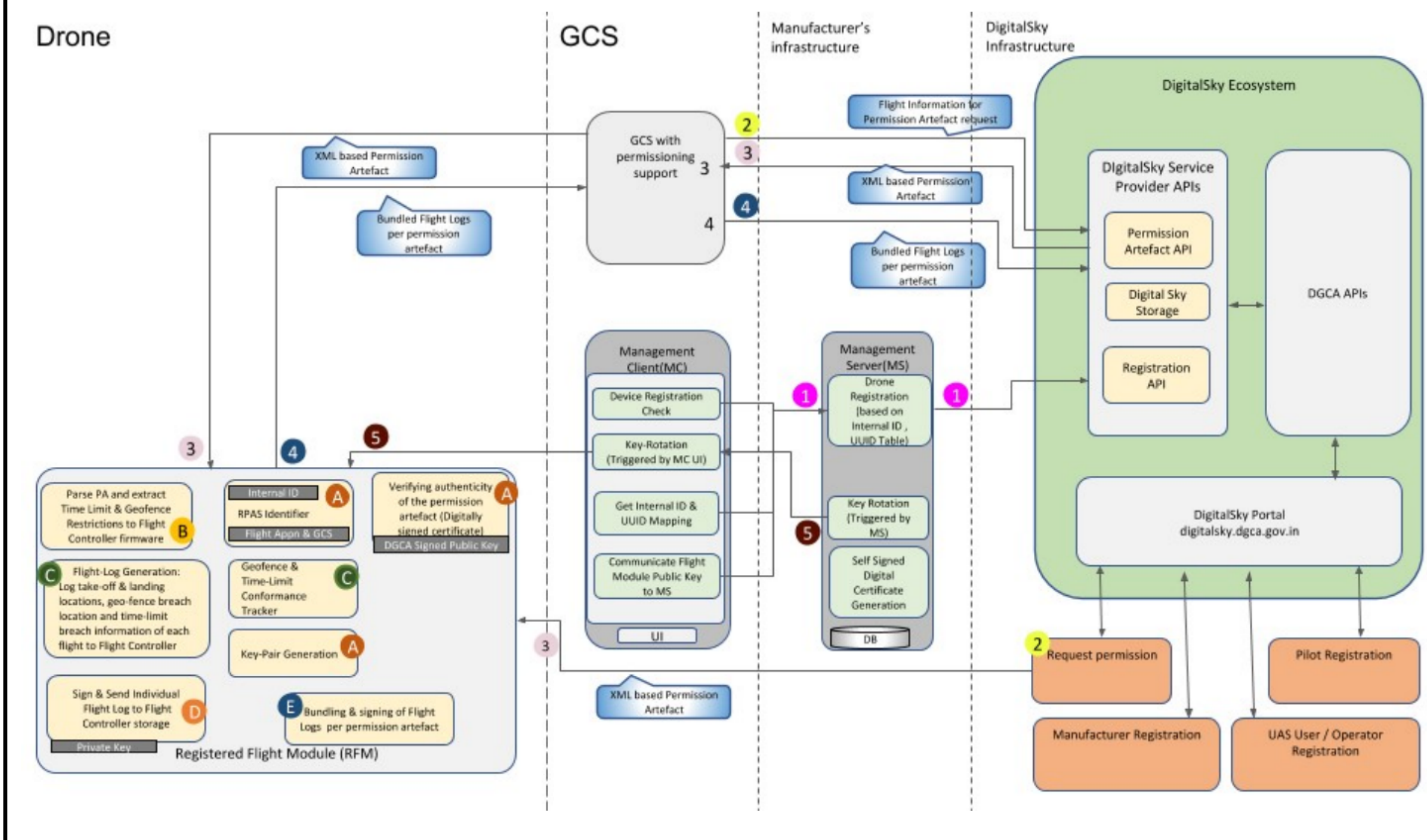
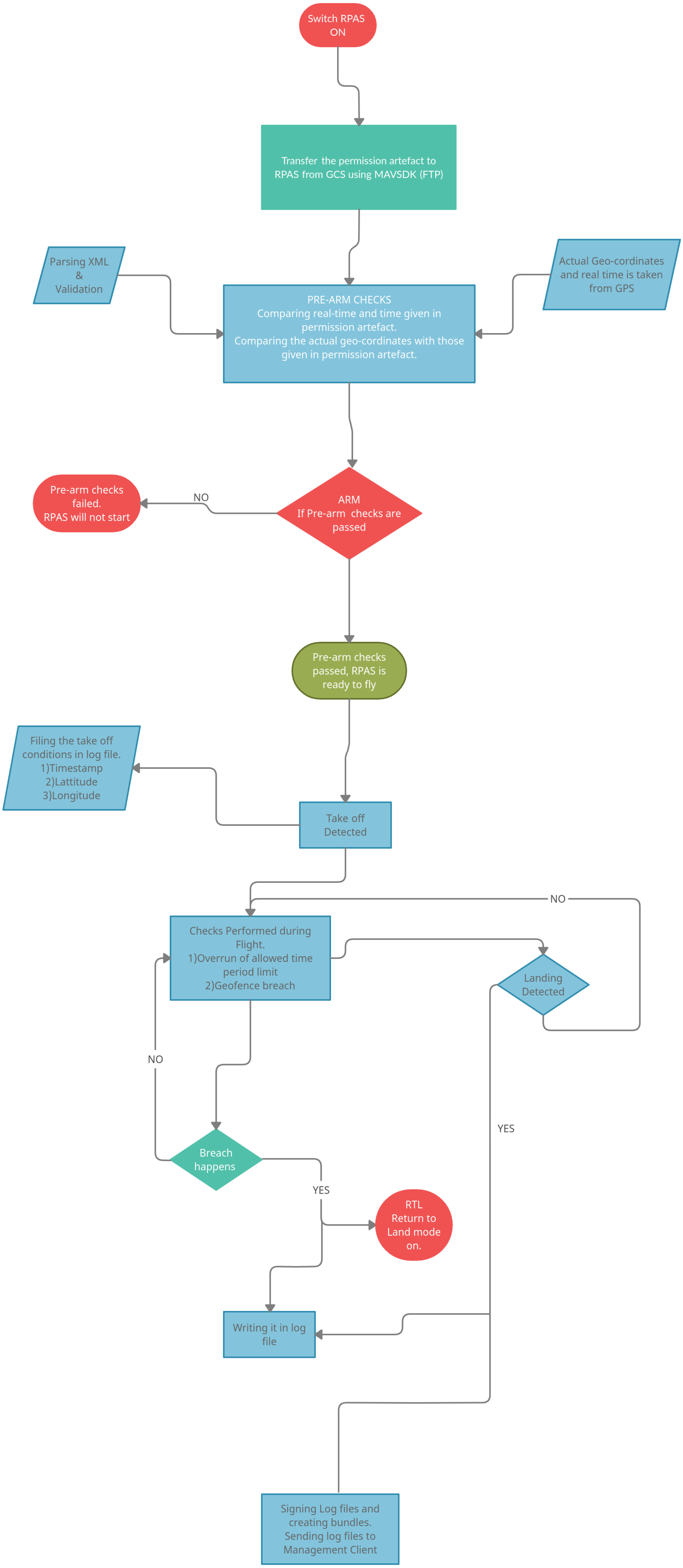
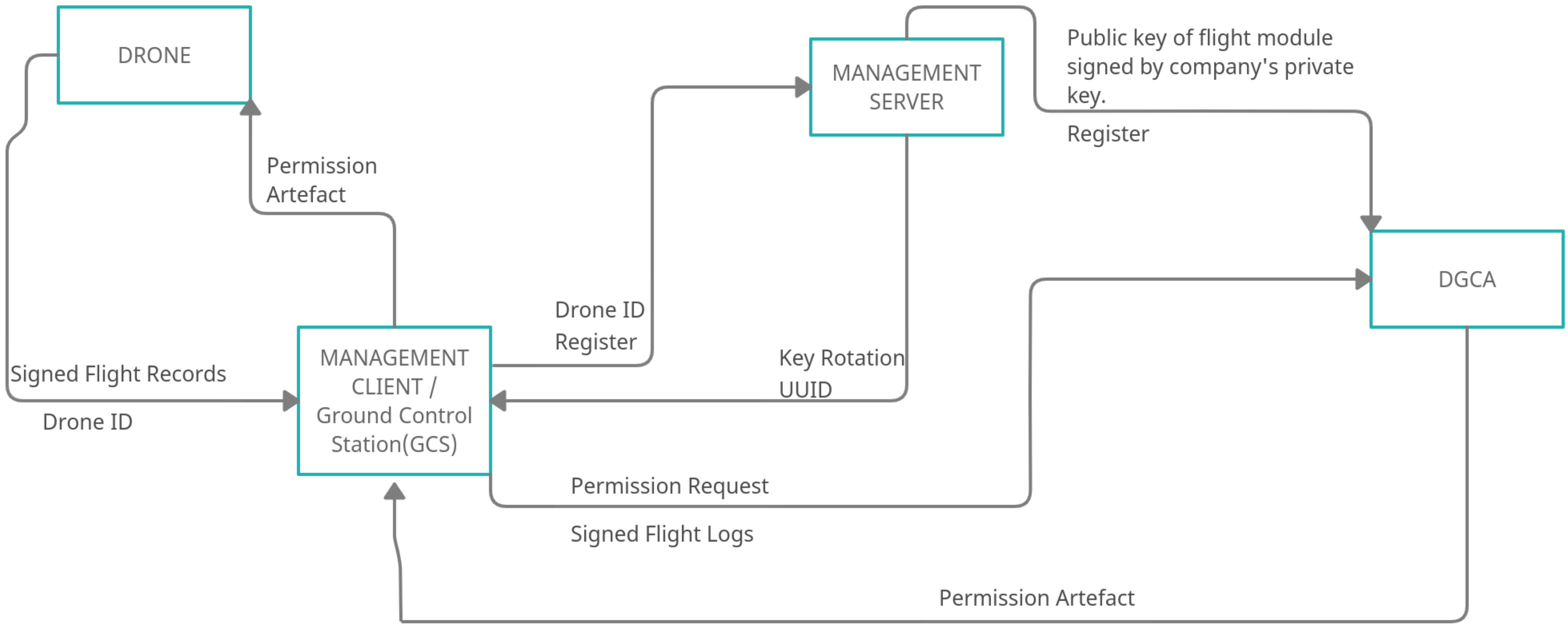
Exhibit C: High Level Sequence Diagram of RPAS, Ground Control Station, Management Client and Server, and Digital Sky




