PROJECTE FINAL DE CARRERA

Estudi de la seguretat de les comunicacions de control de vols no tripulats.

Estudis:	Enginyeria de Telecomunicacions Universitat Politècnica de Catalunya.
Autor:	Ferran Alvarós Alvarez
Codirectors:	Paz Morillo Bosch Jorge Villar Santos
Empresa:	Singular Aircraft
Data:	25 de Juny de 2015

Agraïments

Primerament voldria agrair a la Paz i el Jorge totes les hores dedicades i el suport que m'han donat des del primer dia que em vaig endinsar en aquest projecte. L'oportunitat de poder cursar un projecte amb aquest repte tecnològic tan atractiu i amb la possibilitat de combinar-ho amb una estada en una empresa ha valgut la pena i és gràcies a vosaltres. La criptografia era la meva primera elecció i estic molt content d'haver comptat amb l'ajuda de tots dos, moltes gràcies.

També voldria donar les gràcies a l'empresa Singular Aircraft per confiar en la universitat pública i en especial en l'ETSETB per a investigar i desenvolupar plantejaments que donin solucions a reptes tecnològics tan bonics. A més a més, he tingut la sort de treballar 6 mesos a l'empresa i m'enduc una motxilla plena d'experiències importants. En especial vull agrair el seu tracte proper i sincer al Guillem i al Miquel.

No hauria estat possible arribar fins on he arribat sense l'ajut de moltes persones. Per això, voldria donar les gràcies als molts companys de la universitat amb els que hem compartit decepcions i alegries en especial al Xavi, l'Ignacio i el Robert.
També agrair als amics de sempre, del poble, i en especial a les incombustibles companyes d'estudi a la biblioteca: la Paula i la Pitu, amb algunes incorporacions puntuals. La barbaritat d'històries divertides que hem viscut, amb el temps, taparan els dies que hem passat patint les llums dels florescents.

A la Júlia, per molt lluny que vagis a viure sempre serem veïns. Hem de seguir trobant moments per veure'ns i sortir a passejar.

Als Salouistes, a les Quets, als Amb il·lusió i als boines, a tots. En especial al meu millor amic, el Simpto Gros o l'Ós Bru o el Pius Fulgenci. Tot depèn del dia. Sempre hi ets quan toca ser-hi, sempre puc comptar amb tu. Moltes gràcies.

Abans d'acabar vull agrair a l'Enric que m'ensenyés i em mostrés el camí per, al cap d'uns anys d'universitat, acabar entrant a l'Omega. El valor més gran de la ETSETB, sense cap mena de dubte. L'edifici Omega és la casa de l'estudiant, on els universitaris agrupen inquietuds i emprenedoria per a portar a terme grans fites. En el meu cas la Telecogresca, difícilment mai he après o aprendré tant en tan poc temps. Telecogresca té aquell factor màgic multiplicador on tothom que hi entra surt amb més del que ha donat. De tota la família vull agrair especialment a l'Enric, que m'hi va dur, i a les persones amb les que he compartit les dues Juntes en les que he format part: la Íngrid, el Marcel, el Sala, el Coronado, el Carvajal, l'Arnau, el Trelis i el meu copresidente Jordi Pegueroles.
"¡Telecogresca es un sentimiento!"

Finalment vull fer arribar el meu agraïment als meus pares i a la meva germana gran, els més importants. El projecte porta el meu nom però porta els vostres cognoms, sapigueu que sempre estaré agraït per l'estima i l'educació que m'heu regalat. Sou la meva primera xarxa.

Índex

Índex de Taules i Esquemes:

Índex d'Esquemes

Índex de Taules

Introducció

Entorn i Contingut:

En el següent Projecte Final de Carrera hi trobarem una solució a un repte tecnològic plantejat per Singular Aircraft. Singular Aircraft és una empresa fabricant d'avions no tripulats dirigits a distància mitjançant diverses estacions base situades en furgonetes.

Com veurem, aquests avions disposen d'un doble radioenllaç de comunicació, un enllaç principal i un enllaç d'emergència. Els enllaços actuen a diferents freqüències i tenen característiques molt diferents, tant, que el tractament de cadascun d'ells s'ha fet de forma diferent per assolir els objectius, també diferents, que es marcaven per a cada enllaç.

La feina d'aquest Projecte parteix de la necessitat d'afegir les característiques criptogràfiques de la integritat, la confidencialitat i l'autenticació de missatge en el cas de l'enllaç principal i la de garantir l'autenticació en el cas de l'enllaç d'emergència.

Durant la lectura d'aquest Projecte Final de carrera observarem que s'ha definit un sistema PGP (pretty good privacity) per l'enllaç principal. Aquest sistema criptogràfic es caracteritza per l'ús de criptografia simètrica i asimètrica de forma conjunta aprofitant les millors propietats de cadascuna de les dues criptografies. Rapidesa en el xifrat per part de la criptografia simètrica i robustesa de les claus en el cas de la criptografia asimètrica. En la implementació del PGP per l'enllaç principal s'ha considerat l'estàndard simètric AES (Advanced Encryption Standard) i l'estàndard asimètric RSA. Tots dos protocols criptogràfics usats en molts sistemes actuals.
En el cas de l'enllaç d'emergència, degut a les seves limitacions sobretot en quantitat (bits) d'informació transmesa per trama, s'ha optat per un disseny innovador de criptografia simètrica que hem anomenat criptografia estratificada. El seu funcionament de forma general és dividir la trama a enviar en diferents nivells segons probabilitat d'error de bit i fer ús de signatures digitals truncades per a encabir l'autenticació en la porció de la trama dedicada criptografia. A la vegada, la impossibilitat de generar autenticació amb una sola trama ens ha obligat a fer agrupacions de trames per a garantir el bon funcionament dels algoritmes d'autenticació de missatge que hem fet servir. Això provocarà una diferència de temps entre els missatges rebuts i les autenticacions d'aquests. Hem de procurar trobar un compromís entre quantitat de bits a autenticar i temps retard que li suposa a l'autenticació.

El Projecte Final de Carrera partia inicialment amb l'objectiu de satisfer la seguretat del radioenllaç, tot i així, durant el transcurs d'aquest estudi han anat desencadenant-se noves necessitats que les

hem convertit en objectius d'aquest PFC.

L'ús del protocol PGP, i específicament de l'estàndard RSA, com a base de funcionament criptogràfic de l'enllaç principal va mostrar-nos la necessitat de generar una estructura de gestió de confiances que, adaptada a l'esquema empresarial de Singular Aircraft, facilités la posta a punt d'una missió. Com veurem, contemplarem el sistema de ventes dels avions i les estacions base per part de Singular Aircraft i encaixarem l'estructura d'Autoritats de Certificació (CA's) de la forma més transparent possible.

Finalment, un cop el nostre escenari estava llest per a desenvolupar-se correctament vam creure totalment necessari introduir-nos en l'estudi del transport de claus. Com veurem, les claus i els certificats necessaris no es generaran en els avions o les estacions base, ho faran al que anomenarem Centre de Control de client. Aquesta distància entre la posició de generació i la posició d'ús de les claus ens ha portat a desenvolupar l'últim estudi d'aquest projecte sobre les targetes intel·ligents.

Estructura del treball:

La realització d'aquest projecte ha dut un any sencer de feina a partir de l'acord entre els tutors i l'alumne. L'alumne ha passat sis mesos integrat al cos de treballadors de Singular Aircraft per impregnar-se del dia a dia en la gestió i el muntatge dels diferents productes que ofereix l'empresa.
En la redacció d'aquest projecte hi fem constar 5 capítols que intenten seguir el fil argumental que hem seguit durant tot l'any i explicar els resultats del disseny que hem proposat com a solució al repte tecnològic inicialment plantejat.

El capítol 1 planteja el punt de partida, el per què d'aquest estudi, els objectius i les nocions bàsiques necessàries per entendre la resta del treball.

En el capítol 2 s'expliquen els protocols criptogràfics que usarem en els diferents radioenllaços per a assegurar els objectius marcats per l'empresa Singular Aircraft. El capítol es divideix en dues parts: l'enllaç principal i l'enllaç d'emergència. Hem considerat oportú incloure una forta base teòrica però sempre orientada al nostre escenari.

En el capítol 3 mostrem l'encaix de l'estructura empresarial de ventes per part de l'empresa i el sistema jeràrquic de confiances que proposem per al bon funcionament del sistema. El capítol s'inicia amb una base teòrica orientada estrictament a l'escenari en el qual ens trobem i continua oferint els mètodes pràctics per a posar en funcionament la cadena de confiances i de generació de claus i certificats.

El capítol 4 l'hem dedicat completament al transport del material criptogràfic mitjançant targetes intel·ligents. Al ser aquest món un gran desconegut per a nosaltres, hem cregut convenient explicar el funcionament d'aquestes targetes i els protocols de comunicació entre targetes i lectors/escriptors. Com s'observarà en aquest capítol fem una proposta d'adaptació del protocol estàndard que usen les targetes intel·ligents i finalment mostrem una possibilitat en la programació de la lectura/escriptura de les dades que conté o volem que contingui una targeta.

L'últim capítol està dedicat a les conclusions i les múltiples línies futures que obra aquest treball.

Capítol 1

Nocions bàsiques

1.1: Singular Aircraft:

Aquest projecte s'ha desenvolupat simultàniament en l'entorn del departament de matemàtica aplicada a la criptografia de la UPC i a l'empresa Singular Aircraft.

Singular Aircraft és una empresa que fabrica avions no tripulats de conducció a distància, altrament anomenats UAV (unmanned aerial vehicle). A la vegada aquests avions es tripulen des de terra en unes estacions base que estan integrades en una furgoneta. Aquesta estació base està dotada de tots els controls necessaris per a pilotar l'avió. La comunicació entre l'estació base i l'avió es duu a terme per tres radioenllaços simultàniament: el principal, el d'emergència i el de vídeo.
L'objectiu d'aquest Projecte Final de Carrera i el motiu pel qual van contactar amb l'ETSETB era estudiar la possibilitat d'afegir criptografia en els canals principals i d'emergència tenint en compte que cada enllaç té unes característiques diferents i així doncs unes necessitats criptogràfiques diferents.

La intenció de Singular Aircraft és vendre les parelles conformades entre avions i estacions base. Els seus clients, els compradors, podran fer ús d'aquests avions per a diferents tasques com extinció d'incendis, vigilància nocturna, transport d'ajuda humanitària o transport de mercaderies entre d'altres moltes finalitats. Aquesta venda del producte obliga a la criptografia a generar estructures de confiança per a la certificació de les claus que s'usaran i el transport d'aquestes a qualsevol punta del planeta. Certificar clients, emparellar avions i bases o transport de les claus, entre molts altres punts importants, són aspectes que tractarem als capítols tres i quatre d'aquest projecte final de carrera.

Per a que es puguin fer una idea de l'aspecte que fa l'avió en deixem les característiques seguidament:

Generals:

- Envergadura: 14 metres
- Longitud: 11,5 metres
- Altura: 3,6 metres
- Pes màxim: 3800 kg
- Tren d'aterratge: Patí amb cua retràctil

- Superfícies d'operació: Aigua i Terra.
- Pes quan és buit: 1750 kg
- Capacitat de càrrega: 2050 kg

Enlairament:

- Distància d'enlairament: 313 metres
- Pista d'enlairament: 209 metres
- Ascens: V_Y = 1850 peus/min
- Ascens amb un motor no operatiu: V_Y = 260 peus /min
- VNE = 142 Kts
- Velocitat de creuer potència 75% = 126 Kts
- Velocitat de creuer potència 65% = 103 Kts
- Alçada màxima d'operació: 24.000 peus.

Rendiment:

- Potència instal·lada: 2x340 CV
- Capacitat de combustible (bàsic): 200Kg
- Capacitat de combustible (ferry): 1700 Kg

Autonomia:

- Bàsica: 6h 45 minuts
- Màxima (tasques de vigilància): > 50h

Tot seguit continuem amb el capítol introductori on es procurarà assenyalar i descriure tots els punts que tractarem durant el transcurs de tot el Projecte Final de Carrera.

1.2: Tipus de Radioenllaços:

L'escenari que se'ns planteja en termes de comunicació es basa en dos radioenllaços diferents. El primer l'anomenarem enllaç principal i el segon enllaç d'emergència. Les característiques i limitacions de cadascun d'ells les definim a continuació:

Enllaç Principal:

- S'usa en uplink i downlink
- Integritat mitjançant CRC 32 bits
- Privacitat mitjançant Spread Spectrum propietari del fabricant.
- Privacitat mitjançant protocol propietari del fabricant.
- Opcionalment privacitat mitjançant AES 128 bits.
- Característiques del Uplink:
    - Les dades que circulen per l'enllaç ho fan seguint un protocol específic. A priori no es poden interpretar aquestes dades sense el coneixement del protocol. Tot i així com qualsevol protocol consta de la seva repetitivitat i el seu ordre més o menys simples, motiu pel qual, des del punt de vista de privacitat, podem dir que no n'aporta.
    - L'autenticació de missatges està pendent.

Enllaç d'emergència:

- Només uplink.
- No hi ha integritat
- No hi ha privactitat
- Modulació PPM (4,8 Kbps): Trames de 11 canals de 10 bits cadascun. El temps de trama és de 22,6 ms (44,25 Hz)
    - Per evitar problemes de resolució no s'hauria d'utilitzar més de 8bits/canal (3,9 Kps)
    - Actualment la captura de dades a enviar es fa al voltant de 20Hz i ocupa tota una trama. Per tant, es fa servir gairebé el 50% de la capacitat d'un enllaç (20/44).
    - Es podria afegir integritat posant un hash en les trames lliures.
- Característiques del Uplink:
    - L'enllaç no consta de cap mena de recurs per a la verificació de l'emissor, encriptació o control d'errors.
    - De les 11 trames de pujada, l'objectiu és usar-ne 10 per a informació i 1 per a tècniques criptogràfiques.

Tal com indica el seu nom l'enllaç principal serà la primera opció per a la comunicació entre l'avió i l'estació base. En el cas que aquest enllaç falli, sigui pel motiu que sigui i seguint uns criteris determinats per l'empresa, serà el moment d'usar el canal d'emergència per una comunicació temporal. En el cas que aquest enllaç falli també l'avió passarà a vol automàtic seguint els patrons introduïts en el pla de vol entregat en l'establiment de la connexió, just abans de començar la missió.

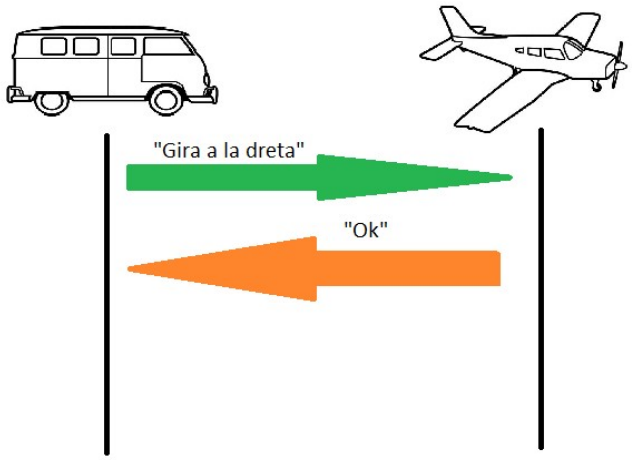
1.3: Esquemes de comunicació i necessitats criptogràfiques.

Enllaç Principal:

En aquest apartat tractarem l'esquema de comunicació entre els agents Avió i Estació Base en el que es basa el funcionament de les nostres missions. L'objectiu principal és, de forma genèrica, remarcar les necessitats criptogràfiques que són necessàries desenvolupar per tal d'assegurar els enllaços.
Per a simplificar els esquemes tractarem les comunicacions amb una relació MxN = 1x1. Essent M={nº d'estacions de control (bases)} i N={nº d'avions}. Veurem més endavant que si aconseguim una relació fiable 1x1 es pot generalitzar a qualsevol dimensió de l'esquema.

Esquema 1: *Sistema Isolat:*

1x1 sense atacants.

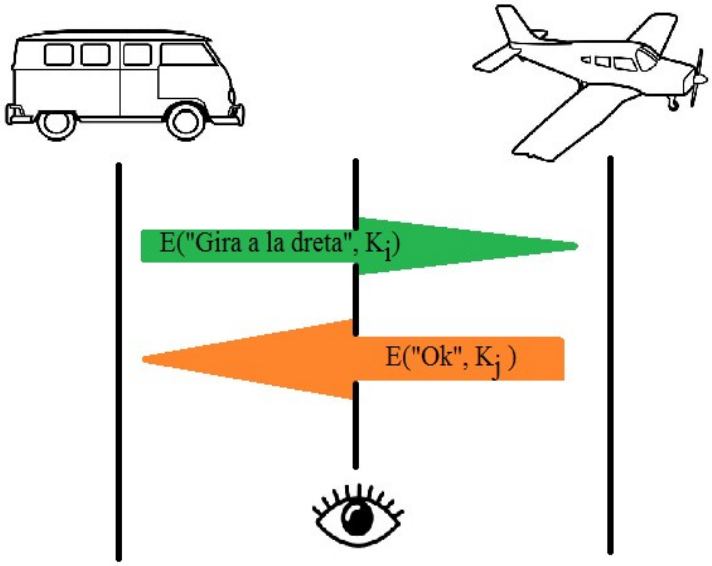


Esquema 1

Al no haver possibles atacants aquest esquema ens permet transmetre en clar i per tant no exigeix cap necessitat criptogràfica.

Esquema 2: *Sistema Oïent:*

1x1 amb observadors que no busquen atacar.



Esquema 2

En aquest esquema la flota, que només consta d'un avió, és coneixedora que només es comunicarà amb la base i viceversa. Tot i així es vol que l'observador no pugui aconseguir informació dels missatges transmesos entre ells per tant apareix una **primera necessitat criptogràfica**.

L'observador amb un atac de *"man in the middle"* com el que s'observa en la figura podria escoltar el transit de missatges entre base i flota, però si d'aquests missatges l'observador no en pot extreure informació es seguirà garantint la **confidencialitat**. Per a que l'observador no en pugui extreure informació dels missatges és necessari que aquests estiguin xifrats.

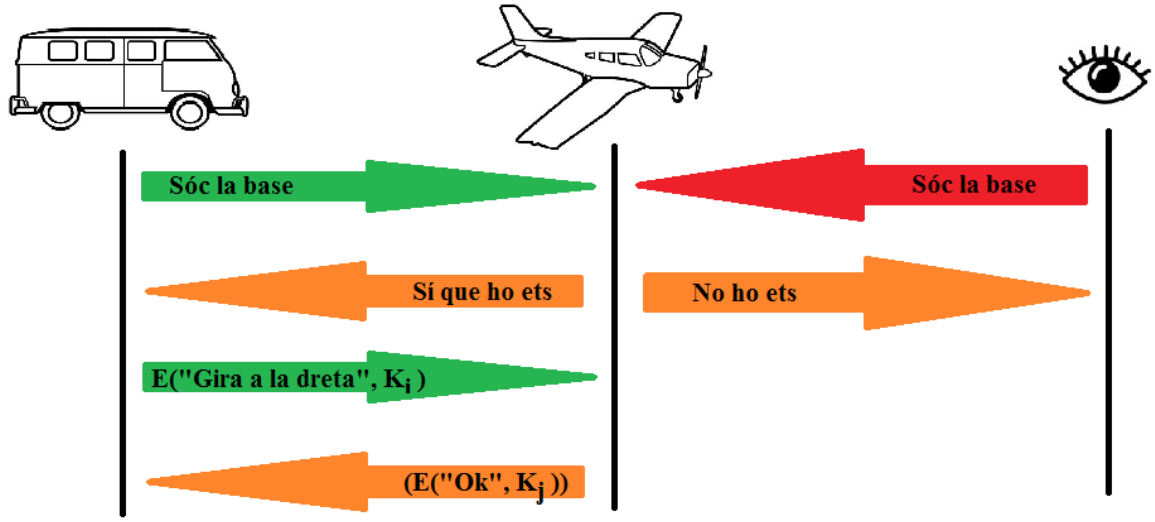
En la figura s'observa com la base llança el missatge xifrat "gira a la dreta" com a *E("Gira a la dreta", K_i)* on *E(A,B)* indica que el missatge *A* és xifrat amb la clau *B*.
La clau, en el cas de la figura 2, anomenada K_i pot ser de molts tipus (p.e. RSA, Sessió) i s'analitzarà més tard quines opcions ofereixen el millor rendiment.
El missatge de tornada de la flota al avió *E("Ok", K_j)* segueix el mateix patró, on K_j és la clau escollida.
No és estrictament necessari que K_i i K_j hagin de ser diferents.

Així doncs d'aquest esquema extraiem el que n'anomenarem **Necessitat 1**.

Necessitat 1: Xifrat
Missatge confidencial

Esquema 3: ***Sistema Atacant a flota:***

1x1 amb atacant a la flota.



Esquema 3

En aquest tercer esquema observem una **segona necessitat criptogràfica,** la base s'ha d'**autenticar** amb la flota per a que aquesta accepti qualsevol instrucció.
Aquesta situació, entre d'altres mesures, facilitarà descartar qualsevol *"atac allau"* on un col·lectiu d'atacants vulguin fer caure el port d'accés de la flota amb un atac massiu d'instruccions. Si un atacant no s'autentica la flota ignorarà els seus missatges.
Un dels punts més importants d'aquest protocol d'autenticació d'entrada és que aquest procés també ens permetrà fer la **gestió de claus** entre flota i base com s'explicarà posteriorment.

En el cas que l'atacant, en comptes d'atacar a l'avió, decidís enganyar a la base, fent-se passar per un avió qualsevol, les necessitats criptogràfiques d'autenticació també seran necessàries i per tant necessitem autenticació bidireccional.

Un cop el camí està establert i autenticat, els missatges s'envien xifrats tal com es feia en l'esquema anterior. A més a més és interessant afegir a la **necessitat 1 integritat, autenticació** i **no repudi** a tots els missatges complementant la ja necessària confidencialitat de l'esquema 2.

Necessitat 1: Xifrat
Confidencialitat en el missatge
Autenticitat de missatge
Integritat en el missatge
No repudi en qualsevol missatge intercanviat.

Taula 1

Necessitat 2: Establiment de connexió
Autenticació Bidireccional d'entrada
Gestió de claus

Taula 2

Enllaç d'emergència:

Com ja hem anteriorment l'enllaç d'emergència només s'usa en una direcció, el de pujada. A més aquest enllaç limita la criptografia a un canal de cada 11 que omplen una trama. Veiem la següent taula:

Trama Canal d'emergència: (110 bits)	
# Canal	**Informació del Canal (10 bits)**
1	Alerons
2	Elevador
3	Deriva
4	Potència motor esquerre
5	Potència motor dret
6	Força frenada fre esquerre
7	Força frenada fre dret
8	Posició Flaps (0º, 15º, 30º, 40º)
9	Fail Safe (ON/OFF), Tren (Up/Down), Bloqueig Cua (Bloquejada/lliure) [3 bits major pes]
10	Motor 1 (ON/OFF), Motor 2 (ON/OFF) [2 bits major pes]
11	Criptografia

Taula 3

Cada trama son 110 bits ja que cada canal consta de 10 bits d'informació. Es genera una trama pel canal d'emergència cada 22.6 ms.

L'enllaç d'emergència només ens reclama **autenticació** i **integritat** de missatge.

Definicions:

En els subapartats anterior ens hem referit a les característiques bàsiques de la seguretat, definim-les:

- Autenticació: Propietat de seguretat que garanteix identificar el generador de la informació.

- Integritat: Propietat de la seguretat que garanteix que les dades rebudes no han estat modificades per tercers. Arriba justament el mateix que es va enviar.

- Confidencialitat: Propietat de la seguretat que garanteix que la informació enviada només és assolida per al destinatari.

- No repudi: Propietat de la seguretat que proporciona protecció contra la interrupció, no repudi simètric és aquell que tant un extrem com l'altre de la comunicació no poden negar haver enviat i/o rebut alguna cosa ja que en emetre o rebre s'intercanvien proves infalsificables generades expressament per aquesta causa.

1.4: Fonaments bàsics de criptografia.

Escenari 1: RSA, criptografia asimètrica i firma digital usant funcions de hash:

El sistema RSA de clau pública és un dels mètodes més usats en la criptografia actual. Aquest sistema basa el seu funcionament en que cada entitat que vol usar el sistema disposa d'un parell de claus, una pública i una privada. Les dues claus es complementen en el sentit que si xifres amb una d'elles, només la seva parella és capaç de desxifrar el missatge.

El xifrat RSA es basa en la matemàtica modular i la particularitat dels nombres primers.

Generació de la parella de claus:

- Generar dos nombres primers p i q, grans i no gaire semblants. Els anomenarem *Nombres RSA*
- Calcular $n = p \cdot q$
- Calcular $\Phi(n) = (p-1) \cdot (q-1)$
- Generar un nombre e tal que no sigui ni múltiple ni submúltiple de $\Phi(n)$. *mcd(e,Φ(n))=1.*
- Trobar $d = e^{-1} \bmod(\Phi(n))$ usant el Algoritme Extès d'Euclides.
- La Clau Pública serà el conjunt **(e,n)** i la Clau Privada serà **d**.

Aquest procés és computacionalment molt costós però només s'ha de realitzar una vegada en la generació de claus. La generació pot ser en qualsevol moment previ a la utilització de les claus, per tant, **no genera un punt crític** de l'Escenari

Xifrat/desxifrat signat/validat amb la parella de claus pública/privada:

Opció	**Xifrat amb clau pública**	**Signat amb clau privada**
Xifrat/Signat	$C = M^e \bmod (n)$	$C = M^d \bmod (n)$
Desxifrat/Validat	$M = C^d \bmod (n)$	$M = C^e \bmod (n)$

Taula 4

Com es pot entendre el cost computacional depèn de la mida dels nombres RSA.

Per fer-ho més entenedor realitzem l'aplicació dels conceptes anteriors en un exemple numèric amb els nombres RSA (p,q) = (3,11):

- Directament s'obtenen els valors següents:
  - n = 33
  - Φ(33) = 20
- Escollim el valor *e* de tal manera que no tingui múltiples comuns amb Φ(33) = 20, per exemple *e=7*.
- A partir de les dades anteriors es calcula la clau privada $d = 7^{-1} \bmod(20)$. En aquest cas tant simple és directe veure que l'invers de 7 en mòdul 20 és 3 ja que compleix la condició següent:
  - $7 \cdot d \bmod (20) = 1$ --> $7 \cdot d = \{21, 41, 61...\}$ --> $7 \cdot 3 = 21$ --> d=3

- Així doncs hem generat la clau pública **(e,n)=(7,33)** i la clau privada **d=3**.
- Ara xifrem el missatge M=9 amb la clau pública (e,n) = (7,33).
  - $C = 9^7 \text{ mod}(33) = 4.782.969 \text{ mod}(33) = 15$
- Finalment desxifrem el criptograma C=15 amb la clau privada d=3.
  - $M = 15^3 \text{ mod } (33) = 3.375 \text{ mod}(33) = 9$
- El missatge ha estat correctament desxifrat.

Com hem pogut copsar el grau de computació amb nombres primers RSA petits ja és força elevat per a fer-ho mentalment. A mesura que escollim nombres primers RSA majors, la dificultat d'implementar l'algoritme RSA augmenta. Per exemple siguin p=23 i q=17, s'obté n=391. Amb una clau pública (3, 391), una clau privada d=235 i un missatge M=34 l'operació de xifrat/desxifrat és:

Xifrat: $34^3 \text{ mod}(391)$
Desxifrat: $204^{235} \text{ mod}(391)$

Observant el nivell de la potència en el desxifrat ens podem fer una idea de l'enorme capacitat de càlcul que serà necessària.

Per què hauríem de confiar en l'algoritme RSA? [1]

Un eventual espia pot conèixer els valors de *e* i *n* ja que són públics. Per a desxifrar un missatge l'espia necessitarà a més a més el valor de *d*, la clau privada. Tot i així anteriorment s'ha vist que el valor de la clau privada es genera a partir dels valors de *e* i de *n* per tant on radica la seguretat?

La seguretat del sistema RSA radica en la necessitat de descompondre el valor de *n* en el producte dels nombres primers *p* i *q*. Si *n* és suficientment gran (de l'ordre de 100 dígits), no hi ha manera coneguda de trobar *p* i *q* en un període de temps raonable.

Autenticació i Integritat mitjançant Firma digital en RSA:

Quan es parla de RSA és essencial definir la **funció de Hash**. La funció de hash transforma un missatge d'entrada M de longitud variable i el converteix en un altre de sortida de longitud fixa H(M). La funció de Hash és la peça clau per a poder intercanviar missatges amb **firma digital**.

Requisits i Propietats de la funció de hash *H(M)=h*:

- M pot ser de qualsevol longitud
- H(M)=h ha de ser d'una longitud fixa
- H(M) és computacionalment fàcil de calcular per a qualsevol x.
- Donat un valor de hash 'h' és computacionalment impossible trobar un missatge 'M' d'entrada que verifiqui H(M)=h.
- No pot passar que dos missatges M_1 i M_2 tinguin el mateix hash; $H(M_i) \neq H(M_j)$ (per tot $i \neq j$) (d'això s'anomena col·lissió)

Imaginem ara un escenari en que Alice i Bob es volen intercanviar un missatge M firmat digitalment. Els passos a seguir serien els següents:

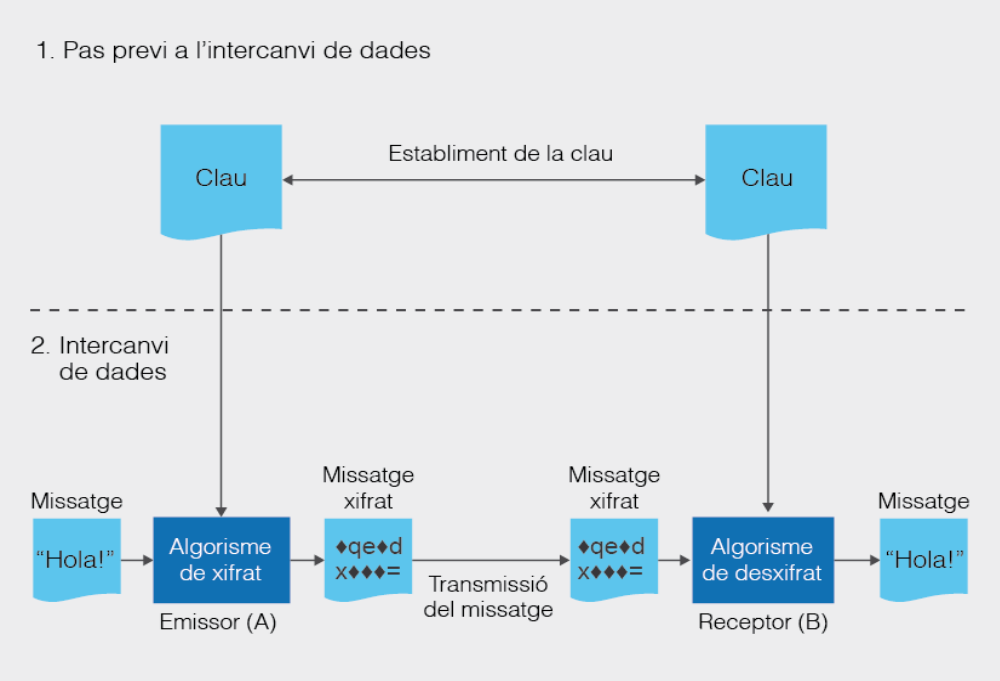
- Alice calcula el hash del missatge M. $H(M)=h$
- Alice signa el hash del missatge amb la seva clau privada. $E(h,s_{Alice}) = h'$
- Alice envia el missatge M i el hash xifrat h' a Bob.
- Bob calcula el hash de M. $H(M)=h$
- Bob valida el hash rebut amb la clau pública d'Alice. $E(h',p_{Alice}) = h''$
- Bob compara h amb h'' si coincideixen la firma és vàlida.

