Symmetric Cryptography Standards

Alexandre Duc

Zoo

Symmetric cryptography is a zoo. There are dozens of primitives and not all are recommended. How do you select which one to use ?

Block Ciphers

- Dozens of designs of block ciphers have been proposed in the academic literature.
- Most of them have not been thoroughly analyzed.
- A large part of them suffers from theoretical and/or practical weaknesses.
- Some of them are “Internet”-standards.

ECRYPT-CSA Recommendation

- https://www.ecrypt.eu.org/csa/documents/D5.4-FinalAlgKeySizeProt.pdf
- European cryptographic researchers that came up with recommendations for choosing cryptography (key sizes, algorithms, ...)
- Still up to date : 2018
- Other document in French from ANSSI (2021) : https://www.ssi.gouv.fr/uploads/2021/03/anssi-guide-selection_crypto-1.0.pdf

Which Block Ciphers would you Use

Primitive
Blowfish
AES
DES
Serpent
Triple-DES
Camelia

Block Cipher Recommendations

Primitive	Legacy	Future
AES	✓	✓
Camelia	✓	✓
Serpent	✓	✓
Triple-DES	✓	x
Blowfish	✓	x
DES	x	x

source of recommendations : https://www.ecrypt.eu.org/csa/documents/D5.4-FinalAlgKeySizeProt.pdf

Triple-DES

- **Block size** of **64 bits**, **key size** of **112 bits** (two-key version) and of **168 bits** (three-keys version).
- Standardized in NIST SP800-67 Revision 1 and in ISO/IEC 18033-3 :2010.
- Effective security lower than key security (2^{112} bits for three-key version)
- **Weak points** : old design, small block size, extremely slow
- Still widely deployed in the financial industry
- **Avoid it** !

AES

- **Block size** of **128 bits**, **key size** of **128, 192** and **256 bits**.
- Standardized in NIST FIPS PUB 197 and ISO/IEC 18033-3 :2010.
- **Strong points** : good security, fast on most platforms, large block size, strong design process
- **Weak point** : the encryption and decryption algorithms are rather different

Camellia

- **Block size** of **128 bits**, **key size** of **128, 192** and **256 bits**.
- Designed by researchers from Mitsubishi and NTT (Japan)
- Standardized by CRYPTREC, NESSIE and in RFC 3713 and ISO/IEC 18033-3 :2010.
- Recommended for use with S/MIME, XML, TLS/SSL, IPSec, OpenPGP, etc.
- **Strong points** : good security, reasonably fast on most platforms, large block size, not a US-approved design
- **Weak points** : less efficient that AES

Serpent

- **Block size** of **128 bits**, **key size** of **128, 192** and **256 bits**.
- Designed by Anderson, Biham, Knudsen.
- Ranked second in the AES competition.
- **Strong points** : believed to have a bigger security margin than AES.
- **Weak points** : much slower than AES.

Block Cipher Block Size

Block Size

Recommended constructions have all **block sizes** of 128 bits. Smaller block sizes should be used only under very specific and controlled situations.

Question

Why is a small block size problematic ?

Padding

- Block ciphers and some mode of operation require the plaintext to have a size **multiple of the block size**.
- This requires some **padding**
- The same paddings can also be used for MACs.

Question

How would you pad a plaintext that is not a multiple of a blocksize ?

Padding Standards

- **Bit Padding** : 10* (defined in NIST 800-38a).
- **Byte Padding : ANSI X9.23** : arbitrary data. Last byte is size of padding.
- **Byte Padding : PKCS#7** : if k bytes are missing, add k bytes with value k.
- PKCS#5 padding is a subset of PKCS#7 for small block sizes.
- They are all secure but beware of **padding oracle attacks**.

Hash Functions

- A lot of hash function designs have been proposed.
- Several designs have been badly **broken** (including MD2, MD4, MD5, SHA1 etc.)

