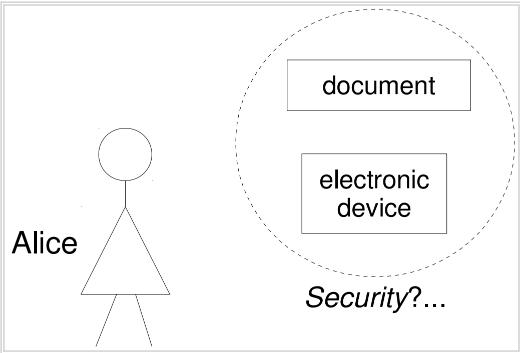
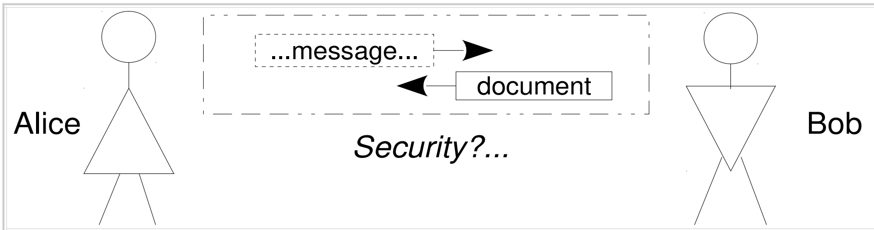
COMPUTER SECURITY

Cryptography: general protection techniques





Protection purpose

- provide **access control** to resources (e.g. users' information)
  - by building secure channels
    - for communication
    - for storage
  - with properties
    - main: confidentiality, integrity and authentication
    - secondary: anonymity, forward secrecy, etc.

Secure channel for communication:

- cryptographically-protected conversation line between two identified subjects
  - called, in some contexts, *security association* (SA)
- basic, expected properties:
  - Authentication – assuring that each subject is talking to the genuine other
  - Integrity – assuring that deletion, change or addition of data is detected
  - Confidentiality – assuring that data is not understandable by anybody else

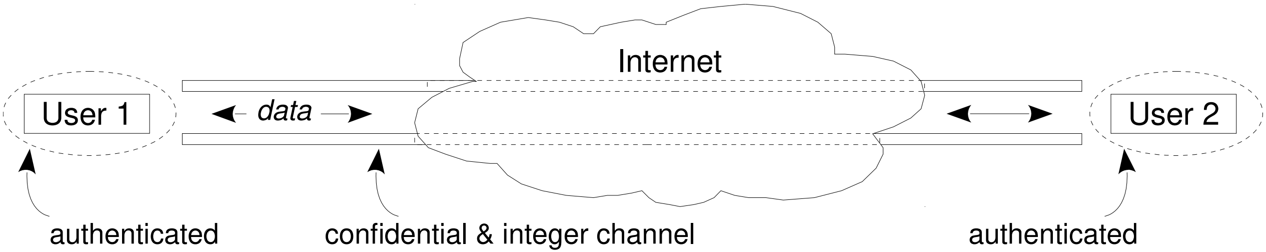


Fig. General secure (communication) channel.

Utilization of a secure channel

- 1st: Authentication of one or both subjects and probable parameter negotiation
  - usually,
    - an asymmetrical cipher is used
    - a "session key"[1] is created
- 2nd: Utilization proper
  - maybe also with protection for
    - integrity
    - confidentiality
  - usually,
    - a symmetrical cipher is used (with above session key)

1 more on this in a following chapter

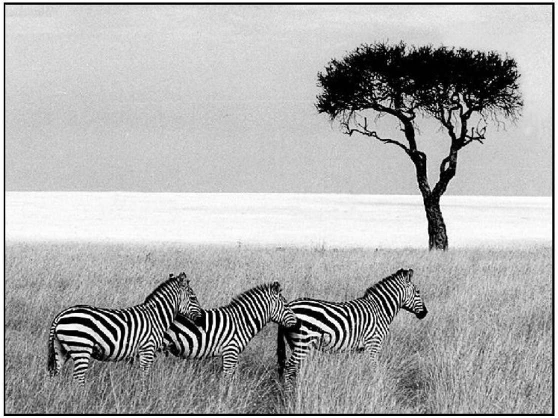
Protecting Communication Channels

Confidentiality

- assurance of limited disclosure of information
  - implies Authentication of the entities involved!

