Developing Effective Test Strategies for Cryptographic Algorithm Implementations

Sydney Pugh[1,3] **M S Raunak[1,2]** D. Richard Kuhn[2] Raghu Kacker[2]

1 Loyola University Maryland, Baltimore, MD USA
2 National Institute of Standards and Technology, Gaithersburg, MD USA
3 University of Pennsylvania, Philadelphia, PA USA

LWC Workshop
Nov 06, 2019

Testing Cryptographic Algorithms is Difficult

Issues

- Lacks test-oracle
  - Developing a test oracle is very costly, often infeasible
- Implementation of cryptographic algorithms are inherently complex
  - Dense with bit manipulations and condition predicates
- Traditional test strategies are generally ineffective
  - Statement and branch coverage

Approach

- Systematically design tests suitable for cryptographic algorithms

Recent Development in Crypto Algorithm Testing

Strong evidence of the application of metamorphic testing to cryptographic algorithm implementations

Appears in IEEE Transactions on Reliability, vol. 67, no. 3, Sept. 2018

Finding Bugs in Cryptographic Hash Function Implementations

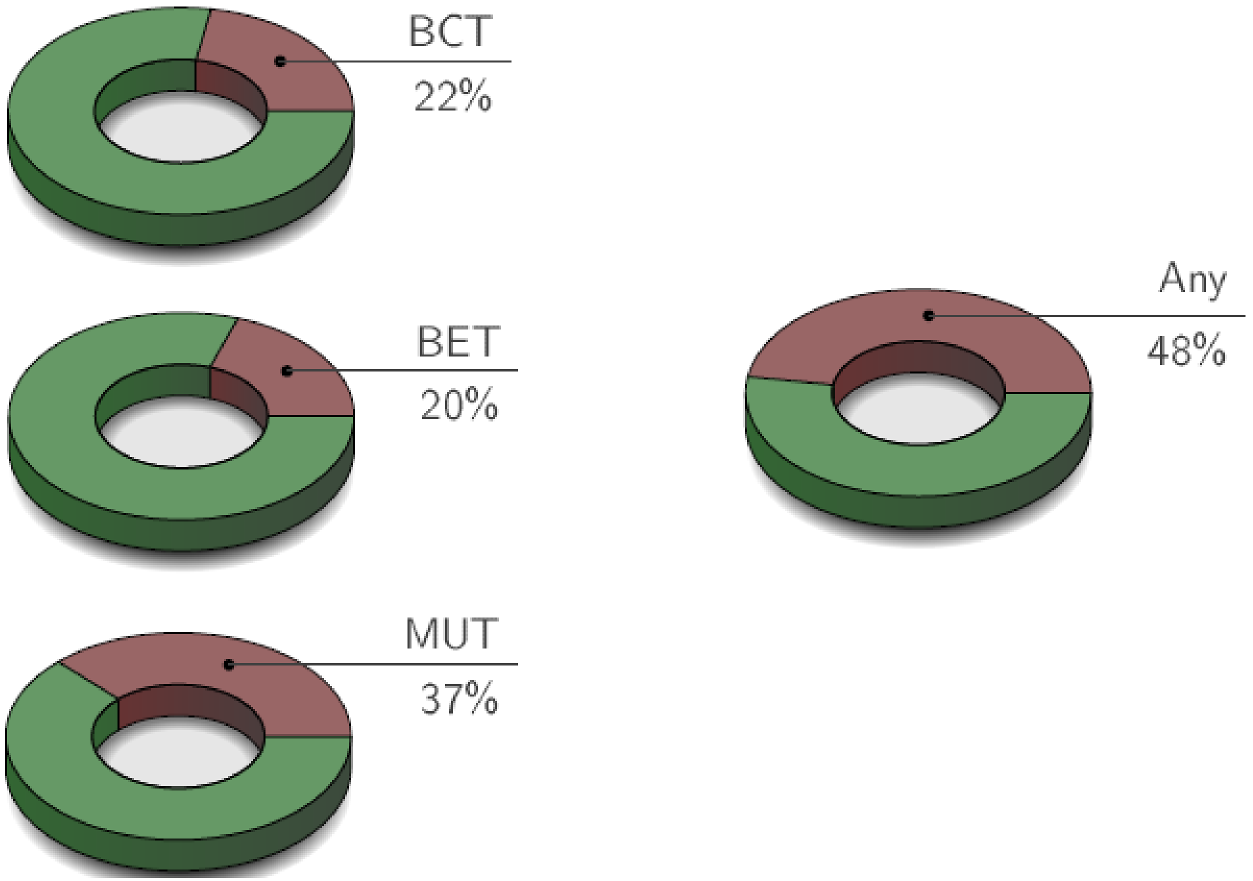
Nicky Mouha, Mohammad S Raunak, D. Richard Kuhn, and Raghu Kacker

2019 IEEE/ACM 4th International Workshop on Metamorphic Testing (MET)

Systematic Testing of Post-Quantum Cryptographic Implementations Using Metamorphic Testing

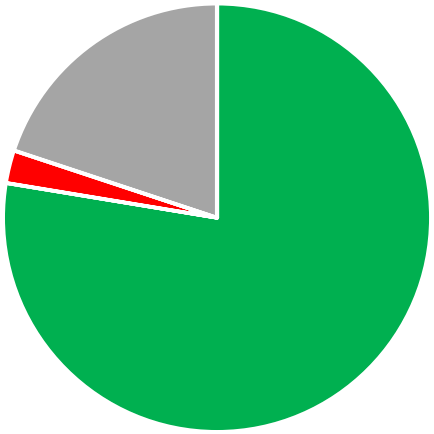
Sydney Pugh, M S Raunak, D. Richard Kuhn, and Raghu Kacker