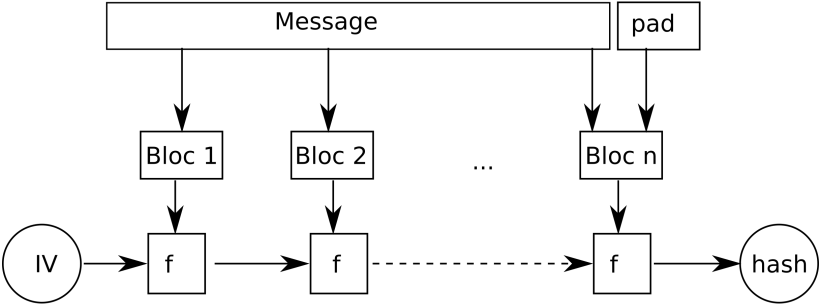
Hash Functions Recommendations

Primitive	Output length	Legacy	Future
SHA-2	256, 384, 512, 512/256	✓	✓
SHA-3	256, 384, 512	✓	✓
SHA-3	SHAKE128, SHAKE256	✓	✓
Whirlpool	512	✓	✓
Blake (1 or 2)	256, 384, 512	✓	✓
SHA-2	224, 512/224	✓	x
SHA-3	224	✓	x
RIPEMD-160	160	✓	x
MD5	128	x	x
SHA1	160	x	x

source of recommendations : `https://www.ecrypt.eu.org/csa/documents/D5.4-FinalAlgKeySizeProt.pdf`

Merkle-Damgård Construction

The Merkle-Damgård construction allows to transfom a **compression function** into a hash function.

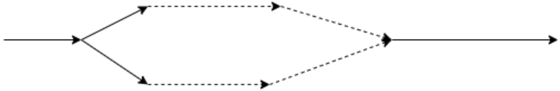


The Merkle-Damgård padding is a 1 followed by 0s follwed by the length of the message encoded over 64 bits.

SHA-1

- Standardized in NIST FIPS 180.
- Output digest of 160 bits
- SHA-1 is **broken** : one can find **collisions** within few seconds on a laptop.
- Should be avoided by all means in new applications.

Shambles Attack



- 2020 Leurent and Peyrin : Improved attack on SHA-1 : **Chosen-prefix collision attack**
- Breaks certificates, PGP, TLS, SSH
- For certificates : possible to find keys k_A and k_B such that $H(\text{Certif}(\text{Alice}, k_A)) = H(\text{Certif}(\text{Bob}, k_B))$.
- Bob asks to sign his certificate. It is a valid certificate for Alice.

SHA-2

- Standardized in NIST FIPS 180.
- Two algorithms : SHA-256 and SHA-512.
- Possible outputs digest are 224 (SHA-256 or SHA-512 truncated), 256 (normal or SHA-512 truncated), 384 (SHA-512 truncated) and 512 bits.
- Also based on the Merkle-Damgård construction.
- Unbroken for the moment.

Merkle-Damgård and Length Extension Attacks

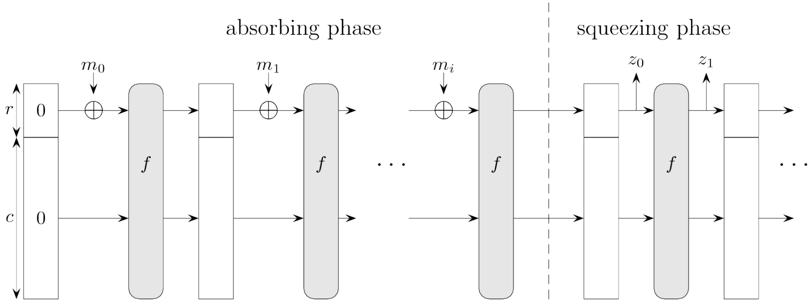
Question

Show how, in the Merkle-Damgård construction, someone can, given a hash, extend the hashed message without knowing it. In which use case is this problematic ?

SHA-3

- Result of the SHA **competition** : Keccak has been selected for becoming SHA-3
- Other finalists where : Blake, JH, Skein, Grøstl
- Standardized in NIST FIPS 202.
- Not based on Merkle-Damgård construction but on **sponge construction**.
- Versions : SHA3-224, SHA3-256, SHA3-384, SHA3-512, SHAKE128, SHAKE256
- 10^*1 padding.

Sponge Construction



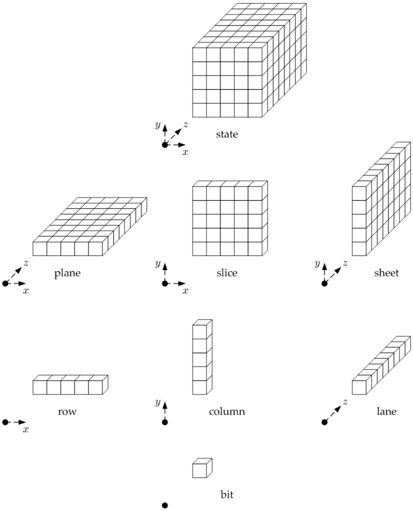
SHA3 Instances

Instance	Output	rate r	capacity c	Collision	Preimage
SHA3-224	224	1152	448	112	224
SHA3-256	256	1088	512	128	256
SHA3-384	384	832	768	192	384
SHA3-512	512	576	1024	256	512
SHAKE128	d	1344	256	$\min(d/2, 128)$	$\min(d, 128)$
SHAKE256	d	1088	512	$\min(d/2, 256)$	$\min(d, 256)$

