Stateless Encryption Scheme of Enhanced Encapsulating Security Payload (EESP)

draft-xia-ipsecme-eesp-stateless-encryption-00

https://datatracker.ietf.org/doc/draft-xia-ipsecme-eesp-stateless-encryption/

Liang Xia, Weiyu Jiang, Huawei

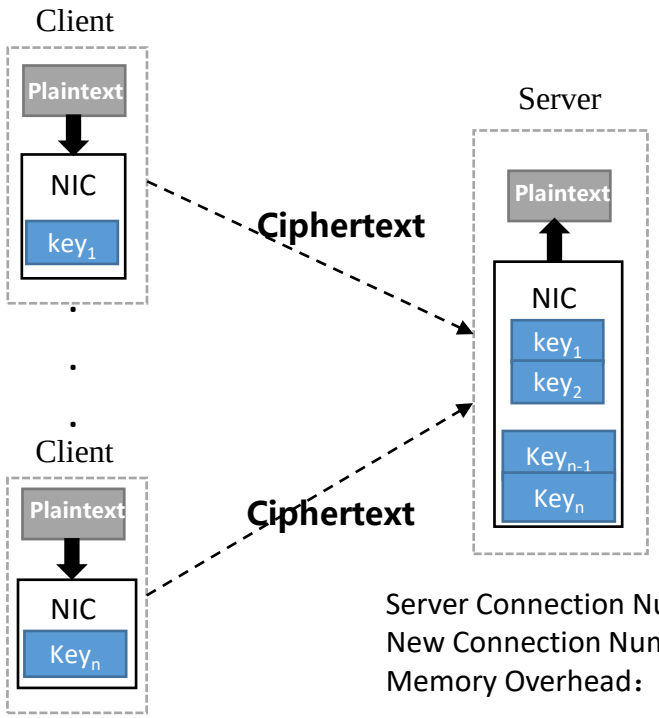
Introduction

- **Basically, stateless encryption here means have only one group key/master key, and perform KDF to get the session key from this group key/master key, together with various combination of security context, and no need IKEv2 key negotiation.**

- This draft first introduces **several use cases** for stateless encryption, analyzes and compares some **existing stateless encryption schemes in the industry**, and then attempts to propose **a general and flexible stateless encryption scheme** based on the **summarized requirements:**
  - **4 Use cases:** C-S Model of General Computing Network, Cluster Communication of HPC Network, NIC/DPU Pool of General Computing Network, AI Computing Network
  - **Existing industry stateless encryption schemes:** Google PSP (PSP Security Protocol), UEC (Ultra Ethernet Consortium) TSS (Transport Security Sublayer)
  - **Summarized requirements:** group key, security enhanced group key, flexible and fine-grained security session isolation, stateless encryption proxy…
- **But, we need a precise/wide consensus definition of stateless encryption:** minimum security state, less security state…, but definitely not totally stateless.

Use Case 1 – C-S Model of General Computing Network: Hardware Memory Limitation, Stateless Encryption for Server Packet Receiving Direction Is Required

