Corso Professionalizzante di Specializzazione (3 CFU)
Ingegneria dell'Informazione o magistrale in Ingegneria Informatica
Automatica, Ingegneria Elettronica,
Ingegneria delle Telecomunicazioni

WSN and VANET Security

Part II: Techniques for WSN and VANET Security

Lecture II.1

Passive Security Functions: Techniques

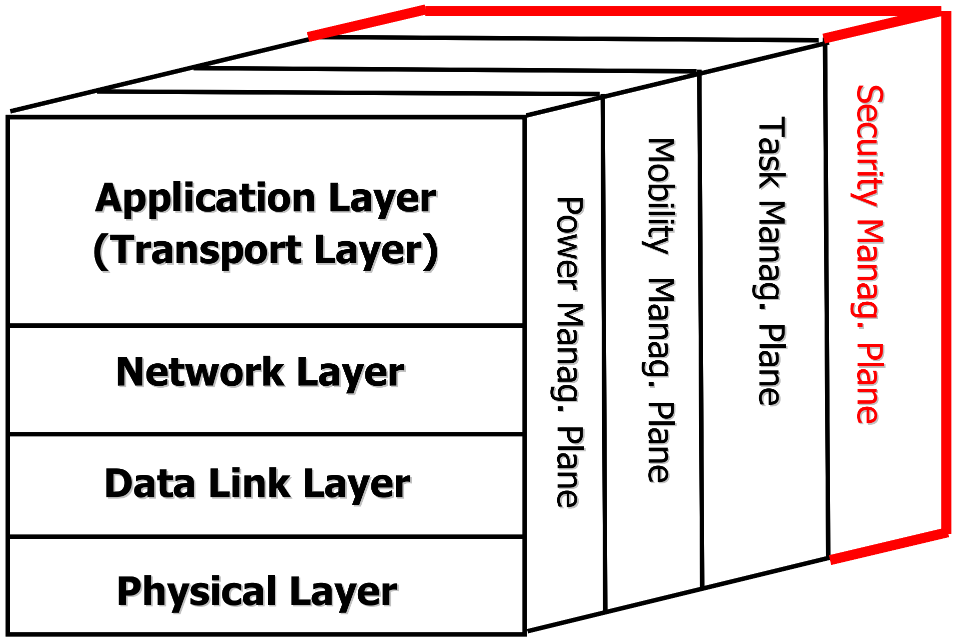
Ing. Marco Pugliese, Ph.D., SMIEEE
Senior Security Manager cert. UNI 10459-2017
marpug@univaq.it
April 26th, 2024

Outline

- Passive Security Functions
  - Ciphering
  - Hash functions
  - Message authentication codes
  - Digital signatures
- Key Establishment Protocols
  - Symmetric KEP
  - Asymmetric KEP
  - ID Based KEP
  - Hybrid KEP
  - Authentication of public key
- Key Management Protocols
  - TinySEC
  - TinyECC
  - TinyIBE
- Passive security techniques for
  - IEEE 802.15.4 MAC
  - Routing
  - ZigBee

Passive Security Functions

- Passive Security Functions (PSFs) concern:
  - Ciphering
  - Hash functions
  - Message authentication codes
  - Digital signatures



PAY ATTENTION:

The same acronym MAC is used for two very different functions: Medium Access Control vs. Message Authentication Code.

In cryptography, is also used the term **MIC (Message Integrity Code)** for MAC.

Basic Terms

Cryptography**: Secret** (*crypto-*) **writing** (*-graphy*)

- **Plain-text:** the original message.
- **Cipher-text:** the ciphered message.
- **Cipher:** an algorithm to add entropy to a plain-text in input so that the resulting cipher-text in output appears as random as possible.
- **Decipher**: an algorithm to substract entropy to a cipher-text in input and extract the related plain-text in output. Cipher and decipher can be the same.
- **Key:** secret information used as a parameter to ciphers and deciphers.
- **Key Establishment Protocol (KEP)**: a scheme to generate keys
- **Key Management Protocol (KMP)**: a scheme to manage keys
- **Authentication:** an algorithm to prove the integrity of the message (MAC).
- **Signature (or Sender Authentication):** an algorithm to prove the integrity of message sender.
- **Symmetric Cryptographic Scheme**: same key to cipher / decipher.
- **Asymmetric Cryptographic Scheme**: different keys to cipher / decipher.

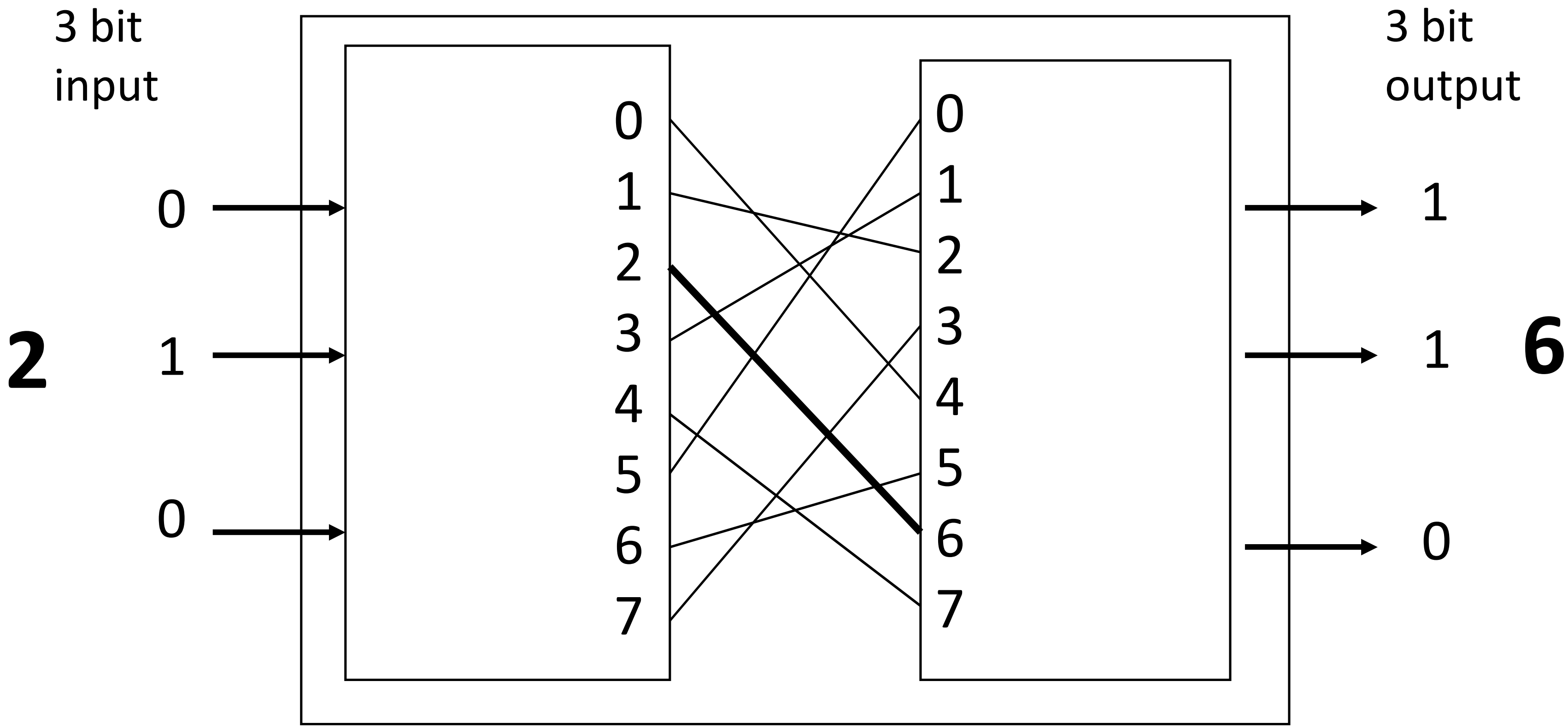
Outline

- Passive Security Functions
  - Ciphering
  - Hash functions
  - Message authentication codes
  - Digital signatures
- Key Establishment Protocols
  - Symmetric KEP
  - Asymmetric KEP
  - ID Based KEP
  - Hybrid KEP
  - Authentication of public key
- Key Management Protocols
  - TinySEC
  - TinyECC
  - TinyIBE
- Passive security techniques for
  - IEEE 802.15.4 MAC
  - Routing
  - ZigBee

Shannon's building blocks

- **product cipher:** two ciphers C_1 and C_2 in sequence (**round**)
- **commuting cipher**: given ciphers C_1, C_2 with key spaces K_1, K_2 respectively, if the space K_{12} of product cipher C_1C_2 is equal to key space K_{21} of C_2C_1, then ciphers C_1 and C_2 commute.
  - The security provided by a product of commuting ciphers **is the same of single ciphers**.
  - If C_1 and C_2 are commuting ciphers and given the same input, the cascades C_1-C_2 and C_2-C_1 produce the same output.
- Shannon proposed the following non-commuting ciphers:
  - **S-Box** (*substitution cipher)* **providing confusion by substitution**. Confusion means that **each binary digit of the cipher-text should depend on several parts of the key.** If one bit of the key is flipped then (statistically) half of the bits in the cipher-text should change. **The property of confusion hides the relationship between the cipher-text and the key**.
  - **P-Box** (*permutation cipher)* **providing diffusion by permutation**. Diffusion means that **the statistical structure of plain-text is dissipated over the cipher-text**. If one bit of the plain-text is flipped then (statistically) half of the bits in the cipher-text should change too, and similarly, if one bit of the cipher-text is flipped then (statistically) one half of the plain-text bits should change too. **The property of diffusion hides the relationship between the cipher-text and the plain-text**.
- **S-P design principle currently in use (e.g. in AES NIST standard)**

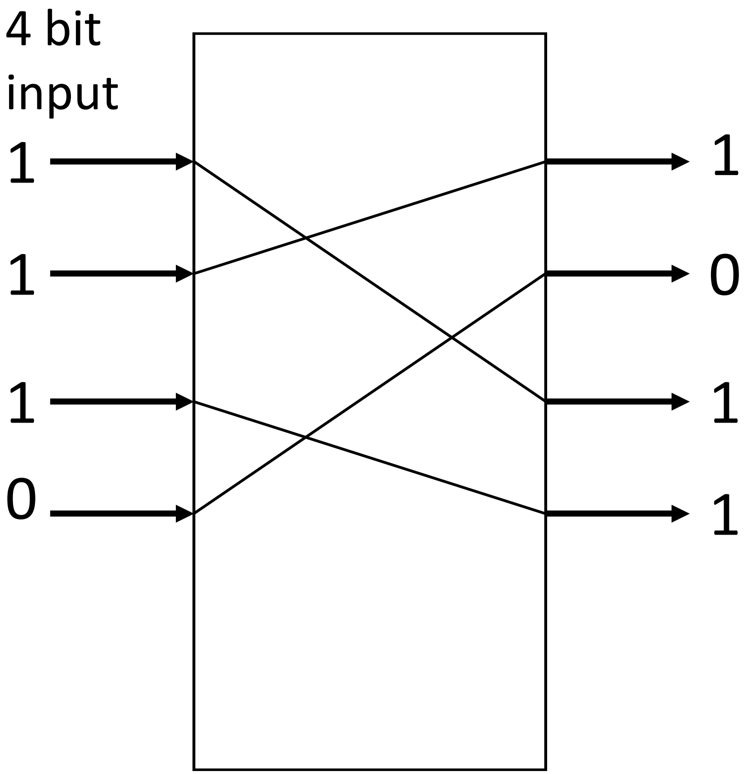
S-box (substitution)



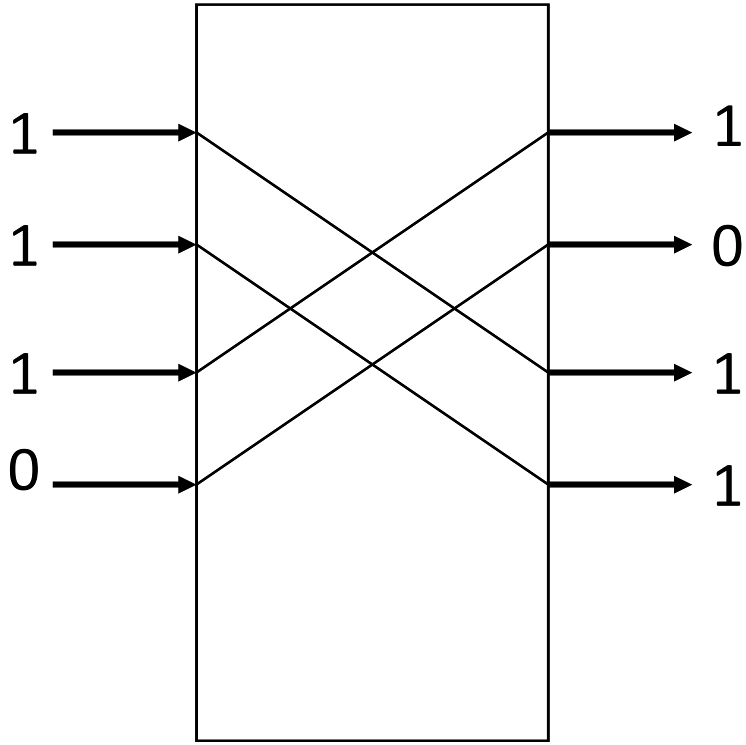
Word size of 3 bits => mapping of $2^3 = 8$ values

Note: a lookup table 8 x 8 is here used (in AES is 16 x 16)

P-box (permutation)



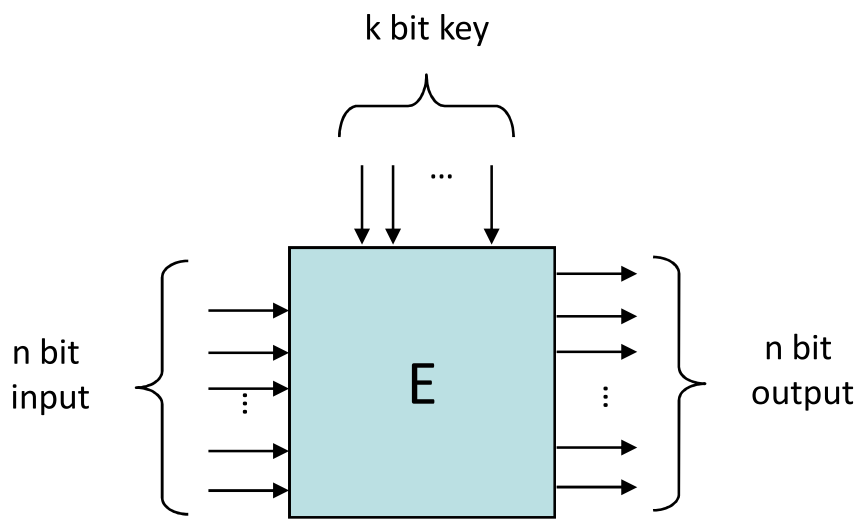
Example 1



Example 2 - swap two halves of input

Block ciphers

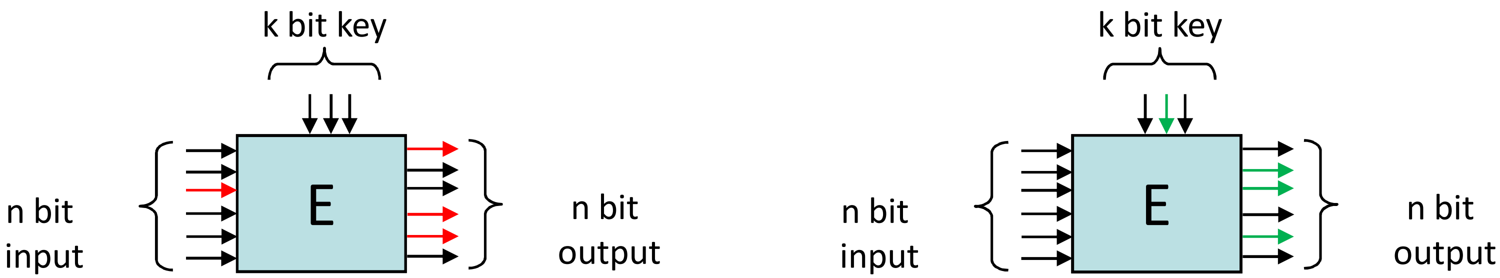
- Block ciphers operate on block (or string, or grams) of binits of the plaintext.
- An n bit block cipher is defined as:
  - given $E \in$ PRF defined as $E: \{0, 1\}^n \times \{0, 1\}^k \rightarrow \{0, 1\}^n$ such that $E^{-1}(E(x, K),K)=x$ holds for $\forall x \in \{0, 1\}^n$ and $\forall K \in \{0, 1\}^k$.
  - given x and K, then E(x,K) must be of **polynomial complexity**;
  - given x, then $E^{-1}(x,K)$ must be **computational infeasible**.
- If E is a random function ($E \in$ RF) then the cipher is **perfect.**



Block Cipher design criteria

- **completeness**
  - **each** bit of the output block should depend on **each** bit of the input block and on **each** bit of the key.

- **avalanche effect (Shannon, Feistel)**
  - changing **one bit** in the input block should change statistically **half of the bits in the output block**.
  - changing **one bit** in the key should change statistically **half of the bits in the output block**.



- **statistical independence**
  - input and output should appear to be statistically independent.

The magic of XOR

- XOR ($\oplus$) is the Exclusive OR. If $c=m \oplus k$:

  $(m=0)\oplus(k=0) = (c=0)$

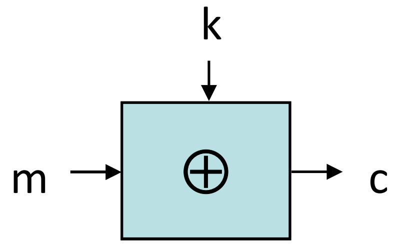
  $(m=0)\oplus(k=1) = (c=1)$

  $(m=1)\oplus(k=0) = (c=1)$

  $(m=1)\oplus(k=1) = (c=0)$

  $\mathbf{x \oplus 0 = x}$

  $\mathbf{x \oplus x = 0}$



- XOR is associative $(a \oplus b) \oplus c = a \oplus (b \oplus c)$ and commutative $a \oplus b = b \oplus a$
- XOR is the addition in GF(2).
- XOR is a **linear** boolean function: it returns 1 for an odd number of operand “1” ($0\oplus\mathbf{1}=\mathbf{1}\oplus 0=1$) and 0 for an even number of operand “1” ($0\oplus 0=\mathbf{1}\oplus\mathbf{1}=0$).
- If $c = m \oplus k$ then $m = c \oplus k$.

  Proof: $c = m \oplus k = (c \oplus k) \oplus k = c \oplus (k \oplus k) = c \oplus 0 = c$.
- XOR performs as (but **is not**) a **random function**.

  Proof: let $p_1= p(x=1)=p(y=1)$ and $p_0= 1-p_1=p(x=0)=p(y=0)$. By definition of the XOR operation we set $p(z=0)=p_0p_0+p_1p_1$ and $p(z=1)=p_0p_1+p_1p_0$. Setting $p(z=0)=p(z=1)$ replacing $p_0= 1-p_1$ and solving for p_1, the unique solution $p_0 =p_1= 0.5$ is obtained. This property does not hold for the other Boolean operators AND and OR.
- XOR is the basic ciphering unit for any developed (block) cipher.

Advanced Encryption Standard

Vincent Rijmen, Joan Daemen (Rijndael), 2001

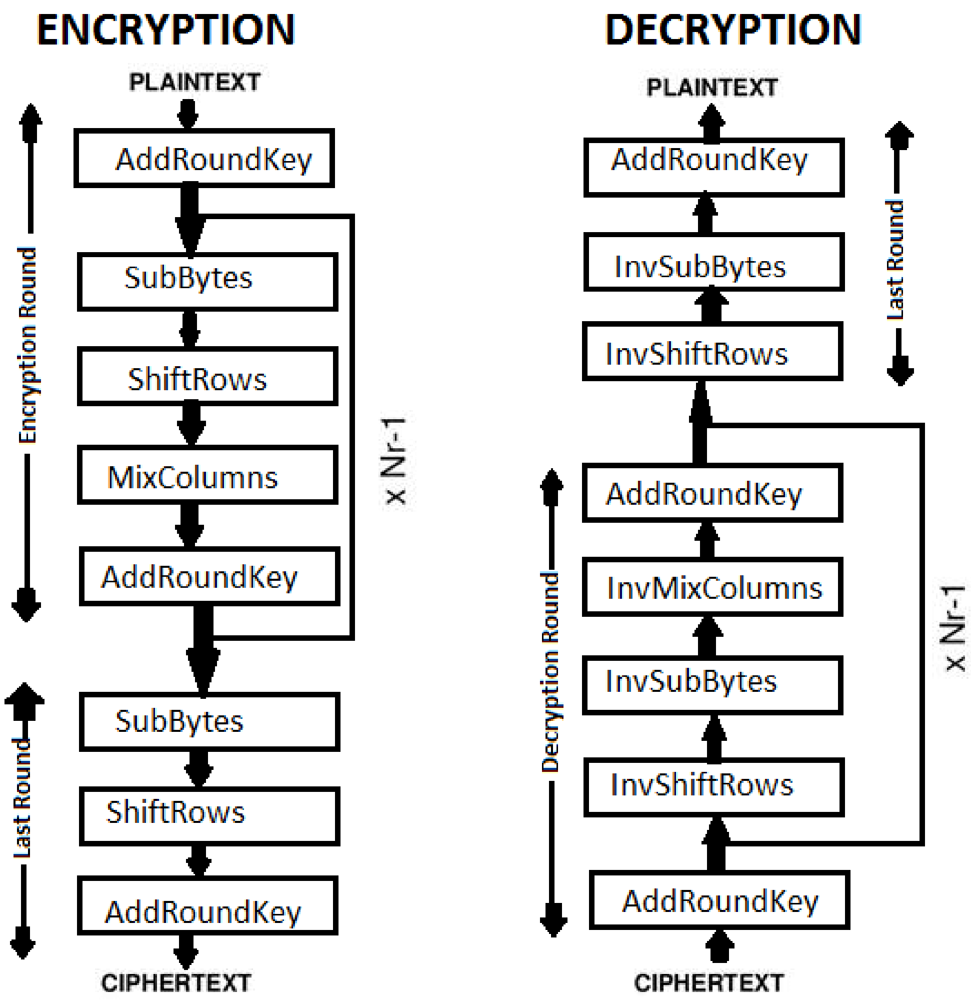
- **Advanced Encryption Standard** (**AES**) is a specification for the encryption of electronic data established by the U.S. National Institute of Standards and Technology (NIST) in 2001 with standard FIPS-197.
- Rijndael is a family of ciphers with different key and block sizes.
- For AES, NIST selected three members of the Rijndael family, each with a block size of 128 bits, but three different key lengths: 128, 192 and 256 bits.
- AES has been adopted by the U.S. Government and is now used worldwide.
- AES supersedes the **Data Encryption Standard (DES)**, 1977.

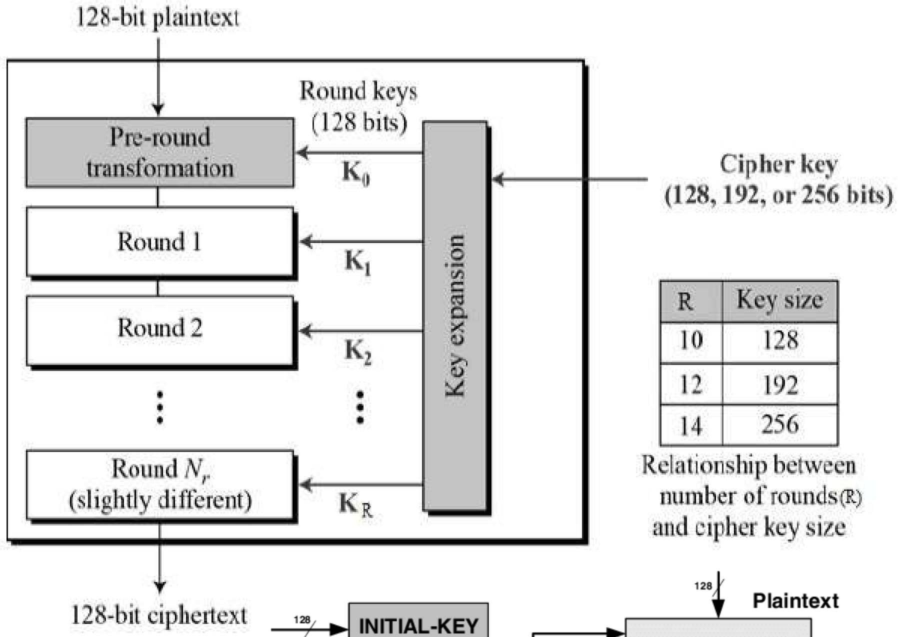
The Design of Rijndael: AES - The Advanced Encryption Standard
J. Daemen, V. Rijmen
Ed. Springer, ISBN 978-3-662-04722-4, 2002
https://autonome-antifa.org/IMG/pdf/Rijndael.pdf

Advanced Encryption Standard

- AES is based on the S-P design principle, fast in both software and hardware.
- AES operates on a 4 × 4 column-major order matrix of bytes, termed the *state*.
- AES calculations are done in $GF(2^8)/x^8+x^4+x^3+x+1$.
- The key size used for an AES cipher specifies the number of repetitions of transformation rounds that convert the plaintext into the ciphertext.
- The number of cycles of repetition are as follows:
  - 10 cycles of repetition for 128-bit keys.
  - 12 cycles of repetition for 192-bit keys.
  - 14 cycles of repetition for 256-bit keys.