Solutions

- hide the sensitive documents
  - physically saving them
  - cunningly disguising them
    - steganography! [FIG[1]]
- encipher documents
  - parties need appropriate keys



1 Presumably, the original of this picture (coloured, 1024×768 pixel), contains in compressed form the complete unabridged text of five Shakespeare's plays, totaling more that 700kB of text. (Tanenbaum, Modern Operating Systems)

Hiding of documents

- not covered here (see steganography examples in the literature)

Encipherment of documents

- symmetrical technique [FIG a)]
- asymmetrically technique [FIG b)]

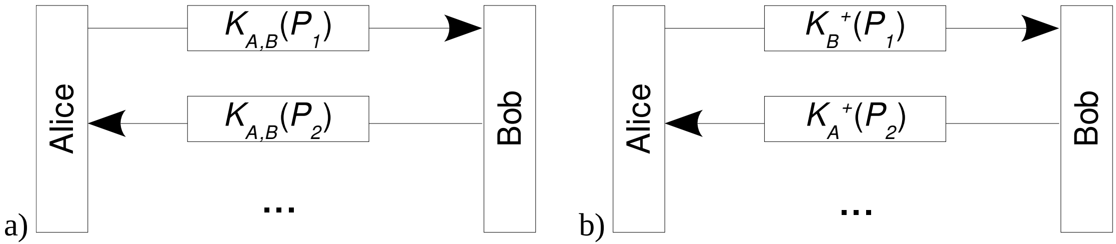
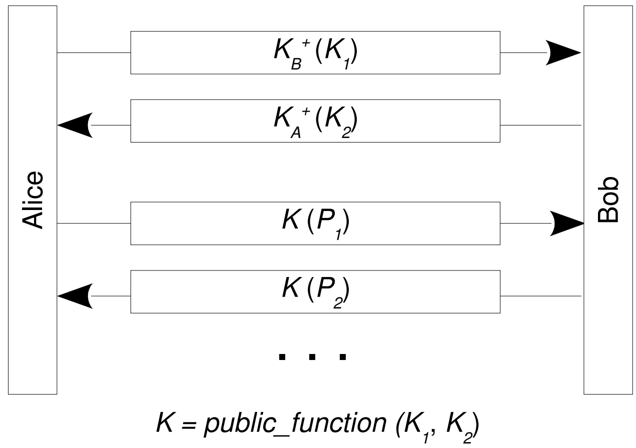


Fig. Base encipherment techniques: a) shared key; b) public key.

Practical problems:

- symmetrical keys are difficult to manage
- asymmetrical operations are very inefficient

- So, usual solution:[1] [FIG]
  1. exchange symmetric key by public-key means
  2. encipher documents with exchanged shared key



1 Conceptually, steps are sometimes called: 1. key encapsulation mechanism (KEM) ; 2. data encapsulation mechanism (DEM).

Integrity

- assuring that a change in "document"[1] is detected[2]
  - implies Authentication of the entities involved!

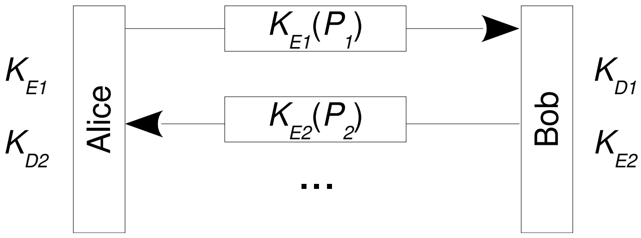
Solutions

- encipher the document (...)
  - with symmetric or asymmetric algorithms
- use integrity code
  - with shared key
- digitally sign the document
  - directly, with private key of sender
  - through its digest (with private key of sender)

1 file, message,...
2 if detected, change cannot be corrected (in general!)

Simple "solution" for integrity problem: encipher everything!

- exchange ciphered information
  - detection of alteration of message (e.g. intelligibility affected)!
  - confidentiality also granted (but not relevant here)



Problems

- symmetric cipher: no origin authenticity (repudiation is possible)!
- asymmetric cipher: low efficiency!
- in any case, alterations can go unnoticed:
  - in applications with general binary data (numbers, pictures...)
  - with some algorithms that guarantee confidentiality but not integrity (e.g. *One-time pad*)!

