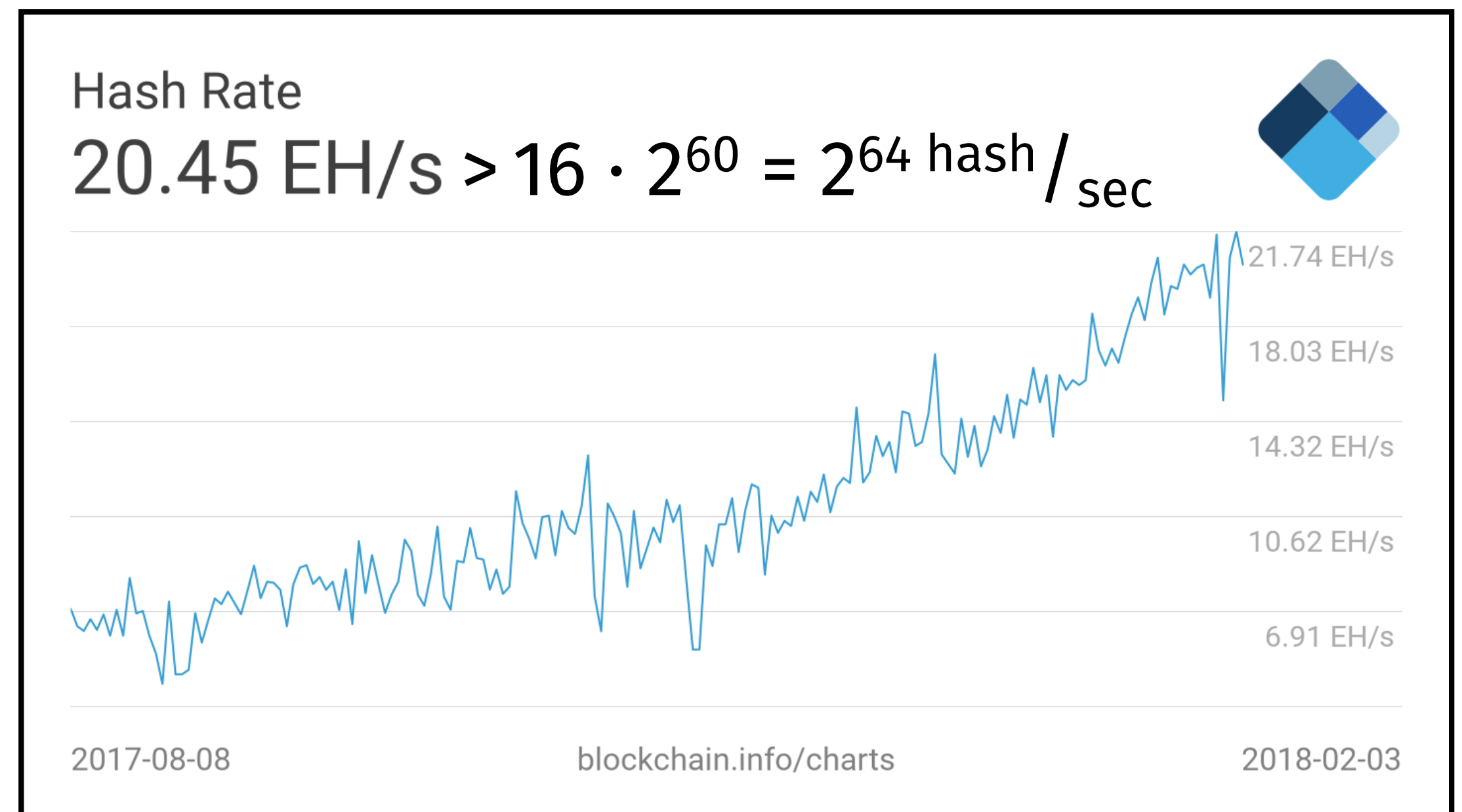
Scale of computation

“One of the largest computations ever completed”

– Google blog post

(<https://security.googleblog.com/2017/02/announcing-first-sha1-collision.html>)

Bitcoin’s collective Double SHA-2 hash rate, Feb 2018
(<https://blockchain.info/charts/hash-rate>)

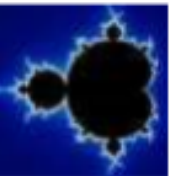


Reactions to the SHA-1 collision



Matthew Green
@matthew_d_green

I hope this goes without saying, but literally the only surprising thing about SHA1 collisions is that it happened in 2017, not 2015.



Adam Langley
@agl_

Remember: things would be much more awkward today if @sleeve_ and @arw and friends hadn't spent years working to rid the Web PKI of SHA-1.



J.C. Jones
@jamespugjones

Tomorrow SHA-1 will be shut off for all @firefox users; we've been rolling that out to increasing %s of users over the last month.



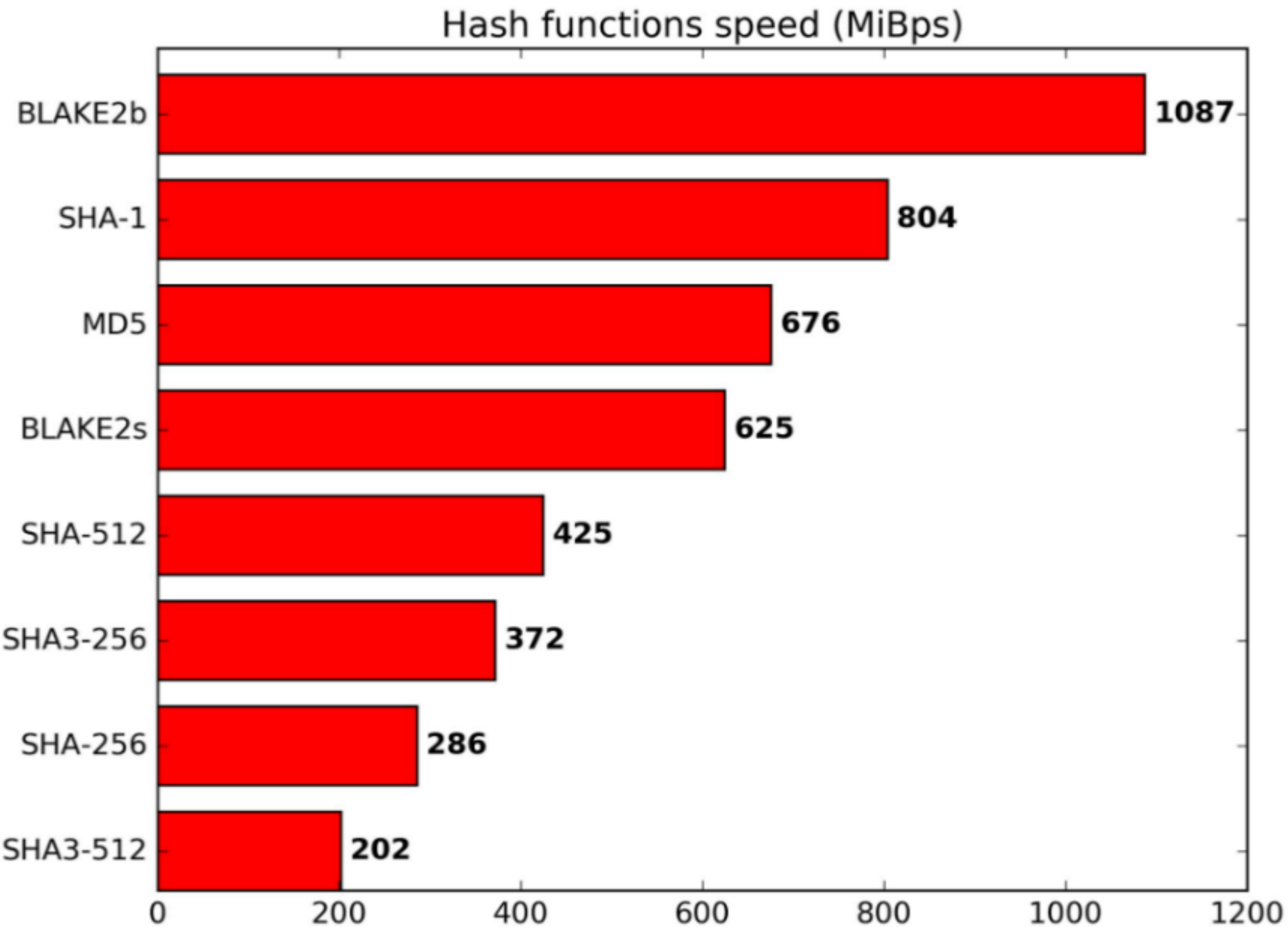
Pinboard
@Pinboard

Real world scenarios: an attacker could use the SHA1 collision to drone on about math endlessly until you feel like you just want to die



JP Aumasson
@veorq

Seriously, now, ditch SHA-1 and use BLAKE2



Replicating the attack

- Finding your own message blocks that collide will take another $2^{63.1}$ effort
- ...Unless you exploit the length extension property of Merkle-Damgard!

SHA1 collider

Quick-and-dirty PDF maker using the collision from the **SHAttered** paper.

Choose two image files (must be JPG, roughly the same aspect ratio). For now, each file must be less than 64kB.

Aspect ratio

x

Browse...

No file selected.

Browse...

No file selected.

Submit Query

Source: <https://alf.nu/SHA1>

Formal comparison of attacks

SHA-1: identical prefix attack

There *exist* specific strings A and B such that for all strings C:

$$\text{SHA1}(A \parallel C) = \text{SHA1}(B \parallel C)$$

MD-5: Chosen prefix attack

Given *any* strings A and B, can compute strings C and D such that

$$\text{MD5}(A \parallel C) = \text{MD5}(B \parallel D)$$

We have used a Sony Playstation 3 to correctly predict the outcome of the 2008 US presidential elections. In order not to influence the voters we keep our prediction secret, but commit to it by publishing its cryptographic hash on this website. The document with the correct prediction and matching hash will be revealed after the elections.

Site: www.win.tue.nl/hashclash/Nostradamus/

A: 	John Edwards.pdf	G: 	Fred Thompson.pdf
B: 	John McCain.pdf	H: 	(hidden)
C: 	Mitt Romney.pdf	I: 	Paris Hilton.pdf
D: 	Ralph Nader.pdf	J: 	Al Gore.pdf
E: 	(hidden)	K: 	Jeb Bush.pdf
F: 	Barack Obama.pdf	L: 	Oprah Winfrey.pdf

All twelve documents we prepared, the ten given above and two hidden ones, have the MD5 hash value

3D515DEAD7AA16560ABA3E9DF05CBC80.

2020: SHA-mbles

- In 2020, SHA-1 is still used in
 - 3% of Alexa top 1 million websites
 - HMAC-SHA1 for 8% of websites
 - ~1% of email certs (PGP and X.509)
- New paper reduces attack cost by another factor of ~10

	<u>Work</u>	<u>GPU cost</u>
Identical prefix attack:	$2^{61.1}$	\$11k
Chosen prefix attack:	$2^{63.4}$	\$45k