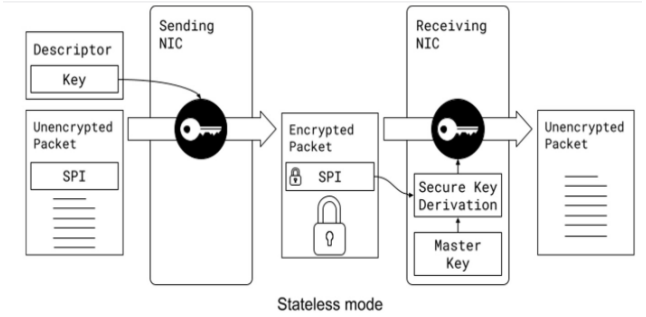
C-S Model of General Computing

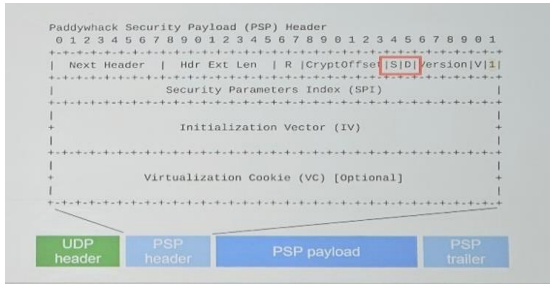


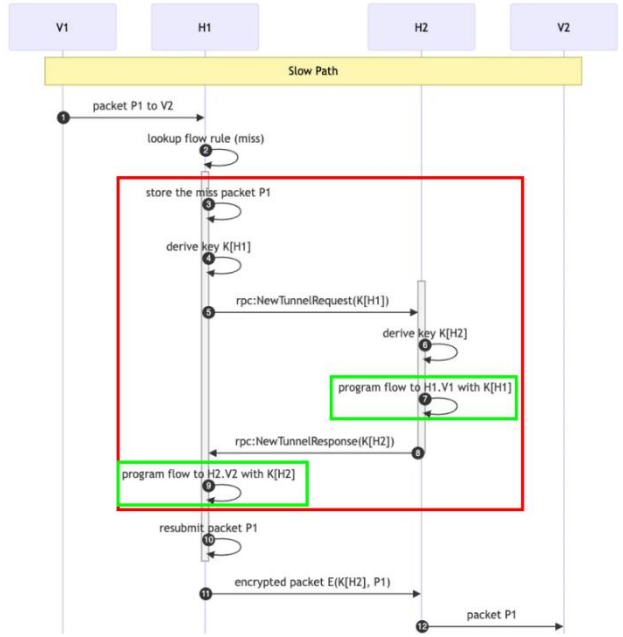
Server Connection Number:**10M+**
New Connection Number per second:**100K+**
Memory Overhead：**256B*2*10M=5GB**

Summary:
1） Hardware supports session-based encryption and decryption, and data keys of different sessions are isolated;
2） The server needs to maintain the security connection context between the server and a large number of clients, and **the hardware cannot store the context**.
3） The client and server do not belong to the same trusted domain.

Google PSP Solution





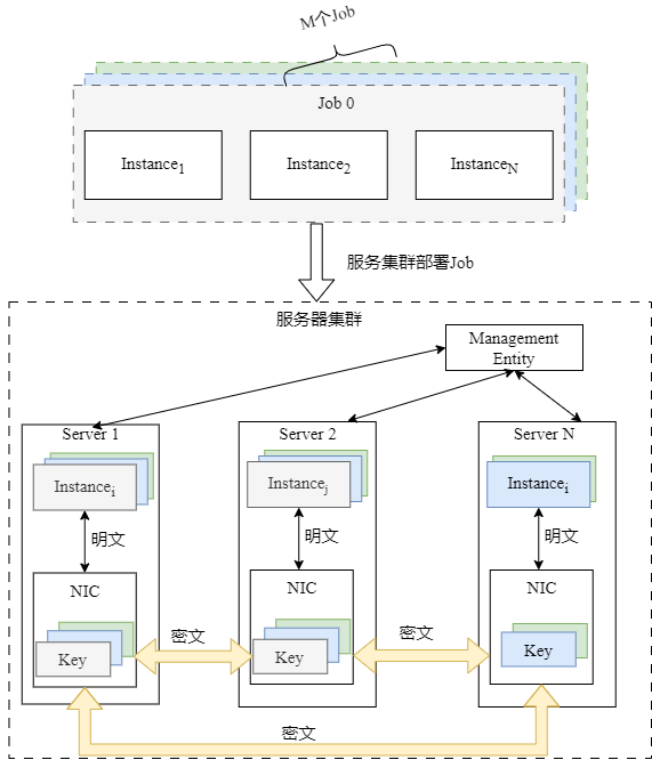


Google PSP Technical Features
1) One-sided key derivation, master key is for receiving direction, so saving half of total session contexts;
2) The master key is not shared (Key leakage affects only one server);
(3) The session key is distributed through out of band channel;

Disadvantage: When a large number of new sessions are created, the first packet is negotiated along the slow path in real time, causing performance degrade.

