ADVANCES IN CYBER-PHYSICAL SYSTEMS

Vol. 4, No. 1, 2019

IMPLEMENTATION OPTIONS OF KEY RETRIEVAL PROCEDURES FOR THE IEEE 802.15.4 WIRELESS PERSONAL AREA NETWORKS SECURITY SUBSYSTEM

Viktor Melnyk

Lviv Polytechnic National University, 12, Bandera Str., Lviv, 79013, Ukraine.
Author's e-mail: *viktor.a.melnyk@lpnu.ua*

Submitted on 24.06.2019

***Abstract*: The paper aims at providing the technical investigation on implementation options for the key retrieval security procedures and consequent security subsystem architecture in the IEEE 802.15.4 compatible devices. Since the security procedures typically consume most processing capacity of IEEE 802.15.4 device, an efficient implementation of the security subsystem is essential. A brief functional overview of the key retrieval procedures has been provided. General investigations on key retrieval procedures implementation have been performed. Three general approaches for implementation of key retrieval procedures in the security subsystem have been considered: a) software implementation; b) hardware implementation; and c) hardware-software implementation. The aim is to determine optimum implementation approach corresponding to low-cost and low-power consumption requirements. An expediency of hardware and software implementation of the key retrieval procedures has been estimated.**

***Index Terms*: Wireless Personal Area Networks, IEEE 802.15.4, key retrieval, wireless security.**

I. INTRODUCTION

In cyber-physical systems [1], the question of organization of secure information exchange between the devices that are part of them is particularly important. First of all, it concerns devices that interact with each other through wireless communication channels. For such communication today, a number of international standards and protocols have been developed and approved. Specifically, this is a series of IEEE 802.11 standards [2] and the Wi-Fi protocol based on them, the IEEE 802.15.4 standard [3], [4] and ZigBee [5], ISA100.11a [6], MiWi, and Thread protocols, each of which further extends the standard by developing the upper layers which are not defined in IEEE 802.15.4.

For the latter standard, the issues of energy saving and the organization of operation under the conditions of minimum energy consumption are very important. This is largely a challenge for developers who are forced to implement the means for complex cryptographic transformations in IEEE 802.15.4 devices to provide secure information exchange between them in terms of their minimum power consumption. Therefore, developers of devices that support secure interaction in IEEE 802.15.4 networks typically implement information security subsystems with hardware-software partitioning, where computationally intensive cryptographic transformations are performed by hardware-oriented specialized encryption processors executing the AES algorithm [7] or the AES-CCM* [8] cryptographic protocol. However, in addition to cryptographic protocols, other security procedures, such as *key retrieval procedures*, that are not computationally intensive, must be executed, but their implementation must take into account the way the keying information is stored in the network and how the components of the security subsystem are implemented.

II. PROBLEM STATEMENT

The security operation procedures have to be performed by the medium access control (MAC) sublayer security subsystem (further named as the security subsystem). Since the security procedures typically consume most processing capacity of IEEE 802.15.4 device, an efficient implementation of the security subsystem is essential.

It is assumed that the security subsystem is a part of the MAC sublayer implementation, which is normally implemented with MAC microcontroller – i.e. programmable microcontroller intended to perform primary MAC functions, excluding security.

In paper [9], the hardware-software approach has been selected for the security subsystem implementation. An implementation of the security lookup procedures has been out of scope of this work. In current paper, we estimate an expediency of hardware / software implementation of the security procedures. During the estimation, the following assumptions are being used:

1) Security subsystem shall be implemented both in hardware and software;

2) AES-CCM* transformation is implemented in hardware;

3) Lookup procedures might be implemented in hardware or in software; keys and other security-related information from MAC PIB (PIB denotes the personal area network (PAN) Information Base) may initially be retained in hardware or in software.

The goal of the research reflected in this paper is to determine an optimum implementation approach corresponding to the IEEE 802.15.4 low-cost and low-power consumption requirements.

III. STRUCTURE OF THE ARTICLE

The paper consists of thirteen sections (including Introduction, problem statement, and the current one). The material of the following sections is structured as follows. The Section IV provides an overview of the security services in IEEE 802.15.4 Std. The Section V gives a frame securing process brief overview in order to show the place of the outgoing frame key retrieval procedure and the key descriptor lookup procedure in there. The Section VI gives a frame unsecuring process brief overview in order to show the place of the security procedures related to a key retrieval in there. In Section VII, functional analysis of lookup procedures is performed, and in the following Section VIII optimization of lookup procedures for reduced-function devices is presented. In Section IX, implementation options of the lookup security procedures are investigated and hardware-software partitioning criteria are formulated. The outgoing frame key retrieval procedure implementation options are analyzed in Section X, while the incoming frame key material retrieval procedure implementation options – in Section XI. The topics for further research are outlined in Section XII. The Section VIII – Conclusions, summarizes options for implementation of the key material retrieval procedures and consequent security subsystem architecture in the IEEE 802.15.4 compatible devices.

IV. SECURITY SUBSYSTEM FUNCTIONAL PARTITIONING

Functionally, the structure of the security subsystem can be broken down into two main units: AES-CCM* Crypto Engine (AES-CCM*-CE) and Security Processing Support Unit (SPSU), as it is shown in Fig. 1.

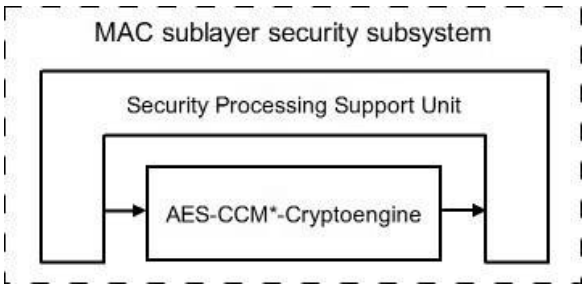

Fig. 1. Security subsystem general structure

AES-CCM*-CE unit performs AES-CCM* forward and inverse transformation and is a basic unit of the security subsystem. Further details of AES-CCM* processing as well as AES-CCM*-CE architecture are out of scope of this paper. We consider AES-CCM*-CE as a "black box", that receives appropriate input data and produces appropriate output data. All input data for AES-CCM*-CE are formed by SPSU. Output data of AES-CCM*-CE are used by SPSU to form the resulting secured or unsecured frame. We assume, that AES-CCM*-CE is hardware implemented, since its software implementation requires high performance microprocessor which energy consumption doesn't allow its application in IEEE 802.15.4 compatible devices.

SPSU unit interacts with AES-CCM*-CE and performs the following security procedures:

- Outgoing frame security procedure;
- Outgoing frame key retrieval procedure;
- Key descriptor lookup procedure;
- Incoming frame security procedure;
- Incoming security level checking procedure;
- Incoming frame key material retrieval procedure;
- Blacklist checking procedure;
- Incoming key usage policy checking procedure.

All input data for the security subsystem are handled by SPSU. They are the frame to be secured/unsecured and security-related information that is used by security subsystem during the security processing. The security-related information involves the MAC PIB security attributes, the parameters of originating primitive, and the MAC sublayer constants. Output data are corresponding secured/unsecured frame, processing status, updated MAC PIB security attributes, and security-related data to generate originating primitives for the upper layers.

