CRYPTOGRAPHIC KEY INFRASTRUCTURE FOR SECURITY SERVICES PROTECTING TT&C AND PAYLOAD LINKS OF SPACE MISSIONS

ESTEC/CONTRACT NO. 4000103681

May 2011 – April 2014

Marcio Juliato, Dept. of ECE, Univ. of Waterloo
Catherine Gebotys, Dept. of ECE, Univ. of Waterloo
Ignacio Aguilar Sanchez, ESTEC, ESA
Table of Contents

Table of Contents ............................................................................................................. 2
List of Acronyms .................................................................................................................. 5
List of Figures ....................................................................................................................... 7
List of Tables ........................................................................................................................ 8
Chapter 1 : Report Summary .............................................................................................. 9
  1.1 Introduction .................................................................................................................. 10
  1.2 Background ................................................................................................................... 10
  1.3 Purpose and Scope ....................................................................................................... 12
  1.4 Programme Outline .................................................................................................... 12
  1.5 Output .......................................................................................................................... 15
  1.6 Relevant References .................................................................................................... 16
Chapter 2 : Key Management Fundamental Requirements ................................................ 18
  2.1 Scope ............................................................................................................................ 19
  2.2 Space Mission Scenarios ............................................................................................ 20
    2.2.1 Orbital Types and Associated Parameters .......................................................... 20
    2.2.2 Communication Scenarios .................................................................................. 21
    2.2.3 Summary of Parameters ..................................................................................... 22
  2.3 Key Roles, Scopes and Hierarchy ................................................................................ 23
    2.3.1 Determining Key Roles and Scopes ..................................................................... 23
    2.3.2 Proposed Key Hierarchy and Justification ............................................................ 24
  2.4 Algorithms and Attacks ............................................................................................... 27
    2.4.1 Assumptions on Threat and Trust Models .......................................................... 27
    2.4.2 Attacks against AES ............................................................................................ 28
    2.4.3 Related Key Attacks ............................................................................................. 29
    2.4.4 Exhaustive Search Attacks ................................................................................... 29
    2.4.5 Successful Forgery ............................................................................................... 36
  2.5 Cryptoperiods and Key Lengths .................................................................................... 38
    2.5.1 Related Work ........................................................................................................ 38
    2.5.2 Data Complexity in Related Key Attacks ............................................................ 40
    2.5.3 Data Complexity of a Successful Forgery ............................................................ 41
2.5.4 Time Complexity of Attacks based on Exhaustive Search ........................................ 42
2.5.5 Determining Key Lengths .......................................................................................... 45
2.5.6 Defining Cryptoperiods ............................................................................................... 48
2.6 Instrument/Payload Data Distribution ........................................................................... 50
2.6.1 ESA-Controlled Data Distribution ............................................................................. 50
2.6.2 Direct Data Distribution with ESA Knowledge of Keys ............................................. 50
2.6.3 Direct Data Distribution with User Key Management ............................................... 51
2.7 Over the Air Re-Keying ................................................................................................ 53
2.7.1 LEO (Sentinel-like) Missions ...................................................................................... 53
2.7.2 GEO (Meteosat-like) Missions .................................................................................... 55
2.8 Appendix: Improved Bounds on CMAC Forgery Probability ........................................ 57
Chapter 3 : Key Management Fundamental Requirements (Supplement) and Trusted Platform Research ................................................................................................................. 58
3.1 Scope ............................................................................................................................. 59
3.2 Revisiting Authentication Cryptoperiods ....................................................................... 60
3.2.1 Updated Equations for Authentication Mechanisms .................................................. 60
3.2.2 Computation of $P_{ue}$ for Reed-Solomon Codes ...................................................... 60
3.2.3 Successful Forgery against Authentication Mechanisms ........................................... 61
3.2.4 Graphical Analysis of $P_{Err}$ and $P_{Forg}$ for TC ..................................................... 62
3.2.5 Graphical Analysis of $P_{Err}$ and $P_{Forg}$ for TM ..................................................... 64
3.2.6 Probability of Accepting Forged Data ....................................................................... 65
3.2.7 Updated Authentication Cryptoperiods ..................................................................... 67
3.2.8 Conclusions and Recommendations ........................................................................ 69
3.3 Trusted Platform to Support Key Management ............................................................. 71
3.3.1 Key Management Goals and Infrastructure ............................................................... 71
3.3.2 Related Work ............................................................................................................. 72
3.3.3 Communication Scenarios ....................................................................................... 74
3.3.4 Secret and Keying Materials ..................................................................................... 74
3.3.5 Rekeying Procedure with Hop-by-Hop Secrecy ......................................................... 76
3.3.6 Rekeying Procedure with End-to-End Secrecy ......................................................... 84
3.3.7 Hop-by-Hop Onboard Integrity Check ..................................................................... 90
3.3.8 End-to-End Onboard Integrity Check ...................................................................... 94
## List of Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AES</td>
<td>Advanced Encryption Standard</td>
</tr>
<tr>
<td>BOINC</td>
<td>Berkeley Open Infrastructure for Network Computing</td>
</tr>
<tr>
<td>CC</td>
<td>Control Center</td>
</tr>
<tr>
<td>CCSDS</td>
<td>Consultative Committee for Space Data Systems</td>
</tr>
<tr>
<td>CMAC</td>
<td>Cipher-Based MAC Mode</td>
</tr>
<tr>
<td>CRC</td>
<td>Cyclic Redundancy Check</td>
</tr>
<tr>
<td>CTR</td>
<td>Counter Mode</td>
</tr>
<tr>
<td>DES</td>
<td>Data Encryption Standard</td>
</tr>
<tr>
<td>D-H</td>
<td>Diffie-Hellman</td>
</tr>
<tr>
<td>ECE</td>
<td>Electrical and Computer Engineering</td>
</tr>
<tr>
<td>ECSS</td>
<td>European Cooperation for Space Standardization</td>
</tr>
<tr>
<td>EDAC</td>
<td>Error Detection and Correction</td>
</tr>
<tr>
<td>ESA</td>
<td>European Space Agency</td>
</tr>
<tr>
<td>ESOC</td>
<td>European Space Operations Centre</td>
</tr>
<tr>
<td>ESTEC</td>
<td>European Space Research and Technology Centre</td>
</tr>
<tr>
<td>GCM</td>
<td>Galois Counter Mode</td>
</tr>
<tr>
<td>GEO</td>
<td>Geostationary Orbit</td>
</tr>
<tr>
<td>GS</td>
<td>Ground Station</td>
</tr>
<tr>
<td>HK</td>
<td>Housekeeping</td>
</tr>
<tr>
<td>IETF</td>
<td>Internet Engineering Task Force</td>
</tr>
<tr>
<td>KDC</td>
<td>Key Distribution Center</td>
</tr>
<tr>
<td>KDF</td>
<td>Key Derivation Function</td>
</tr>
<tr>
<td>KDK</td>
<td>Key Derivation Key</td>
</tr>
<tr>
<td>KDS</td>
<td>Key Derivation Seed</td>
</tr>
<tr>
<td>KEK</td>
<td>Key Encryption Key</td>
</tr>
<tr>
<td>MKHK</td>
<td>Keyed Hash Key</td>
</tr>
<tr>
<td>LEO</td>
<td>Low Earth Orbit</td>
</tr>
<tr>
<td>MAC</td>
<td>Message Authentication Code</td>
</tr>
<tr>
<td>MANET</td>
<td>Mobile Ad-Hoc Network</td>
</tr>
<tr>
<td>MK</td>
<td>Master Key</td>
</tr>
<tr>
<td>MKHK</td>
<td>Master Keyed Hash Key</td>
</tr>
<tr>
<td>MOTS</td>
<td>Master One-Time Secret</td>
</tr>
<tr>
<td>MSC</td>
<td>Master Spacecraft</td>
</tr>
<tr>
<td>MTAK</td>
<td>Master Transport Authentication Key</td>
</tr>
<tr>
<td>MTEK</td>
<td>Master Transport Encryption Key</td>
</tr>
<tr>
<td>NIST</td>
<td>National Institute of Standards and Technology</td>
</tr>
<tr>
<td>NPI</td>
<td>Networking/Partnering Initiative</td>
</tr>
<tr>
<td>Acronym</td>
<td>Full Form</td>
</tr>
<tr>
<td>---------</td>
<td>-----------</td>
</tr>
<tr>
<td>OTAR</td>
<td>Over-the-Air-Rekeying</td>
</tr>
<tr>
<td>OTS</td>
<td>One Time Secret</td>
</tr>
<tr>
<td>OUP</td>
<td>Originator Usage Period</td>
</tr>
<tr>
<td>PAK</td>
<td>Password-Authenticated Key Exchange</td>
</tr>
<tr>
<td>PL</td>
<td>Payload</td>
</tr>
<tr>
<td>PRNG</td>
<td>Pseudo-Random Number Generator</td>
</tr>
<tr>
<td>RS</td>
<td>Reed-Solomon</td>
</tr>
<tr>
<td>SC</td>
<td>Spacecraft</td>
</tr>
<tr>
<td>SEC</td>
<td>Single Error Correction</td>
</tr>
<tr>
<td>SID</td>
<td>Session Identifier</td>
</tr>
<tr>
<td>SK</td>
<td>Symmetric Key</td>
</tr>
<tr>
<td>SKDS</td>
<td>Set of Key Derivation Seeds</td>
</tr>
<tr>
<td>SMK</td>
<td>Security Module Management Key</td>
</tr>
<tr>
<td>SPEKE</td>
<td>Simple Password Exponential Key Exchange</td>
</tr>
<tr>
<td>SSK</td>
<td>Set of Session Keys</td>
</tr>
<tr>
<td>SSM</td>
<td>Set of Secret Material</td>
</tr>
<tr>
<td>TAK</td>
<td>Transport Authentication Key</td>
</tr>
<tr>
<td>TC</td>
<td>Telecommand</td>
</tr>
<tr>
<td>TEK</td>
<td>Transport Encryption Key</td>
</tr>
<tr>
<td>TM</td>
<td>Telemetry</td>
</tr>
<tr>
<td>TPM</td>
<td>Trusted Platform Module</td>
</tr>
<tr>
<td>TTC</td>
<td>Tracking, Telemetry and Command</td>
</tr>
<tr>
<td>UW</td>
<td>University of Waterloo</td>
</tr>
<tr>
<td>VANET</td>
<td>Vehicular Ad-Hoc Network</td>
</tr>
<tr>
<td>WSN</td>
<td>Wireless Sensor Network</td>
</tr>
</tbody>
</table>
List of Figures

Figure 1: Key Hierarchy based on (a) Key Transport and (b) Key Derivation ........... 25
Figure 2: Gain and Accumulation of Computational Power .................................. 33
Figure 3: Graphical Representation of Numerical Approximation .......................... 35
Figure 4: Estimates on Security Levels (50 years timeframe) ................................. 46
Figure 5: Estimates on Security Levels (1 year timeframe) ................................. 47
Figure 6: Estimates on Security Levels (1 month timeframe) ................................. 47
Figure 7: Data Distribution through ESA’s CC .................................................... 50
Figure 8: Direct Data Distribution with ESA Knowledge of Keys ............................ 51
Figure 9: Direct Data Distribution with User Key Management ............................. 52
Figure 10: TC $P_{Err}$ (blue) and $P_{Forg}$ (red), $0 \leq n \leq 10^{18}$ .......................... 63
Figure 11: TC $P_{Err}$, $0 \leq n \leq 10^{10}$ ................................................................. 63
Figure 12: TM $P_{Err}$ (blue) and $P_{Forg}$ (red), $0 \leq n \leq 10^{18}$ .......................... 64
Figure 13: TM $P_{Err}$, $0 \leq n \leq 10^{9}$ ................................................................. 65
Figure 14: TC Number of Frames in Function of the Tag Length ($P_{ue} = 10^{-9}$, 10$^{-16}$, $10^{-22}$, 10$^{-28}$) ................................................................. 66
Figure 15: TM Number of Frames in Function of the Tag Length ($P_{ue} = 10^{-9}$, 10$^{-41}$, $10^{-58}$, 10$^{-75}$) ................................................................. 67
Figure 16: Initial System Configuration to Support Key Transport ......................... 76
Figure 17: Key Transport Protocol, Steps 1 and 2 ............................................. 79
Figure 18: Key Transport Protocol, Step 3 ......................................................... 80
Figure 19: Key Transport Protocol, Steps 4 and 5 ............................................. 80
Figure 20: Key Transport Protocol, Step 6 ......................................................... 81
Figure 21: Key Transport Protocol, Step 6 (Alternative) ..................................... 81
Figure 22: Key Transport Protocol, Step 7 ......................................................... 82
Figure 23: Key Transport Protocol, Steps 8 and 9 ............................................. 82
Figure 24: Key Transport Protocol, Steps 10–12 .............................................. 83
Figure 25: Key Transport Protocol, Step 13 ....................................................... 83
Figure 26: Key Transport Protocol, Step 14 ....................................................... 84
Figure 27: Initial Configuration for Key Transport with End-to-End Secrecy ........ 85
Figure 28: Key Transport Protocol with End-to-End Secrecy, Steps 5 and 6 ....... 87
Figure 29: Key Transport Protocol with End-to-End Secrecy, Step 7 ................. 87
Figure 30: Key Transport Protocol with End-to-End Secrecy, Steps 8 and 9 ....... 88
Figure 31: Key Transport Protocol with End-to-End Secrecy, Steps 10–13 ......... 88
Figure 32: Key Transport Protocol with End-to-End Secrecy, Step 14 ............... 89
Figure 33: Key Transport Protocol with End-to-End Secrecy, Step 15 ............... 89
Figure 34: Hop-by-Hop Onboard Integrity Check, Steps 1–6 ............................. 91
Figure 35: Hop-by-Hop Onboard Integrity Check, Steps 7–8 ............................. 92
Figure 36: Hop-by-Hop Onboard Integrity Check, Steps 9–13 .......................... 93
Figure 37: Hop-by-Hop Onboard Integrity Check, Step 14 ............................... 93
Figure 38: End-to-End Onboard Integrity Check, Step 6 .................................. 94
Figure 39: End-to-End Onboard Integrity Check, Step 7 .................................. 95
Figure 40: End-to-End Onboard Integrity Check, Steps 8–14 .......................... 95
Figure 41: End-to-End Onboard Integrity Check, Steps 15–16 ......................... 96
Figure 42: Onboard storage of SSKs and SSMs .............................................. 134
Figure 43: Direction of SSKs and SSMs utilization for MSC$_1$ and MSC$_2$ ......... 134
List of Tables

Table 1: Parameters for Different Space Missions Scenarios ........................................... 22
Table 2: Interdependence between Key Roles and Scopes .................................................. 24
Table 3: DES Cryptanalytic History ....................................................................................... 33
Table 4: Probability of Frame Error ....................................................................................... 36
Table 5: Number of Years Required for Data Collection in a Related Key Attack ...................... 41
Table 6: Maximum amount of data for a Successful Forgery ................................................ 41
Table 7: Cryptoperiods to avoid a Successful Forgery ........................................................... 42
Table 8: AES-128 Exhaustive Search Times ........................................................................... 43
Table 9: AES-192 Exhaustive Search Times ........................................................................... 44
Table 10: AES-256 Exhaustive Search Times .......................................................................... 45
Table 11: Minimum Security Level for a 24h Cryptoperiod ..................................................... 48
Table 12: LEO Authentication Cryptoperiods ....................................................................... 53
Table 13: LEO Authentication Cryptoperiods based on NIST Recommendations ................... 53
Table 14: Number of Authentication SKs for LEO ................................................................. 54
Table 15: Number of KEKs to Rekey Authentication SKs for LEO with CER=10^{-6} .............. 54
Table 16: Number of Encryption SKs for LEO .......................................................... 54
Table 17: GEO Authentication Cryptoperiods ....................................................................... 55
Table 18: GEO Authentication Cryptoperiods based on NIST Recommendations ................... 55
Table 19: Number of Authentication SKs for GEO ............................................................... 55
Table 20: Number of KEKs to Rekey Authentication SKs for GEO with CER=10^{-6} ............ 56
Table 21: Number of Encryption SKs for GEO .......................................................... 56
Table 22: Number of KEKs for High Data Volumes Links of GEO Missions ......................... 56
Table 23: Bounds for Probability of Forgery for CMAC ......................................................... 57
Table 24: Probability of Undetected Errors ($P_{ue}$) for BCH and RS Codes ......................... 61
Table 25: Summary of Worst-Case Communication Scenarios ............................................... 62
Table 26: Maximum Number of Frames for TC’s and TM’s Authentication Mechanisms ......... 68
Table 27: Authentication Cryptoperiods for TC and TM ......................................................... 68
Table 28: SC Requirements for Scenario 1 ............................................................................. 103
Table 29: SC Requirements for Scenario 2 ............................................................................. 104
Table 30: MSC Requirements for Scenario 2 ........................................................................ 105
Table 31: SC Requirements for Scenario 3 ............................................................................. 106
Table 32: MSC Requirements for Scenario 3 ........................................................................ 107
Chapter 1 : Report Summary
1.1 Introduction

This project was a highly collaborative initiative between Ignacio Aguilar Sanchez at ESTEC and researchers, Marcio Juliato (Postdoctoral Fellow) and Catherine Gebotys (Professor) at the University of Waterloo. The outputs of this project are not only the Phase 1,2,3 reports and paper publications, but a clear advancement of state of the art in security in space. It is hoped that these results will help to provide further understanding and increased confidence in employing controlled as well as autonomous security in space.

The three year project involved 3 phases (one per year). Each year a summary report was produced detailing the research and development of the cryptographic key infrastructure and security protocols and algorithms. Phase 1 detailed key management fundamental requirements. Phase 2 progressed with greater accuracy in the key management fundamental requirements and also introduced the trusted platform research. Finally Phase 3 researched reconfigurable and distributed key management.

The motivation for this research is provided in the background section along with the project purpose and scope of activity. The detailed objectives of each phase are outlined in the programme outline section of this brief report. In particular the Phase 1,2,3 summary reports are listed, each with a short summary of their content.

1.2 Background

Over the last years, the European Space Agency (ESA) have been moving from ‘all-in-the-clear’ telecommand (TC) and telemetry (TM) to more security-aware concepts. Although the ATV, GALILEO, ALPHABUS, and GMES SENTINELs-1, -2, and -3 have already included security requirements, that is not a reality for most of the current missions. In the near future, it is envisioned that all ESA and commercial missions will require security for TC and TM, therefore demanding authentication and confidentiality mechanisms. Such mechanisms require in turn a sound cryptographic key management infrastructure to be in place to support them.

While key management systems for both ESA and commercial space missions are being engineered in a number of ESA-funded projects, there has been no formal specification of cryptographic key management schemes targeting space missions. The particular requirements and constraints of space missions like the substantial communications delay between the spacecraft and the mission control system, the stringent reliability requirements with no maintenance possible once in flight, the extreme environmental effects like radiation demand careful consideration. Additionally, optimization of spacecraft resources like computing and data handling is a must.

Ground-based networks have explored several concepts for performing key management, which can be classified in three categories: Pre-Distribution, Arbitrated Key, and Self-Enforcing.

Pre-distribution schemes rely on a Key Distribution Center (KDC) for loading cryptographic keys into the entities’ systems prior to their deployment. If no update of such keys is necessary after the first set-up, no further interaction with KDC is necessary. If a post-
deployment key update becomes necessary, the KDC becomes again responsible for the key management.

Arbitrated Key schemes employ a trusted third party (or arbiter), which we will refer to as a Trusted Authority (TA), for key establishment and are usually based on symmetric-key cryptography (SKC). The drawback of this scheme is the mandatory interaction with TA to obtain symmetric keys. Also, the TA must store keys for all entities in the network it is responsible for, which may demand considerable memory requirements. The main advantage of this scheme is the utilization of SKC, which favors efficient implementation onboard spacecrafts.

Self-enforcing schemes employ asymmetric primitives, i.e. mechanisms based upon public-key cryptography (PKC). Although this scheme facilitates key establishment, it has several drawbacks. The first one refers to the public-key infrastructure (PKI) which requires a Certification Authority (CA) to issue certificates to the public keys. Second, the complexity of the computational operations involved with this scheme involves much more resources than symmetric-key primitives. Therefore, it is still unknown if schemes based on public-key mechanisms can be efficient implemented onboard spacecrafts.

Identity based cryptography (IBC) relies on public-key primitives without the need of CAs, since the entities’ public-key derives directly from the entities’ unique identifier (ID). Such a scheme, however, demands a TA for issuing private-keys based upon entities’ IDs and a master key of its own. Besides, since IBC involves public-key primitives, its computational complexity is higher than symmetric-key primitives.

Although various terminologies can be found in the literature, this research proposal differentiates KDCs, TAs and CAs according to their categories respectively based on pre-distribution, arbitrated key/IBC, and self enforcing schemes.

Concepts and techniques taken for granted in terrestrial key infrastructures like the use of KDCs, CAs and TAs may require revision and adaptation for the proper utilization in space systems.

In addition, space mission architectures and operations are evolving towards more autonomous spacecraft missions. The implications of such trend on security and cryptographic key infrastructures need to be considered and in particular to constellations and Deep Space missions, where communications with a control center may be temporarily unavailable.

Therefore, in order to complete security engineering research and development effort for space mission security, this project aims at researching and designing a cryptographic key infrastructure suitable for present and future space missions with varying levels of complexity and autonomy.
1.3 Purpose and Scope

The main goals of this project are:

- Optimize the design and implementation of security functions for space missions by focusing on sound cryptographic key management;
- Evaluate higher levels of spacecraft autonomy which can provide temporary operational independence from ground-based CAs and TAs. This may be necessary in specific contingency situations where interactive ground communications for key establishment may not be available.

In pursuing these main goals the following research areas were covered in-depth:

- Theoretical/formal foundations of Symmetric Key Infrastructure (SKI) architectures for space missions (key hierarchies, crypto periods, number of keys of various types/roles, optimization criteria…);
- Dependencies between space mission scenarios and key management concepts. Mission architectures to be investigated include single and multiple spacecraft settings along with single and multiple control centers. Limited access to control centers will also be considered, mainly for achieving constant key management even while in contingency situations;
- Key management based on an on-board trusted platform for various missions architectures including collaboration with other spacecraft;
- Feasibility study of PKI applied to space networks;
- Security in contingency (off-nominal) scenarios;
- Evaluation of security architectures supporting higher autonomy of spacecraft operations and that satisfy operational requirements defined by ESOC.

1.4 Programme Outline

Phase 1) Key management fundamental requirements:
- Analyze space operations concepts and justify a key hierarchy consisting of two or more levels, where the lower level would be session or traffic keys, and the next level up would be the key encryption keys (KEKs) and private keys; Investigate the need for additional levels in the key hierarchy (e.g. to re-key KEKs).
- Research and define the keys scopes, i.e. keys required for authentication and encryption of both TC and TM links and their required secure channels to control the security modules; for the analysis of TM keys it should be considered that platform and payload TM links can either be sharing the same space link or using separate space links.
- Research and define secure instrument/payload data distribution to users (directly and indirectly); consider authentication, confidentiality, access control from the point of view of both spacecraft operator and user(s).
- Research and specify key roles, key lengths and corresponding crypto periods, with credible technical and scientific substantiation for the selected values.

- Investigate a concept for over-the-air-re-keying (OTAR) using a secured channel, in order to satisfy current missions requirements which involve full ground control, as currently favored by the ESA Space Operations Centre (ESOC).

**Phase 2) Trusted Platform research:**

This phase consists of the investigation into how a trusted platform can perform key management tasks to temporarily replace KDCs, TAs, and CAs, respectively for pre-distribution, arbitrated key and self-enforcing schemes. In this phase two candidate platforms will be selected: one to be employed in LEO/GEO orbits, and a second one tailored for deep space missions. Important criteria in the selection of the key management scheme are: reduced onboard storage, minimal bandwidth, and minimal protocols iterations for key management, low requirements in computational power and energy consumption. Furthermore, scalability and temporary autonomy are also important parameters to support different mission architectures with varying number of spacecrafts and ground control centers. Regardless of the key management scheme considered, autonomous operations of spacecraft will be considered along with the investigation of each key management scheme. The goal of higher autonomy is to address mission’s requirements of keeping a minimum contact with ground control centers. Although complete autonomy may not be directly applicable to current missions concepts, certain levels of spacecraft autonomy may be necessary supporting temporary independent operations from a control center, e.g. in contingency situations.

a) Pre-distribution scheme:

- Investigate how an onboard trusted platform can be utilized to temporarily replace a ground KDC, and the operational and implementation requirements involved.

- In order to reduce storage requirements, investigate methods to perform key derivation onboard, along with the associated key synchronization.

- Research a secure means to perform multi-stage post-launch key-distribution in order to achieve higher flexibility and longer lifetime. The control center would utilize the trusted platform to store cryptographic keys for subsequent distribution to other spacecraft. This approach would be employed when the control center cannot reach all the spacecrafts in its domain, e.g. rovers on Mars’ face opposite to Earth. Similar to a store-and-forward approach, one reachable spacecraft (e.g. a satellite orbiting Mars) would be delegated to perform key management on behalf of the control center, therefore allowing almost continuous access to unreachable spacecrafts (e.g. rovers).

b) Arbitrated key scheme:

- Determine how a ground-based TA could be replaced by an onboard trusted platform.

- Analyze the onboard storage requirement for such an approach. Although the computational complexities of SKC primitives are usually lower than the ones required by PKC, the former may demand higher storage requirements to store long lived and session keys for multiple entities in the network.
- Evaluate appropriate key distribution protocols that could be employed for key distribution between control center and spacecraft as well as between spacecrafts, which considers the utilization of a trusted platform onboard.

c) Self-enforcing scheme:
- Research the operational capabilities that a trusted platform should provide to successfully substitute a ground-based CA on a temporary basis

Phase 3) Fail-safe, reconfigurable and distributed key management:

Based upon the two approaches selected in Phase 2, research fail-safe mechanisms for the proposed key management infrastructure, and investigate reconfiguration mechanisms for the trusted platform.

- Investigate how to achieve higher reliability through the efficient and secure utilization of mirrors of the trusted platform onboard other neighboring spacecrafts. The boundaries covered by each trusted platform as well as the level of redundancy involved are subject of research.

- Research schemes for distributed key management involving neighboring spacecraft. Secret sharing schemes are good candidates to perform key agreements based on virtual conferences.

- Explore means to automatically issue temporary keys in contingency situations therefore keeping a secured channel with the mission control system at all times;

- Investigate secure means for the control center to quickly take key management control back from the trusted platform, e.g. when coming out of contingency situations where the spacecraft has temporarily proceeded with autonomous operations.

- Contingency situations in the ground-segment demands yet another mission architecture, which includes control center redundancy. Therefore, the support of such a mission architecture is another point of investigation in this research.

- Revisit and improve the trusted platform to permit secure reconfiguration of algorithms and secure storage, therefore allowing for longer lifetime and higher flexibility of the onboard key management system.
1.5 Output

- Technical Notes/Reports
  
i. Marcio Juliato, Cathy Gebotys, Ignacio Aguilar Sanchez “PHASE 1 SUMMARY: KEY MANAGEMENT FUNDAMENTAL REQUIREMENTS”, 2012, 71pgs

  Focuses on security issues (key hierarchy, lengths, lifetimes, secure data distribution) for the GEO/LEO data link level (specifically TC, TM housekeeping and TM payload links only) to ensure a secure channel between the SC and CC. Unlike previous research, trust/threat models for the space scenario and quantitatively-justified key lengths and cryptoperiods (including CER) are presented. Different schemes for data distribution including updating user’s keys on the SC with or without ESA knowledge of the keys is also presented.

  

  Various mathematical formulations were researched in order to quantify the cryptoperiod for different links and orbits. Based upon the maximum number of frames to be transmitted, as well as upon the datarate of each specific link, it was possible to determine the maximum time frame that a given cryptographic key should be utilized, i.e. the cryptoperiod for a specific link. Additionally, instead of computing cryptoperiods in terms of PErr (for a set of frames), a mathematical formulation based upon Pue (single frames) was also created. This change of approach was crucial to allow for a better analysis involving number of frames to be transmitted, tag lengths, and different Pue. Unlike the TC link where requirements for the undetected error rates in frames are specified, there were no requirements found in the literature for the TM link. Hence assumptions of equivalent undetected error rates were made. Conclusions for the TC link indicated that authentication keys could live long enough to protect the vast majority of missions. Strong conclusions could not be accurately drawn for TM frames given the weak assumptions which had to be made. However the researched formulations for TM indicated the performance of the coding mechanism (RS) was too high to be matched by the security mechanisms utilizing the current key sizes.

  Trusted platform modules are introduced for key management in a constellation of SCs. The use of a Master SC, sets of secret keys, and one time secrets is introduced. Threat models and data distribution algorithms are redefined for this scenario supporting higher levels of autonomy. A complexity evaluation of these schemes utilizing a model is also performed showing the impact on on-board memory, SC security operations and communication bandwidth.

  

  Optimizes phase 2 key transport protocols with fewer messages and passes. Additionally time-out capabilities are incorporated (for improved resistance to attacks). Password based authenticated key exchange is utilized, with the OTS
concept to index the password, in order to generate keys onboard the SC. Extensions with time stamps are provided. Mechanisms for reporting generated keys to CC are provided. Techniques are quantitatively compared with storage and bandwidth. Redundancy is introduced to the concept of in-orbit key distribution centers, supporting two and multiple MSCs.

- **Research Papers:**
  
  i. M. Juliato, C. Gebotys, I. Aguilar Sanchez “Secure and efficient symmetric-key transport for satellite constellations” , 6th International Workshop on Tracking, Telemetry and Command systems for Space Applications (TTC’13), September 2013, ESOC, Darmstadt, Germany, 8 pgs.
  

### 1.6 Relevant References

The following list of relevant references for the project is provided (additional references are listed in each Phase 1,2,3 project report):


- GMES Sentinels Security Operations Procedures (SecOps) (TBC);

- TM&TC System Security Design Study (selection of documents of two parallel ESA study contracts); technical reports covering the following:
  
  o Reference System Architecture;
  
  o Development and Implementation Aspects (HW/SW);
  
  o Operations Aspects.


- NIST, “Part 1, Recommendation for Key Management - Part 1: General” (Revised), SP 800-57.
- NIST, “Recommendation for Key Derivation Using Pseudorandom Functions” (Revised), SP 800-108.
- NIST, “Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths”, SP-800-131A
Chapter 2 : Key Management Fundamental Requirements
2.1 Scope

This chapter is divided into six main sections covering fundamental requirements for key management.

Section 2.2 presents a set of assumptions concerning the communications parameters for missions in Low Earth Orbit (LEO) and Geostationary Earth Orbit (GEO). It includes orbital types and associated parameters, and several communication scenarios.

Section 2.3 presents the roles and scopes of cryptographic keys, as well as provides a justification for a proposed key hierarchy.

Section 2.4 presents the security requirements and assumptions for two reference missions: Sentinel and Meteosat. In addition, it presents the algorithms used in those missions, as well as the modes of operation considered for encryption, authentication, and authenticated encryption. The threat and trust models considered in this research are also introduced. Next, a set of attacks that could be launched against the Advanced Encryption Standard (AES) while in use by space missions is presented.

Section 2.5 proposes a method to determine cryptoperiods and key lengths, based upon time and data complexity of the aforementioned attacks against AES. This method takes into account the increasing availability of computational power to attackers and potential scenarios for cryptanalytic advances.

Section 2.6 describes some scenarios for payload data distribution, and presents a strategy to distribute payload data to users who have their payloads onboard spacecrafts controlled by ESA.

Section 2.7 proposes a concept to perform over-the-air rekeying for TC, TM HK, and TM PL links of both Sentinels and Meteosat missions.
2.2 Space Mission Scenarios

Key management techniques are directly impacted by several mission-specific parameters. In order to set a common ground for the remainder of this document, this section presents the assumptions made for orbital types and associated parameters, communication features, and operational scenarios.

2.2.1 Orbital Types and Associated Parameters

Even though the results of this research can be extended and applied to virtually any type of mission, the results presented in this document are based on Earth observation missions. The reason being is that implementation details of ESA’s Earth observation missions are generally more accessible compared to other types of missions. For instance, navigation missions such as the one involved with Galileo satellites have strong access restrictions to crucial information, which would make the development of this research impossible. In any case, Earth observation missions are paving the way towards the implementation of security mechanisms onboard spacecrafts.

Earth observation missions may implement security functions for individual virtual channels within the TC and TM data links. Without loss of generality, it is assumed in this research that security mechanisms are implemented on the data link level. In other words, they are applied to the TC, TM Housekeeping (HK) and TM Payload (PL) links.

Two types of orbits are considered in this research: Low Earth Orbit (LEO) and Geostationary Orbits (GEO). Communications parameters considered for Sentinels follow standards proposed by the Consultative Committee for Space Data Systems (CCSDS). TC and TM specifications are presented respectively in [CCSDS2000G6] and [CCSDS1000G1].

2.2.1.1 Low Earth Orbit

The Sentinel series of satellites has been used in this research as a reference LEO mission. The size of data frames used by the TC data link layer is 1kb (1024 bits), which is the maximum frame size specified in [CCSDS2320B2]. The data rate utilized by Sentinels’ TC subsystem is 64kbps under normal conditions, which can be reduced to 1kbps under emergency scenarios. The TM link is used for transmitting both HK and PL data, whose data frame size is 8kb (8192 bits). The data rate of the HK channel is 2Mbps, whereas the PL utilizes two channels that can reach up to 260Mbps each, therefore adding up to 520Mbps.

2.2.1.2 Geostationary Earth Orbit

The third generation of Meteosat satellites is used as a GEO reference mission. The data frame sizes is exactly the same as the ones utilized in the Sentinels. Precisely, the TC data link utilizes 1kb frames, whereas the TM HK and PL use 8kb. The TC link has a data rate of 4 kbps under normal operations, which can be reduced to 1 kbps for emergency commanding. The data rate of TM HK is 2Mbps, whereas the TM PL link is capable of transmitting data at 2Gbps.
2.2.2 Communication Scenarios

Communications scenarios play an important role in cryptographic key management. Factors influencing cryptoperiods include communication data rates, data frame sizes, contact times with spacecraft. Besides, different types of operational scenarios should be taken into account in the determination of cryptoperiods. Operational scenarios covered in this research include normal and emergency contexts, as well as fixed and distributed attackers trying to communicate with the SC.

2.2.2.1 Low Earth Orbit

Four scenarios can be devised for LEO missions: normal operation, emergency commanding, attacker at a fixed physical location, and multiple attackers distributed around the globe.

**Normal Operations:** In this scenario the system is operating at nominal conditions, and the CC has complete control over the SC. It is assumed that the CC utilizes only one Ground Station (GS) to communicate with the SC. Each overpass lasts approximately 10 minutes. Even though there may be up to 40 overpasses per day, a commanding campaign is executed in a single overpass. One live TM HK session per day is performed, which is also done within a single overpass. Therefore, the maximum daily communication time of both TC and TM HK between the CC and the SC is 10 minutes. In the worst case scenario, a TM PL link would transmit data in every overpass. This would result in a total contact time of 400 minutes per day.

**Emergency Commanding:** Some off-nominal scenarios may require the utilization of emergency commanding. This can happen in the event of a failure or when the SC enters into safe-mode. In this mode of operation, the data rates of the communication links are reduced in order to minimize the signal to noise ratio. As a consequence, there is a higher chance of a command being correctly received by the SC. Additionally, the CC may continuously send a stream of commands in the hope that the SC receives at least one of them. Assume a daily commanding campaign over 40 overpasses, where each overpass lasts 10 minutes. Then, a total communication time with the SC through the TC link is 400 minutes per day. If the SC responds with telemetry, that would be done in a single 10-minute overpass.

**Fixed Attacker:** An attacker in a fixed geographical position would have the same contact time with the SC as a CC utilizing a single GS. However, such an attacker could send a stream of commands during each SC overpass. Thus, considering 40 overpasses per day, the attacker could have a daily window of 400 minutes to communicate with the SC through the TC link. Notice that this is similar to the emergency commanding scenario. However, since the SC is not in safe-mode, the speed of the TC link is kept as nominal. In the case of TM data, an attacker would observe the same amount of data as a valid CC. Precisely, in the case of TM HK, an attacker would be able to collect TM HK data during a 10-minute overpass in which live telemetry is sent to the CC. The attacker could listen to the TM PL link up to 400 minutes, in the worst case scenario.