Use Case 2 -- Cluster Communication of HPC Network: Massive Secure Sessions Between Job Instances Require an Efficient Key Management Mechanism. Stateless Encryption for both 2 Packet Directions is Required

Cluster Communication in General Computing



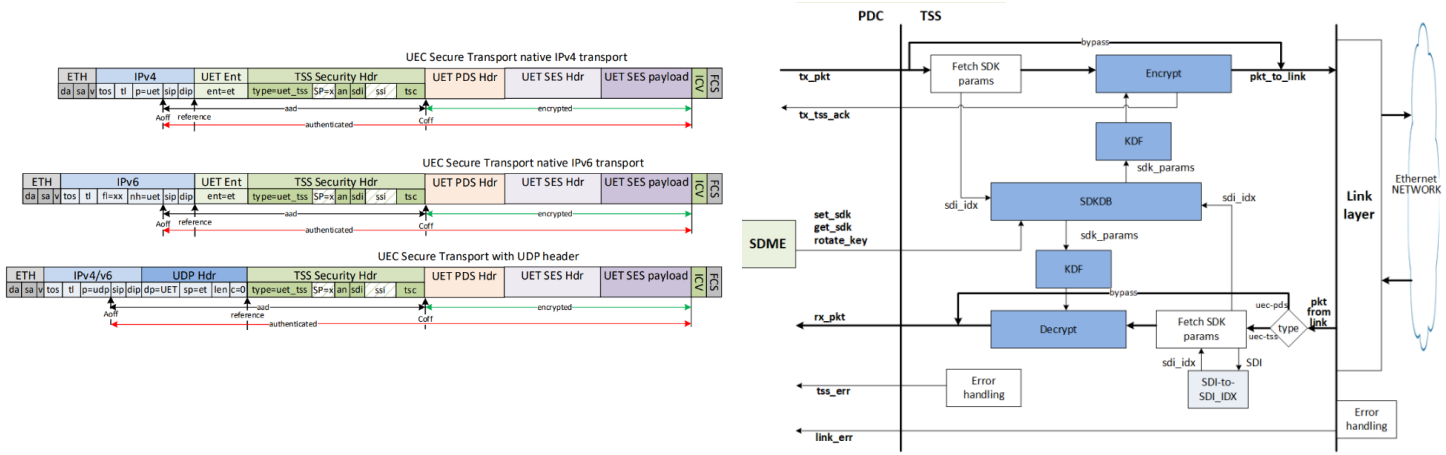
Summary:

1) Encrypted communication is required between different instances of large-scale HPC jobs, the security session number is at the scale of $O(M * N^2)$;

2) **An efficient key management mechanism is required to solve the problem of large-scale security sessions, IKEv2 is not suitable.**

3) **all instances of a HPC job belong to one trusted domain.**

UEC TSS (Transport Security Sublayer) Solution



UEC TSS Technical Features:

1) The group key is shared in the same trusted domain and stored in the SDKDB (Key leakage affects the entire security domain), and is for both receiving and sending directions, so saving all security session contexts;

