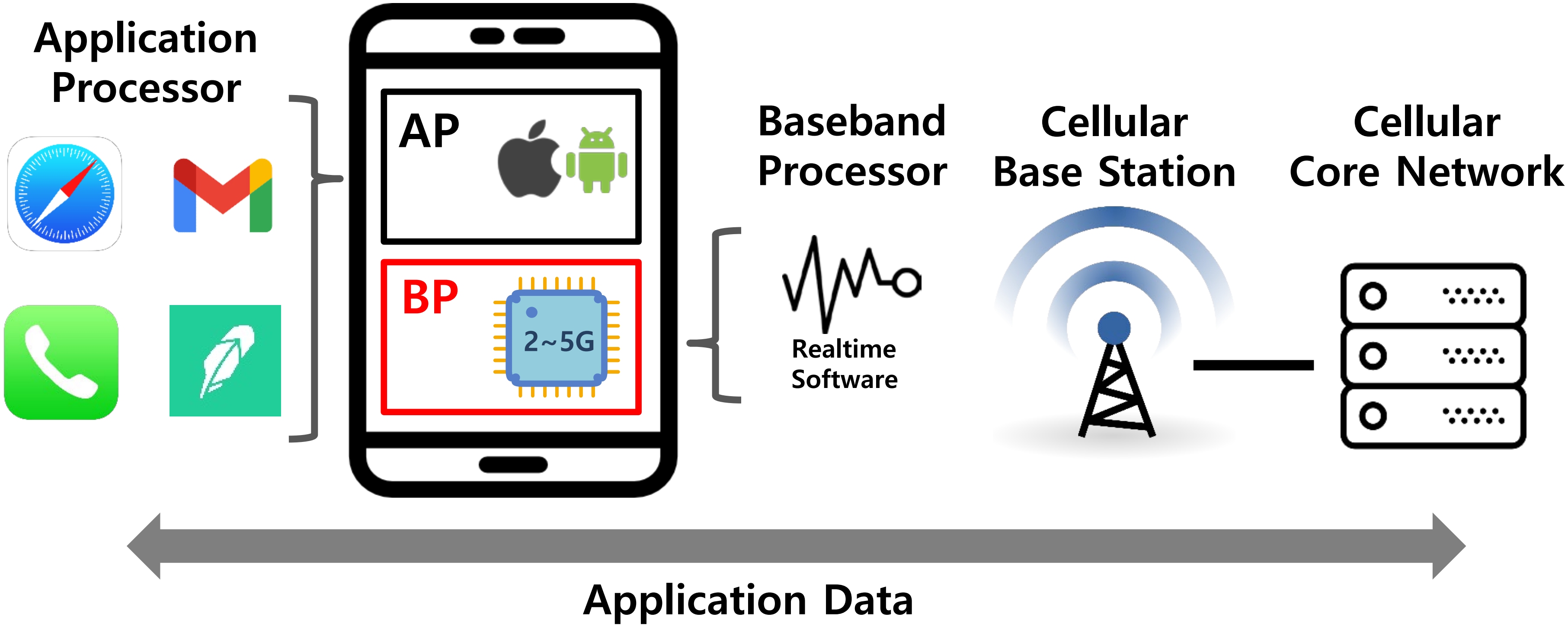
BaseSpec: Comparative Analysis of Baseband Software and Cellular Specifications for L3 Protocols

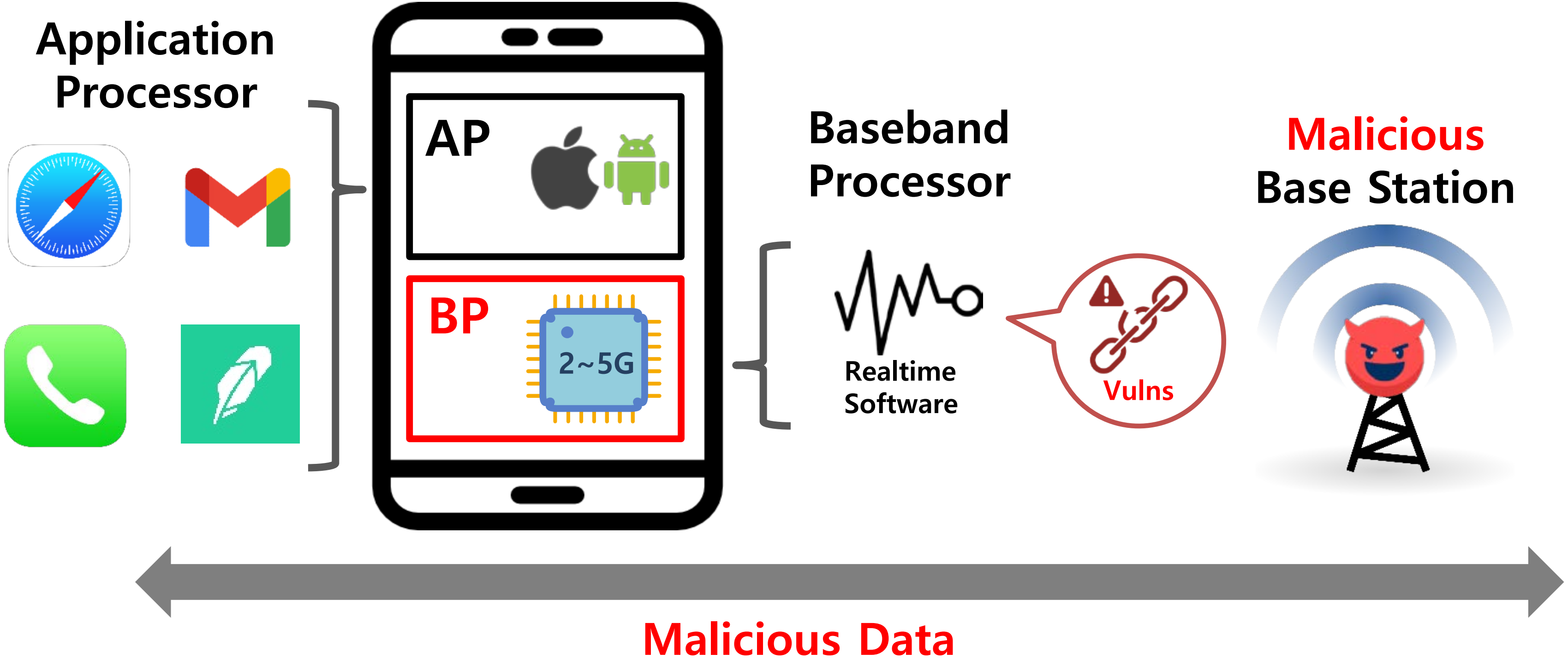
Eunsoo Kim*, Dongkwan Kim*, CheolJun Park,
Insu Yun, and Yongdae Kim
KAIST
NDSS '21

(*: co-first authors)

Processors in smartphone

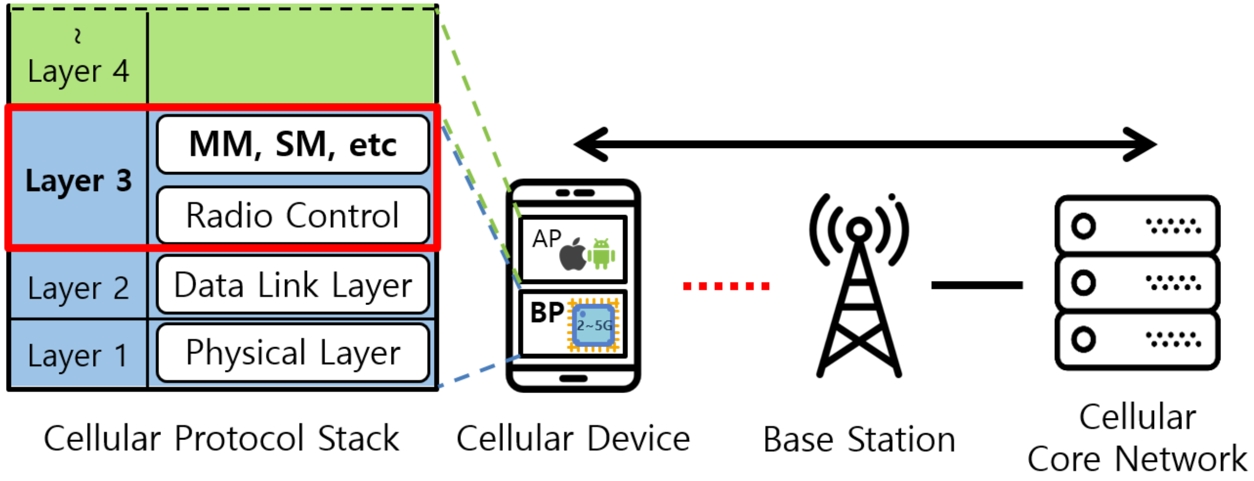


Baseband can be attacked!