**Distributed Attacker(s):** An attacker could utilize multiple GSs distributed across the globe and even join other attackers in other geographic location to increase the communication time with the SC. In either case, the attacker(s) would access the TC link of the SC for 24h (1440 minutes) per day. For both TM HK and PL, the scenario would be the...
same as the normal operations and fixed attacker. So, the access time to TM HK data would be 10 minutes per day, whereas the worst case scenario for TM PL would be 400 minutes per day.

### 2.2.2.2 Geostationary Earth Orbit

For GEO the same communication scenarios are assumed as described for LEO, except for the distributed attacker. The reason being is that a SC in GEO orbit could be in constant line of sight, depending on the geographical position of the attacker. Thus, for GEO satellites a fixed attacker would have the same contact time as a distributed attacker in the LEO case.

**Normal Operations:** Even though SC is always in the GS line of sight, the CC is expected to have approximately 10 minutes of TC and TM HK communications per day during normal operations. In the case of TM PL, it may be the case that the SC downloads data for 24 h (1440 minutes) per day.

**Emergency Commanding:** Emergency commanding is a scenario where the CC may sent a continuous stream of commands to the SC during the whole day in the hope that at least one of them is heard. In this case, the contact time for TC may be up to 1440 minutes per day. If TM HK is sent to the CC, it would happen in the aforementioned 10-minute window. Moreover, this scenario does not expected TM PL traffic to occur.

**Attacker:** An attacker could continuously (24h/day) send commands to the SC in GEO, which is similar to a CC’s emergency commanding. The attacker could also listen to the 10-min TM HK traffic which happens once a day. In addition, TM PL data could be listened for 24h, assuming a constant traffic through that link.

### 2.2.3 Summary of Parameters

The parameters described in Sections 2.2.1 and 2.2.2 are summarized in Table 1. Those parameters allow for the computation of the total amount of data transferred per day, which is an important factor to be considered in the determination of cryptoperiods.

#### Table 1: Parameters for Different Space Missions Scenarios

<table>
<thead>
<tr>
<th>Mission Type</th>
<th>Link Type</th>
<th>Scenario</th>
<th>Data rate [kbps]</th>
<th>Frame size [bits]</th>
<th>Frame rate [frames/s]</th>
<th>Contact time [min/day]</th>
<th>Contact time [s/day]</th>
<th>Contact time [frames/day]</th>
<th>Transfer Rate [frames/day]</th>
<th>Transfer Rate [MB/day]</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEO TC</td>
<td>Nominal</td>
<td>64</td>
<td>1024</td>
<td>64</td>
<td>10</td>
<td>600</td>
<td>38400</td>
<td>4.6875</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LEO TC</td>
<td>Emergency</td>
<td>1</td>
<td>1024</td>
<td>1</td>
<td>400</td>
<td>24000</td>
<td>24000</td>
<td>2.9296875</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LEO TC</td>
<td>Fixed Attacker</td>
<td>64</td>
<td>1024</td>
<td>64</td>
<td>400</td>
<td>24000</td>
<td>1536000</td>
<td>187.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LEO TC</td>
<td>Distributed Attacker</td>
<td>64</td>
<td>1024</td>
<td>64</td>
<td>1440</td>
<td>86400</td>
<td>5529600</td>
<td>675</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LEO TC</td>
<td>Nominal/Attacker</td>
<td>2048</td>
<td>8192</td>
<td>102400</td>
<td>10</td>
<td>600</td>
<td>1536000</td>
<td>150</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LEO TC</td>
<td>Nominal/Attacker</td>
<td>819200</td>
<td>8192</td>
<td>8192</td>
<td>400</td>
<td>24000</td>
<td>2457600000</td>
<td>240000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GEO TC</td>
<td>Nominal</td>
<td>4</td>
<td>1024</td>
<td>1</td>
<td>600</td>
<td>2400</td>
<td>24000</td>
<td>0.29296875</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GEO TC</td>
<td>Emergency</td>
<td>1</td>
<td>1024</td>
<td>1</td>
<td>1440</td>
<td>86400</td>
<td>86400</td>
<td>10.546875</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GEO TC</td>
<td>Attacker</td>
<td>4</td>
<td>1024</td>
<td>4</td>
<td>1440</td>
<td>86400</td>
<td>345600</td>
<td>42.1875</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GEO TC</td>
<td>Nominal/Attacker</td>
<td>2048</td>
<td>8192</td>
<td>256</td>
<td>10</td>
<td>600</td>
<td>1536000</td>
<td>150</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GEO TC</td>
<td>Nominal/Attacker</td>
<td>2097152</td>
<td>8192</td>
<td>262144</td>
<td>1440</td>
<td>86400</td>
<td>22649241600</td>
<td>22118400</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
2.3 Key Roles, Scopes and Hierarchy

This section defines the scope of the keys, therefore showing in which link or subsystem they are used. The roles of cryptographic keys are also determined, i.e. the security functions used in space missions requiring cryptographic keys. In addition, a multi-level key hierarchy is presented to support various security architectures.

2.3.1 Determining Key Roles and Scopes

The security requirements of Sentinels demand authentication for TC and TM HK links. A set of session keys (SKs) is stored onboard and is provided to the authentication mechanism when required. The SKs table is updated every six months. Key transport [NISTSP80057P1] is performed through encrypted and authenticated data packets, in a technique known as key wrapping [2001NISTKWS]. This mechanism demands the utilization of key encryption keys (KEKs). In Sentinels case, the KEKs are also called master keys (MKs), since there is no higher level in the key hierarchy of this mission. The set of KEKs are load onto the SC right before launch and cannot be updated. Since a KEK is used only once, it is necessary to ensure that the SC will have enough keys onboard to cover the entire mission lifetime.

Additionally, authentication of TC and TM links can be disabled through a separate authenticated channel between the CC and the security modules onboard. Notice that authentication of the special channel between the CC and security modules is never disabled. The commands for disabling authentication are authenticated through a set of special keys called Security Module Management Keys (SMKs), which are also called MKs in Sentinel's case. Likewise KEKs, the set of SMKs are also stored onboard before launch and are never updated.

Similarly to the Sentinels, the third generation of Meteosat satellites requires its TC link to be authenticated. In addition, its TM PL link demands authentication and encryption. On the other hand, its TM HK link does not require any security mechanism. Upload of new session keys, as well as the disabling of security mechanisms are performed exactly the same as in the Sentinels' case.

There are other key roles that can be devised. As reported in item IV of Appendix C, the spacecraft recovery mechanisms may require the utilization of one time secrets (OTSs). Even though these secrets are not cryptographic keys themselves, they should be treated as if they were. The reason being is that they are used to perform trusted resets on the onboard computational platform, as well as used to issue temporary keys in emergency situations.

In addition, secure pre-launch key upload might require the utilization of keyed hash functions to check the integrity of key tables. This issue has been described in item V of Appendix C. The same strategy may be utilized to check the integrity of secret tables onboard the SC, which are a fundamental point of trust in recovery mechanisms.

Furthermore, item VI of Appendix C describes the benefits of performing onboard key derivation. Key Derivation Functions (KDFs) utilize a key along with other information to generate new keys. This strategy could be used, for instance, to generate new MKs, so that the risk of running out of this kind of key is eliminated. In addition, key derivation may benefit users with payloads onboard ESA's SCs, who are in charge of performing key
management. This strategy would minimize the traffic through the TC channel, since it would be no longer necessary to upload large blocks of SKs. In this case, only a few commands would be necessary to perform key derivation. Even though the management of these additional key roles are within the scope of this research project, they not addressed in this document.

Therefore, the identified key scope comprises the security mechanisms for TC, TM HK, TM PL links. In addition, keys may be necessary to manage the onboard security modules and recovery platform. Specific key roles include authentication and encryption functions, enabling/disabling of security mechanisms, integrity checking of onboard tables, onboard key derivation, and the secure recovery of onboard cryptographic platforms. Table 2 summarizes the interdependence between the key roles and scopes.

<table>
<thead>
<tr>
<th>Table 2</th>
<th>Interdependence between Key Roles and Scopes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key Roles</td>
<td>TC</td>
</tr>
<tr>
<td>Encryption</td>
<td>✓</td>
</tr>
<tr>
<td>Authentication</td>
<td>✓</td>
</tr>
<tr>
<td>Keyed Hash</td>
<td>✓</td>
</tr>
<tr>
<td>Key Derivation/Transport</td>
<td>✓</td>
</tr>
<tr>
<td>Security Management</td>
<td>✓</td>
</tr>
<tr>
<td>Platform Recovery</td>
<td>✓</td>
</tr>
</tbody>
</table>

As can be noticed, the aforementioned missions do not require full security for all communication links, and utilize only a small subset of security mechanisms that are currently available. Sentinel satellites, for instance, utilize authenticated commanding, but not encryption. However, Meteosat is expected to have an increasing demand for security in upcoming missions. This issue will become especially important if a member state passes a law that ends up requiring full security, e.g. the case of German government law on Earth observation. This point is addressed in item VII of Appendix C. Therefore, this study could be applied not only to the security requirements of Sentinels and Meteosat, but also easily expanded to future missions requiring full security.

### 2.3.2 Proposed Key Hierarchy and Justification

Based upon the key scopes and roles described in the previous section, it is possible to justify the demand for some key hierarchy scenarios. Even though a complete key hierarchy is proposed in this section, mission-specific requirements may end up utilizing just a subset of it. Figure 1 shows a graphical representation of the proposed key hierarchy.

The bottom level in the hierarchy are the SKs, which are utilized to ensure confidentiality and authentication in routine communications between the SC and its CC. TC and TM HK links have relatively low bandwidth compared to TM PL links. Due to their limited data traffic, those links may not require keys to be changed very often. In the case of TM PL links, high data rates may be used. Thus, depending on the key cryptoperiods, it may be necessary to frequently change SKs. As a result, security mechanisms must receive a constant supply of SKs. This level can also include KHKs, which are utilized to perform integrity checks on onboard tables. Even though OTSs are not keys themselves, they should be treated as such. Therefore they are also included in this level. Actually, due to random
properties of OTS, this type of information could eventually be used as temporary keys in emergency scenarios.

Figure 1: Key Hierarchy based on (a) Key Transport and (b) Key Derivation

(a) The second level in the hierarchy comprises the keys that are utilized to supply SKs to the SC. Two techniques can be utilized: key transport and key derivation [NISTSP800108]. If key transport is used, a KEK is utilized to wrap the set of SKs, KHKs, or OTSs. The encrypted set of keys along with an authentication tag are sent to the SC. Notice that, besides encryption and authentication, this technique does not require any additional cryptographic mechanism onboard. Its drawback is the transmission of big blocks of cryptographic keys through the TC link. This is the strategy utilized to upload SKs to Sentinel and Meteosat satellites, where KEKs are usually denoted as MKs.

If key derivation is used, a set of key derivation keys (KDKs) becomes necessary. The drawback of this approach is the requirement of KDFs to be implemented onboard the SC. Also, the CC may want to confirm that key derivation procedures executed correctly, in order to guarantee that the CC and the SC shares the same keys. In spite of that, key derivation brings the benefit of minimizing traffic in the TC link. Besides, as described in item VI of Appendix C, this approach may benefit users with payload hosted on ESA’s SCs. Key derivation becomes specially interesting for TM PL link with high data rates, where frequent key changes are expected. Yet in this hierarchy level are the SMKs. As previously described these keys are used to communicate with the security modules onboard.

(b) The top layer corresponds to the update of KEKs, KDKs and SMKs. This layer would contain master keys (MKs) whose function is to provide support to the transport or derivation of KEKs, KDKs, and SMKs. Most mission requirements may not implement this level, if enough keys are stored onboard prior to launch. However, there are security and reliability benefits of implementing this level. First, every time that the SC enters into safe mode, the TC and TM authentication may be disabled, consequently consuming a SMK. Thus, the more the SC enters into safe mode, the faster it will drain the onboard SMKs.
Furthermore, the integrity and quantity of keys can suffer severe impact from electronics failures and radiation. If for some reason the number of SMKs is reduced, and the SC happens to enter into safe mode very often, then there is a high risk of running out of keys. Second, this layer provides extra flexibility to update key tables onboard, especially if keying information is leaked or compromised on the ground. Obviously, in this case, it would be necessary to impose stringent access control to MKs, given that they would become the ultimate point of trust of the security solution.

In addition, due to advances in cryptanalysis, the cryptoperiods may have to be significantly reduced in order to keep the system at a reasonable security level. Consequently, without the ability to update KEKs, KDKs, and SMKs, the SC would be at risk of running out of keys. Hence, the existence of MKs for mid-levels rekeying would present itself as an essential tool to keep up the space mission security until the end of its lifetime.

Hybrid approaches could be devised as well. For instance, performing key derivation of SKs, but transporting other keys such as KHKs and OTs. Some CCs may consider that it is too risky to derive KEKs, KDKs, and SMKs onboard. In that case, key transport would be preferable solution, which is the technique implemented in Sentinels and Meteosat missions.
2.4 Algorithms and Attacks

This section presents quantitative models of attack time, as well as clarification of assumption made for various threat and trust models. Next, it presents a list of attacks that could be launched against AES when this algorithm is employed in space missions.

2.4.1 Assumptions on Threat and Trust Models

A security solution is always associated with threat and trust models. The threat model specifies the attacker capabilities, therefore allowing for the determination of the vulnerabilities of the system. The trust model specifies the reliable points of a system that cannot be modified by attackers, upon which a security solution can be built.

The block cipher adopted by Sentinels and Meteosat missions is AES. This cipher is utilized in counter mode (AES-CTR) for encryption services. Message authentication codes (MACs) are generally used to achieve data origin authentication. This service is obtained by utilizing AES in cipher-based MAC mode (AES-CMAC). Authenticated encryption is yet another service provided by AES in Galois/Counter Mode of operation (AES-GCM). The AES-CTR, -CMAC and -GCM modes of operation are specified by the National Institute of Standards and Technology (NIST) [NISTSP80032A, NISTSP80032B, NISTSP80032D].

2.4.1.1 Threat Model

The threat model utilized in this research assumes that an attacker:

- Listens to all radio frequency (RF) signals used for communications between the GS and the SC, as well as can tap on the communication lines between the CC and the GS;
- Accumulates and stores data of previous communication sessions in order later utilize them in an attack;
- Exploits the latest attacks available in the literature against the security mechanisms implemented in a specific mission;
- Can send his/her own signals to CCs, GSs, and SCs, during and after communication sessions;
- Knows the data formats, protocols, algorithms and security mechanisms utilized in all communications;
- Does not have unlimited access to computation power, memory, and bandwidth;
- Can utilize distributed computing to increase his/her computational, storage, and communication capabilities.

2.4.1.2 Trust Model

The trusted model comprises the following:

- The agency controlling the SC is the same as the one that has developed and implemented it;
- Subsystem modules, including the security ones, may be provided by third parties. However, it is assumed that such modules conform to the agency’s specifications, and do not have any hidden back door, malware, or any evil functionality;
• It may be possible that a third party (company or agency) launches the SC;
• Cryptographic keys are stored in secure facilities and accessed only by authorized personnel;
• Cryptographic keys could get corrupted onboard, which may be caused by radiation and/or electronics faults. However, it is assumed that error detection and correction (EDAC) mechanisms are in place to mask out these faults;
• Cryptographic keys are generated on the ground in a secure environment, following appropriate standards, e.g. [NISTSP800133];
• Once the keys are generated, it is assumed that they will not be modified intentionally or accidentally during their storage on ground facilities;
• Cryptographic keys are uploaded to the SC in a secure environment by authorized personnel;
• Only authorized personnel has access to the CC control room to command the SC;
• System construction and testing is performed in appropriate and secure environment, so that an attacker cannot collect electromagnetic emanations to launch side channel attacks;
• After every session key transport, a procedure is executed in order to guarantee the synchronism between the SC and CC. New session keys become active only upon successful key confirmation;
• Security mechanisms algorithms and protocols are fully executed, e.g. reduce rounds attacks against a cipher will not happen.

2.4.2 Attacks against AES

The open literature lists several types of attacks against AES, with their respective set of assumptions. Some take into account that the cipher would not complete all its execution rounds, the so called, reduced-round attacks. Others may consider the injection of faults into the execution rounds, whereas others may take into account side channels, such as power consumption, electromagnetic emanations, etc. Given the trust model presented in Section 2.4.1, these types of attacks cannot be launched against a security mechanism onboard a SC in orbit. Hence they are not considered in this research.

Yet another type of attack is based on related keys. This attack should not be feasible if the CC in charge of generating cryptographic keys does it in a secure environment, following sound key generation techniques. That should also be the case for users generating their own cryptographic keys to communicate with their payloads. Related key attacks have been included in this section for the sake of comparison with other attacks. If the conditions of the trust model presented in Section 2.4.1 were met, this kind of attack would not apply to the space segment.

The main attacks against AES considered in this research are exhaustive search, biclique, and successful forgery attacks. Notice that the exhaustive search and the biclique attacks target the underlying cipher, and not the mode of operation in which the latter is employed. Therefore, these attacks could be virtually launched against any mode of operation. The successful forgery attack, in turn, can only be launched against authentication modes such as GCM and CMAC.
2.4.3 Related Key Attacks

Related-key attacks against the full round AES-256 were proposed in [2009BK] and [2009BKN]. In [2009BK] a related-key distinguisher is proposed and works for one out of $2^{32}$ keys with $2^{119}$ data and time complexity, and negligible memory. Hence, in order for the attack to work for every key, $2^{32}$ keys would be necessary. Consequently, the data and time complexity would become $2^{154}$. Also in [2009BK] a related-key recovery is proposed. This approach works for 1 out of every $2^{35}$ keys on average, and has $2^{96}$ data and time complexity, and demands $2^{65}$ memory. Thus, it has a $2^{131}$ data and time complexity.

The attack proposed in [2009BKN] includes both AES-192 and AES-256. More precisely, a related-key boomerang attack is proposed for AES-256, which utilizes four related keys, and has $2^{99.5}$ data and time complexity, and $2^{77}$ memory complexity. A related-key amplified boomerang attack is proposed for AES-192, which has $2^{123}$ data, $2^{176}$ time, and $2^{152}$ memory complexity.

Even if an attacker had enough memory available to launch this attack, she/he would also have to satisfy two other requirements: 1) listen to the communication for a period of time so that enough ciphertexts are collected; 2) spend a certain amount of time searching for the key. Estimates on the time necessary to collect data and perform the search are addressed in Sections 2.5.2 and 2.5.3.

2.4.4 Exhaustive Search Attacks

An attacker can always launch an exhaustive key search in the hope that he/she succeeds in decrypting a ciphertext. Considering that all possible $m$-bit keys would have to be tried, a total of $2^m$ operations are necessary.

In [2011BKR] a technique based on bicliques is proposed, whose goal is to reduce the complexity of attacking AES. Such a technique does not need to consider related keys. The computational complexity of attacking AES-128, AES-192, AES-256 are respectively $2^{126.1}$, $2^{189.7}$, $2^{254.4}$. Similarly to the aforementioned exhaustive search attack, results based on the reduced security of AES due to the biclique attack again considers an attacker with both constant and increasing computational power.

The total time needed to execute attacks based on exhaustive search, for different types of orbits and links, is presented in Section 2.5.4. The results are based on three main scenarios: 1) the attacker does not increment her/his computing power during the attack; 2) the attacker takes advantage of advances in computing (based on Moore's law) to speedup her/his attack; 3) the attacker takes advantage of Moore's law as well as of advances in cryptanalysis.

2.4.4.1 Estimates on Computational Power

The total time to perform an exhaustive search is directly dependent on the computational power of the attacker. Several attack scenarios can be devised, for example, a person using a single computer, an entity utilizing distributed computing, a national government using a supercomputer, or even a distributed attack using a set of networked smart phones. Evidently, many other examples could be listed.
This research takes into account some computational power scenarios, which are listed below, in order to allow for the estimates of approximate attack times. Notice that, some of the data originally listed their performance in floating-point operations per second (FLOPS). However, as cryptographic computations usually not utilize floating point operations, an approximation is made so that FLOPS are considered as instructions per second (IPS). The computational power of each example is listed as million instructions per second (MIPS).

*ESA assumption for Sentinels:* $6 \times 10^4$ MIPS

*Supercomputers:*
  - K Computer (Japan): $10.51 \times 10^9$ MIPS [2011T500]
  - NUDT YH MPP (China): $2.566 \times 10^9$ MIPS [2011T500]
  - Cray XT5-HE (US): $1.759 \times 10^9$ MIPS [2011T500]

*Berkely Open Infrastructure for Network Computing (BOINC):* $5.628 \times 10^9$ MIPS (Total) [2012BOINJC]

*Personal computers:*
  - Frequency of operation of an average desktop processor:
    - AMD Athlon X2 (dual core): $2.3$GHz => $4.6 \times 10^3$ MIPS [2012AMD]
    - Intel i5-2550K (quad core): $3.4$GHz => $13.6 \times 10^3$ MIPS [2012Intel]
  
  Number of computers in US: 164.1 million [2012aneki]
  Total computational power in the US: approximate range $7.55 \times 10^{11} – 2.23 \times 10^{12}$ MIPS

  Number of computers in China: 15.9 million [2012aneki]
  Total computational power in China: approximate range $7.3 \times 10^{10} – 2.16 \times 10^{11}$ MIPS

  Number of computers infected in China in Dec/2011: 4 million [2012CD]
  Total computational power infected in China: $1.84 \times 10^{10} – 5.44 \times 10^{10}$ MIPS

*iPhones/iPods:*
  - Number of iPhones that have been sold in the world: 73.7 million [2011How,2011Apple]
    Frequency of operation: 800MHz => $800$ MIPS
    Total computing power in the world: $5.89 \times 10^{10}$ MIPS

  Number of iPods sold in the world: 1 billion [2010Apple]
  Frequency of operation: 80MHz => $80$MIPS
Total computing power in the world: $8 \times 10^{10}$ MIPS

Total computing power of iPhones and iPods in the world: $13.89 \times 10^{10}$ MIPS

2.4.4.2 Effects of Moore’s Law

The assumption that a brute force attack against a cryptosystem is always performed with constant computational power may produce misleading results. The reason being is an attacker can benefit from Moore’s law and double his/her computational power every 1.5 years. As shown below, Moore’s law has a profound impact in the total time spent to break a cryptosystem. From another perspective, this would correspond to an attacker with fixed computational power attacking a cryptosystem that loses a bit of security every 1.5 years. Therefore, it is currently assumed that a more realistic attack scenario is obtained by considering the Moore’s Law in the scope of this work, unlike previous research in this area. As a consequence, we can estimate how much computational power an attacker may have at some point in the future and the total computation performed up to a certain year. Hence, it is possible to determine how much time it would take for an attacker to break a cryptosystem considering that he/she is benefiting from increased computational power available throughout the years.

The discussions presented in this section are based on the following variables:

- $d$: Moore’s computational power gain [Constant=2];
- $p$: Period for Moore’s computational power gain [Constant=1.5 years];
- $y'$: current year;
- $y$: target year for the computational power estimate;
- $c'$: current (initial) computational power [instructions per year];
- $c$: estimated computational power in a given year [instructions per year];
- $t$: period of time performing computation ($t = y - y'$) [years];
- $g$: gain in computational power in a period of time;
- $G$: total computational power gain accumulated over the years;
- $T$: total computation performed over the years;
- $E$: computational effort required to perform a exhaustive attack;
- $m$: number of bits of a cryptographic key.

In order to define the total amount of computation performed ($E$) over a period of time ($t$), let’s assume that an attacker has fixed amount of computational power ($c$). Thus, $E$ is given by $E = ct$. Besides, the total time of a traditional exhaustive search would be proportional to $2^m$, i.e. $2^m = ct$. Therefore, the total time of a traditional search is defined by:

$$t = 2^m / c.$$  \hspace{1cm} (Equation 2.1)

By the same token, if certain data is to be protected for $t$ years against an attacker with computational power $c$, the security level should be at least $m$ bits. The number of bits of security level is then defined by:

$$m = \log_2(ct).$$  \hspace{1cm} (Equation 2.2)

An attacker could follow Moore’s law to increase his/her computational. In this case, his/her computational power would double every 1.5 years. Thus, an attacker starting the first
year with computational power \( c \) would have \( 2c \) in the end of 1.5 years, \( 4c \) in the end of 3 years, \( 8c \) in the end of 4.5 years and so on.

The computational power gained (\( g \)) in a number of years would be \( g = d^{(\delta p)} \). So, the estimated computational power available in a specific year is given by \( c = g'c' \). In the case of Moore’s law, \( g = 2^{(\delta t)} \).

In order to determine the computational power gain accumulated from the present moment up to \( t \) years in the future it is necessary to integrate \( g = 2^{(\delta t)} \) over the period of time \([0,t]\).

So, \( G = \int_0^t 2^{(\delta t)} \, dt \)

Thus, \( G = (1.5 / \ln(2)) * (2^{(\delta t)} - 1) \).

Furthermore, the total computation performed in \( t \) years is given by \( T = G*c' \).

Hence, \( T = (1.5 / \ln(2)) * (2^{(\delta t)} - 1) * c' \).

The computational effort to break an algorithm through exhaustive search can be determined by \( E = 2^m \).

Since \( E \) is equivalent to \( T \), \( E = T \).

Therefore, \( 2^m = (1.5 / \ln(2)) * (2^{(\delta t)} - 1) * c' \).

This equation can be interpreted in two ways. First, given the size \( m \) of a key, it is possible to determine how many years \( t \) would be necessary to break the system. Second, assuming that the data has to be protected for \( t \) years, it is possible to define the size of the key \( (m) \).

Thus, given the size of the key \( (m) \), the number of years that are necessary to break the system is given by:

\[ t = 1.5 * \log_2((\ln(2) * 2^m) / (1.5c)) + 1) \]  

(Equation 2.3)

On the other hand, if the system is to be protected for \( t \) years, the key size should be:

\[ m = \log_2((1.5c) / \ln(2)) + \log_2(2^{(\delta t)} - 1) \]  

(Equation 2.4)

Figure 2 illustrates the computational power gain, and the total computation performed by an attacker taking into account the computational power gain due to Moore’s law.
2.4.4.3 Cryptanalytic Advances

Even though it can be assumed that advances in cryptanalysis will continue, it is hard to predict when they will happen and the number of security bits compromised by certain technique. If cryptanalysis advances against the DES algorithm are considered, as shown in Table 3, it can be observed that several years have passed until the appearance of the first effective attack in 1992 [1992BS]. In 1994, the security of DES had been reduced to 44.66 bits [1994Mat]. No further advances against DES have been reported in the literature, probably due to the appearance of AES.

DES had 56 bits of security when it was released as an official NIST standard in 1977, and then 44.66 bits in 1994. Put differently, DES lost 11 bits of security in 17 years due to cryptanalytic advances. This corresponds to a loss of one bit every 1.18 years. Obviously, this was not uniformly distributed across the 17 years of cryptanalytic efforts. However, it is interesting to notice a decay ratio similar to Moore’s, but inversely proportional. For instance, the rightmost column of Table 3 lists the expected security of DES, if it had lost 1 bit every 1.5 years.

Table 3: DES Cryptanalytic History

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1977</td>
<td>DES as standard</td>
<td>56</td>
<td>56</td>
</tr>
<tr>
<td>1993</td>
<td>[1993Mat]</td>
<td>47</td>
<td>45.33</td>
</tr>
<tr>
<td>1994</td>
<td>[1994Mat]</td>
<td>43</td>
<td>44.66</td>
</tr>
</tbody>
</table>
Predicting cryptanalytic advance is a very hard, if not, an impossible task. Moreover, analyzing what happened to DES provides no evidence whatsoever that AES will follow the same trend. Since its release in 2001, the best attack against AES (without utilizing related keys) in the literature is the biclique cryptanalysis [2011BKR]. For instance, this attack reduces the security of AES-128 to 126.1 bits. In other words, after 10 years since its adoption as a NIST standard, the security of AES-128 has been reduced in only 2 bits due to cryptanalytic advances. Consequently, it seems at first that a cryptanalytic rate for AES determined by a ratio inversely proportional to Moore’s could be an extreme assumption.

Cryptanalytic advances usually happen in leaps. A new technique may lower the security of an algorithm in several bits followed by a hiatus of several years until a new method is proposed. It was possible to notice this by observing DES history. The initial years after its release have seen no significant crypto advances. However, after 17 years, its level of security was almost as low as if it had lost 1 bit every 1.5 years.

Although this topic may be subject to a long discussion, it is interesting to assume some cryptanalytic advance ratio in order to have quantitative estimates on the exhaustive search time. Let’s assume for now that cryptanalysis advances in such a way that 1 bit is lost every 1.5 years. Put differently, the total amount of effort to break the system is a function of time and halves every 1.5 years. The equation describing the total computational effort function of the year could be written as

\[ E = 2^{m - (t/1.5)} \]

As previously shown, the total computation \( T \) performed in \( t \) years is given by

\[ T = (1.5 / \ln(2)) \ast (2^{(t/1.5)} - 1) \ast c'. \]

The graph of \( E \) and \( T \) will intersect when \( E = T \), so

\[ 2^{(m - (t/1.5))} = (1.5 / \ln(2)) \ast (2^{(t/1.5)} - 1) \ast c'. \]

After taking the \( \log_2 \) of both sides of the equation, it is possible to determine the size of the key should be \( m \) bits to protect the data for \( t \) years. Therefore,

\[ m = \log_2((1.5c / \ln(2))) + \log_2(2^{(2t/1.5)} - 2^{(t/1.5)}). \] (Equation 2.5)

Notice that it is not possible to isolate \( t \) in the previous equation. Hence it is hard to determine the amount of time that the data will be protected given the utilization of an \( m \)-bit key.

It is possible, though, to make the following approximation:

\[ 2^{(2t/1.5)} - 2^{(t/1.5)} = 2^{(t/1.5)} \ast (2^{(t/1.5)} - 1) \approx 2^{(t/1.5)} \ast 2^{(t/1.5)} = 2^{(2t/1.5)}. \]

Hence, the approximated value \( m \) is defined by

\[ m = \log_2((1.5c / \ln(2))) + 2t/1.5. \] (Equation 2.6)

A numerical check can determine if such an approximation is reasonable. The computational power \( c \) varies with the assumptions on the attacker’s capabilities. Since \( c \) is involved in both equations, we can temporarily modify the equations to make the numerical check independent of \( c \).

Let’s take Equation 2.6 and define \( m_1 = m - \log_2((1.5c / \ln(2))) \).

So, \( m_1 = \log_2(2^{(2t/1.5)} - 2^{(t/1.5)}) \). (Equation 2.7)

Likewise, define \( m_2 = m - \log_2((1.5c / \ln(2))) \).

Then, \( m_2 = 2t/1.5 \). (Equation 2.8)
Equations 2.7 and 2.8 are plotted in Figure 3. It can be observed that for \( t \geq 7 \) both graphs are practically the same. Therefore, the aforementioned assumption is reasonable, for \( t \geq 7 \), which is usually the case for the vast majority of scenarios considered in this research.

Hence, it is assumed from now on that:

\[
m = \log_2\left(\frac{1.5c}{\ln(2)}\right) + 2t/1.5. \tag{Equation 2.9}
\]

Moreover, it is possible to determine \( t \) in function of \( m \). So, \( t \) is defined as

\[
t = \frac{1.5m}{2} - \frac{1.5}{2} \cdot \log\left(\frac{1.5c}{2}\right). \tag{Equation 2.10}
\]

2.4.4.4 Additional Parameters

The aforementioned equations take into account the most important parameters that could have a significant impact on cryptoperiods and key lengths. The equations, however, can be further expanded to include other parameters. For instance, it has been assumed that a full execution of AES is executed in a single clock cycle, which is not the case for software implementations. It was intrinsically assumed that the cost of a computing platform remains fixed, in spite of a doubling transistor density every 1.5 years. Besides, it was not taken into account that foundry manufacturing cost may increase to produce such a double-density chip. By the same token, it was not considered that the attacker’s budget may increase with time, as well as the rising costs of equipment due to inflation. The inclusion of these parameters into the aforementioned equations is left as future work.
2.4.5 Successful Forgery

In [ECRYPTII] (page 37) it is noted that besides the key length, the block size may also define upper bounds of security. A small block size may enable the creation of dictionaries; furthermore, non-random behavior, which might be exploitable in attacks, starts to appear after the encryption of around $2^{(b/2)}$ blocks (where $b$ is the number of bits in a block).

Specifically for authentication modes, is mentioned in [ECRYPTII] (page 40) that, “generally speaking, the quantitative measure of security obtained by these proofs decay with the number of message blocks processed under a given key. Security, in a strong sense, is lost as the number of processed blocks approaches $2^{(b/2)}$ for a $b$-bit block cipher. Thus, re-keying should occur well before this bound is reached. In practice, this is only an issue for 64-bit block ciphers”. Still in [ECRYPTII] (page 57), it is mentioned that “if a secure cipher with block length $b$ is used, and no more than $n$ blocks are authenticated under a given secret key, then the probability of successful forgery is bounded by $n^2/2^{(b-2)}$”, where $n$ is the number of blocks and $b$ the number of bits of a block. “For example, with 128-bit AES, if no more than $2^{43}$ blocks are authenticated – whether $2^{40}$ messages of 8 blocks or 8 messages of $2^{40}$ blocks – then the probability of successful forgery is bounded by $2^{36}/2^{126} = 1/2^{40}$.” Thus, the probability of a successful forgery ($P_{\text{Forg}}$) is given by:

$$P_{\text{Forg}} = \frac{n^2}{2^{(b-2)}}.$$

On the other hand, there always exists a probability of having an undetected error ($P_{\text{FrameErr}}$) in the reception of a TC frame. Consider that AES-encrypted data is being received by the TC subsystem. Since each AES block has 128 bits, each encrypted TC frame would comprise of 8 encrypted blocks. The probability of an undetected error in a block ($P_{\text{BlockErr}}$) is given by $P_{\text{BlockErr}} = P_{\text{FrameErr}}/8$.

One assumption that could be made is that the probability of a forgery should be lower or equal than the probability of an undetected block error ($P_{\text{Forg}} \leq P_{\text{BlockErr}}$). Therefore, $P_{\text{BlockErr}}$ would establish an upper bound for the forgery attack. As reported in [ECSSS5004C], this probability of frame error depends on the channel error rate (CER). The $P_{\text{FrameErr}}$ for a single TC frame (1024 bits) is listed below:

<table>
<thead>
<tr>
<th>Channel Bit Error Rate</th>
<th>PFrameErr</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^{-4}$</td>
<td>2.14x10^{-8}</td>
</tr>
<tr>
<td>$10^{-5}$</td>
<td>2.17x10^{-12}</td>
</tr>
<tr>
<td>$10^{-6}$</td>
<td>2.17x10^{-28}</td>
</tr>
</tbody>
</table>

Considering that the TC subsystem will receive $n$ blocks, then:

$$P_{\text{Forg}} \leq n * P_{\text{BlockErr}}$$

$$n^2/2^{(b-2)} \leq n * P_{\text{FrameErr}}/8.$$  

Thus,

$$n \leq 2^{(b-2) * P_{\text{FrameErr}}}/8.$$  \hspace{1cm} (Equation 2.11)

Taking the $P_{\text{FrameErr}}$ mentioned above as reference, it is possible to determine the maximum amount of data that could be transmitted so that $P_{\text{Forg}}$ is equivalent to $n^2P_{\text{BlockErr}}$. Moreover, by considering the data rate of each link, it is possible to compute the cryptoperiod.
of each communication link for LEO and GEO missions. This topic is addressed in Section 2.5.3.
2.5 Cryptoperiods and Key Lengths

This section provides a brief review of previous research, followed by quantitative analysis of cryptoperiods based on the qualitative analysis presented in Sections 2.4.3, 2.4.4 and 2.4.5, along with specific mission parameters introduced in Section 2.2. Estimates on the amount of time that an attacker would require to collect enough data and perform an exhaustive search are provided. These are crucial parameters for determining the key’s cryptoperiods.

2.5.1 Related Work

This section briefly reviews the state of the art in symmetric key management including key lengths and cryptoperiod specifications with a focus on satellites, and also provides a brief general overview of security in LEO satellites.