The Keccak-f Internal Permutation

- 24 rounds R_r.
- The round operation is **easy to invert**.
- Works on a 3D state (was 2D in AES).
- Very elegant design similar to AES : each substep has a purpose.

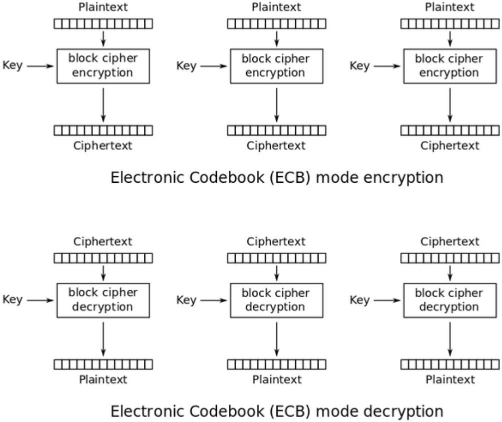
The SHA3 State



Modes of Operation

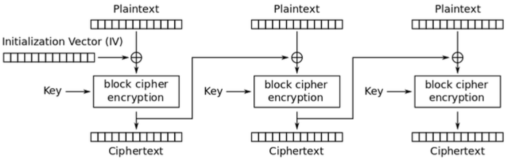
- A mode of operation allows to extend the reach of a block cipher for encrypting data of any length.
- All of them offer a security that is a function of the block length of the underlying block cipher. Due to the birthday effect, they begin to leak information after handling $2^{\frac{\ell}{2}}$ blocks, for an ℓ-bit block cipher.

ECB

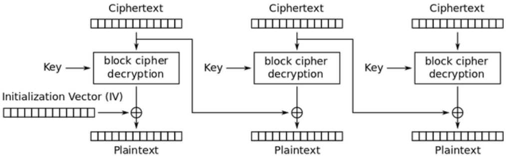


Source of picture : http://en.wikipedia.org/wiki/Block_cipher_modes_of_operation

CBC



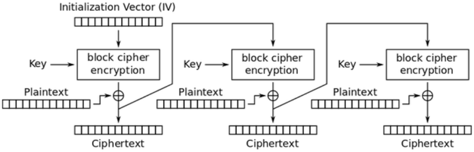
Cipher Block Chaining (CBC) mode encryption



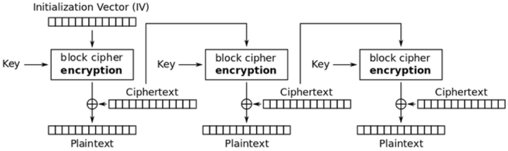
Cipher Block Chaining (CBC) mode decryption

Source of picture : http://en.wikipedia.org/wiki/Block_cipher_modes_of_operation

CFB



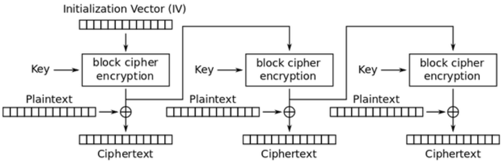
Cipher Feedback (CFB) mode encryption



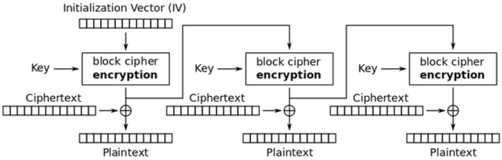
Cipher Feedback (CFB) mode decryption

Source of picture : http://en.wikipedia.org/wiki/Block_cipher_modes_of_operation

OFB



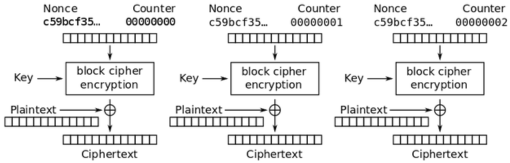
Output Feedback (OFB) mode encryption



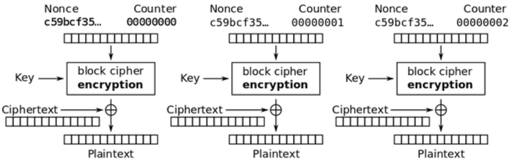
Output Feedback (OFB) mode decryption

Source of picture : http://en.wikipedia.org/wiki/Block_cipher_modes_of_operation

Counter Mode



Counter (CTR) mode encryption



Counter (CTR) mode decryption

Source of picture : `http://en.wikipedia.org/wiki/Block_cipher_modes_of_operation`

Which Mode of Operation would you Use ?

