Misuse-resistant crypto for JOSE/JWT

Neil Madden

OAuth Security Workshop, 2018

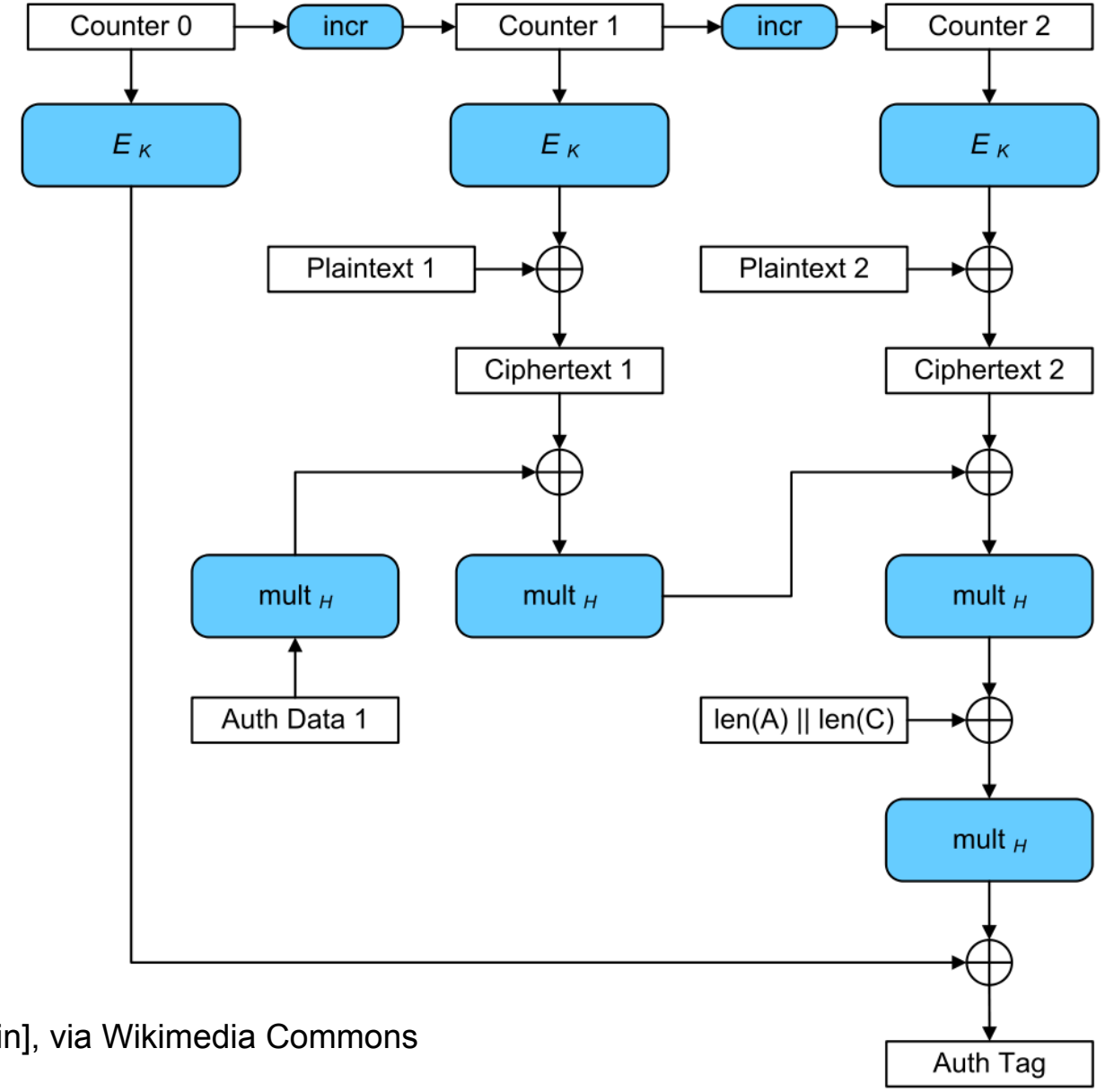
JOSE Content Encryption Methods

- Provide *authenticated encryption*
- AES-CBC with HMAC-SHA2
  - Requires random 128-bit IV
  - Must be unpredictable
- AES-GCM
  - Requires 96-bit *nonce*
  - Nonce can be a simple counter
- Most modern textbooks would recommend GCM: fast, dedicated *AEAD* mode, parallel

GCM

- *Galois Counter Mode*
- CTR-mode for privacy
- GHASH for authentication

- Simple! :-)



By NIST (http://en.wikipedia.org/wiki/File:GCM.png) [Public domain], via Wikimedia Commons

What happens if you reuse a nonce?