Entrarem més a fons en el tipus de signatura que es realitzarà en el capítol 2 d'aquest projecte final de carrera.

Escenari 2: Claus Simètriques o de sessió:

De l'escenari anterior, el terme més preocupant és la capacitat computacional necessària que han de tindre els agents que participen a les missions, a més a més el temps que és necessari invertir en el desxifrat dels missatge. Per a solucionar aquests problemes definim el xifratge amb clau simètrica o altrament dit amb claus de sessió.

Les necessitats d'aquest escenari són les mateixes que en l'anterior i l'ús molt similar com podem veure en la següent figura que escenifica el xifratge d'un missatge mitjançant clau simètrica.



Esquema 4

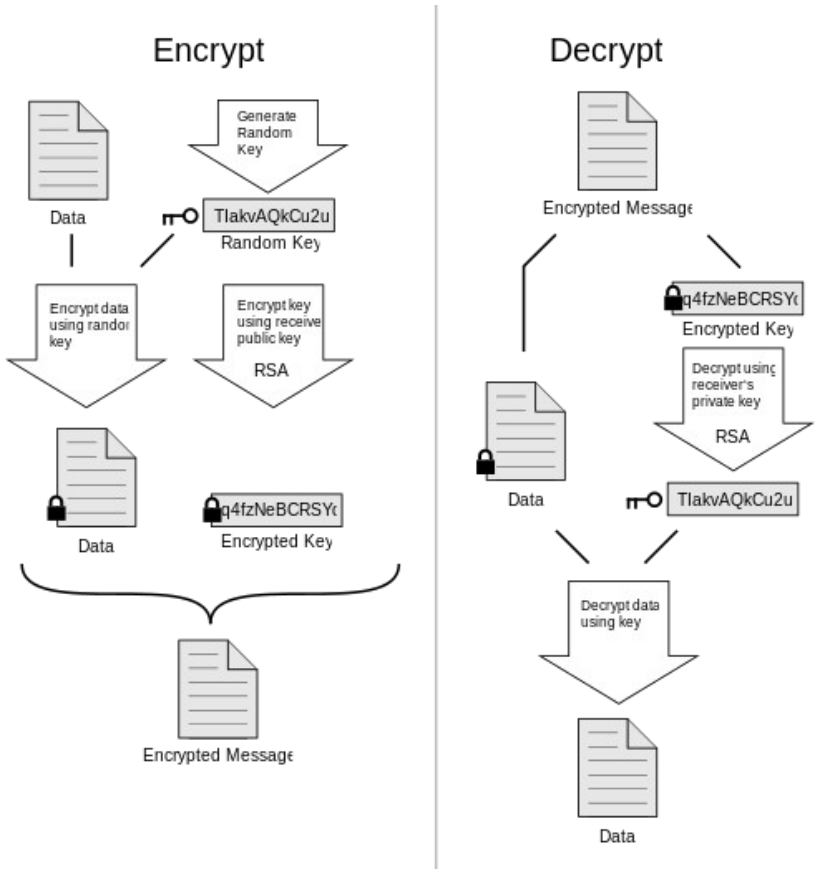
Així doncs podem observar que hi ha dos parts en la comunicació. Un primer pas d'establiment de clau i un segon pas de xifratge.

Tot i així, si l'establiment i renovació de les claus en els diferents agents de comunicació el considerem resolt, el xifrat i desxifrat és senzill i ràpid d'implementar.

Escenari 3: PGP (Pretty Good Privacy):

Vistes les eines de criptografia simètrica i asimètrica, volem destacar un tipus de criptografia híbrida que usarem en el capítol 2 d'aquest projecte final de carrera.

El PGP fa ús dels dos tipus de criptografia vistos anteriorment seguint el protocol de xifrat i desxifrat que explica la figura següent:



Esquema 5

Fixem-nos que funciona de la següent forma:

- Els dos agents de la comunicació han de conèixer només les claus asimètriques.
  - Clau privada pròpia
  - Clau pública pròpia i contrària.
- El transmissor genera una clau de sessió (simètrica) i xifra les dades amb aquesta.
- El transmissor xifra la clau de sessió amb:
  - La seva clau privada. En aquest cas estarà signant el missatge, autenticació. A més a més es pot validar la integritat del missatge.
  - La clau pública del receptor. En aquest cas estarà assegurant la confidencialitat del missatge a més a més de la integritat.
- Es concatena les dades xifrades amb la clau xifrada i s'envia a receptor qui al rebre el bloc de dades:
  - Divideix el bloc de dades entre el missatge xifrat i la clau xifrada.

- Primer de tot aconsegueix la clau de sessió verificada usant la clau necessària:
  - La clau pública del transmissor.
  - La seva clau privada.
- Usant la clau de sessió en clar el receptor ja pot desxifrar el bloc de dades.

Així doncs podem veure que si es pot garantir el funcionament de RSA en el nostre escenari també es podrà garantir l'ús de PGP que sobretot és més eficient en les velocitats dels xifrats i desxifrats.

El més important d'aquesta introducció als diferents tipus de xifrats és adonar-nos que amb les eines vistes fins ara podem encarar un escenari de comunicació segura entre estacions bases i avions.

Notacions importants:

Durant el desenvolupament del projecte ens trobarem punts com els següents:

- $Cert(K_{PA1})[K_{SCl1}]$
- K_{SBS1}
- $Cert_{PER}(\text{“L'avió A000001 pertany al Client 1”})[K_{SSA}]$

En el capítol corresponent ja entrarem a fons en com és un certificat, tot i així, i abans de començar la lectura d'aquest PFC és important que siguin conegudes les següents notacions:

Notació	**Significat**
K_{SBS1}	Clau asimètrica secreta de l'estació base 1
K_{PA1}	Clau asimètrica pública de l'avió 1
$Cert(K_{PA1})[K_{SCl1}]$	Certificat clàssic de la clau pública de l'avió emès i signat per client 1
$Cert_{PER}(\text{“L'avió A000001 pertany al Client 1”})[K_{SSA}]$	Certificat de pertinença, missatge que destaca una relació de pertinença entre dos agents del sistema. Emès i signat per Singular Aircraft.

Taula 5

1.5: Smart Card.

En el capítol 4 d'aquest projecte final de carrera encarem el repte del transport físic de les diverses claus i certificats, entre d'altres dades criptogràfiques. La solució escollida és l'ús de les Smart Card, aquestes són unes targetes de plàstic que tenen integrat un microprocessador.

En el cos del capítol explicarem ben bé què son i per a què serveixen però introduïm ara, fent una pinzellada, la seva utilitat i el seu funcionament.

Com a concepte l'Smart Card és una eina de transport d'informació. En comparació a d'altres elements de transport, com les memòries USB, les Smart Card disposen, més enllà d'una certa quantitat de memòria, d'un microprocessador que les dota de capacitat de càlcul i per tant les fa capaces de processar informació i executar algoritmes prèviament instal·lats.

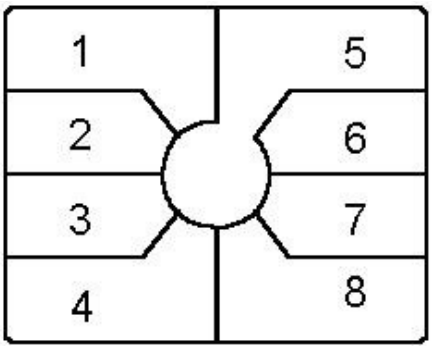
Tipus de memòria de la Smart Card:

Les Smart Card tenen tres tipus de memòria: ROM, EEPROM i RAM.

- ROM (Read Only Memory): S'usa per guardar els programes de la targeta. No fa falta subministrament d'energia externa per a mantindre les dades en aquest tipus de memòria. Tot i així, com diu el seu nom, no s'hi pot escriure un cop fabricada. Normalment inclou les rutines del sistema operatiu així com dades permanents i aplicacions d'usuari.
- EEPROM (Electrical Erasable Programable Read – Only Memory): La diferència significativa amb la memòria de tipus ROM és que el contingut d'aquesta memòria es pot modificar. Les aplicacions d'usuari també poden escriure en ella. Un dels continguts més important és la dada que dóna valor al número de cicles d'escriptura en el temps de vida de la targeta. En la gran majoria de targetes les EEPROM poden suportar com a mínim 100.000 cicles d'escriptura i poden emmagatzemar dades durant 10 anys.
- RAM (Random Access Memory): S'usa com espai temporal de treball per guardar-hi i modificar-hi dades. La RAM no és una memòria persistent, és a dir, la informació que conté no es pot preservar quan la font d'alimentació s'apaga. A la memòria RAM s'hi pot accedir un número il·limitat de cops. Com a dada comparativa amb la EEPROM podem destacar que una cel·la de RAM ocupa quatre vegades l'espai d'una cel·la de memòria EEPROM, tot i així l'escriptura a la RAM és al voltant de 1.000 vegades més ràpida que l'escriptura a la EEPROM.

Electrònica i punts de contacte d'una Smart Card:

Una Smart Card conté vuit punts de contacte, la funcionalitat de cada un és diferent i l'explicarem a continuació. Totes les especificacions de mesures i localització d'aquests contactes es troben especificades a la part 2 de la ISO 7816.



Esquema 6

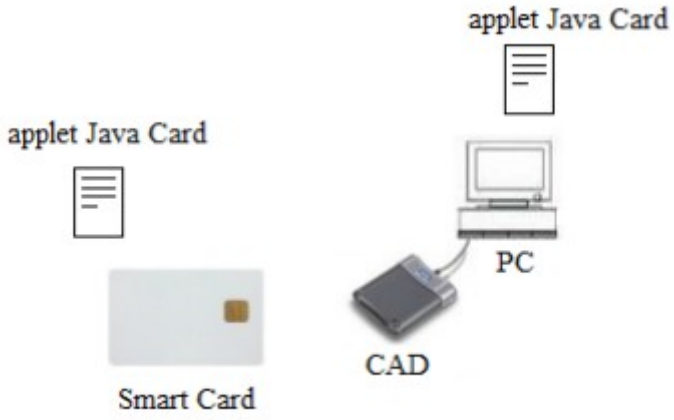
1. V_{CC}: És el punt de subministrament d'energia al xip. Normalment acostuma a tenir un valor de 3 a 5 Volts.
2. RST: S'usa per enviar un senyal de Reset al microprocessador. Aquest tipus de Reset és l'anomenat Reset en calent. L'altre tipus de Reset, el fred, s'executa apagant i encenent la font d'alimentació (per exemple extraient la targeta del lector).
3. CLK: És el punt que proveeix el rellotge extern al xip. D'aquest rellotge extern es deriva el rellotge intern.
4. RFU: Aquest punt està reservat per aplicacions de les Smart Cards Wireless que usen l'enllaç de radiofreqüència tant per alimentar el xip com per a transmetre les dades.
5. GND: Aquest punt s'usa com a punt de referència de voltatge. Es considera que el seu valor ha de ser sempre 0.
6. V_{PP}: És un punt heretat dels primers models d'Smart Card, ja no se'n fa ús. Tot i així aquest punt serveix per a subministrar el voltatge de programació, que té dos nivells (alt i baix). El nivell alt antigament es feia servir per a programar la EEPROM.
7. I/O: És el punt d'entrada i sortida de dades de la Smart Card. El canal és half dúplex que significa que tant hi poden entrar com sortir dades però les trames només es poden transmetre en una direcció en un mateix instant de temps.
8. RFU: Aquest punt està reservat per aplicacions de les Smart Cards Wireless que usen l'enllaç de radiofreqüència tant per alimentar el xip com per a transmetre les dades.

Comunicació entre Host i Smart Card:

Els sistemes de comunicació amb la Smart Card consten de dues parts:

- Software Host: Resident a un ordinador connectat al lector
- Software slave: Resident a la Smart Card.

Aquesta comunicació basa el seu funcionament en Applets de Java Card, llenguatge de programació amb moltes similituds amb el Java. Sigui com sigui, serà necessari instal·lar una Applet a cada extrem de la comunicació tal com s'observa a continuació:



Esquema 7

Les Applets de JavaCard instal·lades serveixen per interpretar les trames de dades que s'intercanvien els dos agents de la comunicació i executar correctament les instruccions que aquestes dades indiquen.

Com és habitual en una comunicació en que hi apareixen elements genèrics com són les Smart Card, per assegurar que les aplicacions generades per diferents programadors i les targetes fabricades per diferents treballadors puguin treballar junts es defineixen estàndards que marquen les pautes a seguir. Aquestes trames de dades que s'intercanvien els dos agents de la comunicació segueixen un protocol anomenat APDU establert en la ISO 7816.

ISO 7816, conceptes bàsics:

La ISO 7816 és l'estàndard més important relacionat amb les targetes intel·ligents. Gestionat conjuntament per l'Organització Internacional de Normalització (ISO) i la Comissió Electrotècnica Internacional (IEC). Algunes de les seves parts més importants són les següents:

- Part 1: Especifica les característiques físiques de les targetes de contacte. Els dos tipus de targetes més utilitzades actualment són:
  - ID_1: Mida de les targetes de crèdit
  - ID_0: Mida de les targetes SIM dels telèfons mòbils.
- Part 2: Especifica les dimensions i ubicacions dels contactes de la targeta.
- Part 3: Especifica la interfície elèctrica i protocols de transmissió per les targetes asíncrones.
- Part 4: Especifica l'organització, la seguretat i les comandes per l'intercanvi de dades.
- Part 15: Especifica l'aplicació de xifrat de la informació.

A partir de la Part 4 totes les que venen són independents de la tecnologia d'interfície física. Tan apliquen a targetes d'accés per contacte físic com per les d'accés per radiofreqüència.

Vist doncs les parts més importants de la ISO 7816, d'on s'ha extret molta informació per al desenvolupament d'aquest projecte final de carrera, creiem necessari la introducció de la tecnologia Java Card ja que en el capítol 4 també en farem ús.

Tecnologia Java Card:

La tecnologia Java Card és l'estàndard de programació que reconeixen les Smart Card. El Java Card ajuda a poder executar programes escrits en llenguatge de programació Java en les Smart Card o altres dispositius petits amb recursos limitats. Els desenvolupadors poden construir, testejar i implementar programes usalt les eines del entorn de desenvolupament estàndard i seguidament convertir-los a un format que pot ser instal·lat en un dispositiu que suporti la tecnologia Java Card. Les aplicacions escrites per la plataforma Java Card s'anomenen Applets.

Tot i així és important comentar que les Smart Card i els petits dispositius amb recursos limitats estan lluny de poder suportar totes les funcionalitats de la plataforma Java degut a les seves limitacions tan en memòria com en capacitat de processament. Així doncs això ens complica la programació ja que la plataforma Java Card suporta un subconjunt de les característiques de Java. Remarquem en la següent taula el subconjunt de característiques suportades:

Característiques suportades	**Característiques no suportades**
Tipus de dades primitives: boolean, byte, short	Tipus de dades primitives: long, double, float, char.
Arrays d'una dimensió	Arrays multidimensionals.
Paquets, classes, interfícies i excepcions de Java.	Càrrega dinàmica de classes Grabage Collector (Recolector de brossa)
Orientació a Objectes	Clonació d'objectes.

Taula 6

A més a més hem de destacar que una de les característiques més importants de l'entorn d'execució Java Card és que proveeix una clara separació entre el sistema intern de gestió de la Smart Card i les aplicacions, ocultant així la complexitat subjacent i els detalls del sistema Smart Card.
L'Arquitectura bàsica d'una targeta Java Card conté varis elements software que descrivim a continuació:

Component Software	**Descripció**
Card SO	Sistema operatiu de la targeta
Native Services	Porten a terme les funcions de I/O, xifrat o assignació de memòria.
Java Card Virtual Machine (JCVM)	La màquina virtual de Java Card estableix l'execució a nivell de byte i el suport del llenguatge Java, incloent el maneig de les excepcions.
Framework	El conjunt de classes que implementen la API. Inclou l'enviament d'APDU's, la selecció de l'Applet i la instal·lació de l'Applet.
API	L'API defineix el conveni entre les peticions per a que l'Applet accedeixi al sistema operatiu i d'altres recursos o serveis.

Java Card runtime environment (JCRE)	L'entorn Java Card en temps real d'execució. Inclou la JCVM, el framework i l'API.
Industry Specific Extensions	Classes afegides que amplien els Applets de la targeta.
Applet	Aplicacions escrites en llenguatge parcial Java per ser usades en les Smart Cards.

Taula 7

Vistos tots aquest elements creiem que el lector ja està preparat per a entendre tot el desenvolupament del capítol 4.

Capítol 2:

Criptografia

Anteriorment hem explicat les característiques dels dos enllaços en que hem d'aportar criptografia i n'hem destacat les necessitats criptogràfiques per a cadascun d'ells. Repassem-les:

- Enllaç Principal (Uplink & Downlink)
  - Autenticació
  - Confidencialitat
  - Integritat
  - No Repudi
  - Gestió de Claus
- Enllaç d'Emergència (Uplink)
  - Autenticació
  - Integritat
  - Gestió de claus.

2.1: Suposicions inicials:

Per a duu a terme les solucions que presentarem i per a poder-les entendre de la forma més clarificadora possible aïllem l'escenari només en repte de xifratge. Per aquest motiu seguim les suposicions següents:

- Deixem la gestió de claus de moment aïllada del sistema i suposem que les claus necessàries són al lloc necessari i al moment necessari per a gestionar la comunicació.

- A la vegada suposem que la comunicació a l'enllaç principal consta de suficient amplada de banda per a que el volum de les dades afegides al gestionar la criptografia no sigui un problema.

- Suposem que la informació crítica (la que s'envia pel canal d'emergència) va duplicada pels dos canals i que per tant només fem cas del canal d'emergència en la situació extrema de fallida del canal principal.

- En el cas que fallin tots dos canals l'avió passa a pilot automàtic i si és necessari retorna i aterra al punt indicat en el pla de vol, que s'introduirà a l'inici de la missió.

- El canal d'emergència és analògic, en el sentit de que la probabilitat d'error és diferent per cada bit (bits menys significatius de cada subcanal tenen major probabilitat d'error).

- De cada subcanal, els bits menys significatius són poc fiables.

- El canal principal es comporta com un canal estàndard, amb probabilitat d'error uniforme.

- L'estació base és capaç de generar claus de sessió durant el transcurs de la missió.

2.2: Enllaç Principal:

El nostre objectiu per l'enllaç principal és aconseguir afegir, al gran bloc de dades que s'envia, les propietats criptogràfiques destacades anteriorment sense entorpir el canal ni afegir retards importants en el processament "on time" de les dades.

És important destacar que per aquests enllaços es transmeten les dades de pilotatge de l'avió. Evidentment la velocitat en el processament de les dades és un punt molt important, tant o més que el d'estar segurs que els missatges que rebem són fiables. Al cap i a la fi la capacitat de reacció del pilot i la immediatesa de l'avió en respondre a les ordres que rep pot ser determinant en qualsevol moment del vol. Per aquest motiu pel canal principal farem ús de la tècnica PGP (Pretty good privacy), una tècnica usada a Internet i també en aplicacions "streaming".

Situem-nos doncs en el punt descrit a continuació:
Una estació base i un avió es troben llestos per a iniciar una missió. Cadascun d'ells té les següents dades:

- Estació Base:
  - Clau Privada interna: $K_{S\,BS\,1}$
  - Clau Pública interna: $K_{P\,BS\,1}$
  - Clau Pública externa: $K_{P\,A\,1}$
  - Clau de sessió temporal: $K_{M\,1}$

- Avió:
  - Clau Privada interna: $K_{S\,A\,1}$
  - Clau Pública interna: $K_{P\,A\,1}$
  - Clau Pública externa: $K_{P\,BS\,1}$

Totes les claus han arribat seguint uns protocols que explicarem més endavant i per tant l'estació base com l'avió confien plenament en aquestes claus, tan les internes com les externes.

Passem ara a veure l'esquema general de comunicació usant PGP + signatura digital HMAC (hash message autentication code).

Transmissió:

Suposem que comencem la comunicació des de l'estació base i que el primer missatge que volem comunicar li diem M.

Missatge M
Dades a enviar.

Aleshores usant la clau simètrica de sessió temporal $K_{M\,1}$ xifrem el missatge M, anomenarem al resultat **M'**.

Missatge M' = E(M)[$K_{M\,1}$]
Dades a enviar xifrades per la clau de sessió.

A continuació xifrem la clau de missió amb la clau asimètrica pública de l'avió, assegurant-ne així la **confidencialitat**. Anomenarem aquest nou paquet de dades **K'**.

M' = E(M)[$K_{M\,1}$]
Dades a enviar xifrades per la clau de sessió.

K' = E($K_{M\,1}$) [$K_{PA\,1}$]
Clau simètrica xifrada amb la clau pública de l'avió receptor.

Seguim amb el protocol establert per PGP ara concatenem aquests dos blocs de dades generant-ne un de contigu. Posteriorment usem totes les dades per a generar un nou bloc de dades que sigui el resultat d'un algoritme irreversible.
Entenem irreversible com per exemple l'ús d'una funció pràcticament impossible d'invertir, existeix la funció que compleix $f(x) = y$ però pràcticament, computacionalment, no existeix la funció inversa que respon a $f^{-1}(y) = x$.
Aquest algoritme irreversible l'anomenarem momentàniament HMAC. Amb aquest pas aconseguim **Autenticat** i **integritat**.
El símbol de concatenar serà sempre el següent: a || b ; a concatenat amb b.

M' || K'
Bloc de dades xifrades.

HMAC
Resultat d'introduir a un algoritme irreversible les dades M' || K'.

Finalment el transmissor ha de concatenar aquests dos blocs i enviar-los.

M' || K' || HMAC
Bloc final de dades xifrades més signades a enviar.

Receptor:

Al rebre el missatge, el receptor comença la validació de la integritat i la autenticació del missatge rebut. Per fer-ho primerament desconcatena el missatge rebut separant el bloc M' || K' de l'HMAC. El segon bloc el reserva mentre calcula l'HMAC del bloc que inclou missatge i clau xifrada. Si coincideix, el receptor podrà assegurar que coneix l'emissor del missatge i que a més a més el missatge no ha estat modificat durant la transmissió. Ho podrà assegurar degut a que la funció HMAC depèn dels paràmetres d'entrada i al crear-la farem ús d'alguna clau que identifiqui perfectament l'emissor. Ho explicarem més endavant en el punt 2.2.3

M' || K'
Bloc de dades rebudes

HMAC
HMAC rebut

HMAC
HMAC calculat a partir del bloc de dades rebut.

Coincideixen?
• **Sí**: L'emissor és conegut i el missatge integre. • **No**: No podem confiar en aquest missatge.

Esquema 8

Si el missatge és correctament autenticat es continua amb el procés d'extracció de les dades. Del bloc de dades rebudes es desconcatena per quedar-nos amb el bloc K' i aquest el desxifrem amb la clau asimètrica privada de l'avió (receptor) $K_{S\,A1}$. El resultat d'aquest desxifrat serà la clau simètrica $K_{M\,1}$ que xifra M'.

M' = E(M)[$K_{M\,1}$]
Dades rebudes xifrades per la clau de sessió.

$K_{M\,1}$ = D(K') [$K_{S\,A\,1}$]
Clau simètrica desxifrada amb la clau privada de l'avió receptor.

Finalment un cop tenim la clau simètrica de sessió desxifrem el missatge M'.

M = D(M')[$K_{M\,1}$]
Ordres de pilotatge rebudes

Vistes l'estructura de la comunicació que plantegem entrem en més detall a cada un dels tres aspectes criptogràfics que usem:

1. Xifratge Simètric
2. Xifratge Asimètric
3. HMAC

2.2.1: Xifratge simètric. AES 128 [2] .

Al gener del 1997, l'Institut Nacional d'Estàndards i Tecnologia (NIST) va iniciar el camí que va acabar amb la definició de l'estàndard de "l'Advanced Encryption Standard" (AES). L'organisme NIST va fer una crida a tota la comunitat tecnològica a enviar propostes per a realitzar aquest nou estàndard. Les úniques limitacions que va posar era que els algoritmes proposats havien de suportar entrades de blocs de dades d'almenys 128 bits, a més a més les mides de les claus havien de ser de 128, 192 i 256 bits.

El procés de selecció es va dividir en dues rondes. A la primera ronda 15 de les 21 propostes van ser acceptades com a candidates. Després d'una discussió pública d'aquestes propostes el NIST va escollir 5 finalistes per esdevenir l'estàndard de xifrat simètric de blocs. Els finalistes van ser:

- MARS (IBM)
- RC6 (RSA)
- Rijndael (Daemen and Rijmen)
- Serpent (Anderson, Biham and Knudsen)
- Twofish (Counterpane)

A l'Octubre del any 2000 NIST va prendre la decisió de seleccionar l'opció Rijndael per a ser l'AES. El disseny inicial del xifrat Rijndael suportava diferents mides dels blocs de dades entrants i claus (128, 160, 192, 224, 256 bits). Finalment l'AES estandarditzat va limitar les mides a 128, 192 i 256 bits.

Entrem doncs a analitzar l'algoritme AES i les seves particularitats.

AES és l'algoritme de clau simètrica més conegut pels usuaris ja que les xarxes Wifi amb tecnologia WPA usen AES com a mètode de xifrat. Com a característiques bàsiques i essencials s'han de destacar les següents:

- AES és un xifrat simètric de blocs.
- El xifrat pot estar implementat tant en software com en hardware.
- AES opera amb blocs de dades i claus de 128, 192 i 256 bits.
- L'algoritme AES treballa repetint els mateixos passos concrets diverses vegades.
- L'AES treballa a nivell de byte tant a l'entrada com a la sortida.
- Com és natural, AES és reversible. Amb AES podem xifrar però també es pot desxifrar.

Tal com hem comentat l'algoritme treballa a nivell de byte. L'entrada i la sortida són considerats i tractats sempre durant tot el xifrat i desxifrat com a blocs unidimensionals de múltiples de 4 bytes (32 bits).

El nombre d'iteracions que farà l'algoritme queda fixat amb el valor que prenguin les mides dels blocs de dades i la clau de sessió. Mostrem a la següent taula el valor d'iteracions en funció de diferents valors típics del bloc de dades i de la clau de sessió.

Mida de la clau de sessió normalitzada (en bits)	**Mida dels blocs de dades normalitzada (en bits)**				
	4	**5**	**6**	**7**	**8**
4	10	11	12	13	14
5	11	11	12	13	14
6	12	12	12	13	14
7	13	13	13	13	14
8	14	14	14	14	14

Taula 8

Entenem el concepte de normalitzar els valors en dividir cada un d'aquests valors (en bits) pel bloc primer de l'algoritme AES de 32 bits. Així doncs si l'execució de l'algoritme AES tingués com a paràmetres d'entrada una clau de 4 x 32 =128 bits i la mida dels blocs que divideixen el missatge entrant fos de 4 x 32 = 128 bits, hauria de iterar 10 vegades el mateix pas que més tard definirem per a xifrar o desxifrar el missatge.

Un dels aspectes més importants de l'AES és el tractament de la clau en cada una de les iteracions de l'algoritme i és que una de les fortaleses de l'AES és que en cada iteració la clau que s'usa és diferent. Per a que s'entengui bé la clau simètrica de sessió és única, però d'aquesta clau l'algoritme s'encarrega de generar una clau ampliada amb la que podrà assolir totes les iteracions que hagi de fer.
Per exemple en un AES amb clau de sessió de 128 bits i blocs de dades de 128 bits necessitarem una clau ampliada d'un total de 1408 bits. Per a fer-ho AES defineix l'algoritme d'ampliació de clau que definim a continuació.

2.2.1.1: Ampliació de clau AES.

Abans de xifrar o desxifrar és necessari fer una ampliació de la clau. La necessitat d'ampliar la clau rau en la obligació per definició d'usar una clau diferent en cada iteració. Explicarem per això el seu ús més endavant. Centrem ara l'atenció en com generar aquesta clau ampliada i les mides necessàries d'aquestes.

La clau ampliada ha de ser suficientment gran per a assumir que al ser trossejada pugui donar resposta al número d'iteracions que es produiran més una ronda extra.

Per tant la mida de la clau ampliada (en bytes) serà sempre igual a:

- 128 x (nombre d'iteracions + 1)

On 128 és la mida dels blocs de dades en bits. Per exemple una taula resum seria:

Mida clau normalitzada (bits)	Blocs de dades normalitzats (bits)	Mida clau ampliada normalitzada (bits)
4	4	44
6	4	52
8	4	60

Taula 9

Així doncs per exemple per a un AES-128 amb clau i blocs de dades de 128 bits necessitaríem 44x32=1408 bits de clau ampliada, o el que és el mateix, 128 bits x (10 iteracions +1) = 1408 bits.

Centrem-nos doncs en com es genera aquesta clau ampliada. Per fer-ho, i degut a la complexitat de l'explicació pel canvi de resultats depenent de la longitud inicial de la clau de sessió, centrem aquesta explicació en l'exemple de clau de 16 bytes i ampliada a 176.