ECRYPT II [2007ECRYPTII] goes as far as providing a 3-star rating to AES all standardized key/block sizes as well as some other ciphers. Additionally CCM, CTR mode modes of operation and Encrypt then MAC achieves a 3-star rating. MACs achieving a 3-star rating include HMAC, CBC and CMAC. However authentication key lengths and cryptoperiods are not recommended. A website [2012BlueKrypt] offers guidance on some security parameters based on various references or formula. For example they suggest some key lengths and cryptoperiods based upon mathematical algorithms, but not algorithm flaws or hardware flaws. For example these suggestions include 128-bit symmetric encryption for 2012-2040 and 256-bit for beyond offering protection against quantum computers, according to ECRYPT. In contrast FNISA [2012BlueKrypt] recommends 128-bit keys for beyond 2020. Additionally the website describes NIST recommendations of 128-bit keys for beyond 2030 and 2 years or less as the cryptoperiod for key-encryption keys, encryption keys, authentication keys and approximately one year for master keys. Lenstra’s equations [2004Len] assumes that a security system is adequate if it costs $40M or more for a successful attack and defines for a symmetric cryptosystem, with \((56+b)\) -bit keys and no known weaknesses, adequate security until year \(1982 + y\) only if \(3b \geq 2y\). This computes a minimum sized 88-bit key as adequate security up to year 2030.

DeGregorio [2005Greg] computes the upper bounds to key lifetimes for typical failure rates, limiting the cryptoperiod of the key so that it is no longer used when the reliability of the system falls below a required level to avoid key exposure. This work is very interesting and may have significance to SEUs in satellites, however it more typically is based upon a larger number of faults, which risk key exposure.

For satellite applications the most relevant document is the recent Green Book released by CCSDS [CCSDS3506G1] describing key management options for space. It is recommended that no more than one function per key be supported. One, two, three and four level key hierarchies are described. In the one level key hierarchy a set of master keys is manually distributed and pre-burned into the SC before launch. This requires a significant amount of protected memory to store a sufficient number of keys for the mission lifetime. The two-level hierarchy consists of master keys which are used to derive or deliver traffic protection keys (TPKs). In the second level of the key hierarchy, the TPKs provide confidentiality and authentication (commonly referred to as session keys). Master keys may also act as key encryption keys in order to deliver the TPK keys. In a three-level hierarchy,
there are master keys, key encryption keys (KEKs) and traffic protection keys. Key encryption keys can be distributed through encryption with the master key or with a higher level KEK. In order to support user access to a specific data reception, one traffic protection key can be utilized for only one downlink session, however keys derived from the TPK can be utilized to encrypt individual data reception for each users access. A four-level key hierarchy is also suggested for user-owned payload modules on the SC. In this case users can send their encrypted TPKs to the operations control center, which embeds them into a TC sent to the user-owned payload module on the SC. A fall back set of master keys can be utilized for contingency operations. This set can be used to reload new sets of TPKs when the SC is back to normal operation.

The general determination of cryptoperiod involves a tradeoff of risk and consequences of exposure [CCSDS3506G1]. Although risk suggests shorter cryptoperiods, more frequent rekeying may in some cases increase the risk of exposure such as that resulting from human error or error in the rekeying process. In general cryptoperiods are not specified nor are key lengths, however general comments and examples are provided. In particular TM payload TPKs are to be replaced more frequently than TC TPKs. It is suggested that new payload TM TPKs be used for every data transfer session. There are long term and short term TPKs where short refers to a single message or communication session. Link encryption, where large amounts of data are encrypted over a short period of time requires encryption keys with a short cryptoperiod, such as a day or week. Keys derived from a master key are also expected to have single use or single communication session cryptoperiods. KEKs are envisioned to have long or short (a day or week) cryptoperiods depending upon whether they encrypt a small or large number of keys over a fixed period of time. Authentication keys are suggested to have single use only if they are protecting very sensitive information. Security commands include uploading of a new key, key revocation, and key switch.

Multicast security has been addressed for satellites through logical key hierarchy systems [2010TSR]. In these systems a tree of keys, including a root key, is utilized for customers located at the leaf nodes. Customers hold keys on the path from their leaf to the root node. Hence multicast is supported by encrypting data using the root key to all customers or alternatively to subgroups using a shared subgroup key in the tree. Thus key updates to groups in LKH are more efficient than utilization of a flat organization where each customer has a different key and there are no group keys. There is also research underway supporting key management for delay-tolerant networks, such as [2010EDS], which is a model for LEO satellites with multiple base stations. Group membership trees and their respective keys are utilized. KEKs as well as membership keys and public-private key pairs are involved.

Quantum key distribution has also been reported as practical with current technologies for satellites [20092009BTDNV], and may support both symmetric and asymmetric authentication keys [2011MBNM]. There are currently experiments underway to make quantum stream ciphers [2009HHAHOY] and quantum key distribution using photon entanglement for satellites [2012Hsu].

Although the ATV, GALILEO (altitude 23km MEO orbit), ALPHABUS, and SENTINELs-1,2,3 have already included security requirements, that is not a reality for most of the current ESA missions. Other LEO satellites include the EUMETSAT METOP polar orbiting satellite with security on downlink including encryption of virtual channel data unit with DES [CCSDS3500G2]. Additionally, the polar orbiting LEOs for photographic intelligence, or imaging satellites, electronic intelligence such as US Navy's white cloud
system (US KH-11s). NASA incorporates added cryptographic integrity checks, utilizing MD5, of an entire image with the Saratoga protocol on the LEO satellite [2010IPSEMTLHNJW] as well as authentication. There appear to be several on-board (LEO) satellite cryptographic hardware components. For example Astrium has units supporting: telecommand encryption (AES-128 to 256) and authentication; telemetry encryption; over the air rekeying and multi-key stores; TRANSEC key stream generation. It supports confidentiality, authentication and integrity, video encryption and has FIPS 140-2, FIPS 197, Common Criteria and COMSEC [2012Astrom]. Thales-Communication Security [2006Thales] also has support for LEO satellites: Telecommand authentication, integrity check, anti-replay protection and deciphering; high data rate payload Telemetry authentication, integrity check, anti-replay protection and ciphering; Key Management, In flight re-keying. Saab Ericsson Space management unit has on-board telecommand authentication, applicable to LEO [2005Saab]. Additionally SSTL-300-S1 imaging system (elevation 500km) supports encryption of all commands and telemetry, including payload data encryption in a 500Km orbit [SSTL].

In summary, although several satellites claim to support security, most do this by relaying already encrypted and authenticated or secure transmissions from a hub to remotes, supporting IP over satellites (e.g. end to end security). In contrast there appear to be few LEO satellites which actually perform encryption or authentication or cryptographic integrity computations on-board. Despite the fact that internet standard security protocols for terrestrial applications have not shown good performance in satellites [CCSDS3506G1], there remains limited research into satellite specific security. In particular cryptographic key management, described in numerous documents including [1996MOV] and [NISTSP80057P1, NISTSP80057P2] for terrestrial applications, has not been sufficiently addressed for secure satellite applications. Although some key sizes and cryptoperiods have been specified, apart from older formulas such as [2004Len], few recommended sizes are justified or applicable to satellite applications.

### 2.5.2 Data Complexity in Related Key Attacks

In order to perform a related key attack, an attacker would have to collect enough ciphertexts encrypted with related keys. As reported in Section 2.4.3, only AES-192 and 256 are subject to this attack, and have data complexities of $2^{123}$ and $2^{99.5}$, respectively. For instance, this would mean that $2^{123}$ bits of encrypted data would have to be collected in the case of AES-192.

The daily contact time of a given mission type, along with the data rate and frame size of each link, as listed in Table 1, allows for the computation of the total time for data collection. Table 5 shows the number of years required to collect enough data in each space mission scenario.
Table 5: Number of Years Required for Data Collection in a Related Key Attack

<table>
<thead>
<tr>
<th>Mission Type</th>
<th>Link Type</th>
<th>Scenario</th>
<th>Data Collection Time [years]</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEO</td>
<td>TC TC</td>
<td>Nominal</td>
<td>7.769E+26</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Emergency</td>
<td>1.243E+27</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Fixed Attacker</td>
<td>1.942E+25</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Distributed Attacker</td>
<td>5.3951E+24</td>
</tr>
<tr>
<td></td>
<td>HK TM</td>
<td>Nominal/Attacker</td>
<td>2.4278E+25</td>
</tr>
<tr>
<td></td>
<td>PL TM</td>
<td>Nominal/Attacker</td>
<td>1.5174E+21</td>
</tr>
<tr>
<td>GEO</td>
<td>TC TC</td>
<td>Nominal</td>
<td>1.243E+28</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Emergency</td>
<td>3.4529E+26</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Attacker</td>
<td>8.6322E+25</td>
</tr>
<tr>
<td></td>
<td>HK TM</td>
<td>Nominal/Attacker</td>
<td>2.4278E+25</td>
</tr>
<tr>
<td></td>
<td>PL TM</td>
<td>Nominal/Attacker</td>
<td>1.6465E+20</td>
</tr>
</tbody>
</table>

It can be noticed that the best case scenario for data collection happens in the PL TM link of GEO missions, where an attacker could listen 24h a day of data transmission. Even in this case, it would be necessary 1.39x10^{13} years for the acquisition of enough data so that an attack can be launched.

As previously mentioned, a sound key generation mechanism would avoid the existence of related keys. However, even if an attacker could manipulate the system to force it to use related keys, the number of years required to acquire data would make such an attack infeasible. Therefore, it is possible to conclude that related key attacks do not represent any threat to the space segment of LEO and GEO missions, and it will no further be considered in this document.

2.5.3 Data Complexity of a Successful Forgery

Prior to the determination of cryptoperiods for authentication modes, it necessary to define the maximum amount of data that the TC link could be transmit without violating the inequality \( n \leq 2^{(b-2)} \times P_{FrameErr} / 8 \). As Table 6 shows, for each channel error rate (CER), there is an associated probability of frame error. Given that each TC frame has 1024 bits and an AES block has 128 bits, each frame contains 8 data blocks. As a result, it is possible to compute the maximum number of blocks \((n)\), and consequently the maximum amount of encrypted data that a key should encrypt, as shown in Table 6.

Table 6: Maximum amount of data for a Successful Forgery

<table>
<thead>
<tr>
<th>Channel Error Rate</th>
<th>(P_{FrameErr})</th>
<th>(P_{BlockErr})</th>
<th>Max # of blocks ((n))</th>
<th>Max # of bits ((128*n))</th>
<th>Max # Tera Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>(10^{-3})</td>
<td>2.14x10^{-15}</td>
<td>2.675x10^{-17}</td>
<td>2.27564x10^{11}</td>
<td>2.91x10^{13}</td>
<td>3.64 x10^{9}</td>
</tr>
<tr>
<td>(10^{-5})</td>
<td>2.17x10^{-26}</td>
<td>2.7125x10^{-24}</td>
<td>2.30754x10^{10}</td>
<td>2.95x10^{12}</td>
<td>3692.064</td>
</tr>
<tr>
<td>(10^{-6})</td>
<td>2.17x10^{-28}</td>
<td>2.7125x10^{-26}</td>
<td>2.31x10^{9}</td>
<td>2.95x10^{11}</td>
<td>0.003692</td>
</tr>
</tbody>
</table>

Moreover, if the data rate of the various communication links are taken into account, it becomes possible to determine the amount of time that a key should be used for each individual type of link. Table 7 presents the cryptoperiods for CER, so that a successful forgery can be avoided.
### Table 7: Cryptoperiods to avoid a Successful Forgery

<table>
<thead>
<tr>
<th>Mission Type</th>
<th>Link Type</th>
<th>Scenario</th>
<th>Data Transfer [MB/day]</th>
<th>Channel Error Rate $10^4$</th>
<th>Channel Error Rate $10^5$</th>
<th>Channel Error Rate $10^6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEO</td>
<td>TC</td>
<td>Nominal</td>
<td>4.6875</td>
<td>7.76751E+14</td>
<td>7.88E+08</td>
<td>787.6403</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Emergency</td>
<td>2.9296875</td>
<td>1.2428E+15</td>
<td>1.26E+09</td>
<td>1260.224</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Fixed Attacker</td>
<td>187.5</td>
<td>1.94188E+13</td>
<td>19691006</td>
<td>19.69101</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Distributed Attacker</td>
<td>675</td>
<td>5.39411E+12</td>
<td>5469724</td>
<td>5.469724</td>
</tr>
<tr>
<td></td>
<td>HK TM</td>
<td>Nominal/Attacker</td>
<td>150</td>
<td>2.42735E+13</td>
<td>24613758</td>
<td>24.61376</td>
</tr>
<tr>
<td></td>
<td>PL TM</td>
<td>Nominal/Attacker</td>
<td>2400000</td>
<td>1517092219</td>
<td>1538.36</td>
<td>0.001538*</td>
</tr>
<tr>
<td>GEO</td>
<td>TC</td>
<td>Nominal</td>
<td>0.29296875</td>
<td>1.2428E+16</td>
<td>1.26E+10</td>
<td>12602.24</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Emergency</td>
<td>10.546875</td>
<td>3.45223E+14</td>
<td>3.5E+08</td>
<td>350.0623</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Attacker</td>
<td>42.1875</td>
<td>8.63057E+13</td>
<td>87515584</td>
<td>87.51558</td>
</tr>
<tr>
<td></td>
<td>HK TM</td>
<td>Nominal/Attacker</td>
<td>150</td>
<td>2.42735E+13</td>
<td>24613758</td>
<td>24.61376</td>
</tr>
<tr>
<td></td>
<td>PL TM</td>
<td>Nominal/Attacker</td>
<td>22118400</td>
<td>164615041.1</td>
<td>166.9227</td>
<td>0.000167**</td>
</tr>
</tbody>
</table>

*2 minutes and 13 seconds
**14.4 seconds

It can be observed that CER has a strong influence on cryptoperiods, which varies from the order of $10^{16}$ days to 14.4 seconds (0.000167 days). The reason being is the equivalence between probability of frame error and the probability of a successful forgery. Hence, the lower the CER, the higher should be security of the system to keep up with a very low probability of frame error. As expected, critical cases happen with CER=$10^{-6}$, in the case of the PL TM links, whose high data rate demands a very short cryptoperiod to keep up an appropriate security level.

#### 2.5.4 Time Complexity of Attacks based on Exhaustive Search

The total time spent executing an exhaustive search is directly proportional to the number of trials and the computational power available. It may not be realistic to consider that an attacker does not take advantages of increased availability of computational power and cryptanalytic advances. Table 8 shows the total amount of time to execute a brute force attack, whose data is based on the discussions provided in Section 2.4.4. It can be observed, for example, that if an attack is launched in 2012, utilizing a Single Intel Quad Core processor, a brute force attack would take $7.9 \times 10^{20}$ years. If the biclique technique is employed, such amount of time is reduced to $2.1 \times 10^{20}$ years.
If the biclique technique is used, and Moore's law is utilized to predict the computational power of a similar processor in 2022 and 2032, the total amount of time is dropped to $7.8 \times 10^{18}$ and $7.7 \times 10^{16}$ years, respectively.

Notice that Moore's law was employed in the aforementioned cases only to estimate the power at the beginning of the attack. It was not considered that the attacker is doubling his/her computational power every 1.5 years to speed up the attack. If that case is considered, as per Equation 2.3, a quad core processor starting the attack in 2012 would break AES-128 in 132 years.

The harshest scenario happens when both Moore's law and cryptanalytic advances are considered. Assuming a cryptanalytic advance ratio presented in Section 2.4.4.3, the total time for a quad core to break AES-128, starting in 2012 and using bicliques, would be only 83 years. Several other scenarios for AES-128 are presented in Table 8. In some cases, the total search time is close to 50 years.

Some missions may require security levels of 192 and 256 bits, for other reasons. Therefore, for the sake of completeness, the total search time for AES-192 and AES-256 are listed respectively in Table 9 and Table 10.

### Table 8: AES-128 Exhaustive Search Times

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Brute Force [years]</td>
<td>Brute Force [years]</td>
<td>Brute Force [years]</td>
</tr>
<tr>
<td>2012</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Single Intel quad core</td>
<td>1.360E+04</td>
<td>7.934E+20</td>
<td>1.324E+02</td>
<td>8.487E+01</td>
</tr>
<tr>
<td>Infected computers in China</td>
<td>5.890E+04</td>
<td>1.832E+20</td>
<td>4.909E+19</td>
<td>8.328E+01</td>
</tr>
<tr>
<td>ESA assumption Sentinels</td>
<td>6.000E+04</td>
<td>1.798E+20</td>
<td>1.292E+02</td>
<td>8.326E+01</td>
</tr>
<tr>
<td>BOINC</td>
<td>5.628E+09</td>
<td>1.917E+15</td>
<td>1.044E+02</td>
<td>7.087E+01</td>
</tr>
<tr>
<td>K Computer</td>
<td>1.051E+10</td>
<td>1.027E+15</td>
<td>1.030E+02</td>
<td>7.020E+01</td>
</tr>
<tr>
<td>World iPhones and iPods</td>
<td>1.389E+11</td>
<td>7.768E+13</td>
<td>9.744E+01</td>
<td>6.740E+01</td>
</tr>
<tr>
<td>2022</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Single Intel quad core</td>
<td>1.382E+06</td>
<td>7.810E+20</td>
<td>1.234E+02</td>
<td>7.987E+01</td>
</tr>
<tr>
<td>Infected computers in China</td>
<td>5.984E+06</td>
<td>3.606E+18</td>
<td>1.207E+02</td>
<td>7.903E+01</td>
</tr>
<tr>
<td>ESA assumption Sentinels</td>
<td>6.096E+06</td>
<td>1.770E+18</td>
<td>1.192E+02</td>
<td>7.826E+01</td>
</tr>
<tr>
<td>BOINC</td>
<td>5.718E+11</td>
<td>1.887E+13</td>
<td>9.438E+01</td>
<td>6.587E+01</td>
</tr>
<tr>
<td>K Computer</td>
<td>1.068E+12</td>
<td>1.011E+13</td>
<td>9.303E+01</td>
<td>6.520E+01</td>
</tr>
<tr>
<td>World iPhones and iPods</td>
<td>1.411E+13</td>
<td>7.647E+11</td>
<td>8.744E+01</td>
<td>6.098E+01</td>
</tr>
<tr>
<td>Personal Computers US</td>
<td>2.266E+14</td>
<td>4.763E+10</td>
<td>8.143E+01</td>
<td>5.940E+01</td>
</tr>
<tr>
<td>2032</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Single Intel quad core</td>
<td>1.404E+08</td>
<td>7.687E+16</td>
<td>1.124E+02</td>
<td>7.487E+01</td>
</tr>
<tr>
<td>Infected computers in China</td>
<td>6.079E+08</td>
<td>3.550E+16</td>
<td>1.107E+02</td>
<td>7.403E+01</td>
</tr>
<tr>
<td>ESA assumption Sentinels</td>
<td>6.193E+08</td>
<td>1.742E+16</td>
<td>1.092E+02</td>
<td>7.326E+01</td>
</tr>
<tr>
<td>BOINC</td>
<td>5.809E+13</td>
<td>1.858E+11</td>
<td>8.438E+01</td>
<td>6.087E+01</td>
</tr>
<tr>
<td>K Computer</td>
<td>1.085E+14</td>
<td>9.947E+10</td>
<td>8.030E+01</td>
<td>6.020E+01</td>
</tr>
<tr>
<td>World iPhones and iPods</td>
<td>1.434E+15</td>
<td>7.527E+09</td>
<td>7.744E+01</td>
<td>5.740E+01</td>
</tr>
<tr>
<td>Personal Computers US</td>
<td>2.302E+16</td>
<td>4.688E+08</td>
<td>7.143E+01</td>
<td>5.440E+01</td>
</tr>
</tbody>
</table>
It can be seen in Table 9 that the fastest search for AES-192 is approximately 100 years. In addition, Table 10 shows that the fastest search when using AES-256 is close to 150 years. Again, it is important to emphasize that these numbers are based on the aforementioned assumptions on computational power of an attacker, Moore's law, and cryptanalytic advances. Therefore, they may vary substantially if those parameters are changed.

Achieving 50 years of data protection with AES-128 seems more than enough for the vast majority of space missions, whose lifetimes vary in the range of 10 to 30 years. However, critical scenarios may eventually happen. For instance, if a breakthrough happens in the cryptanalysis of AES, or if it is discovered that a well funded security agency possesses far more computational power than one estimates. Therefore, it would be interesting to investigate in the future what are reasonable worst case parameters so that critical scenarios are also covered.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2012</td>
<td>Initial Brute Force</td>
<td>Initial Biclique</td>
<td>Brute Force</td>
</tr>
<tr>
<td>Single Intel quad core</td>
<td>1.360E+04</td>
<td>1.464E+40</td>
<td>2.972E+39</td>
<td>2.284E+02</td>
</tr>
<tr>
<td>Infected computers in China</td>
<td>5.890E+04</td>
<td>3.379E+39</td>
<td>6.862E+38</td>
<td>2.252E+02</td>
</tr>
<tr>
<td>ESA assumption Sentinels</td>
<td>6.000E+04</td>
<td>3.317E+39</td>
<td>6.736E+38</td>
<td>2.252E+02</td>
</tr>
<tr>
<td>BOINC</td>
<td>5.628E+09</td>
<td>5.373E+34</td>
<td>7.182E+33</td>
<td>2.004E+02</td>
</tr>
<tr>
<td>K Computer</td>
<td>1.051E+10</td>
<td>3.846E+33</td>
<td>7.182E+33</td>
<td>2.004E+02</td>
</tr>
<tr>
<td>World iPhones and iPods</td>
<td>1.389E+11</td>
<td>1.433E+33</td>
<td>2.910E+32</td>
<td>2.004E+02</td>
</tr>
<tr>
<td>Personal Computers US</td>
<td>2.230E+12</td>
<td>8.926E+31</td>
<td>1.813E+31</td>
<td>1.874E+02</td>
</tr>
<tr>
<td></td>
<td>2022</td>
<td>Moore Estimate</td>
<td>Moore Estimate</td>
<td>Moore Estimate</td>
</tr>
<tr>
<td>Single Intel quad core</td>
<td>1.382E+06</td>
<td>1.44E+38</td>
<td>2.925E+37</td>
<td>2.184E+02</td>
</tr>
<tr>
<td>Infected computers in China</td>
<td>5.984E+06</td>
<td>3.262E+37</td>
<td>6.755E+36</td>
<td>2.152E+02</td>
</tr>
<tr>
<td>ESA assumption Sentinels</td>
<td>6.096E+06</td>
<td>3.265E+37</td>
<td>6.631E+36</td>
<td>2.152E+02</td>
</tr>
<tr>
<td>BOINC</td>
<td>5.718E+11</td>
<td>3.481E+32</td>
<td>7.069E+31</td>
<td>1.904E+02</td>
</tr>
<tr>
<td>K Computer</td>
<td>1.068E+12</td>
<td>1.864E+32</td>
<td>3.785E+31</td>
<td>1.890E+02</td>
</tr>
<tr>
<td>World iPhones and iPods</td>
<td>1.411E+13</td>
<td>2.864E+30</td>
<td>6.958E+29</td>
<td>1.804E+02</td>
</tr>
<tr>
<td>Personal Computers US</td>
<td>2.266E+14</td>
<td>8.786E+29</td>
<td>1.774E+29</td>
<td>1.740E+02</td>
</tr>
</tbody>
</table>
Rigorously speaking, infected computers in China represented by the colored lines. Moreover, the type of line indicates whether or not the timeframe, up to 50 years. Several assumptions in the attacker's computational power are used. Cryptanalytic advances can also severely impact on the security level. If it is assumed that an attacker has Moore's law, Equation 2.5 should be used instead. Cryptanalytic advances can also severely impact on the security level. If the assumptions presented in Section 2.4.4.3 are considered, Equation 2.9 should be used instead.

It is important to mention that key length has been interchangeably used as an indication of security level in this document. Rigorously speaking, \( m \) should be referred to as the number of bits of security, which ultimately determines the security level. The reason being is that an algorithm with an \( m \)-bit key may be susceptible to some attacks that may take away some security bits, e.g. the biclique attack against AES. Furthermore, MAC algorithms using \( m \)-bit keys only provide \( m/2 \) bits of security due to birthday attacks. Therefore, this section uses \( m \) to refer to the security level. Hence, if necessary, the key length should be adjusted accordingly to compensate the number of bits reduced by known attacks.

Figure 4 presents estimates for security levels in function of the protection timeframe, up to 50 years. Several assumptions in the attacker's computational power are represented by the colored lines. Moreover, the type of line indicates whether or not the

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2012</td>
<td></td>
<td>Brute Force</td>
<td>Biclique</td>
<td>Brute Force</td>
</tr>
<tr>
<td>Single Intel quad core</td>
<td>1.382E+06</td>
<td>2.657E+57</td>
<td>8.766E+56</td>
<td>3.144E+02</td>
</tr>
<tr>
<td>Infected computers in China</td>
<td>5.984E+06</td>
<td>6.136E+56</td>
<td>2.024E+56</td>
<td>3.112E+02</td>
</tr>
<tr>
<td>ESA assumption Sentinels</td>
<td>6.096E+06</td>
<td>6.024E+56</td>
<td>1.987E+56</td>
<td>3.112E+02</td>
</tr>
<tr>
<td>BOINC</td>
<td>5.718E+11</td>
<td>6.422E+51</td>
<td>2.118E+51</td>
<td>2.864E+02</td>
</tr>
<tr>
<td>K Computer</td>
<td>1.068E+12</td>
<td>3.439E+51</td>
<td>1.134E+51</td>
<td>2.850E+02</td>
</tr>
<tr>
<td>World iPhones and iPods</td>
<td>1.411E+13</td>
<td>2.602E+50</td>
<td>8.583E+49</td>
<td>2.794E+02</td>
</tr>
<tr>
<td>Personal Computers US</td>
<td>2.266E+14</td>
<td>1.621E+49</td>
<td>5.346E+48</td>
<td>2.734E+02</td>
</tr>
<tr>
<td>2032</td>
<td></td>
<td>Moore Estimate</td>
<td>Brute Force</td>
<td>Biclique</td>
</tr>
<tr>
<td>Single Intel quad core</td>
<td>1.404E+08</td>
<td>2.616E+55</td>
<td>8.629E+54</td>
<td>3.044E+02</td>
</tr>
<tr>
<td>Infected computers in China</td>
<td>6.079E+08</td>
<td>6.040E+54</td>
<td>1.992E+54</td>
<td>3.012E+02</td>
</tr>
<tr>
<td>ESA assumption Sentinels</td>
<td>6.193E+08</td>
<td>5.929E+54</td>
<td>1.956E+54</td>
<td>3.012E+02</td>
</tr>
<tr>
<td>BOINC</td>
<td>5.809E+13</td>
<td>6.321E+49</td>
<td>2.085E+49</td>
<td>2.764E+02</td>
</tr>
<tr>
<td>K Computer</td>
<td>1.085E+14</td>
<td>3.385E+49</td>
<td>1.117E+49</td>
<td>2.750E+02</td>
</tr>
<tr>
<td>World iPhones and iPods</td>
<td>1.434E+15</td>
<td>2.561E+48</td>
<td>8.449E+47</td>
<td>2.694E+02</td>
</tr>
<tr>
<td>Personal Computers US</td>
<td>2.302E+16</td>
<td>1.595E+47</td>
<td>5.262E+46</td>
<td>2.634E+02</td>
</tr>
</tbody>
</table>

2.5.5 Determining Key Lengths

Given the amount of time that confidential information is expected to remain protected, the pertinent security level can be calculated. If it is assumed that an attacker has constant computational power, the security level can be determined by Equation 2.2. In the case of a more realistic scenario, where the attacker benefits from Moore's law, Equation 2.4 should be utilized. Cryptanalytic advances can also severely impact on the security level. If the assumptions presented in Section 2.4.4.3 are considered, Equation 2.9 should be used instead.

It is important to mention that key length has been interchangeably used as an indication of security level in this document. Rigorously speaking, \( m \) should be referred to as the number of bits of security, which ultimately determines the security level. The reason being is that an algorithm with an \( m \)-bit key may be susceptible to some attacks that may take away some security bits, e.g. the biclique attack against AES. Furthermore, MAC algorithms using \( m \)-bit keys only provide \( m/2 \) bits of security due to birthday attacks. Therefore, this section uses \( m \) to refer to the security level. Hence, if necessary, the key length should be adjusted accordingly to compensate the number of bits reduced by known attacks.

Figure 4 presents estimates for security levels in function of the protection timeframe, up to 50 years. Several assumptions in the attacker's computational power are represented by the colored lines. Moreover, the type of line indicates whether or not the security level
Moore's law and cryptanalytic advances were taken into account. For instance, the black line represents computational power compromised by viruses in China in December, 2011, which is close to Sentinel's assumption on the computational power of an attacker (6x10^4 MIPS). For this case, the continuous black line shows that 64 bits would be enough for 10 years of data protection. As previously mentioned, the consideration of an attacker with fixed computational power may represent an unrealistic scenario.

It is strongly recommended to take Moore's law into account (represented by dashed lines) given that such a law may still be followed for a few more decades. If that is the case, 10 years of data protection would require about 68 bits of security. For 30 years of data protection, 82 bits of security should be used. Moreover, if the aforementioned assumptions on cryptanalytic advances are considered (represented by the dotted lines), 10 years of data protection would demand 75 bits of security, whereas 30 years would require 102 bits.

Figure 4: Estimates on Security Levels (50 years timeframe)

Let's now take into account the K Computer that achieves 1.051x10^{10} MIPS and is represented by yellow lines. Considering an attacker benefiting from Moore's law, a protection timeframe of 10 and 30 years would demand, respectively, 86 and 99 bits of security. The same scenarios would demand approximately 93 and 119 bits if cryptanalytic advances are considered.
Even though TM PL links might eventually require long term data protection, this may not be the case for TC and TM HK links. Thus, another interesting aspect to be analyzed is the minimum security to be used in the case of short cryptoperiods. That is usually the case for TC and TM HK links. Figure 5 graphically represents a year timeframe and the
associated security levels, whereas Figure 6 reduces the timeframe to a month. For instance, a cryptoperiod of 24 hours would impose the minimum security levels shown in Table 11. Again, Equations 2.2, 2.4, and 2.9 can be utilized to compute the number of security bits for arbitrary cryptoperiods.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Intel Quad Core</td>
<td>1.36E+4</td>
<td>50.06</td>
<td>50.06</td>
<td>59.69</td>
</tr>
<tr>
<td>Infected Computers China</td>
<td>5.89E+4</td>
<td>52.18</td>
<td>52.18</td>
<td>61.81</td>
</tr>
<tr>
<td>ESA Assumption on Sentinels</td>
<td>6.00E+4</td>
<td>52.20</td>
<td>52.20</td>
<td>61.83</td>
</tr>
<tr>
<td>BOINC</td>
<td>5.63E+9</td>
<td>68.72</td>
<td>68.72</td>
<td>78.35</td>
</tr>
<tr>
<td>K Computer</td>
<td>1.05E+10</td>
<td>69.62</td>
<td>69.62</td>
<td>79.25</td>
</tr>
<tr>
<td>World iPhones and iPods</td>
<td>1.39E+11</td>
<td>73.35</td>
<td>73.35</td>
<td>82.97</td>
</tr>
<tr>
<td>Personal Computer US</td>
<td>2.23E+12</td>
<td>77.35</td>
<td>77.35</td>
<td>86.98</td>
</tr>
</tbody>
</table>

In summary, the determination of the security level in terms of bits should first determine the attack scenario that makes more sense for a specific space mission. Then, the total amount of time required for data protection will dictate the security level to be employed. In addition, the key size should be chosen taking into account two factors: 1) reduced security bits due to attacks; 2) standard keys sizes for AES and its modes of operation. Having in mind that AES utilizes either 128-, 192- or 256-bit keys, these are the keys sizes to taken into account. It is not recommended to utilize shorter keys with padding. Cryptographic keys should match AES standard sizes and be created using sound key generation techniques, as specified in [NISTSP800133].

2.5.6 Defining Cryptoperiods

Based on time and data complexity discussions presented in Sections 2.5.3 and 2.5.4, it is possible to determine appropriate cryptoperiods for space missions. Even considering very harsh scenarios on attacker's computational power and cryptanalytic advances, an exhaustive search on AES-128 key space would take more time than the space mission itself. The fastest exhaustive search, according to the estimates listed in Table 8, would take 53 years. Usually, space missions lifetime vary from a few years to a couple of decades. Thus, cryptoperiods for encryption modes of operation such as AES-CTR and AES-GCM could be, at most, as long as the exhaustive search times reported in Table 8, Table 9 and Table 10. Obviously, a security margin should be adopted for the selected cryptoperiods.

However, it is important to consider the probability of a successful forgery attack against authentication mechanisms. As detailed in Section 2.4.5, this attack along with the probability of frame error can be employed to determine cryptoperiods for CMAC and GCM. Depending upon the CER, the cryptoperiod can vary from a huge number of days, to a few seconds. Critical cases are expected to happen with TM PL links when CER=10^{-6}. In this scenario, an authentication key for the TM PL link of a LEO mission would have a cryptoperiod 2 minutes and 13 seconds. For GEO missions, where the TM PL link is continuously used at high data rates, the cryptoperiod would be only 14.4 seconds.
Recommendations on cryptoperiods are provided by NIST in [NISTSP80057P1]. Notice that NIST generally defines terms such as originator usage period (OUP) and recipient usage period (RUP). Since the keys addressed in this research are utilized to secure communications, not storage, the OUP is approximately the same as the RUP, and is denoted as cryptoperiod. NIST recommendations on symmetry keys are as follows:

- Authentication Key: ≤ 2 years.
- Encryption Key: day to a week for large data volumes, up to a month for smaller volumes.
- Key Wrapping Key: day to a week for large number of keys, up to a month for smaller numbers.
- Master Key: ≤ 1 year.

Besides, NIST also recommends the limitation of keys derived from MKs to a single use/session. No satisfactory scientific justification, however, is provided for the NIST recommended values. The motivation that led NIST to recommend the aforementioned cryptoperiods is subject of future research.

Therefore, the recommendation is to follow cryptoperiods listed in Table 8, Table 9, and Table 10 for AES in CTR mode of operation. If CMAC is utilized, Table 7 should be used instead. Additionally, if CTR along with CMAC are utilized to achieve encryption and authentication, it would be recommended to synchronize the key update of both mechanisms. This would support a reduction in the complexity for key management, since those keys could have the same identifier and be associated with a certain encrypted and authenticated data. In this case, Table 7 would determine the cryptoperiods for both CTR and CMAC modes. In the case of utilizing GCM for authenticated encryption, Table 7, Table 8, Table 9, and Table 10 should be analyzed and the smallest cryptoperiod should be chosen.
2.6 Instrument/Payload Data Distribution

Three main scenarios can be devised for user data distribution: 1) through ESA systems, 2) directly to the user, but with ESA knowledge of users' keys, and 3) direct to the user without ESA knowing the users' keys.

2.6.1 ESA-Controlled Data Distribution

In this scenario it is assumed that ESA generates the cryptographic keys and uploads them onto the SC. These keys are utilized for securing all communication channels between the SC and the CC. ESA is responsible for the entire key management on the space segment, as well as for download of instrument data and its subsequent distribution to the users. Specifically for PL data, the CC checks the authenticity of the packages and decrypts it. Next, the data is re-encrypted and new MAC tags are computed in order to secure transmission of data to the user. Notice that the ground segment, between the CC and the user, can utilize traditional key agreement techniques based on asymmetric techniques. Encryption and authentication between the CC and the user is mandatory to ensure confidentiality and authentication of the downloaded data. This scenario, as depicted in Figure 7, could utilize a two level key hierarchy. The lower level would comprise SKs to secure the channels between the SC and the CC. The upper level would count on KEKs or KDKs to rekey SKs.

![Figure 7: Data Distribution through ESA’s CC](image)

2.6.2 Direct Data Distribution with ESA Knowledge of Keys

Some missions may demand a direct TM PL link with its users, as illustrated in Figure 8. Again, ESA could upload user's keys prior to launch and would either delegate key management to the user or do it itself. TM PL data is downloaded directly to the user and counts on encryption and authentication services. Notice however that ESA would know the keys utilized. Therefore, the agency could have access to the contents of the TM PL link.

If ESA is performing key management, every key update would require informing the user on the new key utilized. Even though the CC-user channel is encrypted in Figure 8, authentication alone would be enough to send a key index to the user - the user would already
possess the key table with the corresponding indexes. The TC link requires encryption and authentication in order to support the upload of new SKs to the SC.