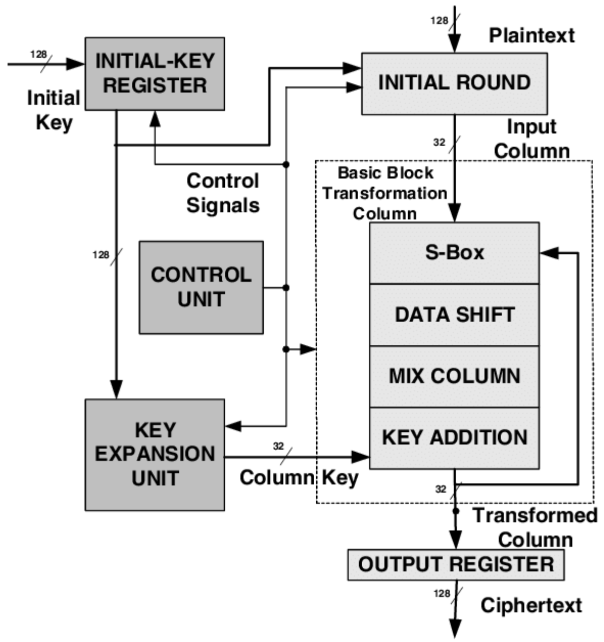
AES Encryption / Decryption Schemes





R	Key size
10	128
12	192
14	256

Relationship between number of rounds(R) and cipher key size

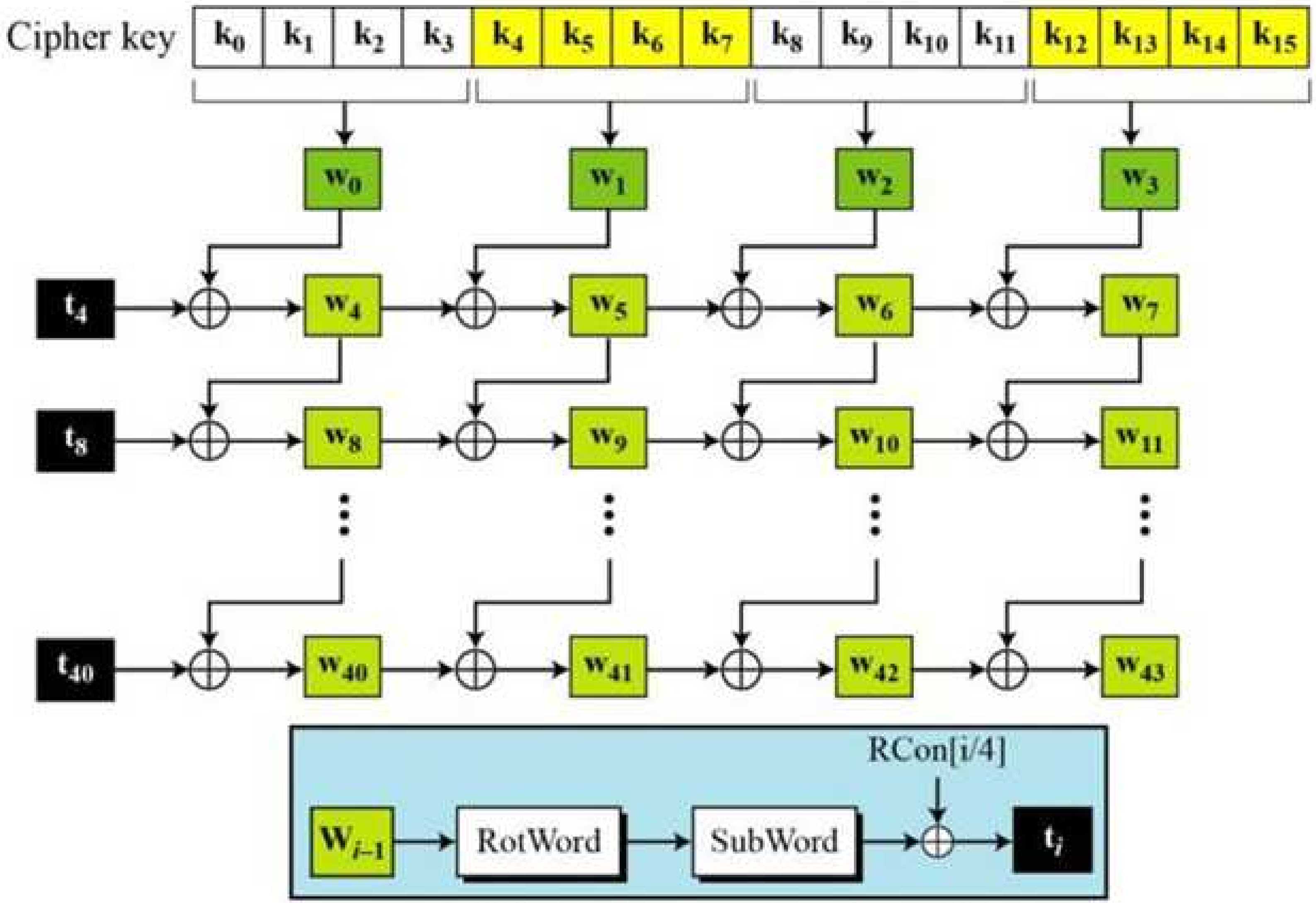


[Animated AES ciphering](#)

Advanced Encryption Standard

- KeyExpansion – round keys are derived from the cipher key using the ”Rijndael key schedule”: it takes a 128-bit key (4 words 32-bit each) and expands into an array of 44 words 32-bit each.
- Initial round key addition:
  - AddRoundKey – each byte of the state is combined with a byte of the round key using bitwise XOR.
- 9, 11 or 13 rounds:
  - SubBytes – a non-linear substitution step where each byte is replaced with another according to a lookup table.
  - ShiftRows – a transposition step where the last three rows of the state are shifted cyclically a certain number of steps.
  - MixColumns – a linear mixing operation which operates on the columns of the state, combining the four bytes in each column.
  - AddRoundKey
- Final round (making 10, 12 or 14 rounds in total):
  - SubBytes
  - ShiftRows
  - AddRoundKey

AES – Rijndael Key Schedule



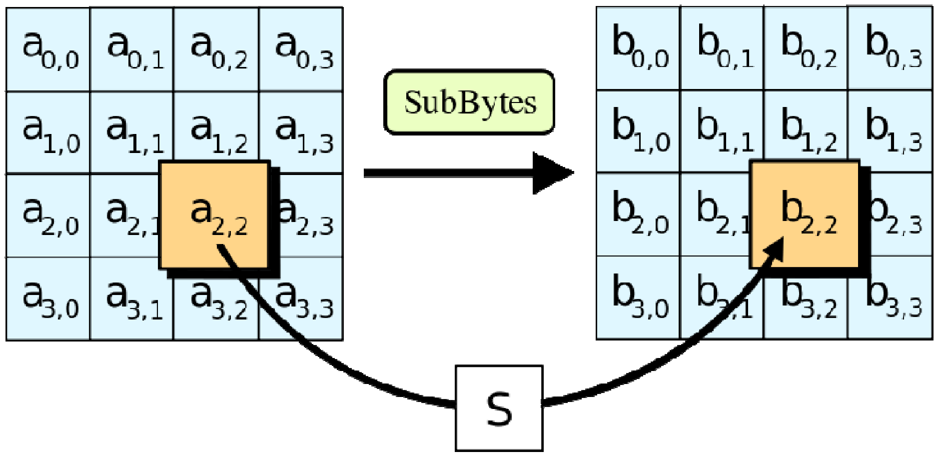
Making of t_i (temporary) words $i = 4\ N_r$

AES – SubBytes step

- In the SubBytes step, each byte in the state is replaced with its entry in a fixed 8-bit lookup table S (elements in $GF(2^8)$ are 8-bit length).
- In the SubBytes step, each byte a_{ij} in the state array is replaced with a SubByte $S(a_{ij})$ using an 8-bit substitution box. **This operation provides the non-linearity in the cipher** combining the multiplicative inverse in $GF(2^8)$ and an affine transformation. $S(a_{ij})$ is computed as follows: given a_{ij}, search along x-axis the row corresponding to its most significant 4 bits (0_{16}-f_{16}) and along y-axis the column corresponding to its less significant 4 bits (0_{16}-f_{16}). E.g. $S(53_{16}) = ed_{16}$.

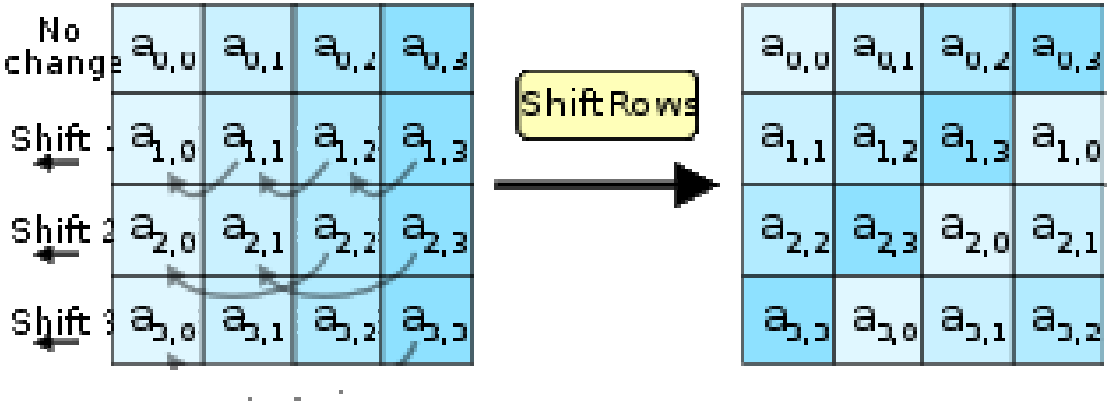
x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

[NIST-FIPS-197]



AES – ShiftRows step

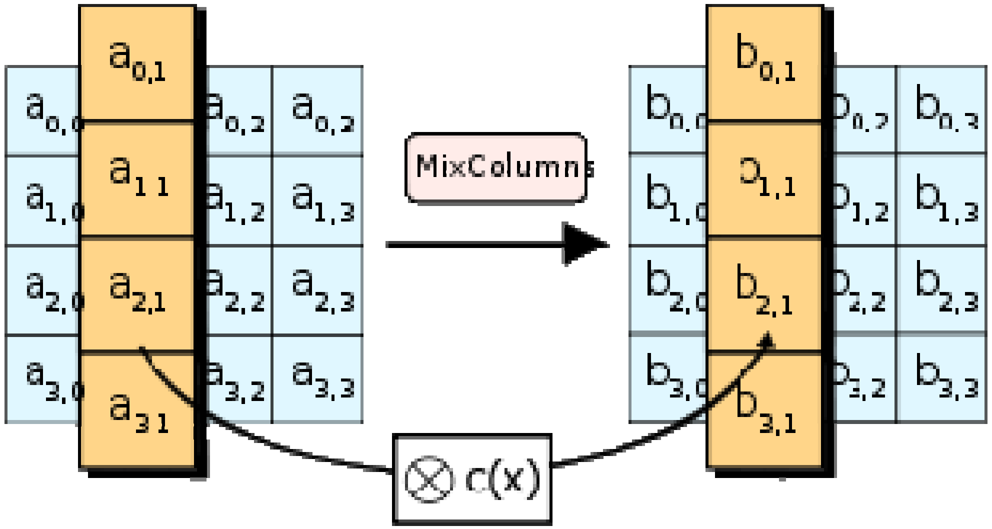
- In the ShiftRows step, bytes in each row of the state are shifted cyclically to the left. The number of places each byte is shifted differs incrementally for each row. It operates on the rows of the state; it cyclically shifts the bytes in each row by a certain offset.
  - The first row is left unchanged. Each byte of the second row is shifted one to the left. Similarly, the third and fourth rows are shifted by offsets of two and three respectively.
- In this way, **each column of the output state of the ShiftRows step is composed of bytes from each column of the input state**.
- The importance of this step is **to avoid the columns being encrypted independently**, in which case AES degenerates into four independent block ciphers.



AES – MixColumns step

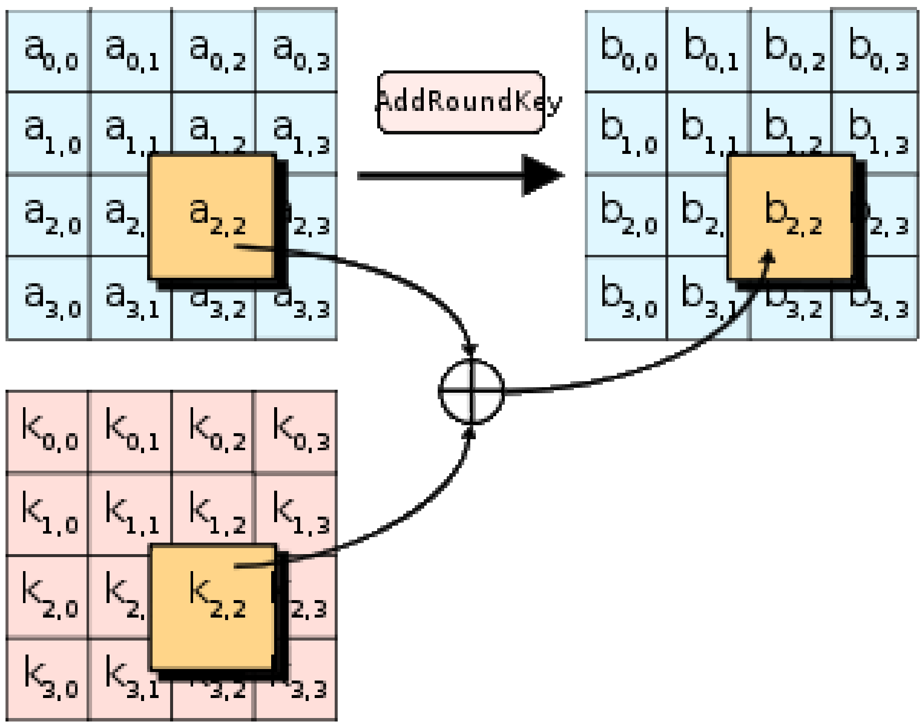
- In the MixColumns step, the four bytes of each column of the state are combined using an invertible linear transformation.
- The MixColumns function takes four bytes as input and outputs four bytes, where each input byte affects all four output bytes. Together with ShiftRows, **MixColumns provides diffusion in the cipher**.
- During this operation, each column is transformed using the fixed matrix c. Matrix multiplication is composed of multiplication and addition in $GF(2^8)/x^8+x^4+x^3+x+1$.

$$\begin{bmatrix} b_{0j} \\ b_{1j} \\ b_{2j} \\ b_{3j} \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \cdot \begin{bmatrix} a_{0j} \\ a_{1j} \\ a_{2j} \\ a_{3j} \end{bmatrix}$$



AES – AddRoundKey step

- In the AddRoundKey step, **the subkey is combined with the state**. For each round, a subkey is derived from the main key using **Rijndael's key schedule**; each subkey is the same size as the state. The subkey is added by combining each byte of the state with the corresponding byte of the subkey using bitwise XOR.



Security of AES

- The National Security Agency (NSA) reviewed all the AES finalists, including Rijndael, and stated that all of them were secure enough for U.S. Government **non-classified** data.
- In June 2003, the U.S. Government announced that AES could be used to protect **classified** information:

  The design and strength of all key lengths of the AES algorithm (i.e., 128, 192 and 256) are sufficient to protect classified information up to the SECRET level. TOP SECRET information will require use of either the 192 or 256 key lengths.

- There is currently no analytical attack from conventional computing against AES known which has a complexity less than a brute-force attack.

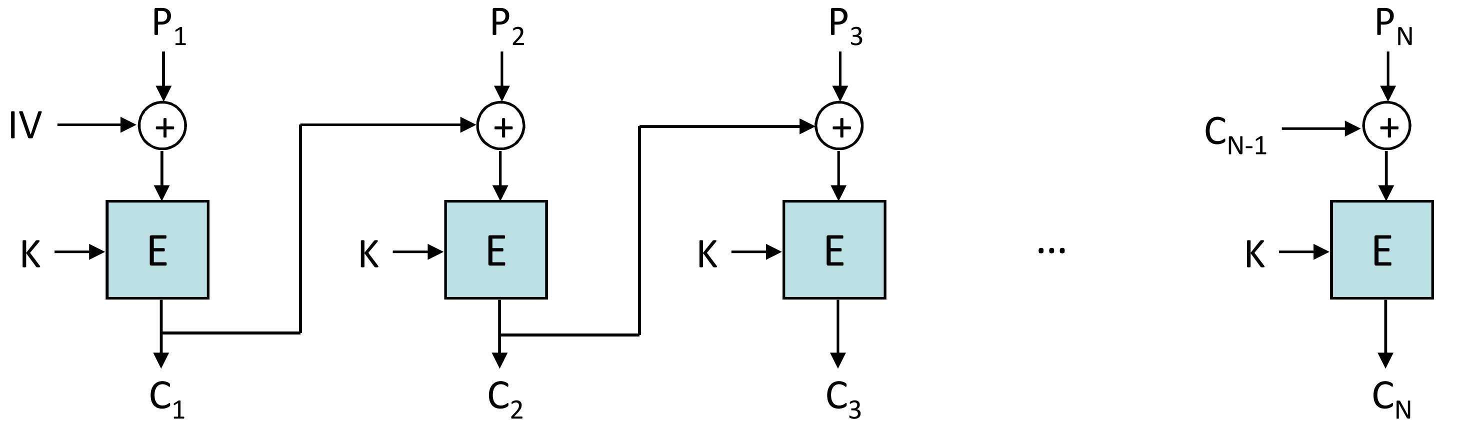
Block Cipher modes of operation

- A mode of operation describes how to repeatedly apply a cipher's single-block operation to securely transform amounts of data larger than a block

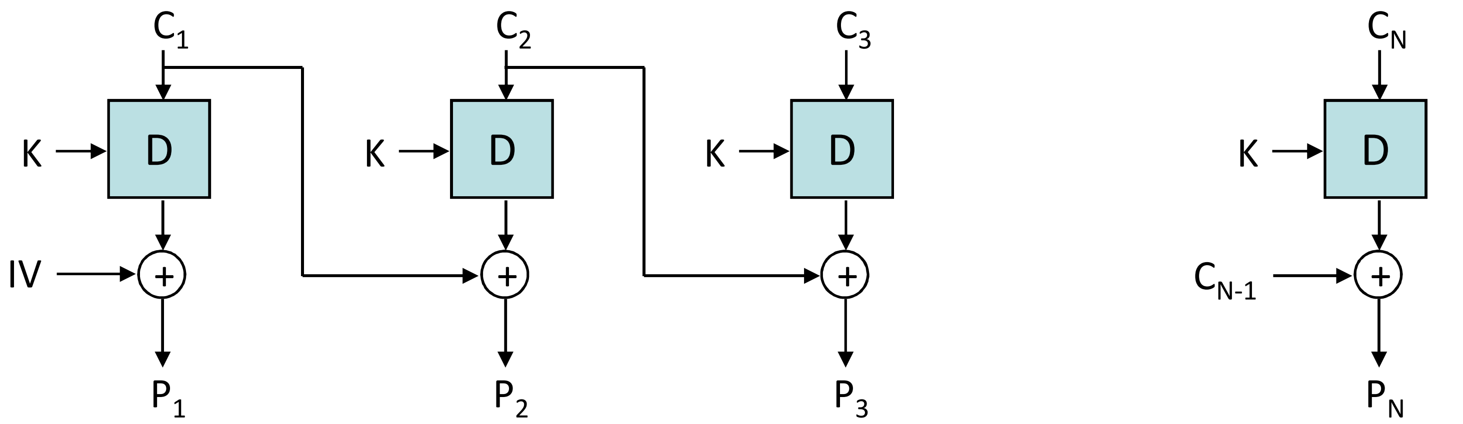
  - **CBC – Cipher Block Chaining**
  - **CTR – Counter**
  - ECB – Electronic Codebook
  - CFB – Cipher Feedback
  - OFB – Output Feedback

CBC mode

- Encrypt (E)



- Decrypt (D=E^{-1})



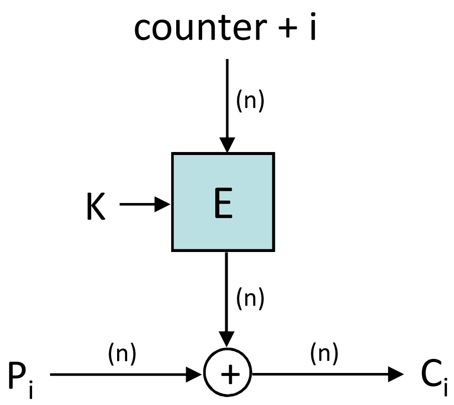
Properties of the CBC mode

Ehrsam, Meyer, Smith and Tuchman, 1976

- **Initialization Vector** (**IV**) is a fixed-size **pseudorandom** value.
- **Ciphertext block C_j depends on P_j and all preceding plain-text blocks.**
- The i-th block cannot be decrypted independently of the others.
  - not parallelizable
  - no random access
- **The IV should be encrypted** to avoid malicious modifications by an attacker to make predictable changes to the first plain-text block recovered.
- Decrypting with the incorrect IV causes the first block of plain-text to be corrupt **but subsequent plain-text blocks will be correct because each block is XORed with the cipher-text of the previous block, not the plain-text. As a consequence, decryption *can* be parallelized.**
- A one-bit change to the cipher-text causes complete corruption of the corresponding block of plain-text, and inverts the corresponding bit in the following block of plain-text (vulnerability to Padding Oracle attack).
- CBC is commonly used mode of operation: main drawbacks are that encryption is sequential (i.e., **encryption *cannot* be parallelized**), and that the message must be padded to a multiple of the cipher block size.

CTR mode

- Encrypt (E)
- Decrypt (D=E)



Properties of the CTR mode

Diffie, Hellman 1979.

- CTR mode uses a **counter rather than an IV** (with non-repeating requirement) or equivalently the IV contains a counter.
- Cycle 2^n length depends on the size of the counter.
- The i-th block can be decrypted independently of the others
  - parallelizable
  - random access
- The values to be XORed with the plaintext can be **precomputed.**
- CTR encrypts as decrypts.
- **At least as secure as the other modes: along with CBC, CTR mode is one of two block cipher modes recommended by Niels Ferguson and Bruce Schneier**.
- CTR mode is well suited to operate on a **multi-processor machine, where blocks can be encrypted in parallel**. If the IV/nonce is random, then they can be combined with the counter **using any invertible operation** (concatenation, addition) to produce the actual unique counter block for encryption. In case of a non-random nonce (such as a packet counter), the nonce and counter **should be concatenated** (not XORed).

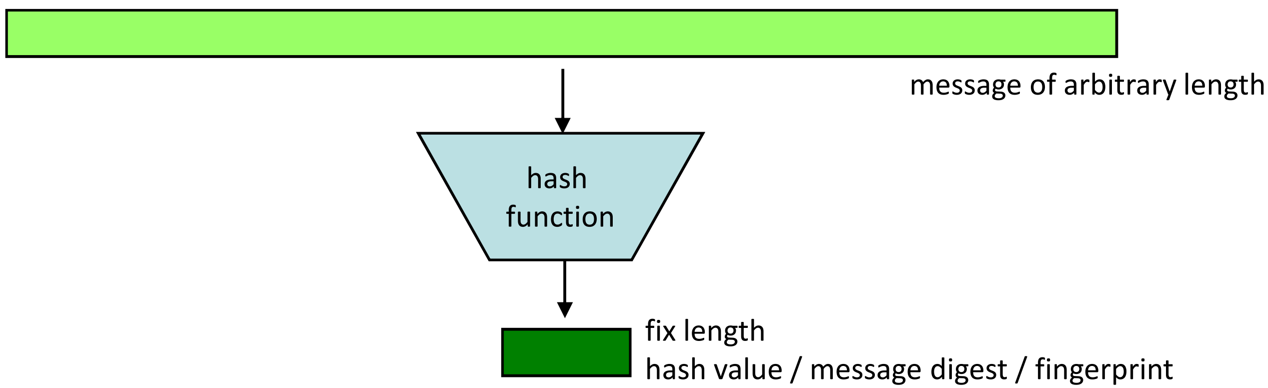
Outline

- Passive Security Functions
  - Ciphering
  - Hash functions
  - Message authentication codes
  - Digital signatures
- Key Establishment Protocols
  - Symmetric KEP
  - Asymmetric KEP
  - ID Based KEP
  - Hybrid KEP
  - Authentication of public key
- Key Management Protocols
  - TinySEC
  - TinyECC
  - TinyIBE
- Passive security techniques for
  - IEEE 802.15.4 MAC
  - Routing
  - ZigBee

Hash Functions

- An **Hash Function** maps bit strings of arbitrary finite length to bit strings of fixed length (n bits).
- The values returned by a hash function are called **hash values**, **hash codes**, **digests**, or simply **hashes**.
- Hash value of a message serves as a compact representative image of the message (fingerprint of the message).
- Many-to-one mapping → collisions are unavoidable but VERY RARE:

  E.g. for a 128 bit hash function, there 2^{128} possible outputs ($\approx 3.4 \cdot 10^{38}$): “only” after $k \approx 2.6 \cdot 10^{10}$ attempts the probability of at least one collision is $\varepsilon = 10^{-18}$ (derived from the Birthday Paradox Problem).