Primerament expliquem les funcions que farem servir per a fer aquest augment de clau:

- **ROT WORD**: Aquesta funció demana d'entrada 4 bytes i els desplaça cíclicament una posició a la dreta. Per exemple:
  - Entrada: 1, 2, 3, 4 (número de byte)
  - Sortida: 2, 3, 4, 1 (número de byte)

- **SUB WORD**: Aquesta funció demana 4 bytes d'entrada i aplica una matriu de substitució a cada un dels bytes d'entrada. La matriu de substitució l'anomenarem **S-box**. Per exemple:
  - Entrada: El byte 1 equival a 0xD9
  - La matriu S-box és una matriu 16x16 on estan definits tots els valors hexadecimals que representen 4 bits.
  - Sortida: El byte 1 equivaldrà al valor fixat a la matriu en la posició S-box(D,9)

- **RCON**: Aquesta funció l'entrada depèn de la volta en que et trobis i la dimensió de la clau de la següent forma:
  - $\frac{Round}{\frac{KeySize}{4}} - 1$

  La sortida d'aquesta funció retorna 4 bytes basats en la taula Rcon_table explícita a continuació.

Entrada	**Sortida**
Rcon(0)	0x 01 00 00 00
Rcon(1)	0x 02 00 00 00
Rcon(2)	0x 04 00 00 00
Rcon(3)	0x 08 00 00 00
Rcon(4)	0x 10 00 00 00
Rcon(5)	0x 20 00 00 00
Rcon(6)	0x 40 00 00 00
Rcon(7)	0x 80 00 00 00
Rcon(8)	0x 1B 00 00 00
Rcon(9)	0x 36 00 00 00
Rcon(10)	0x 6C 00 00 00
Rcon(11)	0x D8 00 00 00
Rcon(12)	0x AB 00 00 00
Rcon(13)	0x 4D 00 00 00
Rcon(14)	0x 9A 00 00 00

Taula 10

Així doncs si per exemple ens trobéssim a la iteració número 4 la nostra entrada seria:

- $\frac{4}{\frac{16}{4}} - 1 = 0$

I la nostra sortida seria el valor:

- Rcon(0) = 0x 01 00 00 00

- **EK**: Aquesta funció reclama a l'entrada un valor d'offset i retorna 4 bytes consecutius de la clau ampliada aplicat l'offset d'entrada. Per exemple:
  - Entrada: 0
  - Sortida: Bytes 0, 1, 2, 3 de la clau ampliada
  - Entrada 4
  - Sortida: Bytes 4, 5, 6, 7 de la clau ampliada

- **K**: Aquesta funció reclama a l'entrada un valor d'offset i retorna 4 bytes consecutius de la clau de sessió inicial aplicant l'offset d'entrada. Per exemple:
  - Entrada: 12
  - Sortida: Bytes 12, 13, 14, 15 de la clau de sessió.

Creiem necessari especificar el significat del concepte Round usat a la funció Rcon.

- Round: És un comptador que vol representar la posició relativa al total de passos a fer per l'algoritme d'ampliació de clau. Per exemple per a generar la clau augmentada a partir de la clau de 16 bytes (128 bits) es necessiten 44 passos.

Tot seguit especifiquem els 44 passos a fer per a generar la clau augmentada a partir d'una clau matriu de sessió de 128 bits que recordem ha d'acabar sent de 176 bytes:

Round	**Número de byte de la clau augmentada**				**Funció**
0	0	1	2	3	**K**(0)
1	4	5	6	7	**K**(4)
2	8	9	10	11	**K**(8)
3	12	13	14	15	**K**(12)
4	16	17	18	19	**Sub Word**(**Rot Word**(**EK**((4-1)*4))) XOR **Rcon**((4/4)-1) XOR **EK**((4-4)*4)
5	20	21	22	23	**EK**((5-1)*4) XOR **EK**((5-4)*4)
6	24	25	26	27	**EK**((6-1)*4) XOR **EK**((6-4)*4)
7	28	29	30	31	**EK**((7-1)*4) XOR **EK**((7-4)*4)
8	32	33	34	35	**Sub Word**(**Rot Word**(**EK**((8-1)*4))) XOR **Rcon**((8/4)-1) XOR **EK**((8-4)*4)
9	36	37	38	39	**EK**((9-1)*4) XOR **EK**((9-4)*4)
10	40	41	42	43	**EK**((10-1)*4) XOR **EK**((10-4)*4)
11	44	45	46	47	**EK**((11-1)*4) XOR **EK**((11-4)*4)
12	48	49	50	51	**Sub Word**(**Rot Word**(EK((12-1)*4))) XOR **Rcon**((12/4)-1) XOR **EK**((12-4)*4)
13	52	53	54	55	**EK**((13-1)*4) XOR **EK**((13-4)*4)
14	56	57	58	59	**EK**((14-1)*4) XOR **EK**((14-4)*4)
15	60	61	62	63	**EK**((15-1)*4) XOR **EK**((15-4)*4)
16	64	65	66	67	**Sub Word**(**Rot Word**(**EK**((16-1)*4))) XOR **Rcon**((16/4)-1) XOR **EK**((16-4)*4)
17	68	69	70	71	**EK**((17-1)*4) XOR **EK**((17-4)*4)
18	72	73	74	75	**EK**((18-1)*4) XOR **EK**((18-4)*4)
19	76	77	78	79	**EK**((19-1)*4) XOR **EK**((19-4)*4)
20	80	81	82	83	**Sub Word**(**Rot Word**(**EK**((20-1)*4))) XOR **Rcon**((20/4)-1) XOR **EK**((20-4)*4)
21	84	85	86	87	**EK**((21-1)*4) XOR **EK**((21-4)*4)
22	88	89	90	91	**EK**((22-1)*4) XOR **EK**((22-4)*4)
23	92	93	94	95	**EK**((23-1)*4) XOR **EK**((23-4)*4)
24	96	97	98	99	**Sub Word**(**Rot Word**(**EK**((24-1)*4))) XOR **Rcon**((24/4)-1) XOR **EK**((24-4)*4)
25	100	101	102	103	**EK**((25-1)*4) XOR **EK**((25-4)*4)

26	104	105	106	107	**EK**((26-1)*4) XOR **EK**((26-4)*4)
27	108	109	110	111	**EK**((27-1)*4) XOR **EK**((27-4)*4)
28	112	113	114	115	**Sub Word**(**Rot Word**(**EK**((28-1)*4))) XOR **Rcon**((28/4)-1) XOR **EK**((28-4)*4)
29	116	117	118	119	**EK**((29-1)*4) XOR **EK**((29-4)*4)
30	120	121	122	123	**EK**((30-1)*4) XOR **EK**((30-4)*4)
31	124	125	126	127	**EK**((31-1)*4) XOR **EK**((31-4)*4)
32	128	129	130	131	**Sub Word**(**Rot Word**(**EK**((32-1)*4))) XOR **Rcon**((32/4)-1) XOR **EK**((32-4)*4)
33	132	133	134	135	**EK**((33-1)*4) XOR **EK**((33-4)*4)
34	136	137	138	139	**EK**((34-1)*4) XOR **EK**((34-4)*4)
35	140	141	142	143	**EK**((35-1)*4) XOR **EK**((35-4)*4)
36	144	145	146	147	**Sub Word**(**Rot Word**(**EK**((36-1)*4))) XOR **Rcon**((36/4)-1) XOR **EK**((36-4)*4)
37	148	149	150	151	**EK**((37-1)*4) XOR **EK**((37-4)*4)
38	152	153	154	155	**EK**((38-1)*4) XOR **EK**((38-4)*4)
39	156	157	158	159	**EK**((39-1)*4) XOR **EK**((39-4)*4)
40	160	161	162	163	**Sub Word**(**Rot Word**(**EK**((40-1)*4))) XOR **Rcon**((40/4)-1) XOR **EK**((40-4)*4)
41	164	165	166	167	**EK**((41-1)*4) XOR **EK**((41-4)*4)
42	168	169	170	171	**EK**((42-1)*4) XOR **EK**((42-4)*4)
43	172	173	174	175	**EK**((43-1)*4) XOR **EK**((43-4)*4)

Taula 11

Fixe'm-nos que les funcions són cícliques cada 4 passos, només varia el número de Round que afecta als valors que introduïm a les funcions anteriorment definides. Analitzem ara particularment les primeres funcions del quadre anterior per a entendre bé què fa aquest algoritme.

En els primers 4 passos es copien exactament igual els 16 bytes de la clau matriu de sessió.

- *K(0), K(4), K(8) i K(12).*

Seguidament en el 5è pas s'executa la següent funció:

- ***Sub Word(Rot Word(EK**((4-1)*4))) XOR **Rcon**((4/4)-1) XOR **EK**((4-4)*4)*

Analitzem poc a poc aquesta funció:

- **EK**((4-1)*4)) = **EK**(12); Copia els bytes 12, 13, 14 i 15 ja existents de la clau augmentada.
- **Rot Word**(12, 13, 14, 15) = 15, 12, 13, 14.
- **Sub Word**(15, 12, 13, 14); Canvia els valors dels bytes d'entrada seguint la matriu 16x16 anomenada **S-box**. Així doncs per a posar un exemple no real:
  - Byte 15 = 0xD0; Posició matriu S-box(D,0) = 0x9A
  - Byte 12 = 0xF7; Posició matriu S-box(F,7) = 0x42
  - Byte 13 = 0x2B; Posició matriu S-box(2,B) = 0x8E
  - Byte 14 = 0xEA; Posició matriu S-box(E,A) = 0x6D

  Aleshores el valor entrant i sortint de la funció seria:
  - Entrant: 0x D0 F7 2B EA
  - Sortint: 0x 9A 42 8E 6D

- **Rcon**((4/4)-1) = **Rcon**(0) = 0x 01 00 00 00
- **EK**((4-4)*4) = **EK**(0); Copia els bytes 0, 1, 2 i 3 ja existents de la clau augmentada. En l'exemple:
  - **EK**(0) = 0x F7 2B EA D0

I així, finalment el que ens queda és el següent:

- 0x 9A 42 8E 6D XOR 0x 01 00 00 00 XOR 0x F7 2B EA D0 = 0x 6C 69 64 BD

Que aquests seran els bytes 16, 17, 18 i 19 de la clau augmentada.

Obtinguts els bytes 16, 17, 18 i 19 de la clau augmentada la següent volta de l'algoritme d'augment de clau serveix per a determinar els bytes 20, 21, 22 i 23. La funció és la següent:

- ***EK**((5-1)*4) XOR **EK**((5-4)*4)*

Analitzant poc a poc aquesta funció:

- **EK**((5-1)*4) = **EK**(16) = Bytes 16, 17, 18 i 19 de la clau augmentada ja generada.
- **EK**((5-4)*4) = **EK**(4) = Bytes 4, 5, 6 i 7 de la clau augmentada ja generada.

Així doncs fixe'm-nos que el resultat dels propers 4 bytes de la clau augmentada que estem generant són combinació de bytes que l'algoritme ja ha generat prèviament.

A partir d'aquest punt l'algoritme és cíclic fins aconseguir els 176 bytes necessaris.

2.2.1.2: Algoritme de xifrat AES.

Un cop vista la solució per a generar una clau augmentada posant com a punt de partida la clau matriu simètrica de sessió podem entrar a comentar el xifrat AES.
Com ja hem comentat anteriorment AES és un xifrat d'iteració de blocs. Això significa que les mateixes operacions o funcions són desenvolupades moltes vegades per a blocs de dades fixats. Aquestes operacions o funcions que s'executen cíclicament ens els diferents blocs de dades es poden resumir de la següent forma:

- **ADD ROUND KEY**: Aquesta funció revela la necessitat d'aplicar l'algoritme d'ampliació de clau. L'Add Round Key executa una XOR dels valors d'estat del bloc de dades a xifrar amb un bloc d'igual mida de la clau augmentada. Els bytes de la clau augmentada usats en una execució d'aquesta funció no es tornen a usar mai més. Per exemple:
  La primera vegada que executem Add Round Key:

Estat dels bytes del bloc de dades a xifrar.		Bytes de la clau augmentada
1	XOR	1
2	XOR	2
3	XOR	3
4	XOR	4
5	XOR	5
6	XOR	6

7	XOR	7
8	XOR	8
9	XOR	9
10	XOR	10
11	XOR	11
12	XOR	12
13	XOR	13
14	XOR	14
15	XOR	15
16	XOR	16

Taula 12

A la segona execució d'aquesta funció els bytes a usar de la clau augmentada seran els compresos entre [17, 32]. Consecutivament per a cada ronda d'execució.

- **BYTE SUB**: Aquesta funció fa ús de la matriu S-box, llegeix l'estat dels bytes entrants i els substitueix per els seu corresponent a la matriu de substitució. Com veiem segueix el mateix procediment que la funció SUB WORD de l'algoritme de generació de clau augmentada.

- **SHIFT ROW**: Aquesta funció afegeix un desplaçament cíclic al estat dels bytes entrants però no de forma plana sinó que fa ús de la generació d'una matriu per afegir-hi complexitat. Per aclarir-ho fem ús d'un exemple:
  Imaginem que les dades entrants son els bytes compresos en el segment [1,16]. Primerament construïm una matriu 4x4 que inclogui aquests bytes.

  1 5 9 13
  2 6 10 14
  3 7 11 15
  4 8 12 16

  A continuació desplacem circularment cada fila zero, una, dues o tres posicions en funció de la fila que sigui.
  - Fila 0: 0 posicions de desplaçament
  - Fila 1: 1 posició de desplaçament.
  - Fila 2: 2 posicions de desplaçament.
  - Fila 3: 3 posicions de desplaçament.

Quedant el següent resultat a la matriu:

1	5	9	13
6	10	14	2
11	15	3	7
16	4	8	12

La funció retorna la matriu desplaçada.

- **MIX COLUMN**: Aquesta funció rep una matriu de 4x4 de 16 bytes i retorna el mateix nombre de bytes modificats. Explicarem el funcionament d'aquesta funció en dos passos. Primerament reflectirem quin procés pateix la matriu d'entrada per a convertir-se en els bytes sortints. Com a segon punt remarcarem i explicarem com aquestes multiplicacions es duen a terme seguint el que s'anomena Espai de Galois.

  Multiplicació de matrius:
  En primer lloc la funció recull la matriu 4x4 entrant que conté el valor de 16 bytes. L'anomenarem Matriu d'Estats:

b1	b5	b9	b13
b2	b6	b10	b14
b3	b7	b11	b15
b4	b8	b12	b16

  Per altra banda la funció conté una Matriu de Multiplicació prefixada com podria ser la següent:

2	3	1	1
1	2	3	1
1	1	2	3
3	1	1	2

  El resultat dels bytes de sortida es calculen multiplicant les files de la matriu de multiplicació per les columnes de la matriu d'estats. El resultat de cada multiplicació interna entre una fila i una columna passen per una XOR per a produir 1 Byte només de sortida.
  Per a posar un exemple senzill, per a generar el primer byte de la matriu de sortida l'expressió seria la següent:
  - $b_1' = (b_1 * 2) XOR (b_2 * 3) XOR (b_3 * 1) XOR (b_4 * 1);$

  Seguiríem el mateix procediment per a generar [b_1', b_{16}'].

Multiplicació en el Cos de Galois:
El contingut teòric del Cos de Galois i la seva aplicació a criptografia es pot trobar a nombrosos artícles realitzats per experts tant de les matemàtiques com de la Criptografia. En aquest Projecte Final de Carrera explicarem tant sols com es realitza la multiplicació en aquest espai essent molt pragmàtics.

Per a executar les multiplicacions farem ús de les taules següents:

Taula E:

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	01	03	05	0F	11	33	55	FF	1A	2E	72	96	A1	F8	13	35
1	5F	E1	38	48	D8	73	95	A4	F7	02	06	0A	1E	22	66	AA
2	E5	34	5C	E4	37	59	EB	26	6A	BE	D9	70	90	AB	E6	31
3	53	F5	04	0C	14	3C	44	CC	4F	D1	68	B8	D3	6E	B2	CD
4	4C	D4	67	A9	E0	3B	4D	D7	62	A6	F1	08	18	28	78	88
5	83	9E	B9	D0	6B	BD	DC	7F	81	98	B3	CE	49	DB	76	9A
6	B5	C4	57	F9	10	30	50	F0	0B	1D	27	69	BB	D6	61	A3
7	FE	19	2B	7D	87	92	AD	EC	2F	71	93	AE	E9	20	60	A0
8	FB	16	3A	4E	D2	6D	B7	C2	5D	E7	32	56	FA	15	3F	41
9	C3	5E	E2	3D	47	C9	40	C0	5B	ED	2C	74	9C	BF	DA	75
A	9F	BA	D5	64	AC	EF	2A	7E	82	9D	BC	DF	7A	8E	89	80
B	9B	B6	C1	58	E8	23	65	AF	EA	25	6F	B1	C8	43	C5	54
C	FC	1F	21	63	A5	F4	07	09	1B	2D	77	99	B0	CB	46	CA
D	45	CF	4A	DE	79	8B	86	91	A8	E3	3E	42	C6	51	F3	0E
E	12	36	5A	EE	29	7B	8D	8C	8F	8A	85	94	A7	F2	0D	17
F	39	4B	DD	7C	84	97	A2	FD	1C	24	6C	B4	C7	52	F6	01

Taula 13

Taula L:

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	-	00	19	01	32	02	1A	C6	4B	C7	1B	68	33	EE	DF	03
1	64	04	E0	0E	34	8D	81	EF	4C	71	08	C8	F8	69	1C	C1
2	7D	C2	1D	B5	F9	B9	27	6A	4D	E4	A6	72	9A	C9	09	78
3	65	2F	8A	05	21	0F	E1	24	12	F0	82	45	35	93	DA	8E
4	96	8F	DB	BD	36	D0	CE	94	13	5C	D2	F1	40	46	83	38
5	66	DD	FD	30	BF	06	8B	62	B3	25	E2	98	22	88	91	10

6	7E	6E	48	C3	A3	B6	1E	42	3A	6B	28	54	FA	85	3D	BA
7	2B	79	0A	15	9B	9F	5E	CA	4E	D4	AC	E5	F3	73	A7	57
8	AF	58	A8	50	F4	EA	D6	74	4F	AE	E9	D5	E7	E6	AD	E8
9	2C	D7	75	7A	EB	16	0B	F5	59	CB	5F	B0	9C	A9	51	A0
A	7F	0C	F6	6F	17	C4	49	EC	D8	43	1F	2D	A4	76	7B	B7
B	CC	BB	3E	5A	FB	60	B1	86	3B	52	A1	6C	AA	55	29	9D
C	97	B2	87	90	61	BE	DC	FC	BC	95	CF	CD	37	3F	5B	D1
D	53	39	84	3C	41	A2	6D	47	14	2A	9E	5D	56	F2	D3	AB
E	44	11	92	D9	13	20	2E	89	B4	7C	B8	26	77	99	E3	A5
F	67	4A	ED	DE	C5	31	FE	18	0D	63	8C	80	C0	F7	70	00

Taula 14

El resultat d'una multiplicació en el Cos de Galois sortirà de la suma additiva dels valors imatge dels dos operands en la taula L i del resultat buscar-ne la imatge a la taula E.
Respecte la suma additiva volem destacar 2 característiques:

- És la representada pel signe + a nivell hexadecimal i no el resultat de fer una AND a nivell de bit.
- La suma és mòdul FF. Si el resultat de la suma és superior a la capacitat d'un bit (FF) aleshores del valor resultat extraurem FF per a quedar-nos amb aquell valor com a resultat.

Per a fer-ho més clarificador seguim amb un exemple:
Imaginem que d'entrada a la funció MIX COLUMN tenim la següent Matriu d'Estats:

D4	A7	25	78
BF	B2	6C	F6
5D	66	2A	4F
30	89	74	35

I posem pel cas que la Matriu de Multiplicació és la següent:

2	3	1	1
1	2	3	1
1	1	2	3
3	1	1	2

Seguint l'execució de la funció hem de realitzar la següent operativa per als primers 4 bytes de sortida:

- $b_1' = (D4 * 2)\ XOR\ (BF * 3)\ XOR\ (5D * 1)\ XOR\ (30 * 1);$

Abans de continuar l'execució d'aquesta funció volem destacar la següent propietat del Cos de Galois en la multiplicació:

- Qualsevol número multiplicat per 1 és igual a ell mateix. *FF * 1 = FF;*

Comentat aquest punt seguim amb el desenvolupament d'aquesta igualtat.

- $b_1' = \mathbf{E}(\mathbf{L}(D4) + \mathbf{L}(02))\ XOR\ \mathbf{E}(\mathbf{L}(BF) + \mathbf{L}(03))\ XOR\ 5D\ XOR\ 30;$

Si busquem els valors demanats a la taula L tenim el següent:

- $b_1' = \mathbf{E}(41 + 19)\ XOR\ \mathbf{E}(9D + 01)\ XOR\ 5D\ XOR\ 30;$

Fem efectives les sumes additives i cap és major de FF per tant no hem d'aplicar la propietat de suma modular (implícitament ho estem fent).

- $b_1' = \mathbf{E}(5A)\ XOR\ \mathbf{E}(9E)\ XOR\ 5D\ XOR\ 30;$

Busquem ara els valors necessaris a la taula E.

- $b_1' = B3\ XOR\ DA\ XOR\ 5D\ XOR\ 30;$

Finalment executem les XOR.

- $b_1' = 04;$

Aquest seria el primer byte que entregaria la funció a la sortida. Per la resta de bytes el procediment és anàleg.

Com a característica, aquesta i només aquesta funció s'aplicarà a totes les iteracions del xifrat menys a la última per a fer possible el desxifrat.

Vistes les funcions necessàries pel xifrat AES vegem ara l'aplicació recorrent d'aquestes funcions per generar a partir d'un bloc de dades el criptograma a enviar. L'exemple és per a una clau i blocs de dades de 128 bits cadascun, si recordem aquesta distribució ens permet fer 10 iteracions que són les que resumim en la següent taula:

Iteració	**Funció**
-	Add Round Key (State)
0	Add Round Key (Mix Column(Shift Row(Byte Sub(State))))
1	Add Round Key (Mix Column(Shift Row(Byte Sub(State))))
2	Add Round Key (Mix Column(Shift Row(Byte Sub(State))))
3	Add Round Key (Mix Column(Shift Row(Byte Sub(State))))
4	Add Round Key (Mix Column(Shift Row(Byte Sub(State))))
5	Add Round Key (Mix Column(Shift Row(Byte Sub(State))))
6	Add Round Key (Mix Column(Shift Row(Byte Sub(State))))
7	Add Round Key (Mix Column(Shift Row(Byte Sub(State))))
8	Add Round Key (Mix Column(Shift Row(Byte Sub(State))))
9	Add Round Key (Shift Row(Byte Sub(State)))

Taula 15

Com es pot observar la funció Add Round Key s'usa 11 vegades i això causa la necessitat abans anunciada de generar una clau ampliada que compleixi el criteri *número d'iteracions +1*.

Com ja sabem, un sistema de xifrat no té cap valor si no es coneix una forma de desxifrar-lo. En aquest Projecte Final de Carrera volem deixar anunciat també el procés per a fer el desxifrat que el resumim en la següent taula:

Iteració	**Funció**
-	Add Round Key (State)
0	Mix Column (Add Round Key(Byte Sub(Shift Row(State))))
1	Mix Column (Add Round Key(Byte Sub(Shift Row(State))))
2	Mix Column (Add Round Key(Byte Sub(Shift Row(State))))
3	Mix Column (Add Round Key(Byte Sub(Shift Row(State))))
4	Mix Column (Add Round Key(Byte Sub(Shift Row(State))))
5	Mix Column (Add Round Key(Byte Sub(Shift Row(State))))
6	Mix Column (Add Round Key(Byte Sub(Shift Row(State))))
7	Mix Column (Add Round Key(Byte Sub(Shift Row(State))))
8	Mix Column (Add Round Key(Byte Sub(Shift Row(State))))
9	Add Round Key (Byte Sub(Shift Row(State)))

Taula 16

Al usar les mateixes funcions ja explicades anteriorment deixarem en aquest punt l'estudi de l'AES-128.

2.2.2: Xifrat Asimètric; RSA.

El xifrat asimètric RSA pren el nom dels seus tres inventors: R. Rivest, A. Shamir i L.Adleman. Aquest xifrat proveeix d'algoritme de xifrat i també de signatura digital i ha aconseguit ser el sistema més popular en el xifrat de clau pública. La robustesa del RSA és deguda a la dificultat de descompondre en factors els nombres (grans) que ens permeten generar el xifrat.

El sistema de fonaments del RSA s'ha explicat a la introducció d'aquest mateix projecte final de carrera, repassem-ne els conceptes claus.

2.2.2.1: Generació de claus i xifrat.

Cada usuari que contingui el sistema al qual volem assegurar les comunicacions, haurà d'obtenir una parella de claus pública i privada. La generació d'aquestes claus es duu a terme en tres passos:

- S'escullen dos nombres primers p i q molt grans (ordre de 100 dígits) i es genera $n = p \cdot q$.
- S'escull un valor e que compleixi que no tingui múltiples comuns amb $\Phi(n)=(p-1)\cdot(q-1)$.
  - La clau pública que fem arribar a tota la resta d'agents amb els que es comparteix comunicació és la parella (n,e).
- Generem $d \cdot e = 1 \bmod (\Phi(n))$
  - Emmagatzemem de forma segura la clau privada d definida per la parella (n, d).

A nivell terminològic anomenarem mòdul a *n*, exponent de xifrat a *e* i exponent de desxifrat a *d*.

Un cop generades les dues claus recuperem com executem el xifrat:

Opció	**Xifrat amb clau pública**
Xifrat	$C = M^e \bmod (n)$
Desxifrat	$M = C^d \bmod (n)$

Taula 17

L'emissor xifra amb la clau pública del receptor. Per tant, aquest xifrat només podrà ser desxifrat pel receptor específic que determini la parella de claus.
En canvi si fem servir el mateix algoritme però usant la clau privada del emissor ja no n'anomenarem xifrat ja que qualsevol usuari coneixedor de la clau pública del emissor (tothom) podrà desxifrar el missatge i per tant anomenarem a aquesta funció signatura/validació digital.

Opció	**Signatura amb clau privada**
Signatura	$C = M^e \bmod (n)$
Validació	$M = C^d \bmod (n)$

Taula 18

Tota la matemàtica inherent a aquest procés i a les tècniques d'optimització en l'elecció dels nombres primers està ja escrita i no la definirem. Algoritmes i teoremes com l'Algoritme d'Euclides, l'Algoritme extès d'Euclides per a la generació d'aquests valors o l'Algoritme de Gordon per a trobar nombres grans i primers de forma més eficient, entre d'altres, seran molt útils i usats en RSA. En aquesta línia per tant recomanem al lector que vulgui aprofundir en els coneixements de RSA que consulti [3].

2.2.2.2: Dimensió de la clau.

Un possible atacant que conegués la clau pública *(n, e)* podria intentar extreure la clau privada descomponent el nombre *n* en dos nombres primers multiplicats per tal de poder conèixer *Φ(n)* i com a conseqüència poder calcular la clau privada *d*. Aquesta tasca però es considera computacionalment impossible en un termini de temps raonable si s'usen valors de *p* i *q* de voltant dels dos cents dígits decimals.

En aquest sentit ens plantegem quina mida de clau agafar, els valors típics són 1024 i 2048 però ens plantegem per què no d'altres. Per a escollir la mida de la clau primer de tot hem de tindre en compte els següents dos aspectes:

- Que sigui prou gran per a que no sigui desxifrable per un atacant.

  Com hem comentat com més gran sigui el mòdul (*n*) que defineix tant la clau pública com la clau privada més complexitat s'afegeix per a un possible atac.
  En l'actualitat es considera que si el mòdul està al voltant de les 400 xifres no és atacable.

Analitzem els valors típics de claus el nombre de xifres que comporten.

Mòdul clau *n* (bits)	**Nombre de xifres que representa.**
1024	$\log(2^{1024}) = 1024 \cdot \log(2) = 308$
2048	$\log(2^{2048}) = 2048 \cdot \log(2) = 616$

Taula 19

Veiem que com comentava com més augmentem el mòdul de la clau més segura la fem, aleshores podríem plantejar-nos claus de 4096 o fins i tot 8192 bits ja que ens disposarien de mòduls molt més grans i més segurs. Tot i així s'exigeix un compromís amb el següent aspecte.

- Que no augmenti desproporcionadament el temps de procés de desxifrat.

  Aquest aspecte seria important en el cas que xifréssim tot el missatge amb tecnologia de clau asimètrica però hem de recordar que l'únic que xifrarem amb aquesta clau serà la clau simètrica d'AES de 128 bits, en aquest cas la dimensió de la clau RSA no modifica els temps de procés en els desxifrats de les dades de forma significativa.

En el nostre escenari prenem la decisió de fer servir claus RSA de 2048 bits i no majors per un aspecte relacionat amb l'entorn en el que el nostre repte tecnològic s'engloba. El fet que ens limita la mida de les claus és que la Smart Card, que serà l'eina que usarem per a transportar aquestes claus, en l'actualitat està completament adaptada per l'emmagatzematge de claus de fins a 2048 bits i, tot i que ja n'hi ha que poden operar amb claus majors, aquesta característica no està estesa a tot el sector.

2.2.3: Codis d'autenticació de missatge MAC; HMAC.

Un cop explicats quin tipus de criptografia usarem tant per la part simètrica com per la part asimètrica volem assegurar també l'autenticació i la integritat del missatge. Per aquest motiu complementem la trama xifrada amb un MAC (Message Authentication Code). Analitzem ara la funcionalitat dels tipus de MAC més comuns.

Definició:

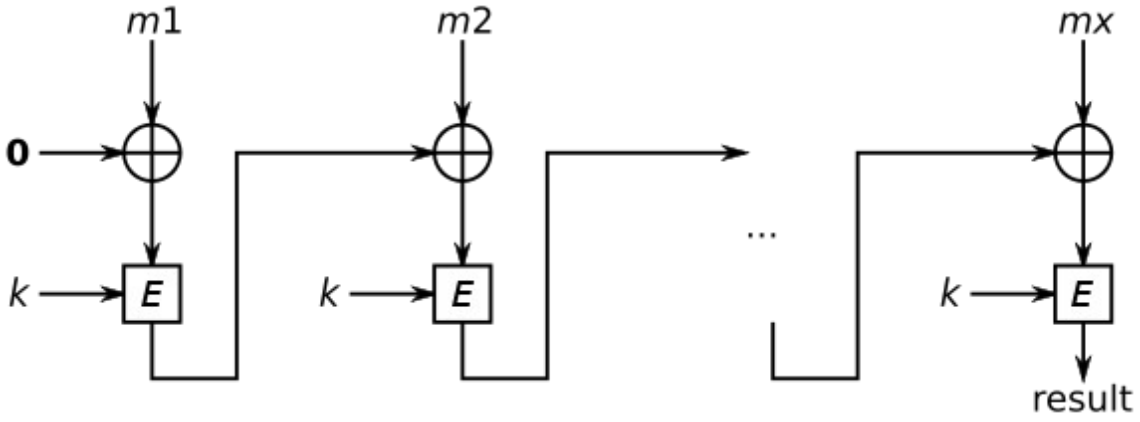
- Definim una funció MAC ideal aquella que tenint un missatge d'entrada M de llargada qualsevol retorna una sortida de n-bits (amb n fixa). Una funció MAC ideal genera sortides molt diferents per a petits canvis en el missatge d'entrada.
- Actualment existeixen tres grans grups de funcions MAC: **CBC-MAC**, **UMAC** i **HMAC**

Principi de funcionament:

- L'entitat origen genera el MAC i l'envia juntament amb el missatge al receptor. El receptor torna a calcular el MAC del missatge i el compara amb el rebut per l'origen. Si coincideixen es verifica l'autenticació del missatge.