If the user is responsible for performing key management, then an encryption and authentication would be required between the CC and the user. The main reason is that the user may need to send confidential information to the CC, for instance new blocks of SKs, so that they can be uploaded to the SC through the TC link.

Key hierarchy would again require the two levels: one for SKs and a second for KEKs and KDKs. Depending on the amount of data downloaded through the TM PL link, it may be required to utilize KDFs in order to minimize traffic caused by frequent of key upload through the TC link.

2.6.3 Direct Data Distribution with User Key Management

The last scenario considers that the agency does not perform key management for the user PL, and does not know the keys utilized by the users in their TC, and TM channels. A set of cryptographic keys are installed onto the SC prior to launch, but they are not released to the agency. The user would count on an authenticated and encrypted direct TC and TM HK channel with the SC. These two user channels would be placed within the TC and TM HK links of the CC, as can be observed in Figure 9.

In this case the user would have entire responsibility in switching SKs and uploading new SK packets. Yet an important feature is the ability for the user to check the integrity of the key tables placed onboard of the host SC. As a consequence, KHKs becomes necessary. Likewise the previous scenario, onboard KDFs could be employed by the user so that TC and TM HK traffic is decreased.
Apart from the three scenarios described above, cryptoperiods would be defined by the amount of data that is sent through the TC and TM channels and the amount of time that it is expected to be protected. Based upon specific mission parameters as shown in Section 2.2, key scopes, roles and hierarchy specified in Section 2.3, as well as the complexity of attacks as described in Sections 2.4 and 2.5, it is possible to determine how over-the-air-rekeying (OTAR) should be done. This is the topic addressed in Section 2.7, which covers several types of missions, links, and security services.

Figure 9: Direct Data Distribution with User Key Management
2.7 Over the Air Re-Keying

This section shows how the aforementioned results from Section 2.5 could be applied to Sentinel- and Meteosat-like missions. Differently from the actual missions, the cases considered in this section assume that full encryption and authentication is in place for all communication links, and may employ AES' CTR, CMAC, or GCM modes of operation. The results shown in the following sections consider AES-128, 192 and 256 as the underlying block cipher.

2.7.1 LEO (Sentinel-like) Missions

The key hierarchy considered in Sentinels is based on key transport, as shown in Figure 1 (a). Two levels are considered: Level 1 for SKs, and Level 2 for KEKs and SMKs, which are loaded onto the SC prior to launch. The onboard storage is assumed to keep separate key sets for encryption and authentication. Besides, it is assumed that a set of 4096 keys for each security mechanism are loaded onto the SC before launch. In addition, a rekeying session is assumed to be performed in a single overpass, which lasts up to 10 minutes.

2.7.1.1 Authentication

Cryptoperiods for CMAC and GCM are directly determined by the successful forgery attack. Even though they vary with the CER, they are independent of the key size utilized. Therefore, the worst case scenario of Table 7 provides the cryptoperiods for authentication, which are summarized in Table 12.

Table 12: LEO Authentication Cryptoperiods

<table>
<thead>
<tr>
<th>Channel Error Rate</th>
<th>TC</th>
<th>TM HK [days]</th>
<th>TM PL</th>
</tr>
</thead>
<tbody>
<tr>
<td>10^{-4}</td>
<td>5.39E+12</td>
<td>2.43E+13</td>
<td>1517092219</td>
</tr>
<tr>
<td>10^{-5}</td>
<td>5469724</td>
<td>24613758</td>
<td>1538.359867</td>
</tr>
<tr>
<td>10^{-6}</td>
<td>5.469724</td>
<td>24.61376</td>
<td>0.00153836</td>
</tr>
</tbody>
</table>

* 2 minutes and 13 seconds

When CER=10^{-4} and 10^{-5}, the cryptoperiods are longer than the lifetime of the space mission. If NIST recommendations are followed for those cases, the cryptoperiods would become as listed in Table 13. Notice, however, that for CER=10^{-6} the cryptoperiod for TM PL links is about 2 minutes. Consequently, if a TM PL session lasts for 10 minutes, the communication session should consider the utilization of five keys.

Table 13: LEO Authentication Cryptoperiods based on NIST Recommendations

<table>
<thead>
<tr>
<th>Channel Error Rate</th>
<th>TC</th>
<th>TM HK [days]</th>
<th>TM PL</th>
</tr>
</thead>
<tbody>
<tr>
<td>10^{-4}</td>
<td>≤ 730*</td>
<td>≤ 730*</td>
<td>≤ 730*</td>
</tr>
<tr>
<td>10^{-5}</td>
<td>≤ 730*</td>
<td>≤ 730*</td>
<td>≤ 730*</td>
</tr>
<tr>
<td>10^{-6}</td>
<td>5.469723972</td>
<td>24.61376</td>
<td>0.001538</td>
</tr>
</tbody>
</table>

* NIST Recommendation
Table 14 shows the total number of keys to protect the entire mission, assuming a 10-, 20-, 30-year mission lifetime.

Table 14: Number of Authentication SKs for LEO

<table>
<thead>
<tr>
<th>Channel Error Rate</th>
<th>10-Year Mission</th>
<th>20-Year Mission</th>
<th>30-Years Mission</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TC</td>
<td>TM HK</td>
<td>TM PL</td>
</tr>
<tr>
<td>10⁻⁴</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>10⁻⁵</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>10⁻⁶</td>
<td>640</td>
<td>143</td>
<td>2275151</td>
</tr>
</tbody>
</table>

It can be observed that if the TM PL link has a CER of 10⁻⁶, a total of 650 keys per day would be demanded. Even if 256-bit keys are used, a few seconds would be enough to upload the keys to the SC. So, a single overpass can indeed be utilized for rekeying.

It is also possible to compute the amount of rekeying that a mission will demand, taking into account the 4096 keys already pre-loaded. Considering the transmission of data through the TC link at 64kbps during 10 minutes, the number of 128-bit keys transported per overpass is 307200. In the case of 192- and 256-bit keys, it is possible to transport a total of 204800 and 153600 keys, respectively. Therefore, the amount of rekeying necessary during the entire lifetime of a mission is shown in Table 15. Since each rekeying operation requires a different KEK, the number of KEKs required for three mission lifetimes (10, 20, and 30 years) and three SK sizes (128, 192 and 256 bits) are shown.

Table 15: Number of KEKs to Rekey Authentication SKs for LEO with CER=10⁻⁶

<table>
<thead>
<tr>
<th>Mission Lifetime [years]</th>
<th>SK Size</th>
<th>128-bit</th>
<th>192-bit</th>
<th>256-bit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>8</td>
<td>12</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>15</td>
<td>23</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>23</td>
<td>34</td>
<td>45</td>
<td></td>
</tr>
</tbody>
</table>

2.7.1.2 Encryption and Authenticated Encryption

Cryptoperiods for encryption modes AES-CTR and AES-GCM are derived from Table 8, Table 9, and Table 10. Even considering state-of-the-art attacks, increasing availability of computational power, and harsh scenarios on cryptanalysis, it can be observed that the cryptoperiods would be at least 50 years. In this case, it would be appropriate to follow NIST recommendations. Specifically, the cryptoperiod should be one day for large data volumes data, such as in TM PL, and one month for smaller volumes as in TC and TM HK. Table 16 shows the total number of keys necessary to cover a 10-, 20-, and 30-year mission. It can be noticed that a maximum of three overpasses are necessary to uploaded all these keys through the TC link. In other words, a maximum of three rekeying sessions is expected to happen in the entire lifetime of a Meteosat mission.

Table 16 : Number of Encryption SKs for LEO

<table>
<thead>
<tr>
<th>Data Volume</th>
<th>10-year</th>
<th>20-year</th>
<th>30-year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>3650</td>
<td>7300</td>
<td>10950</td>
</tr>
<tr>
<td>Large</td>
<td>113150</td>
<td>226300</td>
<td>339450</td>
</tr>
</tbody>
</table>
2.7.2 GEO (Meteosat-like) Missions

The same set of assumptions made for Sentinels in Section 2.7.1, are valid for Meteosat. The only difference is that for GEO missions, communication sessions is used in lieu of overpasses. It is assumed that each GEO communication session is expected to last up to 10 minutes, and a rekeying operation should be performed within that time frame.

2.7.2.1 Authentication

As can be noticed on Table 17, authentication cryptoperiods for all links when CER=10^{-4} and 10^{-3} are longer than the mission lifetime. The exception is a TM PL cryptoperiod of 167 days when CER=10^{-6}. As expected, shortest cryptoperiods are demanded when CER=10^{-6}. If NIST recommendations are followed, the cryptoperiods would be as listed in Table 18.

Table 17: GEO Authentication Cryptoperiods

<table>
<thead>
<tr>
<th>Channel Error Rate</th>
<th>TC</th>
<th>TM HK</th>
<th>TM PL</th>
</tr>
</thead>
<tbody>
<tr>
<td>10^{-4}</td>
<td>8.63E+13</td>
<td>2.43E+13</td>
<td>164615041.1</td>
</tr>
<tr>
<td>10^{-5}</td>
<td>87515584</td>
<td>24613758</td>
<td>166.9227286</td>
</tr>
<tr>
<td>10^{-6}</td>
<td>87.51558</td>
<td>24.61376</td>
<td>0.000166923*</td>
</tr>
</tbody>
</table>

* 14.4 seconds

Table 18: GEO Authentication Cryptoperiods based on NIST Recommendations

<table>
<thead>
<tr>
<th>Channel Error Rate</th>
<th>TC</th>
<th>TM HK</th>
<th>TM PL</th>
</tr>
</thead>
<tbody>
<tr>
<td>10^{-4}</td>
<td>≤ 730*</td>
<td>≤ 730*</td>
<td>≤ 730*</td>
</tr>
<tr>
<td>10^{-5}</td>
<td>≤ 730*</td>
<td>≤ 730*</td>
<td>166.9227286</td>
</tr>
<tr>
<td>10^{-6}</td>
<td>87.51558</td>
<td>24.61376</td>
<td>0.000166923</td>
</tr>
</tbody>
</table>

* NIST Recommendation

The critical case happens for TM PL, whose cryptoperiod of 14.4 seconds will demand a total of 5991 keys per day. Consequently, this will cause either frequent rekeying operations or larger onboard storage. Even if 256-bit keys are used, the upload of a daily batch of keys would take slightly over 6 minutes over a TC link of 4kbps. Thus, a single communication session would suffice for a rekeying operation. In Table 19 shows the total number of keys to protect the entire mission, assuming a 10-, 20-, 30-year mission lifetime.

Table 19: Number of Authentication SKs for GEO

<table>
<thead>
<tr>
<th>Channel Error Rate</th>
<th>TC</th>
<th>TM HK</th>
<th>TM PL</th>
<th>TC</th>
<th>TM HK</th>
<th>TM PL</th>
<th>TC</th>
<th>TM HK</th>
<th>TM PL</th>
</tr>
</thead>
<tbody>
<tr>
<td>10^{-6}</td>
<td>39.99</td>
<td>142.20</td>
<td>20967785.68</td>
<td>79.99</td>
<td>284.39</td>
<td>41935571.37</td>
<td>119.98</td>
<td>426.59</td>
<td>62903357.05</td>
</tr>
</tbody>
</table>

Taking into account the 4096 keys already pre-loaded, and the worst case of CER=10^{-6}, it is possible to compute the number of rekeying that a mission will demand, and consequently the number of KEKs. Table 20 shows the total number of rekeying operations,
for 128-, 192- and 256-bit keys. Again, 10-minute communication session over a TC link at 4kbps is considered. Given that a KEK is used only once, this table also represents the total number of KEKs required for the entire mission lifetime (10, 20, and 30 years).

Table 20: Number of KEKs to Rekey Authentication SKs for GEO with CER=10⁻⁶

<table>
<thead>
<tr>
<th>Mission Lifetime [years]</th>
<th>128-bit</th>
<th>192-bit</th>
<th>256-bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>1092</td>
<td>1638</td>
<td>2184</td>
</tr>
<tr>
<td>20</td>
<td>2184</td>
<td>3276</td>
<td>4368</td>
</tr>
<tr>
<td>30</td>
<td>3277</td>
<td>4915</td>
<td>6553</td>
</tr>
</tbody>
</table>

### 2.7.2.2 Encryption and Authenticated Encryption

Meteosat has exactly the same cryptoperiods as Sentinels. More precisely, one day for large data volumes data, such as in TM PL, and one month for smaller volumes as in TC and TM HK. Table 21 shows the total number of keys necessary to cover a 10-, 20-, and 30-year mission.

Table 21: Number of Encryption SKs for GEO

<table>
<thead>
<tr>
<th>Data Volume</th>
<th>10-year (10-bit)</th>
<th>20-year (192-bit)</th>
<th>30-year (256-bit)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>3650</td>
<td>7300</td>
<td>10950</td>
</tr>
<tr>
<td>Large</td>
<td>113150</td>
<td>226300</td>
<td>339450</td>
</tr>
</tbody>
</table>

Meteosat has a TC data rate of 4kbps, which is different of Sentinels'. This impacts the number of keys that can be uploaded in a rekeying window of 10 minutes. For small data volumes, a single rekeying operation is enough to cover the entire mission lifetime. However, for higher data volumes such as in TM PL links, many more rekeying operations will have to be executed. Precisely, the total number of rekeying operations is listed in Table 22, considering three key sizes (128, 192, and 256 bits). Important to notice that each KEK is used for a single rekeying and then discarded. Hence, the number of rekeying operations directly determines the number of KEKs that the SC should carry onboard.

Table 22: Number of KEKs for High Data Volumes Links of GEO Missions

<table>
<thead>
<tr>
<th>Mission Lifetime [years]</th>
<th>Number of Rekeying Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>128-bit</td>
</tr>
<tr>
<td>10</td>
<td>6</td>
</tr>
<tr>
<td>20</td>
<td>12</td>
</tr>
<tr>
<td>30</td>
<td>18</td>
</tr>
</tbody>
</table>
2.8 Appendix: Improved Bounds on CMAC Forgery Probability

Cryptographic papers often refer to a security bound ($Adv$) or the maximum success or forgery probability of adversaries. Table 23 indicates the progress made on formulating bounds on the probability of forgery using the CMAC standard. Note that OMAC1 is equivalent to CMAC. Initially in this research, the probability published in the ECRYPTII 2011 report [2011ECRYPTII] was utilized. However this formulation of probability was based upon the 2003 bound (see 3rd row of Table 23, INDOCRYPT 2003 publication), which since has been improved. The parameters of the equations are as follows: $q$ represents the number of queries (or the number of authenticated tags produced), $L$ is maximum length of any message which is being authenticated (in terms of the number of blocks), $n$ is the block length of the cipher (128-bits for AES) or number of bits of MAC, $\sigma$ is the total length of all messages summed over all queries in terms of number of blocks (which in our case, $\sigma = Lq$).

<table>
<thead>
<tr>
<th>Bound on Probability of Forgery for CMAC</th>
<th>Publication</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>$[(5L^2+1)q^2+1]/2^n$</td>
<td>Iwata etal. “OMAC: one key CBC MAC” FSE 2003, March 2003 (See Theorem 5.1)</td>
<td></td>
</tr>
<tr>
<td>$[4\sigma^2+1]/2^n$</td>
<td>Iwata etal. “Stronger Security Bounds for OMAC, TMAC, and XCBC” INDOCRYPT 2003, April 2003 (See table 3)</td>
<td>Also in 2003 submitted as “comments to NIST” document with same title</td>
</tr>
<tr>
<td>$10q \sigma/2^n$</td>
<td>In above publication by M.Nandi</td>
<td>In above</td>
</tr>
<tr>
<td>If $L \leq 2^{n/3}$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Chapter 3: Key Management Fundamental Requirements (Supplement) and Trusted Platform Research
3.1 Scope

This chapter is divided into two main parts. The first part is provided as a supplement to some subjects addressed in the first phase of the project. The second part reflects the research done in the second phase of this project.

Section 3.2 provides the results of additional research done on the determination of cryptoperiods. It includes updated equations for upper bounds for forgery attacks on authentication mechanisms. In addition, it includes refined equations for the computation of probability of errors in TM links. These equations comprise telemetry-specific data for the computation of the probabilities of undetected errors in coding schemes based on Reed-Solomon codes.

Section 3.3 presents a key management scheme based on a Trusted Platform Module (TPM), which was the main topic to be addressed in the phase 2 of this project. Specifically, Section 3.3.1 presents the key management goals to be achieved with the proposed mechanism. Also, it specifies details on the infrastructure that had to be taken into account while devising such a scheme. Section 3.3.2 describes related research and the reasons that they cannot be directly applied to the context of space systems. Section 3.3.3 presents the communication scenarios that will be further considered and discussed in the following sections. Section 3.3.4 introduces the sets of secret materials (SSMs) utilized in the protocols and within the trusted platform. It also defines the set of operations utilized in the proposed rekeying procedures. Section 3.3.5 presents a rekeying mechanism to transport keys from a Control Center (CC) to a destination Spacecraft (SC) through an intermediate entity denoted as Master Spacecraft (MSC). The proposed mechanisms achieve hop-by-hop secrecy. A mechanism to achieve end-to-end secrecy is then proposed in Section 3.3.6. Both sections (3.3.5 and 3.3.6) present a detailed description of the protocol to be utilized. In addition, they include a set of illustrations corresponding to the step-by-step execution of the aforementioned protocols. Onboard integrity check is another topic covered in this research project. Section 3.3.7 introduces a hop-by-hop scheme to perform integrity checks on the onboard platform. Complementary to that approach, Section 3.3.8 presents a methodology to execute an end-to-end integrity check of the platform of the destination SC, without disclosing any information to intermediate nodes. In addition, Section 3.3.9 provides a qualitative evaluation of the proposed mechanisms for key transport. Consequently, it allows for estimates of the onboard resources required to support the proposed key transport mechanisms. Finally, Section 3.3.10 provides a quantitative analysis based on three scenarios that may be found in space applications.
3.2 Revisiting Authentication Cryptoperiods

This topic has been included in this document as a supplement to phase 1 results (Chapter 2). The sections below provide updated information and results of further research done in the subject of “Key Management Fundamental Concepts”.

3.2.1 Updated Equations for Authentication Mechanisms

The cryptoperiod studies performed in phase 1 of this project were based on the assumption that the probability of a successful forgery ($P_{\text{Forg}}$) attack against the authentication mechanism (CMAC) would not be higher than the probability of an error ($P_{\text{Err}}$) in the decoding mechanism. The initial assumption on $P_{\text{Forg}}$ reported in the “Phase 1 Report” was based on [2011ECRYPTII] (page 57). That document determined that the probability of a successful forgery ($P_{\text{Forg}}$) would be given by:

$$P_{\text{Forg}} = \frac{n^2}{2^{(b-2)}}$$

where $n$ is the number of blocks and $b$ is the number of bits in a block (128 bits in the case of AES).

Furthermore, in the paper published at ESTEL’12 [2012JGS] the probability of an undetected error was estimated to be $nP_{\text{Err}}$. Therefore, the inequality $n^2/2^{(b-2)} \leq nP_{\text{Err}}$ was utilized to compute the maximum number of frames $n$ that could be transmitted in order to avoid a successful forgery.

As the research progressed, $P_{\text{Forg}}$ and $P_{\text{Err}}$ were updated in order to obtain a more precise mathematical model. For telecommand, the probability of an error $P_{\text{Err}}$ can be computed through the equation $P_{\text{Err}} = (1 - (1 - P_{\text{ue}})^n)$, where $n$ is number of frames, and $P_{\text{ue}}$ is the probability of an undetected error in a frame after being decoded. For telemetry, there are five codewords within an encoded frame, therefore the equation becomes $P_{\text{Err}} = (1 - (1 - P_{\text{ue}})^5n)$.

3.2.2 Computation of $P_{\text{ue}}$ for Reed-Solomon Codes

In the first phase of this project, it was not possible to find in the literature $P_{\text{ue}}$ specific for Reed-Solomon (RS) codes, which are one of the more frequent coding schemes utilized in TM links. As a result, $P_{\text{ue}}$ for TC were utilized as a first approximation. Further investigation was done on how to compute TM-specific $P_{\text{ue}}$ so that a more precise result could be obtained.

In collaboration with research groups in the Politecnico di Torino and Università Politecnica delle Marche, both in Italy, we were able to obtain the equations specific for RS codes. These equations are listed in Section 3.4 (Appendix) and were utilized to compute the $P_{\text{ue}}$ utilized in the following sections. A summary of such computations, based on equations formulated in [1984KL], are as follows.

$$P_{\text{ue}}(\epsilon, \lambda) = \sum_{h=0}^{\lambda} \binom{q-1}{h} q^{-r} (q-1)^h - \epsilon^h (1-\epsilon)^{q-1-h}$$

$$+ \sum_{j=0}^{\min(r-1,q-1-h)} \binom{q-1-h}{j} \frac{\epsilon}{q-1} \left(1-\frac{q\epsilon}{q-1}\right)^{(q-1-h-j)} R_{h,j}(\epsilon)$$

(Equation 3.1)

where $r = 2t$, and
\[
R_{h,j}(\varepsilon) = \sum_{l=0}^{\min(r-1-j,h)} (-1)^{h-l} \left( h \right)_{l} \left( 1-q^{-r+j+l} \right) \left( 1-\frac{\varepsilon}{q-1} \right)^{l} \left( 1-\frac{q\varepsilon}{q-1} \right)^{h-l}, \quad (\text{Equation 3.2})
\]

for \(0 \leq j < r\).

Taking [ECSSEST5001C, CCSDS1310B2] as references for RS parameters, the following literals can be determined:
- \(t = 16\); (number of bits corrected per codeword),
- \(r = 32\); \((r = 2t)\) and
- \(m = 8\); (8 bits per RS symbol)
- \(p = 10^{-4}, 10^{-5},\) and \(10^{-6}\) (channel bit error rate)

As defined in [1984KL]:
- \(q = 2^m;\)
- \(l - \varepsilon = (l - p)^m \Rightarrow \varepsilon = l - (l - p)^m;\)
- \(\varepsilon = (q-1)/q;\) (for the worst channel condition)
- \(\lambda = t = 16\) (the maximum correcting capability is being considered)

Given the aforementioned parameters and Equations 3.1 and 3.2, the probability of undetected error, for three different Channel Error Rates (CERs), are as follows:
- \(p = 10^{-4} \Rightarrow P_{ue} = 2.376666781 \times 10^{-41}\)
- \(p = 10^{-5} \Rightarrow P_{ue} = 2.781893098 \times 10^{-58}\)
- \(p = 10^{-6} \Rightarrow P_{ue} = 2.826045762 \times 10^{-75}\)

For worst channel condition \((\varepsilon = (q-1)/q)\) as per [1984KL]: \(P_{ue} = 2.608888805 \times 10^{-14}\).

### 3.2.3 Successful Forgery against Authentication Mechanisms

The determination of a probability of forgery is currently based on the bounds reported in [2007Nandi], i.e. \(P_{\text{Forg}} = 10n^2L/2^b\), where \(L\) is defined as the number of blocks of the underlying block cipher, and \(n\) is defined as the number of queries. In this research, the number of queries would correspond to the number of frames authenticated. Furthermore, this research assumes the utilization of AES as the underlying block cipher, whose block size is 128 bits \((b = 128)\). If the frame size is defined as \(f\), then \(L\) could be computed as \(L = f / b\).

A summary of \(P_{ue}\) for BCH and RS codes, respectively utilized in the computation of TC and TM cryptoperiods is listed in Table 24. Moreover, Table 25 shows a set of worst-case communication scenarios for several link and mission types. These tables are utilized in the computation of TC and TM cryptoperiods in the next sections.

<table>
<thead>
<tr>
<th>Link</th>
<th>Requirement</th>
<th>Performance</th>
<th>(P_{ue})</th>
<th>Performance</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC (BCH)</td>
<td>(1 \times 10^9)</td>
<td>(2.14 \times 10^{16})</td>
<td>(2.17 \times 10^{22})</td>
<td>(2.17 \times 10^{39})</td>
<td></td>
</tr>
<tr>
<td>TM (RS)</td>
<td>(1 \times 10^9)</td>
<td>(2.38 \times 10^{41})</td>
<td>(2.78 \times 10^{58})</td>
<td>(2.83 \times 10^{71})</td>
<td></td>
</tr>
</tbody>
</table>

61
Table 25: Summary of Worst-Case Communication Scenarios

<table>
<thead>
<tr>
<th>Orbit</th>
<th>Link</th>
<th>Scenario</th>
<th>Data rate [kbps]</th>
<th>Frame size [bits]</th>
<th>Frame rate [frames/s]</th>
<th>Contact time [min/day]</th>
<th>Daily Frame rate [frames/day]</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC</td>
<td>Distributed Attacker</td>
<td>64</td>
<td>1024</td>
<td>64</td>
<td>1440</td>
<td>5529600</td>
<td></td>
</tr>
<tr>
<td>LEO</td>
<td>HK TM</td>
<td>Nominal/Attacker</td>
<td>2048</td>
<td>8192</td>
<td>256</td>
<td>10</td>
<td>153600</td>
</tr>
<tr>
<td></td>
<td>PL TM</td>
<td>Nominal/Attacker</td>
<td>819200</td>
<td>8192</td>
<td>102400</td>
<td>400</td>
<td>2457600000</td>
</tr>
<tr>
<td>TC</td>
<td>Attacker</td>
<td>4</td>
<td>1024</td>
<td>4</td>
<td>1440</td>
<td>345600</td>
<td></td>
</tr>
<tr>
<td>GEO</td>
<td>HK TM</td>
<td>Nominal/Attacker</td>
<td>2048</td>
<td>8192</td>
<td>256</td>
<td>10</td>
<td>153600</td>
</tr>
<tr>
<td></td>
<td>PL TM</td>
<td>Nominal/Attacker</td>
<td>2097152</td>
<td>8192</td>
<td>262144</td>
<td>1440</td>
<td>22649216000</td>
</tr>
</tbody>
</table>

3.2.4 Graphical Analysis of $P_{Err}$ and $P_{Forg}$ for TC

BCH codes are employed for encoding TC frames. Each TC frame has 147 codewords as reported in section 3.2 of [CCSDS2301G1]. The performance of BCH codes for Single Error Correction (SEC) with Cyclic Redundancy Check (CRC) is reported in tables 8-16 of [CCSDS2301G1]. For CERs = $10^{-4}$, $10^{-5}$, and $10^{-6}$, the probabilities of undetected error in a frame ($P_{ue}$) are respectively $2.14\times10^{-16}$, $2.17\times10^{-22}$, $2.17\times10^{-28}$.

Notice that these probabilities correspond to 130 codeblocks. There is no specification in [CCSDS2301G1] for 147 codeblocks, which is the actual case for TC frames. However, as can be inferred from tables 8-16 in [CCSDS2301G1], the $P_{ue}$ for 147 codeblocks would be in the order of $10^{-16}$, $10^{-22}$, and $10^{-28}$. TC frames are 1024 bits long ($f = 1024$), hence $L_{TC} = 8$. Consequently, an equation specific for TC can be determined: $P_{Forg} = 10^{-n^{2}-8/2^{128}}$.

Maple was utilized to investigate a solution for the inequality $P_{Forg} \leq P_{Err}$, which would result in the determination of the number of frames $n$. However, this tool was unable to solve that equation for $P_{ue} = 10^{-9}$, and resulted in a warning (“Warning, solutions may have been lost”). A solution could not be found for $P_{ue} = 10^{-16}$, $10^{-22}$, and $10^{-28}$ as well.

The inability of Maple to solve the inequality $P_{Forg} \leq P_{Err}$ will be left as a subject for future investigation. Besides, note that the $P_{Forg}$ equation utilized is the simplified equation proposed in [2007Nandi], which imposes the limit $L < 2^{L_{TC}/3}$. At this point, it is still unclear why the equation of $P_{Forg}$ allowed the curve to go past the unity, while it should be limited within $[0,1]$. 
A graphical evaluation then took place to evaluate the behavior of the $P_{Err}$ and $P_{Forg}$ curves, as shown in Figure 10. The $P_{uc}$ utilized in the evaluation was $10^{-9}$. In order to have $P_{Forg} = 1$, $n$ would have to be $2.06 \times 10^{18}$. Moreover, as can be noticed in Figure 11, the $P_{Err}$ curve becomes very close to 1 when $n$ is in the order of $10^{10}$. Precisely, if $P_{Err} = 0.9999$, then $n = 9.21 \times 10^9$.

As it can be graphically observed in Figure 10, $P_{Err}$ asymptotically approaches 1 much sooner than $P_{Forg}$. In other words, once a number of frames have been transmitted to make $P_{Forg} = 1$, an error in the decoding scheme will have happened already. Identical
evaluation was performed for $P_{ue} = 2.38\times10^{-41}$, $2.78\times10^{-58}$, and $2.83\times10^{-75}$, in which similar results were obtained. This results show that comparison between $P_{Forg}$ and $P_{Err}$ may not provide reliable results due to the different nature of the two curves, and how fast they approach the unity.

3.2.5 Graphical Analysis of $P_{Err}$ and $P_{Forg}$ for TM

Similar analysis can be done for TM, though some adjustments to $P_{Err}$ have to be made to match the RS encoding scheme used in TM. Each TM frame comprises 5 codeblocks, so that the $P_{Err}$ equation becomes: $P_{Err} = (1 - (1 - P_{ue})^{5n})$. Furthermore, [CCSDS1301G1] does not specify the performance requirement of the RS encoding scheme. Thus, it is for now assumed that the TM has the minimum performance similar to TC, which is $P_{ue} = 10^{-9}$.

According to previous calculations done in this research, the $P_{ue}$ resulting from the performance of RS codes would be $P_{ue} = 2.38\times10^{-41}$, $2.78\times10^{-58}$, and $2.83\times10^{-75}$, respectively for CERs = $10^{-4}$, $10^{-5}$, and $10^{-6}$.

Again, Maple was unable to determine $n$ as a solution of the inequality $P_{Forg} \leq P_{Err}$ for $P_{ue} = 10^{-5}$, $2.38\times10^{-41}$, $2.78\times10^{-58}$, and $2.83\times10^{-75}$. It was possible to compute $P_{Forg} = 1$ to obtain $n$, which would have to be $7.29\times10^{17}$ frames. This is represented in the graph in Figure 12.

![Figure 12: TM $P_{Err}$ (blue) and $P_{Forg}$ (red), $0 \leq n \leq 10^{18}$](image)

Analyzing the $P_{Err}$ curve in Figure 13, it can be notice that it approaches 1 whenever $n$ is in the order of $10^9$. More precisely, $P_{Err} = 0.9999$ when $n = 1.84\times10^9$. Again, the same issue occurs, in which the $P_{Forg}$ and $P_{Err}$ curves present completely different behaviors. Consequently, it is not possible to rely on the comparison $P_{Forg} \leq P_{Err}$ to determine the authentication cryptoperiods.
3.2.6 Probability of Accepting Forged Data

An alternative method to compute cryptoperiods for authentication mechanisms was also investigated. It was motivated by the outcome of the graphical analysis done in Sections 3.2.4 and 3.2.5, as well as the issue with the $P_{\text{Forg}}$ equation proposed by [2007Nandi]. The new approach no longer considers $P_{\text{Err}}$ for the entire decoding sequence. Instead, only $P_{ue}$ is taken into account. Hence, the equations will now reflect the probability of having undetected errors for a type of coding scheme associated with a link, and its expected CER. Additionally, the new approach also takes into consideration the case presented in NIST’s recommendation for applications using hash algorithms [NISTSP800107] (page 15). The rationale behind this approach is that the verification of a MAC tag does not guarantee that its associated text is authentic. There is always a chance of having an attacker forging the tag for a given text, without the knowledge of the key utilized. Obviously, the shorter the tag the higher the chance of a successful forgery. The probability of a forgery can be defined as

$$P_{\text{Forg}} = 1 / 2^l.$$  \hspace{1cm} \text{(Equation 3.3)}

If the $P_{\text{Forg}}$ is made equivalent to $P_{ue}$, then it is possible to determine the minimum tag length ($l$) for both TC and TM links. If the requirement of $P_{ue} = 10^{-9}$ is taken into account, the theoretical minimum tag length would be 30 bits. Evidently, such a tag length would not be recommended. It is to be noticed that this is an estimate of the tag size if $P_{\text{Forg}}$ and $P_{ue}$ were made equivalent.

Considering the channel conditions in which CER = $10^{-4}$, $10^{-5}$, and $10^{-6}$ are expected, the BCH codes employed TC link would result in $P_{ue} = 10^{-16}$, $10^{-22}$, $10^{-28}$. Consequently, the minimum TC tag sizes for these $P_{ue}$ would be, respectively, $l = 53, 72, 92$ bits. In addition, the RS codes utilized in TM links would have $P_{ue} = 10^{-9}$, $10^{-41}$, $10^{-58}$, $10^{-75}$ for the
aforementioned CERs. As a result, the TM tag sizes would be, respectively, $l = 135, 192, 248$ bits.

One more thing to be considered in the scenario where an attacker is trying to provide a valid text/MAC pair is the number of trials allowed. In other words, the higher the number of trials under the same key, the higher the change of the attacker to succeed. Following the convention in [NISTSP800107], let $2'$ be the number of failed trials allowed by the system under the same MAC key, and $l$ the length of the MAC tag. Then, the probability of accepting a forged MAC as authentic is given by

$$P_{\text{Forg}} = 2^{(t' - l)}.$$  \hspace{1cm} (Equation 3.4)

In other words, $2'$ now represents the maximum number of frames ($n$) under the same key, so that $P_{\text{Forg}} = n / (2^l)$. By making $P_{\text{ue}} = P_{\text{Forg}}$, it is possible to determine $n$ in terms of the MAC length ($l$) and a CER-specific $P_{\text{ue}}$ as follows:

$$n = P_{\text{ue}} 2^l.$$  \hspace{1cm} (Equation 3.5)

Figure 14 shows a graphical representation of the number of frames ($n$) for TC in function of the tag length ($l$) and four different $P_{\text{ue}}$. The tag length in this figure varies from 48 to 256 bits. The red, blue, green and cyan curves respectively represents $P_{\text{ue}} = 10^{-9}$, and $10^{-16}, 10^{-22}, 10^{-28}$. The first $P_{\text{ue}} (10^{-9})$ represents the requirement for TC, whereas $10^{-16}, 10^{-22}, 10^{-28}$ represents the performance of BCH codes respectively for CER = $10^{-4}, 10^{-5},$ and $10^{-6}$.

![Figure 14: TC Number of Frames in Function of the Tag Length ($P_{\text{ue}} = 10^{-9}, 10^{-16}, 10^{-22}, 10^{-28}$)](image)

By the same token, Figure 15 illustrates the TM number of frames ($n$) for $P_{\text{ue}} = 10^{-9}, 10^{-41}, 10^{-58}, 10^{-75}$, respectively represented by the red, blue, green and cyan curves. The first
$P_{ue} (10^{-9})$ represents the requirement for TM, whereas $10^{-41}$, $10^{-58}$, $10^{-75}$ represents the performance of RS codes for CER = $10^4$, $10^5$, and $10^6$, respectively.

![Figure 15: TM Number of Frames in Function of the Tag Length ($P_{ue} = 10^{-9}$, $10^{-41}$, $10^{-58}$, $10^{-75}$)]

### 3.2.7 Updated Authentication Cryptoperiods

Let’s now take into account authentication mechanisms utilized by ESA, namely CMAC and GCM schemes. These mechanisms utilized the tag size as big as the block of underlying cipher. Because AES is used, the following results utilize a tag length of 128 bits, i.e. $l = 128$. Consequently, it is possible to compute the maximum number of frames ($n$) that should be transmitted through the TC and TM links under the same authentication key. The results are listed in Table 26.

