



PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
FACULDADE DE ENGENHARIA / FACULDADE DE INFORMÁTICA
ENGENHARIA DE COMPUTAÇÃO



DESENVOLVIMENTO DE UMA INTERFACE DE COMUNICAÇÃO SEGURA UTILIZANDO O PADRÃO MACSEC

Felipe Bortolini Lazzarotto

Volume Final do Trabalho de Conclusão de Curso

Orientador: Prof. Dr. Fernando Gehm Moraes

Porto Alegre, 2016

Felipe Bortolini Lazzarotto

**DESENVOLVIMENTO DE UMA INTERFACE DE COMUNICAÇÃO
SEGURA UTILIZANDO O PADRÃO MACSEC**

Volume Final do Trabalho de Conclusão do
Curso de Engenharia de Computação na
Pontifícia Universidade Católica do Rio
Grande do Sul

Orientador: Prof. Dr. Fernando Gehm Moraes

Porto Alegre, 2016

ABSTRACT

Increasing the sensitive data traffic transmitted through the Internet increases the need to send information securely. To add security to the transmission of the information at the Layer 2 of the OSI model, IEEE published a standard known as MACsec, using encryption to transmit data.

The objective of this end-of-term work is to present the development of a secure communication interface compliant with the MACsec standard. The manuscript addresses the concepts related to the IEEE 802.1X-2010 and IEEE 802.1AE standards, describing the process to exchange the keys and modifications in the Ethernet frame required by the MACsec protocol.

After presenting the IEEE 802.1X-2010 and IEEE 802.1AE protocols, the manuscript details the infrastructure to create and validate a secure communication interface, in software, by emulating in computers the actors involved in the secure communication process: clients, authenticator, RADIUS server. Finally, a hardware module prototyped in FPGA generate packets according to the MACsec protocol, using GCM-AES (for cryptography) and MAC cores.

RESUMO

O aumento do tráfego de dados sensíveis aumenta a necessidade de transmitir informações de modo seguro. Essa necessidade fez com que o IEEE publicasse um padrão conhecido como MACsec, visando agregar segurança à transmissão de informações na camada 2 do modelo OSI, utilizando criptografia na transmissão dos dados.

O objetivo deste trabalho de conclusão de curso é apresentar o desenvolvimento de uma interface de comunicação segura em acordo com o padrão MACsec. Para isso o documento aborda os principais conceitos dos padrões IEEE 802.1X-2010 e IEEE 802.1AE, descrevendo como é realizado o acordo de chaves e modificações necessárias no formato dos pacotes para o padrão MACsec.

Uma vez apresentados os protocolos IEEE 802.1X-2010 e IEEE 802.1AE, detalha-se a infraestrutura necessária para a criação e a validação de uma interface de comunicação segura, em software, emulando-se em computadores os principais atores envolvidos no processo de comunicação segura: clientes, autenticador, servidor RADIUS. Finalmente, apresenta-se um módulo desenvolvido em FPGA, capaz de realizar a geração de pacotes de acordo com o padrão MACsec.

SUMÁRIO

1	INTRODUÇÃO.....	9
1.1	Motivação	9
1.2	Objetivos.....	10
1.3	Estrutura do Documento.....	10
2	PADRÃO IEEE 802.1X-2010	11
2.1	Sequência de mensagens e troca de chaves	11
2.1.1	Autenticação IEEE 802.1X e distribuição da chave MSK	11
2.1.2	Protocolo MACsec Key Agreement (MKA).....	13
2.1.3	Sessão segura.....	14
2.2	Hierarquia de chaves	14
2.2.1	CAK (Connectivity Association Key)	15
2.2.2	CKN (Connectivity Association Key Name).....	15
2.2.3	ICK (Integrity Check Key).....	15
2.2.4	KEK (Key Encrypting Key)	16
2.2.5	SAK (Secure Association Key).....	16
2.3	Arquitetura de controle de acesso à rede.....	16
2.4	Comunicação cliente-cliente.....	17
3	PADRÃO IEEE 802.1AE	19
3.1	Alterações necessárias no formato do <i>frame</i> Ethernet para o <i>frame</i> MACsec.....	19
3.2	Campo SecTAG	20
3.2.1	Campo TAG Control Information (TCI).....	21
3.3	Algoritmo GCM-AES.....	22
3.3.1	Entradas e Saídas do algoritmo GCM-AES.....	22
4	INFRAESTRUTURA PARA A INTERFACE DE COMUNICAÇÃO	24
4.1	Linux Kernel 4.6	25
4.2	Hostapd	25
4.3	WPA Supplicant.....	25
4.4	Servidor RADIUS	26
4.5	Autenticador.....	26
4.6	Cliente.....	27
4.7	Driver MACsec.....	27
5	VALIDAÇÃO DA INTERFACE DE COMUNICAÇÃO	29
5.1	Primeiro cenário – Autenticação com dados de acesso incorretos	29
5.2	Segundo cenário – Autenticação com dados de acesso corretos.....	31
5.3	Terceiro cenário – Validação do protocolo MKA.....	32
5.4	Quarto cenário - Envio de pacotes UDP protegidos	35
6	DESENVOLVIMENTO DE UM MÓDULO DE HARDWARE PARA GERAÇÃO DE PACOTES NO PADRÃO MACSEC	38
6.1	Validação do hardware desenvolvido.....	39
6.2	Testbench de validação do GCM-AES.....	42
6.3	Utilização de área	43
7	CONCLUSÃO E TRABALHOS FUTUROS	45
	REFERÊNCIAS	46
	ANEXO A	48

LISTA DE FIGURAS

FIGURA 1 - DIAGRAMA DE AUTENTICAÇÃO IEEE 802.1X-2010.....	12
FIGURA 2 - DIAGRAMA DA TROCA DE MENSAGENS DO PROTOCOLO MKA.....	13
FIGURA 3 - HIERARQUIA DE CHAVES [IEE10].....	14
FIGURA 4 - CONTROLE DE ACESSO À REDE COM MACSEC [IEE10].....	17
FIGURA 5 - COMUNICAÇÃO SEGURA ENTRE DOIS CLIENTES.	18
FIGURA 6 – ALTERAÇÕES ENTRE O <i>FRAME</i> ETHERNET E O <i>FRAME</i> MACSEC.	20
FIGURA 7 - FORMATO DO CAMPO SECTAG.....	21
FIGURA 8 - FORMATO DO CAMPO TCI.....	22
FIGURA 9 – AMBIENTE DE SIMULAÇÃO UTILIZADO.	24
FIGURA 10 – ENTIDADE SECY, RESPONSÁVEL POR REALIZAR A GERAÇÃO E RECEPÇÃO DE PACOTES NO FORMATO MACSEC.....	28
FIGURA 11 - TROCA DE MENSAGENS ENTRE O CLIENTE E O AUTENTICADOR COM DADOS DE ACESSO INVÁLIDOS.....	29
FIGURA 12 - TROCA DE MENSAGENS ENTRE O AUTENTICADOR E O SERVIDOR RADIUS COM DADOS DE ACESSO INVÁLIDOS.....	30
FIGURA 13 - TROCA DE MENSAGENS ENTRE O CLIENTE E O AUTENTICADOR NA ETAPA DE AUTENTICAÇÃO.....	31
FIGURA 14 - TROCA DE MENSAGENS ENTRE O AUTENTICADOR E O SERVIDOR RADIUS NA ETAPA DE AUTENTICAÇÃO.....	31
FIGURA 15 - CHAVES DERIVADAS PELO CLIENTE APÓS RECEBER A CHAVE MSK.	32
FIGURA 16 - MENSAGEM INICIAL DO PROTOCOLO MKA, ENVIADA PELO CLIENTE.	32
FIGURA 17 - MENSAGEM ENVIADA PELO AUTENTICADOR INFORMANDO SUA PRIORIDADE.....	33
FIGURA 18 - MENSAGEM ENVIADA PELO AUTENTICADOR AO CLIENTE CONTENDO A CHAVE SAK.	34
FIGURA 19 - CLIENTE INFORMA A INSTALAÇÃO DA SAK.	35
FIGURA 20 - CHAVE AES RECEBIDA PELO CLIENTE.....	35
FIGURA 21 - PACOTE UDP ENVIADO DO CLIENTE A.	36
FIGURA 22 - PACOTE MACSEC PROTEGIDO ENVIADO PELO CLIENTE A.....	36
FIGURA 23 - PACOTE MACSEC RECEBIDO PELO CLIENTE B.....	37
FIGURA 24 - PACOTE UDP RECEBIDO PELO CLIENTE B.....	37
FIGURA 25 - DIAGRAMA DOS PRINCIPAIS BLOCOS DA INTERFACE PARA ENVIO DE PACOTES MACSEC.	38
FIGURA 26 - INFRAESTRUTURA PARA SIMULAR O HARDWARE DESENVOLVIDO.	39
FIGURA 27 - PACOTES GERADOS PELA NETFPGA-SUME RECEBIDOS NO AUTENTICADOR.....	40
FIGURA 28 – TRÊS PACOTES MACSEC ENVIADOS VIA FPGA EM SEQUÊNCIA.....	41
FIGURA 29 - TRÊS PACOTES UDP APÓS SEREM VERIFICADOS.....	41
FIGURA 30 - SIMULAÇÃO DO MÓDULO GCM-AES.....	43
FIGURA 31 - POSICIONAMENTO DOS MÓDULOS NA FPGA XC7VX690T.	44
FIGURA 32 - <i>WIRESHARK</i> NO CLIENTE A, MENSAGENS ENVIADAS/RECEBIDAS.	48
FIGURA 33 - <i>WIRESHARK</i> NO AUTENTICADOR, MENSAGENS RECEBIDAS E ENVIADAS ENTRE AS INTERFACES.	49
FIGURA 34 - MENSAGENS RECEBIDAS/ENVIADAS PELO CLIENTE B.	49

LISTA DE TABELAS

TABELA 1 - NOTAÇÃO GCM-AES [RAN11].....	23
TABELA 2 - VETOR DE TESTE	43
TABELA 3 - RECURSOS UTILIZADOS DA NETFPGA.....	44

LISTA DE SIGLAS

AES	Advanced Encryption Standard
AN	Association Number
CA	Connectivity Association
CAK	Connectivity Association Key
CKN	Connectivity Association Key Name
CRC	Cyclic Redundancy Check
FCS	Frame Check Sequence
FPGA	Field Programmable Gate Array
GCM	Gaussian Counter Mode
ICK	Integrity Check Value Key
ICV	Integrity Check Value
IV	Initialization Vector
IEEE	Institute of Electrical and Electronics Engineers
KaY	MAC Security Key Agreement Entity
KEK	Key Encrypting Key
MAC	Media Access Control
MKA	MACsec Key Agreement
MSK	Master Secret Key
PN	Packet Number
RADIUS	Remote Authentication Dial-In User Service
SA	Secure Association
SAK	Secure Association Key
SC	Secure Channel
SCI	Security Channel Identifier
SecTAG	MAC Security TAG
SecY	MAC Security Entity
SL	Short Length
TCI	TAG Control Information

1 INTRODUÇÃO

A cada dia estamos mais conectados, interagindo com diversos dispositivos através da Internet. É a geração da “Internet das Coisas” (*Internet of Things* - IoT), onde cada vez mais dispositivos se tornam capazes de trocar informações. Junto ao aumento no número de dispositivos conectados, houve também o aumento do tráfego de dados sensíveis, como dados de cartão de crédito, informações pessoais, senhas, etc. Devido a isto, segurança, privacidade e confiabilidade se tornaram questões de grande importância em relação à transmissão de dados sensíveis na Internet.

Novos mecanismos de segurança são necessários para prevenir que estas informações sejam acessadas por pessoas não autorizadas. A abordagem original do padrão de comunicação Ethernet 802.3 [IEE12] não considera o problema de segurança nos dados transmitidos. O protocolo Ethernet é conhecido por ter vários problemas de segurança quando utilizado na sua forma original, como ataque de *replay*, *man-in-the-middle*, entre outros [IND12]. O uso do protocolo sem qualquer modificação, portanto, não é a melhor solução para troca de informações sensíveis.

Para resolver este problema, o IEEE (*Institute of Electrical and Electronics Engineers*) publicou em 2006 o padrão 802.1AE *MAC Security* (conhecido como MACsec). Este padrão encapsula os dados do *frame* Ethernet em um novo *frame* no qual os dados são protegidos em termos de confidencialidade, integridade e autenticidade utilizando o algoritmo de criptografia GCM-AES (do inglês, *Galois Counter Mode - Advanced Encryption Standard*) [IEE06]. Como o padrão 802.1AE não descreve como deve ser realizada a troca das chaves de criptografia, o IEEE publicou em 2010 o padrão 802.1X-2010, o qual descreve como as chaves de criptografia são trocadas entre os dispositivos, de forma a garantir confidencialidade das chaves [IEE10].

1.1 Motivação

Com o crescimento do armazenamento de informações na nuvem e da Internet das Coisas, garantir a confidencialidade e integridade dos dados sensíveis se torna muito importante. Para isto o desenvolvimento de novos dispositivos de segurança é necessário. Este trabalho de conclusão de curso permite aprofundar o estudo de alguns dos principais temas vistos durante o curso, como conceitos de redes, sistemas embarcados e prototipação.

Além disto, são abordados temas que são parcialmente abordados durante o curso, que é a área de segurança e criptografia de dados.

1.2 Objetivos

Este trabalho de conclusão tem por objetivo o desenvolvimento de uma interface de comunicação segura, empregando criptografia dos dados. Para isto, é necessário o desenvolvimento de um software para realizar a troca das chaves de criptografia, e integração com *drivers* que realizam a criação e envio de pacotes em interfaces cabeadas.

Além disto, é desenvolvido um módulo de *hardware* para envio de pacotes no padrão MACsec, utilizando um dispositivo FPGA Virtex 7. Para isto é utilizado o módulo XGE_MAC, já utilizado no escopo de um projeto com a empresa TERACOM e disponível no *opencores* [TAN08], inserindo o algoritmo de criptografia GCM-AES para realizar a criptografia de dados na camada MAC.

O projeto exige o estudo dos padrões IEEE 802.1AE e IEEE 802.1X-2010, estudo dos protocolos EAP, EAPoL, RADIUS e integração com *drivers* do Linux. Ao final do projeto, apresenta-se uma interface capaz de realizar a troca de informações com três propriedades da segurança de informação: confidencialidade, integridade e autenticidade. Confidencialidade, para que somente entidades autorizadas tenham acesso à informação. Integridade, propriedade que garante que os dados não foram alterados de forma não autorizada, desde sua criação e transmissão. Autenticidade, para garantir que a informação seja de uma entidade autenticada.

1.3 Estrutura do Documento

Este documento é organizado como segue. O Capítulo 2 descreve o padrão IEEE 802.1X-2010, o qual define os protocolos necessários e como deve ser realizada a troca de chaves de criptografia entre os dispositivos, de forma a prover autenticidade dos dados. O Capítulo 3 apresenta o padrão IEEE 802.1AE (MACsec), introduzindo as alterações necessárias no *frame* Ethernet para prover confidencialidade e integridade. O Capítulo 4 apresenta a infraestrutura utilizada para realização deste trabalho, descrevendo os softwares utilizados e o que foi implementado. O Capítulo 5 apresenta a validação da interface de comunicação desenvolvida, detalhando as mensagens trocadas entre os dispositivos e o envio de pacotes no padrão MACsec. O Capítulo 6 apresenta o *hardware* desenvolvido para envio de pacotes no padrão MACsec.

2 PADRÃO IEEE 802.1X-2010

Este Capítulo tem por objetivo apresentar o padrão *IEEE 802.1X-2010 Port-Based Network Access Control*. Este padrão especifica como realizar a autenticação dos clientes e define os protocolos para realizar a troca das chaves de criptografia utilizadas pelo padrão MACsec. Este padrão foi publicado em 2010, como uma revisão do padrão IEEE 802.1X-2004 [IEE10].

A arquitetura 802.1X especifica três entidades envolvidas na operação de autenticação:

- Cliente (também chamado de Suplicante): o usuário ou cliente que deseja acesso à rede.
- Servidor de autenticação: servidor de usuários, que contém informações dos usuários, tais como login, senha e permissões de acesso. Normalmente é utilizado um servidor RADIUS (do inglês, *Remote Authentication Dial-In User Service*).
- Autenticador: o dispositivo que intermedia a autenticação entre o cliente e o servidor de autenticação, coletando informações, validando-as e liberando acesso. Pode ser um ponto de acesso ou um *switch*.

A seguir é apresentado como é realizado o processo de autenticação e troca de chaves entre as entidades descritas acima.

2.1 Sequência de mensagens e troca de chaves

Uma sequência de troca de mensagens é necessária para estabelecer uma conexão autenticada até podermos utilizar a criptografia do padrão MACsec. Esta sequência pode ser dividida em três etapas: (i) autenticação IEEE 802.1X e distribuição da chave mestre MSK (do inglês, *Master Session Key*); (ii) protocolo MKA (*MACsec Key Agreement*); (iii) troca de dados utilizando o formato de *frame* MACsec. Abaixo descrevemos cada uma destas etapas.