2.2.3.1: CBC – MAC (Cipher block chaining message authenication code).

L'algoritme CBC-MAC usa el xifrat simètric per aconseguir un resultat semblant a una funció aleatòria. L'algoritme divideix el missatge en blocs més petits i els va introduint en les diferents etapes dels registres. El resultat de cada etapa depèn de la sortida de l'etapa anterior.



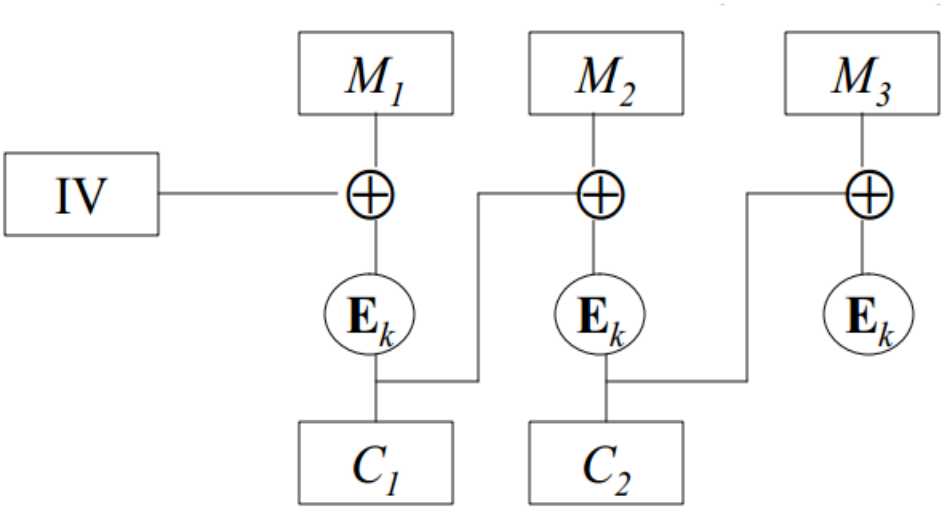
Esquema 9

L'alternativa que proposa aquest sistema és anar introduint els subblocs de 4 bits de dades de cada una de les 10 trames al sistema de registres preconfigurat. Al finalitzar comparar el resultat amb l'enviat als bits dels canals de criptografia.

CBC-MAC només garanteix la seva seguretat per longitud d'entrades fixes. Ara per ara no podem garantir que les entrades del canal principal siguin sempre de la mateixa mida. Singular Aircraft encara no ha determinat la quantitat de dades que s'enviarà per aquest canal i no descarta la possibilitat de que en algunes trames s'envii més informació que a d'altres.

Tot i així posem un exemple teòric de funcionament:

- Donat un bloc de xifratge E de mida m bits.
- Donat un missatge $M = M_1 \,||\, M_2 \,||\, ... \,||\, M_n$
- Donat un Vector inicial que anomenarem IV (Initial Vector)
- El MAC de M és $E_k(M_n \oplus C_{n-1})$
- Les sortides C_1 , C_2 ... C_n responen al següent algoritme:
  - $C_1 = E_k(M_1 \oplus IV)$
  - $C_2 = E_k(M_2 \oplus C_1)$
  - $C_n = E_k(M_n \oplus C_{n-1})$



Esquema 10

Si hi posem valors, imaginem que:

- $M = M_1 \,||\, M_2 \,||\, M_3 = 0010 \,||\, 1101 \,||\, 1010$
- $IV = 0100$
- El bloc sumador equival a una porta XOR
- La clau $k = 3$
- El bloc E del xifrat, només en aquest cas hipotètic, realitza una translació cíclica de k posicions.

Aleshores:

- $C_1 = E_K(M_1 \oplus IV) = E_K\,(0010 \oplus 0100) = E_K\,(0110) = 1100$
- $C_2 = E_K(M_2 \oplus C_1) = E_K\,(1101 \oplus 1100) = E_K\,(0001) = 0010$
- $C_3 = E_K(M_3 \oplus C_2) = E_K\,(\,1010 \oplus 0010) = E_K\,(1000) = 0001 = CBC - MAC$

2.2.3.2: UMAC.

L'algoritme UMAC [4] (Message Authentication Code using Universal Hashing) està dissenyat per ser molt ràpid d'executar.

UMAC basa el seu algoritme en les anomenades funcions de hash universals. Aquestes només consten de sumes de nombres de 32 i 64 bits i alguna multiplicació de 32 bits, operacions que suporten perfectament les màquines actuals i que es poden executar de forma molt veloç. A les universals els hi falta robustesa al no ser funcions criptogràfiques, a més a més exigeix la compartició de dues claus que serviran per a generar nombres pseudoaleatoris que faran que la funció de hash varii en cada autenticació. S'entén així la importància del sincronisme entre els diferents agents del sistema per avançar correctament en la cadena de la família de funcions de hash.

L'autenticació amb UMAC consta de dos passos, un primer de generació del codi MAC molt senzill i un segon que refresca les condicions inicials per a fer ús d'una funció de hash diferent en la següent autenticació.

El primer pas, donat un missatge M que es vol autenticar per a generar el MAC o etiqueta d'autenticació s'usa una funció de hash de la família de funcions de hash universals, que s'aplica al missatge M i a la clau k_1 per a produir un número de longitud fixa i curta (en bits) de sortida.

El segon pas determina la següent funció de la família que s'usarà, el valor del hash sortint es suma en XOR ($\oplus$) amb una clau derivada d'un camí pseudoaleatori fixat en l'establiment de connexió K_2. Del resultat d'aquesta $\oplus$ s'extreu la funció de hash de la família d'universals que haurem d'usar en la següent autenticació.

Amb UMAC podem dir que gaudim d'una estricta seguretat en l'anàlisi de l'autenticitat de missatges. Tot i així per a desenvolupar aquest estil de MAC és necessari que tots els agents de la comunicació comptin internament d'un bloc de xifratge que generi el camí aleatori de la segona clau K_2 a més a més de la família de funcions de hash universal i la clau interna K_1.

A continuació deixem les condicions que s'han de complir necessàriament per a poder generar UMAC de 32 bits en un sistema. Aquest esquema ha estat extret de [4].

```
Input:

K, string of length KEYLEN bytes. //En UMAC podriem parlar d'una clau
                                  AES de 128 bits.

M, string of length less than 2^67 bits. //Entrada inferior a 2^67 bits

Nonce, string of length 1 to BLOCKLEN bytes.

taglen, the integer 4, 8, 12 or 16 (bytes). //Sortida del hash de
                                           longitud variable.
                                       Possibilitat de 32bits.

Output:
```

```
Tag, string of length taglen bytes.

Compute Tag using the following algorithm:

HashedMessage = UHASH(K, M, taglen)// Generem el Hash amb la clau i el
                                   missatge

Pad = PDF(K, Nonce, taglen) //Generem la segona clau pseudoaleatòria
                        mitjançant un nombre "Nonce" que es
                  renova cada cop i la pròpia clau

Tag = Pad XOR HashedMessage //Calculem el UHASH final.

Return Tag
```

Com a resum, les funcions que s'haurien d'usar per a generar els UMAC(família de funcions de hash universals) reuneixen les següents característiques:

- Són funcions no criptogràfiques.
- Responen simplement a propietats de combinatòria (sumes i multiplicacions) i per tant tendeixen a ser molt ràpides.
- La família NH va ser la primera família de funcions hash universals però poc a poc n'han aparegut de noves com Poly1305, Badger, entre d'altres.

2.2.3.3: HMAC.

L'algoritme HMAC [5] usa les funcions de hash com a eina per a la generació de les etiquetes d'autenticació MAC.

La funció de Hash té com a entrada un missatge M de longitud variable i genera com a sortida una seqüència aleatòria de longitud fixa H(M).

Requisits i Propietats de la funció de hash *H(M)=h*:

- M pot ser de qualsevol longitud
- H(M)=h ha de ser d'una longitud fixa
- H(M) és computacionalment fàcil de calcular per a qualsevol x.
- Donat un valor de hash 'h' és computacionalment impossible trobar un missatge 'M' d'entrada que verifiqui H(M)=h.
- No és recomanable que dos missatges M_1 i M_2 tinguin el mateix hash; $H(M_i) \neq H(M_j)$ (per tot i ≠ j). Si aquest fenomen succeeix s'anomena col·lisió. Una bona funció de hash si permet col·lisions sempre serà amb una parella de missatges molt allunyades en contingut. (p.e. M_1=000001 i M_2=101100) així si un atacant canviés un bit a l'entrada podria ser detectat.

L'algoritme HMAC funciona de la següent forma:

- *HMAC(K,M) = H((K⊕opad) | H((K⊕ipad) | M))*

  on:

  - H(·) : És la funció de hash.
  - K: És la clau secreta amb farciment (padding) extra de zeros als bits de menys pes si la demanda (en bits) d'entrada de la funció de hash és major que la mida de la clau. En el cas que la clau sigui de major mida K serà el hash de la clau.
  - M: És el missatge a ser autenticat.
  - ⊕: denota XOR
  - opad: és el farciment de sortida.
  - ipad: és el farciment d'entrada.

  Així doncs veiem que és necessari computar dues vegades la funció de hash per a aconseguir l'etiqueta d'autenticació MAC.

Existeixen per això moltes funcions de hash estandarditzades i cada una d'elles respon a unes condicions diferents, tant de blocs d'entrada de dades permès com de mida fixa de sortida.

A la següent taula es pot observar les demandes de les principals funcions de hash:

Algoritme (Funció de Hash)	**Mida de sortida**	**Mida mínima de bloc.**
MD5	128	512
SHA-1	160	512
SHA-256	256	512
SHA-512,SHA-512/224,SHA-512/256	512/224/256	1024
SHA-3	512	576

Taula 20

Totes les funcions de hash anteriors són funcions molt estandarditzades i usades, ara o en el passat, en aplicacions reals d'autenticació. De totes elles les funcions de hash MD5 i SHA-1 ja han estat relegades per SHA-256 al haver-se trobat formes de trencar els seus algoritmes.

Dels protocols SHA-512/224, SHA-512/256 en parlarem quan arribem a l'enllaç d'emergència al ser algoritmes que permeten que es trunqui la sortida de la funció de hash, concepte que a l'enllaç d'emergència serà molt important.

L'any 2014 va aparèixer el primer esborrany de la funció de hash SHA-3 [6], s'espera per a finals d'estiu 2015 per a que aparegui la versió definitiva però l'esborrany ja ens dóna molta informació. SHA-3 es una evolució de SHA-1 i SHA-2 que repara alguns detalls en temps de procés i afegeix

noves funcions de generació.

SHA-3 proposa tractar les dades entrants en un espai 3D en forma de prisma rectangular on cada bit d'entrada es col·loqui a una posició de la figura. Jugant amb les files, línies i columnes i diferents operands matemàtics es genera un prisma sortint del que no es pot recuperar el prisma entrant. De fet és molt probable que SHA-3 acabi acceptant el format de HMAC sense haver d'aplicar la formulació actual que recordem:

- $HMAC(K,M) = H((K \oplus opad) \,||\, H((K \oplus ipad) \,||\, M))$

Al fòrum de debat pel món criptogràfic que està obert ara mateix [7] ja es comenta que serà acceptat com a HMAC la formulació següent:

- $HMAC\ (K,M) = H(\ K \,||\, M\ )$

Al reduir una execució de la funció de Hash el temps de procés tan per a generar com per validar el codi d'autenticació de missatge és significativament més ràpid.

Destaquem ara les característiques de possibles estats interns i possibles sortides de la funció SHA-3 en les següents taules:

Estats interns	25 bits	50 bits	100 bits	200 bits	400 bits	800 bits	1600 bits

Taula 21

Sortides	224 bits	256 bits	384 bits	512 bits

Taula 22

En el cas que les dades entrants no s'ajustin a aquests valors el mateix algoritme genera un "padding" per a ajustar-nos a les demandes. Respecte els valors que farem servir ens pronunciarem en el següent apartat però ja avancem que els valors definitius estan pendents al testeig del canal principal i a les característiques d'un enllaç que l'empresa Singular Aircraft encara no té especificat.

2.2.4: Dimensionat:

Definits els tres protocols de criptografia que usarem per a generar el PGP a l'enllaç principal volem centrar el següent apartat a comentar el dimensionat de les dades, claus, codis d'autenticació de missatge que usarem. Això ens servirà per saber si hem de renovar les claus, ja siguin les simètriques o les asimètriques i quin protocol seguir.

Com hem comentat inicialment tenim un missatge M de *b* bits de dades que genera una instrucció a l'avió. Aquest bloc de dades el xifrem amb una clau K_{M1} de sessió de 128 bits responent a l'estàndard AES-128.

Missatge M' = E(M)[K_{M1}]
Dades a enviar xifrades per la clau de sessió.

Taula 23

A l'apartat 2.2.1 hem especificat que la clau AES de 128 bits pot fer fins a 10 iteracions per a xifrar blocs d'igual mida de dades amb la mateixa ampliació de la clau K_{M1}. Per tant una mateixa clau pot

xifrar de forma segura 1280 bits, però aquests no són tots els bits que pot xifrar una mateixa clau de sessió. Una mateixa clau de sessió produirà una col·lisió en mitja cada $2^{N/2}$ xifrats. Així doncs en el nostre cas específic estadísticament tindrem una col·lisió quan haguem xifrat 2^{64} blocs amb AES, que sent els nostres blocs de 128 bits (16 bytes) tindrem una col·lisió cada 2^{68} bytes que això són més de 250 milions de Terabytes ($2{,}95 \times 10^{20}$ bytes). Aquesta xifra és significativament gran.

Respecte les claus RSA que farem servir per xifrar i desxifrar la clau de sessió del PGP, hem comentat en l'apartat 2.2.2 que seran de 2048 bits en mòdul cadascuna. Com a criptografia asimètrica que són i per definició no serà necessari renovar-les durant el transcurs d'una missió.

Aleshores el paquet de dades a autenticar amb l'HMAC ens queda de la següent forma:

Primera vegada que ens comuniquem:

M'	**K'**	**HMAC**
Bloc de dades xifrades.	Clau AES de sessió xifrada amb RSA	Resultat d'introduir a un algoritme irreversible les dades M' || K'.

Recordem que:

- HMAC (K, M) = SHA-3 (K || M)
- M: 10 blocs de dades = 1280 bits.
- K_{M1}: Clau de sessió de 128 bits

Com veiem tenim una suma de 1408 bits que volem introduir a la funció HMAC, tot i així la funció HMAC SHA-3 demana una entrada necessària de 1600 bits i per tant haurem d'usar un padding de 192 bits que el propi algoritme SHA-3 ja introduirà.

D'entre totes les possibles sortides d'aquest algoritme ens la plantegem inicialment de 256 bits però és un punt que un cop s'apliqui aquest projecte s'haurà d'analitzar fent proves del canal.

Si suposem sortida HMAC de 256 bits aleshores tindríem:

	Quantitat de dades	Respecte el total
Dades de pilotatge	1280 bits	0,77
Dades de criptografia	384 bits	0,23

Taula 24

A partir d'aquest punt l'enllaç principal no ha de tornar a enviar la clau de sessió fins a que es consideri oportú renovar-la, en tota la successió de dades enviades es seguirà el patró següent:

- Cada 1472 bits de dades d'instrucció enviats es generarà un HMAC concatenant les dades amb la clau de sessió de 128 bits. La sortida s'enviarà seguidament pel radioenllaç. L'avió al rebre anirà seguint el mateix procediment per anar validant les signatures. A la llarga el rendiment del canal s'estabilitzaria en un 0,85.

Una altra alternativa que es planteja per a millorar l'eficiència és no signar tots els bytes que

s'envien, és cert que fer un HMAC de tots els bytes ens costa un ampla de banda que potser l'empresa al aplicar-ho no es pot permetre. L'alternativa que proposem és:

- Fer un HMAC de 1472 bits + 128 de la clau de sessió i deixar *X* bits de dades que anem desxifrant però que no comprovem la seva autenticitat. Si estipulem un temps màxim entre autenticacions, i coneixem la velocitat de transmissió, podem definir cada quants bits fem una trama d'autenticació. Per exemple imaginem que tenim les següents dades:

  - Velocitat de Transmissió de 4,8Kbps

  - Temps màxim entre autenticacions de 0,5segons.

  Aleshores si en aquest mig segon s'envien 2400 bits que són el següent:

  - 2144 bits de dades

  - 255 bits del HMAC que hem de generar i enviar.

  D'aquests 2144 bits de dades només usarem 1472 bits per a generar l'HMAC i els 672 restants no s'autenticaran.

  Aquest seria un exemple per a augmentar el rendiment del canal (en aquest cas 0,89) que s'ha d'extrapolar a mesures reals quan el disseny de l'enllaç principal per part de l'empresa estigui finalitzat.

2.3: Enllaç d'emergència.

En aquest apartat del Projecte Final de Carrera definirem l'estructura que hem dissenyat per afegir criptografia a l'enllaç d'emergència. Un enllaç només de pujada (estació base - avió) que envia dades de la següent forma:

Trama Canal d'emergència: (110 bits)	
# Canal	**Informació del Canal (10 bits)**
1	Alerons
2	Elevador
3	Deriva
4	Potència motor esquerre
5	Potència motor dret
6	Força frenada fre esquerre
7	Força frenada fre dret
8	Posició Flaps (0º, 15º, 30º, 40º)
9	Fail Safe (ON/OFF), Tren (Up/Down), Bloqueig Cua (Bloquejada/lliure) [3 bits major pes]
10	Motor 1 (ON/OFF), Motor 2 (ON/OFF) [2 bits major pes]
11	Criptografia

Taula 25

On cada trama representa 11 canals de 10 bits cadascun dels que només podem fer ús de l'últim per a tasques de criptografia. Cada trama representa 22.6ms.

Principalment la tasca que se'ns demana és la d'autenticació de missatge i per aquest motiu ens plantegem l'ús de la tècnica MAC.

La proposta que reflectirem a continuació dóna solució a l'autenticació dels missatges transmesos per aquest canal que considerem analògic en el sentit de que la probabilitat d'error és diferent per a cada bit (bits menys significatius de cada canal tenen major probabilitat d'error).

La capacitat de transmissió del canal 11 és aproximadament 40 bits per segon, dels que solament una fracció es considera fiable. Per altra banda, un sol bit erroni, ja sigui en una clau com als bits del missatge com al MAC inutilitzen completament la tasca d'autenticació. Per aquest motiu suposarem que la gestió de clau criptogràfica que s'usarà per generar l'HMAC d'aquest canal serà duta a terme pel canal principal. El canal 11 transmetrà exclusivament informació d'autenticació dels bits enviats pels altres 10 canals.

Com a conseqüència directa del tot just explicat s'ha de destacar que pel canal d'emergència no hi haurà la possibilitat de fer un refresc de claus, així doncs remarquem com a excepcional l'ús del canal d'emergència i la impossibilitat d'assegurar aquest canal durant un període llarg en el temps.

Si tenim en compte tot el que hem explicat trobem necessari protegir el sistema contra un atac de repetició. Un atacant podria enviar rèpliques d'un missatge autèntic captat prèviament i l'avió seria incapaç de marcar-les com a falses. Aquesta situació ens fa remarcar que necessitarem afegir al procés d'autenticació una informació única per a cada enviament.

Entrem doncs a definir la proposta d'autenticació per al canal d'emergència:

2.3.1: Autenticació del canal d'emergència. Autenticació Estratificada.

Per a l'autenticació de les trames que s'enviaran pel canal d'emergència farem ús dels mecanismes MAC i específicament HMAC explicats anteriorment. Si recordem, un HMAC és una funció irreversible que requereix d'entrada el missatge a autenticar i una clau que sigui coneguda tan per l'emissor com el receptor del missatge.

Primerament analitzem el missatge que enviarem. Cada trama consta de 100 bits d'informació a autenticar, dels que com a mínim 20 considerem gens fiables. Com a estimació suposem que només 40 bits dels 100 totals són molt fiables. Cada canal d'informació el dividim de la següent manera:

Bloc alt (fiables)				Bloc mig (semi -fiables)				Gens fiables	
9	8	7	6	5	4	3	2	1	0

Esquema 11

Com ja hem explicat, els sistemes d'autenticació estan dissenyats per a canals d'informació sense errors, com que el canal d'emergència no garanteix la fiabilitat dels bits rebuts degut a que no consta d'un control d'errors podem entendre que no podem aplicar mecanismes d'autenticació a la totalitat dels bits d'una trama ja que això provocaria que fallés constantment.
Per aquest motiu dividim la informació en els tres estrats segons fiabilitat. Aquesta divisió també afecta al canal 11 evidentment ja que un error en un bit de la informació d'autenticació també invalidaria el procés complet.

Com veiem dividim la informació en tres grups: Els 4 bits alts (més significatius), els següents 4 bits i els 2 més baixos. L'autenticació del bloc alt (40 bits per trama) serà la més fiable, mentre que la del bloc mig (40 bits per trama) podria fallar amb certa freqüència sense deure's a cap intrusió. A efectes pràctics descartarem els 2 últims bits d'informació per a qüestions d'autenticació.
És important remarcar que el resultat de l'algoritme d'autenticació dels bits del primer bloc serà enviat pel mateix bloc corresponent del canal 11.

Tot i que podríem prescindir de l'autenticació del bloc mig i només autenticar el bloc alt de dades creiem que al cap i a la fi, fer-ho no consumeix més recursos del canal i només demana un càlcul d'un MAC cada un cert temps que està per estipular. A més a més, fent-ho ens aporta informació de monitorització del canal d'emergència.

Les mides de blocs de sortida dels algoritmes de HMAC oscil·la entre els 224 i els 512 bits, per tant no serà convenient realitzar un procés d'autenticació per cada trama, plantegem agrupar la informació d'un bloc de 10 trames per a realitzar les tasques d'autenticació. D'aquesta manera agruparíem 400 bits d'informació del bloc alt i 400 més del bloc mig.

Aquesta agrupació de trames que plantegem generaria una autenticació cada 226 ms, entre 4 i 5 autenticacions per segon que ja és un nombre que compleix les necessitats de l'enllaç. Tot i així al tractar-se d'una agrupació de trames, aquest fet ens produeix un retard en la generació de la informació d'autenticació de 226 ms al que s'hi ha d'afegir el retard de la transmissió d'aquesta informació que és d'uns altres 226 ms. El funcionament del sistema per a cada un dels blocs de bits a autenticar (alt o mig) l'expressem en el següent diagrama:

Num de trama	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Temps (ms)	0	22	45	67	90	113	158	180	203	226	248	271	293	316	339	361	384	406	429	452	474
Info. Generada (bits)	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40
Info Agregada (bits)	40	80	120	160	200	240	280	320	360	400	40	80	120	160	200	240	280	320	360	400	40
MAC calculat base										40											
MAC calculat avió										40											
MAC transmès											4	4	4	4	4	4	4	4	4	4	4
MAC verificat											4	4	4	4	4	4	4	4	4	40	4

Taula 26

Com veiem la intrusió d'un atacant o l'error per qualsevol altra motiu a la verificació d'un MAC seria detectat com a molt mig 452 ms tard. Tot i així la detecció podria donar-se abans, de fet l'avió pot calcular el MAC quan rep la desena trama i a la onzena ja rep els primers 4 bits a verificar, si aquests bits fossin incorrectes ja es podria detectar un error en l'autenticació. Cert és que durant aquest temps de latència l'adversari podria tindre el control de l'avió, per aquest motiu serà necessari acordar amb l'empresa el màxim temps que es permetrà a l'avió seguir instruccions de missatges que no validen correctament l'autenticació.

En primeres converses amb els responsables de l'empresa Singular Aircraft es va acordar que l'avió deixaria de fer ús del canal d'emergència per a passar a vol automàtic en el cas que aquest canal no aconseguís rebre un bloc de 10 trames autenticades durant un període de 5 segons. Aquests 5 segons són temps real des de l'última autenticació correcta, important destacar que el comptador no s'inicia en la primera autenticació incorrecta sinó que el comptador es reseteja cada vegada que es valida una autenticació correctament.
Aquesta quantificació de 5 segons està pendent de proves de l'enllaç i per tant està subjecte a canvis.

Passem ara a estudiar com generem els HMAC tant pel bloc alt com pel bloc mig de fiabilitat:

Recordem que comptem amb 400 bits d'informació a autenticar amb els que hem de generar un MAC que haurem d'encapsular en 40 bits. Un nivell de seguretat de 40 bits seria considerat baix en altres contexts criptogràfics, sobretot en aquells que els atacs per col·lisió són útils. Tot i així, el tipus d'atac del que s'ha de protegir el sistema és que l'atacant prengui el control de l'avió mitjançant la generació de comandes concretes que aconseguissin autenticar correctament amb una bona associació entre missatge i MAC, això obligaria l'adversari a encertar els 40 bits del MAC que suposa una probabilitat de l'ordre de 10^{-12}.
Un altre possible atac que ja hem comentat del que ens hem de protegir és de l'atac de repetició, en aquest tipus d'atac l'adversari pot intentar enviar rèpliques de missatges autèntics que s'hagin transmès anteriorment. La solució que proposem és afegir a la clau que usarem una part variable. Suposant que existeixen rellotges sincronitzats a l'estació base i a l'avió, proposem afegir un annex a

la clau de generació de l'HMAC en forma de *"time stamp"*.

La generació de l'HMAC proposada quedaria de la següent forma:

- Dades entrants:
  - 400 bits d'instrucció.
  - 80 bits de clau simètrica de l'enllaç d'emergència.
  - 32 bits de *time stamp*.
- Dades sortints:
  - 512 bits de MAC.

Com veiem generem un bloc de 512 bits d'entrada on l'algoritme HMAC ja generarà el padding necessari i rebem a la sortida 512 bits que no s'adeqüen a les necessitats del nostre canal. Per fer-ho haurem de truncar la sortida establint per conveni que només enviarem els primers 40 bits d'aquest MAC. És evident que diversos missatges diferents tindran el mateix MAC de sortida ja que de 2^{400} missatges diferents d'informació passem a 2^{40} possibles MAC's però com hem comentat aquest punt no ens afecta ja que a la següent execució de l'algoritme d'autenticació els bits del *time stamp* seran diferents i per tant una mateixa entrada d'informació generarà un MAC completament diferent amb probabilitat molt alta de l'ordre de $(1 - 10^{-25})$.

La clau i el *time stamp*:

Hem definit el *time stamp* com un camp de 32 bits que complementarà la clau. Aquest camp té la capacitat d'aconseguir generar 2^{32} valors diferents en que cada un d'ells servirà per enviar 400 bits d'informació. Amb un *time stamp* cíclic amb l'ús de tots els seus valors (una volta sencera) podríem enviar al voltant de 214 Gbytes d'informació. Si tenim en compte que en una trama s'envien 100 bits d'informació en 22,6 ms. tenim una velocitat de r = 0,55 Kbytes/segon. Una volta sencera del *time stamp* requeriria al voltant de 12 anys.
Com és evident no farem mai ús de tots els bits del *time stamp* en un vol d'emergència però sí que és cert que pot ser necessari ressetejar aquest camp per qualsevol circumstància, i per tant, podria provocar que es repetissin paquets enviats anteriorment. Per aquest motiu plantegem la necessitat d'afegir un mecanisme de renovació de clau.

La clau de 80 bits serà simètrica i en cap cas la seva renovació serà gestionada per l'enllaç d'emergència, ja que recordem aquest canal no codifica, només autentica.
Un refresc de clau ordenat per l'estació base es transmetrà pel canal principal com a bits d'informació i sota un protocol específic que s'haurà d'afegir quan Singular Aircraft dissenyi i defineixi les dades que s'envien pel canal principal.

2.3.2: Excepcions.

Tal com hem plantejat el nostre disseny criptogràfic qualsevol instrucció que rebi l'avió que s'adapti al protocol de les trames d'aquest enllaç, l'avió l'executarà abans de validar l'autenticació fins a un màxim de 5 segons. Tot i així és obligat generar unes quantes excepcions. Hi ha unes ordres concretes que no podem permetre que s'executin sense ésser abans validades. Diferenciem les excepcions en dos grups:

2.3.2.1: Canvis no habituals:

Aquest grup d'excepcions van referides als canals que duen la informació de la posició dels alerons, la deriva, la potència dels motors, la força de frenada o la posició dels flaps. Aquests valors, durant el transcurs habitual d'un vol, segueixen evolucions molt lineals on no existeixen els canvis sobtats. El que nosaltres proposem és que l'avió no accepti una ordre on es reflecteixi un canvi brusc o no habitual en qualsevol d'aquests canals fins a poder validar que l'ordre ha estat enviada pel pilot. Això generarà un retràs de mig segon a la instrucció però tractant-se del canal d'emergència, canal no habitual de comunicació, creiem que és una bona solució.

Proposem en aquest punt una idea interessant que es podria desenvolupar. La idea és la següent:
Un pilot, per condició humana, sempre tindrà uns patrons de pilotatge diferents a d'altres pilots, es podria generar un patró per a cada pilot després d'hores de pilotatge de les seves conductes i respostes habituals a estímuls externs durant el pilotatge. Aquest patró es podria carregar a l'avió i amb aquesta informació l'avió podria conèixer més el seu pilot i acceptar o rebutjar instruccions amb un cert marge de decisió pròpia.
El mateix podríem dir sobre conductes de les diferents etapes del vol: enlairament, pilotatge estable i aterratge. Tres moments en que els tipus d'instruccions habituals que ha de rebre un avió són molt diferents.

