Advanced Network Security

Lecture 4: Decrypting Phone Calls

Harald Vranken, Katharina Kohls

September 29, 2022

Open University Nijmegen
Radboud University Nijmegen

Recap

(1) General introduction to mobile networks
- Terminology: UE, eNodeB, EPC
- Mobile protocol stack
- Security features

(2) Three layer-two attacks
- Website fingerprinting
- Identity mapping
- User data redirection

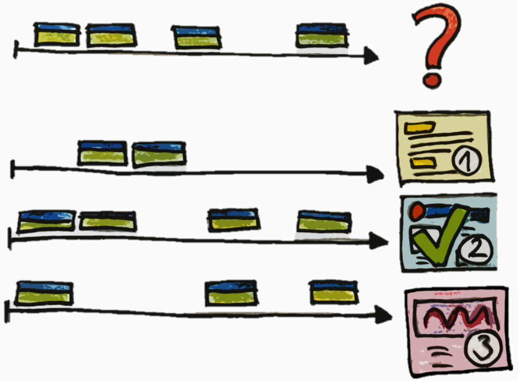
Website Fingerprinting: Classification Attack

Preparation

- Attacker pre-records requests and responses to *n* websites
- Repeats each website several times
- Results in a labeled data set

Classification Attack

- Record traffic of victim
- Compare trace with pre-recorded data
- Highest similarity $\rightarrow$ guess



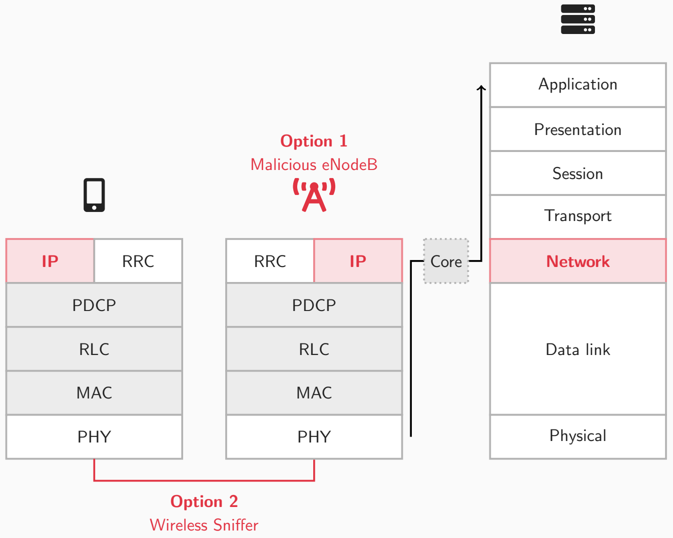
Website Fingerprinting: Malicious Base Station vs. Wireless Sniffer

Option 1: **eNodeB**

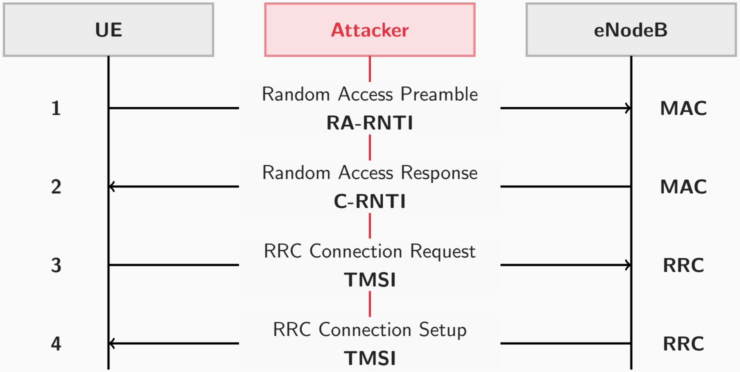
- Access to L1-L3
- LTE encryption

Option 2: **Sniffer**

- Access to air interface
- Only transmissions



Identity Mapping Attack



Identity Mapping Attack: TMSI and IMSI

TMSI

Temporary Mobile Subscriber Identity, randomly assigned temporary identity. For security reasons, the TMSI is a placeholder for the unique IMSI of a user. It can be updated after a certain time period.

IMSI

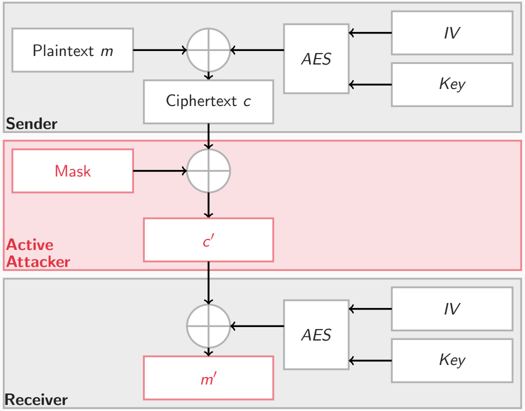
International Mobile Subscriber Identity, uniquely identifies every mobile user. It is *not* the identifier of the SIM card, but still part of the profile.

The TMSI is used for security reasons!
It can be reset if compromised. The IMSI cannot be reset.

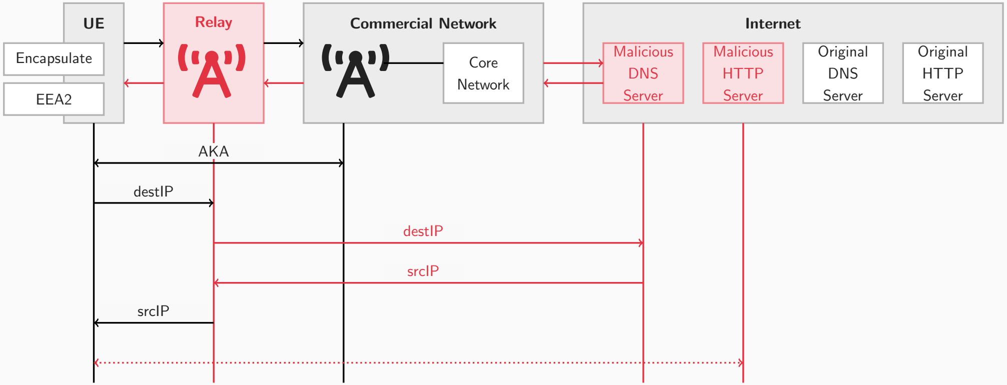
aLTEr: Known-Plaintext Attack

- PDCP encrypts IP packet
- Stream cipher: AES in counter mode
- XOR manipulation mask *m*
- Deterministic manipulation
- Manipulation remains undetected...
  But why?