Cellular Protocol Stack

- ❖ Baseband handles control plane protocols
  - ~100 documents (each has **hundreds of pages**)
- ❖ **Layer 3 (L3)** includes core procedures
  - Call control (CC), Mobility management (MM), session management (SM)
- ❖ Multiple **vulnerabilities** have been found in Layer 3



Analyzing Baseband Security

- Challenge: **Obscurity -** vendors do not release details of baseband

- Manual analysis
  - Baseband Attacks (Weinmann, WOOT'12)
  - Breaking Band (Golde et al., REcon'16)
  - A walk with Shannon (Cama, OPCDE'18)

  ➔ **Limited scalability** and **applicability**
    - Numerous functions (over 90K) for processing hundreds of messages
    - Diverse firmware versions and device models

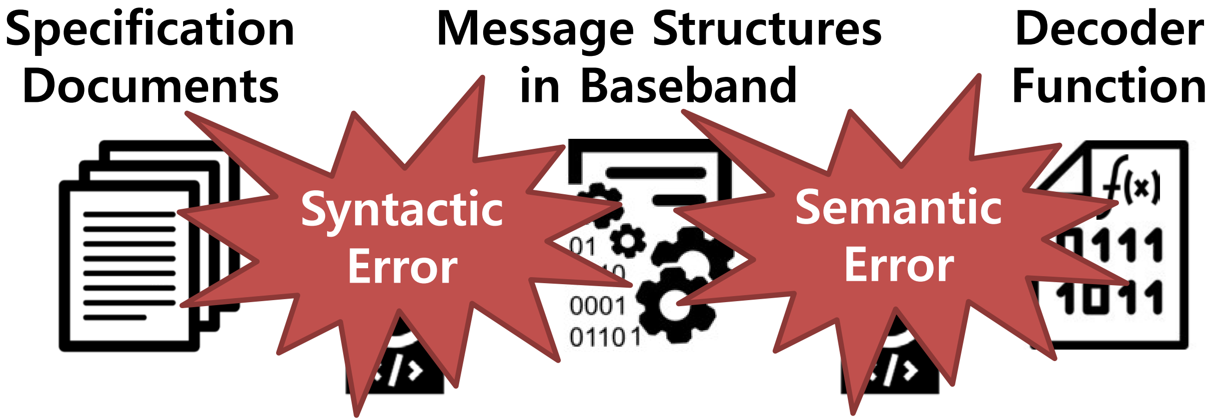
- Dynamic analysis
  - SMS of Death (Mulliner et al., Security'11)
  - Security testing of GSM implementations (Broek et al., ESSoS'14)
  - BaseSAFE (Maier et al., WiSec'20)

  ➔ **Hard to automate**
    - Numerous non-trivial operations (e.g., mobility, session, call, …)
    - Dynamic analysis finds only shallow bugs (e.g., crash)

Observation

- Baseband is software for network communication
  - Receive radio signals
  - **Decode** messages
  - Send responses or update states

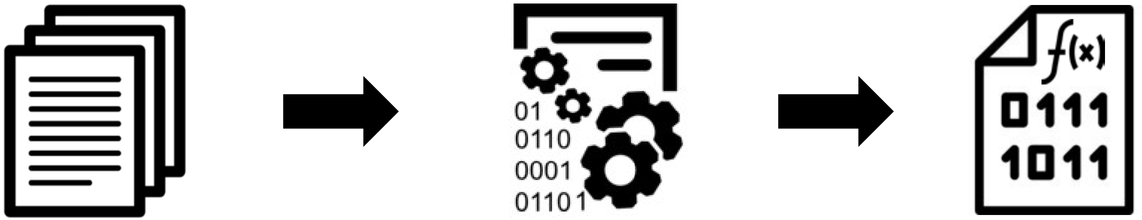
- **Decoder** should implement protocol specifications (hundreds of messages)



Our Approach - BaseSpec

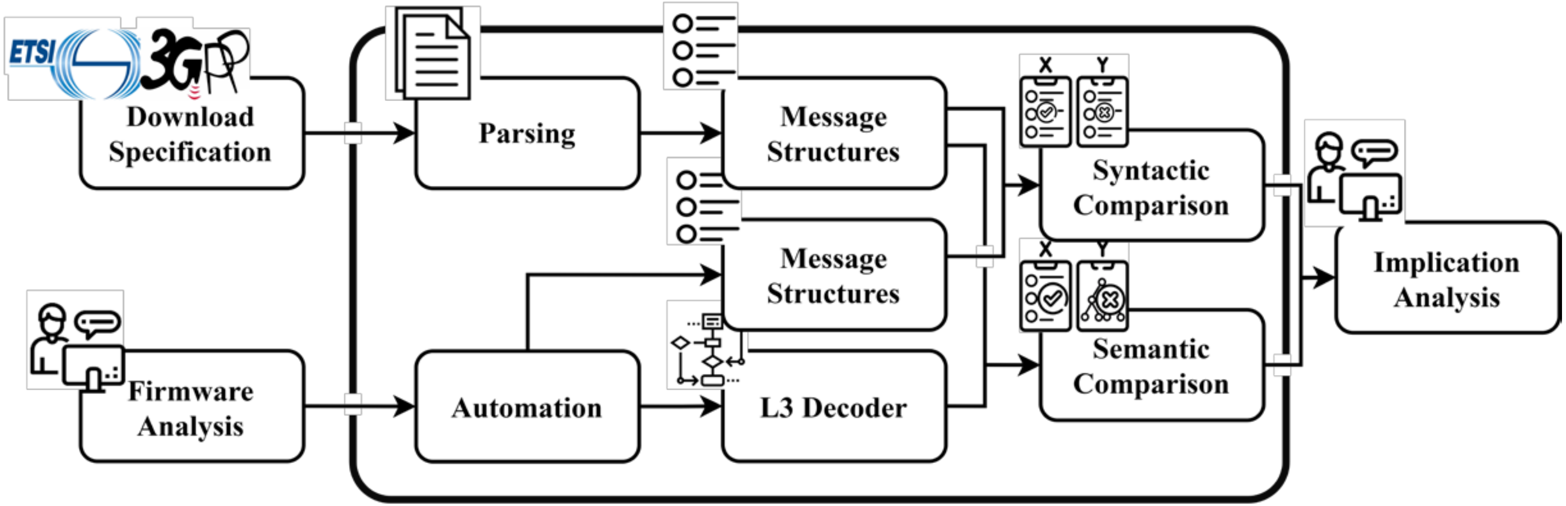
- **Comparative analysis** of baseband and specification
  - Focusing on protocol decoding logic

Specification Documents | **Message Structures in Baseband** | **Decoder Function**



- Message structures are embedded in a **machine-friendly** form
  - ➔ Comparing the structures with the documented specification can be **automated**
- Main decoding logic **rarely changes**
  - ➔ Once analyzed, **applicable** to various firmware versions and device models