Switch RPAS ON
Transfer the permission artefact to RPAS from GCS using MAVSDK (FTP)
Parsing XML & Validation
PRE-ARM CHECKS
Comparing real-time and time given in permission artefact.
Comparing the actual geo-cordinates with those given in permission artefact.
Actual Geo-cordinates and real time is taken from GPS
ARM
If Pre-arm checks are passed
NO
Pre-arm checks failed.
RPAS will not start
Pre-arm checks passed, RPAS is ready to fly
Filing the take off conditions in log file.
1)Timestamp
2)Lattitude
3)Longitude
Take off Detected
NO
Checks Performed during Flight.
1)Overrun of allowed time period limit
2)Geofence breach
Landing Detected
NO
YES
Breach happens
YES
RTL
Return to Land mode on.
Writing it in log file
Signing Log files and creating bundles.
Sending log files to Management Client


DRONE
MANAGEMENT SERVER
Public key of flight module signed by company's private key.
Register
Permission Artefact
DGCA
Drone ID
Register
Key Rotation
UUID
Signed Flight Records
MANAGEMENT CLIENT / Ground Control Station(GCS)
Drone ID
Permission Request
Signed Flight Logs
Permission Artefact

NPNT SOLUTION

-PATANJALI MAITHANI

Scope of this presentation

This presentation specifies a robust solution for a RFM to attain NPNT compliance .

Main objectives of NPNT

For attaining NPNT compliance , a flight module provider is supposed to develop following infrastructure.

1) Inside RFM
2) A Flight Module Management Client
3) A Flight Module Management Server

Goals of Inside RFM Infrastructure

1)**Key pair generation**: We aim to achieve Level 1 compliance (i.e In TEE), in which key pair generation takes place inside the hardware RFM module itself.(TEE-Trusted Execution Environment, more specification yet to come on this in later release of the RPAS guidance manual)

SOLUTION: A program for randomly generating two 1024-bit long prime numbers and performing multi-precision modular arithmetic is flashed upon the RFM module and called upon whenever a key rotation is required.

2) **Parsing permission artefact and validation**: Permission artefact(PA) is an XML document which contains the information of permitted geospace and time period for the flight to be taken by the RPAS. This document is digitally signed by the DGCA to ensure that the information attached to it is non-tampered and has its original source from the DGCA.

SOLUTION: The transfer of PA from management client(MC) to RPAS is done using MAVSDK FTP class(File Transfer Protocol). To make sure that the PA has come from the particular MC, Public Key Cryptography is used, where minimum key length is 128 bit(as specified in the RPAS guidance manual). Once the file is sent to the RPAS, information from various nodes can be fetched and processed as per the standards given in the RPAS guidance manual. XML canonicalization, SHA256 and RSA decryption are the programs that are written inside the RFM module firmware to make the validation of the XML document possible. The DGCA Public key is securely installed on to the RFM, once validated on the management server(MS).

3) **Monitoring Geofence for breach**: If during flight, RPAS overrun the limit of allowed time period or passes the permitted geospace, Return to home(RTH) action should get implemented. Along with the RTH, the timestamp and geo coordinates are supposed to be logged at a minimum rate of 1Hz in the log file.

SOLUTION: A program is written to read the constantly updated position output from the EKF and check if it lies inside the polygon specified by the latitude and longitude coordinates in the permission artefact. If the position output from the EKF turns out to be outside the permitted geospace then the RTH action is enabled and side by side log filing is also initiated.

4)**Signing logs using private key:** The flight logs captured during the period of the permission artefact should be stored on the flight controller along with hash of the logs to ensure tamper- proof records. A record must be maintained on-board to connect next and previous flight logs to avoid omission and the same should be inaccessible to the user. Once the permission artefact is expired or when user wants to submit logs, the complete bundle of such logs should be signed using RFM private key and submitted to the DSP.

SOLUTION: A program is written inside the RFM module to generate the json format log file according to the schema defined in the RPAS guide manual. For signing the flight logs, RSA encryption(private key of flight module is used)and SHA 256 algorithms are used.

Summary for Inside RFM Infrastructure

Following key features are to be embedded in RFM module.

1)Key Pair Generation

2)Parsing permission artefact and validation

3)Monitoring Geofence(time and position)

4)Generation and signing of Flight logs.

