Design and Development of IoT Applications

Dr. –Ing. Vo Que Son

Email: sonvq@hcmut.edu.vn

Content

- Chapter 9: 802.15.4 Link-Layer Security
  - Access Control
  - Message Integrity and Confidentiality
  - 802.15.4 Stack and Protocol
  - Security suites
  - LLSEC in Contiki OS

Overview of 802.15.4 Security

- The basic features provided by the link layer security protocol are
  - Access Control
  - Message Integrity
  - Message Confidentiality
  - Replay Protection

Access Control and Message Integrity

- Unauthorized entities should not be part of a secure network
- A mechanism to detect the above scenario
- Message integrity – message tampering should be detected – MAC
- Requires communicating parties to share a secret

Confidentiality

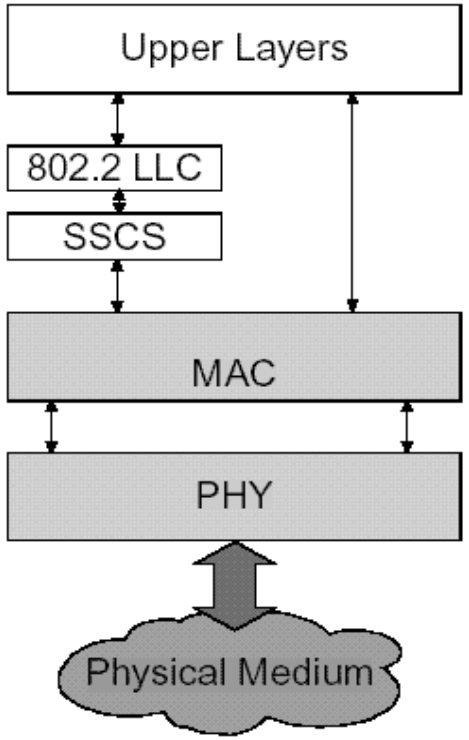
- Means of achieving – encryption
- Notion of “Semantic Security”
- The encryption must prevent an adversary from learning even partial information about the message
- This means that encryption of the same plaintext twice should result in different cipher texts
- Nonce

Replay Protection

- Adversary eavesdrops a message from a legitimate sender and replays it after a time delay ‘x’
- Sequence numbers – increased with every packet

802.15.4 Stack and Protocol

- Each node has
  - 64 – bit Node ID
  - 16 – bit Network ID
- (A node could use 16-bit Node ID)
- Two types of packets (relevant to security)
  - Data
  - ACK (Sender explicitly requests it)