BaseSpec Overview



BaseSpec System

Processing Specification

Specification Documents

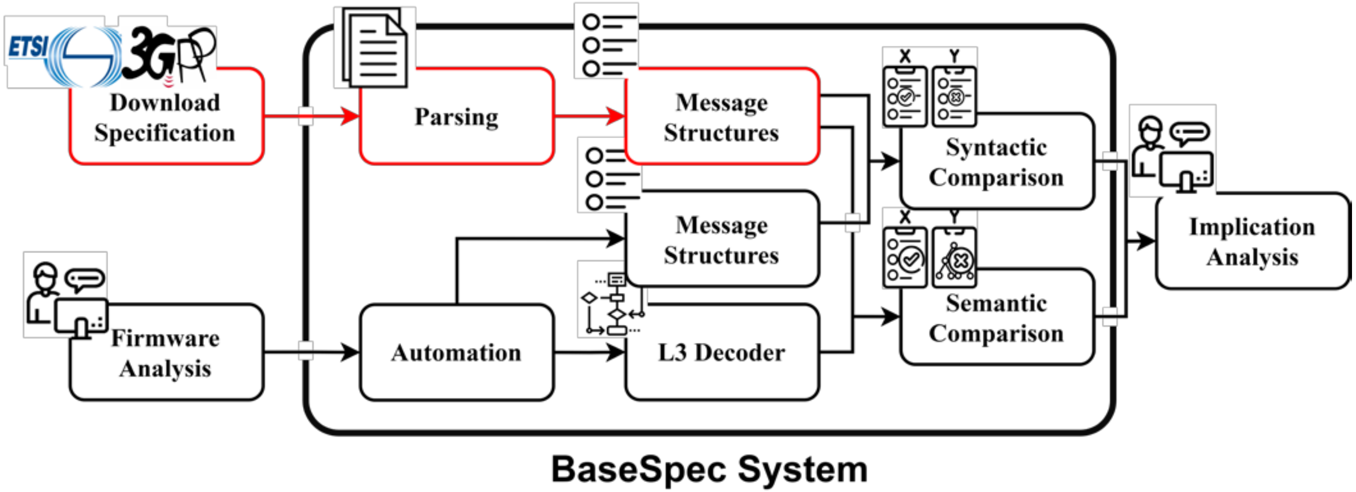
Message Structures in Baseband

Decoder Function

→ Ground Truth

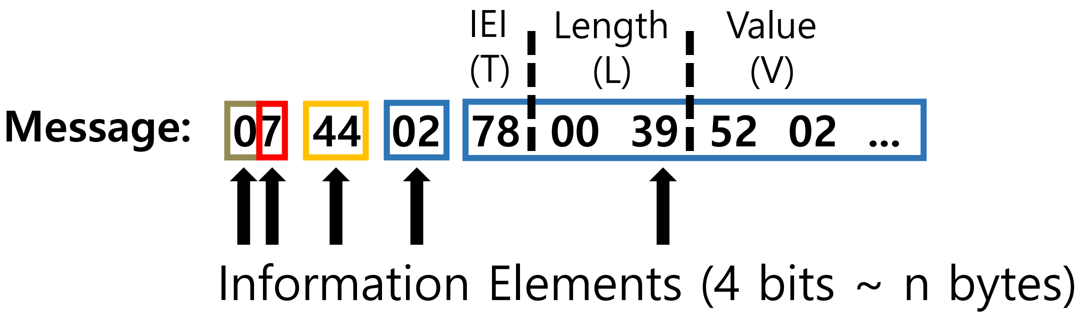
Extracting Msg. Structures from Spec.

1) Download spec documents from the 3GPP (.doc) and ETSI (.pdf) websites
2) Convert documents to raw text
3) Handle inconsistencies (documents are written in a natural language)
4) Parse message structures



Standard L3 Messages

- Have a standardized form



- Message: set of Information Element (IE)
- An IE can have three elements
  - IEI: IE Identifier (T), Length (L), Value (V)
- An IE can be mandatory or optional

Specification TS 24.301 - EMM (0x7)

Table 8.2.3.1: ATTACH REJECT message content

IEI	Information Element	Type/Reference	Presence	Format	Length
	Protocol discriminator	Protocol discriminator 9.2	M	V	1/2
	Security header type	Security header type 9.3.1	M	V	1/2
	Attach reject message identity	Message type 9.8	M	V	1
	EMM cause	EMM cause 9.9.3.9	M	V	1
78	ESM message container	ESM message container 9.9.3.15	O	TLV-E	6-n
		⋮			

Specification TS 24.301 - EMM (0x7)

Table 8.2.3.1: ATTACH REJECT message content

IEI	Information Element	Type/Reference	Presence	Format	Length
	Protocol discriminator	Protocol discriminator 9.2	M	V	1/2
	Security header type	Security header type 9.3.1	M	V	1/2
	Attach reject message identity	Message type 9.8	M	V	1
	EMM cause	EMM cause 9.9.3.9	M	V	1
78	ESM message container	ESM message container 9.9.3.15	O	TLV-E	6-n
	⋮				

IE: Information Element

Specification TS 24.301 - EMM (0x7)

Table 8.2.3.1: ATTACH REJECT message content

IEI	Information Element	Type/Reference	Presence	Format	Length
	Protocol discriminator	Protocol discriminator 9.2	M	V	1/2
	Security header type	Security header type 9.3.1	M	V	1/2
	Attach reject message identity	Message type 9.8	M	V	1
	EMM cause	EMM cause 9.9.3.9	M	V	1
78	ESM message container	ESM message container 9.9.3.15	O	TLV-E	6-n
		⋮			

IE: Information Element

Presence: Mandatory (M), Optional (O)

IEI: Information Element Identifier

Specification TS 24.301 - EMM (0x7)

Table 8.2.3.1: ATTACH REJECT message content

IEI	Information Element	Type/Reference	Presence	Format	Length
	Protocol discriminator	Protocol discriminator 9.2	M	V	1/2
	Security header type	Security header type 9.3.1	M	V	1/2
	Attach reject message identity	Message type 9.8	M	V	1
	EMM cause	EMM cause 9.9.3.9	M	V	1
78	ESM message container	ESM message container 9.9.3.15	O	TLV-E	6-n
		⋮			

IE: Information Element

Presence: Mandatory (M), Optional (O)

IEI: Information Element Identifier

Format: IEI (T), Length (L), Value (V),
Extended (-E)

Specification TS 24.301 - EMM (0x7)

Specification Documents

Table 8.2.1.1: ATTACH ACCEPT message content

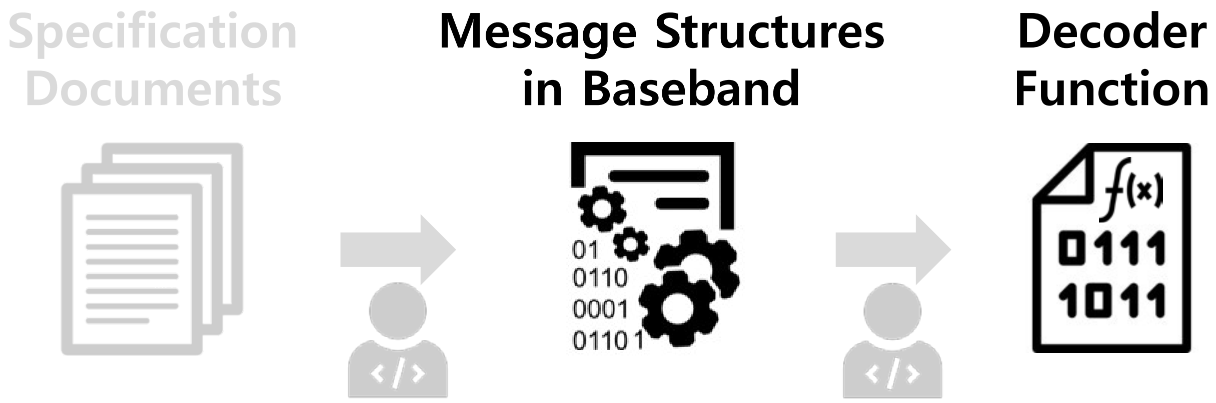
Table 8.2.2.1: ATTACH COMPLETE message content

Table 8.2.3.1: ATTACH REJECT message content

Table 8.2.4.1: ATTACH REQUEST message content

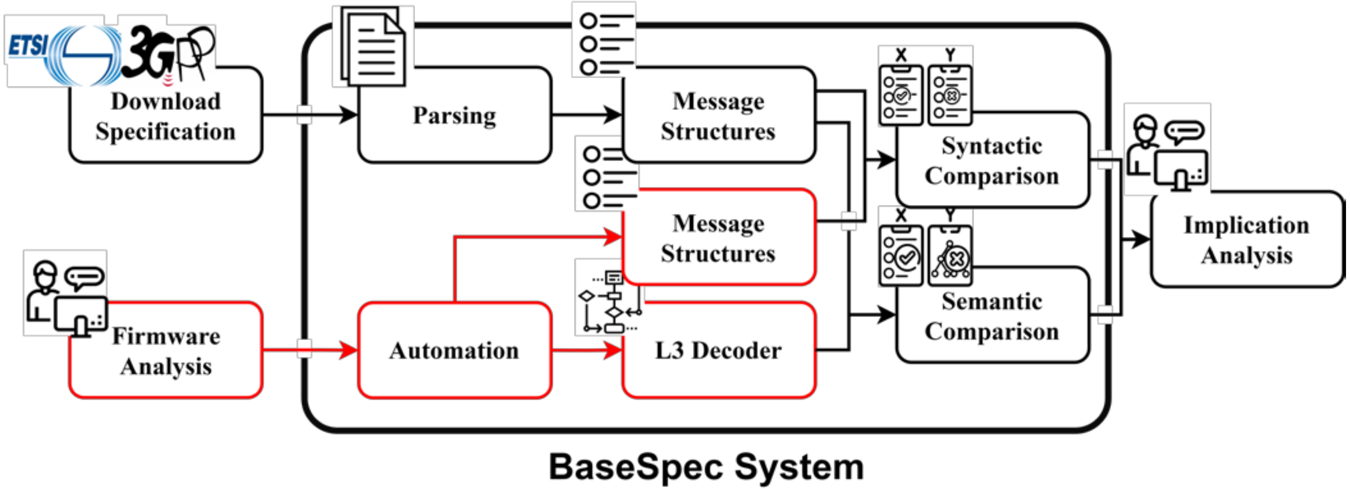
IEI	Information Element	Type/Reference	Presence	Format	Length
	Protocol discriminator	Protocol discriminator 9.2	M	V	1/2
	Security header type	Security header type 9.3.1	M	V	1/2
	Attach request message identity	Message type 9.8	M	V	1
	EPS attach type	EPS attach type 9.9.3.11	M	V	1/2
	NAS key set identifier	NAS key set identifier 9.9.3.21	M	V	1/2
	EPS mobile identity	EPS mobile identity 9.9.3.12	M	LV	5-12
	UE network capability	UE network capability 9.9.3.34	M	LV	3-14
	ESM message container	ESM message container 9.9.3.15	M	LV-E	5-n
19	Old P-TMSI signature	P-TMSI signature 9.9.3.26	O	TV	4

