Blockcipher-Based Key Commitment for Nonce-Derived Schemes

Panos Kampanakis, Shai Halevi,
Nevine Ebeid, and Matt Campagna

Sr. Applied Scientist

Amazon Web Services (AWS)

Outline

- Nonce-Derived Schemes
- Key Commitment
- XAES-256-GCM with Key Commitment
- FIPS Compliance
- Data Limits
- Security Analysis of the Key Commitment
- Performance

Nonce-derived Key Schemes

- Authenticated Encryption with Additional Data (AEAD) schemes have challenges:
  - Small number of messages per key in a cloud system
  - Vulnerable to attacks if IVs are mistakenly used
  - No key commitment
- Nonce-derived key schemes address some of these issues without increasing the block size of the block cipher.

Why Nonce-Derived Schemes?

AES-GCM pain points:

IV is only 96 bits
=> exacerbates nonce-reuse problems

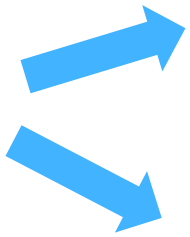
loss of message authentication + leakage of plaintext pairs where re-use occurred

Why Nonce-Derived Schemes?

AES-GCM pain points:

IV is only 96 bits
=> exacerbates nonce-reuse problems

loss of message authentication + leakage of plaintext pairs where re-use occurred

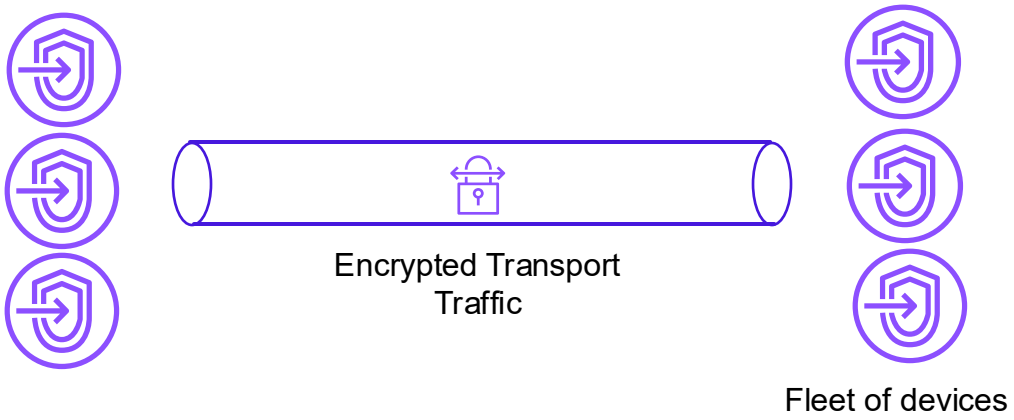


Random IVs => at most 2^{32} messages

Deterministic IVs => small fixed-length field & complexity

Random IVs and the 2^{32} invocation limit

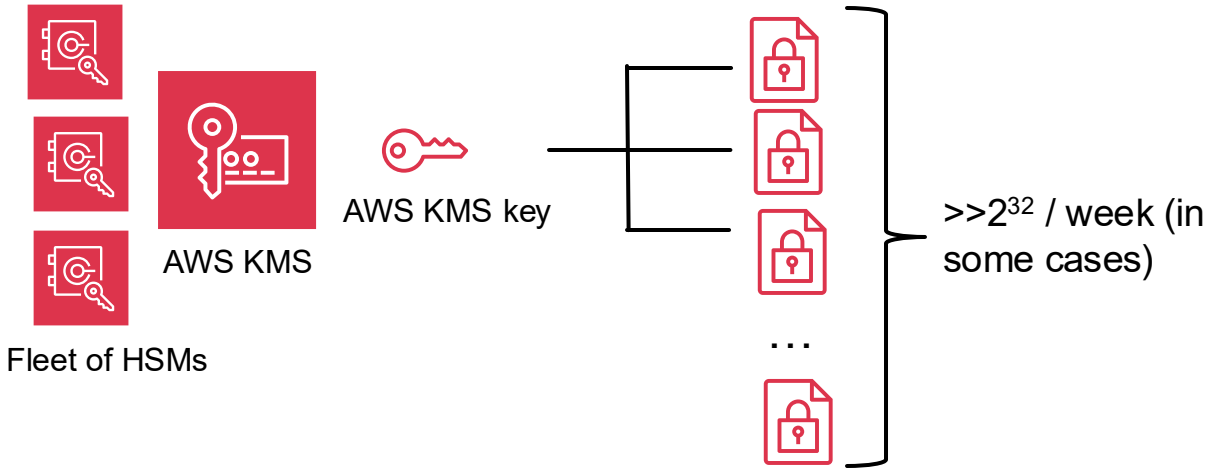
High-volume Transport Encryption for virtualized networks



Distributed transport encryption can collectively encrypt ~2^{32} messages in 2 seconds.

Re-keying every 2 seconds is not practical.

High-volume AWS KMS Encryption

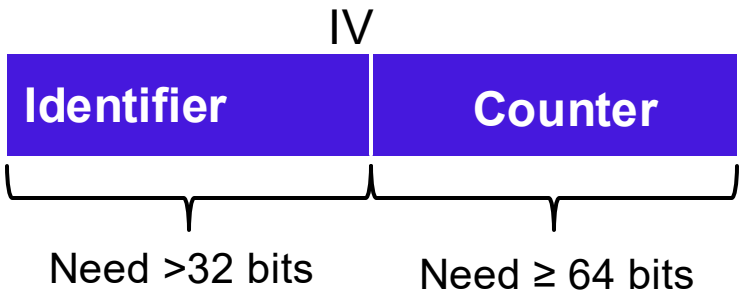


AWS Key Management Service (AWS KMS) key sometimes can encrypt 2^{32} plaintexts / week.

Rekeying weekly and managing AWS keys for thousands of accounts annually adds overhead.

Deterministic 96-bit IVs

Transport Encryption deterministic IV challenges

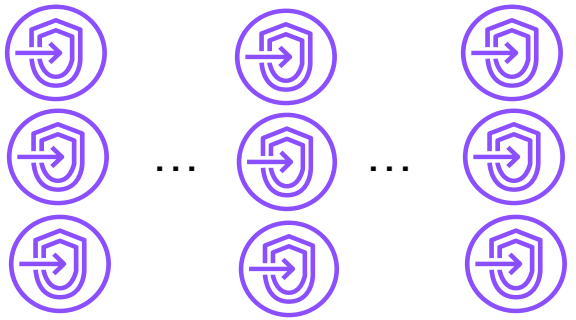


Support for large # of identifiers limits the counter size which means less messages per key.

Unique identifiers in distributed systems add complexity.

We prefer random IVs.

Transport Encryption FIPS challenges

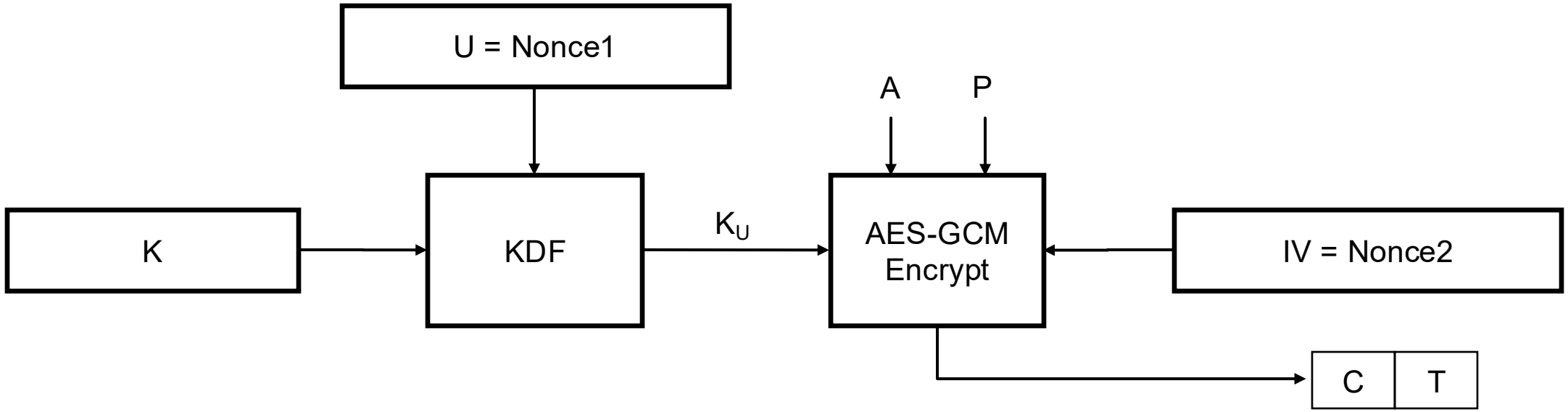


Proving IV uniqueness is hard when there are many devices to manage.

