Open Navigation Surface Working Group

Digital Security Scheme for the Bathymetric Attributed Grid (BAG)

Version 1.1

February 2004

Digital Security Scheme for the Bathymetric Attributed Grid (BAG)

Author: Brian Calder for the Open Navigation Surface Working Group

Version: 1.0
Date: 2004-10-28

Revision History:

Revision	Description	Date	Author
1.0	Initial revision for comment.	2004-10-28	brc
1.1	Minor editing/some revised wording	2005-02-10	brc

Abstract

This document describes the techniques used to implement a Digital Signature scheme in the Open Navigation Surface model for grid-based bathymetric data. The scheme is designed to allow the contents of a Bathymetric Attributed Grid (BAG) to be digitally authenticated, so that a signing authority (i.e., a person given responsibility for preparation or verification of the BAG) can indicate that they have verified the contents of the file, and attest that they have authorized the contents as valid for a particular purpose. The scheme takes the same role for digital hydrographic data as a physical signature does for a traditional smooth sheet summary of a survey. It has the added advantage that it can also show that the contents of the BAG have not been modified, either accidentally or intentionally, between the time that the BAG was authenticated (i.e., signed) and the time that it is verified.

This document describes, at a high-level, the cryptographic techniques used to implement a digital signature algorithm, and their implementation in the BAG format. It also describes a prototype key management scheme that might be used to facilitate signature use.

Background

The Open Navigation Surface (ONS) is a community-led effort to construct a digital file format that contains all of the information required to summarize the results of a hydrographic survey. The project is maintained on an Open Source model, with contributors from industry, government, hydrographic agencies and academia. It is managed by an *ad hoc* Working Group (ONSWG), primarily through an e-mail list, `navsurf_general@ccom.unh.edu`. The ONSWG met for the first time in January 2004 [1], and defined the requirements for the file structure [2] and the file format [3]. Development has continued throughout 2004 on a voluntary basis, and the results will be reported first at the US Hydrographic Conference 2005 in San Diego, CA. The aim of the project is to provide not only a format specification, but also a freely available reference implementation in source-code form.

The ONS file structure is called a Bathymetric Attributed Grid (BAG), and consists of mandatory sections for bathymetry, bathymetric uncertainty, hydrographer's modifications of the bathymetry and meta-data associated with the project. The meta-data follows ISO standards for geospatial data [4, 5], and includes a section for the processing steps that have occurred between data collection and construction of the BAG. 'Processing' here should be interpreted in a very general sense, and this includes the requirement for a statement as to the intended use of the data that is made by a competent authority (e.g., the hydrographer-in-charge). The expectation is that the BAG will take the place of a traditional smooth sheet as the legal archive of a survey. In concept, this statement of intent would follow the current form of a text box containing 'Reviewed and Approved' on a traditional product, followed by an appropriate signature.

However, in a digital product a physical signature cannot be appended, and there is a further requirement to ensure that the product has not been modified (either intentionally or by accident) since it was signed. The BAG structure implements both of these requirements by use of a Digital Signature Algorithm (DSA), a strong cryptographic method that can be used to prove three things:

- The product has been certified to a particular use or stage of processing,
- The certification was done by a particular person or entity, and
- The product has not been modified since it was certified.

This document describes the basis of a DSA, and the implementation of these methods in the BAG format. A feature of a DSA scheme (or any other public key cryptography) is the requirement for a signature authority (SA) – a person or entity designated to sign a document – to know a particular sequence of binary digits, known as the 'private,' or 'secret' key. Since most people have difficulty remembering digital sequences of appropriate length (on the order of 300-400 bytes of data), this sequence must be stored somehow, leading to problems of distribution and management. This document describes one possible method for this, which has been implemented as an example on how to use the source code for ONS's DSA algorithm.