Processing Firmware



→ **Our Analysis Target**

Firmware Analysis



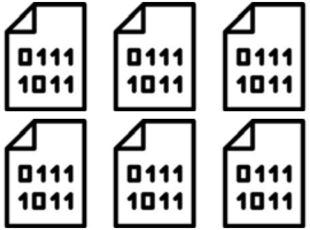
Firmware Analysis

- Challenge
  - **Obscurity** - vendors do not open firmware details
- Target
  - Firmware from 2 major vendors (architecture: ARM)
- Method
  - Manual analysis to uncover the firmware's obscurity
  - Extract decoder function and message structure information

➔ After one-time **manual** analysis, can be **automated**

Firmware Analysis

- ❖ Preprocessing
  - Firmware extraction
  - Memory layout analysis
  - Function boundary identification
- ❖ Identify the L3 decoder
  - Utilize debug information
    - "L3", “Decode", "EMM", …
- ❖ Analyze embedded specification
  (= embedded message structures)



Typical Firmware
(Multiple Binaries)

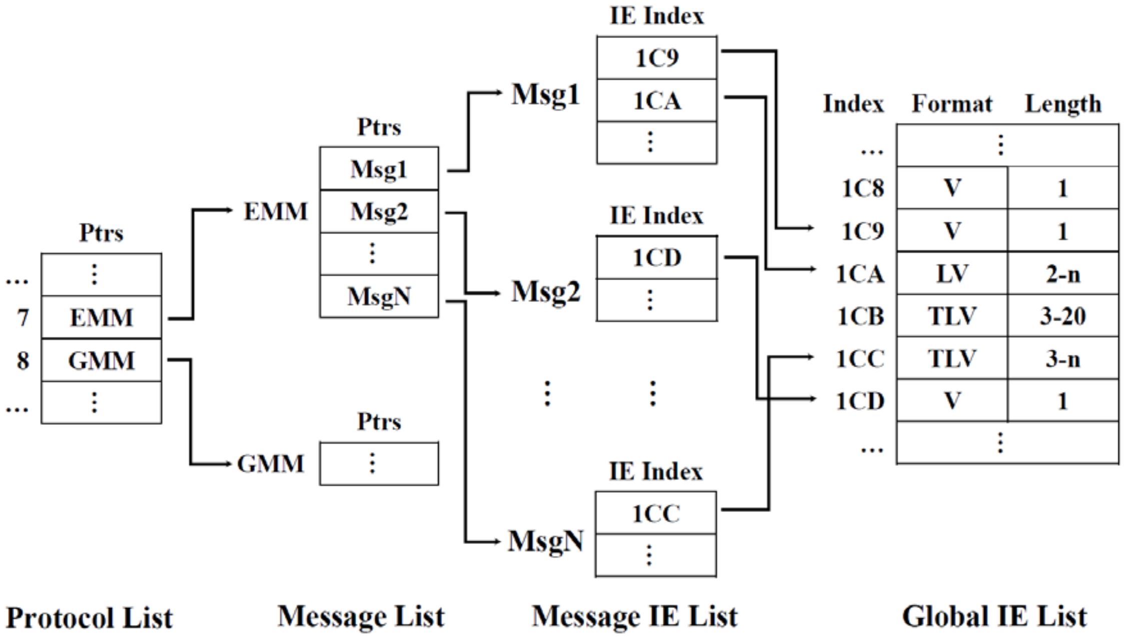
Baseband Firmware
(Single Binary)

```
DCD 0xFECDBA98
DCD aWarnDecodeErro        ; "Warn>Decode Error: 0x%x"
DCD 0xC28
DCD asc_4156E7E0           ; "../../../CALPSS/LteL3/LteSae/
```

Sample Debug Information

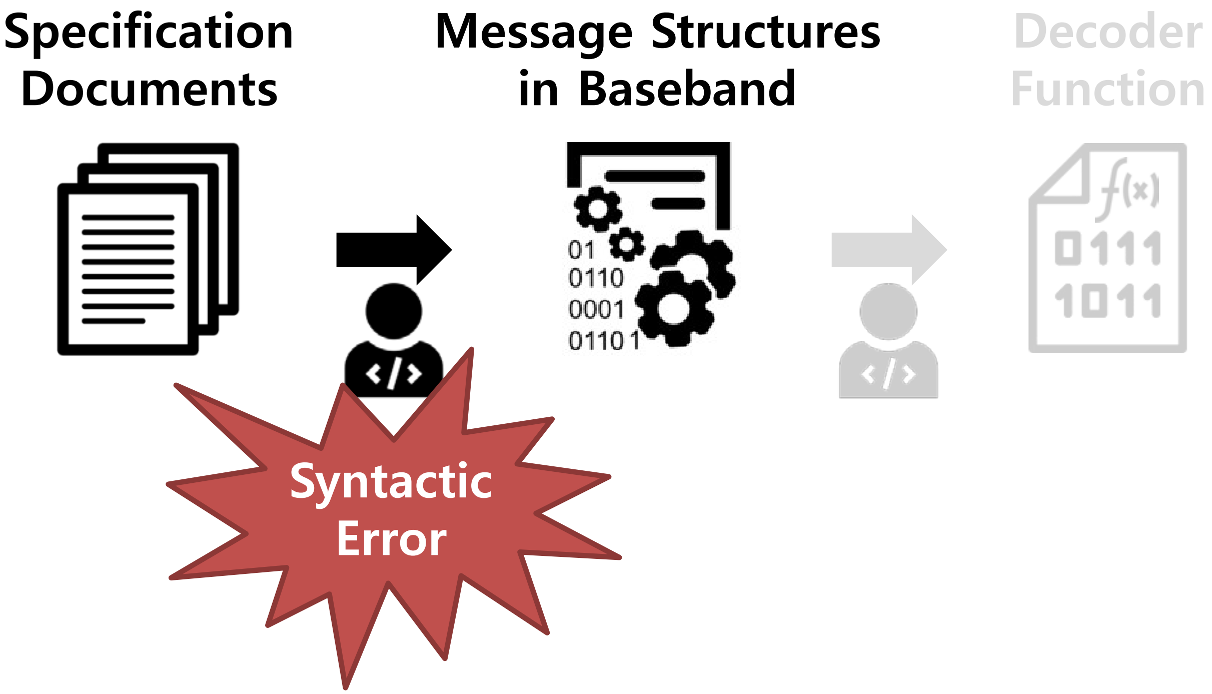
Msg. Structures in Vendor_1 Firmware

- 4 types of linked lists



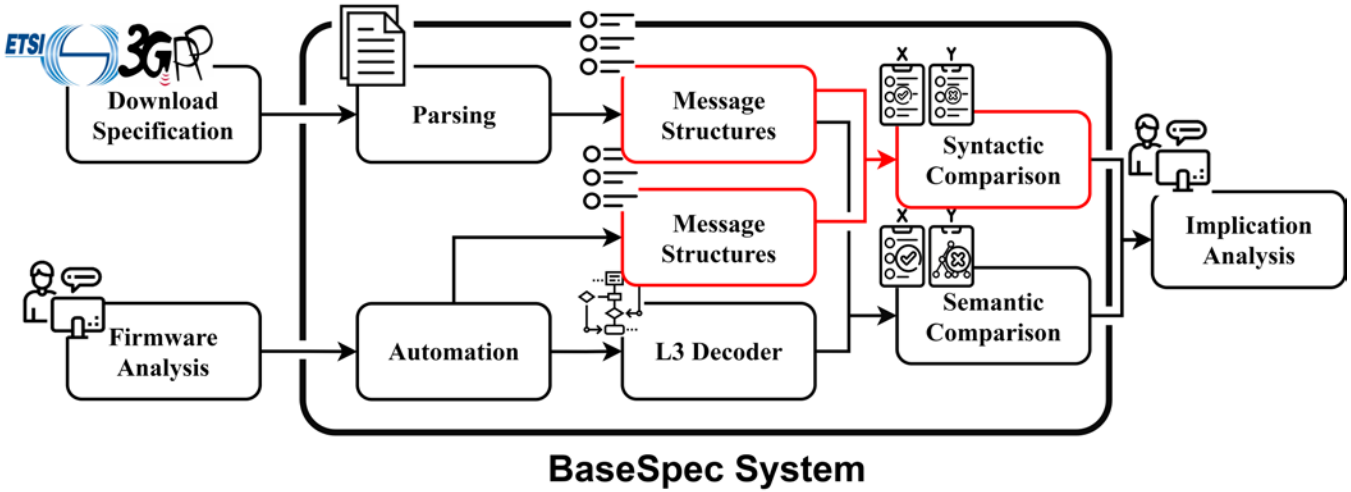
➔ Contains IE information for every implemented message