Question

In your opinion, which mode of operation is the best ?

Modes of Operations Recommendations

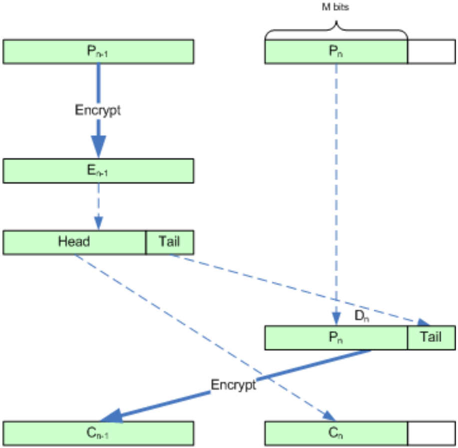
Primitive	Legacy	Future
ECB	x	x
CBC	✓	x
OFB	✓	x
CFB	✓	x
CTR	✓	x
XTS	✓	✓(not for ECRYPT)
EME (patented)	✓	✓
CBC-ESSIV	✓	x
Generic Composition	✓	x
CCM	✓	x
EAX	✓	✓
GCM	✓	✓if used properly
OCB v1.1	✓	✓
ChaCha20 + Poly1305	✓	✓
AES-GCM-SIV	✓	✓

source of recommendations (except the last one) :

Ciphertext-Stealing Mode

- Allows to deal with messages whose length is **not** a multiple of the underlying block cipher's block size, without any expansion of the ciphertext.
- Not standardized, but one can find it implemented in many situations

Ciphertext-Stealing



Source of picture : http://en.wikipedia.org/wiki/Ciphertext_stealing

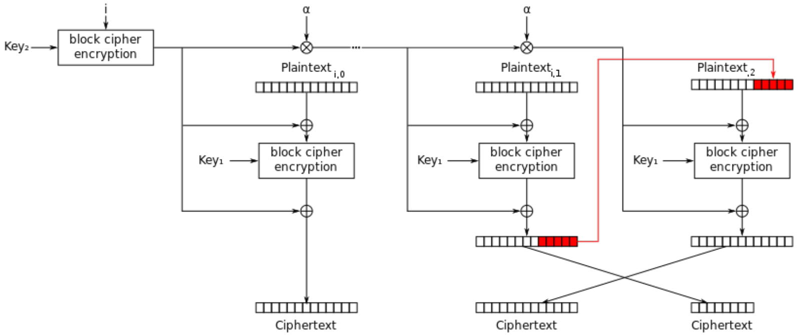
Disk Encryption

- Disk encryption has special requirements :
  - The data on the disk must remain **confidential**
  - Storage and retrieval of data should be **fast**
  - The encryption scheme must **not add overhead**.
- Adversaries can ...
  - ... **read** the raw contents of the disk at any time ;
  - ... request the disk to **encrypt and store arbitrary files** ;
  - ... **modify unused sectors** on the disk and then request their **decryption**.

XTS

- XTS means "XOR-Encrypt-XOR Tweaked-Codebook mode with Ciphertext-Stealing"
- Standardized in NIST SP800-38E.
- Supported by most disk encryption utilities
- Needs to perform multiplications in a Galois field
- Goal : encrypt index j at sector i.
- Requirements : random access with small overhead and no space increase.

XTS



Source of picture : http://en.wikipedia.org/wiki/Disk_encryption_theory

Stream Ciphers

- Stream ciphers are extremely fast symmetric encryption algorithms
- Their security is **less studied** than block ciphers.
- It is usually **preferable** to use a block cipher in CTR mode than a stream cipher (except maybe ChaCha20).

Which Stream Ciphers would you use ?