2.3.2.2: Decisions capitals:

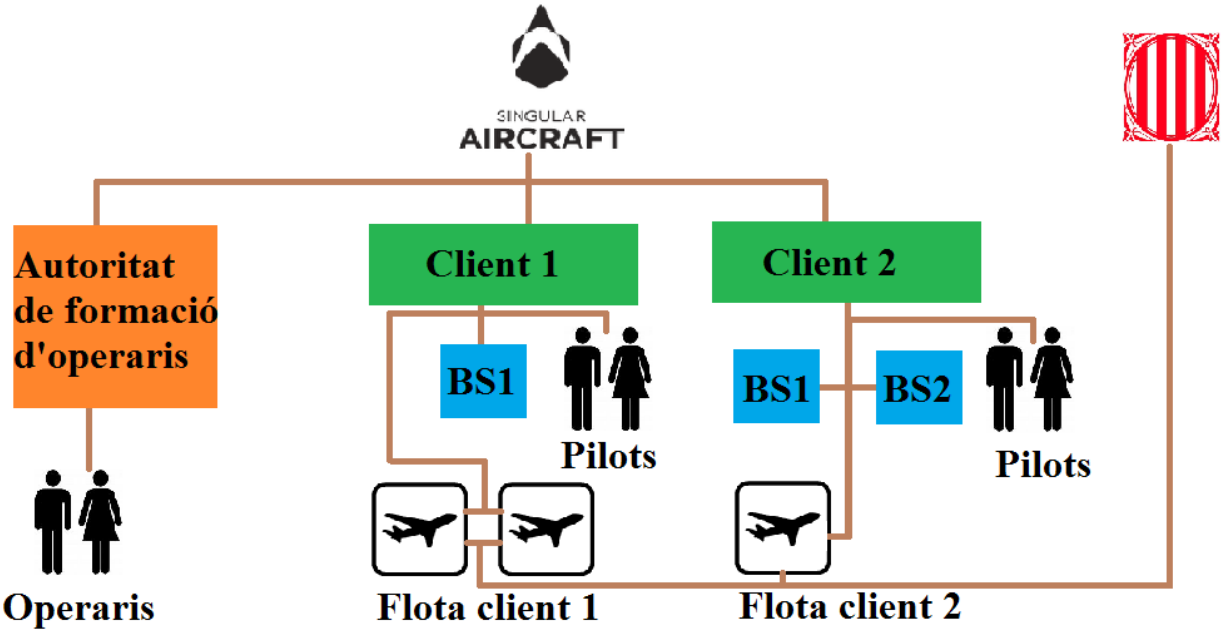
Aquest segon bloc d'excepcions que anomenem decisions capitals fan referència als canals número 9 i 10 de cada trama. Els valors d'aquests canals fan prendre decisions molt importants a l'avió com obrir o tancar un motor o baixar o pujar el tren d'aterratge entre d'altres. Considerem que aquestes instruccions quan s'estigui només amb el canal d'emergència operatiu han de ser estrictament validades i per tant els pilots hauran de tindre en compte la latència de mig segon que tindran les seves ordres que executin aquesta mena de canvis a l'avió.

Capítol 3

PKI, Cadena de confiança i Jerarquització d'Autoritats de Certificació

3.1: Punt de Partida:

L'estudi de jerarquització realitzat en aquest estudi es basa en el següent escenari:



Esquema 12

L'objectiu final d'aquesta metòdica és que les Estacions Base i els avions es puguin comunicar de forma segura.
Per a desenvolupar aquest estudi hem fet les suposicions següents:

- Singular Aircraft només ha d'intervindre en el protocol de seguretat fins al moment de la venda dels avions i les estacions base. A partir d'aquest punt la gestió de la seguretat de les missions va a càrrec del Client i ha de ser "transparent" a Singular Aircraft.
- Suposem que el client no té coneixements de seguretat al Client, per tant Singular ha d'oferir

un servei de gestió de la seguretat a nivell d'usuari. Aquest servei s'ofereix conjuntament amb la venta de l'avió i un cop configurat és client qui en farà ús.
- Tots els avions i estacions base segueixen el mateix procés de producció i es diferencien mitjançant un número de serie únic.
- En el cas que existeixi la revenda d'avions i d'estacions bases entre diferents clients, Singular Aircraft haurà de participar en l'intercanvi ja que haurà d'expedir noves claus que valguin com a certificats de possessió.

Definim ara les tasques a desenvolupar de cada element del sistema. Destacarem essencialment les tasques de seguretat, certificació o validació.

- Singular Aircraft:
  - Produeix i ven avions i estacions bases a diversos Clients. A més a més disposa a cada client d'un "Centre de Control" per a que aquest pugui programar missions segures amb facilitat.
  - A partir de la venta del producte, i més enllà de les revisions i reparacions de les bases i els avions, el proveïdor no intervé durant la gestió de les missions.
- Govern
  - El Govern només es comunicarà amb l'avió en el cas de la necessitat d'una intervenció deguda al mal ús de la flota o qualsevol amenaça seria de la seguretat (per exemple invasió d'espai aeri). Per aquest motiu ha d'existir una porta oculta que només serà coneguda pel Govern i per Singular Aircraft (qui la genera). En cap cas aquesta porta oculta serà entregada al Client.
- Client:
  - Fent qualsevol combinació amb els seus avions i estacions bases pot generar una missió segura.
  - Mitjançant el "Centre de Control" i les seves pròpies claus, Client genera totes les identificacions i claus necessàries per a les missions.
  - Reparteix a cada Pilot i/o operador involucrat en la missió el seu identificador i/o les claus necessàries.
- Operari:
  - L'operari és la persona que fa la posta a punt dels avions. Des del punt de vista de seguretat, l'operari és qui traspassa les claus necessàries a avions i estacions bases per a que es pugui generar la comunicació de forma segura.
- Autoritat de formació d'operaris:
  - Aquesta entitat forma els operaris en totes les tasques necessàries per a fer la posta a punt de la missió. Entre aquestes tasques hi ha els protocols de seguretat (a alt nivell).
  - Un cop els operaris han superat aquesta formació l'Autoritat de Formació els hi ha de certificar la seva vàlua.
- Pilot:
  - El pilot s'ha d'identificar davant l'estació base com a persona autoritzada per a fer-la servir.
- Estació Base (BS):
  - L'estació base ha de validar que tant l'operari com el pilot amb el que es comunica estan autoritzats pel Client. A més a més ha de comprovar també que ella, com a estació base, és propietat del Client en qüestió.

- L'estació base usarà les claus que li transmet l'operari per a iniciar la comunicació amb l'avió amb que està emparellada.
- L'estació base ha de poder gestionar la renovació de claus durant la missió. La renovació de claus de forma periòdica és bàsica per a millorar la fiabilitat de la seguretat de l'enllaç. Per a fer-ho pot generar ella mateixa les noves claus i transmetre-les o seguir altres estratègies.

- Avió:
  - L'avió ha de validar que l'operari amb el que es comunica està autoritzat pel Client. A més a més ha de comprovar també que ell, com a avió, és propietat del Client en qüestió.
  - L'avió usarà les claus que li transmet l'operari per a iniciar la comunicació amb l'estació base amb que està emparellat.
  - L'avió ha d'acceptar i interpretar correctament les ordres de l'estació base en el transcurs de la comunicació. Entre aquestes ordres hi haurà la de renovació de claus.

3.2: Proposta jeràrquica de certificació:

La proposta que fem per a introduir seguretat a les comunicacions es basa en un sistema jeràrquic de certificats digitals que implementa un esquema de confiança entre els diferents agents involucrats. L'escenari de certificats digitals proposat basa el seu funcionament en el sistema RSA de parelles de claus pública/secreta. Ens plantegem una sèrie de preguntes:

- *Quines característiques té un certificat digital?*
- *Qui i quan es generen els certificats?*
- *Necessitem parelles de claus en cada entitat del sistema?*

En la següent anàlisi donem resposta a aquestes preguntes, i tenint en compte les tasques de cada element del sistema definides anteriorment, construïm un sistema de confiances basat en certificats digitals i claus asimètriques que ens permeten complir amb l'objectiu de proporcionar seguretat a un sistema multiagent.

<u>Certificats digitals:</u>

Un certificat digital és un document digital mitjançant el qual un tercer confiable garanteix la vinculació entre la identitat d'un subjecte i la seva clau pública. Aquest tercer confiable s'anomena autoritat certificadora (CA). Per a desenvolupar un sistema basat en certificats digitals totes les entitats han de compartir directe o indirectament una mateixa entitat de confiança (CA-Root). A part poden existir autoritats certificadores de segon ordre que no abasteixin tot el sistema, només una subxarxa d'aquest.
Per a expedir un certificat, l'autoritat certificadora usa la seva clau secreta i per tant tots els elements de l'escenari hauran de conèixer la clau pública de la CA-Root per a validar certificats d'altres.

En l'escenari, Singular Aircraft serà l'Autoritat Certificadora comuna, o CA-Root, per tant tots els elements del sistema (Clients, avions, bases...) hauran de fer-li confiança. Per a fer-ho i per a poder validar certificats identificadors d'altres elements, aquests hauran de conèixer necessàriament la clau pública de Singular Aircraft.

És habitual que en els escenaris PKI de certificació digital jeràrquica apareguin Autoritats Certificadores de segon ordre. En el nostre cas i havent analitzat les tasques que han de dur a terme totes les entitats del sistema creiem necessari dotar d'aquestes prestacions a cadascuna de les entitats Client.
Al no poder obligar al Client a ser expert en criptografia serà necessari desenvolupar un sistema simple i metòdic que, tot i complir tots els estàndards de seguretat, sigui senzill procedimentalment pel Client. L'anomenarem Centre de Control.

Així doncs comptem amb dues autoritats de confiança de diferent rang jeràrquic però amb tasques molt diferents:

- Mitjançant els certificats expedits per Singular Aircraft la resta d'elements es podran identificar.
- Mitjançant els certificats expedits pel Client els operadors i els avions podran demostrar a avions i estacions bases estar autoritzats per a desenvolupar les seves tasques i que es pugui dur a terme la missió.

Cal completar la definició anterior amb els temps aproximats de vigència de cada tipus de certificats:

- Els certificats expedits per Singular Aircraft han de ser vigents per una llarga duració de temps.
- Els certificats expedits pel Client només han de ser vigent durant el transcurs d'una missió i després han de ser revocats.

La revocació de Certificats es pot dur a terme de dues formes diferents:

- Amb una llista de Certificats revocats (CRL) actualitzada i emmagatzemada freqüentment per totes les entitats que hagin de validar identificacions, majoritàriament avions i estacions bases. Abans d'acceptar qualsevol certificat l'agent validador comprovarà que el número de serie del certificat no estigui inclòs en la llista de certificats revocats CRL.
- Temporalment. Aquesta tècnica de revocació de certificats es basa en que en el moment de la generació del certificat es podrà especificar el seu temps de vigència. Un cop aquest temps s'hagi exhaurit el certificat deixarà de tindre validesa i cap agent del nostre sistema el tractarà com a un certificat vàlid.

En el cas dels certificats expedits per Client i que son vàlids només per una missió la tècnica de revocació més adequada serà la temporal. Gestionar una llista de certificats revocats amb les condicions inicials del sistema (no garantida cap connexió a xarxa per part de l'avió o l'estació base) fa no fiable l'ús de la CRL.
En canvi respecte els certificats emesos per Singular Aircraft, certificats de llarga durada, l'ús de la CRL pot ser positiu. Aquests tipus de certificats, per exemple els de relació de pertinença entre Client i Avio, hauran de ser revocables en el cas de que el Client vengui l'avió a un tercer. En aquest cas Singular haurà de generar nous certificats de pertinença per el comprador de l'avió i revocar el certificat del propietari inicial. Per a fer-ho només seria necessari introduir el número de serie del certificat a la CRL i actualitzar-la a tots els agents abans de que es realitzi qualsevol altra missió amb aquells avions o estacions base.

Claus:

Una altra limitació amb la que ens trobem en aquest escenari és que la seguretat de les bases i els avions quan són a terra sense realitzar cap missió no està garantida. Per tant, al ser accessibles, caldrà que les claus privades usades durant les missions siguin destruïdes i no emmagatzemades. Aquesta circumstància obliga als Clients a inserir parelles de claus pública-privades durant la posta a punt de l'avió i l'estació base. Aquesta tasca l'haurà de desenvolupar o bé l'Operari o bé el Pilot. A partir d'aquest punt del document assumim que aquesta tasca la desenvoluparà l'operador ja que és la persona que haurà estat formada per Singular Aircraft, entitat del sistema de major rang.

Aquestes parelles de claus pública-privades serien temporals i serien destruïdes a l'acabar la missió per a tornar a deixar les estacions bases i els avions sense cap informació sensible emmagatzemada.

Resum Claus, números de sèrie i certificats:

Totes les claus necessàries per a formar l'estructura jeràrquica del sistema serien les següents:

1. Singular Aircraft:
    - Claus pública i privada. K_{PSA} K_{SSA}
2. Client:
    - Claus Pública i privada. K_{PCln} K_{SCln}
3. Avió:
    - Claus pública i privada temporals (1 missió). K_{PAn} K_{SAn}
4. Estació Base:
    - Claus pública i privada temporals (1 missió). K_{PBSn} K_{SBSn}

A més a més l'estructura jeràrquica s'ajudarà dels números de sèrie per a realitzar tasques d'identificació i pertinença. Aquests números són personals i únics per a cada element, no han de mantenir-se ocults i seran necessaris en els següents elements:

1. Avió:
    - *A000001, A000002...*
2. Estació Base:
    - *BS00001, BS00002...*

Fixem-nos que excloem els Clients i les Autoritats de formació. Aquests agents constaran d'un identificador (probablement no numèric) que consta en el certificat de la seva clau pública corresponent. En el cas dels Operaris, al ser persones físiques, podem usar el seu DNI com a identificador.

Amb l'ajuda d'aquests números de sèrie es crearan els següents certificats d'identificació o autorització:

Certificats de les claus públiques expedits per Singular Aircraft (No revocables a curt termini):

- $Cert(K_{PSA})[K_{SSA}]$ *//Certificat autosignat de la clau pública de Singular Aircraft.*
- $Cert(K_{PCln})[K_{SSA}]$ *//Certificat emès i signat per Singular de la clau pública de Client.*

Certificats de pertinença expedits per Singular Aircraft (No revocables a curt termini):

- $Cert_{PER}$*("L'avió A000001 pertany al Client n")*$[K_{S\ SA}]$
- $Cert_{PER}$*("L'estació base BS00001 pertany al Client n")*$[K_{S\ SA}]$

Certificat d'identitat d'Operador expedit per Singular Aircraft (No revocable a curt termini):

- $Cert_{ID}$*("L'operador DNI_OPERADOR de l'autoritat de formació AF_id")*$[K_{S\ SA}]$

Certificats de les claus públiques que s'usaran en el desenvolupament d'una missió expedits pel Client:

- $Cert(K_{P\ BS\ n})[K_{S\ Cl\ n}]$
- $Cert(K_{P\ A\ n})[K_{S\ Cl\ n}]$

A més a més, el Client firmarà també el que n'anomenarem pla de vol. El pla de vol podrà incloure diferents aspectes com per exemple la relació entre les diferents estacions bases amb els avions que els hi toqui pilotar. El pla de vol inclourà unes ordres bàsiques a nivell tècnic de com ha d'actuar l'avió en el cas que el senyal dels diferents canals es perdi o sigui interferida. Un exemple bàsic podria ser el següent:

- $Cert_{Pla\_de\_vol}$*("L'avió A000001 es relacionarà amb l'estació base BS00001 inicialment; A part es podrà relacionar amb altres bases si és necessari (BS00002,BS00003...))*$[K_{S\ Cl\ n}]$

Cal comentar també que tot certificat inclou molta més informació de la que s'indica aquí, per exemple els camps d'un certificat informen, entre d'altres, dels aspectes següents:

- Id Emisor i Propietari.
- Clau Pública de l'entitat validada.
- Clau de verificació.
- Tipus d'algoritme de criptografia usat.

Aquests punts seran abordats més endavant d'aquest mateix capítol.

3.3: Metòdica en una missió:

Imaginem un escenari en que un Client vol realitzar una missió 1x1 (nº avions, nº bases). Tant l'avió com l'estació base han estat comprats a Singular Aircraft. El Client ha de poder realitzar aquesta missió de forma segura sense la necessitat de contactar amb Singular Aircraft.
El Client contacta amb un operador i un pilot per a que duguin a terme aquesta operació.
Considerarem per aquest exemple que la missió es podrà dur a terme amb seguretat quan aconseguim una primera connexió segura entre l'avió i l'estació base.

Escenari inicial:

La situació inicial de cada component d'aquest escenari és la següent:

- Client:
  El Client disposa de la seva clau secreta i el certificat de la seva clau pública expedit per Singular Aircraft. A més a més, el Client disposa també de dos certificats de pertinença, un respecte el número de sèrie de l'avió i l'altre respecte el de l'estació base. A més a més, també disposa d'infraestructura per a generar certificats i claus mitjançant la seva clau secreta que l'anomenem Centre de Control. A més a més el Client també disposa de la llista d'identificadors dels possibles pilots i operaris.
    - Claus pública-privada: $K_{P\,Cl\,1}$ $K_{S\,Cl\,1}$
    - Cert. Clau Pública: $Cert(K_{P\,Cl\,1})[K_{S\,SA}]$
    - Cert. pertinença: $Cert_{PER}$*("L'avió A000001 pertany al Client 1")*$[K_{S\,SA}]$ *i* $Cert_{PER}$*("L'estació base BS00001 pertany al Client 1")*$[K_{S\,SA}]$
    - Centre de control.

- Operador:
  L'operador disposa del seu certificat d'identitat d'Operador. Espera ordres del Client.
    - $Cert_{ID}$*("operador OP00001 de l'autoritat de formació AF00001")*$[K_{S\,SA}]$

- Pilot:
  El Pilot no ha de disposar de res més que dels coneixements per a pilotar l'avió.

- Estació Base:
  L'estació base comprada pel Client no té cap informació respecte certificats però consta d'un número de sèrie fix generat en la seva producció i la clau pública de Singular Aircraft en hardware.
    - Número de sèrie: *BS00001*
    - Clau Pública de Singular Aircraft: $K_{P\,SA}$

- Avió:
  L'avió, igual que l'estació base, no té coneixements de certificats però té inclosos en producció un número de sèrie únic i fix i la clau pública de Singular Aircraft en hardware.
    - Número de sèrie: *A000001*
    - Clau Pública de Singular Aircraft: $K_{P\,SA}$

Primer pas:

- Client → Operador:
  Mitjançant el Centre de Control el Client genera les claus publica-privades per a la comunicació entre l'estació base i l'avió, de les públiques en genera també el certificat firmant-les amb la seva clau secreta.
  Vinculant els números de sèrie en genera els certificats temporals "plans de vol" de base-avió i avió-base que els entrega a l'operador.
  El Client entrega a pilot el certificat de la seva clau pública.
  A més a més també fa entrega de les claus secretes per a l'avió i per l'estació base i dels certificats de pertinença d'ambdós respecte el Client generats per Singular Aircraft.
    - $Cert(K_{P\,A\,1})[K_{S\,Cl\,1}]$
    - $Cert(K_{P\,BS\,1})[K_{S\,Cl\,1}]$
    - $Cert_{Pla\_de\_vol}$*("L'avió A000001 es relacionarà amb l'estació base BS00001 inicialment)*$[K_{S\,Cl\,1}]$
    - $K_{S\,A\,1}$
    - $K_{S\,BS\,1}$
    - $Cert(K_{P\,Cl\,1})[K_{S\,SA}]$
    - $Cert_{PER}$*("L'avió A000001 pertany al Client 1")*$[K_{S\,SA}]$
    - $Cert_{PER}$*("L'estació base BS00001 pertany al Client 1")*$[K_{S\,SA}]$

  Volem remarcar en aquest punt el doble significat que estem fent ús de la paraula certificat. En un entorn d'estructura jeràrquica de confiances amb tecnologia PKI el certificat s'usa per a validar una clau pública, en el nostre sistema aquests serien els casos dels certificats de les claus públiques de Client, avions i estacions bases. Tot i així nosaltres usem la nomenclatura *Cert(A)[B]* per a referir-nos als certificats de pertinença o al pla de vol tot i no continguin una validació de clau pública. Aquests són en definitiva signatures digitals dels missatges transmesos.
  Per a que s'entengui la diferència, un certificat digital fent ús de la tècnica PKI només són aquell que al validar-los pots determinar una correspondència entre una identitat i una clau pública. Així doncs, tot i fer ús de signatura digital per assegurar la procedència i la integritat de les dades els plans de vol i els certificats de pertinença no són certificats digitals d'una estructura PKI.

Segon Pas:

L'operador, el Pilot, l'estació base i l'avió es traslladen al punt inicial de la missió. A partir d'aquest punt el Client ja no participa en el procés. En aquest segon pas establim la primera connexió segura entre estació base i avió. En la següent explicació es mostra per passos la introducció dels certificats a l'avió. Aquesta situació és només acadèmica per a poder anar justificant l'ús d'aquests, en una situació real l'Operador introduiria totes les dades i seguidament l'Avió en faria la comprovació.

- Operador → Avió:
  Inicialment l'operador introdueix el Certificat d'identitat que el valida com a operari format per una autoritat de formació concreta validada per Singular Aircraft.
    - $Cert_{ID}$*("operador DNI_Operador de l'autoritat de formació AF_id")*$[K_{S\,SA}]$

Seguidament introduïm el certificat de la clau pública del Client que relaciona el Id de Client amb la seva clau pública. Verifiquem que el Client existeix.

- $Cert(K_{P\,Cl\,1})[K_{S\,SA}]$

A continuació el client introdueix a l'avió el pla de vol que en aquest cas verifica les següents dues relacions:

- El client ha designat l'operador
- L'avió ha d'iniciar comunicació amb una estació base que també sigui propietat de Client.

Per a fer-ho doncs el nostre pla de vol serà a grans trets el següent document.

- $Cert_{Pla\_de\_vol}$*("L'operador DNI_Operador ha estat designat per Cilent_i per a dur a terme la posta a punt de L'avió A000001 que es relacionarà amb l'estació base BS00001 inicialment")*$[K_{S\,Cl\,1}]$

A continuació l'operador introdueix el certificat de pertinença al client generat per Singular Aircraft. Aquest certificat vincula el número de sèrie de l'avió amb l'ID del Client de qui és propietat.

$Cert_{PER}$*("L'avió A000001 pertany al Client 1")*$[K_{S\,SA}]$

Un cop l'avió ja ha validat mitjançant les claus públiques de singular i de client les instruccions anteriors l'operador hi introdueix el certificat de la clau pública de l'estació base amb la que començarà comunicació i la parella de claus asimètriques que li corresponen a l'avió.

- $K_{S\,A\,1}$
- $Cert(K_{P\,A\,1})\ [K_{S\,Cl\,1}]$
- $Cert(K_{P\,BS\,1})[K_{S\,Cl\,1}]$

Cal remarcar que tot l'anterior és un esquema i que sobretot en el pla de vol es pot introduir molta més informació si fos necessari. En un cas real caldrà fer mencions a instruccions de vol, informacions de sensors, entre moltes altres dades que depenen de departaments diferents d'aquesta empresa. En definitiva tot allò que sigui necessari comunicar a l'avió o a l'estació base abans d'iniciar un vol.

- Avió:

A mesura que l'avió va rebent aquesta informació la va validant i assumint. Aquesta operació la pot realitzar gràcies a que tot i no confiar d'entrada amb l'operador que l'està parlant ambdós comparteixen una autoritat de confiança que és Singular Aircraft. Per aquest motiu és essencial que l'avió tingui la clau pública de Singular Aircraft emmagatzemada en tot moment.

- $Cert_{ID}$*("operador DNI_Operador de l'autoritat de formació AF_id")*$[K_{S\,SA}]$

Inicialment l'operador introdueix el certificat d'identitat que ell valida amb la seva clau pública de Singular Aircraft. Amb aquest missatge l'avió pot identificar l'operador que li parla i a més pot conèixer en quina autoritat de formació va rebre els seus fonaments d'operador.

- $Cert(K_{P\,Cl\,1})[K_{S\,SA}]$

Quan l'avió valida aquest certificat amb la clau pública de Singular Aircraft en els seus camps interns relaciona l'ID del Client *"Client 1"* amb la clau pública d'aquest.

- $Cert_{Pla\_de\_vol}$*("L'operador DNI_Operador ha estat designat per Cilent_i per a dur a terme la posta a punt de L'avió A000001 que es relacionarà amb l'estació base BS00001 inicialment")*$[K_{S\,Cl\,1}]$

En rebre el Pla i validar que aquesta informació procedeix d'un client certificat per Singular Aircraft, l'avió comprova que l'operari ha estat designat per Client i a més a més descobreix amb quina estació base haurà d'iniciar comunicació.

- $Cert_{PER}$*("L'avió A000001 pertany al Client 1")*$[K_{S\,SA}]$

A continuació, mitjançant la clau pública de Singular Aircraft l'avió reconeix el seu número de sèrie en el certificat i comprèn que forma part de la flota del Client 1 que té un cert ID *"Client 1"*. El Client és propietari de l'avió.

- $Cert(K_{P\,BS\,1})[K_{S\,Cl\,1}]$
- $Cert(K_{P\,A\,1})\ [K_{S\,Cl\,1}]$
- $K_{S\,A\,1}$

Finalment l'avió rep les claus generades per Client, una parella de claus asimètriques per a ell, i la clau pública de l'estació base amb la que haurà d'iniciar la comunicació.

Simètricament al que s'ha vist per a l'avió l'operador ho realitza amb l'estació base. En aquest segon cas haurem de contemplar la possibilitat que el pilot s'hagi d'autenticar o identificar amb l'estació base.
Les possibilitats en aquest punt són molt àmplies. Es podria desenvolupar mitjançant un codi pin, una targeta de coordenades o per lectures biomètriques de l'empremta digital entre moltes d'altres tècniques.

Tercer pas:

Amb els dos passos anteriors ja hem establert la comunicació entre l'avió i la base. Els dos coneixen la clau pública de l'altra entitat i la seva pròpia clau secreta.
En aquest pas el Pilot pot començar a introduir el pla de vol a l'avió des del panell de control situat a l'estació base sense cap risc.

Com es gestionaran cada un dels enllaços entre avió i estació base són temes que es tracten en capítols diferents ja que no influeixen en el model de jerarquització.

Quart pas:

Al finalitzar la missió la confiança generada mitjançant els diferents certificats expirarà degut a la pèrdua de validesa del pla de vol i de les claus de missió. Amb l'absència d'aquests certificats serà impossible iniciar una missió i per tant l'avió i l'estació base quedaran inutilitzables fins a que arribi un nou permís expedit pel client de qui són propietat.

3.4: Autoritats de Certificació; Muntatge de l'estructura i generació de claus i certificats:

3.4.1: Punt de Partida i anàlisi de les necessitats:

Anteriorment, en altres apartats d'aquest mateix projecte, s'ha definit la necessitat de que el client pogués programar les seves pròpies missions. Per a fer-ho vam definir que seria necessari que Client tingués la capacitat de generar Claus temporals per el desenvolupament d'aquestes. Aquestes claus han d'estar signades pel propi Client. Per a que aquest Client pugui realitzar aquestes tasques serà necessari que se l'acrediti com a CA en la nostra estructura jeràrquica de PKI (Infraestructura de claus públiques).

A més a més de la certificació dels clients com a autoritats de certificació serà necessari generar tot el material criptogràfic necessari per el transcurs d'una missió.

Seguint la mateixa estructura del punt anterior, les dades necessàries per el desenvolupament de la missió seran:

- $Cert(K_{P\,A\,1})[K_{S\,Cl\,1}]$
- $Cert(K_{P\,BS\,1})[K_{S\,Cl\,1}]$
- $Cert_{Pla\_de\_vol}$*("L'operador DNI_Operador ha estat designat per Cilent_id per a dur a terme la posta a punt de L'avió A000001 que es relacionarà amb l'estació base BS00001 inicialment")*$[K_{S\,Cl\,1}]$
- $K_{S\,A\,1}$
- $K_{S\,BS\,1}$
- $Cert(K_{P\,Cl\,1})[K_{S\,SA}]$
- $Cert_{PER}$*("L'avió A000001 pertany al Client 1")*$[K_{S\,SA}]$
- $Cert_{PER}$*("L'estació base BS00001 pertany al Client 1")*$[K_{S\,SA}]$

Que com podem veure, efectivament són tot dades certificades o signades per Singular Aircraft o per Client.

Com a conseqüència d'èsser una CA, el Client haurà de disposar d'un software i un hardware específic que li permeti fer les següents tasques:

- Emmagatzemar la clau privada de Client.
- Generar parelles de claus temporals pel desenvolupament de missions específiques.
- Certificar les claus generades amb les claus de Client.
- Introduir les claus generades a les SmartCard

A més a més totes aquestes tasques les ha de poder desenvolupar d'una forma protocolaria, senzilla i entenedora. Al Client se li ha d'ofertar un full de ruta de com activar els seus avions pas per pas. És estratègicament contraproduent per a l'empresa obligar al Client a obtenir coneixements de seguretat.

3.4.2: Propostes de solució:

Al abraçar la jerarquització i la formació de les autoritats de certificació inicialment ens trobem amb la disjuntiva de quina mena de mesures es poden prendre per a pal·liar les necessitats exposades anteriorment. En el sector de la seguretat hi ha diverses opcions per a generar i emmagatzemar claus, en format hardware i en format software així com hem explicat a la introducció.

A continuació en fem un curt esment per a posar-nos en situació.

En format Hardware:

La manera més habitual d'emmagatzemar claus estàticament (no s'han de desplaçar) és usar un disc dur dels tipus HSM (Hardware Security Modul).
Aquests tipus de disc durs són mòduls llestos per a desenvolupar tasques criptogràfiques com:

- Emmagatzematge segur de les claus
- Xifrat
- Signatura
- Generació de claus

Els avantatges d'usar aquest tipus d'eina són:

- Velocitat de procés ja que el seu processador està dissenyat exclusivament per aquesta tasca.
- Seguretat física. No es poden modificar els algoritmes i les claus que es vulguin protegir no trobaran un millor entorn.
- Instal·lació. Al ser un mòdul robust i tancat és molt senzill d'instal·lar en qualsevol espai que es necessiti.

Aquest hardware del que ens disposa el mercat compleix totes les necessitats que hem definit a l'anàlisi. Tot i així té més d'un inconvenient en l'escenari que ens trobem nosaltres.