Efficient counter management adds complexity.

We prefer random IVs.

Nonce-Derived Schemes

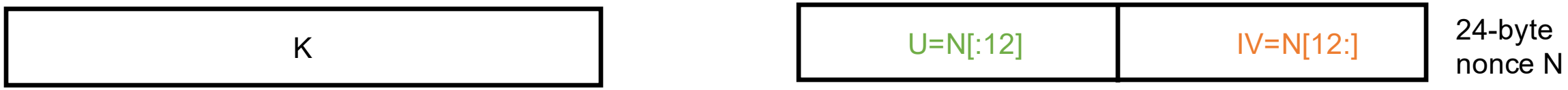


- A new key, K_U, is derived for each invocation on a new message.
- Reduces (K_U, IV) collision probability.

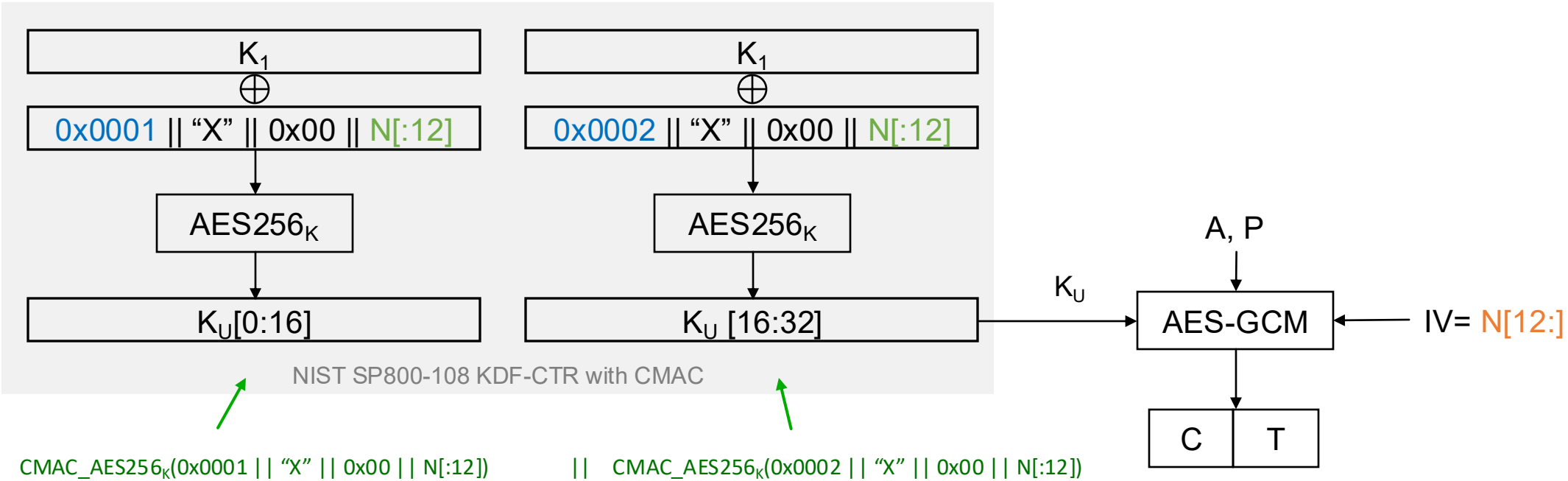
XAES-256-GCM

Filippo Valsorda

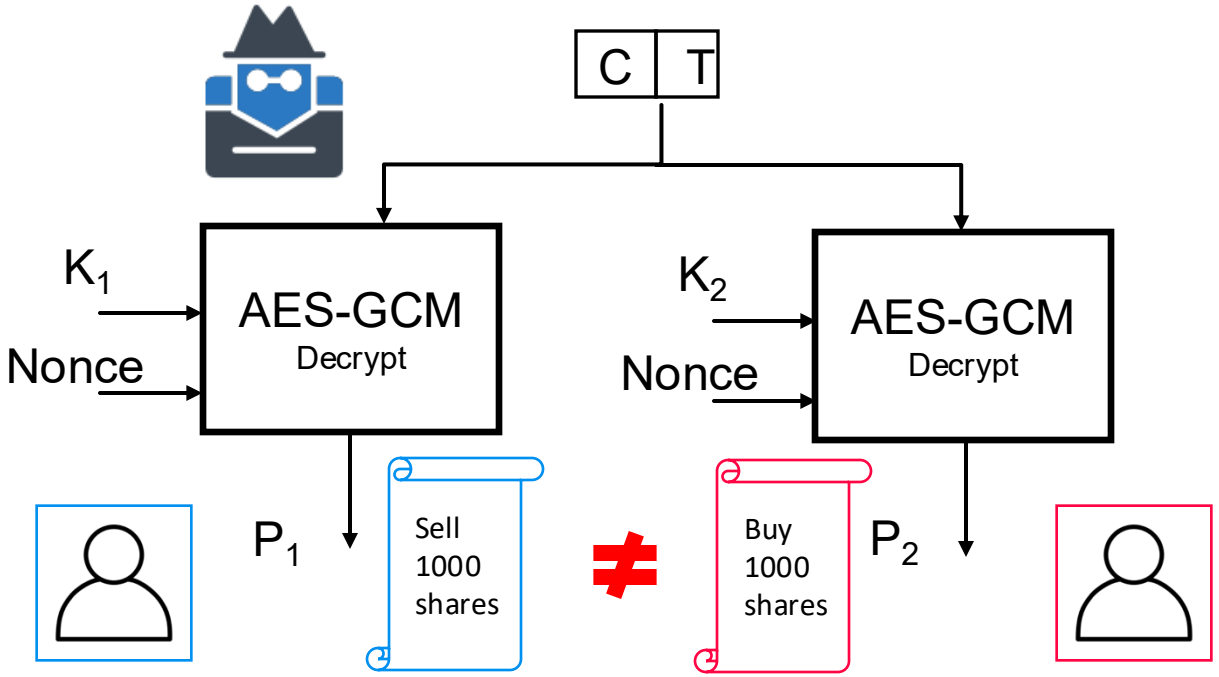
https://github.com/C2SP/C2SP/blob/main/XAES-256-GCM.md



Init: K1 := $X \times$ **AES256$_K$**(0) // multiplication by X in $\mathrm{GF}(2^n)$



Why Key Commitment?



- AES-GCM has no protection against a ciphertext decrypting to **two valid plaintexts under two different keys**.
- Solution: Bind the Key to inputs such as the Nonce and put a “stamp” on it.