Structural Requirements of Hash Functions

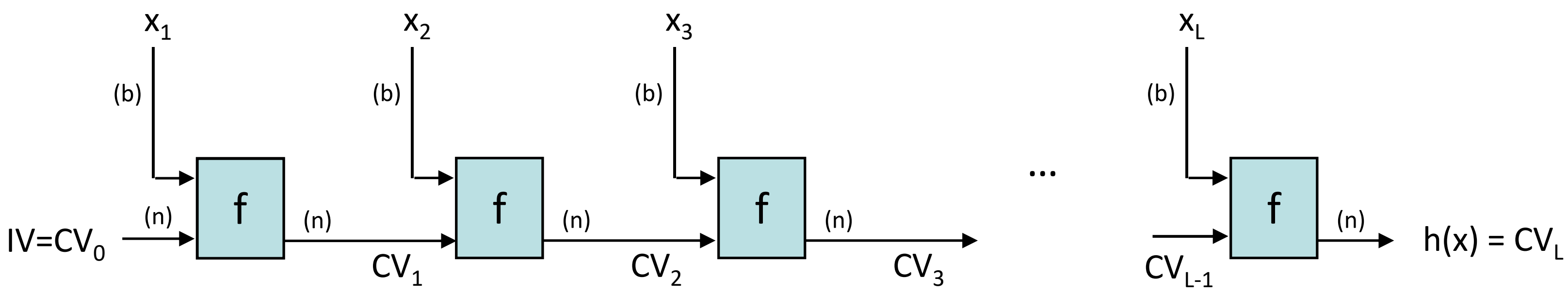
- **Ease of computation**
  - Given an input x, the hash value H(x) should be a low complexity algorithm.
- **One-way property (inverse image or preimage resistance)**
  - Given a hash value y, to find any input x s.t. H(x) = y should be a **computationally infeasible.**
- **Weak collision resistance (2nd preimage resistance)**
  - Given an input x, it should be **computationally infeasible** to find a second input x' such that H(x') = H(x).
- **Strong collision resistance (collision resistance)**
  - It should be **computationally infeasible** to find any two distinct inputs x and x' such that H(x) = H(x').

The compliance to **easy computation** and **one-way** requirements makes hash functions suited for cryptosystems (in this case are denoted as **cryptographic hash functions**).

Iterated Hash Functions

- Input is divided into fixed length blocks x_1, x_2, …, x_L (last block padded if necessary).
- f is called the **compression function,** cv the **compressed vector** (f maps an input x of arbitrary finite bit length, in this case n+b, to an output h(x) of fixed bit length n).
- Each stage of an iterated hash function compresses a block.
- The compression of the last stage returns the hash of the input.

- Each input block is processed according to the following scheme:



Secure Hash Algorithm

- **Secure Hash Algorithm** (**SHA**) is a family of Cryptographic Hash functions published by the National Institute of Standards and Technology (NIST) as a U.S. Federal Information Processing Standard (FIPS).

- Corresponding standard is FIPS 180-4 **Secure Hash Standard (SHS) (2015).** It specifies secure hash algorithms - SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 and SHA-512/256. However on March 2023, NIST has released a Planning Note saying that after two rounds of public comment, NIST has decided to revise FIPS 180-4.

Key Derivation Function

- **Key Derivation Functions** (**KDF**) are particular hash functions.
- It derives one or more secrets from a shared secret value.
- It is typically used to stretch keys into longer keys or to obtain keys of a required format.
- KDF(SS) = (k_1,k_2) where c = E_{k1}(m) and t = MAC_{k2}(c): it is used when different keys for encryption and authentication are needed.

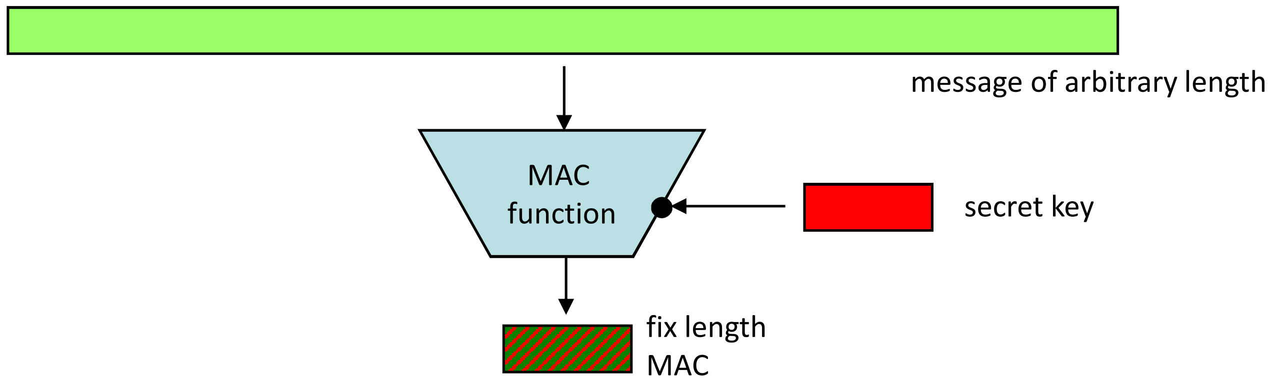
- KDFs are often used as components in KEPs.
- **KDF1-2-3-4 are standardized in ISO/IEC 18033-2**.

Outline

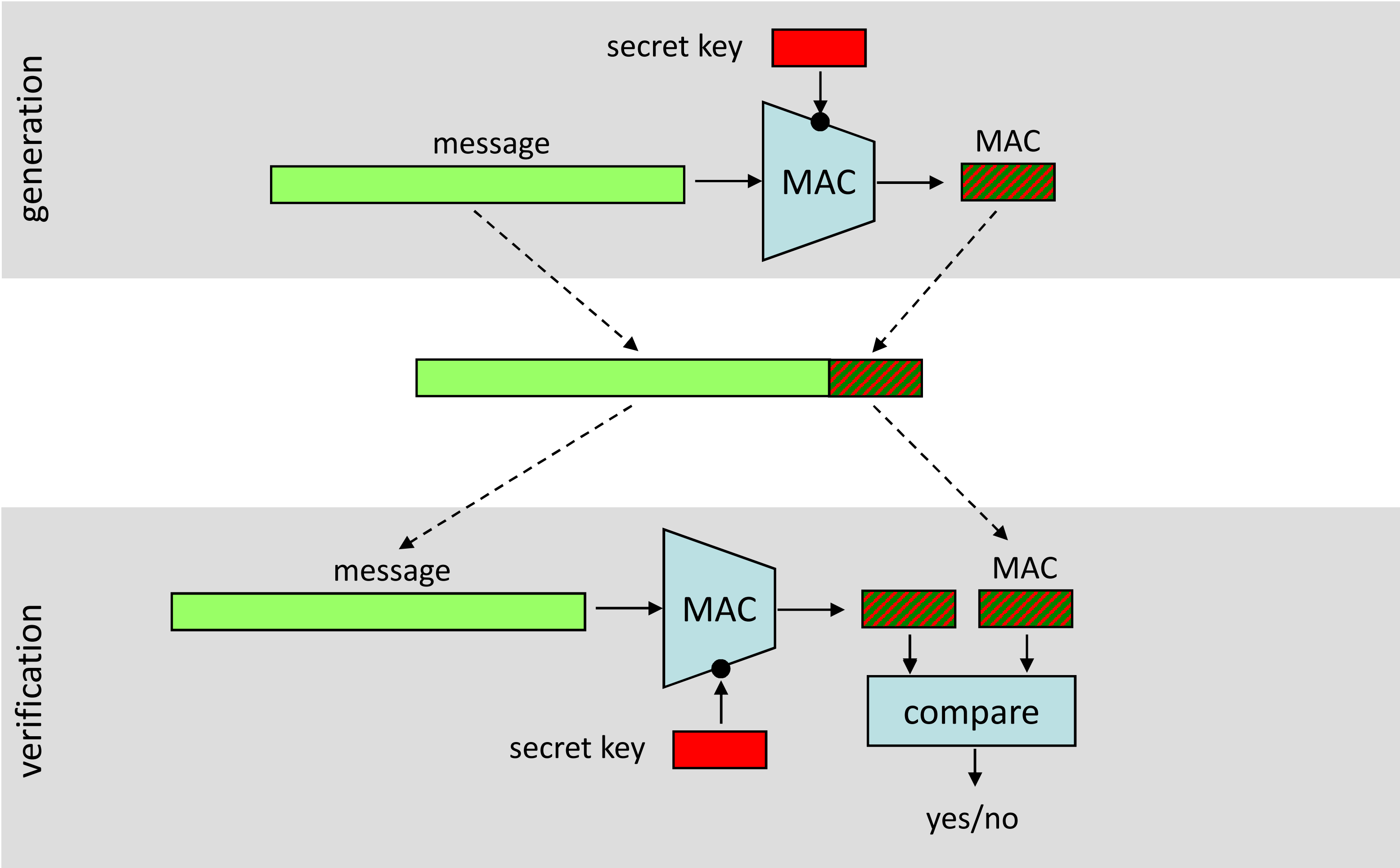
- Passive Security Functions
  - Ciphering
  - Hash functions
  - Message authentication codes
  - Digital signatures
- Key Establishment Protocols
  - Symmetric KEP
  - Asymmetric KEP
  - ID Based KEP
  - Hybrid KEP
  - Authentication of public key
- Key Management Protocols
  - TinySEC
  - TinyECC
  - TinyIBE
- Passive security techniques for
  - IEEE 802.15.4 MAC
  - Routing
  - ZigBee

Message Authentication Codes (MACs)

- **MAC is a specific application of cryptographic hash functions.**
- MAC functions are hash functions with two functionally distinct inputs: a message and a secret key.
- **This hash produces a fixed size output called the MAC.**
- From the properties of cryptographic hash functions:
  - it should be **computationally infeasible** to produce a correct MAC for a message without the knowledge of the secret key.
  - it should be **computationally infeasible** to find any two distinct inputs x and x' such that H(x) = H(x').
- MAC functions are often used to implement data integrity services.



MAC generation and verification



Structural Requirements of MAC functions

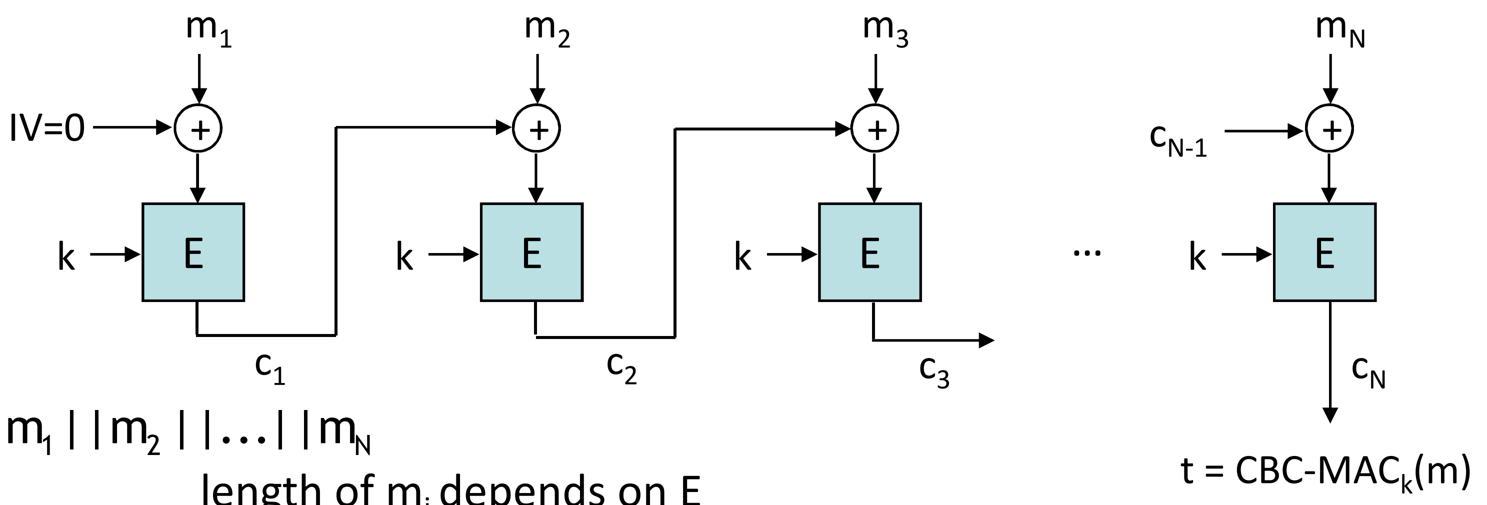
The same for hash functions plus "Key non-recovery" requirement.

- **Ease of computation**
  - Given an input x and a key k, the hash value $MAC_k(x)$ should be a low complexity algorithm.
- **One-way property (inverse image or preimage resistance)**
  - Given a hash value y and a key k, to find any input x s.t. $MAC_k(x) = y$ should be a **computationally infeasible.**
- **Weak collision resistance (2nd preimage resistance)**
  - Given an input x and a key k, it should be **computationally infeasible** to find a second input x' such that $MAC_k(x') = MAC_k(x)$.
- **Strong collision resistance (collision resistance)**
  - It should be **computationally infeasible** to find any two distinct inputs x and x' and the same key k such that $MAC_k(x) = MAC_k(x')$.
- **Key non-recovery**
  - it should be **computationally infeasible** to recover the key k, given one or more pairs $(x_i, MAC_k(x_i))$ for that k.

EXEMERGE CELLENCE

UNIVERSITÀ DEGLI STUDI DELL'AQUILA

CBC-MAC vs. CMAC



$m = m_1 || m_2 || \ldots || m_N$

length of m_i depends on E

- It's a technique for constructing a MAC from a block cipher in CBC mode.
- **If the block cipher E is secure (e.g. AES) then CBC-MAC is proven to be secure for fixed-length messages (N blocks m_i).**
- CMAC (Cipher-based MAC) is recommended by NIST for **variable-length messages** and fixes security vulnerability of CBC-MAC applied to variable-length messages, even if it requires more keys.

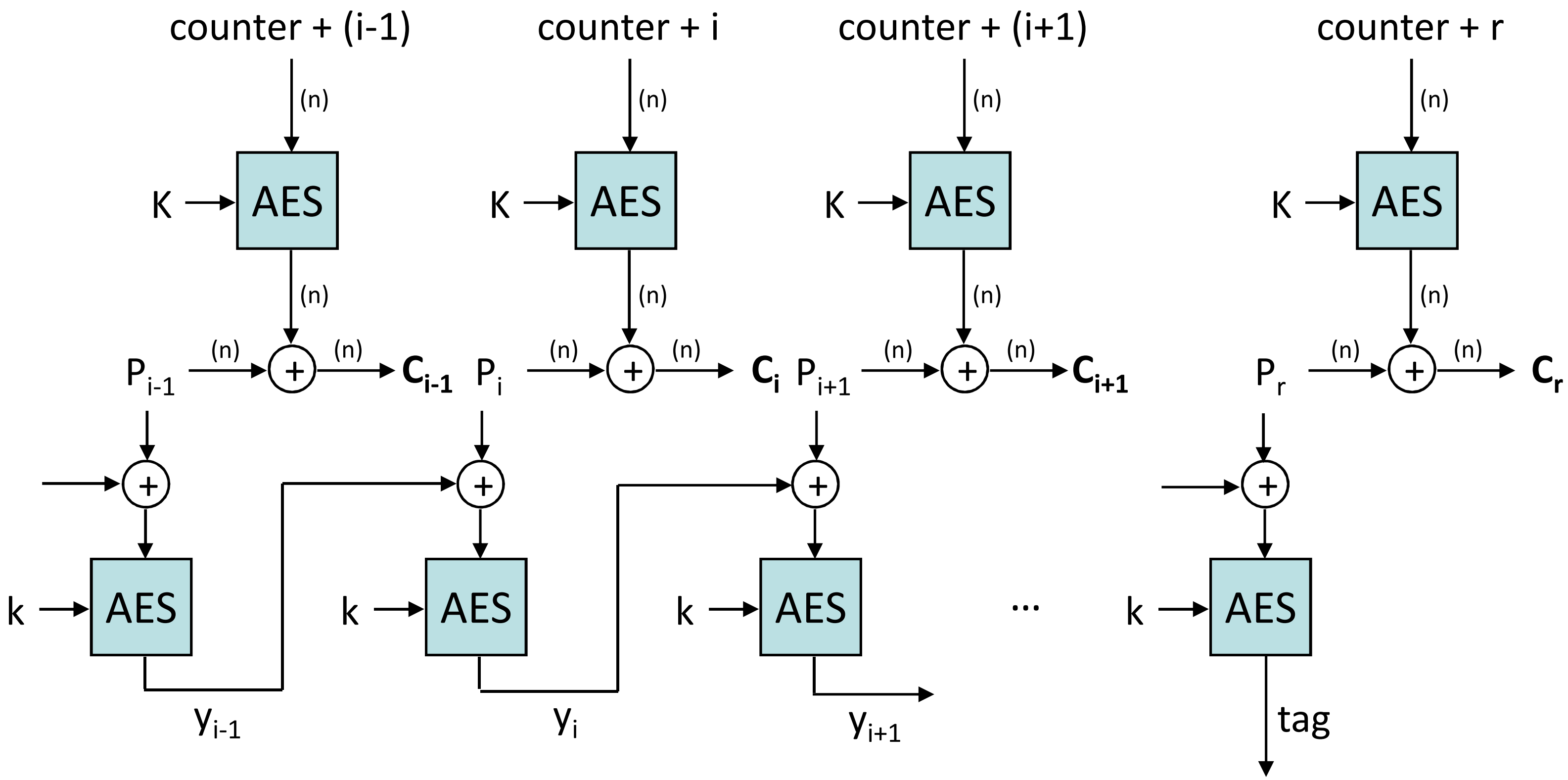
- KDF(SS) = (k_1,k_2) where c = $AES_{k1}(m)$ and t = $\text{AES-CBC-MAC}_{k2}(c)$ or t = $\text{CMAC}_{k2}(c)$

AES-CCM mode

- AES-CCM provides **both encryption and authentication** using the AES block cipher. This is a widely used mode since it requires only a single cryptographic primitive. That primitive is used in two different modes: CBC and CTR mode. The following shows how AES-CCM generally works:
  - **AES-CBC mode (AES-CBC-MAC)** is used to generate a **nice** "**authentication tag**". If a single byte changed anywhere in the data fed into the AES-CBC block, the final output will differ.
  - **AES-CTR mode** is used for the **actual data encryption**. Note AES-CTR encryption and decryption is the same operation, as AES-CTR is basically generating a unique "pad" we XOR with the data.
- Additional usage information:
  - **A nonce format is required for AES-CTR**. This nonce can be based on information in the packet, such as source address, or be random.
  - **An IV is required for the AES-CCM block**. This IV can be sent (possibly encrypted) to the AES-CCM block, or be part of secret information stored in the bootloader.
- A minor variation of CCM, called CCM*, is used in the Zigbee standard. CCM* includes all of the features of CCM and additionally offers encryption-only capabilities.

AES-CCM mode

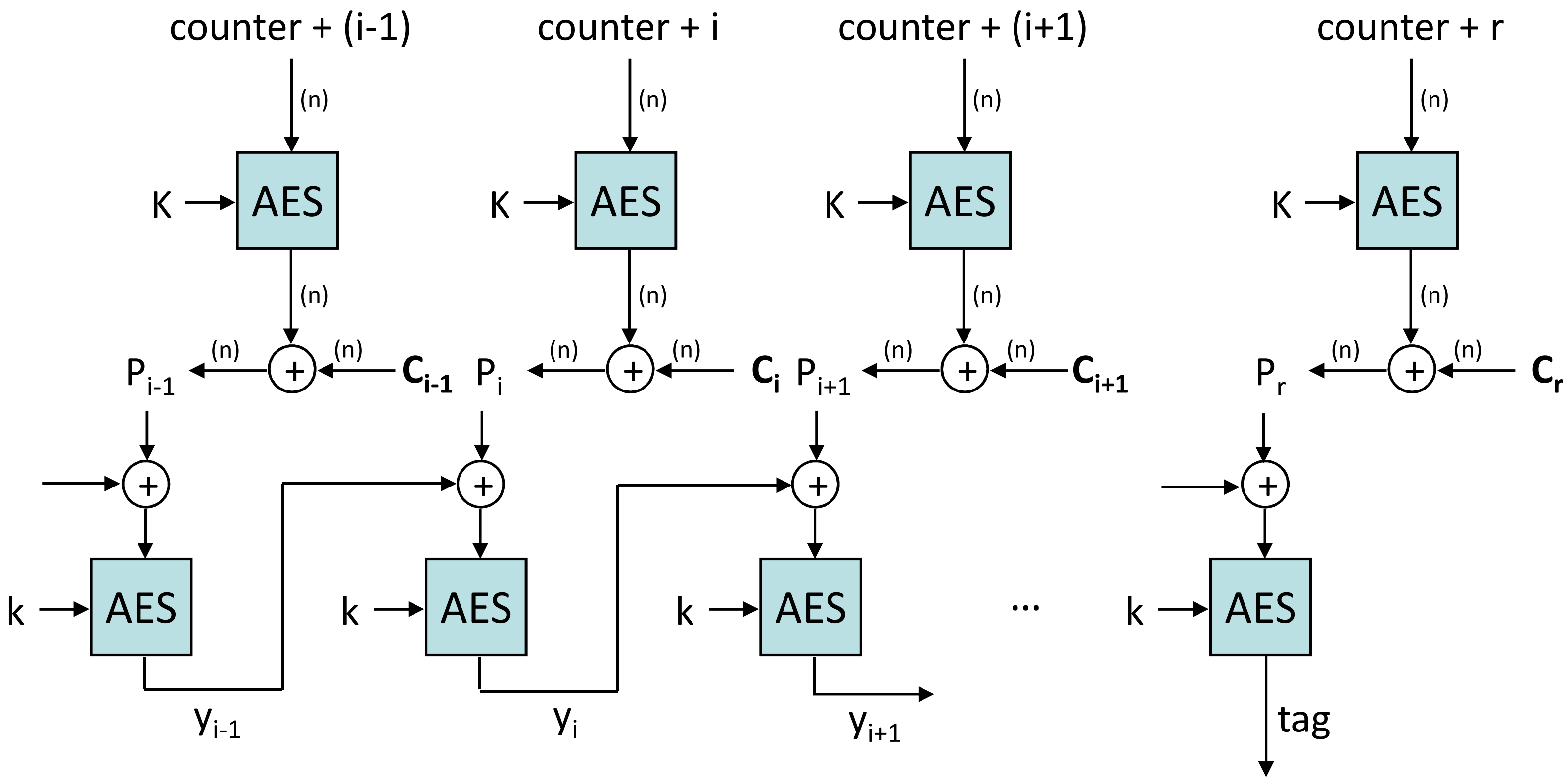
- encrypt



expected tag = $\text{CBC-MAC}_k(P_{i-1}|P_i|P_{i+1}\ldots|P_r)$

AES-CCM mode

- decrypt



computed tag = $\text{CBC-MAC}_k(P_{i-1}|P_i|P_{i+1}...|P_r)$

AES-CCM mode

- The “computed tag" and “expected tag" are compared together, **and only if they match is the decrypted data used**. A change of any of the data blocks OR the header would change the calculated tag, resulting in an error.
- Some nice features of AES-CCM:
  - Can decrypt any data block, or decrypt blocks out of order due to AES-CTR usage.
  - Authentication Tag provides authentication that data has not been modified in transit.
  - Auth tag can include non-encrypted information, such as a header with address or length information.
  - Auth tag can be shortened (i.e., not full 16-byte length) for use with protocols with very sensitive length limitations.

