Mike Hamburg, Rambus Cryptography Research

The STROBE protocol framework

Secure, simple, and small

What is STROBE?

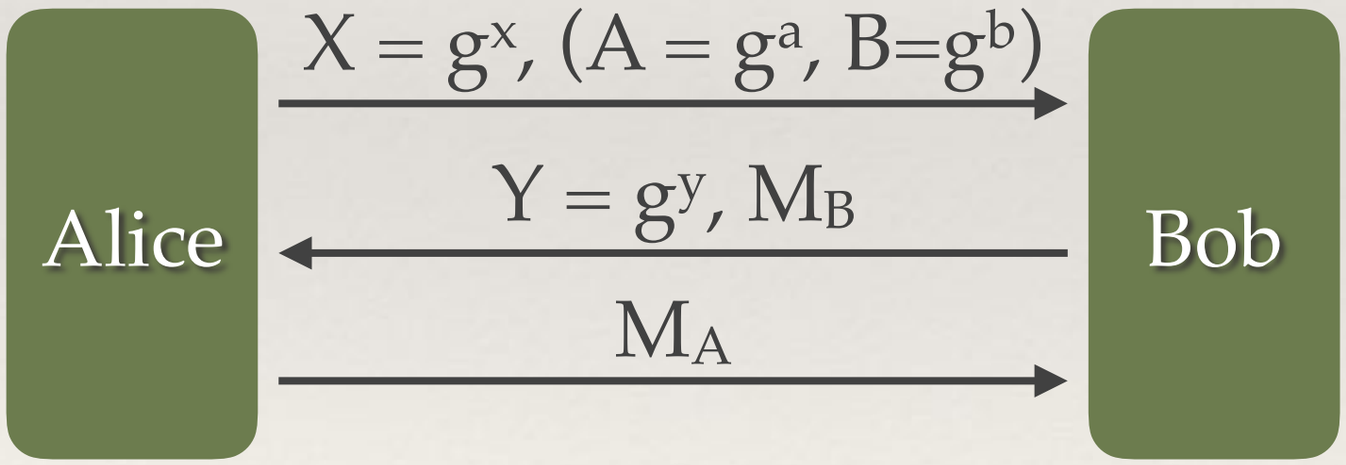
- Protocol framework with embedded focus
  - Simple protocols and handshakes
    - Encrypt, MAC, hash, sign…
- Simple, easy to analyze
- Non-terrible performance
- Can be an instance of NIST [cSHAKE]

Motivation: bespoke protocols

- Best practice: use TLS or IPSEC
- Real-world protocols have diverse requirements
  - Public key encryption/auth algorithms
  - Message flow
  - Code size and memory requirements
- Result: lots of custom protocols!
  - Design and analysis are a pain
  - Often insecure

Motivation: academic protocols

- Hash, sign, encrypt and MAC on tuples, key confirm
- [FHMQV]-C:



$$d = H(X,Y,A,B)$$

$$e = H(Y,X,A,B)$$

$$\sigma = g^{(x+da)(y+eb)}$$

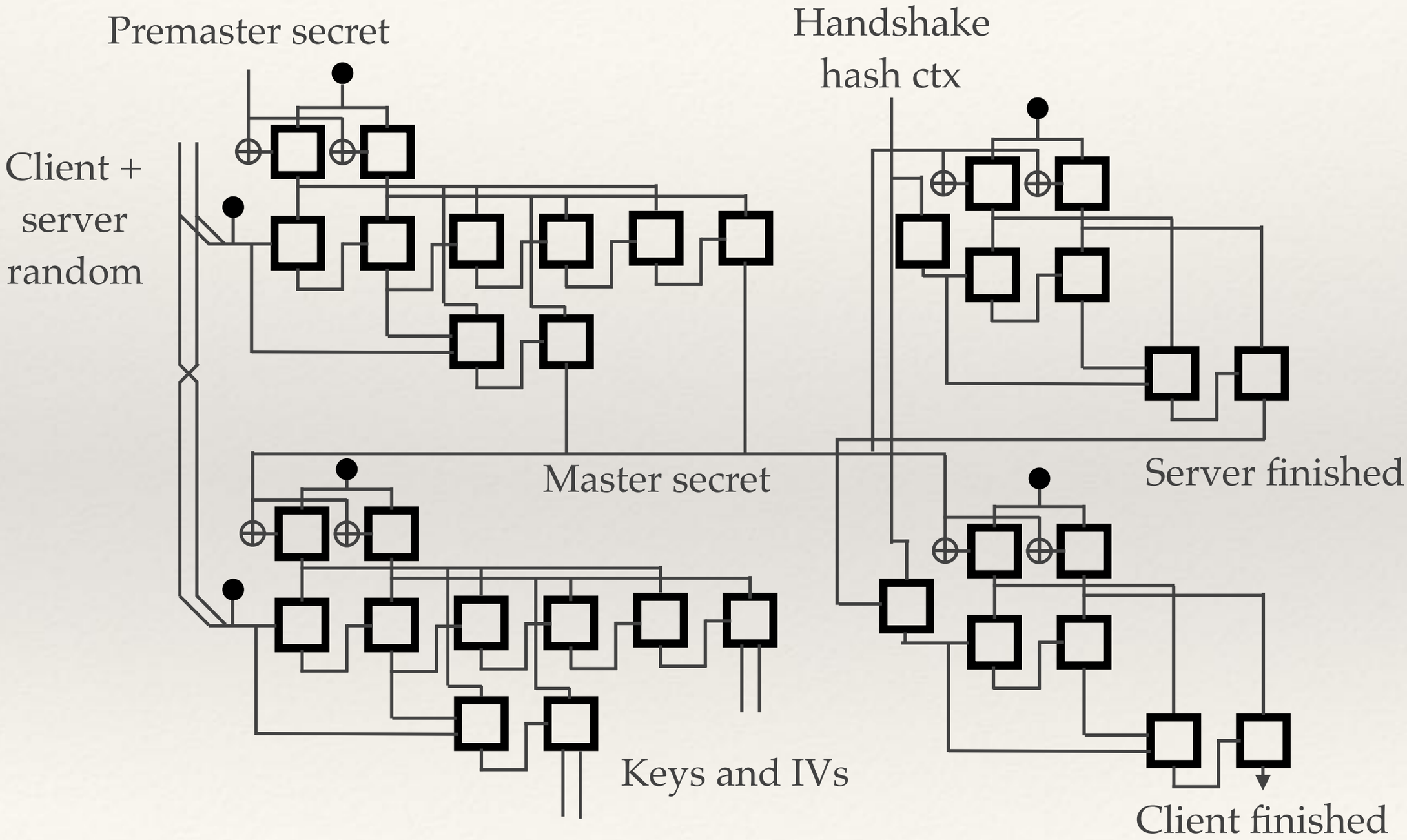
$$K_1 = KDF_1(\sigma,A,B,X,Y)$$

$$M_A = MAC(K_1; A,X)$$

$$M_B = MAC(K_1; B,Y)$$

$$K_2 = KDF_2(\sigma,A,B,X,Y)$$

Motivation: [TLS 1.2]



Finished is also encrypted, but I got bored before drawing the cipher calls.

The modern solution

Eg: [TLS 1.3], [Noise], [BLINKER]

STROBE overview

All messages pass through STROBE

(at the least, to update the hash)

Partially trusted
Sets keys
Payload plaintext
Metadata
Protocol framing
Asymmetric crypto

Trusted
Hashes all messages
Stores cipher/hash state
Optional encryption
Computes MAC, hash

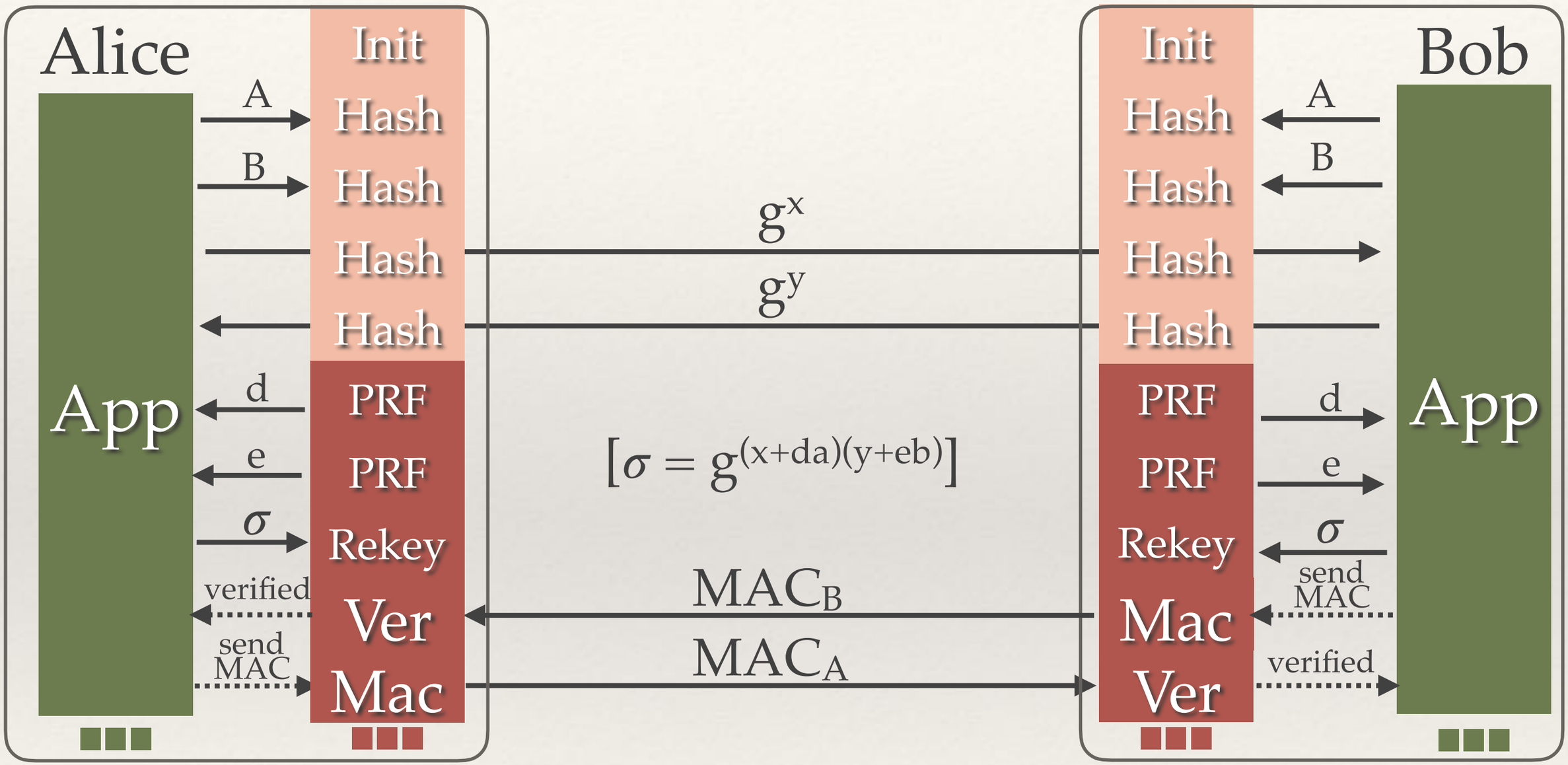
Untrusted
Network or flash
Sees ciphertext, MAC
Reliable when no attack
In-order