Omnipresent Robot Tech
NPNT Document

May 29, 2021

Omnipresent Robot Tech

Contents

1 Introduction

1.1 Structure of this document

This document provides the architecture of the NPNT implementation for the RPAS manufactured in Omnipresent Robot Tech. The document discusses different scenarios with well-established protocols that are needed to maintain aspects like security, traceability, and integrity of the RPAS in the Drone ecosystem of the country. The basis of this document lies in the chapter seven of the RPAS guidance manual available at the digital sky website. Chapters in this document are scenario-specific. The protocols followed in each scenario are elucidated with their basis upon the chain of trust implemented using PKI infrastructure.
The keys used in the architecture are as follows:
1)Flight Module Provider(FMP) Key-pair
2)Registered Flight Module (RFM) Key-pair
3)Key pair used in Management Client(MC)
4)Key used inside RFM for encryption and decryption(this will be reffered as Key D)

The algorithms used for hashing and signing files for information exchange are:
1)SHA256 for creating a 256-bit long hash of the information
2)RSA (2048)encryption for signing the hash with 2048-bit long modulus.

Different scenarios have used files as a means for information exchange. Manipulating its read, write, and execution access makes it secure for usage. Also, they become suitable for information change when signed, maintaining the non-repudiation, traceability, and integrity of its content. Explanation of each scenario is accompanied by the files playing a crucial role in implementation.

The solution to NPNT present in this document is Level 0 compliant. The RFM key pair is kept encrypted inside the RFM module and can only be decrypted inside the RFM by the key stored in the firmware code of RFM. Key generation is done inside the RFM module itself. The details of the same are in chapter 5.

There is no companion computer used in this implementation therefore there is no need to worry about the concerns related to inter-modular security.

The functionality of hardware in the loop is disabled by changing the appropriate parameters inside the firmware code. Even if the user tries to operate the RFM in HITL mode using Qgroundcontrol, he/she won't succeed in it as the parameter is locked from inside of the drone. Some flight parameters can be modified by the user on sending a request to the Management Server. The authentic way in which it is carried out is mentioned in chapter 6.

If any hardware accessories get damage while operating RPAS or the user wants to change particular hardware in the RPAS, an authentic way of carrying out the hardware change process is mentioned in chapter 7.

As the first scenario that will be faced by the RPAS in its lifetime would be its registration, the second chapter(next chapter) of this document discusses the same in detail. On each power-on cycle, RFM makes some pre-arm checks before it gets ready to take off. These pre-arm checks and the sequence in which they are performed are mentioned in chapter 3. This chapter is referred in all other chapters to get a more clear idea of the region in which the processing is carried out.

Chapter 4 and Chapter 8 discuss the NPNT policy and the logic for managing flight logs. At last, Chapter 8 details the procedure of firmware update.

2 Registration Scenario

Please first read the chart on the next page for better intuition of the upcoming text. The following text is like a supplement to the chart. Chart itself is enough to know the "Know how" process.

During the manufacturing phase, the drone ID of the RPAS is calculated using the device IDs of its hardware accessories (inside the company's lab) and is coded inside the firmware. This Drone ID will be unique to each drone and will remain unchanged throughout the lifetime of the RPAS. A file named DroneID.txt(inside RFM), signed using RFM private key, contains this DroneID.

Along with the DroneID, the DroneID.txt file has RFM public key, HardwareID, Firmware version, RPAS category, UUID number, and some other relevant information. This file is fetched by Management Client(MC) and is later sent to the Management Server. By this, identification of the particular RPAS is done at the Management server side.

Before the registration, the RFM is coded with the private-public key pair during the manufacturing phase. The public key of this key pair is available with the Flight module provider and is used by it to validate the DroneID.txt file when sent for the first time.

In Figure 2 of this document, one can see the logic running inside the RFM for checking if the Drone is registered or not. If the drone is not registered, then it will not get armed and a warning message will flash up on the QGC saying "Connect to MC for registration".

UUID is generated at the time of registration at the MS side. UUID.txt file contains the DroneID and UUID of a particular RPAS. This file is signed with the Flight module provider's(FMP) private key and is later verified inside the RFM with the FMP's public key(which is coded inside the firmware during the manufacturing phase). Thus how the integrity of the file is maintained. Once the registration is complete, the UUID.txt file will remain in the MC. So that even if there is no internet, which is the case while operating RPAS in remote areas, the drone can confirm its registration by validating the UUID.txt file. The same is represented in detail in the flow chart in Figure2.

A unique product key is given to the buyer, along with the RPAS. When the Registration process is triggered, from the management server and registration UI is opened at the MC end, the buyer type in this product key with other credentials. This way, out of the band authentication of the product is done. Once the product key and signature of DroneID.txt is verified using the public key of the RFM(corresponding to the DroneID in DroneID.txt), UUID is generated inside the Management Server(MS).

In later steps, as shown in Figure 1, MS sends the drone's DroneID, generated UUID, and signed public key(signed with FMP private key) to the Digital sky for registration. Once the registration is done, UUID.txt and authenticated RFM public key are sent back to MC, and consequently to RFM. KeyChangePerm.txt file contains the authenticated RFM public key signed with FMP private key. More details of KeyChangePerm.txt and its usage will be in chapter 5 (related to key rotation).