LLC – Logical Link Control

SSCS – Service Specific Convergence Sub layer

802.15.4. Packet format

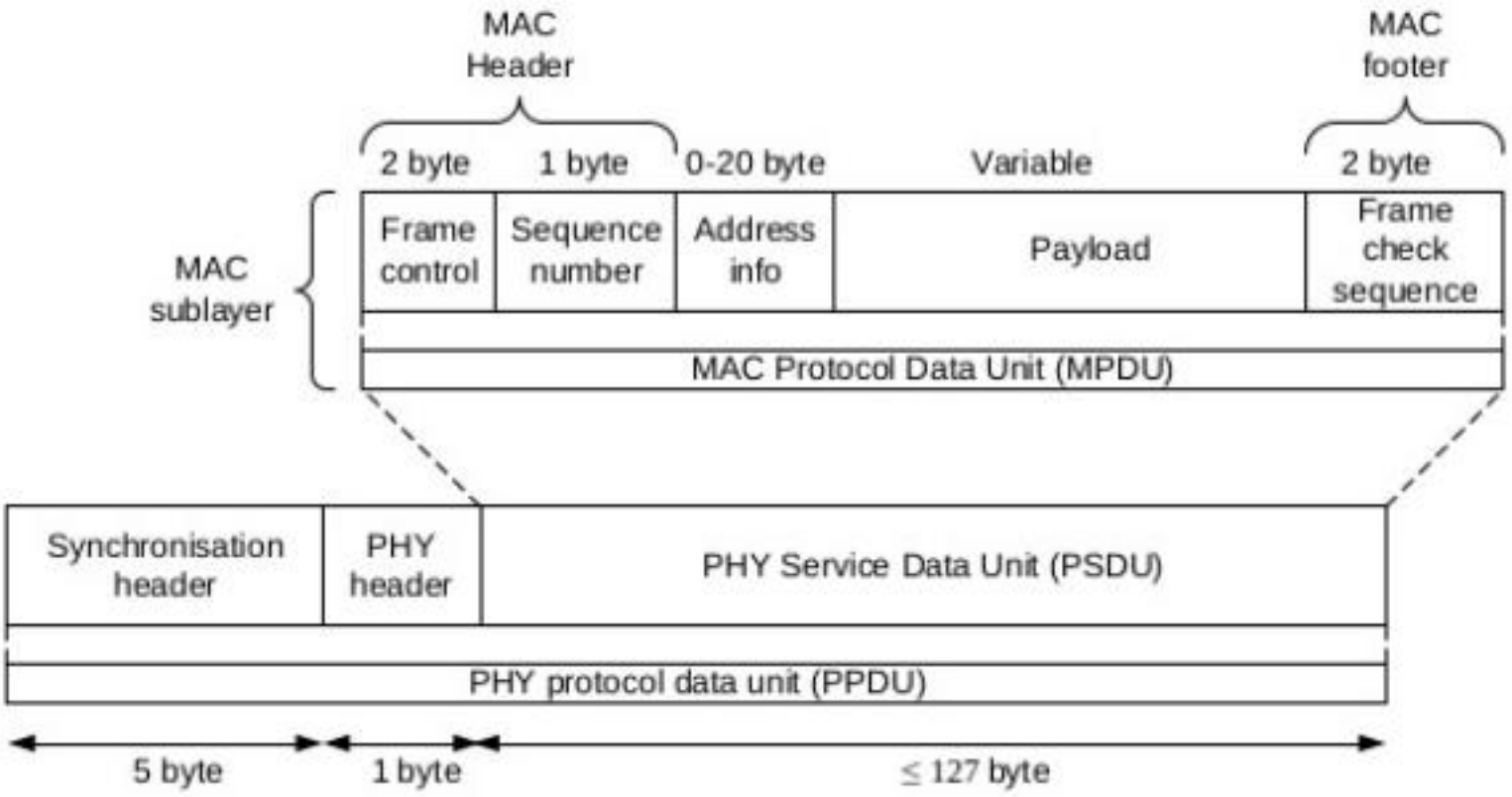
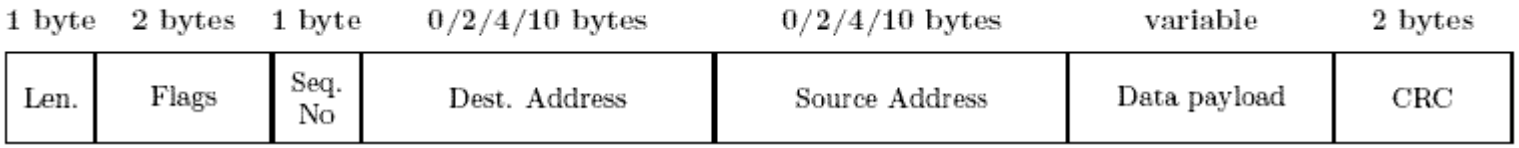
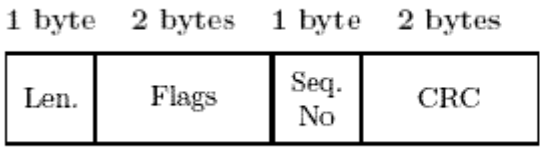