- NIST SP-800-38D on GCM:

  “An important caution to the use of GCM is that a breach of the requirement in Sec. 8 for the uniqueness of the initialization strings ***may compromise the security assurance almost entirely****”*

  “In practice, this requirement is almost as important as the secrecy of the key.”

Nonce reuse attacks on GCM

- If a nonce is reused for the same key just once, results are catastrophic:
  - Recover information about encrypted plaintexts
  - Recover authentication sub-key
  - Produce arbitrary forgeries of associated data
  - Can often produce forgeries of encrypted ciphertext too:

`{“sub”: “peter”, … }` ➡️ `{“sub”: “admin”, … }`

Nonce reuse in reality

- KRACK attacks against WPA2
- Forced nonce reuse by weaknesses in protocol

  “If the victim uses […] GCMP encryption protocol, instead of AES-CCMP, the impact is especially catastrophic. **Against these encryption protocols, nonce reuse enables an adversary to not only decrypt, but also to forge and inject packets**.” (krackattacks.com)

How to avoid?

- NIST recommends either:
  1. Use random IV
  2. Use deterministic counter
- Both can be problematic
- Failures of RNG, e.g. SSH keys generated too soon on first boot, Android SecureRandom failures leading to BitCoin wallet compromise, IoT devices
- Counters are hard to synchronise across servers
- Only 96-bit IV

Is CBC/HMAC better?

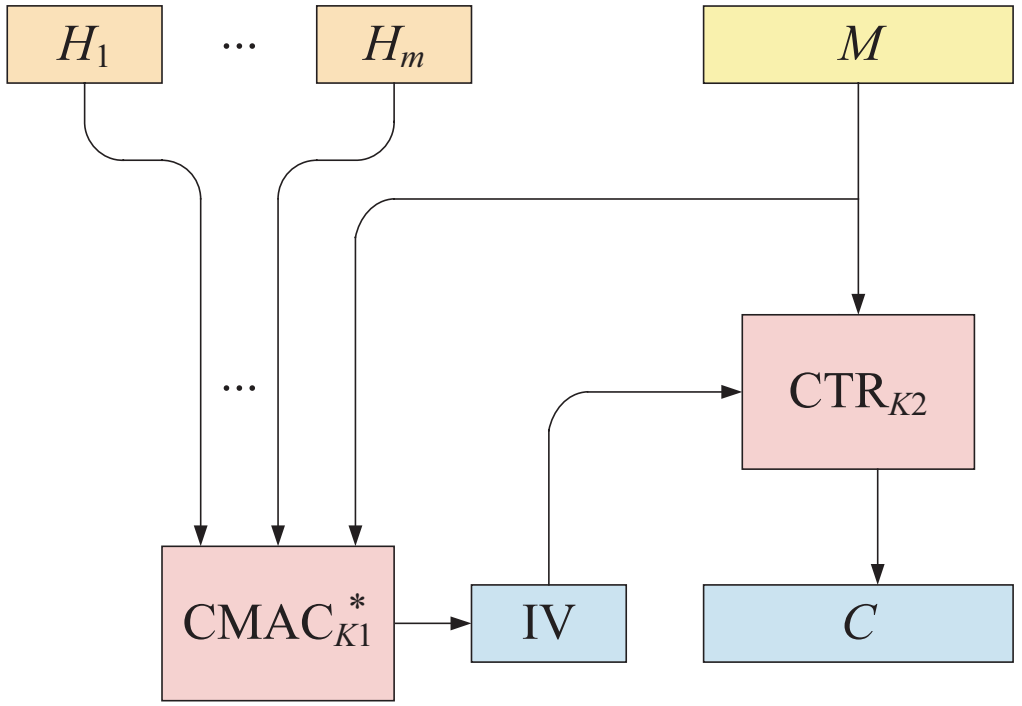
- Yes and no
- CBC has its own problems:
  - Padding oracle attacks
  - If IV is *predictable* then plaintext can be recovered (BEAST)
  - Worse security bounds than CTR mode
- Unpredictable IV is a more strict requirement than non-repeating nonce
- HMAC prevents some of these attacks, but not necessarily all – e.g., if attacker can inject plaintext via logon username