Files in Play:
1)UUID.txt 2)DroneID.txt 3)KeyChangePerm.txt

Keys in Play:
1)FMP Key pair 2)RFM key pair 3)Product key

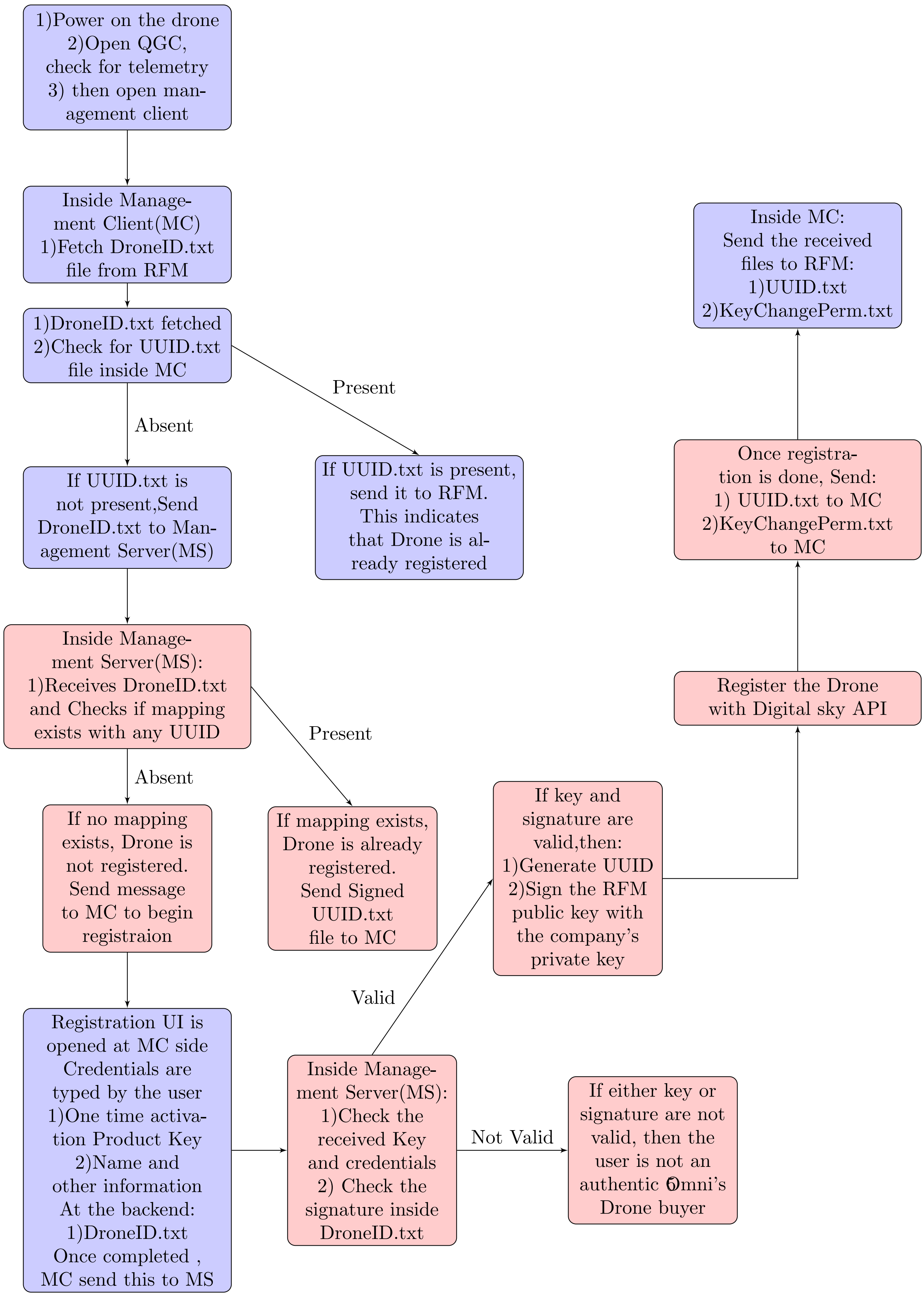


Figure 1: Registration Scenario

3 Inside RFM processes

The chart in Figure 2 shows the Pre-arm checks before the drone gets ready to fly.

The Checks are in the following order:

1)RPAS identifier: This is to make sure if the hardware attached to the RPAS is authentic or not. A function, at every power-on cycle, calculates a Hardware number using the device IDs of the attached electronic hardware. The logic of doing so is kept private to the company's knowledge. A valid Hardware number is present in the HardwareInUse.txt file. This file is encrypted inside the RFM using a key (key D). If the value of the Hardware number calculated at each power-on cycle matches the valid one, then the RPAS is authentic. More information on these files is in chapter 7(Secure hardware change).

2)Registration: Check for the UUID.txt file is done at each power-on cycle to ensure that the drone is registered or not. (chart in Figure 1 discusses more of this)

3)Check of recentPA.txt: The function of this file is to manage flight logs. Some scenarios where this file comes into use are:

3.1) When a crash takes place, the log file remains unfinished and unsigned: this text file helps in completing the log file, sign it and store it.

3.2)When the validity of the recently used permission artifact(PA) has ended: this file helps in bundling and signing the group of log files corresponding to the recently used PA. (chart in Figure 7 discusses more on this)

4)Check for Key rotation: Key rotation is triggered by the management server(MS). The main file to be checked is KeyRotation.txt which if validated with FMP public key, initiates the process of key generation. (chart in Figure 4 discusses more on this)

5)Check for Change in Parameter: This check is done using the ChangePerm.txt file (signed using FMP private key). This is done to check if there are some parameters whose values are needed to be changed or not. (chart in Figure 5 discusses more on this)

6)Check for Permission Artefact(PA): Check for PA is done to know if the time and place conditions as specified by the Digital sky are met or not. PA is an XML file(.xml extension). (chart in Figure 3 discusses more on this)

7)At last, sensor readiness checks are performed to ensure a safe flight.

Files in play:

(1)HardwareChange.txt 2)UUID.txt 3)PA.xml 4)KeyRotation.txt 5)ParamChangePerm.txt 6)ParamInUse.txt 7)KeyChangePerm.txt 8)DroneID.txt

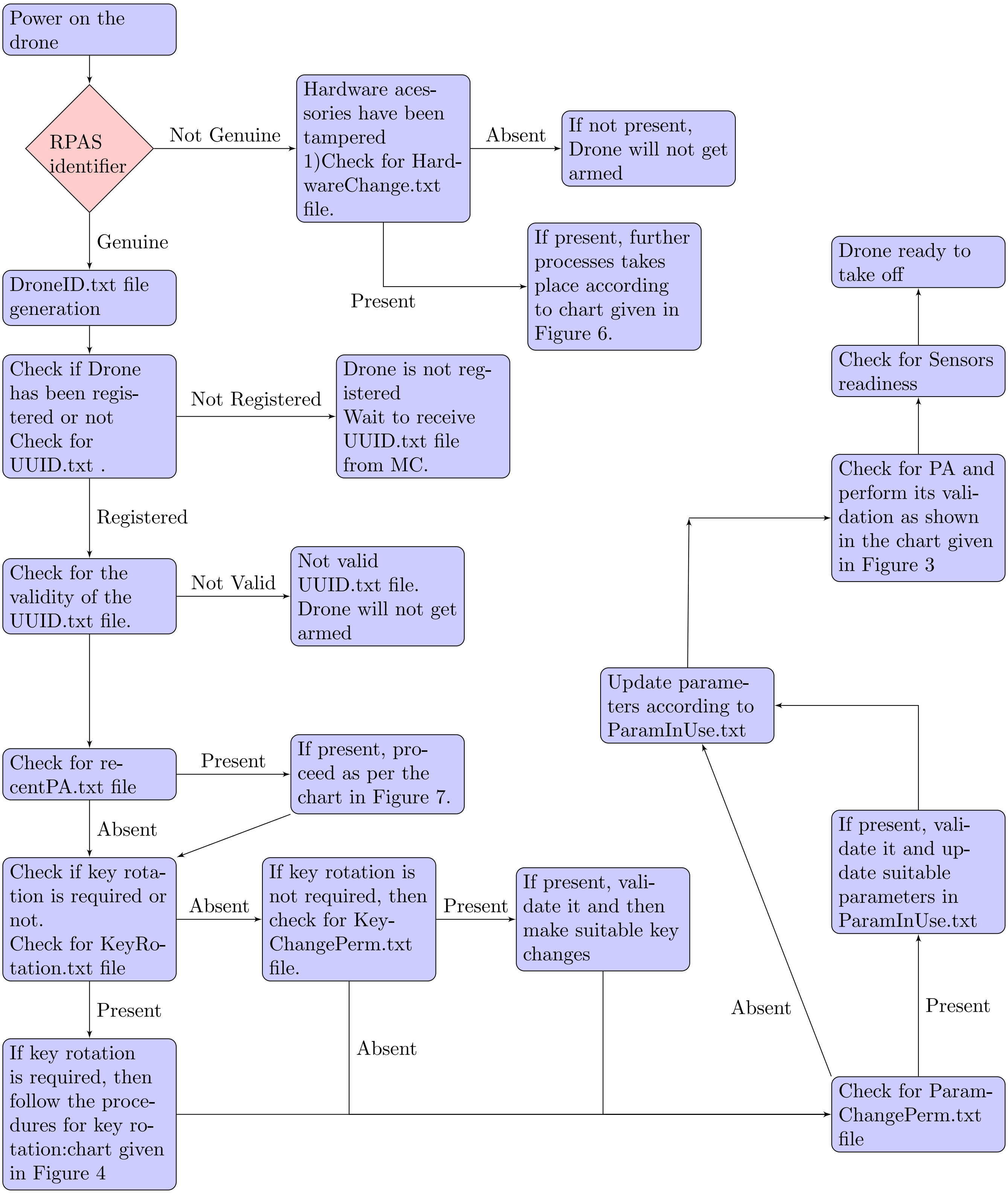


Figure 2: Inside RFM flow chart

4 NPNT Policy

The chart in Figure 3 represents the flow of the implementation of the NPNT policy. The chart covers up all the majour details in the process. Some of the important points are as follows:

User makes requests to Digital sky for permission artifact through the Management Client(MC) UI.