En primer lloc, la robustesa, que per a protegir les claus és tan avantatjosa, ens pot provocar problemes en la generació de claus i certificats digitals de la forma específica que els volem nosaltres. El nostre escenari no segueix fil per randa els estàndards en la generació de certificats i per tant aquest mòdul ens pot deixar la feina a mitges. En la mateixa línia i pel mateix motiu cal destacar que aquest hardware no suporta gens bé les actualitzacions.
A més a més del problema de la robustesa del hardware cal destacar que és una solució més cara que la de software i que un mal manteniment per part de l'usuari pot fer que es deteriori amb rapidesa.

En format software:

Fent ús de software criptogràfic aconseguim una major flexiblitat en els algoritmes a usar ja que serà tasca del programador decidir el tractament de les claus i el xifratge o les signatures d'aquestes.
El software més estandarditzat entre els programadors de criptografia es l'*Open SSL* un estàndard que permet la generació de claus oferint un ampli ventall d'instruccions.

L'Open SSL es desenvolupa tan en el sistema operatiu Ubuntu com en Windows. En el primer dels sistemes operatius s'executa mitjançant el Terminal i en Windows l'execució és en el cmd (centre de comàndes).

Aquesta solució aporta flexibilitat, portabilitat i facilitat en l'actualització dels algoritmes. Aquestes propietats són essencials pel nostre escenari ja que els Clients no estaran centralitzats. Tot i així haurem de buscar una bona solució per a emmagatzemar les claus de cadascun dels nostres clients. Les claus pròpies i úniques de cada client no són temporals i és a través d'aquestes que es fonamenta tot el nostre sistema criptogràfic.

3.4.3: Generació d'una CA amb l'OpenSSL:

En el següent punt exposarem els passos a seguir per a certificar Singular Aircraft com a una autoritat de certificació dins una xarxa privada d'infraestructura de clau pública.

El concepte 'privada' en una PKI i en una autoritat de certificació és important en el sentit que no serà reconeguda per a tothom. Al contrari, només aquells usuaris que en coneguin la identitat en confiaran, entenent-se com a identitat una copia fiable de la clau pública. La generació d'una xarxa privada d'autoritats de certificació i usuaris és la millor opció per el nostre escenari ja que no farem ús de la xarxa Internet. Nosaltres mateixos generem les relacions de confiança entre les entitats que les necessitin.

Posant com a punt de partida un ordinador amb suficient capacitat de càlcul i memòria RAM executat amb sistema operatiu Ubuntu definim els següents punts:

3.4.3.1: Estructura de fitxers:

Per a mantindre un ordre en la gestió de certificats és recomanable seguir una estructura de directoris que permeti tindre classificats els diferents elements inherents al procés de gestió de certificació. L'estructura que seguirem en el nostre cas serà la següent:

```
Nom_Client/
Nom_Client/certs              # Directori contenidor de certificats
Nom_Client/private            # Contenidor de cakey.pem (clau privada)
Nom_Client/newcerts           # Contenidor dels nous certificats emesos.
Nom_Client/csr                # Contenidor dels arxius de demandes de certificació
Nom_Client/crl                # Contenidor dels certificats revocats
Nom_Client/OpenSSL.cnf        # Configuració per defecte dels certificats
Nom_Client/index.txt          # Fitxer número de certificats signats
Nom_Client/serial             # Contenidor de SN dels certificats expedits.
Nom_Client/crlnumber          # Contenidor de SN dels certificats revocats
```

Essent Nom_Client en aquest primer cas Singular_Aircraft. Per a especificar-ho de forma genèrica usarem “Client”.

Per a facilitar la creació de certificats a Client serà convenient fer ús de la possibilitat que ofereix OpenSSL de brindar algunes configuracions per defecte. Aquestes s'inclouen a l'arxiu OpenSSL.cnf que en cas de la versió per Ubuntu el podem trobar a /etc/ssl. El procediment convenient és generar-ne una copia al directori Nom_Client/ de la nostra estructura de directoris i modificar la copia.

Aquesta estructura per defecte dels certificats ens permetrà que Client no hagi d'introduir totes les dades que es requereixen cada vegada que vulgui generar un certificat. Al cap i a la fi, la majoria de dades que s'han d'introduir poden ser fixes en la majoria de les nostres autoritats de certificació.
Els paràmetres que ens demanarà introduïr OpenSSL en la generació d'un certificat digital són:

- Country Name (2 letter code) [AU]: *ES*
- State or Province Name (full name) [Some-State]: *BCN*
- Locality Name (e.g., city): *BCN*
- Organization Name (eg, company): *Singular Aircraft*
- Organizational Unit Name (eg, section): *IT Service*
- Common Name (eg, YOUR name): *Ferran_Alvarós*
- Email Address: *...@singularaircraft.com*

3.4.3.2: Inicialització d'una CA:

Per a inicialitzar una CA cal generar un parell de claus (pública i privada) i crear un certificat digital de clau pública que pot estar desenvolupat de dues maneres diferents:

- Certificat autosignat.
  En el cas de les autoritats certificadores de primer ordre de la jerarquia, al no tindre cap autoritat superior, es genera un certificat de la seva clau pública que s'autosigna la mateixa CA.
- Certificat signat per una CA de jerarquia superior:
  En el cas de les autoritas certificadores que no són de primer ordre es genera el certificat digital signat per qualsevol CA de jerarquia superior.

En el nostre cas Singular Aircraft és la CA-Root i tots els altres clients els hem d'acreditar com a Autoritat de Certificació signant les seves claus públiques amb la privada de Singular Aircraft.

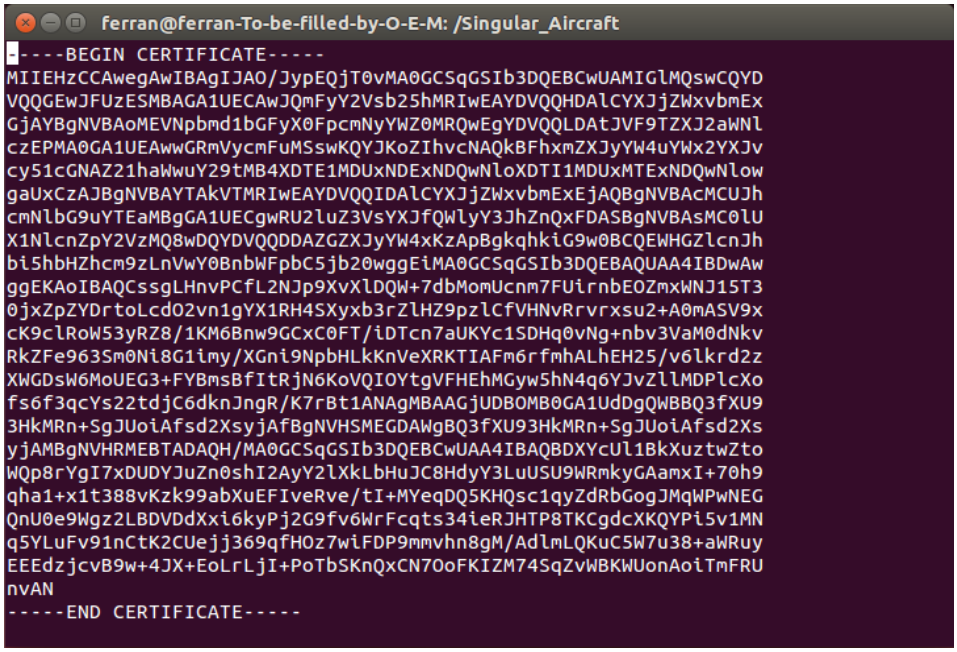
En el cas de Singular Aircraft la instrucció necessària en un entorn Ubuntu i amb la paqueteria OpenSSL instal·lada correctament serà:

- *openssl req -new -x509 -days 3650 –keyout private/cakey.pem -out cacert.pen -config openssl.cnf*

Aquesta instrucció generarà dos fitxers:
- **cacert.pem**: Certificat arrel, el qual pot ser publicat i haurà de ser extès per a totes les entitats de la infraestructura de clau pública que estem dissenyant.
- **cakey.pem**: L'arxiu que conté la clau privada de la CA-Root. S'usarà per a signar els certificats digitals que expedeixi aquesta CA. S'ha de guardar de forma segura.

De la instrucció cal destacar la crida a la configuració per defecte *openssl.cnf* situada al final de la comanda i l'especificació dels dies de vigència d'aquest certificats en *-days 3650* .

En la següent imatge es pot observar el certificat digital generat:

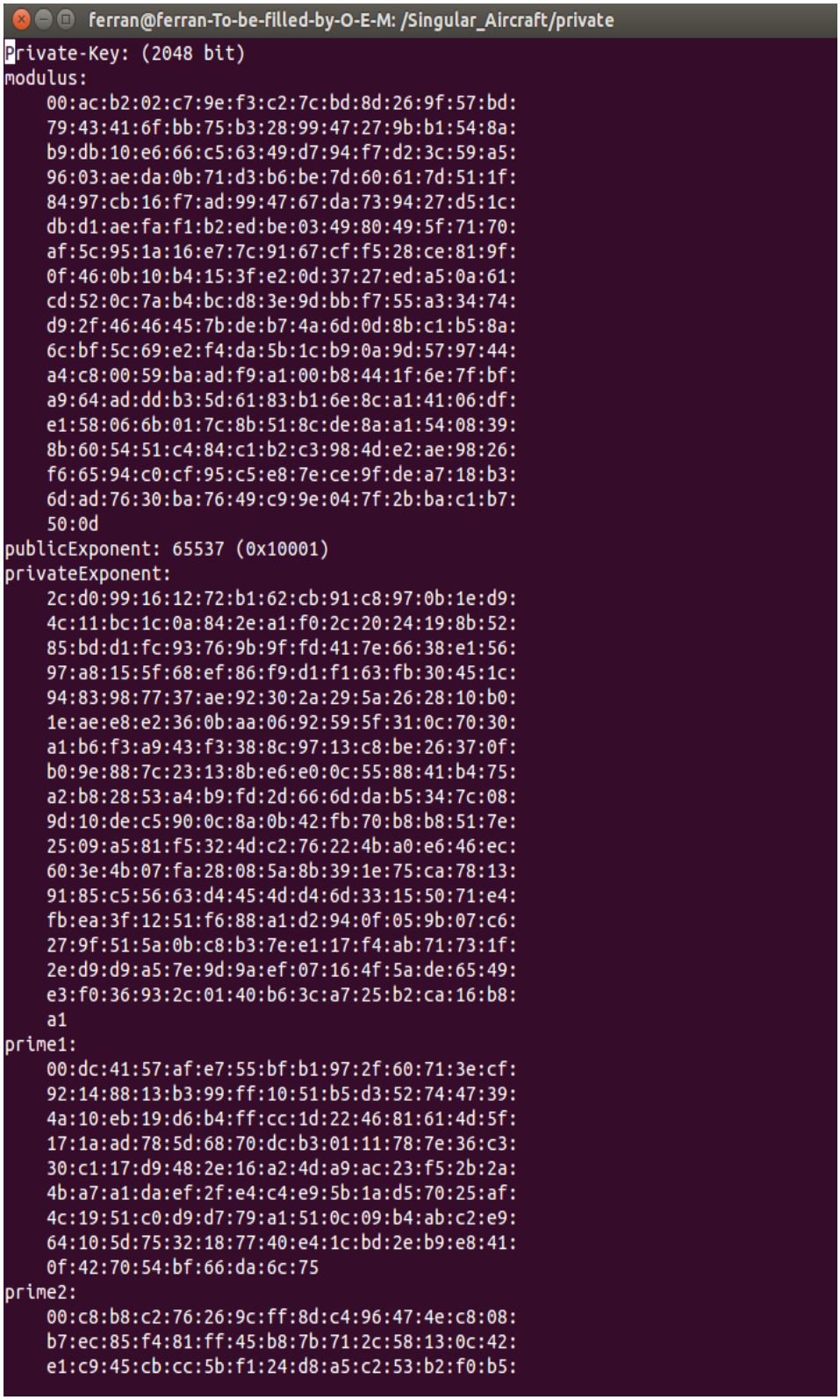


Esquema 13

Mitjançant instruccions podem generar un fitxer que tradueixi la clau pública o la privada generades a hexadecimal. Per exemple per a veure la clau privada (cakey.pem) en format hexadecimal podem fer ús de la següent instrucció:

openssl rsa -in cakey.pem -text -out cakey_text.pem

Una part de la clau quedaria com es pot observar en la imatge de la següent pàgina:


ferran@ferran-To-be-filled-by-O-E-M: /Singular_Aircraft/private
Private-Key: (2048 bit)
modulus:
00:ac:b2:02:c7:9e:f3:c2:7c:bd:8d:26:9f:57:bd:
79:43:41:6f:bb:75:b3:28:99:47:27:9b:b1:54:8a:
b9:db:10:e6:66:c5:63:49:d7:94:f7:d2:3c:59:a5:
96:03:ae:da:0b:71:d3:b6:be:7d:60:61:7d:51:1f:
84:97:cb:16:f7:ad:99:47:67:da:73:94:27:d5:1c:
db:d1:ae:fa:f1:b2:ed:be:03:49:80:49:5f:71:70:
af:5c:95:1a:16:e7:7c:91:67:cf:f5:28:ce:81:9f:
0f:46:0b:10:b4:15:3f:e2:0d:37:27:ed:a5:0a:61:
cd:52:0c:7a:b4:bc:d8:3e:9d:bb:f7:55:a3:34:74:
d9:2f:46:46:45:7b:de:b7:4a:6d:0d:8b:c1:b5:8a:
6c:bf:5c:69:e2:f4:da:5b:1c:b9:0a:9d:57:97:44:
a4:c8:00:59:ba:ad:f9:a1:00:b8:44:1f:6e:7f:bf:
a9:64:ad:dd:b3:5d:61:83:b1:6e:8c:a1:41:06:df:
e1:58:06:6b:01:7c:8b:51:8c:de:8a:a1:54:08:39:
8b:60:54:51:c4:84:c1:b2:c3:98:4d:e2:ae:98:26:
f6:65:94:c0:cf:95:c5:e8:7e:ce:9f:de:a7:18:b3:
6d:ad:76:30:ba:76:49:c9:9e:04:7f:2b:ba:c1:b7:
50:0d
publicExponent: 65537 (0x10001)
privateExponent:
2c:d0:99:16:12:72:b1:62:cb:91:c8:97:0b:1e:d9:
4c:11:bc:1c:0a:84:2e:a1:f0:2c:20:24:19:8b:52:
85:bd:d1:fc:93:76:9b:9f:fd:41:7e:66:38:e1:56:
97:a8:15:5f:68:ef:86:f9:d1:f1:63:fb:30:45:1c:
94:83:98:77:37:ae:92:30:2a:29:5a:26:28:10:b0:
1e:ae:e8:e2:36:0b:aa:06:92:59:5f:31:0c:70:30:
a1:b6:f3:a9:43:f3:38:8c:97:13:c8:be:26:37:0f:
b0:9e:88:7c:23:13:8b:e6:e0:0c:55:88:41:b4:75:
a2:b8:28:53:a4:b9:fd:2d:66:6d:da:b5:34:7c:08:
9d:10:de:c5:90:0c:8a:0b:42:fb:70:b8:b8:51:7e:
25:09:a5:81:f5:32:4d:c2:76:22:4b:a0:e6:46:ec:
60:3e:4b:07:fa:28:08:5a:8b:39:1e:75:ca:78:13:
91:85:c5:56:63:d4:45:4d:d4:6d:33:15:50:71:e4:
fb:ea:3f:12:51:f6:88:a1:d2:94:0f:05:9b:07:c6:
27:9f:51:5a:0b:c8:b3:7e:e1:17:f4:ab:71:73:1f:
2e:d9:d9:a5:7e:9d:9a:ef:07:16:4f:5a:de:65:49:
e3:f0:36:93:2c:01:40:b6:3c:a7:25:b2:ca:16:b8:
a1
prime1:
00:dc:41:57:af:e7:55:bf:b1:97:2f:60:71:3e:cf:
92:14:88:13:b3:99:ff:10:51:b5:d3:52:74:47:39:
4a:10:eb:19:d6:b4:ff:cc:1d:22:46:81:61:4d:5f:
17:1a:ad:78:5d:68:70:dc:b3:01:11:78:7e:36:c3:
30:c1:17:d9:48:2e:16:a2:4d:a9:ac:23:f5:2b:2a:
4b:a7:a1:da:ef:2f:e4:c4:e9:5b:1a:d5:70:25:af:
4c:19:51:c0:d9:d7:79:a1:51:0c:09:b4:ab:c2:e9:
64:10:5d:75:32:18:77:40:e4:1c:bd:2e:b9:e8:41:
0f:42:70:54:bf:66:da:6c:75
prime2:
00:c8:b8:c2:76:26:9c:ff:8d:c4:96:47:4e:c8:08:
b7:ec:85:f4:81:ff:45:b8:7b:71:2c:58:13:0c:42:
e1:c9:45:cb:cc:5b:f1:24:d8:a5:c2:53:b2:f0:b5:

Esquema 14

Així doncs ja hem creat una CA-Root i aquesta ja estarà habilitada per a signar tot el que sigui necessari d'ara en endavant.

Com ja s'ha comentat anteriorment en altres apartats d'aquest projecte, pel bon desenvolupament d'aquesta xarxa de confiances nosaltres garantirem que qualsevol ens partícip en les missions serà coneixedor de la clau pública de Singular Aircraft, així doncs qualsevol dada signada amb la clau privada de Singular podrà ser validada per a qualsevol avió o estació base.

En l'escenari que ens trobem és Client qui generarà les claus temporals per al desenvolupament de la missió entre els seus avions i estacions base.

En aquest punt recentment explicat rau la necessitat de convertir en autoritats de confiança i de certificació als nostres Clients.

Un cop creada doncs la CA-Root necessitem crear noves autoritats de certificació. Una per a cada client. Anteriorment hem anomenat aquestes autoritats de certificació com a Centre de Control.

3.4.3.3: Inicialització del Centre de Control:

Per a que aquestes autoritats puguin signar i certificar operacions els haurem d'acreditar amb:

- Una parella de claus pública i privada
- Un certificat de la seva clau pública signat per Singular Aircraft.

Els Clients usaran la seva clau privada per a signar qualsevol dada que generin i faran arribar el certificat digital a tothom qui el necessiti.
Els avions i les bases podran extreure la clau pública del Client del certificat emès per Singular Aircraft i amb aquesta clau podran desxifrar o validar les dades que hagi generat el Client.

Evidentment el Client haurà d'obtindre un hardware especial per a la custòdia de la clau privada que se li entregui.

Per a configurar Client com a autoritat de confiança caldrà configurar en el centre de comandament de client una estructura de fitxers ubicada en el directori arrel del administrador del sistema (/root).

Primerament creem un directori que l'anomenarem amb el nom “CA” /root/CA i en aquest generem tota l'estructura de fitxers definida anteriorment que confirmarà a Client com a autoritat de confiança.

Directori / Arxiu	**Descripció**
/root/CA/certs	Directori on s'emmagatzemen els certificats ja signats i retornats a usuari
/root/CA/newcerts	Directori on s'emmagatzemen els certificats que acaben de ser signats
/root/CA/crl	Directori on són emmagatzemats els certificats

	revocats
/root/CA/csr	Directori on són emmagatzemades les peticions de certificats pendents de signar
/root/CA/private	Directori on s'emmagatzema la clau privada de l'autoritat de certificació, a més a més de totes les demés claus privades generades pel desenvolupament de les missions
/root/CA/serial	Contindrà inicialment el valor 01 seguit d'un salt de línia. Indica el número de sèrie que tindrà el següent certificat que es signi
/root/CA/crlnumber	Contindrà inicialment el valor 01 seguit d'un salt de línia. Indica el número que tindrà la següent llista de certificats revocats (CRL)
/root/CA/index.txt	Inicialment buit. Serà una base de dades que contindrà informació de tots els certificats signats.

Taula 27

Com es pot observar respecte l'anterior definició del sistema de fitxers d'una CA ens falta l'arxiu *openssl.cnf* aquest és opcional però per el tipus de CA que és client, inexpert en protocols de seguretat però confiable per l'entorn, creiem indispensable fer-ne ús i expliquem a continuació com usar-lo.

Recordem abans que res que l'arxiu *openssl.cnf* proporciona una estructura per defecte dels certificats que es generaran i permet modificacions per a ser més o menys estrictes en les dades introduïdes.

Inicialment aquest document el podem trobar en la paqueteria oficial de OpenSSL (*/etc/pki/tls/openssl.cnf*) i està estructurat en els següents apartats:

- [CA]
  Defineix la secció per defecte del document que resumeix les característiques de la CA
- [CA_default]
  Defineix l'estructura de fitxers de la CA i alguns paràmetres per defecte com la duració de la validesa del certificat o la renovació de la CRL
- [req]
  Defineix les mides de les claus i la versió de certificat que s'usarà
- [req_distinguished_name]
  Defineix les especificacions per defecte del certificat.
- [req_attributes]
  Defineix els paràmetres no obligatoris del certificat.
- [usr_cert]
  Defineix els paràmetres que han d'estar presents en qualsevol certificat i ofereix algunes estructures opcionals.
- [v3_req]

Extensions afegides al certificat.

- [v3_ca]
  Extensions de l'autoritat de certificació.
- [crl_ext]
  Extensions de la llista de certificats revocats.
- [proxy_cert_ext]
  Extensió per evitar que s'interpreti un certificat d'usuari final com a una CA.
- [tsa]
  Defineix les estructures bàsiques per a fer ús dels certificats amb la tècnica criptogràfica del timestamp que es basa en la seguretat del qui signa que les dades existien abans d'un temps específic.
- [tsa_config1]
  Defineix la configuració dels certificats que segueixen la tècnica criptogràfica del tsa.

Com es pot observar el document es divideix en diferents seccions, en el nostre cas només serà necessari modificar aquestes tres:

- [*CA_default*]
- [*req_distinguished_name*]
- [*req*]

Inicialment hem de modificar totes les referències a directoris que evidentment no compleixen el disseny d'estructura de fitxers que hem dissenyat.

A la secció [CA_default]:

Variable	**Valor**	**Descripció**
dir	/root/CA	Directori arrel de l'autoritat de certificació
certs	$dir/certs	Directori on s'emmagatzemen els certificats ja signats.
crl_dir	$dir/crl	Directori on s'emmagatzemen els certificats revocats.
database	$dir/index.txt	Arxiu amb la base de dades dels certificats
new_certs_dir	$dir/newcerts	Directori on es guarden els certificats acabats de signar.
certificate	$dir/cacert.pem	Arxiu amb la clau pública de l'autoritat de certificació // En aquest cas certificada//
serial	$dir/serial	Arxiu amb el número de serie dels certificats.
crlnumber	$dir/crlnumber	Arxiu amb el número de serie de revocació. // Si no es volguéssim usar crl només hauriem de comentar aquesta línia//
crl	$dir/crl.pem	Llista dels certificats revocats
private_key	$dir/private/cakey.pem	Arxiu que conté la clau privada de l'autoritat de certificació

RANDFILE	$dir/private/.rand	Arxiu amb el número aleatori privat
x509_extensions	usr_cert	Extensions que han d'afegir-se al certificat.
name_opt	ca_default	Format en que es mostrarà el nom dels certificat abans de que sigui signat.
cert_opt	ca_default	Format en que es mostrarà un certificat abans de que sigui signat.
default_days	30	Dies per defecte per els que es signa un arxiu
default_crl_days	30	Dies per defecte en que s'ha d'actualitzar la llista de certificats revocats d'aquesta autoritat.
default_md	default	Missatge utilitzat per defecte md5
preserve	no	Indica si s'ha de mantindre o no l'ordre Domain Name
policy	policy_match	Política per defecte a aplicar si no se n'especifica cap.

Taula 28

De totes les variables anteriors és important destacar-ne unes quantes:

- default_days: Com ja s'ha comentat en altres seccions d'aquest mateix projecte, la revocació dels certificats que generarà cada Client haurà de ser temporal. Per aquest motiu aquesta variable és molt important. Hem especificat a cinc dies la durada dels certificats de forma arbitrària. Tenint en compte que els certificats que signarà aquesta autoritat han de revocar-se en el moment que ja s'hagin acabat d'usar per la missió, caldrà en cada missió prendre la decisió de quina fracció de temps ens convé més.
- policy: La variable policy de la taula anterior pot prendre dos valors, *policy_match* o *policy_anything*, que es defineixen a continuació:
  - *policy_match:*
    - countryName = match
    - stateOrProvinceName = match
    - organitzationName = match
    - organizationalUnitName = optional
    - commonName = supplied
    - emailAddress = optional
  - *policy_anything*
    - countryName = optional
    - stateOrProvinceName = optional
    - organitzationName = optional
    - organizationalUnitName = optional
    - commonName = supplied
    - emailAddress = optional

El valor *match* indica que aquell camp en el certificat que es vagi a signar haurà de coincidir amb el que conté l'autoritat de certificació. El valor *supplied* indica que és obligatori que aquell camp sigui proporcionat en la petició de signatura del certificat, podent ésser diferent a l'existent a l'autoritat de

certificació. El camp *optional* significa que no és necessari que sigui indicat en la petició de signatura d'un certificat.

Resumint, la política *policy_match* serveix per a que una autoritat de certificació pugui signar els seus propis certificats, mentre que la política *policy_anything* serveix per a que una autoritat de certificació signi certificats de qualsevol organització.

En tot cas, la decisió que s'ha pres per a les autoritats de certificació dels clients és que segueixin la política *policy_match* ja que no els hi haurien d'arribar peticions de certificació externes.

A la secció [req_distinguished_name] haurem d'especificar els següents valors:

Variable	**Valor**	**Descripció**
countryName_default	ES	Estat d'emisió de certificat.
stateOrProvinceName_default	Barcelona	Comunitat autònoma o província d'emisió
localityName_default	Barcelona	Localitat d'emisió del certificat.
0.organizationName_default	Singular_Aircraft	Nom de l'organització
organizationalUnitName_default	*Client X*	Nom de la secció
commonName_defualt	*Autoritat de certificació del Client X*	Nom de l'autoritat
emailAddress_default	*CA_clientX@...*	Direcció de correu del responsable de l'autoritat de certificació.

Taula 29

Com podem observar aquest darrer bloc de modificacions dependrà de cada client i de les seves particularitats.

A la secció [req]:

Aquesta secció només l'hauríem de modificar si n'estem interessats. Els valors que nosaltres implantarem per defecte inicialment seran els següents:

Variable	**Valor per defecte**	**Descripció**
default_bits	2048	Bits per defecte de la clau privada
default_md	sha1	Algoritme de signatura per defecte

Taula 30

Un cop configurat tots els passos anteriors ja estem a punt per a generar les claus pública i privada de Client. A més de fer la demanda de certificació d'aquesta clau per part de la CA-Root (Singular Aircraft) amb les següent comandes:

Generem la clau privada de la següent forma:

openssl genrsa -des3 -out /client1/CA/private/client1.key 1024

Generem el CSR (Certificate Signing Request) que és un fitxer que ha de generar el client per tal de demanar a Singular Aircraft que el signi, validant així que les dades transmeses són correctes. Si Singular verifica el client, aquest podrà validar/certificar altres claus de elements més baixos en la cadena de confiança PKI. Per a generar el CSR fem ús de la següent comanda:

openssl req -new -key /client1/CA/private/client1.key -out /client1/CA/client1.csr

En l'execució el terminal ens sol·licitarà una contrasenya d'accés a la clau (en el cas que li vulguem posar) i totes les dades necessàries de país, localitat, etc. Si el fitxer de configuració ha estat correctament configurat totes les dades per defecte seran vàlides.

Aquesta petició l'hem de fer arribar a Singular Aircraft per a que sigui validada. Quan la petició arribi a la CA-Root (Singular Aircraft) aquesta haurà d'executar una comanda com:

openssl x509 -req -days 3650 -in /Singular_Aircraft/csr/client1.csr -CA /Singular_Aircraft/cacert.pem -Cakey /Singular_Aircraft/private/cakey.pem -out /Singular_Aircraft/newcerts/client1cert.pem -set_seial 01 -sha1

Finalment per a que la CA del client sigui operativa caldrà retornar el certificat just creat a client.

Arribats a aquest punt en que client ja està certificat i validat per Singular Aircraft i que per tant ja forma part de l'estructura de xarxa de confiances i pot actuar com a CA hem d'encarar el cas real d'una missió i la generació de totes les dades necessàries.

3.4.4: Generació de les dades criptogràfiques necessàries per el transcurs d'una missió:

Rescatem dels punts anteriors les dades que Client ha d'entregar al operador o al pilot per a poder fer la posta a punt de l'avió:

- $Cert(K_{P\,A\,1})[K_{S\,Cl\,1}]$
- $Cert(K_{P\,BS\,1})[K_{S\,Cl\,1}]$
- $Cert_{Pla\_de\_vol}$*("L'avió A000001 es relacionarà amb l'estació base BS00001 inicialment)*$[K_{S\,Cl\,1}]$
- $K_{S\,A\,1}$
- $K_{S\,BS\,1}$
- $Cert(K_{P\,Cl\,1})[K_{S\,SA}]$
- $Cert_{PER}$*("L'avió A000001 pertany al Client 1")*$[K_{S\,SA}]$
- $Cert_{PER}$*("L'estació base BS00001 pertany al Client 1")*$[K_{S\,SA}]$

D'aquests punts fins ara hem explicat com generar-ne el següent:

- $Cert(K_{P\,Cl\,1})[K_{S\,SA}]$ *//Certificat de la clau pública de Client1 expedit per SA*

Podem separar tota la resta en els següents blocs:

- Dades signades per Singular Aircraft
    - $Cert_{PER}$*("L'avió A000001 pertany al Client 1")*$[K_{S\ SA}]$
    - $Cert_{PER}$*("L'estació base BS00001 pertany al Client 1")*$[K_{S\ SA}]$
- Dades generades, signades i/o certificades per Client.
    - $Cert(K_{P\ A\ 1})[K_{S\ Cl\ 1}]$
    - $Cert(K_{P\ BS\ 1})[K_{S\ Cl\ 1}]$
    - $Cert_{Pla\_de\_vol}$*("L'avió A000001 es relacionarà amb l'estació base BS00001 inicialment)*$[K_{S\ Cl\ 1}]$
    - $K_{S\ A\ 1}$
    - $K_{S\ BS\ 1}$

3.4.4.1: Dades signades per Singular Aircraft:

En el moment en que Singular Aircraft fa una venda a un cert Client d'un primer joc d'avió + estació base, Singular Aircraft generarà, a més a més del Centre de Control explicat anteriorment, un parell de certificats de pertinença. Un per a cada avió o base que adquireixi. Aquest document senzillament serà un document de text signat amb la clau privada de Singular Aircraft on s'especificarà que l'avió o la base pertanyen al Client en qüestió.