Finding Syntactic Error



Syntactic Comparison

- Check whether the embedded structures are correct
  - Directly indicate developers' mistakes (e.g., mistyping length)



SysSec
System Security Lab

Syntactic Comparison

- ❖ Check existence / length
- ❖ Correct
- ❖ Invalid Mismatch
  - – Incorrect length
- ❖ Missing Mismatch
  - – Not in firmware
- ❖ Unknown Mismatch
  - – Only in firmware

From Specification

IEI	IE Name	Format	Value Length
-	Protocol disc…	V	1/2
-	Security header…	V	1/2
-	Attach reject …	V	1
-	EMM cause	V	1
78	ESM …	TLV-E	**3-n**
5F	T3346 value	TLV	1

Correct

Invalid

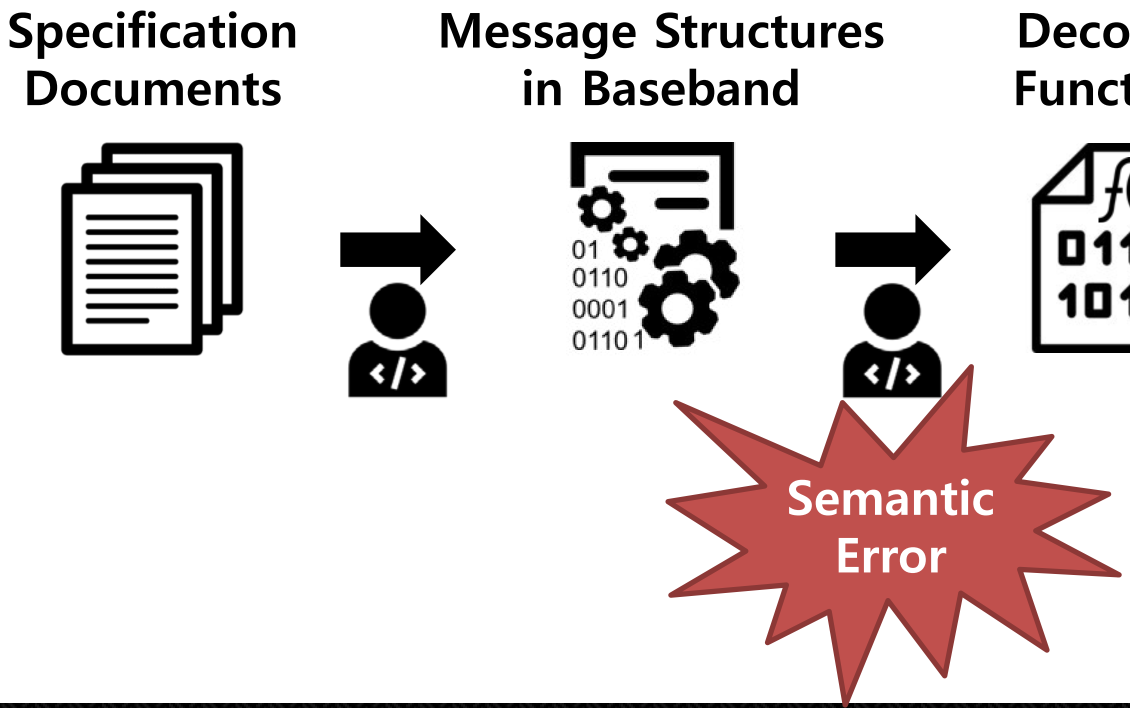
Missing

Unknown

From Firmware

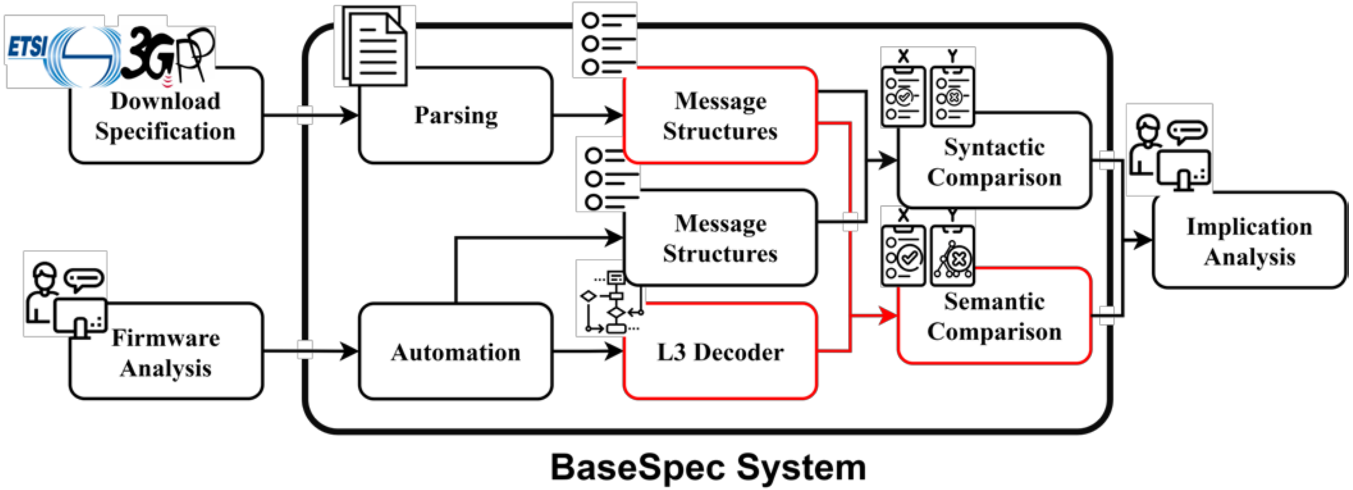
IEI	Value Length
-	-
-	-
-	-
-	1
78	**0-n**
-	-
FF	1

Finding Semantic Error



Semantic Comparison

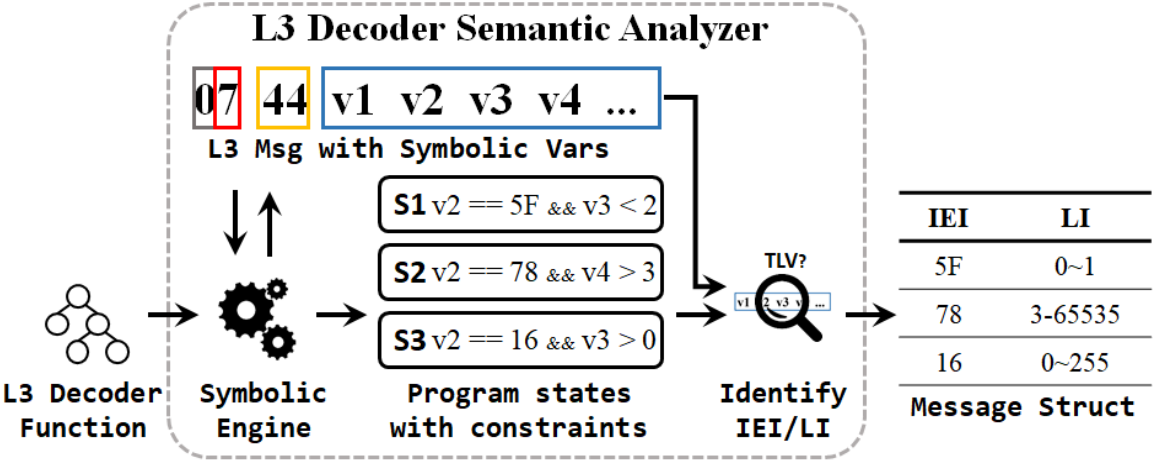
- Check whether the decoder operates correctly
  - Can identify missing logic (e.g., length check) or exceptional cases
- Use symbolic execution to analyze the decoding logic