Inside RFM, PA is verified with the Digital Sky Public Key. Digital Sky Public Key is coded inside the firmware and can't be accessed. If the Digital Sky Public key is changed, a firmware update for the RPAS is released with the new Digital Sky Public Key coded inside.

The process of the Firmware update is given in chapter 9. RPAS is programmed such that if a valid PA is not found inside the RPAS, it will not get armed. The reason for which a PA is invalid is also shown on the QGC. Permission to perform further sensor checks is given only when there is no breach of time and position.

During the flight, continuous checking for knowing the breaches(time or position) is done at a rate of 1 Hz. Upon encountering a breach, the RPAS logs that event along with its timestamp to the log file. The log file is signed at the end of the flight using the RFM private key. The more detailed log management is discussed in chapter 8.

RPAS turns on its RTL(return to launch) mode upon sensing a breach.

For validating the PA.xml file inside the RFM, xml file was first converted into its canonical form. Later, the hash of the canonical form was calculated using SHA256 and compared to the one decrypted using the RSA public key.

Files in play:

1)recentPA.txt 2)PA.xml

Keys in play:

1)Digital Sky public key: for verifying the PA.

2)RFM private key: for signing the log file at the end of the flight and while bundling group of such log files when the validity of the PA gets over.

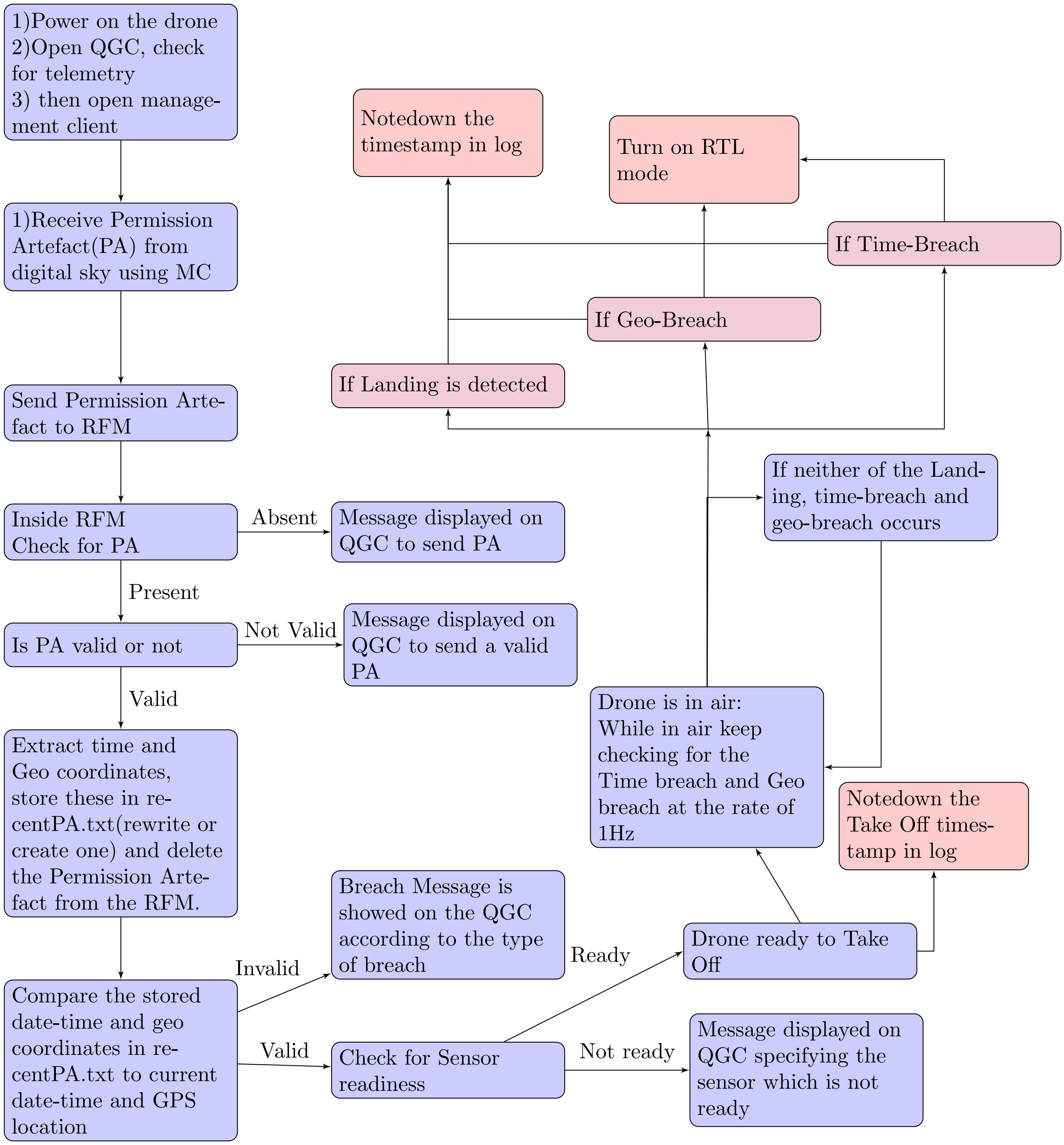


Figure 3: Flight Flow chart

5 Key Rotation

Key rotation can only be triggered by Management Server. A KeyRotation.txt file is signed with Flight Module Provider's Private key and sent to Management Client(MC). From Management Client, this file is sent to RFM. All other activities like fetching of logs and permission artifacts are stopped until the newly formed Public Key is received at the MC end.

Inside RFM, a check for the presence and validity of the KeyRotation.txt file is performed as shown in the chart in Figure 2.

Accordingly, the process of key generation is initiated. The files playing crucial role in iformation exchange in this process are as follows:

1)KeyRotation.txt: Signed using FMP's private key for telling RFM to initiate the process of key generation.

2)PublicKeyNew.txt: Signed using RFM's private key(the one present in PublicPrivateInuse.txt), this is the file fetched by Management Client(MC) and later sent to Management Server(MS).

3)PublicPrivateNew.txt: This is the file inside the RFM, where newly generated keys are kept in a buffer state. This file is encrypted using a secure key(Key D) which is stored in the RFM code.

4)KeyChangePerm.txt: This is the file that initiates the process of copying the contents of PublicPrivateNew.txt to PublicPrivateInUse.txt. This file is signed using FMP's private key.

5)PublicPrivateInuse.txt: This is the file that contains the RFM public-private keys in an encrypted format. It can only be decrypted using a secured Key(Key D). For signing logs, the private key in this file is first decoded and then used.

Keys in Play:

1)FMP private-public key

2)RFM private-public key(old and new)

3)Key D used inside RFM for encryption and decryption.

