
東海大學
TUNGHAI UNIVERSITY
1955


DSNSL
資料庫與網路安全實驗室
Database System and Network Security Lab

5G 認證流程

以 *Mininet* 拓樸，*Ryu* 控制

St425

0 Install

在 *Virtual Box* 上建立 *Linux* 虛擬環境，

ubuntu-18.04.5-desktop-amd64.iso / ubuntu-20.04.2.0-desktop-amd64.iso

使用 *Mininet* 拓樸，使用 *Ryu* 控制傳輸，模擬 *5G* 認證過程。

環境：

Mac(IOS): *Big Sur 11.1*

App: *Virtual Box 6.1.16 r140961*

Virtual Box:

RAM: *8192 MB*

虛擬硬碟大小: *20.0 GB*

環境：

Win: *Win10*

App: *Virtual Box 6.1.16 r140961*

Virtual Box:

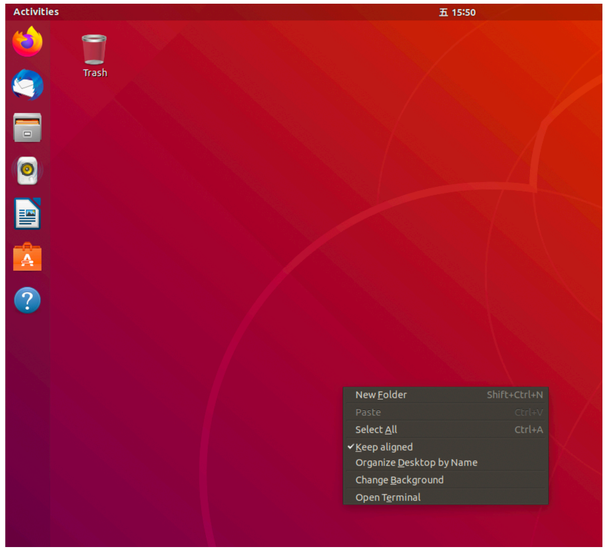
RAM: *8192 MB*

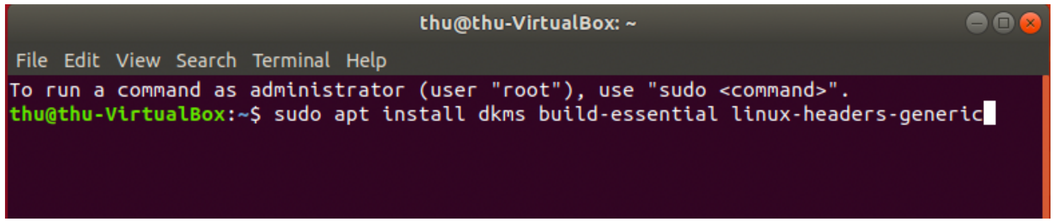
虛擬硬碟大小: *20.0 GB*

1 Copy & Paste

1 Install
Ubuntu 複製貼上

進入 *Ubuntu* 內，為了方便能從 *Mac* 到 *Virtual Box(Ubuntu)* 複製貼上，請進行以下操作。



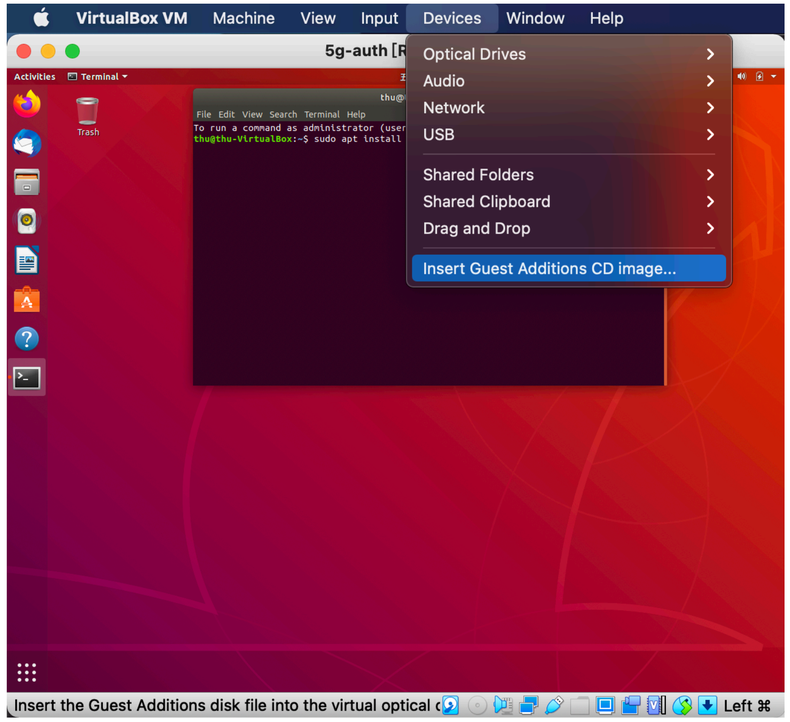


Right Click & Select: open Terminal

$ sudo apt install dkms build-essential linux-headers-generic

1 Install

Ubuntu 複製貼上



VirtualBox Menu > Devices > Insert Guest Additions CD images …

1 Install
Ubuntu 複製貼上

$ *shutdown -f now*

VirtualBox > 設定 > 一般 > 進階 > 共用剪貼簿 > 雙向

——*Restart Ubuntu*——

即可從 *Mac* 到 *Virtual Box(Ubuntu)* 複製貼上

2

Mininet
Ryu

2 Mininet + Ryu

Mininet install

$ *sudo apt-get update*

$ *sudo apt-get upgrade*

$ *sudo apt-get install -y git*

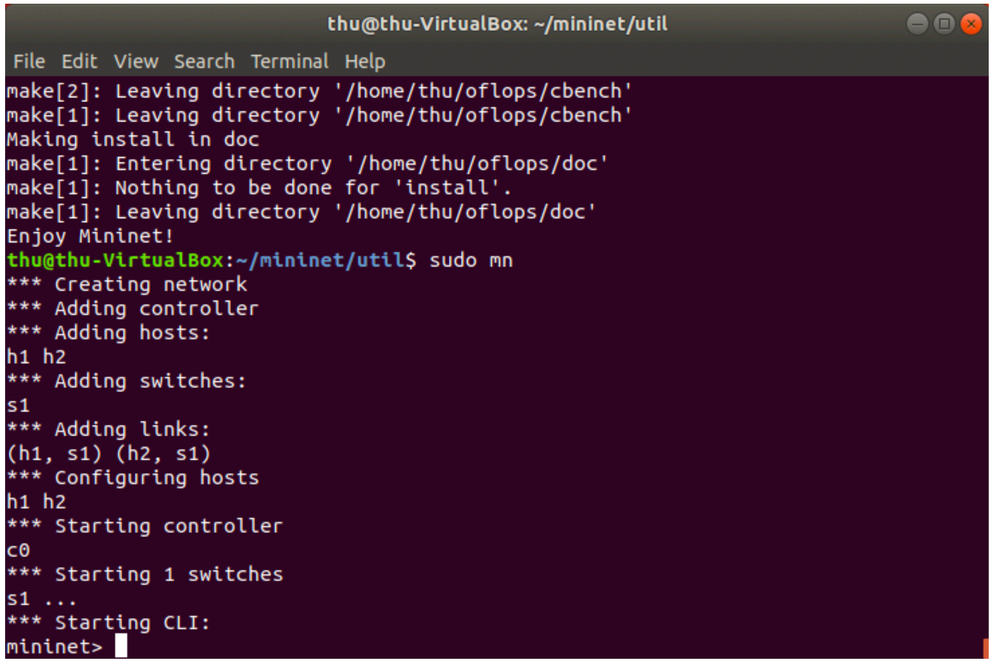
$ *git clone git://github.com/mininet/mininet*

$ *cd mininet/util*

$ *sudo ./install.sh –a*

$ *sudo mn*

測試 Mininet 安裝是否成功



mininet> exit
即可退出

2 Mininet + Ryu

Ryu install

```
$ cd ~
$ sudo apt-get update
$ sudo apt-get install libxml2-dev libxslt1-dev python-pip python-eventlet python-routes
  python-webob python-paramiko
$ sudo pip install msgpack-python eventlet==0.15.2
$ sudo pip install six --upgrade
$ sudo pip install oslo.config q --upgrade


$ sudo apt install python3-pip
```