Previous Testing Success – SHA-3



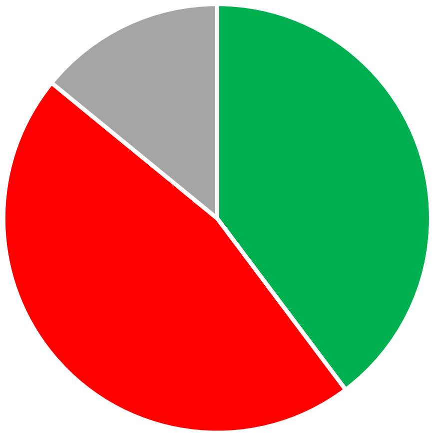
Previous Testing Success – PQC

Bit Contribution



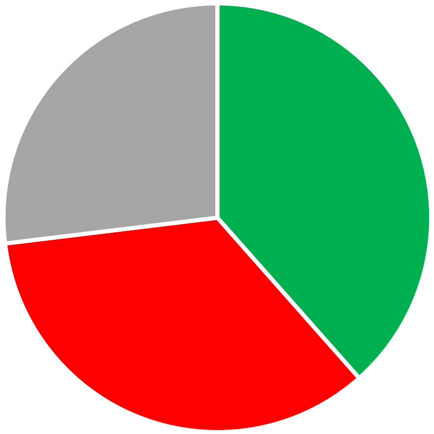
2.56% Failed or Error

Bit Exclusion



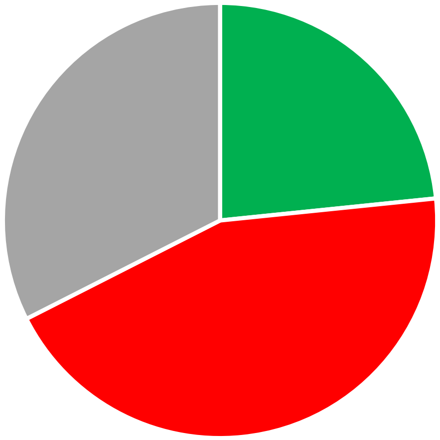
46.15% Failed or Error

Bit Verify



34.62% Failed or Error

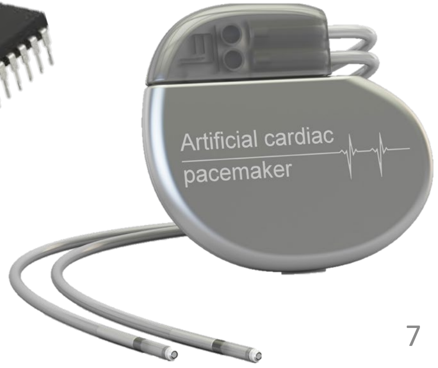
Encrypt Decrypt Check



40.96% Failed or Error

Need for LWC: Growth of Small Computing Devices

- Radio Frequency IDentification (RFID) Tags
- Smart Cards
- Microcontrollers
- Embedded Systems
- Sensor Networks
- IoT Devices



Lightweight Cryptography (LWC)

Develop a new standard for Authenticated Encryption with Associated Data (AEAD) and hash functions designed for resource-constrained devices

Timeline

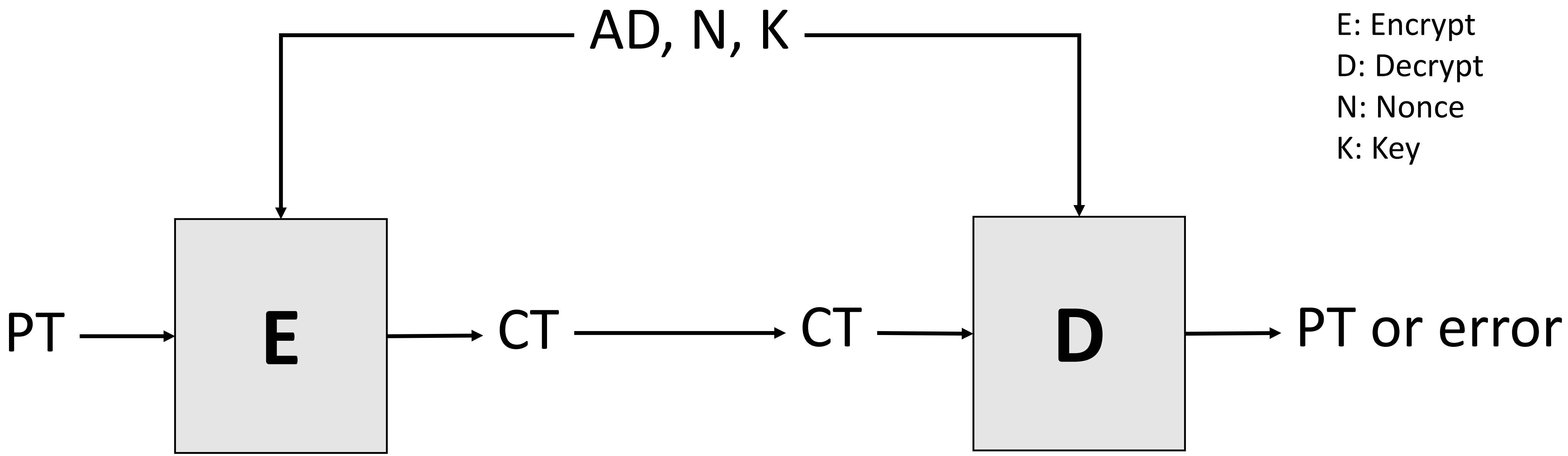
- Aug 2018: Formal Call for Proposals
- Feb 2019: Deadline for Submissions (57 received)
- Apr 2019: 56 Submissions Advance to Round 1
- *Sept 2019: Round 2 Begins*