V. IEEE 802.15.4 FRAME SECURING PROCESS BRIEF OVERVIEW

Functionally, frame securing process consists in execution of three procedures: outgoing frame security procedure, outgoing frame key retrieval procedure, and key descriptor lookup procedure. The outgoing frame key retrieval procedure is invoked by the outgoing frame security procedure. Also, it invokes the key descriptor lookup procedure.

The outgoing frame security procedure was shown in details in [9]. In this paper we consider the outgoing frame key retrieval procedure and the key descriptor lookup procedure. Each of them contains a number of functional operations, or steps. Let us take an overview of these procedures step by step.

A. OUTGOING FRAME KEY RETRIEVAL PROCEDURE OVERVIEW

Outgoing frame key retrieval procedure is invoked by outgoing frame security procedure in order to obtain encryption key from the MAC PIB of device. Procedure uses frame itself, information from MAC PIB, and information from originating primitive parameters. Sequence of the procedure steps is listed and described below.

1) *Key lookup data and key lookup size determination.* The aim of this step is to form the identification data which shall be used to retrieve appropriate key from the MAC PIB of device. These identification data are named as key lookup data and key lookup size.

2) *Obtaining of the key descriptor.* This step invokes the *key descriptor lookup procedure*, which is

overviewed in the next subsection. If that procedure fails, the procedure terminates and indicates failure.

3) *Setting of the key*. This step consists in assignment of key that was retrieved from the MAC PIB of device to key value which shall be used in AES-CCM* transformation.

4) *Outputs assignment*. This step is formal and means that a key retrieval process is successfully completed and a key is produced.

B. KEY DESCRIPTOR LOOKUP PROCEDURE OVERVIEW

In order to communicate securely in the personal area network (PAN) the device may use one or more encryption keys. Keys and key-related information, which is used during securing/unsecuring process, are contained into key descriptors. Each key descriptor contains exactly one key. Beside key and key-related information each key descriptor contains auxiliary information that identifies this key. All key descriptors are placed into MAC PIB of the device. The aim of key descriptor lookup procedure is to determine appropriate key among all others listed in the MAC PIB. Determination is performed by comparison of identification data formed at a first step of the outgoing frame key retrieval procedure with that data containing info key descriptors in MAC PIB. If matching entries were not found, procedure would fail.

VI. BRIEF OVERVIEW OF IEEE 802.15.4 FRAME UNSECURING PROCESS

Functionally, frame unsecuring process consists in execution of six procedures: incoming frame security procedure, incoming security level checking procedure, incoming frame key material retrieval procedure, incoming key usage policy checking procedure, key descriptor lookup procedure (the same as it was overviewed earlier), and blacklist checking procedure.

In this paper, we consider the incoming security level checking procedure, incoming frame key material retrieval procedure, incoming key usage policy checking procedure, which use the lookup operations on the MAC PIB security tables. Other procedures are out of scope of this paper and were shown in details in [9]. Let us take an overview of these procedures step by step.

C. INCOMING SECURITY LEVEL CHECKING PROCEDURE OVERVIEW

The aim of the procedure is to determine whether the security level specified in the incoming frame corresponds to the minimum requirements of the security level for actual frame type. Types and subtypes1 of the frames with the associated minimum security levels are contained into the MAC PIB of device. Determination is performed by comparison of the security level specified in the incoming frame with the minimum security level for actual frame type and subtype. Beside the minimum security level, value MAC PIB contains information indicating whether the remote device can override security minimum. The procedure completes successfully if security level specified in the incoming frame is greater than or equal to minimum security level. Otherwise, the procedure fails if remote device cannot override security minimum, and passes conditionally if it can override.

D. INCOMING FRAME KEY MATERIAL RETRIEVAL PROCEDURE OVERVIEW

Incoming frame key material retrieval procedure is invoked by incoming frame security procedure in order to obtain encryption key and other key-related information from the MAC PIB of device. Procedure uses frame itself and information from the MAC PIB. Sequence of the procedure steps is listed and described below (first two steps are the same as in outgoing frame key retrieval procedure).

1) Key lookup data and key lookup size determination.

2) Obtaining of the key descriptor.

3) Device lookup data and device lookup size determination. The aim of this step is to form the identification data, which shall be used to retrieve security-related information regarding remote device originating the incoming frame. This information is stored in the MAC PIB of device. The identification data are named as device lookup data and device lookup size.

4) Obtaining of the device descriptor and key device descriptor. This step invokes the blacklist checking procedure, which is overviewed in the next subsection. If that procedure fails, the procedure terminates and indicates failure.

5) Outputs assignment. This step is formal and means that key material retrieval process is successfully completed, and key and key-related information are produced.

E. BLACKLIST CHECKING PROCEDURE OVERVIEW

The aim of the procedure is to determine whether the remote device originating the incoming frame belongs to devices that may use identified key. The list of devices associated with each key is contained into the MAC PIB of device. Determination is performed by comparison of identification data formed at the third step of incoming frame key material retrieval procedure with that data contained info the MAC PIB of device. If matching entries were not found, procedure would fail. Otherwise, procedure shall check whether device is marked as “blacklisted”, i.e. whether keying material associated with this device may be used. If so, the procedure completes successfully, in other case it fails.

F. INCOMING KEY USAGE POLICY CHECKING PROCEDURE OVERVIEW

The aim of the procedure is to determine whether the identified key may be used to unsecure the frame of

[1] Frame subtype means type of the MAC command frame.

present type. Types and subtypes of frames, which can be unsecured with appropriate key, are listed in the MAB PIB of device for each key. Determination is performed by comparison of incoming frame type and subtype with those ones associated with identified key in MAB PIB. If matching entries were not found, procedure would fail.

VII. FUNCTIONAL ANALYSIS OF IEEE 802.15.4 LOOKUP PROCEDURES

Lookup operations on MAC PIB security tables are performed in MAC PIB. These operations require significant amount of processor time, since the tables are big enough and sometimes include several embedded lookups. If the tables are implemented in the dedicated memory, the lookup operations can be implemented in software or in hardware as well. If they reside in the MAC microcontroller memory, the lookup operations shall be implemented in software only.

Following subsection deals with four procedures: KeyDescriptor lookup procedure, blacklist checking procedure, incoming key usage policy checking procedure, and incoming security level checking procedure. These procedures use the lookup operations on the MAC PIB security tables.

In fact, implementation of the lookup procedures represents serious standing alone task, which implies the following:

- Determination or investigation of a lookup algorithm in order to reduce a lookup time;
- Determination of the maximum number of the devices in the PAN;
- Determination of the number of keys (or PAN keying model);
- Analysis of the MAC PIB security tables contents and relations between and inside of them;
- Optimisation of the tables in order to reduce their sizes and redundancy;
- Design of the memory architecture (either MAC microcontroller memory or a dedicated one) etc.

Further, we will perform general analysis of the lookup procedures for a *reduced-function device* (RFD) and a *full-function device* (FFD), review and estimate their implementation options, determine their features and highlight their advantages and disadvantages. The FFD can operate in three modes serving as a PAN coordinator, a coordinator, or a device. An FFD can talk to RFDs or other FFDs, while an RFD can talk only to an FFD. An RFD is intended for applications that are extremely simple, such as a light switch or a passive infrared sensor; they do not have the need to send large amounts of data and may only be associated with a single FFD at a time. Consequently, the RFD can be implemented using minimal resources. An FFD has to be able to communicate with a number of RFDs or/and other FFDs; this feature makes it more "resource-hungry".