Primitive
ChaCha20
RC4
E0
A5/1
A5/2
HC-128
Salsa20/20
SOSEMANUK
Grain128a
Grain
Mickey
Trivium
Rabbit

Stream Cipher Recommendations

Primitive	Legacy	Future
ChaCha20	✓	✓
HC-128	✓	✓
Salsa20/20	✓	✓
SOSEMANUK	✓	✓
Grain128a	✓	✓
Grain	✓	x
Mickey 2.0	✓	x
Trivium	✓	x
Rabbit	✓	x
RC4	x	x
A5/1	x	x
A5/2	x	x
E0	x	x

source of recommendations : https://www.ecrypt.eu.org/csa/documents/D5.4-FinalAlgKeySizeProt.pdf

eSTREAM Portofolio

- Cryptographic competition to find a successor to RC4. Ended in 2008.
- Software-oriented :
  - **HC-128** : 128-bit key, 128-bit IV
  - **Rabbit** : 128-bit key, 64-bit IV, existing theoretical distinguisher
  - **Salsa20/12** : 128-bit or 256-bit key, 64-bit nonce, 64-bit. 12 rounds is not recommended for future use. Updated into **Salsa20/20** or **ChaCha20**.
  - **SOSEMANUK** : 128-bit key, 128-bit IV
- Hardware-oriented :
  - **Grain v1** : 80-bit key, 64-bit IV. Too small parameters. Updated into **Grain128a** : 128-bit key and 128-bit IV.
  - **Mickey 2.0** : 80-bit key, 80-bit IV. Too small parameters.
  - **Trivium** : 80-bit key, 80-bit IV. Too small parameters.

ChaCha20

- Designed by Dan Bernstein.
- Most used stream cipher nowadays. Replacement for RC4 in TLS for Google.
- Resistant to timing and cache attacks.
- 128-bit or 256-bit **key**.
- Uses a 64-bit **nonce** and a 64-bit **position counter**.
- **Variant** with 96-bit **nonce** and 32-bit **position counter** (RFC-7539).
- $f(\text{nonce}, \text{counter}, \text{key}) \rightarrow \{0,1\}^{512}$.
- Possibility to encrypt or decrypt a bloc of 512 bits without computing the whole stream.
- Possibility to have a **bigger nonce** with XChaCha20.

ChaCha20 or AES ?

- Both are well-analyzed and well-studied.
- AES is **faster** when the processor has dedicated instructions.
- Otherwise, ChaCha20 is faster.
- ChaCha20 is a stream cipher. You need to protect the integrity of the messages.
- ChaCha20-Poly1305 used in TLS1.3, SSH, ...

Message Authentication Codes

- Message authentication codes are symmetric signature schemes.
- They do **not** provide repudiation.

MACs Recommendations

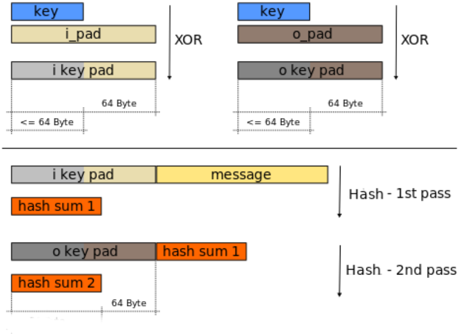
Primitive	Legacy	Future	Comment
HMAC	✓	✓	hash functions
CMAC	✓	✓	Block cipher
EMAC	✓	✓	Block cipher
AMAC	✓	✓	Block cipher
UMAC	✓	✓	Universal hash function
Poly1305	✓	x	When used alone
GMAC	✓	x	Used alone (not in GCM)
CBC-MAC	x	x	

source of recommendations : https://www.ecrypt.eu.org/csa/documents/D5.4-FinalAlgKeySizeProt.pdf

HMAC

- Transforms a hash function in a message authentication code.
- Standardized in RFC 2104 and NIST FIPS PUB 198.
- HMAC-SHA1 and HMAC-MD5 are used in TLS/SSL and IPSec, notably.
- Slow transition towards HMAC-SHA2.

HMAC

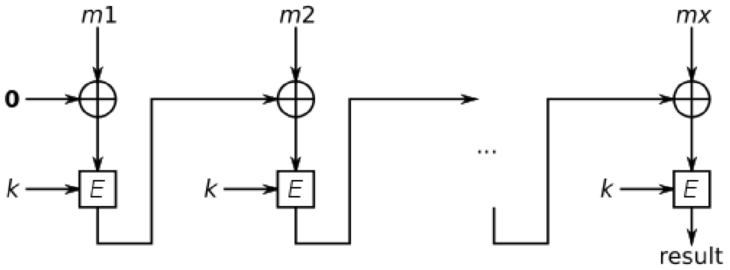


$$\begin{aligned} \text{opad} &= \texttt{0x5C5C5C}\ldots\texttt{5C} \\ \text{ipad} &= \texttt{0x363636}\ldots\texttt{36} \end{aligned}$$

Source of picture : http://en.wikipedia.org/wiki/Hash-based_message_authentication_code

CBC-MAC

- CBC-MAC
  - Based on the CBC encryption mode
  - Uses a block cipher $E_K(.)$
  - Only secure for messages with a **fixed size** !