ACE	PHOTON-Beetle
ASCON	Pyjmask
Bleep64	Qameleon
CiliPadi	Quartet
CLAE	REMUS
CLX	Romulus
COMET	SAEAES
DryGASCON	Saturnin
Elephant	Shamash & Shamashash
ESTATE	SIMPLE
FlexAEAD	SIV-Rijndael256
ForkAE	SIV-TEM-PHOTON
Fountain	Skinny
GAGE-InGAGE	SNEIK
GIFT-COFB	SPARKLE
Gimli	SPIX
Grain-128AEAD	SpoC
HERN & HERON	Spook
HYENA	Subterranean 2.0
ISAP	SUNDAE-GIFT
KNOT	Sycon
LAEM	TGIF
Lilliput-AE	TinyJambu
Limdolen	Triad
LOTUS & LOCUS	TRIFLE
mixFeed	WAGE
ORANGE	Xoodyak
Oribatida	Yarara & Coral

Authenticated Encryption with Associated Data

- AEAD is a symmetric encryption scheme

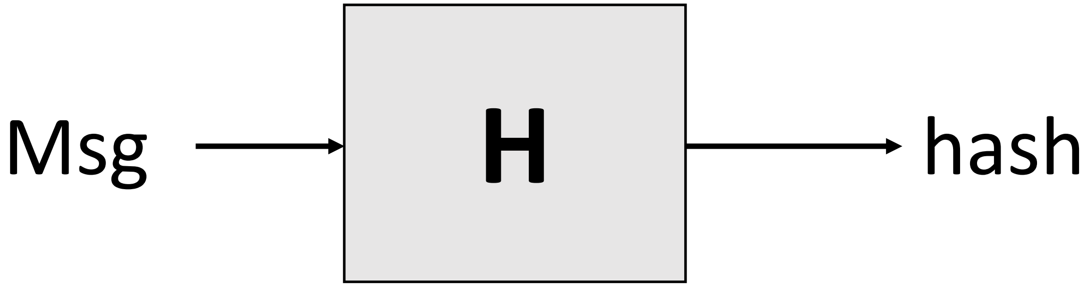
PT: Plain Text
CT: CipherText
AD: Associated Data
E: Encrypt
D: Decrypt
N: Nonce
K: Key



Cryptographic Hash Functions

Cryptographic Hash functions convert a message into a unique, fixed-length digest

- Collision resistance
- Preimage resistance
- Second-preimage resistance




```
        H(“NIST”) = FCE07FF980244E6D
        H(“FIST”) = 70F44C69CA82041B
H(“National Institute of…”) = C034262E461C6474
```

Testing Approach

Design Tests Based on Cryptographic Properties

- Implementations should satisfy the algorithmic properties of AEAD and HASH

Tests

- Bit Exclusion
- Bit Contribution (3 variations)
- Buffer Check
- Ciphertext Length Check