Each device in the PAN can communicate with one or more another devices. Depending on which *keying model* [10] is used in the PAN, one or more encryption keys are used by device for communication. All keys are contained into *KeyDescriptor* entries of the *macKeyTable* of the MAC PIB. Besides the key *KeyDescriptor* contains key-related information that is used during frame securing/unsecuring process, and auxiliary information that is used for *KeyDescriptor* identification.

Selection of appropriate key and key-related information performs KeyDescriptor lookup procedure. It compares the pair of input values *KeyLookupData* and *KeyLookupSize* with pair of values contained into the *KeyDescriptor*. Their coincidence means that communicating pair of devices uses exactly this *KeyDescriptor* (i.e. exactly this key and key-related information).

The KeyDescriptor lookup procedure is applied either during securing or unsecuring of a frame.

The number of *KeyDescriptor* entries that *macKeyTable* contains is equal to the number of keys that this device uses. One key may be used by device to communicate with one or more remote devices within the PAN.

During input frame unsecuring, the device has to determine whether the sending remote device belongs to the devices that use identified key. For this reason, each *KeyDescriptor* contains the list of devices, for which this key is used. It is contained into the *KeyDeviceList* entry of the *KeyDescriptor*. Verification of the device in the *KeyDeviceList* entry performs blacklist checking procedure (with embedded DeviceDescriptor lookup procedure).

Also, during unsecured input frame, the device has to determine whether identified key may be used to unsecure the frame of present type. The frame types unsecured with identified key are supported are determined in the *KeyUsageList* entry of the *KeyDescriptor*. Verification of the frame type in the *KeyUsageList* entry performs key usage policy checking procedure.

Since these three lookup procedures operate in the same MAC PIB security table: *macKeyTable*, they shall be implemented in the same manner. Usage of different implementation approaches within these procedures is inexpedient.

During input frame unsecuring, device has to check whether the frame to be unsecured corresponds to the minimum security requirements that are applied to this type of MAC frame. For this reason, the MAC PIB consumes *macSecurityLevelTable* with the set of *SecurityLevelDescriptor* entries. Each *SecurityLevel Descriptor* determines minimum security level used for appropriate frame type. If the frame is a MAC Command frame, security minimum may differ depending on the command type. Verification of the security minimum requirements in *macSecurity LevelTable* performs incoming security level checking procedure.

Since incoming security level checking procedure operates in *macSecurityLevelTable*, its implementation

does not depend on implementation of three other lookup procedures.

VIII. OPTIMIZATION OF LOOKUP PROCEDURES FOR RFD

Stating that the RFD can communicate only with one single device in the PAN, the lookup procedures for RFD may be significantly simplified. Common keying models [10] that are appropriate for sensor networks are the following:

- Network shared keying;
- Pairwise keying;
- Group keying;
- Hybrid keying.

These modes imply that all nodes, groups of nodes, or pairs of nodes use one symmetric key to communicate with each other. Depending on the keying model, FFD may use n, $1 \le n \le k$ different keys, where k is a number of devices that the FFD communicates with. In present case, the number of the *KeyDescriptor* entries in the *macKeyTable* is equal to k.

The RFD, which can communicate only with one single device in the PAN, may use only one key. Consequently, the number of the *KeyDescriptor* entries in the *macKeyTable* is 1. This *KeyDescriptor* contains one *KeyDeviceDescriptor* and one *DeviceDescriptor* in *KeyDeviceList*. Therefore:

- The KeyDescriptor lookup procedure might be eliminated for RFD, since only one KeyDescriptor is in its possession.
- The blacklist checking procedure might be simplified for RFD to check the Blacklisted element of the KeyDeviceDescriptor.
- The incoming key usage policy checking procedure and incoming security level checking procedure seem to be the same (remain unchangeable) for the FFD and the RFD.

Selection of appropriate key and key-related information is done by comparison of the pair of input values *KeyLookupData* and *KeyLookupSize* with contained into the *KeyDescriptor* pair of values. However, since RFD uses just one key descriptor, there is no need to perform selection. Hence, the *KeyLookupData* and *KeyLookupSize* values do not need to be determined in *outgoing frame key retrieval procedure* and in *incoming frame key material retrieval procedure* for the RFD. Moreover, *KeyDescriptor* entry, which is stored in *macKeyTable* of RFD, shall not include *LookupData* and *LookupSize* entries, and consequently *KeyLookupList* and *KeyLookupListEntries* as well.

Verification of the *DeviceDescriptor* in *KeyDevice Descriptor* entry is done by comparison of input values *DeviceLookupData* and *DeviceLookupSize* with corresponding values constructed from *KeyDevice Descriptor* entries, but only if *UniqueDevice* element of the *KeyDeviceDescriptor* is FALSE. Reasoning that RFD communicates with one device only, that element will always be TRUE. Thus, verification of the *DeviceDescriptor* in *KeyDeviceDescriptor* entry shall not be performed for RFD and *UniqueDevice* element of the *KeyDeviceDescriptor* shall be eliminated. Hence, the *DeviceLookupData* and *DeviceLookupSize* values do not need to be determined in incoming frame key material retrieval procedure for the RFD. Moreover, *DeviceDescriptor* entry, which is stored in *macKeyTable* of RFD, shall not include *PAN ID*, *ShortAddress*, and *ExtAddress* elements.

Since *KeyDeviceList* element of the *KeyDescriptor* shall contain only one entry, the value of *KeyDeviceListEntries* element of the *KeyDescriptor* will always be 1, thus it can be eliminated.

The *KeyLookupData* and *KeyLookupSize* values do not need to be determined in *outgoing frame key retrieval procedure* and in incoming frame key material retrieval procedure for the RFD, if the *KeyDescriptor lookup procedure* is eliminated for the last.

The *DeviceLookupData* and *DeviceLookupSize* values do not need to be determined in *incoming frame key material retrieval procedure* for the RFD, if the *blacklist checking procedure* is eliminated for the last.

Primary *macKeyTable* structure for the RFD with eliminated entries and elements is shown in Fig. 2. Eliminated entries and elements are delineated shaded.

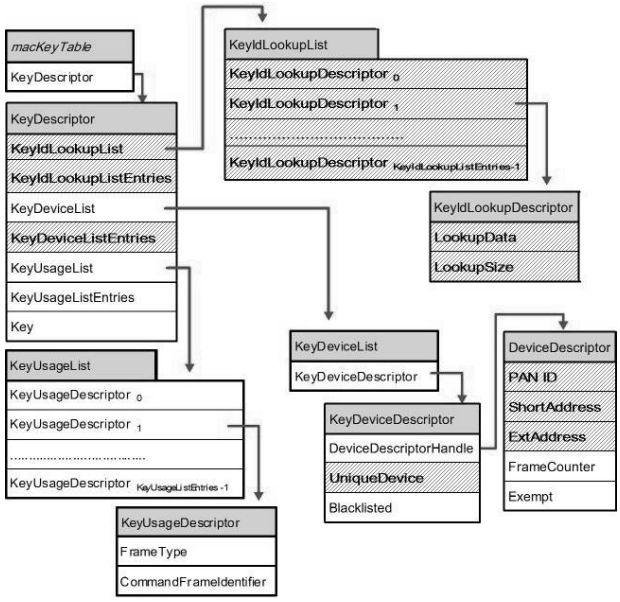

Fig. 2. Primary macKeyTable structure for the RFD with eliminated entries and elements

Optimized *macKeyTable* structure for RFD shall contain *KeyDeviceDescriptor* entry, *KeyUsageList* entry, *KeyUsageListEntries* element and *Key* element. The *KeyDeviceDescriptor* entry shall contain *FrameCounter*, *Exempt*, and *Blacklisted* elements. The structure of the remaining parts of the *macKeyTable* will remain the same for RFD and for FFD. Optimized *macKeyTable* structure for RFD is shown in Fig. 3.

