October 2019

MEDIA ACCESS CONTROL SECURITY KEY DISTRIBUTION USING BLOCKCHAIN AND PUBLIC KEY CRYPTOGRAPHY

Niranjan M. M

Nagaraj Kenchaiah

Vijay Kothamasu

MEDIA ACCESS CONTROL SECURITY KEY DISTRIBUTION USING BLOCKCHAIN AND PUBLIC KEY CRYPTOGRAPHY

AUTHORS:
Niranjan M M
Nagaraj Kenchaiah
Vijay Kothamasu

ABSTRACT

Techniques are described herein for sophisticated authentication and encryption methods that do not require manual configuration or a centralized server. These techniques use blockchain and public key cryptography to exchange Media Access Control security (MACsec) keys securely between router links and thereby by avoid manual configuration for MACsec. This simplifies existing MACsec key configuration approaches, which use static security mode with manually-configured security keys and dynamic security mode with keys distributed from a centralized Authentication, Authorization, and Accounting (AAA) server over Extensible Authentication Protocol Transport Layer Security (EAP-TLS).

DETAILED DESCRIPTION

For simplicity, the Asymmetric Encryption (AE) algorithm discussed herein is represented as the key generation, encryption, and decryption algorithms (KeyGeneration, Encryption, Decryption) respectively. Similarly, the Digital Signature (DS) algorithm discussed herein is represented as the key generation, signature, and verification algorithms (KeyGeneration, Signature, Verification) respectively. The pair of keys used in the algorithms are the Public Key (PK) and private (Secret) Key (SK). These are basic requirements for public key cryptography, where SK is kept secret and PK is broadcast over the network.

Media Access Control security (MACsec) is an Institute of Electrical and Electronics Engineers (IEEE) 802.1AE standard authenticating and encrypting mechanism between MACsec-capable devices. MACsec allows encryption and decryption of data packets at line-rate, and can be used on the links connecting routers. There can be many inter-connected routers depending on the specific deployment. Each of these links connecting routers must be manually programmed with identical key information (identical

Connectivity Association Key (CAK) and identical Connectivity Association Key Name (CKN)). The CAK is the secret key and should not be compromised.

There are many issues with manually programming these MACsec keys. Manual configuration is time consuming since each port has to be programmed independently and can be error prone. Also, the user must keep track of which ports on two routers are physically connected and ensure they are programmed with identical key information. Moreover, any mismatch in MACsec key information can result in the network link going down which may affect the performance and resiliency of the network. Moreover, existing MACsec key configuration uses static security mode wherein security keys are configured manually, and dynamic security mode wherein keys are distributed from a centralized Authentication, Authorization, and Accounting (AAA) over Extensible Authentication Protocol Transport Layer Security (EAP-TLS).

MACsec currently supports two security modes: pre-shared key security mode and dynamic key security mode. In pre-shared key security mode, the user needs to configure the same key across nodes manually. There are two forms of pre-shared key security mode. The first form is Static Secure Association Key (SAK) Security Mode, where the user needs to manually configure two SAKs across nodes of a point-to-point Ethernet link. The second form is static CAK security mode, where the user needs to manually configure the CKN and the CAK across nodes of the point-to-point Ethernet link. Static CAK security mode ensures security by frequently refreshing to a new random security key and by only sharing the security key between the two devices on the MACsec-secured point-to-point link. However, this method is not considered robust because the user needs to configure the CAK and the CKN.

In dynamic key security mode, the EAP-TLS mutual authentication method is used to authenticate a supplicant (node/router) and a server (e.g., a Remote Authentication Dial-In User Service (RADIUS) server) by exchanging certificates. The Master Secret Key (MSK) and session Identifier (ID) are populated and distributed. Dynamic key security mode avoids the aforementioned disadvantage in pre-shared key security mode, but the RADIUS server must support the EAP-TLS authentication framework. For example, the server must use IEEE 802.1X authentication, even when it is multiple hops from deployment.

Thus, one method requires manual intervention and another requires a centralized server to distribute the MACsec keys. To overcome these disadvantages, a sophisticated authentication and encryption method is needed that does not require manual configuration or a centralized server. Accordingly, techniques are described herein which simplify this process by avoiding manual configuration while providing secure and decentralized MACsec key distribution between routers. In particular, manual intervention is removed and key distribution is decentralized. Blockchain and public key cryptography (also referred to as asymmetric cryptography) may be used to generate and securely distribute MACsec keys among routers, which requires no manual configuration and provides a decentralized (distributed) approach that does not need a centralized authority.

