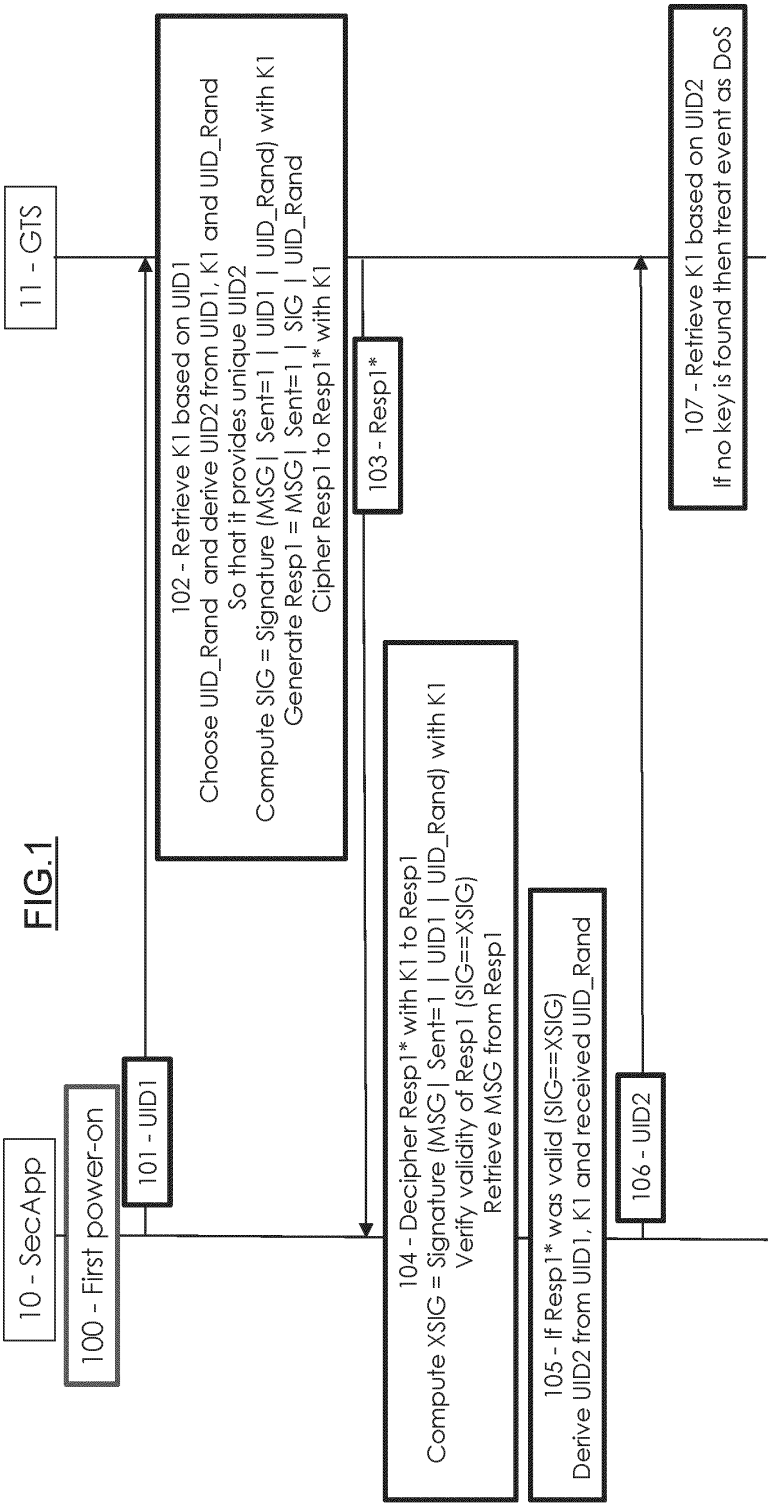
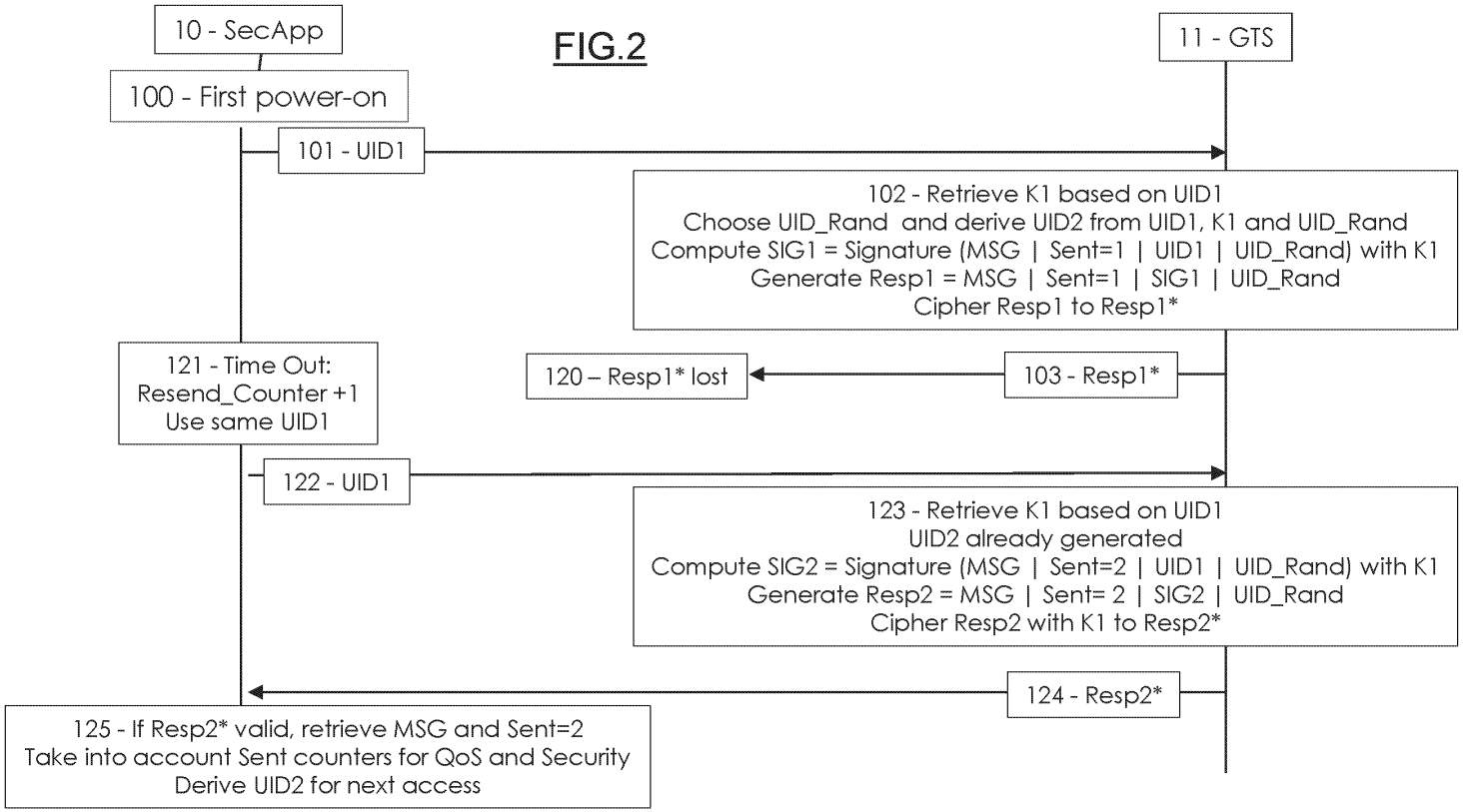