STROBE two-party protocols

Alice and Bob's STROBE instances advance in lockstep

If a message is changed on the wire, the next MAC will fail

STROBE example: FHMQV-C



Everything is based on running hash

$d = H(A,B,g^x,g^y)$ $e = H(A,B,g^x,g^y,d)$

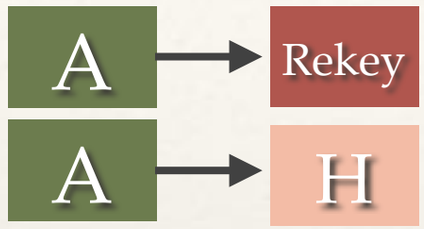
$MAC_B = H(A,B,g^x,g^y,d,e,\sigma)$

(roughly)

STROBE operations

Key

AD

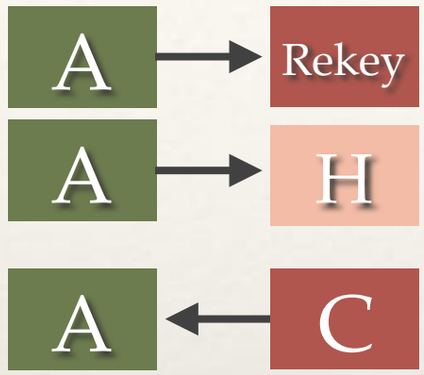


STROBE operations

Key

AD

PRF

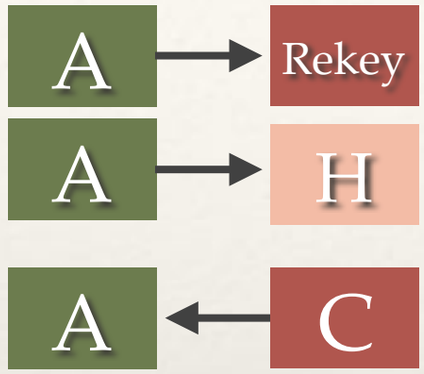


STROBE operations

Key

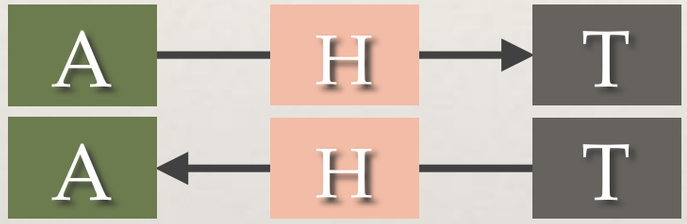
AD

PRF



Send clear

Recv clear

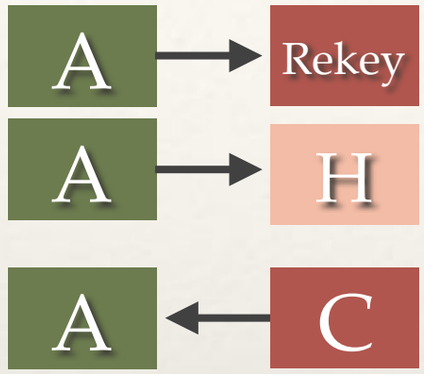


STROBE operations

Key

AD

PRF

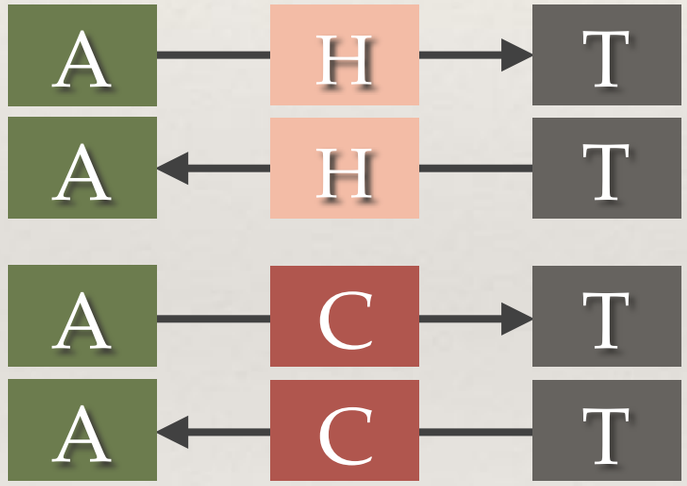


Send clear

Recv clear

Send enc

Recv enc



STROBE operations

Key 

AD 

PRF 

Send clear 

Recv clear 

Send enc 

Recv enc 

Send MAC 

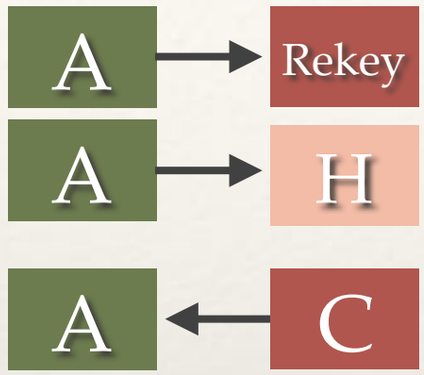
Verify MAC 

STROBE operations

Key

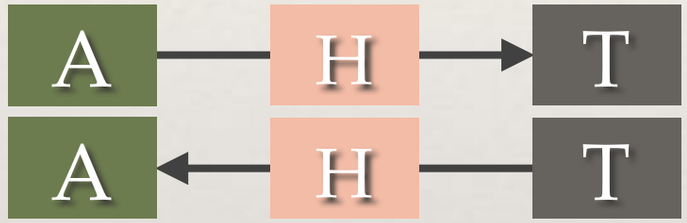
AD

PRF



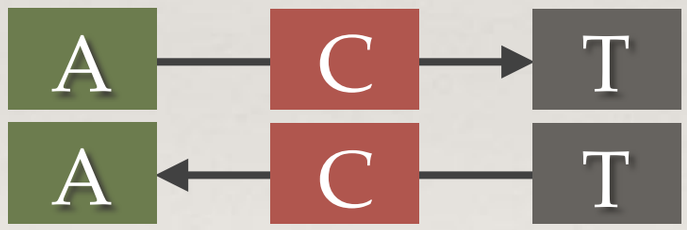
Send clear

Recv clear



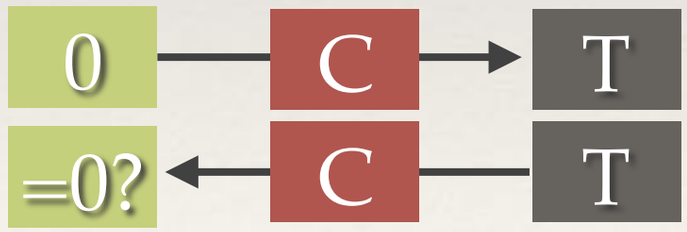
Send enc

Recv enc



Send MAC

Verify MAC

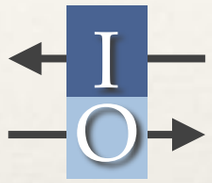


Ratchet

STROBE operations

Key	A	→	Rekey
AD	A	→	H
PRF	A	←	C
Send clear	A	— H →	T
Recv clear	A	← H —	T
Send enc	A	— C →	T
Recv enc	A	← C —	T
Send MAC	0	— C →	T
Verify MAC	=0?	← C —	T
Ratchet	0	→	Rekey

All described by 4 features:



Data flow direction

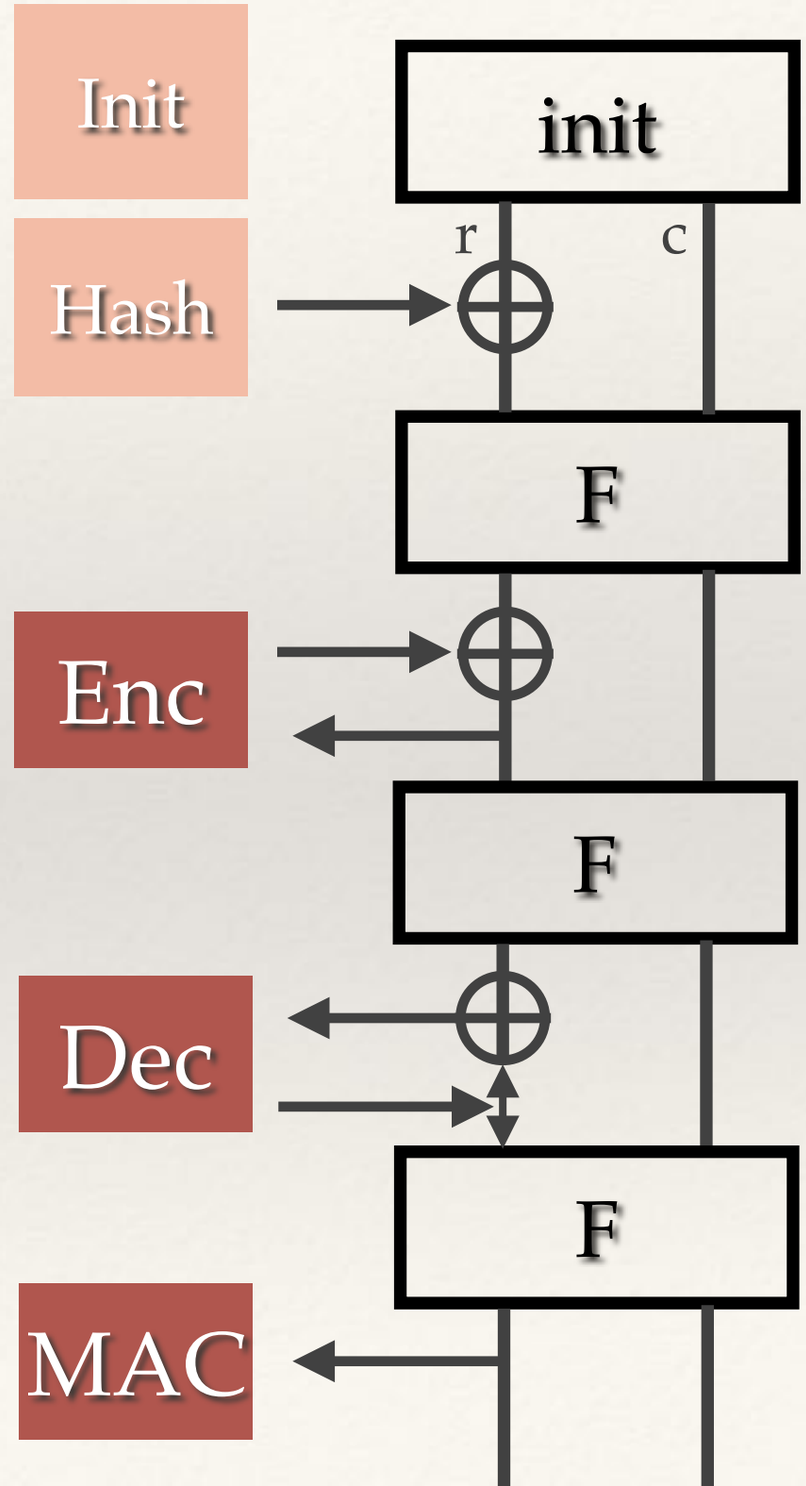
Data goes to/ from app

Data goes to/ from cipher (else just hash)

Data goes to/ from transport

STROBE implementation

- Duplex sponge construction [RadioGatún, KECCAK, Duplex]
- State is divided into two parts:
  - Rate gets xor'd with input block
  - Capacity is kept separate
  - (r, c) = F(r⊕m, c)



“Hash all the things”

- Goal: output of Strobe is a **random oracle**

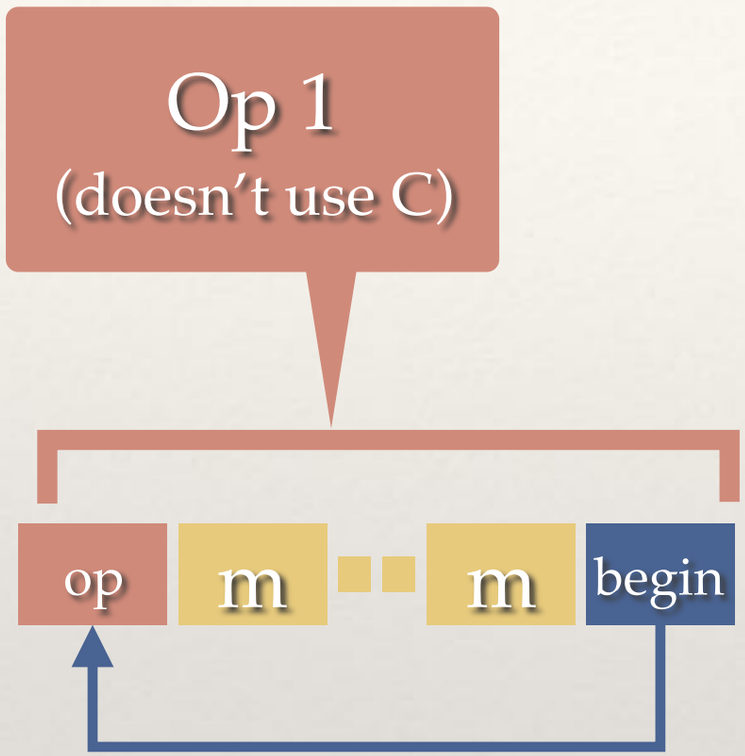
- Input is all previous operations
  - H(“abc”) != H(“a”,“bc”)
  - Includes operation type and data
  - Includes intended use of output