As described herein, routers become peers when a link is established between them. Mutual peer authentication may take place as follows based on blockchain (or any suitable distributed ledger). Each router may generate an SK/PK pair using public key cryptography. The SK may be kept secret by the device that generates and sends the PK to the peer. For authentication, the sender may use its SK to sign a message and send the message to the receiver. The receiver may in turn use the PK shared earlier by the sender in the received message to verify the sender.

For simplicity, notation for the signature is "DS.Signature(Private Key of the Sender, Identification of the Sender)." Similarly, notation for the verification is "DS.Verification(Public Key of the Sender, Identification of the Sender)." Furthermore, a transaction having metadata is created and added to the blockchain to avoid reply attacks, malicious key updates, etc.

In the MACsec context, routers are defined which are participating in the point-to-point link as shown below:

R1 = (ID_R1, PK_R1, SK_R1)

R2 = (ID_R2, PK_R2, SK_R2)

Here, ID_R1 and ID_R2 are the respective port/interface MAC addresses of router R1 and router R2, PK_R1 and PK_R2 are the respective PKs of the router link, and SK_R1 and SK_R2 are the respective SKs of the router link. Sender R1 signs using SK_R1 and router R2 performs verification using PK_R1. This may be repeated for every message sent

from router R1 to router R2. A similar operation may be performed in the opposite direction (from router R2 to router R1) as well.

All active histories of the routers, including operations performed when the routers participate and act in the network, are built into metadata of transactions on blockchains as T_R = (ID_R, PK_R, DS.Signature(SK_R, ID_R), KEY_UPDATE_R), where T_R is the transaction ID of the router, ID_R is the identity of the router (e.g., the port/interface MAC address of the router), PK_R is the PK of the router, SK_R is the SK of the router, and KEY_UPDATE_R is an encrypted tuple added to the transaction used for key updating (e.g., in case of reconnect, reboot of the router, etc.). Here, ID_R is verified by the algorithm DS.Verification(PK_R, DS.Signature(SK_R, ID_R)) to prove that ID_R is linked with PK_R and is registered with the blockchain. SK_R is not added to the transaction directly, and instead the signature generated by DS.Signature() using SK_R and ID_R is populated in the transaction.

The transactions may include transactions of registered entities that not only effectively indicate the existences of those entities but also indicate the relationships connecting them. For router R1 these transactions may be defined as T_R1 = (MAC_R1, PK_R1, DS.Signature(SK_R1, MAC_R1), KEY_UPDATE_R1), where DS_Signature(SK_R1, MAC_R1) is the digital signature signed by router R1 using its SK (which is used for authentication), and KEY_UPDATE_R1 is the AE.Encryption(PK_R2, nonce, DS.Signature(SK_R1, nonce)) (which is the encrypted tuple added to the transaction used for key updating).

Similarly, for router R2 these transactions may be defined as T_R2 = (MAC_R2, PK_R2, DS.Signature(SK_R2, MAC_R2), KEY_UPDATE_R2), where DS_Signature(SK_R2, MAC_R2) is the digital signature signed by router R2 using its SK, which is used for authentication, and KEY_UPDATE_R2 is the AE.Encryption(PK_R1, nonce, DS.Signature(SK_R2, nonce)) (which is the encrypted tuple added to the transaction used for key updating).

Table 1 below illustrates an example distributed ledger that has been updated to include these transactions.

No	Transaction ID	Source ID	Public Key	Digital Signature	Key Update Tuple	Destination ID
1	T_R1	MAC_R1	PK_R1	Signature_R1	KEY_UPDATE_R1	MAC_R2
2	T_R2	MAC_R2	PK_R2	Signature_R2	KEY_UPDATE_R2	MAC_R1

Table 1

Optionally, Link Layer Discovery Protocol (LLDP) may be used to identify the peer port (node-id and port-id). Local node-id, local port-id, peer node-id, and peer port-id may be used to create a key-name that is unique within the deployment and add to the transaction T_R and hence to the blockchain.