2.1.1 Autenticação IEEE 802.1X e distribuição da chave MSK

O padrão IEEE 802.1X usa o protocolo EAP (do inglês, *Extensible Authentication Protocol*) para permitir o uso de uma variedade de mecanismos de autenticação. O protocolo EAP é baseado em um esquema de troca de mensagens utilizando quatro diferentes tipos de mensagens: *EAP Request* (pedido EAP), *EAP Response* (resposta EAP), *EAP Success* (sucesso EAP) e *EAP Failure* (falha EAP).

A Figura 1 apresenta as trocas de mensagens entre as entidades envolvidas na autenticação. A autenticação inicia com o Cliente enviando um pacote do tipo *EAPOL-Start*. Ao receber este pacote, o Autenticador solicita o identificador do Cliente que deseja ser autenticado, transmitindo um pacote *EAP Request-Identity* para o cliente. No momento que o cliente recebe este pacote, este responde com um pacote *EAP Response-Identity* contendo um identificador (nome do usuário). O autenticador ao receber a resposta do cliente, encapsula esta resposta em um pacote *RADIUS Access-Request* e a envia para o servidor de autenticação. O servidor de autenticação responde ao autenticador com um pacote *RADIUS Access-Challenge*, contendo uma mensagem *EAP Request* especificando o método EAP que o cliente deve utilizar. O autenticador envia ao cliente um pacote *EAP Request* informando o método EAP e aguarda uma resposta do cliente aceitando o método EAP solicitado ou negocia a utilização de outro método EAP. No momento em que o cliente e o servidor de autenticação chegarem a um acordo, o servidor de autenticação envia um pacote *RADIUS Access-Accept* (ou *RADIUS Access-Reject* se a autenticação falhar), transmitindo neste pacote a chave MSK. O Autenticador ao receber a mensagem envia ao Cliente uma mensagem do tipo *EAP-Success* (ou *EAP-Failure*, em caso de falha) e finaliza a etapa de autenticação.

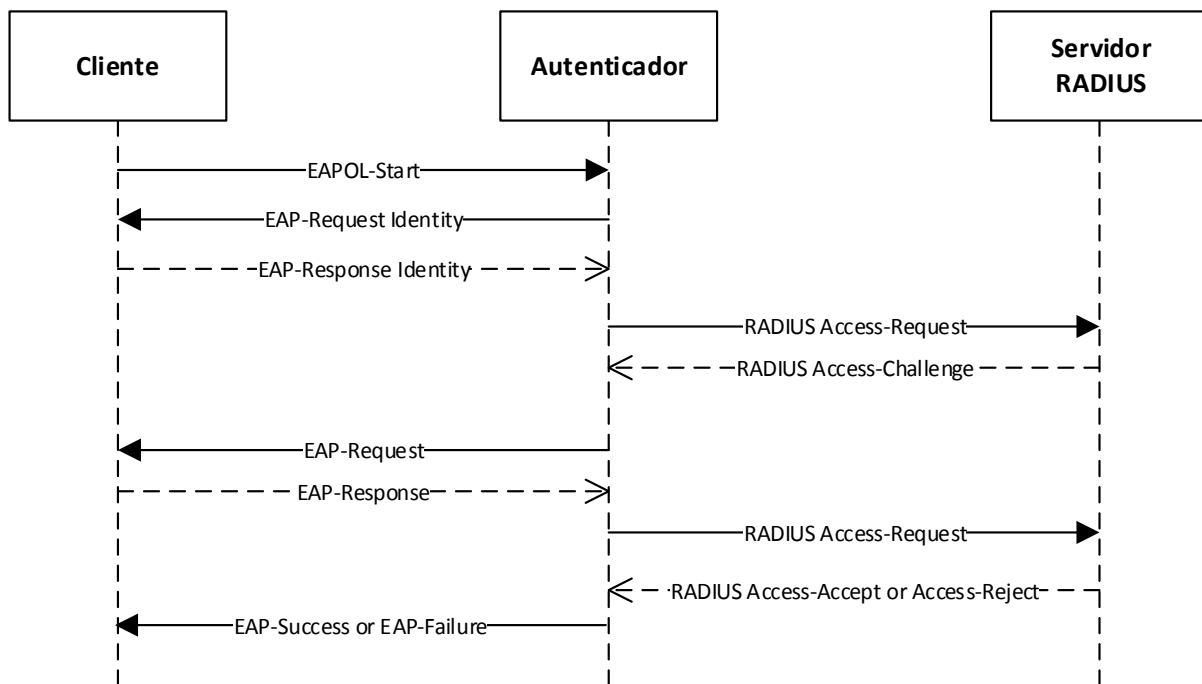


Figura 1 - Diagrama de autenticação IEEE 802.1X-2010.

2.1.2 Protocolo MACsec Key Agreement (MKA)

Durante esta etapa o cliente e autenticador irão eleger o servidor de chaves, responsável pela geração da chave SAK (do inglês, *Secure Association Key*) e derivar todos os parâmetros necessários para o protocolo MACsec, como o tamanho da chave de criptografia que será utilizada, identificador do canal seguro (SCI, do inglês *Secure Channel Identifier*), etc.

A eleição do servidor de chaves é baseada em prioridade, sendo o dispositivo com maior prioridade eleito o servidor de chaves. É definido que a maior prioridade será representada pelo menor valor numérico, por exemplo, se um dos dispositivos tiver uma prioridade 5 e outro dispositivo uma prioridade 1, o servidor de chaves será o dispositivo com prioridade 1. Essa eleição e demais parâmetros são definidos através da troca de mensagens do tipo *EAPOL-MKA*, como apresentado na Figura 2. Nesta Figura, o autenticador foi eleito o servidor de chaves, sendo responsável por gerar a chave SAK, que será derivada da chave CAK (*Connectivity Association Key*), que por sua vez é derivada da chave MSK, recebida na etapa anterior (na Subseção 2.2 veremos em detalhes todas as chaves envolvidas e como elas são derivadas).

Para que os dados possam ser criptografados e descriptografados com sucesso, o Cliente e o Autenticador devem utilizar a mesma chave SAK. Esta chave é enviada pelo Autenticador ao Cliente. Para que esta chave não fique exposta ao ser trocada, a chave é criptografada utilizando a chave KEK (*Key Encrypting Key*) e a função *AES Key Wrap*, descrita em [SCH02]. Após descriptografar a mensagem e receber a chave SAK, o Cliente envia uma mensagem informando que a chave foi instalada.

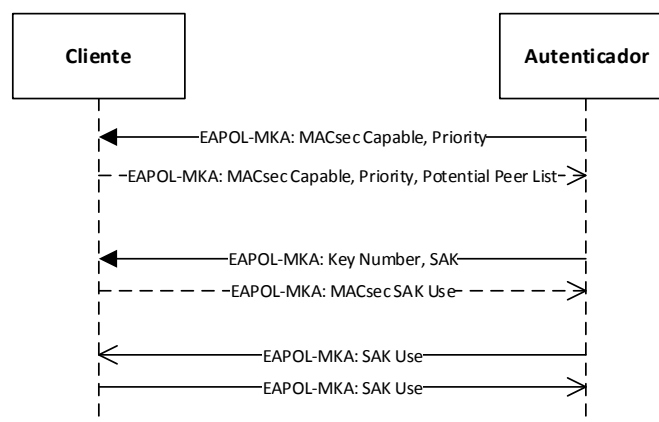


Figura 2 - Diagrama da troca de mensagens do protocolo MKA.

2.1.3 Sessão segura

Nesta etapa o cliente e autenticador já possuem a mesma chave SAK e começam a transmitir e receber dados utilizando criptografia e o formato do *frame* MACsec, que será apresentado no próximo Capítulo.

2.2 Hierarquia de chaves

Cada chave utilizada pelo protocolo MKA é derivada da MSK utilizando a função de derivação KDF (*Key Derivation Function*) em *Counter Mode*, descrita em [CHE09]. A partir da MSK, distribuída pelo servidor RADIUS durante a etapa de autenticação, é derivada a CAK, a qual é utilizada para derivação de todas as outras chaves associadas. Tanto o Cliente como o Autenticador derivam as mesmas chaves, com exceção da SAK que é enviada pelo servidor de chaves eleito.

A CAK é identificada pela CKN (também conhecida como *CAK Name*). A CKN é derivada a partir da MSK. A CKN é enviada nos pacotes do protocolo MKA para identificar a CAK que esta sendo utilizada. A partir da CAK, são derivadas as chaves ICK (*Integrity Check Key*) e KEK (*Key Encrypting Key*). A ICK é utilizada para gerar e verificar o campo ICV dos pacotes do protocolo MKA. A KEK é utilizada para transmitir a SAK de forma segura para o outro dispositivo.

A SAK, que será utilizada para criptografar os dados na comunicação segura, pode ser derivada a partir da CAK ou pode ser utilizado um gerador de chaves randômico. Esta hierarquia está descrita na Figura 3, e cada transformação realizada para derivação é descrita à seguir. Uma nova SAK é gerada após serem transmitidos 0xC0000000 pacotes utilizando esta chave.

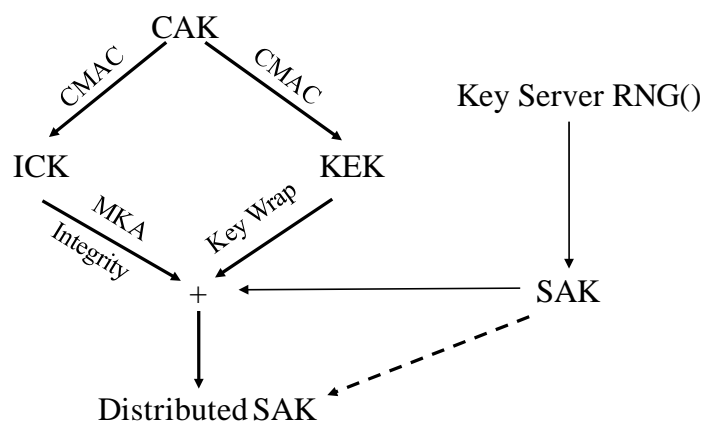


Figura 3 - Hierarquia de chaves [IEE10].

2.2.1 CAK (Connectivity Association Key)

A chave CAK é derivada da chave MSK utilizando a seguinte transformação:

$$\text{CAK} = \text{KDF}(\text{Key}, \text{Label}, \text{Mac1} \mid \text{Mac2}, \text{CAKLength})$$

Onde:

- Key = MSK[0-15] para uma chave CAK de 128 bits ou MSK[0-31] para uma chave de 256 bits.
- Label = "IEEE8021 EAP CAK"
- Mac1 = o menor dos dois endereços MAC utilizados no pacote EAP.
- Mac2 = o maior dos dois endereços MAC utilizados no pacote EAP.
- CAKLength = dois octetos representando o tamanho da chave que se deseja (128 para uma chave de 128 bits e 256 para uma chave de 256 bits).

2.2.2 CKN (Connectivity Association Key Name)

A chave CKN é derivada da chave MSK utilizando a seguinte transformação:

$$\text{CKN} = \text{KDF}(\text{Key}, \text{Label}, \text{ID} \mid \text{Mac1} \mid \text{Mac2}, \text{CKNlength})$$

Onde:

- Key = MSK[0-15] para um nome CKN de 128 bits ou MSK[0-31] para um nome de 256 bits.
- Label = "IEEE8021 EAP CKN"
- ID = EAP-Session-ID
- Mac1 = o menor dos dois endereços MAC utilizados no pacote EAP.
- Mac2 = o maior dos dois endereços MAC utilizados no pacote EAP.
- CKNLength = dois octetos representando o tamanho da chave que se deseja (128 para uma chave de 128 bits e 256 para uma chave de 256 bits).

2.2.3 ICK (Integrity Check Key)

A chave ICK é derivada da chave CAK utilizando a seguinte transformação:

$$\text{ICK} = \text{KDF}(\text{Key}, \text{Label}, \text{KeyID}, \text{ICKLength})$$

Onde:

- Key = CAK
- Label = "IEEE8021 ICK"
- KeyID = os 16 primeiros octetos do CKN, se necessário completados com zeros.

- ICKLength = dois octetos representando o tamanho da chave que se deseja (128 para uma chave de 128 bits e 256 para uma chave de 256 bits).

2.2.4 KEK (Key Encrypting Key)

A chave KEK é derivada da chave CAK utilizando a seguinte transformação:

$$\text{KEK} = \text{KDF}(\text{Key}, \text{Label}, \text{KeyID}, \text{KEKLength})$$

Onde:

- Key = CAK
- Label = "IEEE8021 KEK"
- KeyID = os 16 primeiros octetos do CKN, se necessário completados com zeros.
- KEKLength = dois octetos representando o tamanho da chave que se deseja (128 para uma chave de 128 bits e 256 para uma chave de 256 bits).

2.2.5 SAK (Secure Association Key)

A chave SAK é derivada da chave CAK utilizando a seguinte transformação:

$$\text{SAK} = \text{KDF}(\text{Key}, \text{Label}, \text{KS-nonce} \mid \text{MI-value list} \mid \text{KN}, \text{SAKLength})$$

Onde:

- Key = CAK
- Label = "IEEE8021 SAK"
- KS-nonce = um número aleatório com o mesmo tamanho da chave SAK.
- MI-value list = lista de identificação dos membros de uma CA concatenados.
- KN = o número da chave em quatro octetos.
- SAKLength = dois octetos representando o tamanho da chave que se deseja (128 para uma chave de 128 bits e 256 para uma chave de 256 bits).

2.3 Arquitetura de controle de acesso à rede

O padrão 802.1X define duas entidades de porta lógica para uma porta autenticada: a porta controlada e a porta não controlada. A porta controlada é manipulada pelo PAE (do inglês, *Port Access Entity*) para permitir ou evitar o recebimento/envio de pacotes. O PAE utiliza a porta não controlada para transmitir e receber pacotes do tipo EAPOL, visando a autenticação. O PAE é responsável por toda a parte de autenticação e protocolo MKA,

informando a SecY (do inglês, *Security Entity*) a criação de uma associação segura e a chave SAK que deve ser utilizada para geração e recebimento de pacotes protegidos.

A Figura 4 ilustra a arquitetura de controle de acesso à rede. O dispositivo da direita representa o Autenticador, realizando a comunicação com um Servidor RADIUS externo. O dispositivo da esquerda representa o Cliente desejando ser autenticado. Inicialmente o tráfego é recebido pela porta não controlada (identificado na Figura 4 com o número 1), sendo permitido somente tráfego 802.1X. Após a autenticação é permitido o recebimento de qualquer tráfego, sendo o tráfego seguro controlado pela entidade SecY (2 na Figura). A SecY compreende a parte de formatação do pacote MACsec e o módulo GCM-AES para criptografia e descryptografia (3). A SecY será detalhada na Subseção 4.4 deste trabalho.

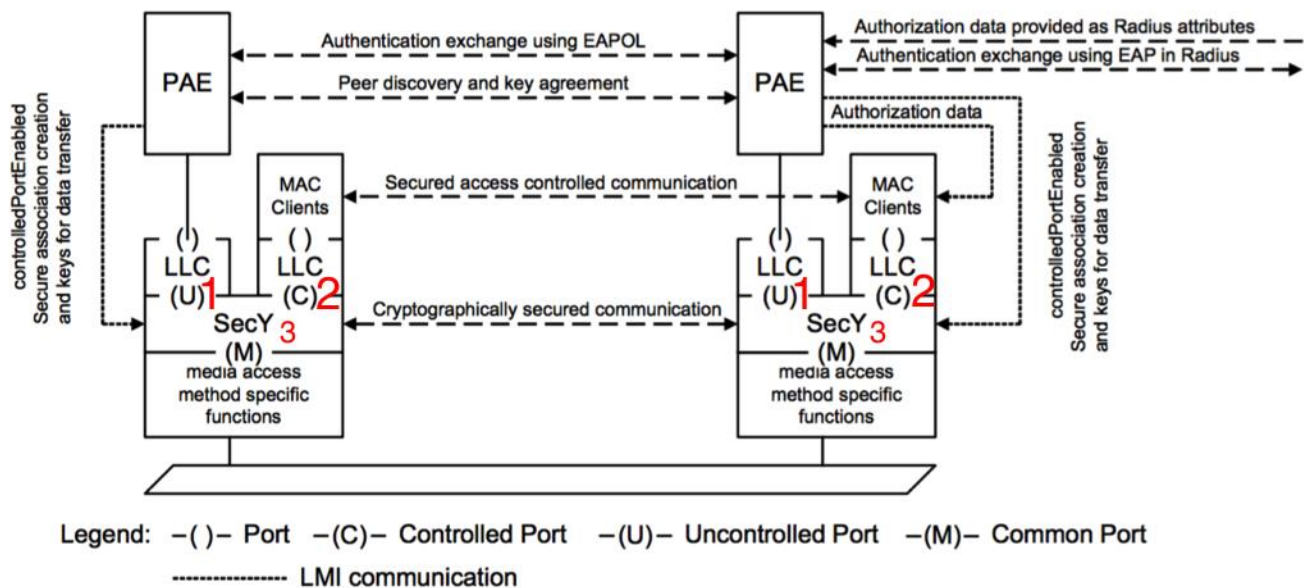


Figura 4 - Controle de acesso à rede com MACsec [IEE10].

