**Notes:**

**-                      Tidying up the content**

**-                      Finishing the file format tables + example base64 data**

**-                      Service elements**

**-                      Alterations to account for multiple CA descriptions**

**S-100 – Part xx**

**Data Encryption and Authentication**

**Contents.**

Contents

[1. Introduction 4](#_Toc477353404)

[2. Scope 5](#_Toc477353405)

[2.1. Structure under S-100 5](#_Toc477353406)

[*Participants in the Scheme* 5](#_Toc477353407)

[Scheme Administrator 5](#_Toc477353408)

[Data Servers 6](#_Toc477353409)

[Data Clients 6](#_Toc477353410)

[Original Equipment Manufacturers (OEM) 6](#_Toc477353411)

[2.2. The Operation of the Data Protection Scheme. 6](#_Toc477353412)

[3. Dataset Compression and Copy Protection 7](#_Toc477353413)

[3.1. Introduction 7](#_Toc477353414)

[3.2. Compression. 8](#_Toc477353415)

[3.3. Data encryption – Blowfish. 8](#_Toc477353416)

[3.4. Application of the encryption algorithm 9](#_Toc477353417)

[3.5. Detail - User Permits and cell permits. 9](#_Toc477353418)

[3.5.1. User Permit specification. 10](#_Toc477353419)

[3.5.2. M\_ID/M\_KEY format. 11](#_Toc477353420)

[3.5.3. Backwards compatibility. 11](#_Toc477353421)

[3.5.4. Steps to create a user permit. 11](#_Toc477353422)

[3.5.5. Dataset Permit Specification. 11](#_Toc477353423)

[4. Authentication, Identity and Digital Signatures 14](#_Toc477353424)

[4.1. *The operation of the data protection scheme.* 14](#_Toc477353425)

[4.1.1. Responsibilities of participants under the data protection scheme. 15](#_Toc477353426)

[4.1.2. The steps involved in digital signature production. 16](#_Toc477353427)

[4.1.3. Notes on implementation of digital signatures within other product specifications. 16](#_Toc477353428)

[4.2. File Formats for Digital Signatures. 17](#_Toc477353429)

[Definition: Key Pairs: Public Key 18](#_Toc477353430)

[Definition: Key Pairs: Private Key 19](#_Toc477353431)

[Definition: Self Signed Key. 20](#_Toc477353432)

[Definition: signedPublicKey – returned to data client 21](#_Toc477353433)

[Note: This is different if the product specification specifies its own public key format. Where the key format is different the bytes in RED are replaced with the bytes representing the public key and the base64 encoding is also similarly changed.Definition: dataset Digital Signature 22](#_Toc477353434)

[5. Annex A – Example data? 25](#_Toc477353435)

[Reference implementation of data. 25](#_Toc477353436)

# Introduction

Copyright infringement and data piracy are pervasive problems of the digital era. Electronic Navigational Charts (ENC) are not exempt from these issues. As well as the economic impact, the unofficial distribution of nautical information also gives rise to significant safety concerns. As a result, the publishers of official nautical information have sought to protect their data and provide the mariner with a certificate of authenticity through the adoption of a security scheme.

In September 2000, IHO Member States were polled on their views on developing a single IHO Recommended Security Scheme (RSS) and supported the adoption of the Primar Security Scheme as the IHO RSS, as it was at the time the de facto standard for ENC protection. The IHO Committee on Hydrographic Requirements for Information Systems (CHRIS, now HSSC: Hydrographic Services and Standards Committee), at its 13th meeting (Athens, Greece, September 2001), revisited the issue of an RSS and agreed that a small advisory expert group investigate the implications of IHB becoming the security scheme administrator for an RSS and assuming responsibility for the maintenance of an RSS.

The Data Protection Scheme Working Group (DPSWG) reported back to the IHB in January 2002 that there were no technical implications to the IHB becoming the security scheme administrator and that the level of effort to administer the security scheme would be limited and within the IHB resources. As a result, Edition 1.0 of the IHO Data Protection Scheme was adopted in October 2003 as Publication S-63.

The 18th CHRIS meeting (2006) tasked the DPSWG to develop a revised edition of S-63 with the following guidance:

• There would be no introduction of new features; changes would be kept to a minimum.

• Published S-63 guidelines would be included in the standard.

• S-63 would be reorganized to group issues relevant to the IHB as Scheme Administrator, to Data Servers, and to OEMs, respectively.

• There would be a more precise description of the correct implementation of the IHO standard.

Accordingly, a draft Edition 1.1 of S-63 was prepared by DPSWG and endorsed by CHRIS at its 19th meeting (Rotterdam, Netherlands, November 2007). In April 2012, small changes were made to Edition 1.1 to remove the hexadecimal limitation of M\_ID in order to extend the number of possible M\_ID values that the scheme is able to accommodate. This has resulted in this edition 1.1.1 of S-63 which supersedes the previous edition.

Proposals received by the DPSWG at S-100WG 1 (Tokyo), and subsequently HSSC8 (Monaco 2016) supported the view that the most pragmatic way forward with IHO S-63 is to incorporate its content into the evolving IHO S-100. This recognises that:

* data integrity is apart from issues of distribution protection
* that multiple product specifications

# Scope

## Structure under S-100

S-100WG and DPSWG decided to fold all the S-63 content into S-100. The following describes how data protection and authentication is achieved within S-100 product specifications:

* This part of S-100 defines how compression, encryption and authentication may be achieved within any S-100 product specification
* It also describes the operation and technical details surrounding the Data Protection Scheme, run by the Scheme Administrator. This scheme provides the mechanisms for assuring identity within its scope.
* Three individual sections within this part define how compression, encryption and authentication may be implemented both by data servers (those applying the standard to data) and data clients (those using data encoded according to this part)
* Service elements – these elements of the former standard S-63 are now located in ECDIS specific product specifications, namely S-101.