2 Mininet + Ryu

Ryu install

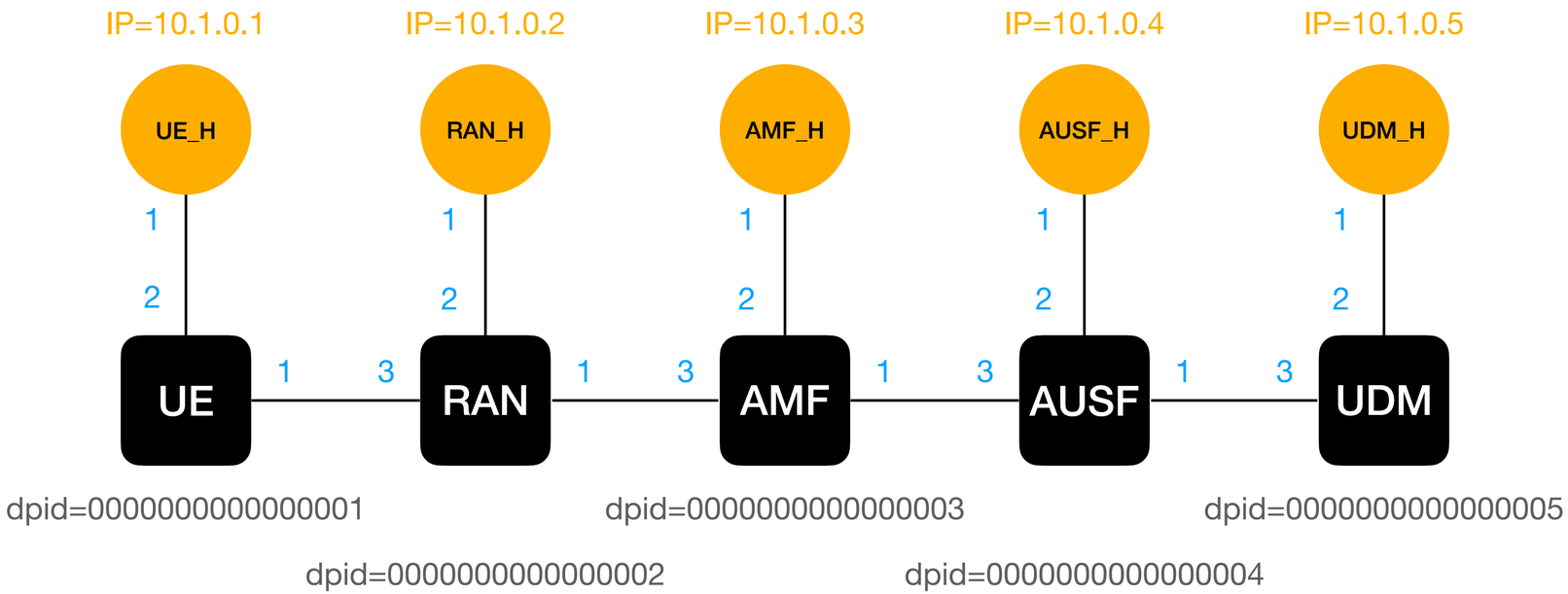
```
$ cd ~
$ git clone https://github.com/faucetsdn/ryu.git
$ cd ryu
$ pip3 install .
$ sudo apt install python-ryu


$ ryu-manager --verbose ryu.app.ofctl_rest
```