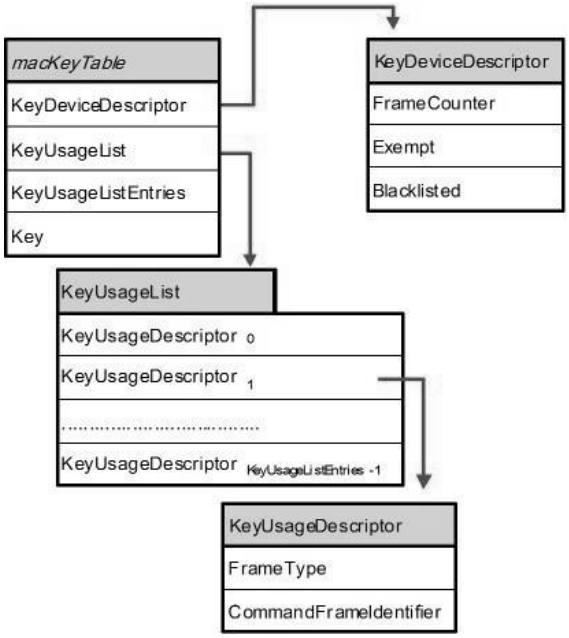

Fig. 3. Optimized macKeyTable structure for the RFD

Further structure and logic optimisations in the MAC PIB security tables for RFD are possible.

It is proved that the *macKeyTable* of the RFD contains one *KeyDescriptor* entry, one *KeyDevice Descriptor* entry, and one *DeviceDescriptor* entry. However, it is not proved that having one *KeyDescriptor* entry means that this entry is the right one, having one *DeviceDescriptor* entry means that this entry is the right one as well. Therefore, further in this paper we assume that both variants are possible, and the lookups might still be needed for the RFD. If so, the *macKeyTable* structure for the RFD will not be simplified, however its size will be reduced to one *KeyDescriptor* entry, one *KeyDeviceDescriptor* entry, and one *DeviceDescriptor* entry. All lookup procedures will be the same for the FFD and for the RFD, and all data to perform the look-ups still must be determined. Further investigations to determine whether the RFD needs lookups must be done.

Assuming that another keying model that implies usage of *n* different keys in communication within pair of nodes is used in the PAN, the *macKeyTable* shall contain *n KeyDescriptor* entries each with the same *KeyDeviceDescriptor* and *DeviceDescriptor*. However, such keying models require usage of additional key selection parameters but it is not provided in [3].

IX. IMPLEMENTATION OPTIONS OF IEEE 802.15.4 LOOKUP SECURITY PROCEDURES

G. APPROACHES TO IMPLEMENTATION OF LOOKUP PROCEDURES

Generally, execution of the lookup procedures needs:

- Software- or hardware-implemented data fetch and analysis facilities;
- Software- or hardware-implemented MAC PIB security tables.

The interaction scheme for the lookup procedures is shown in Fig. 4.

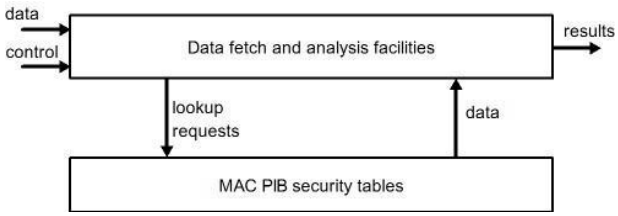

Fig. 4. Interaction scheme for the lookup procedures

H. SOFTWARE IMPLEMENTATION OF LOOKUP PROCEDURES

Software implementation of the lookup procedures provides data fetch and analysis by the means of MAC microcontroller. The MAC PIB security tables may be implemented either in software or in hardware, as it is shown in Fig. 5 and in Fig. 6, respectively. The first software implementation option for the lookup procedures is featured by scalability, simplicity, flexibility, and minimum hardware needs. However, it has a number of disadvantages:

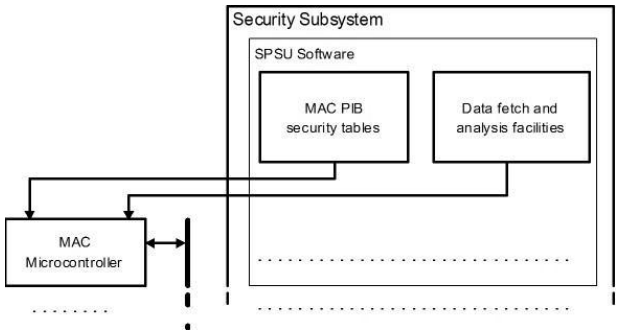

Fig. 5. Software implementation of lookup procedures with software-implemented MAC PIB security tables

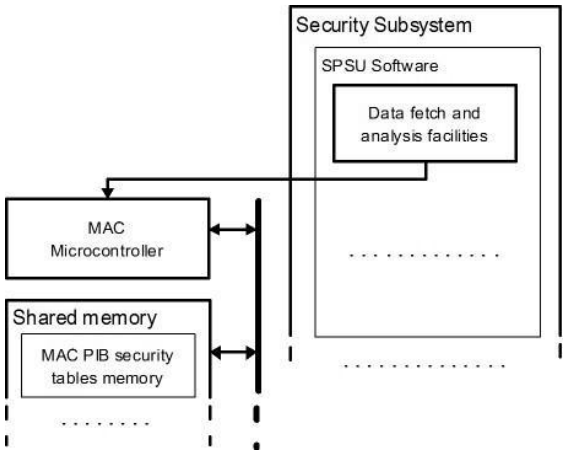

Fig. 6. Software implementation of lookup procedures with MAC PIB security tables in separate hardware memory

- Time expenses for the microcontroller are high;
- Since the MAC PIB security tables contain keys and key-related information, software implementation of

the MAC PIB security tables might be unsafe, because this information is represented as a plain source code and might be easily accessed by intruder. In this case, usage of protected memory areas for keys and key-related information storage is preferable. For more information refer to [11];

- Storage of the MAC PIB security tables in software requires significant resources of the external flash memory (where MAC software shall be stored), as they must be respectively coded, interpreted/compiled etc.

The second software implementation option for the lookup procedures implies usage of the separate hardware memory (e.g. RAM) to store the MAC PIB security tables and it is featured by following:

- Due to hardware implementation of the MAC PIB security tables, data fetch process is faster than it is in the former case, but MAC microcontroller is still busy with data fetch and analysis.
- Keys and key-related information are represented not as a plain source code, which might be accessible to intruders, that is much safer then it is in former case.
- Hardware implementation of MAC PIB security tables requires additional memory resources.

I. HARDWARE IMPLEMENTATION OF LOOKUP PROCEDURES

Hardware implementation of the lookup procedures provides data fetch and analysis by the specialized hardware means like ASIC or programmable hardware (e.g. FPGA, CPLD). In this case the MAC PIB security tables shall be implemented in hardware as well (other implementation is inexpedient). Two major options for hardware implementation of the lookup procedures are highlighted. The first option implies usage of the separate hardware memory (e.g. RAM) to store the MAC PIB security tables. It is shown in Fig. 7.