Better solution: use Message Integrity Codes, MIC[1]

- parties agree on a (shared) key
- sender builds an *hash* of "message *plus* key" (*keyed hash* technique):
  - that is the MIC! E.g. MIC = $h\ (m \,\|\, K)$ (|| means concatenation)
- sender transmits both message and MIC
- receiver can check message's integrity, repeating hash operation

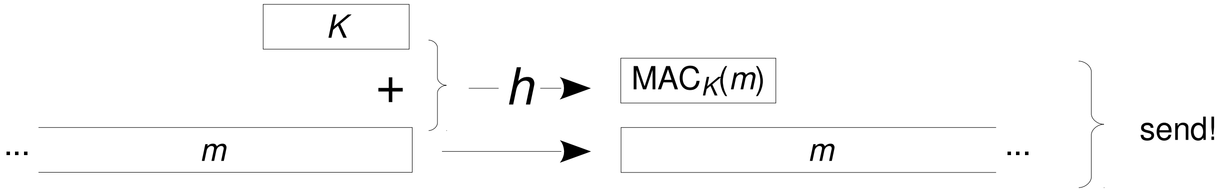


Fig. General construction principle and usage of Message Integrity Codes.

1 The Message Integrity Check term, originally presented in RFC 1421 (Privacy Enhancement for Internet Electronic Mail), is currently not much used; instead, the designation in fashion is Message Authentication Code, MAC.
Some authors make a slight distinction between the two (e.g. see Menezes et al.' Handbook of Applied Cryptography); I will not. Also, I will prefer MIC, as it is more clear.

...Integrity Protection with message integrity codes (cont.)...

Problems

- uses a shared key
  - parties must exchange it, somehow
  - there is no prevention for:
    - message alteration (or forging) by the recipient
    - message repudiation by the sender!

Exercise:

- What vulnerability would turn up if in the *keyed hash technique* MIC/MAC was instead defined as $h\ (K \| m)$?

A famous MIC: the HMAC

- HMAC, *Hashed Message Authentication Code*, IETF RFC 2104
  - MAC = $h\ \{(K \oplus opad)\ \|\ h\ [(K \oplus ipad)\ \|\ m)]\}$

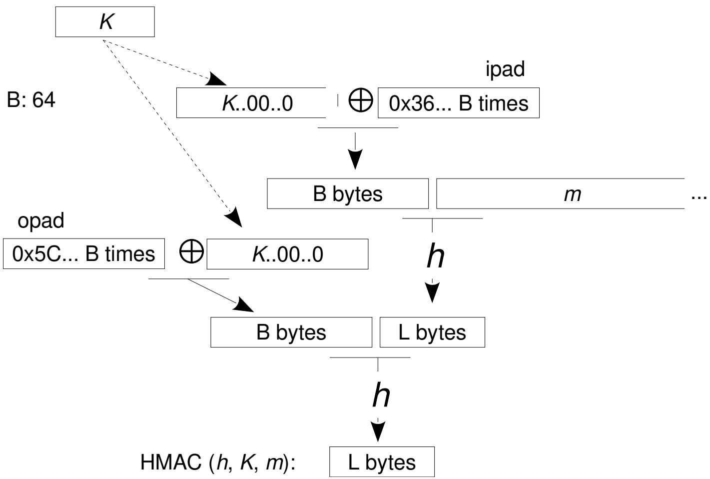


Fig. HMAC with SHA-1:

- h = SHA-1
- K = 128 b
- B = 64 bytes (512 b)
- L = 160 b

E.g.

HMAC_SHA1("key", "HMAC") =
6f0cd789d86644081ce5fe03caa9a70478d1f14e

Great solution: use digital signatures

- allows:
  - checking of a document for alteration
  - associating a document to its author
- and so:
  - only author can change the original document
  - readers are assured of the identity of author
  - author is not able to deny authorship of document (repudiate it)

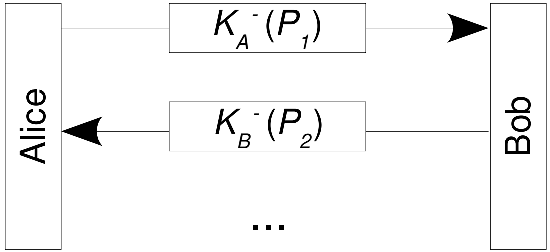
Techniques

- public key[1]
- message digest (<u>with public key</u>!)