A safer alternative

- *Misuse Resistant Authenticated Encryption* (MRAE)
- Developed by Rogaway & Shrimpton while analysing AES KeyWrap
- When unique nonce used then has same properties as GCM, CBC+HMAC, etc: authenticated encryption
- If nonce is reused then loses a minimum amount of security:
  - Authenticity is not compromised at all
  - Privacy only (slightly) compromised if the the *same* message is encrypted with *same key, nonce, and associated data*.

Synthetic IV (SIV)

- Achieves MRAE
- Basic idea: use MAC of associated data (header) and plaintext as the IV for encryption
- AES-SIV: MAC is AES-CMAC, encryption is AES-CTR
- RFC 5297



From http://web.cs.ucdavis.edu/~rogaway/papers/siv.pdf

Advantages

- Simple and provably secure scheme
- Original AES-SIV only uses AES in encrypt direction: efficient on constrained devices (similar to AES-CCM)
- Can substitute other MACs and ciphers (with some caveats)
  - For instance, HMAC, PMAC (parallel), Blake2 etc
  - Other (stream) ciphers, e.g. XSalsa20/XChaCha20
  - About to be published by IRTF (CFRG): AES-GCM-SIV
- Versatile: content encryption, key-wrapping, deterministic encrypt
- Subjective: Well-respected mode amongst cryptographers

Disadvantages

- Must make two passes over the input
- Cannot be streamed
- If no unique value in header then completely deterministic
- Not great for low-entropy inputs (e.g., passwords)

- On the other hand:
  - Many JOSE inputs are small (JWTs)
  - Decryption cannot (safely) be streamed in any case
  - Few encryption schemes *are* secure for passwords

Proposed new modes

“enc”	“alg”
`A128SIV`	`A128SIVKW`
`A128SIV-HS256`	`A128SIVKW-HS256`
`A192SIV-HS384`	`A192SIVKW-HS384`
`A256SIV-HS512`	`A256SIVKW-HS512`

- JWE IV *should* be a random 128-bit value
- Fixed IV for -KW variants

Code (Java + Bouncy Castle)

```
byte[] iv = secureRandomBytes(16);
Mac cmac = Mac.getInstance("AESCMAC");
cmac.init(macKey);
cmac.update(ascii(b64url(header) + ".." + b64url(iv) + '.'));
byte[] siv = cmac.doFinal(plaintext);

Cipher aes = Cipher.getInstance("AES/CTR/NoPadding");
aes.init(Cipher.ENCRYPT_MODE, encKey,
        new IvParameterSpec(siv));
byte[] ciphertext = aes.doFinal(plaintext);
```

Code - Key Wrap

```
byte[] iv = secureRandomBytes(16);
Mac cmac = Mac.getInstance(“AESCMAC”);
cmac.init(macKey);
cmac.update(ascii(“A128SIV..."));
byte[] siv = cmac.doFinal(cek);

Cipher aes = Cipher.getInstance(“AES/CTR/NoPadding”);
aes.init(Cipher.ENCRYPT_MODE, encKey,
         new IvParameterSpec(siv));
byte[] ciphertext = aes.doFinal(cek);
```

Misuse of other JOSE algorithms?

- Signatures mostly ok apart from ES ones
  - Nonce reuse for NIST ECDSA led to Playstation 3 hack, Bitcoin theft, etc.
  - Use RFC 6979 or EdDSA
- Public key encryption
  - Less of a problem?
  - *Hedged PKE*
- Password-based encryption can be hedged by increasing rounds (maybe consider memory-hard hash algorithms: Scrypt, Argon2)

Internet Draft

- https://tools.ietf.org/html/draft-madden-jose-siv-mode-02
- 03 coming soon…
- What variants to support?
  - Just AES-SIV?
  - HMAC variants?
  - AES-GCM-SIV?
  - A non-AES alternative (e.g., XChaCha20-HS384-SIV)?
- Would OAUTH WG adopt this?

Thank You

FORGEROCK