This part of S-100 is designed to be applied after the relevant encoding of data has taken place. It can be applied to any file-based encoding (e.g iso8211 or GML) of an S-100 product specification.

[Note on compatibility with existing scheme for S-57]

## *Participants in the Scheme*

There are several types of users within the scheme, these are:

* The Scheme Administrator (SA). There may be many SAs.
* The Data Server (DS), of which there can be many.
* The Data Client (DC), of which there are many.
* The Original Equipment Manufacturer (OEM) of which there are many.

### Scheme Administrator

The Data Protection Scheme is split into multiple domains. These domains may be defined according to any criteria, e.g regional, functional (e.g IHO vs IALA vs WMO for instance) or they may be local to a particular member state. Each domain is identified by a single name (e.g the IHO domain of hydrographic offices, ECDIS and ENC producers is called “IHO”)

The Scheme Administrator (SA) is solely responsible for maintaining and coordinating the scheme within its domain. The SA role is currently operated by The International Hydrographic Bureau (IHB), as secretariat of the IHO, on behalf of the IHO member states within the IHO domain.

A domain SA is responsible for controlling membership of the scheme and ensuring that all participants operate according to defined procedures. The SA keeps track of which organisations are registered to the scheme and ensures they have access to all the necessary codes and documentation to allow them to implement the standard. The SA maintains the top level digital certificate used to operate the S-63 Data Protection Scheme and is the only body that can certify the identity of the other participants of the scheme. The Scheme Administrator (SA), authenticates the identity of the other participants within the scheme.

All Data Servers and System Manufacturers (OEMs) must apply to the SA to become participants in the scheme and, on acceptance, are supplied with proprietary information unique to them.

The SA is also the custodian of all documentation relating to the S-63 Data Protection Scheme.

### Data Servers

Data Servers are responsible for the encrypting and signing data in compliance with the procedures and processes defined in the scheme. Data Servers issue licences (permits) so that Data Clients, with valid user permits, can decrypt data.

Data Servers will use the M\_KEY and HW\_ID information, as supplied by the SA, to issue encrypted ENC cell keys to each specific installation. Even though the cell keys used to encrypt each cell are identical, they will be encrypted using the unique HW\_ID and therefore cannot be transferred between other ECDIS from the same manufacturer.

**Hydrographic Offices, Value Added Resellers and RENC organisations are examples of Data Servers.**

### Data Clients

Data Clients are the end users of dataset information and will receive protected information from the Data Servers. The Data Client’s software application (OEM System) is responsible for authenticating any digital signatures and decrypting the dataset information in compliance with the procedures defined in the scheme.

***ECDIS/ECS systems are examples of Data Clients.***

### Original Equipment Manufacturers (OEM)

OEMs subscribing to the IHO S-63 DPS must build a software application according to the specifications set out in this document and self-verify and validate it according to the terms mandated by the SA. The S-63 standard contains test data for the verification and validation of OEM applications. The SA will provide successful OEM applicants with their own unique manufacturer key and identification (M\_KEY and M\_ID).

The manufacturer must provide a secure mechanism within their software systems for uniquely identifying each end user installation. The scheme requires each installation to have a unique hardware identifier (HW\_ID).

The software application will be able to decrypt the cell keys using the HW\_ID stored in either the hard lock or soft lock devices attached to or programmed within the application to subsequently decrypt and decompress the data. The digital signature applied to the data can then be verified.

The implementation of data integrity within S-100 is re-usable by S-10x product specifications should they require it. It contains no ENC-specific elements within it.

## The Operation of the Data Protection Scheme.

This document specifies a method of securing datasets and maintaining data integrity with multiple providers serving a large customer base. The purpose of the data protection described here is threefold:

1. Piracy Protection: To prevent unauthorised use of data by encrypting the ENC information.
2. Selective Access: To restrict access to ENC information to only those cells that a customer has been licenced for.
3. Authentication: To provide assurance that the ENC data has come from approved sources

Piracy protection and selective access are achieved by encrypting the data and providing “permits” to decrypt them. Data Servers will encrypt data before supplying it to the Data Client. The encrypted data is then decrypted by the ECS/ECDIS prior to being reformatted and imported into the client system. Authentication is provided by means of digital signatures within the data.

The scheme does not specifically address how information can be protected once it is within an end-user application. This is the responsibility of the OEMs and its responsibilities according to its membership of the data protection scheme.

The scheme allows for compression, copy protection and digital signatures to be applied independently of each other. Datasets can be digitally signed without requiring encryption.

The scheme allows for the mass distribution of encrypted data either online or though the supply of hard media which can be accessed and used by customers with a valid licence containing a set of permits. Selective access to individual datasets is supported by providing users with a licenced set of permits containing the encrypted keys. This licence is created using a unique hardware identifier of the end user system and is specific to each Data Client. Consequently licences cannot be exchanged between individual Data Clients.

This part of S-100 also specifies a compression algorithm to reduce the size of the dataset. Unencrypted data may contains many repeating patterns of information, e.g. coordinate information. Compression is therefore always applied before the dataset is encrypted and uncompressed after the decryption on the data client system.