1 In reality, a digital signature is made with a *private* key!

Digital signatures: (plain) public key technique

- encipherment with sender's private key
- decipherment with sender's public key



Problems

- "major":
  - asymmetric cipher: low efficiency!
- "minor":
  - sender's private key must be kept secret
  - sender's public key must be known in advance
  - longevity of protection of sent document implies safe keeping of key pair

Digital signatures: message digest (with public key) technique

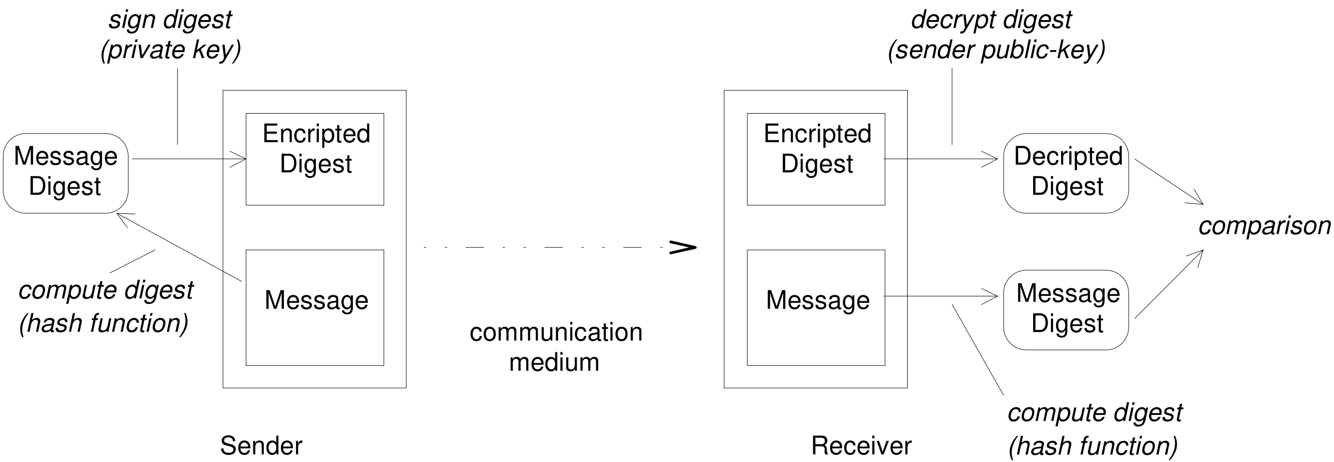


Fig. Integrity protection with digital signatures: message digest technique. (*in* Tanenbaum, ...)

...Digital signatures: message digest technique (cont.)

Problems

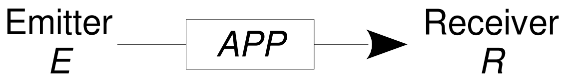
- "major":
  - greater complexity
    - (but no efficiency penalty as hashing is very fast!)
- "minor":
  - same as (simple) public key's technique

Exercise (Integrity protection):

- Present an advantage and a disadvantage of each of the different techniques for integrity protecting messages.

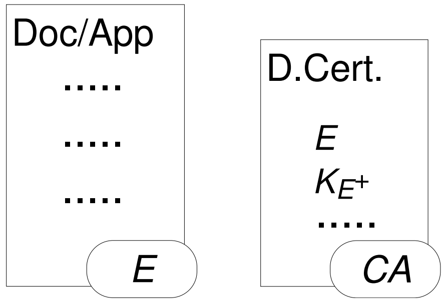
...Digital signatures (cont.)

Example: Secure distribution of documents or software



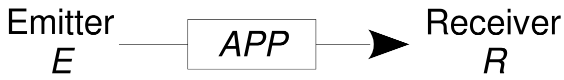
Part I: Emission

- Emitter *E* of application/document *APP*
  - digitally signs *APP*
    - public-key technique, digest technique...
    - generates $[APP]_E$ [1]
  - appends to $[APP]_E$ a digital certificate[2] $[DC(E)]_{CA}$
    - certificate has K_E^+
    - is signed by *CA* (also trusted by Receiver!)
  - sends everything to Receiver
    - $APP + [APP]_E + [DC(E)]_{CA}$



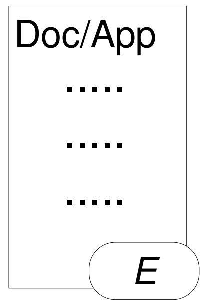
1 Notation of digital signature: $[DOC]_E \Longleftrightarrow K_E^-$ (DOC) or $[DOC]_E \Longleftrightarrow K_E^-$ (h(DOC))
2 much more on this in a following chapter

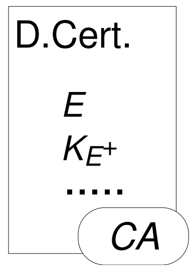
…Example: Secure distribution of documents or software (cont.)