10 - SecApp
11 - GTS
100 - First power-on
101 - UID1
102 - Retrieve K1 based on UID1
Choose UID_Rand and derive UID2 from UID1, K1 and UID_Rand
So that it provides unique UID2
Compute SIG = Signature (MSG| Sent=1 | UID1 | UID_Rand) with K1
Generate Resp1 = MSG| Sent=1 | SIG | UID_Rand
Cipher Resp1 to Resp1* with K1
103 - Resp1*
104 - Decipher Resp1* with K1 to Resp1
Compute XSIG = Signature (MSG| Sent=1 | UID1 | UID_Rand) with K1
Verify validity of Resp1 (SIG==XSIG)
Retrieve MSG from Resp1
105 - If Resp1* was valid (SIG==XSIG)
Derive UID2 from UID1, K1 and received UID_Rand
106 - UID2
107 - Retrieve K1 based on UID2
If no key is found then treat event as DoS

FIG.1

FIG.2


10 - SecApp
11 - GTS
100 - First power-on
101 - UID1
102 - Retrieve K1 based on UID1
Choose UID_Rand and derive UID2 from UID1, K1 and UID_Rand
Compute SIG1 = Signature (MSG | Sent=1 | UID1 | UID_Rand) with K1
Generate Resp1 = MSG | Sent=1 | SIG1 | UID_Rand
Cipher Resp1 to Resp1*
121 - Time Out:
Resend_Counter +1
Use same UID1
120 – Resp1* lost
103 - Resp1*
122 - UID1
123 - Retrieve K1 based on UID1
UID2 already generated
Compute SIG2 = Signature (MSG | Sent=2 | UID1 | UID_Rand) with K1
Generate Resp2 = MSG | Sent= 2 | SIG2 | UID_Rand
Cipher Resp2 with K1 to Resp2*
124 - Resp2*
125 - If Resp2* valid, retrieve MSG and Sent=2
Take into account Sent counters for QoS and Security
Derive UID2 for next access