Authentication and Digital Signatures

All DSAs are based on publically-available key cryptographic methods, now commonly used for electronic transactions across public networks such as the Internet. The essence of a public key scheme is a mathematical computation that is easy to do in one direction,

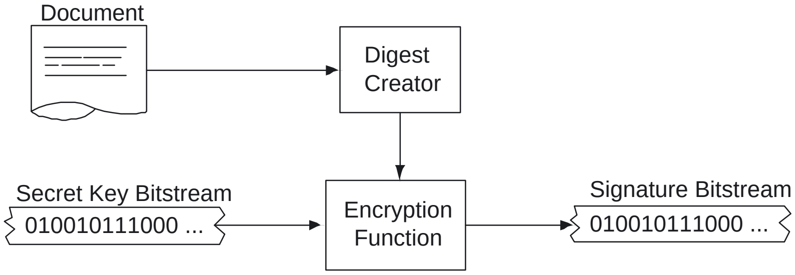


Figure 1: Generic Digital Signature Algorithm (DSA) flow-path. A digest (probabilistically unique number based on the source document) is encrypted with the Signature Authority's secret key to generate the signature.

but very difficult (read: essentially impossible) to reverse. One such example is factorization of large numbers into primes: it is simple, -- given two prime numbers --, to compute their product. However, given only the product, it is extremely difficult to determine the two prime factors that are its only divisors. Other examples include computation of discrete binary logarithms and the solution of elliptic curve equations. All of these methods have been turned into both cryptographic schemes (i.e., for encrypting data to ensure that it can only be read by the appropriate person) and DSAs [6].

All public key systems contain methods for manipulating two pieces of information, known as the public and private (or secret) keys. The public key is made generally known by publishing it on a web site, or somehow making it available for download. The secret key is known only to the user for whom the keys were made, and is not published. These key pairs are sequences of binary digits representing large numbers. In general, they are converted into ASCII representations and then stored in suitably protected files, ready for use. Each SA must have a key pair generated for them individually, and no two key pairs are identical.

In order to sign a document, Fig. 1, the SA uses software to encrypt the document using the secret key (i.e., applies an appropriate mathematical manipulation of the document) to end up with a sequence of binary digits. In practice, it is a complex process to encrypt the entire document. Hence, most algorithms build a summary of the document, called a digest or hash, that is cryptographically secure (i.e., is characteristic of only that document and no other), and then encrypt just the digest. This encrypted digest, or signature, is characteristic of both the document and the secret key. If either changes, the signature would change. Therefore, this digital signature has the same properties as a physical signature: it applies only to one particular document, and it can only be constructed by the SA, since only the SA knows the appropriate secret key. The signature is appended to the document, and is offered to the recipient as an attestation that the document came from the SA.

To verify a document, Fig. 2, the recipient must recompute the digest from the document in exactly the same way that the SA did. Then, the SA's public key is combined with the digest and the signature using an appropriate mathematical manipulation. The important property of this manipulation is that a particular condition can only be true if the secret key used to sign the original digest was the corresponding half of the public key attributed to the SA. If the document is modified, the digest changes and the verification fails; if the wrong public key is used, the verification also fails. Verification of the signature hence assures the recipient that the document has not

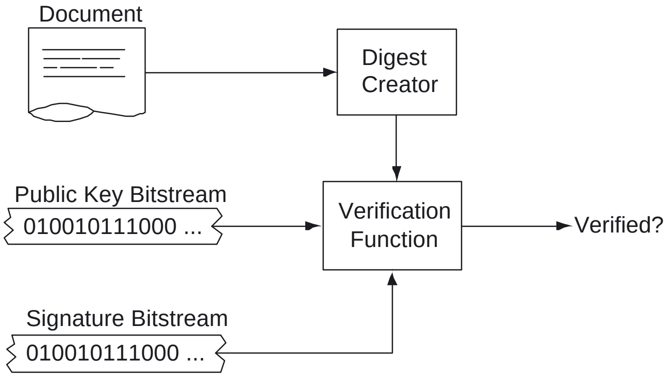


Figure 2: Generic Digital Signature (DS) verification flow-path. The recreated digest is combined with the Signature Authority's public key and compared with the signature. Verification is determined by a particular mathematical property of the combined results.

been tampered with or been received in error. It also confirms that the person purported to have signed the document knew the appropriate secret information that only the SA should know. Hence, the recipient is assured that the document is valid and comes from the SA.

The process of signature and verification is mostly transparent to the user; as long as the appropriate key files are provided, the details of the computation need not be known. It should be noted that this general description does not say anything about how the keys are stored or managed; this is characteristic of each particular scheme. It is also typically the most complex part of the scheme to get right, and the easiest part to get wrong.