No user plane integrity protection 👎



aLTEr: Attack Procedure



Call Me Maybe

Call Me Maybe:
Eavesdropping Encrypted LTE Calls With REVOLTE

David Rupprecht
Ruhr University Bochum
david.rupprecht@rub.de

Katharina Kohls
Ruhr University Bochum
katharina.kohls@rub.de

Thorsten Holz
Ruhr University Bochum
thorsten.holz@rub.de

Christina Pöpper
NYU Abu Dhabi
christina.poepper@nyu.edu

Abstract

Voice over LTE (VoLTE) is a packet-based telephony service seamlessly integrated into the Long Term Evolution (LTE) standard and deployed by most telecommunication providers in practice. Due to this widespread use, successful attacks against VoLTE can affect a large number of users worldwide. In this work, we introduce REVOLTE, an attack that exploits an LTE implementation flaw to recover the contents of an encrypted VoLTE call, hence enabling an adversary to eavesdrop on phone calls. REVOLTE makes use of a predictable keystream reuse on the radio layer that allows an adversary to decrypt a recorded call with minimal resources. Through a series of preliminary as well as real-world experiments, we successfully demonstrate the feasibility of REVOLTE and analyze various factors that critically influence our attack in commercial networks. For mitigating the REVOLTE attack, we propose and discuss short- and long-term countermeasures deployable by providers and equipment vendors.

1 Introduction

Millions of people worldwide use the latest widely deployed mobile communication standard LTE daily. Besides high-speed Internet access, LTE also provides the packet-based telephony service VoLTE. VoLTE promises low call-setup times and high-definition voice quality while being seamlessly integrated into the standard call procedure. With more than 120 providers worldwide and over 1200 different device types supporting VoLTE [23], it is an essential part of our communication infrastructure. At the same time, the use of VoLTE is fully transparent to the user and improves the call quality without requiring any further interaction. Consequently, any practical vulnerability in the VoLTE standard has far-reaching consequences for users all over the world, without them even realizing that they may be affected.

LTE not only improves the performance of prior mobile network generations, but it also defines a series of fundamental security aims to protect further the sensitive information of phone calls, web browsing, etc. One crucial aspect of these security aims is providing data confidentiality [8] for all voice calls, which protects LTE communication from eavesdropping. This is achieved by implementing publicly reviewed encryption algorithms like AES that protect the radio-layer transmission. In addition, VoLTE can establish an *additional* layer of security that further protects all signaling messages (IPsec tunnel) and voice data (SRTP). We will later see how these additional security features must be considered in the design of our attack. Breaking these protection mechanisms and thus the data confidentiality of LTE, allows us to recover the information of an arbitrary phone call. In a setting where the underlying mobile network generation promises strong security aims, this might reveal highly sensitive information that was assumed to be protected.

While prior work demonstrates that the aims of location and identity privacy [13,43] and an attacker can break the integrity of user data [38], a technical report by Raza and Lu [36] recently indicated that the data confidentiality of LTE might contain a fundamental flaw. By jamming particular messages and reinstalling a key, the authors introduce a concept that theoretically allows eavesdropping on a VoLTE connection. Although their work presents the foundation for breaking the essential security aim—data confidentiality—of the LTE communication standard, their work only covers a theoretical evaluation of the attack vector. It lacks any evidence that the concept is actually feasible in a real-world setup and at a sufficiently large scale.

In this work, we build upon the concept of key reinstallation and break the data confidentiality aim of LTE in a commercial network setup. This attack vector is the starting point for REVOLTE: An attack concept that uses a *passive* downlink sniffer instead of active jamming, and provides insights on numerous adjustments to the technical requirements and challenges of a real-world implementation of the attack. REVOLTE is a *layer-two* attack that allows us to Reuse Encrypted VoLTE traffic to eavesdrop on an encrypted voice call. Keystream reuse can occur when two calls are made within one radio connection.

Today:
Decrypting VoLTE Calls

`https://revolte-attack.net/`

VoLTE: Voice over LTE

VoLTE

- Uses a 4G *data* connection instead of the standard voice network to operate phone calls.
- Has smaller packet headers and is more efficient than VoIP/LTE.
- Device, firmware, and provider must all support VoLTE.

VoLTE Networks

Is this a thing?

- 253 operators
- 113 countries

Why is this a thing?

- Better call quality
- LTE *security* features

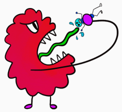
Target and Keystream Call

Target Call

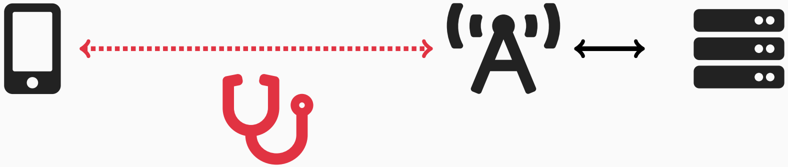
Record VoLTE Call

Keystream Call

Recover keystream!



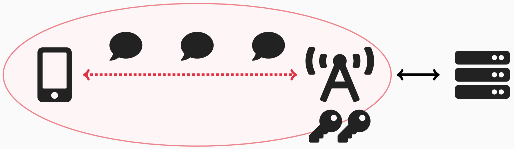
Decrypting Encrypted Calls



Call me Maybe!?