Apply Tests to LWC Standardization Process Submissions

- All variants of reference implementations

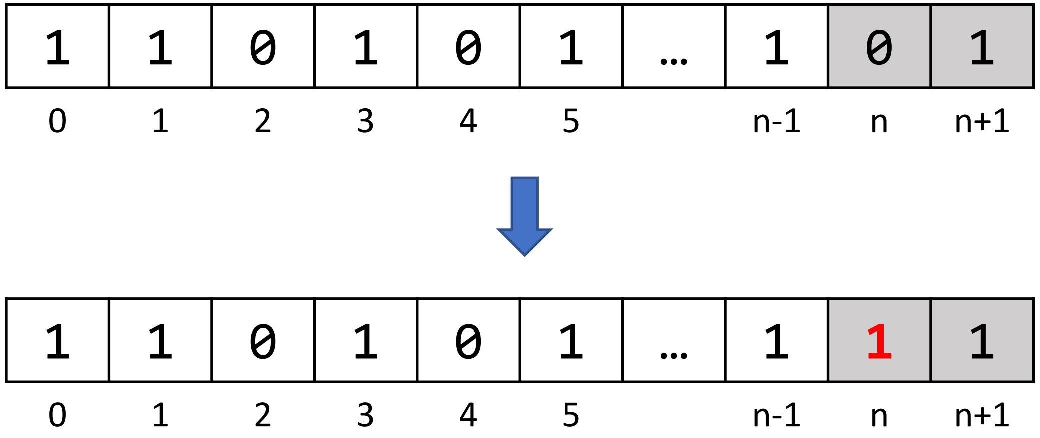
Bit Exclusion

Motivation

Bits beyond the specified input message length should be ignored

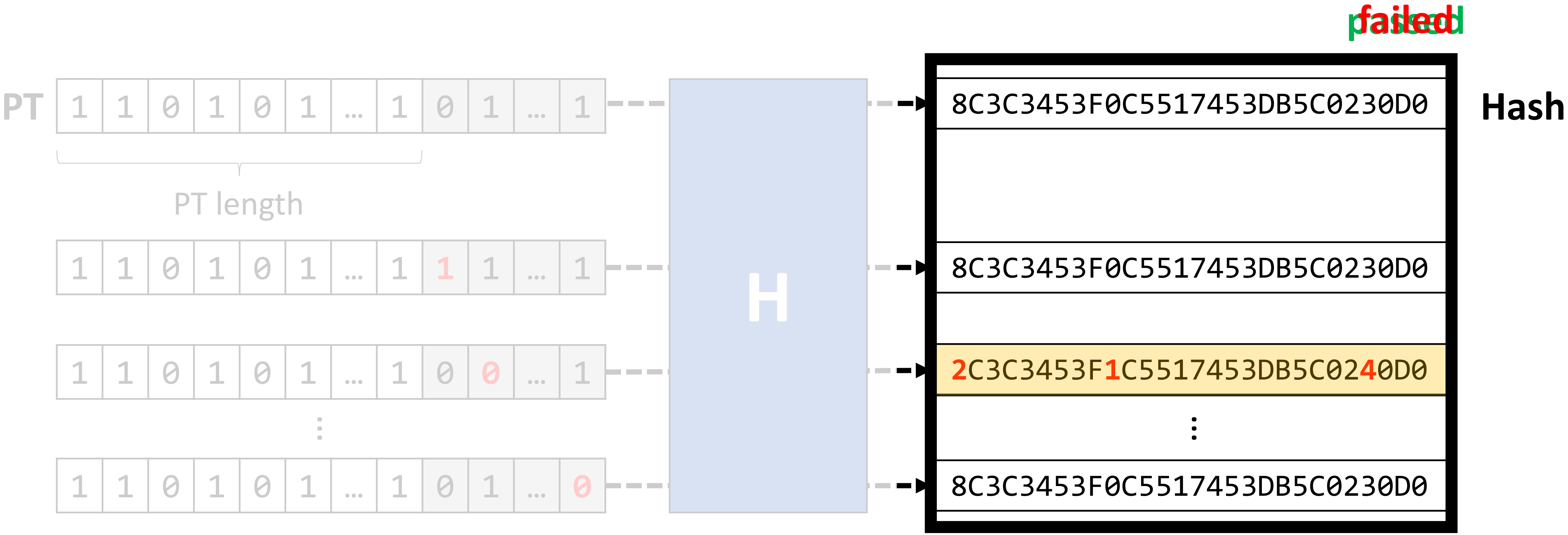
Strategy

- Generate a plaintext message *m* of length *n*
- Flip one bit of *m* outside length *n*, call this m’
- Check ***H****(m)* = ***H****(m’)*?
  - If no, then **fail**



Bit Exclusion

Bits beyond the specified input length should be ignored



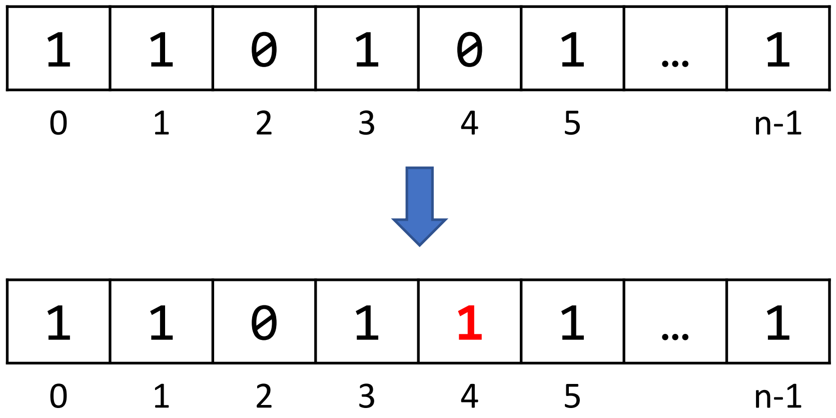
Bit Contribution for Plaintext

Motivation

Second-Preimage Resistance: given a message *m* and hash function *H*, it should be difficult to find a $m' \neq m$ such that $H(m') = H(m)$

Strategy

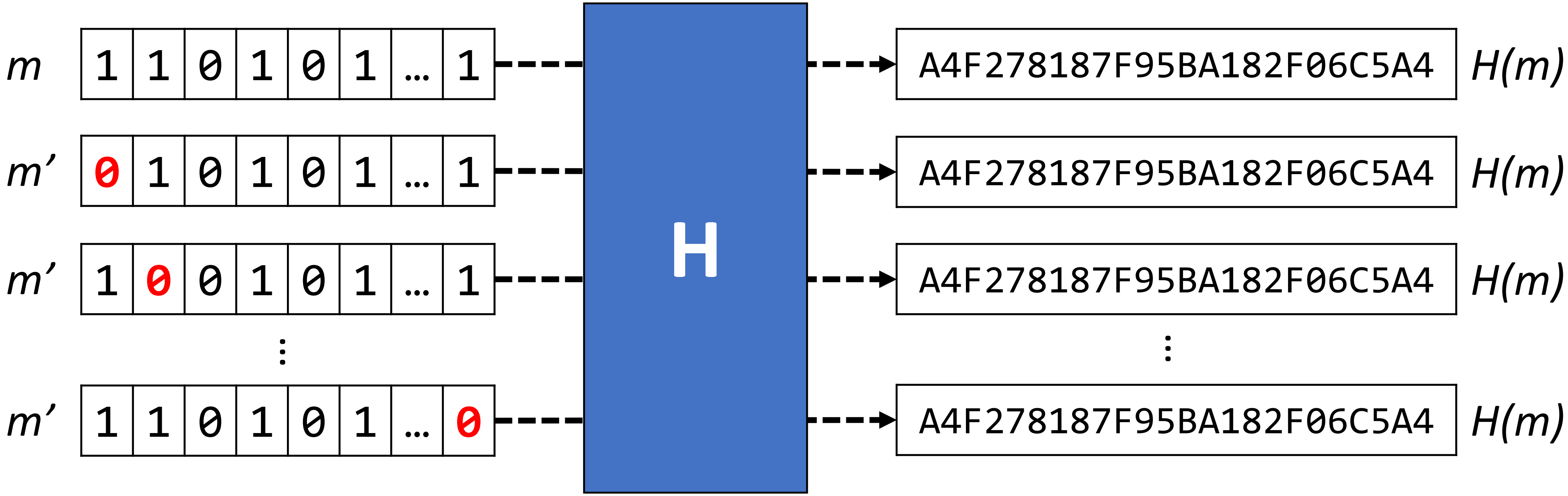
- Generate a plaintext message *m* of length *n*
- Flip one bit of *m*, call this m'
- Check ***H****(m)* = ***H****(m')*?
  - If yes, then **fail**



Bit Contribution for Plaintext

Second-Preimage Resistance

Given a message *m* and hash function *H*, it should be difficult to find a *m'* ≠ *m* such that *H(m')* = *H(m)*