Key Commitment

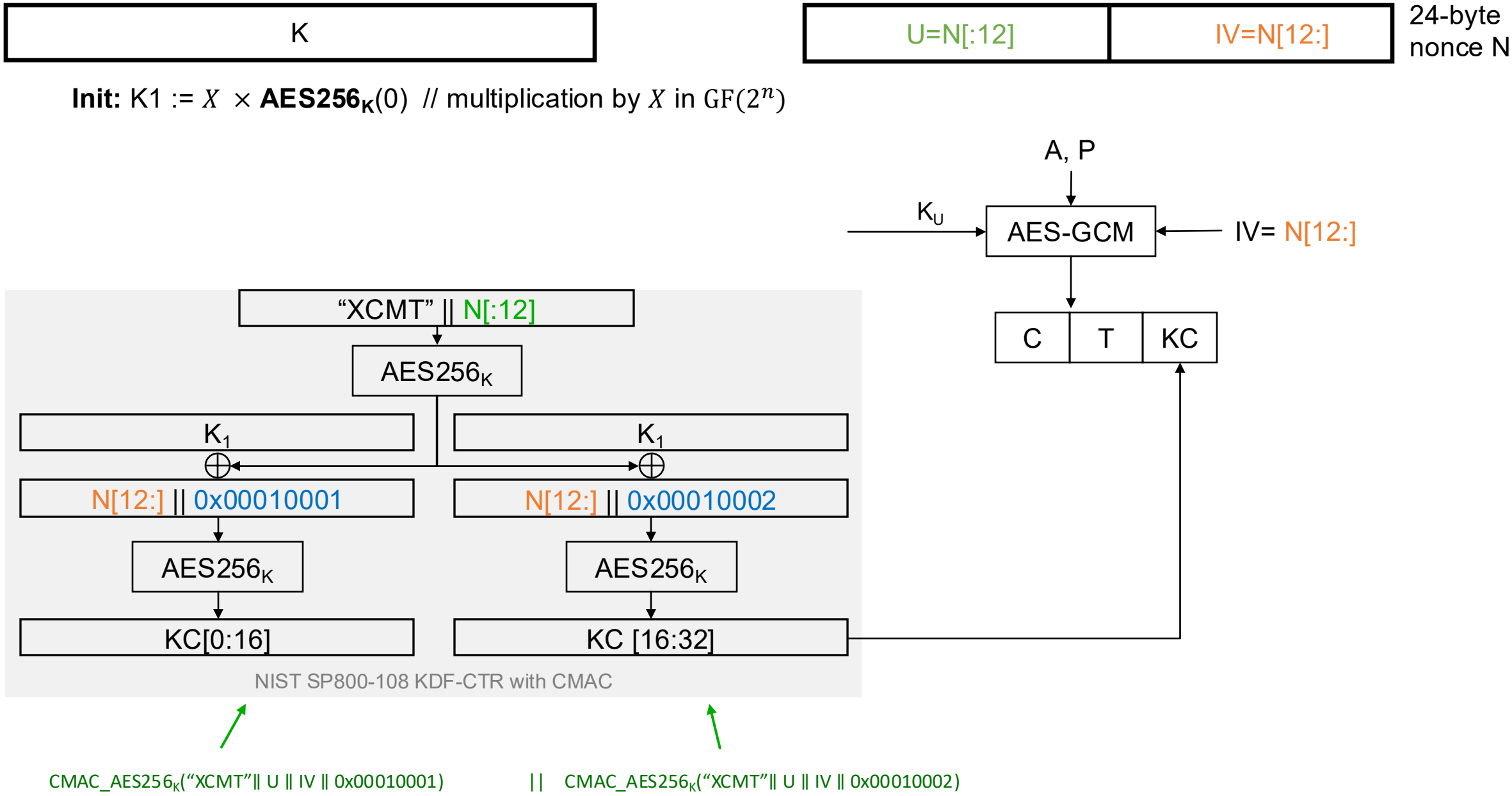
Recall: Authenticated Encryption with Additional Data (AEAD) scheme:

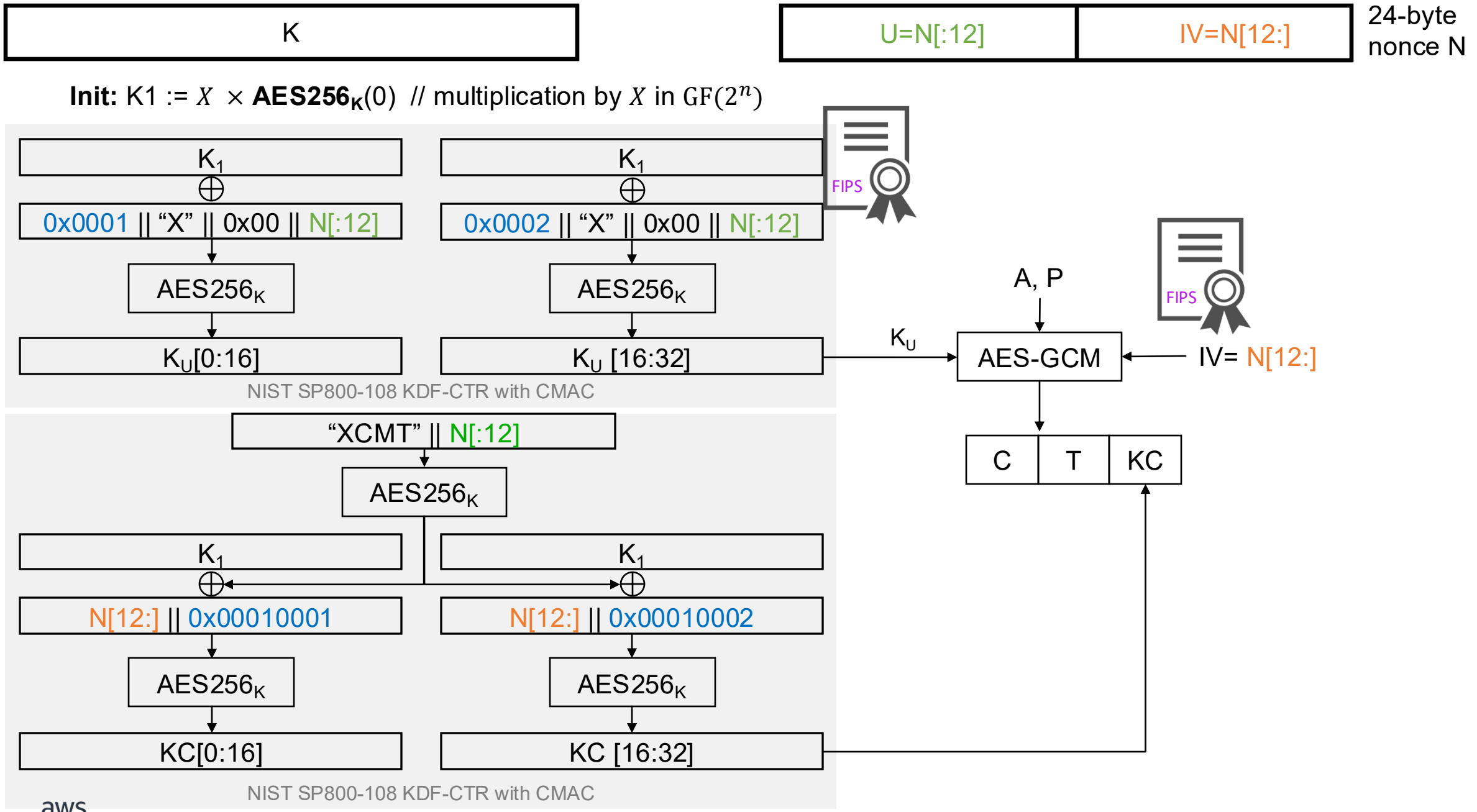
$$\text{Encrypt}(K, N, A, P) \longmapsto C, \text{Decrypt}(K, A, C) \longmapsto P/\perp$$

Key commitment notions,
you **cannot** obtain the same C when you Encrypt with these input differences:

- FOR-KC: $K \neq K'$, one N, two pairs $(A, P), (A', P')$ [Farshim et al., '17]
- CMT-1: $K \neq K'$, two tuples $(N, A, P), (N', A', P')$ [Bellare-Hoang '22]
- CMT-2: $(K, N) \neq (K', N')$, two pairs $(A, P), (A', P')$ [Takeuchi et al. '24]
- CMT-4: $(K, N, A, P) \neq (K', N', A', P')$, committing to A and P [Bellare-Hoang '22]

Not distinct in typical AEAD when N is stored with C





FIPS Compliance

- Key-Based Key Derivation Function (KBKDF) is CMAC-AES256 using *counter mode* as in NIST.SP.800-108r1-upd1.
- Derived Key is used with AES-256-GCM and a random nonce $\mathrm{IV} = N[12:24]$.