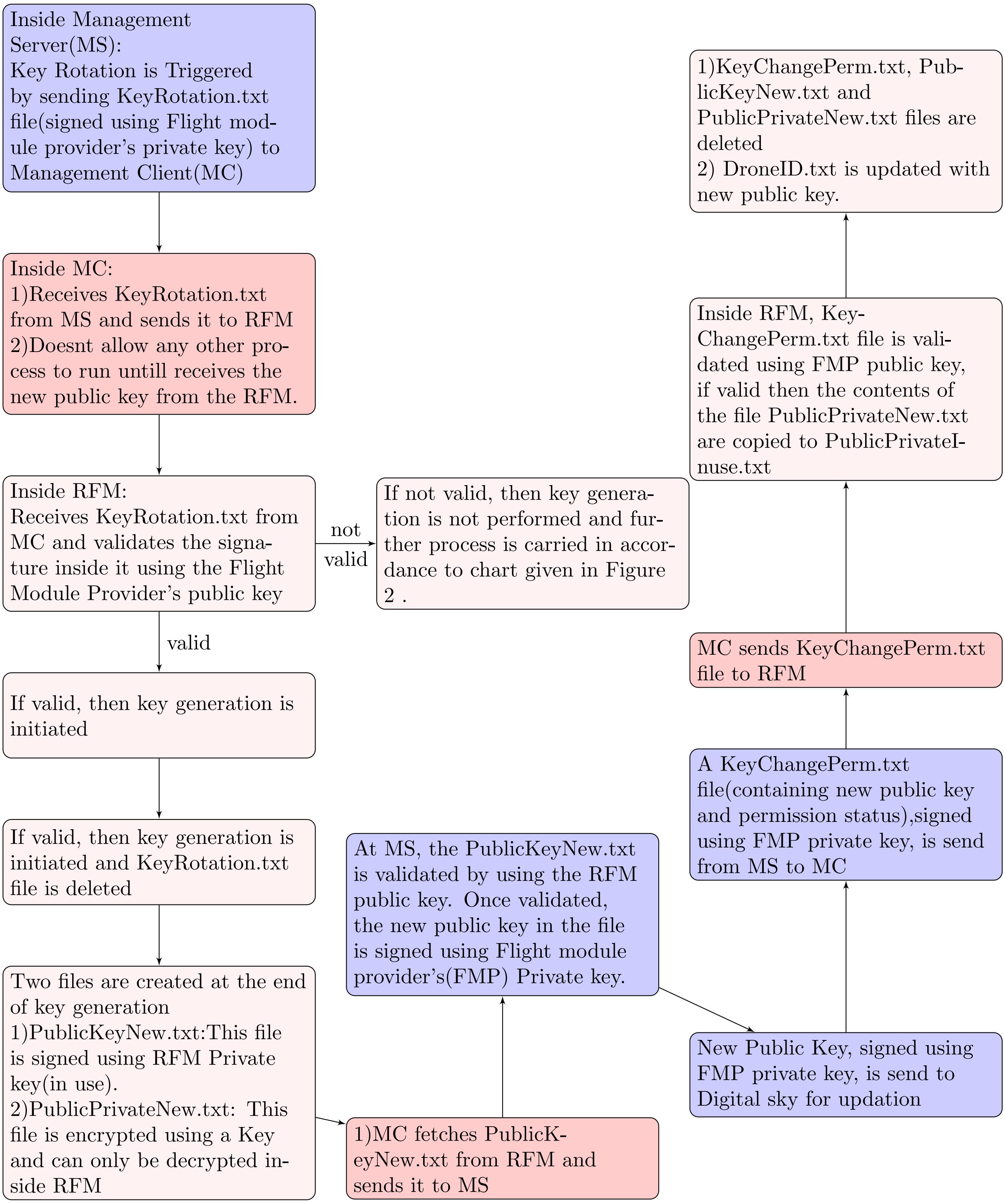


Figure 4: Key Rotation and Key Management

6 Secure Flight Parameters Change

There are some crucial parameters, changing access for which is not provided to the user. These parameters are locked inside the firmware. Some examples of such parameters are:

1)BAT_CRIT_THR

2)BAT_EMERGENCY_THR

3)BAT_LOW_THR

4)COM_LOW_BAT_ACT

5)COM_HOME_IN_HR

6)GF_ACTION

7)LNDFW_AIRSDD_MAX

There might be some situations where the user wants the parameters to be changed according to his need. An authentic and secure channel for doing the same is provided in the chart in Figure 5.

The setting up of the flight parameters takes place before the arming of the drone, as shown in the chart in Figure 2. The information exchange in the chart in Figure 5 is carried out using the following files:

1)ParamChangePerm.txt: File signed using Flight Module Provider's private key. It contains the list of the parameters and their values that are needed to be modified in the RFM code.

2)ParamInuse.txt: This is the file kept inside the RFM from which RFM reads and set value at the start of each power-on cycle.

Keys in Play:

1)FMP private-public key: Used for signing the ParamChangePerm.txt file. The same is validated inside the RFM using the public key of FMP.

The information change between MC and MS is using https.