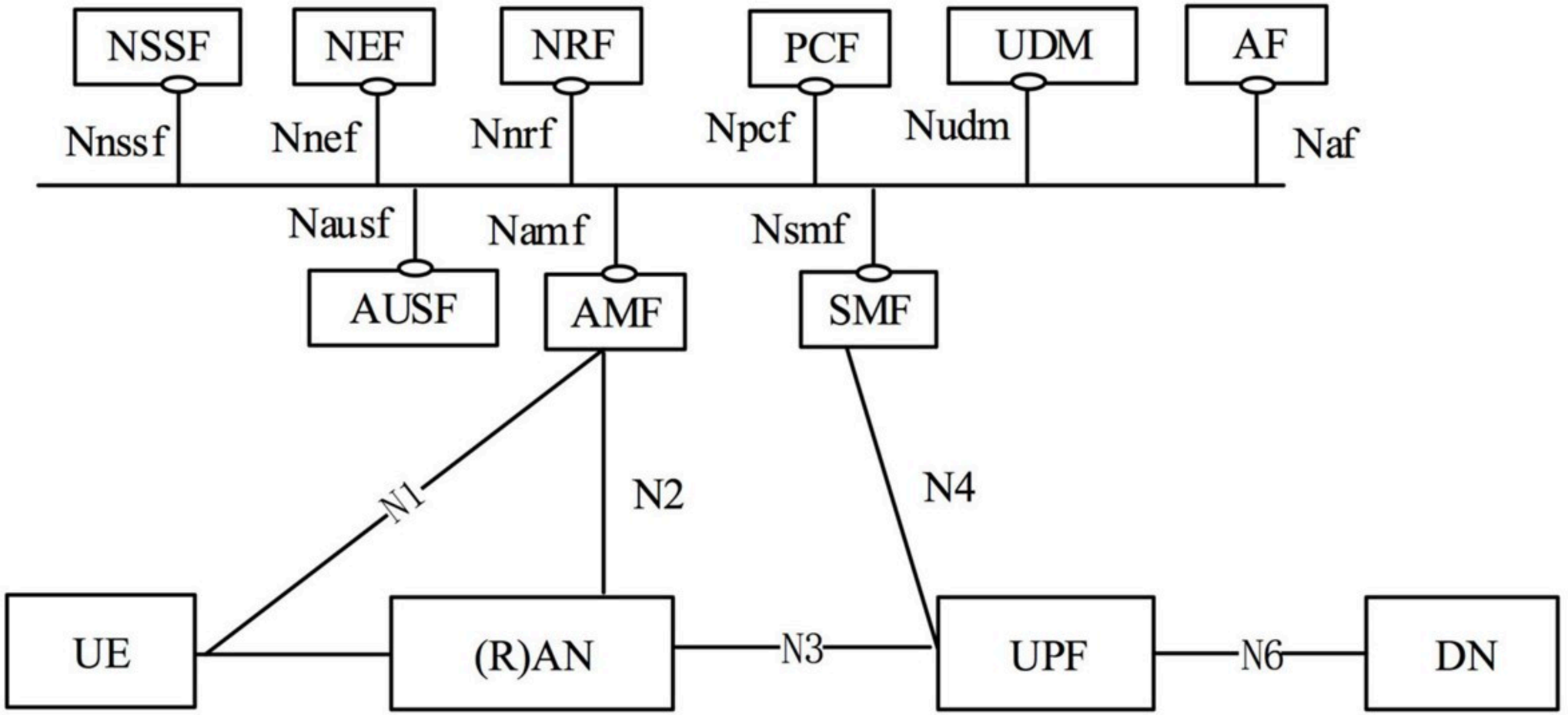
3

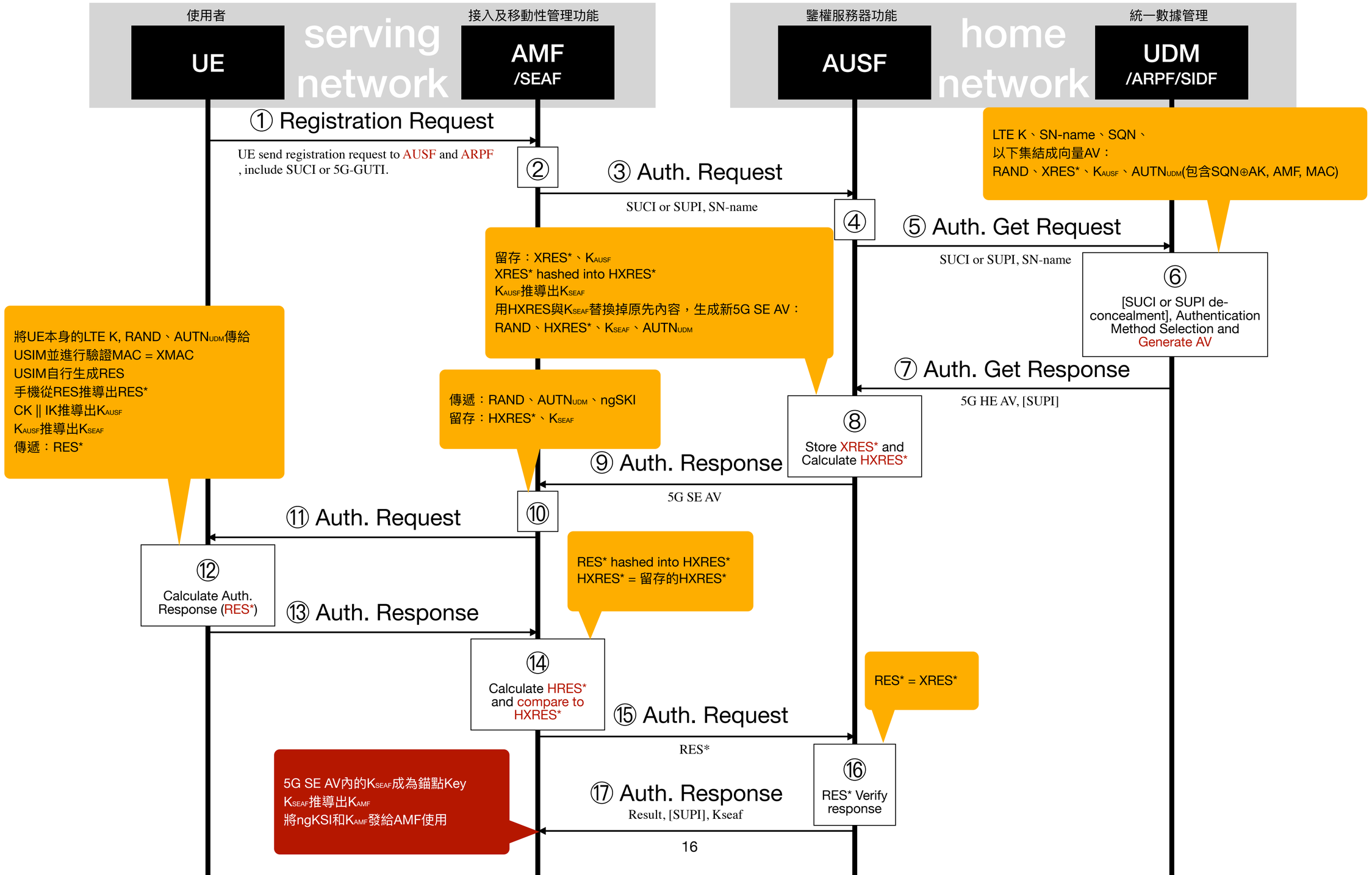
5G Topo & Architecture

3 5G Topo
Architecture



Service-Based Architecture (SBA)




使用者
UE
serving network
接入及移動性管理功能
AMF /SEAF
鑒權服務器功能
AUSF
home network
統一數據管理
UDM /ARPF/SIDF
① Registration Request
UE send registration request to AUSF and ARPF , include SUCI or 5G-GUTI.
②
③ Auth. Request
SUCI or SUPI, SN-name
④
⑤ Auth. Get Request
SUCI or SUPI, SN-name
LTE K、SN-name、SQN、
以下集結成向量AV：
RAND、XRES*、KAUSF、AUTNUDM(包含SQN⊕AK, AMF, MAC)
⑥
[SUCI or SUPI de-concealment], Authentication Method Selection and Generate AV
留存：XRES*、KAUSF
XRES* hashed into HXRES*
KAUSF推導出KSEAF
用HXRES與KSEAF替換掉原先內容，生成新5G SE AV：
RAND、HXRES*、KSEAF、AUTNUDM
⑦ Auth. Get Response
5G HE AV, [SUPI]
⑧
Store XRES* and Calculate HXRES*
將UE本身的LTE K, RAND、AUTNUDM傳給
USIM並進行驗證MAC = XMAC
USIM自行生成RES
手機從RES推導出RES*
CK || IK推導出KAUSF
KAUSF推導出KSEAF
傳遞：RES*
傳遞：RAND、AUTNUDM、ngSKI
留存：HXRES*、KSEAF
⑨ Auth. Response
5G SE AV
⑩
⑪ Auth. Request
⑫
Calculate Auth. Response (RES*)
⑬ Auth. Response
RES* hashed into HXRES*
HXRES* = 留存的HXRES*
⑭
Calculate HRES* and compare to HXRES*
⑮ Auth. Request
RES*
RES* = XRES*
⑯
RES* Verify response
5G SE AV內的KSEAF成為錨點Key
KSEAF推導出KAMF
將ngKSI和KAMF發給AMF使用
⑰ Auth. Response
Result, [SUPI], Kseaf
16