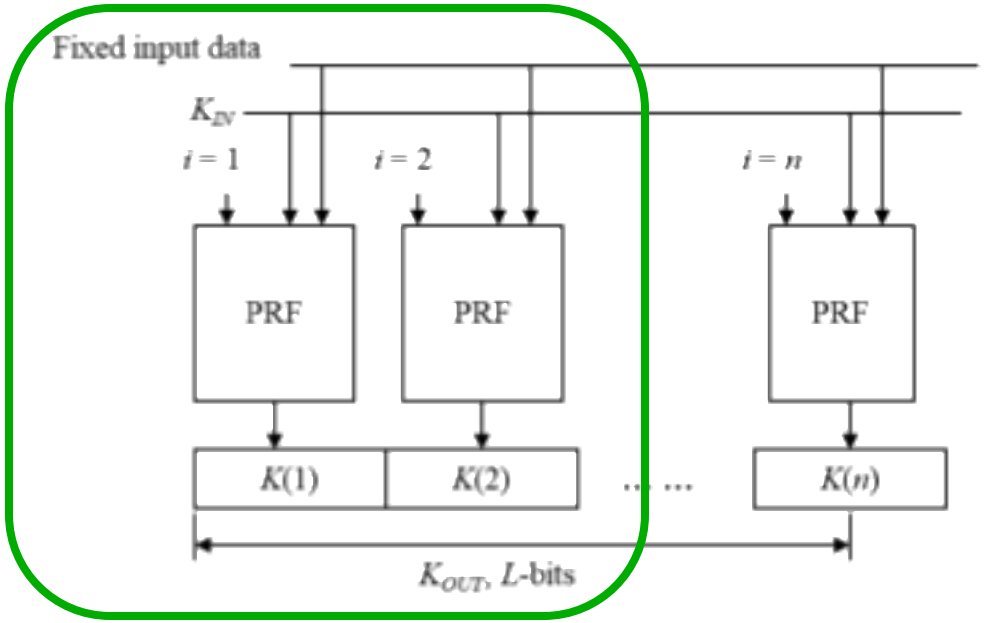


Fig. 1. KDF in Counter Mode

Data Limits

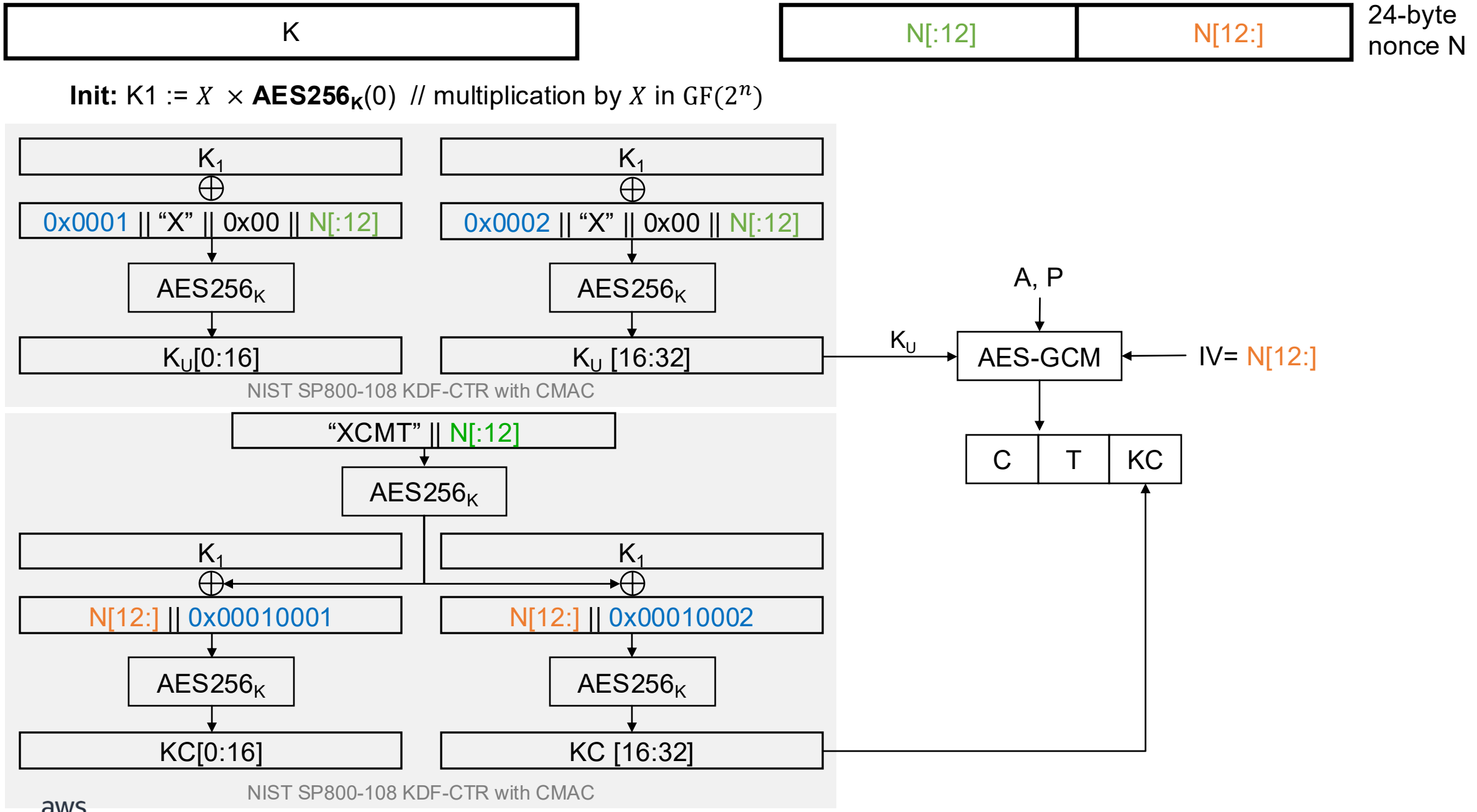
- **Number of messages:** 2^{80} with the same main key K and with random nonces
  before reaching nonce collision probability of 2^{-32}.
  Compare with AES-GCM's limit of 2^{32} messages with random IVs.

- **Max message length:** with 2^{80} messages per main key, each message be practically at the same 64GB limit ($2^{36} - 32$ bytes) per message as in AES-GCM.
  If the application is conservative and wants to enforce no-more-than-X-bytes per key, then no single message encrypted under XAES should be larger than X / 7 bytes (see manuscript).

Security Analysis of the Key Commitment

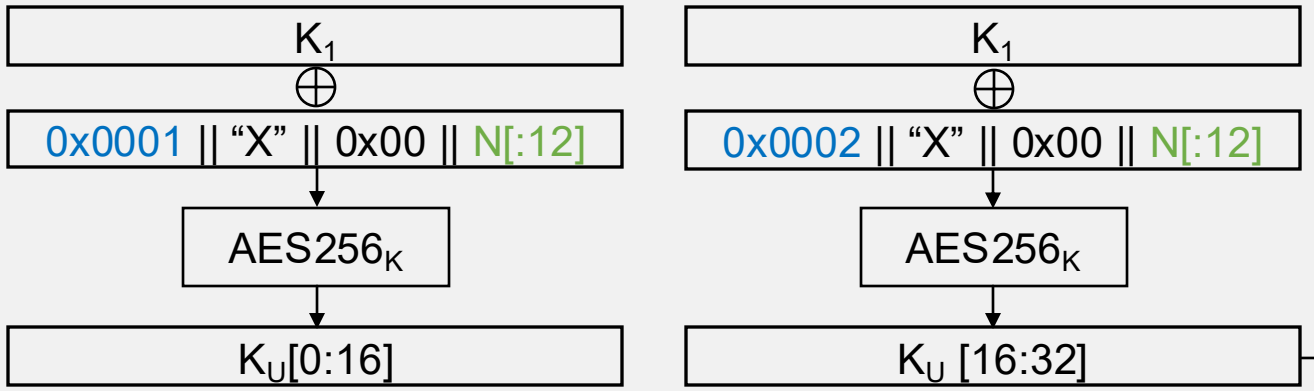
Complexity of Finding Collisions

- FOR-KC ($K \neq K'$, one N; nonce is part of ciphertext, so constant)
  - for n-bit blockcipher, queries $q \ll 2^n$, complexity $\approx O(q)/2^n$
    - Proof shows probability of collision $\approx O(q/2^n + q^2/2^{2n})$