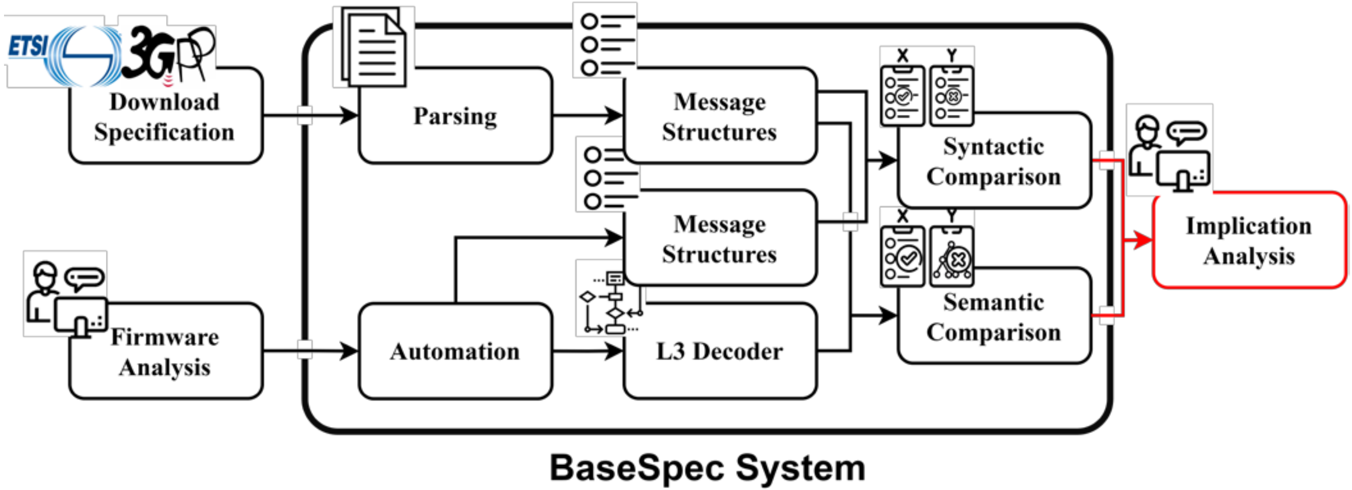
SysSec System Security Lab

Semantic Analysis

- Run symbolic execution on the decoder function
  - Collect symbolic variables and constraints
    - Created when the decoder checks IE Identifiers (IEIs) or lengths
  - Identify IE Identifier (IEI) and Length Indicator (LI) and build message structures
    - Compared with specifications to find mismatches (invalid, missing, and unknown)



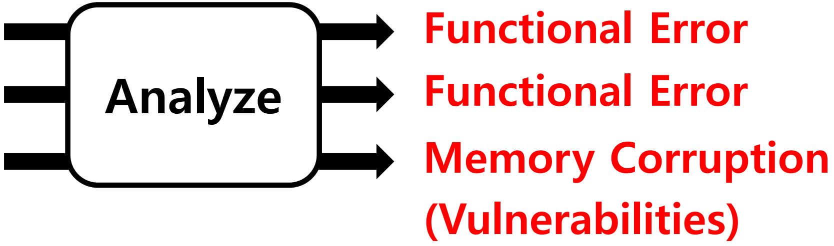
Implication Analysis



SysSec
System Security Lab

Implication Analysis

- Comparison reports mismatches
  - **Missing**: IEs not in firmware
  - **Unknown**: IEs only in firmware
  - **Invalid**: IEs with incorrect lengths



- Some mismatches may not cause errors
  - Additional check routines after decoder function
  - Optional to implement
  - ➔ **Manual** analysis is required
    But mismatches can pinpoint erroneous parts

Evaluation

- Implemented a prototype with IDA Pro and angr
- Target
  - 2 major vendors (ARM)
  - 18 firmware images from $\textbf{Vendor}_1$ (9 models × 2 versions)
  - 3 firmware images from $\textbf{Vendor}_2$ (3 models)
- Details are anonymized upon the vendor's request
  - We reported all the findings to the vendors

Evaluation Results

- Hundreds of mismatches are reported from every firmware

- Implication analysis results
  - $Vendor_1$
    - 5 Functional Errors
    - 4 Memory-related vulnerabilities
      ➔ 2 critical Remote Code Execution (RCE) vulnerabilities
  - $Vendor_2$
    - 1 Memory-related vulnerability

Mismatch Results – $Vendor_1$

			In Binary		Common Mismatch						Syntactic-only Mismatch						Semantic-only Mismatch						Case Study Results								
			# of		Missing		Unknown		Invalid		Missing		Unknown		Invalid		Missing		Unknown		Invalid		Functional†					Memory-related			
	Model	Build Date	Msgs	IEs	i-IE	n-IE	i-IE	n-IE	i-IE	n-IE	i-IE	n-IE	i-IE	n-IE	i-IE	n-IE	i-IE	n-IE	i-IE	n-IE	i-IE	n-IE	E1	E2	E3	E4	E5	E6‡	E7	E8‡	E9
Latest Firmware	Model A	May/2020	268	1204	1	164	0	36	38	109	3	19	6	13	21	52	1	6	0	9	35	203	✓	✓	✓	✓	✓	✓	·	✓	✓
	Model B	May/2020	268	1201	1	167	0	36	38	109	3	19	6	13	21	52	1	6	0	9	35	200	✓	✓	✓	✓	✓	✓	·	✓	✓
	Model C	May/2020	268	1201	1	167	0	36	38	109	3	19	6	13	21	52	1	6	0	9	35	200	✓	✓	✓	✓	✓	✓	·	✓	✓
	Model D	Jun/2020	268	1200	1	179	0	36	41	111	3	18	6	13	21	52	1	6	0	9	32	186	✓	✓	✓	✓	✓	✓	·	✓	✓
	Model E	Jun/2020	268	1200	1	179	0	36	41	111	3	18	6	13	21	52	1	6	0	9	32	186	✓	✓	✓	✓	✓	✓	·	✓	✓
	Model F	Apr/2020	268	1198	1	179	0	36	41	111	3	18	6	13	21	52	1	6	0	9	32	186	✓	✓	✓	✓	✓	✓	·	✓	✓
	Model G	Apr/2020	268	1198	1	179	0	36	41	111	3	18	6	13	21	52	1	6	0	9	32	186	✓	✓	✓	✓	✓	✓	·	✓	✓
	Model H	Apr/2020	263	1096	1	212	0	3	40	39	4	19	8	34	21	118	1	327	0	1	32	71	·	✓	✓	·	·	·	·	✓	✓
	Model I	Apr/2020	263	1096	1	212	0	3	40	39	4	19	8	34	21	118	1	327	0	1	32	71	·	✓	✓	·	·	·	·	✓	✓

*IE: Information Element (= message field)   i-IE: imperative (≈mandatory) IE   n-IE: non-imperative (≈optional) IE

Mismatch Results – $Vendor_1$

- Missing imperative (≈mandatory) & Unknown IEs
  - Directly indicate **functional errors**

			In Binary		Common Mismatch						Syntactic-only Mismatch						Semantic-only Mismatch						Case Study Results								
			# of		Missing		Unknown		Invalid		Missing		Unknown		Invalid		Missing		Unknown		Invalid		Functional†					Memory-related			
	Model	Build Date	Msgs	IEs	i-IE	n-IE	i-IE	n-IE	i-IE	n-IE	i-IE	n-IE	i-IE	n-IE	i-IE	n-IE	i-IE	n-IE	i-IE	n-IE	i-IE	n-IE	E1	E2	E3	E4	E5	E6‡	E7	E8‡	E9
Latest Firmware	Model A	May/2020	268	1204	1	164	0	36	38	109	3	19	6	13	21	52	1	6	0	9	35	203	✓	✓	✓	✓	✓	✓	·	✓	✓
	Model B	May/2020	268	1201	1	167	0	36	38	109	3	19	6	13	21	52	1	6	0	9	35	200	✓	✓	✓	✓	✓	✓	·	✓	✓
	Model C	May/2020	268	1201	1	167	0	36	38	109	3	19	6	13	21	52	1	6	0	9	35	200	✓	✓	✓	✓	✓	✓	·	✓	✓
	Model D	Jun/2020	268	1200	1	179	0	36	41	111	3	18	6	13	21	52	1	6	0	9	32	186	✓	✓	✓	✓	✓	✓	·	✓	✓
	Model E	Jun/2020	268	1200	1	179	0	36	41	111	3	18	6	13	21	52	1	6	0	9	32	186	✓	✓	✓	✓	✓	✓	·	✓	✓
	Model F	Apr/2020	268	1198	1	179	0	36	41	111	3	18	6	13	21	52	1	6	0	9	32	186	✓	✓	✓	✓	✓	✓	·	✓	✓
	Model G	Apr/2020	268	1198	1	179	0	36	41	111	3	18	6	13	21	52	1	6	0	9	32	186	✓	✓	✓	✓	✓	✓	·	✓	✓
	Model H	Apr/2020	263	1096	1	212	0	3	40	39	4	19	8	34	21	118	1	327	0	1	32	71	·	✓	✓	·	·	·	·	✓	✓
	Model I	Apr/2020	263	1096	1	212	0	3	40	39	4	19	8	34	21	118	1	327	0	1	32	71	·	✓	✓	·	·	·	·	✓	✓