SHA-1 is a Shambles

First Chosen-Prefix Collision on SHA-1
and Application to the PGP Web of Trust

Gaëtan Leurent¹ and Thomas Peyrin^{2,3}

¹ Inria, France

² Nanyang Technological University, Singapore

³ Temasek Laboratories, Singapore

gaetan.leurent@inria.fr, thomas.peyrin@ntu.edu.sg

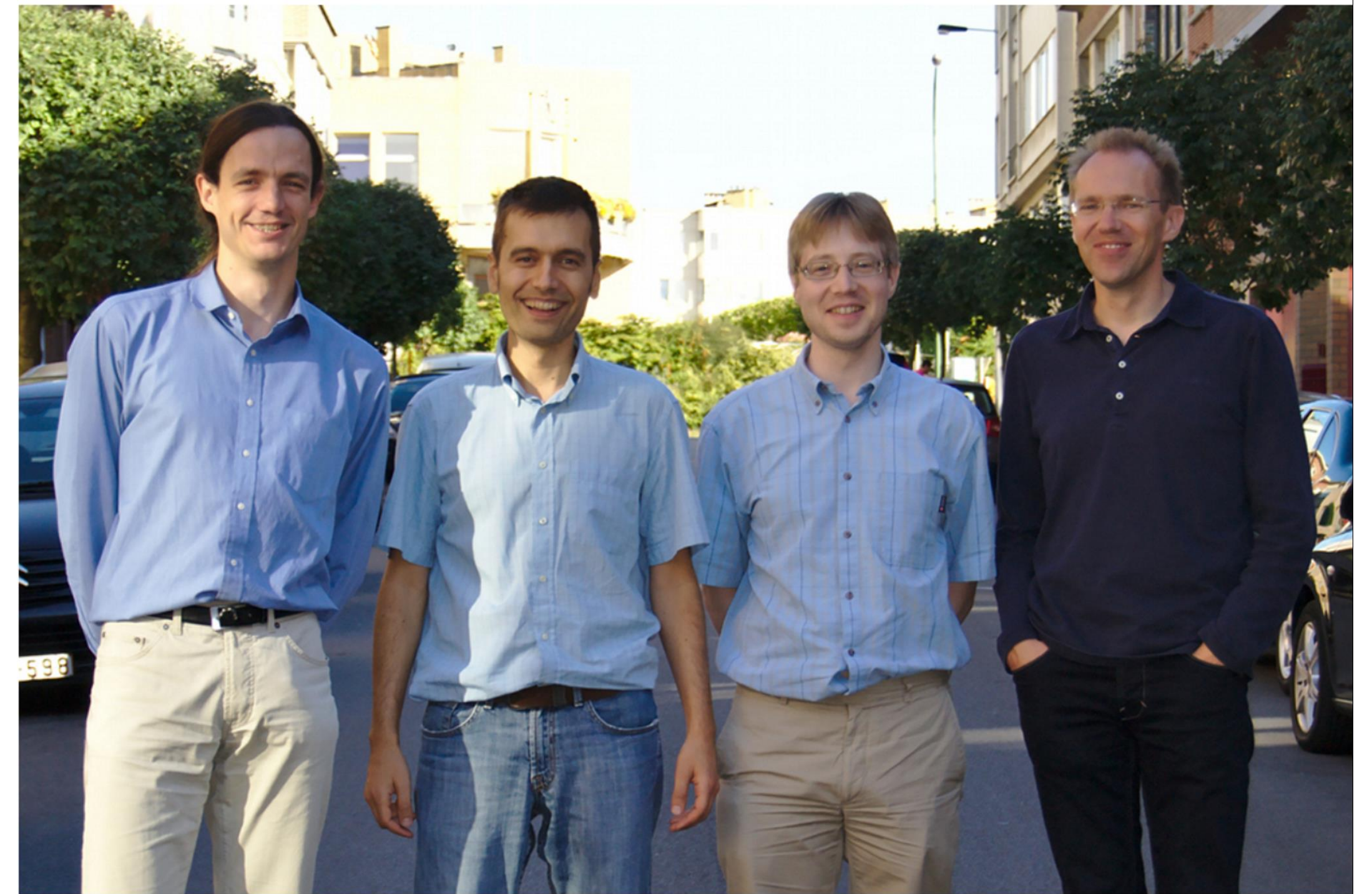
<https://sha-mbles.github.io/>

Source: <https://sha-mbles.github.io>

2. Sponge functions and Keccak

SHA-3: quest for a Merkle-Damgard alternative

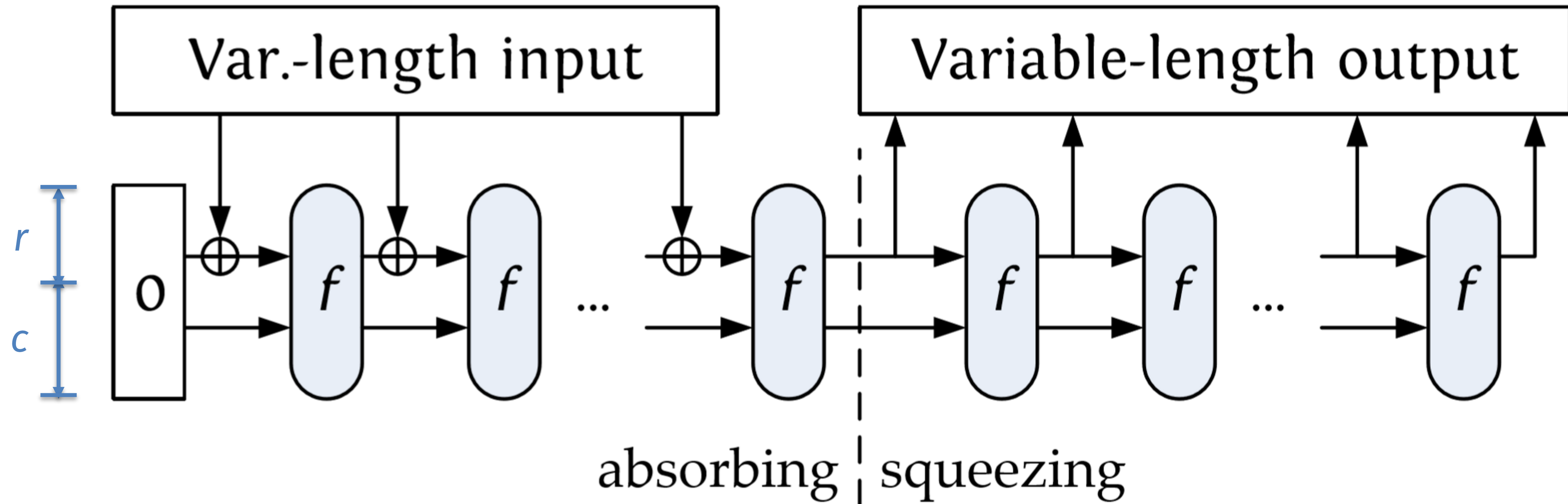
- 2004: Weakness found in Merkle-Damgard, eventually leading to SHA-1 break in 2017
- 2007: Call for submissions
- 2008: 64 submissions received
- 2009-12: Three workshops, one before each cutover: $64 \rightarrow 51 \rightarrow 14 \rightarrow 5 \rightarrow 1$
- Oct 2012: Keccak announced as winner, created by Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche
- Aug 2015: NIST publishes Federal Information Processing Standard (FIPS) 202 standardizing Keccak



Why NIST chose Keccak, in their words

1. “Offers acceptable performance in software, and *excellent performance in hardware.*”
2. “Has a *large security margin*, suggesting a good chance of surviving without a practical attack during its working lifetime.”
3. “A fundamentally new and different algorithm that is entirely *unrelated to the SHA-2 algorithms.*”

Sponge functions



Split state into two components

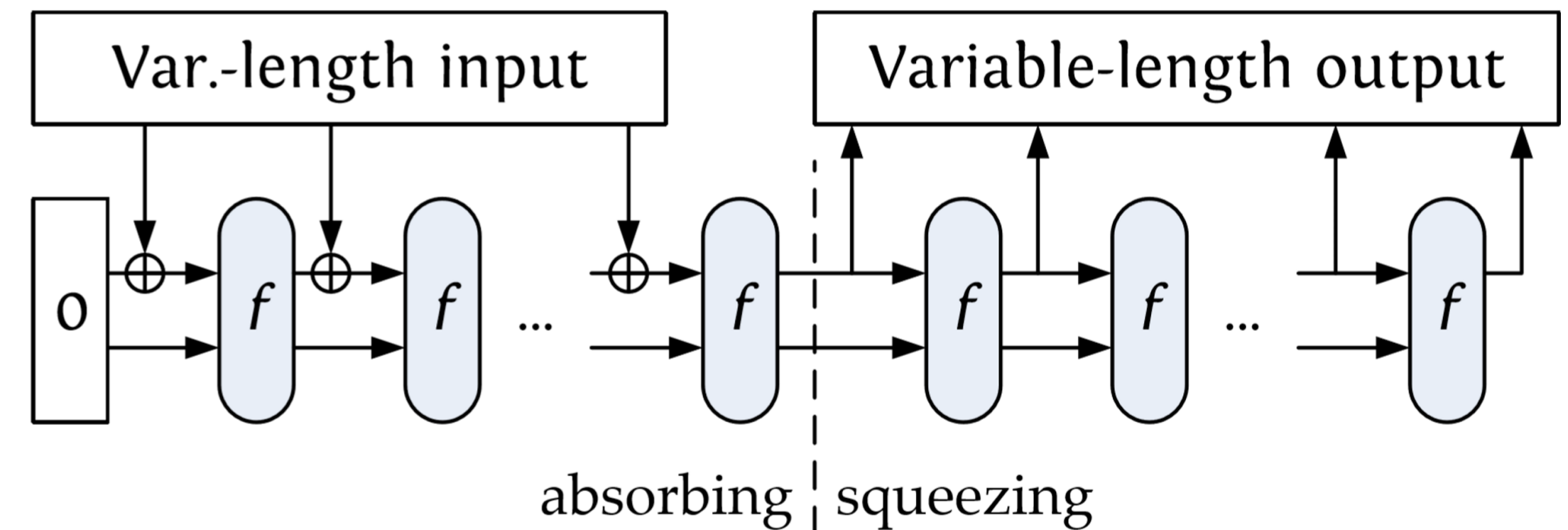
- r = rate, which influences speed
- c = capacity, which influences security

NIST standard

- One specific codebook f
- c = 224, 256, 384, or 512 (like SHA-2)

Benefits of sponge functions

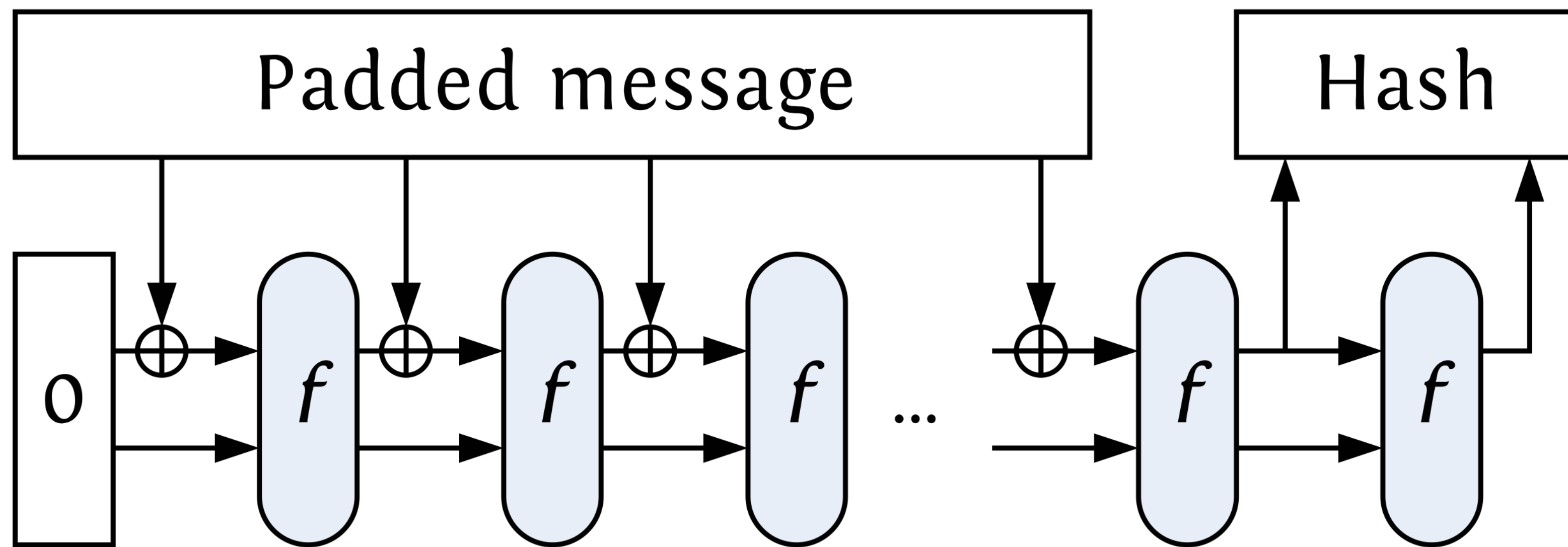
- *Simple design*: Need 1 codebook (subset of block cipher goal)
- *Tunable length*: Output can be bigger or smaller than input. Don't even need to decide upfront.
- *Security*: No concern about length-extension attacks. (Why?)
- *Utility*: Can produce a keyed function, encryption scheme, MAC directly from sponge functions without generic transformations