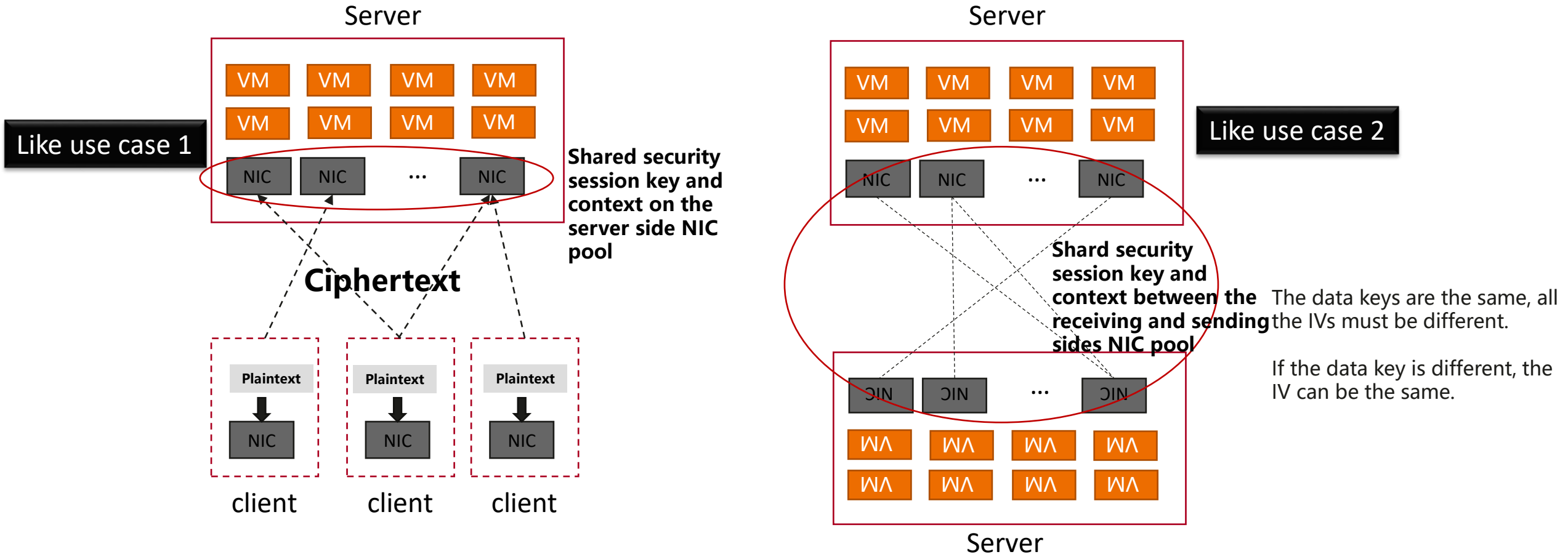
2) The communication parties do not need to store data keys, and the increase of the number of connections does not affect the number of security contexts.

3) Data keys are derived on the fast path, and the first packet delay is small.

4) The context content can be generated based on the SSI/SIP/DIP field. Although the context content is flexible, the calculation overhead increases.

5) Data keys can be updated through the TSC.epoch.

Use Case 3 – NIC/DPU Pool of General Computing Network: Difficult to Collaborate and Share Security Session Key and Context. Stateless Encryption for both 2 packet directions is Required



The data keys are the same, all the IVs must be different.

If the data key is different, the IV can be the same.

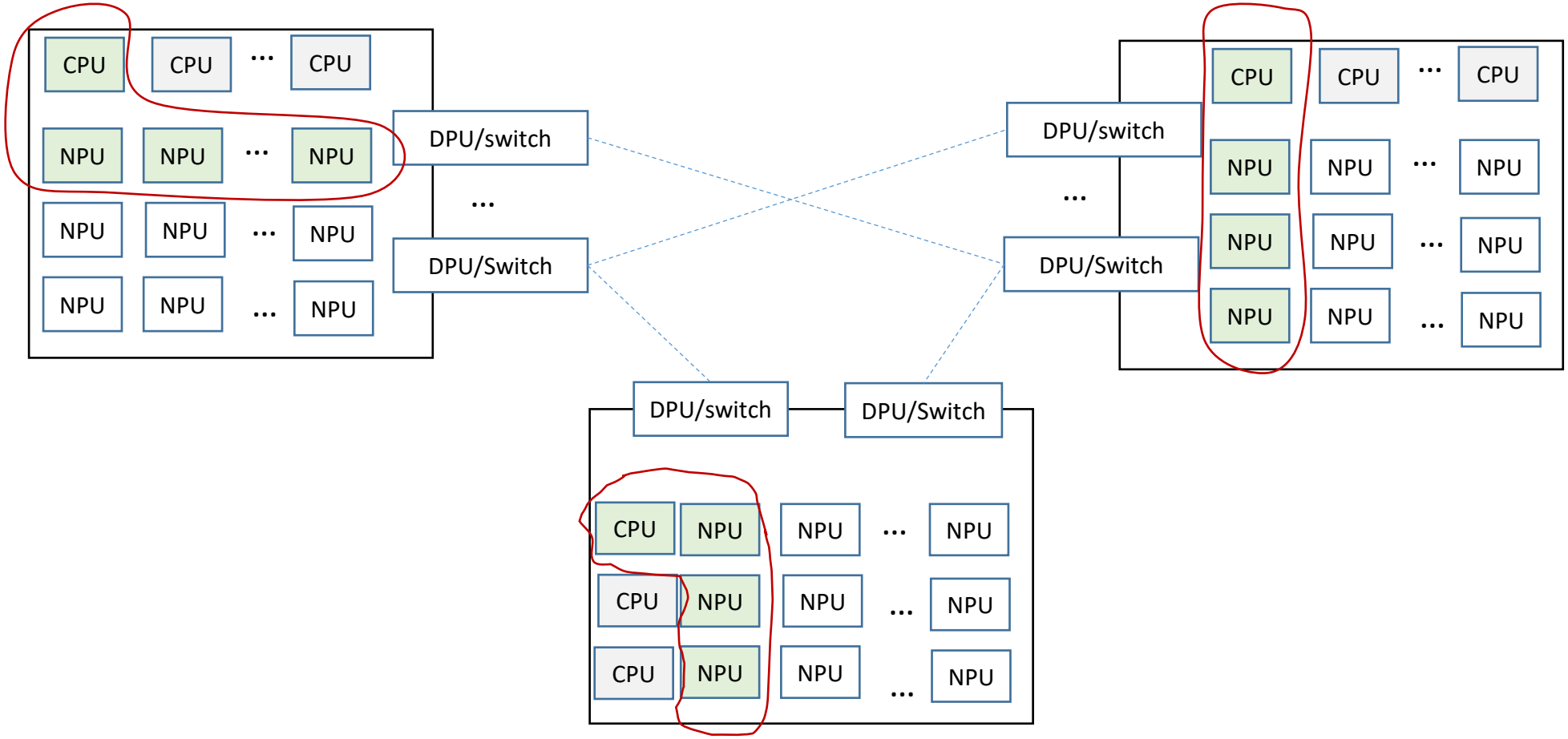
Summary: **During a request or response, traffic is processed by any NIC/DPU in the pool.** The NIC must be able to encrypt or decrypt subsequent traffic.