Notice that the computation of the maximum number of frames is an intermediate step in the determination of the cryptoperiods. Based upon the maximum number of frames to be transmitted, as well as upon the datarate of each specific link, it is then possible to determine the maximum time frame that a given cryptographic key should be utilized, i.e. the cryptoperiod for a specific link. In addition, the consideration of number of frames allows for a more straightforward graphical analysis, independently of the link considered. If cryptoperiods were considered instead, the graphical analysis would have to be performed for each individual link. Note that TC and TM links utilize respectively BCH and RS coding schemes. Therefore, for the sake of simplicity, Table 26 lists $n$ in function of the requirement of each coding scheme and their performance in face different CERs.
Table 26: Maximum Number of Frames for TC’s and TM’s Authentication Mechanisms

<table>
<thead>
<tr>
<th>Link</th>
<th>Requirement ((P_{ue}=10^9))</th>
<th>Maximum Number of Frames ((n))</th>
<th>Coding Performance for CER=10^{-4}</th>
<th>Coding Performance for CER=10^{-5}</th>
<th>Coding Performance for CER=10^{-6}</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC (BCH)</td>
<td>3.40x10^{29}</td>
<td>7.28x10^{22}</td>
<td>7.38x10^{16}</td>
<td>7.38x10^{10}</td>
<td></td>
</tr>
<tr>
<td>TM (RS)</td>
<td>3.40x10^{29}</td>
<td>8.10x10^{-13}</td>
<td>9.46x10^{-20}</td>
<td>9.63x10^{-37}</td>
<td></td>
</tr>
</tbody>
</table>

Finally, by utilizing the communications scenarios presented in Table 25 along with the maximum number of frames listed in Table 26 it is possible to compute the cryptoperiods for authentication mechanisms specifically for both TC and TM links, as shown in Table 27. It can be observed a wide range of cryptoperiods in Table 27. Depending on the scenario considered, the cryptoperiods may vary from \(10^{-24}\) to \(10^{-47}\) days.

Table 27: Authentication Cryptoperiods for TC and TM

<table>
<thead>
<tr>
<th>Orbit</th>
<th>Link</th>
<th>Scenario</th>
<th>Requirement ((P_{ue}=10^9))</th>
<th>Cryptoperiod ([\text{days}]) CER=10^{-4}</th>
<th>Cryptoperiod ([\text{days}]) CER=10^{-5}</th>
<th>Cryptoperiod ([\text{days}]) CER=10^{-6}</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEO</td>
<td>TC</td>
<td>Distributed Attacker</td>
<td>6.15x10^{22}</td>
<td>1.32x10^{16}</td>
<td>1.34x10^{10}</td>
<td>1.34x10^{4}</td>
</tr>
<tr>
<td></td>
<td>HK TM</td>
<td>Nominal/Attacker</td>
<td>2.22x10^{24}</td>
<td>5.27x10^{8}</td>
<td>6.16x10^{25}</td>
<td>6.27x10^{42}</td>
</tr>
<tr>
<td></td>
<td>PL TM</td>
<td>Nominal/Attacker</td>
<td>1.38x10^{20}</td>
<td>3.30x10^{12}</td>
<td>3.85x10^{29}</td>
<td>3.92x10^{46}</td>
</tr>
<tr>
<td>GEO</td>
<td>TC</td>
<td>Attacker</td>
<td>9.85x10^{24}</td>
<td>2.11x10^{17}</td>
<td>2.14x10^{41}</td>
<td>2.14x10^{55}</td>
</tr>
<tr>
<td></td>
<td>HK TM</td>
<td>Nominal/Attacker</td>
<td>2.22x10^{21}</td>
<td>5.27x10^{8}</td>
<td>6.16x10^{25}</td>
<td>6.27x10^{42}</td>
</tr>
<tr>
<td></td>
<td>PL TM</td>
<td>Nominal/Attacker</td>
<td>1.50x10^{19}</td>
<td>3.58x10^{13}</td>
<td>4.18x10^{30}</td>
<td>4.25x10^{47}</td>
</tr>
</tbody>
</table>

Let’s first consider the TC results. For LEO, the shortest cryptoperiod is \(1.34x10^{4}\) (CER=10^{-6}), which represents almost 37 years. It can also be observed that cryptoperiods are inversely proportional to the performance of the coding mechanisms (BCH codes). For CER=10^{-4}, for example, the cryptoperiod becomes \(1.32x10^{16}\) days. Moreover, for the TC link requirement \((P_{ue}=10^9)\), as determined in section 9.5.1 of [CCSDS2301G2], the cryptoperiod reaches the order of \(10^{22}\) days. Cryptoperiods for GEO missions follow exactly the same trend as the LEO ones, but are one order of magnitude bigger. For instance, the cryptoperiod to protect TC links of GEO missions, following the requirement \((P_{ue}=10^9)\), is \(9.85x10^{24}\) days.

Differently from TC, [CCSDS1301G2] does not specify the requirements for undetected error rates in TM frames. It has been assumed for TM the same undetected error rate in frames \((P_{ue}=10^9)\) as the one recommended for TC. Ideally, it would be important to have this information specified in [CCSDS1301G2] so that precise cryptoperiods could be computed for TM. For \(P_{ue}=10^9\), the cryptoperiod for HK TM links on both LEO and GEO missions are the same: \(2.22x10^{24}\) days. Note that HK TM links are considered in this study to have the same speed in both LEO and GEO, therefore the same cryptoperiods. Furthermore, PL TM links in LEO and GEO missions have, respectively, cryptoperiods in the order of \(10^{20}\) and \(10^{19}\) days. Again, this represents a huge number of years. On the other hand, if the
performance of RS codes are considered (CER=10⁻⁴, 10⁻⁵, and 10⁻⁶) the proposed approach to compute cryptoperiod does not hold. It is possible to notice in Table 27 that the biggest cryptoperiod achieved is 5.27x10⁻⁸ days. This represents a cryptoperiod of 4.5ms, which, in other words, are not of practical use. Besides, this happens for all cryptoperiods computed based on the code performance, as shaded in Table 27. From another perspective, the undetected error rate in TM frames is extremely low, which makes the performance of the coding mechanism (RS) too high to be matched by the security mechanisms utilizing the current key sizes. Increasing the strength of the cryptosystem could help balancing the equation, however that would impose higher costs on the security end. Furthermore, that would only become necessary if the performance of the coding mechanism is followed. If the requirement is taken as reference, the cryptoperiods are presented above have shown to be enough to protect the missions for a huge number of years.

3.2.8 Conclusions and Recommendations

This section provided a complement to a topic of study investigated in phase 1 of this project. New equations for $P_{\text{Forg}}$ and $P_{\text{Err}}$ were investigated. As a result, it was discovered that the bound on the probability reported in [2007Nandi] could not be used as a probability. Furthermore, based on the assumption that each frame should be authenticated, it started to make more sense to us to investigate the probability of having a frame forged, instead of a set of frames. Thus, instead of computing cryptoperiods in terms of $P_{\text{Err}}$ (for a set of frames), it was now computed based upon $P_{\text{ue}}$ (single frames). This change of approach was crucial to allow for a better analysis involving number of frames to be transmitted, tag lengths, and different $P_{\text{ue}}$.

A new set of cryptoperiods were calculated for a wide variety of scenarios. As described in the previous section, the whole set of cryptoperiods for TC indicates that authentication keys could live for a huge number of years, long enough to protect (practically) the vast majority of missions. On the other hand, the study for TM presented some issues. First, in [CCSDS1301G2] does not specify the undetected error rate for TM frames. Consequently, we utilized the TC requirement. It is important to notice that TC and TM utilize different coding mechanisms (respectively BCH and RS), different frame sizes, among other parameters. Therefore, it might be possible that TM demands a different requirement than the one for TC. Hence, it would be very useful if CCSDS could specify a requirement for TM in [CCSDS1301G2], similarly to what has been done for TC in [CCSDS2301G2]. A second issue found in TM is that RS codes have very good performance (extremely low probability of undetected errors). Consequently, it was noticed that the approach utilized to compute cryptoperiods for TC links could not hold for the TM ones when the performance of the coding mechanism was taken into account. All resulting cryptoperiods were a fraction of a second, which do not make sense in a practical setting. Cryptoperiods for TM are of some significance only when the TC requirement was utilized ($P_{\text{ue}}=10^{-9}$). Again, in order to obtain more precise results, it would be important to define the requirements for TM. For the time being, we will continue to use the TC requirement for TM, until we have some formal publication determining parameters specific for TM.

In summary, cryptoperiods for TM that made no sense were discarded. The remaining ones, for both TC and TM, were long enough to protect the mission for a huge number of years. Obviously, it would not be recommended to keep a key in use for such a number of
years. It would be recommended, though, to change keys as often as possible. Ideally in every communication session. Thus, it would be very interesting to have a study on the implications of key management on onboard resources and operational requirements. The next sections presents some study cases on the onboard platform needed to support key management. In addition to that, it would be very interesting to determine the minimum acceptable cryptoperiods based upon operational costs. It can be foreseen that operators would have a tendency not to change keys. However, such a study could provide some valuable input on how cryptographic keys should be managed.
3.3 Trusted Platform to Support Key Management

This section presents an approach based on trusted platform modules (TPMs) to perform key transport to destinations SCs. Key management goals tailored to space systems are defined and related work are presented. Next, the communication scenarios are listed and the set of secret information necessary to support key management is introduced. Two rekeying methods are described in detail along with two techniques to perform onboard integrity checks after the completion of the rekeying operations. Lastly, a complexity analysis based upon the study of a constellation of SCs is presented in order to determine resources needed to support the proposed key management approach, such as onboard storage, number of rekeying operations, among other parameters.

3.3.1 Key Management Goals and Infrastructure

Space systems differ from their ground counterparts in many different ways. As reported in Section 3.3.2, it is not trivial to port approaches devised for general networks on the ground to the space segment. Therefore, several particularities of space systems have been taken into account while devising the design criteria for the proposed key transport scheme. Two main goals are the minimization of CC communications (time and data) and platform requirements (computation and storage to distribute keys to a constellation of SCs).

The first goal is of utmost importance in the event of limited communications capabilities between the SC and the CC. This is the case in contingency scenarios where, for instance, the baud rate of communication channels is reduced. In addition, a limited time-window to communicate with a destination SC does not allow for long communication sessions to be dedicated to key management operations. Besides, in critical scenarios it may be the case that the CC is completely unavailable. To support the latter case, it is necessary to have some in-orbit resources that would allow for higher autonomy of the SCs to perform key management functions so that secure communications are not disturbed.

The minimization of the infrastructure requirements to support key management in a constellation of SCs is also crucial. Unlike general-purpose computing systems on the ground, the onboard platform (such as those in microsatellites) has limitations on the computational power and storage. Hence, the simpler the computation to be performed, the better. By the same token, it is important to minimize the amount of onboard storage required for key management. Furthermore, in some extreme scenarios the SC may not have a secured channel with the CC to perform key transport. Therefore, it is interesting to provide means to allow the CC to upload new keys to the SC and establish a secure channel from scratch.

These goals may be achieved with the proposed trusted platform onboard the SCs utilizing symmetric cryptography. In addition, a SC is selected to serve as a Master Spacecraft (MSC) to support higher levels of autonomy for the whole constellation. In the absence of the CC, the MSC takes the role of a key distribution center (KDC) in orbit. Even though the MSC may have its own payloads and functions within the constellation, it is assigned one additional task, specifically to act as KDC performing key management. This comprises the temporary storage of keying material onboard its own TPM. According to the demand of cryptographic keys by the destination SCs, the MSC will opportunistically forward keying material to support the continuous maintenance of security services. Even
though the MSC is launched with a set of secret materials onboard, the demand for keys may be high enough to require the CC to upload additional key material to the MSC after launch. This functionality is also supported. It has been currently considered that a single MSC is present in the constellation. However, more advance approaches providing redundancy are envisioned and are the subject of further investigation in this research project.

Yet another important feature investigated in this research is the integrity check of keying materials uploaded to the MSC and SCs. Once new keying materials are uploaded to the MSC or its destination SC, it is crucial to securely check the integrity of received information prior to its utilization. This is also applicable to the scenario where it is necessary to check sets of secret data loaded onto the SC memories prior to launch, without the disclosure of its contents.

### 3.3.2 Related Work

This section reviews related work in the areas of general adhoc networks, including vehicular adhoc networks (VANETs), mobile adhoc networks (MANETs) and wireless sensor networks (WSNs). The focus of this section is on key management cryptographic functions and processes.

A VANET generally is a network of vehicles, each with keys preassigned by a Certificate Authority (CA), which signs them and cryptographically associates the keys with the vehicle. The objective is to support vehicles sending messages each with an attached signature (supported with asymmetric cryptography). Privacy in this network is defined as the inability to associate a signature with a particular vehicle or group of vehicles. It has been shown that key assignment schemes that ensure no single key is used by more than one vehicle are advantageous [2009Ha]. Vehicles can easily be preloaded with many keys, hence key reuse is not an issue.

Advantages of asymmetric cryptography in MANETs largely come from the use of fewer keys. Asymmetric cryptography is primarily utilized for short messages. Generally, longer messages utilize symmetric cryptographic techniques. Some symmetric cryptographic key management schemes for MANET include predistributed key schemes, key infection, and peer intermediary key establishment [2012Da]. In the later scheme, it is assumed with some probability that each pair of mobile nodes shares a secret key with at least one or more intermediaries, and a process is executed to try to find out which secret key is shared. The key infection scheme assumes all nodes are trusted and broadcasts until another node responds to it. Other schemes rely upon a trusted authority to distribute keys or employ \((k,n)\) threshold cryptography where any \(k\) nodes in the group of \(n\) nodes can generate private keys from their shares of the master key. Other schemes have also been proposed for MANETs including logical key hierarchy (LKH) which was originally proposed for satellite systems. In this scheme a group key concept is used to distribute secret data to more than one satellite (instead of delivering the secret data \(i\) times to each \(SC_i\)), thus saving bandwidth. In general, most schemes do not address the communication capacity or bandwidth efficiency or robustness against link loss under a given power consumption of MANETs which is likely more important than the current focus on energy consumption and computational power [2006He].

Key management in ad hoc networks includes asymmetric as well as symmetric schemes. Secure pebble networks [2002Fo] is an example of a symmetric approach. In this
scheme, it is assumed that all nodes have a protected copy of the secret group identity key $K_g$ whose cryptoperiod is the entire network lifetime. A node is selected within the group to serve as a ‘cluster head’ which has an identifier, $ID_c$. Once this cluster head node is established, a cluster key ($K_c$) is distributed using $E_{Kh}(ID_c|K_c|MAC_{K_g}(ID_c|K_c))$ such that $K_h = H(K_g)$, where $E_{k}(.)$ is symmetric encryption, $MAC_{k}(.)$ is a MAC, $H(.)$ is the hash function, and ‘|’ is concatenation. The cluster key is then utilized to distribute traffic encryption keys ($K_{tek}$) to all nodes within the group, using $E_{Kc}(ID_c|K_{tek}|MAC_{K_g}(ID_c|K_{tek}))$.

This process is repeated after a period of time, for example, the new cluster head is established, and new traffic keys are then distributed. Its main weaknesses are that nodes must have tamper-resistant storage for the shared secret group identity key ($K_g$), and additionally only group authentication is supported [2005Ge].

WSNs include RFID tags and networks of devices with very limited computation and memory. Secure control of access to the devices tag is essential. For example, the data tag must be protected and when the tag value is computed by another trusted node it is used to index a key. Typically, networks are composed of 1,000 to 10,000 nodes each storing 30–100 keys of 128-bits. Original proposals suggested communicating a hash (or modulo multiplication) of the tag ($H(ID)$, where the tag is $ID$), however this scheme would allow someone to track the device. Thus it was suggested that tags would choose a random value, $r$, and communicate ($r, H(ID|r)$) which prevents tracking. This however complicates the search for the correct key [2009La]. Other schemes have also been proposed including tree-based key spaces.

### 3.3.2.1 Comparison with Related Work

The symmetric scheme proposed in this research can be compared to previous work. Previous key management schemes are forced to utilize asymmetric cryptography due to the unacceptably large number of symmetric keys which would be required for their large network size. In contrast, the proposed use of symmetric cryptography keeps the key management simple while the number of keys remains manageable since the satellite constellations are assumed to be at most 100 nodes. In addition, the simpler computations of symmetric cryptography provide lower energy consumption than their asymmetric counterparts, which may be critical for LEOs and microsatellites. Yet another serious issue in utilizing asymmetric mechanisms in delay tolerant networks (DTNs) is the propagation of certificate revocation lists. As there is no time guarantee of package delivery, certificate lists can easily get outdated therefore affecting the security of the system.

A general weakness of symmetric cryptography is the lack of source authentication [2006He]. In our proposal source authentication is strengthened by the utilization of a Trusted Platform Module (TPM) which provide mechanisms for source authentication. In addition, incoming satellite signal information can typically provide additional data useful in identifying the source. In general, source authentication is appropriate for identifying the CC, MSC and even SCs.

The indexing of shared secrets can be compared to previous research which utilized a hash to index keys [2009La]. In the proposed scheme, a long random word is utilized to index providing a larger amount of brute force security. Replay attacks are not possible since the indexes are removed once they are utilized. The use of a hash would be possible but would reduce the security since hash lengths would be smaller than the bit size of the proposed index.
A single MSC may be a possible single point of failure in the proposed key management scheme, hence multiple MSCs may be appropriate. The robustness of the proposed technique is good, for example there is no disruption caused by sending an index multiple times. Although a quantitative comparison with previous research utilizing size and number of messages is difficult to perform, it will be investigated in the future. This kind of comparison is also currently lacking in the literature. Some previous research utilizes hashes or modulo multiplication to generate new keys unlike this proposal which transports keys. The advantage of the proposal approach is that keys may be guaranteed with respect to security properties, unlike the case of the key generation from a hash or multiplication.

3.3.3 Communication Scenarios

The communication scenarios described below assumes that all SCs have means to communicate with each other. However, there is no guarantee regarding the elapsed time-frame between two opportunistic communication sessions. The same assumption is valid for communications between the CC. There are two main communication scenarios identified:

Regular Communications:

Regular communications are those involving transfers of payload data, commands, and HK telemetry data between the CC and the SC, as well as between two SCs (including the MSC). The links utilized for regular communications are TC, TM HK, and TM PL.

Rekeying Operations:

Rekeying operations are communication sessions with the goal of transferring fresh keying material to a destination SC. Since the MSC is a special denomination for a SC that possesses key management capabilities, all operations related to regular SC also applies to the MSC. This type of transfer can be classified as direct and indirect.

Direct transfers occur only between the CC and the destination SC, and transport of keying material does not involve intermediate nodes. Indirect transfers involve the MSC, which keeps the keying material under custody until its final delivery to the destination SC. This kind of transfer occurs in two steps. In a first moment, the CC opportunistically uploads sets of keying material onboard of the MSC. Later on, whenever the destination SC is in line of sight, the MSC forwards the packages of keying material to its final destination.

Rekeying operations are not expected to happen very often, which may vary to months to decades, as exemplified in the study cases shown in Section 3.3.10. Additionally, rekeying operations can be divided into several rekeying sessions. The number of sessions needed depends on the link bandwidth and on the time frame that the destination SC is in line of sight. The links involved with rekeying operations are the TC and TM HK links.

3.3.4 Secret and Keying Materials

There are four Sets of Secret Material (SSMs) utilized in the proposed key transport scheme: One-Time Secrets (OTSS), Transport Encryption Keys (TEKs), Transport Authentication Keys (TAKs), and Keyed Hash Key (KHKs). All SSMs are generated on the ground and uploaded to the SC prior to launch. Additionally, there are the Symmetric Keys (SKs), which are organized and transported as Sets of Secret Keys (SSKs).
TEKs and TAKs are responsible respectively for the establishment of encrypted and authenticated channel between two entities. Through this secured channel, keying material is securely transported to the destination SC. TEKs and TAKs are used once and then discarded. KHKs, in turn, are utilized to perform integrity checks on the SSKs uploaded to the SCs. SKs are the keys utilized by encryption and authentication mechanisms to secure regular communications.

In order to allow for the SC to operate for a certain amount of time after launch, a number of SSKs are launched onboard the SC. Whenever rekeying transfers happen, one or multiple SSKs are transferred to the SC. The advantage of transmitting more than one SSK is the better utilization of the OTSs onboard. OTSs are random numbers utilized once and then discarded. Hence, the more SSKs are transmitted in each rekeying operation, the least OTSs have to be stored onboard to support key transport. Notice that there is no provision in the proposed approach to upload new SSMs (which include OTSs) after launch. This is a research topic to be covered in the next phase of this project.

Two entities (CC, SC, or MSC) who wish to perform a rekeying operation must have the same OTS within its onboard platform. The CC is responsible to initiate the process with the MSC and destination SCs. During rekeying of SCs, the MSC is responsible to initiate the process. Each OTS can be associated with a set (or subset) of keying material (TEKs, TAKs, SKs, and KHKs). From another perspective, the goal of the OTS is to obtain the associated information from within the TPM. Right after the utilization of an OTS, the TPM permanently removes it from its internal OTS table. Consequently, the hardware guarantees that there are no means to reutilize an OTS in a replay attack. Detailed information on the TPM operation and security are extensively discussed in [2008JG, 2013JG].

In order to ensure higher reliability, it is important to guarantee the integrity of the SSKs uploaded to the SC. That can be achieved by computing a keyed hash over the transmitted SSKs, and such an operation requires KHKs. Sections 3.3.7 and 3.3.8 provide further information on integrity checking operations.

As previously mentioned, it is possible to either transmit a single or multiple SSKs to the MSC/SC. Whenever the second case is utilized, multiple SSKs are concatenated therefore allowing for a single KHK to be used in the integrity check.

The following functions are used in the protocols throughout this document:

- $E_{TEK}(\cdot)$: Encryption operation using TEK.
- $D_{TEK}(\cdot)$: Decryption operation using TEK.
- $MAC_{TAK}(\cdot)$: Generation of a MAC tag utilizing TAK, with subsequent storage and/or transmission to the communication channel.
- $MAC^*_{TAK}(\cdot)$: Computation of a MAC tag utilizing TAK over the information received via the communication channel in order to check its validity.
- $TPM_{RetrKeys}(OTS, SID)$: TPM function to retrieve a TEK/TAK pair associated with the OTS. If OTS is valid, outputs TEK/TAK and associates SID with TEK/TAK; Else, abort rekeying operation.
- $TPM_{StoreSSK}(SID, SSKs)$: Stores SSKs within the TPM and associates it with SID. A single SID can be associated with multiple SSKs.
- $TPM_{LoadSSK}(SID)$: Retrieves the SSKs associated with a given SID.
Indexes $i$, $j$, and $k$ are used to indicate association between OTSs, TEKs and TAKs. Index $n$ is used to indicate which SSK is being referred to in the protocol. Symbol $|$ represents concatenation.

Notice that even though this section mentions communication sessions between the CC and the SCs, this should be implicitly assumed that all communication coming out of the CC will be sent to Ground Stations (GSs), and the latter will establish the actual link with the SC.

### 3.3.5 Rekeying Procedure with Hop-by-Hop Secrecy

The rekeying procedure described below is based on a system configuration comprised of a CC, a MSC and a single SC. Figure 16 illustrates the initial SSM setup of the CC and the onboard TPMs to support key transport operations.

![Figure 16: Initial System Configuration to Support Key Transport](image)

A constellation of destination SCs is omitted in the next sections, without loss of generality, as they follow exactly the same procedure. The proposed key upload is based on the principle that CC and the MSC share the same set of OTS. Besides, it is assumed that the MSC and the destination SC share another set of OTS. Additionally, each OTS $i$ is associated with a pair of keys TEK$_i$ and TAK$_i$. Every rekeying operation is linked with a Session Identifier (SID) so that it is possible to keep track of the associations between TEKs/TAKs and the SSKs involved in a given key transport operation. For instance, whenever an entity wants to determine which TEK/TAK pair is linked with a certain SSK, it will use the SID as a link to retrieve both TEK/TAK and SSK. Without the SID, it is not possible to discover which TEK/TAK pair was used for encryption and authentication of SSKs.

#### 3.3.5.1 Key Transport Protocol

The protocol utilized to transport SSKs from the CC to the MSC is described below. It is assumed that the initial communications are performed in the clear due to the lack of an initial secured channel, as it may be the case in contingency scenarios. This initial “in-the-clear” information does not compromise the security of the system due to the way that the TPM receives, handles and processes the OTSs.
Replay attacks are forbidden directly by the trusted platform. If an attacker tries to compromise the system by guessing OTSs, it would be necessary to perform an exhaustive search over the space indexed by the OTSs. That, by itself, would be infeasible and guaranteed by the size of the OTS. On top of that, there is always the processing time required by the trusted platform to process the OTS, which naturally imposes an additional time-delay.

Additionally, if necessary, it is trivial to penalize illegitimate users trying to compromise the protocol. In this case, if an OTS is not received correctly for \( N \) times, a penalty delay \( T \) would be imposed after each subsequent trial. Consequently, an attacker could start an exhaustive search, but would not proceed too much as the penalty delay would be triggered after \( N \) trials. A legitimate entity would also be capable of sending valid secrets and eventually perform the desired operation before \( N \) trials have been reached. The penalty delay would end after a certain amount of time \( T \). Moreover, parameters \( N \) and \( T \) would have to be specified according to the space mission security and availability requirements. For example, some context may favor minimum \( N \) and larger \( T \), while others would not tolerate a large \( T \) and would have to increase \( N \) to achieve a more satisfactory trade-off. It would be recommended, though, to choose \( T \) in a way that could give a chance for the ground station to (potentially) communicate with the spacecraft within its overpass time frame. Obviously, if an attacker is situated in the neighborhood of the ground station, a denial of service attack could potentially happen.

On the one hand, this approach would provide increased resistance against exhaustive search with minimum impact on implementation complexity, i.e. only a timer is needed to create such a delay. On the other hand, its main drawback is that illegitimate trials would trigger delays that could also penalize legitimate system users (depending on their ground location). If an attacker is located close enough to a GS, the spacecraft could overpass that GS with the delay active. If such a delay is too long, the spacecraft may be already out of reach of the ground station at the moment that the timer elapses. Thus, there would be necessary to wait for the next spacecraft overpass to perform the protocol, luckily without the interference of a neighboring attacker. Such a scenario would not favor quick recoveries in contingency situations.
### Protocol 3.1: Key Transport Between the CC and MSC with Subsequent Rekeying of SC

<table>
<thead>
<tr>
<th>Step</th>
<th>Entity</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>CC</td>
<td>((\text{TEK}_i, \text{TAK}<em>i) = \text{TPM}</em>{\text{RetKeys}}(\text{OTS}_i, \text{SID}_i));</td>
</tr>
<tr>
<td>2.</td>
<td>CC → MSC</td>
<td>(\text{OTS}_i, \text{SID}_i)</td>
</tr>
</tbody>
</table>
| 3.   | MSC | \((\text{TEK}_i, \text{TAK}_i) = \text{TPM}_{\text{RetKeys}}(\text{OTS}_i, \text{SID}_i)\);  
\[p1 = E_{\text{TEK}}(\text{OTS}_i - 1);\]  
\[t1 = \text{MAC}_{\text{TAK}}(E_{\text{TEK}}(\text{OTS}_i - 1)).\] |
| 4.   | MSC → CC | \(p1, t1\) |
| 5.   | CC | \(D_i = D_{\text{TEK}}(p1);\)  
\[M'_i = \text{MAC}_{\text{TAK}}^*(p1);\]  
If \(D_i = (\text{OTS}_i - 1)\) and \(t1 = M'_i\), Then key is confirmed, \(\text{SSK}_i = \text{TPM}_{\text{LoadSSK}}(\text{SID}_i)\);  
\(p2 = E_{\text{TEK}}(\text{SSK}_i);\) \(t2 = \text{MAC}_{\text{TAK}}(E_{\text{TEK}}(\text{SSK}_i))\); Else reinitiate rekeying operation (go to Step 1 and repeat). |
| 6.   | CC → MSC | \(p2, t2\) |
| 7.   | MSC | \(D_j = D_{\text{TEK}}(p2);\)  
\[M''_j = \text{MAC}_{\text{TAK}}^*(p2);\]  
If \(M''_j = t2\), Then \(\text{TPM}_{\text{StoreSSK}}(\text{SID}_j, \text{SSK}_j)\); Else request retransmission from CC and go to Step 6. |
| 8.   | MSC | \(\text{TPM}_{\text{RetKeys}}(\text{OTS}_j, \text{SID}_j)\);  
\[M_j = \text{MAC}_{\text{TAK}}(E_{\text{TEK}}(\text{OTS}_j - 1));\] |
| 9.   | MSC → SC | \(\text{OTS}_j, \text{SID}_j\) |
| 10.  | SC | \(\text{TPM}_{\text{RetKeys}}(\text{OTS}_j, \text{SID}_j);\)  
\[p3 = E_{\text{TEK}}(\text{OTS}_j - 1);\] \(t3 = \text{MAC}_{\text{TAK}}(E_{\text{TEK}}(\text{OTS}_j - 1));\) |
| 11.  | SC → MSC | \(p3, t3\) |
| 12.  | MSC | \(D_j = D_{\text{TEK}}(p3);\)  
\[M'_j = \text{MAC}_{\text{TAK}}^*(p3);\]  
If \(D_j = (\text{OTS}_j - 1)\) and \(t3 = M'_j\), Then key is confirmed, \(\text{TPM}_{\text{LoadSSK}}(\text{SID}_j)\);  
\(p4 = E_{\text{TEK}}(\text{SSK}_n);\) \(t4 = \text{MAC}_{\text{TAK}}(E_{\text{TEK}}(\text{SSK}_n))\); Else reinitiate rekeying operation (go to Step 8 and repeat). |
| 13.  | MSC → SC | \(p4, t4\) |
| 14.  | SC | \(D_{\text{TEK}}(p4);\)  
\[M''_j = \text{MAC}_{\text{TAK}}^*(p4);\]  
If \(M''_j = t4\), Then \(\text{TPM}_{\text{StoreSSK}}(\text{SID}_j, \text{SSK}_n)\); Else, request retransmission from MSC and go to Step 13. |
3.3.5.2 Step-by-Step Protocol Illustration

The step-by-step illustrations shown below assumes that the CC and the MSC share a set of OTS [OTS\(_1\), OTS\(_2\), ..., OTS\(_{10}\)], and the MSC and the SC (as well as the CC since it preloaded this data) share a second set [OTS\(_{11}\), ..., OTS\(_{20}\)]. In addition, it is assumed that each OTS\(_i\) is already associated with a TEK\(_i\)/TAK\(_i\) pair.

**Steps 1 and 2:**

The CC is the entity that initiates the key upload protocol. It starts by retrieving, for instance, OTS\(_1\), as depicted in Figure 17. Then the CC creates a unique SID, e.g. SID\(_1\), associates it with the TEK\(_1\)/TAK\(_1\) pair. Additional meta-information could be linked to the SID, such as identity of source and destination nodes, timestamp of the transfer, etc. Consequently, the SIDs could be made unique within the constellation of SCs. Next, the CC sends the SID\(_1\) and OTS\(_1\) pair to the MSC. As described in detail in [2013JG], this initial communication session can be done in the clear.

**Step 3:**

After receiving OTS\(_1\), the MSC’s TPM checks the authenticity of such a secret. If OTS\(_1\) is a valid secret, the TPM outputs the two keys associated with it: TEK\(_1\) and TAK\(_1\). The TPM also associates the received SID with the TEK/TAK pair. At the same time, OTS\(_1\) is permanently erased from the TPM internal table so that replay attacks are avoided. The CC does not necessarily need to delete OTS\(_1\) from its records. In fact, it may be beneficial to keep a record of the utilized OTS for future reference. In the figures below, the OTS on the CC side has been blacked out to indicate that the OTS has already been utilized.
If, for some reason, the OTS is not a valid secret, the protocol would have to be reinitiated. An OTS may have been corrupted while in transit, or an attacker is potentially trying to send his/her own (invalid) secret to the SC.

**Steps 4 and 5:**

At this point, the MSC can utilize $\text{TEK}_1/\text{TAK}_1$ to establish a secured channel with the CC, as represented by the pink pipe in Figure 19. In other words, the pink pipe represents that the information going through this channel has been encrypted under TEK and also possesses a MAC tag computed using TAK. Key confirmation is done at this point and should be executed as recommended in [2013JG]. This would allow the CC to certify that the MSC received the OTS, addressed its TPM tables correctly, and recovered the proper TEK/TAK pair. Besides, without any feedback from the MSC the CC cannot assume that the OTS has been received.
**Step 6:**

The CC employs TEK\textsubscript{1} and TAK\textsubscript{1} to encrypt and generate a MAC tag for SSK\textsubscript{1}. It also links SSK\textsubscript{1} to the SID\textsubscript{1}, and consequently to the TEK\textsubscript{i}/TAK\textsubscript{i} pair, as shown by the bold arrow linking SID\textsubscript{1} to SSK\textsubscript{1} in Figure 20. This is important to keep track of the association between SSKs and TEK/TAK pairs, as well as for future auditing purposes.

![Figure 20: Key Transport Protocol, Step 6](image)

**Step 6 (Alternative):**

Alternatively, the CC can upload multiple SSKs to the MSC which would provide a higher efficiency in terms of TEK/TAK pair utilization. In other words, the CC can utilize a single TEK/TAK pair to encrypt and generate MAC tags for multiple SSKs, as shown in Figure 21. In this case, SID\textsubscript{1} is linked with multiple SSKs, e.g. SSK\textsubscript{1} and SSK\textsubscript{2}.

![Figure 21: Key Transport Protocol, Step 6 (Alternative)](image)
**Step 7:**

The MSC receives, decrypts, and checks authentication of the SSKs packages. If those operations are successful, the SSKs are stored onboard. Figure 22 shows SSK$_1$ and SSK$_2$ stored onboard the MSC, as well as its association with SID$_1$.

![Figure 22: Key Transport Protocol, Step 7](image)

**Steps 8 and 9:**

The transport of SSKs from the MSC to destination SCs follows a very similar pattern to Steps 1 – 7. Whenever the destination SC is in line of sight, the MSC can initiate a transport of keying material. It starts with the MSC selecting one OTS that is shared with the destination SC, for instance OTS$_{11}$. A new SID is created for this transport operation, e.g. SID$_2$. Then, as shown in Figure 23, the MSC sends SID$_2$ and OTS$_{11}$ to the destination SC. The SC’s TPM receives OTS$_{11}$ and checks its authenticity.

![Figure 23: Key Transport Protocol, Steps 8 and 9](image)
**Steps 10 – 12:**

If OTS\textsubscript{11} is valid, then the SC’s TPM removes it from its internal table, outputs TEK\textsubscript{11}/TAK\textsubscript{11} and associates SID\textsubscript{2} with those keys. The TEK/TAK pair may now be used to send encrypted and authenticated messages to the MSC, as represented by the pink pipe in Figure 24. Likewise, Steps 4 and 5, key confirmation can happen at this point. After confirming that the SC correctly received OTS\textsubscript{11} and output the TEK/TAK pair, the MSC’s TPM can also erase OTS\textsubscript{11} from its internal OTS table.

![Figure 24: Key Transport Protocol, Steps 10–12](image)

**Step 13:**

The MSC encrypts SSK\textsubscript{1} and generate a MAC tag for the package being transmitted to the SC. An association between the TEK\textsubscript{11}/TAK\textsubscript{11} and SID\textsubscript{2} is also made at this point so that the MSC can later consult which key pair has been used to transmit a certain SSK. Notice that the SID can include additional metadata so that it is also possible to determine which destination SC received a given SSK. Even though Figure 25 shows a single SSK being uploaded (SSK\textsubscript{1}), due to the same reasons previously explained, the MSC could also transport multiple SSKs under the same TEK/TAK key pair.

![Figure 25: Key Transport Protocol, Step 13](image)
Step 14:

The destination SC receives the encrypted SSK₁ and its MAC tag, as shown in Figure 26. By possessing the current TEK/TAK pair used to communicate with the MSC, the SC can decrypt the package and compute its own MAC tag. If these operations finish successfully, the SKs within SSK₁ are made available to the onboard security mechanisms.

3.3.6 Rekeying Procedure with End-to-End Secrecy

Analyzing the approach in Section 3.3.5, it is possible to notice that the MSC has access to all the information being relayed to the destination SCs. Though it might be acceptable in some cases, there may be some scenarios where end-to-end secrecy is required, in which the CC and SC are the only entities knowing the contents of the SSK packages.

An advanced approach has been derived from the previous method that guarantees the secrecy of SSMs to intermediate nodes. This extended approach is also based on TPMs and OTSs, but adds an additional layer of security to ensure end-to-end secrecy, i.e. between the source CC and the destination SC. No additional infrastructural elements are needed to achieve such a requirement. However, it is mandatory to have the CC and the destination SC share a common set of OTS as well as an associated common set of TEKs/TAKs.