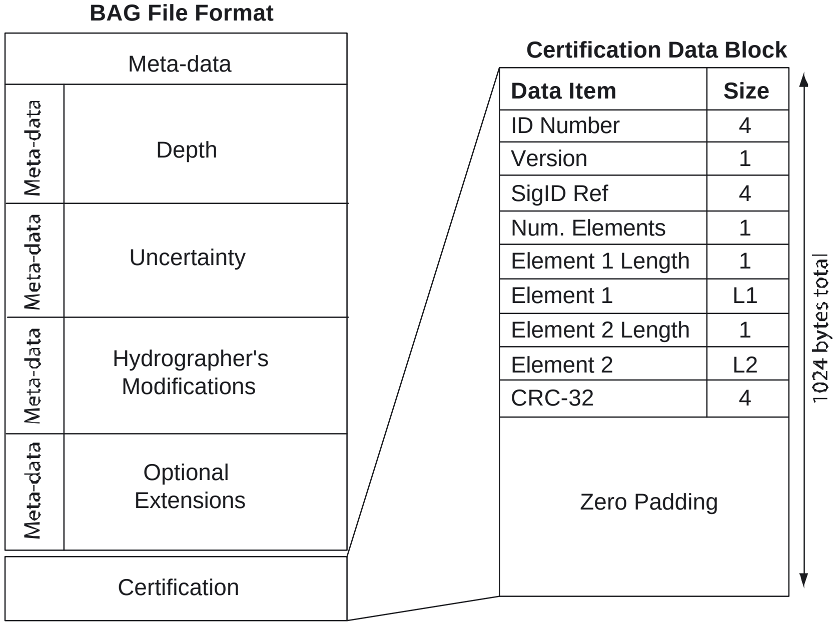
The Open Navigation Surface Implementation

The BAG format (Fig. 3) contains a section of fixed length at the end of the binary file that holds the signature information. This section contains only the signature plus a sequence number that identifies to which meta-data entry the signature applies. This information is appended the first time that a signature is applied, and is replaced every time the file is signed (i.e., after every manipulation that is applied to the data, or at each inspection stage, etc.) The DSA is applied to the rest of the file, hence protecting the entire contents, including the meta-data.

The Open Navigation Surface implementation uses the NIST-approved DSA (as detailed in FIPS 186-2 [7]), with the NIST-approved Secure Hash Standard (FIPS 180-2 [8]) SHA-1 used to compute the message digest to be signed. The signature consists of approximately 40-bytes of data to be stored. At all stages of the process, CRC-32 checksums are computed to ensure reliable detection of problems in transmission or storage of the keys and signatures. The base cryptographic library is BeeCrypt, an Open Source project that is freely available and actively maintained. All of the source code is available from the Open Navigation Surface Project's CVS repository.

The BAG format meta-data contains a section for signature information. Although not directly specified, the ONSWG envisage that any organization would build a list of approved signature types, e.g.,

- This data was gathered with reference to project instructions A, reference B, and meets all of the requirements for hydrographic survey with respect to standards C.



Data Item	Size
ID Number	4
Version	1
SigID Ref	4
Num. Elements	1
Element 1 Length	1
Element 1	L1
Element 2 Length	1
Element 2	L2
CRC-32	4

Figure 3: BAG data structure and certification block. The fixed size of block makes it easier to determine where the block is with respect to the rest of the file.

- This data was verified against original binary data, and was found to be acceptable for use in charting as specified in standard A.
- This data is certified as suitable for the construction of navigational products as defined in standard A.
- This data is certified as suitable for navigation.
- This data represents the best available information for the area, but has not been verified as suitable for navigation.

Multiple certifications may occur in a file, indicating the history of the data as it is manipulated from raw data to the final product used for navigation. Each certification has a reference number, which is included in the signature block. Mismatch of the certification reference number provided by the user and that in the signature block also counts as a verification failure.

It is expected that certain signatures would be restricted in use. For example, ‘suitable for navigation’ might be restricted to a key pair available only to the head of a hydrographic office *ex officia*. Multiple key pairs may be assigned to a particular SA and keys may be retired from time to time. The BAG format intentionally does not mandate any of these conditions, although it allows for them in the implementation structure.