STROBE padding

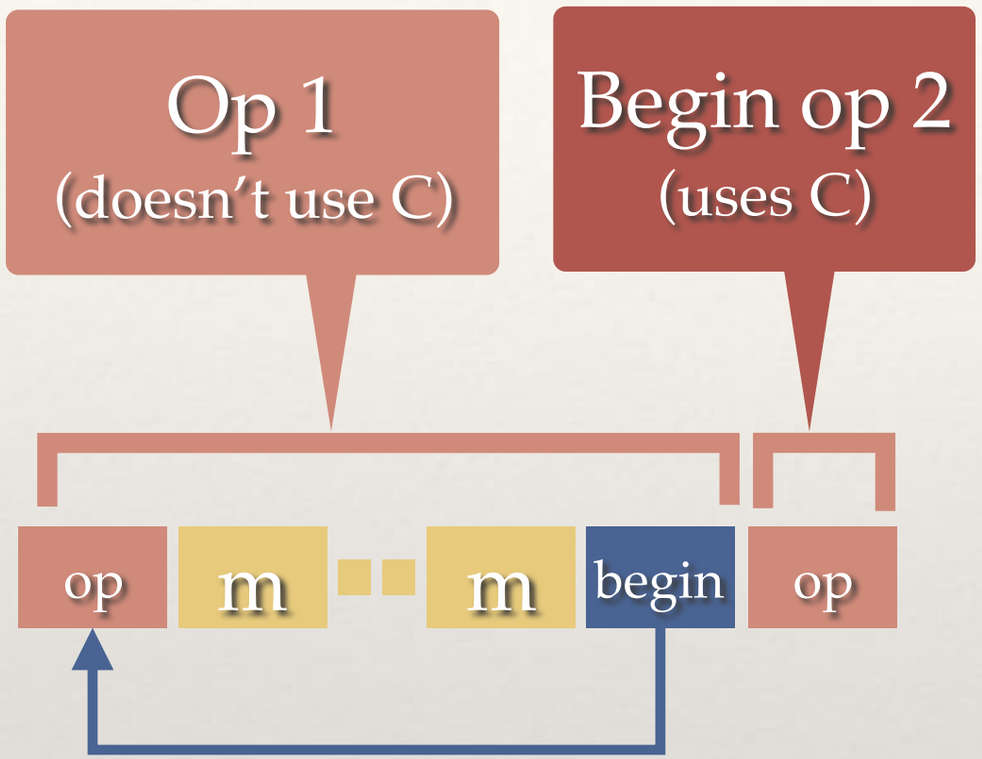
- Theorem from [Duplex]: sponge output is a **random oracle** on previous inputs (if F is a random fn/perm)

- ⇒ Requirement: each time F is called, can parse:
  - Entire previous transcript
  - Intended use of output