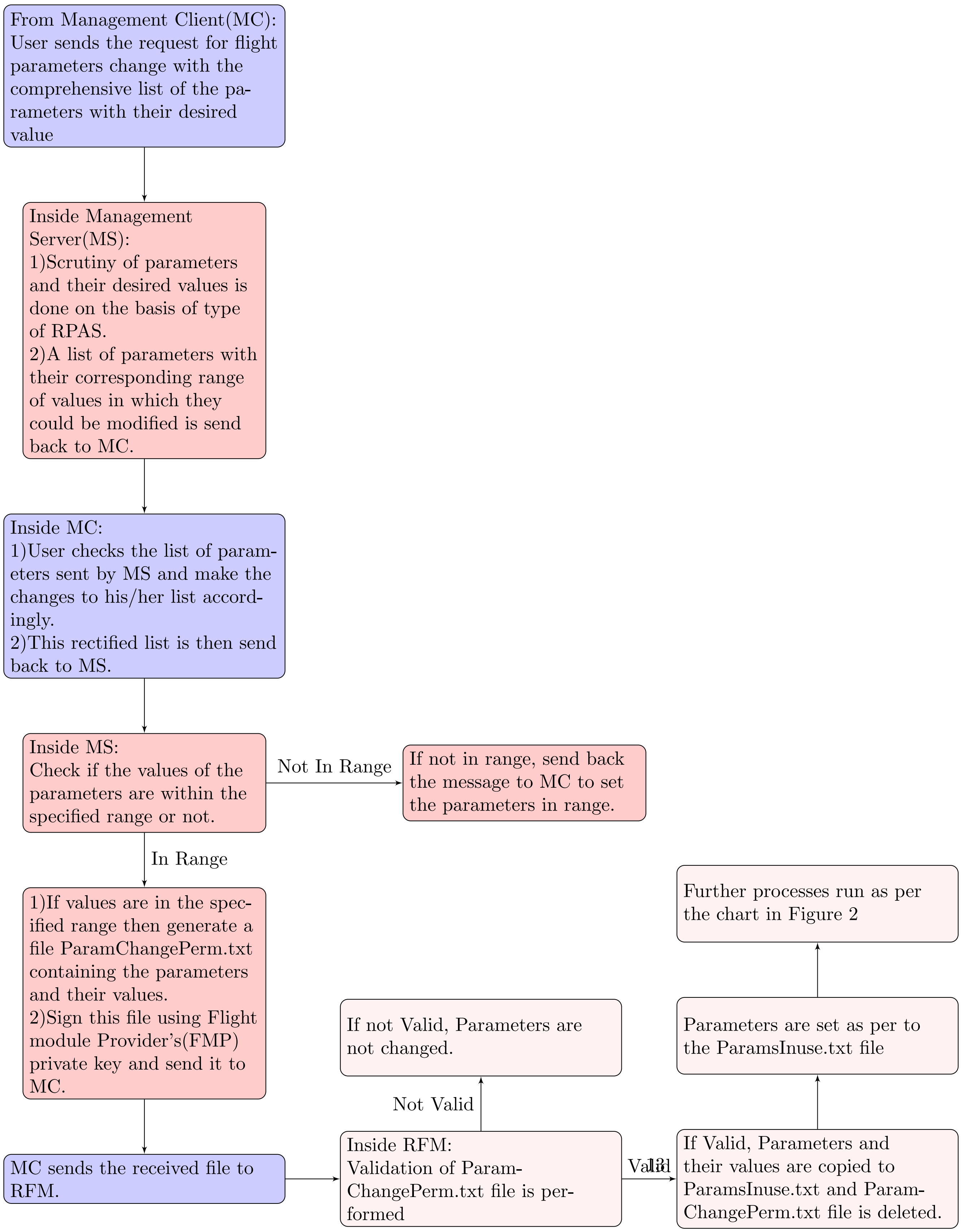


Figure 5: Authentic Parameter Change

7 Secure Hardware Change

All the electronic hardware components are verified at each power-on cycle. This is done with the help of each device's unique Device ID and the true Hardware number. This is how Hardware tamper-avoidance is implemented in the RPAS. Hardware number and DroneID are two different parameters. Hardware number is changed as per the replacement of the existing hardware, but DroneID is constant throughout the lifetime of the RPAS.

The chart given in Figure 6 shows the detailed process for the replacement of the damaged hardware. In this way, it is easy to keep track of the changes brought in hardware in the lifetime of the RPAS.

Files in Play:

1)HardwareChange.txt: This is the file sent by the Management Server through HTTPS to Management Client. It is signed by the company's private key. It consists of the Device ID of the new hardware.

2) HardwareInUse.txt: This file is inside RFM and contains device IDs of all the hardware components attached to the RPAS and the True hardware number of the RPAS. This file is used for comparing the Hardware number generated at each power-on cycle. It is encrypted inside the RFM and can only be read/updated inside the drone.

3)SystemLog.txt: This file contains the log of the periodic changes occurring in the RFM, like hardware change, key change, parameters change. When the user tries to attach an unauthentic hardware component to the RPAS, then also the event is registered in this file.

Keys in play:

1)FMP private-public key

2)RFM encrypting/decrypting key(Key D)

3)RFM private-public key

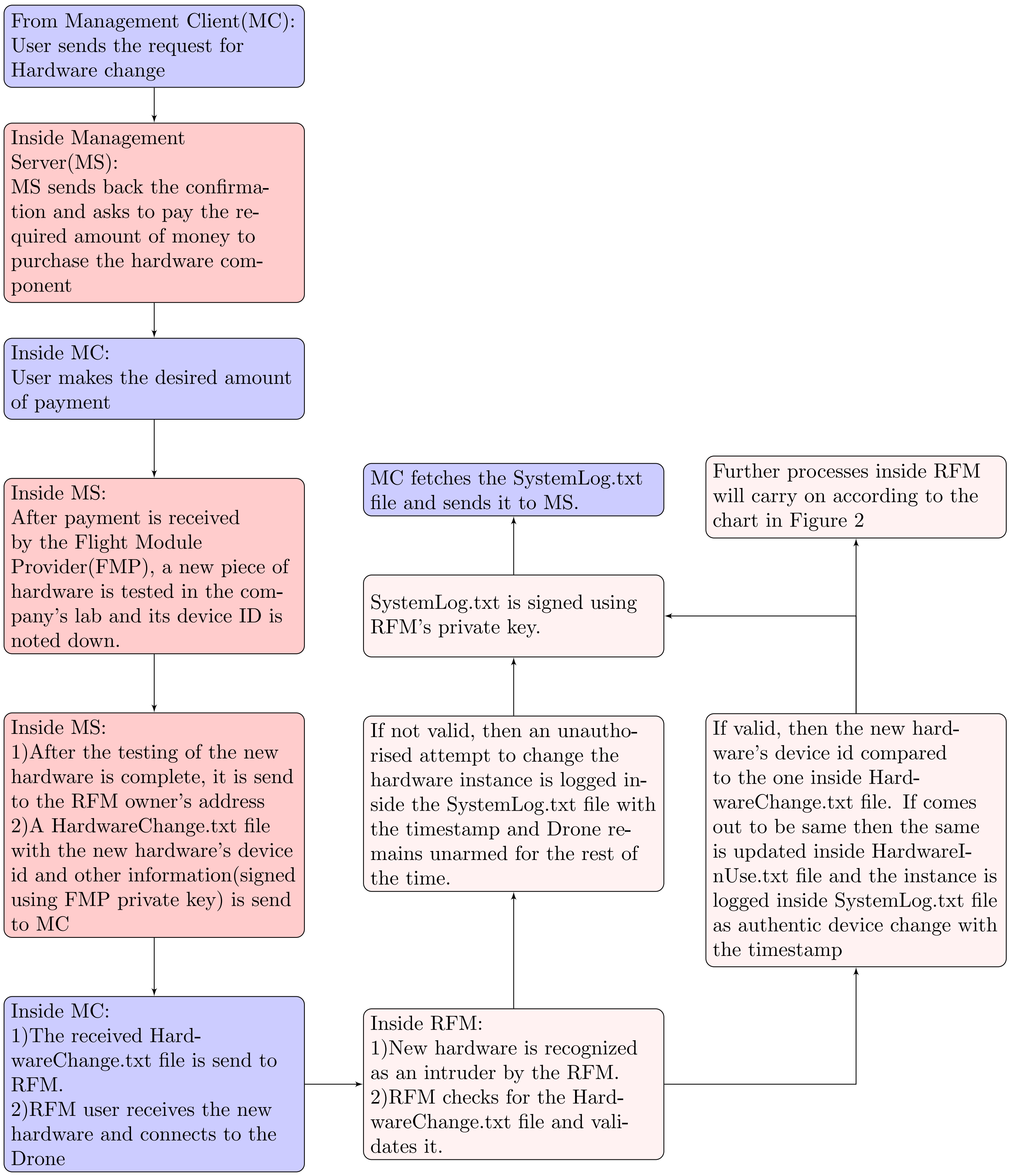


Figure 6: Authentic Hardware Change

8 Flight Log Management

The chart in Figure 7 discusses how log files are handled during and after the flight.

Log files can be modified only inside the RFM. If an accident happens and the log remains unfinished and unsigned, logic is implemented in the RFM to finish and sign such type of log files in the next power-on cycle. After the valid period of PA ends, bundling of flight logs will automatically start. And the drone will remain unarmed until the MC fetches all the log files with their combined signature and sent back a confirmation file to the RFM, confirming the successful download of all the flight logs. Later all the flight logs are automatically sent to the Digital Sky from the MC when the internet connection is established.

Files in play are:

1)recentPA.txt: This is the file where the information of the recent permission artifact and flight log is stored. Its usage is very much evident in the charts given in Figure 3 and Figure 7.

2)bundleSig.txt: This is the file used to store the calculated combined hash of all the log files. This hash is signed using the RFM private key. Thus, maintaining the integrity of the flight logs.

3)fetched.txt: This file is signed using the private key of the Management client and sent to the RFM to indicate that the log files have been successfully downloaded. After this RFM can erase the flight logs from its memory.

Keys in play:

1)RFM private-public key

2)Management Client Private-Public key: Public key of MC is stored inside the RFM and is used to validate the fetched.txt.