Key Management: Certificates and Hardware Tokens

Storage, dissemination and management of the public and secret keys for the DSA can be problematical. Public keys must be made available; secret keys must be kept secure. At the same time secret keys must be readily portable since they must follow the SA from place to place. In addition, it is essential that some method be found to ensure that the public key provided by an entity does in fact belong to it. Otherwise, it would be possible for an entity to construct their own key pair and sign products just as a valid authority does, as long as they can convince the recipient to accept the appropriate public key from them. While this is acceptable (and might even be encouraged) under some circumstances, the potential for fraudulent signatures is high, and a scheme to authenticate public keys (typically referred to as ‘binding’ the key to the SA’s identity) is required. The BAG format does not mandate how this scheme should be implemented: this is a matter for the organization using the data. However, an example of how this might be done is included in the reference implementation, primarily as an example of how the ONS/DSA proper would be used in practice.

A certificate is a digital file that contains information on the identity of a particular SA (person or corporate entity), along with their public key. Fig. 4 shows a simple example built for a particular SA with the tools provided in the reference implementation. The certificate is formatted so that it is readily read by the signing and verification algorithms. It also provides a convenient method to transport the information. In order to inextricably link the identity and public key, the certificate is sent to a Certificate Signing Authority (CSA). The CSA verifies the information, and then digitally signs the certificate file with a secret key of its own, certifying that the public key belongs to the person identified. This DS by the CSA plays the same role as the DS in a BAG. It ensures that the certificate cannot be modified without the user knowing, and that the CSA guarantees that the key is valid.

```
user {
      name              Brian Calder
      organisation      Center for Coastal and Ocean Mapping and \
                        NOAA-UNH Joint Hydrographic Center
      id                0001
}
public_key {
------ OpenNavigationSurface Public Key ------
03460481008C755FC39808D7D9908A60EFAD34EB4B5D8148722EEC6AC4E5
2946568E510D99C53A850C6C77AB1C82A609E3A5239DCA72773EF8AB5C84
CA2A967F6B1A98E40E8903B94507FA5E7D7E6E3C93C9E2353133028CBC7A
1E98767D7E372C0D914990E1B36167302DC790114E0548D22C0731B7C93F
178D346A0A0EB32CB6DFDAFC315009033C50BA44F10FCD1CDD280B9E9357
C94749A8980629A09EFCB7E2783A674C1D2F936C36956659B1280B2503DA
CA74178C054A849748C051E260023D1BF123D4980C62A8E512A118EEDF8B
7B2F310F1270BD749360D99C7E5893F5EF5315F79B39040985C71E8FFD2C
E24989D77FAEE886790FDCA8CA797071E87EC53DCC2FB463F1E2150657F6
FBA0988CB210C85AA3971B1297B805AE3ED87BD7910E58C02232D9D2B323
C0EB565BA6D83689420B2FAE68FCBACFF89E7302369A950B91CB817AD5E4
F6B0159D96505347A06213E47AA3CC7A91FCE606D703A03994E32486D1E6
1C06F899D251097E4E8438E1D9D1269E0A2A7FC6EF142CD03BCB252F839F
692FD3F3C625CD3FC9DB112DFA6CBE2ECD20ACD735919FE4CF8F6CF94443
A7=
}
signature {
      auth        CCOM Signing Authority
      keysource   http://www.opennavsurf.org/keychain.html
      value       0215008519694BC966114B64AD673EE935482873F51F\
                  771433A448B73A120527AA7CDC4E7EDEC718B78B9EB7\
                  379524DFFF165BA
}
```

Figure 4: A typical simple certificate for a particular SA, containing the required user information, public key and CSA signature.

Anyone wishing to verify a BAG first obtains the certificate of the CSA and that of the purported SA identified in the BAG. Then, the CSA's certificate (Fig. 5) is used to ensure that the SA's certificate is valid using the verification process outlined above (Fig. 6) before the SA's certificate is used to verify the BAG, Fig. 7. The CSA's certificate is signed using the CSA's secret key, essentially self-certifying the identity of the CSA. This implies that the CSA's key management is secure, and therefore that the CSA should be some trusted agency (e.g., a national hydrographic office).