Maintaining metadata of transactions on the blockchain may help overcome reply attacks and identity-verification challenges in the key updating phase. For example, suppose a compromised router launches a reply attack to a legitimate router by frequently sending its information. By auditing T_R on the blockchain, a router may refuse a replied request when the replied identity (ID_R and PK_R) is already part of the blockchain that stores timestamp and communication histories among the routers.

With respect to key updating, T_R also includes the tuple "KEY_UPDATE_R = AE.Encryption(PK_R2, nonce, DS.Signature(SK_R1, nonce))," which is for encryption using the public key of (destination) router R2. This includes a nonce selected by router R1 when the connection is first built. Specifically, when the same router R1 reconnects with router R2, router R2 may audit the transaction T_R on the blockchain. The encrypted tuple may be extracted from the T_R and generate an identity-verification challenge for router R1. If router R1 can successfully respond with the correct nonce that the R2 verifies using the encrypted value, the PK of the nonce may be equal to the value extracted from the logged connection transaction. If so, the verification is successful and router R2 continues to share MACsec keys CAK and CKN with router R1.

Communication between the routers R1 and R2 is secured by encrypting at sending router R1 using PK_R2 of receiving router R2 and decrypting at receiving router R2 with its SK_R2. Notation used herein for encryption is "Encryption_R1 = AE.Encryption(Public Key of R2 , DS.Signature(Private Key of R1))," and notation used herein for decryption is "Decryption_R2 = AE.Decryption(Private Key of R2, DS_Verification(Public Key of R1))." In addition to encryption and decryption, message authentication may also be performed by adding a signature at the sender and verifying at the receiver. For example,

the sender may add the signature using SK_R1, and this may be verified by the receiver using PK_R1.

Figure 1 below illustrates an example sequence diagram that shows the steps involved between routers to exchange MACsec keys using public key cryptography.