計算消息認證碼(MAC)：$MAC=F^1_K(SQN|| RAND || AMF)$ ；

計算期望的認證應答(XRES)：$XRES=F^2_K(RAND)$；

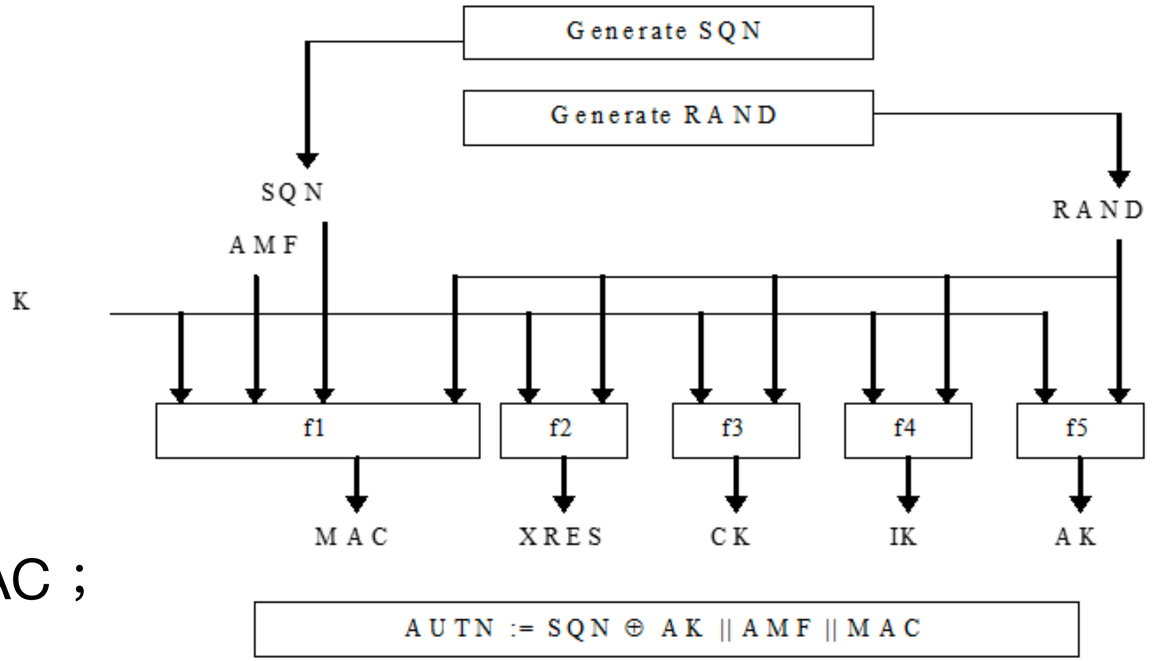
計算保密性密鑰(CK)：$CK=F^3_K(RAND)$；

計算完整性密鑰(IK)：$IK=F^4_K(RAND)$；

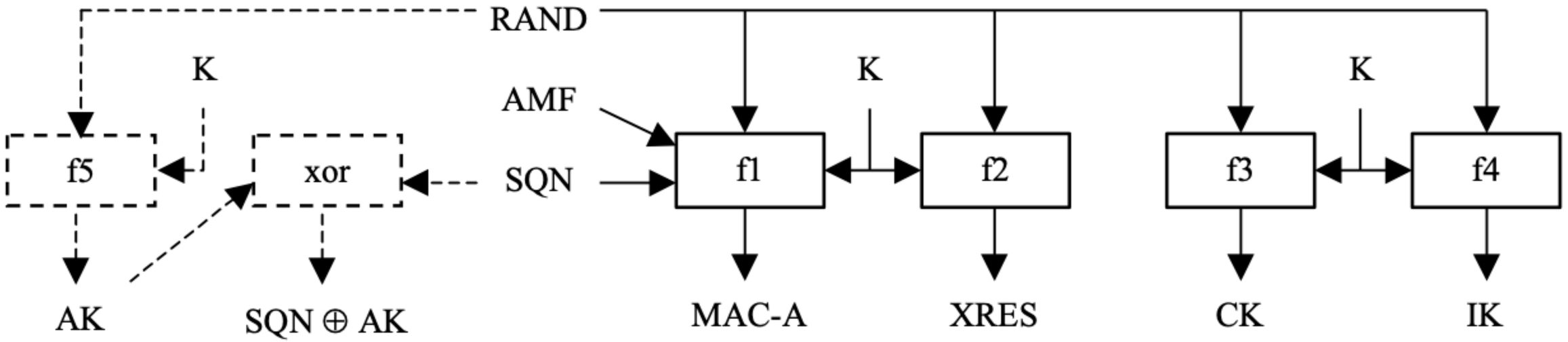
匿名密鑰(AK)：$AK=F^5_K(RAND)$；

網絡認證令牌(AUTN)：$AUTN=SQN \oplus AK|| AMF || MAC$；

AV：AV=RAND||XRES||CK||IK||AUTN；



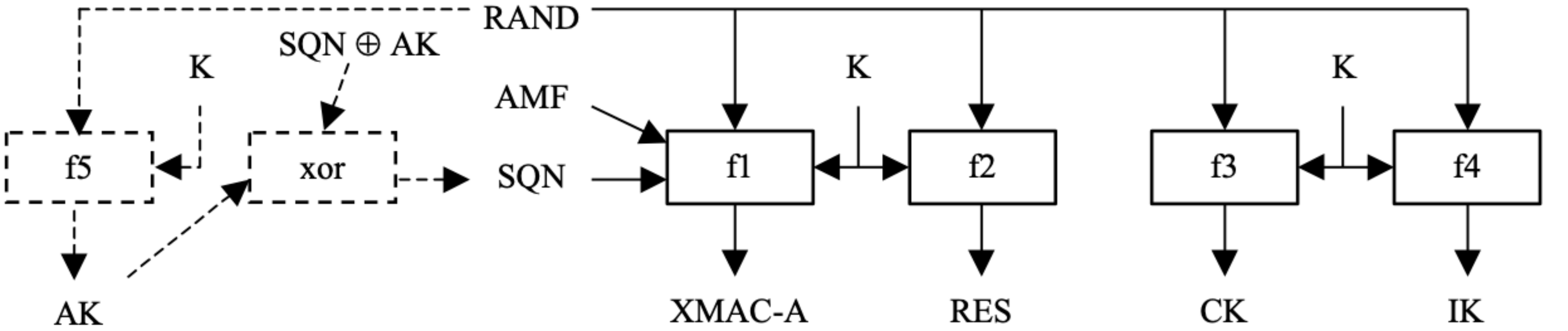
AK用來隱藏SQN，因為SQN可能會暴露用戶的位置信息。如果不需要隱藏SQN，那麼AK被設置為0。

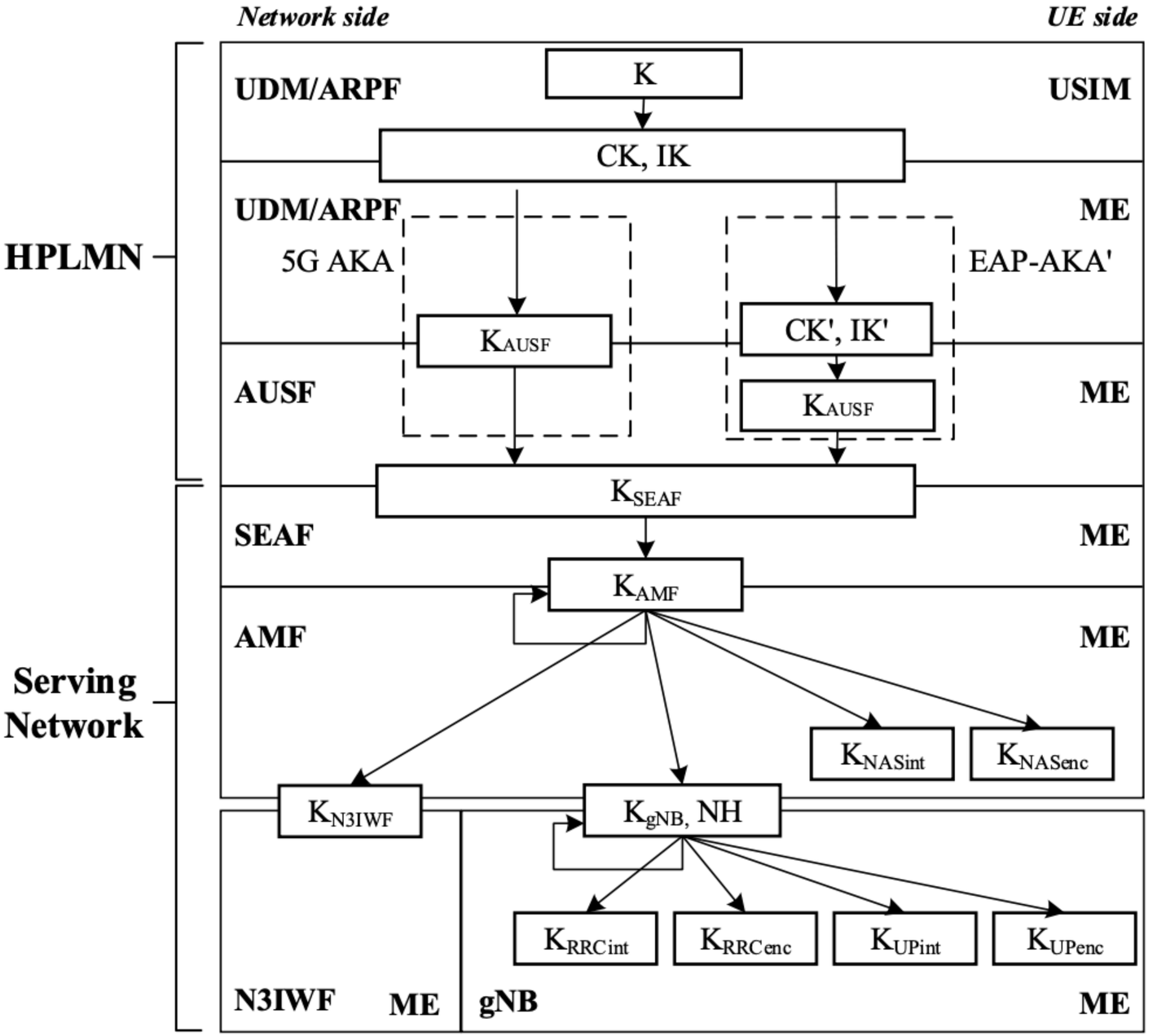


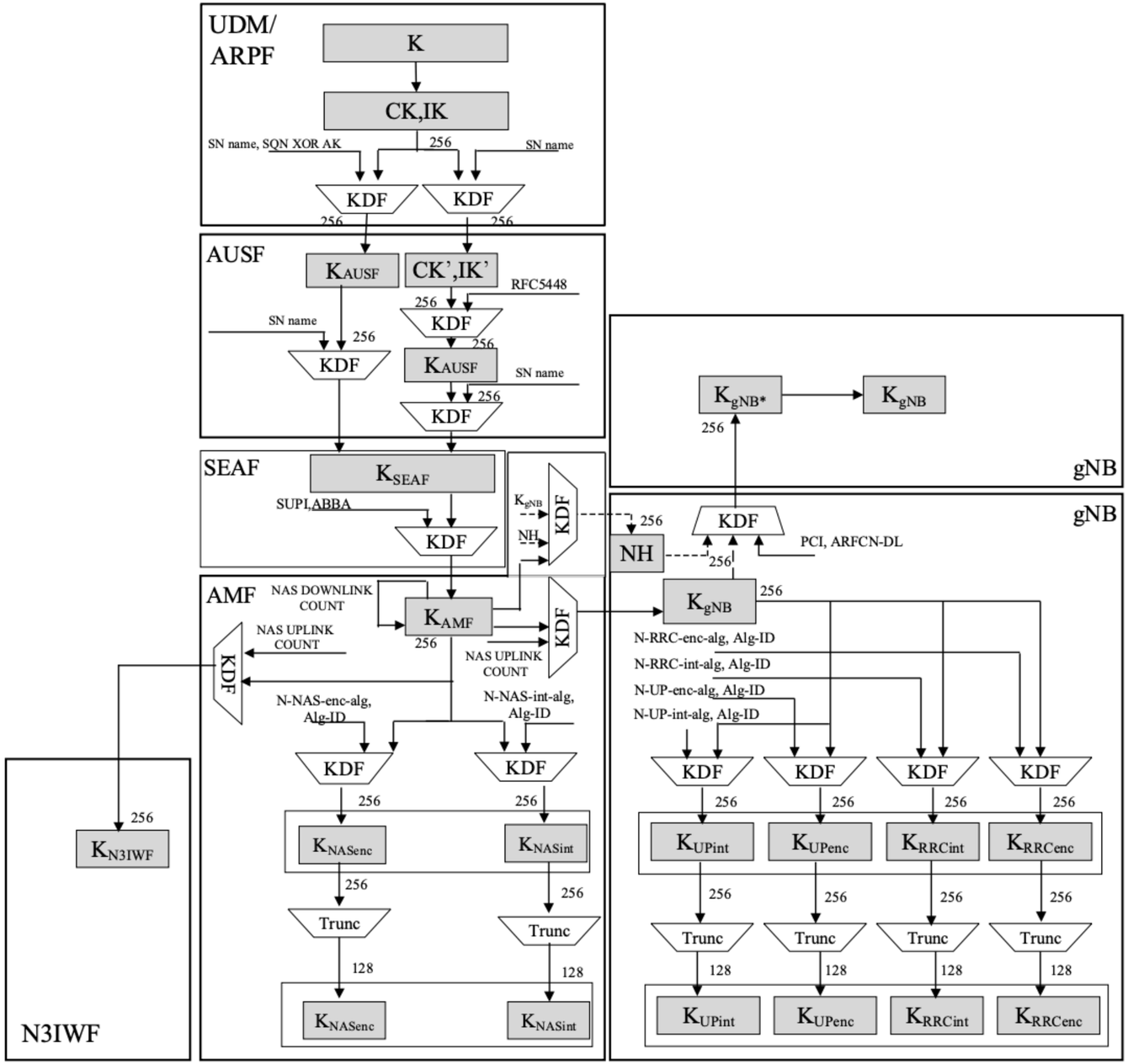
$$AUTN = SQN\ [\oplus\ AK] \parallel AMF \parallel MAC\text{-}A$$
$$Quintet = (RAND, XRES, CK, IK, AUTN)$$