# Dataset Compression and Copy Protection

## Introduction

It is designed to implement a copy protection scheme allowing for selective unlocking of individual datasets and their installation/processing on a host system which is assigned a unique identification number. The standard sets out the processes, technologies and algorithms for supplying data in an encrypted form and issuing keys to end users to allow them to selectively unlock datasets. The two main properties of the copy protection part of S-63 are to ensure:

1. That data distributed to an end user may only be installed by that end user on a single uniquely named system
2. That data may not be copied for use on another system whether named or not.
3. That individual datasets, identified by a single key, can be unlocked by the end user.

**[TODO: Diagram showing how copy protection works]**

In the following sections the individual components processes of S-63 are documented along with all the necessary technical details. In these sections it is assumed that the input dataset is contained in a file called ***dataset.txt [change this to an S-101 filename for concrete examples]***

***This section defines the following :***

1. Hardware ID (HW\_ID) creation and management by the data client
2. Compression and Encryption of data
3. Construction of permit for data unlocking.
4. Distribution of permit and unlocking of data.

## Compression.

The compression stage is very simple. S-100 uses the ZIP algorithm[[1]](#footnote-1) [6] to compress and decompress datasets. It is identical to the algorithm used in many commercial applications e.g. WinZip, PKZIP. Digital signatures, however, are defined on unencrypted datasets before compression (this allows for selective implementation of digital signatures without an accompanying copy protection scheme) so it is important that the order application is followed correctly.

The ZIP entry [reference] should contain the name of the dataset name and the zip file should be named [datasetname].zip. e.g in our example

***dataset.txt*** is compressed and the resultant file is called ***dataset.txt.zip***

[**TODO**: insert something on how we can zip up auxiliary files too – this allows for encryption of all files within a particular dataset and makes sure that the rest of the standard only has a single file as entry point – ideally all auxiliary files are zipped into the same archive – this is dependent on having a known destination for all auxiliary files, it may be prudent to only allow a single file within a zip archive?]

**TODO: Make sure the zip technical spec is correct.**

## Data encryption – Blowfish.

* ***Default is to use same algorithm***
* ***How Blowfish is applied to file (or byte stream) based data.***
* ***File naming conventions (stays the same – catalogue defines names)***
* ***Key lengths have been extended to 8 bytes.***
* ***Need to specify mode of BF algorithm.***

Only one encryption algorithm is specified within the S-63 data protection scheme itself and this algorithm can be used as a default for new S-100 product specifications if required. The S-63 format works at a dataset level, so it encrypts the complete content of the dataset data files passed to it.

The cell keys used to encrypt the data files are themselves encrypted by the Data Server and supplied to Data Clients as “cell permits”. Information about the encryption algorithm is available in section X.x.x [TODO: Insert reference to algorithm online]

*The scheme encrypts all information using the Blowfish algorithm [9]. The algorithm is unpatented and available in the public domain (*[*www.counterpane.com*](http://www.counterpane.com)*). Blowfish is a block cipher algorithm that operates on 64 bit (8 byte) quantities. It requires that the data sources must be padded if they are not a multiple of 8 bytes. The protection scheme uses the “DES in CBC Mode” padding algorithm defined in [8] whenever any data sources must be padded. This complies with the ECB (Electronic Code Book) mode of DES*

The defined key length is 64 bits arranged in an 8 byte group. Although short by modern standards this key length is used as the baseline for S-100 data. If a longer key length is required this can be defined within an individual product specification and modifications made to the catalogue metadata to indicate the keylength. In this case the permit key lengths will require change as well and specific implementation by data servers for that product specification.

## Application of the encryption algorithm

S-100 only specifies the Blowfish (TODO: insert mode) algorithm for encrypting data files. The resultant file should be given the same filename without the .zip extension. This is so that decryption and unpacking of the data can be done without overwriting any source files. Using our example the data server would take the file produced by the zip algorithm, ***dataset.txt.zip*** , and encrypt it with a chosen encryption key, renaming the output file ***dataset.txt***. The S-100 catalogue metadata notes the dataset name and marks it as encrypted under S-63 edition 2.0.

Example of complete process of compression and encryption of dataset.txt file.

***TODO: Diagram showing simple application along with sample encryption key “43 F2 AC C9 29 94 B8 13” – must show dataset.txt -> dataset.txt.zip -> dataset.txt with key providing the encryption. We can then use the key in the next section to show how the cell permits are created.***

## Detail - User Permits and cell permits.

Introduction:

*To operate the scheme effectively there must be a means where Data Client systems can unlock the encrypted data. To unlock the data the Data Client’s system must have access to the encryption keys. These keys are supplied to the Data Client as a “permit”, encoded in a file.*

*To make each set of cell permits exclusive to an individual end user the cell keys must themselves be encrypted using something that is unique to the Data Clients system. Data Clients assign a unique hardware identifier (HW\_ID) to each individual client system and provide an protected copy of this, in the form of a ”userpermit”, to each Data Client.*

*OEMs encrypt the HW\_ID with their own unique manufacturer key (M\_KEY) so that a HW\_ID cannot be duplicated by another manufacturer. Data Servers have access to each OEMs M\_KEYs and can therefore decrypt the HW\_ID stored in the userpermit. Data Servers encrypt their cell keys with the manufacturers HW\_ID when producing a set of cell permits. This makes them unique to the Data Client and non transferable between Data Client systems.*

The Scheme Administrator must ensure that the data servers maintain the confidentiality of the M\_KEY/M\_ID pairs used to decode user permits (and create cell permits) and the individual OEMs must ensure confidentiality of their individual M\_KEY/M\_ID pair.