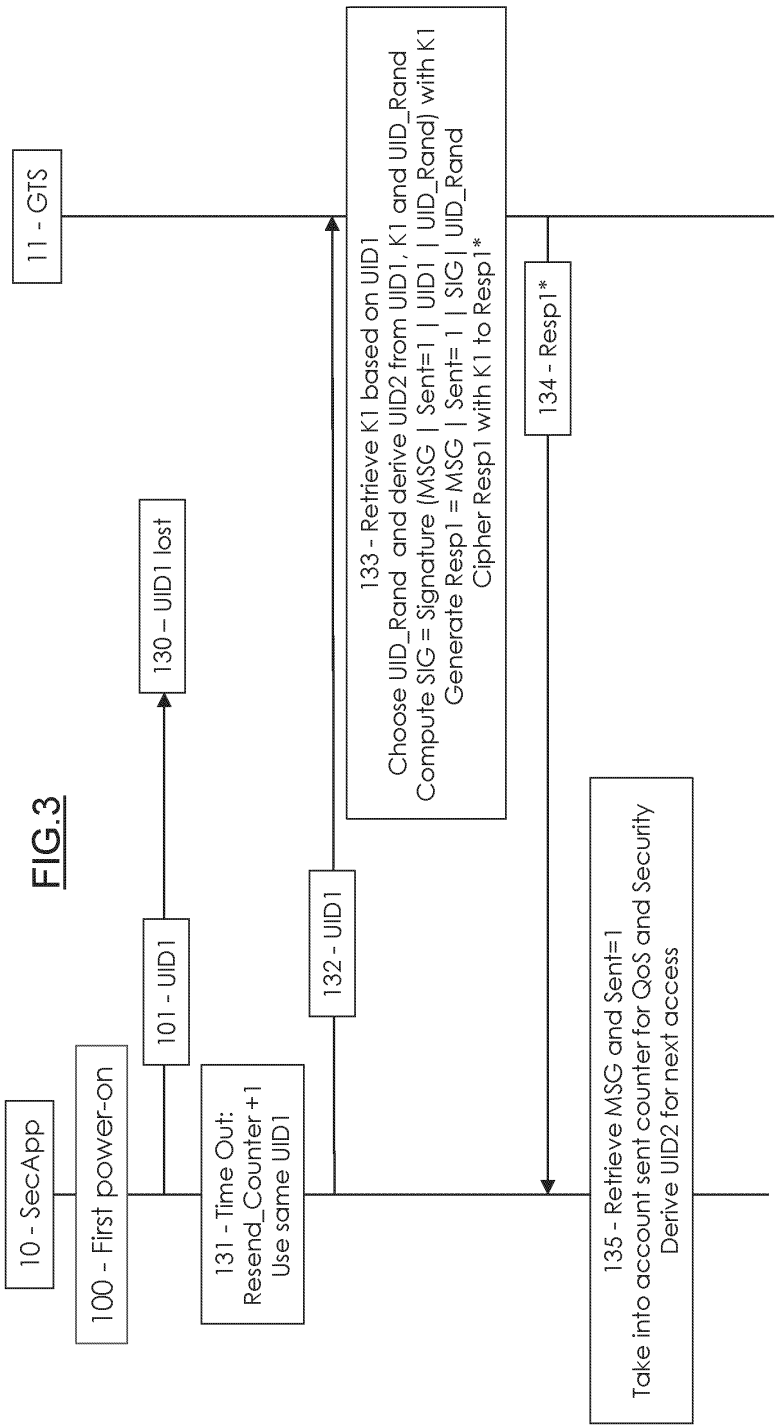

10 - SecApp
100 - First power-on
101 - UID1
130 – UID1 lost
131 - Time Out:
Resend_Counter +1
Use same UID1
132 - UID1
11 - GTS
133 - Retrieve K1 based on UID1
Choose UID_Rand and derive UID2 from UID1, K1 and UID_Rand
Compute SIG = Signature (MSG | Sent=1 | UID1 | UID_Rand) with K1
Generate Resp1 = MSG | Sent= 1 | SIG| UID_Rand
Cipher Resp1 with K1 to Resp1*
134 - Resp1*
135 - Retrieve MSG and Sent=1
Take into account sent counter for QoS and Security
Derive UID2 for next access

FIG.3