The initial system configuration is shown in Figure 27. Notice that this scenario, differently from the one shown in Figure 16, has a set of OTSs only known by the CC and the destination SC.
### 3.3.6.1 Key Transport Protocol

The protocol utilized to achieve end-to-end secrecy between the CC and the destination SC is introduced below. The final key confirmation between the CC and the destination SC can be done via MSC or directly with the CC. Protocol 3.2 shows key confirmation being performed directly with the CC. If the MSC is employed for final key confirmation, it would keep under custody all the key confirmation materials, which are encrypted and authenticated with a key pair only known by the CC and the SC.

#### Protocol 3.2: Key Transport with End-to-End Secrecy Between the CC and the Destination SC

<table>
<thead>
<tr>
<th>Step</th>
<th>Entity</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>CC</td>
<td>$TPM_{RerKey}(OTS_i, SID_i)$; $M_i = MAC_{TAKi}(E_{TEKi}(OTS_i - 1))$;</td>
</tr>
<tr>
<td>2.</td>
<td>CC → MSC</td>
<td>$OTS_i, SID_i$</td>
</tr>
<tr>
<td>3.</td>
<td>MSC</td>
<td>$TPM_{RerKey}(OTS_i, SID_i); p_1 = E_{TEKi}(OTS_i - 1); t_1 = MAC_{TAKi}(E_{TEKi}(OTS_i - 1))$</td>
</tr>
<tr>
<td>4.</td>
<td>MSC → CC</td>
<td>$p_1, t_1$</td>
</tr>
<tr>
<td>5.</td>
<td>CC</td>
<td>$D_i = D_{TEKi}(p_1); M_i' = MAC_{TAKi}^*(p_1)$</td>
</tr>
</tbody>
</table>

If $D_i = (OTS_i - 1)$ and $M_i' = t_1$, Then key is confirmed, $TPM_{LoadSSK}(SID_i)$; Else reinitiate rekeying operation.

Prepare $m_k = OTS_k, E_{TEKi}(SSK_n), MAC_{TAKi}(OTS_k | E_{TEKi}(SSK_n))$;

$p_2 = E_{TEKi}(m_k); t_2 = MAC_{TAKi}(m_k)$.  

<table>
<thead>
<tr>
<th>Step</th>
<th>Entity</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.</td>
<td>CC → MSC</td>
<td>$p_2, t_2$</td>
</tr>
<tr>
<td>7.</td>
<td>MSC</td>
<td>$m_k = D_{TEKi}(p_2)$; $M_i'' = MAC_{TAKi}^*(m_k)$</td>
</tr>
</tbody>
</table>

![Diagram of the Key Transport Protocol](image)

Figure 27: Initial Configuration for Key Transport with End-to-End Secrecy
If $M_{i'} = t_2$, Then $TPM_{StoreSSK}(m_k)$; Else, reinitiate SSK\textsubscript{n} upload.

8. MSC : $TPM_{RetrKey}(OTS\textsubscript{k}, SID\textsubscript{j})$

9. MSC $\rightarrow$ SC : $OTS\textsubscript{j}, SID\textsubscript{j}$

10. SC : $TPM_{RetrKey}(OTS\textsubscript{k}, SID\textsubscript{j})$

\[ p_3 = E_{TEK_j}(OTS\textsubscript{j} – 1); t_3 = MAC_{TAK_j}(E_{TEK_j}(OTS\textsubscript{j} – 1)). \]

11. SC $\rightarrow$ MSC : $p_3, t_3$

12. MSC: $D_j = D_{TEK_j}(p_3), M_j = MAC_{TAK_j}(p_3)$

If $D_j = (OTS\textsubscript{j} – 1)$ and $M_j = t_3$, Then key is confirmed, $TPM_{LoadSSK}(SID\textsubscript{j})$;

\[ p_4 = E_{TEK_k}(m_k); t_4 = MAC_{TAK_k}(m_k); \] else reinitiate rekeying operation.

13. MSC $\rightarrow$ SC : $p_4, t_4$

14. SC : $m_k = D_{TEK}(p_4)$;

\[ M_{j''} = MAC_{TAK_k}(m_k). \]

If $M_{j''} = t_4$, Then $TPM_{StoreSSK}(SID\textsubscript{k}, SSK\textsubscript{n})$; $p_5 = E_{TEK_k}(OTS\textsubscript{k} – 1)$,

\[ t_5 = MAC_{TAK_k}(E_{TEK_k}(OTS\textsubscript{k} – 1)); \] else, reinitiate SSK\textsubscript{n} upload.

15. SC $\rightarrow$ CC : $p_5, t_5$

16. CC : $D_k = D_{TEK_k}(p_5), M_k = MAC_{TAK_k}(p_5)$

If $D_k = (OTS\textsubscript{k} – 1)$ and $M_k = t_5$, Then key is confirmed; Else reinitiate rekeying operation.

### 3.3.6.2 Step-by-Step Protocol Illustration

The procedures illustrated below have the same set of assumptions as in Section 3.3.5.2, in which the CC and the MSC share a set of OTS [OTS\textsubscript{1},... ,OTS\textsubscript{10}], and the MSC and the SC share a second set [OTS\textsubscript{11},...,OTS\textsubscript{20}]. In addition, it is assumed that the MSC and the SC share a third set of OTS [OTS\textsubscript{21},...,OTS\textsubscript{30}], which are associated respectively with [TEK\textsubscript{21},...,TEK\textsubscript{30}] and [TAK\textsubscript{21},...,TAK\textsubscript{30}]. In order to achieve more objective illustrations, the following figures omit the SIDs as well as other information that are not relevant to the understanding of the proposed mechanism. The reason being is that SIDs are a few bytes long, nonces may potentially reach a few hundred bits, and metadata comprises some additional few bytes. Overall, the bulk of the key transport is SSKs which are thousand of bytes long. Thus, transport of SSKs is the main subject to be investigated, whereas SIDs, nonces and metadata can be omitted without compromising the completeness and precision of the descriptions. SIDs, nonces, etc, would function exactly as described in Section 3.3.5.2.

The CC utilizes OTS\textsubscript{1} and proceeds with Steps 1 through 5 (as described in Section 3.3.5.2) to establish a secured channel with the MSC. In order to assure secrecy of the SSK packages, the protocol proceeds differently from Step 6 on.
**Step 5 and 6:**

The CC retrieves OTS$_{50}$ and the TEK$_{50}$/TAK$_{50}$ pair. In the sequence, it uses those keys to encrypt SSK$_1$, as shown in Figure 28. The resulting package, represented by (SSK$_1$)$_{TEK50}$, is sent to the MSC along with OTS$_{50}$. A MAC tag is also computed over (SSK$_1$)$_{TEK50}$ and is sent to the MSC as well.

![Figure 28: Key Transport Protocol with End-to-End Secrecy, Steps 5 and 6](image)

**Step 7:**

The MSC decrypts and checks authentication tags on the packages received under the TEK$_1$/TAK$_1$ pair. Notice in Figure 29 that the received package is encrypted under TEK$_{50}$/TAK$_{50}$. Given that the MSC does not have the keys associated with OTS$_{50}$ within its TM, it cannot decrypt SSK$_1$ to read its contents. This is how it is guaranteed that the MSC is unable to read the packages intended to another SC, i.e. how end-to-end secrecy is obtained. The MSC stores the encrypted SSKs onboard for further distribution to the constellation.

![Figure 29: Key Transport Protocol with End-to-End Secrecy, Step 7](image)
Step 8 and 9:

Whenever the destination SC is in line of sight, its rekeying is started. In preparation to establishing a secured channel with the SC, the MSC now utilizes OTS_{11} to retrieve TEK_{11}/TAK_{11}, as illustrated in Figure 30.

Figure 30: Key Transport Protocol with End-to-End Secrecy, Steps 8 and 9

Step 10 – 13:

Next, by following Steps 10, 11, and 12, the MSC establishes a secured channel with the SC. Then, the MSC sends to the SC the encrypted (SSK_{1})_{TEK_{50}} that it had under custody along with the associated OTS_{50}. Notice that this is done through a secure channel (pink pipe in Figure 31). Information stored and transferred under custody is never sent out in the clear. Rather, they are transmitted through a secure channel established between the two communicating entities.

Figure 31: Key Transport Protocol with End-to-End Secrecy, Steps 10–13
**Step 14:**

At this point, as can be observed in Figure 32, the SC decrypts and checks authentication of the material that has been transferred under the custody of the MSC.

![Figure 32: Key Transport Protocol with End-to-End Secrecy, Step 14](image)

**Step 15 and 16:**

Key confirmation can be done directly between the SC and the CC, or via the MSC. This process utilizes the TEK$^{50}$/TAK$^{50}$ key pair along with OTS$^{50}$ to establish a secured channel between the SC and the CC, as shown in Figure 33.

![Figure 33: Key Transport Protocol with End-to-End Secrecy, Step 15](image)
3.3.6.3 Discussion on the Rekeying Procedures

The approach presented in Section 3.3.5 has a main advantage, resulting from the fact that all the SSKs uploaded to the MSC do not have to be linked to their destination SCs. Therefore, the MSC has a higher degree of freedom while distributing the SSKs. In this case, the MSC can keep a single pool of SSKs onboard and distribute to the SCs on demand, without worrying with the ownership of the SSKs.

The approach presented in this section has the benefit of providing end-to-end secrecy in terms of the contents of each SSK. However, it can be observed that each SC will demand its own pool of SSKs to be stored onboard the MSC.

Keeping a single pool of SSKs on the MSC allows for lower complexity in terms of management, as there is no discrimination to which destination SC a given SSK will be uploaded. In practice, it is not expected that all SCs have the same SK consumption rate. As a result, it may be the case that a certain number of SCs require their pool to be refilled more frequently than others. Hence, a more complex management of the SSKs onboard the MSC is expected.

3.3.7 Hop-by-Hop Onboard Integrity Check

Once the SSKs are transported, it is fundamental to have means to check the integrity of the SSK packages stored onboard of the destination SC. Without such a check, it cannot be guaranteed that the SSKs are in an integer state prior to its utilization. This is especially important if the keys are to be stored onboard for a long period of time, which may leave them susceptible to corruption caused by radiation effects. Two methods are proposed to support integrity checking. This section introduces a Hop-by-Hop method, whereas the next section describes an End-to-End approach.

The integrity check of a SSK (or multiple SSKs) can be executed through the computation of a MAC, utilizing a KHK provided by the TPM. Therefore, in order to support integrity checking operations, each OTS has to be associated with a KHK in addition to the TEK/TAK pair. Furthermore, a nonce is used to ensure freshness in MAC re-computations over the same SSK utilizing the same KHK. The MAC computation is represented by $MAC_{KHK}(\text{Nonce} | \text{SSK})$.

The protocols remain the same as the ones presented in Sections 3.3.5.1 and 3.3.6.1. The difference now is that, once the SSKs are stored onboard a MAC is computed over it. Thus, the resulting MAC tag is sent back to the source entity in the end of the protocol execution. The following discussions assume that both CC and MSC share a set of KHKs, e.g. [KHK1,...,KHK10].
3.3.7.1 Step-by-Step Protocol Illustration

**Steps 1 – 6:**

Whenever the CC wants to transport an SSK to the MSC, it would proceed as in Protocol 3.1. Specifically, it would initiate the process by sending OTS₁ to the SC. Both the CC and MSC would recover their TEK₁/TAK₁ pair from their TMs. In this case, their TM will also provide KHK₁ associated with OTS₁, as illustrated in Figure 34. After the establishment of the secured channel under TEK₁/TAK₁, the CC uploads SSK₁ along with Nonce₁. Nonces are used to bring freshness to the integrity check process. The goal is to allow the same KHK to be re-utilized in the future with a different Nonce, which consequently allows for the minimization in the number of KHKs stored onboard.

![Figure 34: Hop-by-Hop Onboard Integrity Check, Steps 1–6](image)

**Steps 7 – 8:**

Upon the receipt of SSK₁ and Nonce₁, the MSC has all the parameters to compute a MAC over the SSK table onboard. The MAC computation is performed as follows: $\text{MAC}_1 = \text{MAC}_1^{\text{KHK}_1}(\text{Nonce}_1 | \text{SSK}_1)$.

The CC can utilize KHK₁ along with Nonce₁ to perform its own computation over SSK₁. Thus, once the MAC computed by the MSC is received, the CC checks whether its value matches the expected one.

Note that the MAC computation can be performed either in the concluding part of the protocol and/or at a later time. If the MAC is performed in the end of the protocol, MAC₁ could be returned to the CC immediately.

If performed at a later time, parameters KHK₁ and Nonce₁ are kept onboard until the CC requests such a computation to be executed. The CC can request this integrity check to be performed after a period of time after its upload to the MSC, and/or alternatively prior to the further distribution of SSKs to the destination SCs.

It is straightforward to support multiple integrity checks without switching the associated KHK (e.g. upon receipts of SSKs, and prior to their utilization), if required by the CC. For instance, the CC can upload multiple Nonces to the MSC which are all stored onboard. Next, each integrity check will consume a fresh Nonce along the current KHK.
Steps 9 – 13:

Prior to the distribution to the destination SC, the MSC computes a MAC over the SSK that is about to be transmitted, i.e. \( \text{MAC} = \text{MAC}_{\text{KHK}_1}(\text{Nonce}_1, \text{SSK}_1) \). Then, it compares it to \( \text{MAC}_1 \), which had been previously computed during the receipt of SSK. If both values match, then it is guaranteed that \( \text{SSK}_1 \) has not been corrupted while stored onboard the MSC. Otherwise, \( \text{SSK}_1 \) would have to be discarded. Note that this MAC computation cannot only be compromised by the corruption of \( \text{SSK}_1 \). If \( \text{KHK}_1, \text{Nonce}_1, \) and \( \text{MAC}_1 \) are corrupted for some reason, the integrity check will result in a different MAC. Therefore, these elements are strongly recommended to be protected with EDAC techniques. This would allow any error to be potentially corrected, or in the worst-case scenario determine which parameter has been corrupted.

In order to transfer \( \text{SSK}_1 \) to the SC, the MSC utilizes OTS\(_{11} \). The first step is to recover \( \text{TEK}_{11}/\text{TAK}_{11} \) and \( \text{KHK}_{11} \). Next, a different nonce is selected, e.g. \( \text{Nonce}_{11} \), and the pair \( \text{SSK}_1/\text{Nonce}_{11} \) is sent to the destination SC. At this point, the MSC computes a MAC utilizing \( \text{SSK}_1 \), \( \text{Nonce}_{11} \), and \( \text{KHK}_{11} \), and stores it onboard.

All nonces utilized by the MSC can either be generated onboard or transported by the CC. Generating nonces onboard would require a pseudo-random number generator (PRNG), but would not require the CC transport of any additional data. Generating the nonces on the ground brings the complexity to the CC, therefore avoiding the extra burden of having the MSC generating them. However, in this case, those nonces have to be transported to the MSC and be stored onboard for future use.
Step 14:

After receiving $SSK_1$ and $Nonce_1$, and storing these data onboard, the destination SC computes a MAC over $SSK_1$. This operation involves $KHK_{11}$ provided by its TPM and $Nonce_{11}$ which was just received. Upon the completion of the MAC computation, $MAC_{11}$ is generated and sent to the MSC. Based on the MAC value that the MSC had previously computed, it could certify that the $SSK$ stored onboard the destination SC has not been corrupted. Again, notice that it may be the case that the parameters involved in the computation ($KHK$ and $Nonce$) were the values corrupted, not the $SSK$ itself.
3.3.8 End-to-End Onboard Integrity Check

The Hop-by-Hop integrity check reveals a set of information to the intermediate node, in this case the MSC. In some scenarios, this may not be an issue such as in the case of having the same entity owning and operating MSC along with the constellation. However, in other scenarios the CC may want to avoid revealing any information to the intermediate nodes, as it was the case for key transport in Section 3.3.6. This section introduces a method, based on Protocol 3.2, to perform integrity check on the destination SC so that End-to-End secrecy is achieved.

3.3.8.1 Step-by-Step Protocol Illustration

The CC executes steps 1 to 6 in order to establish a secured channel with the MSC. Next, the CC utilizes OTS_{50} to recover TEK_{50}/TAK_{50} and KHK_{50}, and continues with Step 6 as follows.

**Step 6:**

A nonce (Nonce_{1}) is generated and a package containing SSK_1 and Nonce_{1} is created. This package is encrypted with TEK_{50}, its MAC is created utilizing TAK_{50}, and it is sent to the MSC along with OTS_{50}. Note that the communication channel between the CC and the MSC is encrypted and authenticated under TEK_{1}/TAK_{1}.

![Figure 38: End-to-End Onboard Integrity Check, Step 6](image)

**Step 7:**

The MSC receives the package, decrypts and checks its MAC, and stores its contents onboard. More precisely, at this point the MSC has access to OTS_{50} and (SSK_1, Nonce_{1})TEK_{50}. However, the MSC does not have OTS_{50}, as well as its associated keys, in its TPM. Therefore, it is not possible to read the contents of the encrypted package protected under TEK_{50}. 

94
Step 8 – 14:

The MSC establishes a secured channel under TEK_{11}/TAK_{11}, and transmits OTS_{50} along with \((SSK_1, \text{Nonce}_1)_{TEK_{50}}\) to the SC. The destination SC utilizes OTS_{50} to recover TEK_{50}, TAK_{50}, and KHK_{50} from its TPM. With this information, the SC can decrypt the received package and check its authentication tag.

Step 15 – 16:

After decrypting and checking authentication of the received package, the SC stores it onboard. Additionally, the SC can utilize KHK_{50} and \text{Nonce}_1 to perform an integrity check over the SSK stored on its memory. As previously explained in Section 3.3.7.1, this
operation can be either performed as a concluding part of the protocol, or at a later time. Whenever it is done, the SC returns the generated MAC tag (MAC₁ in this case) to the CC. This operation is encrypted and authenticated under TEK₅₀/TAK₅₀, which are only known by the SC and the CC. Hence, this operation can go through the MSC without disclosing any information. Alternatively, if the SC has the CC in line of sight, a direct communication session could be established without going through any intermediate entity.

Figure 41: End-to-End Onboard Integrity Check, Steps 15–16

### 3.3.9 Complexity Evaluation

Given the key transport scheme proposed in the previous section, it is fundamental to determine communication and onboard resources required to support key management. As previously mentioned in this document, an additional topic to be investigated in the future is the impact that the proposed approach would cause on operations workload. Actually, operations may also be a factor in the determination of cryptoperiods. This section provides an evaluation of the impact of different mission parameters on rekeying operations and on the trusted platform requirements. It defines all the parameters involved with key management and the requirements of a single SC. Based on those parameters, the requirements of the MSC is also determined.

#### 3.3.9.1 Mission and Communication Parameters

(L , days) **Mission Lifetime:** Is the number of years that the cryptographic mechanisms protecting the constellation is expected to remain active. For the sake of simplicity, it is assumed that all SCs become active at the same point in time, as well as that all SCs are retired simultaneously in the end of the mission lifetime.

(N) **Number of Spacecrafts in the Constellation:** Represents the maximum number of SCs expected in the constellation. For long mission lifetimes, it could be expected that some SC may be retired. If that is the case, it is assumed that they are replaced by another SC.

(S) **Number of Secured Channels per SC:** Even though each SC may carry a set of links (TC, TM HK, TM PL), it is assumed that each link may carry a set of internal (multiplexed)

96
channels. Thus, independently of the number and type of links, this parameter specifies the total number of channels for a single SC which must be secured.

\textbf{(SKc) Number of SKs Required per Secured Channel:} Each channel demands a set of cryptographic mechanisms, for example encryption and data origin authentication, the most common ones. This parameter determines the number of SKs required for each channel. For instance, if encryption and authentication are supported, two SKs are required \((SK_c = 2)\). It is assumed that all channels will have the same number of \(SK_c\).

\textbf{(C, days) Cryptoperiod:} This is the number of days that a cryptographic key will remain active. It is assumed that encryption and authentication keys will have the same cryptoperiod, and that both keys will be replaced at the same time.

\textbf{(BTC, bps) TC Link Bandwidth during Rekeying:} Specifies the bandwidth of the TC link available during rekeying operations, in bits per second (bps). Notice that this is the bandwidth consumed only during rekeying. At other times, the bandwidth is completely available for regular communications.

\textbf{(RTSC, minutes) SC Rekeying Time Window:} This is the time window, in minutes, in which the MSC has the SC is in line of sight during an ‘overpass’ to perform a rekeying operation. It is assumed that a rekeying operation is split into several rekeying sessions if this time window is not long enough to allow for the upload of all keying material to the destination SC. In other words, the rekeying time window is what determines the length of a single rekeying session, and ultimately impacts on the number of rekeying sessions within a single rekeying operation. The frequency that the SC is in line of sight only impacts on the total amount of time to finish the rekeying operation, but it does not interfere with the number of rekeying sessions. By the same token, the number of rekeying sessions is only dependent on the rekeying time window and on the TC bandwidth.

\textbf{(RTMSC, minutes) MSC Rekeying Time Window:} Similarly to \(RTSC\), \(RTMSC\) refers to the amount of time that the MSC is in line of sight of the CC during an ‘overpass’. Again, the \(RTMSC\), along with the \(BTC\), is utilized to compute the total number of rekeying sessions of the MSC.

\textbf{OTS Size (|OTS|):} Number of bits of a single OTS.

\textbf{TEK Size (|TEK|):} Number of bits of a single TEK.

\textbf{TAK Size (|TAK|):} Number of bits of a single TAK.

\textbf{KHK Size (|KHK|):} Number of bits of a single KHK.

\textbf{SK Size (|SK|):} Number of bits of a single SK.

\textbf{SSK Size (|SSK|):} Number of SKs within a single SSK.

\textbf{(SSKLSC) SSKs Launched Onboard SC:} Number of SSKs launched onboard of the SC.

\textbf{(SSKuSC) SSKs Sent to SC during Rekeying:} Number of SSKs that is uploaded to the SC in a single rekeying operation.

\textbf{(SSKLMSC) SSKs Launched Onboard MSC:} Number of SSKs launched onboard of the MSC.

\textbf{(SSKuMSC) SSKs Sent to MSC during Rekeying:} Number of SSKs that is uploaded to the MSC in a single rekeying operation.
3.3.9.2 Spacecraft Platform Requirements

Based upon the mission parameters listed in section 3.3.9.1 it is possible to compute the rekeying and platform requirements for an individual SC. This section determines how each requirement can be computed. The dot operator (\cdot) represents a multiplication operation in the equations presented below.

\((SK_{tSC})\) **Total Number of SKs for the SC Mission:** Total number of SKs that will be necessary to secure all channels during the entire mission lifetime of a single SC. It is a function of the mission lifetime \((L)\), number of secured channel per SC \((S)\), the number of SKs required per channel \((SKc)\), and the cryptoperiod of the keys \((C)\). It is defined as:

\[
SK_{tSC} = \left\lceil \frac{365 \cdot L \cdot S \cdot SKc}{C} \right\rceil. \tag{Equation 3.6}
\]

\((SSK_{tSC})\) **Total Number of SSKs for the SC Mission:** Total number of sets of SKs that will be demanded for the entire mission lifetime of a single SC. It is a function of the total number of keys needed \((SK_{tSC})\) and SK set size \(|SSK|\). It is defined as:

\[
SSK_{tSC} = \left\lfloor \frac{SK_{tSC}}{|SSK|} \right\rfloor. \tag{Equation 3.7}
\]

\((SK_{pSC})\) **Number of SKs to be Uploaded Post-Launch:** Total number of keys that will have to be uploaded to a single SC after launch. The upload of keys only becomes necessary after their onboard keys (loaded pre-launch) were consumed. In other words, if the SC is launched with enough keys onboard, post-launch upload may not become necessary. It is a function of the total number of keys needed for the mission \((SK_{tSC})\) and the number of keys launched onboard \((SK_{lSC} \cdot |SSK|)\). It is defined as:

\[
SK_{pSC} = \begin{cases} SK_{tSC} - (SSK_{lSC} \cdot |SSK|), & \text{if } SSK_{lSC} \cdot |SSK| < SSK_{tSC} \\ 0, & \text{otherwise} \end{cases} \tag{Equation 3.8}
\]

\((SSK_{pSC})\) **Number of SSKs to be Uploaded Post-Launch:** Total number of SSKs to be uploaded to a single SC after launch. Again, if \(SSK_{lSC} \geq SSK_{tSC}\), post-launch upload may not become necessary. The total number of SSKs is a function of the total number of SSKs required for the mission \((SSK_{tSC})\) and the number of sets launched onboard \((SSK_{lSC})\), and is defined as:

\[
SSK_{pSC} = \left\lfloor \frac{SK_{pSC}}{|SSK|} \right\rfloor. \tag{Equation 3.9}
\]

Or, alternatively as:

\[
SSK_{pSC} = SSK_{tSC} - SSK_{lSC}. \tag{Equation 3.10}
\]

\((Ro_{SC})\) **Number of Rekeying Operations:** Total number of rekeying operations required by a single SC after launch. It takes into account that the SC is launched with a number of SSKs onboard \((SSK_{lSC})\), as well as that a number of SSKs are uploaded in a rekeying operation \((SSK_{uSC})\). It is defined as:

\[
Ro_{SC} = \left\lfloor \frac{SSK_{pSC}}{SSK_{uSC}} \right\rfloor. \tag{Equation 3.11}
\]

98
**Number of Rekeying Sessions:** The total number of rekeying sessions required is a function of the TC bandwidth \((B_{TC})\), rekeying time window \((Rt_{SC})\), number of SSKs to be uploaded during rekeying \((SSKu_{SC})\), size of the SSK \(|SSK|\), and size of SKs \(|SK|\) within the SSKs. It is defined as:

\[
R_{SC} = \lceil \frac{(SSKu_{SC} \cdot |SK| \cdot |SSK|)}{(60 \cdot B_{TC} \cdot Rt_{SC})} \rceil.
\]  
(Equation 3.12)

**SK Utilization Rate:** The daily consumption rate of SKs of a single SC is a function of the number of secured channels \((S)\), SKs required per channel \((SKc)\), and the cryptoperiod \((C)\). It is defined as:

\[
SK_{RS} = \frac{(S \cdot SKc)}{C}.
\]  
(Equation 3.13)

**First Rekeying Operation:** Given a number of SSKs \((SSKl_{SC})\) launched with the SC, and a given daily key consumption \((SKr_{SC})\), it is possible to compute the number of days elapsed between launch and the first rekeying operation, as follows:

\[
Rf_{SC} = \frac{(SSKl_{SC} \cdot |SSK|)}{SKr_{SC}}.
\]  
(Equation 3.14)

**Subsequent Rekeying Operations:** Provided that each rekeying operation uploads a number of sets of SSKs \((SSKu_{SC})\) to the SC, the number of days for subsequent rekeying operations can be computed as:

\[
Rr_{SC} = \frac{(SSKu_{SC} \cdot |SSK|)}{SKr_{SC}}.
\]  
(Equation 3.15)

**Memory Required to Store all Secret Information:** A set of secret information (OTS, TEK, TAK, KHK) is necessary to support key transport for the entire mission lifetime of the SC. This set of secret information is uploaded to the SC prior to launch and will remain onboard of the SC until the end of its mission lifetime. Notice that each rekeying operation requires an OTS, a TEK, a TAK, and a KHK. The total amount of memory (bytes) required to store this secret information is a function of the size of the set of secret information needed by a SC during its lifetime. It can be computed as follows:

\[
M_{SS} = R_{OS} \cdot (|OTS| + |TEK| + |TAK| + |KHK|) / 8.
\]  
(Equation 3.16)

**Memory Required for Buffering SSKs:** The memory requirements (bytes) to store SSKs onboard is a function of the number of the SK size as well as the number of SKs within the SSK. It is assumed that the onboard memory of the SC is as large as the number of keys launched with the SC. The memory requirement (bytes) is determined as:

\[
M_{K} = (SSKL_{SC} \cdot |SK| \cdot |SSK|) / 8.
\]  
(Equation 3.17)

**Memory Required to Store all SSKs:** It is also possible to determine the total amount of storage (bytes) that would be required if the SC was launched with all secret information and SKs needed to support its entire mission lifetime. The total memory requirement (bytes) would be computed as:

\[
M_{T} = (SSKT_{SC} \cdot |SK| \cdot |SSK|) / 8.
\]  
(Equation 3.18)
3.3.9.3 Master Spacecraft Platform Requirements

Similar to section 3.3.9.2, it is possible to determine the MSC requirements to support the operation of the entire constellation of SCs. Most of equations for the MSC are identical for those specified for single SC, with the exception that the requirements for the MSC consider a constellation of N SCs.

\( (SKt_{MSC}) \) **Total Number of SKs Needed by the Constellation:** The number of SKs needed for the constellation takes into account that the SCs were launched with \( SSKl_{SC} \) keys onboard. In other words, \( SSKl_{MSC} \) refers to the number of keys that will have to be uploaded to the SCs after they utilize their original supplies of SKs. Thus, the total number of keys can be determined by:

\[
SKt_{MSC} = N \cdot SKp_{SC}.
\]  
(Equation 3.19)

\( (SSKt_{MSC}) \) **Total Number of SSKs Needed by the Constellation:** Likewise \( SSKl_{SC} \), \( SSKt_{MSC} \) is a function of the total of keys needed and the size of the SSK. It can be determined by:

\[
SSKt_{MSC} = SKt_{MSC} / |SSK|.
\]  
(Equation 3.20)

\( (SKp_{MSC}) \) **Number of SKs to be Uploaded Post-Launch:** Given that the MSC is launched with \( SSKl_{MSC} \) sets of keys onboard, the number of keys that need to be uploaded to the MSC post-launch is given by:

\[
SKp_{MSC} = SKt_{MSC} - (SSKl_{MSC} \cdot |SSK|).
\]  
(Equation 3.21)

\( (SSKp_{MSC}) \) **Number of SSKs to be Uploaded Post-Launch:** The total number of SSKs that will have to be uploaded post-launch is given by:

\[
SSKp_{MSC} = \lceil SKp_{MSC} / |SSK| \rceil.
\]  
(Equation 3.22)

\( (RKo_{MSC}) \) **Number of Rekeying Operations:** The number of rekeying operations of the MSC is given by:

\[
Ro_{MSC} = \lceil SSKp_{MSC} / SSKu_{MSC} \rceil.
\]  
(Equation 3.23)

\( (RKs_{MSC}) \) **Number of Rekeying Sessions:** Given a rekeying time window in which a CC may have the MSC in line of sight to perform rekeying, the number of rekeying sessions can be determined by:

\[
Rs_{MSC} = \lceil (SSKu_{MSC} \cdot |SK| \cdot |SSK|) / (60 \cdot B_{TC} \cdot R_{tMSC}) \rceil.
\]  
(Equation 3.24)

\( (SKr_{Const}, SKs \ per \ day) \) **SK Utilization Rate:** The SK utilization rate for the entire constellation is given by:

\[
SKr_{Const} = N \cdot SKr_{SC}.
\]  
(Equation 3.25)
**First Rekeying Operation:** In order to compute the amount of time until the first rekeying of the MSC takes place, it is necessary to consider the consumption rate of cryptographic keys of the entire constellation ($SK_{r_{const}}$). Moreover, the MSC and the SCs in the constellation have been launched with a set of keys onboard (respectively $SSK_{l_{MSC}}$ and $SSK_{l_{SC}}$). Hence, the MSC rekeying will only occur after all the keys onboard the SC and the MSC have been consumed. The time frame (days) between launch and the first rekeying of the MSC is given by:

$$R_{f_{SC}} = \left( |SSK| \cdot (SSK_{l_{MSC}} + (N \cdot SSK_{l_{SC}})) \right) / SK_{r_{const}}.$$  \hspace{1cm} (Equation 3.26)

**Subsequent Rekeying Operations:** Subsequent rekeying of the MSC will happen after all the sets of keys uploaded ($SSK_{u_{MSC}}$) are consumed. The number of days for subsequent rekeying is determined by:

$$R_{r_{MSC}} = \left( SSK_{u_{MSC}} \cdot |SSK| \right) / SK_{r_{const}}.$$  \hspace{1cm} (Equation 3.27)

**Memory Required to Store Secret Information:** The MSC stores all the secret info (OTS, TEK, TAK, KHK) to perform the rekeying of the entire constellation of SCs. It also stores all the secret information necessary to be rekeyed by the CC. This information is uploaded to the MSC prior to launch and will remain onboard until the end of the constellation lifetime. The total amount of memory (bytes) required to store this secret information is determined by:

$$M_{s_{MSC}} = \left( N \cdot R_{o_{MSC}} + R_{o_{MSC}} \right) \cdot (|OTS| + |TEK| + |TAK| + |KHK|) / 8.$$  \hspace{1cm} (Equation 3.28)

**Memory Required for Buffering SSKs:** The memory requirement (byte) to store SSKs onboard of the MSC depends on the maximum number of SSKs to be stored ($SSK_{l_{MSC}}$) as well as number of SKs within the SSKs ($|SSK|$).

$$M_{k_{MSC}} = (SSK_{l_{MSC}} \cdot |SK| \cdot |SSK|) / 8.$$  \hspace{1cm} (Equation 3.29)

**Memory Required to Store all SSKs:** It is possible to determine the amount of memory onboard of the MSC that would be required to perform the rekeying of the constellation during its entire mission lifetime. This includes all secret information necessary for key transport, as well as all the SSKs to be distributed to the destination SCs. Given that each SC is launched with a number of SSKs onboard, the SSKs onboard the MSC are only used for rekeying the SCs after they run out of their original supply of SSKs.

$$M_{t_{const}} = (SSK_{l_{MSC}} \cdot |SK| \cdot |SSK|) / 8.$$  \hspace{1cm} (Equation 3.30)

### 3.3.10 Study Cases

In order to provide an estimate of the onboard resources required to implement the TPM, some study cases were analyzed. A set of assumptions common to all study cases is listed below:

- Each rekeying operation utilizes a single OTS between two SCs;
• All rekeying operations are done through the MSC. No direct rekeying between the CC and the SCs are considered in the study cases;

• The estimates provided in this section focuses on the bulk of data to be transmitted which are the SSKs. Transportation and storage of SIDs, Nonces, and metadata represents a minor portion of data compared to the SSKs. More precisely, SID may be a few bytes long, nonces may potentially reach a few hundred bits, and metadata additional few bytes. Overall, the bulk of the key transport are SSKs which are thousand of bytes long. Thus, transport of SSKs are the main subject to be investigated. SIDs, Nonces and metadata are therefore not considered in the estimates provided in this section;

• Depending on the SC’s orbit, each rekeying opportunity would be limited to the time that the SC is in line of sight. The rekeying time window is what ultimately determines the number of rekeying sessions necessary to perform one rekeying operation. It is assumed that the SC rekeying time window is limited to 5 minutes ($R_{SC} = 5$). Similar situation is assumed for the MSC. It is assumed that the CC would have a 10 minute time-window to rekey the MSC ($R_{MSC} = 10$);

• Each SSK contains 4096 SKs ($|SSK| = 4096$);

• Each SC is launched with ten SSKs onboard ($SSK_{SC} = 10$). This memory space is considered to be the maximum memory space available onboard to store SSKs. However, a rekeying operation does not necessarily need to fill up the entire memory space;

• In each SC rekeying, five SSKs are uploaded onto the SC ($SSK_{uSC} = 5$);

• The MSC is launched with ten SSKs onboard ($SSK_{MSC} = 10$), which are utilized in rekeying operations;

• The MSC receives five SSKs in each rekeying operation ($SSK_{uMSC} = 5$).

• The datarate assumed for the TC link speed is 1kbps ($B_{TC} = 1$kbps). This speed reflects the worst-case scenario of a contingency situation, in which the channel datarate is reduced.

The cases presented next have a set of parameters that varies according to the study case therefore reflecting different mission scenarios. All these parameters are utilized in conjunction with the equations presented in Sections 3.3.9.2 and 3.3.9.3. As a result, a set of requirements to support rekeying operations can be determine for both the SC and MSC. Notice that all parameters presented in the following scenarios are subjective and chosen almost arbitrarily. They are utilized to explore different mission scenarios and the implications on the proposed key transport mechanism.

3.3.10.1 Scenario 1

This scenario reflects a small constellation consisted of twenty SCs ($N = 20$) and a mission lifetime of ten years ($L = 10$). The bit length of all keys and SSMs are 128 bits ($|OTS| = |TEK| = |TAK| = |KHK| = |SK| = 128$). The SK cryptoperiod considered in this scenario is seven days ($C = 7$). Each SC has ten channels to be protected ($S = 10$), and that
each channel requires one encryption and one authentication key \((SK_c = 2)\). Based upon these parameters, a summary of requirements for an individual SC is shown in Table 28.