### User Permit specification.

A user permit is defined as follows

|  |  |  |  |
| --- | --- | --- | --- |
| **Encrypted HW\_ID (E\_HWID)** | **Check Sum (CRC)** | **M\_ID (Manufacturer ID)** | **M\_MDL (Model)** |
| 8 bytes | 4 bytes | 4 bytes | 2 bytes |

Example:

**09 B8 35 4C AD B0 8D CA BD 08 CA AD 09 89 80 93 55 22**

**|---------| |----------------------||----------||------|**

**CRC E\_HWID M\_ID M\_MDL**

User Permit

* Remains as string of hex characters for ease of transmission
* Length of M\_ID stays as 4 characters (but M\_KEY -> 8 bytes).
* CRC32 value stays as a check (but not a control)
* Length of HW\_ID = 8 bytes (1block of BF).
* E\_HWID same length as before (because of BF padding)
* M\_MDL (model number - optional, useful, in agreement?).
* No restrictions on characters/bytes used for M\_ID/M\_KEY.

### M\_ID/M\_KEY format.

The M\_ID/M\_KEY pairs are defined as follows:

1. 1.M\_ID – the individual identification of the data client system. This is a random 4 byte number encoded in the userpermit in hexadecimal.
2. M\_KEY the corresponding key used to produce the encrypted HW\_ID (EHW\_ID). This is held securely by the data client, used to construct the userpermit and decrypt the permits when received from the data server.

The user permit is sent from the end user system to the data server and allows the data server to extract the HW\_ID. This is then used to encrypt the individual cell keys.

### Backwards compatibility.

*[ a key question for me is whether the fixed M\_ID/M\_KEY scenario is long term sustainable. If an M\_KEY is compromised then it can’t be revoked. Using DH key exchange (with each ECDIS having its own PK) would be a more secure and established way of doing it but is significantly more complex (or could just be in the streams section of the standard] . Should be debated]*

*[agreed within group discussion to leave user permit as unchanged as possible – it’s readability is useful and familiar to users – by all means strengthen (by extendingthe bit length of the E\_HWID and M\_KEY and M\_MDL) but don’t e.g XML or base64 encode it).*

### Steps to create a user permit.

[This procedure is carried out by the implementing end user system, the data client:]

1. Construct 8 byte HW\_ID
2. Get M\_KEY (8 bytes)
3. Encrypt HW\_ID with M\_KEY
4. Append M\_ID
5. Append M\_MDL
6. Calculate CRC32
7. Append rest of userpermit to CRC32.

[ The original standard has some nice tabular step by step diagrams on how to make these up.] we need some reference ones using genuine M\_ID/M\_KEY and illustrating the new M\_MDL and extended length of the M\_KEY]

### Dataset Permit Specification.

**Cell/dataset Permit specification**

XML Specification for permit.

* Includes all new fields.
  + ID – detail needed here – should we standardise how ID is generated and/or require dataserver certificate from the IHO here?.
  + userPermit – originating user permit.
  + serialNum – freetext unique indentifier
  + dateOfIssue [date of issue of permit]
* XML wrapper of elements known from last edition.
  + datasetName – (oldcell name)
  + startDate – NEW start date of permit
  + endDate – old expiry date
  + permitString – just the encrypted dataset key.
  + crc –encrypted permit crc (not cell CRC) – this is calculated from the preceding fields CRCd together then encrypted with the HW\_ID. This prevents modification of the permit [should we replace this with a digital signature of the entire permit?]
* Contains dataset name instead of cell name – this facilitates its use by other products and is non filename format specific. All that is required is an ASCII representation of the filename which can be included in the XML.
* Identifies host system by userpermit. Optional Serial number. [provider?]
* Algorithm uses ASCII encoding of cell, st/end date and calculates permit/encrypted checksum.
* Adds in (encrypted) CRC32 of permit/ (start/end)date and cell so they can’t be changed?
* Permits are then zipped into a PERMIT.ZIP *– name should be neutral but a convention of using the userpermit as the filename might be a good idea(? – discuss). Also multiple permits could be embedded in the same file?*