3. Building cryptosystems from sponge functions

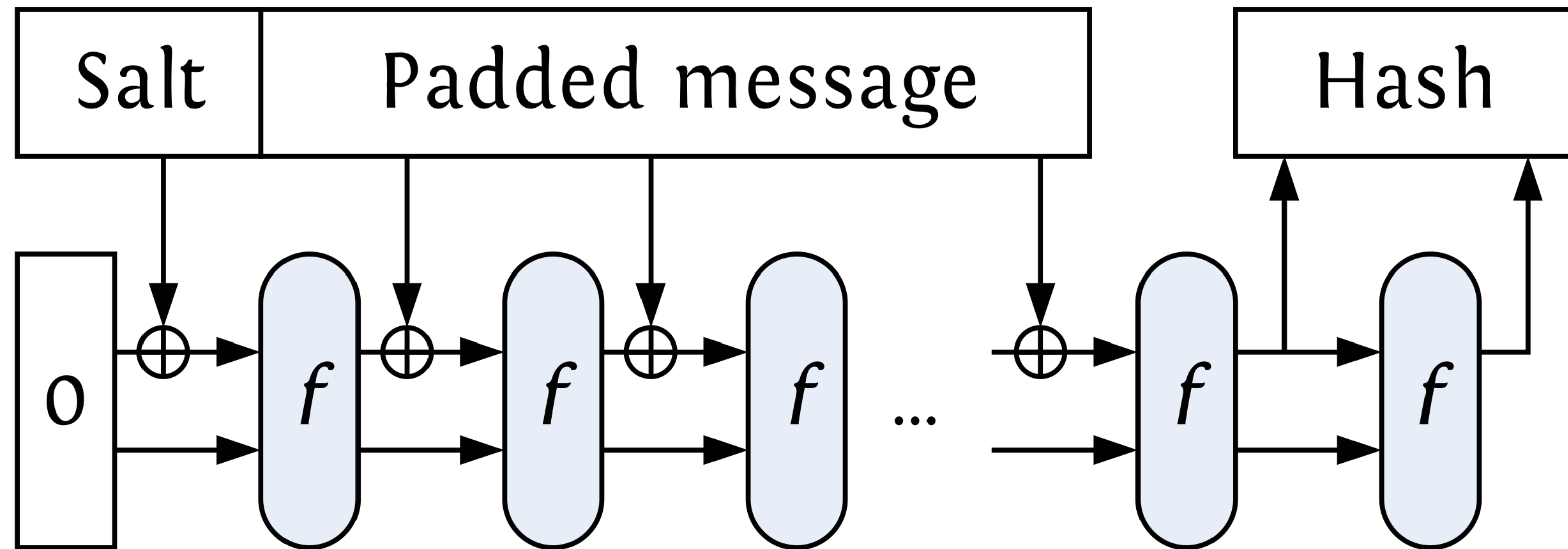
Source for slides: passwords12.atifi.uio.no/Joan_Daemen_Passwords12.pdf

Regular hashing



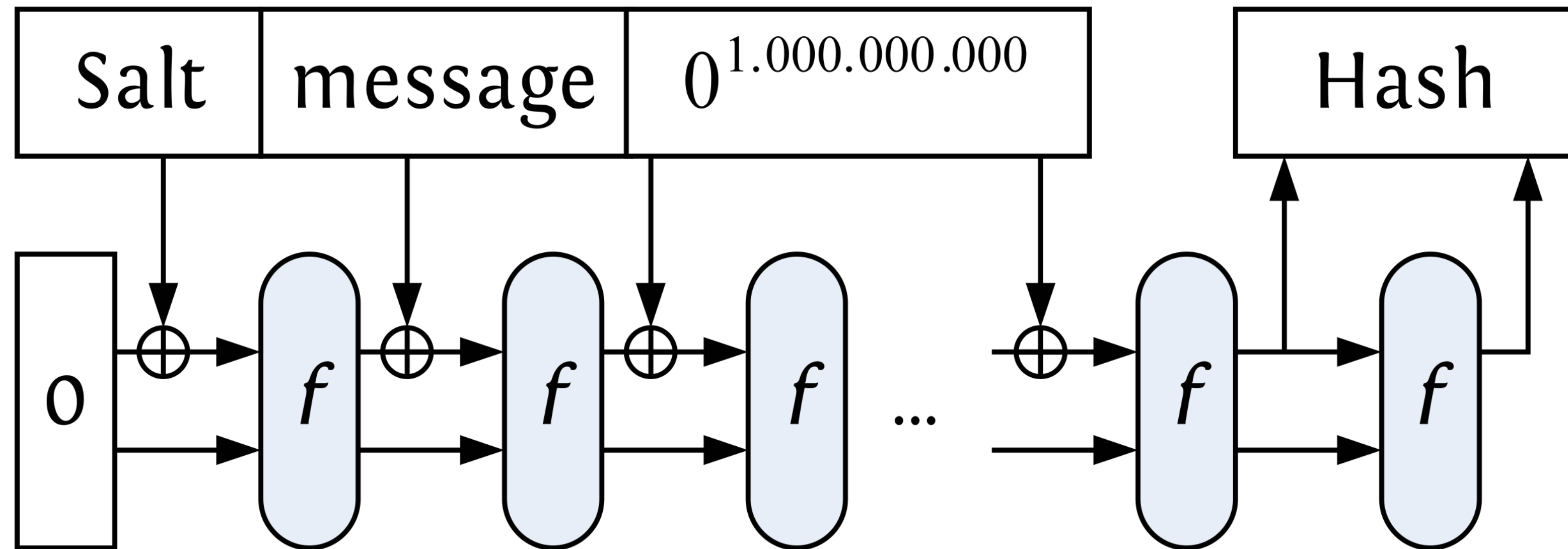
- Electronic signatures
- Data integrity (*shaXsum ...*)
- Data identifier (*Git, online anti-virus, peer-2-peer ...*)

Salted hashing



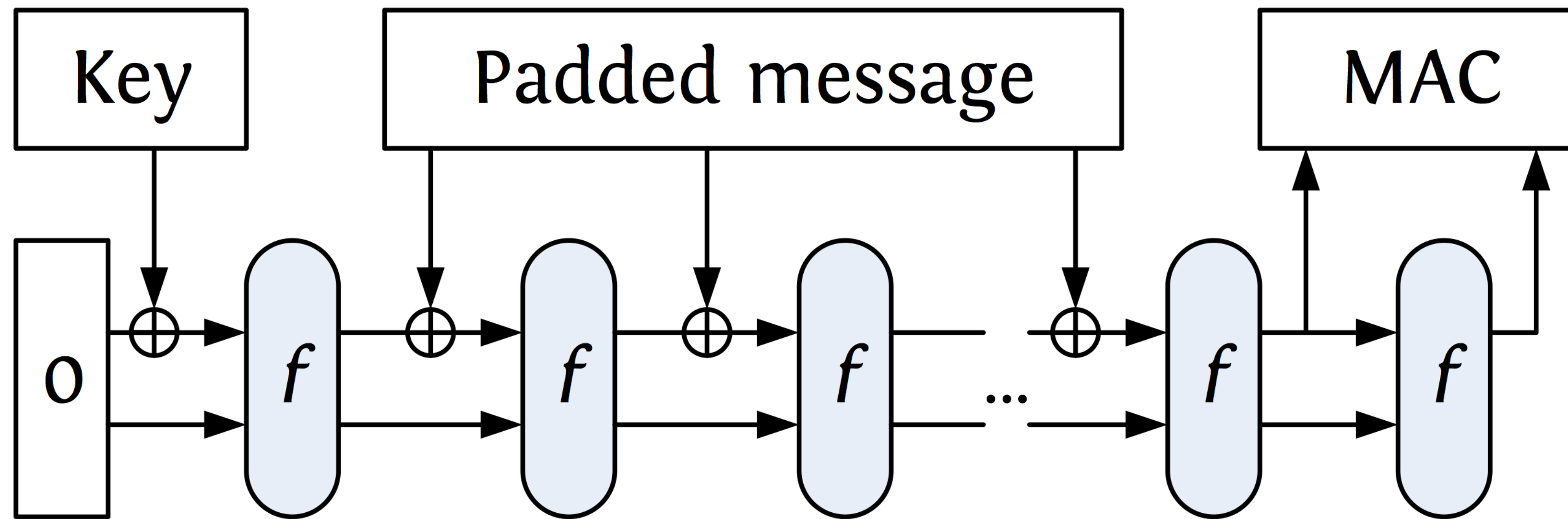
- Randomized hashing (RSASSA-PSS)
- Password storage and verification (*Kerberos*, /etc/shadow)

Salted hashing



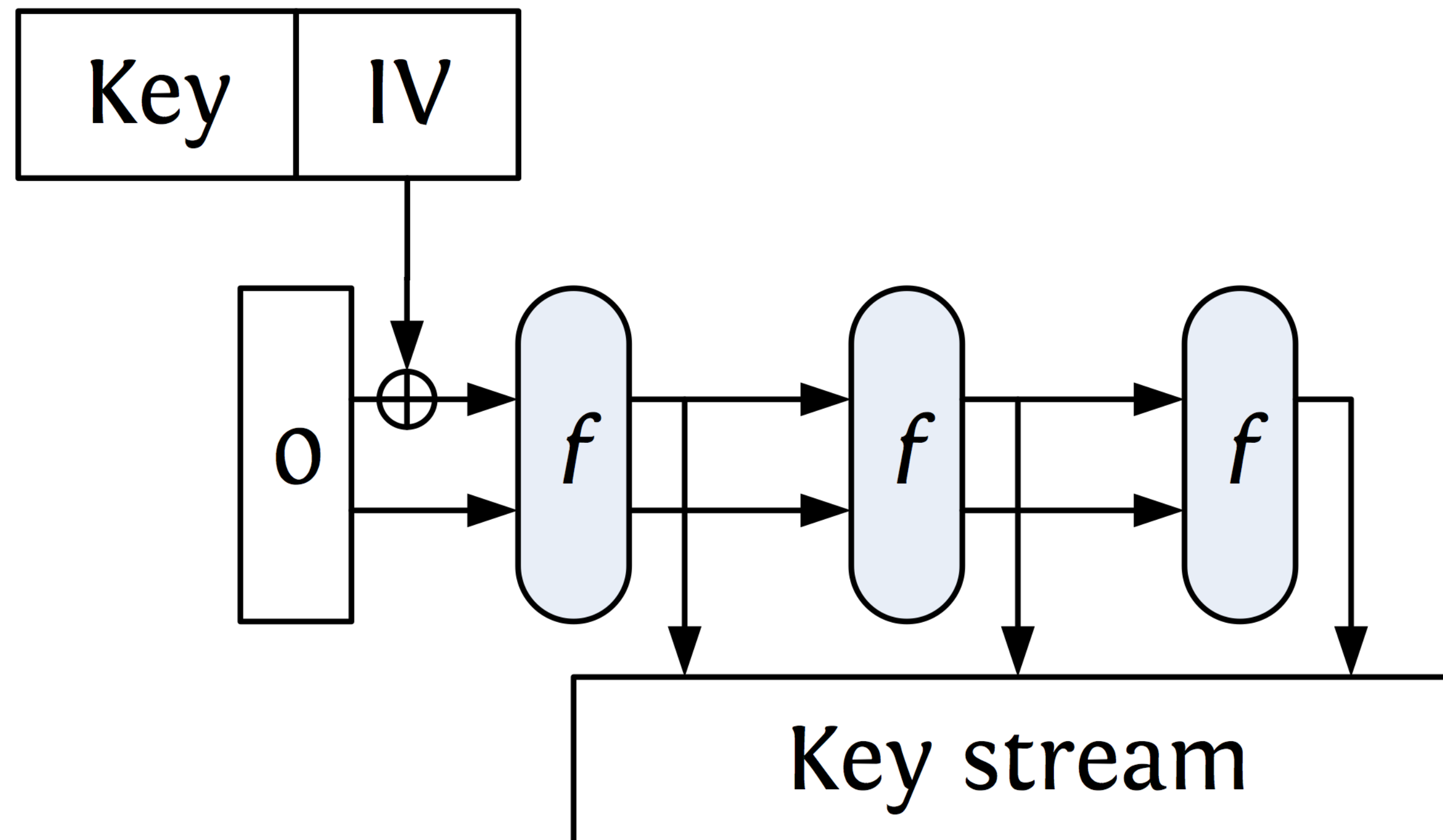
- Randomized hashing (RSASSA-PSS)
- Password storage and verification (*Kerberos*, /etc/shadow)
 - ...Can be as **slow** as you like it!