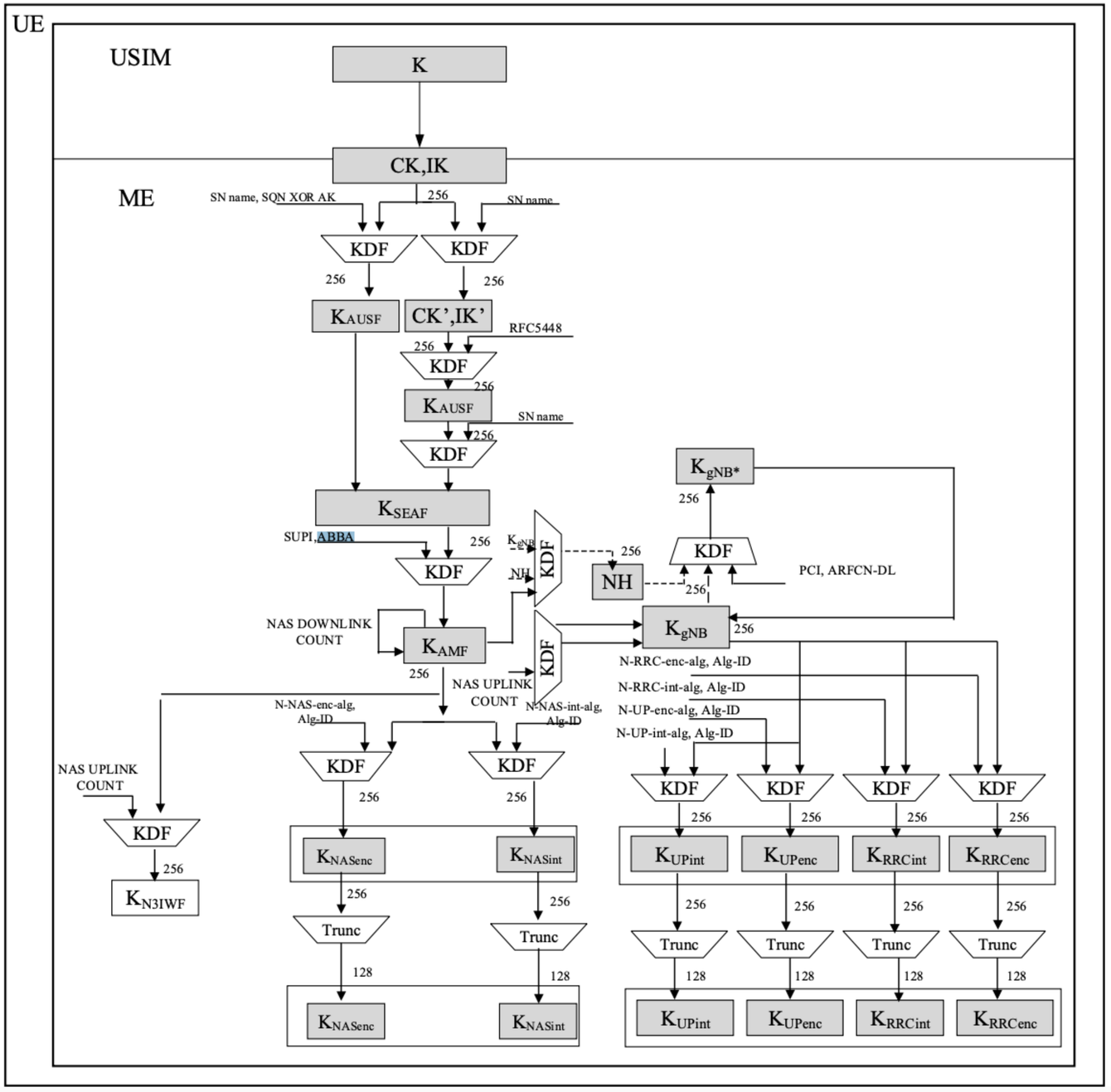
3GPP	3rd Generation Partnership Project
AES	Advanced Encryption Standard
AMF	Authentication Management Field
AK	Anonymity key
AuC	Authentication Centre
AUTS	Re-synchronisation Token
CK	Cipher Key
DPA	Differential Power Analysis
$E(X)_K$	Encryption of X under key K
IK	Integrity Key
K	Subscriber key
MAC	Message Authentication Code
MAC-A	Network Authentication Code
MAC-S	Resynchronisation Authentication Code
OP	a 128-bit Operator Variant Algorithm Configuration Field that is a component of the functions ***f1, f1*, f2, f3, f4, f5 and f5****
OP_C	a 128-bit value derived from OP and K and used within the computations of the functions ***f1, f1*, f2, f3, f4, f5 and f5****.
OFB	Output Feedback
RAND	Random Challenge
RES	Response to Challenge
RNC	Radio Network Controller
SAGE	Security Algorithms Group of Experts
SAGE 3GPP AF TF	
SAGE Task Force for the design of the 3GPP Authentication and Key Agreement Functions	
SQN	Sequence Number
SPA	Simple Power Analysis
TA	Timing Attack
UE	User Equipment
UMTS	Universal Mobile Telecommunications System
USIM	User Services Identity Module

UDM


RAND
SQN ⊕ AK
K
AMF
SQN
f5
xor
f1
f2
f3
f4
K
K
AK
XMAC-A
RES
CK
IK


Network side
UE side
HPLMN
Serving Network
UDM/ARPF
K
USIM
CK, IK
UDM/ARPF
ME
5G AKA
EAP-AKA'
KAUSF
CK', IK'
AUSF
KAUSF
ME
KSEAF
SEAF
ME
KAMF
AMF
ME
KNASint
KNASenc
KN3IWF
KgNB, NH
KRRCint
KRRCenc
KUPint
KUPenc
N3IWF
ME
gNB
ME


UDM/
ARPF
K
CK,IK
SN name, SQN XOR AK
256
SN name
KDF
KDF
256
256
AUSF
KAUSF
CK',IK'
RFC5448
256
KDF
SN name
256
KDF
256
KAUSF
SN name
256
KDF
SEAF
KSEAF
SUPI,ABBA
KDF
KgNB
NH
KDF
AMF
NAS DOWNLINK COUNT
KAMF
256
NAS UPLINK COUNT
KDF
NAS UPLINK COUNT
KDF
N-NAS-enc-alg, Alg-ID
N-NAS-int-alg, Alg-ID
KDF
KDF
256
256
KNASenc
KNASint
256
256
Trunc
Trunc
128
128
KNASenc
KNASint
256
KN3IWF
N3IWF
KgNB*
256
KgNB
gNB
gNB
256
KDF
PCI, ARFCN-DL
NH
256
KgNB
256
N-RRC-enc-alg, Alg-ID
N-RRC-int-alg, Alg-ID
N-UP-enc-alg, Alg-ID
N-UP-int-alg, Alg-ID
KDF
KDF
KDF
KDF
256
256
256
256
KUPint
KUPenc
KRRCint
KRRCenc
256
256
256
256
Trunc
Trunc
Trunc
Trunc
128
128
128
128
KUPint
KUPenc
KRRCint
KRRCenc


UE
USIM
K
CK,IK
ME
SN name, SQN XOR AK
256
SN name
KDF
KDF
256
256
KAUSF
CK',IK'
RFC5448
256
KDF
256
KAUSF
SN name
256
KDF
KSEAF
SUPI, ABBA
256
KDF
KgNB
NH
KDF
256
NH
256
KgNB*
256
KDF
PCI, ARFCN-DL
KgNB
256
NAS DOWNLINK COUNT
KAMF
256
NAS UPLINK COUNT
KDF
N-NAS-enc-alg, Alg-ID
N-NAS-int-alg, Alg-ID
N-RRC-enc-alg, Alg-ID
N-RRC-int-alg, Alg-ID
N-UP-enc-alg, Alg-ID
N-UP-int-alg, Alg-ID
NAS UPLINK COUNT
KDF
256
KN3IWF
KDF
KDF
256
256
KNASenc
KNASint
256
256
Trunc
Trunc
128
128
KNASenc
KNASint
KDF
KDF
KDF
KDF
256
256
256
256
KUPint
KUPenc
KRRCint
KRRCenc
256
256
256
256
Trunc
Trunc
Trunc
Trunc
128
128
128
128
KUPint
KUPenc
KRRCint
KRRCenc