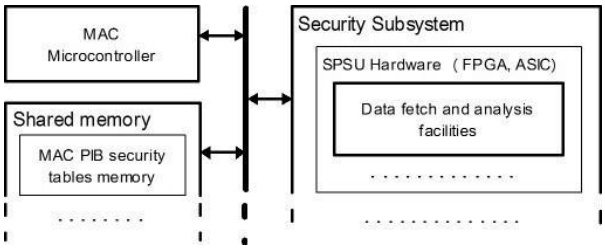

Fig. 7. Hardware implementation of lookup procedures with MAC PIB security tables in a separate hardware memory

This option has the following advantages:

- Fast data analysis;
- The keys and key-related information storage safety level is not lower than in the former case;
- MAC microcontroller has direct access to the MAC PIB security tables, that is important for data initial load and reload.
- The disadvantages of this option are the following:
- Usage of the standard memory interface for data fetch brings time overheads;
- Hardware implementation is featured by higher power consumption.

The second hardware implementation option for the lookup procedures implies usage of embedded memory of the programmable chip (FPGA or CPLD) or integrated memory within ASIC to store the MAC PIB security tables. It is shown in Fig. 8.

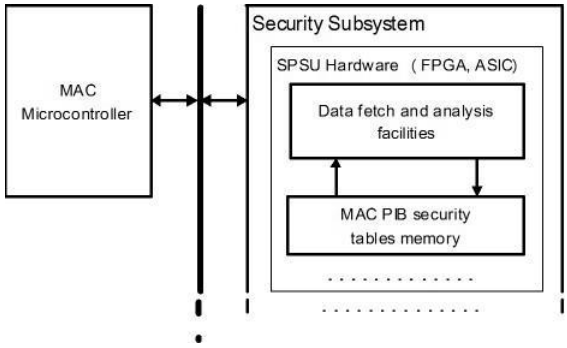

Fig. 8. Hardware implementation of lookup procedures with MAC PIB security tables in an embedded or integrated memory

The advantages of this option are the following:

- Fast data fetch and analysis (due to memory interface optimization). Full hardware implementation provides the highest performance;
- The highest storage safety level for the keys and key-related information;
- No separate hardware memory needs to store the MAC PIB security tables.

Along with advantages, this option has disadvantages as well:

- High hardware overheads and power consumption;
- Additional mechanisms to perform data initial load and reload into the MAC PIB security tables are required; interaction mechanisms between microcontroller and embedded or integrated memory are required.

In order to determine which hardware or software option is expedient, it is necessary to investigate questions of effective execution of the lookup algorithms and perform their timing estimations for software and for hardware taking into account lookup memory sizes for RFD and FFD.

J. IMPLEMENTATION OF LOOKUP PROCEDURES FOR RFD WITH OPTIMIZED MAC PIB SECURITY TABLES

Actual Subsection investigates the lookup procedures implementation for the RFD only if the KeyDescriptor lookup procedure is eliminated and blacklist checking procedure is simplified for the last.

The blacklist checking procedure, if it is simplified, shall perform checking of the *Blacklisted* element of the *KeyDeviceDescriptor*. The procedure has no input

arguments and has only BLC_STATUS output. The pseudo code below explains blacklist checking procedure operation for RFD.

```
IF KeyDeviceDescriptor(Blacklisted) = FALSE THEN
   BLC_STATUS <= PASSED;
ELSE BLC_STATUS <= FAILED;
END IF;
```

The 1-bit length *Blacklisted* element may be implemented either in software or in hardware depending on implementation of the blacklist checking procedure for the RFD.

The incoming key usage policy checking procedure and incoming security level checking procedure are the same for RFD and FFD. They use the lookups and should be implemented according to one of the options described before.

X. IMPLEMENTATION OPTIONS OF OUTGOING FRAME KEY RETRIEVAL PROCEDURE

The outgoing frame key retrieval procedure consists of the following steps:

- Key lookup data and key lookup size determination;
- Obtaining of the KeyDescriptor;
- Setting of the key;
- Outputs assignment.

Unlike the outgoing frame security procedure, the steps of the outgoing frame key retrieval procedure can only be executed sequentially, since each following step uses the results of the previous step. The procedure shall be executed in four conditional periods as it is shown in Fig. 9.

The procedure uses two types of operations: a) operations on originating primitive parameters, MAC PIB security attributes, and MAC constants, and b) operations on a frame fields. Operations on originating primitive parameters and operations of analysis of the frame fields' values could be effectively implemented in either software or hardware. Implementation effectiveness of the operations of usage of the frame fields to form the resulting data sets and internal values depends on the size of appropriate fields.

Since the basic step of the procedure is to obtain the key descriptor, thus it shall be considered firstly.

K. "OBTAINING OF THE KEYDESCRIPTOR" IMPLEMENTATION OPTIONS

The inputs for this step are *KeyLookupDataValue* and *KeyLookupSizeValue*. The outputs are *KeyDescriptor Value* and *status*. Since this step represents a separate procedure, which belongs to the lookup procedures, all further considerations are detailed in Section IX.

L. "SETTING OF THE KEY" AND "OUTPUTS ASSIGNMENT" IMPLEMENTATION OPTIONS

These steps consist in trivial output assignment and can be practically integrated with the "obtaining of the KeyDescriptor" step in Period 2 according to Fig. 9.

Implementation options for these steps are fully dependent on the "obtaining of the key descriptor" implementation. For FFD and RFD, if the KeyDescriptor lookup procedure is not eliminated for the last, these steps shall be implemented either in software or hardware dependent on data fetch and analysis facilities on the KeyDescriptor lookup procedure implementation.

For the RFD, if the KeyDescriptor lookup procedure is eliminated (actually, this is only an assumption), OFKR_STATUS output may not be used, thus it can be removed. Both steps can be optimized and perform key assignment as it is shown by pseudo code below.

```
OFKR_Key <= macKeyTable(Key);
```

In this case, since the key is used in hardware-implemented AES-CCM* transformation, for RFD it is reasonable to store the key in the same hardware unit that contains AES-CCM*-CE.

M. "KEY LOOKUP DATA AND KEY LOOKUP SIZE DETERMINATION" IMPLEMENTATION OPTIONS

If the KeyDescriptor lookup procedure is eliminated for RFD, the *KeyLookupData* and *KeyLookupSize* values do not need to be determined. Thus, this step shall be eliminated for RFD as well. The following investigations concern implementation options for FFD and RFD, if the KeyDescriptor lookup procedure is not eliminated for the last.

The data fetch and analysis facilities potentially may be implemented in hardware or in software.

In each *KeyDescriptor* lookups are performed using *KeyLookupData* and *KeyLookupSize* values.

In the case of explicit key identification, *KeyLookupData* and *KeyLookupSize* values are formed with operations on originating primitive parameters and MAC PIB attributes, which are in possession of the MAC microcontroller. This means that if key lookup data and key lookup size determination is implemented in hardware, the MAC microcontroller shall be still busy with the originating primitive parameters and MAC PIB attributes transmission, but hardware resources shall grow. Thus, hardware implementation is inexpedient if explicit key identification is used, and key lookup data and key lookup size software determination is expedient if explicit key identification is used.