Part II: Reception

- Receiver R of application/document
  - gets K_E^+ of Emitter (if he does not yet have it)
    - by processing the digital certificate $[DC(E)]_{CA}$
      - must already know, or somehow get, K_{CA}^+
      - checks the integrity of $[DC(E)]_{CA}$
  - checks the integrity of $[APP]_E$
  - uses *APP* with confidence!





Integrity + Confidentiality: Authenticated Modes

- as already said, even "mixed" confidentiality operation modes are vulnerable to undetectable modifications of ciphertext
- so, some type of integrity protection must be added
  - basic example: combine secrecy with digital signatures [FIG]
  - in general: use *authenticated encipherment* protocols

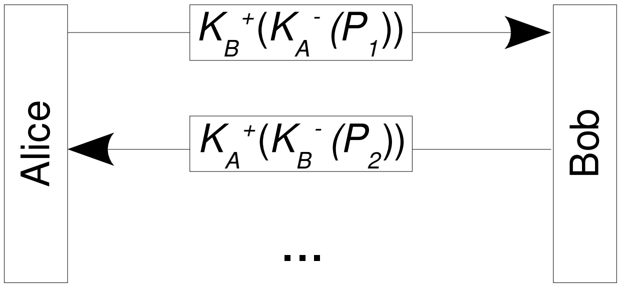


Fig. Confidentiality with integrity protection.

Authenticated ciphering protocols (modes)[1]

- special protocols developed to aggregate both protections
  - in general, integrity protection is provided by Message Integrity[2] Codes
  - but digital signing can also be used (of course) [previous FIG]
- the main approaches are:
  - (external) combination of protective techniques[3]
    - prone to failures due to incorrect implementation
  - "intrinsic" combination
    - several standardized schemes
    - sponge functions can be used in *duplex mode*!
    - *signcryption*: "low-cost" combination of digital signing and ciphering[4]

1 Authenticated Encryption with Associated Data (AEAD) applies when it is explicitly necessary to assure integrity protection of plaintext data that is to accompany ciphertext (e.g. network packets might need a visible header that should be integrity protected as well as the secret payload).

2 or Authentication ;-)

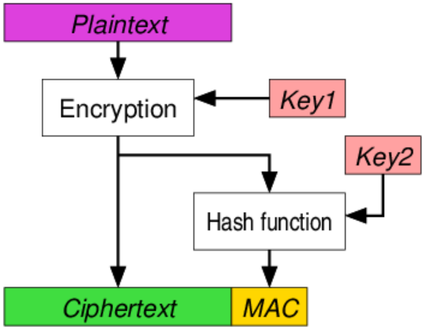
3 also called "generic composition" of schemes used separately for achieving confidentiality and integrity protection

4 Digital Signcryption or How to Achieve Cost(Signature & Encryption)..., Y. Zheng, CRYPTO '97

Authenticated Modes - "generic composition"

Encrypt-then-MAC, EtM

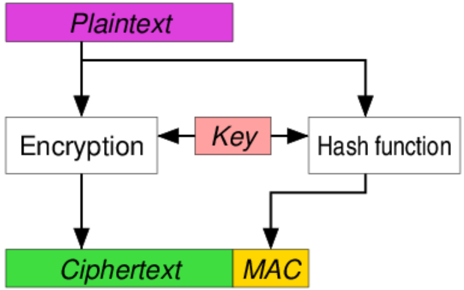
- ISO/IEC 19772:2009
- process [FIG - *in* Wikipedia]
  - 1st, encipher; 2nd, calculate MIC
  - non-parallelizable
- different keys K_E, K_{MAC} !
- "normal" padding
- reverse process:
  - verify integrity of ciphertext; decipher to get plaintext
  - parallelizable
- considered the more secure method (compared with the following)[1]



1 see, for instance, Bellare & Namprempre "Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm" (2008)

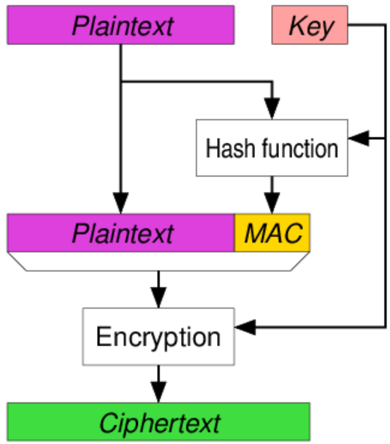
Encrypt-and-MAC (E&M)

- process [FIG - *in* Wikipedia]
  - encipher; calculate MIC
  - parallelizable
- apparently, a single key is enough!
- "normal" padding
- reverse process:
  - 1st, decipher to get plaintext;
    2nd, verify integrity of plaintext
  - non-parallelizable