Figure 5.1: IEEE 802.15.4 Packet Description

Data and ACK packet formats

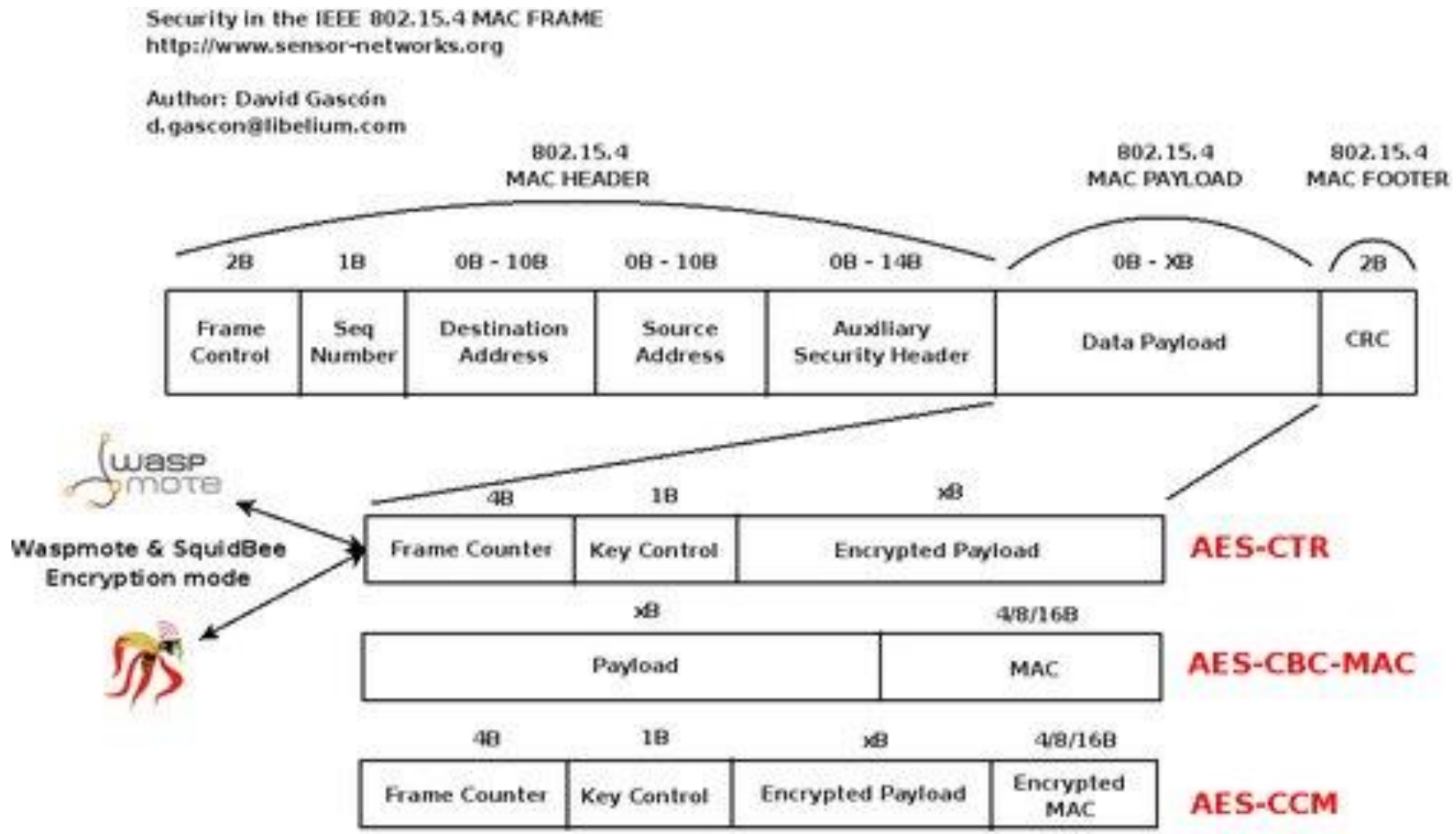


(a) Data packet format



(b) Acknowledgment packet format

Data and ACK packet formats



Where is security ?

- Handled by the Media access control layer
- The application controls the security required
- By default – “NO Security”
- Four types of packets
  - Beacon, Data, ACK, Control packets for MAC Layer
- NO Security for ACK packets
- The other packets can optionally use encryption or integrity checks

Security Suites

Security Level	Name	Description	
0x00	Null	No Security	Data is not encrypted. Data authenticity is not validated.
0x04	AES-CTR	Encryption only, CTR Mode	Data is encrypted. Data authenticity is not validated.
0x03	AES-CBC-MAC-128	128 bit MAC	Data is not encrypted. Data authenticity is validated.
0x02	AES-CBC-MAC-64	64 bit MAC	Data is not encrypted. Data authenticity is validated.
0x01	AES-CBC-MAC-32	32 bit MAC	Data is not encrypted. Data authenticity is validated.
0x07	AES-CCM-128	Encryption & 128 bit MAC	Data is encrypted. Data authenticity is validated.
0x06	AES-CCM-64	Encryption & 64 bit MAC	Data is encrypted. Data authenticity is validated.
0x05	AES-CCM-32	Encryption & 32 bit MAC	Data is encrypted. Data authenticity is validated.

How does it work?

- Application decides the choices on the security level. (A bool value)
- Access Control Lists are used to enforce these security levels (max up to 255 entries)
- If security is enforced then the MAC layer looks up the ACL table for the cryptographic material for the destination
- On packet reception, based on the flags the MAC layer decides how to process the packet

Address	Security Suite	Key	Last IV	Replay Ctr

ACL Entry Format

How does it work?

- **Address**: of the node we want to communicate with
- **Security Suite**: the security police which is being used (AEC-CTR, AES-CCM-64, AES-CCM-128,,...)
- **Key**: the 128b key used in the AES algorithm
- **Last Initial Vector (IV) and Replay Counter**: both are the same field. The Last IV is used by the source and the Replay Counter by the destination as a message ID in order to avoid reply attacks.

Address	Security Suite	Key	Last IV	Replay Ctr

ACL Entry Format

How does it work?

- Replay Counter
  - Applicable to AES-CTR and AES-CCM suites
  - The replay counter is 5 bytes long
    - Frame counter
    - Key Counter (MSB)
- Note: This is same as the nonce but is used for a different purpose.

Address	Security Suite	Key	Last IV	Replay Ctr

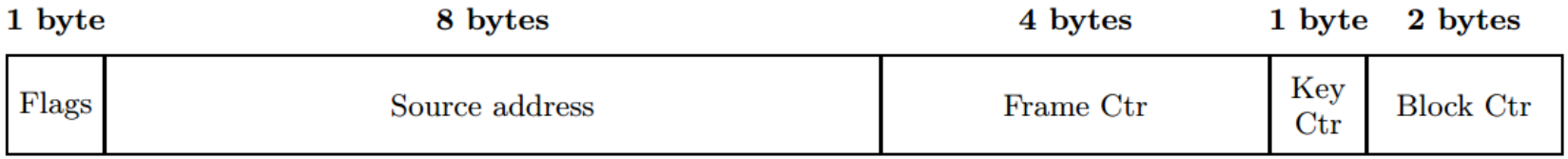
ACL Entry Format

Details of Security Suites

- NULL – no security, mandatory in all chips
- AES-CTR (Confidentiality alone)
  - Break plain text into 16-byte blocks $p_1,\ldots,p_n$
  - Compute cipher text $c_i = p_i$ *xor* $E_k(x_i)$
  - *CTR or Nonce (or IV)* x_i is necessary for the receiver to decrypt
- Clearly, the recipient needs the counter value x_i in order to reconstruct p_i