**Example: Cell Permit**

***<permit version = "2.0"> - should we include an XML format descriptor URI?***

***<header>***

***<id>***

***[Who is sending the permit out]***

***[If this is signed then the id is here]***

***</id>***

***<userpermit>09B8354CADB08DCABD08CAAD09898</userpermit>***

***<serialNum>424-93242-342</serialNum>***

***<date> [do we need a date here (date of issue)] </date>***

***</header>***

***<permits>***

***<dataCategory name="S-101"> [can be multiple ones]***

***<permits>***

***<permit>***

***<dataset>GB400797</dataset>***

***<startDate>20151201</startDate>***

***<endDate>20161231</endDate>***

***<permitString>CC053093B43539A</permitString>***

***<crc>46BB2901</crc>***

***</permit>***

***<permit> <***

***<dataset>GB308210</dataset>***

***<startDate>20151201</startDate>***

***<endDate>20161231</endDate>***

***<permitString>CC9238957253539A</permitString>***

***<crc>77DD3452</crc> - not needed if we go with signed permits***

***</permit>***

***</permits>***

***</dataCategory>***

***</permits>***

***<sig>***

***[ Should we offer ability to sign permits? YES – look at meeting notes…]?***

***[ this can be a single base64 string ]***

***</sig>***

***</permit>***

**Permit XML details.**

|  |  |  |
| --- | --- | --- |
| **Field.** | **Description** | **Example** |
| dataCategory | The S-100 product name | “S-101” |
| datasetName | The name of the dataset which is encrypted (n.b link to metadata catalogue name) |  |
| startDate | The start date of the permit. Format “YYYYMMDD” |  |
| endDate | The expiry date of the permit. Format “YYYYMMDD” |  |
| permitString | The encrypted key of the dateset. The permitString is an ASCII encoded hexadecimal string containing the encryption key encrypted (using the Blowfish Algorithm) with the HW\_ID extracted by the data server from the userpermit supplied by the data client system. |  |
| crc | An encrypted crc of the permit fields (without the XML formatting). This provides a check of individual permits contained within a single permit.zip to ensure they are free from modification. The crc is encrypted with the HW\_ID taken from the userpermit **[if we digitall sign the entire permit then this CRC is not necessarily required]** |  |
| Sig | A digital signature of the entire permit file. This is formatted as per the specification in section [x.x.x] |  |

The Annex Part 13:A contains a complete worked example of a userpermit and associated dataset permits for several different product specifications.

# Authentication, Identity and Digital Signatures

This section defines how digital signatures are constructed for file based data products. It shows how to create digital signatures for data files, how the signatures are encoded and comprehensive examples.

The intention is that product specifications can implement this section directly and that the data created (a base64 encoded string) is embedded into the relevant parts of the catalogue metadata – this is to ensure no separate digital signature files are required for implementation..

*There are two specific sections to this part – one part defines how the data server applies the digital signature algorithm to an individual dataset and produces a digital signature which can be embedded in catalogue metadata. The second part defines the surrounding identity management environment (essentially the data protection scheme which the scheme administrator operates) and the formats and processes defined.It is assumed that the reader is familiar with the concept of digital signatures and public key cryptography. A comprehensive introduction is available at:*

[*https://en.wikipedia.org/wiki/Digital\_signature*](https://en.wikipedia.org/wiki/Digital_signature)

[*https://en.wikipedia.org/wiki/Digital\_Signature\_Algorithm*](https://en.wikipedia.org/wiki/Digital_Signature_Algorithm)

*S-100 uses a pre-existing digital signature algorithm. A detailed explanation of the algorithm itself is beyond the scope of this document and the reader is referred to the Digital Signature Standard (DSS), FIPS Pub 186 (*[*www.itl.nist.gov/div897/pubs/fip186.htm*](http://www.itl.nist.gov/div897/pubs/fip186.htm)*) for a more detailed and accessible explanation. This part of S-100 does not specify code or algorithms for digital signature verification or construction (this is dealt with in the above documentation) but it does provide a set of test data to ensure that implementations are correct.*

### *The operation of the data protection scheme.*

*The digital signature technique used within the data protection scheme uses a standard algorithm and key exchange mechanism widely used elsewhere. S100 digital signatures use asymmetric public key algorithms within a PKI-like infrastructure scheme to unbreakably bind a data file with the identity of the issuer.*

*The scheme relies on asymmetric encryption[[2]](#footnote-2) of a checksum of a data file. By verifying the signature against the issuer’s public key, and also verifying the issuer’s public key against a root level identity the user is assured of the signer’s identity.*

*Operation of the data protection scheme can be considered to have three distinct phases:*

1. *A Scheme Administrator (SA) verifies the identity of a data server and provides them with a certified public key. This is embedded with distributed datasets.*
2. *A Data Server (the implementer of the standard) issues data signed with their private key and supplies their SA-signed public key along with the data*
3. *The data client can then verify the Data Server’s identity (by its association with the SA) and the integrity of the signed data.*

***[]***

### Responsibilities of participants under the data protection scheme.

1. SA
   * Enrolling participants within the data protection scheme.
   * Producing signedPublicKey certificates for end users. Establishing correct identity of applications for signed public keys.
   * Maintaining and distributing the list of M\_ID/M\_KEY pairs.
   * Maintaining the SA key root level keypair.
2. Implementing data clients
   * Maintaining the security of the M\_ID/M\_KEY pairs (if encryption is being implemented by the product specification)
   * Producing systems capable of outputting valid userpermits and importing/decoding valid permits.
   * Producing systems which decode and correctly verify digital signatures.
   * Maintenance of an independently updateable SA certificate on the data client.
3. Data servers
   * Keeping their private keys safe.
   * Compression/encryption of data (if required)
   * Production of valid digital signatures for data
   * Identity authentication with Scheme Administrator and the use of the authenticated identity for promulgation of data.

### The steps involved in digital signature production.

1. When setting up the data protection scheme, the scheme administrator
   1. sets up their own private/public key pair
   2. self-signs their own public key
   3. publicises their own self-signed public key as the “SA Certificate”
   4. This SA certificate is distributed to all data clients for embedding within end user systems.
2. Each data server constructs a public/private key pair
   1. The data server self-signs their public key
   2. The data server sends the self-signed public key to the SA for verification and certification.
   3. The SA verifies the self-signed key and signs it with their own private key. The SA then sends the signed public key back to user.
   4. The data server then embeds the signed public key (their “certificate”) with digital signatures of data (constructed with their private key) and the data itself.
   5. The data client system verifies that:
      1. The SA certificate correctly authenticates the data server’s certificate
      2. The data digital signature is verified by the data server’s public key against the data itself.