The secret key cannot simply be placed in a file, since knowledge of the SA's secret key would allow any user to sign products as the SA. The example implementation avoids this problem by encrypting the secret key information with a cryptographic scheme (Advanced Encryption Standard, AES [9]) that is based on a pass-phrase that is readily memorized (e.g., 'Now is the time for all good men to come to the aid of the party'), Fig. 8. When the DSA key pair is being generated, the SA provides the pass-phrase in secret. The key generation algorithm uses the Secure Hash Algorithm (SHA-256) to convert the pass-phrase into a digest of length 256-bits. The AES-256 algorithm

```
user {
      name                  CCOM Signing Authority
      organisation          Center for Coastal and Ocean Mapping and \
                            NOAA-UNH Joint Hydrographic Center
      id                    0001
      keysource             http://www.opennavsurf.org/keychain.html
}
public_key {
------ OpenNavigationSurface Public Key ------
03460481009C8C663940D6F6F42FA9E0F77F0A985F2DE9D7CA9E7B20B882
9E91D7353C65727B5190F4722394FE2371F63254482B9EB9E1F3BF30FD44
A6D1F337280D9F12C54F26926F92DAC5B63B6DAD0681D08BF2904E615DEF
F0AF082FC26CACCD465EF410A20E7AC57F1EAD5898695C79AFAEE40413E7
D8D0200B33ECAD3C89E76C5EEB1500A72C4D29370E242707C72408127AB7
857DED29EB805D27A106719033CD4E722A6125E42D4F4EC538D0A16859C7
3F8086F9E5DB6162790E0DAB8ADFA65D6195756B1D26386474E18E247475
615C3498A2A483071ECFA177F8DFE75C6EA9C52576B8E9DCB2D99F9B1DD1
256BAC9A1FAA17CD5022558830EF1CED571D84621910C11C71D7DF4F5EA1
8B3036225A45460F2D16C7B09A0080213FB02F44B74B2A589A31ECC1094D
E463EA6ED27413BF87C266D535D436612D9F79F748B786B9660C988AB314
454AF2611E484B11416B8D371968FB453B6B42461198B0E368DDE2772741
96548EA4E72D199B811DE8AB222D435AE595ABF5E4D87E80DBC2E84BDA38
0AF0B8CAC25AB4BD55C450981DE1228B34AD2E959BCE78474DE337EAB924
DF=
}
signature {
      auth        CCOM Signing Authority
      keysource   http://www.opennavsurf.org/keychain.html
      value       021448C6CA0222E2E4ED0F430CE2C0F1737E63838A37\
                  1459B316FA0D1C684A2179021D89FA23B345939BF740\
                  1C1038B64CF01D
}
```

Figure 5: A typical CSA certificate with required information on the identity of the CSA and signature derived from the CSA's own secret key. This self-sign feature is used to terminate the chain-of-trust with the assumption that the CSA's key management is secure.

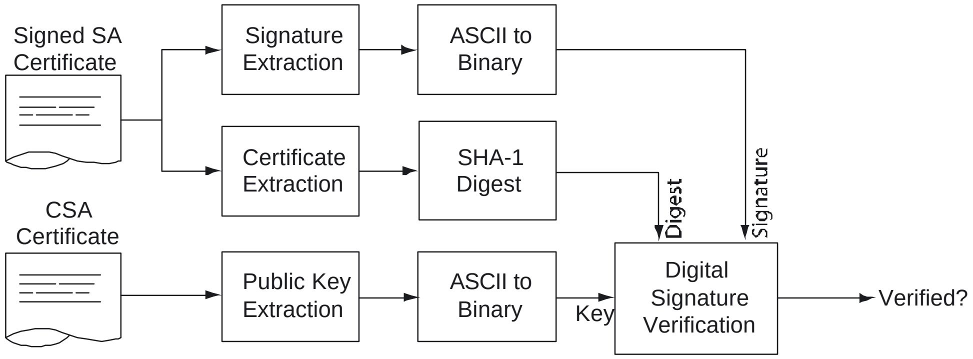


Figure 6: Certificate verification flow-path in example certificate management scheme.