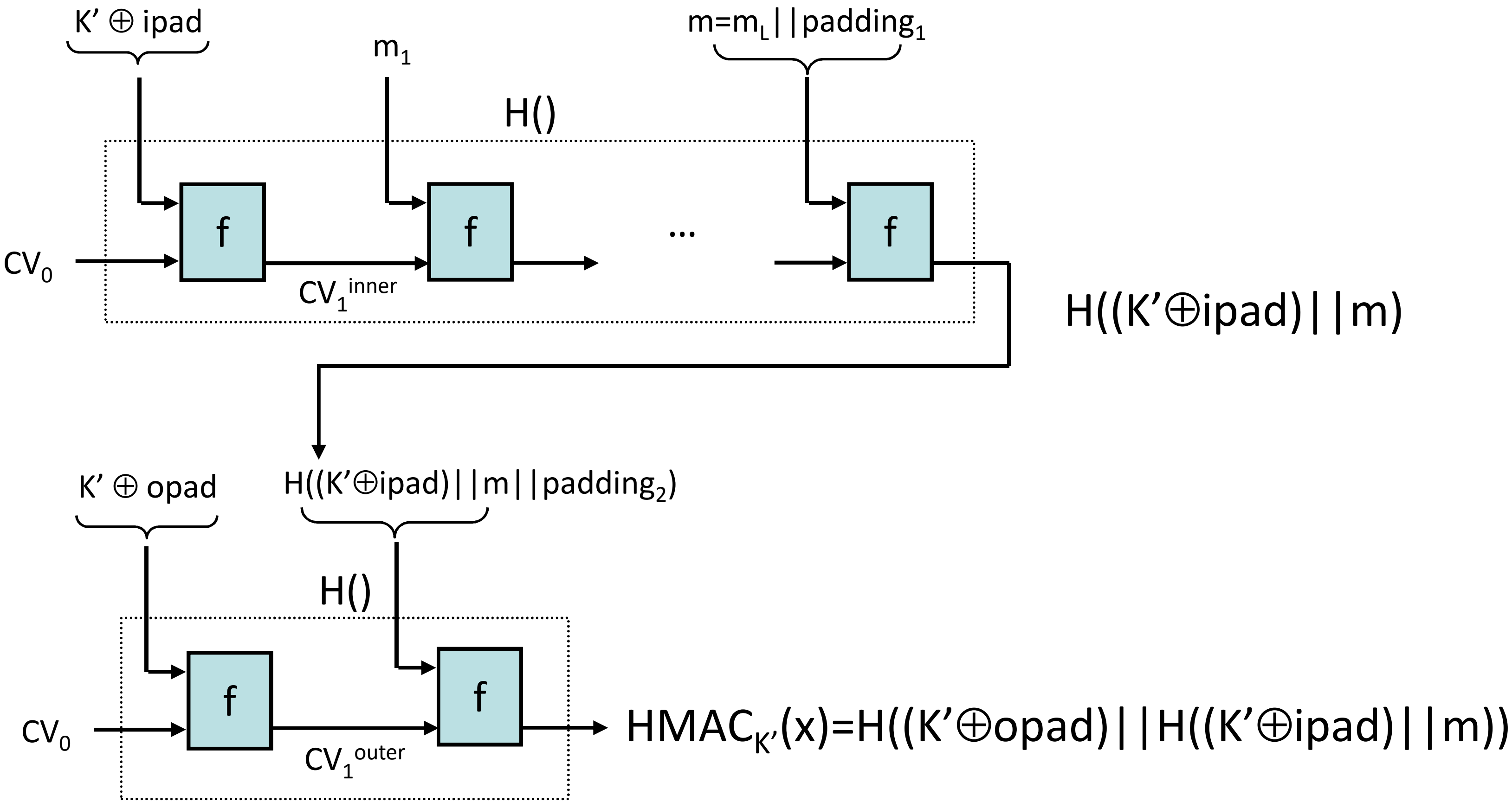
HMAC

- A **keyed-Hash Message Authentication Code** (**HMAC**) is a technique for constructing a MAC from a cryptographic iterated hash function H() in combination with a cryptographic key (FIPS-PUB 198-1)
- HMAC is denoted as HMAC-<name of hash function>
  - e.g., HMAC-SHA-1, HMAC-SHA-2, HMAC-SHA-256.

- K' = K if |K| = block_size
- K' = K + zero padding if |K| < block_size
- K' = H(K) if |K| > block size, therefore |H(K)| = block_size
- m the message to be authenticated with |m| = block_size
- ipad (inner pad) = 00110110 repeated block_size/8 times
- opad (outer pad) = 01011100 repeated block_size/8 times
- ⊕ is the XOR operator
- || is the chain operator

$$HMAC_{K'}(m) = H((K' \oplus opad) || H((K' \oplus ipad) || m))$$

HMAC


K' ⊕ ipad
m1
m=mL||padding1
H()
CV0
f
f
...
f
CV1inner
H((K'⊕ipad)||m)
K' ⊕ opad
H((K'⊕ipad)||m||padding2)
H()
CV0
f
f
CV1outer
HMACK'(x)=H((K'⊕opad)||H((K'⊕ipad)||m))

Outline

- Passive Security Functions
  - Ciphering
  - Hash functions
  - Message authentication codes
  - Digital signatures
- Key Establishment Protocols
  - Symmetric KEP
  - Asymmetric KEP
  - ID Based KEP
  - Hybrid KEP
  - Authentication of public key
- Key Management Protocols
  - TinySEC
  - TinyECC
  - TinyIBE
- Passive security techniques for
  - IEEE 802.15.4 MAC
  - Routing
  - ZigBee

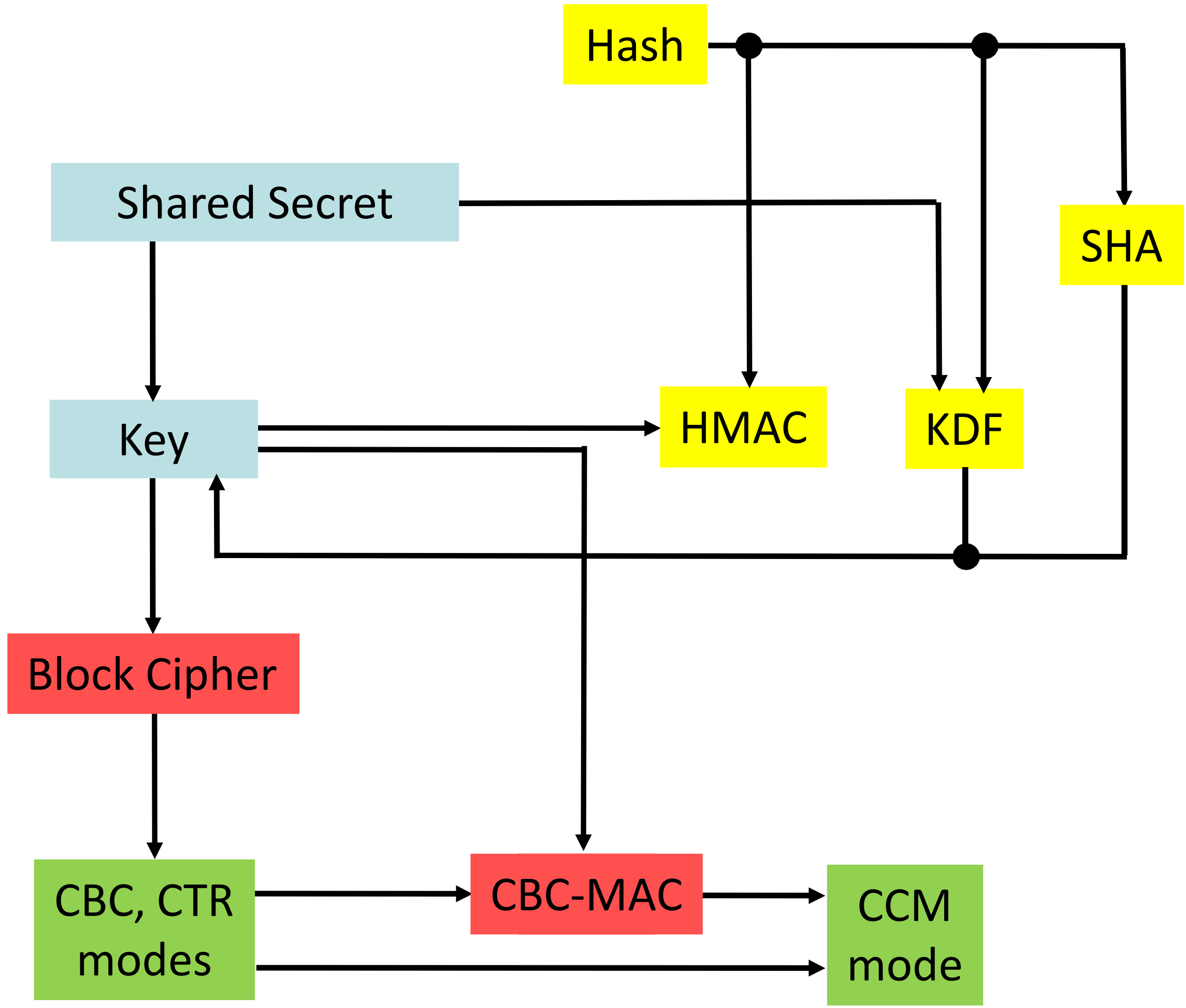
Digital Signatures

- DS is used for the authentication and non-repudiation of message origin (sender).

- DS is based on **public-key cryptography**.
  - private key PRK_A of the signer A defines a **signing transformation S**
    - $S(m, PRK_A) = \sigma$
  - public key PUK_A of the signer A defines a **verification transformation V**
    - $V(m, PUK_A) = \sigma$ is true then "signature accepted"
    - $V(m, PUK_A) = \sigma$ is false then "signature refused".

- DS protocols must satisfy these properties:
  - **Completeness**: if S is true, the honest verifier will be convinced of this fact.
  - **Soundness**: if S is false, no cheating prover can convince the honest verifier that it is true.
  - **Zero-knowledge**: if S is true, no cheating verifier learns anything other than this fact.

Summary


Hash
Shared Secret
SHA
Key
HMAC
KDF
Block Cipher
CBC, CTR modes
CBC-MAC
CCM mode

Outline

- Passive Security Functions
  - Ciphering
  - Hash functions
  - Message authentication codes
  - Digital signatures
- **Key Establishment Protocols**
  - Symmetric KEP
  - Asymmetric KEP
  - ID Based KEP
  - Hybrid KEP
  - Authentication of public key
- Key Management Protocols
  - TinySEC
  - TinyECC
  - TinyIBE
- Passive security techniques for
  - IEEE 802.15.4 MAC
  - Routing
  - ZigBee

Key Establishment Protocols

- A **Key Establishment Protocol** (**KEP**), or Key Agreement Protocol, among parties generates a **ciphering key shared among parties** (shared secret) by exchanging **information which does not reduce uncertainty on the generated ciphering key: H(K|exchanged information for Key Generation) ≈ H(K)**

- Key Establishment reduces to Key Distribution when a trusted party directly and autonomously generates and distributes the secrets associated to be shared among parties (not suitable for WSN).

  Key Establishment = Key Transport + Key Processing (Key Generation)

- Requirements for KEPs in a WSN are:
  - **Key Transport**: should be based on a state machine as simpler as possible, (req/conf or 2-phase), better if stateless (1-phase without confirmation) robust against topology changes, independent on communication patterns and key types.
  - **Key Processing**: should be as lighter as possible for energy constrained devices.

The Need of KEPs

- **Trusted Key Distribution Center (KEP reduces to a KDP, Key Distribution Protocol)**
  - Ciphering keys are **not generated by the parties but by a (trusted) third party**.
  - Parties ask KDC for a key.
  - Access Point can relay requests to KDC but it becomes a single point of failure.
  - However ciphering keys must be transmitted by TTP over a secure link: a preloaded **network key** should considered for the start-up: this represents a vulnerability (spoofing threat).

- **TKDC can be considered only if TTP can be authenticated** i.e. the TKDC should sign the generated ciphering key and the receiver party should verify sender authenticity (sender integrity).

- Pure ad-hoc networks need KEPs to self generate shared ciphering keys.
- Hybrid ad-hoc networks (with Access Point) can employ a KDP for shared ciphering keys.

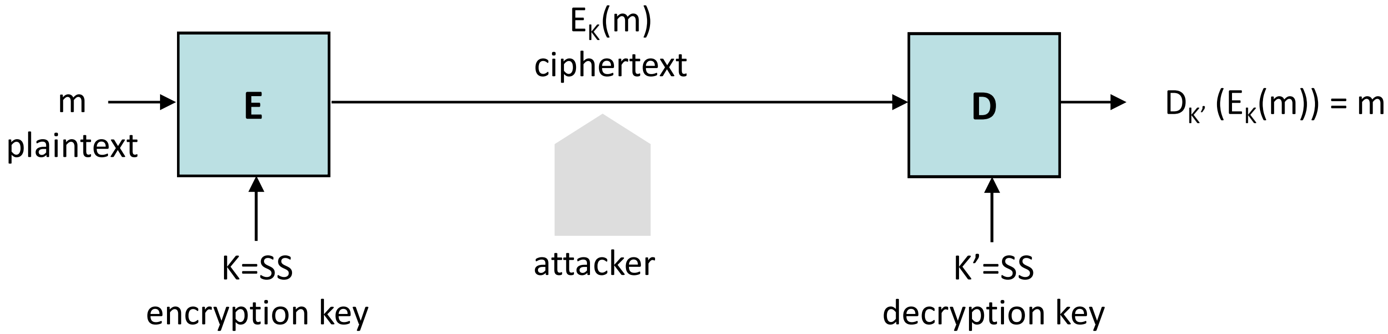
Key Authentication Problem

- **A certificate C (containing the credentials) binds a name A to the key K and a lifetime T and is represented as a triple C = (A, K, T).**
- A trusted entity is the only legitimated authority to **generate credentials and to release signed certificates.**
- Trusted entities can be
  - **Key Distribution Centers** for the generation and distribution of authenticated secrets in *symmetric key schemes*.
  - **Public Key Generators** (**Certification Authorities)** for the generation of authenticated public keys in *asymmentric key schemes*.
  - **Private Key Generators** for the generation of authenticated private keys in (*asymmetric*) *identity-based schemes*.

Outline

- Passive Security Functions
  - Ciphering
  - Hash functions
  - Message authentication codes
  - Digital signatures
- Key Establishment Protocols
  - Symmetric KEP
  - Asymmetric KEP
  - ID Based KEP
  - Hybrid KEP
  - Authentication of public key
- Key Management Protocols
  - TinySEC
  - TinyECC
  - TinyIBE
- Passive security techniques for
  - IEEE 802.15.4 MAC
  - Routing
  - ZigBee

Symmetric-key Scheme



- Symmetric-key scheme for encryption/decryption
  - **Shared Secret (SS)** = K' = K
- Each party shares the secret K generated by a suited KEP.
- A good symmetric key scheme should avoid:
  - key preload on the party
  - key transmission over unsecure channels.
- **Here shared secret management is a vulnerability**

EXEMERGE CELLENCE

Classic Symmetric KEPs for WSN

- **Probabilistic Key Establishment Protocols**
  – **Random Key Pre-Distribution scheme**

    L. Eschenauer, V.Gligor, "A Key Management Scheme for Distributed Sensor Networks," Proc. of the 9th ACM Conference on Computer and Communication Security, pages 41-47, 2002

  – **q-composite rand key pre-distribution**

    H. Chan, A. Perrig, D. Song, "Random Key Predistribution Schemes for Sensor Networks," Proc. of the 2003 IEEE Symposium on Security and Privacy, pages 197-213, 2003

- **Deterministic Key Establishment Protocols**
  – **Polynomial based key pre-distribution**

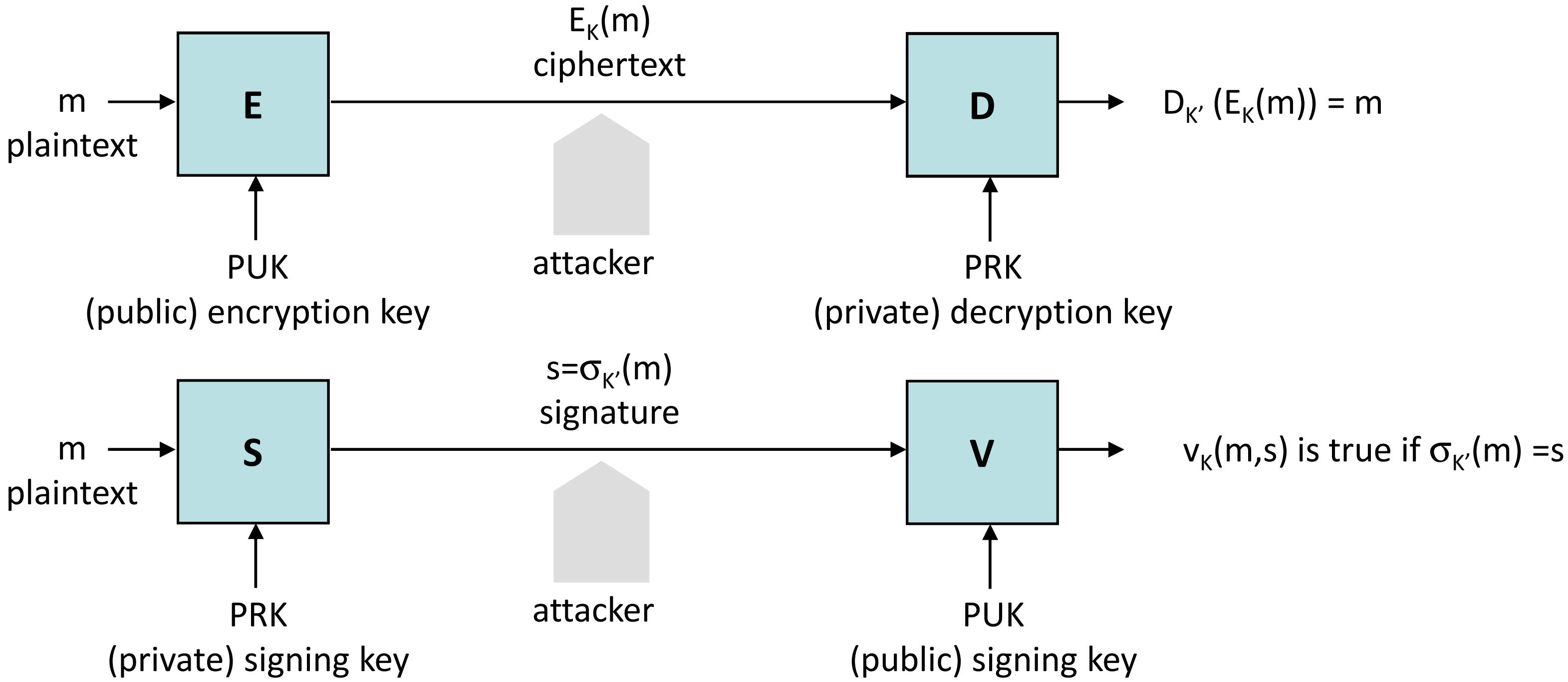
    R.Blom, "An Optimal Class of Symmetric Key Generation Systems," Proc. of EUROCRYPT, pages 335–338, 1984

    C. Blundo, A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung, "Perfectly-Secure Key Distribution for Dynamic Conferences," Proc of CRYPTO, pages 471–486, 1992

Outline

- Passive Security Functions
  - Ciphering
  - Hash functions
  - Message authentication codes
  - Digital signatures
- Key Establishment Protocols
  - Symmetric KEP
  - Asymmetric KEP
  - ID Based KEP
  - Hybrid KEP
  - Authentication of public key
- Key Management Protocols
  - TinySEC
  - TinyECC
  - TinyIBE
- Passive security techniques for
  - IEEE 802.15.4 MAC
  - Routing
  - ZigBee

Asymmetric-key Scheme



- Asymmetric-key scheme for encryption/decryption/signature.
  - PRK is **private** (i.e. secret), PUK is **public**
  - PUK=f(PRK), f **one-way function:** as PRK=f^{-1}(PUK), secrecy of PRK relies on the hardness of the inverse problem f^{-1}
- An authority that assigns private / public keys is needed. Encryption and decryption are generally energy consuming functions.

Asymmetric-key Scheme: RSA

- **RSA key generation**
  - Given p, q secret primes, e.g. p=3 and q=11
  - Compute n = pq, e.g. $n = 3 \cdot 11 = 33$, n is public
  - Compute $\varphi(n) = (p-1)(q-1)$, $\varphi(n)$ is public, e.g. $\varphi(33) = 2 \cdot 10 = 20$
  - Choose an integer e such that $1 < e < \varphi(n)$ and $\gcd(e, \varphi(n)) = 1$, i.e. e and $\varphi(n)$ are coprime. **Public key (or public exponent) is e**, e.g. 7, Public key = 7.
  - Determine d as $d \equiv e^{-1} \bmod \varphi(n)$ or $d \cdot e \bmod \varphi(n) = 1$. **Private key (or private exponent) is d**, e,g, d = 3 ($7 \cdot 3 \bmod 20 = 1$). Private key = 3.

- **RSA encryption and decryption**
  - Suppose the plaintext message is m =15
  - Encryption: **$c=m^e \bmod n$**, $c = 15^7 \bmod 33 = 27$
  - Decryption: **$m=c^d \bmod n$**, $m = 27^3 \bmod 33 = 15$

- To avoid RSA encryptions / decryptions, the sender party can generate and encrypt a pseudorandom as a new shared secret, transmit it the receiver party which decrypts it. Now a symmetric scheme is obtained.

Outline

- Passive Security Functions
  - Ciphering
  - Hash functions
  - Message authentication codes
  - Digital signatures
- Key Establishment Protocols
  - Symmetric KEP
  - Asymmetric KEP
  - ID Based KEP
  - Hybrid KEP
  - Authentication of public key
- Key Management Protocols
  - TinySEC
  - TinyECC
  - TinyIBE
- Passive security techniques for
  - IEEE 802.15.4 MAC
  - Routing
  - ZigBee

Identity-Based Encryption and Signature

- **Identity-Based Encryption (IBE)** and **Identity-Based Signature (IBS)** are an old idea originally proposed by Adi Shamir, co-inventor of the RSA Algorithm, in 1984.
- Driving concepts:
  - **Public key is based on public domain parameters** (e.g. mail address).
  - **Private key is generated on request by a Trusted Third Party** (Private Key Generator).
  - **No need of a Certification Authority: each party can execute an Authentication Check for its own private key.**
- Vulnerabilities:
  - Key Escrow Problem
  - Private Key transmission over unsecure channels

A. Shamir, Identity-Based Cryptosystems and Signature Schemes
Proceedings of Crypto 1984 on Advances in Cryptology, pp. 47-53
https://discovery.csc.ncsu.edu/Courses/csc774-S08/reading-assignments/shamir84.pdf

Basic concepts of IBE

- The sender Alice can **use the receiver's public key which can be directly known (without a CA) by the interested parties** to encrypt a message**:** it can be an email address or a vehicle plate.
- The receiver Bob **obtaines a private key PRK_{Bob}** from the TTP Private Key Generator (PKG) or and can decrypt the received ciphertext. **PRK can be preloaded at Bob's side.**
- An IBE scheme can be described with the following steps.

1. **Setup**: The PKG generates its **private key PRK_{PKG}** and its **public (or master) key PUK_{PKG}** pair (note that **PUK_{PKG}** is publicly known).
2. **Private Key Generation**: The receiver Bob authenticates himself to PKG and obtains an authenticated private key PRK_{Bob}. Bob performs a **Private Key Authentication Check** to verify if $\hat{e}(PRK_{Bob}, G) = \hat{e}(ID_{Bob}, \text{Master Key})$ holds.
3. **Encryption**: Using Bob's identity ID_{Bob} ($=PUK_{Bob}$) and PUK_{PKG} (master key), the sender Alice encrypts her plaintext M and obtains a ciphertext C.
4. **Decryption**: Upon receiving the ciphertext C from Alice, Bob decrypts it using his private key $prik_{Bob}$ and the Master Key to recover the plaintext M.