Message authentication codes



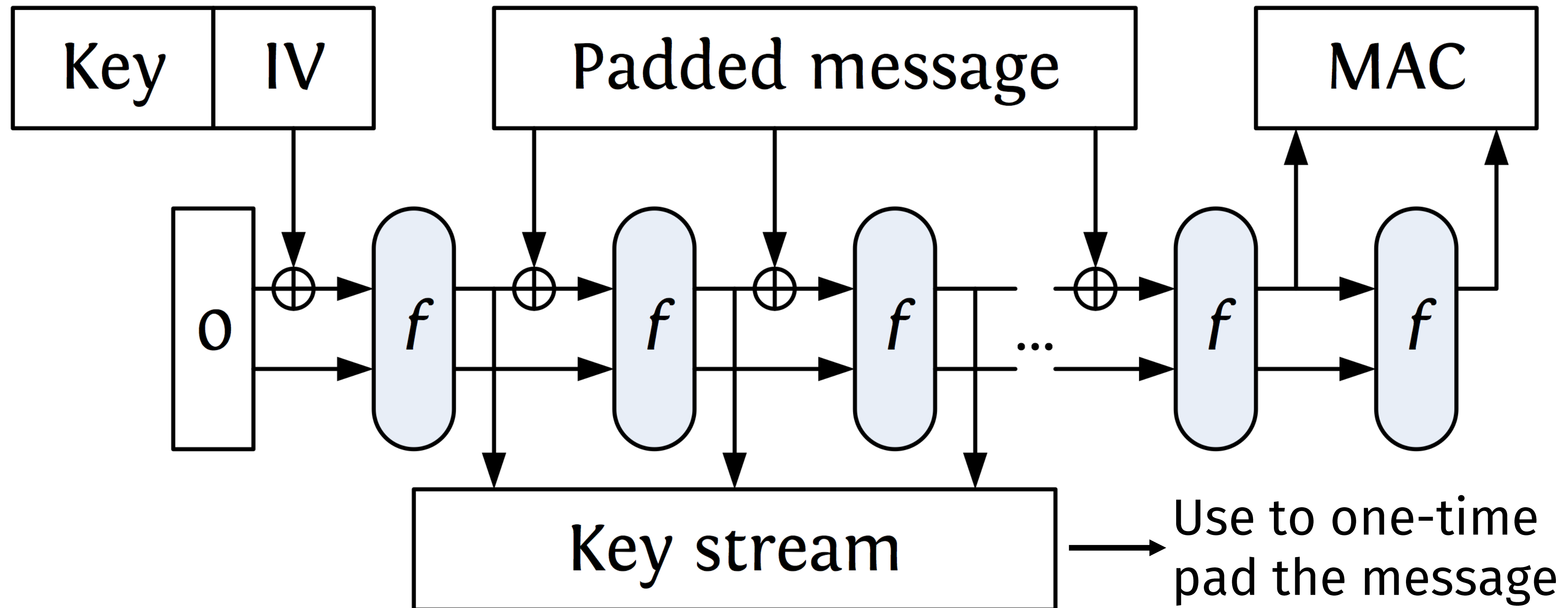
- As a message authentication code
- Simpler than HMAC [FIPS 198]
 - Required for SHA-1, SHA-2 due to length extension property
 - No longer needed for sponge

Stream encryption



- As a stream cipher
 - Long output stream per IV: similar to OFB mode
 - Short output stream per IV: similar to counter mode

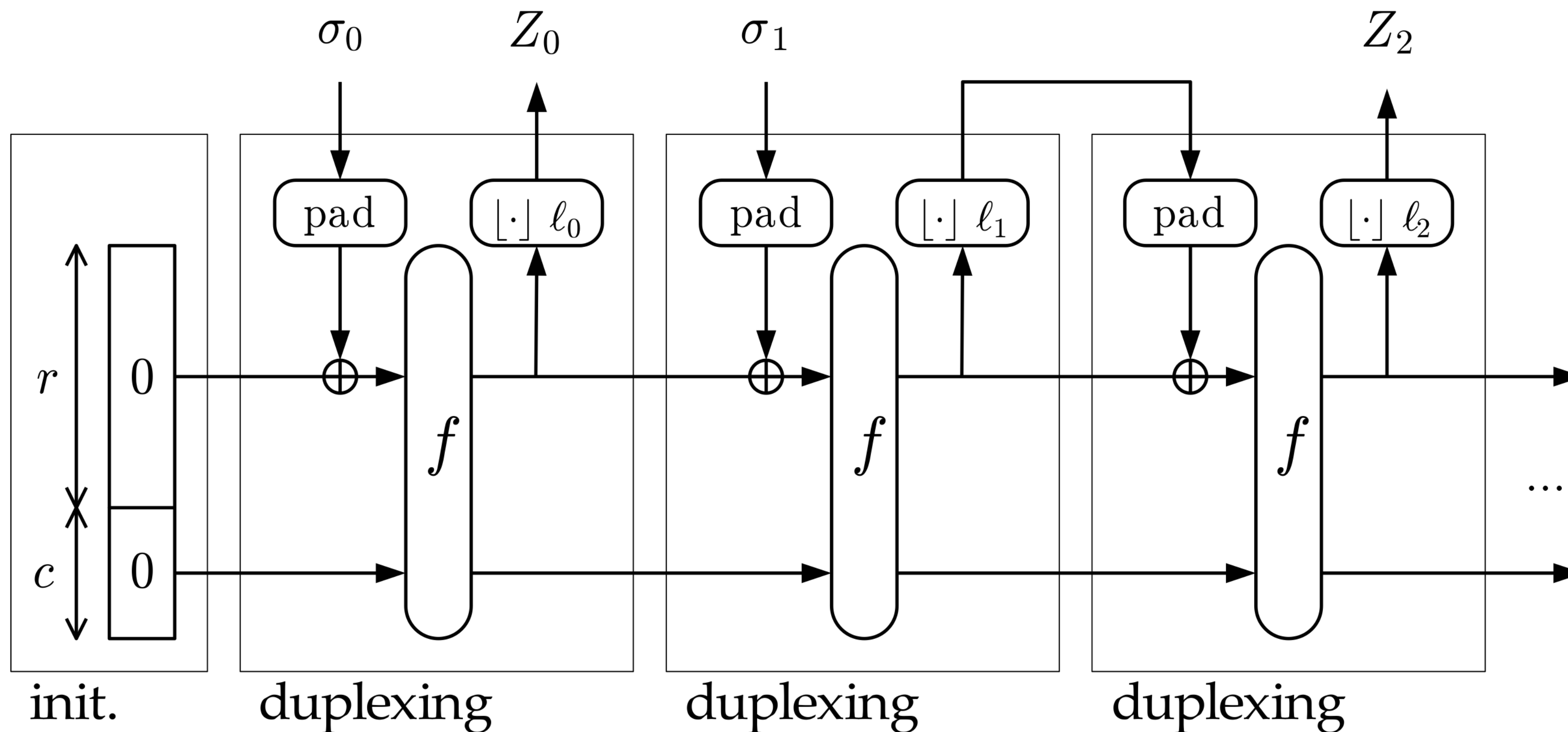
Single pass authenticated encryption



- Authentication and encryption in a **single** pass!
- Secure messaging (*SSL/TLS, SSH, IPSEC* ...)

Reseedable pseudorandom sequence generator

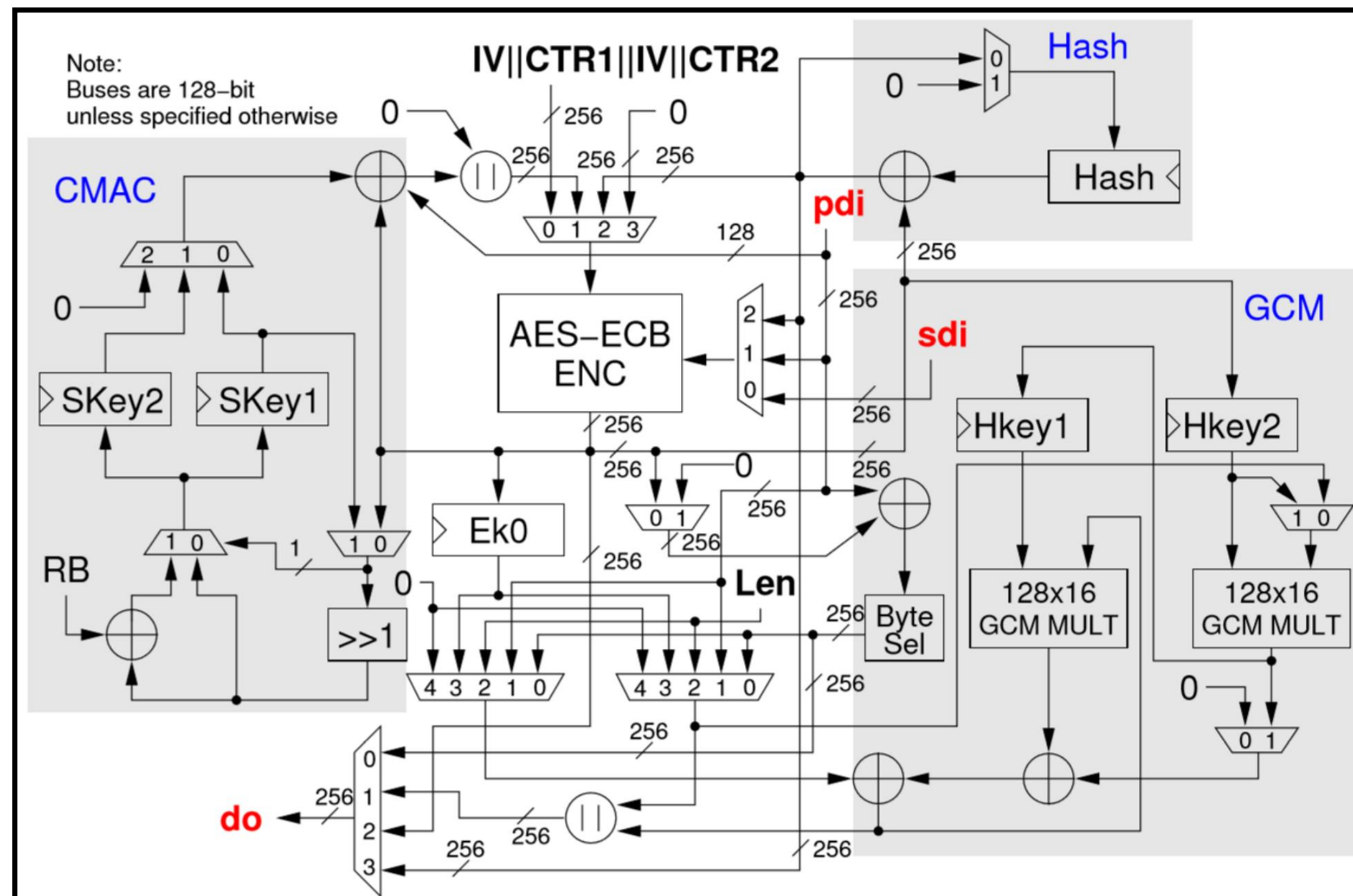
- Defined in [Keccak Team, CHES 2010] and [Keccak Team, SAC 2011]
- Support for forward secrecy by *forgetting* in duplex:



Comparing crypto implementations in hardware

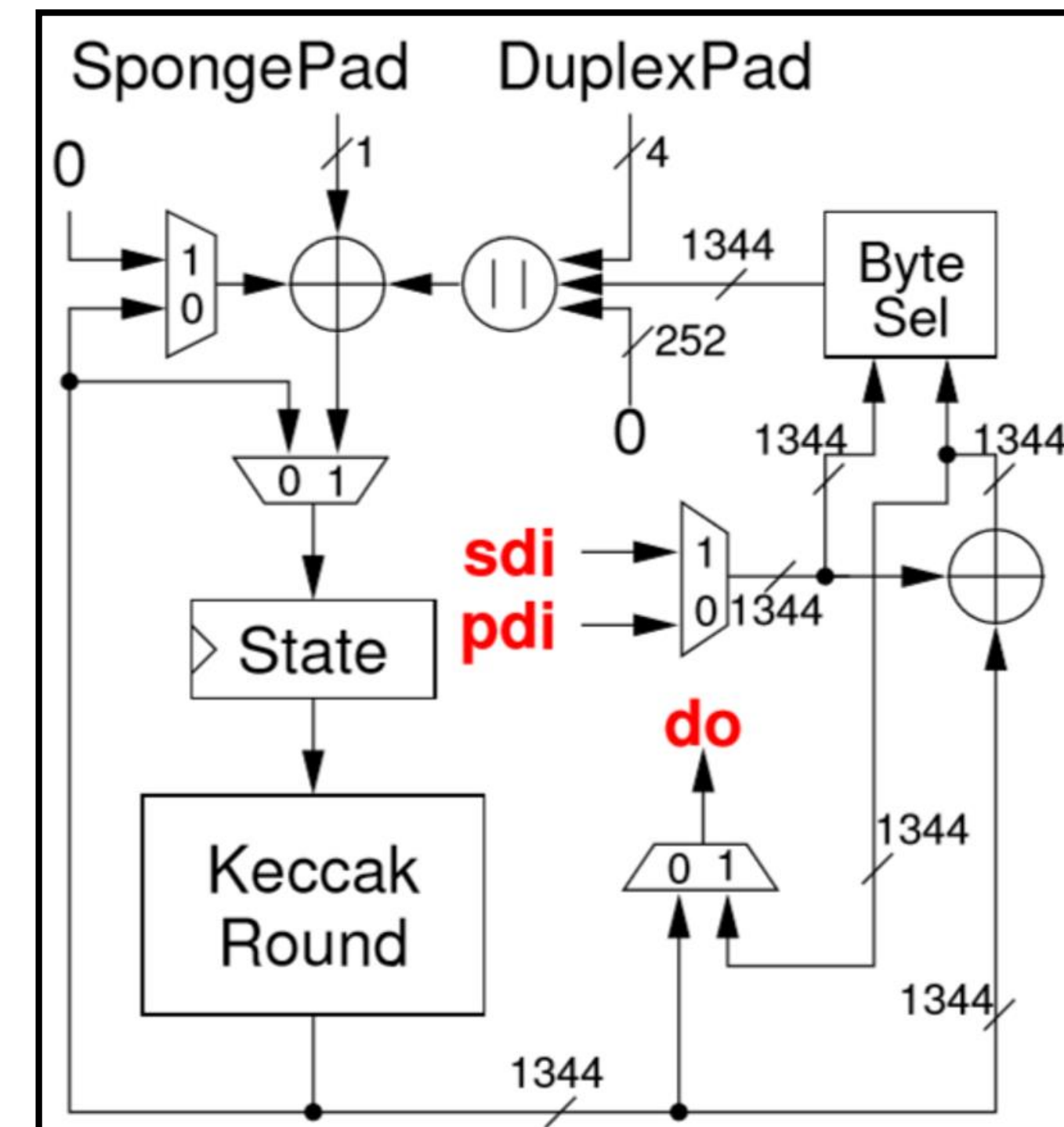
AES based

- Rijndael itself: small space
- Support for all modes: large space



SHA-3 based

- Keccak itself: large space
- Support for all modes: small space



4. The Keccak-f permutation

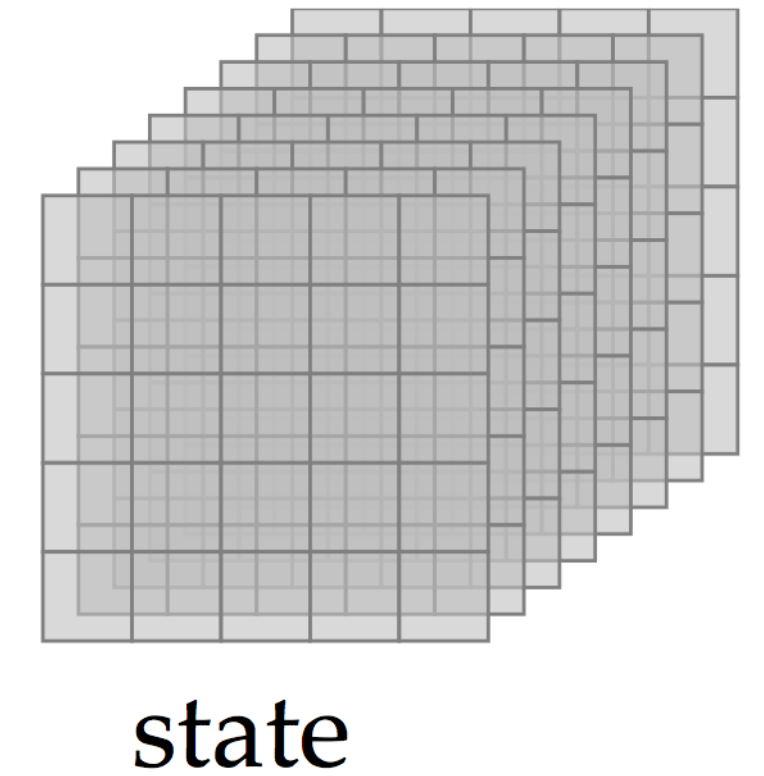
Designing the permutation KECCAK- f

Our mission

To design a permutation called KECCAK- f that cannot be distinguished from a random permutation.

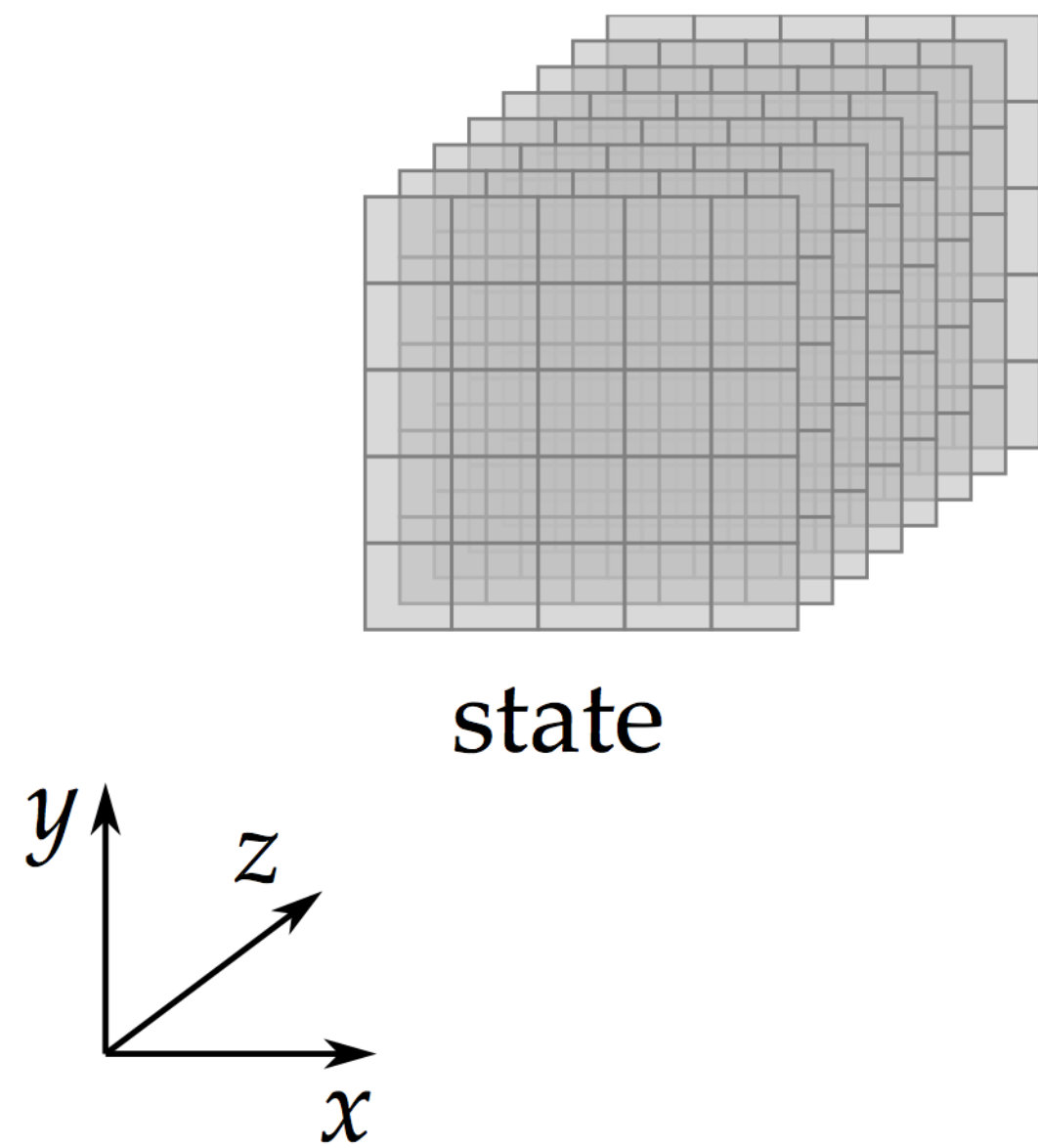
- Like a block cipher
 - sequence of identical rounds
 - round function that is nonlinear and has good diffusion
- ...but not quite
 - no need for key schedule
 - round constants instead of round keys
 - inverse permutation need not be efficient

- Instantiation of a *sponge function*
- the **permutation** KECCAK- f
 - 7 permutations: $b \in \{25, 50, 100, 200, 400, 800, 1600\}$
- Security-speed trade-offs using the same permutation, e.g.,
 - SHA-3 instance: $r = 1088$ and $c = 512$
 - permutation width: 1600
 - security strength 256: post-quantum sufficient
 - Lightweight instance: $r = 40$ and $c = 160$
 - permutation width: 200
 - security strength 80: same as SHA-1



KECCAK- f : the permutations in KECCAK

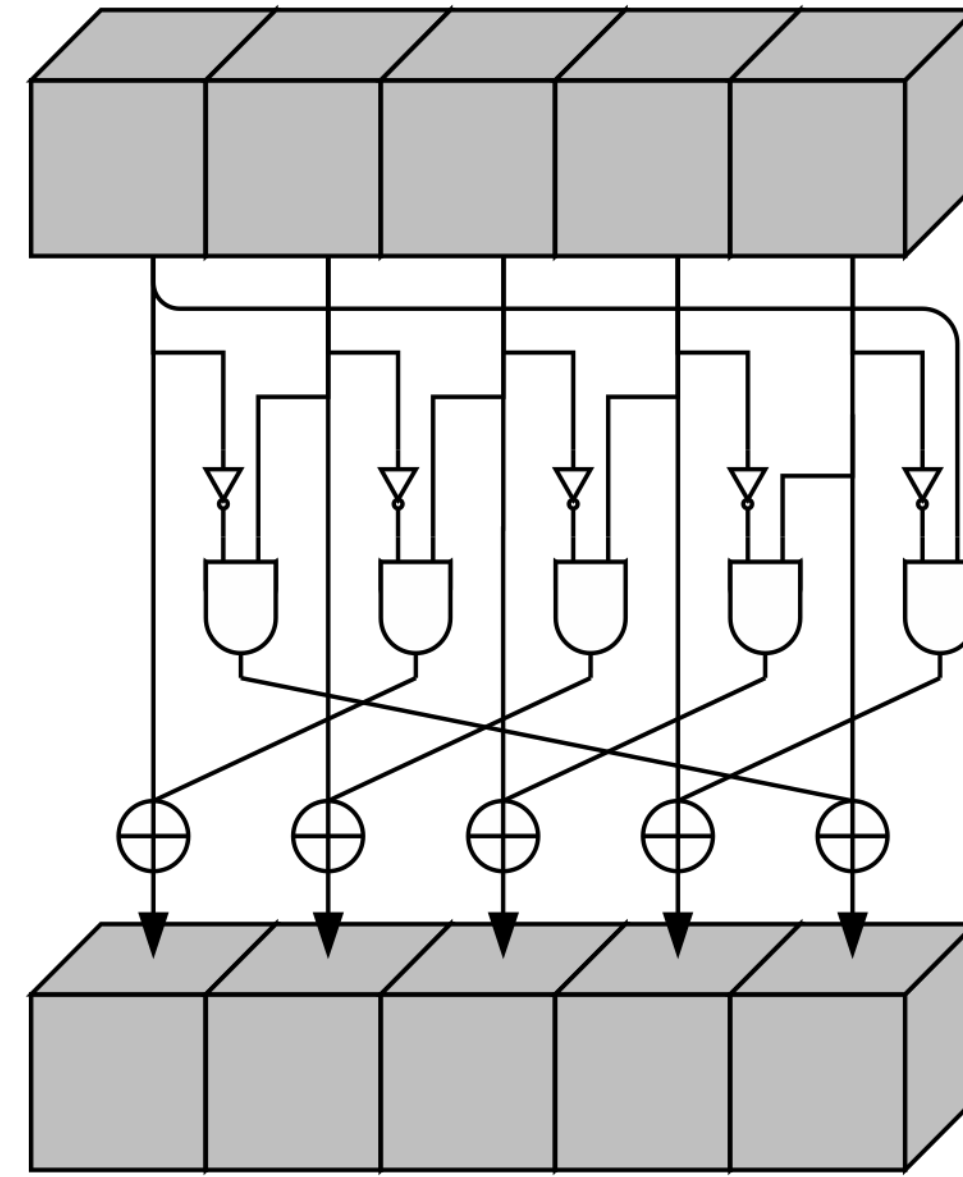
Operates on 3D state:



- (5×5) -bit **slices**
- 2^ℓ -bit **lanes**

- Round function R with 5 steps:
 - θ : mixing layer
 - ρ : bit transposition
 - π : bit transposition
 - χ : non-linear layer
 - ι : round constants
- # rounds: $12 + 2\ell$ for $b = 2^\ell 25$
 - 12 rounds in KECCAK- $f[25]$
 - 24 rounds in KECCAK- $f[1600]$