3 5G Topo

建立 5gtopo.py 在 mininet/custom/5gAuth 內

```
from mininet.topo import Topo

class Topo1(Topo):
 def __init__(self):
  #Initialize topology
  Topo.__init__(self)

  #add Host and Switch
  #UE
  ue=self.addHost('UE',ip='10.1.0.1')

ues=self.addSwitch('UEs',dpid='0000000000000001')
  #RAN
  ran=self.addHost('RAN',ip='10.1.0.2')

rans=self.addSwitch('RANs',dpid='0000000000000002')
  #AMF
  amf=self.addHost('AMF',ip='10.1.0.3')

amfs=self.addSwitch('AMFs',dpid='0000000000000003')
  #AUSF
  ausf=self.addHost('AUSF',ip='10.1.0.4')

ausfs=self.addSwitch('AUSFs',dpid='0000000000000004')
  #UDM
  udm=self.addHost('UDM',ip='10.1.0.5')

udms=self.addSwitch('UDMs',dpid='0000000000000005')

  #add Links
  self.addLink(ue,ues,port1=1,port2=2)
  self.addLink(ues,rans,port1=1,port2=3)
  self.addLink(rans,ran,port1=2,port2=1)
  self.addLink(rans,amfs,port1=1,port2=3)
  self.addLink(amfs,amf,port1=2,port2=1)
  self.addLink(amfs,ausfs,port1=1,port2=3)
  self.addLink(ausfs,ausf,port1=2,port2=1)
  self.addLink(ausfs,udms,port1=1,port2=3)
  self.addLink(udms,udm,port1=2,port2=1)


topos={'mytopo':(lambda:Topo1())}
```

3 5G Topo

Ryu + Mininet

開啟 第一個 *Terminal (Ryu)*

```
$ ryu-manager ryu.app.simple_switch_13
```

開啟 第二個 *Terminal (Mininet)*

```
$ cd ~/mininet/custom/5gAuth
$ sudo mn --custom 5gtopo.py --topo
  mytopo --
  controller=remote,ip=127.0.0.1,port=6633
  --switch ovs,protocols=OpenFlow13
$ mininet > pingall
$ mininet > xterm UE RAN
```

3 5G Topo
Client

在 ~/mininet/custom/5gAuth 寫入 Client.py

```
$ import socket
$ host = "10.1.0.2"
$ port = 1
$ s = socket.socket(socket.AF_INET,
  socket.SOCK_STREAM)
$ s.connect((host, port))
$ while True:
$     cmd = input("Please input msg: ")
$     s.send(cmd.encode('UTF-8'))
$     data = s.recv(1024)
$     print("server send: %s" % (data.decode('UTF-8')))
```

3 5G Topo
Server

在 ~/mininet/custom/5gAuth 寫入 Server.py

```
$ import socket
$ bind_ip = "10.1.0.2"
$ bind_port = 1
$ s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
$ s.bind((bind_ip, bind_port))
$ s.listen(5)
$ while True:
$     c, addr = s.accept()
$     print("Connected by: ", addr)
$     while True:
$         data = c.recv(1024)
$         print("Client recv data: %s" % (data.decode('UTF-8')))
$         c.send("ACK!".encode('UTF-8'))
```