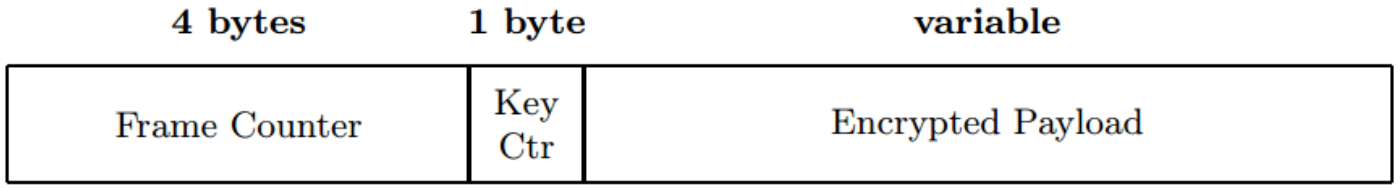
Nonce (IV)

- Is made up of
  - Static *flags* field
  - *Sender's address*
  - 3 counters
    - 4 byte *frame counter* (identifies the packet)
    - 1 byte *key counter*
    - 2 byte *block counter* (numbers the 16 byte blocks in a packet)



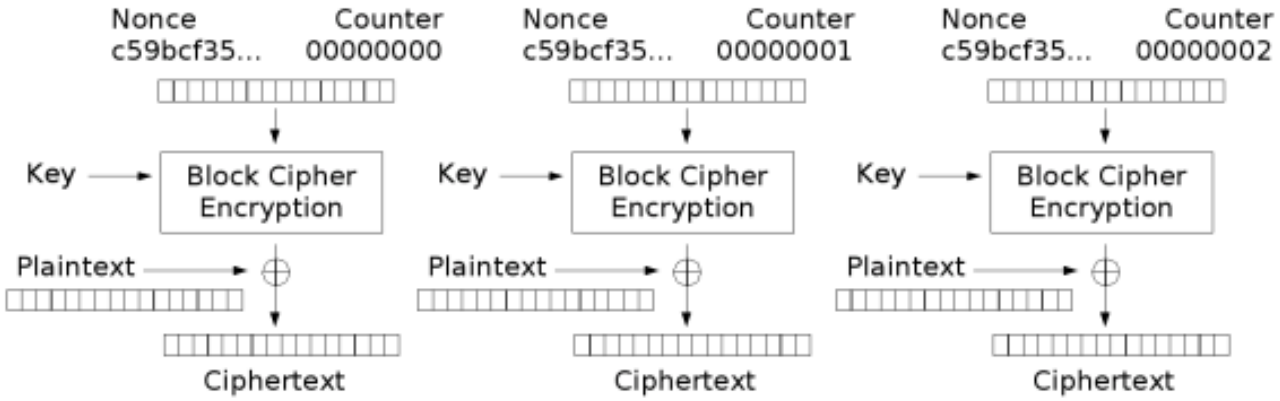
Nonce (IV)

- *Frame counter* controlled by the hardware radio
  - Sender increments it after every packet
  - When reaches max value no further encryptions are possible
- *Key counter* – application's control
  - It can be incremented if the *frame counter* ever reaches its maximum value.
  - Goal of *frame counter* and *key counter* is to prevent nonce reuse (in a single key's life-time)
- Use of *block counter*
  - ensure different *nonce*'s are used for each block
  - need not be transmitted