Bit Contribution for Nonce

Motivation

LWC requirements states, “AEAD algorithms are expected to maintain security as long as the nonce is unique (not repeated under the same key)”

Strategy

- Generate a random *PT*, *AD*, *N*, and *K*
- Process *PT, AD*, *N*, and *K*, yielding *CT*
- Flip one bit of *N*, call this *N’*
- Process PT, AD, *N’*, and *K*, yielding *CT’*
- XOR *CT* and *CT’*, and add result to matrix

	CT_0	CT_1	CT_2	CT_3	CT_4	…	CT_C
N_0	5204	5102	4802	5219	4787	…	5223
N_1	4883	5209	4778	5247	4792	…	5213
N_2	5204	5209	4778	5183	5211	…	4985
N_3	5085	5201	5179	5183	5211	…	5014
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
N_N	5226	**8406**	4800	5214	**7001**	…	4985

Really small or really large matrix values imply a failure

Bit Contribution for Key

Motivation

AEAD algorithms are expected to maintain security when the key is unique

Strategy

- Generate a random *PT*, *AD*, *N*, and *K*
- Process *PT, AD*, *N*, and *K*, yielding *CT*
- Flip one bit of *K*, call this *K'*
- Process PT, AD, *N*, and *K'*, yielding *CT'*
- XOR *CT* and *CT'*, and add result to matrix

	CT_0	CT_1	CT_2	CT_3	CT_4	…	CT_C
K_0	5204	5102	4802	5219	4787	…	5223
K_1	4883	5209	4778	5247	4792	…	5213
K_2	5204	5209	4778	5183	5211	…	4985
K_3	5085	5201	5179	5183	5211	…	5014
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
K_N	5226	8406	4800	5214	7001	…	4985

Really small or really large matrix values imply a failure

Buffer Check (Decryption Failure Test)

Motivation

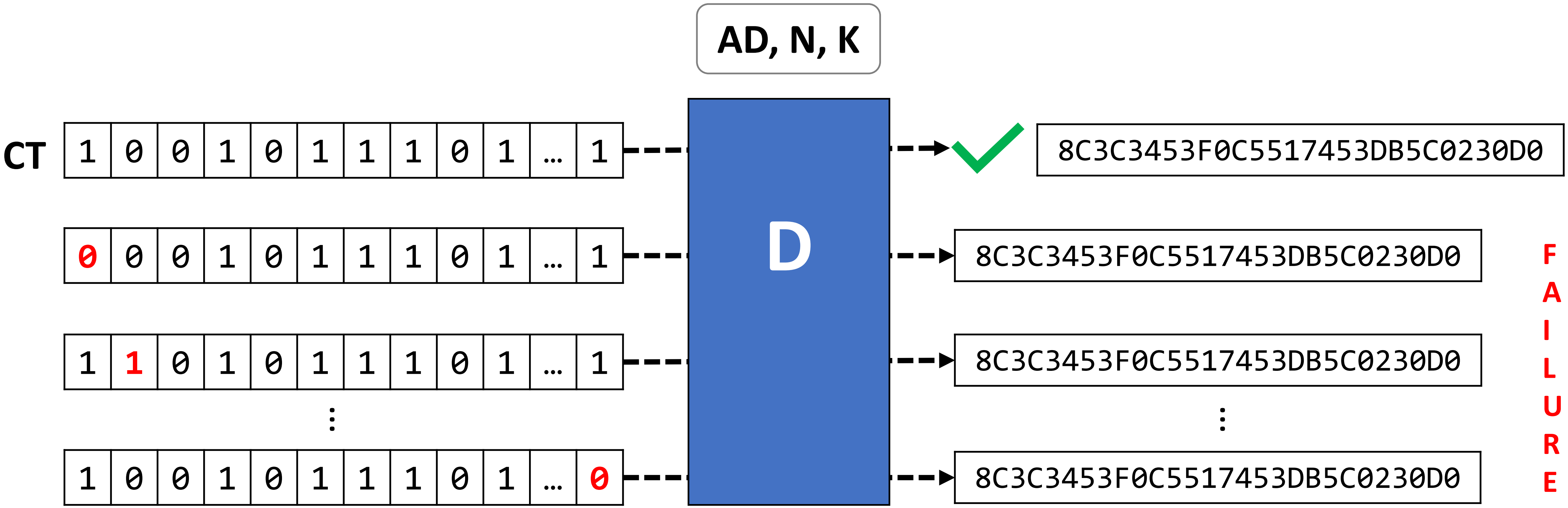
"Plaintext **should not** be returned by the decryption-verification process if the ciphertext is invalid."

Strategy

- Generate a ciphertext $CT_{PT,\ AD,\ N,\ K}$
- Flip one bit of $CT_{PT,\ AD,\ N,\ K}$
- Invoke *decrypt* function
- Check the buffer where plaintext was to be stored
  - If the buffer has a consecutive 10-byte match to *PT*, then **fail**

Buffer Check (Decryption Failure Test)

"Plaintext **shall not** be returned by the decryption-verification process if the ciphertext is invalid."