- Record VoLTE phone calls → Wireless sniffer
- Exploit keystream reuse → Implementation flaw
- Decrypt recorded call → Attack goal

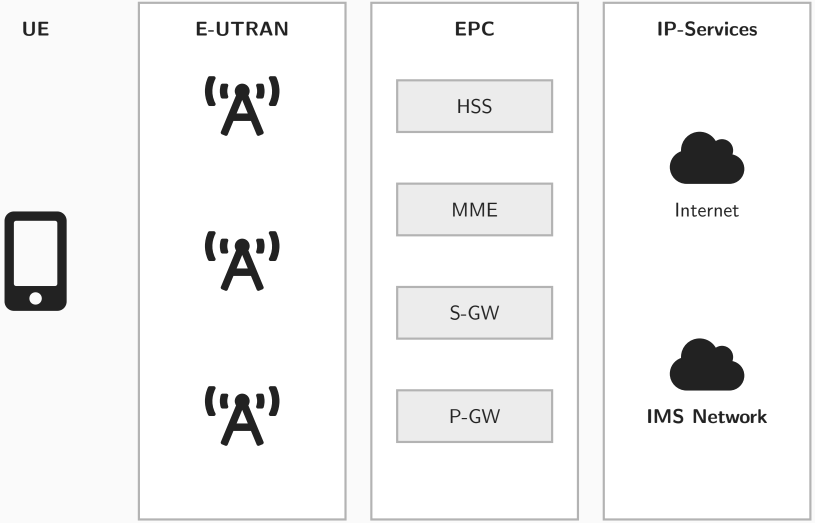
Attack Challenges



- **Wireless sniffer:** Be in the same radio cell, record traffic between UE and eNodeB.
- **Keystream reuse:** Call is encrypted. If the *same* keystream is used again, we can decrypt the call.
- **Challenges:** Everything that happens in the wireless transmission of voice data.

Attack Challenges

VoLTE Infrastructure



VoLTE Infrastructure

📱 User, making a phone call
- Make initial call
- Answer a subsequent call

📡 Radio cell
- Set the codec and ROHC
- Apply security measures

☁ Packet-switched phone call through the IMS
- Transport protocols

- Home Subscriber Service (HSS)
- Mobility Management Entity (MME)
- Serving Gateway (S-GW)
- PDN Gateway (P-GW)

Challenges

Performance

- **Codecs and comfort noise**
- **Robust header compression**
- **Radio data bearers**

IP Multimedia Subsystem (IMS)

- Control and media planes
- Packet-switched architecture

Security

- Radio-layer encryption
- Additional AKA
- Secure Real-Time Transport Protocol (SRTP)

Codecs and Comfort Noise

Multimedia Codec

transforms signals between different representations with either optimizing the data consumption or the quality.

Three possible codecs:

(1) Enhanced Voice Services (EVS)[1]
(2) Adaptive Multi-Rate (AMR)
(3) Adaptive Multi-Rate Wideband (AMR-WB)

[1] https://www.3gpp.org/news-events/1639-evs_news

Optimize Bitrates

Optimization Methods

- Comfort Noise
  - Save data in periods where the calling partner is silent.
  - Comfort noise is generated through a *seed*.
  - Seed is smaller, transmitted on lower frequency.
  - **Example:** AMR-WB uses 40bit over 160ms, actual voice uses 477bit every 20ms.
- Transcoding occurs when calling partners use different codecs.
  - **Example:** Radio-layer problems enforce a downsampling.

Why are codecs relevant for us?

We will eavesdrop a phone call,
everything is encoded by the applied multimedia codec.

What we need to know:

- What codec was applied?
- How can we reconstruct the information?
- How to handle comfort noise?
- How to handle transcoding?

Challenges

Codecs and comfort noise

Robust header compression

Radio data bearers

Robust Header Compression (ROHC)

Robust Header Compression (ROHC)

is a technique to save bits in the headers of IP, TCP, UDCP, and RTP packets. The compression saves bandwidth by removing redundancies from packet headers with the same connection endpoints.

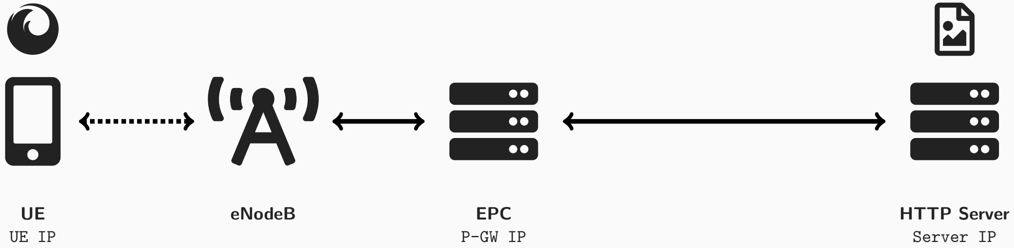
ROHC and ReVoLTE:

(1) Addressing packets in a mobile connection

(2) Involved protocols

(3) Protocol headers and redundant information

Robust Header Compression (ROHC)



What protocols are involved?

What header fields remain the same?

Protocol Headers

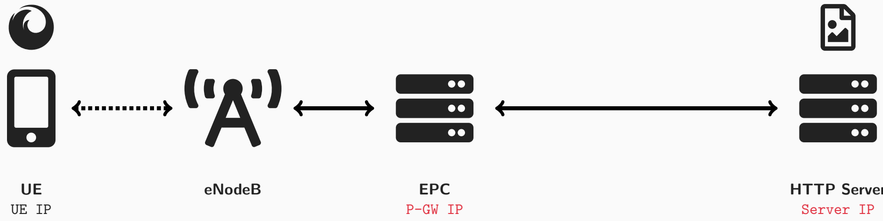
IPv4 Header

Version	IHL	Type of Service	Total Length
Identification	Flags	FragmentOffset	
Time to Live	Protocol	Header Checksum	
Source IP Address			
Destination IP Address			
Options	Padding		

UDP Header

Source Port	Destination Port
Length	Checksum
Data	

Redundant Information



192.168.1.163 107.77.235.125 172.217.17.78

Robust Header Compression (ROHC)

Remove Redundancy:

- Send the first packet with all header information included
  $\rightarrow$Both sides learn relevant and static information.
- In the following packets, remove all headers that do not change.

The eNodeB activates ROHC profiles:

- Profile 1: compress RTP, UDP, IP headers;
  only transmit RTP data with ROHC header.
- Profile 2: compress UDP, IP; only transmit UDP payload with ROHC header.

Demo: ROHC

Demo:

ROHC Profiles

```
lte-rrc.drb_Identity  || (lte-rrc.c1 == 4)

# demo_revolte_not_possible.pcap
# 1 RRCConnectionReconfiguration
# 6     RRCConnectionReconfiguration
```

Why is ROHC relevant?

ROHC compresses the transmitted packets and influences the data that we can eavesdrop.

What we need to know:

- What kind of compression is enabled?
- What information do we lose?
- When does this occur?