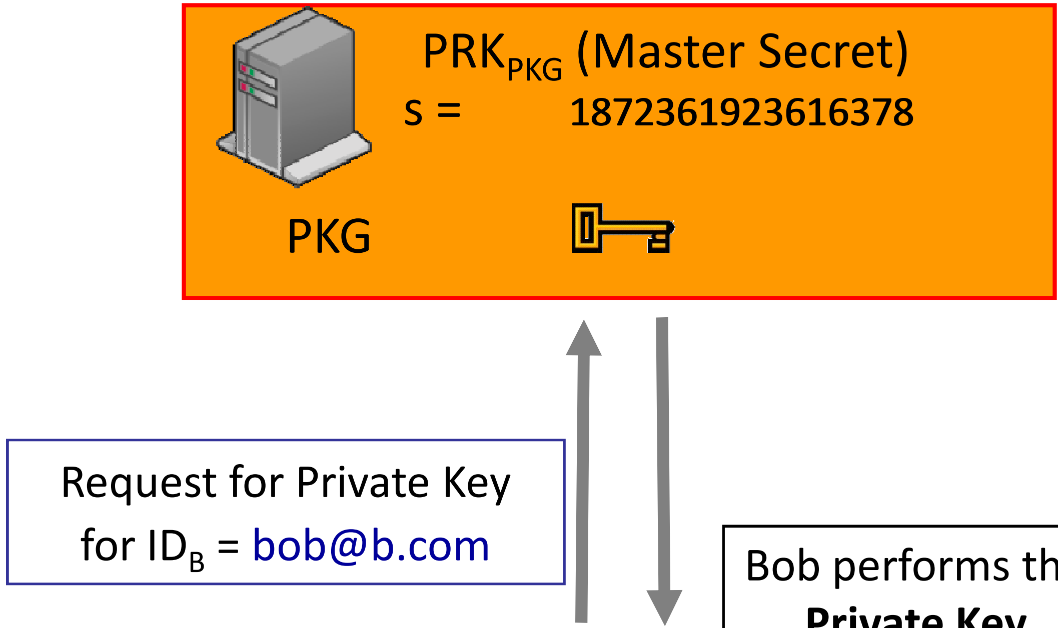
EXCELLENCE EMERGE

Basic concepts of IBS

- The signer Alice first obtains a **signing** $\mathbf{PRK_{Alice}}$ from PKG. She then signs a message using this signing key.
- The verifier Bob uses Alice's ID_A (= PUK_{Alice}) to verify Alice's signature. **No needs for Bob to get Alice's certificate**.
- An IBS scheme can be described using the following steps.

1. **Setup**: The PKG generates its **private key** $\mathbf{PRK_{PKG}}$ (or **Master Secret**) and its **public key** $\mathbf{PUK_{PKG}}$ **(or Master Key)** pair (note that $\mathbf{PUK_{PKG}}$ is publicly known).
2. **Private Key Generation**: The signer Alice authenticates herself to PKG and obtains an authenticated private key $\mathbf{PRK_{Alice}}$ associated with her identity ID_{Alice}. Alice performs a **Private Key Authentication Check** to verify if $\hat{e}(PRK_{Alice}, G) = \hat{e}(ID_{Alice}, \text{Master Key})$ holds.
3. **Signature Generation**: Using her private key PRK_{Alice}, Alice creates a signature σ on her message M.
4. **Signature Verification**: Having obtained the signature σ and the message M from Alice, the verifier Bob checks whether σ is a genuine signature on M using Alice's identity ID_{Alice} and the Master Key PUK_{PKG}. If it is, he returns “Accept”, otherwise, he returns “Reject”.

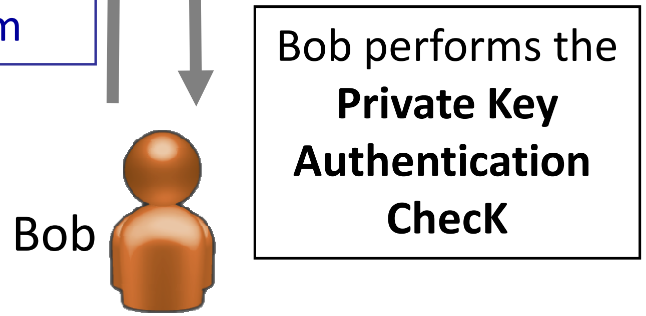
The Private Key Generator



Parameters: **E**, **G**, **ê**

Hash function: $H_1 : \{0,1\}^m \rightarrow E$

r: order of E



PKG uses master secrets to generate keys:

- The **Master Secret** is picked up randomly: $s \in (1, r-1)$
- The **Master Key** is generated from the Master Secret: **sG**
- The **Public Key** of Bob is $\mathbf{ID_B}$ = bob@b.com
- The **Private Key** of Bob is generated by PKG from its Master Secret and Bob Identity: $\mathbf{sH_1(ID_B)}$

IBE Private Keys

Private Key Authentication Check

After receiving the Private Key from the PKG, the party can execute this check:

ê(Private Key, G) = ê(Identity, Master Key)

If successful, then the binding between
user's Private Key issued by PKG and user's identity
is authenticated.

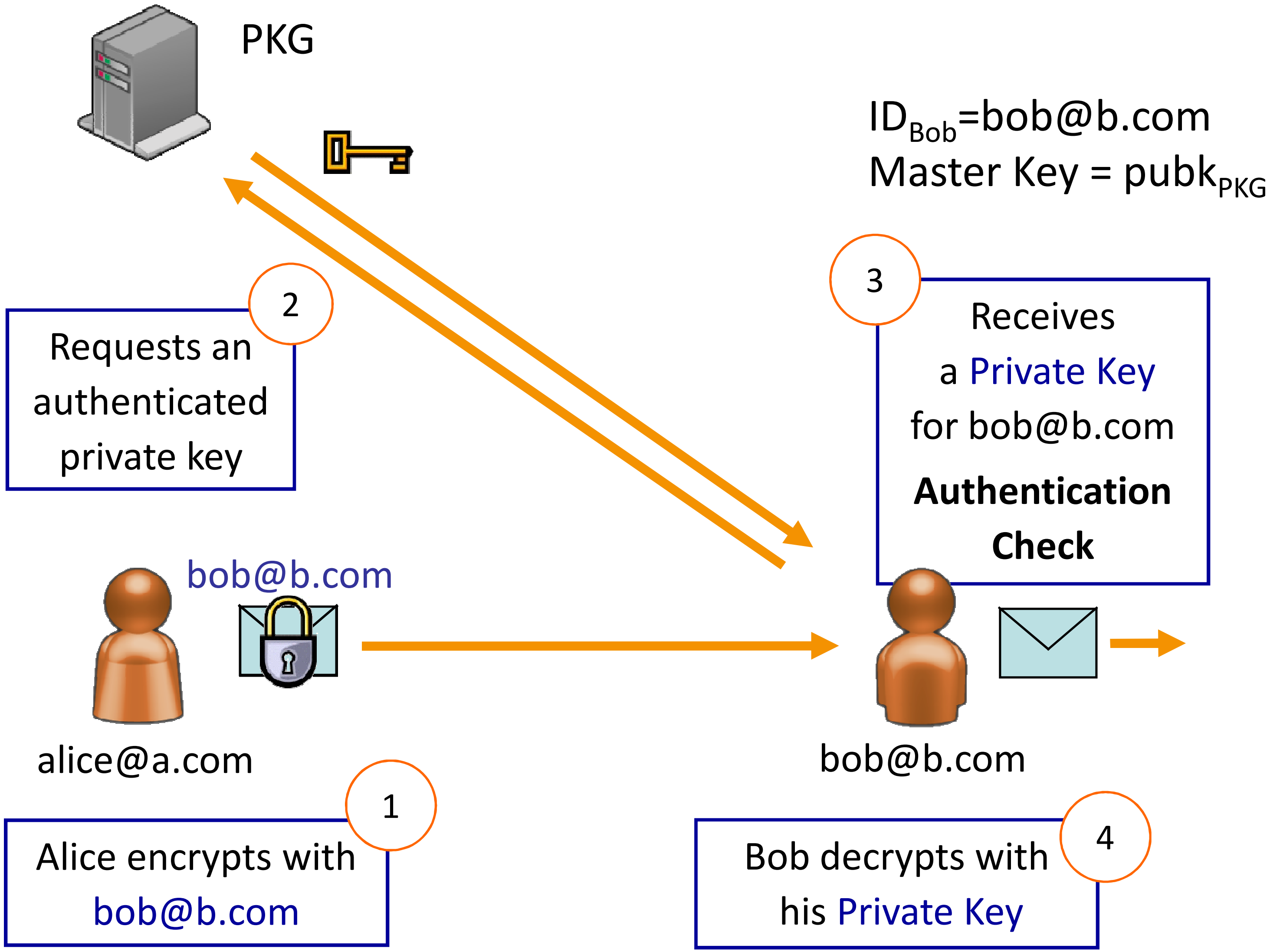
Proof: from the pairing property $\hat{e}(sA,B)=\hat{e}(A,sB)$ is

$$\hat{e}(sH(ID),G)=\hat{e}(H(ID),sG)$$

.

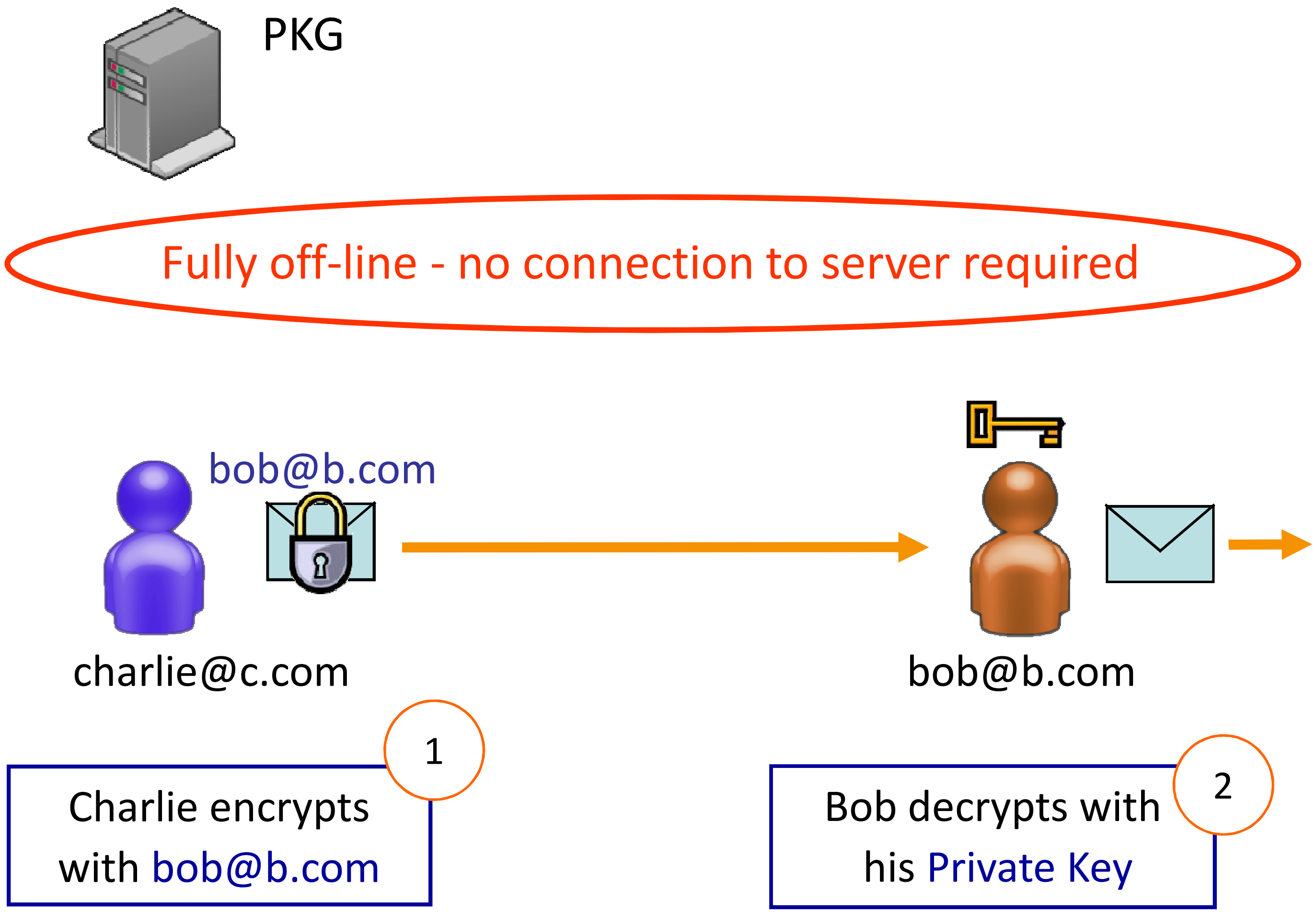
How IBE works in practice

Alice sends a Message to Bob



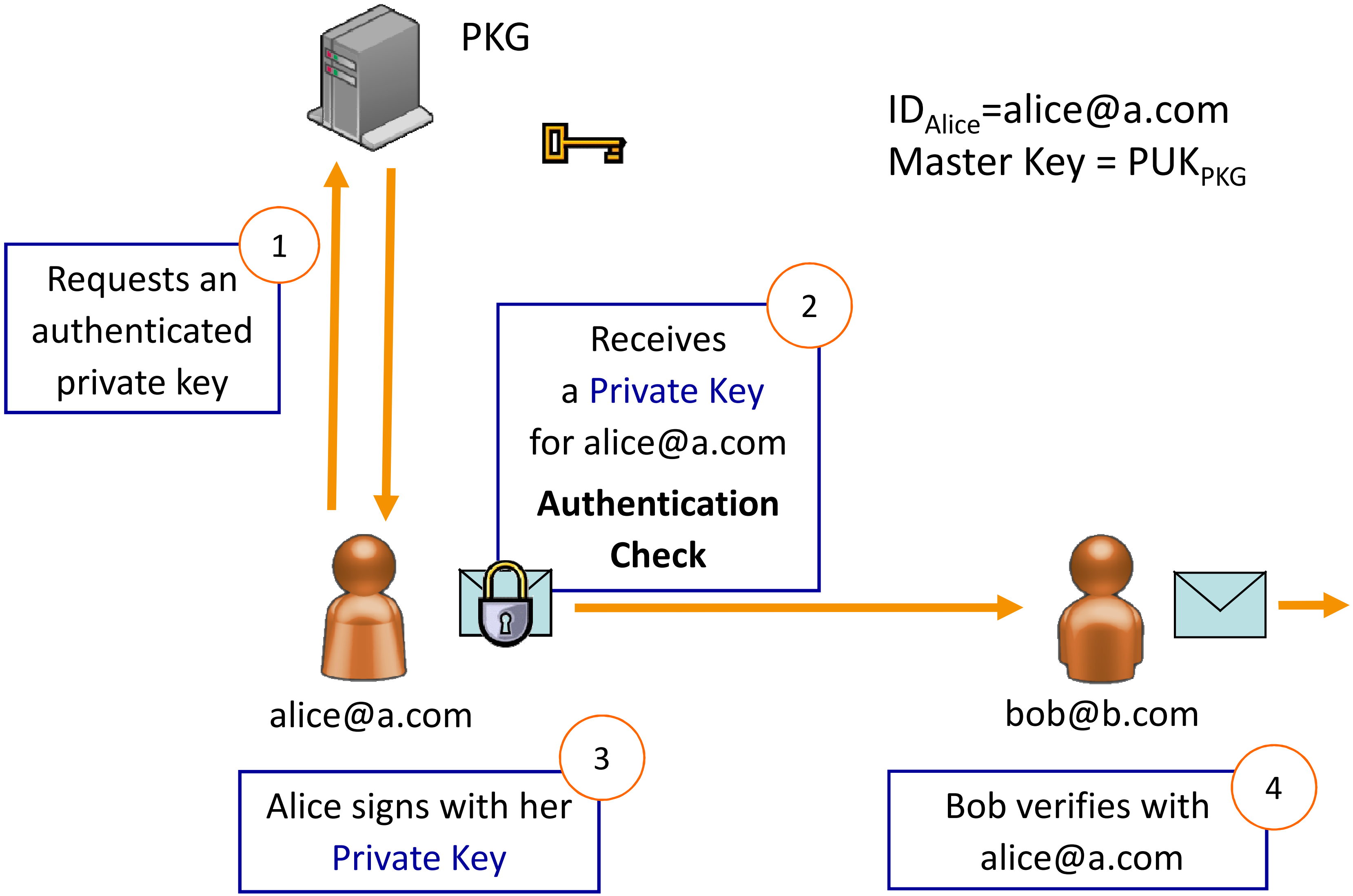
How IBE works in practice

Alice sends a Message to Bob



How IBS works in practice

Alice signs a Message to Bob



IBE using Weil pairing

Parameters: E, G (gen. in E(GF())), ê, hash functions H1 and H2: $H_1 : \{0,1\}^m \rightarrow E$

$H_2 : GF() \rightarrow \{0,1\}^m$

Setup: **Bob's public key is K_B** = $H_1(ID_B)$.
PKG has private key **s**∈ (1,r-1), r order of E, and **Master Key *sG***.
PKG computes **Bob's private key *s*K_B**.

Encryption: To send M, Alice selects a random r∈ (1,q-1) and computes

R = rG and c = M ⊕ $H_2(\hat{e}(K_B, sG)^r)$. She sends Bob (R, c).

Decryption: Bob uses his **private key sK_B** and the **Master Key *sG*** to compute
c ⊕ $H_2(\hat{e}(sK_B,R))$ = c ⊕ $H_2(\hat{e}(sK_B, rG))$ = c ⊕ $H_2(\hat{e}(K_B, sG)^r)$ = M.

Anyone other than Bob wishing to decrypt the message from (R, c) needs to be able to compute $\hat{e}(K_B, sG)^r = \hat{e}(K_B, G)^{rs}$ given G, K_A, S, and R. This requires solving the bilinear Diffie-Hellman problem.

Identity-Based Encryption from the Weil Pairing
D. Boneh, M. Franklin
Proceedings of Crypto 2001, pp. 213-229, Springer-Verlag, 2001
http://courses.cs.vt.edu/~cs6204/Privacy-Security/Papers/Crypto/IBE-Weil-Pairing.pdf

IBS using Weil pairing

(ISO/IEC 14888-3:2018)

Parameters: E, G (gen. in E(GF)), ê, hash functions H1 and H2:

$$H_1 : \{0,1\}^m \to E$$

$$H_2 : GF() \to \{0,1\}^m$$

Setup: **Alice's public key is K_A** $= H_1(ID_A)$.

PKG has private key **s**$\in$ (1,r-1), r order of E, and **Master Key sG**.

PKG generates **Alice's private key sK_A**.

Sign: To send M, Alice selects a random k$\in$ (1,q-1) and computes $T = ê(sK_A,G)^k$, $h = H_2(m,T)$, $S = (k-h)sK_A$. She sends Bob (h,S).

Verification: Bob uses **Alice's public key K_A** and the **Master Key sG** to compute

$T = ê(S,G)ê(K_A,sG)^h$

$= ê(ksK_A,G)ê(-hsK_A,G)ê(K_A,sG)^h$

$= ê(sK_A,G)^k ê(sK_A,G)^{-h} ê(sK_A,G)^h$

$= ê(sK_A,G)^k$

if $H_2(m,T)=h$ then accept, otherwise refuse.

Efficient Identity-Based Signature Schemes based on Pairings
F. Hess
Proceedings of Selected Areas in Cryptography (SAC 2002), pp. 310-324, Springer-Verlag, 2002
https://link.springer.com/content/pdf/10.1007/3-540-36492-7_20.pdf

Open issues with IBE and IBS

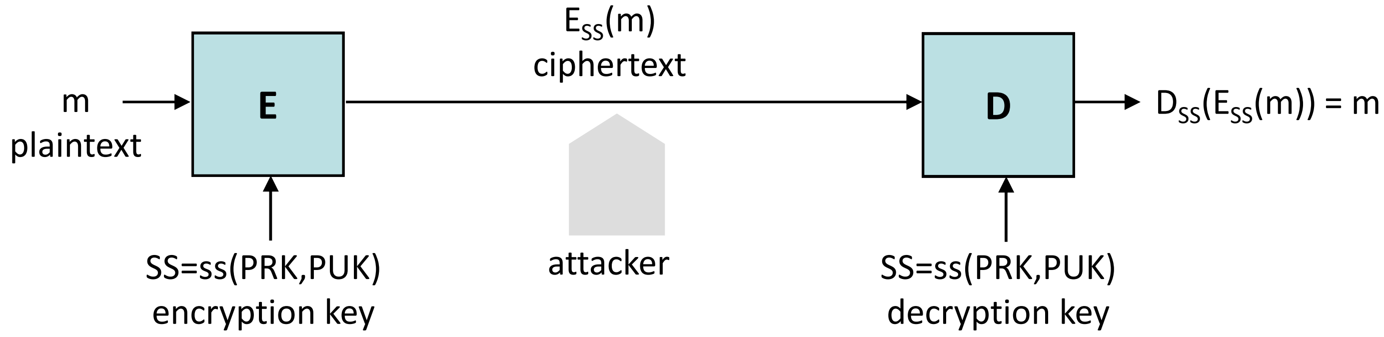
- **Key Escrow Problem (deposito chiavi).** Identity-based cryptographic schemes have inherent weakness, a “key escrow" property: recall that in IBE and IBS schemes, **the PKG issues private keys for user using its (public) Master Key**. **As a result, the PKG is able to decrypt or sign any messages**.

  In terms of encryption, this property *might be useful* in some situations where user's privacy can possibly be limited, for example, due to the involvement in the crime, the user's message should be opened by a court order. However, **in terms of signature**, key escrow property is not desirable at all since “non-repudiation" property is one of the essential requirement of digital signature schemes (non-repudiation means that only an entity which possesses a signing key can create a valid signature).
  - As a possible countermeasure against key escrow problem, given a set of PKGs, each PKG’s Master Key **could be distributed using Shamir's secret sharing technique** into the other PKGs: in this way any single PKG cannot known its own Master Key unless other PKGs agree as well.
- Generated private key are transmitted **over unsecured channels**.
  - As a possible countermeasure, **private keys can be preloaded**. In this case the Private Key Authentication Check can be useful to reveal private key compromissions / alterations

Outline

- Passive Security Functions
  - Ciphering
  - Hash functions
  - Message authentication codes
  - Digital signatures
- Key Establishment Protocols
  - Symmetric KEP
  - Asymmetric KEP
  - ID Based KEP
  - Hybrid KEP
  - Authentication of public key
- Key Management Protocols
  - TinySEC
  - TinyECC
  - TinyIBE
- Passive security techniques for
  - IEEE 802.15.4 MAC
  - Routing
  - ZigBee

Hybrid Encryption

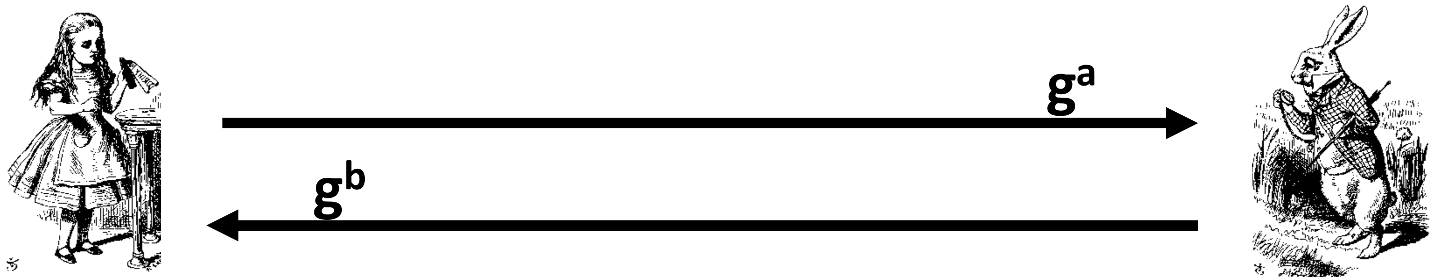


- Hybrid encryption:
  - Asymmetric mechanism for the generation of a shared symmetric key.
  - No asymmetric encryptions or decryptions at all.
  - A shared secret is directly generated by the parties using their private / public keys.

  - **Therefore Shared Secret SS = g(PRK,PUK)** where g() is a one-way function.

Diffie-Hellman Key Exchange

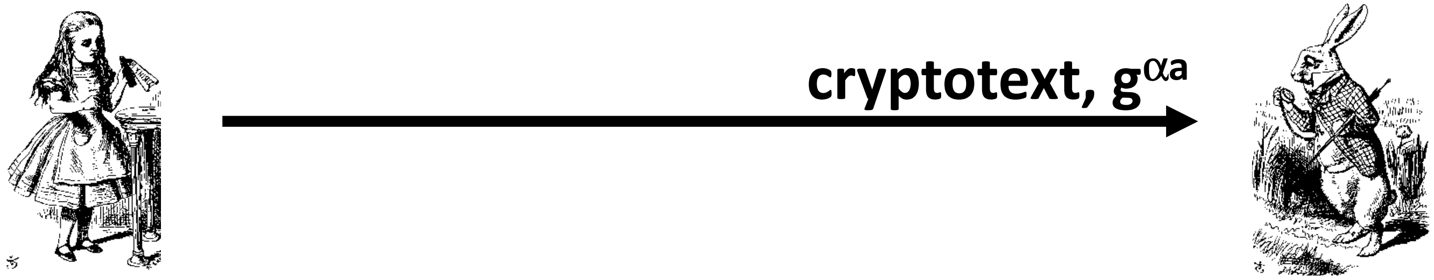
- DHKE is a 2-phase KEP: let g be a generator in GF(p), a,b, K ∈ GF()
  - Alice: Private Key = p_A = a, Public Key = $P_A = g^a$
  - Bob: Private Key = p_B = b, Public Key = $P_B = g^b$
  - Alice computes $K_{AB} = (g^b)^a$ and Bob computes $K_{BA} = (g^a)^b$
  - Shared Secret = K = g^{ab} and then Alice sends the ciphered message c



- Solving g^a **for a** or g^b **for b** is called the **Discrete Logarithm Problem** (**DLP**) or **DH Problem**: infact is $a=\log_g(g^a)$ or $b=\log_g(g^b)$ where a,b∈ GF().
- Main vulnerability of 2-phase schemes is the “Man in Middle” threat: Charlie can be between Alice and Bob and create separate link with them and share Shared Secrets with them and neither Alice or Bob can know this.