*IE: Information Element (= message field)   i-IE: imperative (≈mandatory) IE   n-IE: non-imperative (≈optional) IE

Mismatch Results – $Vendor_1$

- ❖ Missing imperative (≈mandatory) & Unknown IEs
  - Directly indicate **functional errors**
- ❖ Invalid IEs
  - Numerous incorrect length limit / ad-hoc length checks after decoder function
  - Can lead to **memory-related bugs**

			In Binary		Common Mismatch						Syntactic-only Mismatch						Semantic-only Mismatch						Case Study Results								
			# of		Missing		Unknown		Invalid		Missing		Unknown		Invalid		Missing		Unknown		Invalid		Functional†					Memory-related			
	Model	Build Date	Msgs	IEs	i-IE	n-IE	i-IE	n-IE	i-IE	n-IE	i-IE	n-IE	i-IE	n-IE	i-IE	n-IE	i-IE	n-IE	i-IE	n-IE	i-IE	n-IE	E1	E2	E3	E4	E5	E6‡	E7	E8‡	E9
Latest Firmware	Model A	May/2020	268	1204	1	164	0	36	38	109	3	19	6	13	21	52	1	6	0	9	35	203	✓	✓	✓	✓	✓	✓	·	✓	✓
	Model B	May/2020	268	1201	1	167	0	36	38	109	3	19	6	13	21	52	1	6	0	9	35	200	✓	✓	✓	✓	✓	✓	·	✓	✓
	Model C	May/2020	268	1201	1	167	0	36	38	109	3	19	6	13	21	52	1	6	0	9	35	200	✓	✓	✓	✓	✓	✓	·	✓	✓
	Model D	Jun/2020	268	1200	1	179	0	36	41	111	3	18	6	13	21	52	1	6	0	9	32	186	✓	✓	✓	✓	✓	✓	·	✓	✓
	Model E	Jun/2020	268	1200	1	179	0	36	41	111	3	18	6	13	21	52	1	6	0	9	32	186	✓	✓	✓	✓	✓	✓	·	✓	✓
	Model F	Apr/2020	268	1198	1	179	0	36	41	111	3	18	6	13	21	52	1	6	0	9	32	186	✓	✓	✓	✓	✓	✓	·	✓	✓
	Model G	Apr/2020	268	1198	1	179	0	36	41	111	3	18	6	13	21	52	1	6	0	9	32	186	✓	✓	✓	✓	✓	✓	·	✓	✓
	Model H	Apr/2020	263	1096	1	212	0	3	40	39	4	19	8	34	21	118	1	327	0	1	32	71	·	✓	✓	·	·	·	·	✓	✓
	Model I	Apr/2020	263	1096	1	212	0	3	40	39	4	19	8	34	21	118	1	327	0	1	32	71	·	✓	✓	·	·	·	·	✓	✓

*IE: Information Element (= message field) i-IE: imperative (≈mandatory) IE n-IE: non-imperative (≈optional) IE

Mismatch Results – $Vendor_1$

- Missing imperative (≈mandatory) & Unknown IEs
  - Directly indicate **functional errors**
- Invalid IEs
  - Numerous incorrect length limit / ad-hoc length checks after decoder function
  - Can lead to **memory-related bugs**
- Missing non-imperative (≈optional) IEs
  - May not be buggy

			In Binary		Common Mismatch						Syntactic-only Mismatch						Semantic-only Mismatch						Case Study Results								
			# of		Missing		Unknown		Invalid		Missing		Unknown		Invalid		Missing		Unknown		Invalid		Functional†					Memory-related			
	Model	Build Date	Msgs	IEs	i-IE	n-IE	i-IE	n-IE	i-IE	n-IE	i-IE	n-IE	i-IE	n-IE	i-IE	n-IE	i-IE	n-IE	i-IE	n-IE	i-IE	n-IE	E1	E2	E3	E4	E5	E6‡	E7	E8‡	E9
Latest Firmware	Model A	May/2020	268	1204	1	164	0	36	38	109	3	19	6	13	21	52	1	6	0	9	35	203	✓	✓	✓	✓	✓	✓	·	✓	✓
	Model B	May/2020	268	1201	1	167	0	36	38	109	3	19	6	13	21	52	1	6	0	9	35	200	✓	✓	✓	✓	✓	✓	·	✓	✓
	Model C	May/2020	268	1201	1	167	0	36	38	109	3	19	6	13	21	52	1	6	0	9	35	200	✓	✓	✓	✓	✓	✓	·	✓	✓
	Model D	Jun/2020	268	1200	1	179	0	36	41	111	3	18	6	13	21	52	1	6	0	9	32	186	✓	✓	✓	✓	✓	✓	·	✓	✓
	Model E	Jun/2020	268	1200	1	179	0	36	41	111	3	18	6	13	21	52	1	6	0	9	32	186	✓	✓	✓	✓	✓	✓	·	✓	✓
	Model F	Apr/2020	268	1198	1	179	0	36	41	111	3	18	6	13	21	52	1	6	0	9	32	186	✓	✓	✓	✓	✓	✓	·	✓	✓
	Model G	Apr/2020	268	1198	1	179	0	36	41	111	3	18	6	13	21	52	1	6	0	9	32	186	✓	✓	✓	✓	✓	✓	·	✓	✓
	Model H	Apr/2020	263	1096	1	212	0	3	40	39	4	19	8	34	21	118	1	327	0	1	32	71	·	✓	✓	·	·	·	·	✓	✓
	Model I	Apr/2020	263	1096	1	212	0	3	40	39	4	19	8	34	21	118	1	327	0	1	32	71	·	✓	✓	·	·	·	·	✓	✓

*IE: Information Element (= message field)   i-IE: imperative (≈mandatory) IE   n-IE: non-imperative (≈optional) IE

Mismatch Results – $Vendor_1$

- Missing imperative (≈mandatory) & Unknown IEs
  - Directly indicate **functional errors**
- Invalid IEs
  - Numerous incorrect length limit / ad-hoc length checks after decoder function
  - Can lead to **memory-related bugs**
- Missing non-imperative (≈optional) IEs
  - May not be buggy

➡ **9 erroneous cases affecting 33 distinct messages**

			In Binary		Common Mismatch						Syntactic-only Mismatch						Semantic-only Mismatch						Case Study Results								
			# of		Missing		Unknown		Invalid		Missing		Unknown		Invalid		Missing		Unknown		Invalid		Functional†					Memory-related			
	Model	Build Date	Msgs	IEs	i-IE	n-IE	i-IE	n-IE	i-IE	n-IE	i-IE	n-IE	i-IE	n-IE	i-IE	n-IE	i-IE	n-IE	i-IE	n-IE	i-IE	n-IE	E1	E2	E3	E4	E5	E6‡	E7	E8‡	E9
Latest Firmware	Model A	May/2020	268	1204	1	164	0	36	38	109	3	19	6	13	21	52	1	6	0	9	35	203	✓	✓	✓	✓	✓	✓	·	✓	✓
	Model B	May/2020	268	1201	1	167	0	36	38	109	3	19	6	13	21	52	1	6	0	9	35	200	✓	✓	✓	✓	✓	✓	·	✓	✓
	Model C	May/2020	268	1201	1	167	0	36	38	109	3	19	6	13	21	52	1	6	0	9	35	200	✓	✓	✓	✓	✓	✓	·	✓	✓
	Model D	Jun/2020	268	1200	1	179	0	36	41	111	3	18	6	13	21	52	1	6	0	9	32	186	✓	✓	✓	✓	✓	✓	·	✓	✓
	Model E	Jun/2020	268	1200	1	179	0	36	41	111	3	18	6	13	21	52	1	6	0	9	32	186	✓	✓	✓	✓	✓	✓	·	✓	✓
	Model F	Apr/2020	268	1198	1	179	0	36	41	111	3	18	6	13	21	52	1	6	0	9	32	186	✓	✓	✓	✓	✓	✓	·	✓	✓
	Model G	Apr/2020	268	1198	1	179	0	36	41	111	3	18	6	13	21	52	1	6	0	9	32	186	✓	✓	✓	✓	✓	✓	·	✓	✓
	Model H	Apr/2020	263	1096	1	212	0	3	40	39	4	19	8	34	21	118	1	327	0	1	32	71	·	✓	✓	·	·	·	·	✓	✓
	Model I	Apr/2020	263	1096	1	212	0	3	40	39	4	19	8	34	21	118	1	327	0	1	32	71	·	✓	✓	·	·	·	·	✓	✓