Challenges

Codecs and comfort noise

Robust header compression

Radio data bearers

Radio Data Bearers

Radio Bearers

- Signaling Radio Bearers SRB
- **Dedicated Radio Bearers DRB**

Idle and Active Connections

- Inactivity triggers `idle` mode
- Active radio connections use a dedicated radio bearer
- Logical link of UE and eNodeB
- Define transmission requirements

Bearer	Purpose	Bearer ID
DRB1	Internet	1
DRB2	SIP (IMS)	2
DRB3..32	RTP	3..32

- DRB1: Standard Internet connection
- DRB2: Signalling traffic for the IMS
- DRB3..32: Phone calls, is immediately removed after the call

Demo:
Setting the DRB

```
lte-rrc.drb_Identity || (lte-rrc.c1 == 4)

# demo_revolte_not_possible.pcap
# 1 RRCConnectionReconfiguration
# 6       RRCConnectionReconfiguration

# demo_PDCP.pcap
# 35 RRCConnectionReconfiguration
```

Why are Bearers relevant?

We will later exploit a keystream reuse. Using the same bearer ID contributes to keeping the same keystream.

What we need to know:

- How does the eNodeB pick the bearer ID?
- Does the bearer ID stay the same?

Challenges

Performance

- Codecs and comfort noise
- Robust header compression
- Radio data bearers

IP Multimedia Subsystem (IMS)

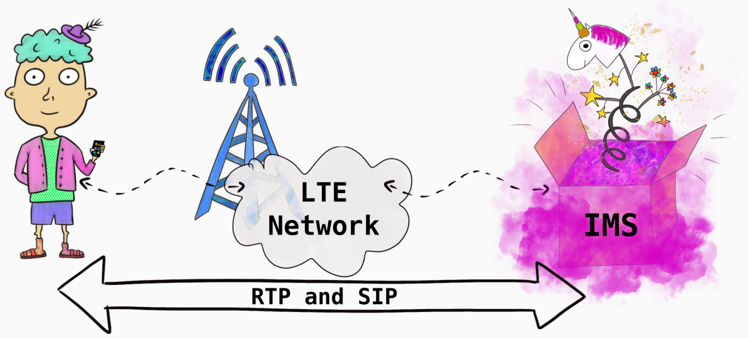
- **Control and media planes**
- **Packet-switched architecture**

Security

- Radio-layer encryption
- Additional AKA
- SRTP

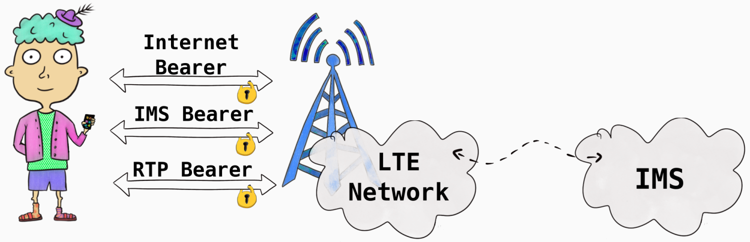
IP Multimedia Subsystem (IMS)

IP Multimedia Subsystem (IMS)



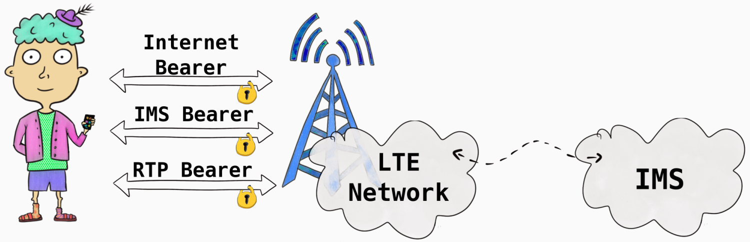
- **IMS:** Generic system for IP-based multimedia
- **RTP:** Transport audio and video via IP
- **SIP:** Build and control communication session

Radio Bearers



Bearer	Purpose	Bearer ID
DRB1	Internet	1
DRB2	SIP (IMS)	2
DRB3..32	RTP	3..32

Why is the IMS relevant?



What we need to know:

- RTP bearer is the most important for our attack.
- Two security features are relevant:
  (1) RTP versus SRTP
  (2) Bearer ID as input for the keystream generation.

Challenges

Performance

- Codecs and comfort noise
- Robust header compression
- Radio data bearers

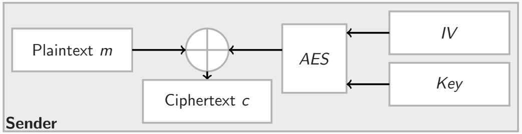
IP Multimedia Subsystem (IMS)

- Control and media planes
- Packet-switched architecture

Security

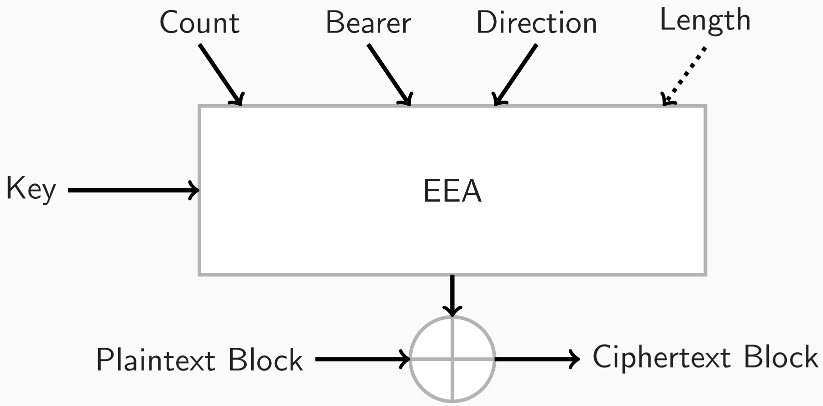
- **Radio-layer encryption**
- **Additional AKA**
- **SRTP**

Radio-Layer Encryption

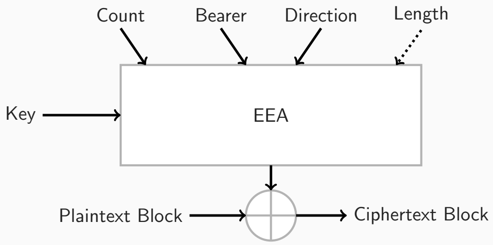


- PDCP applies the encryption algorithm EEA.
  - EEA1: Snow3G
  - EEA2: AES in counter mode
  - EEA3: ZUC
- Plaintext gets XOR-ed with keystream
- Keystream blocks are generated individually for each plaintext block

Keystream Generation



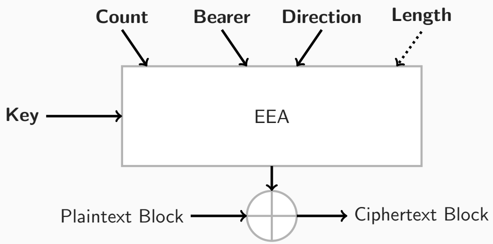
Keystream Generation



- **Key** Use plane key, established for each new radio connection.
- **Count** PDCP sequence number + PDCP hyperframe number
- **Bearer** Bearer identity
- **Direction** Uplink or downlink
- **Length** Length of the keystream block (does not influence the keystream generation)

Why is the keystream generation relevant?