Ephemeral Diffie-Hellman Key Exchange

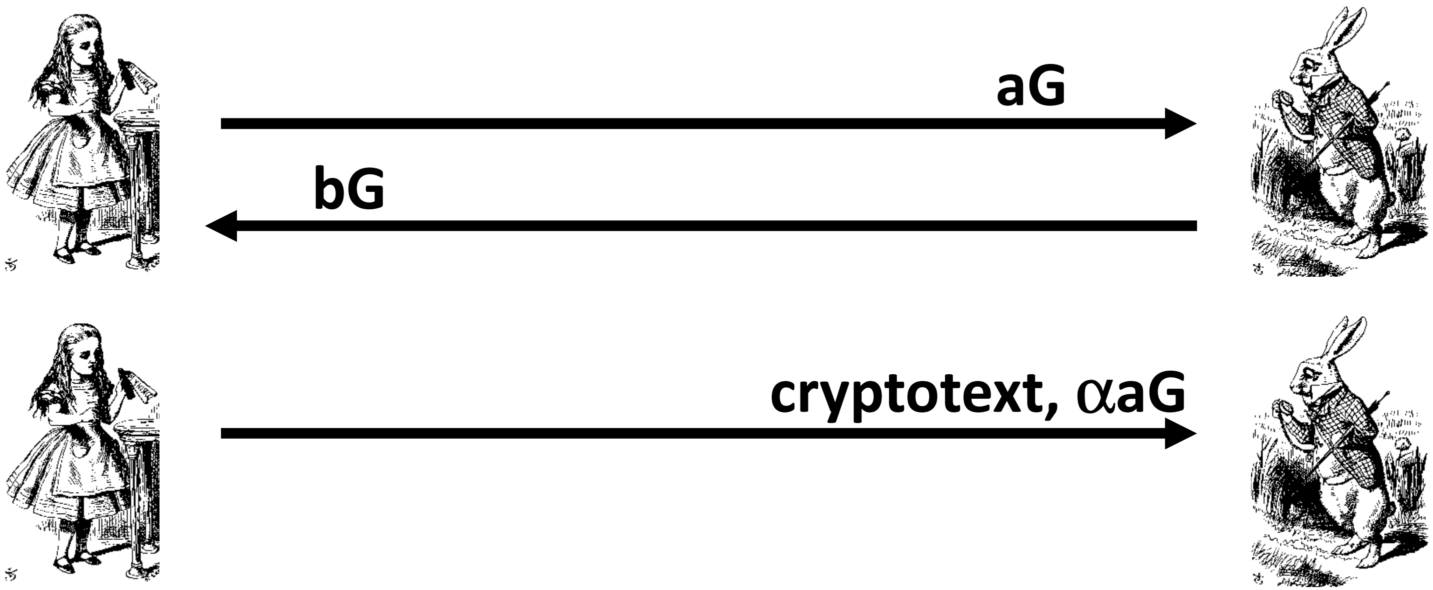
- EDHKE is 1-phase KEP: let g be a generator in GF(q), $a,b,\alpha \in 1, 2, .., q-1$, $K \in GF(q)$
  - α is a pseudorandom generated by Alice
  - Alice: Private Key = $\mathbf{p_A = a}$, Ephemeral Public Key = $\mathbf{P_A = g^{\alpha a}}$
  - Bob: Private Key = $\mathbf{p_B = b}$, Public Key = $\mathbf{P_B = g^b}$
  - Alice computes $K_{AB} = (g^b)^{\alpha a} = g^{\alpha ab}$
  - Shared Secret = $K = g^{\alpha ab}$ and then Alice sends the cryptotext ciphered using K
  - Bob computes $K_{AB} = (g^{\alpha a})^b = g^{\alpha ab}$



- Solving $g^{\alpha a}$ for αa is again a Discrete Logarithm Problem (DLP) or DH Problem, but the pseudorandom factor α hides the solution for a.
- Now the “Man in Middle” threat is not possible anymore: Charlie cannot create separate link with them because he cannot share the Shared Secret.

ECC Diffie-Hellman Key Exchange

- ECDHKE and ECEDHKE extends DHKE and EDHKE to ECC. Let G be a generator in EC(), a,b,$\alpha \in$ 1, 2, .., p-1, K$\in$ GF(p)
  - Alice: Private Key = $\mathbf{p_A = a}$, Public Key = $\mathbf{P_A = aG}$, Ephemeral Public Key = $\mathbf{P_A = \alpha aG}$,
  - Bob: Private Key = $\mathbf{p_B = b}$, Public Key = $\mathbf{P_B = bG}$
- ECDHKE: Shared Secret = K = abG, ECEDHKE: Shared Secret = K = αabG
- The EC Discrete Logarithm Problem (ECDLP) replaces DLP:
  - EC Point **Addition** (P+Q) replaces element **product** (pq) in GF(p)
  - EC Point **Doubling** (2G) replaces element **squaring** (g^2) in GF(p)
  - ECDLP replaces DLP with the same complexity.



IES / DSA vs. ECIES / ECDSA

- ECC allows Diffie-Hellman Key Exchange to be implemented over WSN:
  - ECDLP replaces DLP with the same complexity
  - DH security relies on the Discrete Logarithm Problem
  - ECDH security relies on Elliptic Curve Logarithm Problem

- IES (Integrated Encryption Scheme) and DSA (Digital Signature Algorithm) due to V. Shoup
- ECIES and ECDSA are the ECC extension to IES and DSA.

A Proposal for an ISO Standard for Public Key Encryption (v. 2.1), 2001
V. Shoup
http://www.shoup.net/papers/iso-2_1.pdf

ECIES pseudocode

Alice (the Encypher): given EC Domain Parameters D=(p,a,b,G,n,h) or D=(m,a,b,G,n,h), **Bob's public key $\mathbf{P_B}$**, the message m, return (c,R,t) where c is the enciphered message (the cipher-text):

1. Select an integer k such that $1 \le k \le n-1$
2. Compute R = kG e Z = hk$\mathbf{P_B}$. If Z = 0 go back to 1
3. Compute KDF(z_x, r_x) = (k_1,k_2) where z_x and r_x are x-coordinate of Z and R
4. Compute c = ENC_{k1}(m) and t = MAC_{k2}(c)
5. Return (c,R,t)

Bob (the Decipher): given EC Domain Parameters D=(p,a,b,G,n,h) or D=(m,a,b,G,n,h), **Bob's private key $\mathbf{p_B}$**, the 3-pla (c,R,t), determine the message m or refuse:

1. Compute Z = h$\mathbf{p_B}$R = (z_x,z_y). If Z = 0 then "refuse"
2. Compute KDF(z_x, r_x) = (k_1,k_2) where z_x and r_x are x-coordinate of Z and R
3. Compute t' = MAC_{k2}(c). If t ≠ t' then "refuse"
4. Compute m = DEC_{k1}(c)

ECDSA pseudocode

Alice (the Prover): given EC Domain Parameters D=(q,a,b,G,n,h) or D=(m,a,b,G,n,h), **Alice's private key p_A**, the message m, compute signature (r,s).

1. Select an integer k such that $1 \leq k \leq n-1$
2. Compute $kG = (x_1,y_1)$ and consider $r = x_1 \bmod n$. If r = 0 go back to 1
3. Compute e = H(m)
4. Compute $s = k^{-1}(e+p_A r) \bmod n$. If s = 0 go back to 1
5. Return (r,s)

Bob: (the Verifier): given EC Domain Parameters D=(q,a,b,G,n,h) or D=(m,a,b,G,n,h), **Alice's public key P_A**, the message m and the signature (r,s), accept or refuse the sign.

1. Verify that $1 \leq r,s \leq n-1$. If not then "refuse"
2. Compute e = H(m)
3. Compute $w = s^{-1} \bmod n$. Compute $u_1 = ew \bmod n$. Compute $u_2 = rw \bmod n$
4. Compute $X = \mathbf{u_1 G + u_2 P_A}$. If X = 0 then "refuse"
5. Compute $v = x_1 \bmod n$ where x_1 is the x-coordinate of X
6. If v = r then "accept" otherwise "refuse"

Cryptography is still secure ?

- **Classical Computing:** 1 binit $\rightarrow$ 2 classical states: 0 **or** 1: voltage low / high
- **Quantum Computing: 1 qubit** $\rightarrow$ superposition of 2 quantum states 0 **and** 1: spin up / down in electrons, H/V polarization in photons.
- Therefore n qubits in input for an operation in a quantum computer, the result of 2^n operations for 2^n inputs **with just one computational step** is returned as compared to 2^n computation steps needed by a classical computer, **where n is the number of electrons / photons !!**
- **Computation Capacity is improved by a 2^n factor** (truly parallel computing).
- **Post-quantum cryptography** refers to public key algorithms that are thought to be secure against an attack by a quantum computer.
- Up to now, problems such as integer factorization (RSA), DLP (DH) and ECDLP (ECDH) could be **efficiently broken** by a sufficiently large quantum computer running Shor's algorithm.
- NIST includes post-quantum algorithms in Commercial National Security Algorithm Suite 2.0 and recommends timing for the complete transition from CNSA v. 1.0 to 2.0 **by 2030**. The suite includes: AES 256 bit, ECDH and ECDSA with curve P-384, SHA-2 with 384 bits, Diffie–Hellman key exchange and RSA with a minimum 3072-bit modulus.

Outline

- Passive Security Functions
  - Ciphering
  - Hash functions
  - Message authentication codes
  - Digital signatures
- Key Establishment Protocols
  - Symmetric KEP
  - Asymmetric KEP
  - ID Based KEP
  - Hybrid KEP
  - Authentication of public key
- Key Management Protocols
  - TinySEC
  - TinyECC
  - TinyIBE
- Passive security techniques for
  - IEEE 802.15.4 MAC
  - Routing
  - ZigBee

Authentication of Public Keys

- A **Certification Authority (CA)** is a trusted party that authenticates public keys (or private keys), i.e. **certifies** the binding <identity, key> .
- The CA public key is assumed known by any party.
- Certificates can be explicit, implicit or certificate-less.

- An **explicit** certificate (X. 509-based) of a public key is the 3-pla:

  <$\mathbf{pubk_{user}}$, $\mathbf{ID_{user}}$, $\mathbf{sigk_{CA}(pubk_{user}, ID_{user})}$> signed by CA using its $prik_{CA}$.

  Suppose Bob requests Alice’s public key

  1. **Cert(Alice) = (**$\mathbf{pubk_{Alice}}$**,** $\mathbf{ID_{Alice}}$**,** $\mathbf{sigk_{CA}(pubk_{Alice},ID_{Alice})}$**)**

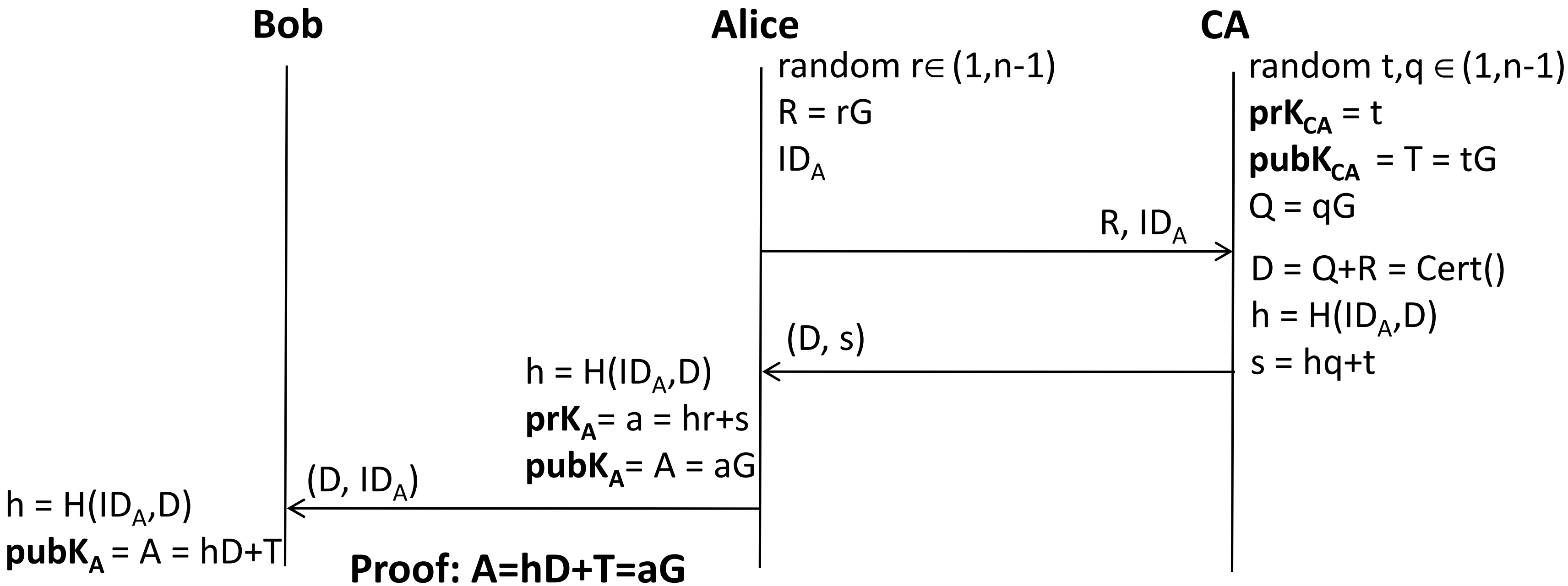
  2. Bob verifies $\mathbf{pubk_{Alice}}$ using $\mathbf{pubk_{CA}}$

  A drawback of explicit certification is the size: standard X.509 certificate is about 1KB

- Similarly for the certification of a private key.
- In an Identity-based scheme, the subject's identity itself is used to derive their public key; there is no certificate (certificate-less scheme). The corresponding private key is calculated and issued to the subject by a TTP.

Authentication of Public Keys

- An **implicit** certificate (e.g. ECQV, EC Qu Vanstone certificate) consists of identification data (ID) and a single EC point. **Any party (say Alice) can compute its own pair (prK_A,$pubK_A$) and any other party (say Bob) can computes $pubK_A$** without any transmission of private keys and CA signatures. ECQV certificates can be considerably smaller than explicit certificates (useful for resource constrained nets).
  Given an EC domain, be n the order of the generator G:



Proof: A=hD+T=aG

$A=aG=(hr+s)G=hrG+(hq+t)G=hR+hQ+T=h(R+Q)+T=hD+T$ q.e.d.

SEC 4: Elliptic Curve Qu-Vanstone Implicit Certificate Scheme (ECQV).
Standard for Efficient Cryptography. Jan. 2013. Available online: http://www.secg.org/sec4-1.0.pdf

Authentication of Public Keys

Different approaches can be followed:

- **Web of Trust**: **no CA are assumed**; it relies entirely on trust a-priori relationships between parties. If Alice trusts Bob, it is assumed that she also wants to trust all other users whom Bob trusts. Every party in this WoT implicitly trusts parties whom it does not know. **Sign-encryption schemes should be used**.
- **Chain of Trust**: **multiple interoperating CAs are assumed**; users communicate with other whose certificates are issued by different CAs: this requires cross-certification of CAs, e.g. CA_1 certifies the public key of CA_2. If Alice trusts her CA_1, cross certification ensures that she also trusts CA_2 ("trust is delegated"). **This approach results suited for ad-hoc networks that can be parceled into subnetworks , each subnet referring to a specific CA**, e.g. in VANETs that refer on multiple RAs (Regional trusted Authorities). Chain of trust is also applied for blockchains.
- **Key Material Preloading**: **an external CA is assumed**: CA computes private keys for each node and the needed public keys and this key material is off-line preloaded into nodes avoiding requests for public keys towards CA. A dynamic refresh (to achieve Forward Secrecy) of the keys through nonces should be performed for any communication session (ephemeral private / public keys). **This approach results suited for large ad-hoc networks**.

Outline

- Passive Security Functions
  - Ciphering
  - Hash functions
  - Message authentication codes
  - Digital signatures
- Key Establishment Protocols
  - Symmetric KEP
  - Asymmetric KEP
  - ID Based KEP
  - Hybrid KEP
  - Authentication of public key
- Key Management Protocols
  - TinySEC
  - TinyECC
  - TinyIBE
- Passive security techniques for
  - IEEE 802.15.4 MAC
  - Routing
  - ZigBee

Key Management Protocol

- Once a key has been established, the Key Management Protocol (KMP) manages the key during system life-cycle (e.g. node additions, node deletions, key refresh, …).
- KMP should work in undefined deployment environment
- KMP should be independent on topology.
- KMP returns useful information to Intrusion Detection System.
- It provides the basic management operations on key material (key components, configuration parameters, link keys, network keys, …)
  - Assignment
  - Revocation
  - Updates

- **Session keys** are shared secrets **updated each session**
  - To limit available ciphertext for cryptanalysis.
  - To limit exposure caused by the compromise of a session key.
  - To avoid long-term storage of secret keys.
  - Need of CSPRNG.

Outline

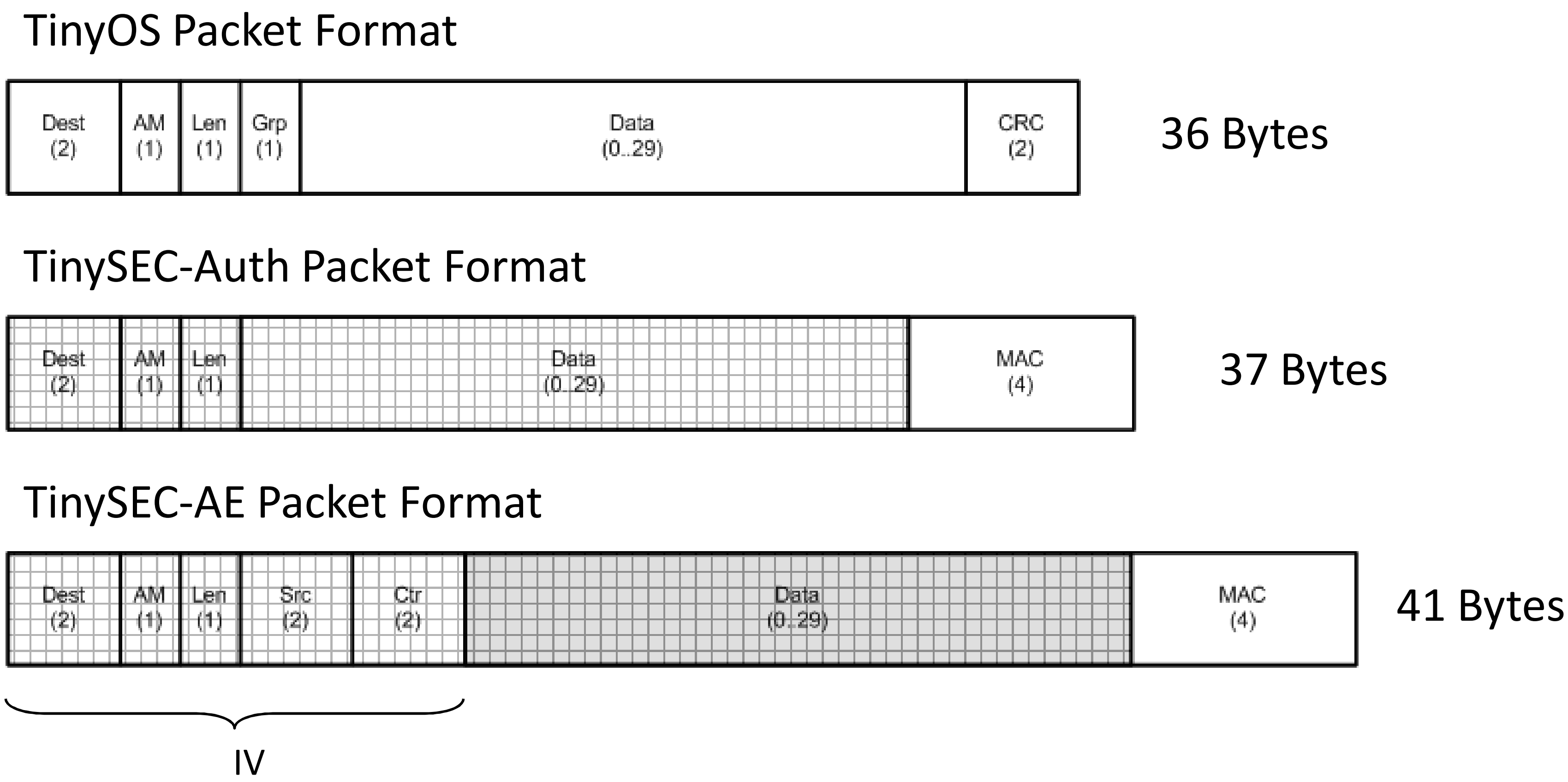
- Passive Security Functions
  - Ciphering
  - Hash functions
  - Message authentication codes
  - Digital signatures
- Key Establishment Protocols
  - Symmetric KEP
  - Asymmetric KEP
  - ID Based KEP
  - Hybrid KEP
  - Authentication of public key
- Key Management Protocols
  - TinySEC
  - TinyECC
  - TinyIBE
- Passive security techniques for
  - IEEE 802.15.4 MAC
  - Routing
  - ZigBee

TinySEC

- **It is the basic KMP over TinyOS.**
- It manages link-layer security for WSN.
- It is independent on ciphering algorithms and key establishment protocols.
- It guarantees authenticity, integrity and confidentiality.

- Each service a type of **packet format**
  - TinyOS (No TinySEC)
    - No Authentication & No Encryption
  - TinySEC-AE
    - Authentication & Encryption (CBC)
    - MAC computed over encrypted data and the packet header
  - TinySEC-Auth
    - Authentication Only

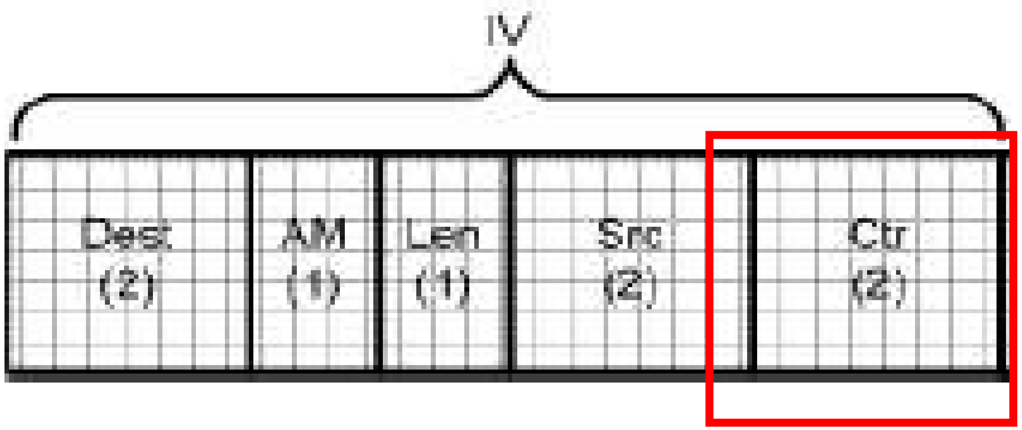
TinySEC Packet format



TinySEC IV format

- In TinySEC the Initialitation Vector is a counter (not a pseudo-random) → IV could be reused
  - IV too long - add unnecessary bits to the packet
  - IV too short – frequent repetitions
- A counter IV repeat after $2^n + 1$, n the bit length of the counter.
- **CBC works better with reused IV**

8 byte but the Counter in only 2 bytes (16 bits)



Security Analysis

Combination of carefully formatted IVs, low data rates and CBC mode for encryption achieves high confidentiality in TinySEC.

- **Message Integrity and Authenticity**
  - Based on MAC length (4 bytes for TinySEC)
  - 1 out 2^{32} chance to guess it
  - Adversary must send 2^{32} packets to correctly fake a message
  - This is not OK for regular networks but given the **low rate** data in WSN, this is ok for WSN.
    - Even if the adversary flood the channel (suppose 25 kbps), he can send only 80 forgery attempts/sec (each packet is 40 byte long), s.t. sending 2^{32} would take about 20 months.
    - Battery operated nodes do not have that much energy to collect all those packets.