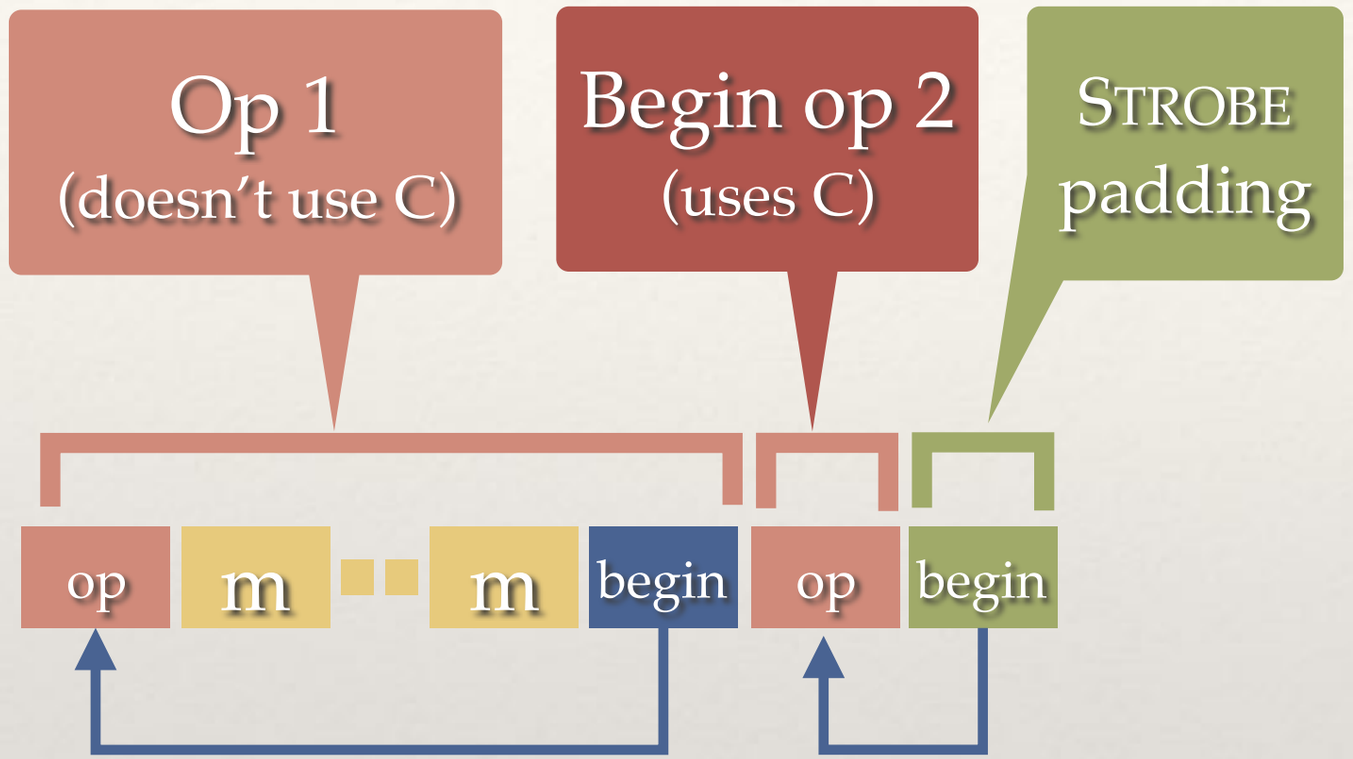
STROBE padding



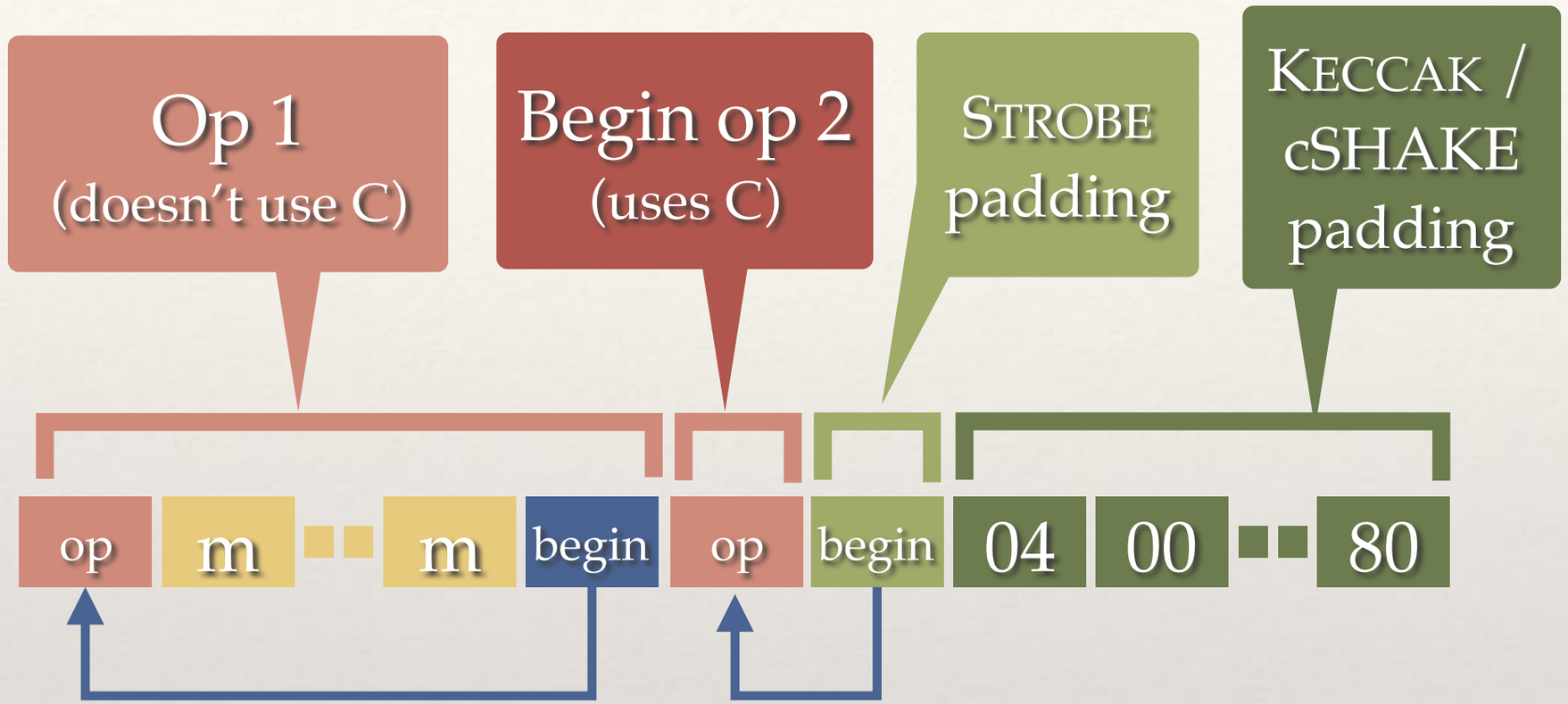
STROBE padding



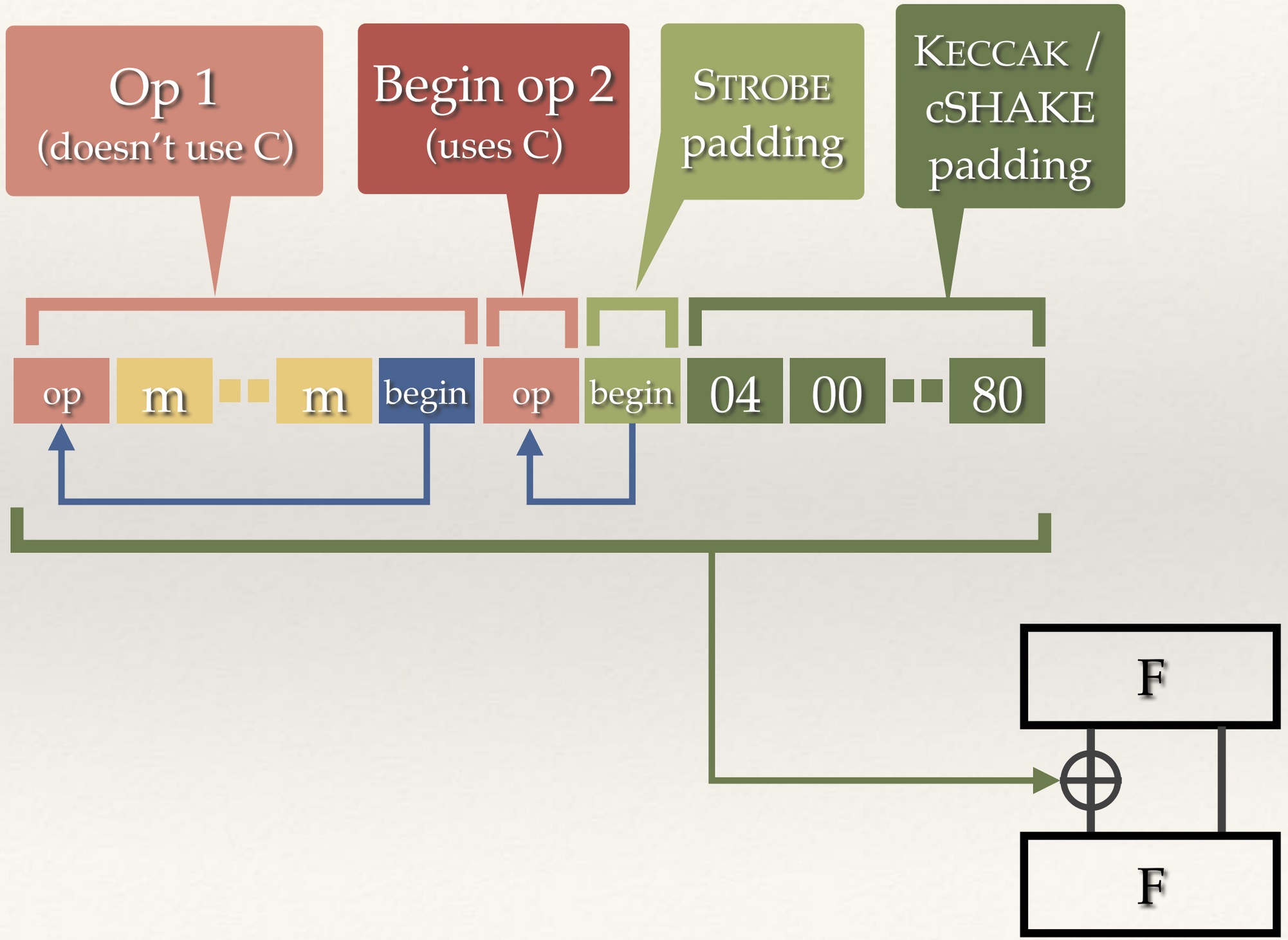
STROBE padding



STROBE padding



STROBE padding



Operations with metadata

- Output depends on its **intended use**
  - "Will be used to encrypt a message" isn't good enough
  - What kind of message? How long?

- Disambiguate with metadata operations
  - Metadata AD/CLR/ENC before each operation
  - Can be (tag, length) of protocol framing
  - Optional but recommended. Cheap.

Implementation

- Prototype C code at https://strobe.sourceforge.io/
- Optimized for size on embedded devices
- Includes simple callback-based IO engine
- Curve25519 code may be of independent interest

Implementation results

- KECCAK-*f*[800], Cortex-M3/M4 C
  - < 2 KB code, <350 B stack
- With Curve25519 ECDH/sign/verify; PRNG support
  - < 3.5KB code, 700B stack, 120B PRNG pool