Ciphertext Length Check

Motivation

Algorithms must make sure that the ciphertext is ***at most*** ***CRYPTO_ABYTES*** longer than the plaintext

Strategy

- Generate a random *PT* of length *n*, *AD*, *N*, and *K*
- Process (encrypt) *PT*, *AD*, *N*, and K, yielding *CT*
- Make sure |*CT*|>= *n* and |*CT*|<=*n* + CRYPTO_ABYTES
  - If no, then **fail**

Experimentation

AEAD

56 algorithms,
157 reference implementations
(All variants),

- Bit Contribution for Plaintext
- Bit Contribution for Nonce
- Bit Contribution for Key
- Bit Exclusion
- Buffer Check
- Ciphertext Length Check

HASH

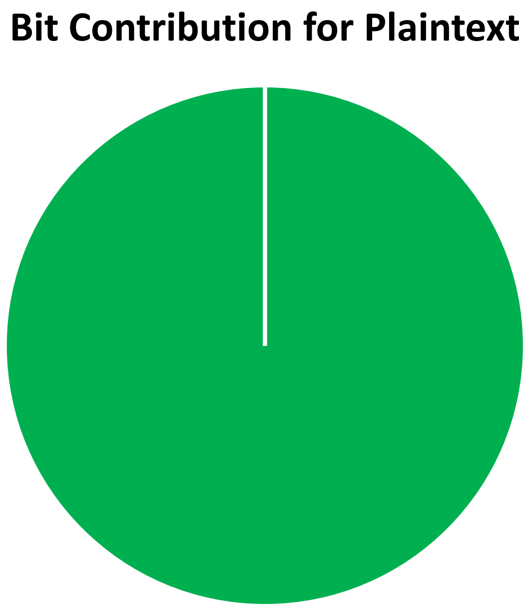
22 algorithms,
39 reference implementations,
(All variants)

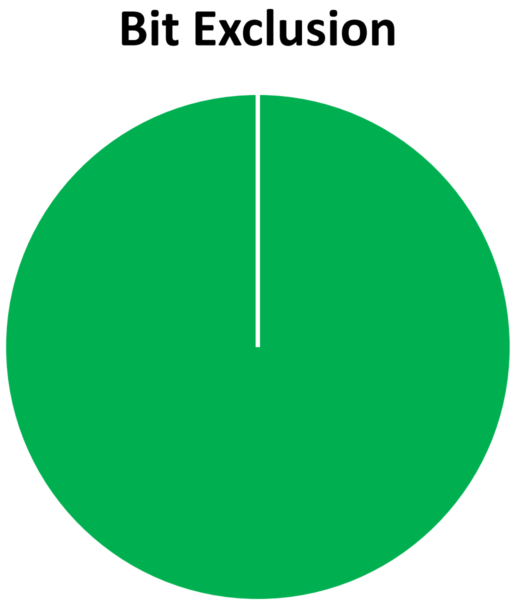
- Bit Contribution for Plaintext
- Bit Exclusion