Security Analysis

- **Message Confidentiality**
  - Security based on IV length, assuming no reuse is 8 byte counter or 16 byte random
  - However, we have an 8 byte **total** IV
    - 2 Destination, 1 AM, 1 Length, 2 Source and **2 Counter**
  - Therefore IV repeats after 2^{16} packets
    - Considering a monitoring application with sample period 1 observation per minute, IV reuse will not occur before about 45 days ($2^{16}/(60*24) \approx 45$).

Outline

- Passive Security Functions
  - Ciphering
  - Hash functions
  - Message authentication codes
  - Digital signatures
- Key Establishment Protocols
  - Symmetric KEP
  - Asymmetric KEP
  - ID Based KEP
  - Hybrid KEP
  - Authentication of public key
- Key Management Protocols
  - TinySEC
  - TinyECC
  - TinyIBE
- Passive security techniques for
  - IEEE 802.15.4 MAC
  - Routing
  - ZigBee

TinyECC

TinyECC: A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks (v.2.0)

A. Liu, P. Ning

Proceedings of the 7th International Conference on Information Processing in Sensor Networks (IPSN 2008), SPOTS Track, pp. 245-256, April 2008.

http://discovery.csc.ncsu.edu/software/TinyECC/TR-2007-36.pdf

TinyECCK: Efficient Elliptic Curve Cryptography Implementation over GF(2^m) on 8-bit MICAz Mote

S. C. Seo, D. Han, H.C. Kim, S. Hong

IEICE Transactions on Info and Systems E91-D(5), pp. 1338-1347, May 2008.

https://www.researchgate.net/publication/31467778_TinyECCK_Efficient_Elliptic_Curve_Cryptography_Implementation_over_GF2m_on_8-Bit_Micaz_Mote

Efficient Implementation of NIST-Compliant Elliptic Curve Cryptography for 8-bit AVR-Based Sensor Nodes

Z. Liu, H.Seo, J.Großschädl, H. Kim

IEEE Transactions on Information Forensics and Security, vol. 11, n.7, pp. 1385-1397, July 2016.

https://orbilu.uni.lu/bitstream/10993/12934/1/ICICS2013.pdf

TinyECC Design Principles

- TinyECC is a sw package over TinyOS for ECC-based KEPs .
  - TinyECC is based on GF(p), p prime.
  - TinyECCK is based on GF(2^n).

- TinyECC includes ECC schemes such as
  - Elliptic CurveDigital Signature Algorithm (ECDSA).
  - Elliptic Curve Diffie-Hellman (ECDH).
  - Elliptic Curve Integrated Encryption Scheme (ECIES).

- TinyECC is independent on sensor platform through TinyOS.

Optimization tricks in TinyECC

- **Barrett Reduction:** integer modular reductions without divisions.
- **Hybrid Multiplication:** windowed double and add algorithm.
- Use of **Projective Coordinates.**
- **Shamir's Trick:** This optimization is only used for the verification of ECDSA signatures: the verification of ECDSA signature requires the computation of the form aP+bQ, where a,b are integers and P,Q are two points on an elliptic curve.
- **Curve Specific Optimization:** use pseudo-Mersenne primes as specified by NIST and SECG.

Outline

- Passive Security Functions
  - Ciphering
  - Hash functions
  - Message authentication codes
  - Digital signatures
- Key Establishment Protocols
  - Symmetric KEP
  - Asymmetric KEP
  - ID Based KEP
  - Hybrid KEP
  - Authentication of public key
- Key Management Protocols
  - TinySEC
  - TinyECC
  - TinyIBE
- Passive security techniques for
  - IEEE 802.15.4 MAC
  - Routing
  - ZigBee

TinyIBE Design Principles

- TinyIBE is a sw package over TinyOS for ID-Based Encryption.
  - **suited for hierarchical (clustered) WSN**
  - simple variant of the Sakai-Kasahara IBE scheme
  - **only convergecast (from leaves to CH)**: no need of PRK/PUK for leaf nodes.
  - SS is generated by a leaf node and then shared with its own CH
- off-line (2 phases)
  - **Setup**: $P \in E(GF(q))$ order r, $s \in (1, r-1)$, Q=sP, g=ê(P,P)
  - **Extract**: assign $PUK_{CH} = H_1(ID_{CH})$, $PRK_{CH} = (1/(s+PUK_{CH}))P$.

    Pre-load PRK_{CH} into CH nodes; pre-load PUK_{CH}, P,Q, g into leaf nodes.

$$H_1 : \{0,1\}^* \rightarrow Z_q^*$$

$$H_2 : GF(q) \rightarrow \{0,1\}^n$$

- run-time (2 phases)
  - **Encrypt at leaf node**: random w, $SS = t \in Z^*_q$

    compute and send $C_1 = w(Q + PUK_{CH}P)$, $C_2 = t \oplus H_2(g^w)$ to CH node
  - **Decrypt at CH node**: $SS = t = H_2(\hat{e}(PRK_{CH}, C_1)) \oplus C_2$

TinyIBE: Identity-Based Encryption for Heterogeneous Sensor Networks
P. Szczechowiak, M. Collier, 5th International Conference on Intelligent Sensors, Sensor Networks and Information Processing , 2009
ID-based Cryptosystems with Pairing on Elliptic Curve
R. Sakai, M. Kasahara, Cryptology ePrint Archive, Report 2003/054, 2003.

Outline

- Passive Security Functions
  - Ciphering
  - Hash functions
  - Message authentication codes
  - Digital signatures
- Key Establishment Protocols
  - Symmetric KEP
  - Asymmetric KEP
  - ID Based KEP
  - Hybrid KEP
  - Authentication of public key
- Key Management Protocols
  - TinySEC
  - TinyECC
  - TinyIBE
- Passive security techniques for
  - IEEE 802.15.4 MAC
  - Routing
  - ZigBee

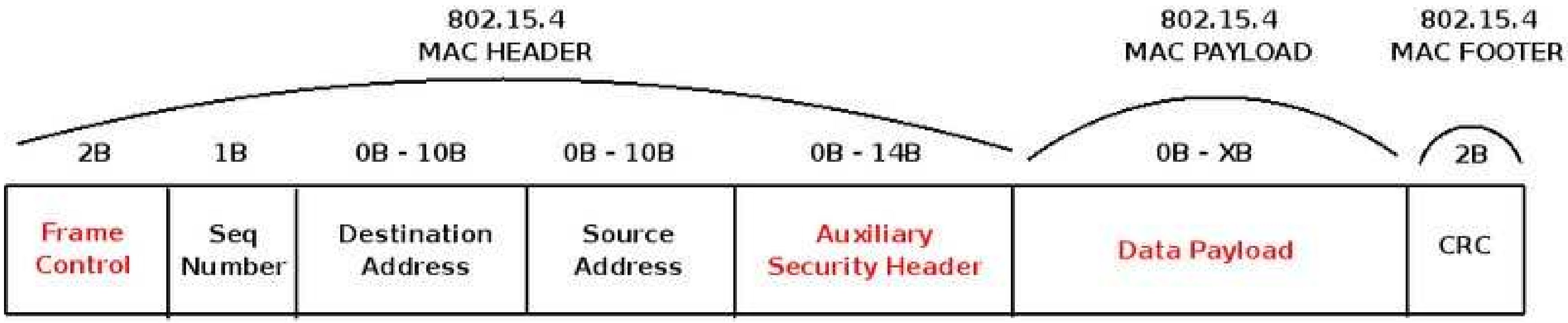
802.15.4 Security

- IEEE 802.15.4 security deals with the security functions provided by the standard to protect the link layer functions.
- Security functions are:
  - Message Confidentiality through AES ciphering.
  - Message Integrity through Message Authentication Code (MAC).

- AES algorithm is not only used to encrypt the information but to validate the data which is sent (**Data Integrity)** and it is achieved using a MAC (**Message Authentication Code**) which is appended to the message.
- This code ensures integrity of the MAC header and MAC payload attached (**here MAC is Media Access Control**).
- To avoid confusion, the term **Message Authentication Code** will be replaced by the term **Message Integrity Code (MIC).**
- MIC can have different sizes: 32, 64, 128 bits, however it is always created using the **AES-128** algorithm.

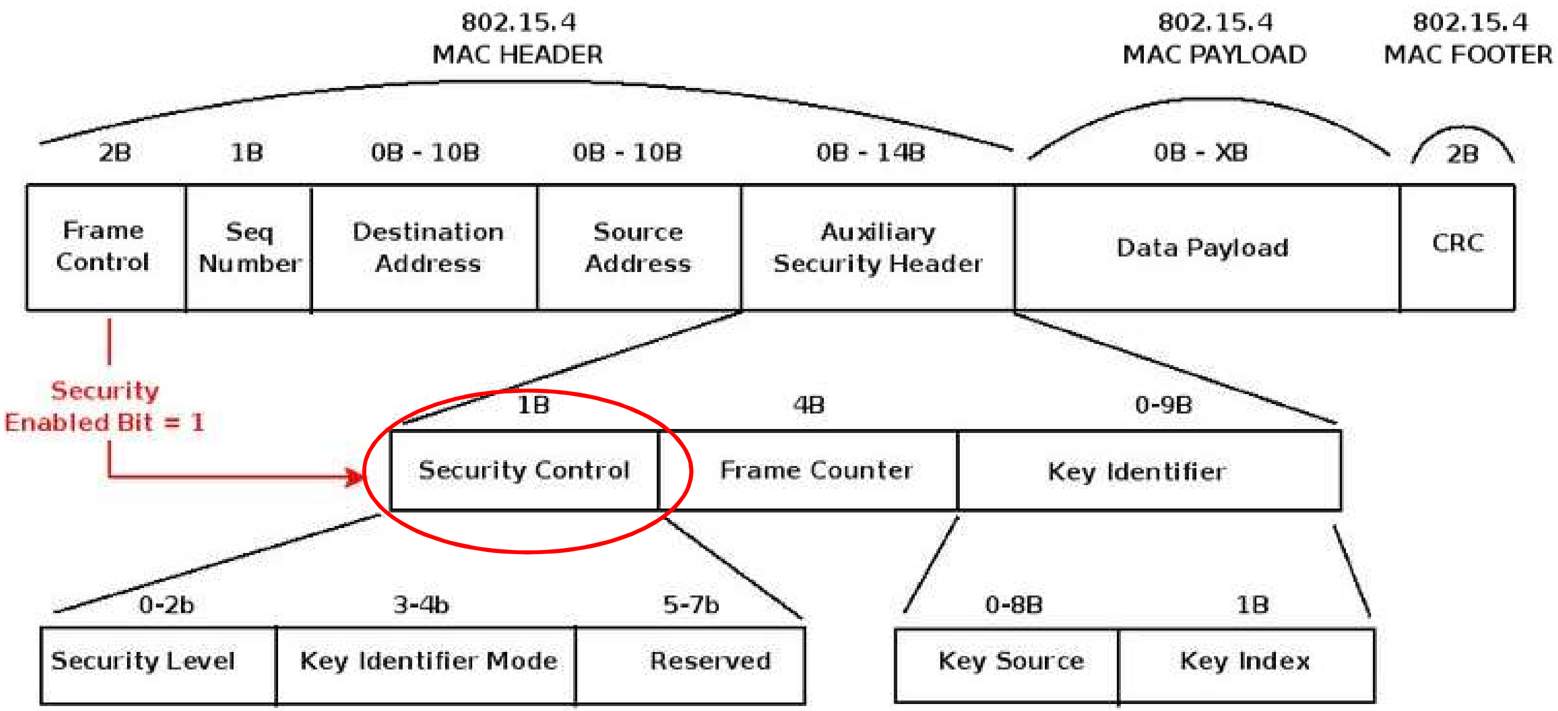
802.15.4 Security

- Recall from Lesson 1 that a coordinator on a PAN can optionally bound its channel time using a "**superframe structure**".
- A superframe is bounded by the transmission of a beacon frame and can have an active portion and an inactive portion. The coordinator may enter a low-power (sleep) mode during the inactive portion.
- During the active portion (=CAP+CFP), frames can access the medium.
- 4 frame types: BEACON, DATA, ACK and MAC command frame.
- 3 fields in the IEEE 802.15.4 MAC DATA frame related to security:
  - **Frame Control** (located in the MAC Header)
  - **Auxiliary Security Control** (in the MAC Header)
  - **Data Payload** (in the MAC Payload field)



802.15.4 Security

- The **Auxiliary Security Frame** is enabled if the **Security Enabled Bit** subfield of the **Frame Control Frame** is set to 1. This special header has 3 fields:
  - **Security Control** (1B) specifies which kind of protection is used.
  - **Frame Counter** (4B) is a counter given by the source of the current frame in order to protect the message from replaying. For this reason each message has a unique sequence ID represented by this field.
  - **Key Identifier** (0-9B) specifies the information about the key we are using with the node we are communicating with.

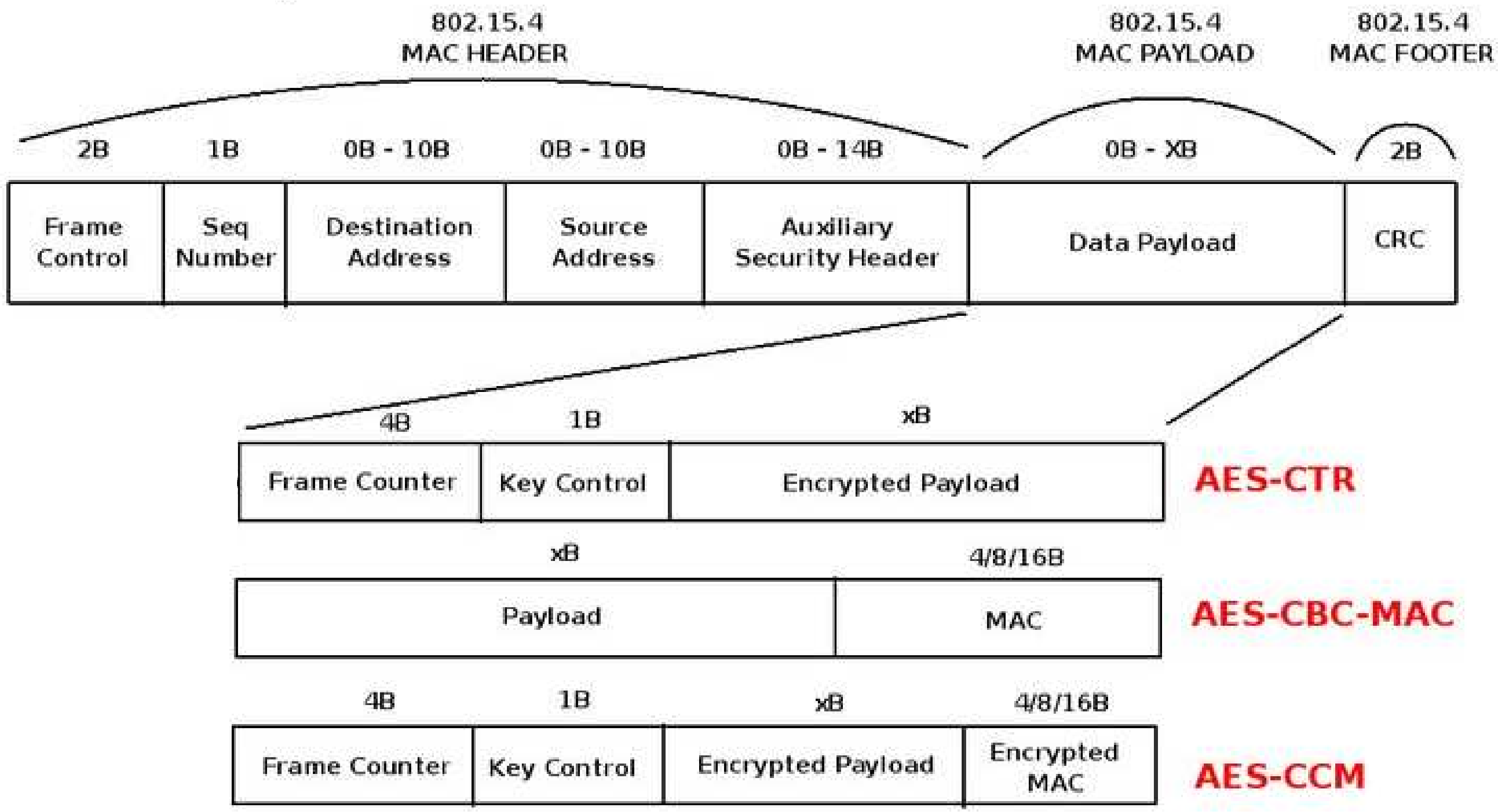


802.15.4 Security

- The **Security Control** is the field where global Security Policy is set: the **0x00** value sets no encryption so nor the data is encrypted (no data confidentiality) or the data authenticity is validated. From the **0x01 to 0x03** the data is authenticated using the encrypted Message Authentication Code (**MAC**). The value **0x04** encrypts the payload ensuring **Data Confidentiality**. The **0x05 to 0x07** range ensures both data confidentiality and authenticity.

0x00			No security. Data is not encrypted. Data authenticity is not validated.
0x01	AES-CBC-MAC-32	MIC-32	Data is not encrypted. Data authenticity is validated.
0x02	AES-CBC-MAC-64	MIC-64	Data is not encrypted. Data authenticity is validated.
0x03	AES-CBC-MAC-128	MIC-128	Data is not encrypted. Data authenticity is validated.
0x04	AES-CTR	ENC	Data is encrypted. Data authenticity is not validated.
0x05	AES-CCM-32	AES-CCM-32	Data is encrypted. Data authenticity is validated.
0x06	AES-CCM-64	AES-CCM-64	Data is encrypted. Data authenticity is validated.
0x07	AES-CCM-128	AES-CCM-128	Data is encrypted. Data authenticity is validated.

802.15.4 Security



802.15.4 Security

- **Data Payload** field can have three different configurations depending on the previously defined security fields:
  - **AES-CTR**: all the data is encrypted using the defined 128 bit key and the AES algorithm. The Frame Counter sets the unique **message ID**, and the **Key Counter** (Key Control subfield) is used by the application layer if the **Frame Counter** max value is reached.
  - **AES-CBC-MAC**: MIC is attached to the end of the data payload. Its length depends on the level of security specified in the Security Policy field. The MIC is created encrypting information from the 802.15.4 MAC header and the data payload.
  - **AES-CCM** (**CCM** = **C**TR with **C**BC-**M**AC): CCM mode combines CBC-MAC with CTR mode of encryption. These two primitives are applied in an "authenticate-then-encrypt" manner:
    - CBC-MAC is first computed on the message to obtain a tag t
    - the message and the tag are then encrypted using counter mode.
  - One key insight is that the same encryption key can be used for both

The Access Control List

- Each **802.15.4 transceiver** has to manage a **list** to control its ***"trusted brothers"***. For this reason each node has to control its own **Access Control List (ACL)** which stores at least the following fields:
  - **Address**: the destination node
  - **Security Suite**: the security police which is being used (e.g. AES-CCM-128)
  - **Key**: the 128 bit key length used in AES
- When a node wants to send a message to a specific node or receives a packet, it looks at the **ACL** to see if it is a **trusted brother** or not.
  - In the case it is, the node uses the data inside the specific row apply the security measures.
  - In the case the node is not in the list or its message is rejected, an authentication process starts.
- Same key in multiple ACL entries
  - If used, very likely that nonce will be reused (loss of confidentiality)
- Loss of ACL from power failure: recommended re-keying.
- ACL is stored in MAC PAN Information Base (PIB) and is accessed and modified similar to other MAC attributes.

Outline

- Passive Security Functions
  - Ciphering
  - Hash functions
  - Message authentication codes
  - Digital signatures
- Key Establishment Protocols
  - Symmetric KEP
  - Asymmetric KEP
  - ID Based KEP
  - Hybrid KEP
  - Authentication of public key
- Key Management Protocols
  - TinySEC
  - TinyECC
  - TinyIBE
- Passive security techniques for
  - IEEE 802.15.4 MAC
  - Routing
  - ZigBee

EXEMERGE CELLENCE

Secure Routing

UNIVERSITÀ DEGLI STUDI DELL'AQUILA

- First: messages must be signed.
- Second: an IDS must be running.

- Sybil attack
  - Limit the number of neighbors for a node.
  - IDS: search for ID conflicts
- HELLO flood attack
  - Verify if links with these new nodes are bidirectional
  - IDS: search for HELLO generators with conflicting IDs.
- Wormhole, sinkhole attack
  - Robust routing protocol design (e.g. spanning tree).
  - IDS: measure packet delivery latencies
- Selective forwarding
  - Multi-path routing: message routed over n paths whose nodes are completely disjoint.
  - Dynamically pick next hop from a set of candidates.
  - IDS: search for nodes with frequent requests to re-transmit.

Outline

- Passive Security Functions
  - Ciphering
  - Hash functions
  - Message authentication codes
  - Digital signatures
- Key Establishment Protocols
  - Symmetric KEP
  - Asymmetric KEP
  - ID Based KEP
  - Hybrid KEP
  - Authentication of public key
- Key Management Protocols
  - TinySEC
  - TinyECC
  - TinyIBE
- Passive security techniques for
  - IEEE 802.15.4 MAC
  - Routing
  - ZigBee

ZigBee Security

- **The Trust Center is usually the network coordinator**. It is responsible for the following security roles:
  - Trust Manager: to authenticate devices that request to join the network;
  - Network Manager: to maintain and distribute network keys;
  - Configuration Manager: to enable end-to-end security between devices
- **Master Key**: this optional key is not used to encrypt frames but is used as an initial shared secret between two devices when they perform the Key Establishment Procedure (SKKE) to generate Link Keys. Keys that originates from the Trust Center are called Trust Center Master Keys, while all other keys are called Application Layer Master Keys.
- **Network Key**: this key performs security Network Layer on a ZigBee network. All devices on a ZigBee network share the same key.
  - High Security Network Keys must always be sent encrypted over the air
  - Standard Security Network Keys can be sent either encrypted or unencrypted. Note that High Security is supported only for ZigBee PRO.
- **Link Key**: this optional key secure unicasts messages between two devices at the Application Layer. Keys that originate from the Trust Center are called Trust Center Link Keys, while all other keys are called Application Layer Link Keys.

ZigBee Security

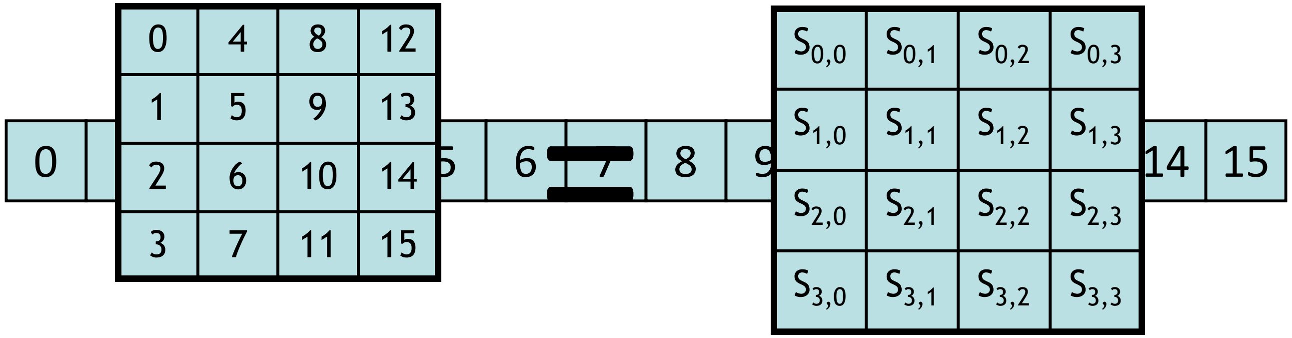
2. **Authentication and Data Encryption***:* Data is encrypted using 128-bit AES with CCM mode (remind **CCM** = **C**TR with **C**BC-**M**AC) allowing authentication and data encryption.
   - AES-CCM is FIPS-complaint
   - ZigBee uses a slightly modified version of CCM called CCM*, which gives more flexibility than the standard CCM

3. **Integrity and Freshness of Data***:* Message Integrity Code (MIC) can be used to make sure that the data has not been altered in transit.
   - ZigBee supports 16, 32, 64, and 128 bit MIC lengths.
   - MIC is generated using the CCM* protocol.