- Significantly smaller with asm intrinsics (unreleased)

Future work

- Better documentation and example protocols
- Improve engine code
- Non-sponge implementation
- Formal analysis
  - Most work is done by [Duplex]
  - Rollback resistance, full protocol analysis
- Post-quantum analysis

Works cited

- [BLINKER]: Markku-Juhani Saarinen. “Beyond Modes: Building a Secure Record Protocol from a Cryptographic Sponge Permutation.” CT-RSA 2014, https://eprint.iacr.org/2013/772
- [cSHAKE]: John Kelsey, Shu-jen Chang Ray Perlner. “SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash and ParallelHash.” NIST SP 800-185, December 2016, http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-185.pdf
- [Duplex]: Guido Bertoni, Joan Daemen, Michaël Peeters, Gilles Van Assche. "Duplexing the sponge: single-pass authenticated encryption and other applications.” SAC 2011, http://sponge.noekeon.org/SpongeDuplex.pdf
- [FHMQV]: Augustin Sarr, Philippe Elbaz-Vincent, Jean-Claude Bajard. “A secure and efficient authenticated diffie–hellman protocol.” European PKI Workshop 2009, https://eprint.iacr.org/2009/408

Works cited

- [KECCAK]: Guido Bertoni, Joan Daemen, Michaël Peeters, Gilles Van Assche. “The KECCAK sponge function family.” NIST SHA-3 submission, http://keccak.noekeon.org/
- [Noise]: Trevor Perrin. “Noise Protocol Framework.” http://www.noiseprotocol.org/
- [RadioGatún]: Guido Bertoni and Joan Daemen, Michaël Peeters and Gilles Van Assche. “RadioGatún, a belt-and-mill hash function.” Cryptographic Hash Workshop 2006, http://eprint.iacr.org/2006/369
- [TLS 1.2]: Tim Dierks and Eric Rescorla. “The Transport Layer Security (TLS) Protocol, Version 1.2.” RFC 5246 (2008), https://www.ietf.org/rfc/rfc5246.txt
- [TLS 1.3]: Eric Rescorla. "The Transport Layer Security (TLS) Protocol Version 1.3.” draft, https://tlswg.github.io/tls13-spec/

FIN

Questions?