We need to reconstruct the keystream to decrypt a recorded call.



What we need to know:

- Do these inputs change over multiple connections?
- What influences changes?

Background: Authentication and Key Agreement AKA

Understanding the basics:

- LTE has *mutual authentication*
- This is achieved through the AKA
- For VoLTE an *additional* AKA is used
- **We'll first learn how the AKA works**
- We need this to understand the VoLTE AKA

Authentication and Key Agreement AKA

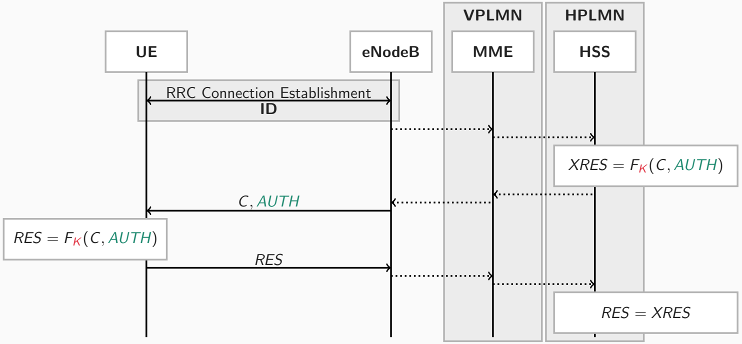
Mutual Authentication in LTE:

- LTE uses a challenge-response protocol to establish **mutual authentication** between the UE and the network
- The protocol uses symmetric key cryptography
- The UE has its secret K on the SIM card
- The operator stores their secrets K in the core network (HSS)

Authentication and Key Agreement AKA:

- Before the AKA, the RRC Connection Establishment takes place
- (Remember the Identity Mapping attack of last week, RNTIs, ...)
- In this process, the UE sends its ID towards the network
- The ID is used to check the correct individual information

Authentication and Key Agreement



Authentication and Key Agreement

(1) After connection was established, network sends the challenge C and authentication token $AUTH$

(2) Network generates individual $XRES$

(3) UE uses secret K to generate RES

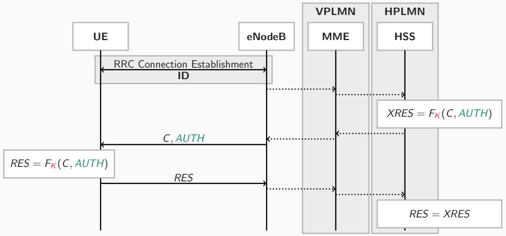
(4) Send RES towards network, where it's compared to $XRES$

Important:

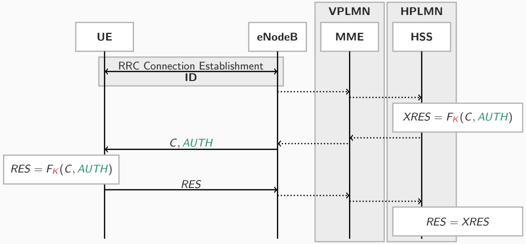
- The authentication token $AUTH$ authenticates the network towards the UE
- $RES = XRES$ authenticates the UE towards the network
- The eNodeB only does the communication. All important computations are done in the *core network*.

AKA Core Components

- Challenge C: Like a nonce
- Authentication Token AUTH: ID-specific
  - Sequence number, receives updates whenever used
  - In sync between HSS and UE
  - Authenticates network to UE
- Cryptographic function F: Generate tokens RES and $XRES$
- Secret K: Symmetric key



VoLTE AKA



- UE and Evolved Packet Core (EPC) use the AKA for mutual authentication.
- The IMS is a new component in this setting. It also uses mutual authentication.
- To achieve this, the User Equipment (UE) needs to go through a *second* AKA.

Second AKA

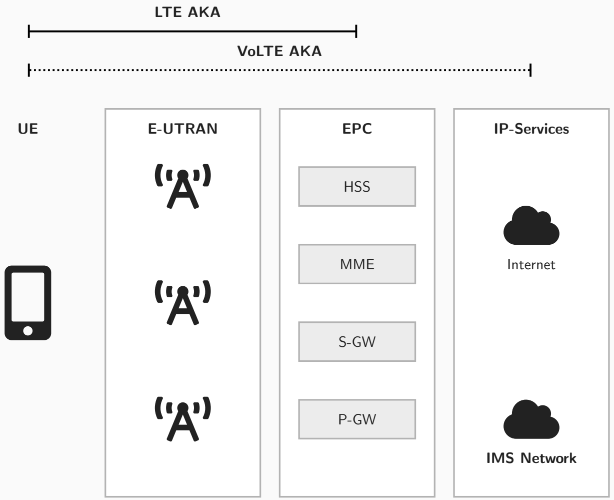
Initial Authentication and Key Agreement (AKA)

- Initial AKA authenticates UE and EPC
- After that, all traffic is encrypted.