### Table 28: SC Requirements for Scenario 1

<table>
<thead>
<tr>
<th>SC Mission Lifetime Rekeying Requirements</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Number of SKs for the SC Mission ((SK_tSC))</td>
<td>(10.4 \times 10^3) [keys]</td>
</tr>
<tr>
<td>Total Number of SSKs for the SC Mission ((SSK_tSC))</td>
<td>3 [sets]</td>
</tr>
<tr>
<td>Number of SKs to be Uploaded Post-Launch ((SK_pSC))</td>
<td>0 [keys]</td>
</tr>
<tr>
<td>Number of SSKs to be Uploaded Post-Launch ((SSK_pSC))</td>
<td>0 [sets]</td>
</tr>
<tr>
<td>Number of Rekeying Operations (1 SC) ((RoSC))</td>
<td>0 [ops]</td>
</tr>
<tr>
<td>Number of Rekeying Sessions (1 SC) ((R_sSC))</td>
<td>N/A [sessions]</td>
</tr>
<tr>
<td>SK Utilization Rate (1SC) ((SK_rSC))</td>
<td>2.86 [keys/day]</td>
</tr>
<tr>
<td>First Rekeying Operation ((R_fSC)) in</td>
<td>N/A [years]</td>
</tr>
<tr>
<td>Subsequent Rekeying Operations ((R_rSC)) every</td>
<td>N/A [years]</td>
</tr>
<tr>
<td>Memory Req. for Secret Info (1 SC) ((M_sSC))</td>
<td>0 [bytes]</td>
</tr>
<tr>
<td>Memory Req. for Buffering SSKs (1 SC) ((M_kSC))</td>
<td>640 [kbytes]</td>
</tr>
<tr>
<td>Memory Req. to Store all SSKs (1SC) ((M_tSC))</td>
<td>192 [kbytes]</td>
</tr>
</tbody>
</table>

Given an average utilization of 2.86 keys per day \((SK_rSC)\), a total of 10.4 thousand keys \((SK_pSC)\) will be necessary to secure the communications channels for the entire mission lifetime of one SC. However, in this case, the SC is launched with enough SSKs onboard so that rekeying operations \((RoSC)\) are, in principle, not necessary.

The memory requirements onboard each individual SC to all SSKs launched with the SC is 640kbytes. Out of this memory, only 192 kbytes are utilized to store the SKs actually needed to support the SC operation. Notice that this is an optimistic estimate. It does not take into account that keys may get corrupted onboard the SC, for instance, due to SEUs. On the other hand, this study provides a good estimate of the order of magnitude of the onboard storage, which is completely feasible using today’s technology.

### 3.3.10.2 Scenario 2

This scenario is similar to the first one \((N = 20, L = 10, S = 10)\), with the exception that the cryptoperiods are reduced to one day \((C = 1)\). The goal in this case is to show the MSC onboard requirements to support the rekeying of the constellation described in this context.

As can be noticed in Table 29, the SC is not launched with enough SKs to support the entire mission lifetime. Precisely, each SC requires about 104 thousand SKs \((SK_tSC = 73 \times 10^3)\). From another perspective, eight SSKs will have to be uploaded to the SC after launch \((SSK_pSC = 8)\), which will be performed in two rekeying operations \((RoSC = 2)\). As previously mentioned, each rekeying operation may be broken into separate rekeying sessions. The number of rekeying sessions depends on the TC link bandwidth, amount of information to be
transmitted, as well as on the rekeying time window. Specifically to this scenario, each rekeying operation will divided into nine rekeying sessions ($R_{SC} = 9$).

Table 29: SC Requirements for Scenario 2

<table>
<thead>
<tr>
<th>SC Mission Lifetime Rekeying Requirements</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Number of SKs for the SC Mission ($SK_{t\text{SC}}$)</td>
<td>$7 \times 10^3$ [keys]</td>
</tr>
<tr>
<td>Total Number of SSKs for the SC Mission ($SSK_{t\text{SC}}$)</td>
<td>18 [sets]</td>
</tr>
<tr>
<td>Number of SKs to be Uploaded Post-Launch ($SK_{p\text{SC}}$)</td>
<td>32040 [keys]</td>
</tr>
<tr>
<td>Number of SSKs to be Uploaded Post-Launch ($SSK_{p\text{SC}}$)</td>
<td>8 [sets]</td>
</tr>
<tr>
<td>Number of Rekeying Operations (1 SC) ($R_{o\text{SC}}$)</td>
<td>2 [ops]</td>
</tr>
<tr>
<td>Number of Rekeying Sessions (1 SC) ($R_{s\text{SC}}$)</td>
<td>9 [sessions]</td>
</tr>
<tr>
<td>SK Utilization Rate (1 SC) ($SK_{r\text{SC}}$)</td>
<td>20 [keys/day]</td>
</tr>
<tr>
<td>First Rekeying Operation ($R_{f\text{SC}}$) in</td>
<td>5.6 [years]</td>
</tr>
<tr>
<td>Subsequent Rekeying Operations ($R_{r\text{SC}}$) every</td>
<td>2.8 [years]</td>
</tr>
<tr>
<td>Memory Req. for Secret Info (1 SC) ($M_{SC}$)</td>
<td>128 [bytes]</td>
</tr>
<tr>
<td>Memory Req. for Buffering SSKs (1 SC) ($M_{k\text{SC}}$)</td>
<td>640 [kbytes]</td>
</tr>
<tr>
<td>Memory Req. to Store all SSKs (1SC) ($M_{t\text{SC}}$)</td>
<td>1.13 [Mbytes]</td>
</tr>
</tbody>
</table>

Notice, however, that the SC is launched with enough keys to operate for longer than five years ($R_{f\text{SC}} = 5.6$). Thus, after 5.6 years, a rekeying operation must be performed. Since each rekeying operation uploads a number ($SSK_{u\text{SC}}$) of SSKs to the SC, subsequent rekeying operations will happen every 2.8 years ($R_{f\text{SC}} = 2.8$). If the SC were to store all SKs onboard for its entire mission lifetime, a total of 1.13 Mbytes would be necessary. However, only 128 bytes would be required to store SSMs (OTS, TEK, TAK, KHK) necessary to support key management.

Since all rekeying operations are performed via the MSC, it is necessary to determine its onboard requirements to support such operations. Table 30 lists a set of parameters that comprises the MSC requirements for the entire mission lifetime of the constellation.
Table 30: MSC Requirements for Scenario 2

<table>
<thead>
<tr>
<th>MSC Lifetime Rekeying Requirements</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of SKs for the Constellation (SKt&lt;sub&gt;MSC&lt;/sub&gt;)</td>
<td>640.8x10&lt;sup&gt;3&lt;/sup&gt; [keys]</td>
</tr>
<tr>
<td>Number of SSKs for the Constellation (SSKt&lt;sub&gt;MSC&lt;/sub&gt;)</td>
<td>157 [sets]</td>
</tr>
<tr>
<td>Number of SKs to be Uploaded Post-Launch (SKp&lt;sub&gt;MSC&lt;/sub&gt;)</td>
<td>599840 [keys]</td>
</tr>
<tr>
<td>Number of SSKs to be Uploaded Post-Launch (SSKp&lt;sub&gt;MSC&lt;/sub&gt;)</td>
<td>147 [sets]</td>
</tr>
<tr>
<td>Number of Rekeying Operations (MSC) (Ro&lt;sub&gt;MSC&lt;/sub&gt;)</td>
<td>30 [ops]</td>
</tr>
<tr>
<td>Number of Rekeying Sessions (MSC) (Rs&lt;sub&gt;MSC&lt;/sub&gt;)</td>
<td>5 [sessions]</td>
</tr>
<tr>
<td>SK Utilization Rate (Constellation) (SKr&lt;sub&gt;Const&lt;/sub&gt;)</td>
<td>400 [keys/day]</td>
</tr>
<tr>
<td>First Rekeying Operation (Rf&lt;sub&gt;MSC&lt;/sub&gt;) in</td>
<td>5.9 [years]</td>
</tr>
<tr>
<td>Subsequent Rekeying Operations (Rr&lt;sub&gt;MSC&lt;/sub&gt;) every</td>
<td>51.2 [days]</td>
</tr>
<tr>
<td>Memory Req. for Secret Info (MSC) (Ms&lt;sub&gt;MSC&lt;/sub&gt;)</td>
<td>4.375 [kbytes]</td>
</tr>
<tr>
<td>Memory Req. for Buffering SSKs (MSC) (Mk&lt;sub&gt;MSC&lt;/sub&gt;)</td>
<td>640 [kbytes]</td>
</tr>
<tr>
<td>Memory Req. to Store all SSKs (Constellation) (Mt&lt;sub&gt;Const&lt;/sub&gt;)</td>
<td>9.81 [Mbytes]</td>
</tr>
</tbody>
</table>

Given that the constellation will demand about 641 thousand keys (SKt<sub>MSC</sub> = 640.8x10<sup>3</sup>) for the entire mission lifetime, a total of thirty rekeying operations (Ro<sub>MSC</sub> = 30) will be needed. Considering that each SC and the MSC are launched with a certain number of keys onboard, the first rekeying of the MSC will happen in approximately six years (Rf<sub>MSC</sub> = 5.9). Subsequent rekeying of the MSC will happen every 51 days (Rr<sub>MSC</sub> = 51.2). If rekeying of the MSC were to be avoided, and if the MSC were to store all SSKs onboard to support secure communications for the entire constellation lifetime, a total of 9.81 Mbytes would be necessary.

Note that a relatively small memory for nowadays standards would be enough to store all the SKs needed to support the SC operation. In spite of that, it is strongly recommended to support rekeying of the SCs and the MSC. The main reason is the radiation found in space and its effects on electronics, which may cause the corruption of secret and keying materials stored onboard. This becomes a more serious issue for longer mission/constellation lifetimes, and this scenario is explored next.

### 3.3.10.3 Scenario 3

This scenario extrapolates the previous ones by analyzing a large constellation with a long lifetime. Moreover, the SCs utilize longer secrets and keying materials with short cryptoperiods. The goal is to show the implementation feasibility of the proposed approach in face of a very demanding scenario. In this case a constellation of 200 SCs is planned to survive for up to 100 years. The cryptoperiod is reduced to a single day (C = 1), whereas the bit length of all secret and keying materials is now doubled, i.e. 256 bits. Additionally, it is assumed that each SC has 200 channels (S = 200), which would allow them to communicate with the CC and all SCs in the constellation. It may be hard to find a real mission that requires the aforementioned number of channels. However, again, these numbers where only utilized to exercise the parameters and determine the requirements of the key transport mechanism.
Given the long mission lifetime of the entire constellation, it is obvious that SCs will become nonoperational after a period of time. However, it is assumed that they will eventually be replaced by new SCs so that the resulting number of SCs in the constellation remains the same.

Table 31: SC Requirements for Scenario 3

<table>
<thead>
<tr>
<th>SC Mission Lifetime Rekeying Requirements</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Number of SKs for the SC Mission ($SK_{tSC}$)</td>
<td>$14.6 \times 10^6$ [keys]</td>
</tr>
<tr>
<td>Total Number of SSKs for the SC Mission ($SSK_{tSC}$)</td>
<td>3565 [sets]</td>
</tr>
<tr>
<td>Number of SKs to be Uploaded Post-Launch ($SK_{pSC}$)</td>
<td>14559040 [keys]</td>
</tr>
<tr>
<td>Number of SSKs to be Uploaded Post-Launch ($SSK_{pSC}$)</td>
<td>3555 [sets]</td>
</tr>
<tr>
<td>Number of Rekeying Operations (1 SC) ($R_{oSC}$)</td>
<td>711 [ops]</td>
</tr>
<tr>
<td>Number of Rekeying Sessions (1 SC) ($R_{sSC}$)</td>
<td>18 [sessions]</td>
</tr>
<tr>
<td>SK Utilization Rate (1SC) ($SK_{rSC}$)</td>
<td>400 [keys/day]</td>
</tr>
<tr>
<td>First Rekeying Operation ($R_{fSC}$) in</td>
<td>102.4 [days]</td>
</tr>
<tr>
<td>Subsequent Rekeying Operations ($R_{rSC}$) every</td>
<td>51.2 [days]</td>
</tr>
<tr>
<td>Memory Req. for Secret Info (1 SC) ($M_{sSC}$)</td>
<td>88.88 [kbytes]</td>
</tr>
<tr>
<td>Memory Req. for Buffering SSKs (1 SC) ($M_{kSC}$)</td>
<td>1.25 [Mbytes]</td>
</tr>
<tr>
<td>Memory Req. to Store all SSKs (1SC) ($M_{tSC}$)</td>
<td>445.63 [Mbytes]</td>
</tr>
</tbody>
</table>

It can be observed in Table 31 that almost 15 million of keys ($SK_{tSC} = 14.6 \times 10^6$) are required by the entire constellation to support secure communications during its entire lifetime. This implies in 711 rekeying operations to be performed for each SC. The first rekeying operation of the SC will happen in about 102 days after launch, with subsequent rekeying happening every 51 days. The total amount of memory required to store SSMs necessary for the entire mission lifetime would be less than 90 kbytes. If all SKs were to be launched with the SC, a total onboard storage of 446 Mbytes would be required.

Table 32 lists the MSC requirements to support the rekeying of the constellation. Notice that a total of $2.9 \times 10^9$ keys are needed for this scenario, which would require the MSC to be rekeyed 142177 times. After the MSC launch, the first rekeying would happen in about 103 days. Due to the amount of SSKs uploaded to the MSC (only 5 SSKs per rekeying in this case), subsequent rekeying would happen every 0.3 days, i.e. approximately three times a day. Note that a 1-day cryptoperiod is utilized, which causes a high consumption of keys ($80000$ keys/day).

Despite the first rekeying takes a bit longer to happen due to all SSKs launched onboard the SCs, subsequent rekeying has to be performed multiple times a day. This is an indication that this scenario requires more SSKs to be uploaded in each rekeying operation. As a consequence, several rekeying sessions may become necessary if the TC link does not have a high datarate. Furthermore, if the number of rekeying sessions is not completed within the time frame where the next rekeying operation is planned to start, the constellation may enter into starvation.
Concerning the TPM to support this scenario, it can be observed that the amount of onboard memory \((M_{MSC})\) required to store all secret information is only 35 Mbytes. If the MSC were to be launched with all SKs necessary \((M_{Const})\) to support the constellation for its entire lifetime, a total of 89 Gbytes would be required.

It can be observed that this scenario utilizes a huge number of crypto keys, reaching the order of \(10^9\). This study did not consider related key attacks as it would be assumed that all keys are generated on the ground and then uploaded to the SC. It is important to reemphasize this point, not only to this scenario, but for any context utilizing the results of this research. Related keys would give an advantage for an attacker to explore other attack scenarios. As a consequence, the consideration of related keys attacks could potentially lead to shorter cryptoperiods. Notice, however, that this is an issue in this research because of the way that key generation is assumed to be performed. But, on the other hand, if more advanced approaches are taken into consideration such as onboard key generation and/or derivation, related key attacks would become a concern.
3.4 Appendix: Probability of Undetected Errors for Reed-Solomon Codes

This appendix presents copy of a document in its entirety provided by the coding researchers in the Politecnico di Torino and Università Politecnica delle Marche, both in Italy. This document has been used in our research to compute the Probability of Undetected Errors ($P_{ue}$) for Reed-Solomon codes, as reported in Section 3.2.1.

**Calculus of the probability of undetected error ($P_{ue}$) for Reed-Solomon codes**

Authors:
- R. Garello, Politecnico di Torino, Italy
- F. Chiaraluce, M. Baldi, M. Bianchi, Università Politecnica delle Marche, Italy

Let us consider an ($n, k, q$) RS code. The data symbols and check symbols are elements from a finite field with $q$ symbols.

Let us denote by $(1 - \varepsilon)$ the probability that any transmitted symbol is received correctly at the receiver and by $\varepsilon/(q - 1)$ the probability it is transformed into any of the $q - 1$ other symbols.

Let us suppose that the code is used for pure error detection. An explicit expression for $P_{ue}(\varepsilon)$ is as follows [1]:

$$P_{ue}(\varepsilon) = \sum_{i=1}^{n} A_i \left(\frac{\varepsilon}{q-1}\right)(1-\varepsilon)^{n-i},$$

(1)

where $A_i$ is the number of codewords having exactly $i$ nonzero symbols. The knowledge of $A_i$’s corresponds to know the weight distribution of the code.

Eq. (1) generalizes the well-known expression valid for the binary case. It follows directly from the fact that an error pattern will go undetected by the decoder if and only if it is identical to one of the nonzero codewords. This certainly is the case if the all-zero codeword is transmitted, but also holds for the transmission of an arbitrary codeword due to the linearity assumption.

The weight distribution of Reed-Solomon codes has been completely determined [2]. In particular, for a $t$-error-correcting RS code of length $n = q - 1$ with symbols from GF($q$), the number of codewords of weight $i$ is given by:

---

\[ A_i = \left( q^{-1} \right)^i q^{-2t} \left( q-1 \right)^i + \sum_{j=0}^{2t} (-1)^{i+j} \binom{i}{j} \left( q^{2t} - q^j \right) \right] , \] (2)

for \( 2t + 1 \leq i \leq q - 1 \) and \( d = 2t + 1 \) is the minimum distance of the code.

Expression (2) can be also rewritten as [3]:

\[ A_i = \sum_{j=0}^{i-d} (-1)^i \binom{i}{j} \left( q^{i-j} - 1 \right) = \binom{n}{i} (q-1) \sum_{j=0}^{i-d} (-1)^j \binom{i-1}{j} q^{i-d-j} . \] (3)

Using the weight distribution, the probability of undetected error, for this case, results in [4]:

\[ P_{ue}(\varepsilon) = q^{-r} \left[ 1 + \sum_{i=0}^{r-1} \binom{n}{i} \left( q^r - q^i \right) \left( \frac{\varepsilon}{q-1} \right)^i \left( 1 - \frac{q\varepsilon}{q-1} \right)^{n-i} - q^r (1 - \varepsilon)^n \right] , \] (4)

where \( r = 2t \).

Like all BCH codes, however, RS codes can be used for simultaneous error correction and detection. Let us denote by \( \lambda \) a nonnegative integer less than \( d/2 \) (i.e., \( \leq t \)), and suppose that the RS code is used to correct all error patterns with \( \lambda \) or fewer symbol errors. In this case, (1) is no longer valid as, when the number of errors introduced by the channel is \( \geq d - \lambda \), the received sequence may fall within the decoding sphere of a codeword different from the transmitted one; when this happens, trying to correct, the decoder introduces \( \lambda \) additional symbol errors, and this event is undetected as well.

If \( d - \lambda \) or more errors occur, it is in general rather difficult to calculate, or even estimate, the probability of undetected error. For the case of RS codes, however, the problem has been effectively solved.

Let us denote by \( P_{ue}(\varepsilon, \lambda) \) the probability of undetected error after correction. It is possible to prove [4] that:

\[ P_{ue}(\varepsilon, \lambda) = \sum_{h=0}^{\lambda} \binom{q-1}{h} \left[ q^{-r} (q-1)^h - \varepsilon^h (1-\varepsilon)^{q-1-h} \right. \]
\[ + \left. \sum_{j=0}^{\min(r-1,q-1-h)} \left( q-1-h \right) \binom{\varepsilon}{q-1}^j \left( 1 - \frac{q\varepsilon}{q-1} \right)^{q-1-h-j} R_{h,j}(\varepsilon) \right] , \] (5)

where
\[ R_{h,j}(\epsilon) = \min_{r \leq j < h} \sum_{l=0}^{h-l} (-1)^{h-l} \left(\begin{array}{c} h \\ l \end{array}\right) \left(1-q^{-r+j+l}\right)^l \left(1-q^e \right)^{h-l}, \]  

(6)

for \(0 \leq j < r\).

As expected, (5) reduces to (4) by setting \(\lambda = 0\), which implies pure error detection. On the opposite side, if the RS code is used at the most of its correction capability, (5) must be applied by setting \(\lambda = t\).

**Remark 1**: According to the discussion above, the problem of determining the probability of undetected error seems to be solved since (at least) 1984 (i.e., the Kasami-Lin’s paper). However, as expression (5), in particular, is rather hard to compute, further work has been done for finding reliable upper bounds to \(P_{ue}(\epsilon, \lambda)\). A first contribution, in this sense, appeared just in the Kasami-Lin’s paper, while an improved bound was presented in [5] and [6]. The main targets of these papers were: i) (as mentioned) to simplify the computation, though at the expense of slightly worse precision; ii) to demonstrate that a linear MDS code, which possesses rigid algebraic and combinatoric structures, behaves (in some sense) like a random code with no structure at all. Property ii), in particular, gives the rationale to explain the validity of the simplified bounds. Though conceptually interesting, these alternative approaches seem not essential, in the sense that computation of (5) should be possible for RS codes of practical interest (e.g., the (255, 223) RS code included in the CCSDS “TM Synchronization and Channel Coding Standard”). For this reason, they have not been discussed in detail in this note.

**Remark 2**: Eqs. (4) and (5) express the probability of undetected error as a function of the symbol error rate. If considered of more interest, it is simple to express the same results as a function of the probability of a bit error at the input of the RS decoder. In particular, noting the latter as \(p\) and assuming \(q = 2^m\), we have \(1 - \epsilon = (1-p)^m\).

**Appendix References**


Chapter 4 : Reconfigurable and Distributed Key Management
4.1 Scope

This chapter is divided into four main sections, which complements and expands the research topics covered in Phase 2 of this project. The approach for key transport that had been previously proposed was based on the assumption that the CC is responsible for generating the keys on the ground, which are later transported to the destination SCs with the support of a MSC. In addition to that approach, some additional strategies tailored to space systems were investigated.

Section 4.2 presents several optimizations to the key transport protocols introduced in Phase 2. A reduced number of messages was achieved with the optimizations, as well as the capability of timing-out messages therefore allowing for higher resistance against replay attacks.

In addition to key transport, key agreement schemes were analyzed. In Section 4.3, a password-based Diffie-Hellman method was modified so that it could be utilized in conjunction with the onboard trusted platform. The motivation behind this extension is to allow the TPM to provide the passwords to the SCs so they may be utilized within the Diffie-Hellman protocol.

Section 4.4 describes a method to securely upload new sets of secret materials (SSMs) to the SC’s TPM after launch. This operation is important to guarantee that the SC will always have enough SSMs onboard, even in the presence of harsher scenarios. For instance, there may be an unpredictable reduction of cryptoperiods, due to leakage of SSMs caused by some (internal) attack.

Redundancy of in-orbit key distribution centers are introduced in Section 4.5. Two scenarios are taken into account: the first one utilizes two MSCs and the second one considers multiple MSCs. The two-MSC method is very similar to the single-MSC approach, except for the fact that the latter splits the onboard table to avoid collisions with its mirror MSC. Synchronization is also performed when those two MSCs are in line of sight. Due to the complexity of keeping the multiple onboard tables synchronized, the multiple-MSC method employs a different approach, which is based upon onboard key derivation.
4.2 Key Transport

This section presents some updates on the techniques introduced in phase 2 (Chapter 3). Improvements to the original approaches comprise the reduction in the number of messages in the protocol. This goal is achieved through the utilization of nonces and timestamps.

4.2.1 Original Key Transport Approach

During phase 2 of this research project, a protocol has been proposed to transport SSKs between the CC and an MSC (summarized in Protocol 4.1). In subsequent months, further research has been done and some alternatives have been developed.

**Protocol 4.1: Summary of the Key Transport Proposed in Chapter 3**

<table>
<thead>
<tr>
<th>CC</th>
<th>MSC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selects <em>OTS</em></td>
<td>Initialization phase.</td>
</tr>
<tr>
<td>Initiates protocol</td>
<td>Retrieves <em>TEK</em>, <em>TAK</em>, and <em>KHK</em></td>
</tr>
<tr>
<td><em>OTS</em></td>
<td></td>
</tr>
<tr>
<td>Checks <em>OTS-1</em></td>
<td>Key confirmation</td>
</tr>
<tr>
<td><em>E(OTS-1)TEK</em>, <em>MAC( E(OTS-1)TEK)TAK</em></td>
<td></td>
</tr>
<tr>
<td>Uploads <em>SSK</em></td>
<td>Recovers <em>SSK</em> and stores it onboard</td>
</tr>
<tr>
<td><em>E(SSK)TEK</em>, <em>MAC( E(SSK)TEK)TAK</em></td>
<td>If required, perform integrity check utilizing <em>KHK</em></td>
</tr>
</tbody>
</table>

The main advantage of Protocol 4.1 is that its first message is very simple (no encryption/authentication) and short (basically only *OTS* is sent). It can be received quickly and no cryptographic computations are involved. After receiving *OTS*, the MSC could take an entire orbit to respond to the CC. This time could be utilized to perform the onboard computation of the key confirmation package and the message authentication code. This approach would be more suitable to scenarios where the communication window with the MSC is very limited. For instance, when the MSC is tumbling and its antenna is not pointed towards Earth for a long period of time. In general, the messages of this protocol are shorter than Protocols 4.2 and 4.3 (described below).

The main drawback of this protocol is that it is a 3-pass protocol, and a response from the MSC is required before continuing with the SSK upload. Furthermore, it can be noticed that the bulk of the transmission happens in third message due to the transport of SSKs. But, at that point in time, key confirmation has been done and the CC can split the third message...
into smaller ones. Next, TEK/TAK could be utilized to encrypt/authenticate parts of the third message. Each of these sub-messages would be transmitted in different overpasses. This way, its size can be adjusted to fit into the time-window available for communications.

### 4.2.2 Improved Key Transport based on Nonces

If more bandwidth is available in the communication channel, Protocol 4.1 can be modified with the goal of reducing the number of passes. Protocol 4.2 shows an improvement based on nonces using a 2-pass protocol. The first message carries OTS and nonce $n$ in the clear, plus an encrypted SSK and an authentication tag for the whole message. At this point, the encrypted part of the message cannot be read by the MSC, because it still does not have the proper TEK. The same happens to the authentication tag, which cannot be checked right away.

From an attacker’s perspective, he/she would receive the same data. The attacker will know OTS and the nonce, but would not be able to read the remaining message. The MSC, however, can use the OTS to access the TPM and acquire TEK, TAK and KHK. Next, it can decrypt the message and check its authentication tag. An attacker, on the other hand, would not be able to perform similar actions, and the best that he/she can do is to guess the encryption keys and generate the authentication tag, or try to forge the message. The MSC can also store the SSKs onboard and perform a keyed hash over the memory utilizing KHK.

The second message of the protocol carries the key confirmation, the integrity check tag of the SSKs stored onboard, and a message authentication code computed over the entire message. After receiving this message the CC can confirm that the MSC recovered the proper TEK. If the keyed hash tag does not match, two things may have happened: 1) the SSKs is corrupted, 2) the KHK has not been recovered correctly, or 3) the second message has not been transmitted correctly (which will also be noticed while checking the authentication tag); This may be due to channel errors or attackers trying to compromise the communications.

Last, the CC can also certify that the MSC recovered the proper TAK by computing the message authentication tag of the message received. If the message authentication code does not match, it may be the case that: 1) the TAK was not recovered correctly, 2) the second message has not been transmitted correctly.
### Protocol 4.2: Improved Key Transport Protocol Based on Nonces

**CC**
- Selects OTS and n
- Initiates protocol
- Uploads SSK

**MSC**
- Uploads SSK, $E( SSK \}_TEK$, $MAC( OTS, n, E( SSK )_TEK )_TAK$

**Initialization phase.**
- Retrieves $TEK$, $TAK$, and $KHK$
- Decrypts $E()_TEK$
- Checks MAC

- $E( OTS-1, n )_TEK$
- $H( n, SSK )_KHK$
- $MAC( E( OTS-1, n )_TEK, H( n, SSK )_KHK )_TAK$

**Key confirmation**
- Integrity check
- Authentication

**Checks OTS-1**
**Checks n**
**Checks MAC**
**Checks $H$ of stored SSKs against expected value**

The main advantage of this approach is that only 2 passes are required to perform key transport. On the other hand, in order to avoid replay attacks, the MSC has to keep track of all nonces utilized. It is also susceptible to jamming (likewise Protocol 4.1), where an attacker completely disrupts the first message. As a consequence, the MSC will not receive it, but the attacker might have listened to its contents. If that is the case, the attacker will be able to replay the same message at a later time. This happens because of the nonce used, which would seem fresh from the MSC’s perspective, but in reality it had been used already. In the end, what the attacker can accomplish is exactly the same as the CC initially intended. However, the attacker would have the ability to perform such an operation on his/her own will, and this may not be appropriate in some scenarios. It can be observed that this attack is also feasible in Protocol 4.1. Mechanisms to allow for a time-out would be very beneficial to avoid this kind of attack.
4.2.3 Improved Key Transport based on Timestamps

A method that utilizes timestamps, instead of nonces, has also been devised. Protocol 4.3 shows this alternative method, which is an improvement of Protocol 4.2. The first message has exactly the same structure as in Protocol 4.2, but it employs time stamp, \( t_1 \), in lieu of nonce, \( n \). The goal is to allow for the MSC to detect that the message is outdated by checking its timestamp.

An attacker could potentially try to send its own OTS and timestamp. However, in order to succeed, he/she would also have to send a valid message authentication tag. Since TAK is unknown to the attacker, the only way would be either to exhaustively search for the key or to forge the authentication tag.

Jamming can always be utilized by the attacker to disrupt the communication between the CC and the SC. It can be also assumed that he/she can do it to acquire the first message aiming at performing a replay attack. Still, the attacker would have to use this message within a short time-frame, while it is valid. Otherwise, it will expire and will be declined by the MSC. As a consequence, the attacker can only delay the action of the CC for a certain amount of time, which is as short as the time-out period.

The second message performs key confirmation and can also provide the CC with the guarantee that the SSKs stored onboard has not been corrupted/compromised. Both key confirmation and integrity tag utilize \( t_1 \) and \( t_2 \) to allow for time-out to be implemented. In addition, a message authentication tag is also sent to the CC, therefore allowing the CC to confirm the origin of such a message.

Similar to previous cases, this second message also protects against impersonation (attacker trying to act on behalf of the MSC). First, as the attacker does not know TEK, TAK and KHK, he/she would not be able to (feasibly) forge the parts of this message. The attacker could, again, disrupt the communications with jamming, but \( t_2 \) would impose a time-out period. Hence, the most that an attacker could accomplish with a replay attack would be to delay the action of the MSC by a period of time as long as the time-out period.

Given the utilization of timestamps, it is obvious that the MSC and the CC must keep some synchronization between their timestamp generators. Actually, this is the main drawback of this method.
Protocol 4.3: Improved Key Transport Protocol Based on Timestamps

CC

Selects OTS
Acquires $t_1$
Initiates protocol
Uploads SSK

MSC

Initializes phase.
Retrieves $TEK$, $TAK$, and $KHK$
Decrypts $E(TEK)$
Checks MAC and $t_1$
Acquires $t_2$

Key confirmation
Integrity check
Authentication

$OTS, t_1, E(SSK)_{TEK}, MAC(OTS, t_1, E(SSK)_{TEK})_{TAK}$

Deciphers $E(TEK)$
Checks $OTS-1$
Checks $t_1$ and $t_2$
Checks MAC
Checks whether $H$ of stored $SSK$s matches expected value

$E(OTS-1, t_1, t_2)_{TEK}, H(t_1, t_2, SSK)_{KHK}, MAC(E(OTS-1, t_1, t_2)_{TEK}, H(t_1, t_2, SSK)_{KHK})_{TAK}$
4.3 Key Agreement

This section presents some strategies to perform key agreement between the CC and a SC, or between two SCs. This is a valuable alternative to key transport since keys can be established between 2 parties without the participation of the CC and this allows for key derivation thus not having to rely on large sets of stored keys. A password-based Diffie–Hellman algorithm is utilized along with a TPM to allow for efficient key agreement protocols tailored to the space segment.

4.3.1 Diffie-Hellman

In addition to key transport mechanisms suggested in Section 4.2, a key agreement method is proposed in this section. It utilizes a trusted platform module (TPM) along with a variation of the PAK Diffie-Hellman (D-H) [RFC5683] to achieve authenticated key agreement between two entities. Key agreement provides some advantages when compared with key transport approaches previously investigated.

First, it avoids transferring or storing sets of session keys (SSKs), therefore saving bandwidth and memory. Second, from the SC perspective, it allows for increased freedom as key agreement can be done directly between two SCs, without the participation of the MSC or CC. The key agreement establishes a shared secret key between two SCs, referred to as a session key, which are used in encryption and authentication mechanisms. This means that in order to obtain both keys, for example, the protocol must be executed twice, which could have two instances running in parallel. Alternatively, the agreed key can serve as a seed for a key derivation mechanism on board, which, in turn, is used to generate other keys (e.g., for encryption, authentication, keyed hash).

The keys established in the proposed method are analogous to those transported in the previous methods. The main difference in this alternative approach is that keys are agreed on demand, not transported ahead of time. In other words, the protocol presented in this section will be executed only when the previous key lifetimes have expired and new keys need to be utilized. Some missions may allow for higher levels of SC autonomy, therefore permitting them to perform key agreement autonomously, with limited reporting to the CC/MSCs or none at all. In other scenarios, the CC may wish to have a tighter control of what is happening in the space segment, so that reporting to the CC/MSCs is mandatory. In order to support the second scenario, a set of supplementary reporting steps have been provided in Protocol 4.9.

4.3.2 Man-In-The-Middle Attacks

The utilization of D-H [1976DH] requires some precaution. It is well known that a classical man-in-the-middle attack (MITM) can be easily performed against the D-H algorithm. The original D-H algorithm is shown in Protocol 4.1, in which parameters $g$ and $p$ are known by the communicating parties. The parameters $R_a$ and $R_b$ are random numbers.

In order to perform a MITM attack, an adversary has to be placed between two entities (consider them as CC and SC, as in Protocol 4.4) and disrupt the communication between them. In addition, the adversary has to create new messages and execute a separate D-H protocol with each entity to be able to impersonate as the CC to the SC, and vice-versa.
Protocol 4.4: Diffie-Hellman Protocol

CC

Selects $R_a$;
Computes
$u = g^{R_a} \pmod{p}$

SC

Selects $R_b$;
Computes
$v = g^{R_b} \pmod{p}$

Computes the key
$K = u^{R_b} = (g^{R_a})^{R_b} \pmod{p}$

Computes the key
$K = v^{R_a} = (g^{R_b})^{R_a} \pmod{p}$

Assuming that communications occur between a CC and a SC (or two SCs in space), it is questionable how feasible a MITM attack would be. The reason being is that, for an adversary to succeed, he/she would have to receive the radio frequency (RF) signals from the transmitting entity (e.g. CC), modify them on-the-fly, and retransmit the superposed signal to the destination (e.g. SC). The resulting (superposed) signal would have to correspond to a (new) D-H protocol, now being performed between the adversary and the SC. This does not appear to be trivial. In fact, it would require some computation to be performed to analyze the received signal and to generate the new signal to be transmitted. Thus, by the time that the adversary sends out its modified signal, the original one would have reached the destination SC. There is no means to take advantage from previous run of the protocol, as each individual instance utilizes a fresh $R_a$ and $R_b$. It is also infeasible for the adversary to guess $R_a$ and $R_b$ currently in use.

The attack scenario becomes different, however, if an attacker is considered to have jamming capabilities. More specifically, in this context the adversary would have to be able to receive the incoming signal and, at the same time, jam it so that the destination is not capable of receiving it. If these attack requirements are satisfied, the MITM attack becomes easier to perform, as shown in Protocol 4.5. For instance, the transmitting entity would send its signal, which is received by the adversary. Simultaneously, the adversary jams this signal so that the destination does not receive it. At a later time, the attacker would send its own RF signal to the destination SC. The same procedure is valid on the other direction of the key agreement protocol. In the end, this adversary will have performed a MITM attack. Notice that the adversary does not necessarily have to be in space to perform this kind of attack.
Protocol 4.5: Man-in-the-Middle Attack Based on Jamming

CC

Selects \( R_a \);
Computes \( u = g^{R_a} \pmod{p} \);

Adversary

Selects \( R_c \);
Computes \( x = g^{R_c} \pmod{p} \);

SC

Selects \( R_b \);
Computes \( v = g^{R_b} \pmod{p} \);

Selects \( R_y \);
Computes \( y = g^{R_y} \pmod{p} \);

Selects \( R_b \);
Computes \( \text{key } (SC) K_{yb} = y^{R_b} = (g^{R_y})^{R_b} \pmod{p} \);

Selects \( R_c \);
Computes \( \text{key } (CC) K_{ya} = y^{R_c} = (g^{R_y})^{R_c} \pmod{p} \);

In this case, the detection of jamming would help to discover that an attacker might be tampering with the channel and the operation could be aborted. In addition to that, there are some cryptographic alternatives to avoid the MITM attacks against D-H, in which several of those utilize asymmetric primitives (e.g. digital signatures).