[ Note that the SA must use the key lengths and algorithms as defined in this document. If a product specification wants to define its own digital signature algorithms (or different/longer key pairs) it can still be supported within the data protection scheme as the SA only has to verify the identity and sign the public key, then send it back to the participant. The product specification merely specifies its own equivalent of the content in section B of this document to correspond to its own requirements.]

### Notes on implementation of digital signatures within other product specifications.

It is worth noting that the digital signatures of datasets can be independent of the scheme used to assure identity although within this standard they use the same algorithms (this is for simplification of usage and reduction of implementation complexity by implementing systems).

For product specifications requiring digital signatures but requiring algorithms other than those specified in this document (and therefore requiring a digital identity within the IHO data protection scheme) the following process should be followed:

1. The product specification should define its own data integrity measure either as a section in the main document or as a separate (referenced specification)
2. The product specification should ensure it provides an encoding for an end user’s public key in byte form.
3. Participants will create key pairs according to the product specification – these can be of any kind and their byte encodings should be sent to the SA for certification.
4. The SA will certify by signing the public key and will return a DSA digital signature of the users’ public key. This is then the user’s signedPublicKey in the context of S-63.
5. A digital signature then consists of the signature (as defined by the product specification), the SA public key (as defined by Section C of this document) and the signedPublicKey as provided by the SA.
6. Note that the SA will need to independently verify the identity of the implementing organisation and recognise its status before providing a signedPublicKey for use within the data protection scheme.

### File Formats for Digital Signatures.

**This section defines each of the file formats and record contents for the authentication mechanisms and digital signature creation.**

Digital signatures are calculated on unencrypted data to allow for their usage independently of any copy protection scheme also being implemented. It is intended primarily for S-101 but should be able to be adapted to any file based S-10x product. The main outputs of the digital signatures are base64 encoded number sequences which can be slotted into a product specifications’ catalogue metadata.

**For each type of record the following tables show:**

1. **A description of the content of the record.**
2. **An XML representation of the data content.**
3. **A table with each element described.**
4. **A base64 encoding example (this is the encoding used in S-63 ed2.0 as the output format)**

**The following process is followed by the data server in order to create a digital signature of file based data.**

**[ Step by step guide for creation of key pair, issuing of signed public key and receipt of signedPublicKey.]**

### Definition: Key Pairs: Public Key

Notes: Key length for public/private keys for product specifications implementing this section is fixed at XXX bytes.

|  |  |  |
| --- | --- | --- |
| **Description**: A public key is one half of a key pair. As a DSA public key it is composed of three numbers, the XML representation below shows an example encoded in hexadecimal.  Enhanced with “Role” attribute | | |
| <publicKey>  <P> FCA6 82CE 8E12 CABA 26EF CCF7 110E 526D B078 B05E DECB CD1E B4A2 08F3 AE16 17AE 01F3 5B91 A47E 6DF6 3413 C5E1 2ED0 899B CD13 2ACD 50D9 9151 BDC4 3EE7 3759 2E17</P>  <Q>962E DDCC 369C BA8E BB26 0EE6 B6A1 26D9 346E 38C5</Q>  <G>6784 71B2 7A9C F44E E91A 49C5 147D B1A9 AAF2 44F0 5A43 4D64 8693 1D2D 1427 1B9E 3503 0B71 FD73 DA17 9069 B32E 2935 630E 1C20 6235 4D0D A20A 6C41 6E50 BE79 4CA4</G>  <Y>2A64 D7D5 B1A0 6537 DD90 5B6B 2C44 4210 9294 B586 6C25 E959 1C1E CD5C CDE6 75C2 4C26 E32E 904C EAAC C040 E810 7B6E E8DA 8353 8E35 8CD1 E2F3 5A61 E7E8 10BE 29B2</Y>  </publicKey> | | |
| **Field** | **Type / Values** | **Example Content.** |
| ***Public Key Parameter P*** | Integer [128 bytes] | FCA6 82CE 8E12 CABA 26EF CCF7 110E 526D B078 B05E DECB CD1E B4A2 08F3 AE16 17AE 01F3 5B91 A47E 6DF6 3413 C5E1 2ED0 899B CD13 2ACD 50D9 9151 BDC4 3EE7 3759 2E17 |
| ***Public Key Parameter Q*** | Integer [40 bytes] | 962E DDCC 369C BA8E BB26 0EE6 B6A1 26D9 346E 38C5 |
| ***Public key Parameter G*** | Integer [128 bytes] | 6784 71B2 7A9C F44E E91A 49C5 147D B1A9 AAF2 44F0 5A43 4D64 8693 1D2D 1427 1B9E 3503 0B71 FD73 DA17 9069 B32E 2935 630E 1C20 6235 4D0D A20A 6C41 6E50 BE79 4CA4 |
| ***Public Key Y*** | Integer [128 bytes] | 2A64 D7D5 B1A0 6537 DD90 5B6B 2C44 4210 9294 B586 6C25 E959 1C1E CD5C CDE6 75C2 4C26 E32E 904C EAAC C040 E810 7B6E E8DA 8353 8E35 8CD1 E2F3 5A61 E7E8 10BE 29B2 |
| Base64 Representation.  **<publicKey>/KaCzo4Syrom78z3EQ5SbbB4sF7ey80etKII864WF64B81uRpH5t9jQTxeEu0ImbzRMqzVDZkVG9xD7nN1kuFw==li7dzDacuo67Jg7mtqEm2TRuOMU=**  **Z4Rxsnqc9E7pGknFFH2xqaryRPBaQ01khpMdLRQnG541Awtx/XPaF5Bpsy4pNWMOHCBiNU0NogpsQW5QvnlMpA==KmTX1bGgZTfdkFtrLERCEJKUtYZsJelZHB7NXM3mdcJMJuMukEzqrMBA6BB7bujag1OONYzR4vNaYefoEL4psg==</publicKey>** | | |