2.4 Comunicação cliente-cliente

O padrão MACsec especifica uma comunicação ponto a ponto, protegendo a comunicação entre dispositivos confiáveis. A Figura 5 ilustra como é realizada a comunicação entre dois dispositivos clientes. Inicialmente, a mensagem é criptografada no Cliente A, utilizando a chave SAK trocada entre o Cliente A e o Autenticador (Key A), e após é enviada para o Autenticador (1 na Figura). Chegando no Autenticador, a mensagem é descryptografada e verificada a integridade do dado (2 na Figura). Ainda no Autenticador, a mensagem é criptografada utilizando a chave SAK trocada entre o Autenticador e o Cliente B (Key B) e

enviada para o Cliente B (3 na Figura), finalizando a comunicação entre os dois clientes (4 na Figura).

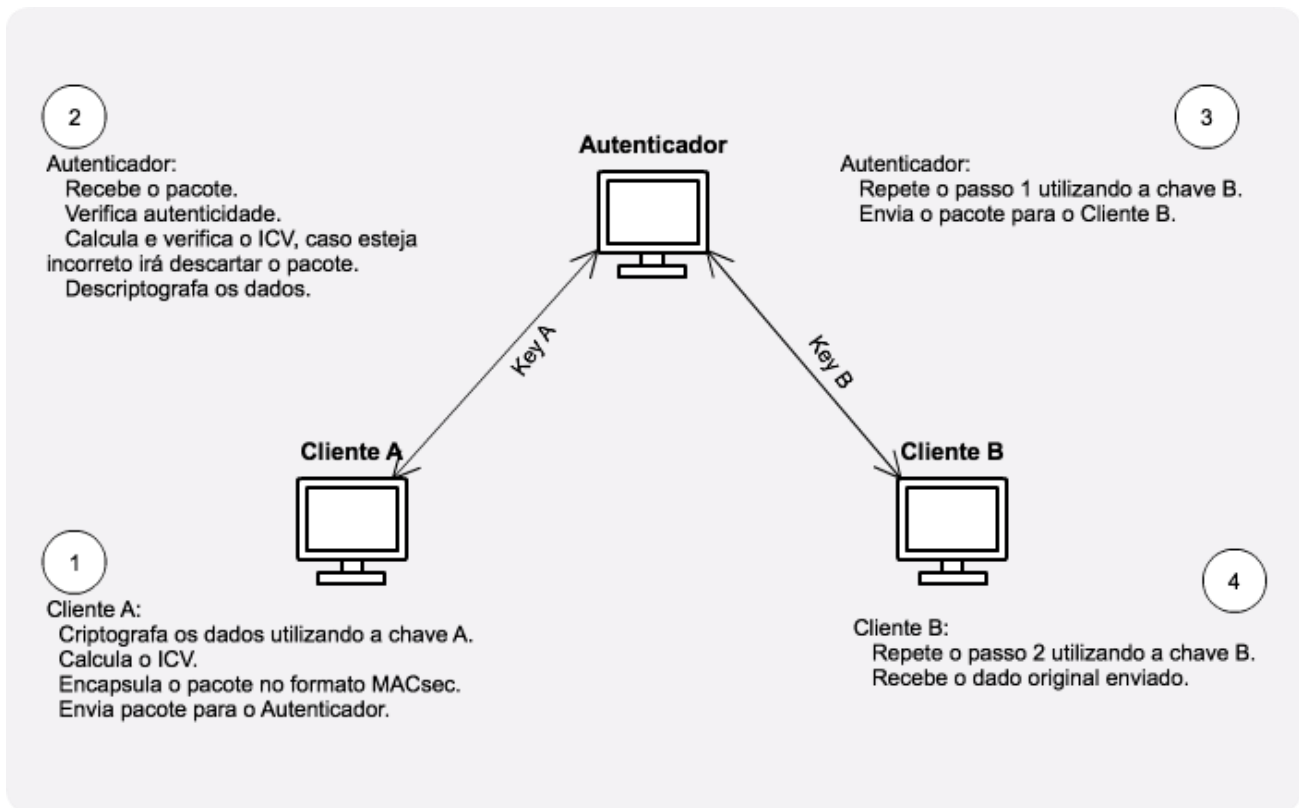


Figura 5 - Comunicação segura entre dois clientes.

Analisando esta comunicação, seria mais eficiente que a mensagem não fosse descriptografada e criptografada novamente no autenticador, realizando-se uma conexão do tipo *end station-to-end station*. Entretanto, o padrão não esclarece quais protocolos seriam utilizados para que uma das estações finais descobrisse a outra e de que forma seria realizada a configuração de uma conexão segura entre os dois clientes [SEA13].

3 PADRÃO IEEE 802.1AE

Este Capítulo tem por objetivo apresentar o padrão IEEE 802.1AE MAC Security, conhecido como MACsec. Este padrão especifica os requisitos para confidencialidade, integridade e autenticidade dos dados para entidades que operam de forma transparente à camada MAC. O padrão foi publicado originalmente em 2006, recebendo alterações nas emendas IEEE 802.1AEbn e IEEE 802.1AEbw em 2011 e 2013, respectivamente [CAR15].

O padrão MACsec detalha como é realizado o encapsulamento do *frame* Ethernet desprotegido em um novo *frame* onde os dados estão protegidos. Para isto é utilizado o algoritmo de criptografia GCM-AES, utilizando uma chave de criptografia de 128 ou 256 bits. O padrão não descreve como são trocadas as chaves de criptografia entre as entidades, processo este definido pelo padrão IEEE 802.1X-2010, o qual foi descrito no Capítulo 2 deste trabalho.

3.1 Alterações necessárias no formato do *frame* Ethernet para o *frame* MACsec

O pacote Ethernet contém, além dos dados úteis, os endereços de origem e destino do pacote, para permitir a comunicação ponto a ponto dos dispositivos conectados à uma mesma rede. O pacote é formatado em campos de octetos determinados pelo protocolo Ethernet para permitir a decodificação de cada informação dentro do pacote. O tamanho total do *frame* deve conter um número inteiro de octetos. Os principais campos são (parte superior da Figura 6):

- **Destination/Source Addresses:** endereços MAC de 6 octetos, utilizados na identificação da origem e destino do pacote recebido.
- **EtherType:** este campo de 2 octetos é utilizado para dois propósitos distintos. O primeiro é informar o número de bytes do pacote, quando este for inferior a 1500 octetos. Valores maiores que 1536 definem o tipo de protocolo ao qual pertence o pacote (*EtherType*). Como exemplo, o protocolo IPV4 utiliza o *EtherType* 0x0800 (valor em hexadecimal). Valores entre 1501 e 1535 não são utilizados.
- **Data:** são os dados úteis, este campo pode variar entre 46 a 1500 octetos.
- **Frame Check Sequence (FCS):** esse campo consiste de 4 octetos que armazenam o valor de CRC (*Cyclic Redundancy Check*), calculado através dos dados do pacote e utilizado para verificação da integridade do mesmo no receptor.

O padrão MACsec estabelece algumas alterações na formatação do pacote Ethernet. Essas alterações visam garantir a integridade, confidencialidade e autenticidade. As alterações estão em destaque na Figura 6 e são a inclusão dos seguintes campos:

- **SecTAG:** este campo pode ter 8 ou 16 octetos e compreende um conjunto de informações relativas ao protocolo MACsec, este campo será detalhado na Subseção 3.2.
- **Secure Data:** são os campos *EtherType* e *Data* do pacote Ethernet criptografados, este campo não pode possuir tamanho nulo.
- **Integrity Check Value (ICV):** este campo possui 16 octetos e é gerado pelo algoritmo GCM-AES. Esse campo tem por objetivo garantir a integridade do pacote, permitindo verificar no recebimento se o pacote sofreu alguma modificação.

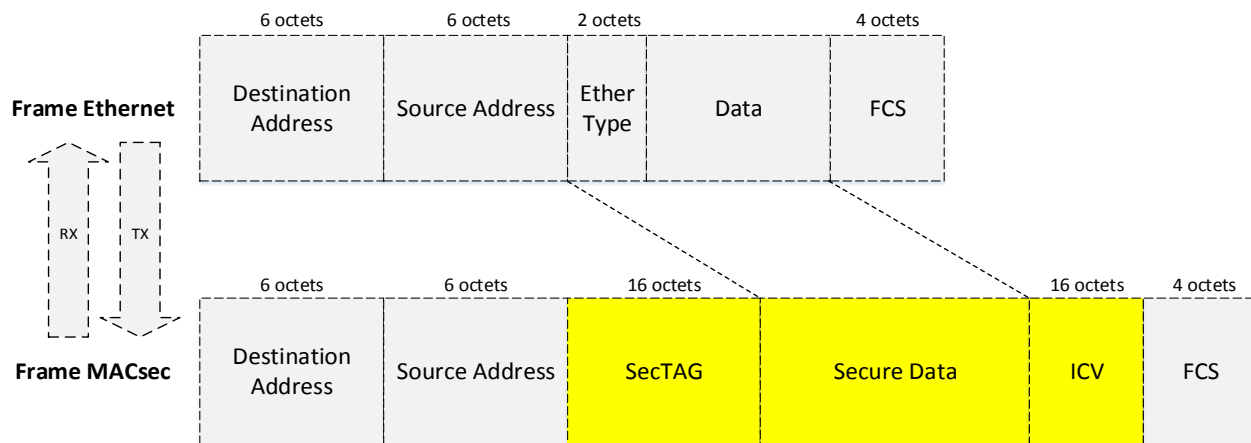


Figura 6 – Alterações entre o *frame* Ethernet e o *frame* MACsec.

3.2 Campo SecTAG

O campo SecTAG é identificado pelo MACsec Ethertype. O formato do campo SecTAG é ilustrado na Figura 7 e compreende os seguintes campos:

- **MACsec EtherType:** este campo possui 2 octetos e identifica que o pacote é do tipo MACsec, por padrão o valor deste campo é 0x88E5.
- **TAG Control Information (TCI):** este campo compreende o terceiro octeto do campo SecTAG, este campo é detalhado na Subseção 3.2.1.
- **Short Length (SL):** este campo compreende o quarto octeto, e identifica o número de octetos presentes no campo *Secure Data*, quando o número de octetos do campo for

menor que 48. Caso o tamanho do campo *Secure Data* seja maior que 48 octetos, este campo deverá ser zero. Os bits 7 e 8 são sempre zero.

- **Packet Number (PN):** este campo possui 4 octetos e contém um número único para identificar cada pacote transmitido utilizando o mesmo SA (do inglês, *Secure Association*). Desta forma obtém-se proteção contra mensagens repetidas.
- **Secure Channel Identifier (SCI):** este campo é opcional e possui 8 octetos. Se o bit SC (do inglês, *Secure Channel*) do campo TCI for igual a 1, este campo será utilizado. Os primeiros 6 octetos são compostos do endereço MAC associado ao SecY. Os dois últimos octetos são a identificação da porta utilizada [IEE06].

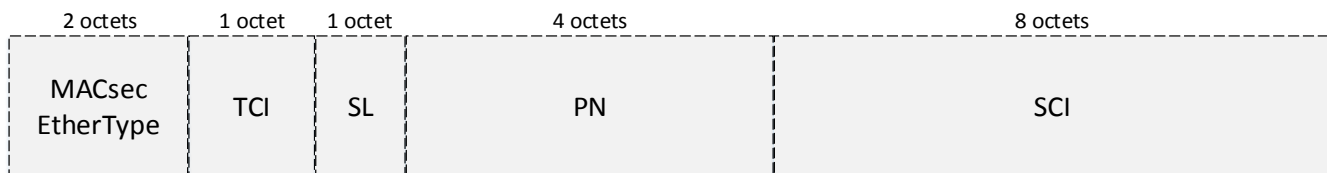


Figura 7 - Formato do campo SecTAG.

3.2.1 Campo TAG Control Information (TCI)

O campo TCI compreende o terceiro octeto do campo SecTAG. Seu formato é detalhado na Figura 8, e é composto por:

- **Version (V):** é o número da versão, como ainda não existe outra versão, este bit deve ser zero.
- **End Station (ES):** este bit deve ser setado caso os 6 primeiros octetos do campo SCI sejam iguais ao endereço MAC de origem. Caso seja setado, o bit 6 (SC bit) deverá ser zero e não deve ser utilizado o campo SCI. Este bit será zero caso o endereço de origem não seja utilizado para determinar o SCI.
- **Secure Channel (SC):** deve ser setado caso seja utilizado o campo SCI.
- **Single Copy Broadcast (SCB):** este bit deve ser setado caso seja suportado EPON Single Copy Broadcast.
- **Encryption (E):** este bit deve ser setado caso seja utilizada criptografia no campo *Secure Data*.

- **Changed Text (C):** este bit deve ser setado caso o dado original tenha sido alterado. Este bit só deverá ser zero, caso o campo *Secure Data* seja exatamente igual ao dado original.
- **Association Number (AN):** este campo compreende os bits 1 e 2 do terceiro octeto e é utilizado para identificar a SA.

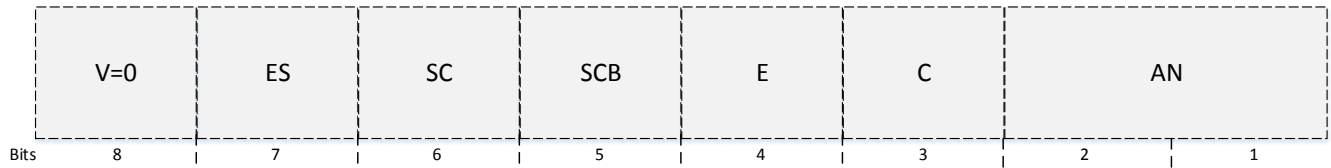


Figura 8 - Formato do campo TCI.

3.3 Algoritmo GCM-AES

O padrão IEEE 802.1AE define a utilização do algoritmo GCM-AES para fornecer autenticação e confidencialidade de dados. A autenticidade dos dados é fornecida pelo GCM, utilizando uma função *hash* universal [MOZ12]. A confidencialidade dos dados é fornecida pelo algoritmo AES. O algoritmo AES foi aceito pelo *National Institute of Standards and Technology* (NIST) em 2001, em substituição aos padrões de criptografia anteriores. Desde então tem sido utilizado em diversas aplicações, como nos padrões de redes sem fio WiMAX e Wi-Fi, segurança de *smart cards*, mecanismos de segurança do arquivo *bitstream* em FPGAs.

O GCM-AES é o algoritmo de criptografia padrão utilizado pelo MACsec. Originalmente, o padrão previa somente a utilização da chave de criptografia de 128 bits, sendo alterado em 2011, na emenda IEEE 802.1AEbn, possibilitando a utilização de chaves de 128 bits ou 256 bits. O padrão MACsec especifica as entradas e saídas do algoritmo GCM-AES, como veremos a seguir.

3.3.1 Entradas e Saídas do algoritmo GCM-AES

O algoritmo GCM-AES possui duas operações, a primeira se refere a criptografia autenticada de dados e a segunda a descryptografia autenticada. A Tabela 1 apresenta um resumo das notações utilizadas pelo algoritmo GCM-AES.

A operação de criptografia possui quatro entradas:

- Uma chave secreta K , que é a chave SAK (do inglês, *Secure Association Key*) de 128 bits ou 256 bits.

- Um vetor de inicialização *IV* de 96 bits, que são os seguintes campos concatenados: campos *SCI* e *PN* da *SecTAG*.
- Um texto não codificado *P*, que é o dado que será criptografado. São os campos *Ethertype* e *User Data* do *frame* Ethernet, descrito na Figura 6.
- Um dado adicional de autenticação *A*, que são os seguintes campos concatenados: *MAC Destination Address*, *MAC Source Address* e *SecTAG*.

Tabela 1 - Notação GCM-AES [RAN11].

K	The AES key (either 128-bit or 256-bit)
IV	initial value used by GCM
A	additional authenticated data
P	plaintext (user data)
C	encrypted data
T	integrity check value (ICV)

A operação de criptografia gera duas saídas:

- Um texto criptografado *C*, que é o campo *Secure Data* do *frame* MACsec, possuindo o mesmo tamanho do texto não codificado *P*.
- Uma *tag* de autenticação *T*, que possui 128 bits e é o campo *ICV* do *frame* MACsec.

A operação de descriptografia é a função inversa e possui cinco entradas: a chave SAK, o vetor de inicialização *IV*, o texto criptografado *C*, o dado adicional de autenticação *A* e a tag de autenticação *T*, e possui uma única saída, que é o texto não codificado *P* ou um símbolo informando que as entradas não são autênticas [IEE06][MCG04].

4 INFRAESTRUTURA PARA A INTERFACE DE COMUNICAÇÃO

Os Capítulos anteriores corresponderam ao estudo necessário para a execução do presente trabalho. Os próximos Capítulos apresentam a principal contribuição deste TCC, que é o desenvolvimento de uma interface de comunicação segura.

Este Capítulo apresenta os softwares e dispositivos necessários para a criação do ambiente de simulação. Foram utilizados 4 computadores para a realização deste trabalho, cada um representa um dos dispositivos ilustrados na Figura 9. O Autenticador é um computador com três placas de redes instaladas para simular a função de um *switch*. O Autenticador executa o *software hostapd*. Os dispositivos Cliente executam o *software wpa_supplicant* e no Servidor RADIUS foi instalado o *software FreeRADIUS*. Foi utilizada a versão 4.6 do Kernel Linux nos dispositivos Clientes e Autenticador para utilização do *driver* MACsec. A seguir serão detalhados os softwares e recursos utilizados.