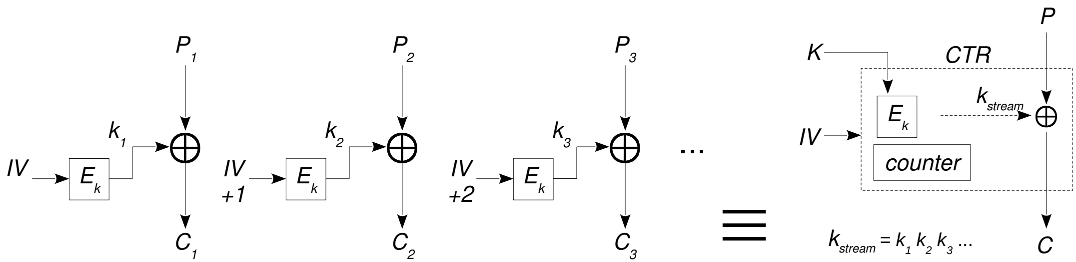
MAC-then-Encrypt (MtE)

- process [FIG - *in* Wikipedia]
  - 1st, calculate MIC; 2nd, encipher
  - non-parallelizable
- apparently, a single key is enough!
- padding after hashing
- reverse process:
  - 1st, decipher to get plaintext and MAC; 2nd, verify integrity of plaintext
  - non-parallelizable



Authenticated Modes - "intrinsic"

- here, there is an integration of the 2 protections
  - the schemes are built with provision to provide both
- the usual procedure is
  - use a primary key (*seed*) to feed an extended key-generation function
  - use the generated long key, to encipher *P* in *stream* mode
    - typically, a variant of Counter Mode is used [FIG]
  - use part of the generated key to produce a MIC of the ciphered (or plain) text



Galois/Counter Mode (GCM)

- NIST 800-38D
- process [FIG]
- confidentiality:
  - AES-128b is typical
- integrity protection: GMAC [FIG next page]
  - ciphertext + Associated Data
- apparently, highly performative (parallelization by inter-leaving & pipelining?)
- some obs:
  - *AD* and *C* are padded separately before being concatenated; *IV* is used sequentially in GMAC first and then in CTR; internal intermediate states are to be kept private

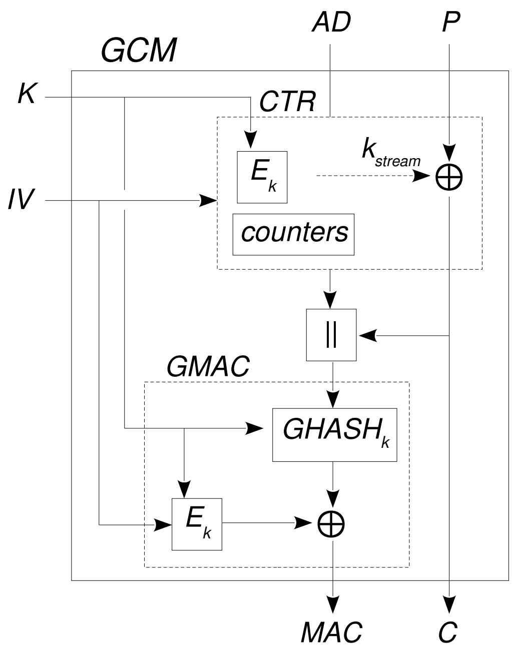
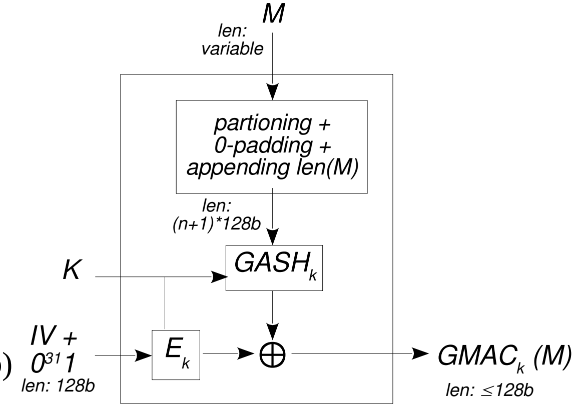
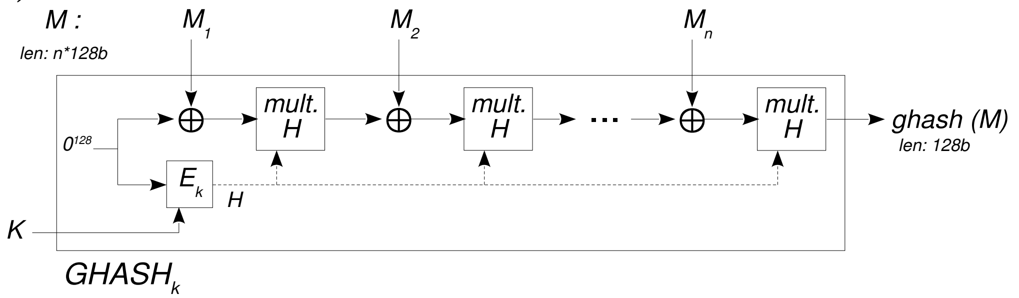