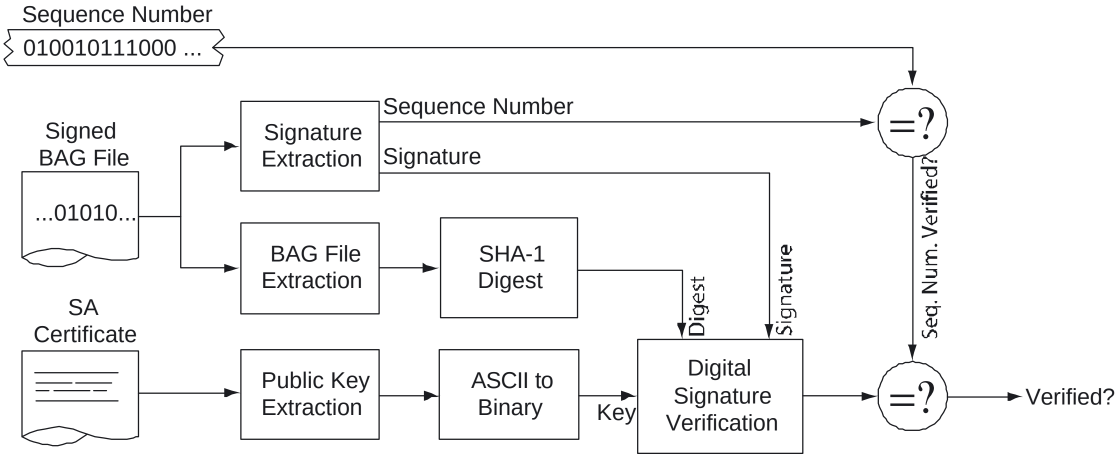


Figure 7: BAG File verification flow-path in the example certificate management scheme. The sequence number is used to link the certification statement in the meta-data to the signature itself.

is then used to encrypt the secret key using the digest, and the encrypted secret key is then written into the EEPROM of a standard USB dongle (Fig. 9).

To sign a document, the SA plugs the dongle into a spare USB slot, and then provides the pass-phrase, again in secret, from memory. The signature algorithm uses SHA-256 to reconstruct the digest from the pass-phrase, reads the EEPROM from the dongle, and decrypts the secret key. The signature scheme outlined above then proceeds as normal, with the secret key never being exposed outside of the running signature program (Fig.10). When the signature is computed, the USB dongle can be removed to a safe location. The same scheme can be used to sign certificates (Fig. 11).

This scheme ensures that the SA does not have to memorize the secret key, ensure that a file containing the secret key is kept secret, or move an encrypted secret key file from place to place. The AES-256 algorithm is sufficiently secure to ensure that the secret key cannot be recovered in any reasonable time without knowledge of the pass-phrase in the event that the USB dongle is lost or stolen. The intent is not that the secret key should never be recoverable. Instead, it should not be recoverable in the time

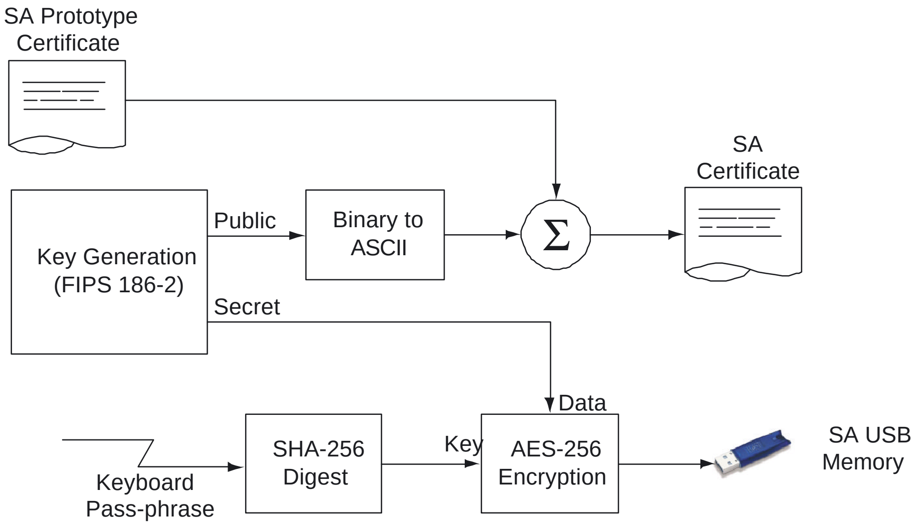


Figure 8: Certificate construction flow-path with the example certificate management scheme.