χ , the nonlinear mapping in Keccak-f



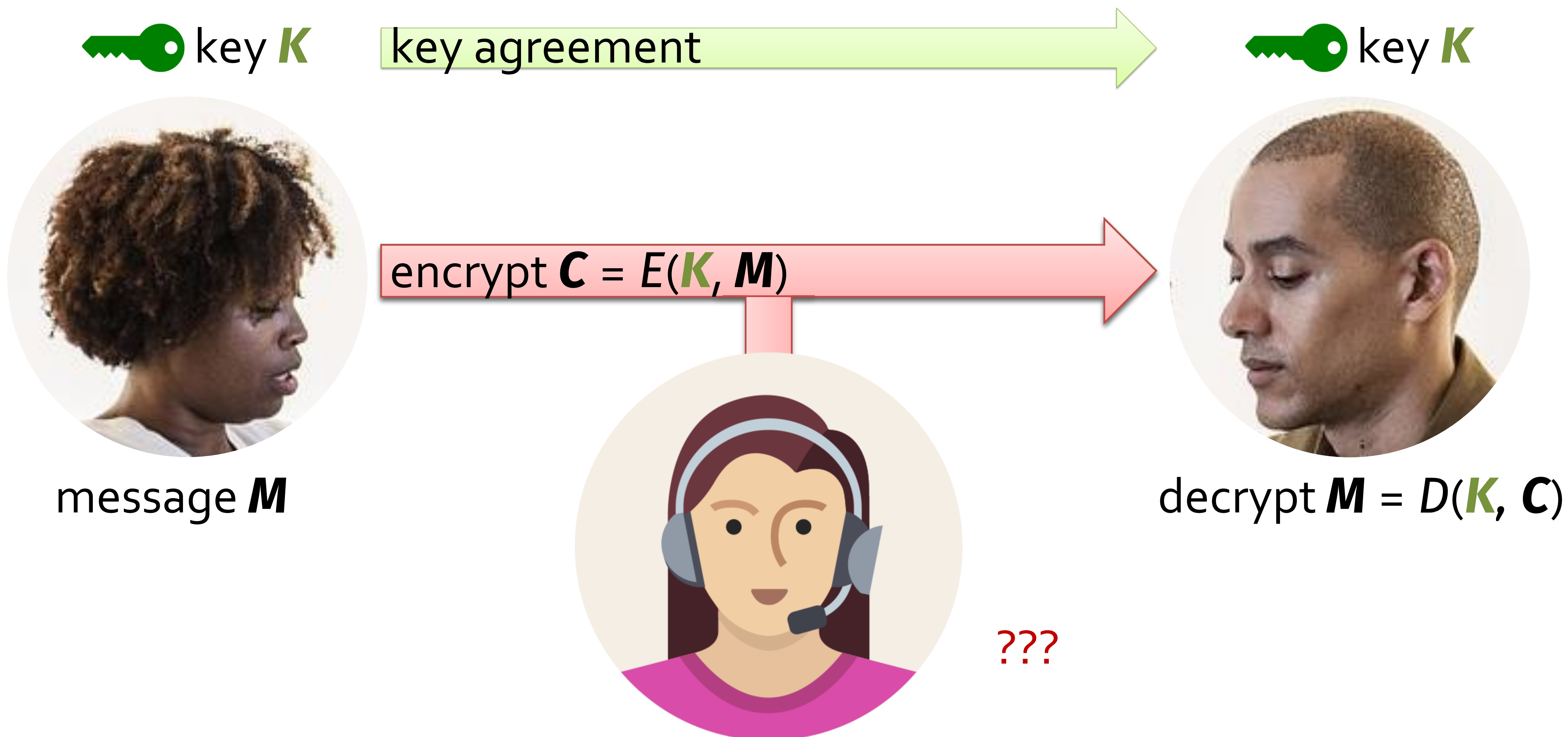
- “Flip bit if neighbors exhibit 01 pattern”
- Operates independently and in parallel on 5-bit rows
- Algebraic degree 2, inverse has degree 3
- LC/DC propagation properties easy to describe and analyze

Bounds for differential trails in KECCAK- f [1600]

Rounds	Lower bound	Best known
1	2	2
2	8	8
3	32 [KECCAK team]	32 [Duc et al.]
4		134 [KECCAK team]
5		510 [Naya-Plasencia et al.]
6	74 [KECCAK team]	1360 [KECCAK team]
24	296	???

5. Conclusion

Goal: Secure communication in presence of adversary

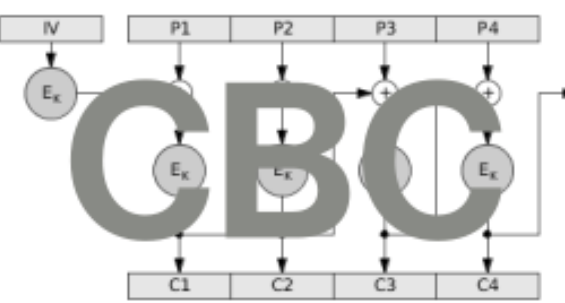
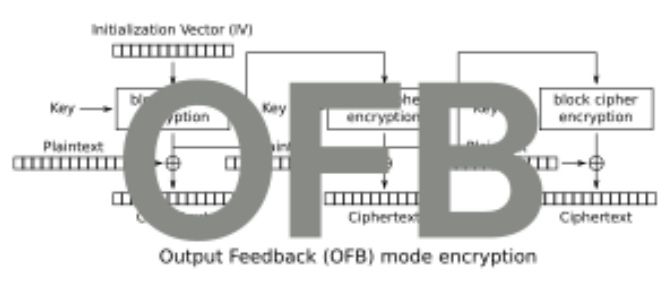
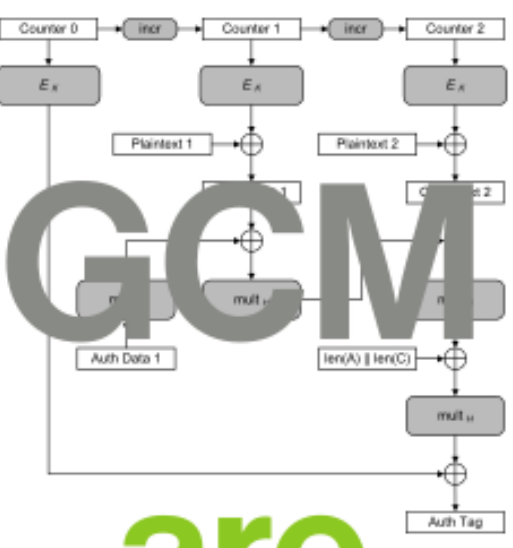
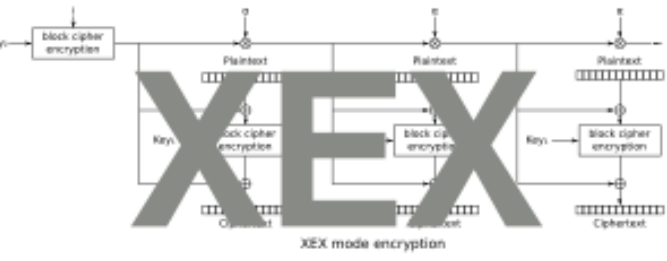
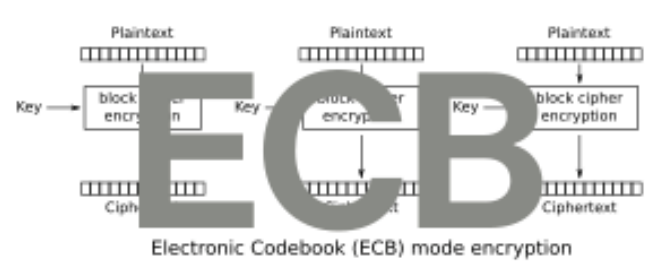
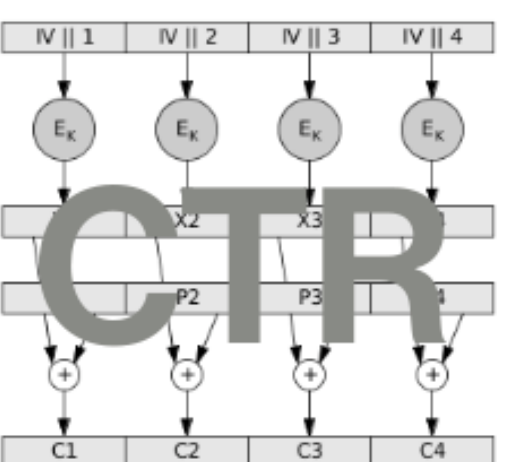
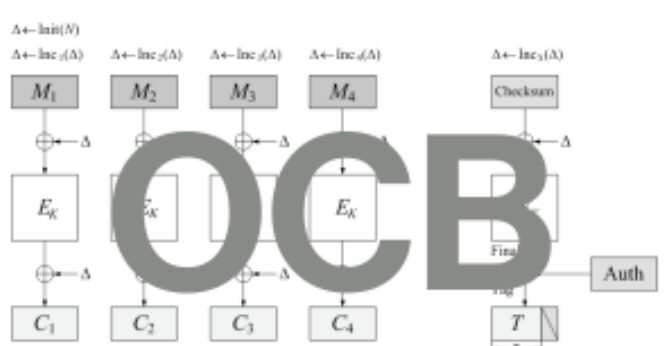
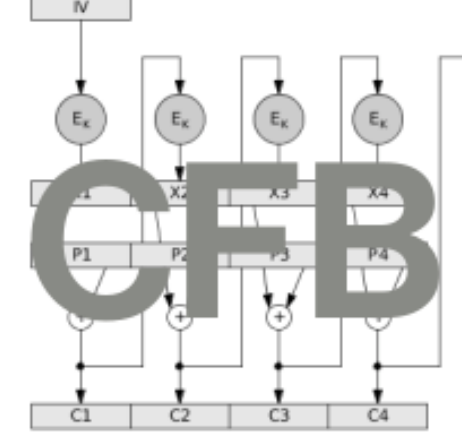
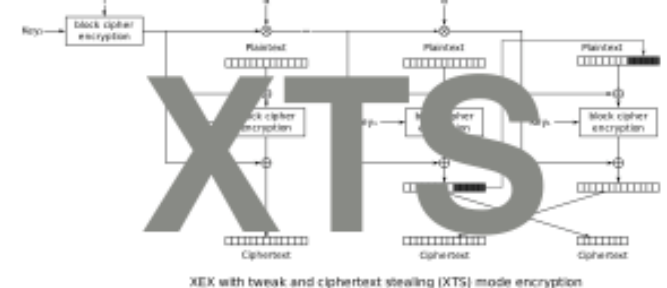