In the case of implicit key identification, *KeyLookupData* and *KeyLookupSize* values are formed with operations on the frame fields and operations on originating primitive parameters and MAC PIB attributes. Operations on the frame fields require the frame analysis and subfields extraction to be performed firstly. Frame analysis and subfields extraction may be performed by software or by hardware. If it is performed by software, key lookup data and key lookup size software determination are expedient. Otherwise, hardware determination is expedient. Thus, key lookup data and key lookup size software determination is

expedient if implicit key identification is used, if frame analysis and subfields extraction are also implemented in software, otherwise hardware determination is expedient.

Stating on written above, implementation of this step is going to be divided between the software and hardware. However, such approach adds quite a bit of complexity to the system. Hence, it is necessary to use one implementation approach for the entire step determining more expedient implementation option among the two.

Assuming that the software can use the originating primitive parameters and MAC PIB attributes instead of the frame subfields, frame analysis can be avoided there. Thus, software implementation of the entire step seems to be more reasonable. In addition, it seems expedient to implement key lookup data and key lookup size determination in software.

Proposed implementations of the outgoing frame key retrieval procedure are shown in Fig. 10. Indexes HW (hardware) and SW (software) show proposed implementation option for each step.

XI. INCOMING FRAME KEY MATERIAL RETRIEVAL PROCEDURE IMPLEMENTATION OPTIONS

The incoming frame key material retrieval procedure, including transferred from the incoming frame security procedure “incoming key usage policy check” step, consists of the following steps:

- Key lookup data and key lookup size determination;
- Obtaining of the *KeyDescriptor*;
- Device lookup data and device lookup size determination;
- Obtaining of the *DeviceDescriptor* and *KeyDeviceDescriptor*;
- Incoming key usage policy check;

Outputs assignment.

According to the analysis, the steps can be executed in semi-parallel manner in 4 conditional periods (layers), as it is shown in Fig. 11. Parallel (or semi-parallel) execution of procedure steps is reasonable for hardware and hardware-software procedure implementation, as it provides higher efficiency of hardware usage, reduces the execution time, and has no impact on the resources.

The basic steps of the procedure are the following: “obtaining of the key descriptor”, “obtaining of the DeviceDescriptor and KeyDeviceDescriptor”, and “incoming key usage policy check”, thus they shall be considered firstly. They all represent separate procedures, which belong to the lookup procedures. According to the analysis performed above0, they should be implemented in the same manner.

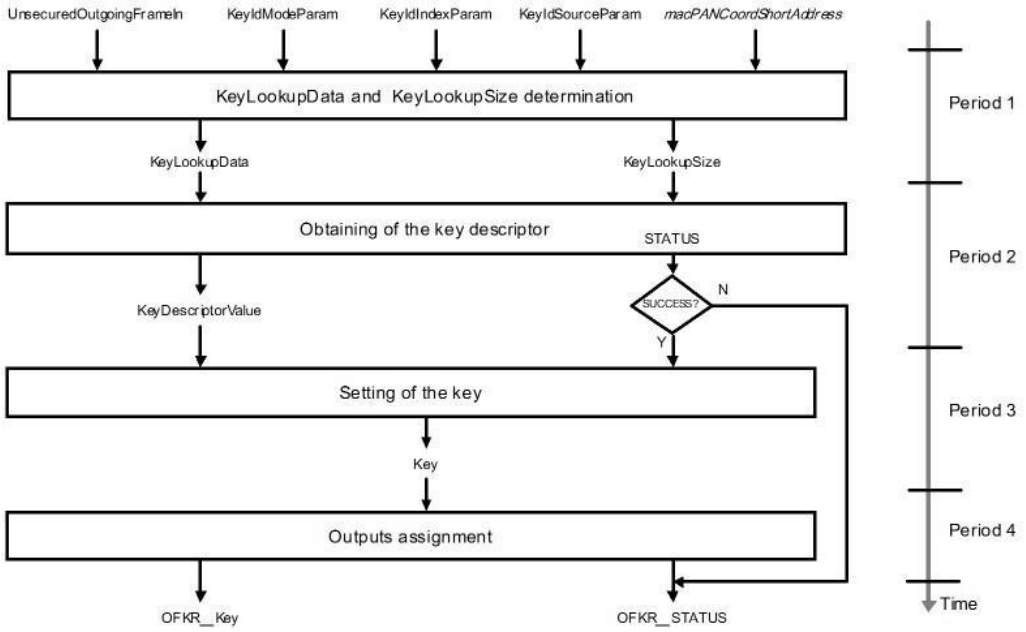

Fig. 9. Execution of outgoing frame key retrieval procedure steps

1	**SW**	*KeyLookupData* and *KeyLookupSize* determination
2	**HW**	Obtaining of the *KeyDescriptor*, Setting of the key, Outputs assignment

a) If the Key Descriptor lookup procedures is hardware-implemented

1	**SW**	*KeyLookupData* and *KeyLookupSize* determination
2	**SW**	Obtaining of the *KeyDescriptor*, Setting of the key, Outputs assignment

b) If the Key Descriptor lookup procedures is software-implemented

Fig. 10. Proposed implementations of the outgoing frame key retrieval procedure

N. "OBTAINING OF THE KEYDESCRIPTOR", "OBTAINING OF THE DEVICEDESCRIPTOR AND KEYDEVICEDESCRIPTOR", AND "INCOMING KEY USAGE POLICY CHECK" IMPLEMENTATION OPTIONS

Since these steps represent separate procedures, which belong to the lookup procedures, all further considerations are detailed in Section IX.

If the blacklist checking procedure is simplified for RFD, it shall operate to check the device's blacklist status only. The procedure operation is shown by pseudo code below:

```
IF macKeyTable(KeyDeviceDescriptor(Blacklisted)) = TRUE THEN
    BLC_STATUS <= FAILED;
ELSE BLC_STATUS <= PASSED;
END IF;
```

O. "OUTPUTS ASSIGNMENT" IMPLEMENTATION OPTIONS

This step consists in trivial outputs assignment and practically can be integrated with the "obtaining of the *DeviceDescriptor* and *KeyDeviceDescriptor*" step in Period 3 according to Fig. 11.

Implementation options for this step are fully dependent on the "obtaining of the *DeviceDescriptor* and *KeyDeviceDescriptor*" step implementation. For FFD and RFD, if the KeyDescriptor lookup procedure is eliminated, and the blacklist checking procedure is simplified for the last, this step shall be implemented in either software or hardware depending on the data fetch and analysis facilities of the lookup procedures implementation.

For the RFD, if the KeyDescriptor lookup procedure is eliminated and blacklist checking procedure is simplified, this step will look as it is shown by pseudo code below:

```
IF BLC_STATUS = PASSED THEN
    IFKR_KeyDescriptor <= macKeyTable;
    IFKR_STATUS <= PASSED;
ELSE IFKR_STATUS <= FAILED;
END IF;
```

The *macKeyTable* shall contain one *KeyDescriptor*, which includes one entry of the *KeyDeviceDescriptor* combined with *KeyDescriptor*, as it is shown in Fig. 3. In this case, since the key and key-related material are used in hardware-implemented AES-CCM* transformation, for RFD it is reasonable to store them in the same hardware unit that contains AES-CCM*-CE.