### Definition: Key Pairs: Private Key

|  |  |  |
| --- | --- | --- |
| **Description**:  The private key | | |
| <privateKey>  <P> FCA6 82CE 8E12 CABA 26EF CCF7 110E 526D B078 B05E DECB CD1E B4A2 08F3 AE16 17AE 01F3 5B91 A47E 6DF6 3413 C5E1 2ED0 899B CD13 2ACD 50D9 9151 BDC4 3EE7 3759 2E17</P>  <Q>962E DDCC 369C BA8E BB26 0EE6 B6A1 26D9 346E 38C5</Q>  <G>6784 71B2 7A9C F44E E91A 49C5 147D B1A9 AAF2 44F0 5A43 4D64 8693 1D2D 1427 1B9E 3503 0B71 FD73 DA17 9069 B32E 2935 630E 1C20 6235 4D0D A20A 6C41 6E50 BE79 4CA4</G>  <Y>7963 182F F2AA C641 A50D 77D1 2057 9F52 C9C8 04C4 0270 C764 6E50 19BA C51A 7E95 300F 44F4 F9D3 A883 09E2 18F6 FC8F 9905 7548 121F 9ADA 09FF 7DF5 D23B 700A 8F56</Y>  <X>9583 A9EE 035C B385 46A7 6F8D 770F AF05 AC5A A5C5</X>  </privateKey> | | |
| **Field** | **Type / Values** | **Example Content.** |
| ***Public Key Parameter P*** | Integer [128 bytes] | FCA6 82CE 8E12 CABA 26EF CCF7 110E 526D B078 B05E DECB CD1E B4A2 08F3 AE16 17AE 01F3 5B91 A47E 6DF6 3413 C5E1 2ED0 899B CD13 2ACD 50D9 9151 BDC4 3EE7 3759 2E17 |
| ***Public Key Parameter Q*** | Integer [40 bytes] | 962E DDCC 369C BA8E BB26 0EE6 B6A1 26D9 346E 38C5 |
| ***Public key Parameter G*** | Integer [128 bytes] | 6784 71B2 7A9C F44E E91A 49C5 147D B1A9 AAF2 44F0 5A43 4D64 8693 1D2D 1427 1B9E 3503 0B71 FD73 DA17 9069 B32E 2935 630E 1C20 6235 4D0D A20A 6C41 6E50 BE79 4CA4 |
| ***Public Key Y*** | Integer [128 bytes] | 7963 182F F2AA C641 A50D 77D1 2057 9F52 C9C8 04C4 0270 C764 6E50 19BA C51A 7E95 300F 44F4 F9D3 A883 09E2 18F6 FC8F 9905 7548 121F 9ADA 09FF 7DF5 D23B 700A 8F56 |
| ***Private Key X*** | Integer [40 bytes] | 9583 A9EE 035C B385 46A7 6F8D 770F AF05 AC5A A5C5 |
| <privateKey>  /KaCzo4Syrom78z3EQ5SbbB4sF7ey80etKII864WF64B81uRpH5t9jQTxeEu0ImbzRMqzVDZkVG9xD7nN1kuFw==li7dzDacuo67Jg7mtqEm2TRuOMU=Z4Rxsnqc9E7pGknFFH2xqaryRPBaQ01khpMdLRQnG541Awtx/XPaF5Bpsy4pNWMOHCBiNU0Nogps  QW5QvnlMpA==eWMYL/KqxkGlDXfRIFefUsnIBMQCcMdkblAZusUafpUwD0T0+dOogwniGPb8j5kFdUgSH5raCf999dI7cAqPVg==lYOp7gNcs4VGp2+Ndw+vBaxapcU=</privateKey> | | |

### Definition: Self Signed Key.

|  |  |  |
| --- | --- | --- |
| **Description**:  The self signed key is submitted by the data server to the SA for signing. It is a digital signature of the data server’s public key (i.e signed by the data servers’ private key). | | |
| Content:  <selfsignedKey>  <publicKey>  <p></p>  <q></q>  <g></g>  <y></y>  <role>  [official body]/[private producer]  </role>  </publicKey>  <R></R>  <S></S>  </selfsignedKey> | | |
| **Field** | **Type / Values** | **Description.** |
| ***Public Key Y – the users’ public key with the individual parameters.*** | DSA public key  P = n bytes  Q = n bytes  G = n bytes  Y = n bytes |  |
| ***Role*** | 2 bytes (encodes producer status] |  |
| ***Signature Part R – DSA signature of public key.*** | Integer [ n bytes] |  |
| ***Signature Part S*** | Integer [n bytes] |  |
| **Hex Dump of entire selfSignedKey bytes** | | |
| **Eventual base64 encoding of selfSignedKey**  <selfsignedKey>  </selfsignedKey> | | |