2 Challenges:

1) without group key, session key and context need to be synchronized in the pool: so that pooled NICs can correctly encrypt & decrypt data, which introduce complex process and consumes memory resources.

2) **IV collaboration: Different IVs must be generated for multiple NICs. Otherwise, security problems may occur. However, real-time IV synchronization between multiple NICs causes a delay.**

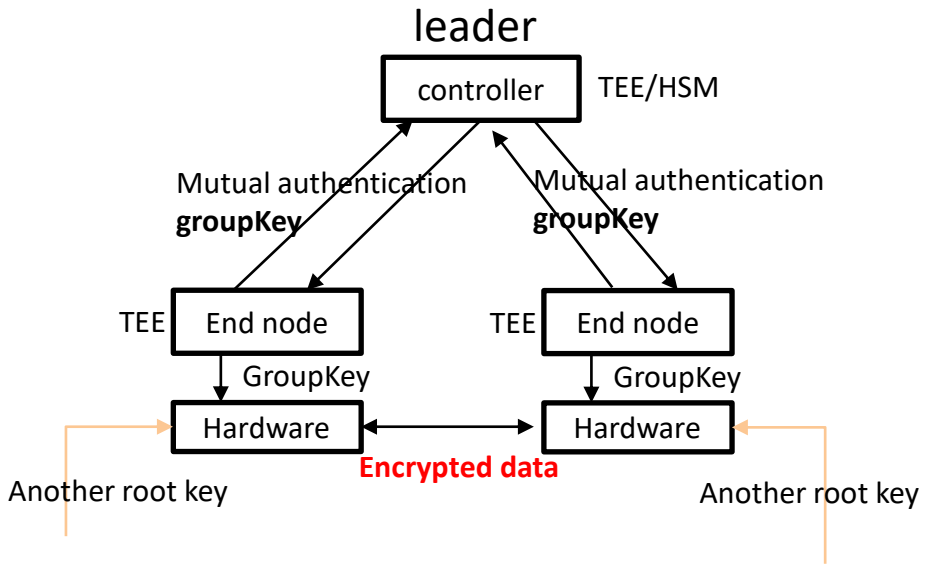
Use Case 4 – AI Computing Network: Fine-grained key management is complex for single/multi-task cross-node communication – Stateless Encryption is required to solve the problem of frequent negotiation and synchronization of fine-grained keys with IKEv2



Summary: Chips that perform the same task belong to the same security **domain. Encrypted communication is required for cross-chassis interconnection between chips.**

Challenge/Requirement: **trusted domain, multi-tenant, multi-task, and virtual pod scenarios require multi-level (aka, fine-grained) group keys and fine-grained session key derivation (SIP, DIP, 5-turple, DeviceID, VMID, taskID) and encrypted traffic isolation**.