4.3.3 Related Work

Password authenticated key exchange (PAKE) protocols establish a key and provides authentication based upon knowledge of a shared secret password. Passwords are not transmitted, but users are assumed to know the password(s). Security properties of PAKE schemes are 1) resists off line dictionary attack (must not reveal password or hash of password), 2) forward secrecy (secure even if password revealed at a later date), 3) known session security (prevents a disclosed session from affecting other session security), and 4) resists on line dictionary attack (attacker can only test one password per session). Passwords are also used in other areas such as password-based encryption [RFC5683], where a salt and/or iteration count are concatenated with the password and input to a key derivation function), etc. Unlike these techniques, PAKE provides good security, specifically a high entropy cryptographic key, even when a low entropy password is used [2008Hao].

The PAKE schemes fall into two categories, that of balanced and augmented. Augmented protocols are more complex and typically involve server-client scenarios, yet have not been shown to be provably better than balanced types (apart from having a server compromise resistance). Balanced types of PAKE create a shared symmetric key. Most
balanced PAKE schemes which resist known attacks have been patented [2008Hao]), including EKE, SPEKE (simple password exponential key exchange, used in Blackberrys, Entrust, etc… [2009Vacca]), SRP, etc.

One scheme, which is not patented, has been standardized by the Internet Engineering Task Force (IETF) (see [RFC5683]). This scheme is password-authenticated key Diffie-Hellman exchange, or PAK. This balanced PAKE scheme, is as secure as Diffie-Hellman even when a low entropy password is used. If the password is known to the attacker, he/she will be able to compute the session key, thus the password must be kept secret.

Since PAKE assumes users already are in possession of passwords, there is some research [2009Paterson] addressing the generation of one-time passwords using hash chains [2005Abdalla]. Given a set of passwords sending the index to select a password is also suggested, in addition to alternatively selecting a password in response to a challenge. For example a session ID may include the index of the one-time password.

### 4.3.4 Password Authenticated Key Exchange

Simpler approaches which rely on shared secrets include the Simple Password Exponential Key Exchange (SPEKE) and Password Authenticated Key Exchange (PAK), which are respectively described in [RFC5683] and [RFC5683]. It appears that SPEKE and some of its derivations are protected by patents, thus PAK was utilized as a basis for the proposed approach.

Both SPEKE and PAK require a secret password to be utilized in the protocol. Therefore, some means are needed to store and retrieve this secret information securely. This is where a TPM is employed along with one-time secrets (OTSs). A modification to the PAK algorithm, which will be referred to as Protocol 4.6, is proposed and includes an OTS in the first message. Each OTS is used to retrieve a password, which is securely stored within the TPM. This password is subsequently utilized in the protocol.

### 4.3.5 TPM-Supported Key Agreement Protocol

By relying on a single OTS (transferred in the clear), a pair of entities can perform multiple runs of the key agreement protocol. The parameters $R_a$ and $R_b$ are random numbers generated on board each SC, and must be at least 384 bits long [RFC5683]. These are needed for each key agreement or each subsequent run of the protocol when key lifetimes have expired. The main drawback of the key agreement approach, compared with key transport mechanisms, is the utilization of more complex mathematical computations such as modular multiplication, inversion, and exponentiation. Depending on how it is implemented, it might potentially leave the CC without the knowledge of the keys agreed in orbit. A reporting mechanisms has also been devised, and will be later detailed (Protocol 4.9).
Protocol 4.6: Alternative Key Agreement Protocol – First Run

SC1

Selects $OTS$;
Retrieves associated $P$;
Selects $R_a$;
Computes $X = H_1(A, B, P)^g_{Ra}$

Computes
$Y_{ba} = Y^* (H_2(A, B, P))^*$
$S_1' = H_3(A, B, P, g_{Ra}, Y_{ba}, Y_{ba}^{Ra})$
If $S_1 = S_1'$ then
$K = H_5(A, B, P, g_{Ra}, Y_{ba}, Y_{ba}^{Ra})$
$S_2 = H_4(A, B, P, g_{Ra}, Y_{ba}^{Ra})$

SC2

Uses $OTS$ to retrieve $P$;
Computes $X_{ab} = X^* H_1(A, B, P)^*$
Selects $R_b$;
Computes
$Y = H_2(A, B, P)^g_{Rb}$
and
$S_1 = H_3(A, B, P, X_{ab}, g_{Rb}, X_{ab}^{Rb})$

$Y, S_1$

$OTS, A, X$

If $S_2 = S_2'$ then
$K = H_5(A, B, P, g_{Rb}, X_{ab}, X_{ab}^{Rb})$

Notice that the first protocol run (Protocol 4.6) requires the SCs to use an $OTS$ to obtain a password, $P$. The parameters $A$, $B$ represent identities of SC1 and SC2 respectively. The five hash functions are $H_{1:5}$ and are defined in [RFC5683]. Subsequent key agreements can use the same $P$ by selecting fresh values for $R_a$ and $R_b$. Therefore, $OTS$ will be sent as part of the first message only in the initial run of the protocol. Subsequent runs will have the SCs assume the same $P$, for as long as its cryptoperiod. A given SC would know when a new $P$ has to be acquired by checking if $OTS$ is part of the first message. Subsequent runs of the protocol are identical to the original PAK, as can be noticed in Protocol 4.7.
Protocol 4.7: Alternative Key Agreement Protocol – Subsequent Runs

SC1

Keeps same $P$; Selects $R_{a}$; Computes $X = H_{1}(A, B, P)^{g_{Ra}}$

$A, X$

Assumes $P$ to be the same; Computes $X_{ab} = X^{*} H_{1}(A, B, P)^{y_{l}}$
Selects $R_{b}$; Computes $Y = H_{2}(A, B, P)^{y_{l}} g_{Rb}$ and $S_{1} = H_{3}(A, B, P, X_{ab}, g_{Rb}, X_{ab})$

$Y, S_{1}$

Computes $Y_{ba} = Y^{*} (H_{2}(A, B, P)^{y_{l}})$ $S_{1}' = H_{3}(A, B, P, g_{Ra}, Y_{ba}, Y_{ba})$ If $S_{1} = S_{1}'$ then $K = H_{4}(A, B, P, g_{Ra}, Y_{ba}, Y_{ba})$ $S_{2} = H_{4}(A, B, P, g_{Ra}, Y_{ba}, Y_{ba})$

$S_{2}$

$S_{2}' = H_{4}(A, B, P, X_{ab}, g_{Rb}, X_{ab})$ If $S_{2} = S_{2}'$ then $K = H_{4}(A, B, P, g_{Rb}, X_{ab}, X_{ab})$

SC2

Another alternative utilizing timestamps has been derived with the goal of allowing message time-out. This will be discussed in Section 4.3.6.

4.3.6 TPM-Supported Key Agreement Protocol with Timestamps

Message time-out is an important feature to be considered if it is assumed that an attacker could jam the communication between two SCs and at the same time obtain the message being transmitted. In this case, the attacker would be able to replay the message at a later time, and the destination SC would not realize that it is outdated. In the key agreement protocols, as well as in the key transport protocols previously proposed, this is not a major concern. The reason being is that, even if an attacker delays the execution of the protocol, he/she would end up doing exactly the same action as the original entity intended to. However, this has been a point of criticism from some reviewers of our previous papers which is worth consideration in this work as well. The improved protocol is shown in Protocol 4.8 utilizing timestamps $t_{1-3}$.

The TPM utilized in the proposed approach is simpler than the one utilized in the key transport one. In the key transport approach, the number of transport operations would determine the number of OTSs and associated secret information (e.g. TEK, TAK, KHK,
SSKs). Note that each SSK may have thousands of secret keys (SKs). On the other hand, the key agreement dramatically reduces the storage requirements. The only information needed to be stored within the TPM is the pairs of OTS and P.

Protocol 4.8: Alternative Key Agreement Protocol With Timestamps

SC1

Selects OTS;
Retrieves associated P;
Selects $R_a$;
Acquires $t_1$;
Computes $X = H_1(t_1, A, B, P)^* g^{R_a}$

$t_1$, OTS, A, X

If $t_1$ >> or << current time abort.
Else Uses OTS to retrieve P;
Computes $X_{ab} = X^* H_1(t_1, A, B, P)^{t_1}$
Selects $R_b$;
Acquires $t_2$;
Computes $Y = H_2(t_1, t_2, A, B, P)^* g^{R_b}$
and $S_1 = H_3(t_1, t_2, A, B, P, X_{ab}, g^{R_b}, X_{ab}^{R_b})$

$t_2$, Y, S1

If $t_2$ >> or << current time abort
Else Computes $Y_{ba} = Y^* (H_2(t_1, t_2, A, B, P)^{t_1})$
$S_1' = H_3(t_1, t_2, A, B, P, g^{R_a}, Y_{ba}, Y_{ab}^{R_a})$
If $S_1 = S_1'$ then
Acquires $t_3$;
$K = H_5(t_1, t_2, t_3, A, B, P, g^{R_a}, Y_{ba}^{R_a}, Y_{ab}^{R_a})$
$S_2 = H_4(t_1, t_2, t_3, A, B, P, g^{R_a}, Y_{ba}^{R_a}, Y_{ab}^{R_a})$

$t_3$, S2

If $t_3$ >> or << current time abort.
Else $S_2' = H_4(t_1, t_2, t_3, A, B, P, X_{ab}, g^{R_b}, X_{ab}^{R_b})$
If $S_2 = S_2'$ then
$K = H_5(t_1, t_2, t_3, A, B, P, g^{R_b}, X_{ab}, X_{ab}^{R_b})$

SC2
4.3.7 TPM-Supported Key Agreement Protocol with Control Center Reporting

Assuming that two SCs can now perform autonomous key agreement, the CC will not have access to the keys utilized in the communication channel between two SCs. In some missions the CC would have an interest in knowing as much information about the system as possible. A potential solution to this problem is to have the SCs report the keys agreed in orbit to the CC at a later time. Again, the TPM can provide a means to achieve that. The idea is to have OTSs not only linked to a P, but also linked with a TEK/TAK pair.

The TEK/TAK pair is exclusively used to establish a secured communication channel with the CC. In other words, when P is issued, so is a TEK/TAK pair. An important observation is that this TEK/TAK pair is used exclusively for reporting information about key agreement, and it is not utilized to secure regular traffic. Regular data traffic between the CC and the SC utilizes a separate channel which is secured by its own session keys. The keys utilized to secure data communication between the SC and the CC can also utilize the aforementioned method for key agreement. In other words, the channel used to transport data can use multiple SKs while keeping the same P. Again, this is done independently if the communication is occurring between two SCs or between an SC and the CC. In either case, there is always a separate channel, secured by its own keys (TEK/TAK) for reporting data to the CC.

Reporting to the CC would be performed as follows. The TPM onboard of both SCs would obtain the TEK/TAK associated with a given OTS. The first SC would report R_a, K, and the timestamps. The second SC performs similar operation, i.e., would report the timestamps, R_b and K. Notice, however, that a given OTS will output the same P on both SCs. This is not true for the TEK/TAK pair. Each SC will have its own TEK/TAK pairs. In other words, OTS_1 on SC1 would lead to TEK_a/TAK_a, whereas OTS_1 in SC2 would lead to a different pair, TEK_b/TAK_b. As a consequence, each SC has its own independent and secured channel with the CC. The main goal is to utilize this secure channel to report onboard data relative to the execution of the key agreement protocol. Protocol 4.9 shows how the SCs can securely update the CC with the secret information being utilized onboard.

Notice that in this approach, a single TEK/TAK pair is linked to one OTS, and consequently to P. It means that, as far as the SC uses the same P, the TEK/TAK pair will be also kept the same. Observe that this approach is different from the key transport, and much more efficient in terms of onboard storage. More precisely, in the key agreement approach, each tuple within the TPM has one OTS, one P, and one TEK/TAK pair. In the key transport approach, each OTS would be linked to a TEK/TAK pair, a KHK, and an SSK. The latter is the set of session keys, whose size is in the order of thousands of hundreds of bits.

Obviously key agreement could also be utilized to have the CC and the SC agree on a key. However, in order to agree on a key, the CC and the SC would have to exchange three messages and rely on an additional OTS/P pair. Further details can be found in the discussion section of this chapter.
Protocol 4.9: Key Agreement with Reporting to the CC

SC1

Selects OTS;
Retrieves associated P;
Retrieves associated TEK/TAK;
Selects $R_a$;
Acquires $t_1$;
If $t_1 >>$ or $<<$ current time abort
Else Computes
\[ X = H_d(t_1, A, B, P) g^{R_a} \]

Uses OTS to retrieve $P$ and associated TEK/TAK;
Computes
\[ X_{ab} = X \cdot H_d(t_1, A, B, P) g^{R_b} \]
Selects $R_b$;
Acquires $t_2$;
If $t_2 >>$ or $<<$ current time abort
Else Computes
\[ Y = H_d(t_1, t_2, A, B, P) g^{R_b} \]
and
\[ S_1 = H_d(t_1, t_2, A, B, X_{ab}, g^{R_b}, X_{ab}) \]

Computes
\[ Y_{ba} = Y \cdot H_d(t_1, t_2, A, B, P) g^{R_a} \]
\[ S_1' = H_d(t_1, t_2, A, B, X_{ab}, g^{R_a}, X_{ab}) \]
If $S_1 = S_1'$ then
Acquires $t_3$;
\[ K = H_d(t_1, t_2, t_3, A, B, P, g^{R_a}, g^{R_b}, X_{ab}, X_{ab}) \]
\[ S_2 = H_d(t_1, t_2, t_3, A, B, P, g^{R_a}, g^{R_b}, X_{ab}, X_{ab}) \]
If $t_3 >>$ or $<<$ current time abort
Else Computes
\[ S_2' = H_d(t_1, t_2, t_3, A, B, P, g^{R_a}, X_{ab}, X_{ab}) \]
If $S_2 = S_2'$ then
K = H_d(t_1, t_2, t_3, A, B, P, g^{R_a}, X_{ab}, X_{ab})
Computes
Acquires $t_5$;
\[ E_2 = E_d(t_1, t_2, t_3, t_5, R_b, K)_{TEK_b} \]
and
\[ M_2 = MAC(t_5, OTS, E_2)_{TAK_b} \]

Retrieves TEK_a/TAK_a using OTS;

CC

CC

126
If tag $M_1$ is OK, continue, Else abort;
If $t_4 \gg \text{current time}$ abort, Else
Decrypts $E_1$ to obtain $t_1', t_2', t_3', R_a, K'$;
Retrieves $\text{TEK}/\text{TAK}$ using $\text{OTS}$;
If tag $M_2$ is OK, continue, Else abort;
If $t_5 \gg \text{current time}$ abort, Else
Decrypts $E_2$ to obtain $t_1'', t_2'', t_3'', R_b, K''$;
Checks whether $t_1', t_2', t_3'$ and $K'$
match with $t_1'', t_2'', t_3'', K''$;
Can (optionally) recalculate $K$.

### 4.3.8 Discussions

For regular communications between two SC, why not load $\text{TEK}/\text{TAK}$ pairs onboard instead of passwords ($P$)?

The reason is that each session would require a $\text{TEK}/\text{TAK}$ pair (for encryption and authentication). Therefore, every time that a cryptoperiod is over, two fresh keys would be needed. As a result, the onboard storage would be proportional to the number of times that the $\text{TEK}/\text{TAK}$ pair is changed during the mission lifetime. Once the SCs run out of $\text{TEK}/\text{TAK}$ pairs, a key transport mechanism would have to take place. As previously discussed, key transport mechanisms depend on the availability of the CC or a MSC. An advantage of that method, though, is that it requires a single protocol message; One SC has just to inform its counterpart on the next key pair to be utilized.

In the case of using key agreement, at most two passwords ($Ps$) are needed to generate multiple encryption and authentication SKs, until the cryptoperiod of $P$ is over. The only new data needed for each run of the protocol are fresh $R_a$ and $R_b$, which are generated onboard. Therefore, the onboard storage is proportional to the number of SKs needed, divided by the number of times that $P$ is reutilized in the key agreement protocol. Hence, there is a considerable saving in terms of onboard storage. Besides memory savings, the key agreement method also allows for higher level of autonomy of the SCs. The drawback of this method is that the protocol takes three passes to complete and utilizes more complex mathematical operations (such as modular multiplication, inversion, and exponentiation).

Let’s consider that all 100 SKs are transferred at the same time. Let’s also assume that these SKs are not specific for authentication or encryption so that they can be used for either function. Thus, the onboard storage per SC is:
- Key transport: 1 $\text{OTS}$, 1 $\text{TEK}/\text{TAK}$ (to establish the secured channel), 1 $\text{KHK}$ (to check the integrity if requested by the CC), and 100 SKs.
- Key agreement: 1 $\text{OTS}$, 1 $P$, 1 $\text{TEK}/\text{TAK}$ (for reporting to the CC).

The reason being is that a single $P$ is used to derive all 100 SKs and the same $\text{TEK}/\text{TAK}$ is used to report to the CC – the only thing that is unique for each key agreement are parameters $R_a$ and $R_b$.

Now let’s consider that we differentiate authentication and encryption keys.
- Key transport: 1 $\text{OTS}$, 1 $\text{TEK}/\text{TAK}$, 1 $\text{KHK}$, and 200 SKs. (100 encryption keys, 100 authentication keys)
• Key agreement: 1 OTS, 2 P, 1 TEK/TAK. (One P to agree on encryption keys, and one for agreeing on authentication keys)

In this case, each P derives 100 SKs. Actually, depending on the cryptoperiod of P, it can generate much more than 100 SKs.

Hence, the key agreement demands more processing power but ends up with storage savings. In spite of that, it allows for higher autonomy as the SCs do not require the CC and MSC for transporting keys.

More generally, assume that a mission needs a number of SKs, represented by N.

For the key agreement approach, the number of times that the key agreement protocol has to run is proportional to the total number of keys needed, and is given by N.

Also assume that each password can be reutilized n times before being replaced with a fresh one. The total number of Ps and OTSs are given by N/n, since each OTS/P pair reutilized n times. Similarly, the number of TEK/TAK pairs is also given by N/n.

Therefore, the total amount of onboard storage to support key agreement is given by

\[ S = \frac{N}{n} \times (|OTS| + |P| + |TEK| + |TAK|) \]

Now consider the key transport approach. Each protocol run would transport a set of SKs, named SSK. Suppose that each SSK has m SKs. Thus, the total number of protocol runs (R_t) is given by R_t = N/m. Each protocol run employs one OTS, one TEK, one TAK, and one KHK. Hence, the amount of onboard storage (S_t) in this case is given by:

\[ S_t = \frac{N}{m} \times (|OTS| + |TEK| + |TAK| + |KHK|) + |SSK| \]

It can be noticed that the key transport requires considerable more onboard storage than the key agreement. That is a result of all memory required to store the SSKs for future use.

The two approaches can also be compared in terms of protocol complexity. The optimized version of the key transport protocol utilizes two messages, whereas the key agreement utilizes three.

The size of the messages in the key transport protocol is as follows:

1\textsuperscript{st} message: \[ |M_{t1}| = |OTS| + |SSK| + |timestamp| + |MAC| \]

2\textsuperscript{nd} message: \[ |M_{t2}| = |OTS + 2 \times timestamp| + |MAC| + |MAC| \]

In which the transmission of the first MAC is optional.

Again, it can be observed that the biggest message is the first one, in which the SSKs are transmitted. The size of the SSK can be hundreds of thousands of bits long.

On the other hand, the size of the messages of the key agreement protocol is as follows:

1\textsuperscript{st} message (1\textsuperscript{st} run): \[ |Ma_1| = |timestamp| + |OTS| + |SCID| + |modulo p| \]

1\textsuperscript{st} message (subsequent runs): \[ |Ma_1| = |timestamp| + |SCID| + |modulo p| \]

2\textsuperscript{nd} message: \[ |Ma_2| = |timestamp| + |modulo p| + |H| \]

3\textsuperscript{rd} message: \[ |Ma_3| = |timestamp| + |H| \]

In term of complexity of operations required in the protocols:

Key transport:
- 2 Encryptions (where one of them is over the large set of SKs);
- 2 MACs; and
- 1 Hash (over the SSK).

Key agreement:
- 2 Random Number Generations;
- 5 Hashes;
- 2 Modular Exponentiations
- 4 Modular Multiplications
- 2 Modular Inversions.
- If reporting to the CC is required, 2 Encryptions and 2 MACs would also be needed.

The internal architecture of the TPM performs very similar operation on both schemes. Thus they do not play a major role in terms of operation complexity.

Why not use key agreement in the synchronization between the SC and the CC (at the end of the protocol)?

This can be done in both ways. However, key agreement presents fewer advantages in this case. If key agreement was utilized, the TPM would have to store two additional passwords ($P's$) for the key agreement between the CC and the SC; one for encryption and one for authentication. Thus the number of $P’s$ is proportional to the number of synchronizations. In addition, the CC and the SC would have to exchange three messages to agree on those keys.

Storing $TEK/TAK$ pairs onboard avoids the modular multiplication, inversion, and exponentiation. Additionally, synchronization with the CC can be done with a single protocol message. Most importantly, the $TEK/TAK$ pair can be reutilized multiple times for each $P$. Thus, the number of $TEK/TAK$ pairs to be stored onboard is proportional to the number of $P$s. As a result, it makes more sense to utilize key already stored onboard for the synchronization operations.

**4.3.9 Future Work**

The proposed modification demands the inclusion of timestamps as additional inputs of the hash functions ($H_1$ through $H_5$). Each of those employs internally a series of hash computations. SHA-1 is the hash function listed in [RFC5683]. It would be recommended to utilize SHA-2 instead, due to security concerns on SHA-1. As a consequence, it is necessary to analyze whether those additional values may eventually demand additional hashes to be computed.
4.4 Onboard Trusted Platform Reconfiguration

This section introduces a mechanism that can be utilized to perform the reconfiguration of the onboard trusted platform after the launch of the SC.

4.4.1 Trusted Platform Module

The Trusted Platform Module (TPM) can be utilized to support several cryptographic operations, such as key transport, key agreement, onboard integrity checking, key derivation, just to cite a few. The TPM internal architecture remains basically the same, except for the secrets stored in its memory. For instance, key transport requires encryption, authentication and keyed hash keys be stored; key agreement demands password, and encryption and authentication keys for CC reporting; key derivation requires a seed and keyed hash keys. In all cases, these secrets are indexed by OTSs. Independently of their function, let’s call these tuples as Set of Secret Material (SSM). Some examples of SSMs are:

<table>
<thead>
<tr>
<th>Key Transport:</th>
<th>OTS</th>
<th>TEK</th>
<th>TAK</th>
<th>KHK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key Agreement:</td>
<td>OTS</td>
<td>P</td>
<td>TEK</td>
<td>TAK</td>
</tr>
<tr>
<td>Onboard Key Derivation:</td>
<td>OTS</td>
<td>Seed</td>
<td>TEK</td>
<td>TAK</td>
</tr>
<tr>
<td>Onboard Integrity Checking:</td>
<td>OTS</td>
<td>KHK</td>
<td>TAK</td>
<td></td>
</tr>
</tbody>
</table>

During the course of a mission, these onboard resources are utilized at a certain rate, determined in the pre-launch phase. The number of SSMs is usually a function of the key cryptoperiods and/or number of operations to be performed. For instance, in key transport, the number of SSMs is directly related to the size of SSKs and the SK cryptoperiod. In key agreement, they are related directly to the number of times that P is utilized. In the onboard key derivation and integrity checking contexts, they are a function of number of operations that the CC intends to perform during the mission lifetime.

Even though some margins should be considered while determining the number of SSMs pre-loaded onto the SC, some unexpected events may happen. For example, a memory failure may reduce the number of SSMs available. Radiation in space might eventually corrupt the integrity of the SSMs, which may then need to be discarded. Efficient attacks against crypto mechanisms may force a reduction in the cryptoperiod of keys to keep up with the desired security level.

Hence in some scenarios, SSMs onboard the TPM may need to be “refilled” to maintain their operational duties. In order to achieve that, some reconfiguration mechanism
has to be available in the TPM. An approach is to implement another layer around the TPM, to support its reconfiguration.

### 4.4.2 Uploading Sets of Secret Material

An effective method to upload SSMs is to rely on the same approach based on OTSs. In the reconfiguration layer, the OTS will be named Master OTS (MOTS). Likewise previous approaches, each MOTS is linked to master encryption keys (MTEK) and master authentication (MTAK) keys, as well as master keyed hash keys (MKHKs). Their functions are similar to the key transport approach. MTEK supports confidentiality, MTAK supports data origin authentication, and MKHK supports the integrity check of the materials stored onboard.

The process proceeds in two stages. First, the SSMs are transported to the SC. The SC decrypts its package, checks its MAC, and then store the SSMs into a temporary storage. Second, a keyed hash is computed over this temporary storage, and is compared with the keyed hash value received from the CC. If both values match, then the SSMs have not been corrupted. Thus, then the SC autonomously sets this temporary storage as the main one.

The protocol utilized to upload SSMs to the TPM is shown in Protocol 4.10. The CC initiates the upload by sending a first message to the SC. This message contains a MOTS and an encrypted package under MTEK. The encrypted package has all the new SSMs being uploaded to the TPM. It also has the CC’s expected value for the keyed hash computed over the SSM under TKHK. Additionally, this initial message also has a MAC tag to allow for data origin authentication.

Once the SC receives the first message, it utilizes MOTS to retrieve MTEK, MTAK, and MKHK. Then, it decrypts the encrypted package, checks its MAC and the timestamp, \( t_1 \), of the message. If the MAC does not match with the computed value, or if \( t_1 \) shows an unreasonable discrepancy with the current onboard time, the protocol is aborted. This requires the protocol to be re-executed. On the other hand, if the MAC and \( t_1 \) pass this initial test, then the SSMs are stored onboard, and a keyed hash is computed over it. If the keyed hash \( H' \) matches the one received from the CC, then the SC assumes this SSM as the primary set of onboard secrets to be utilized in future operations.

Next, the SC responds to the CC with an encrypted package. It contains MOTS-1, which is used for key confirmation, the timestamps \( t_1 \) and \( t_2 \). Timestamp \( t_2 \) is used to check the freshness of this message against the CC’s time. If it seems to be an outdated message, the CC may ask the SC to replay it with a fresh timestamp. Last, but not least, this second protocol message has the keyed hash \( (H'(SSM))_{MKHK} \) computed onboard over the SSMs stored into the TPM. This value must match the one pre-computed on the ground, otherwise the CC must request the SC to replay the second message of the protocol. It might be the case that the encrypted package from the SC got corrupted during transmission, but in that case the MAC will also fail. If the MAC has been checked as ok, it may then be the case that the keyed hash value has been corrupted onboard the SC prior to transmission.
Protocol 4.10: Uploading new SSMs onto the TPM

**CC**
- Selects *MOTS*
- Acquires $t_1$
- Initiates protocol
- Uploads SSM

**TPM**

\[ \text{MOTS, } E(t_1, SSM, H(SSM)_{MKHK}^{MTK})_{MTK} \]
\[ \text{MAC(MOTS, } E(t_1, SSM, H(SSM)_{MKHK}^{MTK})_{MTK} \) }_{MTAK} \]

Initialization phase.
- Retrieves *MTK*, *MTAK*, and *MKHK*
- Decrypts $E(t_1, SSM, H(SSM)_{MKHK}^{MTK})_{MTK}$
- Checks MAC and $t_1$
- Stores SSMs
- Acquires $t_2$
- Compute keyed hash over the stored SSMs ($H'$)

\[ \text{E(MOTS-1, } t_1, t_2, H'(SSM)_{MKHK}^{MTK})_{MTK} \]
\[ \text{MAC( } E( \text{MOTS-1, } t_1, t_2, H'(SSM)_{MKHK}^{MTK})_{MTK} \) }_{MTAK} \]

Key confirmation
- Authentication

Checks *MOTS-1*
- Checks MAC
- Checks $t_1$ and $t_2$
- Checks $H'$ of stored SSMs against expected value $H'$
4.5 In-Orbit Redundancy of Key Distribution Centers

This section presents an approach to allow for higher autonomy and redundancy in the mechanisms in the space segment responsible for key management. It is possible to reduce the dependency of a single MSC for key distribution by utilizing multiple MSCs in orbit. Two main cases are proposed, based on two and multiple MSCs. Each approach has its own set of requirements and implementation complexity.

It is worth mentioning that the topics presented in this section were the last ones developed in the very last phase of this research project. The ideas and approaches presented here should be taken as initial concepts only, not as a final approach to in-orbit redundancy for key distribution. Further development of this topic is left as future work.

4.5.1 Master Spacecraft Redundancy – 2 MSCs

The approach employing two MSCs is based on a very similar concept to the key transport mechanism described in Section 4.2 4.1, in which a single MSC is utilized. The goal is to achieve a simple mechanism that relies on the infrastructure previously proposed. As a consequence, that would allow for the launch of a first MSC to provide initial support for key transport. In a second moment, the scheme would be complemented by a second MSC. In fact, this second MSC is the element bringing redundancy and increased autonomy into the system. From this point on, these two MSC will be referred to as MSC$_1$ and MSC$_2$.

In comparison with the single-MSC approach, the main difference of the proposed mechanism is on how its onboard SSKs are utilized. In this approach, both MSC$_1$ and MSC$_2$ carry the same onboard sets of SSKs and SSMs. These sets are loaded onto the MSCs prior to launch time. Update of the set of secret materials onboard can be done while in flight through key transport mechanisms as detailed in Sections 4.2 and 4.4.

The two sets of secret information stored onboard are illustrated in Figure 42. There are $k$ SSKs, indexed $1..k$, which are not initially linked to any SC. In other words, at key transport time, the MSC will perform the link between the SSK and the destination SC in the constellation. There are also multiple SSMs which are the SSMs needed to transport SSKs to each of the $m$ destination SCs in the constellation. For each SC, there are $n$ SSMs, therefore supporting the execution of $n$ key transport operations.

Individual key transport operations would follow exactly the same mechanism as described in Section 4.2. For instance, to transport SSK$_5$ to SC$_1$, the MSC would determine the current SSM to be utilized. For the sake of the exercise, assume that it is SSM$_{1,4}$. Then, the MSC would utilize its OTS$_{1,4}$, TEK$_{1,4}$, TAK$_{1,4}$, and KHK$_{1,4}$, which is also shared with SC$_1$, to transport SSK$_5$. Independently of being MSC$_1$ or MSC$_2$, the same approach is used.
Figure 42: Onboard storage of SSKs and SSMs

Figure 43: Direction of SSKs and SSMs utilization for MSC₁ and MSC₂
The main difference though is how the MSCs utilize these onboard tables. It is important to mention that both SSKs and SSMs are always utilized in sequence. However, MSC\(_1\) indexes the onboard tables in a crescent manner, whereas MSC\(_2\) indexes them in a decrescent way. In other words, in order for MSC\(_1\) to utilize SSK\(_i\), it has to have utilized SSK\(_{i-1}\) already. On the other hand, in order for MSC\(_2\) to utilize SSK\(_i\), it has to have utilized SSK\(_i\) already. The same happens for the SSMs.

The direction in which the onboard tables are utilized is illustrated in Figure 43. While MSC\(_1\) uses the table “top-down”, MSC\(_2\) indexes it “bottom-up”. The reasoning behind it is that both MSCs should be able to transport SSKs to the destination SCs without the consent of its counter-part. This feature is important since it is assumed that the MSCs are not always in line of sight of each other. This way, they should have certain autonomy to proceed with key transport without running the risk of using an SSK/SSM that has already been utilized. The “black” regions are usually not utilized in normal operations, as it is the region of exclusive use of the other MSC. However, it is stored onboard both MSCs as a form of backup. This resource can be requested by the counter-MSC in case of integrity problems. It can eventually be used a primary source of SSKs/SSMs in the total absence of the counter-MSC, for instance due to a major failure.

Both MSCs can freely utilize their SSKs/SSMs in the aforementioned directions, which lies in the white area of Figure 43. However, it can be done until a certain critical point, which is indicated as an “X” in Figure 43. That is a critical zone that should be taken as spare materials, to be temporarily used until the MSC is rekeyed by the CC. Whenever either MSC reaches that point, it cannot assume that the SSKs/SSMs are readily available for use. At that point, it would have to first synchronize with the second MSC to confirm that the materials have not been used previously. Then, a number of SSKs would be reserved so that they can be utilized while the MSCs are not rekeyed. Rekeying of the MSCs happens in conformance to the procedures in Section 4.2.

**4.5.2 Master Spacecraft Redundancy – Multiple MSCs**

Besides the cases just described, higher levels of redundancy may become desirable which may employ more than two MSCs. In this scenario, a sub-constellation consisted of MSCs are specialized for key generation and distribution. It is important to notice that this section of the report outlines the main idea of this approach. This topic can surely be expanded into very deep technical details and has been left as future research.

In a multi-MSC scenario, the 2-MSC approach would present an increasing difficulty in managing the usage of the onboard tables. Splitting a replicated table of secrets and keeping them synchronized with all other MSC in the sub-constellation would quickly grow in complexity as more MSCs are added to the constellation.

A more elegant solution requires the MSCs to have higher levels of autonomy from the CC. In previous approaches, the CC was always responsible for the key generation and upload onto the MSCs. In a multi-MSC approach, each MSC would take the responsibility of generating keys onboard. This does not mean that the MSCs cannot be launched with pre-loaded sets of SSKs. That can surely be done, however, each MSC would also have the capability of generating keys onboard after launch. In order to do that, each MSC would also carry seeds for onboard key derivation. From this point on, key derivation seeds are referred
to as KDS, and they are not to be confused with other abbreviation (e.g. key derivation server) in the literature for key management.

The onboard set of KDS (SKDS) is unique to each MSC. Its function is to allow for the execution of onboard (post-launch) key derivation. The investigation and determination of specific key derivation functions (KDF) that would better fit the needs of the space environment have been left as future work. However, NIST standards [NISTSP80056C, NISTSP800108] would serve as a reference starting point.

If the mission chooses to launch the MSC with SSKs onboard, the set of seeds should be unique to each MSC. When it comes to the SKDS, two approaches could be adopted: 1) The MSCs are always launched with the same SKDS; 2) The MSCs are launched with distinct SKDSs.

In the first approach, the MSC would have to use a random salt value at key derivation time to generate a set of SSKs that differs from the other MSCs. The random salt could be obtained from an entropy source onboard. The salt would have to be shared with the CC at a later time. This would allow the CC to reproduce and verify on the ground the key derivation procedures executed onboard. The main advantage of this approach is that if the onboard SKDS gets corrupted, other MSCs in the constellation would also have the exact same copies that could serve as a backup.

The second approach assumes that MSCs are launched with their unique SKDSs. Consequently, their KDFs would always generate a distinct set of SSKs from the other MSCs. No salt as well as no reporting of the salt value to the CC are necessary. If an entry in the SKDS becomes corrupted, it has to be marked as invalid and then discarded, and this event should be reported to the CC.

In either case, the MSC has to report to the CC to keep it up to date of the onboard key derivations. Likewise in Protocol 4.9, the MSC would utilize TEK, TAK and KHKs to perform this reporting operation. Whenever a random salt is used, that also has to be communicated to the CC through the secured channel. The KHKs would be utilized to hash the SSKs generated onboard, so that its MAC can be later used by the CC to replay the key derivations performed onboard.

Update of SSMs used for key derivation would follow the methods introduced in Section 4.4. Thus, the key derivation mechanisms can be supported for the entire mission lifetime of the MSC.
References


[CCSDS3506G1] CCSDS, Space Missions Key Management Concept, CCSDS 350.6-G-1, November, 2011.