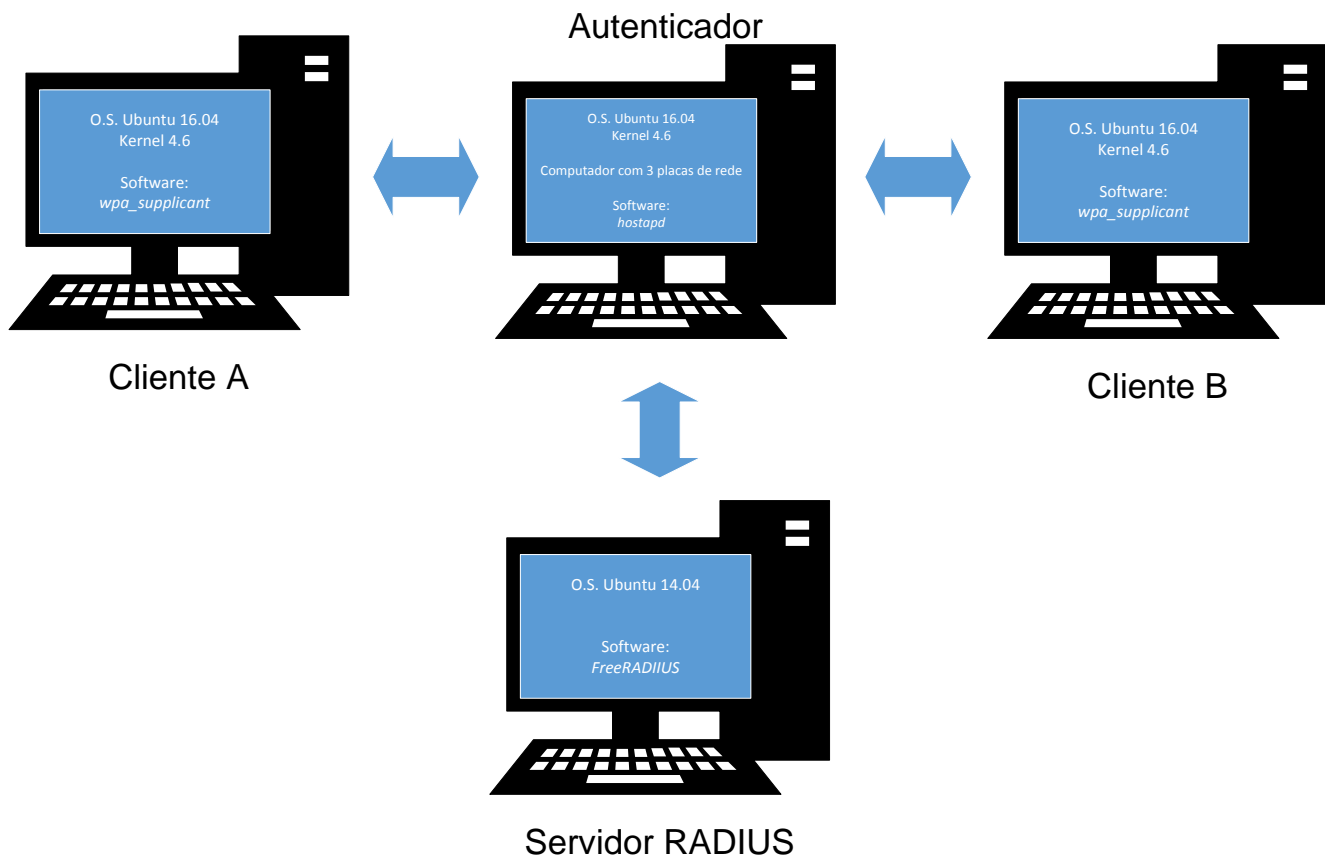


Figura 9 – Ambiente de simulação utilizado.

4.1 Linux Kernel 4.6

Nos computadores Cliente e no computador Autenticador, foi instalado o sistema operacional Ubuntu 16.04 e utilizada a versão 4.6 do Kernel. Essa versão do Kernel foi lançada em 15 de maio de 2016 e entre os novos recursos está o suporte ao MACsec, possuindo um *driver* capaz de gerar e receber pacotes no padrão MACsec [LIN16].

4.2 Hostapd

O *hostapd* é um daemon *open source* para pontos de acesso e servidores de autenticação. Este programa implementa o padrão IEEE 802.11 (padrão para redes sem fio), autenticação IEEE802.1X, WPA, WPA2 e EAP, cliente RADIUS, entre outros recursos. Este programa é *open source* e escrito na linguagem C. O *hostapd* possui suporte a diversos métodos EAP, como EAP-TLS, EAP-PEAP, EAP-TTLS/MSCHAPv2, EAP-SIM [HOS16]. O padrão MACsec define que pode ser utilizado qualquer método EAP que realize a transmissão de uma chave MSK. Neste trabalho é utilizado o método EAP-TTLS/MSCHAPv2 por este ser um dos métodos suportados.

Primeiramente, foi utilizado o *hostapd* com o objetivo de realizar a autenticação dos clientes com o Servidor RADIUS externo. Após validada a autenticação dos usuários, o programa foi modificado para possuir suporte ao protocolo MKA, realizar a troca das demais chaves necessárias, e realizar a interface com o *driver* MACsec do Kernel.

4.3 WPA Supplicant

O *wpa_supplicant* é um daemon *open source* usado em estações do tipo Cliente. Ele implementa negociação de chaves com o Autenticador. Assim como o *hostapd*, o *wpa_supplicant* possui suporte a diversos métodos EAP [WPA16].

Neste trabalho utilizamos o programa *wpa_supplicant* para realizar a autenticação do usuário com o servidor RADIUS e para realizar todo o protocolo MKA, trocando as chaves necessárias com o Autenticador. Após validar a troca correta das chaves e recebimento da chave SAK, foi necessário alterar o programa para que ele se comunicasse com o driver MACsec do Kernel, realizando a troca de informações de forma segura.

4.4 Servidor RADIUS

RADIUS é um protocolo de rede responsável por definir regras para a comunicação entre dispositivos de rede. O protocolo RADIUS possui três funções: autenticar usuários ou dispositivos antes que estes tenham acesso à rede, autorizar os usuários a utilizar recursos e serviços específicos da rede e contabilizar e monitorar a utilização destes serviços. Neste trabalho, foi utilizado o FreeRADIUS, que é um servidor RADIUS *open source* [FRE16]. O FreeRADIUS foi instalado em um computador com o sistema operacional Ubuntu 14.04.

Entre as configurações necessárias para utilização do servidor, foram necessários adicionar os usuários e senhas dos clientes que irão se conectar, definir o endereço IP e senha de acesso do Autenticador e definir o método de autenticação EAP permitido para a conexão. Neste trabalho foi definido o método EAP-TTLS/MSCHAPv2.

4.5 Autenticador

O Autenticador realiza a comunicação entre os dispositivos Cliente e o servidor RADIUS na etapa de autenticação e após é o servidor de chaves para o protocolo MACsec Key Agreement. Em um ambiente real, o Autenticador seria um *switch* com suporte à autenticação IEEE 802.1X e suporte ao protocolo MACsec.

Neste trabalho, como não tínhamos disponível um *switch* com suporte ao protocolo MACsec, substituímos o *switch* por um computador com três placas de rede. Neste computador, foi utilizado o software *hostapd* para realizar a etapa de autenticação [HOS16]. Após, foi necessário adicionar o protocolo MKA para realizar a troca de chaves com os dispositivos Cliente. Além disto, foram necessárias algumas alterações para que as chaves derivadas fossem criadas corretamente.

Após ter realizado a troca de todas as chaves corretamente, foi criada uma interface com o *driver* MACsec do Kernel do Linux. A Subseção 4.4 detalhada esta interface.

Para simular a função de um *switch*, foi desenvolvido um programa escrito em linguagem C que tem por função reenviar os pacotes recebidos através de uma interface para outra interface. Por exemplo, um pacote recebido através da interface `eth0` é reenviado para a interface `eth1`.

4.6 Cliente

Os dispositivos do tipo Cliente são computadores que executam o software *wpa_supplicant* para realizar a autenticação e troca de chaves do protocolo MKA. As alterações necessárias no software foram a criação uma interface com o *driver* MACsec do Kernel do Linux.

Para verificar o envio e recebimento de pacotes protegidos, foi desenvolvido um programa escrito em linguagem C que envia pacotes do tipo UDP para um determinado endereço MAC.

4.7 Driver MACsec

Para realizar a criação dos pacotes no formato MACsec, como visto no Capítulo 3 deste trabalho, foi utilizado o driver MACsec presente na versão 4.6 do Kernel do Linux. Este driver é responsável por implementar a SecY, realizando a verificação dos pacotes recebidos e geração dos pacotes seguros. Na Figura 10 está ilustrada a SecY.

A interface segura funciona da seguinte forma. Primeiramente, é criada uma nova interface de rede (na Figura 10 ilustrado com o nome `macsec_eth0`). Através desta interface são recebidos e enviados os pacotes. Todos os pacotes recebidos e transmitidos por essa interface estarão protegidos.

Através da interface `macsec_eth0`, os pacotes são enviados e recebidos no formato do padrão Ethernet. O *driver* MACsec, compreende a SecY (identificado na Figura 10 pelo retângulo vermelho), ele realiza a formatação do pacote (1) no formato MACsec (*Secure Frame Generation*), utilizando o Cipher GCM-AES-128 (2) para realizar a criptografia e geração do ICV e após enviará o pacote através da porta comum.

Ao receber um pacote pela porta comum, se o pacote estiver no padrão MACsec, o *driver* realiza a verificação e descriptografa o pacote (3) (*Secure Frame Verification*), disponibilizando o pacote na interface `macsec_eth0` no formato Ethernet. Os pacotes protegidos no padrão MACsec podem ser vistos através da interface `eth0`. Estes pacotes estarão formatados conforme o padrão MACsec.

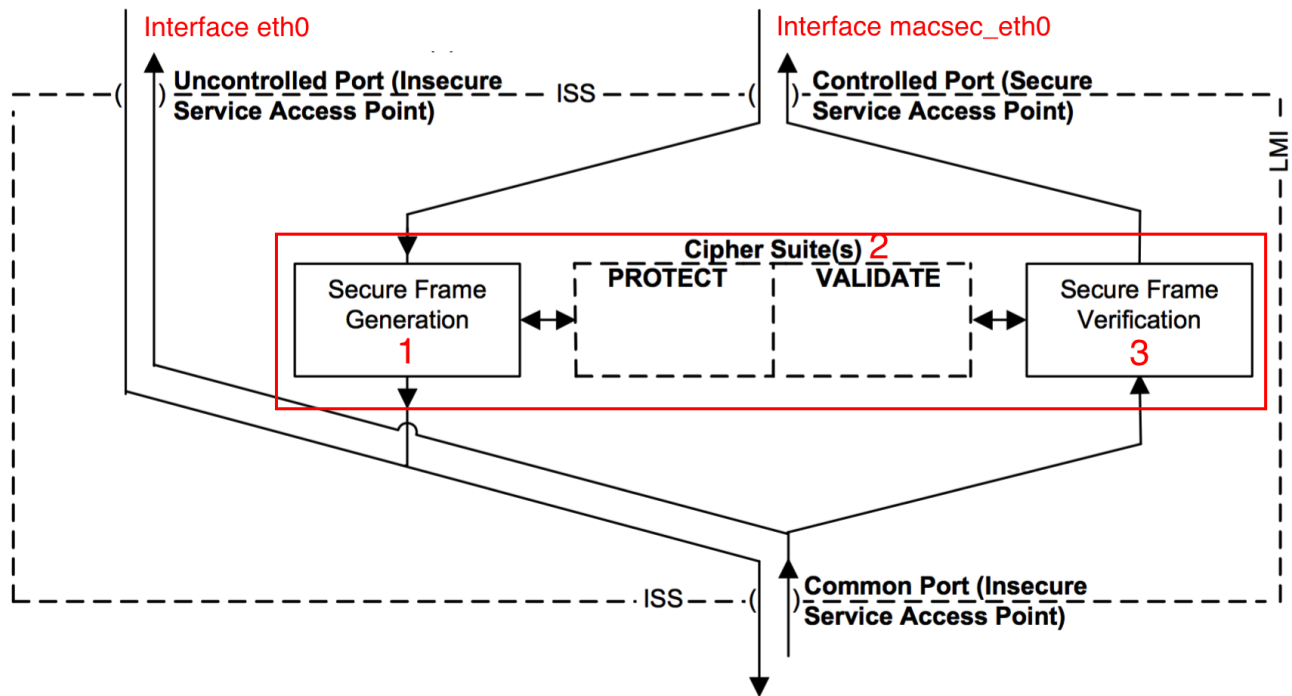


Figura 10 – Entidade SecY, responsável por realizar a geração e recepção de pacotes no formato MACsec.

5 VALIDAÇÃO DA INTERFACE DE COMUNICAÇÃO

Para validar o ambiente de testes, descrito na Figura 9, foi utilizado o software de monitoramento de redes *Wireshark*, juntamente com mensagens de *debug* nos programas *hostapd* e *wpa_supplicant* para identificar as chaves derivadas no Cliente e no Autenticador. A seguir são descritos os cenários criados para validar o correto funcionamento da interface.

5.1 Primeiro cenário – Autenticação com dados de acesso incorretos

O primeiro cenário consiste na validação da autenticação dos usuários. Para validar que somente usuários e senhas válidos estão sendo autenticados, foi realizada uma tentativa de autenticação onde a senha do usuário estava incorreta. Ao tentar se conectar com dados inválidos, espera-se que a conexão seja rejeitada, finalizando com uma mensagem *EAP-Failure*. Na Figura 11 visualizamos as mensagens trocadas entre o Cliente e o Autenticador, onde a conexão é rejeitada e o Cliente não tem acesso a rede, essas mensagens são as mensagens descritas na Figura 1.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	HewlettP_df:d1:64	Nearest	EAPOL	1 60	Start
2	0.012797706	D-LinkCo_90:bd:d8	Nearest	EAP	2 64	Request, Identity
3	0.013018592	HewlettP_df:d1:64	Nearest	EAP	3 60	Response, Identity
4	0.014625218	D-LinkCo_90:bd:d8	Nearest	EAP	4 64	Request, Tunneled TLS EAP (EAP-TTLS)
5	0.015110827	HewlettP_df:d1:64	Nearest	SSL	325	Client Hello
6	0.024245228	D-LinkCo_90:bd:d8	Nearest	TLSv1	1042	Server Hello, Certificate, Server Key Ex...
7	0.024643123	HewlettP_df:d1:64	Nearest	EAP	6 60	Response, Tunneled TLS EAP (EAP-TTLS)
8	0.025877374	D-LinkCo_90:bd:d8	Nearest	TLSv1	183	Server Hello, Certificate, Server Key Ex...
9	0.028351015	HewlettP_df:d1:64	Nearest	TLSv1	158	Client Key Exchange, Change Cipher Spec, ...
10	0.032277182	D-LinkCo_90:bd:d8	Nearest	TLSv1	87	Change Cipher Spec, Encrypted Handshake ...
11	0.032903356	HewlettP_df:d1:64	Nearest	TLSv1	146	Application Data, Application Data
12	1.034148559	D-LinkCo_90:bd:d8	Nearest	EAP	8 64	Failure

Figura 11 - Troca de mensagens entre o Cliente e o Autenticador com dados de acesso inválidos.

A sequência de troca de mensagens compreende (identificadas em vermelho na Figura 11):

1. *EAPOL Start*: Cliente envia mensagem informando que deseja se conectar.
2. *EAP-Request Identity*: Autenticador solicita a identificação do Cliente (nome do usuário).
3. *EAP-Response Identity*: Cliente responde o Autenticador com sua identificação.
4. *EAP-Request, Tunneled TLS EAP*: O Autenticador envia mensagem ao cliente informando o método EAP que deve ser utilizado.
5. *Mensagens do protocolo EAP-TTLS*: O Autenticador e o Cliente trocam mensagens para entrar em um acordo sobre o *Cipher* utilizado e demais parâmetros do protocolo.

6. *EAP-Response, Tunneled TLS EAP*: Cliente responde ao Autenticador informando que aceita utilizar o método EAP-TTLS.
7. *Handshake EAP-TTLS*: o método EAP-TTLS realiza um *handshake* para acordar entre os parâmetros utilizados na criação da chave MSK.
8. *EAP-Failure*: Autenticador envia mensagem ao cliente informando que a autenticação falhou.

A Figura 12 apresenta as mensagens trocadas entre o Autenticador e o Servidor RADIUS, onde o servidor rejeita a conexão por não ter conseguido autenticar os dados do usuário, estas são as mensagens trocadas entre o Autenticador e o Servidor RADIUS na Figura 1.