BACKUP SLIDES

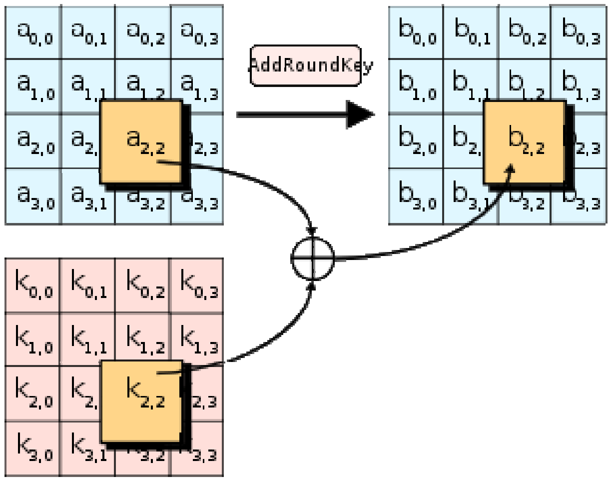
Convert to State Array

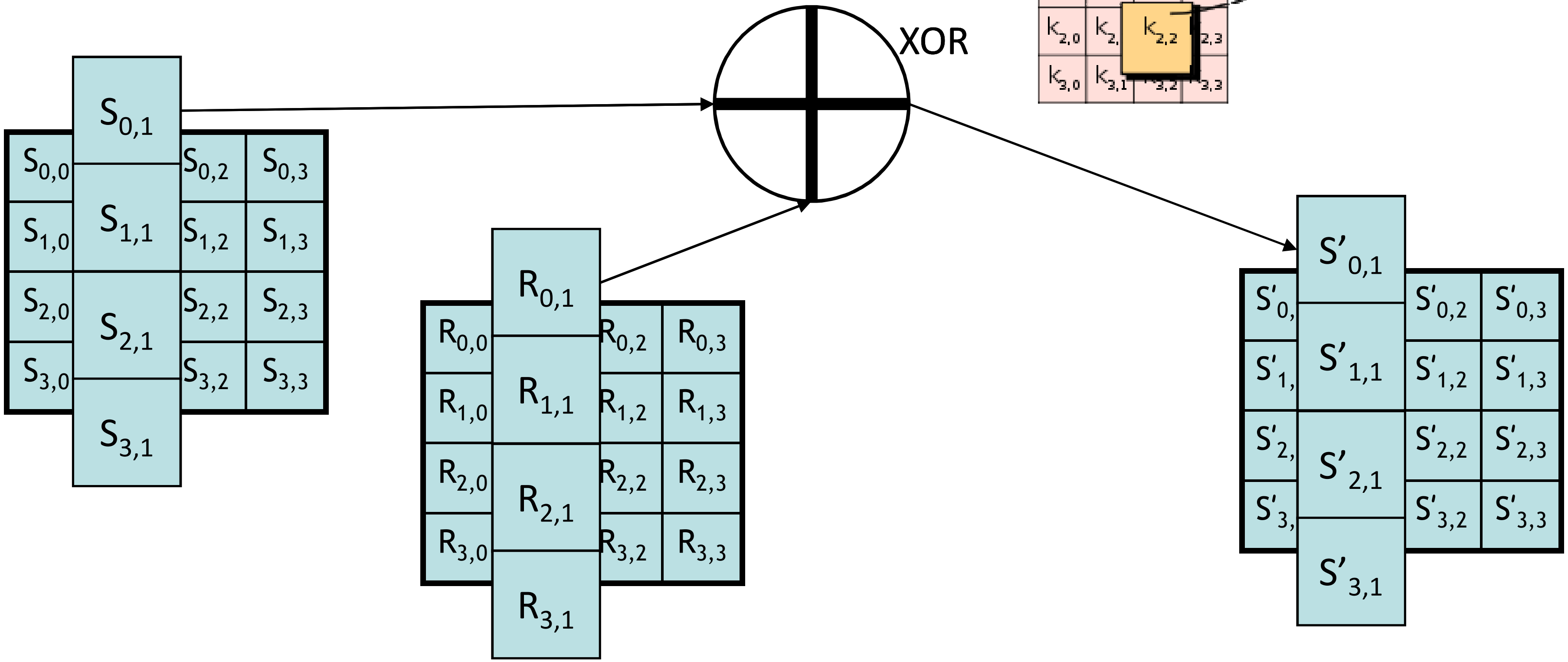
Input block:



AddRoundKey

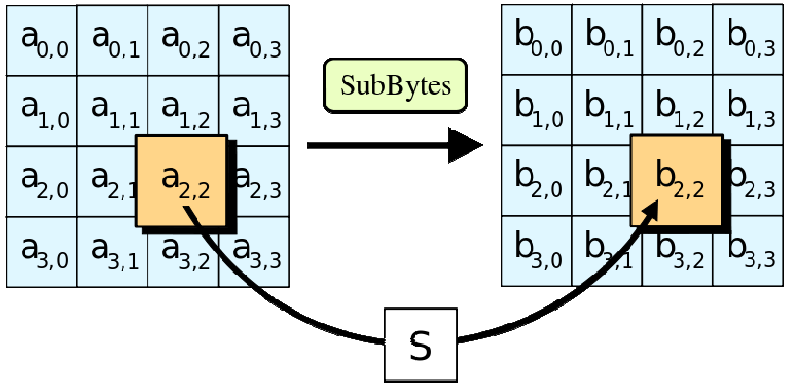
- XOR each byte of the Round Key with its corresponding byte in State Array





SubBytes

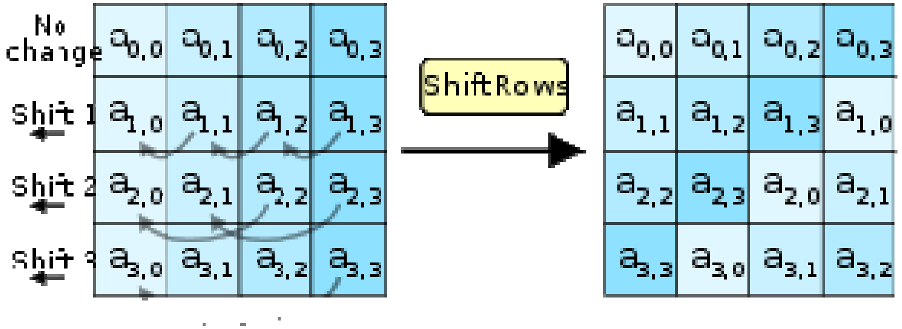
- Replace each byte in the state array with its corresponding value from the State Array.

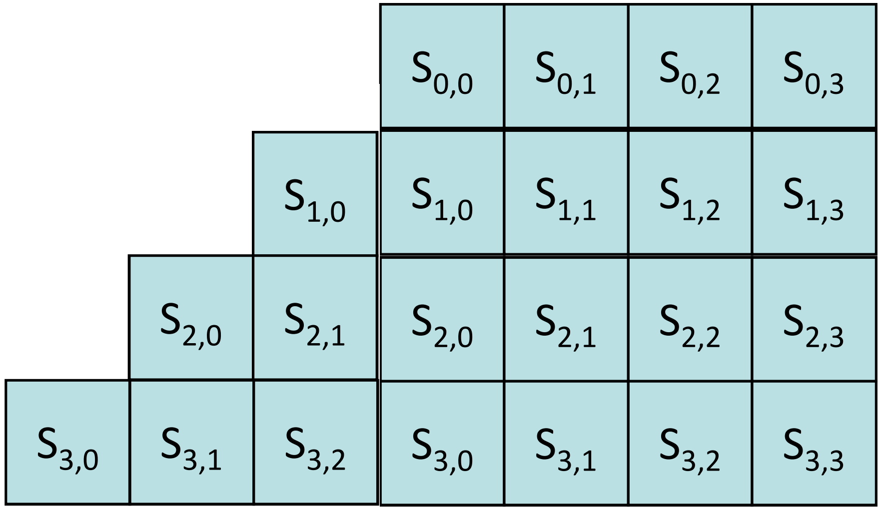


x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

ShiftRows

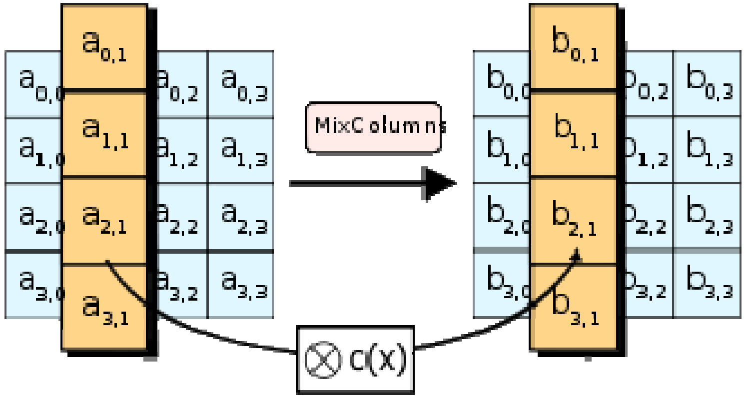
- Last three rows are cyclically shifted





MixColumns

- Apply MixColumn transformation to each column



$$S'_{0,c} = (\{02\} \bullet S_{0,c}) \oplus (\{03\} \bullet S_{1,c}) \oplus S_{2,c} \oplus S_{3,c}$$

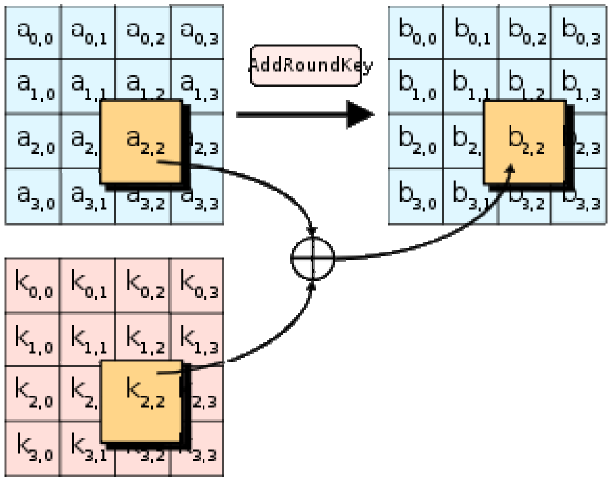
$$S'_{1,c} = S_{0,c} \oplus (\{02\} \bullet S_{1,c}) \oplus (\{03\} \bullet S_{2,c}) \oplus S_{3,c}$$

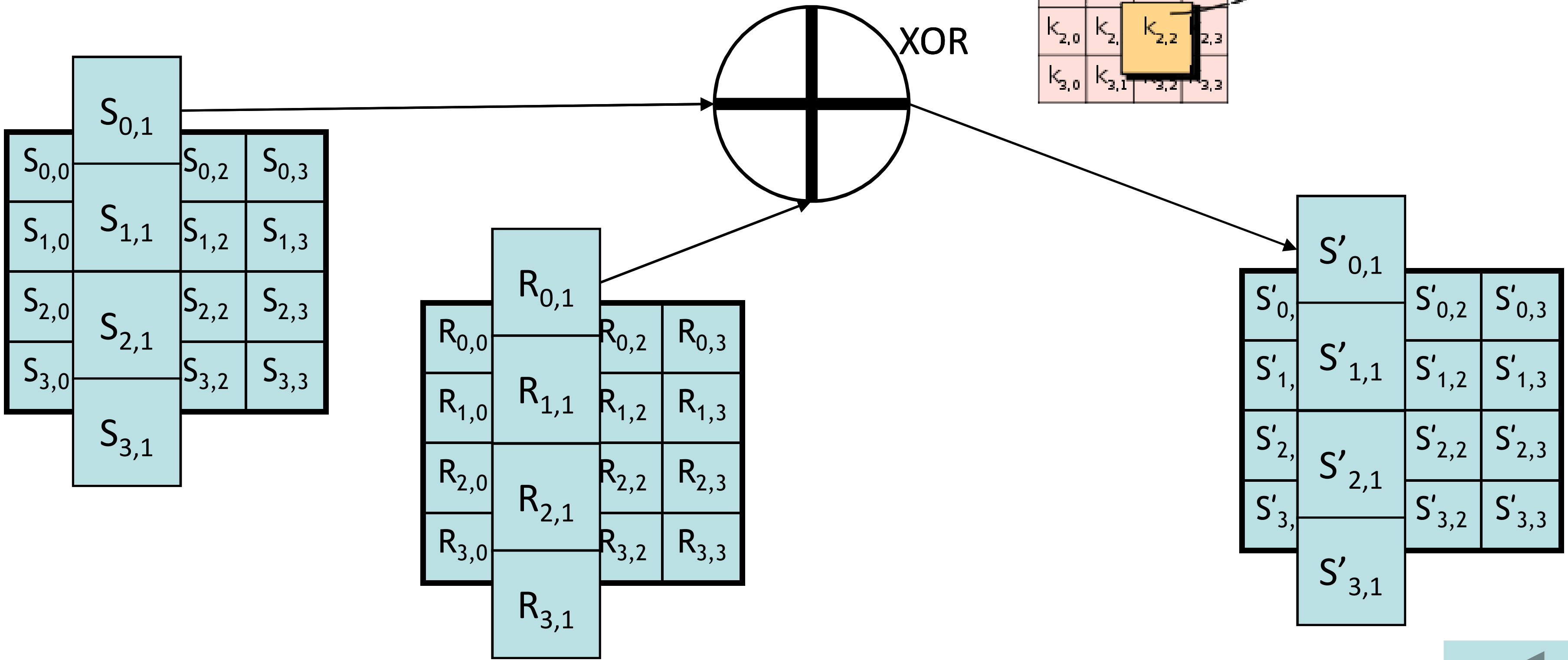
$$S'_{2,c} = S_{0,c} \oplus S_{1,c} \oplus (\{02\} \bullet S_{2,c}) \oplus (\{03\} \bullet S_{3,c})$$

$$S'_{3,c} = (\{03\} \bullet S_{0,c}) \oplus S_{1,c} \oplus S_{2,c} \oplus (\{02\} \bullet S_{3,c})$$

AddRoundKey

- XOR each byte of the Round Key with its corresponding byte in State Array





The Birthday Paradox

- Given a set of N elements, from which we sample k elements (k<<N) randomly (with replacement). What is the probability of encountering at least one repeating element?
- First, the probability of **no repetition for each sample respect to the previous ones** is:
  - The first element x_1 can be anything then $Pr(x_1)= 1/N$
  - When choosing the second element x_2 then $Pr(x_2 \neq x_1) = 1\text{-}1/N$
  - When choosing x_3, then $Pr(x_3 \neq x_2 \text{ and } x_3 \neq x_1) = 1-(1/N+1/N) = 1-2/N$
  - When choosing x_k, then $Pr(x_k \neq x_{k-1} \ldots, \text{and } x_k \neq x_1) = 1-(k-1)/N$
- Second, the probability of **no repetition for any sample** is the joint probability

  $(1 - 1/N)(1 - 2/N)\ldots(1 – (k-1)/N)$

  If k<<N is verified the approximation $(1-k/N) \approx e^{-k/N}$ applies.

  Hence $(1 - 1/N)(1 - 2/N)\ldots(1 – (k-1)/N) \approx e^{-1/N}e^{-2/N} \ldots e^{-(k-1)/N} = e^{-k(k-1)/2N}$

- Hence the probability of at least one repetition after k samples is:

$$\mathbf{1 - e^{-k(k-1)/2N}}$$

The Birthday Paradox

- How many samples k in a set of N elements do you need if you want the probability of at least one repetition to be ε ?

  Solve for k equation $1 - e^{-k(k-1)/2N} = \varepsilon$
- Therefore:

  $\varepsilon = 1 - e^{-k(k-1)/2N}$

  $k(k-1) = -2N \ln(1/(1-\varepsilon)) = 2N \ln(1-\varepsilon)$

  (1) $\mathbf{k \approx sqrt(2N\ln(1-\varepsilon))}$

  if $\varepsilon << 1$ then $2N \ln(1-\varepsilon) \approx 2N\varepsilon$

  (2) $\mathbf{k \approx sqrt(2N\varepsilon)}$

- Example: the Birthday Paradox with N=365 and $\varepsilon = 0.5$. Use (1):

  k ≈ 1.177 sqrt(N) ≈ 23.

- Example: an hash function 128 bit ($N=3.4\cdot10^{38}$) with probability of at least one collision is $\varepsilon = 10^{-18}$. Use (2):

  k ≈ sqrt(2Nε) = 2.6 $\cdot10^{10}$.

  "only" after k $\approx 2.6 \cdot 10^{10}$ attemps the probability of a collision is 10^{-18} !!

CMAC

To generate an ℓ-bit CMAC tag (t) of a message (m) using a b-bit block cipher (E) and a secret key (k), one first generates two b-bit sub-keys (k_1 and k_2) using the following algorithm (this is equivalent to multiplication by x and x^2 in $GF(2^b)$).

Let $\ll$ denote the standard left-shift operator and $\oplus$ denote bit-wise XOR:

- Calculate a temporary value $k_0 = E_k(0)$.
- If $msb(k_0) = 0$, then $k_1 = k_0 \ll 1$, else $k_1 = (k_0 \ll 1) \oplus C$; where C is a certain constant that depends only on b. (Specifically, C is the non-leading coefficients of the lexicographically first irreducible degree-b binary polynomial with the minimal number of ones: 0x1B for 64-bit, 0x87 for 128-bit, and 0x425 for 256-bit blocks.)
- If $msb(k_1) = 0$, then $k_2 = k_1 \ll 1$, else $k_2 = (k_1 \ll 1) \oplus C$.
- Return keys (k_1, k_2) for the MAC generation process.

CMAC

As a small example, suppose $b = 4$, $C = 0011_2$, and $k_0 = E_k(0) = 0101_2$.

Then $k_1 = 1010_2$ and $k_2 = 0100 \oplus 0011 = 0111_2$.

CMAC tag generation process is as follows:

- Divide message into b-bit blocks $m = m_1 \parallel \ldots \parallel m_{n-1} \parallel m_n$, where $m_1, \ldots, m_{n-1}$ are complete blocks. (The empty message is treated as one incomplete block.)
- If m_n is a complete block then $m_n' = k_1 \oplus m_n$ else $m_n' = k_2 \oplus (m_n \parallel 10\ldots0_2)$.
- Let $c_0 = 00\ldots0_2$.
- For $i = 1, \ldots, n - 1$, calculate $c_i = E_k(c_{i-1} \oplus m_i)$.
- $c_n = E_k(c_{n-1} \oplus m_n')$
- Output $t = \text{msb}_\ell(c_n)$.

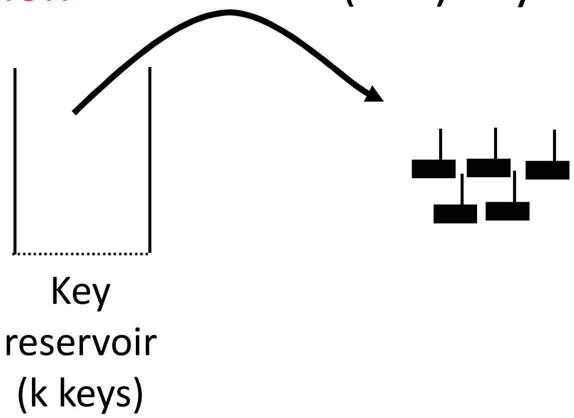
The verification process is as follows:

- Use the above algorithm to generate the tag.
- Check that the generated tag is equal to the received tag.

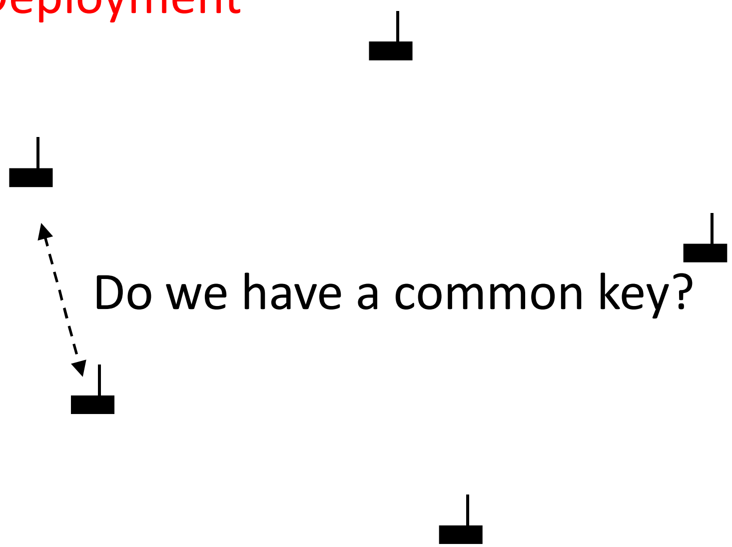
Random Key Pre-Distribution Scheme

1. Initialization

m (<<k) keys in each sensor ("key ring of the node")



2. Deployment



Probability for any 2 nodes to have a common key:

$$p = 1 - \frac{((k-m)!)^2}{k!(k-2m)!}$$

Basic Scheme

- **Initialization phase**
  - A large pool S of unique keys are picked at random.
  - For each node, m keys are selected randomly from S and pre-loaded in the node (**key ring**).
- **Direct Key Establishment phase**
  - After deployment, each node finds out with which of its neighbors it shares a key (e.g., each node may broadcast the list of its key IDs).
  - Two nodes that discover that they share a key verify that they both actually possess the key (e.g., execute a challenge-response protocol).
- **Path Key Establishment phase**
  - Neighboring nodes that **do not have a common key** in their key rings establish a shared key through a path of intermediaries
  - Each link of the path is secured in the direct key establishment phase

Setting the parameters

- Connectivity of the graph resulting after the direct key establishment phase is crucial
- A result from Random Graph Theory [Erdős-Rényi, 1959]: in order for a random graph to be connected with probability *c<1* (e.g., *c* = 0.9999), the expected degree *d* of the edge (expected number of arcs from that edge over the total number of edges) should be:

$$d = \frac{n-1}{n}\left(\ln(n) - \ln(-\ln(c))\right) \qquad (1)$$

- In this case, *d* = *pn'* (2), where *p* is the probability that two nodes have a common key in their key rings, and *n'* is the expected number of neighbors (for a given deployment density)
- *p* depends on the size *k* of the pool and the size *m* of the key ring

$$c \xrightarrow{(1)} d \xrightarrow{(2)} p \xrightarrow{(3)} k, m$$

$$p = 1 - \frac{((k-m)!)^2}{k!(k-2m)!} \qquad (3)$$

EXCELLENCE EMERGE

Setting the parameters – an example

UNIVERSITÀ DEGLI STUDI DELL'AQUILA

- Number of nodes: n = 10000
- Expected number of neighbors: n' = 40
- Required probability of connectivity after direct key establishment: c = 0.9999

- using (1) required node degree after direct key establishment: d = 18.42
- using (2) required probability of sharing a key: p = 0.46
- using (3) appropriate key pool and key ring sizes:

  k = 100000, m = 250

  k = 10000, m = 75

  …

Qualitative Analysis

- **Advantages**:
  - No need for intensive computation
  - Path key establishment have some overhead
    - decryption and re-encryption at intermediate nodes
    - communication overhead
  - No assumption on topology
  - Easy addition of new nodes

- **Disadvantages**:
  - Node capture affects the security of non-captured nodes too
    - if a node is captured, then its keys are compromised
    - these keys may be used by other nodes too
  - If a path key is established through captured nodes, then the path key is compromised
  - No authentication is provided

q-composite rand key pre-distribution

- **Basic idea:**
  - Two nodes can set up a shared key if they have **at least *q* common keys in their key rings**
  - The pairwise key is computed as the hash of all common keys
- **Advantages:**
  - in order to compromise a link key, all keys that have been hashed together must be compromised
- **Disadvantages:**
  - probability of being able to establish a shared key directly is smaller (it is less likely to have *q* common keys, than to have one)
  - Optimum key ring and key pool sizes:
    - key ring size should be increased (but: memory constraints)
    - key pool size should be decreased (but: effect of captured nodes)

Polynomial based key pre-distribution

- Let f be a bivariate t-degree polynomial over GF(q), q prime, or GF(2^n), s.t. f(x, y) = f(y, x)

$$f(x,y) = \sum_{i,j=0}^{t} a_{ij} x^i y^j = \sum_{i,j=0}^{t} a_{ij} x^j y^i$$

- Each node is pre-loaded with a polynomial share f(i, y), where i is the ID of the node, a_{ij} randon values in GF(q)
- Any two nodes i and j can compute a shared key by
  - i evaluating f(i, y) at point j and obtaining f(i, j), and
  - j evaluating f(j, y) at point i and obtaining f(j, i) = f(i, j)
- This scheme can be **unconditionally secure** and is **t-secure**
  - **unconditionally secure:** uncertainty on shared secret is not reduced by information exchange if destination IDs are known
  - **t-secure:** any coalition of at most t compromised nodes knows nothing about the shared keys computed by any pair of non-compromised nodes
- Memory requirement of the nodes is (t +1) log(q), s.t. t is limited by the memory constraints of the sensors

Polynomial based key pre-distribution

- **Operation**:
  - Let S be a pool of bivariate t-degree polynomials
  - For each node i, we pick a subset of m polynomials from the pool
  - Pre-load into node i the polynomial shares of these m polynomials computed at point i
  - Two nodes that have polynomial shares of the same polynomial f can establish a shared key f(i, j)
  - If two nodes have no common polynomials, they can establish a shared key through a path of intermediaries

- **Statistically** the scheme can tolerate the capture of more than t nodes. Infact:
  - In order to compromise a polynomial, the adversary needs to obtain t + 1 shares of that polynomial
  - It is very unlikely that t + 1 randomly captured nodes have all selected the same polynomial from the pool