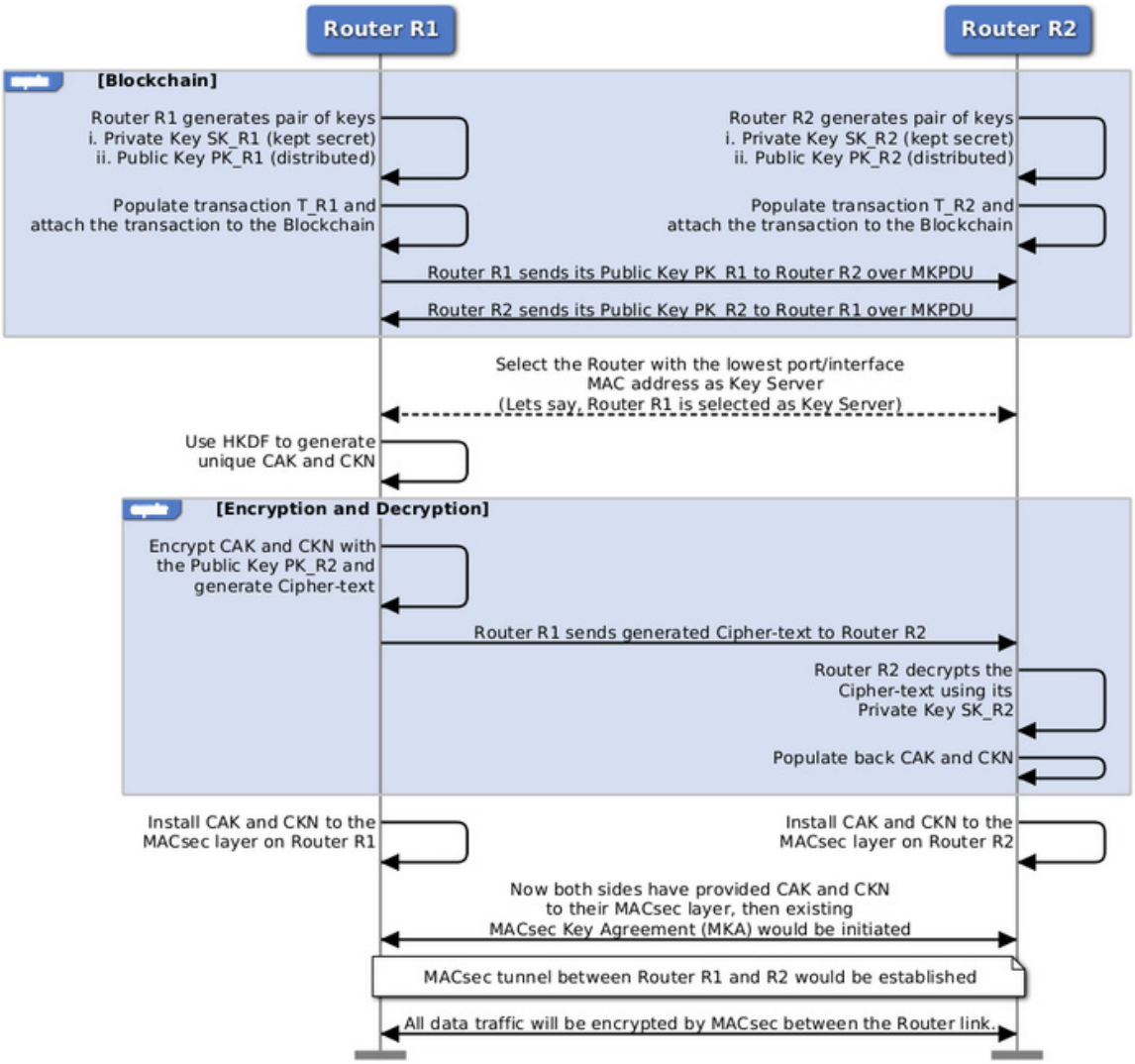

Figure 1

This encryption and decryption method may be used to send MACsec keys between routers. MACsec is used across nodes of a point-to-point Ethernet link. This example considers MACsec between routers, although it will be appreciated that similar operations are applicable between switches or between a host and a switch, for example. When MACsec is enabled on router R1, it generates a pair of keys: SK1 and PK1. SK1 is kept secret by router R1 and PK1 is sent to peer router R2 over an existing MACsec Key

Agreement Protocol Data Unit (MKPDU). The transaction may be populated at R1 as T_R1 = (ID_R1, PK_R1, DS.Signature(SK_R1, ID_R1), KEY_UPDATE_R1), and the transaction T_R1 may be attached to the blockchain and hence to the public ledger.

Similarly, router R2 generates a pair of keys: SK2 and PK2. SK2 is kept secret by router R2, and PK2 is sent to peer router R1 over an existing MKPDU. The transaction is populated at router R2 as T_R2 = (ID_R2, PK_R2, DS.Signature(SK_R2, ID_R2), KEY_UPDATE_R2). The transaction T_R2 is attached to the blockchain and hence to the public ledger.

The router of the point-to-point link with the lowest port/interface MAC address may be selected as the key server. In this example, router R1 is selected. Hash-based Message Authentication Code (HMAC) Key Derivation Function (HKDF) may be used to generate a unique CAK and CKN on the selected key server R1 for this router link. HKDF is used to derive an encryption key from a pass phrase. Initially, HKDF creates a Pseudorandom Key (PRK) using a pass phrase and a salt value, in order to produce an HMAC hash function (such as HMAC-SHA256), and along with a salt value. Next the PRK output is used to produce a key of the required length. If a 16-byte output (32 hex characters) is generated, it will generate a 128-bit key, and for a 32-byte output (64 hex characters), it will generate a 256-bit key.

The CAK and CKN may be encrypted with the PK_R2 to generate cipher-text, which is sent to router R2. Router R2 may decrypt the cipher-text using SK_R2 to populate the CAK and CKN. Now both routers R1 and R2 have the CAK and CKN, and install the CAK and CKN to the MACsec layer. Once both sides have provided the CAK and CKN to their MACsec layer, the existing MACsec Key Agreement (MKA) may be initiated on the router link. Upon successful MKA session creation, all data traffic may be encrypted via MACsec between the router links.

The techniques described herein require no manual configuration for MACsec key generation and distribution. There is also no need for a RADIUS server, which is required for EAP-TLS. Similar methods may be used for a MACsec tunnel/session between switches or between the host and the switch, or data centers between fabric links.

In summary, techniques are described herein for sophisticated authentication and encryption methods that do not require manual configuration or a centralized server. These

techniques use blockchain and public key cryptography to exchange MACsec keys securely between router links and thereby by avoid manual configuration for MACsec. This simplifies existing MACsec key configuration approaches, which use static security mode with manually-configured security keys and dynamic security mode with keys distributed from a centralized AAA server over EAP-TLS.