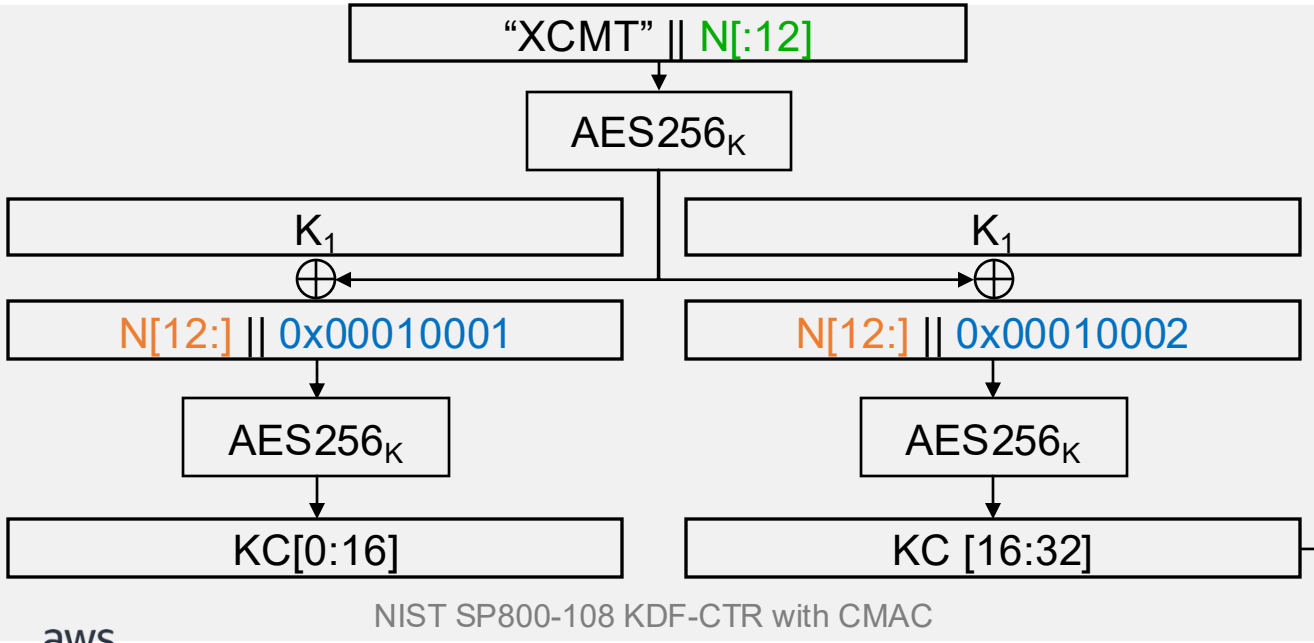


24-byte nonce N

Init: K1 := $X \times$ **AES256$_K$**(0) // multiplication by X in $\mathrm{GF}(2^n)$

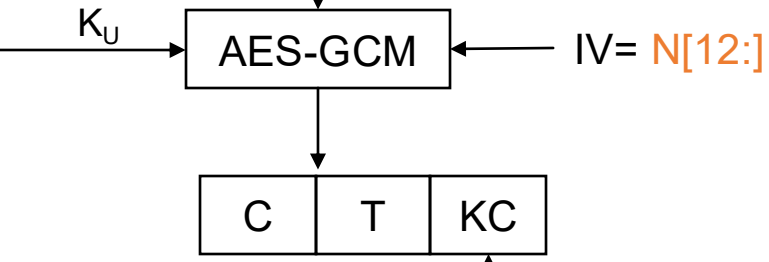


NIST SP800-108 KDF-CTR with CMAC



NIST SP800-108 KDF-CTR with CMAC

		AES-GCM	XAES		
				Overhead	
Init		0.136	0.054	-0.082	-60%
Encrypt	32B	0.087	0.247	0.160	184%
	128B	0.096	0.258	0.162	169%
	256B	0.123	0.285	0.162	132%
	512B	0.176	0.333	0.158	90%
	1024B	0.220	0.377	0.157	71%



Init of AES-GCM:

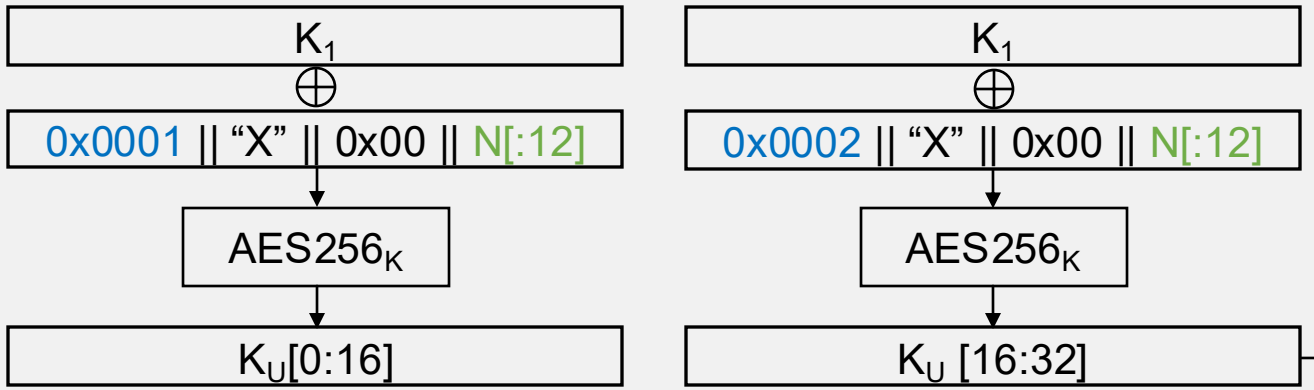
a) a key scheduling

b) an encryption of an all-zeroes input

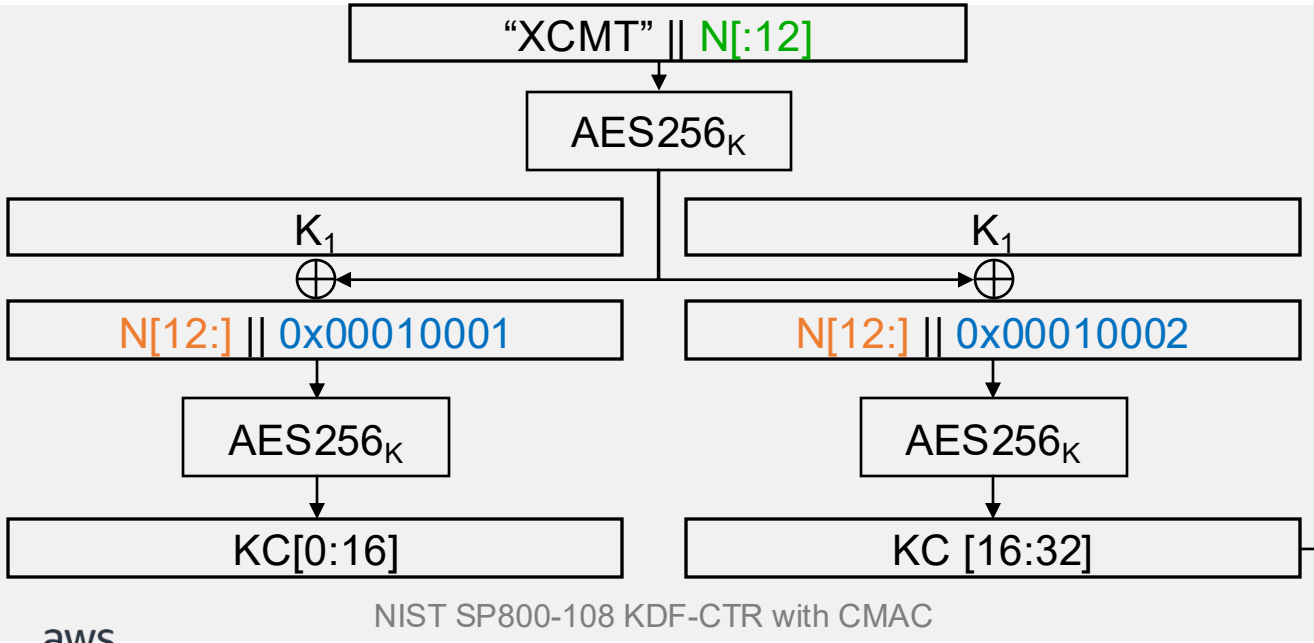
c) A precomputation of the GHASH constants