### Definition: signedPublicKey – returned to data client

|  |  |  |
| --- | --- | --- |
| Description: | | |
| Content:  <signedPublicKey>  <publicKey id=”**idname**”>  <p> </p>  <q> </q>  <g> </g>  <y> </y>  <publicKey>  <publicKey id=”SA”>  <p> </p>  <q> </q>  <g> </g>  <y> </y>  <digitalSignature>  <R></R>  <S></S>  </digitalSignature>  </signedPublicKey> | | |
|  | **Type / Values** | **Description.** |
| ***Public Key [id – this is the users’ public key and may be different depending on product specification.*** | For a DSA public key  **P = n bytes**  **Q = n bytes**  **G = n bytes**  **Y = n bytes** |  |
| ***Public Key [this is the SA public key]this should match (at a byte level) the SA public key installed on the system.*** | DSA public key  P = n bytes  Q = n bytes  G = n bytes  Y = n bytes |  |
| ***Signature Part R***  ***(DSA signature of the users public key by the SA) defined by SA.*** | Integer [ n bytes] |  |
| ***Signature Part S*** | Integer [n bytes] |  |
| **Hex Dump of entire signedPublicKey bytes** | | |
| **Eventual base64 encoding of signedPublicKey**  <signedPublicKey id=”**idname**”>  <signedPublicKey> | | |

### Note: This is different if the product specification specifies its own public key format. Where the key format is different the bytes in RED are replaced with the bytes representing the public key and the base64 encoding is also similarly changed.Definition: dataset Digital Signature

|  |  |  |
| --- | --- | --- |
| Description: | | |
| <digitalSignature id="idname”>  <R>23235252525</R>  <S>55546785347</S> **[signs content + file]**  </digitalSignature> | | |
| **Field** | **Type / Values** | **Description.** |
| ***Signature Part R – DSA signature of data, uses users private key to generate.*** | Integer [ n bytes] |  |
| ***Signature Part S*** | Integer [n bytes] |  |
| **Hex Dump of entire digitalSignature bytes** | | |
| **Eventual base64 encoding of digitalSignature**  <digitalSignature publicKey=”id”>  <digitalSignature> | | |

This is the signature which is calculated per dataset.

**Base64 encoding table for Signature Parameters:**

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Example Content.** |
| ***Public Key Parameter P*** | Integer [128 bytes] | FCA6 82CE 8E12 CABA 26EF CCF7 110E 526D B078 B05E DECB CD1E B4A2 08F3 AE16 17AE 01F3 5B91 A47E 6DF6 3413 C5E1 2ED0 899B CD13 2ACD 50D9 9151 BDC4 3EE7 3759 2E17 |
| ***Public Key Parameter Q*** | Integer [40 bytes] | 962E DDCC 369C BA8E BB26 0EE6 B6A1 26D9 346E 38C5 |
| ***Public key Parameter G*** | Integer [128 bytes] | 6784 71B2 7A9C F44E E91A 49C5 147D B1A9 AAF2 44F0 5A43 4D64 8693 1D2D 1427 1B9E 3503 0B71 FD73 DA17 9069 B32E 2935 630E 1C20 6235 4D0D A20A 6C41 6E50 BE79 4CA4 |
| ***Public Key Parameter Y*** | Integer [128 bytes] | 2A64 D7D5 B1A0 6537 DD90 5B6B 2C44 4210 9294 B586 6C25 E959 1C1E CD5C CDE6 75C2 4C26 E32E 904C EAAC C040 E810 7B6E E8DA 8353 8E35 8CD1 E2F3 5A61 E7E8 10BE 29B2 |

Base 64 representation of the above P,Q,G,Y public key specification:

|  |
| --- |
| <publicKey>/KaCzo4Syrom78z3EQ5SbbB4sF7ey80etKII864WF64B81uRpH5t9jQTxeEu0ImbzRMqzVDZkVG9xD7nN1kuFw==li7dzDacuo67Jg7mtqEm2TRuOMU=  Z4Rxsnqc9E7pGknFFH2xqaryRPBaQ01khpMdLRQnG541Awtx/XPaF5Bpsy4pNWMOHCBiNU0NogpsQW5QvnlMpA==KmTX1bGgZTfdkFtrLERCEJKUtYZsJelZHB7NXM3mdcJMJuMukEzqrMBA6BB7bujag1OONYzR4vNaYefoEL4psg==</publicKey> |

.

# Annex A – Example data?

## Reference implementation of data.

1. Scheme Admin private/public key
2. Self signed SA certificate
3. Example M\_ID/M\_KEY for data client.
4. HW\_ID and EHW\_ID
5. Userpermit
6. Example data files
7. Permits to unlock data files
8. Digital signatures for data files.

Multiple product specs. Example of using RSA keys rather than DSA keys.

EXAMPLE: [Use of RSA keys rather than DSA keys. This example will use RSA keys for production of digital signatures but DSA keys for identity assurance. This allows the scheme administrator to fix their infrastructure algorithms without restricting the choices available to end user product specifications]

1. <http://en.wikipedia.org/wiki/ZIP_%28file_format%29> [↑](#footnote-ref-1)
2. Asymmetric cryptography relies on algorithms where encryption and decryption take place with different cryptographic keys. Therefore one person can encrypt data and make available a decryption key for others to decrypt it. These keys are referred to as the “private key” and the “public key”, collectively known as a “key pair” [↑](#footnote-ref-2)