P. "DEVICE LOOKUP DATA AND DEVICE LOOKUP SIZE DETERMINATION" IMPLEMENTATION OPTIONS

If the blacklist checking procedure is simplified for RFD, the *DeviceLookupData* and *DeviceLookupSize* values do not need to be determined in incoming frame key material retrieval procedure for the RFD. Thus, this step shall be eliminated for RFD as well. The following investigations concern its implementation options for the FFD and the RFD, if the blacklist checking procedure is not simplified for the last.

This step uses operations on the frame fields and operation on the MAC PIB attributes. Input data for this step are the frame to be unsecured and *macPANCoord ShortAddress* and *macPANCoord ExtendedAddress* MAC PIB attributes.

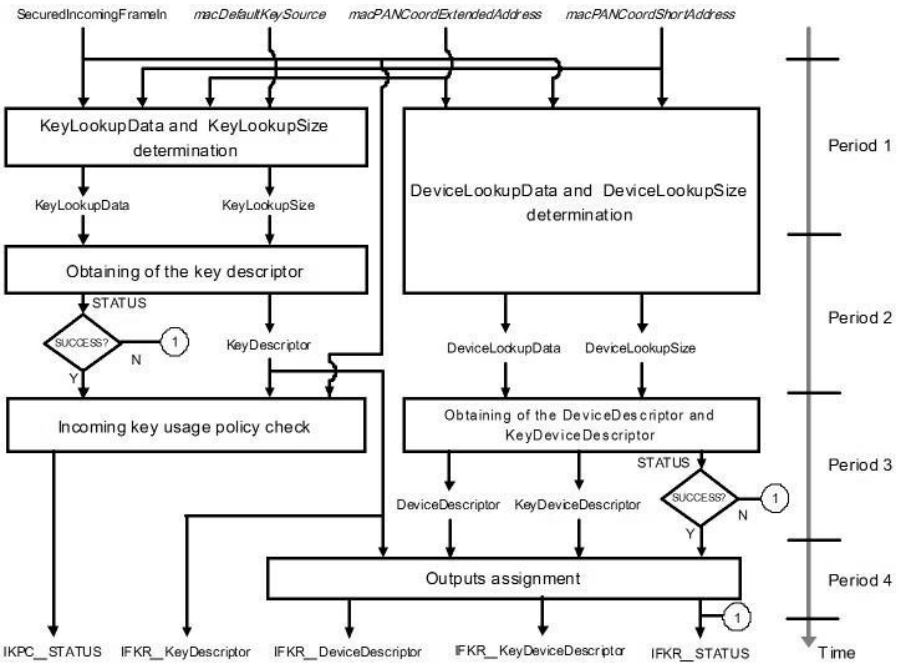

Fig. 11. Semi-parallel execution of incoming frame key material retrieval procedure steps

The step consists in simple concatenations and assignments, but in order to execute them, the frame analysis and subfields extraction have to be performed firstly. If it is performed in software, the *Device LookupData* and *DeviceLookupSize* software determination is expedient. Otherwise, hardware determination is expedient. Thus, it is reasonable to implement the step in the same manner as the frame analysis and subfields extraction is implemented.

Q. "KEY LOOKUP DATA AND KEY LOOKUP SIZE DETERMINATION" IMPLEMENTATION OPTIONS

If the KeyDescriptor lookup procedure is eliminated for RFD, the *KeyLookupData* and *KeyLookupSize* values do not need to be determined in the incoming frame key material retrieval procedure for the RFD. Thus, this step shall be eliminated for RFD as well. The following investigations concern implementation options for the FFD and the RFD, if the KeyDescriptor lookup procedure is not eliminated for the last.

As analysis has shown, the same algorithm of determination and the same frame subfields are used to determine both pairs of values: *KeyLookupData* & *KeyLookupSize*, and *DeviceLookupData* & *Device LookupSize*, if implicit key identification is used. The following pseudo code shows an alternative and effective solution for *KeyLookupData* and *KeyLookup Size* determination (implicit key identification) using *DeviceLookupData* and *DeviceLookupSize* values.

```
IF KeyIdentifierModeSubfield = "00" THEN
   KeyLookupDataValue <= DeviceLookupDataValue && 0x00;
   KeyLookupSizeValue <= DeviceLookupSizeValue + 1;
END IF;
```

It is reasonable to implement this step for implicit key identification in the same manner as *DeviceLookupData* and *DeviceLookupSize* determination. It should be implemented in the same manner as the frame analysis and subfields extraction is implemented.

In the case of explicit key identification, *KeyLookupData* and *KeyLookupSize* values shall be determined (the method of determination of them is out of scope of this paper). Input data that are used are the frame to be unsecured and *macDefaultKeySource* MAC PIB attribute.

The step consists in simple concatenations and assignments, but in order to execute them, the frame analysis and subfields extraction have to be performed firstly. If it is performed in software, the *KeyLookupData* and *KeyLookupSize* software determination is expedient. Otherwise, hardware determination is expedient. It is reasonable to implement entire step in the same manner as the frame analysis and subfields extraction are implemented.

Proposed implementations of the incoming frame key material retrieval procedure, including "incoming key usage policy check" step, are shown in Fig. 12. Indexes HW (hardware) and SW (software) show proposed implementation option for each step.

If the Key Descriptor lookup procedure is eliminated and the blacklist checking procedure is simplified for the RFD, the incoming frame key material retrieval procedure will look as it is shown by the pseudo code below. We suggest implementing it in hardware.

1	**SW**	*DeviceLookupData* and *DeviceLookupSize* determination
2	**SW**	*KeyLookupData* and *KeyLookupSize* determination
3	**SW**	Obtaining of the *KeyDescriptor*
	SW	Incoming key usage policy check
4	**SW**	Obtaining of the *DeviceDescriptor* and *KeyDeviceDescriptor*, Outputs assignment

a) Software lookup procedures implementation, software frame analysis and subfields extraction

1	**SW**	*DeviceLookupData* and *DeviceLookupSize* determination		
2	**SW**	*KeyLookupData* and *KeyLookupSize* determination		
3	**HW**	Obtaining of the *KeyDescriptor*		
4	**HW**	Obtaining of the *DeviceDescriptor* and *KeyDeviceDescriptor*, Outputs assignment	**HW**	Incoming key usage policy check

c) Hardware lookup procedures implementation, software frame analysis and subfields extraction

1	**HW**	*DeviceLookupData* and *DeviceLookupSize* determination	**HW**	*KeyLookupData* and *KeyLookupSize* determination
2	**SW**	Obtaining of the *KeyDescriptor*		
	SW	Incoming key usage policy check		
3	**SW**	Obtaining of the *DeviceDescriptor* and *KeyDeviceDescriptor*, Outputs assignment		

b) Software lookup procedures implementation, hardware frame analysis and subfields extraction

1	**HW**	*DeviceLookupData* and *DeviceLookupSize* determination	**HW**	*KeyLookupData* and *KeyLookupSize* determination
2	**HW**	Obtaining of the *KeyDescriptor*		
3	**HW**	Obtaining of the *DeviceDescriptor* and *KeyDeviceDescriptor*, Outputs assignment	**HW**	Incoming key usage policy check

d) Hardware lookup procedures implementation, hardware frame analysis and subfields extraction

Fig. 12. Proposed implementations of the incoming frame key material retrieval procedure

```
IF macKeyTable(KeyDeviceDescriptor(Blacklisted)) = FALSE
THEN
    IFKR_KeyDescriptor <= macKeyTable;
    EXECUTE incoming_key_usage_policy_checking_procedure (
                    IFKR_KeyDescriptor,
                    FrameTypeSubfield);
    IFKR_STATUS <= PASSED;
  ELSE IFKR_STATUS <= FAILED;
  END IF;
```

XII. TOPICS FOR FURTHER RESEARCH

6) To investigate questions of the lookup procedures implementation for the RFD. The result will be answers to the questions: is the KeyDescriptor lookup procedure eliminated for the RFD, and is the blacklist checking procedure simplified for the RFD.