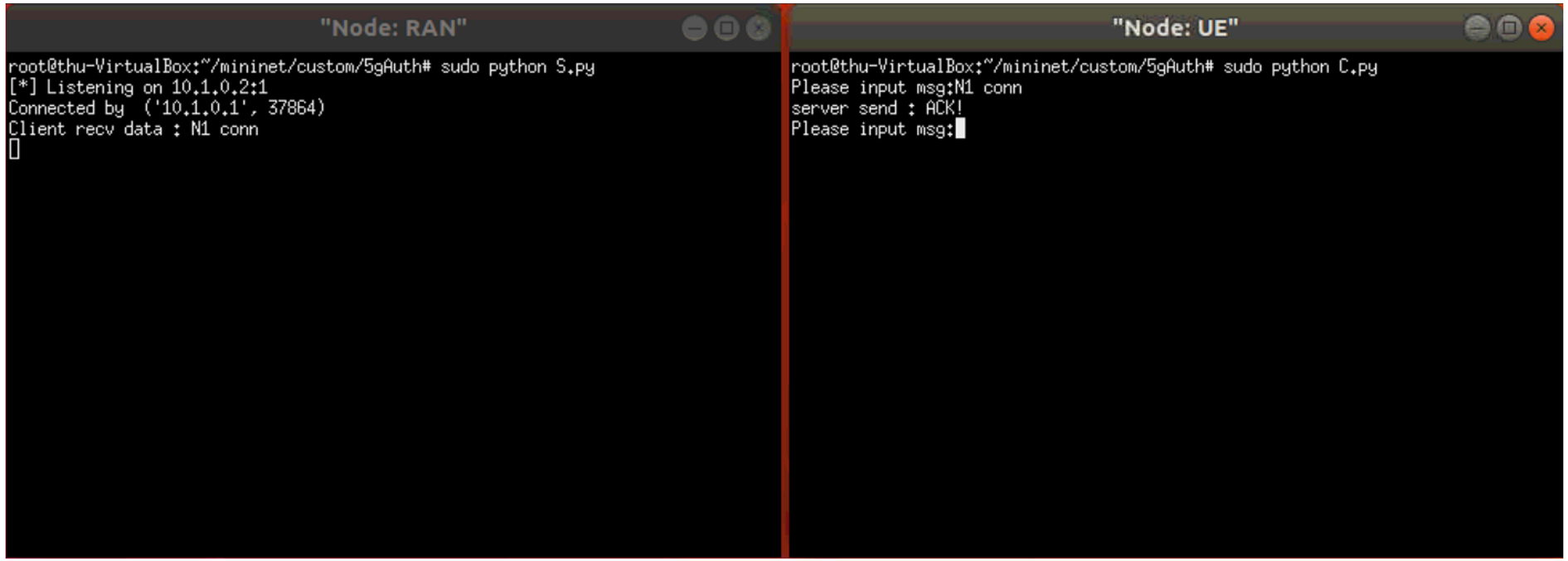
3 5G Topo
UE connect to RAN

xterm (RAN) >

$ sudo python Server.py

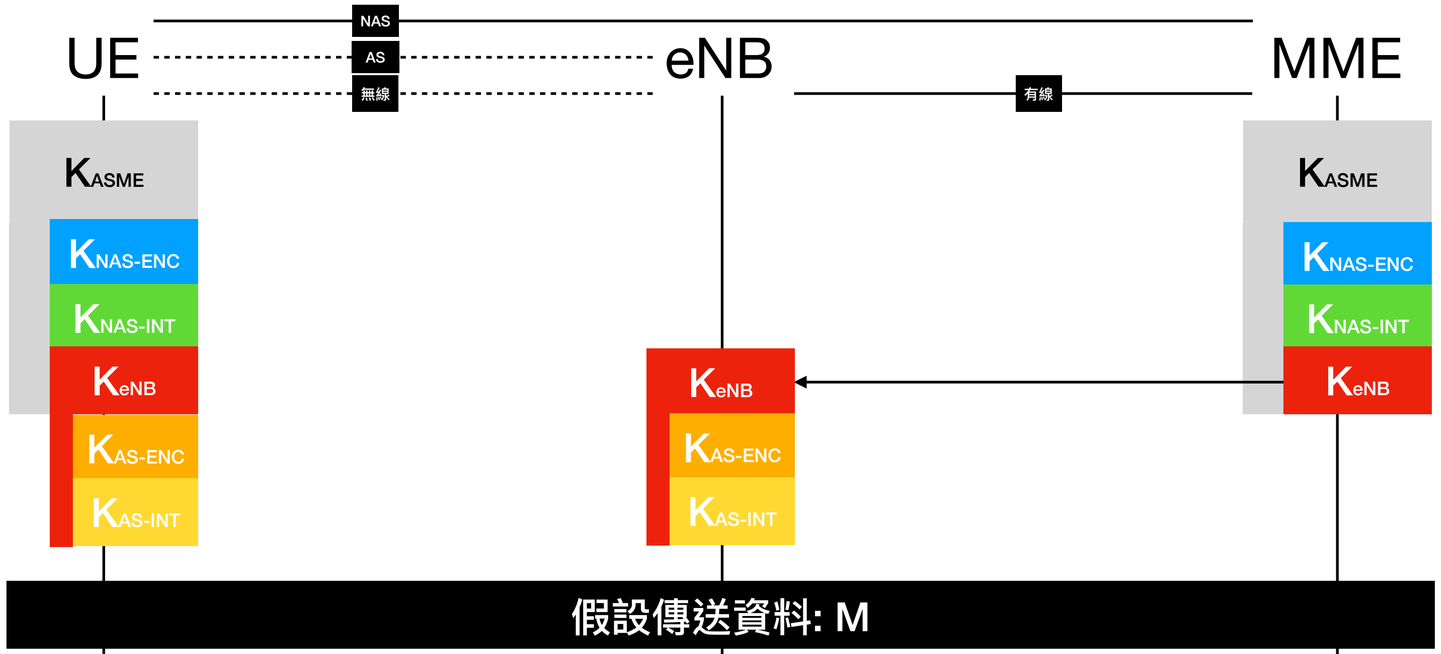
xterm (UE) >

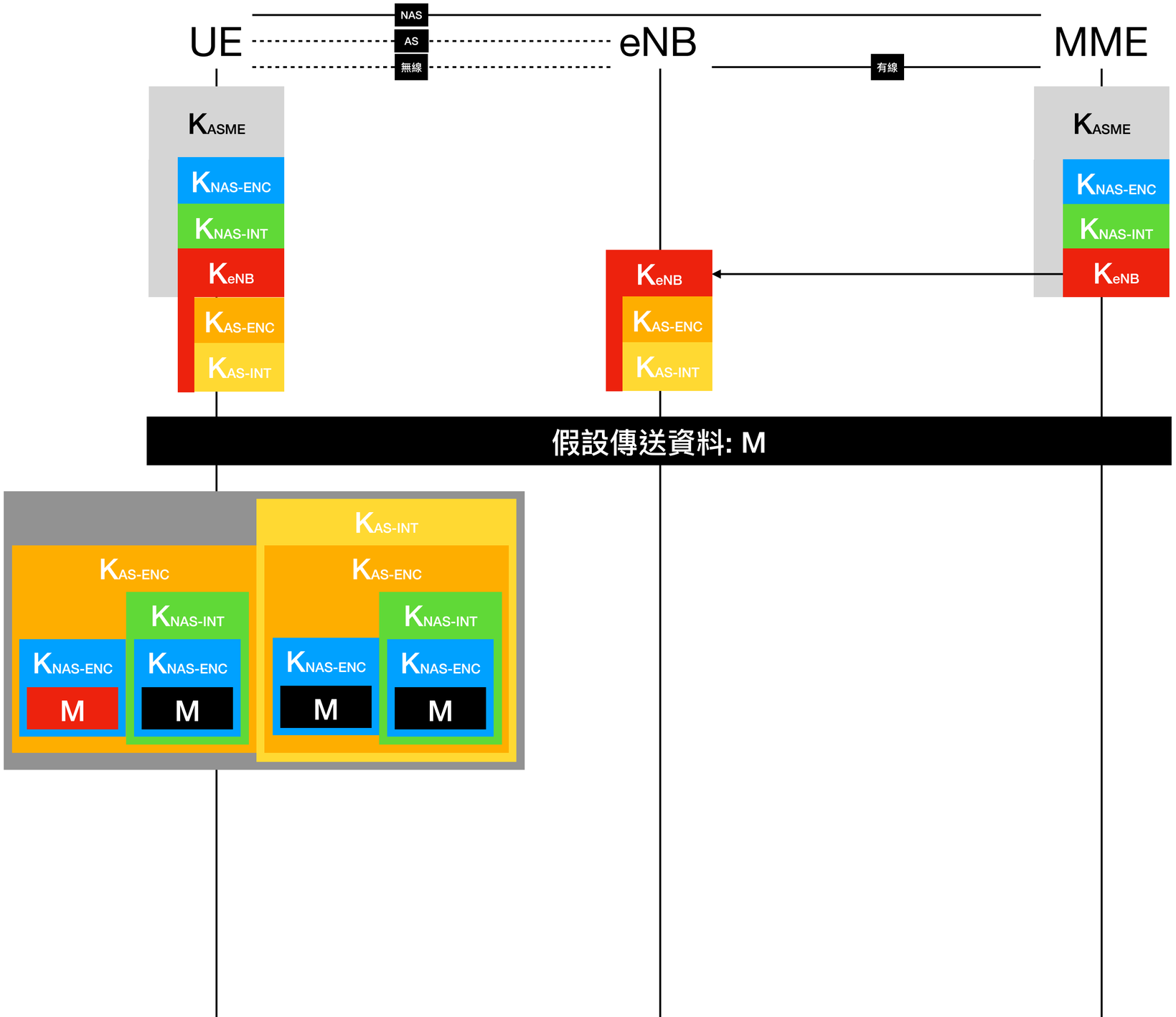
$ sudo python Client.py

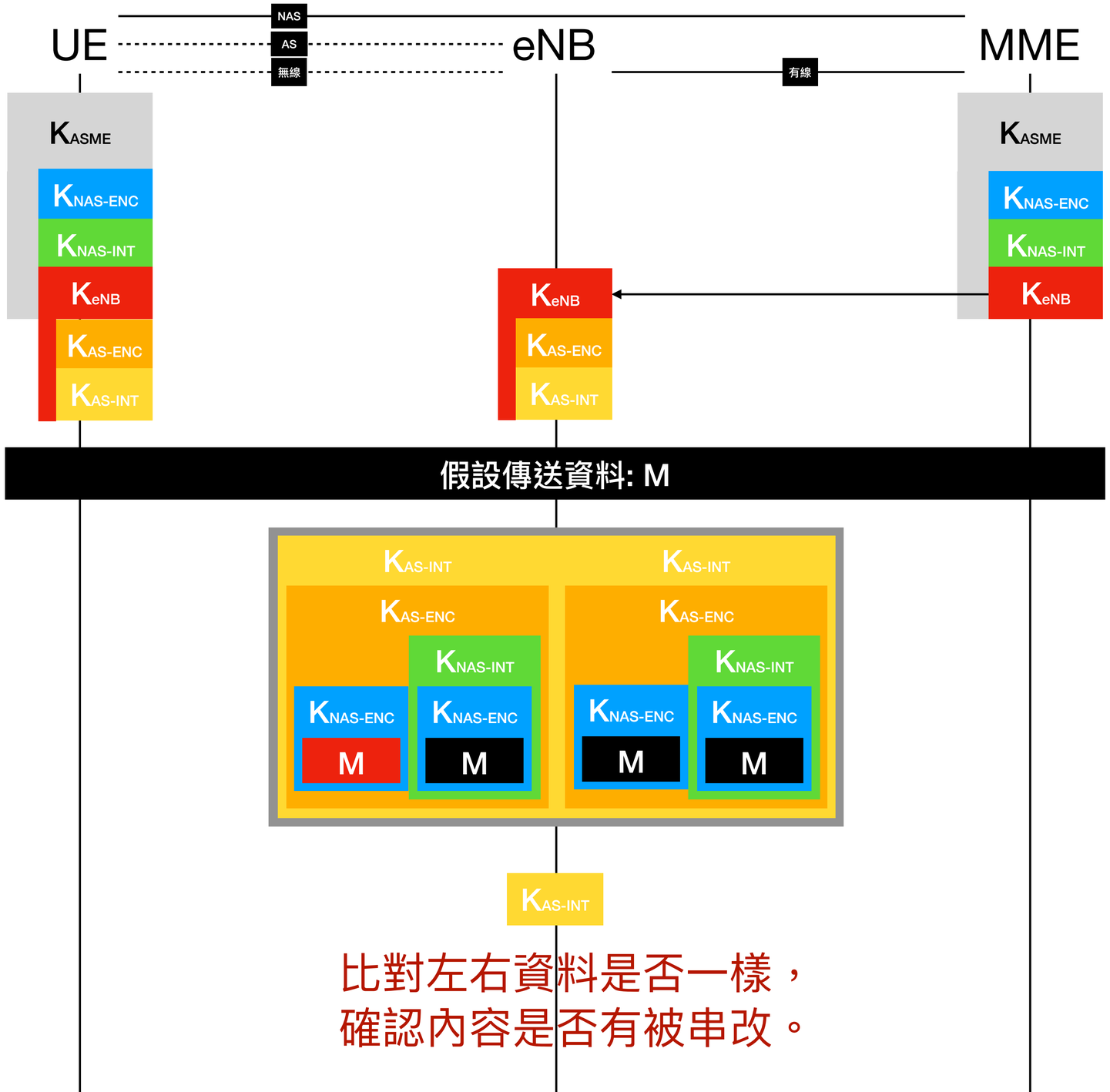


可從 UE 傳遞信息至 RAN

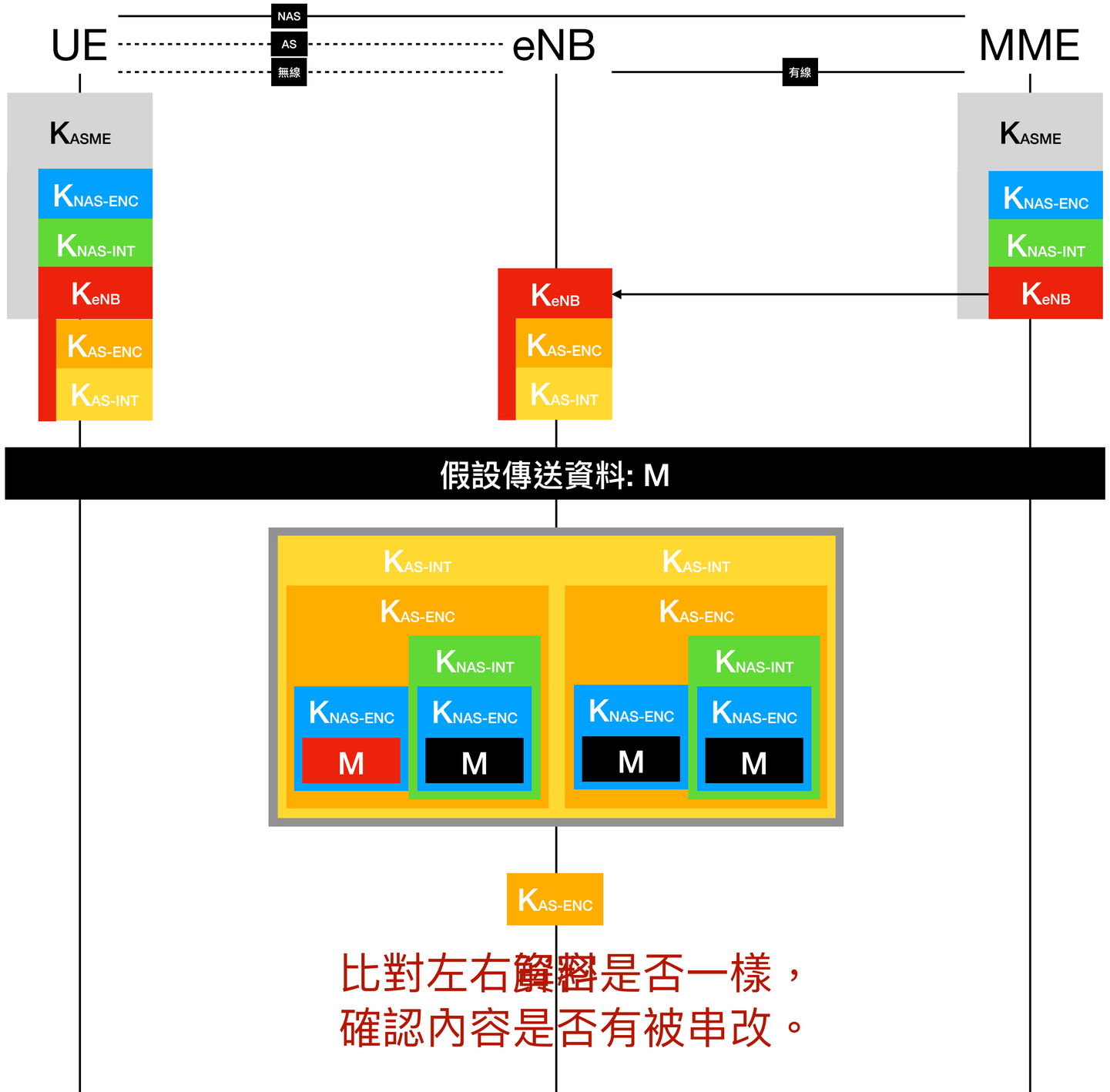
4 UE <-> AMF

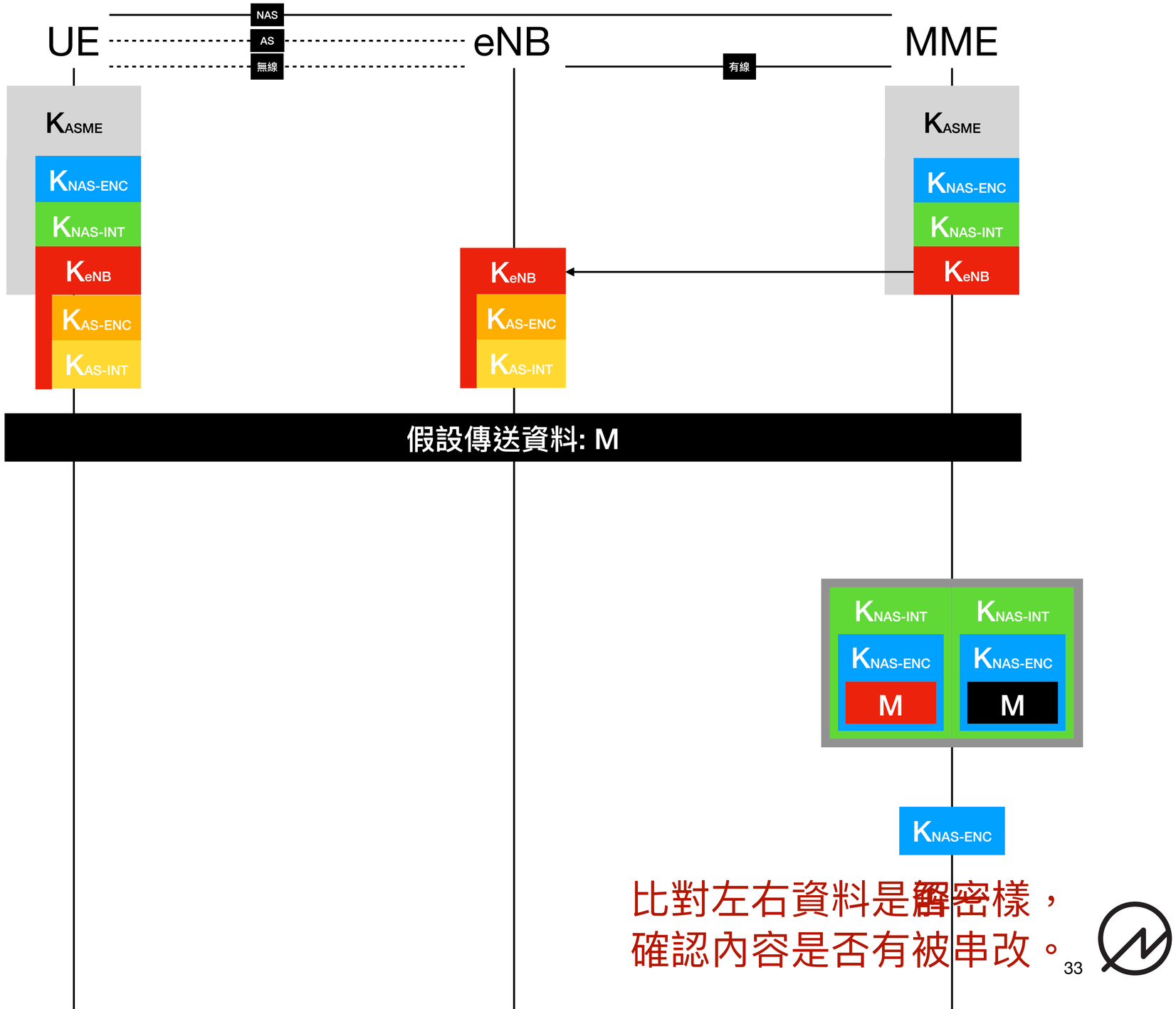

NAS
AS
無線
有線
UE
eNB
MME
KASME
KNAS-ENC
KNAS-INT
KeNB
KAS-ENC
KAS-INT
KASME
KNAS-ENC
KNAS-INT
KeNB
KeNB
KAS-ENC
KAS-INT
假設傳送資料: M


NAS
AS
無線
UE
eNB
有線
MME
KASME
KNAS-ENC
KNAS-INT
KeNB
KAS-ENC
KAS-INT
KeNB
KAS-ENC
KAS-INT
KASME
KNAS-ENC
KNAS-INT
KeNB
假設傳送資料: M
KAS-INT
KAS-ENC
KNAS-INT
KNAS-ENC
M
KNAS-ENC
M
KAS-ENC
KNAS-INT
KNAS-ENC
M
KNAS-ENC
M



比對左右資料是否一樣，
確認內容是否有被串改。


NAS
AS
無線
UE
eNB
有線
MME
KASME
KNAS-ENC
KNAS-INT
KeNB
KAS-ENC
KAS-INT
KeNB
KAS-ENC
KAS-INT
KASME
KNAS-ENC
KNAS-INT
KeNB
假設傳送資料: M
KAS-INT
KAS-INT
KAS-ENC
KAS-ENC
KNAS-INT
KNAS-INT
KNAS-ENC
KNAS-ENC
KNAS-ENC
KNAS-ENC
M
M
M
M
KAS-ENC
比對左右簽章是否一樣，
確認內容是否有被串改。



比對左右資料是解密樣，
確認內容是否有被串改。