Additional Modes for ASCON

Rhys Weatherley
Southern Storm Software, Pty Ltd
NIST Sixth Lightweight Cryptography Workshop, June 2023

History

Lots of little cryptography experiments from 1990’s to today

1996: SSL 2.0 and 3.0 implementations for Oracle PowerBrowser

2015 to present: Arduino Cryptography Library

- AES, SHA-1, SHA-2, SHA-3, BLAKE-2, ChaCha, SPECK
- GCM, CTR, EAX, CBC, CFB, OFB, HMAC, HKDF, Poly1305
- Curve25519, Ed25519, P521, NewHope
- Added CAESAR finalists ASCON-128 and ACORN-128 in 2018
- AVR, ARM Cortex, and ESP32 platforms
- https://github.com/rweather/arduinolibs

2019 to 2022: Implementing and benchmarking LWC candidates in rounds 2 and 3

Recently: ASCON Suite and Additional Modes for ASCON

ASCON-CXOF - Customizable Hashing

Customizable XOF mode similar to cSHAKE (NIST SP 800-185)

ASCON-CXOF(X, L, N, C, a, b, r)

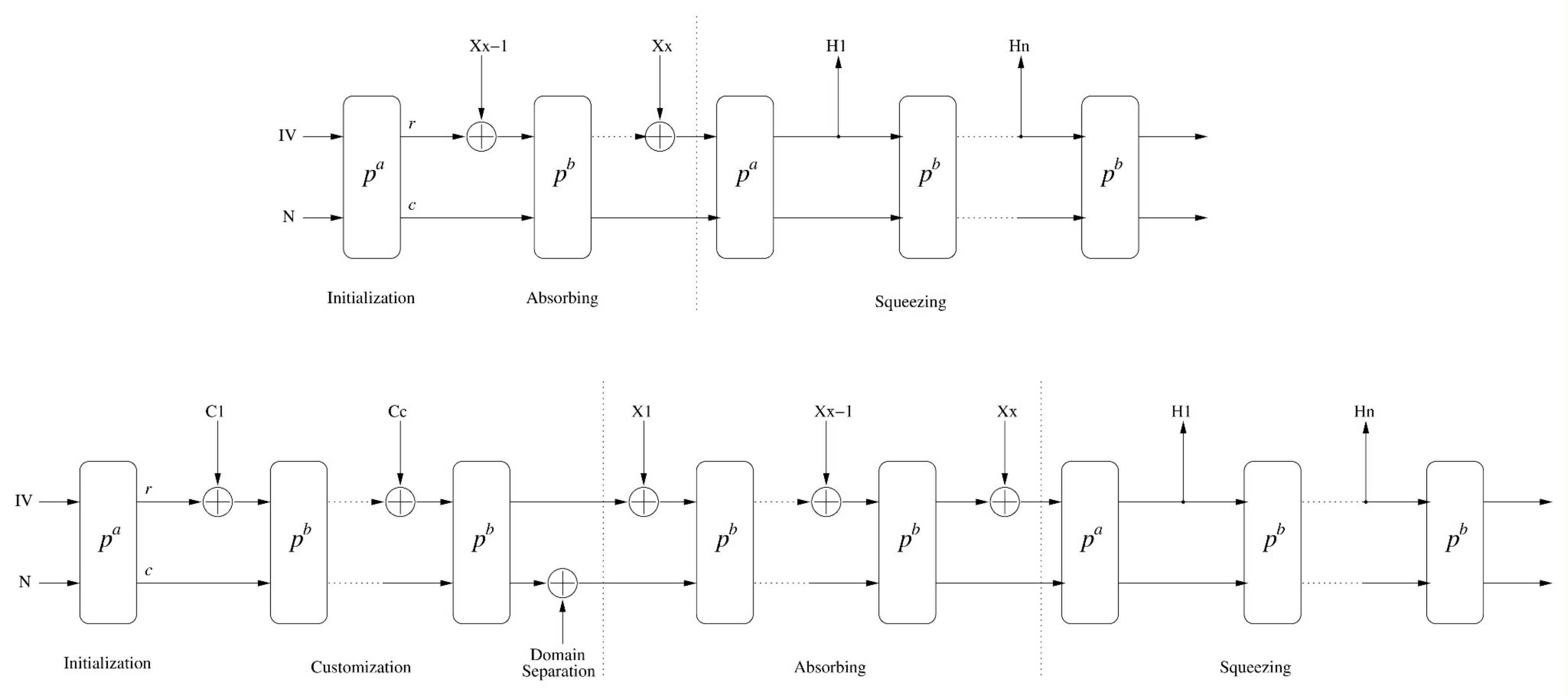
- X - Input string of any length
- L - An integer representing the desired output length (0 for indefinite)
- N - Function name string; e.g. “KMAC”, “KDF”, “PRNG”, etc (may be empty)
- C - Customization string for application-specific variants on N (may be empty)
- a - Number of ASCON rounds for initialization and finalization ($1 \le a \le 12$)
- b - Number of ASCON rounds for absorbing and squeezing ($1 \le b \le a$)
- r - Rate for absorbing and squeezing (64 or 128)

Except for the handling of N and C, ASCON-CXOF is the same as regular ASCON hashing.