CBC-MAC Variants

- CBC-MAC is dangerous to use and should not be used.
- **EMAC** encrypts the result with a second secret key.
- The **CMAC standard** was proposed by Black and Rogaway.
- Standardized in RFC 4493 and RFC 4494, as well as in NIST SP800-38B.
- XORs a constant to the last block that depends on the key and truncates the result. The key-dependent constant is $E_K(0)$ shifted by one bit to the left and XOR a constant if there is a carry.
- **AMAC** is another variant of EMAC. Used a lot in banking applications with DES.

Poly1305

- Proposed by Dan Bernstein.
- Often used with Chacha20 or Salsa20.
- Standardized in RFC 7539.
- Requires very simple operations.
- All the integers are represented in **little endian**.

Poly1305 Algorithm (IETF version) – Keys

1. From a **nonce** and a ChaCha20 key, derive a **32-byte key**.
2. Let r be the first 16 bytes and s be the last 16 bytes.
3. Clear the 4 most significant bits of $r[3], r[7], r[11], r[15]$.
4. Clear the 2 least significant bits of $r[4], r[8], r[12]$.
5. We see r as a 16-byte integer.

Poly1305 Algorithm (IETF version) – Computing the MAC

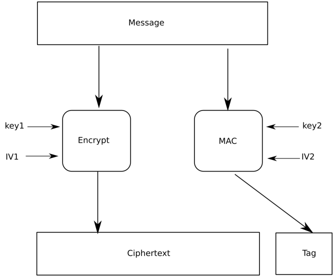
1. Let $p = 2^{130} - 5$ and $\mathrm{Acc} = 0$.
2. Divide the message into blocks of 16 bytes. (last one might be shorter)
3. For each message block :
    1. append the byte `0x01` to each block to obtain a 17-byte block. For the shorter block, append further with 0s.
    2. Compute $\mathrm{Acc} = (\mathrm{Acc} + \mathrm{block} \cdot r) \bmod p$
4. The final result is $(\mathrm{Acc} + s) \bmod 2^{128}$.

Combining Encryption and MAC

Question

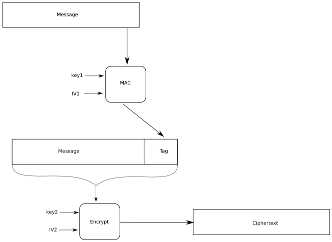
How would you combine a symmetric encryption scheme with a MAC ?

Encrypt-and-MAC



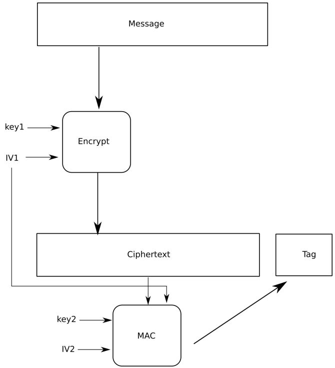
- Globally **not** secure.
- No integrity of the ciphertext : attack on OpenSSH.
- Bad interaction between Encryption and MAC.

MAC-then-Encrypt



- Globally **not** secure.
- Padding oracle attacks on TLS.
- No integrity of the ciphertext.

Encrypt-then-MAC



- Globally the **least risky** solution.
- Protects integrity of the plaintext and the ciphertext.

Same Key for MAC and Encrypt ?

- Globally **risky**.
- Sometimes catastrophic : CBC with CBC-MAC.
- Sometimes no known interaction : AES with SHA2.