A sequência de troca de mensagens compreende as mensagens da Figura 11 encapsuladas no protocolo RADIUS (as mensagens encapsuladas estão identificadas na Figura 12 com o mesmo número utilizado na Figura 11):

3. *Access-Request*: é a mensagem *EAP-Response Identity* (3 na Figura 11) encapsulada em uma mensagem do protocolo RADIUS.
4. *Access-Challenge*: é a mensagem *EAP-Request TTLS* encapsulada pelo protocolo RADIUS.
5. Mensagens do protocolo *EAP-TTLS*: mensagens *Cliente Hello* e *Server Hello* encapsuladas.
6. *Access-Request*: Mensagem *EAP-Response TTLS* encapsulada pelo protocolo RADIUS.
7. *Handshake EAP-TTLS* encapsulado pelo protocolo RADIUS.
8. *Access-Reject*: mensagem *EAP-Failure* encapsulada, onde o Servidor RADIUS informa que a autenticação falhou.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.32.162.51	10.32.162.130	RADIUS	3	226 Access-Request(1) (id=0, l=184)
2	0.000637643	10.32.162.130	10.32.162.51	RADIUS	4	106 Access-Challenge(11) (id=0, l=64)
3	0.001922835	10.32.162.51	10.32.162.130	RADIUS	5	538 Access-Request(1) (id=1, l=496)
4	0.009829281	10.32.162.130	10.32.162.51	RADIUS	5	1132 Access-Challenge(11) (id=1, l=1090)
5	0.011472151	10.32.162.51	10.32.162.130	RADIUS	6	235 Access-Request(1) (id=2, l=193)
6	0.011877206	10.32.162.130	10.32.162.51	RADIUS	6	265 Access-Challenge(11) (id=2, l=223)
7	0.016183070	10.32.162.51	10.32.162.130	RADIUS	7	369 Access-Request(1) (id=3, l=327)
8	0.018201536	10.32.162.130	10.32.162.51	RADIUS	7	169 Access-Challenge(11) (id=3, l=127)
9	0.019696691	10.32.162.51	10.32.162.130	RADIUS	8	357 Access-Request(1) (id=4, l=315)
10	1.020188209	10.32.162.130	10.32.162.51	RADIUS	8	86 Access-Reject(3) (id=4, l=44)

Figura 12 - Troca de mensagens entre o Autenticador e o Servidor RADIUS com dados de acesso inválidos.

5.2 Segundo cenário – Autenticação com dados de acesso corretos.

O segundo cenário, consiste na validação da autenticação utilizando usuário e senha corretos. Na Figura 13 visualizamos a troca de mensagens entre o Cliente e o Autenticador, esta troca de mensagens é semelhante a troca descrita na Figura 11, sendo a diferença a ultima mensagem, que neste caso é a mensagem *EAP-Success* (em destaque na Figura), informando que a autenticação foi realizada com sucesso.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	Tp-LinkT_03:28:ec	Nearest	EAPOL	60	Start
2	0.010724365	Dell_3b:d4:fc	Tp-LinkT_03:28:ec	EAP	64	Request, Identity
3	0.010953189	Tp-LinkT_03:28:ec	Nearest	EAP	60	Response, Identity
4	0.012979470	Dell_3b:d4:fc	Tp-LinkT_03:28:ec	EAP	64	Request, Tunneled TLS EAP (EAP-TTLS)
5	0.013470814	Tp-LinkT_03:28:ec	Nearest	TLSv1	325	Client Hello
6	0.023308816	Dell_3b:d4:fc	Tp-LinkT_03:28:ec	TLSv1	1042	Server Hello, Certificate, Server Key Exchange, Server Hello Done
7	0.023690488	Tp-LinkT_03:28:ec	Nearest	EAP	60	Response, Tunneled TLS EAP (EAP-TTLS)
8	0.025434302	Dell_3b:d4:fc	Tp-LinkT_03:28:ec	TLSv1	183	Server Hello, Certificate, Server Key Exchange, Server Hello Done
9	0.027842885	Tp-LinkT_03:28:ec	Nearest	TLSv1	158	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
10	0.031197575	Dell_3b:d4:fc	Tp-LinkT_03:28:ec	TLSv1	87	Change Cipher Spec, Encrypted Handshake Message
11	0.031771823	Tp-LinkT_03:28:ec	Nearest	TLSv1	146	Application Data, Application Data
12	0.034527354	Dell_3b:d4:fc	Tp-LinkT_03:28:ec	EAP	64	Success

Figura 13 - Troca de mensagens entre o Cliente e o Autenticador na etapa de autenticação

Na Figura 14 estão as mensagens trocadas entre o Autenticador e o Servidor RADIUS. Estas mensagens são semelhantes as da Figura 12, sendo a diferença que ao final da autenticação a conexão é aceita e o Servidor RADIUS envia uma mensagem do tipo *Access-Accept* (em destaque na Figura), contendo a chave MSK.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.32.162.51	10.32.162.130	RADIUS	228	Access-Request(1) (id=0, l=186)
2	0.000637833	10.32.162.130	10.32.162.51	RADIUS	106	Access-Challenge(11) (id=0, l=64)
3	0.002553445	10.32.162.51	10.32.162.130	RADIUS	539	Access-Request(1) (id=1, l=497)
4	0.010418393	10.32.162.130	10.32.162.51	RADIUS	1132	Access-Challenge(11) (id=1, l=1090)
5	0.012618469	10.32.162.51	10.32.162.130	RADIUS	236	Access-Request(1) (id=2, l=194)
6	0.013048205	10.32.162.130	10.32.162.51	RADIUS	265	Access-Challenge(11) (id=2, l=223)
7	0.016871190	10.32.162.51	10.32.162.130	RADIUS	370	Access-Request(1) (id=3, l=328)
8	0.018859620	10.32.162.130	10.32.162.51	RADIUS	169	Access-Challenge(11) (id=3, l=127)
9	0.020722792	10.32.162.51	10.32.162.130	RADIUS	358	Access-Request(1) (id=4, l=316)
10	0.021354330	10.32.162.130	10.32.162.51	RADIUS	215	Access-Accept(2) (id=4, l=173)

Figura 14 - Troca de mensagens entre o Autenticador e o Servidor RADIUS na etapa de autenticação

A fase de autenticação é concluída com sucesso quando o Autenticador recebe a mensagem *Access-Accept* (em destaque na Figura 14) do Servidor RADIUS e envia a mensagem *EAP-Success* (em destaque na Figura 13) ao Cliente. Ao final desta etapa, o Cliente e o Autenticador possuem a chave mestre MSK e se iniciará a segunda etapa, que é o protocolo MKA, o qual é validado no terceiro cenário.

5.3 Terceiro cenário – Validação do protocolo MKA

Após concluída a etapa de autenticação, são derivadas as demais chaves necessárias para a comunicação segura. Através de mensagens de *debug* no software *wpa_supplicant*, podemos ver a chave MSK recebida e as chaves derivadas a partir da chave MSK. Na Figura 15 podemos ver a chave MSK recebida (1 na Figura), sendo a partir dela derivadas as chaves CAK (2) e CKN (3). A partir da chave CAK são derivadas as chaves KEK (4) e ICK (5). As chaves derivadas são utilizadas pelo protocolo MKA. Estas chaves são derivadas pelo Cliente e pelo Autenticador da mesma forma.

```

1: MSK: - hexdump(len=16): 52 ba ce 7f 57 14 00 65 3d a0 92 da d3 a2 d1 8b
2: Derived CAK - hexdump(len=16): 37 cd b0 8b fd 37 da 13 ef 69 e3 bd 1c e7 9e 1d
3 "Derived CKN - hexdump(len=16): 02 81 3c d5 b0 99 2f d8 4c 19 b1 57 07 a7 f1 d2
4 :KaY: Derived KEK - hexdump(len=16): dd 89 5e a8 be 44 f2 a9 c0 9d 83 79 0c 84 df 53
5: KaY: Derived ICK - hexdump(len=16): 4b 75 8f ea 40 93 c9 b3 e8 0b 57 16 e6 c3 b1 cd

```

Figura 15 - Chaves derivadas pelo cliente após receber a chave MSK.

Após derivadas as chaves inicia-se a troca de mensagens do protocolo MKA. O protocolo MKA se inicia com cada dispositivo enviando uma mensagem *broadcast* contendo sua prioridade para ser o servidor de chaves (*Key Server Priority*) e uma lista de participantes (*Potential Peer List*). Quando todos os participantes entrarem em acordo em relação à lista de participantes, o dispositivo com maior prioridade é escolhido como servidor de chaves e este gera e envia a chave SAK.

O protocolo MKA possui uma propriedade de *liveness* baseada em *heartbeat*. Portanto, a cada 2 segundos cada dispositivo envia uma mensagem informando que está “vivo”. Na Figura 16, o Cliente inicia o protocolo MKA enviando uma mensagem *multicast* informando sua prioridade (1 na Figura), o SCI (2), seu identificador (3), o número da mensagem (4) e a chave CAK (5).

```

► Frame 13: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface 0
► Ethernet II, Src: Tp-LinkT_03:28:ec (f8:1a:67:03:28:ec), Dst: Nearest (01:80:c2:00:00:03)
▼ 802.1X Authentication
  Version: 802.1X-2010 (3)
  Type: MKA (5)
  Length: 68
▼ MACsec Key Agreement
  ▼ Basic Parameter set
    MKA Version Identifier: 1
    1 Key Server Priority: 255
    0... .. = Key Server: False
    .1.. .. = MACsec Desired: True
    ..11 .. = MACsec Capability: MACsec Integrity with confidentiality offset (3)
    ... 0000 0010 1100 = Parameter set body length: 44
    2 SCI: f81a670328ec0001
    3 Actor Member Identifier: c13c4fedcc7ab9292a33e14d
    4 Actor Message Number: 00000001
    Algorithm Agility: 0080c201
    5 CAK Name: 02813cd5b0992fd84c19b15707a7f1d2
  ► Integrity Check Value Set

```

Figura 16 - Mensagem inicial do protocolo MKA, enviada pelo Cliente.

A Figura 17 mostra a segunda mensagem enviada. Esta mensagem é enviada pelo Autenticador em *multicast* informando sua prioridade (1 na Figura), o SCI (2), seu identificador (3), o número da mensagem (4), a chave CAK (5) e uma lista de participantes (6). Como podemos observar, o Cliente já aparece na lista de participantes potenciais (o Actor Member Identifier corresponde ao Actor Member Identifier do Cliente (3 na Figura 16)).

- ▶ Frame 14: 106 bytes on wire (848 bits), 106 bytes captured (848 bits) on interface 0
- ▶ Ethernet II, Src: Dell_3b:d4:fc (00:14:22:3b:d4:fc), Dst: Nearest (01:80:c2:00:00:03)
- ▼ 802.1X Authentication
 - Version: 802.1X-2010 (3)
 - Type: MKA (5)
 - Length: 88
 - ▼ MACsec Key Agreement
 - ▼ Basic Parameter set
 - MKA Version Identifier: 1
 - 1 Key Server Priority: 48
 - 1... = Key Server: True
 - .1.. = MACsec Desired: True
 - ..11 = MACsec Capability: MACsec Integrity with confidentiality offset (3)
 - 0000 0010 1100 = Parameter set body length: 44
 - 2 SCI: 0014223bd4fc0001
 - 3 Actor Member Identifier: ad7f52c0358e91a187db40ca
 - 4 Actor Message Number: 00000001
 - Algorithm Agility: 0080c201
 - 5 CAK Name: 02813cd5b0992fd84c19b15707a7f1d2
 - ▼ Potential Peer List Parameter set
 - Parameter set type: Potential Peer List (2)
 - 6 { 0000 0001 0000 = Parameter set body length: 16
 - Actor Member Identifier: c13c4fedcc7ab9292a33e14d
 - Actor Message Number: 00000001
 - ▶ Integrity Check Value Set

Figura 17 - Mensagem enviada pelo Autenticador informando sua prioridade.

Após 2 segundos, o Cliente envia uma nova mensagem *multicast* com as mesmas informações da mensagem enviada anteriormente, porém agora o autenticador estará na sua lista de participantes.

Neste momento, como todos participantes possuem a mesma lista de participantes vivos, o participante com maior prioridade se torna o servidor de chaves e envia a chave SAK para os demais participantes. Neste caso, o Autenticador foi definido com uma prioridade maior para que seja o servidor de chaves. A chave SAK gerada é protegida sendo utilizado o algoritmo *AES Key Wrap* e a chave KEK. Na Figura 18 temos a mensagem enviada pelo Autenticador, contendo a chave SAK protegida (1 na Figura), o número da chave (2) e o número da associação (3).

- ▶ Frame 16: 138 bytes on wire (1104 bits), 138 bytes captured (1104 bits) on interface 0
- ▶ Ethernet II, Src: Dell_3b:d4:fc (00:14:22:3b:d4:fc), Dst: Nearest (01:80:c2:00:00:03)
- ▼ 802.1X Authentication
 - Version: 802.1X-2010 (3)
 - Type: MKA (5)
 - Length: 120
 - ▼ MACsec Key Agreement
 - ▼ Basic Parameter set
 - MKA Version Identifier: 1
 - Key Server Priority: 48
 - 1... = Key Server: True
 - .1.. = MACsec Desired: True
 - ..11 = MACsec Capability: MACsec Integrity with confidentiality offset (3)
 - 0000 0010 1100 = Parameter set body length: 44
 - SCI: 0014223bd4fc0001
 - Actor Member Identifier: ad7f52c0358e91a187db40ca
 - Actor Message Number: 00000002
 - Algorithm Agility: 0080c201
 - CAK Name: 02813cd5b0992fd84c19b15707a7f1d2
 - ▶ Live Peer List Parameter set
 - ▼ Distributed SAK parameter set
 - Parameter set type: Distributed SAK (4)
 - 300.. = Distributed AN: 0
 - ..00 = Confidentiality Offset: 0
 - 0000 0001 1100 = Parameter set body length: 28
 - 2Key Number: 00000001
 - 1AES Key Wrap of SAK: 76df379c7488e5ec022f2d72095c2886377be262d479386a
 - ▶ Integrity Check Value Set

Figura 18 - Mensagem enviada pelo Autenticador ao Cliente contendo a chave SAK.

Ao receber a chave SAK, os participantes enviam uma mensagem informando a instalação da chave. Na Figura 19, o cliente informa a utilização da chave SAK (em destaque na barra azul), atualizando os dados relativos a chave. Neste caso, como é a primeira chave transmitida, os dados das chaves antigas estão todos em zero.

Através de mensagens de *debug* no software *wpa_supplicant* do Cliente, conseguimos observar a chave *AES Key Wrap of SAK* transmitida pelo Autenticador (1 da Figura 18) e a chave SAK que será utilizada na criptografia dos pacotes (*AES Key Unwrap of SAK*). As duas chaves podem ser vistas na Figura 20.

Após a instalação da chave SAK, os participantes continuaram enviando mensagens *heartbeat* a cada 2 segundos para informarem que continuam vivos, informando também o número do último pacote recebido (*Lowest Acceptable PN*), para controle da geração de novas chaves pelo servidor de chaves. Uma nova chave SAK é gerada quando forem enviados 0xC0000000 pacotes utilizando a chave atual (em uma conexão a 10Gb/s, seria necessário gerar uma nova chave a cada 5 minutos).

```

▶ Frame 17: 150 bytes on wire (1200 bits), 150 bytes captured (1200 bits) on interface 0
▶ Ethernet II, Src: Tp-LinkT_03:28:ec (f8:1a:67:03:28:ec), Dst: Nearest (01:80:c2:00:00:03)
▼ 802.1X Authentication
  Version: 802.1X-2010 (3)
  Type: MKA (5)
  Length: 132
  ▼ MACsec Key Agreement
    ▼ Basic Parameter set
      MKA Version Identifier: 1
      Key Server Priority: 255
      0... .... = Key Server: False
      .1.. .... = MACsec Desired: True
      ..11 .... = MACsec Capability: MACsec Integrity with confidentiality offset (3)
      .... 0000 0010 1100 = Parameter set body length: 44
      SCI: f81a670328ec0001
      Actor Member Identifier: c13c4fedcc7ab9292a33e14d
      Actor Message Number: 00000003
      Algorithm Agility: 0080c201
      CAK Name: 02813cd5b0992fd84c19b15707a7f1d2
    ▶ Live Peer List Parameter set
    ▼ MACsec SAK Use parameter set
      Parameter set type: MACsec SAK Use (3)
      00.. .... = Latest Key AN: 0
      ..0. .... = Latest Key tx: False
      ...1 .... = Latest Key rx: True
      .... 00.. = Old Key AN: 0
      .... ..0. = Old Key tx: False
      .... ...0 = Old Key rx: False
      0... .... = Plain tx: False
      .0.. .... = Plain rx: False
      ...0 .... = Delay protect: False
      .... 0000 0010 1000 = Parameter set body length: 40
      Latest Key: Key Sever Member Identifier: ad7f52c0358e91a187db40ca
      Latest Key: Key Number: 00000001
      Latest Key: Lowest Acceptable PN: 00000001
      Old Key: Key Sever Member Identifier: 00000000000000000000000000000000
      Old Key: Key Number: 00000000
      Old Key: Lowest Acceptable PN: 00000001
    ▶ Integrity Check Value Set

```

Figura 19 - Cliente informa a instalação da SAK.

```

*** Distributed SAK ***
Distributed AN.....: 0
Confidentiality Offset: 1
Body Length.....: 28
Key Number.....: 1
AES Key Wrap of SAK...: - hexdump(len=24): 76 df 37 9c 74 88 e5 ec 02 2f 2d 72 09 5c 28 86 37 7b e2 62 d4 79 38 6a
AES Key Unwrap of SAK: - hexdump(len=16): 86 cd 49 84 4e 1a 95 ba 59 13 2d 65 37 e9 e0 f5

```

Figura 20 - Chave AES recebida pelo cliente.