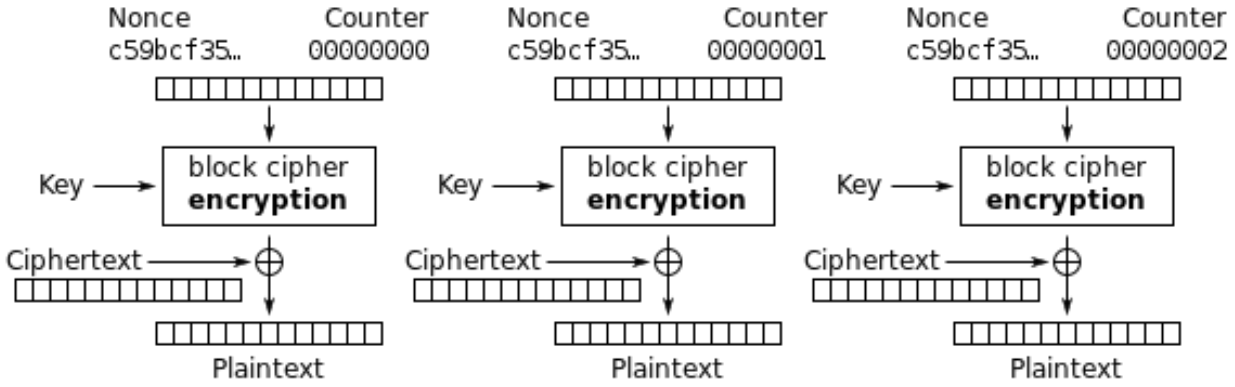


(a) AES-CTR

CTR



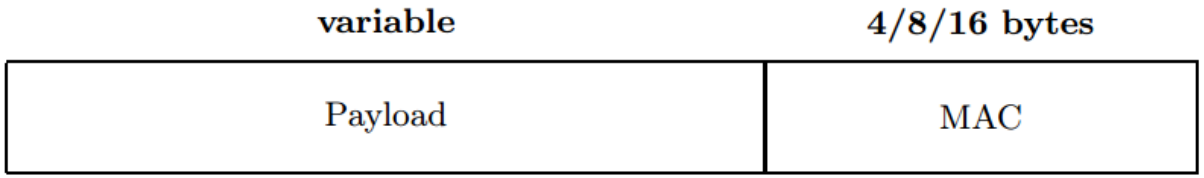
Counter (CTR) mode encryption



Counter (CTR) mode decryption

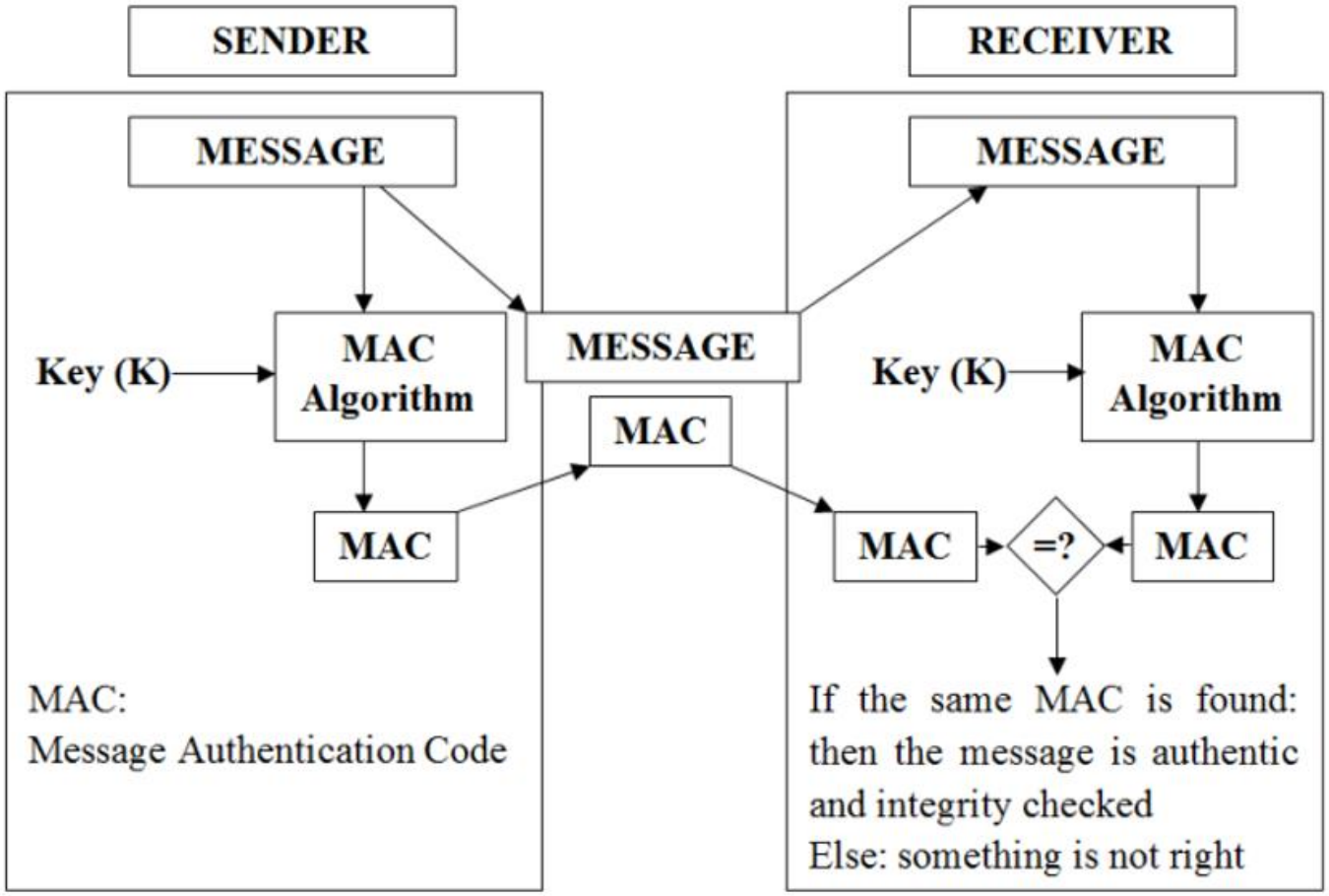
AES-CBC-MAC

- This suite provides integrity protection using CBC-MAC [11]. The sender can compute either a 4, 8, or 16 byte MAC using the CBC-MAC algorithm, leading to 3 different AES-CBC-MAC variants.
- The MAC (Message Authentication Code) can only be computed by parties with the symmetric key. The MAC protects packet headers as well as the data payload. The sender appends the plaintext data with the MAC.