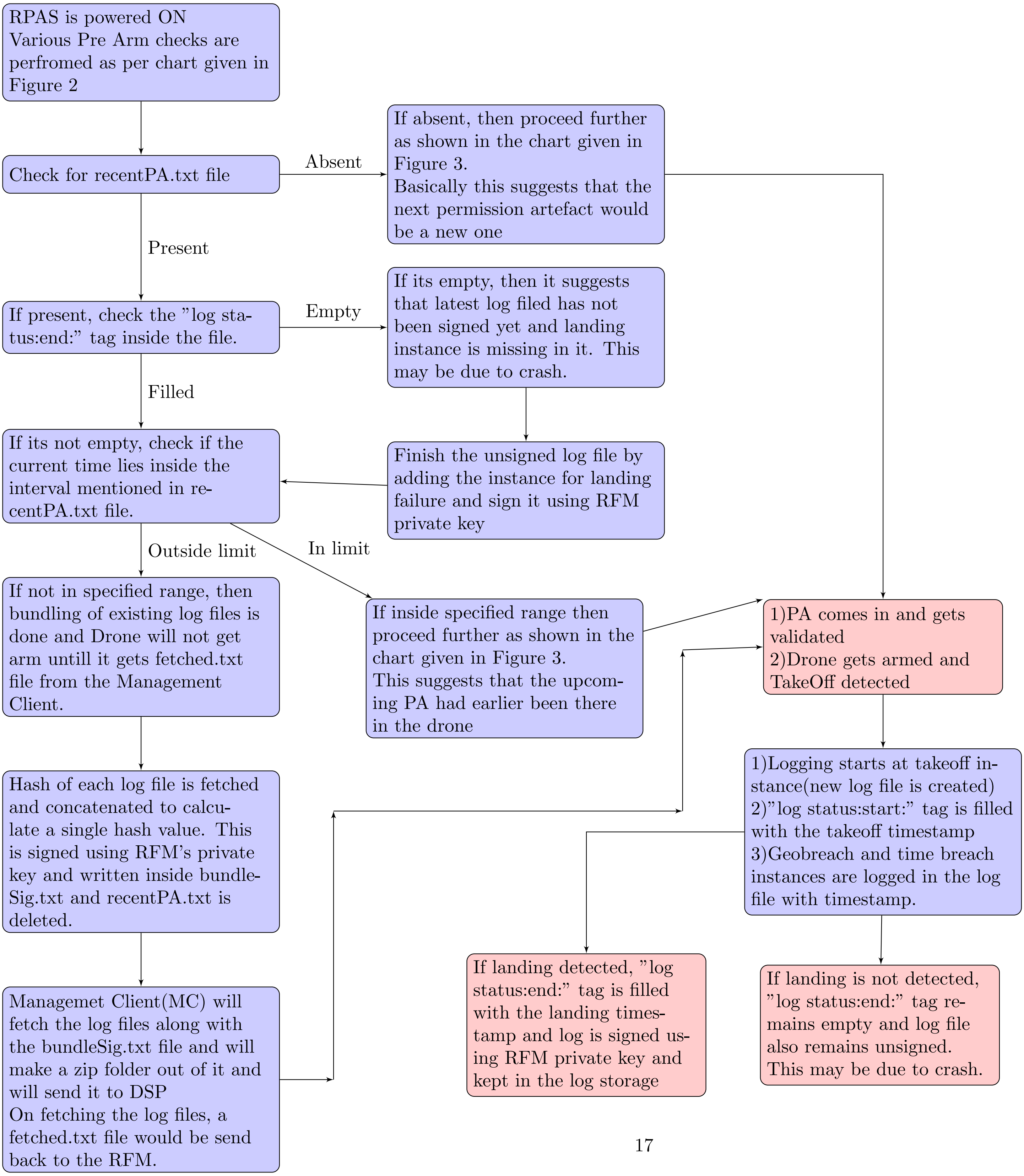


Figure 7: Flight log management

9 Secure boot for firmware update

For securing the firmware update process, a bootloader with signing support is flashed upon the controller. The FMP public key is installed inside the bootloader code. In this way, whenever an attempt to upload a new firmware binary file is made, a signature checking is performed. Only if the binary file is signed using the FMP private key, the flashing would be successful, otherwise, the code won't get outside the bootloader code, and new firmware won't get uploaded. Binary of the firmware is signed using the FMP private key.

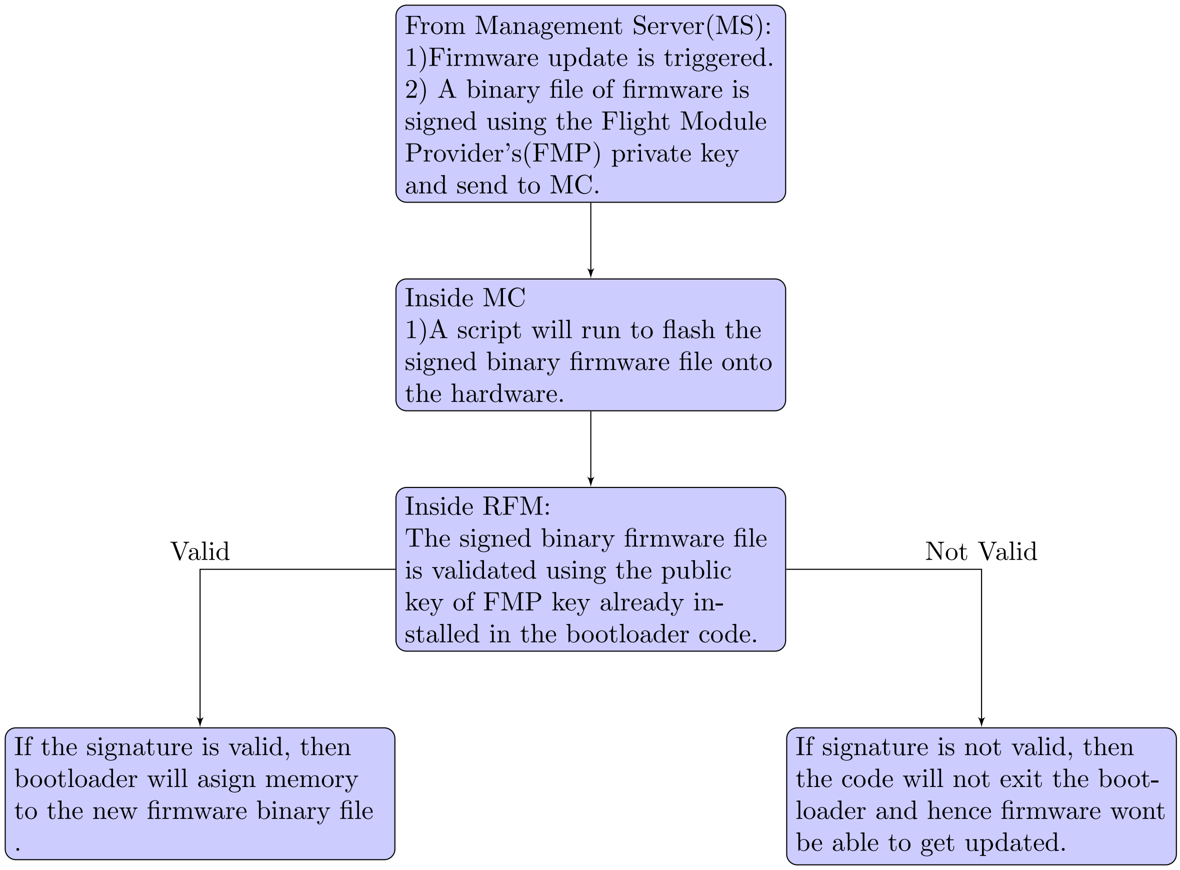


Figure 8: Secure Firmware Update