5.4 Quarto cenário - Envio de pacotes UDP protegidos

Os cenários anteriores são realizados pelo Cliente A e Cliente B. Ao final do terceiro cenário os dois Clientes instalaram suas chaves SAK. Com a instalação da chave SAK é possível iniciar o envio de informações protegidas. Para validarmos o envio dos pacotes foi criado um programa em C que realiza o envio de pacotes do tipo UDP para um determinado endereço MAC. Através deste programa foi enviada a seguinte informação: “abcdef123456789abcdef1234567891abcde”. Na Figura 21, podemos ver o pacote UDP enviado pelo Cliente A (a informação dos dados está em destaque na barra azul).

Antes de ser enviado pelo Cliente A o pacote é modificado para o padrão MACsec. Os campos EthernetType (1 na Figura 21,), IPV4 (2), UDP (3) e Data (4) são criptografados e inseridos no campo Data (1 na Figura 22) do novo pacote.

```

▶ Frame 47: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 1
▼ Ethernet II, Src: Tp-LinkT_03:28:ec (f8:1a:67:03:28:ec), Dst: HewlettP_df:d1:64 (00:21:5a:df:d1:64)
  ▶ Destination: HewlettP_df:d1:64 (00:21:5a:df:d1:64)
  ▶ Source: Tp-LinkT_03:28:ec (f8:1a:67:03:28:ec)
  Type: IPv4 (0x0800) 1
▶ Internet Protocol Version 4, Src: 162.110.0.50, Dst: 192.168.0.35 2
▶ User Datagram Protocol, Src Port: 3423 (3423), Dst Port: 5342 (5342) 3
▼ Data (18 bytes) 4
  Data: abcdef123456789abcdef1234567891abcde
  [Length: 18]

```

Figura 21 - Pacote UDP enviado do Cliente A.

Na Figura 22 visualizamos o pacote UDP da Figura 21 já no formato do padrão MACsec. Este é o pacote enviado pela rede. Como podemos observar, o campo Data (1 na Figura) possui 48 bytes, correspondente aos campos EthernetType, IPV4, UDP e Data da Figura 21 e está criptografado. Além disto, foi adicionada a SecTAG (2), o MACsec EtherType (3) e o ICV ao pacote (4). O campo *System Identifier* (5) da SecTAG possui o endereço MAC do dispositivo que esta enviando a mensagem, neste caso o endereço MAC do Cliente A.

```

▶ Frame 46: 92 bytes on wire (736 bits), 92 bytes captured (736 bits) on interface 0
▼ Ethernet II, Src: Tp-LinkT_03:28:ec (f8:1a:67:03:28:ec), Dst: HewlettP_df:d1:64 (00:21:5a:df:d1:64)
  ▶ Destination: HewlettP_df:d1:64 (00:21:5a:df:d1:64)
  ▶ Source: Tp-LinkT_03:28:ec (f8:1a:67:03:28:ec)
  Type: Unknown (0x88e5) 3
▼ 802.1AE Secure tag 2
  ▶ TCI=0x2c: V=0, ES=0, SC=1, SCB=0, E=1, C=1, AN=0
    0000 0000 = Short length: 0
    0000 0000 0000 0000 0000 0000 0011 0100 = Packet number: 52
    5 System Identifier: Tp-LinkT_03:28:ec (f8:1a:67:03:28:ec)
    0000 0000 0000 0001 = Port number: 1
    4 ICV: f2536e350dcb30484c49983a68d0de16
▼ Data (48 bytes)
  1 Data: 1978b513b087b50992a216f411db579c79959179d21cb559...
  [Length: 48]

```

Figura 22 - Pacote MACsec protegido enviado pelo Cliente A.

Ao chegar no Autenticador, este pacote é descriptografado com a chave SAK trocada com o Cliente A. Após será criptografado com a chave trocada com o Cliente B e enviado ao Cliente B.

Na Figura 23 apresentamos o pacote MACsec recebido pelo Cliente B. Podemos observar que o campo Data (1 na Figura) mudou em relação ao campo Data da Figura 22, isto se deve ao fato de ele ter sido criptografado com outra chave, as outras diferenças estão no campo ICV (4) que foi alterado devido a ter sido gerado com outra chave. O campo *System Identifier* (3) da SecTAG (2) foi alterado e possui o endereço MAC do switch.

```

▶ Frame 47: 92 bytes on wire (736 bits), 92 bytes captured (736 bits) on interface 0
▼ Ethernet II, Src: Tp-LinkT_03:28:ec (f8:1a:67:03:28:ec), Dst: HewlettP_df:d1:64 (00:21:5a:df:d1:64)
  ▶ Destination: HewlettP_df:d1:64 (00:21:5a:df:d1:64)
  ▶ Source: Tp-LinkT_03:28:ec (f8:1a:67:03:28:ec)
  Type: Unknown (0x88e5)
▼ 802.1AE Secure tag 2
  ▶ TCI=0x2c: V=0, ES=0, SC=1, SCB=0, E=1, C=1, AN=0
    0000 0000 = Short length: 0
    0000 0000 0000 0000 0000 0000 0011 0100 = Packet number: 52
  3 System Identifier: D-LinkCo_90:bd:d8 (00:21:91:90:bd:d8)
    0000 0000 0000 0001 = Port number: 1
  4 ICV: b503d5a1df571dc20be5ef5eda17a028
▼ Data (48 bytes)
  1 Data: 0eb089b5112c5b21cd83517ccf985f75fb65e249903991ee...
    [Length: 48]

```

Figura 23 - Pacote MACsec recebido pelo Cliente B.

O Cliente B, ao receber o pacote MACsec irá descriptografá-lo. Na Figura 24 temos o pacote UDP recebido pelo Cliente B. Podemos observar que ele é idêntico ao pacote UDP enviado pelo Cliente A (Figura 21), sendo recebido o dado enviado pelo Cliente A (em destaque na Figura 24).

```

▶ Frame 48: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 1
▼ Ethernet II, Src: Tp-LinkT_03:28:ec (f8:1a:67:03:28:ec), Dst: HewlettP_df:d1:64 (00:21:5a:df:d1:64)
  ▶ Destination: HewlettP_df:d1:64 (00:21:5a:df:d1:64)
  ▶ Source: Tp-LinkT_03:28:ec (f8:1a:67:03:28:ec)
  Type: IPv4 (0x0800)
▶ Internet Protocol Version 4, Src: 162.110.0.50, Dst: 192.168.0.35
▶ User Datagram Protocol, Src Port: 3423 (3423), Dst Port: 5342 (5342)
▼ Data (18 bytes)
  Data: abcdef123456789abcdef1234567891abcde
    [Length: 18]

```

Figura 24 - Pacote UDP recebido pelo Cliente B.

Como visto, conseguimos transmitir informações entre dois dispositivos Cliente de forma segura, validando a interface proposta na Figura 9. O Anexo A mostra os pacotes trocados entre dois clientes e mostra como é realizado o reenvio dos pacotes pelo Autenticador ao receber os pacotes.

6 DESENVOLVIMENTO DE UM MÓDULO DE HARDWARE PARA GERAÇÃO DE PACOTES NO PADRÃO MACSEC

No Capítulo anterior foi apresentada a validação da interface de comunicação desenvolvida em software. Após validado o envio e recebimento de pacotes criptografados no padrão MACsec, iniciamos a segunda parte deste trabalho, que se refere ao desenvolvimento de um módulo de hardware capaz de gerar pacotes no padrão MACsec, realizando a criptografia dos dados GCM-AES e formatando o pacote conforme o padrão MACsec.

Este módulo de hardware foi desenvolvido em uma placa NetFPGA Sume, a qual utiliza um dispositivo FPGA modelo Virtex 7 XC7VX690T, com capacidade de até quatro interfaces 10GbE através de dispositivos ópticos SFP+ [ZIL14]. Para o envio e recebimento de dados através dos módulos SFP+, foi utilizado o projeto desenvolvido anteriormente pela parceria GAPH e TERACOM LTDA. A Figura 25 apresenta uma visão geral dos principais blocos utilizados. Os módulos reutilizados compreendem o GT_gtwizard, PCS, MAC, I2C Interface, UART Interface. O módulo GTH_gtwizard realiza a interface com o transceiver GTH (interface de alta velocidade – 10 Gbps). O módulo PCS realiza a interface entre a camada física e de enlace. O módulo MAC é responsável por controlar o recebimento e envio de pacotes. O módulo I2C Interface realiza a configuração do chip de modelo SI5324 externo ao FPGA que gera a referência de clock externa de 156.25MHz conectada ao banco de transceivers GTH.

No contexto deste trabalho desenvolveu-se o módulo MACsec (em vermelho na Figura). Este módulo instância o módulo GCM-AES (em amarelo) responsável pela criptografia dos dados, sendo capaz de realizar a geração de pacotes no padrão MACsec.

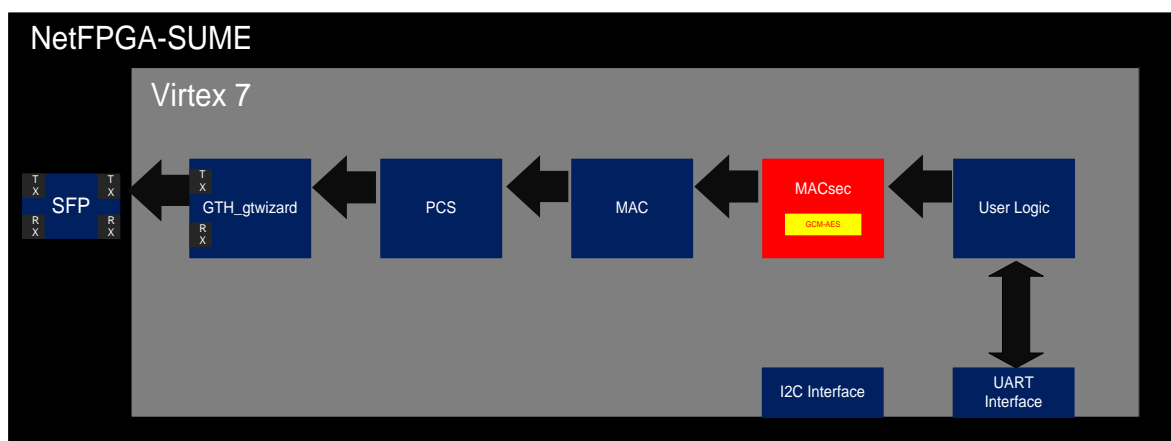


Figura 25 - Diagrama dos principais blocos da interface para envio de pacotes MACsec.

Para o desenvolvimento do módulo MACsec foi utilizado o algoritmo GCM-AES disponível no *opencores* [TAR10]. Este módulo é conectado ao módulo MAC, um módulo também disponível no *opencores* responsável por realizar a interface necessária para comunicação com o módulo de codificação da camada PCS, conforme apresentado na Figura 25. A interface de comunicação *UART Interface* realiza o envio e recebimento de dados através do protocolo UART (*Universal asynchronous receiver/transmitter*). Esse módulo permite a comunicação de um software com a placa, via porta USB (*Universal Serial Bus*).

O módulo GCM-AES utilizado opera em blocos de 128 bits de dados, consumindo 10 ciclos de clock para realizar a criptografia de um bloco de 128 bits. Este módulo utiliza o algoritmo AES-128 para realizar a criptografia dos dados e o GCM para realizar a autenticação dos dados. O GCM utilizado neste módulo produz o resultado (que será o campo ICV) em 8 ciclos de clock. A operação do modulo GCM-AES é apresentado na na Subseção 6.2, onde utilizamos um *testbench* para validar a operação.

6.1 Validação do hardware desenvolvido

Para validar o hardware desenvolvido, foi criado o cenário apresentado na Figura 26. Neste cenário, o Cliente B e o Autenticador estão conectados através de um Switch. O switch foi utilizado por dois motivos: o primeiro motivo é realizar a interface entre as portas elétricas e a porta óptica da NetFPGA, o segundo motivo é que o hardware desenvolvido não implementa a autenticação e acordo de chaves do protocolo MKA, pois necessitaria de um processador embarcado para este fim.

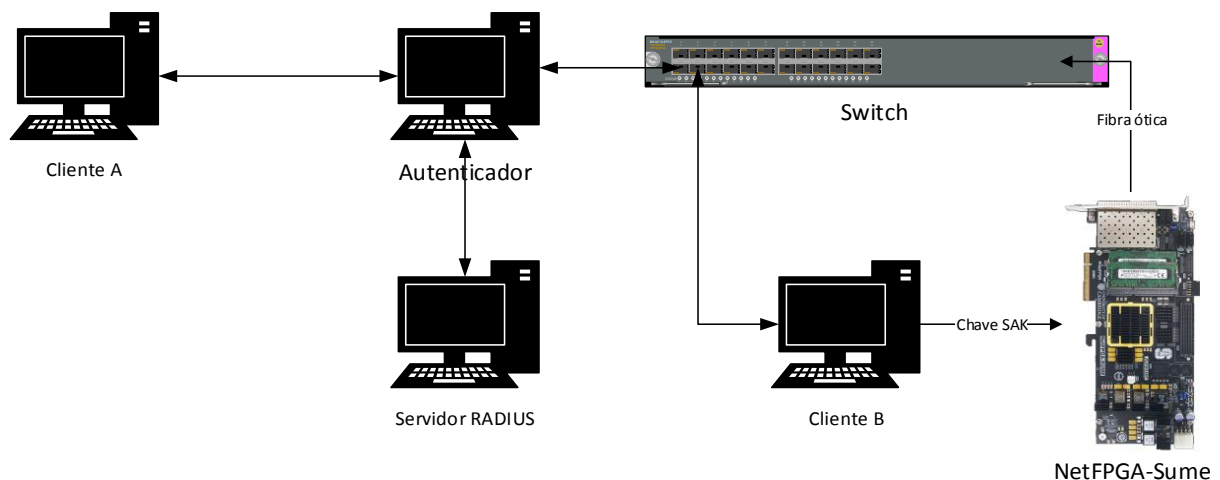


Figura 26 - Infraestrutura para simular o hardware desenvolvido.

Neste ambiente, o Cliente B realiza a autenticação e trocas de chaves com o Autenticador da mesma forma que foi validado no Capítulo 5 deste trabalho. Após realizar a troca de chaves o Cliente B é conectado com a NetFPGA-SUME via *UART Interface* para transferência da Chave SAK, seu endereço MAC e o endereço MAC para o qual deseja enviar um pacote e outros parâmetros necessários. Estas informações são gravadas em registradores utilizados para a geração dos pacotes.

Através da NetFPGA-SUME foi realizado o envio de pacotes com destino ao Cliente A. Com a ferramenta *Wireshark* foi possível monitorar o recebimento e descryptografia dos pacotes pelo Autenticador, validando a geração de pacotes realizada pelo hardware desenvolvido. Através do software utilizado para comunicação via *UART Interface*, configuramos o envio de três pacotes em sequência. A Figura 27 apresenta os 3 pacotes MACsec transmitidos em sequência pela FPGA e recebidos pelo Autenticador (1 na Figura). Ao receber estes pacotes, o *driver* MACsec realiza a descryptografia e verificação, recuperando o pacote transmitido pela FPGA (2 na Figura).

Estes 3 pacotes MACsec contêm a mesma informação, porém cada pacote é criptografado gerando um dado diferente. Isto se deve ao campo *Packet number* da *SecTAG* (identificado com o número 1,2 e 3 na Figura 28), este campo é incrementado a cada pacote enviado para evitar ataques do tipo *Replay*.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	Tp-LinkT_03:28:ec	Broadcast	EAPOL-...	150	
2	0.562446865	D-LinkCo_90:bd:d8	Broadcast	EAPOL-...	150	
1 { 3	4.663126492	Tp-LinkT_03:28:ec	HewlettP_df:01:64	MACSEC	92	MACsec frame
4	4.663159476	Tp-LinkT_03:28:ec	HewlettP_df:01:64	MACSEC	92	MACsec frame
5	4.663179516	Tp-LinkT_03:28:ec	HewlettP_df:01:64	MACSEC	92	MACsec frame
2 { 6	4.663126492	192.168.0.35	162.110.0.50	UDP	60	3423 → 5342 Len=18
7	4.663159476	192.168.0.35	162.110.0.50	UDP	60	3423 → 5342 Len=18
8	4.663179516	192.168.0.35	162.110.0.50	UDP	60	3423 → 5342 Len=18
9	5.001521937	Tp-LinkT_03:28:ec	Broadcast	EAPOL-...	150	
10	5.563306183	D-LinkCo_90:bd:d8	Broadcast	EAPOL-...	150	
11	10.004160358	Tp-LinkT_03:28:ec	Broadcast	EAPOL-...	150	
12	10.564408337	D-LinkCo_90:bd:d8	Broadcast	EAPOL-...	150	

Figura 27 - Pacotes gerados pela NetFPGA-SUME recebidos no Autenticador.