Per a que aquest procés funcioni correctament és essencial seguir sempre, en cada cas, el mateix patró de fitxer de text.
A la vegada, per a que no tingui fissures, és important definir els identificadors de Client, avió i estació base.

En el cas dels avions i les estacions bases el seu identificador serà el número de serie de fabricació.
En el cas dels Clients l'identificador haurà de coincidir amb l'identificador especificat en la generació del seu certificat.
Recordem que justament en l'apartat anterior hem certificat als clients com a autoritats de certificació de segon ordre i per a fer-ho Singular Aircraft ha generat un certificat de la parella de claus publica-privada del Client. En aquest certificat s'hi han introduït dades personals o de la Societat que conforma el Client, sigui quin sigui el seu rol jurídic. Aquestes dades hauran de coincidir amb les que hi apareguin en el certificat de pertinença.

Per tant el format del fitxer per als avions seguirà les següents pautes:

[version info]
major_version=1.0
minor_version=1.0123
[owner id]
client_name=...
SingularAircraft_client_id=...
date=...
address1=...

[item id]
item_type=plane
item_description=model name, ...
item_unique_id= serial number

I anàlogament amb les estacions base:

[version info]
major_version=1.0
minor_version=1.0123
[owner id]
client_name=...
SingularAircraft_client_id=...
date=...
address1=...
[item id]
item_type=base_station
item_description=model name, ...
item_unique_id= serial number

Aquest parell de documents els desarem amb els noms següents:

- /Singular_Aircraft/pertcert_*num_serie_avio*.txt
- /Singular_Aircraft/pertcert_*num_serie_bs*.txt

El següent pas a seguir és la signatura digital d'aquest parell de documents. Per a fer-ho farem ús de la clau privada de Singular Aircraft. En aquest sentit no xifrem per a tindre confidencialitat, el que busquem per aquests dos missatges és integritat i per això ho fem mitjançant un algoritme de signatura digital.

Per a fer-ho podem usar la funcionalitat del programari openssl:

openssl dgst -c sha1 -sign /Singular_Aircraft/private/cakey.pem -out /Singular_Aircraft/percert_num_serie_avio_signatura.sig /.../percert_num_serie_avio.txt

Instrucció que serveix per a signar digitalment i que es resumeix de la següent manera:

*openssl dgst -c **A** -sign **B** -out **C** **D***

on:

- A: Tecnologia de signatura usada (sha1)
- B: Clau usada per a signar
- C: Document on es guarda la signatura
- D: Document a signar.

Anàlogament amb el de la estació base i ja tenim signats els certificats de pertinença.

3.4.4.2: Dades xifrades, signades i/o certificades per Client:

Definim què són les dades que ha de generar Client:

- $Cert(K_{P\,A\,1})[K_{S\,Cl\,1}]$
  Certificat de la clau pública de missió de l'avió emès per client.
- $Cert(K_{P\,BS\,1})[K_{S\,Cl\,1}]$
  Certificat de la clau pública de missió de l'estació base emès per client.
- $Cert_{Pla\_de\_vol}$*("L'avió A000001 es relacionarà amb l'estació base BS00001 inicialment)*$[K_{S\,Cl\,1}]$
  Pla de vol de l'avió generat per Client i signat per client.
- $K_{S\,A\,1}$
  Clau privada de sessió de l'avió.
- $K_{S\,BS\,1}$
  Clau privada de sessió de l'estació base.

Respecte el pla de vol el procediment és el mateix que en el dels certificats de pertinença explicats recentment. En aquest cas haurem de signar amb la clau privada de client en comptes de la clau privada de Singular Aircraft. Tot i així destaquem l'estructura del fitxer a continuació:

```
[version info]
major_version=1.0
minor_version=1.0123
[owner id]
client_name=...                          //Nom del client
SingularAircraft_client_id=...           //Id del client
date=...                                 //data
address1=...
[mission_agents id]                      //Informació dels agents participants en el vol
mission_id=...                           //Id de la missió
operator_unique_id=...                   //DNI del operador
pilot_unique_id=...                      //DNI del pilot
plane_unique_id=...                      //número de serie únic de l'avió
base_station_unique_id=...               //número de serie únic de la BS
plane_certificate_fingerprint=...        //Hash (Sha1) del certificat K_P A1 de l'avió
base_station_certificate_fingerprint=... //Hash (Sha1) del certificat K_P BS1 de la BS
[emergency_flight info]                  //Informació del vol d'emergència.
location=...                  //Coordenades on s'ha de dirigir l'avió en emergència.
flight_altitude=...                      //Altitud de vol d'emergència
...
[thresholds_flight_info]                 //Informació dels llindars d'emergència
engine_temperature_sensor_cold           //Llindar de mínima temperatura del motor
engine_temperature_sensor_hot            //Llindar de màxima temperatura del motor
...
```

El pla de vol conté les identitats de tots els agents participants en el transcurs d'una missió, els hash del certificat de les claus públiques de missió generades per Client i informació addicional que és necessari que conegui l'avió abans del enlairament com poden ser els exemples explícits dels llindars acceptables en la temperatura del motor o les coordenades d'emergència en el cas de passar a vol automàtic.
El pla de vol queda pendent d'actualitzacions en els paràmetres que el defineixen a futures converses amb el pilot.

Solucionat el pla de vol, només ens queda la generació de dues parelles de claus publica-privades i la certificació per part de client de la correcta correspondència entre la clau pública i la identitat de a qui correspon.
Per a generar el parell de claus cal fer ús d'OpenSSL, les instruccions necessàries per terminal que serien necessàries serien les següents:

Generem la clau privada de l'avió i de l'estació base:

openssl genrsa -des3 -out /client1/avio1/private/clau_privada_avio.key 1024
openssl genrsa -des3 -out /client1/bs1/private/clau_privada_bs.key 1024

Generem les peticions CSR de certificació per a cada parella de claus:

openssl req -new -key /client1/avio1/private/clau_privada_avio.key -out /client1/avio1/avio1.csr -config /client1/CA/openssl.cnf
openssl req -new -key /client1/bs1/private/clau_privada_bs.key -out /client1/bs1/bs1.csr -config /client1/CA/openssl.cnf

D'aquest punt cal destacar que fem ús del fitxer openssl.cnf de la CA de client. Aquest document és el que hem modificat anteriorment per a deixar unes condicions molt marcades. En executar aquesta comànda ens demanarà dades respecte la identitat de qui està demanant aquest certificat. Al haver usat la política *policy_match* només ho donarà per vàlid en el cas que coincideixin els paràmetres amb els especificats.

Aquestes dues peticions les enviem al servidor de la CA de client. En aquest cas només cal traslladar aquest dos arxius .csr a la carpeta */client1/CA/csr.* Ho podem fer de la següent forma.

mv /client1/avio1/avio1.csr /client1/CA/csr/avio1.csr
mv /client1/bs1/bs1.csr /client1/CA/csr/bs1.csr

Un cop les peticions es troben a la carpeta de la CA de Client és aquesta qui ha d'executar la validació i signatura d'aquestes. Per fer-ho caldrà seguir el mateix model que hem seguit abans amb la CA de Singular Aircraft.

openssl x509 -req -days 5 -in /client1/CA/csr/avio1.csr -CA /client1/CA/cacert.pem -Cakey /client1/CA/private/cakey.pem -out /client1/CA/newcerts/avio1_cert.pem -set_serial 01 -sha1

openssl x509 -req -days 5 -in /client1/CA/csr/bs1.csr -CA /client1/CA/cacert.pem -Cakey /client1/CA/private/cakey.pem -out /client1/CA/newcerts/bs1_cert.pem -set_serial 02 -sha1

En aquest cas cal fixar-se en que hem certificat aquesta parella de claus només per als pròxims 5 dies. En cada situació de missió hauríem de prendre la millor decisió.
A continuació només hauríem de retornar aquests certificats als emissors o entregar-los directament a pilot o operador.

Així doncs hem aconseguit generar una estructura PKI de cadena de confiances que dóna solució al repte que inicialment havíem plantejat.

3.5: Transport de les claus i els certificats:

Tot i aconseguir dissenyar una xarxa PKI que respongui a les nostres necessitats, un dels punts crítics d'aquest sistema és el transport de les claus des de que es generen al Centre de Control del Client fins que arriba el moment de inserir-les a l'avió o l'estació base.

Descartat per condició inicial del repte tecnològic que encarem la possibilitat de transferir les dades usant Internet no ens queda més que el transport físic d'aquest material criptogràfic.

Com a possibilitats en el transport físic existeixen un parell d'opcions que destaquen respecte la resta:

- Memòria flash usb
- Smart Card (Targeta intel·ligent)

Com ja hem explicat a la introducció d'aquesta memòria la Smart Card està dissenyada expressament, entre d'altres tasques, pel transport de material criptogràfic o informació sensible per tant serà la millor alternativa. La memòria flash USB seria una elecció més tradicional poc justificable observant que tots els sectors estan mudant tots els seus sistemes d'identificació i transport de claus a xarxes controlades per Smart Card.

Analitzant una mica més a fons el seu funcionament i la seva seguretat s'ha de comentar que una targeta Smart Card respon a l'estàndard de seguretat de *"alguna cosa que tens"*. Un possible atacant només podrà trencar aquest protocol de seguretat obtenint aquestes targetes. Això sí, si aconsegueix fer-se amb les targetes i és coneixedor del protocol de posta a punt de l'avió no tindrà cap dificultat per a agafar-ne el control.

Així doncs és sensat plantejar-se l'opció de complementar aquesta seguretat del *"alguna cosa que tens"* amb un protocol que treballi amb l'estàndard del *"alguna cosa que saps"*. Aquest segon estàndard pot ser representat per una contrasenya del tipus pin entre de moltes altres tècniques vàlides. *"Allò que saps"* validaria que l'operari és verdaderament la persona que està dient que és.

A nivell de hardware els avions i les estacions bases, així com el Centre de Control de Client, haurien d'afegir en producció uns lectors d'aquest tipus de targetes intel·ligents.

En el pròxim capítol concretem el funcionament de l'Smart Card i expliquem el tipus d'ús que Singular Aircraft n'haurà de fer.

Capítol 4

SmartCard, eina per la jerarquització d'entitats i la gestió del transport de claus

4.1: Objectius del capítol:

- Analitzar l'ús de la Targeta Intel·ligent com a eina de transport de les claus i els certificats digitals.
- Analitzar els requeriments d'aquest sistema tant en software com en hardware.
- Compatibilitzar-ho amb l'escenari definit en el capítol anterior.
- Detallar l'ús del protocol APDU d'accés a la Smart Card per a emmagatzemar les claus.

4.2: Punt de partida:

4.2.1: Dades a transportar:

La situació de partida és l'escenari definit en el capítol anterior. Per a que tot aquell esquema de xarxa PKI dissenyat funcioni correctament serà necessari gestionar el transport dels certificats digitals i les claus temporals.

Centrem la nostra anàlisi en el punt on hem deixat el capítol anterior. És a dir en el moment que client genera i entrega a operador o pilot les claus i certificats necessaris per a dur a terme una missió. Aquestes dades es transporten amb l'avió i l'estació base inicial al punt de partida de la missió i han de començar la posta a punt. És en el transport on apareix la necessitat logística de fer ús de les Smart Card.

Per a resituar-nos, les dades a transportar seran les següents:

- $Cert(A000001, K_{P\,BS\,1})[K_{S\,Cl\,1}]$
- $Cert(BS00001, K_{P\,A\,1})[K_{S\,Cl\,1}]$
- $K_{S\,A\,1}$
- $K_{S\,BS\,1}$
- $Cert_{PER}(\text{“L'avió A000001 pertany al Client 1”})[K_{S\,SA}]$
- $Cert_{PER}(\text{“L'estació base BS00001 pertany al Client 1”})[K_{S\,SA}]$
- $Cert_{ID}(\text{“operador DNI\_Operador de l'autoritat de formació AF\_id” })[K_{S\,SA}]$

- *Pla de vol.*
- *Cert($K_{P\,Cl\,1}$) [$K_{S\,SA}$]*

En aquest capítol i des del punt de vista del transport podem separar aquestes dades criptogràfiques en dos blocs: *públiques i secretes*.

Bloc 1: Dades públiques.

Les dades públiques són totes aquelles que en el nostre escenari no importa que siguin conegudes per tothom, ja que amb aquesta informació no n'hi ha prou per a donar ordres a l'avió. Amb la definició establerta seria lògic pensar que aquests missatges es podrien quedar emmagatzemats dins l'avió, que tot i ser vulnerable quan no està en ús, no ens preocupa que hi puguin accedir. Considerem igualment que el millor és que els avions segueixin el mateix procediment en cadena de fabricació i que sigui només el número de sèrie el que variï entre dos avions.
Un altre motiu pel qual ens hem de preocupar pel transport d'aquests blocs de dades és que alguns d'ells destaquen per la seva particularitat d'ésser temporals. Generats per a una missió. Neixen al Centre de Control abans de la missió i moren quan la missió acaba per tant no ens ajuda el fet de deixar emmagatzemats aquests criptogrames a l'avió ja que no es podran usar altra vegada.

Els criptogrames d'aquest bloc són:

- *Cert(A000001, $K_{P\,BS\,1}$)[$K_{S\,Cl\,1}$]*
- *Cert(BS00001, $K_{P\,A\,1}$)[$K_{S\,Cl\,1}$]*
- *$Cert_{PER}$(“L'avió A000001 pertany al Client 1”)[$K_{S\ SA}$]*
- *$Cert_{PER}$(“L'estació base BS00001 pertany al Client 1”)[$K_{S\ SA}$]*
- *$Cert_{ID}$(“operador DNI_Operador de l'autoritat de formació AF_id”)[$K_{S\,SA}$]*
- *Pla de vol.*
- *Cert($K_{P\,Cl\,1}$) [$K_{S\,SA}$]*

Veiem així que totes aquestes dades requereixen únicament solucionar la logística de la **integritat** en el seu transport. Tot i així al ser certificats verificables, en que la integritat criptogràfica ja està garantida pel sistema de confiances descrit anteriorment, aquestes dades només hem de decidir com transportar-les.

Bloc 2: Dades secretes:

Els blocs de dades privades són aquells que a més a més de la **integritat** en el transport, com és el cas en els públics, també requereixen **confidencialitat**. En el nostre escenari, els criptogrames privats són les claus secretes de missió temporals que usaran l'avió i l'estació base.

- $K_{S\,A\,1}$
- $K_{S\,BS\,1}$

Així doncs, s'estableix com a punt de partida de l'estudi la necessitat de transportar de forma segura un conjunt de dades criptogràfiques.

A més a més el transport d'aquestes dades ha de complir els següents requisits:

- Respectar limitacions dels avions i les estacions base com:
  - La identificació no pot suposar canvis significatius en la cadena de muntatge dels avions i les estacions base.
- Respectar la limitació del Client:
  - El Client no serà considerat expert en criptografia. Les accions a desenvolupar per a l'establiment d'una connexió segura han de ser merament protocol·làries al seu entendre.
- Respectar les limitacions de l'escenari:
  - Algunes de les dades criptogràfiques s'han de generar en la planificació de la missió, per tant, poc temps abans que hagin de ser usades.
  - Les dades a generar han de ser emeses pel Client i entregades a l'Operador o al Pilot.
  - En condicions inicials l'enllaç de radiofreqüència no és segur.
  - No es pot garantir l'enllaç inicial entre el client, les diferents estacions base i els avions corresponents a cada missió en una mateixa xarxa compartida. Ni tan sols d'accés a Internet en tot moment.

Tenint en compte totes aquestes limitacions proposem la solució d'ús de **Smart Card**.

4.2.2 Smart Card; Què és i per a què serveix:

Com a concepte, una SmartCard és un element segur. La informació guardada a dins és impossible de ser llegida si no es coneix la clau necessària, normalment un PIN. Aquesta característica la fa especialment idònia per l'emmagatzematge de material criptogràfic (claus secretes) o altra informació sensible.

Uns quants exemples d'SmartCards que usem tots en el dia a dia són, les targetes SIM (emmagatzemen informació de la subscripció), targetes bancàries (guarden informació sobre el compte bancari) o el DNI electrònic (que entre altres coses protegeix una clau secreta única).

Hi ha dos tipus de Smart Card segons la tecnologia que continguin:

1. Amb memòria: Targeta de tipus moneder que són d'un sol ús i que només pot generar decrements a una quantitat inicial.
2. Amb Microprocessador: Targetes amb capacitat de càlcul i processament. Les dades emmagatzemades no podran ésser mai directament accessibles per a aplicacions externes. Aquest tipus de targetes són les més utilitzades en aplicacions tan habituals com els controls d'accés d'edificis, aplicacions bancàries i totes aquelles aplicacions on la seguretat de les dades és primordial.

En l'escenari que ens pertoca analitzar són les del segon tipus les que necessitarem.

Avui en dia, la majoria d'Smart Cards implementen una API (application programing interface), anomenada Java Card. Aquesta API és estàndard i està molt establerta a la indústria. Una aplicació Java Card funciona amb qualsevol targeta, sigui quin sigui el seu fabricant.

A partir de la API Java Card es poden programar Applets usant una semisintaxi de Java i s'instal·len a la SmartCard usant un lector/escriptor de targetes.

Per tant com podem veure les SmartCard són programables i es poden configurar perquè facin el que sigui necessari en cada moment. Un ús molt freqüent de les Smart Card és en els telèfons mòbils. De fet en una mateixa SmartCard si hi instal·les un Applet GSM la converteixes en una SIM (usable pel telèfon mòbil), si hi instal·les una Applet de VISA la converteixes en una targeta de crèdit. Si hi instal·les totes dues podrem realitzar el sistema NFC de pagament a través de l'antena del mòbil.

Les limitacions d'aquest sistema està en les pròpies Applets de JavaCard, per les seves característiques i el seu alt nivell de seguretat, l'experiència d'usuari és majoritàriament força complicada i farragosa. Els problemes solen venir donats per l'entorn (Lector, Sistema Operatiu del PC, Llenguatge de programació JavaCard, Navegador...) més que no per la Smart Card.

4.2.3:Solució proposada:

La solució proposada per a paliar els requeriments de transport i seguretat de l'escenari de Singular Aircraft basat en Smart Card es basa en els següents punts de hardware i interacció:

- Centre de Control del Client: En el Centre de Control del Client es crearan i distribuiran les diferents Smart Cards necessàries pel bon desenvolupament de la missió. El Client mitjançant un lector de targetes i Applet de Java Card podrà introduir els certificats, les claus secretes, les bases de dades i el pla de vol necessari.
- Ordinador d'accés + Lector de Targetes: A l'Avió hi haurà un lector de targetes vinculat a un petit ordinador que comptarà amb el suport d'una pantalla perquè la interacció amb l'operador sigui més fàcil. Aquest ordinador gestionarà el Control d'Accés a l'Ordinador Central. L'Ordinador Central de l'avió no s'engegarà si no hi ha una prèvia identificació a l'ordinador d'accés.
  En el lector o lectors d'SmartCard introduirem les targetes. Tota la tasca de certificats només necessitarà ser llegida una vegada de la targeta, en canvi la SmartCard que conté la clau secreta temporal (generada específicament per aquesta missió) haurà de ser físicament a l'avió durant tot el transcurs del vol, ja que la mateixa targeta és la clau secreta.
  Tot i així, la visió mostrada anteriorment està subjecte a canvis depenent de les proves i les converses amb els fabricants de les targetes i els lectors. Al cap i a la fi, és un risc que la clau privada depengui d'una lectura electrònica en l'entorn de canvis importants en el que vola l'avió. Caldrà estudiar si un canvi sobtat i important d'acceleració, temperatura, pressió o humitat pot fer que baixi la fiabilitat del sistema de lectura o no. En tot cas és una qüestió menor de la que es poden trobar alternatives.

Com a complement d'aquest projecte d'identificació que fa ús de la Smart Card (respon a l'exigència en el món de la seguretat de l'alguna cosa que tens) i de l'ús d'un codi PIN de reconeixement (l'alguna cosa que saps) es planteja la possibilitat d'introduir un lector biomètric ja sigui de l'empremta digital o de la retina (l'alguna cosa que ets). Amb aquests tres filtres d'identificació es garanteix una total seguretat en l'accés i només una mala praxis d'un operador podria dur-nos a una situació d'accés fraudulent.

Com ja s'ha especificat a la introducció d'aquest projecte no entrarem en la programació de les Applets sent aquest un bloc suficientment extens en treball i investigació com per a donar per a un projecte final de carrera sencer. Tot i així passem ara a explicar el funcionament intern de la Smart Card, el protocol amb el que es comunica amb el lector i les especificacions que hauria de tindre aquesta Applet de JavaCard per a poder presentar-se com a solució del nostre repte tecnològic.

4.3: Casos d'ús del protocol APDU en la comunicació entre targeta i lector.

En la comunicació d'una Smart Card i intervenen dos elements, la pròpia Smart Card i un lector de targetes intel·ligents.

Esquema 15

4.3.1: Protocol APDU:

Les Smart Card es comuniquen amb el lector de targetes mitjançant el protocol APDU definit a la ISO 7816. Les targetes no inicien mai la comunicació, sempre responen a trames APDU rebudes del lector. Aquesta situació ens defineix dos tipus de trames APDU:

- APDU de comanda. (Lector --> Smart Card)
- APDU de resposta. (Smart Card --> Lector)

Els camps d'aquestes trames es distribueixen de la següent forma:

APDU de comanda:

Capçalera Obligatòria				**Cos Opcional**		
CLA	INS	P1	P2	LC	DATA	LE
1 byte	1 byte	1 byte	1 byte	1 byte	0-255 bytes	1 byte

Esquema 16

A la capçalera obligatòria de la trama hi trobem quatre bytes, un byte per a cada subcamp de la capçalera:

- CLA: Classe de la instrucció, indica el tipus de comanda (e.g. *Interindustry or propietary*)
- INS: Codi d'instrucció, indica la comanda específica dins d'una classe (e.g. *write data*)
- P1 i P2: Paràmetres referits a la comanda (e.g. Offsets d'escriptura...)

En el cas del cos opcional aquest té longitud variable depenent dels camps de dades que es vulguin omplir. Els paràmetres d'aquest cos són els següents:

- LC: Longitud del camp de dades enviat (00...FF)
- DATA: Des de 0 a 255 bytes de dades, fixades per LC

- LE: Longitud del camp de dades esperat a la resposta (00...FF) (00 = 256 bytes)

APDU de resposta:

DATA (opcionals)	SW1	SW2
0-256 bytes	1 byte	1 byte

Esquema 17

- DATA: Des de 0 a 256 bytes de dades, fixades pel camp LE rebut en la trama APDU comanda. Inclosa la possibilitat de que el camp sigui Nul.
- SW1 i SW2: Estats de processament de comanda. Referits al estat de la comunicació.

Fins ara ens hem centrat en definir el significat d'aquests camps i en mostrar l'estructura de les trames comanda i resposta. Entrem ara més en detall als camps que acabem de definir, tractem-los un per un i fem un anàlisi seriós sobretot dels camps més ambigus com són CLA i INS.

CLA:

Aquest camp s'usa per indicar quin protocol segueix la trama qual precedeix. Com veurem depenent del valor que agafi aquest byte la trama seguirà els estàndards establerts en la ISO7816 en diferents graus o deixarà llibertat per a estructurar el protocol de forma lliure.

Els valors típics són els següents:

Valor	Significat
0x0X	Estructura i codificació de les comandes i respostes acord amb la ISO/IEC 7816. (Per al valor X veure taula següent)
0x10 fins a 0x7F	Reservat per RFU (Radiofreqüència Unit)
0x8X, 0x9X	Estructura de les comandes i respostes acord amb la ISO/IEC 7816. La codificació i significat de comandes i respostes són definides per propietari.
0xB0 fins a 0xCF	Estructura de les comandes i respostes acord amb la ISO/IEC 7816.
0xD0 fins a 0xFE	Estructura i codificació de les comandes i respostes escollides per propietari
0xFF	Reservat per a PTS (Protocol Type Selection)

Taula 31

Per a poder conèixer el significat dels últims 4 bits en els casos que no està determinat en la taula anterior ho resumim seguidament:

b4 b3 b2 b1	**Significat**
x x -- --	Format de Missatgeria Segur. (SM)
0 x -- --	En el cas de que existeixi SM no és acord al estàndard 1.6. [8]
0 0 -- --	No es fa ús del SM.
0 1 -- --	Format SM definit per propietari.
1 x -- --	Format SM d'acord amb l'estàndard 1.6. [8]
1 0 -- --	Capçalera de comanda no autenticada.
1 1 -- --	Capçalera de comanda autenticada. Els diferents usos es troben al 1.6.3.1 de [8]
-- -- x x	Número del canal lògic (1.5[8]). En el cas de b2b1=00 no seleccionem cap canal lògic.

Taula 32

Per entendre millor aquestes dues taules volem remarcar el concepte de “propietari”. Quan parlem de que una estructura o una codificació està definida per propietari ens referim a que el protocol ens reserva aquells valors del byte CLA per a no lligar-nos a la estructura o la codificació definida a la ISO/IEC 7816. L'inconvenient de no lligar-nos a l'estàndard serà la programació de les Applets de Java Card, i és que facilita molt fer ús de funcions o valors estàtics ja definits en JavaCard per l'ús comú de tothom.

Tot i així és cert també que basar tot el nostre sistema en l'estàndard que defineix la ISO 7816 no és un bon remei. Aquesta defineix en un cas molt genèric de totes aquestes funcions i nosaltres en necessitarem de més específiques per a poder donar una bona resposta al nostre escenari.

En aquest cas ens decidim per a la CLA **0x80** que respon a les següents característiques:

- Estructura de les comandes i respostes acord amb la ISO/IEC 7816.

  Que significa que usarem l'estructura de trames comanda i trames resposta definida anteriorment. Amb tots els camps i longituds definides.

- Codificació i significat de les comandes i respostes definits pel propietari.

  Podrem interpretar els valors d'aquests de forma lliure i generar funcions per a realitzar tasques no definides a la ISO.

- Deixarem a 0 els quatre últims bits perquè no farem ús de SM ni el canal lògic.

  En les funcions d'escriptura i lectura que hem de desenvolupar no fa falta afegir-hi el tipus de criptografia simètrica de blocs que ens proposa el mode SM.

INS:

El camp d'instrucció defineix el tipus de comanda que es tracta i el perquè de la comunicació entre targeta i lector.
La ISO 7816 defineix les següents instruccions com a reservades per a comunicacions de la targeta:

Valor	**Nom de la comanda**
0x0E	Esborrar binari
0x20	Verificar
0x70	Canal de Gestió
0x82	Autenticació Externa
0x84	Obtenir un Repte (get challenge)
0x88	Autenticació Interna
0xA4	Seleccionar arxiu
0xB0	Llegir binari
0xB2	Llegir Registre
0xC0	Obtenir Resposta
0xC2	"Envelope"
0xCA	Obtenir dades
0xD0	Escriure binari
0xD2	Escriure Registre
0xD6	Actualitza binari
0xDA	Introducció de dades
0xDC	Actualització de dades
0xE2	Annexar Registre

Taula 33

Aquestes instruccions, de les que en podem fer ús sempre que volem, tenen el byte d'instrucció reservat i per tant l'Applet de JavaCard, si s'ha importat correctament la llibreria de la ISO 7816, podrà executar la instrucció pertinent sense nosaltres haver de desenvolupar el codi.

Tot i així més endavant d'aquest projecte explicarem les bases del nostre protocol i mostrarem un exemple per a la programació de les funcions d'escriptura i lectura de les que haurà de constar la nostra Smart Card.

Com a resum volem destacar que una instrucció és sinònim d'una funció en l'Applet de JavaCard.

Paràmetres P1 i P2:

Degut a la tria que hem fet en el tipus de classe (0x80) podem interpretar com vulguem aquests paràmetres en les funcions que nosaltres desenvolupem a JavaCard. A més a més podrem referir-nos i extreure'n el seu valor de forma molt senzilla si al programar importem la llibreria de la ISO. Ho podrem veure al exemple del final del capítol.

Camps opcionals:

En el cas dels camps opcionals cal destacar-ne que poden ser no existents. Si no s'han d'usar no s'envien. És feina de l'Applet de JavaCard interpretar les dades que rep en les trames APDU i es programarà per a poder-ho fer. Els quatre casos possibles són els següents:

- El cos opcional no envia dades ni n'encarrega de resposta:
  - L_C = Null;
  - Data = Null;
  - L_E = Null;
- El cos opcional només encarrega dades de resposta:
  - L_C = Null;
  - Data = Null;
  - L_E = Valor;
- El cos opcional només envia dades:
  - L_C = Valor;
  - Dades = L_C Bytes;
  - L_E = Null;
- El cos opcional envia dades i n'espera de resposta:
  - L_C = Valor;
  - Dades = L_C Bytes;
  - L_E = Valor;

SW1 i SW2:
Els camps SW1 i SW2 defineixen un cop acabat un intercanvi de trames APDU comanda/resposta l'estat de la comunicació, les possibilitats amb 2 bytes son força àmplies i uns exemples que ens seran útils en el nostre escenari seran:

SW1	**SW2**	**Significat**
90	00	La comunicació ha estat correcta
62	81	Possible corrupció de les dades retornades
67	00	Longitud incorrecta
6B	00	Paràmetres P1 – P2 incorrectes
6D	00	Codi d'instrucció no suportat o no vàlid
6E	00	Classe no suportada
6F	00	No hi ha un diagnosi precís

Taula 34

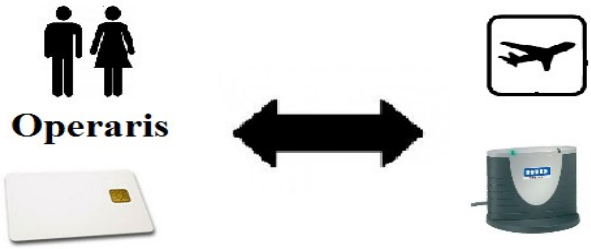
4.3.2: En el nostre escenari, lectura i escriptura a la Smart Card:

Ara que ja sabem com funciona el protocol APDU i també les dades que inclouen les targetes és adequat analitzar la metòdica en una missió.
De tota la missió centrarem ara el nostre anàlisi en la posta a punt de l'avió. En aquell punt l'operari té com a missió validar els certificats digitals expedits per Singular Aircraft o el Client corresponent per a aconseguir fer l'establiment d'una primera comunicació segura.
Per a posar-nos altra vegada en situació repassem què tenim en aquell mateix punt.
L'Operador, el Pilot, l'estació base i l'avió s'han traslladat al punt inicial de la missió i l'operador vol començar la posta a punt. Per a fer la posta a punt a nivell criptogràfic (identificar-se entre d'altres objectius) l'operador disposa d'una Smart Card amb les següents dades emmagatzemades:

- *Cert(A000001, $K_{P\,BS\,1}$)[$K_{S\,CI\,1}$]*
- *Cert(BS00001, $K_{P\,A\,1}$)[$K_{S\,CI\,1}$]*
- *$K_{S\,A\,1}$*
- *$K_{S\,BS\,1}$*
- *$Cert_{PER}$("L'avió A000001 pertany al Client 1")[$K_{S\,SA}$]*
- *$Cert_{PER}$("L'estació base BS00001 pertany al Client 1")[$K_{S\,SA}$]*
- *$Cert_{ID}$("operador DNI_Operador de l'autoritat de formació AF_id")[$K_{S\,SA}$]*
- *Pla de vol.*
- *Cert($K_{P\,CI\,1}$) [$K_{S\,SA}$]*

Així mateix l'avió disposa d'un lector de Smart Card i internament en hardware la clau pública de Singular Aircraft.