Figure 9: USB and parallel port dongles, which contain a sufficient amount of memory to store an encrypted secret key. The example certificate management scheme uses the AES symmetric encryption algorithm to encrypt an SA's secret key using a pass-phrase supplied by the SA, and then writes it into the dongle for portability and easy use. Image source: Aladdin Knowledge Systems Ltd.

required to realize that the dongle is gone and then have the CSA repudiate the SA's current certificate, create a new key pair (with a new dongle) and sign the new certificate.

Summary

The Open Navigation Surface (ONS) project is designed to specify a file format to contain information about bathymetric data. To ensure the integrity of this data, and provide an audit trail equivalent to physical signatures on current (hardcopy) products, a Digital Signature (DS) scheme has been proposed.

In the ONS scheme, each signature authority (SA) that wants to provide certified data must construct a key pair according to the FIPS 186-2 [7] Digital Signature Algorithm (DSA). The public key is published, and the secret key is secured. To sign a

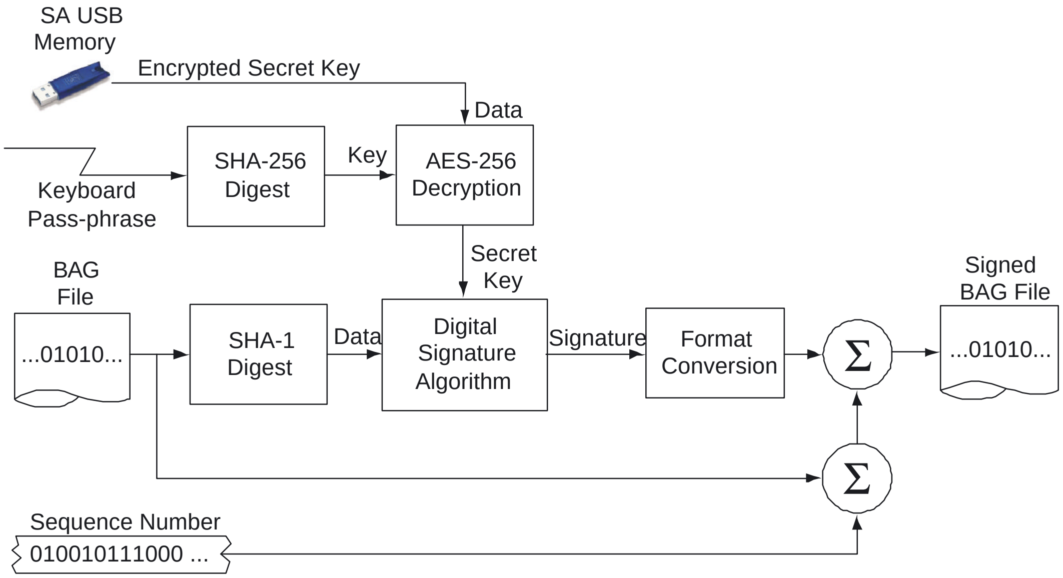


Figure 10: BAG File signature process using the example certificate management.