Na Figura 28 estão representados os três pacotes gerados em sequência pela FPGA (1 na Figura 27). Podemos observar que o campo *Packet number* (1, 2 e 3) foi incrementado, causando a alteração dos campos ICV (4, 5 e 6) e Data (7, 8 e 9). Desta forma, os três pacotes são enviados de forma segura.


```

▶ Frame 3: 92 bytes on wire (736 bits), 92 bytes captured (736 bits) on interface 0
▶ Ethernet II, Src: Tp-LinkT_03:28:ec (f8:1a:67:03:28:ec), Dst: HewlettP_df:01:64 (00:21:5a:df:01:64)
▼ 802.1AE Secure tag
  ▶ TCI=0x2c: V=0, ES=0, SC=1, SCB=0, E=1, C=1, AN=0
    0000 0000 = Short length: 0
    0000 0000 0000 0000 0000 0000 0001 1110 = Packet number: 30 1
    System Identifier: Tp-LinkT_03:28:ec (f8:1a:67:03:28:ec)
    0000 0000 0000 0001 = Port number: 1
    ICV: 080e1d427b70edd5c0563d22f4b38029 4
▼ Data (48 bytes)
  Data: 6704f8ca245be6f0baed078f2e0d2182513f4b3abb636f30... 7
  [Length: 48]

▶ Frame 4: 92 bytes on wire (736 bits), 92 bytes captured (736 bits) on interface 0
▶ Ethernet II, Src: Tp-LinkT_03:28:ec (f8:1a:67:03:28:ec), Dst: HewlettP_df:01:64 (00:21:5a:df:01:64)
▼ 802.1AE Secure tag
  ▶ TCI=0x2c: V=0, ES=0, SC=1, SCB=0, E=1, C=1, AN=0
    0000 0000 = Short length: 0
    0000 0000 0000 0000 0000 0000 0001 1111 = Packet number: 31 2
    System Identifier: Tp-LinkT_03:28:ec (f8:1a:67:03:28:ec)
    0000 0000 0000 0001 = Port number: 1
    ICV: 9b600cecb1b86fac402f27f6e28667bb 5
▼ Data (48 bytes)
  Data: b523a83eeb1cc6ea8305364ab595630f2b04cb8ca848f3bd... 8
  [Length: 48]

▶ Frame 5: 92 bytes on wire (736 bits), 92 bytes captured (736 bits) on interface 0
▶ Ethernet II, Src: Tp-LinkT_03:28:ec (f8:1a:67:03:28:ec), Dst: HewlettP_df:01:64 (00:21:5a:df:01:64)
▼ 802.1AE Secure tag
  ▶ TCI=0x2c: V=0, ES=0, SC=1, SCB=0, E=1, C=1, AN=0
    0000 0000 = Short length: 0
    0000 0000 0000 0000 0000 0000 0010 0000 = Packet number: 32 3
    System Identifier: Tp-LinkT_03:28:ec (f8:1a:67:03:28:ec)
    0000 0000 0000 0001 = Port number: 1
    ICV: 1ba976c1ab618f7b7adc1aaa41f1393c 6
▼ Data (48 bytes)
  Data: 3d9a79b67f8f4b4db68875ec1e425e7670d899d886b999de... 9
  [Length: 48]

```

Figura 28 – Três pacotes MACsec enviados via FPGA em sequência.

Na Figura 29 apresenta-se os 3 pacotes UDP enviados pela FPGA. Estes são os pacotes da Figura 28 após eles serem verificados e descriptografados. Como podemos observar, a informação enviada é a mesma nos 3 pacotes (1, 2 e 3 na Figura).

```

▶ Frame 6: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 1
▶ Ethernet II, Src: Tp-LinkT_03:28:ec (f8:1a:67:03:28:ec), Dst: HewlettP_df:01:64 (00:21:5a:df:01:64)
▶ Internet Protocol Version 4, Src: 192.168.0.35, Dst: 162.110.0.50
▶ User Datagram Protocol, Src Port: 3423 (3423), Dst Port: 5342 (5342)
▼ Data (18 bytes)
  Data: abcdef123456789abcdef1234567891abcde 1
  [Length: 18]

▶ Frame 7: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 1
▶ Ethernet II, Src: Tp-LinkT_03:28:ec (f8:1a:67:03:28:ec), Dst: HewlettP_df:01:64 (00:21:5a:df:01:64)
▶ Internet Protocol Version 4, Src: 192.168.0.35, Dst: 162.110.0.50
▶ User Datagram Protocol, Src Port: 3423 (3423), Dst Port: 5342 (5342)
▼ Data (18 bytes)
  Data: abcdef123456789abcdef1234567891abcde 2
  [Length: 18]

▶ Frame 8: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 1
▶ Ethernet II, Src: Tp-LinkT_03:28:ec (f8:1a:67:03:28:ec), Dst: HewlettP_df:01:64 (00:21:5a:df:01:64)
▶ Internet Protocol Version 4, Src: 192.168.0.35, Dst: 162.110.0.50
▶ User Datagram Protocol, Src Port: 3423 (3423), Dst Port: 5342 (5342)
▼ Data (18 bytes)
  Data: abcdef123456789abcdef1234567891abcde 3
  [Length: 18]

```

Figura 29 - Três pacotes UDP após serem verificados.

6.2 Testbench de validação do GCM-AES

Para compreender e validar o funcionamento do módulo GCM-AES, foi utilizado um *testbench* e utilizado como entrada um dos vetores de teste disponíveis em [RAN11]. Os valores de cada campo do vetor de teste estão descritos na Tabela 2. É esperado que o algoritmo GCM-AES gere como saída os valores dos campos *Secure Data* e *ICV*.

A Figura 30 apresenta a validação do algoritmo GCM-AES. Os pontos principais estão enumerados em vermelho e descritos abaixo:

1. O primeiro passo é a etapa de *setup*, onde é transmitida a chave SAK. No momento em que a chave SAK estiver disponível deve ser ativado o sinal *cii_ctl_vld* por um ciclo de clock.
2. Após deve ser passado o vetor de inicialização através do sinal *dii_data* e ativado o sinal *cii_IV_vld* até que o sinal *dii_data_not_ready* seja zero. Este processo consome 20 ciclos de clock. Neste momento está concluída a etapa de *setup*.
3. Ao enviar o dado adicional de autenticação A o sinal *dii_data_type* deve ser 1 para informar que esta sendo transmitido um dado adicional.
4. São enviados os primeiros 128 bits do campo User Data.
5. Após 10 ciclos de clock os 128 bits criptografados estarão disponíveis na saída (primeira linha do campo *Secure Data* da Tabela 2).
6. Ao baixar o sinal *dii_data_not_ready*, são enviados os próximos 128 bits do campo User data.
7. Após 10 ciclos de clock os 128 bits criptografados estarão disponíveis na saída (segunda linha do campo *Secure Data* da Tabela 2).
8. São enviados os últimos 128 bits do campo User Data e é ativado o sinal *dii_last_word*, informando que este é o último dado a ser criptografado e que deve ser gerado o ICV.
9. Após 10 ciclos de clock os 128 bits criptografados estarão disponíveis na saída (terceira linha do campo *Secure Data* da Tabela 2).
10. Após 8 ciclos de clock os 128 bits do campo ICV da Tabela 2 estarão disponíveis na saída.

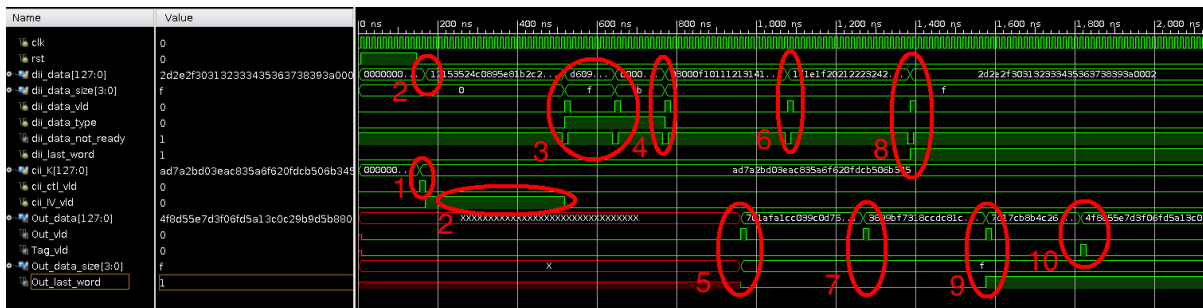


Figura 30 - Simulação do módulo GCM-AES

Tabela 2 - Vetor de teste

Campo	Valor
MAC DA (48 bits)	D6 09 B1 F0 56 63
MAC SA (48 bits)	7A 0D 46 DF 99 8D
User Data (384 bits)	08 00 0F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F 30 31 32 33 34 35 36 37 38 39 3A 00 02
MACsec EtherType (16 bits)	88 E5
TCI e NA (8 bits)	2E
SL (8 bits)	00
PN (32 bits)	B2 C2 84 65
SCI (64 bits)	12 15 35 24 C0 89 5E 81
IV (96 bits)	SCI & PN
SecTAG (128 bits)	MACsec EtherType & TCI e AN & SL & PN & SCI
A (224 bits)	MAC DA & MAC SA & SecTAG
Chave SAK (128 bits)	AD7A2BD03EAC835A6F620FDCB506B345
Secure Data (384 bits)	70 1A FA 1C C0 39 C0 D7 65 12 8A 66 5D AB 69 24 38 99 BF 73 18 CC DC 81 C9 93 1D A1 7F BE 8E DD 7D 17 CB 8B 4C 26 FC 81 E3 28 4F 2B 7F BA 71 3D
ICV (128 bits)	4F 8D 55 E7 D3 F0 6F D5 A1 3C 0C 29 B9 D5 B8 80

6.3 Utilização de área

A partir do relatório gerado ao final da síntese física pode-se observar uma baixa utilização de lógica para a implementação da interface segura. Conforme apresentado na Tabela 3, foi utilizado menos de 2% da área disponível na FPGA. Entretanto, foi possível

observar que o módulo GCM-AES representa um custo relativamente alto, quando comparado ao restante da lógica necessária para realizar o envio de pacotes.

A partir da Figura 31, pode-se ter uma melhor visualização da área utilizada em relação ao total de recursos disponíveis no dispositivo FPGA. O módulo GCM-AES está destacado na cor amarela. Podemos observar que este módulo ocupa aproximadamente 40% da área de toda a lógica utilizada. Os demais módulos em destaque são o MAC/PCS (em azul) e o módulo GTH_gtwizard (em vermelho).

Tabela 3 - Recursos utilizados da NetFPGA

Tipo de recurso	GCM-AES	Total utilizado	Total na FPGA	Utilização (%)
FF/Latch	2075	8581	866400	0.99
LUTs	4231	9038	433200	2,09
BRAM	0	11	1470	0.75

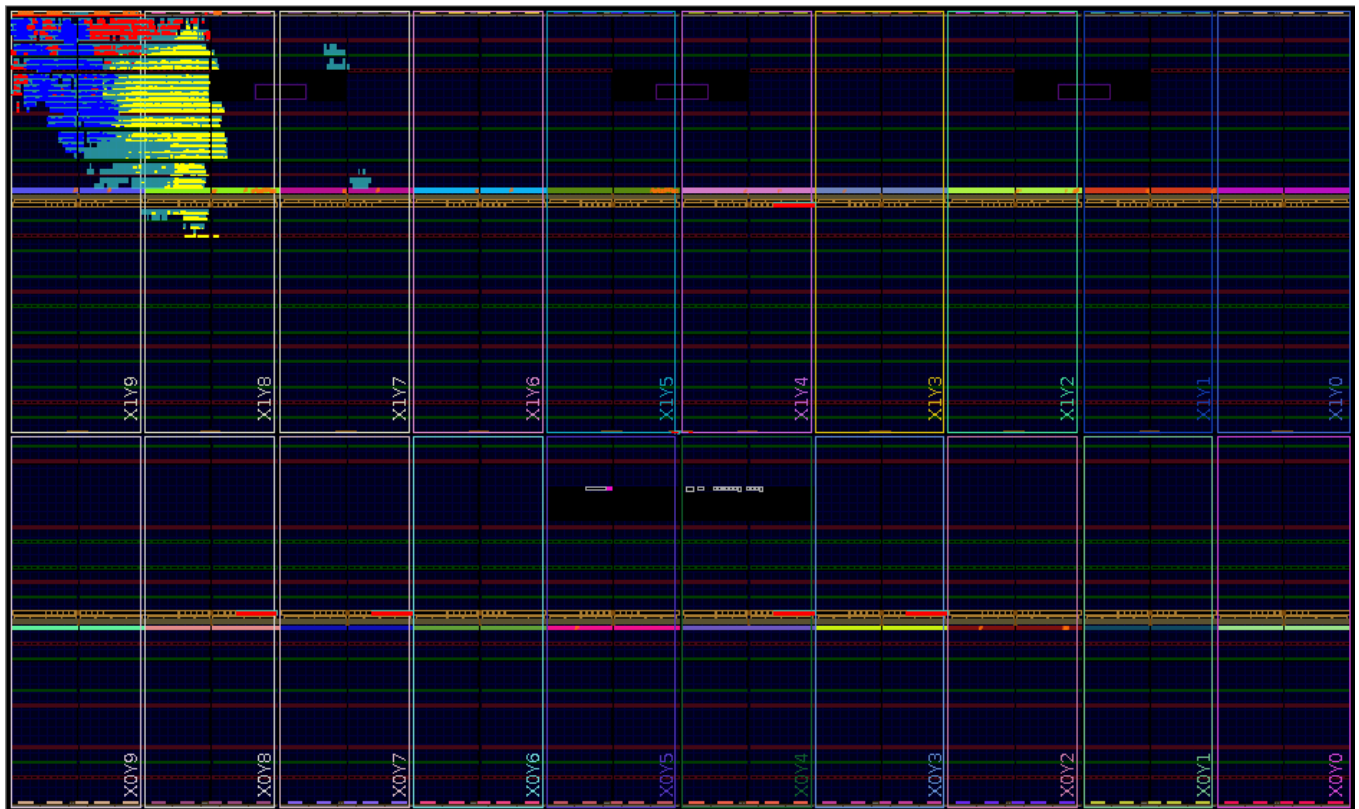


Figura 31 - Posicionamento dos módulos na FPGA XC7VX690T.

7 CONCLUSÃO E TRABALHOS FUTUROS

Segurança de dados é um requisito fundamental para o projeto de sistemas computacionais. Com o advento da chamada “Internet das Coisas” (IoT), onde dados sensíveis são transmitidos entre inúmeros elementos de processamento, distribuídos fisicamente, agregar segurança nas camadas inferiores de rede é uma importante contribuição deste trabalho.

O trabalho descrito neste documento apresentou os conceitos necessários para a implementação de uma interface de comunicação segura, de acordo com o padrão MACsec, agregando segurança na camada 2 do modelo OSI. Durante o desenvolvimento deste trabalho, foram abordados dois tópicos principais: o desenvolvimento e validação de uma interface de comunicação segura em software e uma versão em FPGA capaz de realizar o envio de pacotes MACsec.

Os conhecimentos desenvolvidos nesse trabalho, além de abordar áreas de grande importância dentro do curso de Engenharia de Computação, como de redes de computadores, sistemas embarcados e prototipação em FPGAs, deram a oportunidade de abordar conhecimentos que não são aprofundados durante o curso, que são a área de segurança e criptografia de dados.

O estudo e desenvolvimento realizado neste trabalho pode ser utilizado como base para projetos a serem realizados em trabalhos futuros, como o desenvolvimento de um switch com suporte ao MACsec e a prototipação da interface completa em FPGAs.