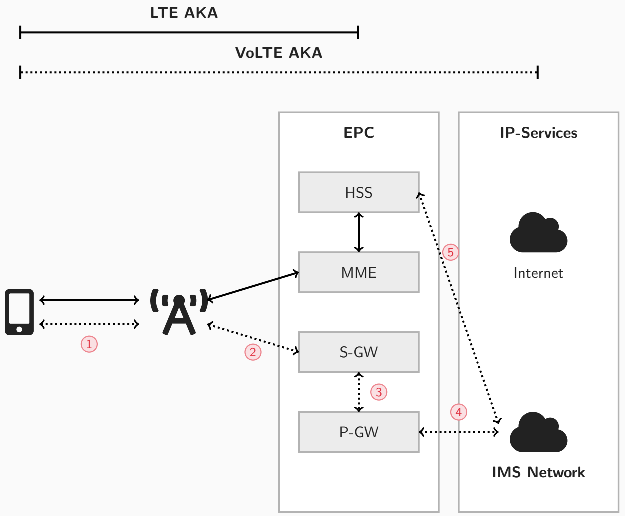
VoLTE AKA

- When the UE wants to make a VoLTE call, it establishes mutual authentication with the IMS.
- A second AKA takes place.
- All messages use the already authenticated LTE connection.

VoLTE AKA


LTE AKA
VoLTE AKA
UE
E-UTRAN
EPC
IP-Services
HSS
MME
S-GW
P-GW
Internet
IMS Network

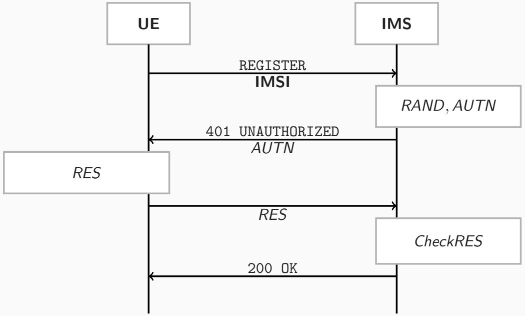
VoLTE AKA: Step by Step



VoLTE AKA

1. Radio connection between UE and eNodeB
2. VoLTE traffic is treated as user plane data $\rightarrow$ handled in Serving Gateway (S-GW).
3. PDN Gateway (P-GW) receives traffic and forwards it to the IP network.
4. IMS checks whether user is allowed to receive service.
5. Checks identity at HSS.

Session Initiation Protocol (SIP)



VoLTE AKA Protocol

- The VoLTE AKA uses SIP.
- UE sends *IMSI* for identification at the HSS.
- Caller and callee both authenticate.

Secure Real-Time Transport Protocol (SRTP)

SRTP

The SRTP adds encryption, message authentication and integrity, and replay attack protection to RTP.

SRTP Security

- SRTP would add an additional layer of encryption.
- This encryption happens on higher layers of the protocol stack.

VoLTE security measures are optional.
It is up to the provider to configure these features.

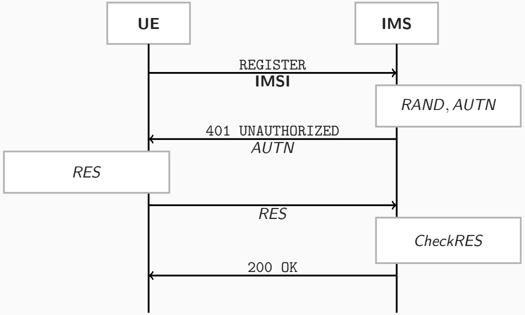
First Lecture: Mobile Lifecycle

$$\begin{tikzcd}\text{Specification} \arrow[r] & \text{Implementation} \arrow[r] & \text{Configuration}\end{tikzcd}$$

Define things on paper	**Transform it into code**	**Fine-tune the live network**
Things are flawed by definition.	All devices with this implementation are flawed.	Local network has a flawed configuration.
Update specification 😳	Patch implementation ☹	Patch configuration 😅

Why is the VoLTE AKA relevant?

The second AKA and SRTP enable additional encryption.



What we need to know:

- How often does the network apply the second VoLTE AKA?
- How often is SRTP used for an additional layer of encryption?

Challenges

We record a call and reconstruct the keystream from a second call.
These are the challenges:

Performance: How codecs & compression affect traffic.

IMS: RTP bearer as main data transportation.

Security: VoLTE AKA and encryption through SRTP.

Challenges: Summary

We record a call and reconstruct the keystream from a second call.
These are the challenges:

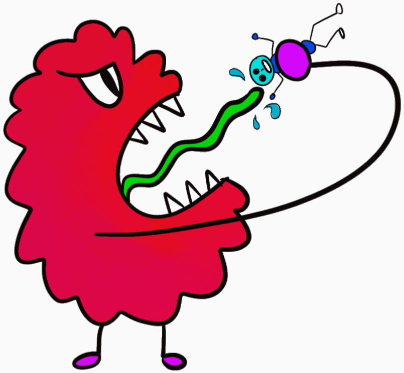
- **Attack:** Decrypting VoLTE calls by exploiting a keystream reuse.
- **We need to consider**:
  1. Performance: Codecs and compression and how this affects traffic
  2. IMS: Data bearers and the RTP bearer as main data transportation.
  3. Security: VoLTE AKA and encryption through SRTP.
- **Next:** Understanding the ReVoLTE attack

Challenges: Exam Examples

- What is Robust Header Compression?
- Why do we need to consider ROHC and codecs for the attack?
- What do we exploit for the ReVoLTE attack?
- Why does SRTP influence the attack?
- Why does VoLTE use a second AKA?
- What protocol is used for the VoLTE AKA?

Attack

Call me Maybe!



(1) Unclear specification

(2) Keystream reuse

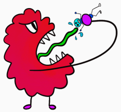
(3) Wireless sniffing

(4) Decrypt calls

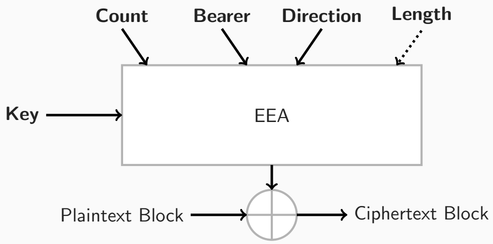
Target and Keystream Call

Target Call

Keystream Call



Stream Cipher

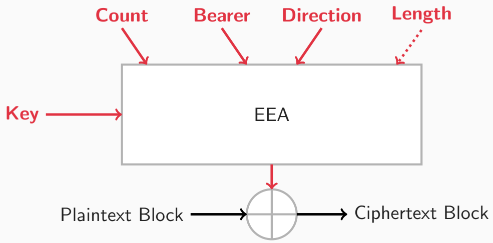


Stream Cipher

- **Key:** VoLTE user traffic key
- **Count:** Sequence number of packets
- **Bearer:** Bearer identity
- **Direction:** Uplink or downlink
- **Length:** Keystream block length

Same input generates **same** keystream!