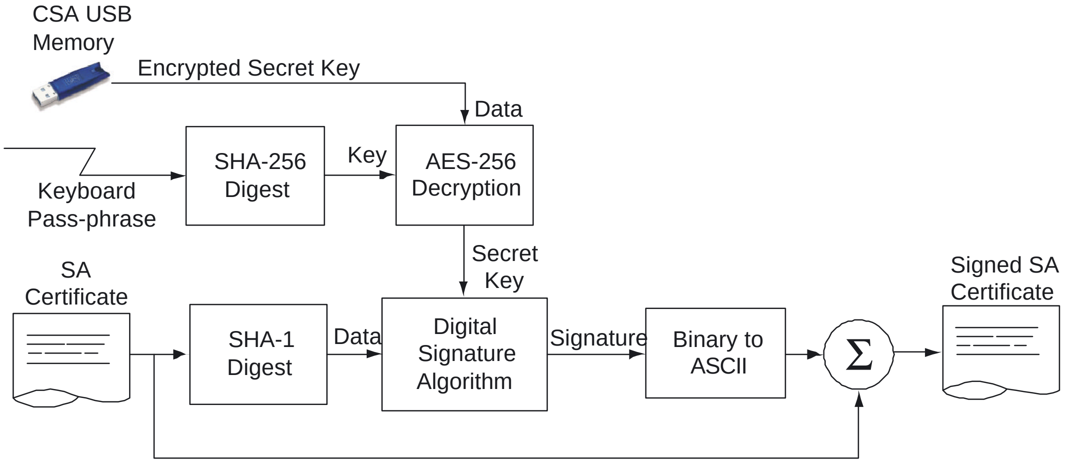


Figure 11: Certificate signature with the example certificate management scheme. The CSA's USB dongle and pass-phrase are used to extract the secret key required to sign the SA's certificate.

Bathymetric Attributed Grid (BAG), an appropriate algorithm (SHA-1 [8]) is used to transform the file into a number essentially unique to that BAG; this number is combined with the secret key to generate another number, called the signature. The signature is stored appended to the end of the BAG in a suitable format. To verify the file, the signature is read and combined appropriately with the public key. If a particular property holds, the user is assured that (a) the file has not been modified since it was signed, and (b) the SA associated with the public key did in fact know the corresponding secret key used to compute the signature (i.e., that the SA did sign the BAG).

The BAG format provides for meta-data to be embedded in the BAG body. This meta-data should contain a certification statement appropriate to the intended use of the data, its provenance, or other qualifications. The signature contains a reference number that may be used to tie the signature to a particular certification in the meta-data (there may be more than one as the data proceeds from source to product). However, the BAG does not enforce this requirement directly, leaving this to the user level code.

The BAG format does not mandate a particular method of representing or managing the key pair required for the DSA. This is something that must be decided by the sponsoring entity. However, an example implementation of how a certificate scheme might be implemented is provided for reference (and as an example of how the code proper should be used). The public keys are represented in ASCII text files with ASCII signatures based on a chain-of-trust model terminating with a Certificate Signing Authority's self-signed certificate. This secret key management scheme uses the SHA-256 algorithm to hash a pass-phrase into an AES-256 key, with which it encrypts the secret key before writing it to the non-volatile memory of a standard USB dongle. To sign a document, the SA presents the USB dongle and enters the pass-phrase. To verify a document, the SA's certificate is parsed to extract the public key. Methods for generating certificates, signing them and verifying their integrity are also provided.

All of the source code for the DSA scheme, including the example key management implementation is available from the Open Navigation Surface's CVS repository. The encryption toolbox is BeeCrypt, an Open Source project that is freely available and actively maintained.

References

[1] Open Navigation Surface Meeting Notes. Inaugural Meeting, January 2004. Available in electronic form; send e-mail to `navsurf_general@ccom.unh.edu`.

[2] Open Navigation Surface: Requirements. The Open Navigation Surface Working Group, January 2004. Available in electronic form as for [1].

[3] The Open Navigation Surface: Bathymetric Attributed Grid Format, V1.0. The Open Navigation Surface Working Group, January 2004. Available in electronic form as for [1].

[4] ISO Standard 19115:2003 Geographic information – Meta-data (`www.iso.ch`)

[5] ISO Proposed Standard 19139: Meta-data – Implementation Specification (`www.iso.ch`)

[6] Applied Cryptography: protocols, algorithms, and source code in C. B. Schneier. Wiley (New York), 1996.

[7] The Digital Signature Standard. National Institute of Standards and Technology, FIPS 186-2, 2000 (`http://csrc.nist.gov/publications/fips/fips186-2/fips186-2-change1.pdf`)

[8] The Secure Hash Standard. National Institute of Standards and Technology, FIPS 180-2, 2002 (`http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf`)

[9] The Advanced Encryption Standard. National Institute of Standards and Technology, FIPS 197, 2001 (`http://csrc.nist.gov/publications/fips/fips197/fips197.pdf`)