Results – HASH

No failures were discovered for the hash function implementations

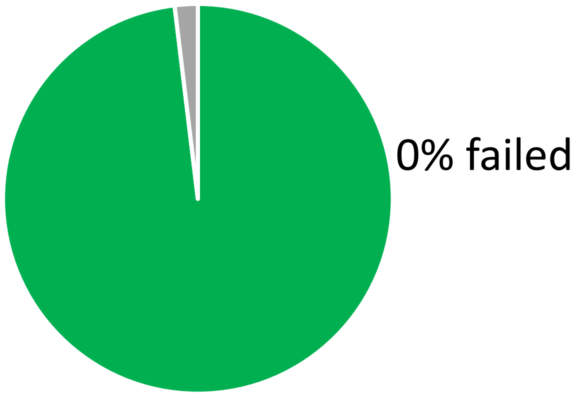
- Does not guarantee there are no bugs



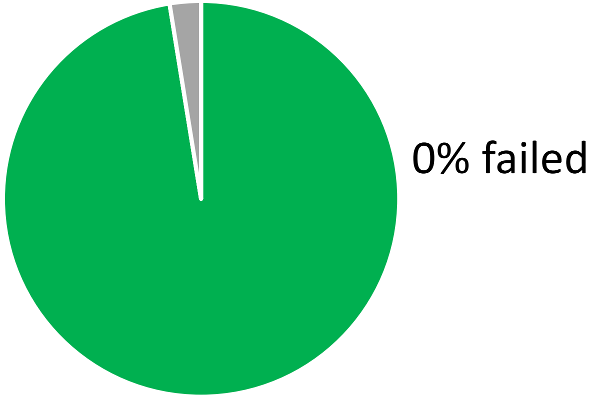


Results – **AEAD**

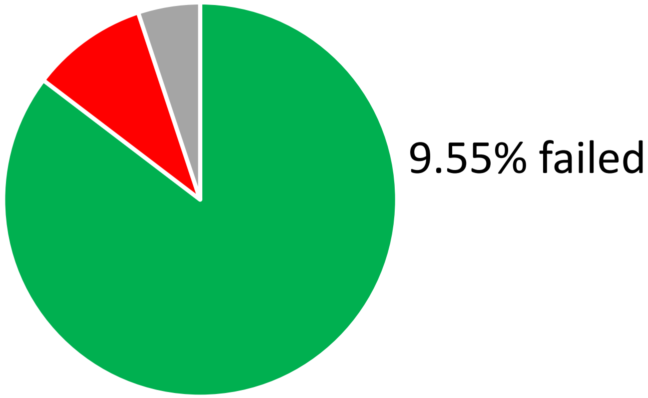
Bit Exclusion



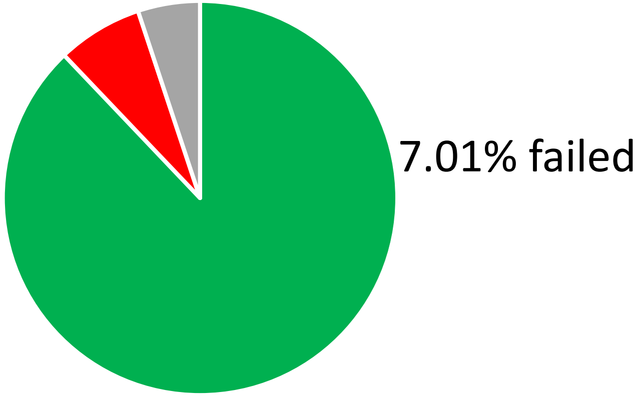
Bit Contribution for Plaintext



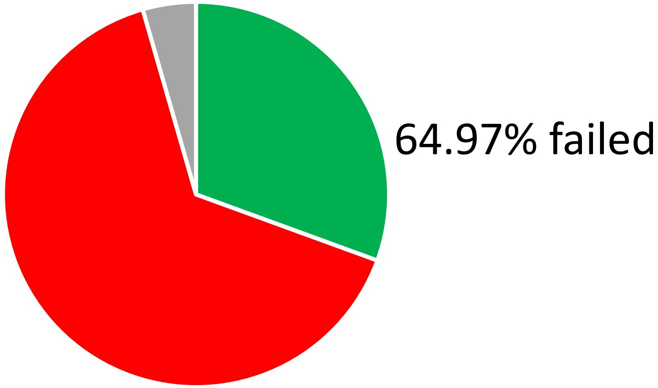
Bit Contribution Nonce



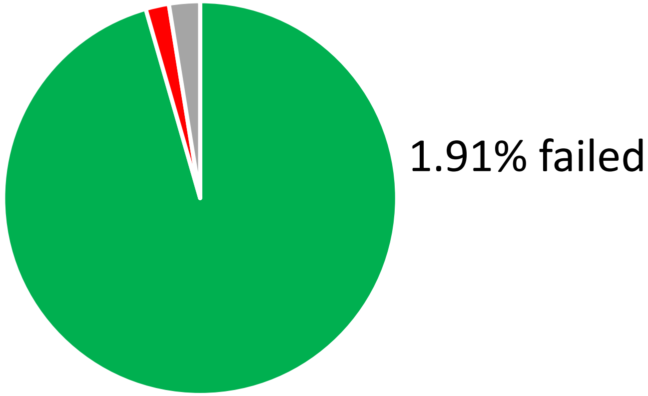
Bit Contribution Key



Buffer Check (Decryption Failure)



Ciphertext Length Check



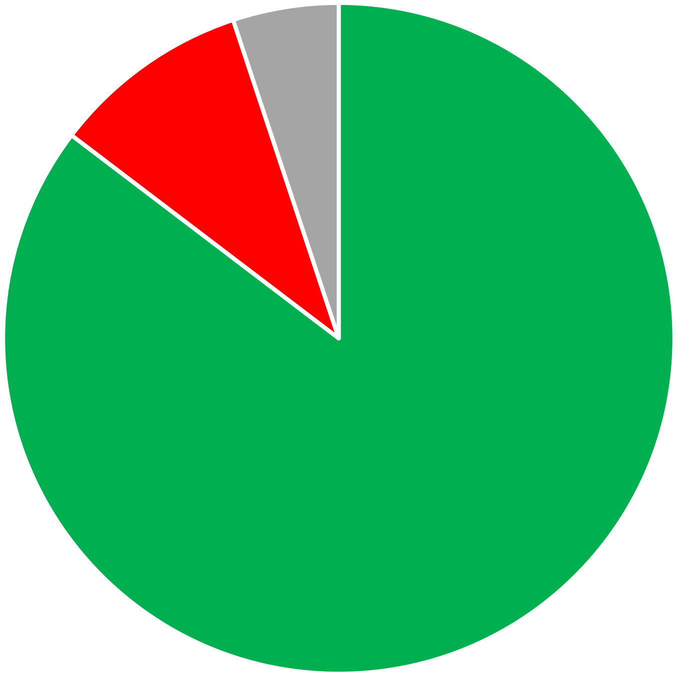
AEAD Results—**Bit Contribution for Nonce**

85.35% passed

5.10% indeterminate

9.55% failed

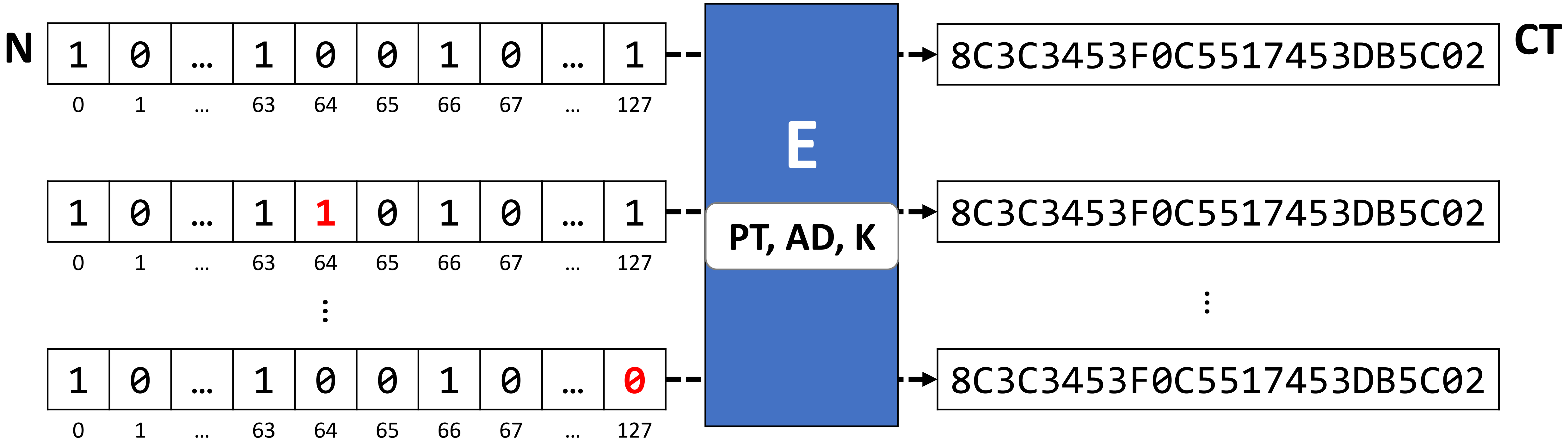
- 8/15 failed implementations are definitive failures
  - bleep64
  - lotus
  - orange
  - qameleon128128128v1
  - qameleon12812896v1
  - qameleon12812864v1
  - quartet
  - wage



Lotus and Locus—A. Chakraborti, et al.