Input and Output

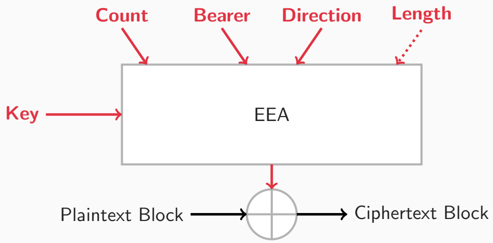


Same input generates **same** keystream!

$$(Plain_A \oplus Keystream) \oplus (Plain_B \oplus Keystream) = (Plain_A \oplus Plain_B)$$

$$(Plain_A \oplus Plain_B) \oplus Plain_B = Plain_A$$

Reconstructing the Plaintext



Adversary has control over $\boldsymbol{Plain_B}$

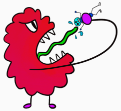
$$(Plain_A \oplus Keystream) \oplus (\boldsymbol{Plain_B} \oplus Keystream) = (Plain_A \oplus \boldsymbol{Plain_B})$$

$$(Plain_A \oplus \boldsymbol{Plain_B}) \oplus \boldsymbol{Plain_B} = Plain_A$$

Keystream Call $\rightarrow$ $Plain_B$

Target Call

Keystream Call



Roadmap

Two Attack Components:

(1) VoLTE data encrypted with a *stream cipher* 😳

(2) Keystream *reuse* 😳

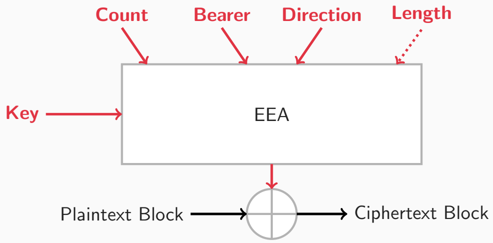
Attack Concept

(1) **Target Call**
- User makes a phone call
- Adversary monitors encrypted traffic
- Waits until call ends

(2) **Keystream Call**
- Call the victim
- Talk for same duration
- Monitor traffic

(3) **Reconstruct First Call**

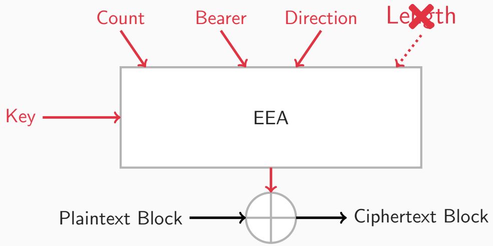
Keystream Generation



- **Key:** VoLTE user traffic key
- **Count:** Sequence number of packets
- **Bearer:** Bearer identity
- **Direction:** Uplink or downlink
- **Length:** Keystream block length

Step-by-step check of all inputs for the keystream

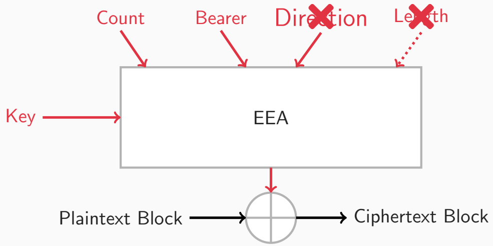
Length



Length

- Keystream block length.
- Does not influence the keystream generation.
- *Does not change the keystream.*

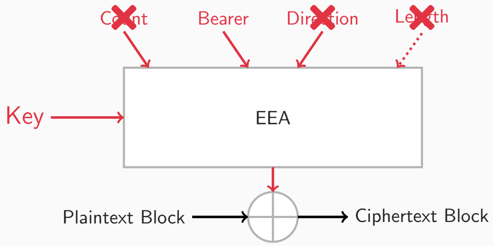
Direction



Direction

- Defines the direction of the traffic.
- Can either be uplink or downlink traffic.
- We monitor traffic in both directions.
- *Does not change the keystream.*

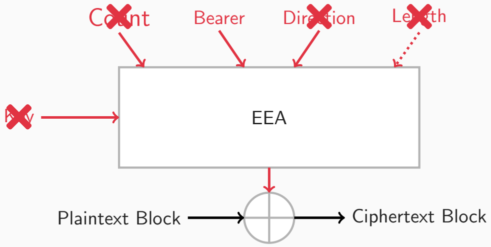
Key



Key

- Use plane key k_{up}
- Established for each new radio connection.
- If we manage to stay in the same session, k_{up} does not change.
- *Does not change the keystream.*

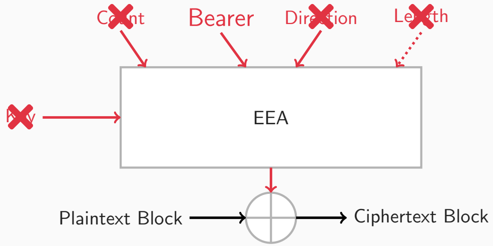
Count



Count

- Count consists of the PDCP sequence number and the PDCP hyperframe number.
- For each new call, the count will be reset
- *Does not change the keystream.*

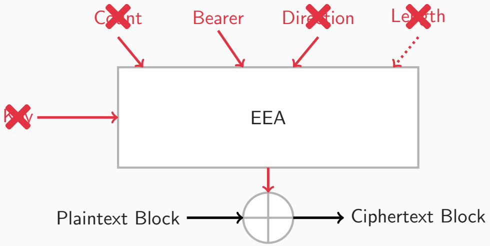
Bearer ID



Bearer ID

- Bearer ID for the RTP bearer can be in range 2..32
- If a new Bearer ID is used, the keystream changes.
- If the same Bearer ID is reused, it is an *implementation flaw.*
- *Might not change the keystream.*

Keystream Checklist



- **Key:** VoLTE user traffic key
- **Count:** Sequence number of packets
- **Bearer:** Bearer identity??
- **Direction:** Uplink or downlink
- **Length:** Keystream block length

What happens to the Bearer ID?