- $K_{P\,SA}$



Esquema 18

4.3.2.1: Lectura de dades de la Smart Card.

El primer pas de la validació és validar el Certificat d'identitat que el valida com a operari que depèn de Singular Aircraft.

- *$Cert_{ID}$("operador DNI_Operador de l'autoritat de formació AF_id")[$K_{S\,SA}$]*

Al ser l'Smart Card una eina passiva, que només respon a peticions del seu *màster* (el lector), qui necessàriament iniciarà la comunicació serà el lector de l'avió.

APDU comanda:

Els APDU comanda serviran per a demanar a la Smart Card que transfereixi els certificats necessaris per a gestionar la identificació.

Capçalera Obligatòria				**Cos Opcional**		
CLA	INS	P1	P2	LC	DATA	LE
1 byte	1 byte	1 byte	1 byte	1 byte	0-255 bytes	1 byte

Esquema 16

Capçalera obligatòria; CLA, INS, P_1 i P_2:

Com ja s'ha comentat prèviament les trames APDU són la forma de comunicar-se entre el lector i la targeta de forma bidireccional. En els dos vèrtex de la comunicació hi haurà instal·lada una Applet de Java i Javacard respectivament que amb l'ajuda d'un protocol establert interpretaran les dades rebudes per a generar les respostes corresponents en cada cas.

El protocol ha de servir per a que el lector sàpiga quines trames ha d'enviar i quin format han de tindre aquestes per a que el procés d'autenticació pugui ser validat. A la vegada ha de servir per a que la Smart Card interpreti les dades que rep i doni la resposta esperada pel lector.

Anteriorment hem definit els camps obligatoris de la trama APDU comanda i n'hem avançat alguns dels valors típics dels camps que la conformen. Entrem més en detall als valors que poden agafar tots aquests camps en el cas específic de lectura.

CLA: Camp de classe.

Tal com s'ha justificat anteriorment farem ús de la classe **0x80**.

INS: Camp d'instrucció.

Aquest camp defineix el tipus d'instrucció que serà la comanda. És part fonamental del protocol de comunicació entre la Smart Card i el Lector i el camp més important que enviarem en aquesta trama APDU comanda.

Com hem comentat és l'Applet de JavaCard (programada per nosaltres mateixos) la que interpretarà aquestes dades i per tant tenim llibertat per a generar funcionalitats específiques per a quasi tots els valors del camp INS.

És comprensible ara que la primera funció que creem i per tant el primer camp de INS que ocuparem en el nostre protocol de propietari és la de lectura.

INS	**Interpretació en recepció**
0x02	*Llegeix dades*

Taula 35

Que s'hi analitzem les diferents lectures que haurà de realitzar són:

Lectures	**Necessitats de lectura:**
1	*Passem el certificat d'identitat del operador*
2	*Passem el certificat de clau pública de client.*
3	*Passem el pla de vol.*
4	*Passem el certificat que verifica que l'avió és propietat del Client*
5	*Passem la parella de claus asimètriques de l'avió.*

Taula 36

Des del punt de vista del lector de targetes de l'avió (màster) haurem de fer us de un grup de trames comanda per anar llegint tota la informació que necessitem i poder-la anar validant. El lector esperarà a rebre correctament la resposta de cadascuna d'aquestes comandes per enviar la següent.

P1 i P2: Paràmetres.

Els paràmetres serveixen per a indicar diferents detalls respecte la instrucció, per a cada instrucció el seu significat pot ser diferent i en la nostra estructura de propietari els significats que els hi volem donar per lectura són:

- Si no volem indicar res: P1 = P2 = **0x00**.
- Offset de lectura: P1 = 0x00; P2 = OFFSET.

Com veurem seguidament el camp de dades de les trames de retorn pot fer-ho fins a 256 bytes per tant l'offset el tractarem de la següent manera:

Offset real (bytes)	**Offset enviat per paràmetre P2 (1 byte)**
0 bytes	0x00
64 bytes	0x01
128 bytes	0x02
192 bytes	0x03
256 bytes	0x04
320 bytes	0x05
384 bytes	0x06
448 bytes	0x07
512 bytes	0x08
...	...
16320 bytes	0xFF

Taula 37

El byte P2 marcarà el offset usant blocs de 64 bytes. Això ens obliga a treballar en blocs de 64 bytes en totes les trames que escriguin/llegeixin dades menys en l'últim. Amb aquest dimensionat del offset podrem escriure/llegir fins a un màxim de 16384 bytes.

Les Smart Card que es comercialitzen actualment compten amb la memòria EEPROM de mínim 128Kbits que són al voltant de 16000 bytes per tant trobem que els blocs de l'offset són els encertats.

És important remarcar que en qualsevol cas sempre han d'estar inicialitzats a algun valor.

Cos Opcional: L_C, Dades, L_E :

L_E:
En el cos opcional només serà necessari omplir el byte de L_E que defineix la longitud necessària per a retornar el que es demana en el camp de dades. Per aquest motiu el protocol haurà de ser molt clar quan parlem de quants bytes ha de ser la longitud dels diferents certificats. El valor estarà entre 0x01 i 0xFF (1...255 bytes) o 0x00 per el cas de 256 Bytes. Només en el cas de L_E és vàlid el valor **0x00** per a que ens retornin 256 bytes.

Així doncs si plantegem el tractament de tots els certificats com a un únic bloc continu de dades en que les longituds de cada part és coneguda podem usar una mateixa funció de JavaCard per a anar llegint totes les dades i l'ordinador de bord de l'Avió les pot anar validant.

Una aproximació del ordre de magnitud que ens ocuparà aquest bloc continu de dades de tots els certificats que hem de transportar és la següent:

Dades a emmagatzemar	**(Bytes):**
Cert(A000001, K_{PBS1})[K_{SCI1}]	*2 KB*
Cert(BS00001, K_{PA1})[K_{SCI1}]	*2KB*
$Cert_{PER}$(“L'avió A000001 pertany al Client 1”)[$K_{S\ SA}$]	*2KB*
$Cert_{PER}$(“L'estació base BS00001 pertany al Client 1”)[$K_{S\ SA}$]	*2KB*
$Cert_{ID}$(“operador DNI_Operador de l'autoritat de formació AF_id”)[K_{SSA}]	*2KB*
Pla de vol.	*[2 – 10] KB*
Cert(K_{PCI1}) [K_{SSA}]	*2 KB*
K_{SA1}	*2 KB*
K_{SBS1}	*2KB*
TOTAL	*24KB*

Taula 38

L_C i Dades:
Els camps de dades i L_C queden buits en lectura ja que no tenim cap necessitat d'enviar informació en aquesta trama. Evidentment no significa que si algun dia es volgués usar no es podria fer, només seria necessari adaptar la funció de JavaCard amb la que executem la lectura.

Trama APDU resposta:

Dades opcionals:
Evidentment en el cas de lectura que ens trobem les dades no seran opcionals sinó que seran la part fonamental de la resposta. La pròpia funció de JavaCard anirà retornant totes les dades que se li demanin fent cas al paràmetre d'offset rebut en la trama APDU comanda.

Camp d'estats obligatori:
Si la comunicació ha sigut bona la resposta del camp d'estats serà $\mathbf{SW_1 = 0x90}$ i $\mathbf{SW_2 = 0x00}$. Per altres possibles sortides es pot consultar la taula de l'inici del document.

Exemple de Lectura:

Explicat totes les propietats de la lectura de dades de la Smart Card fent ús de les trames APDU comanda i resposta, resumim en el següent esquema un possible cas pràctic de lectura del bloc de dades de 512 bytes:

Comanda 1:

Capçalera Obligatòria				**Cos Opcional**		
CLA	INS	P1	P2	LC	DATA	LE
0x80	**0x02**	**0x00**	**0x00**	**--**	**--**	**0x00**

Resposta 1:

DATA (opcionals)	**SW1**	**SW2**
[256 bytes]	**0x90**	**0x00**

D'aquest 1r traspàs d'informació hem de fixar-nos en els detalls següents:

- El camp INS=0x02 perquè estem en lectura.
- Els dos paràmetres P1 i P2 inicialitzats a 0 perquè no volem mostrar cap offset de lectura.
- L_C i Dades = Null.
- L_E és igual a 0x00 per a demanar com a resposta 256 bytes al camp de dades.
- La resposta inclou els 256 bytes demanats per la trama APDU comanda que ha de correspondre al certificat d'identitat de l'operador.
- El camp d'estats de la resposta confirma que la comunicació ha estat finalitzada correctament.

Comanda 2:

Capçalera Obligatòria				**Cos Opcional**		
CLA	INS	P1	P2	LC	DATA	LE
0x80	**0x02**	**0x00**	**0x04**	**--**	**--**	**0x00**

Resposta 2:

DATA (opcionals)	SW1	SW2
[256 bytes]	0x90	0x00

Esquema 19

On la gran majoria de paràmetres es mantenen igual que a la primera comunicació, el canvi significatiu que s'ha de destacar és l'ús del paràmetre P2 per a indicar un offset de 64 x 4 = 256 Bytes.

4.3.2.2: Escriptura de dades a la Smart Card:

En l'apartat anterior hem analitzat la funcionalitat en el punt de lectura de la Smart Card i el protocol basat en tecnologia APDU per la comunicació entre el lector i la Smart Card.

És evident associar que per a poder fer la lectura d'aquestes dades prèviament haurem d'haver dut a terme l'escriptura d'aquestes a la Smart Card.
El punt d'escriptura es localitza al Centre de Control del Client just abans de començar la missió el Client entregarà a l'operador la Smart Card amb les següents dades emmagatzemades:

- *Cert(A000001, $K_{P\,BS\,1}$)[$K_{S\,Cl\,1}$]*
- *Cert(BS00001, $K_{P\,A\,1}$)[$K_{S\,Cl\,1}$]*
- *$K_{S\,A\,1}$*
- *$K_{S\,BS\,1}$*
- *$Cert_{PER}$("L'avió A000001 pertany al Client 1")[$K_{S\,SA}$]*
- *$Cert_{PER}$("L'estació base BS00001 pertany al Client 1")[$K_{S\,SA}$]*
- *$Cert_{ID}$("operador DNI_Operador de l'autoritat de formació AF_id")[$K_{S\,SA}$]*
- *Pla de vol.*
- *Cert($K_{P\,Cl\,1}$) [$K_{S\,SA}$]*

Per a seguir la mateixa estructura que al capítol anterior, analitzem els paràmetres de les trames APDU i els camps d'aquestes que canviaran la seva funcionalitat respecte lectura.

APDU comanda:

Capçalera Obligatòria				Cos Opcional		
CLA	INS	P1	P2	LC	DATA	LE
1 Byte	**1 Byte**	**1 Byte**	**1Byte**	**1 Byte**	**0-255 bytes**	**1 Byte**

- CLA: El tipus de classe és la **0x80** per a poder fer interpretació de propietari dels camps definits en l'estructura de la trama.
- INS: Així com hem definit un camp INS per a lectura haurem de fer el mateix per a escriptura.

INS	**Interpretació en recepció**
0x04	*Escriptura de dades*

Taula 39

El valor **0x04** per al camp INS provocarà que s'executi una funció JavaCard a la Smart Card d'escriptura.

- Paràmetres P1 i P2: El paràmetre **P1 = 0x00**, en canvi el paràmetre P2 l'usarem per a indicar l'offset d'escriptura del bloc de dades.
- L_C: El camp L_C ha d'indicar la quantitat a dades que s'enviaran seguidament, a l'ésser el valor entre [0,255] Bytes i ser 255 un valor no múltiple del bloc de bytes que hem definit com a offset hem de limitar el camp de dades a blocs màxims de 192 bytes (0xC0), valor més proper i inferior a 255 que compleix ser múltiple de 64.
  En la última trama d'escriptura ens podrem permetre escriure un bloc que no sigui múltiple de 64. Serà l'única excepció.
- Dades: Inclouran el número de bytes de dades definit per el camp L_C.
- L_E: En un principi no demanarem cap tipus de resposta per part de la Smart Card més enllà dels camps d'estat, per això no inclourem aquest camp a la trama. **L_E = Null**

Capçalera Obligatòria				**Cos Opcional**		
CLA	INS	P1	P2	LC	DATA	LE
0x80	**0x04**	**0x00**	**offset**	**1 Byte**	**[0-255] bytes**	**--**

Esquema 20

APDU resposta:

SW1	**SW2**
0x90	**0x00**

Esquema 21

En el cas de la resposta només necessitarem els camps d'estats que confirmin que l'operació s'ha executat amb èxit.

Per a acabar mostrem en un esquema un exemple d'escriptura a una Smart Card d'un bloc de 512 bytes:

Comanda 1:

Capçalera Obligatòria				**Cos Opcional**		
CLA	INS	P1	P2	LC	DATA	LE
0x80	**0x04**	**0x00**	**0x00**	**0xC0**	**[192] bytes**	**--**

Resposta 1:

SW1	**SW2**
0x90	**0x00**

D'aquest 1r traspàs d'informació hem de fixar-nos en els detalls següents:

- El camp INS=0x04 perquè estem en escritura.
- Els dos paràmetres P1 i P2 inicialitzats a 0 perquè no volem mostrar cap offset d'escriptura.
- L_C = 0xCA (192) perquè és el bloc múltiple de 64 més gran en el segment [0,255]
- Dades = [192 Bytes]. Són els primers 192 Bytes d'informació a escriure.
- L_E = Null perquè no volem dades resposta.
- El camp d'estats de la resposta confirma que la comunicació ha estat finalitzada correctament.

Queden 512 – 192 = 320 Bytes ha enviar. Seguim.

Comanda 2:

Capçalera Obligatòria				**Cos Opcional**		
CLA	INS	P1	P2	LC	DATA	LE
0x80	**0x04**	**0x00**	**0x03**	**0xC0**	**[192] bytes**	**--**

Resposta 2:

SW1	**SW2**
0x90	**0x00**

Aquest 2n traspàs d'informació és idèntic al primer però hi ha la següent modificació que tot seguit comentem:

- El camp P2 =0x03 perquè l'offset d'escriptura l'hem de situar després del tercer bloc de 64bytes. 3 x 64 = 192 Bytes. 192 Bytes són els que hem enviat en la primera trama

Abans d'aquesta segona comunicació quedaven 320 bytes per enviar, al ser major a 256 hem tornat a enviar un bloc de 192. Ara mateix portem 384 Bytes escrits i en queden 128.

Comanda 3:

Capçalera Obligatòria				**Cos Opcional**		
CLA	INS	P1	P2	LC	DATA	LE
0x80	**0x04**	**0x00**	**0x06**	**0x80**	**[128] bytes**	**--**

Resposta 3:

SW1	**SW2**
0x90	**0x00**

Esquema 22

Amb aquest tercer i definitiu intercanvi de trames hem assolit l'objectiu d'escriptura dels 512 Bytes.

4.3.3: Programació funció JavaCard per escriptura i lectura:

```
// File: ReadWrite.java
//--------------------------------------------------------------------------
// Especificacions de la funció amb disseny de propietari READ
//   command APDU              CLA = '80' || INS = '02' ||
//                             P1 (= '00') || P2 (= offset [byte]) ||
//                             Le (= number of bytes to read = DATA)
//   response APDU (good case) DATA (with length Le) || SW1 || SW2
//   response APDU (bad case)  SW1 || SW2
//
// Especificacions de la funció amb disseny de propietari WRITE
//   command APDU              CLA = '80' || INS = '04' ||
//                             P1 (= '00') || P2 (= offset [byte]) ||
//                             Lc (= number of bytes to write) ||
//                              DATA (bytes to write)
//   response APDU (all cases) SW1 || SW2
//
//--------------------------------------------------------------------------

package packreadwrite;          // Nom de la package
import javacard.framework.*; //Importa les packages necessàries per a Java Card.

public class ReadWrite extends Applet { // Classe d'escriptura i lectura
  final static byte CLASS       = (byte) 0x80;  //CLA del les trames comanda
  final static byte INS_READ    = (byte) 0x02;  //INS per la trama de lectura
  final static byte INS_WRITE   = (byte) 0x04;  //INS per la trama d'escriptura
  final static short SIZE_MEMORY= (short) 256; // Mida del contenidor de memòria
  static byte[] memory;                        // Memòria de dades de l'aplicació

  //----- Instal·lació i registre de l'applet -------
  public static void install(byte[] buffer, short offset, byte length) {
    memory = new byte[SIZE_MEMORY*64];  // Contenidor de memòria de dades
    new ReadWrite().register();
  }  // tancament de install

  //----- Executor de comandes -----
  public void process(APDU apdu) {
    byte[] cmd_apdu = apdu.getBuffer();

    if (cmd_apdu[ISO7816.OFFSET_CLA] == CLASS) {  // És correcte el byte CLA
      switch(cmd_apdu[ISO7816.OFFSET_INS]) {      // Depenent del byte INS
        case INS_READ:                            // Toca llegir
          cmdREAD(apdu);                          // Crida a la Funció de llegir
          break;
        case INS_WRITE:                           // Toca escriure
          cmdWRITE(apdu);                         // Crida la funció d'escriure
          break;
        default :                                 // Error en el camp de INS
          ISOException.throwIt(ISO7816.SW_INS_NOT_SUPPORTED);
      }  // Tancament switch
    }  // Tancament if
    else {                                        // Error en el camp CLA
      ISOException.throwIt(ISO7816.SW_CLA_NOT_SUPPORTED);
    }  // Tancament else
  }  // Tancament process
```

```
  //----- Codi funció de lectura READ -----
  private void cmdREAD(APDU apdu) {
    byte[] cmd_apdu = apdu.getBuffer();
    //----- Comprovem les condicions inicials -----
    // P1 hauria de ser = 0
    if (cmd_apdu[ISO7816.OFFSET_P1]!=0)
            ISOException.throwIt(ISO7816.SW_WRONG_P1P2); //Saltem error
    // Comprovem que l'offset del P2 hi cap a l'array de memòria
    short offset = (short) (cmd_apdu[ISO7816.OFFSET_P2]); // calculem offset
    if (offset >= SIZE_MEMORY) ISOException.throwIt(ISO7816.SW_WRONG_P1P2);
    // Comprovem que l'offset P2+Le (Dades a rebre) caben a l'array de memòria.
    short le = (short)(cmd_apdu[ISO7816.OFFSET_LC]);  // calculem Le
//----Afegim el cas le = 0x00 com a 256 bytes -------------------
    switch(le){
      case 0:
            le=256;
            break;
      default:
            break;
    } // Tanquem switch
    if ((offset*64 + le) > SIZE_MEMORY*64)
            ISOException.throwIt(ISO7816.SW_WRONG_LENGTH); //Saltem error

    //--Totes les condicions inicials es compleixen, podem enviar les dades--
    apdu.setOutgoing();                  //canvia el sentit de comunicació
    apdu.setOutgoingLength((short)le);   // comunica el número de bytes a enviar
    apdu.sendBytesLong(memory, (short)offset, (short)le);
                    // Envia el número de bytes a la IFD (Image file directory)
  }  // cmdREAD  // Tanquem funció cmdRead

  //----- program code for the APDU command WRITE -----
  private void cmdWRITE(APDU apdu) {
    byte[] cmd_apdu = apdu.getBuffer();
    //----- Comprovem les condicions inicials -----
    // Comprovem que P1=0
    if (cmd_apdu[ISO7816.OFFSET_P1] != 0)
            ISOException.throwIt(ISO7816.SW_WRONG_P1P2);
    // Comprovem que l'offset del P2 hi cap a l'array de memòria
    short offset = (short) (cmd_apdu[ISO7816.OFFSET_P2]); // calculem offset
    if (offset >= SIZE_MEMORY) ISOException.throwIt(ISO7816.SW_WRONG_P1P2);
    // Comprovem si l'offset P2+Lc (Dades a escriure) caben a l'array de memòria
    short lc = (short)(cmd_apdu[ISO7816.OFFSET_LC]); // calculem Lc
    if ((offset*64 + lc) > SIZE_MEMORY*64)
            ISOException.throwIt(ISO7816.SW_WRONG_LENGTH);
    // Comprovem si Lc és 0, no ho pot ser.
    if (lc == 0) ISOException.throwIt(ISO7816.SW_WRONG_LENGTH); //Saltem error

    receiveAPDUBody(apdu);    //  funció per rebre el cos de la trama APDU
    //--Totes les condicions inicials es compleixen, podem escriure les dades--
    Util.arrayCopy(cmd_apdu,(short)ISO7816.OFFSET_CDATA, memory, offset, lc);
      // Aquest procediment de copia és atòmic
    ISOException.throwIt(ISO7816.SW_NO_ERROR);  //comanda executada correctament
  }  // Tanquem funció cmdWRITE

  //----- Funció per rebre el cos de la trama APDU
  public void receiveAPDUBody(APDU apdu) {
    byte[] buffer = apdu.getBuffer();
    short lc = (short)buffer[ISO7816.OFFSET_LC];  // Calculem Lc
```

```
    // Comprovem si Lc és diferent al número de bytes rebuts per escriure
    if (lc != apdu.setIncomingAndReceive())
      ISOException.throwIt(ISO7816.SW_WRONG_LENGTH); // Si no concorda Error
  }  // Tanquem funció receiveAPDUBody
}  // Tanquem la class

//-----------------------------------------------------------------------------
// El codi anterior està basat en el codi font descrit seguidament:

// This source code is under GNU general public license
// (see www.opensource.org for details).
// Please send corrections and ideas for extensions to Wolfgang Rankl
// (www.wrankl.de)
// Copyright 2003-2004 by Wolfgang Rankl, Munich

// Per la realització de les funcions d'escriptura i lectura necessàries pel
// desenvolupament d'aquest PFC s'han afegit parts de codi per part de Ferran
// Alvarós.

//-----------------------------------------------------------------------------
```

4.3.4: Alternativa APDU extended.

Val la pena comentar en aquest PFC que degut al increment recent de memòries EEPROM de les Smart Card i la capacitat de poder emmagatzemar blocs de dades superiors als 2Kb (256 Bytes) s'ha començat a implementar una versió extesa del protocol APDU. Aquesta facilita la transimsió de blocs de dades superiors als 256 bytes no havent de fer ús de diferents seguits de trames APDU.

A nivell de trames APDU canvia molt poca cosa l'estructura de l'APDU comanda queda de la següent forma:

Capçalera Obligatòria				**Cos Opcional**		
CLA	INS	P1	P2	LC	DATA	LE
1 byte	1 byte	1 byte	1 byte	0-3 bytes	0-16777215 bytes	0-3 bytes

Esquema 23

A nivell d' APDU resposta:

DATA (opcionals)	**SW1**	**SW2**
0-16777215 bytes	1 byte	1 byte

Esquema 24

Essent així un exemple per escriure un bloc de dades de 512 bytes:

Comanda 1:

Capçalera Obligatòria				**Cos Opcional**		
CLA	INS	P1	P2	LC	DATA	LE
0x80	**0x04**	**0x00**	**0x00**	**0x0200**	**[512 bytes]**	**-**

Resposta 1:

SW1	**SW2**
0x90	**0x00**

Esquema 25

Val mencionar que de moment les característiques de l'estàndard no estan encara prou generalitzades i pot provocar problemes a l'hora de programar les Applets de JavaCard per a depen de quins models de targetes. En el moment de redactar aquest projecte creiem que la millor alternativa és configurar el transport de les dades criptogràfiques amb el protocol APDU, ja que hi ha una millor acceptació al sector. Tot i així no volem perdre de vista la possibilitat que en el moment de desenvolupar aquest projecte a l'empresa l'APDU extended ja hagi aconseguit assentar-se entre els fabricants d'Smart Card. Sigui com sigui en la programació només hi canviarien petits detalls.

Capítol 5:

Conclusions i línies futures:

Arribem així al final d'aquest Projecte Final de Carrera i en volem destacar les conclusions i línies futures que se'n poden extreure.
En els capítols anteriors s'ha pogut observar l'anàlisi de l'escenari que implica la venta, per part de Singular Aircraft, d'avions i estacions bases a tercers. Seguidament hem fet una proposta per a pal·liar les necessitats criptogràfiques dels dos radioenllaços: principal i emergència, que presenten característiques molt diferents entre ells. Per una banda, a l'enllaç principal hem assegurat confidencialitat, integritat i autenticació de missatge mitjançant PGP (Pretty Good Privacy) tecnologia molt usada en el sector que combina la criptografia de clau simètrica amb la de clau asimètrica. En segon lloc, l'enllaç d'emergència que presenta la limitació més important en la poca informació, en quantitat de bits, que s'hi transmet per a cada trama d'instrucció, ens ha obligat a idear un format de criptografia que l'hem anomenat estratificada per assegurar l'autenticació dels missatges que s'envien des de l'estació base cap a l'avió.

Degut a l'ús pel canal principal de PGP vam trobar la necessitat de donar una resposta a l'obligació d'adaptar el model de gestió de confiances amb el sistema de ventes previst per Singular Aircraft. Com a resposta a aquest repte hem plantejat una estructura jeràrquica de confiances basada amb la tecnologia dels certificats digitals, tecnologia que encaixa a la perfecció amb la tècnica de clau asimètrica RSA que hem usat per a desenvolupar el PGP. En aquest sentit hem constituït a Singular Aircraft com a Autoritat Certificadora arrel (CA-root) i hem plantejat que qualsevol client que adquireixi un lot d'avió i estació base se'l constitueixi CA de segon ordre dotant-lo d'una eina que hem anomenat Centre de Control amb la que pugui generar tot el material criptogràfic necessari de forma puntual pel desenvolupament de cada missió.
El sistema desenvolupat basa la seva fortalesa en la confiança que tot agent participant de la missió diposita a Singular Aircraft, l'autoritat certificadora arrel. Tot i així gràcies a que els avions i les estacions base reconeixen l'autoritat de Singular Aircraft també confiaran amb el Client que Singular els ordeni que han de confiar.
Per a desenvolupar tot aquest material criptogràfic com claus RSA, claus simètriques, certificats digitals o certificats de pertinença, entre d'altres, a més a més de la constitució de les diferents CA's que estructuraran el sistema empresarial de Singular Aircraft hem exposat la utilització del programari OpenSSL que com s'ha vist en el capítol 3 respon a les nostres necessitats.

Finalment vam observar la necessitat de transportar les claus i certificats que s'usen durant una missió des del Centre de Control de client fins al punt on l'operari inicia la posta a punt de l'avió i l'estació base per a poder realitzar-la. Per a pal·liar aquesta necessitat hem plantejat un sistema basat

en targetes intel·ligents. En el Capítol 4 hem especificat el protocol necessari per a gestionar la comunicació entre un lector/escriptor de targetes i les pròpies Smart Card.

Un dels punts més importants d'aquest Projecte Final de Carrera és que tots els dissenys o definicions de protocols en cada una de les necessitats tecnològiques s'adapta a l'estructura de negoci existent de Singular Aircraft sense obligar a fer grans canvis en el seu model intern de treball. Aquest era un objectiu fonamental del projecte que, creiem, s'ha assolit amb èxit.

A partir d'aquest projecte de disseny sorgeixen moltes línies de desenvolupament. Internament l'empresa Singular Aircraft haurà d'aplicar i testejar totes les solucions aquí plantejades:

- Configuració de la xarxa de confiances que garanteixi el bon funcionament del model criptogràfic dissenyat, que comporta els següents punts:
  - Constituir Singular Aircraft com a CA Root capaç de generar claus RSA i tota la mena de certificats definits.
  - Constituir els clients com a CA's certificades per Singular Aircraft amb el desenvolupament de l'eina Centre de Control que faciliti la generació de tot aquell material criptogràfic necessari pel transcurs d'una missió.
  - Dotar als clients del hardware necessari per a emmagatzemar la respectiva clau privada.
- Afegir als enllaços dels avions i les estacions base el software necessari per xifrar/desxifrar les comunicacions segons els protocols establerts en aquest Projecte Final de Carrera.
- Desenvolupar les Applets Javacard per a les targetes que usaran els clients per a transportar el material criptogràfic que usaran en cada missió.

En definitiva, el món dels vols no tripulats és un sector tecnològic innovador que majoritàriament usa tècniques ja existents. Aspectes com l'electrònica, la termodinàmica, les cadenes de producció o muntatge o el consum entre molts d'altres, no són aspectes nous en el món aeronàutic. El que fa innovador aquest sector és l'aposta pel control remot. La integració d'aquest control remot en una tecnologia amb tanta història com és l'aeronàutica obre nous mercats, empresarialment parlant. Ens atrevim a portar els avions a entorns on abans era impensable i per tant, s'obren també nous horitzons d'investigació. La seguretat d'un radioenllaç en serveis “on time” (en directe) n'és un però n'hi ha molts més. En definitiva, els UAV (Unmanned aerial vehicle) ja no són el futur, són actualitat cent per cent.

Bibliografia

[1] Juan Gómez – Matemáticos, espías y piratas informáticos

[2] Adam Berent. *AES: (Advanced Encryption Standard) Simplified.*

[3] Hans Delfs, Helmut Knebl. *Introduction to Cryptography.* Springer 2007

[4] Network Working Group. *UMAC: Message Authentication Code using Universal Hashing.* Ed. T. Krovetz - March 2006

[5] http://csrc.nist.gov/publications/PubsFIPS.html. List of currents FIPS publications on NIST website.

[6] Federal Information Processing Standards Publication. *SHA-3 Standard: Permutation – Based Hash and Extendable-Output Functions.* Information Technology Laboratory, National Institute of Standards and Technology Gaithersburg. Maig 2014

[7] International Discusion Forum about Hash system. *hash-forum@nist.gov*

[8] ISO 7816 Part 1: Physical Charcteristics of Integated Circuit Cards.