24-byte nonce N

Init: K1 := $X \times$ **AES256$_K$**(0) // multiplication by X in $\mathrm{GF}(2^n)$

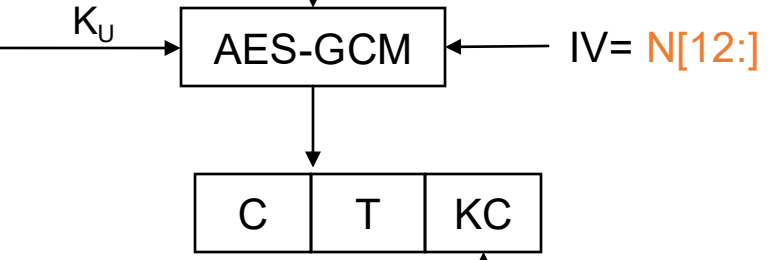


NIST SP800-108 KDF-CTR with CMAC



NIST SP800-108 KDF-CTR with CMAC

		AES-GCM	XAES		
				Overhead	
Init		0.136	0.054	-0.082	-60%
Encrypt	32B	0.087	0.247	0.160	184%
	128B	0.096	0.258	0.162	169%
	256B	0.123	0.285	0.162	132%
	512B	0.176	0.333	0.158	90%
	1024B	0.220	0.377	0.157	71%



Init of AES-GCM:

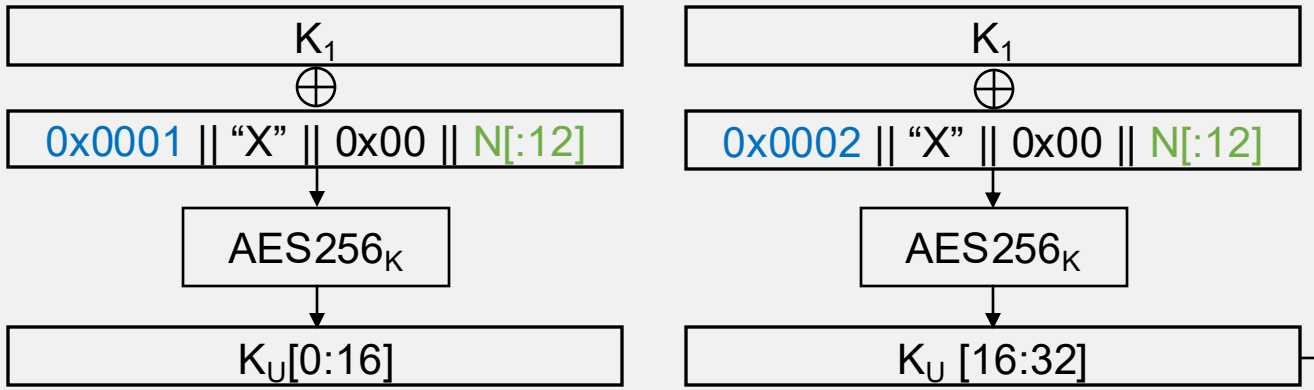
a) a key scheduling

b) an encryption of an all-zeroes input

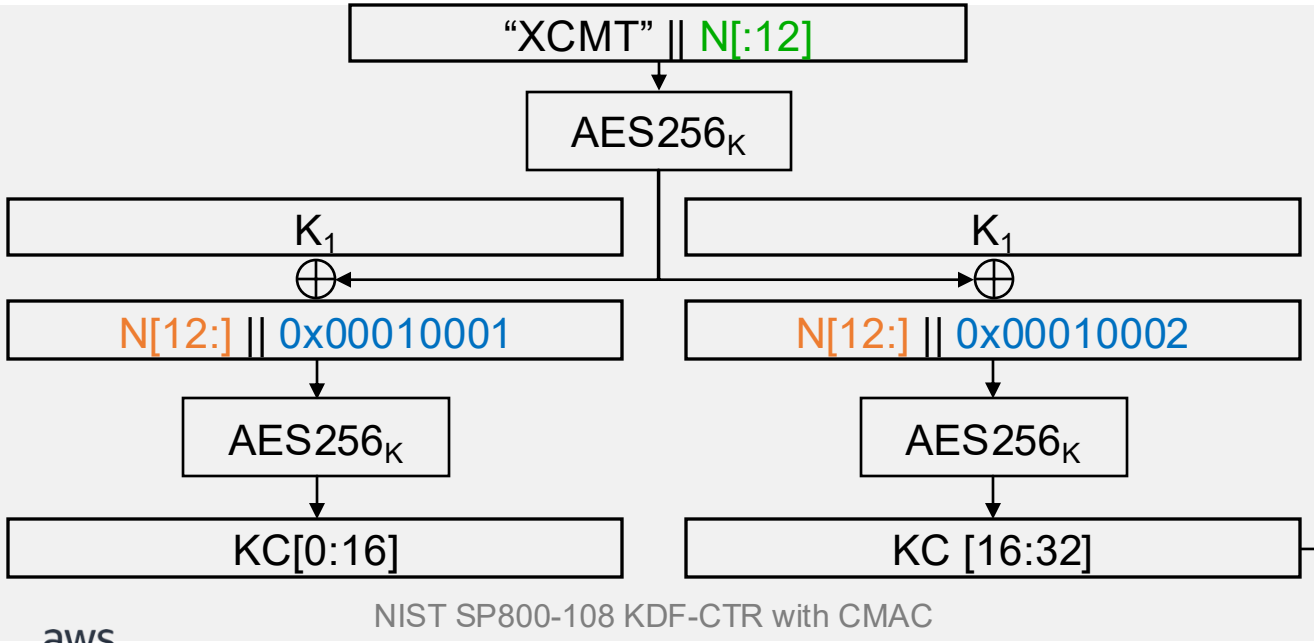
c) A precomputation of the GHASH constants

24-byte nonce N

Init: K1 := $X \times$ **AES256$_K$**(0) // multiplication by X in $\mathrm{GF}(2^n)$

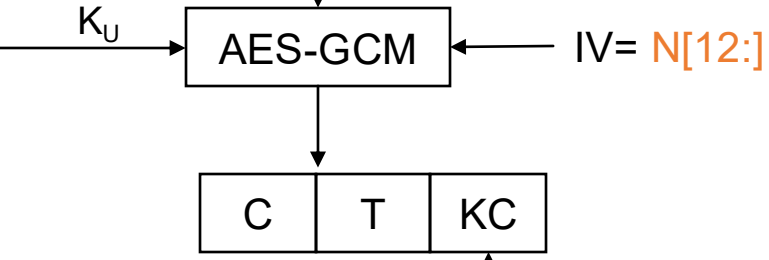


NIST SP800-108 KDF-CTR with CMAC



NIST SP800-108 KDF-CTR with CMAC

		AES-GCM	XAES		
				Overhead	
Init		0.136	0.054	-0.082	-60%
Encrypt	32B	0.087	0.247	0.160	184%
	128B	0.096	0.258	0.162	169%
	256B	0.123	0.285	0.162	132%
	512B	0.176	0.333	0.158	90%
	1024B	0.220	0.377	0.157	71%



Init of AES-GCM:

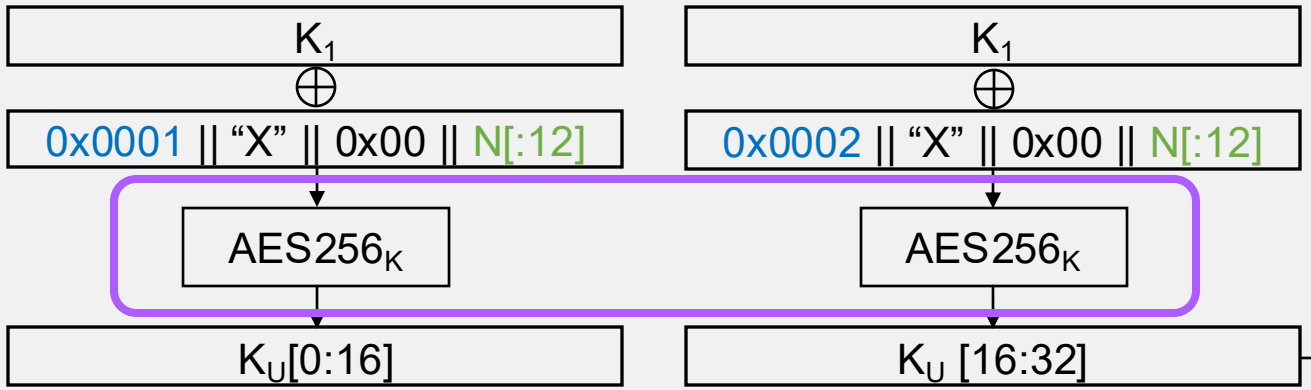
a) a key scheduling

b) an encryption of an all-zeroes input

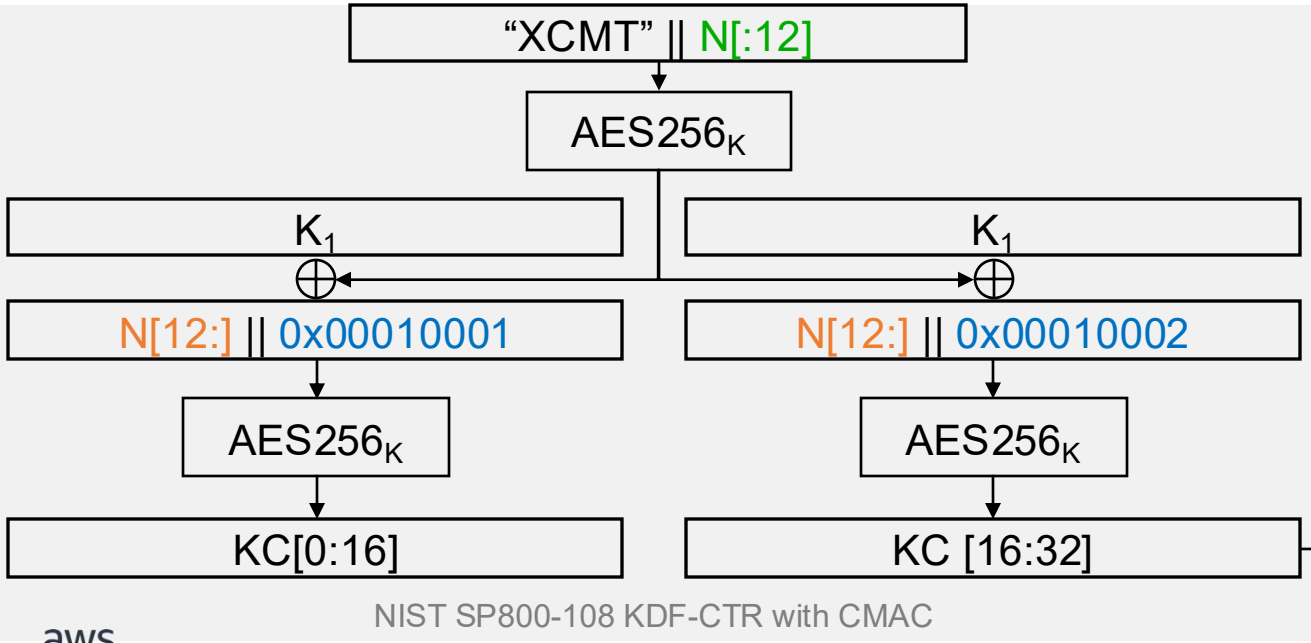
c) A precomputation of the GHASH constants

24-byte nonce N

Init: K1 := $X \times$ **AES256$_K$**(0) // multiplication by X in $\mathrm{GF}(2^n)$

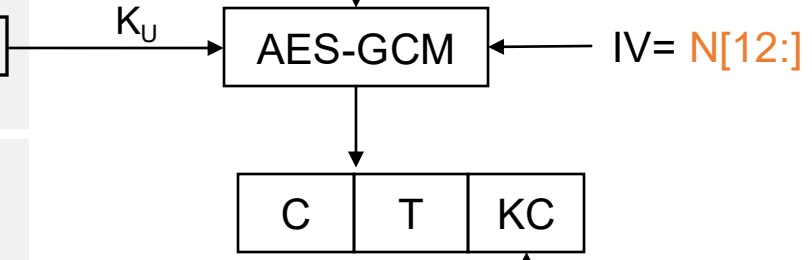


NIST SP800-108 KDF-CTR with CMAC



NIST SP800-108 KDF-CTR with CMAC

		AES-GCM	XAES		
				Overhead	
Init		0.136	0.054	-0.082	-60%
Encrypt	32B	0.087	0.247	0.160	184%
	128B	0.096	0.258	0.162	169%
	256B	0.123	0.285	0.162	132%
	512B	0.176	0.333	0.158	90%
	1024B	0.220	0.377	0.157	71%



Init of AES-GCM:

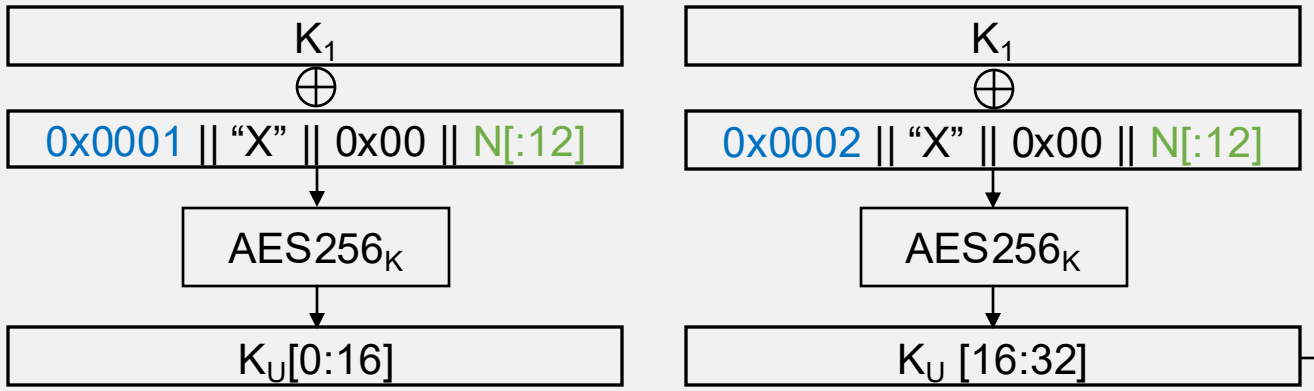
a) a key scheduling

b) an encryption of an all-zeroes input

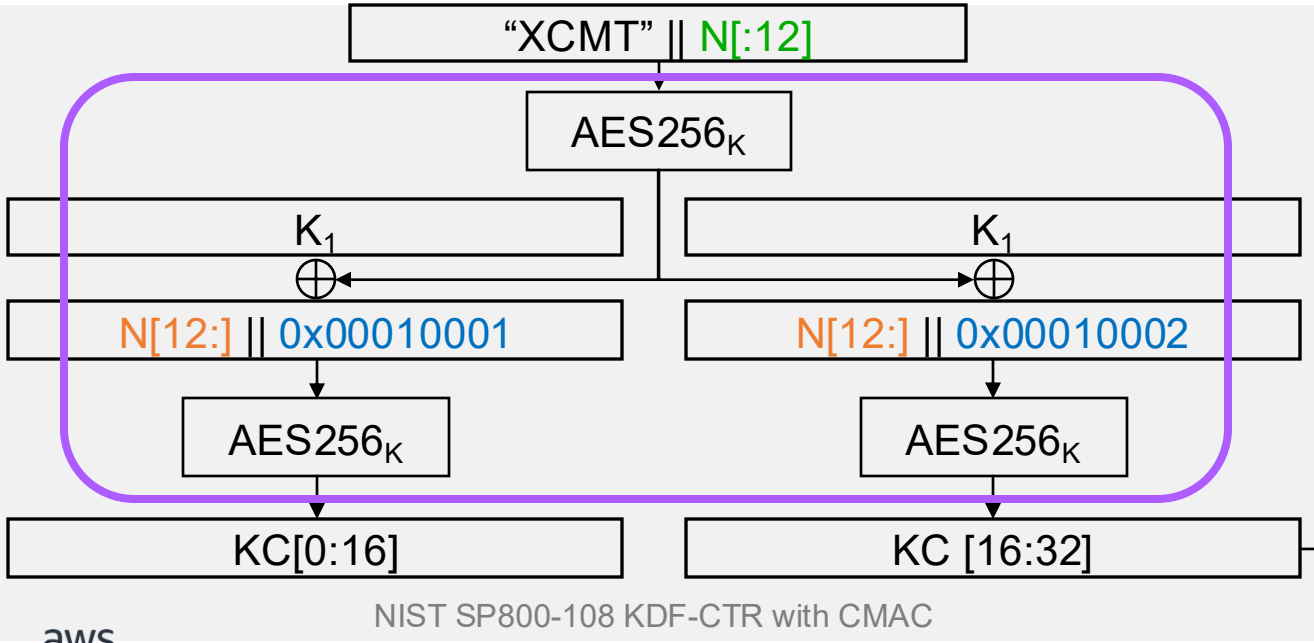
c) A precomputation of the GHASH constants

24-byte nonce N

Init: K1 := $X \times$ **AES256$_K$**(0) // multiplication by X in $\mathrm{GF}(2^n)$

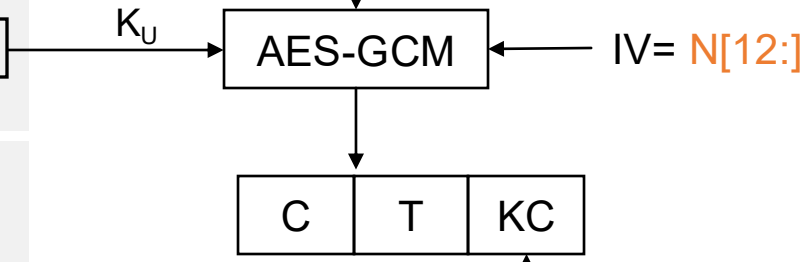


NIST SP800-108 KDF-CTR with CMAC



NIST SP800-108 KDF-CTR with CMAC

		AES-GCM	XAES		
				Overhead	
Init		0.136	0.054	-0.082	-60%
Encrypt	32B	0.087	0.247	0.160	184%
	128B	0.096	0.258	0.162	169%
	256B	0.123	0.285	0.162	132%
	512B	0.176	0.333	0.158	90%
	1024B	0.220	0.377	0.157	71%



Init of AES-GCM:
a) a key scheduling
b) an encryption of an all-zeroes input
c) A precomputation of the GHASH constants

KC-XAES		
	Overhead	
0.055	-0.082	-60%
0.303	0.216	248%
0.313	0.218	227%
0.341	0.218	178%
0.388	0.213	121%
0.429	0.209	95%

Performance

Intel® Xeon®
Platinum 8375C
(x86-64 processor)

		AES-GCM	XAES		KC-XAES	
Init		0.134	0.053	-60%	0.054	-60%
Encrypt	32B	0.087	0.244	180%	0.296	240%
	1024B	0.219	0.376	72%	0.432	97%
	16KB	1.448	1.620	12%	1.686	16%
	1MB	91.242	90.738	-1%	90.128	-1%
Decrypt	32B	0.094	0.242	157%	0.298	217%
	1024B	0.225	0.373	66%	0.431	92%
	16KB	1.457	1.618	11%	1.671	15%
	1MB	85.403	86.326	1%	85.932	1%

Performance

Intel® Xeon® Platinum 8375C (x86-64 processor)

		AES-GCM	XAES		KC-XAES	
Init		0.134	0.053	-60%	0.054	-60%
Encrypt	32B	0.087	0.244	180%	0.296	240%
	1024B	0.219	0.376	72%	0.432	97%
	16KB	1.448	1.620	12%	1.686	16%
	1MB	91.242	90.738	-1%	90.128	-1%
Decrypt	32B	0.094	0.242	157%	0.298	217%
	1024B	0.225	0.373	66%	0.431	92%
	16KB	1.457	1.618	11%	1.671	15%
	1MB	85.403	86.326	1%	85.932	1%

Init of HMAC-AES:
2 SHA-256 calculations

		AES-GCM	HMAC-AES		KC-HMAC-AES	
Init		0.136	0.142	4%	0.142	4%
Encrypt	32B	0.086	0.357	314%	0.501	481%
	1024B	0.219	0.487	122%	0.637	191%
	16KB	1.462	1.718	17%	1.882	29%
	1MB	87.231	86.642	-1%	88.108	1%
Decrypt	32B	0.093	0.361	289%	0.506	445%
	1024B	0.225	0.493	119%	0.640	184%
	16KB	1.458	1.726	18%	1.879	29%
	1MB	85.334	85.224	0%	85.998	1%

Performance

Intel® Xeon® Platinum 8375C (x86-64 processor)

		AES-GCM	XAES		KC-XAES	
Init		0.134	0.053	-60%	0.054	-60%
Encrypt	32B	0.087	0.244	180%	0.296	240%
	1024B	0.219	0.376	72%	0.432	97%
	16KB	1.448	1.620	12%	1.686	16%
	1MB	91.242	90.738	-1%	90.128	-1%
Decrypt	32B	0.094	0.242	157%	0.298	217%
	1024B	0.225	0.373	66%	0.431	92%
	16KB	1.457	1.618	11%	1.671	15%
	1MB	85.403	86.326	1%	85.932	1%

Init of HMAC-AES:
2 SHA-256 calculations

		AES-GCM	HMAC-AES		KC-HMAC-AES	
Init		0.136	0.142	4%	0.142	4%
Encrypt	32B	0.086	0.357	314%	0.501	481%
	1024B	0.219	0.487	122%	0.637	191%
	16KB	1.462	1.718	17%	1.882	29%
	1MB	87.231	86.642	-1%	88.108	1%
Decrypt	32B	0.093	0.361	289%	0.506	445%
	1024B	0.225	0.493	119%	0.640	184%
	16KB	1.458	1.726	18%	1.879	29%
	1MB	85.334	85.224	0%	85.998	1%

Performance

AWS Graviton4 - Neoverse V2, (AArch64 processor)

		AES-GCM	XAES		KC-XAES	
Init		0.082	0.041	-50%	0.042	-49%
Encrypt	32B	0.089	0.209	135%	0.251	182%
	1024B	0.250	0.411	64%	0.454	82%
	16KB	2.644	2.767	5%	2.810	6%
	1MB	164.182	164.404	0%	164.508	0%
Decrypt	32B	0.096	0.217	126%	0.258	169%
	1024B	0.254	0.415	63%	0.457	80%
	16KB	2.587	2.711	5%	2.752	6%
	1MB	160.418	160.292	0%	160.554	0%

		AES-GCM	HMAC-AES		KC-HMAC-AES	
Init		0.083	0.091	10%	0.091	10%
Encrypt	32B	0.089	0.285	220%	0.395	344%
	1024B	0.250	0.485	94%	0.594	138%
	16KB	2.643	2.841	7%	2.951	12%
	1MB	164.573	164.313	0%	164.759	0%
Decrypt	32B	0.096	0.290	202%	0.399	316%
	1024B	0.254	0.487	92%	0.595	134%
	16KB	2.587	2.784	8%	2.892	12%
	1MB	160.767	160.429	0%	160.577	0%

Thank you!