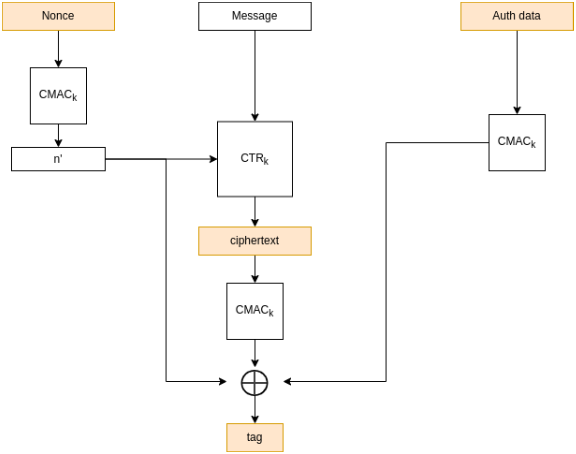
Authenticated Encryption

- **Authenticated encryption** (AE) schemes offer two security properties at the same time : **confidentiality and authenticity**.
- They usually combine a standard mode of operation and a MAC in a clever way.
- Possibility to authenticate data without encrypting it (authenticated data (AD)). AEAD is commonly used for Authenticated encryption with authenticated data.
- **20.02.2019** : announcement of winners of CAESAR competition !
- **Always prefer authenticated encryption when possible !**

CCM

- CCM is a combination of CBC-MAC and the CTR mode.
- CCM uses a block cipher with a 128-bit block size.
- Initialization vectors (IV) must **never** be repeated.
- Standardized in RFC 3610 in combination with AES.
- CCM is used in 802.11i (aka CCMP), IPSec and TLS 1.2.

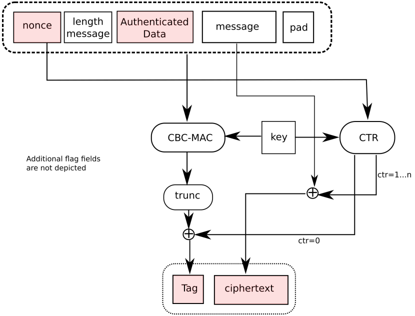
EAX



Question

What are the improvements compared to CCM ?

EAX vs CCM

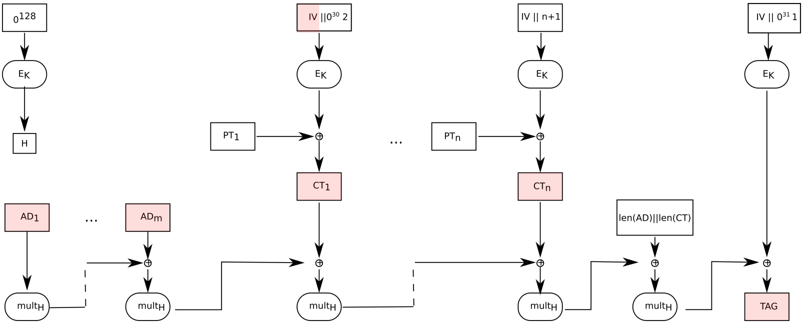


- Strict improvement of CCM.
- Allows to verify the tag before decrypting.
- Can process a stream of data.
- Can pre-process associated data.

GCM

- GCM relies on a block cipher with a 128-bit block size, and accept initialization vectors of any size.
- GCM uses multiplications in $GF(2^{128})$ constructed as $\mathbb{Z}_2[x]/(x^{128} + x^7 + x^2 + x + 1)$
- Standardized in NIST SP800-38D and is used in IPSec, TLS and SSH, notably.
- The IV (nonce) should **never** be repeated.
- GCM is used in IPSec, TLS, and SSH.
- Better performances than CCM.

GCM



0^{128}
E_K
H
IV $||0^{30}$ 2
E_K
PT_1
CT_1
...
IV || n+1
E_K
PT_n
CT_n
IV || 0^{31} 1
E_K
AD_1
...
AD_m
$mult_H$
$mult_H$
$mult_H$
$mult_H$
len(AD)||len(CT)
$mult_H$
TAG

Using GCM

- GCM is **not easy** to use properly.
- The **number of messages** one can encrypt under the same key is **limited**.
- The **size** of a message one can encrypt is **limited**.
- The **IV size** has to be **96 bits**.
- Consult the NIST **special publication 800-38D** for details on what has to be done to obtain a secure implementation.
- Many problems are solved in AES-GCM-SIV (used for instance in BoringSSL by Google).

Birthday Paradox Reminder

- In a space of size d, the probability of having a collision when drawing n values is approximatively

$$1 - e^{-n^2/(2d)} \; .$$

- This implies that you can find with good probability a collision on ℓ bit hashes with $2^{\frac{\ell}{2}}$ evaluations of the hash function.