7) To investigate structures of the outgoing frame key retrieval procedure and the incoming frame key material retrieval procedure for the RFD.

8) To determine the lookup algorithm for implementation of the lookup procedures.

9) To estimate timing expenses for the lookup procedures execution in software in order to make sure that the lookups software implementation will suit overall security subsystem performance constraints. The result will be determination of the certain architecture approach for the security subsystem.

10) To develop algorithms of the unsecured and secured frame analysis and subfields extraction.

XIII. CONCLUSIONS

The conclusions summarize options for the key material retrieval procedure implementation and consequent security subsystem architecture in the IEEE 802.15.4 compatible devices.

R. SUMMARY OF OUTGOING FRAME KEY RETRIEVAL PROCEDURE IMPLEMENTATION OPTIONS

If the *Key Descriptor lookup procedure* is eliminated for the RFD, we suggest implementing the *outgoing frame key retrieval procedure* in hardware. All the following paragraphs in the actual summary summarize the *outgoing frame key retrieval procedure* implementation for FFD and RFD, if the *KeyDescriptor lookup procedure* is not eliminated for the last.

11) The “obtaining of the key” step consists in execution of the *key descriptor lookup procedure*. No certain option for the lookup procedures implementation has been proposed yet. Therefore, two implementation options: software and hardware, which potentially are possible, will be considered in this summary.

12) The *Key Descriptor lookup procedure* implementation approach determines the implementation for the “setting of the key” and “outputs assignment” steps.

13) The following steps: “obtaining of the key”, “setting off the key”, and “outputs assignment”, should be combined into one step and executed in one conditional period.

14) The *KeyLookupData* and *KeyLookupSize* values should be determined in software.

15) Since the retrieved key shall be used in hardware-implemented AES-CCM* transformation, we are not considering key transmission back into software.

S. SUMMARY OF INCOMING FRAME KEY MATERIAL RETRIEVAL PROCEDURE IMPLEMENTATION OPTIONS

All the following paragraphs in the actual summary summarize the incoming frame key material retrieval procedure implementation for FFD and RFD, if the KeyDescriptor lookup procedure is not eliminated and the blacklist checking procedure is not simplified for the last.

16) The following steps: “obtaining of the KeyDescriptor”, “obtaining of the DeviceDescriptor and KeyDeviceDescriptor”, and “incoming key usage policy check”, consist in execution of the appropriate lookup procedures. No certain option for the lookup procedures implementation has been proposed yet. Therefore, two implementation options: software and hardware, which potentially are possible, will be considered in this summary.

17) The KeyDescriptor lookup procedure, the blacklist checking procedure, and the incoming key usage policy checking procedure should be implemented in the same manner.

18) The following steps: “obtaining of the DeviceDescriptor and KeyDeviceDescriptor”, and “outputs assignment”, should be combined into one step and executed in one conditional period.

19) Implementation approach for the following procedures: the KeyDescriptor lookup procedure and the blacklist checking procedure determines the implementation for the “obtaining of the KeyDescriptor”, “obtaining of the DeviceDescriptor and KeyDevice Descriptor”, and “outputs assignment” steps.

20) The following steps: “KeyLookupData and KeyLookupSize determination” and “Device LookupData and DeviceLookupSize determination”, should be implemented in the same manner as the frame analysis and subfield extraction are.

21) If the following steps: “KeyLookupData and KeyLookupSize determination” and “DeviceLookup Data and DeviceLookupSize determination”, are implemented in hardware, they should be executed in parallel.

22) If the following steps: “obtaining of the DeviceDescriptor and KeyDeviceDescriptor” and “incoming key usage policy check”, are implemented in hardware, they should be executed in parallel, since they operate in different entries of the *macKeyTable*.

23) If the following steps: “KeyLookupData and KeyLookupSize determination” and “Device LookupData and DeviceLookupSize determination”, are implemented in software, the “DeviceLookupData and DeviceLookupSize determination” step should be executed firstly.

24) Since the retrieved key material shall be used in hardware-implemented AES-CCM* transformation, we are not considering key material transmission back into software.

REFERENCES

[1] Melnyk A. Cyber-Physical Systems Multilayer Platform and Research Framework // *Advances in cyber-physical systems*. - 2016. – Vol. 1, Num. 1. – C. 1–6.

[2] *IEEE 802.11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. (2016 revision). IEEE-SA. 14 December 2016. doi:10.1109/IEEESTD.2016.7786995. ISBN 978-1-5044-3645-8.

[3] *IEEE Std 802.15.4TM*, IEEE Standard for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements—Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs). Second edition, September 2006.

[4] *IEEE Std 802.15.4TM 2011*, IEEE Standard for Local and metropolitan area networks—Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs). Revision of IEEE Std 802.15.4-2006, Approved 14 August 2012 by American National Standards Institute.

[5] Gascón, David (February 5, 2009). "*Security in 802.15.4 and ZigBee networks*". [Online]. Available: http://www.libelium.com/security-802-15-4-zigbee/ [Accessed: Nov. 25, 2018].

[6] "*ISA100 Committee Home Page*": https://www.isa.org/isa100/ [Accessed: Nov. 25, 2018].

[7] Federal Information Processing Standards (FIPS) Publication 197. *Announcing the ADVANCED ENCRYPTION STANDARD (AES).* November 26, 2001. Available at http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf [Accessed: Nov. 25, 2018]

[8] NIST Special Publication 800-38C. Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality. May 2004.

[9] Viktor Melnyk. Security Architecture Technical Investigation for IEEE 802.15.4 Low-Rate Wireless Personal Area Networks. *Scientific-Technical Journal "Advances in Cyber-Physical Systems". Vol. 1, No. 2*, 2018. – pp. 103–118.

[10] Sastry, N., and Wagner, D., "Security considerations for IEEE 802.15.4 networks", Proceedings of the 2004 ACM workshop on Wireless security. Philadelphia, PA, USA Pages: 32–42: 2004. Available at www.cs.berkeley.edu/~kwright/nest_papers/15.4-wise04.pdf [Accessed: Nov. 25, 2018]

[11] *Security requirements for cryptographic modules*. Federal information proceedings standard publication 140-2, 1999. – 50 p.

Viktor Melnyk is a professor of the Department of Information Technology Security at Lviv Polytechnic National University, Ukraine. He was awarded with the academic Doctor of Philosophy degree in 2004, and Doctor of Technical Sciences in 2013 at Lviv Polytechnic National University. He has scientific, academic and hands-on experience in the field of computer systems research and design, proven contribution into IP Cores design methodology and high-performance reconfigurable computer systems design methodology. He is experienced in computer data protection, including cryptographic algorithms, cryptographic processors design and implementation, wireless sensor network security. Mr. Melnyk is an author of more than 90 scientific papers, patents and monographs.