Fig. Basic constructs of Galois/Counter Mode for integrity protection:
a) hash function GHASH;
b) Message Authentication Code GMAC.

b)

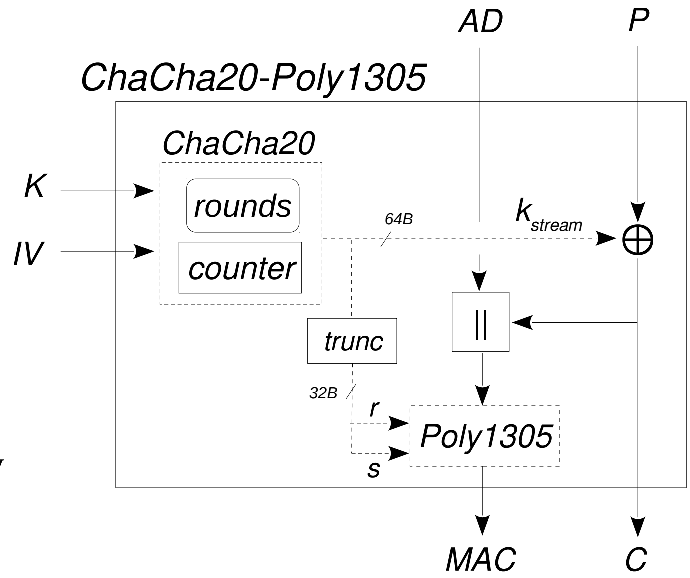


a)



ChaCha20-Poly1305

- RFC 8439
- designed by D. J. Bernstein
  - ChaCha20[1] stream cipher
  - Poly1305 *authenticator*
- process [FIG]
  - key stream feeds message integrity code function first (counter=0) and then XOR cipher (counter>0)
  - *AD* and *C* are padded separately before being concatenated



1 20-round version of ChaCha

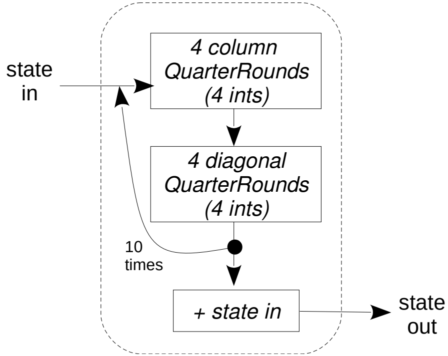
- **Chacha20:**
  - input: 32B (256b) key, 12B (96b) IV (***nonce***), 4B (32b) counter [FIG]
  - output: stream key in 64B (512b) blocks
  - internal state: 4 x 4 x 4B (16 32b-integers) = 64 B (512b)
  - block function: [FIG]
    - sequence of 10 double "quarter"-rounds
    - quarter-round: set of operations on 4 numbers (addition modulo 2^{32}, XOR, left-shift of *n* bits)
    - final sum with input
  - encipher algorithm:
    - for each iteration (increasing counter), use key stream to cipher 64B block of Plaintext
  - deciphering is obvious

state (4x4 32b ints) in:

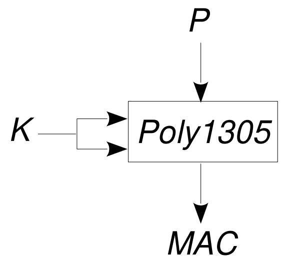
Cnst	Cnst	Cnst	Cnst
Key	Key	Key	Key
Key	Key	Key	Key
Ctr	IV	IV	IV