- The recipient verifies the MAC by computing the MAC and comparing it with the value included in the packet



(b) AES-CBC-MAC-b, $b \in \{4, 8, 16\}$ MAC size

AES-CBC-MAC

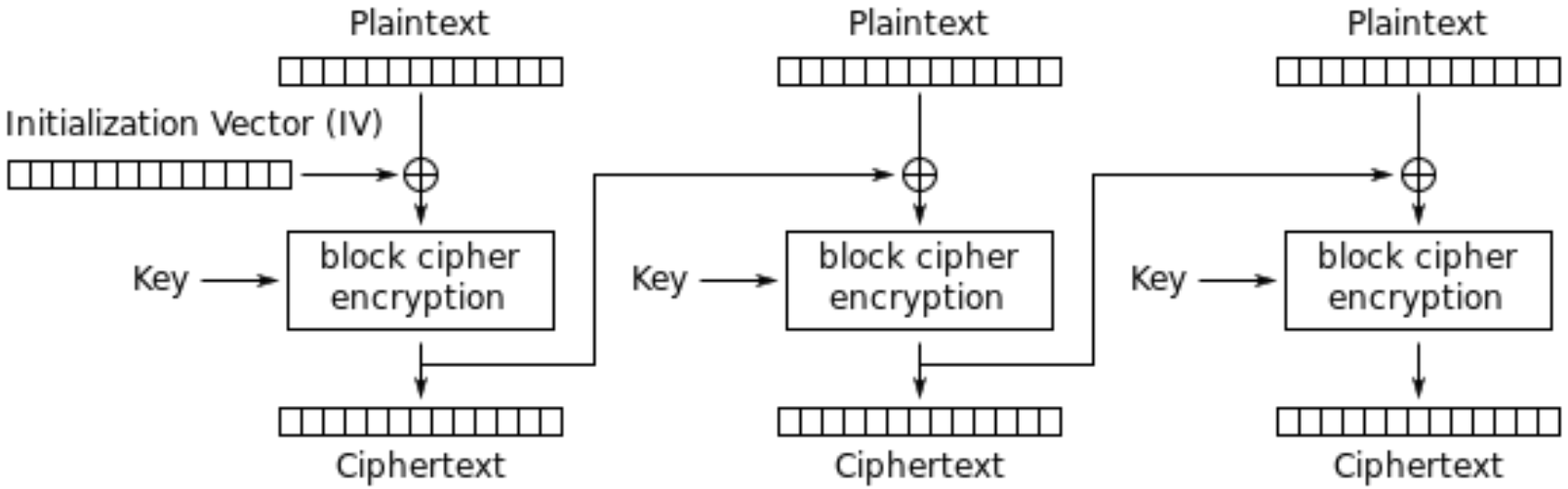

SENDER
RECEIVER
MESSAGE
MESSAGE
MESSAGE
Key (K)
MAC Algorithm
Key (K)
MAC Algorithm
MAC
MAC
MAC
=?
MAC
MAC:
Message Authentication Code
If the same MAC is found:
then the message is authentic
and integrity checked
Else: something is not right