Implementation *lotus* failed the Bit Contribution for Nonce test

- Bits 64 to 127 do not affect the ciphertext produced



Lotus and Locus—A. Chakraborti, et al.

Implementation *lotus* failed the Bit Contribution for Nonce test

- Bits 64 to 127 do not affect the ciphertext produced

ltwegift64lotus/encrypt.c, corrected

```
void init(u8 *nonced_key, u8 *nonced_mask, const u8 *key, const u8 *nonce)
{
    u8 twk;
    u8 zero[CRYPTO_BLOCKBYTES] = { 0 };
    u8 enc_zero[CRYPTO_BLOCKBYTES];
⋮
    // compute K_N = K + N
    memcpy(nonced_key, key, CRYPTO_KEYBYTES);
    xor_bytes(nonced_key, nonce, CRYPTO_NPUBBYTES);
```

AEAD Results—**Buffer Check Test**

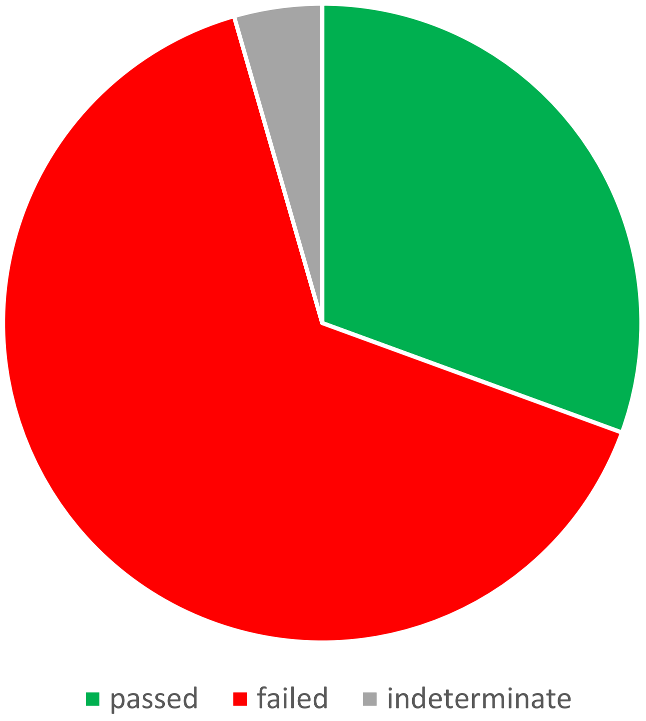
30.57% passed

4.46% indeterminate

64.97% failed

Some implementations acknowledge that they do not clear the buffer

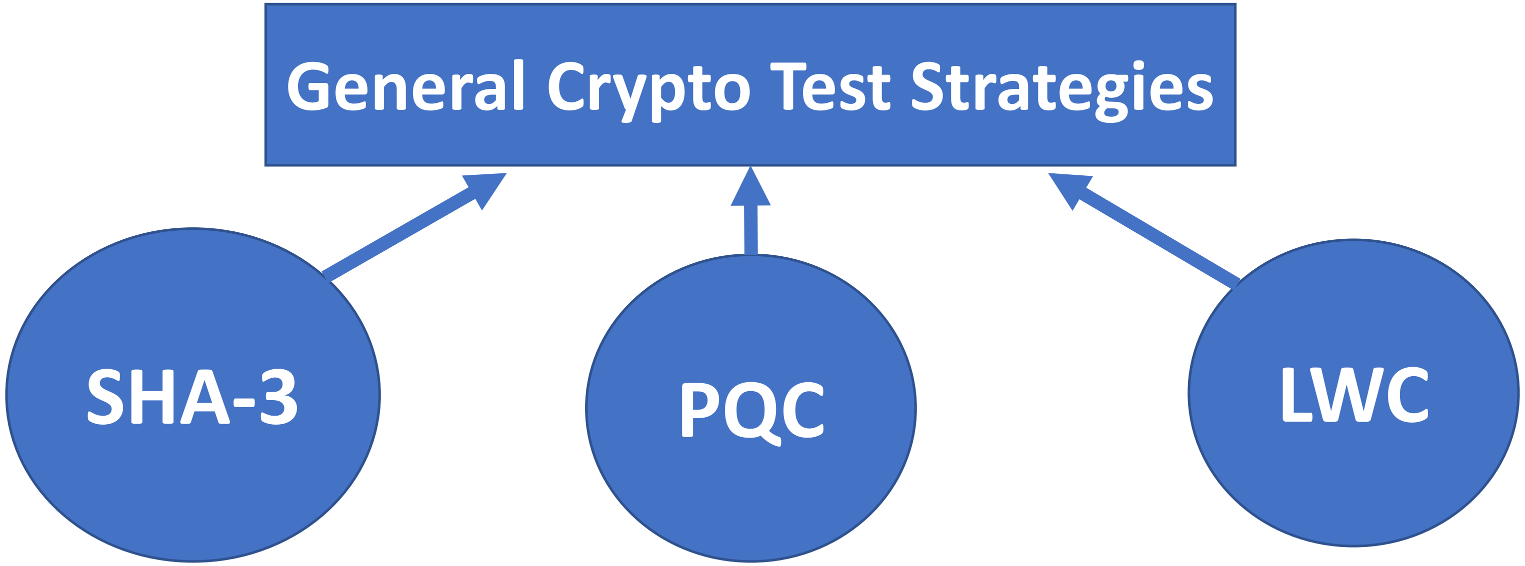
Possible Solution: Use an additional temporary buffer.



Conclusion

Metamorphic tests based on cryptographic properties is effective

- We have seen many test failures and found several source code bugs



Future Work

- Test the optimized implementations
- Develop a generic testing approach for cryptographic algorithms