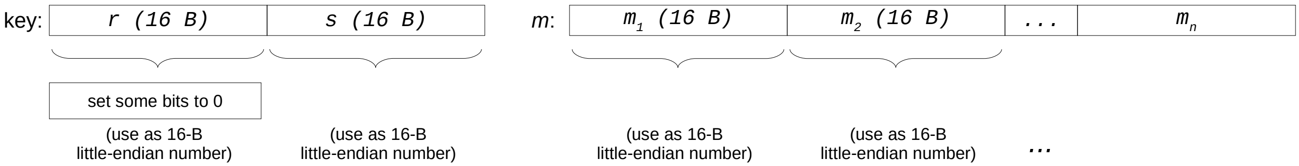
Cnst	Cnst	Cnst	Cnst:
"expa"	"nd 3"	"2-by"	"te k"

Chacha20



- **Poly1305**
  - input:
    - 32B (256b) **one-time**, two-part key: *r* (16B) || *s* (16B)
    - arbitrary-length message
  - output: 16B (128b) MAC
  - arithmetic operations with 16B groups used as numbers






```
for i: 1..n
      Acc = 0
      Acc = Acc + m_i
      Acc = Acc x r
      Acc = Acc % (2^130-5)
Acc = Acc + s
```

MAC: (use Acc as 16-B string) =

$(m_1 r + m_2 r^2 + \ldots + m_n r^n) \bmod (2^{130}-5) + s$

Fig. D. J. Bernstein's Poly1305 *authenticator*: 128b MAC.

SpongeWrap

- sponge construct in duplex mode

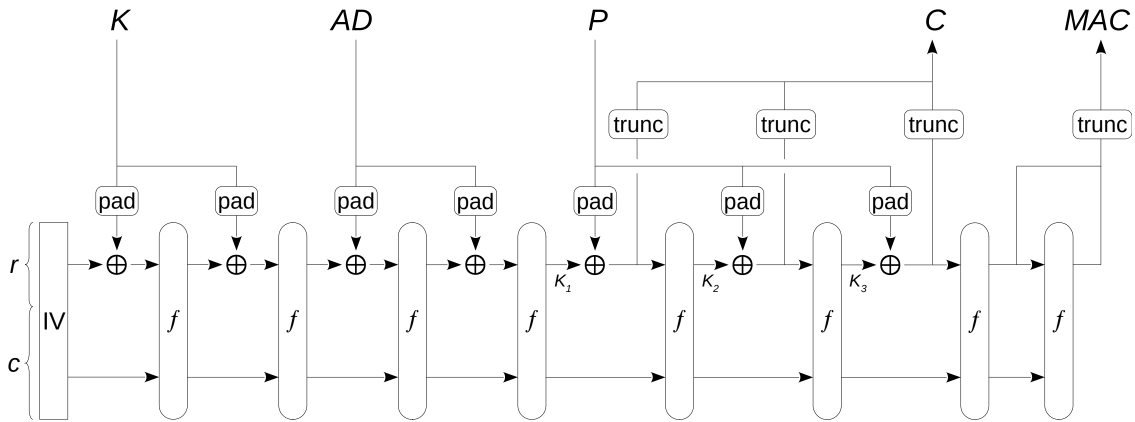


Fig. Sponge construct in duplex-mode for authenticated enciphering (AEAD): notice that plaintext *P* is XORed, block by block, with *f*'s outputs - the *keystream*, k_i *!* The function *pad* is used for padding and separation of data segments. The *trunc* removes padding and truncates the MAC.
(in Y.Sasaki and K.Yasuda, 2015)

Authentication (*to be presented*)

- assuring the identity of the entities involved
- *topic to be presented*!

Pointers...

- **Steganography: Hiding Data Within Data**, 2001 – Gary Kessler
  - www.garykessler.net/library/steganography.html
- The “**HMAC RFC**”, 1997 – H. Krawczyk, M. Bellare, R. Canetti
  - tools.ietf.org/html/rfc2104
- “**Authenticated encryption**”, Wikipedia
  - en.wikipedia.org/wiki/Authenticated_encryption
- “**Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC**”, 2007 – M. Dworkin, NIST
  - nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38d.pdf
- “**The Poly1305-AES Message-Authentication Code**”, 2005 – D. Bernstein
  - link.springer.com/content/pdf/10.1007/11502760_3.pdf
- “**ChaCha, a variant of Salsa20**”, 2008 – D. Bernstein
  - cr.yp.to/chacha/chacha-20080120.pdf
- “**Duplexing the sponge: single-pass authenticated encryption...**”, 2011 – G. Bertoni, J. Daemen, M. Peeters, G.Van Assche
  - eprint.iacr.org/2011/499.pdf