CBC

If the first block has index 1, the mathematical formula for CBC encryption is

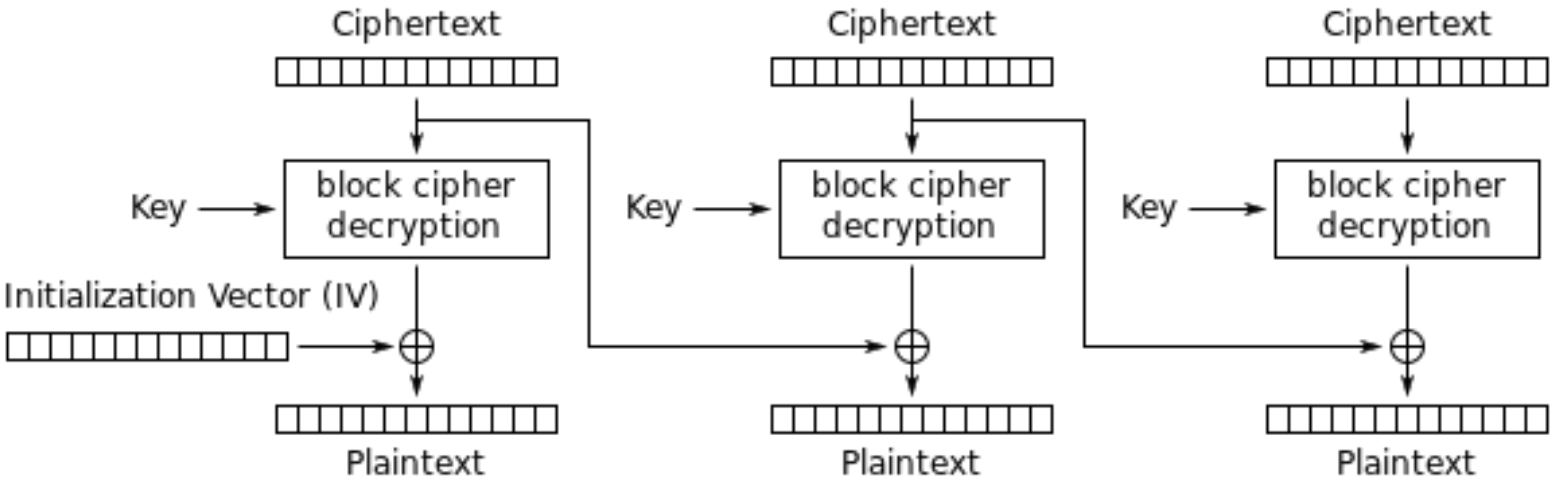
$$C_i = E_K(P_i \oplus C_{i-1}),$$

$$C_0 = IV$$



Cipher Block Chaining (CBC) mode encryption

CBC



Cipher Block Chaining (CBC) mode decryption

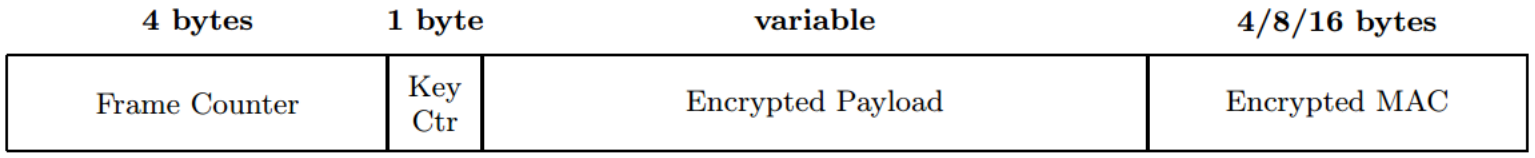
while the mathematical formula for CBC decryption is

$$P_i = D_K(C_i) \oplus C_{i-1},$$

$$C_0 = IV$$

AES-CCM

- This security suite uses CCM mode for encryption and authentication. Broadly, it first applies integrity protection over the header and data payload using CBC-MAC and then encrypts the data payload and MAC using AES-CTR mode. As such, AES-CCM includes the fields from both the authentication and encryption operations: a MAC, and the frame and key counters.
- These fields serve the same function as above.



(c) AES-CCM-b, $b \in \{4, 8, 16\}$ MAC size

Problems

- Three classes
  - Nonce management
  - Key management
  - Integrity protection

Nonce Problems

- Same Key in Multiple ACL entries
  - If used, very likely that nonce is also reused
  - This could break the confidentiality
- Simple solution
  - If using the same key then use only one ACL entry, after sending the first packet, change the ACL entry (the software must keep track of which destinations are in play)

Nonce Problems

- Loss of ACL – Power Failure
  - Assuming that the ACL is restored, what about the nonces?
  - The authors say that the 802.15.4 lacks clarity in this issue
- Recommendation – re-key after power failure
- Related issue – what happens when the node is sleeping
- Again there is presumably no clarity in the specification

Key Management Problems

- No group keying
  - First approach: Assign different ACL entries for all the nodes in the group and use the same key (Nonce management problem, ACL will be large)
  - Second approach: Single ACL entry – heavy, since for every destination the ACL has to be changed. Assuming the receivers also employ the same strategy then the receiver must switch the ACL before a packet arrives from a destination other than that in the current ACL

Key Management Problems

- Replay protection is incompatible with group keying
  - Node x transmits 100 messages, so replay counter is at 99 now
  - Node y starts transmission, its replay counter is going to 0, so the packets are dropped.

Nonce and Replay Counter

- Nonce is bound to the key – incremented with every message that is sent should use a different nonce (if two different entries use the same key, they share a nonce)
- Replay counter is bound to the sender's address. If two ACL entries share a key, they still have different replay counters.

Integrity protection Problems

- Unauthenticated encryption
  - Recommendation – do not use AES-CTR
- DoS on AES-CTR
  - Two parties X and Y communicating (Replay protection is enabled)
  - Attacker sends a very high value for the key and frame counter with garbage with sender ID X
  - This would cause the messages from legitimate sender to be dropped.