GCM Limitations

- The size of **one message** should not be more than $2^{32}-2$ blocks.
- For GCM with **random IVs**, the **number of messages** encrypted under one key should not be larger than 2^{32}.
- This is obtained with the **birthday paradox** : we want the probability of collision to be $< 2^{-32}$.
- For **deterministic IVs**, the maximum number of messages is 2^{64} (based on how fields are splited in the IV).
- Reusing an IV in GCM breaks **confidentiality** (like for CTR) and **integrity**. One can then authenticate **any message**.

CAESAR Competition

- Started in 2013. About 60 candidates.
- Not managed by a government or a standardization entity.
- Portofolio of winners announced in 2019.
- Winners chosen based on security, analysis quality and performances.

CAESAR Winners

- **Lightweight applications :** Ascon (first choice), ACORN (second choice)
- **High-performance applications :** AEGIS-128 and OCB
- **Defense in depth :** Deoxys-II (first choice), COLM (second choice)

CAESAR : Lightweight Applications

- Fits on small hardware area.
- Fast on 8-bit CPUs.
- Good hardware performances.
- Usually for short messages.

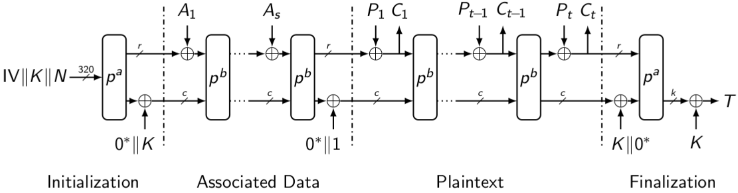
CAESAR : High-Performance Applications

- Efficient on 32-bit/ 64-bit CPU.
- Efficient on dedicated hardware.
- Usually for long messages.

CAESAR : Defense in Depth

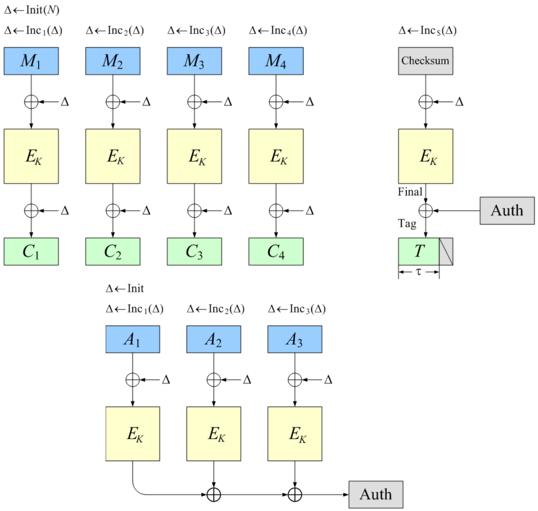
- Protection against **nonce misuse**.
- Limits damage when plaintext are decrypted although not authentic.

ASCON



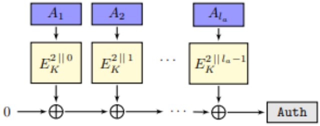
https://ascon.iaik.tugraz.at/specification.html

OCB

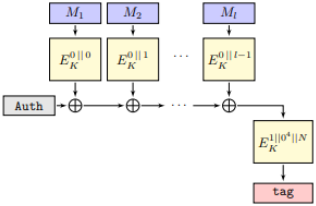


http://web.cs.ucdavis.edu/~rogaway/ocb/ocb-faq.htm

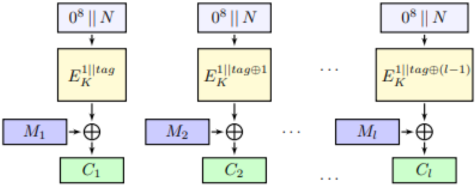
Deoxys-II



(a) Without padding.



(a) Without padding.



(a) Message-length is a multiple of the block size.

https://competitions.cr.yp.to/round3/deoxysv141.pdf

Authenticated Encryption Recommendations

Primitive	Legacy	Future
Generic Composition	✓	x
CCM	✓	x
EAX	✓	✓
GCM	✓	✓if used properly
OCB v1.1	✓	✓
ChaCha20 + Poly1305	✓	✓
AES-GCM-SIV	✓	✓

source of recommendations (except the last one) :

https://www.ecrypt.eu.org/csa/documents/D5.4-FinalAlgKeySizeProt.pdf

Conclusion


AAAAAAAAAAAAA
stands for
All Asian, African, American, And Australian Association Against Acronym And Abbreviation Abuse Anonymous
ABBREVIATIONS
Abbreviations.com