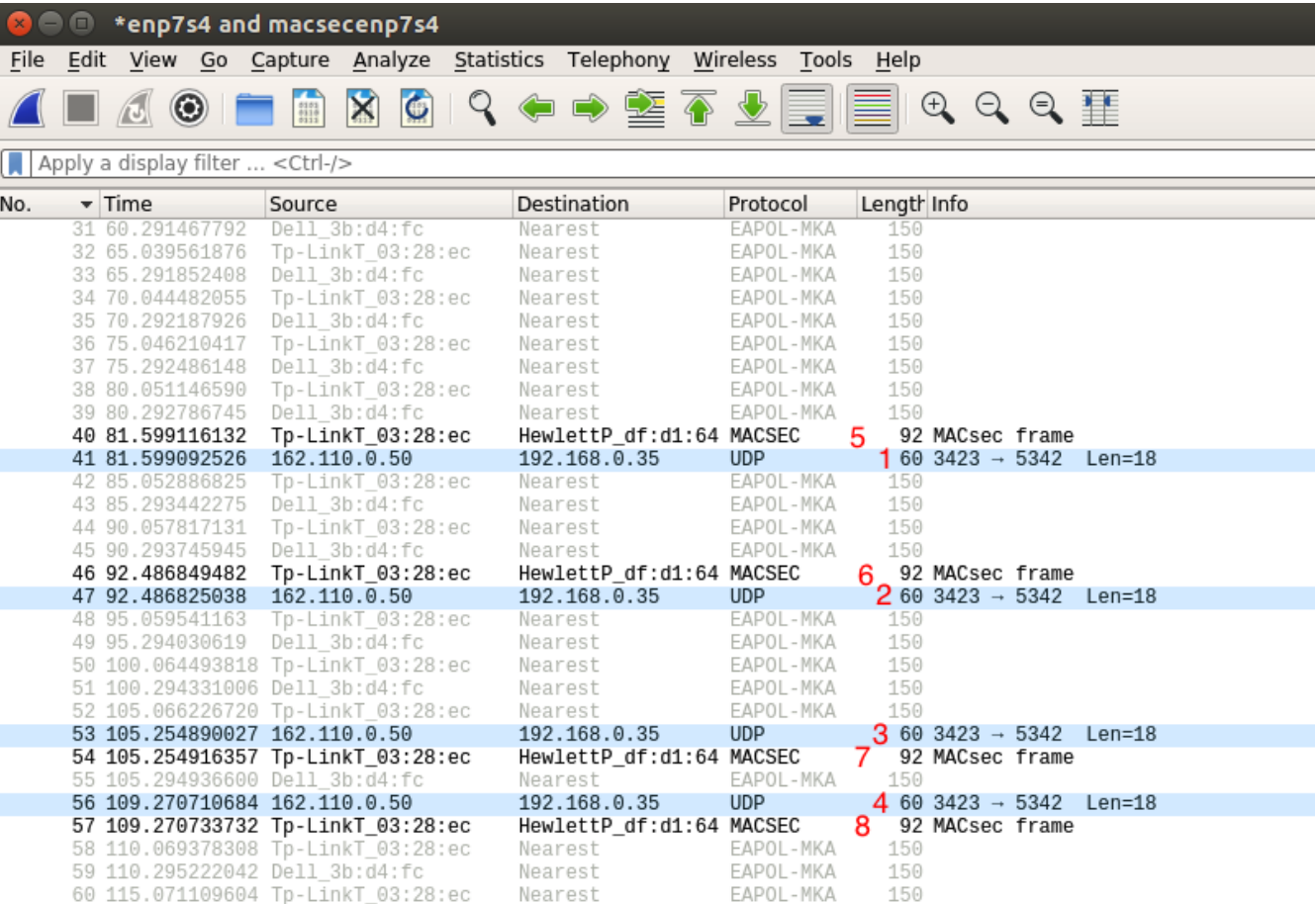
REFERÊNCIAS

- [CAR15] Carnevale, B.; Falaschi, F.; Crocetti, L.; Hunjan, H.; Bisase, S.; Fanucci, L. “An implementation of the 802.1AE MAC Security Standard for in-car networks.”. In: 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT), 2015, pp. 24-28.
- [CHE09] Chen, L., “Recommendation for Key Derivation Using Pseudorandom Functions”, NIST Special Publication 800-108. Capturado em: <http://csrc.nist.gov/publications/nistpubs/800-108/sp800-108.pdf>, outubro, 2009.
- [FRE16] FreeRADIUS. Capturado em: <http://freeradius.org>, junho, 2016.
- [HOS16] Hostapd. Capturado em: <http://w1.fi/hostapd/>, junho, 2016.
- [IEE06] The Institute of Electrical and Electronic Engineers Inc. (IEEE). “IEEE Standard for Local and Metropolitan Area Networks: Media Access Control (MAC) Security”, Capturado em: <http://standards.ieee.org/findstds/standard/802.1AE-2006.html>, agosto, 2006.
- [IEE10] The Institute of Electrical and Electronic Engineers Inc. (IEEE). “IEEE Standard for Local and Metropolitan Area Networks: Port-Based Network Access Control”, Capturado em: <https://standards.ieee.org/findstds/standard/802.1X-2010.html>, fevereiro, 2010.
- [IEE12] The Institute of Electrical and Electronic Engineers Inc. (IEEE). “IEEE Standard for Ethernet, Section 4”. Capturado em: <https://standards.ieee.org/about/get/802/802.3.html>, agosto, 2012.
- [IND12] Indukuri, N. “Layer 2 Security for Smart Grid Networks”. In: IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), 2012, pp. 99-104.
- [LIN16] Linux Kernel 4.6. Capturado em: http://kernelnewbies.org/Linux_4.6, maio, 2016.
- [MCG04] McGrew, D. A., Viega, J., “The Security and Performance of the Galois/Counter Mode (GCM) of Operation (Full Version)”. Capturado em: <http://eprint.iacr.org/2004/193.pdf>, 2004.
- [MOZ12] Mozaffari-Kermani, M., & Reyhani-Masoleh, A. “Efficient and high-performance parallel hardware architectures for the AES-GCM.” IEEE Transactions on Computers. 2012 .61(8), pp. 1165-1178.

- [RAN11] Randall, K., "MACsec GCM-AES Test Vectors". Disponível em: <http://www.ieee802.org/1/files/public/docs2011/bn-randall-test-vectors-0511-v1.pdf>, abril, 2011.
- [SEA13] Seaman, M., "MACsec hops". Capturado em: <http://www.ieee802.org/1/files/public/docs2013/ae-seaman-macsec-hops-0213-v02.pdf>, fevereiro, 2013.
- [SCH02] Schaad, J., Housley, R., "Advanced Encryption Standard (AES) Key Wrap Algorithm". Capturado em: <https://www.ietf.org/rfc/rfc3394.txt>, setembro, 2002.
- [TAN08] Tanguay, A., "10GE MAC Core Specification". Capturado em: http://opencores.org/project,xge_mac, maio, 2008.
- [TAR10] Tariq, A., "Galois Counter Mode Advanced Encryption Standard GCM-AES". Capturado em: <http://opencores.org/project,gcm-aes>, outubro, 2010.
- [WPA16] WPA-Suppliant. Capturado em: http://w1.fi/wpa_supplicant/, junho, 2016.
- [ZIL14] Zilberman, N., Audzevich, Y., Covington, G., Moore, A., "NetFPGA SUME: Toward 100 Gbps as Research Commodity," In: IEEE Micro, 2014, vol.34, no.5, pp.32-41.

ANEXO A

Este Anexo mostra a troca de pacotes entre dois Clientes, com capturas do *wireshark*. A Figura 32 mostra as mensagens enviadas e recebidas pelo cliente A em um determinado tempo. Podemos observar que estão sendo enviados 4 pacotes UDP (numerados de 1 a 4 na Figura), ao enviá-los o *driver* MACsec realiza a formatação do pacote para o padrão MACsec (5 a 8). Como nesta Figura estão sendo capturados dados das interfaces *enp7s4* e *macsecenp7s4*, os dados podem aparecer desordenados. Os demais pacotes que aparecem (*EAPOL-MKA*) são os pacotes enviados periodicamente pelos dispositivos para informarem que ainda estão vivos.



No.	Time	Source	Destination	Protocol	Length	Info
31	60.291467792	Dell_3b:d4:fc	Nearest	EAPOL-MKA	150	
32	65.039561876	Tp-LinkT_03:28:ec	Nearest	EAPOL-MKA	150	
33	65.291852408	Dell_3b:d4:fc	Nearest	EAPOL-MKA	150	
34	70.044482055	Tp-LinkT_03:28:ec	Nearest	EAPOL-MKA	150	
35	70.292187926	Dell_3b:d4:fc	Nearest	EAPOL-MKA	150	
36	75.046210417	Tp-LinkT_03:28:ec	Nearest	EAPOL-MKA	150	
37	75.292486148	Dell_3b:d4:fc	Nearest	EAPOL-MKA	150	
38	80.051146590	Tp-LinkT_03:28:ec	Nearest	EAPOL-MKA	150	
39	80.292786745	Dell_3b:d4:fc	Nearest	EAPOL-MKA	150	
40	81.599116132	Tp-LinkT_03:28:ec	HewlettP_df:d1:64	MACSEC	5	92 MACsec frame
41	81.599092526	162.110.0.50	192.168.0.35	UDP	1	60 3423 → 5342 Len=18
42	85.052886825	Tp-LinkT_03:28:ec	Nearest	EAPOL-MKA	150	
43	85.293442275	Dell_3b:d4:fc	Nearest	EAPOL-MKA	150	
44	90.057817131	Tp-LinkT_03:28:ec	Nearest	EAPOL-MKA	150	
45	90.293745945	Dell_3b:d4:fc	Nearest	EAPOL-MKA	150	
46	92.486849482	Tp-LinkT_03:28:ec	HewlettP_df:d1:64	MACSEC	6	92 MACsec frame
47	92.486825038	162.110.0.50	192.168.0.35	UDP	2	60 3423 → 5342 Len=18
48	95.059541163	Tp-LinkT_03:28:ec	Nearest	EAPOL-MKA	150	
49	95.294030619	Dell_3b:d4:fc	Nearest	EAPOL-MKA	150	
50	100.064493818	Tp-LinkT_03:28:ec	Nearest	EAPOL-MKA	150	
51	100.294331006	Dell_3b:d4:fc	Nearest	EAPOL-MKA	150	
52	105.066226720	Tp-LinkT_03:28:ec	Nearest	EAPOL-MKA	150	
53	105.254890027	162.110.0.50	192.168.0.35	UDP	3	60 3423 → 5342 Len=18
54	105.254916357	Tp-LinkT_03:28:ec	HewlettP_df:d1:64	MACSEC	7	92 MACsec frame
55	105.294936600	Dell_3b:d4:fc	Nearest	EAPOL-MKA	150	
56	109.270710684	162.110.0.50	192.168.0.35	UDP	4	60 3423 → 5342 Len=18
57	109.270733732	Tp-LinkT_03:28:ec	HewlettP_df:d1:64	MACSEC	8	92 MACsec frame
58	110.069378308	Tp-LinkT_03:28:ec	Nearest	EAPOL-MKA	150	
59	110.295222042	Dell_3b:d4:fc	Nearest	EAPOL-MKA	150	
60	115.071109604	Tp-LinkT_03:28:ec	Nearest	EAPOL-MKA	150	

Figura 32 - Wireshark no Cliente A, mensagens enviadas/recebidas.

Na Figura 33 o Autenticador recebe os pacotes pela interface *enp4s2* (lado esquerdo da Figura), descriptografa e envia pela interface *enp2s0* (lado direito da Figura) criptografando os pacotes com a nova chave. O pacote é recebido no padrão MACsec (1, 5, 9, 13) é

descriptografado (2, 6, 10, 14), enviado para a outra interface no formato UDP (3, 7, 11, 15) é criptografado (4, 8, 12, 16 e enviado para o Cliente B.

The image shows two Wireshark capture windows. The left window, titled '*enp412 and macsecenp412', shows a sequence of MACsec frames (92 bytes) and UDP packets (60 bytes) between interfaces. The right window, titled '*enp130 and macsecenp130', shows a similar sequence of MACsec frames and UDP packets. Both captures show a mix of EAPOL-MKA and MACsec frames, with some frames being descriptographed and others being cryptographed.

No.	Time	Source	Destination	Protocol	Length	Info
25	50.006125572	D-LinkCo_90:bd:08	Nearest	EAPOL-MKA	150	
26	51.961456927	HewlettP_df:d1:64	Nearest	EAPOL-MKA	150	
27	55.006737971	D-LinkCo_90:bd:08	Nearest	EAPOL-MKA	150	
28	56.963295965	HewlettP_df:d1:64	Nearest	EAPOL-MKA	150	
29	60.007346634	D-LinkCo_90:bd:08	Nearest	EAPOL-MKA	150	
30	61.965584009	HewlettP_df:d1:64	Nearest	EAPOL-MKA	150	
31	65.007391555	D-LinkCo_90:bd:08	Nearest	EAPOL-MKA	150	
32	66.007391554	HewlettP_df:d1:64	Nearest	EAPOL-MKA	150	
33	70.008537244	D-LinkCo_90:bd:08	Nearest	EAPOL-MKA	150	
34	71.960191255	HewlettP_df:d1:64	Nearest	EAPOL-MKA	150	
35	72.154966893	Tp-LinkT_03:28:ec	HewlettP_df:d1:64	MACSEC	92	MACsec frame
36	72.154966893	162.110.0.50	192.168.0.35	UDP	60	3423 → 5342 Len=18
37	75.009122088	D-LinkCo_90:bd:08	Nearest	EAPOL-MKA	150	
38	76.971480882	HewlettP_df:d1:64	Nearest	EAPOL-MKA	150	
39	80.009715248	D-LinkCo_90:bd:08	Nearest	EAPOL-MKA	150	
40	81.976954599	HewlettP_df:d1:64	Nearest	EAPOL-MKA	150	
41	83.042965666	Tp-LinkT_03:28:ec	HewlettP_df:d1:64	MACSEC	92	MACsec frame
42	83.042965666	162.110.0.50	192.168.0.35	UDP	60	3423 → 5342 Len=18
43	86.012912255	D-LinkCo_90:bd:08	Nearest	EAPOL-MKA	150	
44	86.975533691	HewlettP_df:d1:64	Nearest	EAPOL-MKA	150	
45	90.010971756	D-LinkCo_90:bd:08	Nearest	EAPOL-MKA	150	
46	91.977727238	HewlettP_df:d1:64	Nearest	EAPOL-MKA	150	
47	95.011443576	D-LinkCo_90:bd:08	Nearest	EAPOL-MKA	150	
48	95.011443576	Tp-LinkT_03:28:ec	HewlettP_df:d1:64	MACSEC	92	MACsec frame
49	95.011443576	162.110.0.50	192.168.0.35	UDP	60	3423 → 5342 Len=18
50	99.821761154	HewlettP_df:d1:64	Nearest	EAPOL-MKA	150	
51	99.821761154	Tp-LinkT_03:28:ec	HewlettP_df:d1:64	MACSEC	92	MACsec frame
52	99.821761154	162.110.0.50	192.168.0.35	UDP	60	3423 → 5342 Len=18
53	100.012027028	D-LinkCo_90:bd:08	Nearest	EAPOL-MKA	150	
54	101.981863429	HewlettP_df:d1:64	Nearest	EAPOL-MKA	150	
55	105.012608867	D-LinkCo_90:bd:08	Nearest	EAPOL-MKA	150	
56	105.012608867	HewlettP_df:d1:64	Nearest	EAPOL-MKA	150	
57	119.013178394	D-LinkCo_90:bd:08	Nearest	EAPOL-MKA	150	

Figura 33 - Wireshark no Autenticador, mensagens recebidas e enviadas entre as interfaces.

Na Figura 34 os pacotes MACsec são recebidos (1, 3, 5, 7) e são descriptografados, recuperando-se o pacote UDP (2, 4, 6, 8) transmitido pelo Cliente A (1, 2, 3, 4 na Figura 32). Os demais pacotes são pacotes periódicos do protocolo MKA.

The image shows a Wireshark capture window titled '*enp0s25 and macsecenp0s25'. It displays a sequence of MACsec frames (92 bytes) and UDP packets (60 bytes) between interfaces. The capture shows a mix of EAPOL-MKA and MACsec frames, with some frames being descriptographed and others being cryptographed.

No.	Time	Source	Destination	Protocol	Length	Info
31	65.007160061	D-LinkCo_90:bd:d8	Nearest	EAPOL-MKA	150	
32	66.962359242	HewlettP_df:d1:64	Nearest	EAPOL-MKA	150	
33	70.007657331	D-LinkCo_90:bd:d8	Nearest	EAPOL-MKA	150	
34	71.964100884	HewlettP_df:d1:64	Nearest	EAPOL-MKA	150	
35	75.008200486	D-LinkCo_90:bd:d8	Nearest	EAPOL-MKA	150	
36	76.966280351	HewlettP_df:d1:64	Nearest	EAPOL-MKA	150	
37	80.008687071	D-LinkCo_90:bd:d8	Nearest	EAPOL-MKA	150	
38	81.968025964	HewlettP_df:d1:64	Nearest	EAPOL-MKA	150	
39	85.009195166	D-LinkCo_90:bd:d8	Nearest	EAPOL-MKA	150	
40	86.970171140	HewlettP_df:d1:64	Nearest	EAPOL-MKA	150	
41	87.155626916	Tp-LinkT_03:28:ec	HewlettP_df:d1:64	MACSEC	92	MACsec frame
42	87.155626916	162.110.0.50	192.168.0.35	UDP	60	3423 → 5342 Len=18
43	90.009730290	D-LinkCo_90:bd:d8	Nearest	EAPOL-MKA	150	
44	91.971940988	HewlettP_df:d1:64	Nearest	EAPOL-MKA	150	
45	95.010236290	D-LinkCo_90:bd:d8	Nearest	EAPOL-MKA	150	
46	96.974070379	HewlettP_df:d1:64	Nearest	EAPOL-MKA	150	
47	98.043336659	Tp-LinkT_03:28:ec	HewlettP_df:d1:64	MACSEC	92	MACsec frame
48	98.043336659	162.110.0.50	192.168.0.35	UDP	60	3423 → 5342 Len=18
49	100.010695986	D-LinkCo_90:bd:d8	Nearest	EAPOL-MKA	150	
50	101.975823395	HewlettP_df:d1:64	Nearest	EAPOL-MKA	150	
51	105.011212951	D-LinkCo_90:bd:d8	Nearest	EAPOL-MKA	150	
52	106.977962005	HewlettP_df:d1:64	Nearest	EAPOL-MKA	150	
53	110.011684799	D-LinkCo_90:bd:d8	Nearest	EAPOL-MKA	150	
54	110.811391522	Tp-LinkT_03:28:ec	HewlettP_df:d1:64	MACSEC	92	MACsec frame
55	110.811391522	162.110.0.50	192.168.0.35	UDP	60	3423 → 5342 Len=18
56	111.979754691	HewlettP_df:d1:64	Nearest	EAPOL-MKA	150	
57	114.827205753	Tp-LinkT_03:28:ec	HewlettP_df:d1:64	MACSEC	92	MACsec frame
58	114.827205753	162.110.0.50	192.168.0.35	UDP	60	3423 → 5342 Len=18
59	115.012179764	D-LinkCo_90:bd:d8	Nearest	EAPOL-MKA	150	
60	116.981892184	HewlettP_df:d1:64	Nearest	EAPOL-MKA	150	
61	120.012692888	D-LinkCo_90:bd:d8	Nearest	EAPOL-MKA	150	
62	121.982526971	HewlettP_df:d1:64	Nearest	EAPOL-MKA	150	

Figura 34 - Mensagens recebidas/enviadas pelo Cliente B.