Group Key Management and Distribution



Even if the group key leaks, another root key can also provide another protection!

Why we need group key, because without it:

1. End-to-end key negotiation incurs high overhead and latency > **2RTT; not friendly to latency-sensitive and short-connection services**.
2. Real-time distribution of negotiated keys to hardware on-chip storage **also introduces some latency**.
3. The large number of end-to-end **keys cannot be stored in hardware.**

Group Key Risk evaluation:

Group key may be used by many nodes, when it **leaks**, Security risks can spread.

Session key without fine-grained isolation will increase the key leakage influence scope

Therefore, we should carefully give a secure group key distribution mechanism, with higher security by introducing new security features:

1、two or more root keys of group key should be adopted.

2、more flexible and fine-grained security session isolation with the fine-grained context definition

3、Context information update by itself can enable session key update

Comparison of UEC TSS and PSP

	UEC TSS	**PSP**
Key deriviation	KDF(SDKDB-key, labelCluster+TSC.epoch+TSC.counter+**SIP/DIP/SSI**+L) —256 bit key	KDF is based on CMAC(CKDF not HKDF)：KDF(K,**SPI**） --256 or 128 bit key
Keys update	1. AN(**one group/master key: old key, new key,** key update every 24 hours)； 2. **Update TSC-Epoch-TSC.counter**, update keys	**One group/master key updates when SPI rotation**; SA lifetime and is typically set to 24 hours,
IV Construction	IVmask; Epoch； IV=(((pkt.ssi OR pkt.sdi) ‖pkt.tsc) XOR IVMASK)（96bit）	IV=timestamp(8B)+SPI(4B)
Flexibility	**A-offset**, c-offset; Fields that are not carried with the package, search the SDKDB table, one SDI one profile	Only support crypto offset in every packet
Replay attack detection	No, depends on upper layer	No, depends on upper layer
Master key	Secure domain	Server

Requirements Summary – part 1

- S**upport nodes within a trusted trust domain to share the same master key;**

- **Group key supports multi-level combination design. In a trust domain, the master key is composed of multiple root keys of different types and levels**, such as trust domain root key, tenant root key, task group root key, etc. This enhances the overall security of the master key and supports fine-grained encryption traffic isolation (e.g., all nodes in a trust domain, nodes of the same tenant in a trust domain, nodes of the same computing task in a trust domain, etc.);

- Different types of root keys have different security levels and lifecycles, and corresponding **key rotation mechanisms** need to be defined. The master key update will trigger the data key update;

- The key rotation of each type of root key should support **multiple key rotations**, such as pre_key, current_key, and next_key, to **support rapid rotation while ensuring that real-time encryption and decryption are not affected**;

Requirements Summary – part 2

- The key derivation of the data key is based on the group key, context, and KDF. **KDF** must support packet-by-packet data key calculation in most cases (except when the session key is cached in memory), which requires extremely high performance, **must support cryptographically secure, hardware-concurrent high-performance algorithms; maybe more than one KDF algorisms should support.**

- To support real-time derivation of the session key**, context information and IV information need to be carried with the message**. To support different scenarios and different granularities of session key calculation and encryption traffic isolation (based on stream, based on source IP, based on source ID, etc**.), multiple combinations of context and IV need to be supported**, and different combination algorithms need to be distinguished through specific fields in the message;

- **Context information enables dynamic updates of the session key**, such as carrying an epoch in the context. When the epoch changes, the session key is also refreshed accordingly;

- It is necessary to **support encryption proxy capabilities across trusted domains**. At the edge nodes across trust domains (such as DPU, Switch, etc.), support for master keys and stateless encryption of two trust domains (local trust domain and global trust domain) is required, and proxy conversion of message encryption and decryption between the two trust domains must be completed.

Thanks