Modes of operation for authentication and/or encryption

Cryptographic doom principle

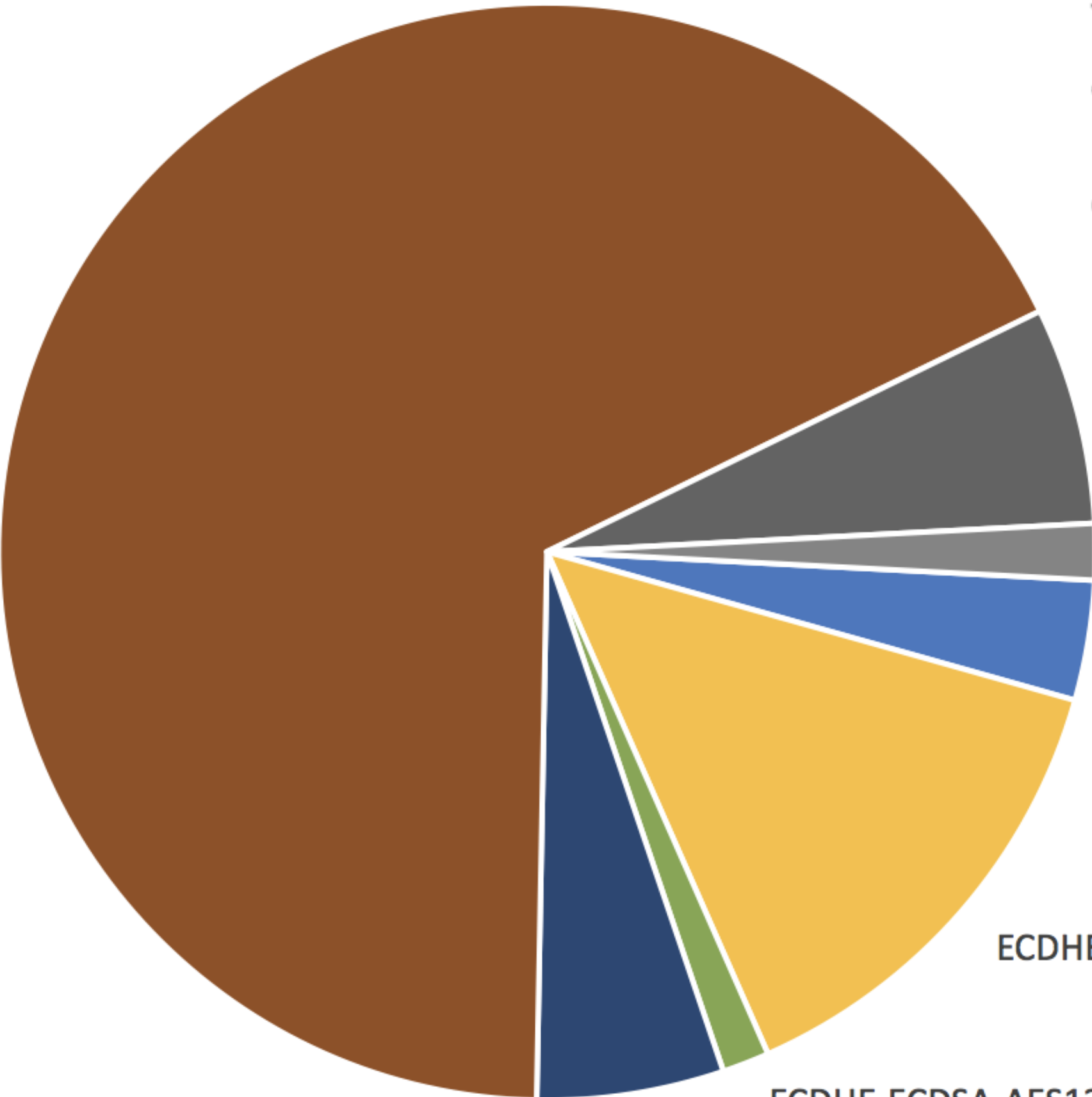
If you have to perform any crypto operation before verifying the MAC on a message you've received, it will somehow inevitably lead to doom!

– Moxie Marlinspike

 <p>CBC</p> <p>All</p>	 <p>OFB</p> <p>modes</p>	 <p>GCM</p> <p>are</p>
 <p>XEX</p> <p>beautiful</p>	 <p>ECB</p> <p>not you</p>	 <p>CTR</p> <p>and</p>
 <p>OCB</p> <p>deserve</p>	 <p>CFB</p> <p>to be</p>	 <p>XTS</p> <p>used</p>

Widespread use of good crypto: Auth Enc, SHA-2, ...

ECDHE-ECDSA-AES128-GCM-SHA256
67%



ECDHE-ECDSA-CHACHA20-POLY1305
6%

ECDHE-ECDSA-AES128-SHA
1%

ECDHE-RSA-AES128-GCM-SHA256
14%

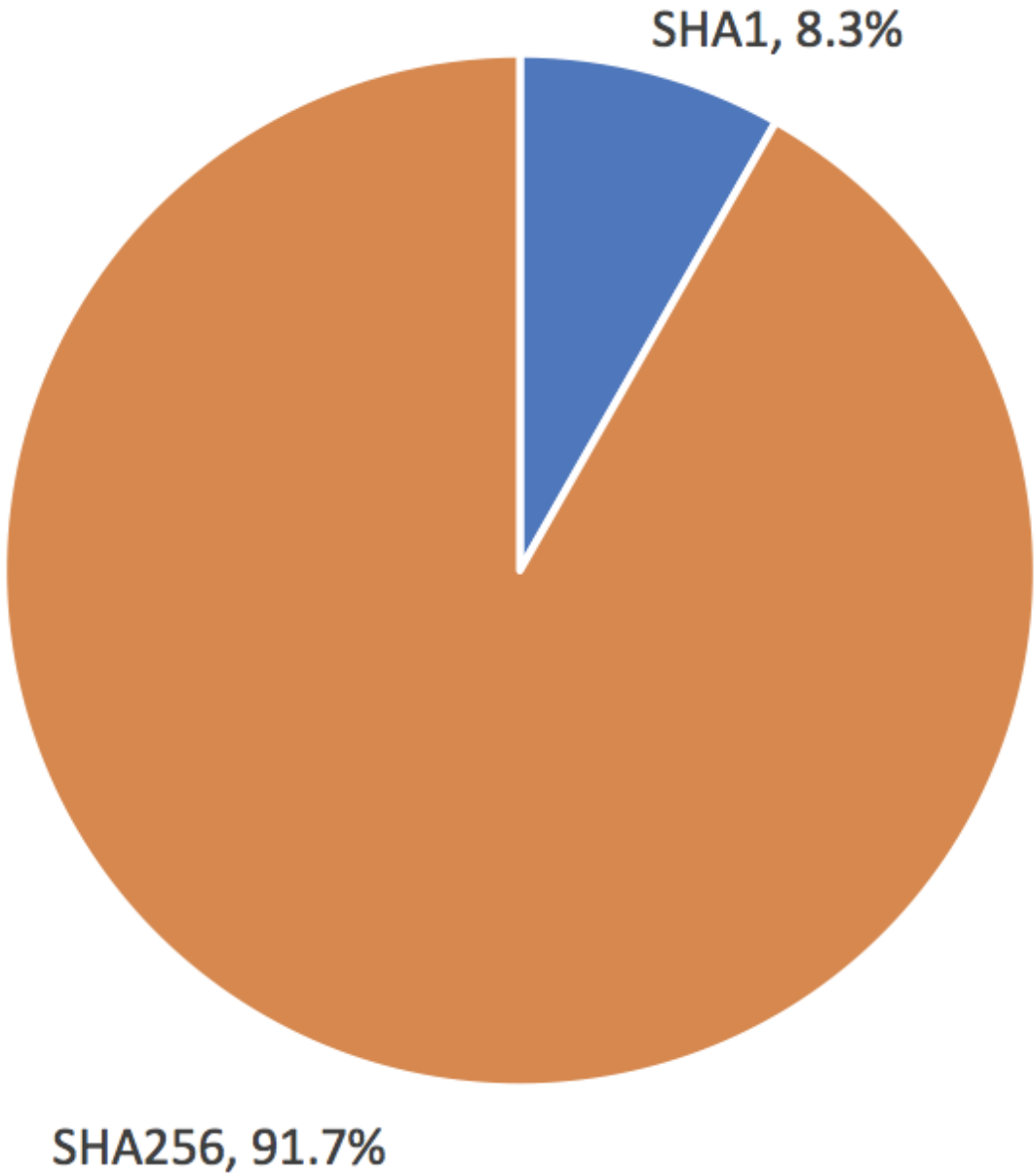
ECDHE-RSA-CHACHA20-POLY1305
4%

Other
2%

ECDHE-RSA-AES128-SHA
6%

Secure Connection

The connection to this site is encrypted and authenticated using a strong protocol (QUIC), a strong key exchange (ECDHE RSA with X25519), and a strong cipher (AES_128_GCM).

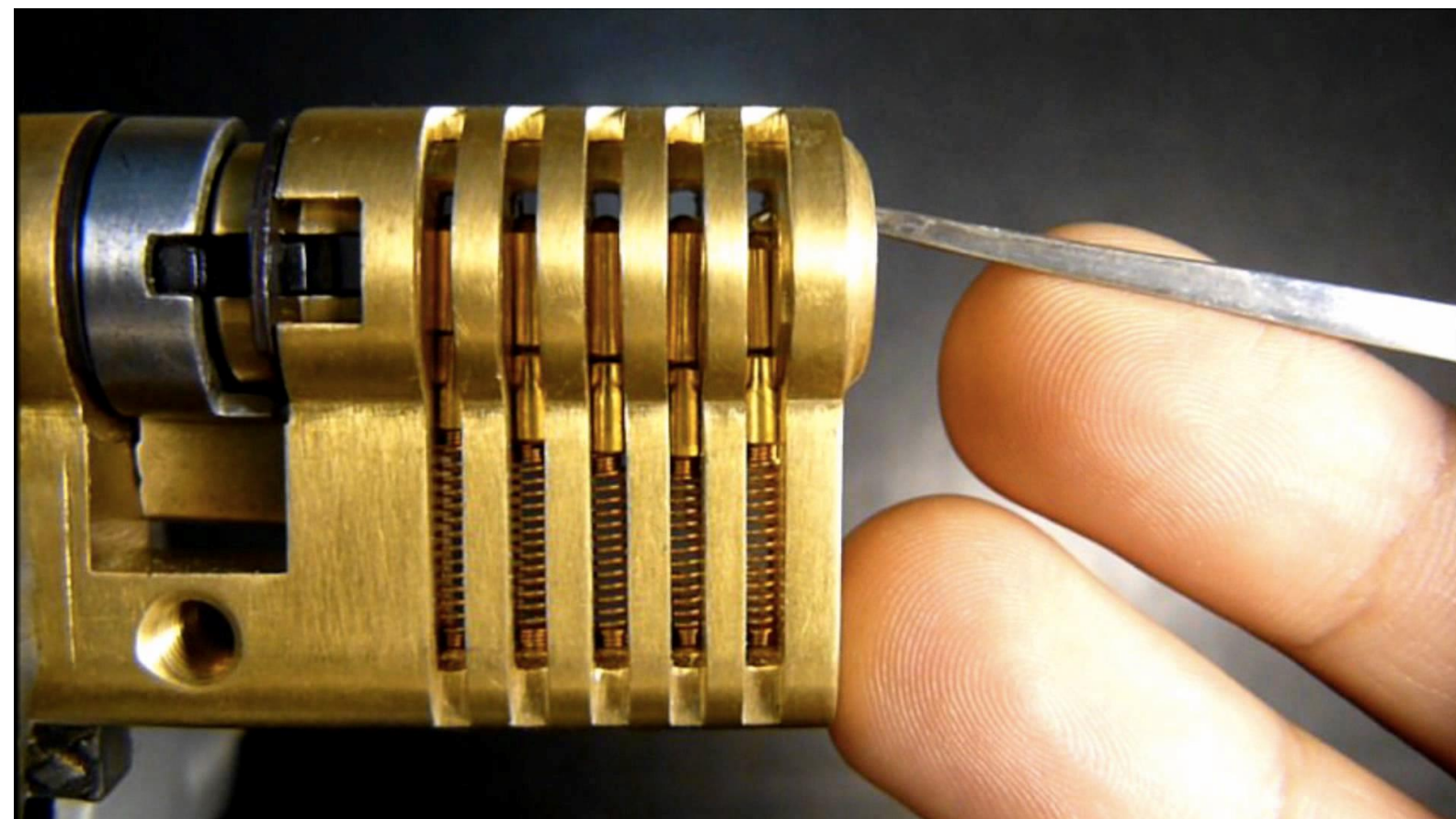


Divide-and-conquer \Rightarrow Side channels + cryptanalysis

Foot-Shooting Prevention Agreement

I, _____, promise that once
Your Name

I see how simple AES really is, I will
not implement it in production code
even though it would be really fun.

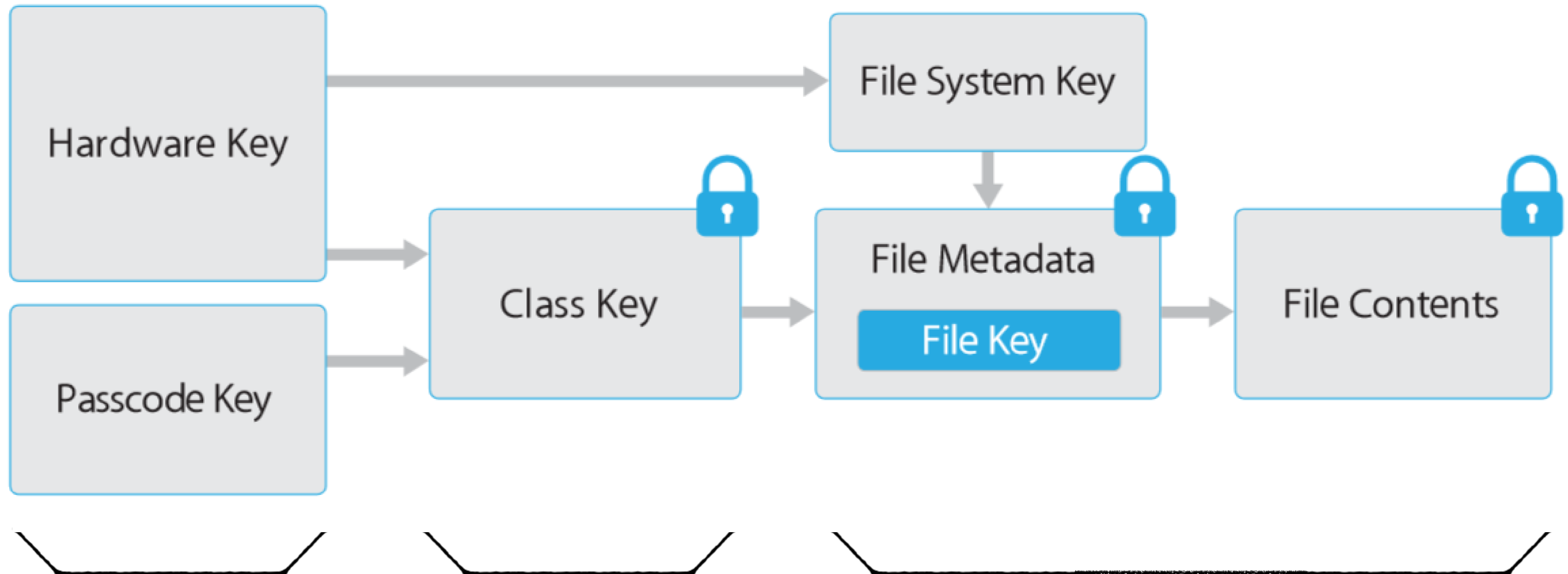


This agreement shall be in effect
until the undersigned creates a
meaningful interpretive dance that
compares and contrasts cache-based,
timing, and other side channel attacks
and their countermeasures.