Ascon-cXof - Pseudocode

```
Ascon-cXof($X, L, N, C, a, b, r$):
    $State \leftarrow p^a$(Format-First-Block($L, N, a, b, r$))
    if len(C) > 0 then
        $State \leftarrow$ Absorb($State$, pad($C$), $b, r$)
        $State \leftarrow$ State $\oplus$ 1
    $State \leftarrow$ Absorb($State$, pad($X$), $b, r$)
    $State \leftarrow p^a(State)$
    return Squeeze($State, L, b, r$)
```

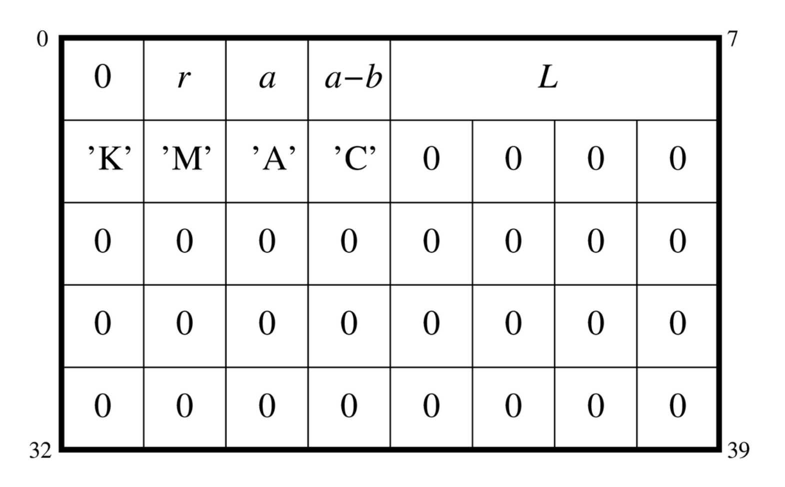
ASCON-CXOF - Visual Structure



Ascon-cXof - Handling the Function Name

ASCON’s hashing mode already encodes L, a, b, and r in the initial block.

We can add the function name N to the initial block:



0	r	a	$a-b$	L			
’K’	’M’	’A’	’C’	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

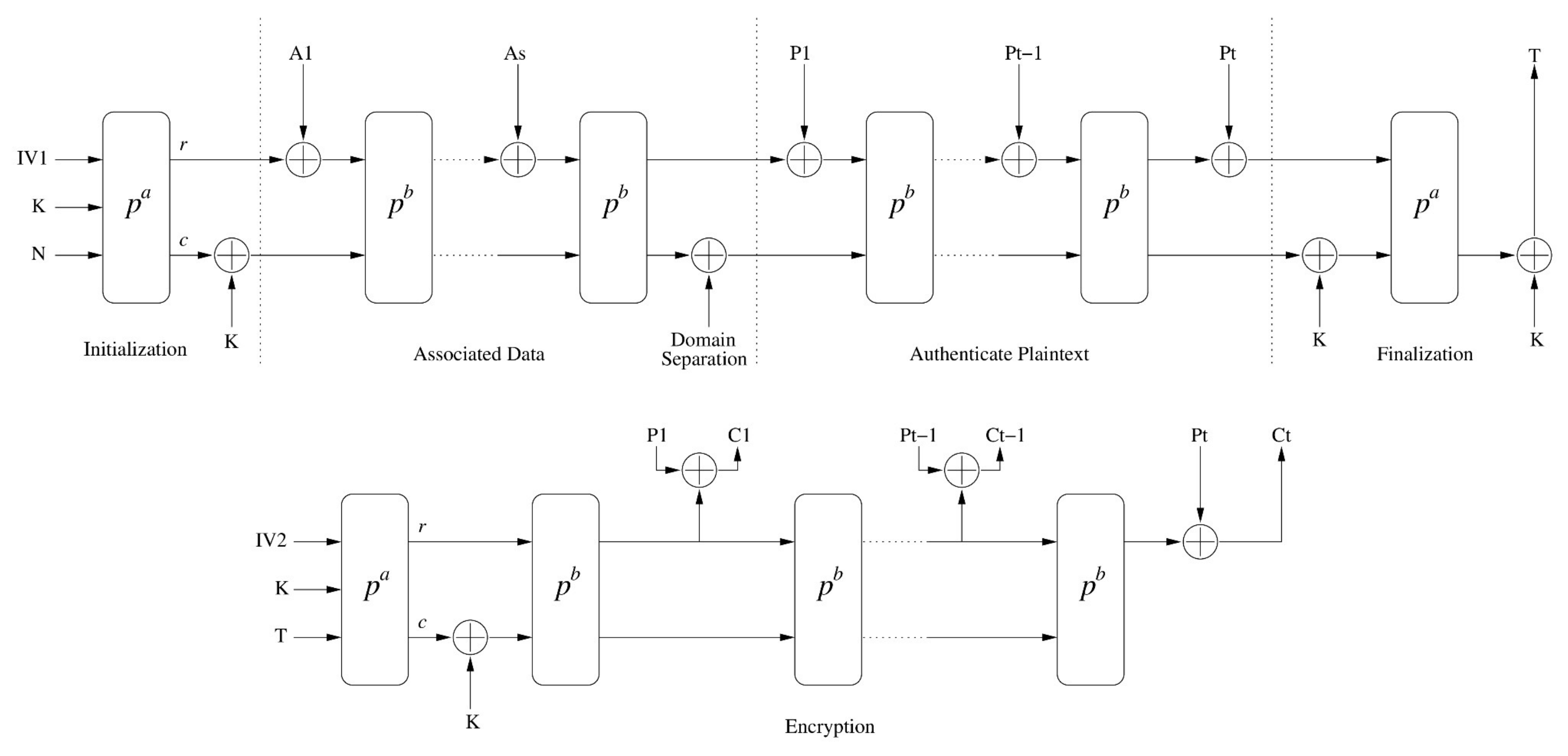
If len(N) > 256, then set $N \leftarrow$ Ascon-Hash(N)