Demo: Bearer Identity

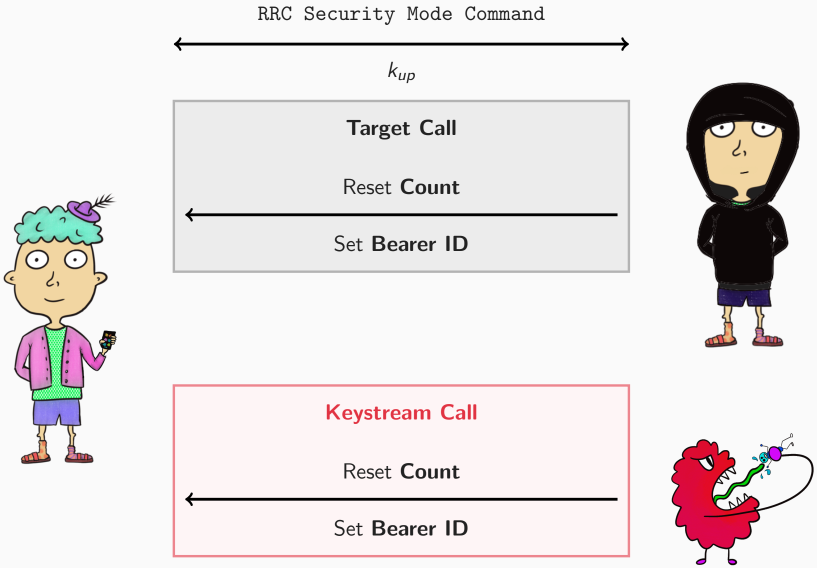
Demo:

ROHC Profiles

```
lte-rrc.drb_Identity  || (lte-rrc.c1 == 4)

# demo_revolte_not_possible.pcap
# demo_revolte_possible.pcap
```

Target and Keystream Call

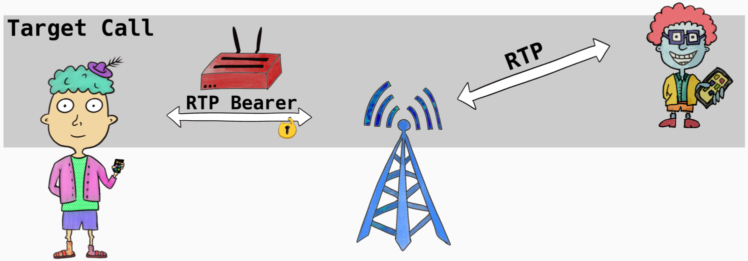


Vulnerable Networks

3/15 eNodeBs increase the Bearer ID

12/15 eNodeBs **reuse** the same keystream

Conducting the Attack



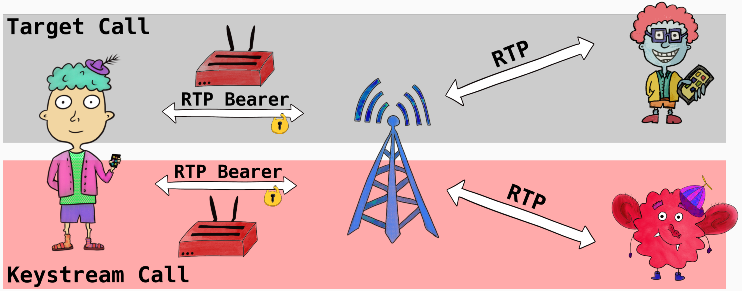
Target Call

(1) Victim places call

(2) Adversary monitors RTP bearer: Encrypted voice data

(3) Passive sniffer in the same radio cell

Conducting the Attack



Keystream Call

(1) Adversary calls victim immediately after target call

(2) Again, record RTP bearer *with same bearer ID*

(3) Reconstruct plaintext of first call

Reconstructing the Plaintext

$$(Plain_A \oplus Keystream) \oplus (\boldsymbol{Plain_B} \oplus Keystream) = (Plain_A \oplus \boldsymbol{Plain_B})$$
$$(Plain_A \oplus \boldsymbol{Plain_B}) \oplus \boldsymbol{Plain_B} = Plain_A$$

Controlling $\boldsymbol{Plain_B}$:

- $\boldsymbol{Plain_B}$ is the keystream call
- We replay something we know
- Example: Recorded conversation

Remember the Challenges?

- Avoid comfort noise
- Anticipate the codec
- Only works without SRTP

Attack: Summary

How the keystream reuse happens and how we conduct the attack.

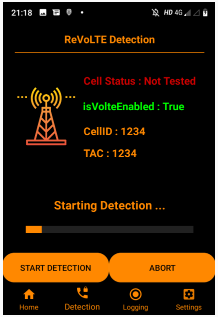
- **Keystream Generation**:
  - Direction and length are not really relevant.
  - User plane key remains the same for a session.
  - Count is reset, as defined in the specification.
  - Last remaining input: Bearer ID
- **Attack:** Record target call, place keystream call, reconstruct plaintext of first call.

Attackt: Exam Examples

- What is the critical component that causes the keystream reuse?
- How could we fix this attack vector?
- Is the ReVoLTE attack something that could happen in the real world?
- Can you remember the technical challenges we discussed in the beginning? Does everything make a little more sense now?
- Why is there a subsequent keystream call?

Results

- **Implementation flaw**
- Specification was ambiguous, this is resolved now
- Providers can patch the flaw if known


21:18

ReVoLTE Detection

Cell Status : Not Tested

isVolteEnabled : True

CellID : 1234

TAC : 1234

Starting Detection ...

www.revolte-attack.net

Acronyms

AKA	Authentication and Key Agreement
eNodeB	Evolved NodeB
EEA	EPS Encryption Algorithm
EPC	Evolved Packet Core
E-UTRAN	Evolved Universal Terrestrial Radio Access
HPLMN	Home PLMN
HSS	Home Subscriber Service
IMS	IP Multimedia Subsystem
MAC	Medium Access Control
MCC	Mobile Country Code
MME	Mobility Management Entity
MNC	Mobile Network Code
NAS	Non-Access Stratum
P-GW	PDN Gateway
PCRF	Policy and Charging Rules Function
PDCP	Packet Data Convergence Protocol
PDN	Packet Data Network
PHY	Physical Layer
RA-RNTI	Random Access RNTI
RLC	Radio Link Control
RNTI	Radio Network Temporary Identity
ROHC	Robust Header Compression
RRC	Radio Resource Control
RTP	Real-Time Transport Protocol
S-GW	Serving Gateway
SIP	Session Initiation Protocol
SRTP	Secure Real-Time Transport Protocol
UE	User Equipment
VPLMN	Visiting PLMN