X _____
Signature Date

Source:
moserware.com/2009/09/stick-figure-guide-to-advanced.html

Data at rest: derive keys from a password



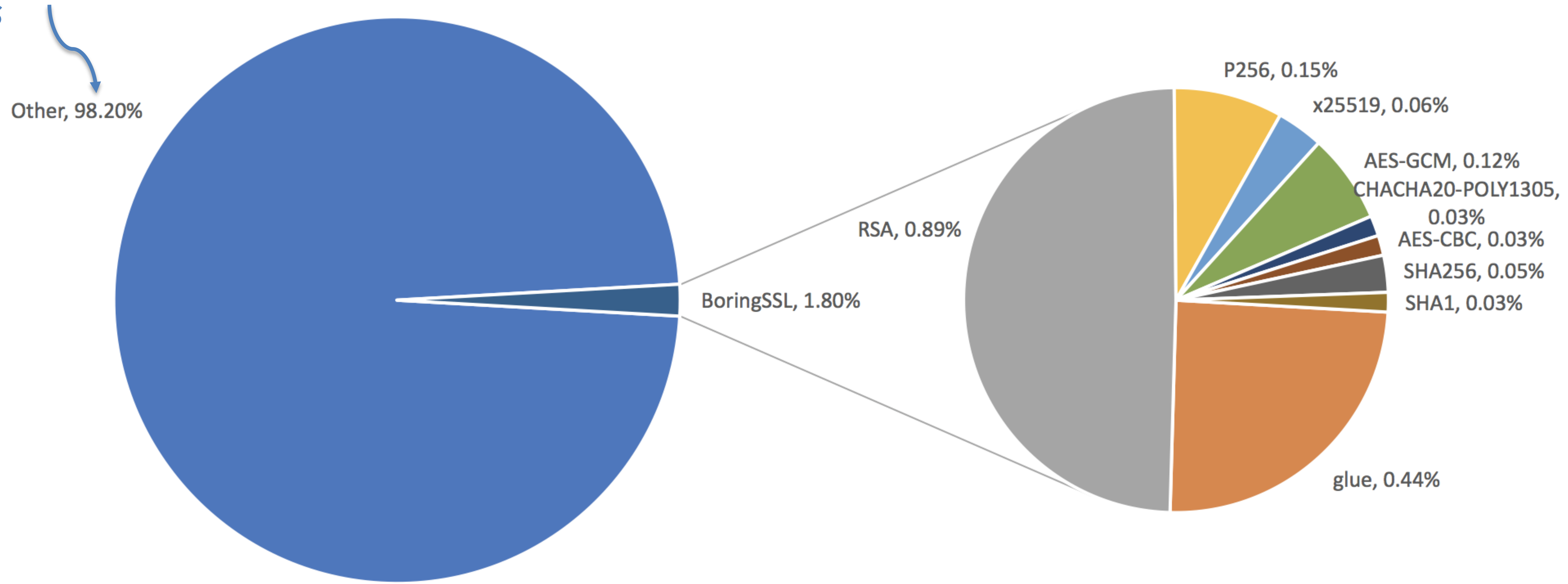
Master key generated from things that you know, are, and have

Derive keys that live only for limited time

Use these to wrap a unique key for each file

Data in transit: crypto in TLS is really fast!

*everything else
the server
does*



Key management \Rightarrow Access control

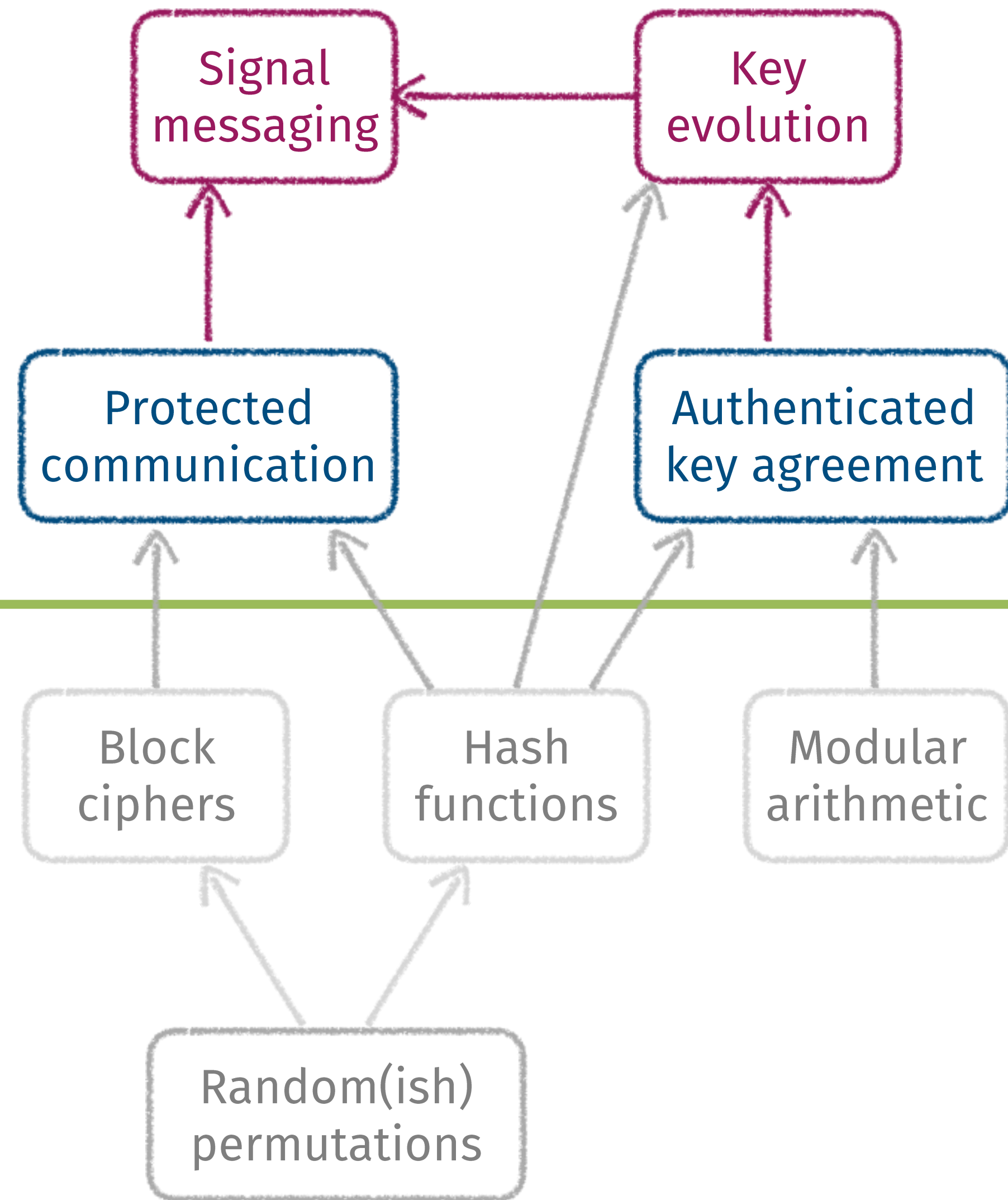


“This bar is pretty good, but you have to go stand in line for a ticket before they serve you.”

Source:
twitter.com/sweis/status/98227289194842112
0

Signal: Deniability, forward + backward secrecy

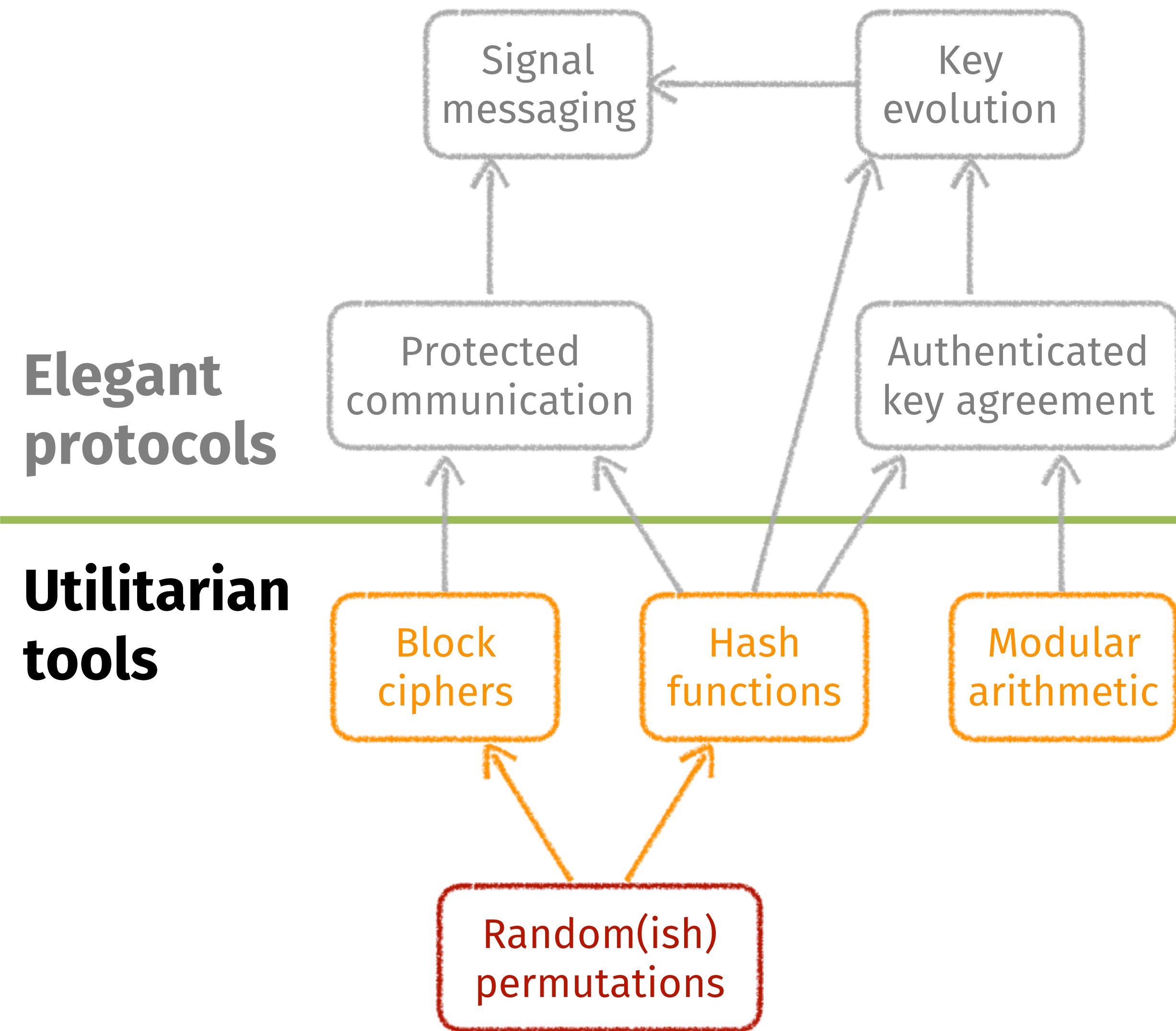
**Elegant
protocols**



**Utilitarian
tools**



Cryptanalysis informs design of lowest-layer tools



Cryptography *enables* data analysis *without* data sharing



Next week:
Cryptography and the law