ASCON-CXOF - Parameterization for Common Uses

- ASCON-HASH(X) = ASCON-CXOF(X, 256, “”, “”, 12, 12, 64)
- ASCON-HASHA(X) = ASCON-CXOF(X, 256, “”, “”, 12, 8, 64)
- ASCON-XOF(X) = ASCON-CXOF(X, 0, “”, “”, 12, 12, 64)
- ASCON-XOFA(X) = ASCON-CXOF(X, 0, “”, “”, 12, 8, 64)

- ASCON-KMAC(K, L, X, C) = ASCON-CXOF($K \,||\, X$, L, “KMAC”, C, 12, 12, 64)
- ASCON-KDF(K, L, C) = ASCON-CXOF(K, L, “KDF”, C, 12, 12, 64)
- ASCON-PRNG(*Seed*, C) = ASCON-CXOF(*Seed*, *0*, “PRNG”, C, 12, 12, 64)
- …

ASCON-SIV - Synthetic Initialization Vector



Other things in Ascon Suite

- Drop-in replacements for HMAC, HKDF, and PBKDF2.
- Safely transitioning from squeezing back to absorbing.
- Reseedable PRNG using the SpongePRNG construction.
- …

And obviously:

- Ascon-128, Ascon-128a, Ascon-80pq
- Ascon-Hash, Ascon-Hasha, Ascon-Xof, Ascon-Xofa
- Ascon-Mac, Ascon-Prf, Ascon-PrfShort
- ISAP-A-128, ISAP-A-128a

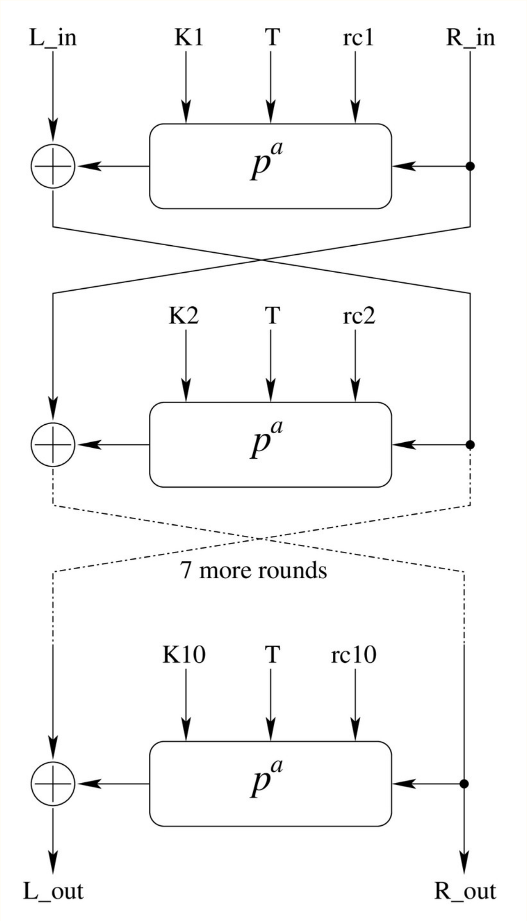
ASCON as a tweakable block cipher (yes, really)

- Tweakable block ciphers are required for memory and disk encryption.
- On-the-fly memory encryption (using tweaked versions of AES) is increasingly common in microcontrollers.

- AEAD modes are unsuitable because nonce reuse is fatal.
- SIV modes are suitable only if there is extra storage for the tag.

Ascon and Luby-Rackoff

- Luby-Rackoff is a method to turn a set of pseudorandom functions F_i into a Feistel block cipher.
- Break the 128-bit input block up into L and R halves.
- For each round, $L \leftarrow L \oplus F_i(R)$
- Swap the two halves in every round except the last.
- 10 rounds are enough for everyone!
- For Ascon: $F_i(R) = \lfloor p^a(R \,||\, K_i \,||\, T \,||\, rc_i) \rfloor_{64}$
- Reduced-round versions of p^a to improve performance.
- Or … just use Skinny-128-384+ instead.



Thank You!

https://github.com/rweather/ascon-suite

https://eprint.iacr.org/2023/391/