*IE: Information Element (= message field) i-IE: imperative (≈mandatory) IE n-IE: non-imperative (≈optional) IE

Mismatch Results – $Vendor_1$

- Let's see E1 (functional) and E6 (memory-related)
  - Appear in new models (Model A to G)

			In Binary		Common Mismatch						Syntactic-only Mismatch						Semantic-only Mismatch						Case Study Results								
			# of		Missing		Unknown		Invalid		Missing		Unknown		Invalid		Missing		Unknown		Invalid		Functional†					Memory-related			
	Model	Build Date	Msgs	IEs	i-IE	n-IE	i-IE	n-IE	i-IE	n-IE	i-IE	n-IE	i-IE	n-IE	i-IE	n-IE	i-IE	n-IE	i-IE	n-IE	i-IE	n-IE	E1	E2	E3	E4	E5	E6‡	E7	E8‡	E9
Latest Firmware	Model A	May/2020	268	1204	1	164	0	36	38	109	3	19	6	13	21	52	1	6	0	9	35	203	✓	✓	✓	✓	✓	✓	·	✓	✓
	Model B	May/2020	268	1201	1	167	0	36	38	109	3	19	6	13	21	52	1	6	0	9	35	200	✓	✓	✓	✓	✓	✓	·	✓	✓
	Model C	May/2020	268	1201	1	167	0	36	38	109	3	19	6	13	21	52	1	6	0	9	35	200	✓	✓	✓	✓	✓	✓	·	✓	✓
	Model D	Jun/2020	268	1200	1	179	0	36	41	111	3	18	6	13	21	52	1	6	0	9	32	186	✓	✓	✓	✓	✓	✓	·	✓	✓
	Model E	Jun/2020	268	1200	1	179	0	36	41	111	3	18	6	13	21	52	1	6	0	9	32	186	✓	✓	✓	✓	✓	✓	·	✓	✓
	Model F	Apr/2020	268	1198	1	179	0	36	41	111	3	18	6	13	21	52	1	6	0	9	32	186	✓	✓	✓	✓	✓	✓	·	✓	✓
	Model G	Apr/2020	268	1198	1	179	0	36	41	111	3	18	6	13	21	52	1	6	0	9	32	186	✓	✓	✓	✓	✓	✓	·	✓	✓
	Model H	Apr/2020	263	1096	1	212	0	3	40	39	4	19	8	34	21	118	1	327	0	1	32	71	·	✓	✓	·	·	·	·	✓	✓
	Model I	Apr/2020	263	1096	1	212	0	3	40	39	4	19	8	34	21	118	1	327	0	1	32	71	·	✓	✓	·	·	·	·	✓	✓

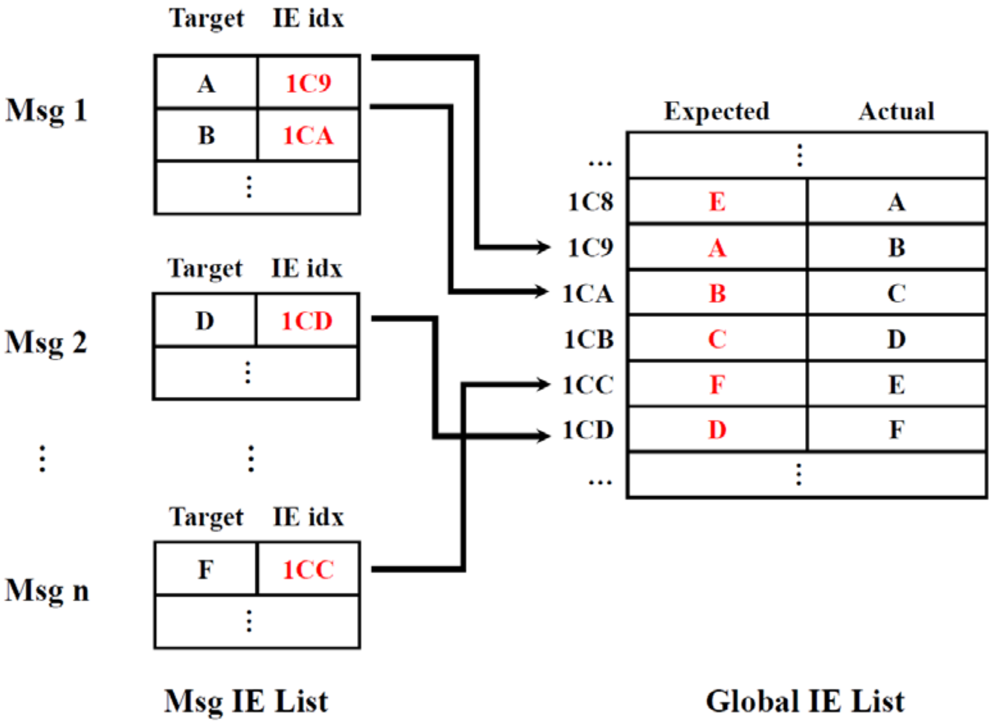
Case Study - E1

- Problem
  - Developers embedded IEs in an incorrect order

- Buggy IEs (six IEs)

Header compression configuration
Control plane only indication
User data container
Release assistance indication
Extended protocol configuration options
Serving PLMN rate control

- Result
  - **22 mismatches**



SysSec
System Security Lab

Case Study - E6

- Vulnerable message & IE
  - P-TMSI REALLOCATION COMMAND
  - Allocated P-TMSI IE
- Reported by invalid mismatch
  - Spec: 5 bytes
  - Firmware: takes upto 255 bytes
  - Not all IEs are checked properly
- Result
  - Stack-based buffer overflow
  - No protection (**exploitable**)

```
void handle_ptmsi_rellocation()
{
    char allocated_ptmsi[5];  ← 5 bytes buffer
    ...
    get_ie_bytes(allocated_ptmsi,
                 ALLOCATED_PTMSI_IDX);
    ...
}

void get_ie_bytes(char *buf, enum IE_IDX idx)
{
    int length;
    char *value;

    // Get a length of the IE in the message (Controllable)
    length = get_ie_length(idx);

    // Check lengths for certain IEs
    if(idx == PLMN_LIST_IDX && length > 45)
        length = 45;
    if(idx == LSA_ID_IDX && length > 3)  ← No length check for Allocated P-TMSI
        length = 3;

    // Get a value of the IE (Controllable)
    value = get_ie_value(idx);
    memcpy(buf, value, length);  ← Copy to buffer
}
```

Discussion & Limitations

- Fully automating bug discovery
  - Requires additional efforts for implication analysis
  - Other techniques (e.g., fuzzing, symbolic analysis) can be combined
- Applicability of BaseSpec
  - Only standard L3 messages are supported currently
  - Similar approach is applicable to other cellular protocols (e.g., ASN.1)
- Other types of bugs
  - Only covers bugs in the decoding logic
  - Cannot cover state-related bugs

Conclusion

- Systematically compared cellular baseband firmware with the specification for standard L3 messages
  - Found 10 error cases including 2 critical RCE vulnerabilities

- Lessons learned
  - Many errors occur in the development process from specifications
  - Comparative analysis can find such errors
  - Various firmware versions and device models can be analyzed (w/o real device)

Questions (1/3)

❖ Yeongbin Hwang

- In the case of an encrypted message, some fields are added to the structure of the existing message. So I think the function to decode this is also a little different, can you find this case properly using a BASESPEC?

➔ Answers

- Actually, we did not have to consider encrypted messages because the decryption process is completely separated from the decoder function.
- One main key assumption of BaseSpec is that developers may follow good programming practices
  - Machine-friendly embedding of message structures rather than hard-coding everything with many if-else clauses
  - Clear separation of different tasks (decryption / parsing)

Questions (2/3)

- Tuan Hoang Dinh
  - Is this method applicable for other wireless protocols and chipsets, for example, GPS, Wifi.
- Wooyoung Go
  - If I apply this method to other cellular protocol, what should I do??

➔ Answers

- The protocol should be extractable from both specification and firmware in a comparable format
  - Apply other techniques (fuzzing / emulation / source-code analysis) first if possible

Questions (3/3)

- Taehwa Lee
  - Have the vendors taken action on the vulnerabilities found?
- Youngjin Jin
  - This paper does not explicitly state any defense measures or countermeasures.

➔ Answers

- The vendor_1 fixed all the bugs, but vendor 2 did not respond
- The bugs are traditional memory corruptions (BOF)
  - Vendor_1 recently adopted stack-protector (canary)
- However, vendor_1 wanted to anonymize the details

Thank You!

hahah@kaist.ac.kr
dkay@kaist.ac.kr