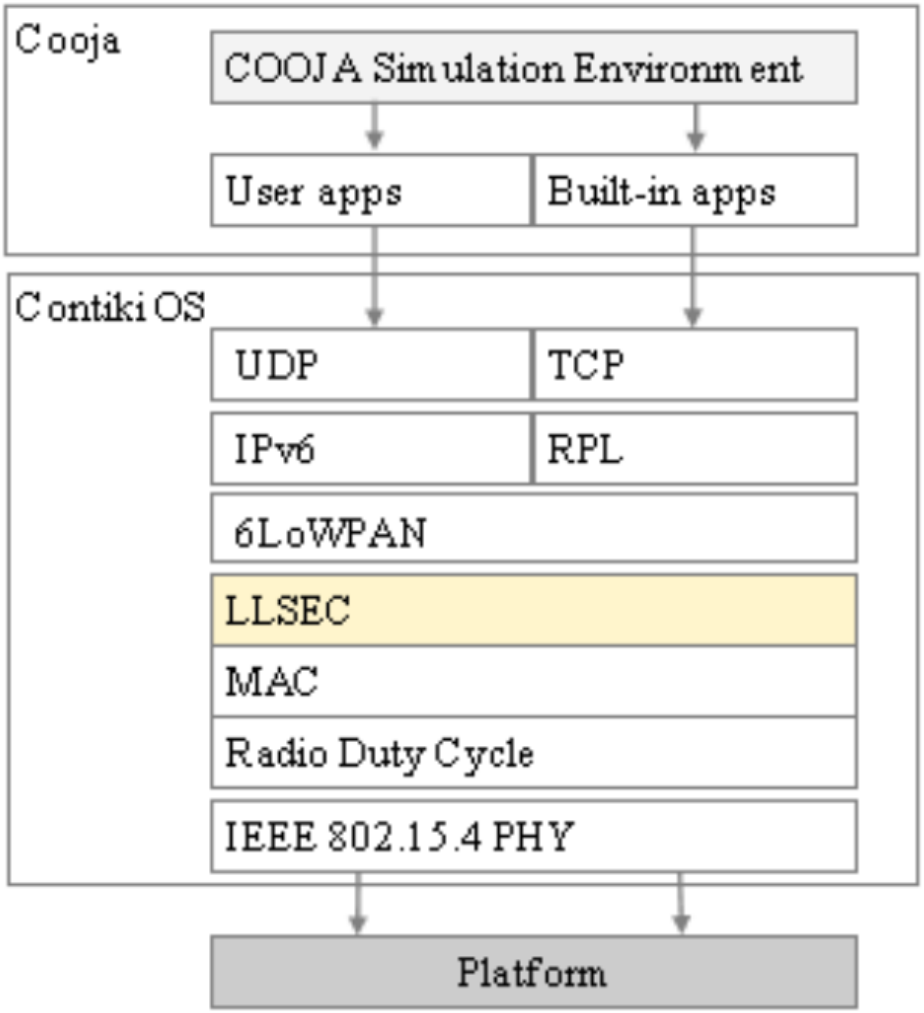
Integrity protection Problems

- ACK without MAC
  - Sequence number sent in clear
  - Adversary can send ACK's

Take-away

- Application Designers
  - Do not use AES-CTR
  - Do not rely on ACK
- Hardware Designers
  - Support for 255 ACL entries (depends on the size of the network)
  - Retail ACL in low power mode (if possible nonces too)
  - Do not support AES-CTR
- Specification writers
  - Warn against Dangerous ACL configurations
  - Better support for various keying models
  - Do not support AES-CTR
  - Authenticated ACK

LLSEC in Contiki-OS



Enable LLSEC in project-conf.h

```
/* configuration of Link Layer Security - LLSEC */
#undef AES_128_CONF
#define AES_128_CONF        aes_128_driver      /* software-based AES */

#undef LLSEC802154_CONF_ENABLED
#define LLSEC802154_CONF_ENABLED        1
#undef NETSTACK_CONF_FRAMER
#define NETSTACK_CONF_FRAMER        noncoresec_framer
#undef NETSTACK_CONF_LLSEC
#define NETSTACK_CONF_LLSEC         noncoresec_driver
#undef NONCORESEC_CONF_SEC_LVL
#define NONCORESEC_CONF_SEC_LVL  1

#define LLSEC_ANTIREPLAY_ENABLED                    0   /* disable anti-replay */
#define LLSEC_REBOOT_WORKAROUND_ENABLED             1
#define NONCORESEC_CONF_KEY { 0x00 , 0x01 , 0x02 , 0x03 , 0x04 , 0x05 , 0x06 , 0x07 ,
0x08 , 0x09 , 0x0A , 0x0B , 0x0C , 0x0D , 0x0E , 0x0F }
#endif
```

Modify Makefile to enable LLSEC

MODULES += core/net/mac core/net core/net/mac/sicslowmac core/net/mac/contikimac core/net/llsec/noncoresec