Stable Implementation Agreements for Open Systems Interconnection Protocols: Part 12 - Security

Output from the March 1991 NIST Workshop for Implementors of OSI

SIG Chair: **Dr. James Galvin**
SIG Editor: **Maj Doug Naegele, USAF**

Foreword

This part of the Stable Implementation Agreements was prepared by the Security Special Interest Group (SECSIG) of the National Institute of Standards and Technology (NIST) Workshop for Implementors of Open Systems Interconnection (OSI). See Procedures Manual for Woprkshop charter.

Text in this part has been approved by the Pleanary of the above-mentioned Workshop. This part replaces the previously existing chapter on this subject. There is significant technical change from this text as previously given.

Future changes and additions to this version of these Implementor Agreements will be published as change pages. Deleted and replaced text will be shown as strikeout. New and replacement text will be shown as shaded.

Table of Contents

List of Figures

List of Tables

Part 12 - Security

Editor's Note - Previous material in this part has been deleted and is no longer applicable.

0 Introduction

1 Scope

2 Normative References

[**1**] ISO/IEC 9594-8 (CCITT X.509 Recommendation)*Information Technology - Open Systems Interconnection - The Directory - Part 8: Authentication Framework.*

[**2**] ISO 8649: 1988/DAD 1 *Service Definition for the Association Control Service Element, Addendum 1: Peer-Entity Authentication During Association Establishment.*

[**3**] ISO 8650: 1988/DAD 1 *Protocol Specification for the Association Control Service Element, Addendum 1: Peer-Entity Authentication During Association Establishment*.

[**4**] ISO/IEC 9594-3 (CCITT X.511 Recommendation) *Information Technology - Open Systems Interconnection - The Directory - Part 3: Abstract Service Definition*.

[**5**] ISO 10021-4 (CCITT X.411 Recommendation) *Information Processing Systems - Text Communication - MOTIS - Message Transfer System : Abstract Service Definition and Procedures.*

[**6**] ISO 7498-2 *Information Processing Systems - Open Systems Interconnection Reference Model - Part 2: Security Architecture, February 1989.*

3 Definitions

4 Symbols and Abbreviations

5 Architectures

5.1 Introduction

5.2 General OIW Application Environments

5.3 Security Profiles

5.4 Guidelines for OIW Application Profile Development

6 Key Management

7 Lower Layers Security

8 Upper Layers Security

9 Message Handling System (MHS) Security

All current MHS security relevant text appears in Part 8, clause 11.

10 Directory Services Security

11 Network Management Security

11.1 Threats

11.2 Security Services

11.3 Security Mechanisms

12 Security Algorithms

12.1 Integrity

12.1.1 MD4 Hash

12.1.2 SQ-Mod-N

Recent research regarding the square-mod-n one-way hash function described in Annex D of the Directory Documents, ISO 9594 - Part 8, has revealed that the function is not secure. Its use, therefore, is discouraged.

12.1.3 DES MAC

12.2 Authentication

12.2.1 MD4 with RSA Signature

NOTE - This text was moved from Part 11, clause 14.

12.2.2 ElGamal

The information in this subclause includes a tutorial description of the ElGamal scheme for digital signature using the notation defined in the Directory Documents, ISO 9594 - Part 8. It is intended that much of the tutorial information provided in this subclause will be moved to the security agreements sometime in the future.

12.2.2.1 Background

The ElGamal digital signature scheme is based on earlier work done by Diffie and Hellman [DIFF76] in which it was suggested that a likely candidate for a one-way function is the *discrete exponential function*

$$f(x) \equiv \alpha^{x} (\mathrm{mod} p) \tag{1}$$

where *x* is an integer between 1 and *p*-1 inclusive, where *p* is a very large prime number, and where α is an integer such that $1 \leq \alpha \geq p$ and $\{\alpha \bmod p, \alpha^2 \bmod p, \ldots, \alpha^{p-1} \bmod p\}$ is equal to the set $\{1, 2, \ldots, p-1\}$. In algebraic terminology, such an α is called a *primitive element*. References on the topic of primitive roots and elements are [McCl79] and [PATT87].

Now, in the real number system, if $y = \alpha^x$, then by definition of the logarithm we can solve for *x* using $x = \log_\alpha(y)$. The same idea extends to solving eq (1) for x so that inverting $f(x)$ requires calculating *discrete logarithms*. The reason Diffie and Hellman suspected eq (1) is one-way is that for suitable p, it is computationally difficult to invert $f(x)$. According to the current state of the art, computing discrete logs for suitable *p* has been found to require a number of operations roughly equivalent to

$$\exp(\sqrt{cb\ln b}) \tag{2}$$

where *b* is the number of bits in *p*, and *c* is estimated at $c = .69$ according to [ODLY]. This can be compared to only about $2 \log_2 p$ multiplications for discrete exponentiation. If in fact the best known algorithm for computing discrete logs is near optimal then Expression (2)is a good measure of the problem's complexity (for a properly chosen *p*) and the discrete exponential function has all the qualities of a one-way function as described by Diffie and Hellman.

12.2.2.2 Digital Signature

Private Key: X_s denotes the private key for user *X*. X_s is a randomly chosen integer which user *X* keeps secret.

Public Key: X_p denotes the public key for user *X* and is calculated using the corresponding private key such that

$$X_p \equiv \alpha^{X_s} (\mathrm{mod}\, p) \tag{3}$$

where

a) *p* is a prime satisfying the requirements listed in 12.2.2.4.

b) α is a primitive element mod *p*.

c) Note that *p* and α could be used globally, but because they should be easily changeable (see 12.2.2.4 for information about why these two parameters should be easily changeable) it would probably be preferable for each user to choose his/her own *p* and α. If users choose their own, then *p* and α must be made available to the recipient for use in the signature verification process.

Signing Procedure: Suppose user *A* wants to sign a message intended for recipient *B*. The basic idea is to compute a two part signature (*r*, *s*) for the message *m* such that

$$\alpha^{h(m)} \equiv (\mathrm{A}p)^r r^s (\mathrm{mod}\, p) \tag{4}$$

where *h* is a one-way hash function.

Compute the signature (*r*, *s*) as follows.

a) Choose a random number *k*, uniformly between 0 and *p*-1 such that *k* and *p*-1 have no common divisor except 1 (i.e., gcd(*k*,*p*-1)=1).

b) Compute *r* such that

$$r \equiv \alpha^{k} (\mathrm{mod}\, p) \qquad (5)$$

c) Use *r* to solve for the corresponding *s* as follows.

1) rewrite eq (4) using eq (5) and the definition of the public key to get

$$\alpha^{h(m)} \equiv \alpha^{(A_s)r} \alpha^{ks} (\mathrm{mod}\, p) \qquad (6)$$

Combining exponents, get

$$\alpha^{h(m)} \equiv \alpha^{(A_s)r+ks} (\mathrm{mod}\, p) \qquad (7)$$

eq (7) implies that

$$h(m) \equiv (A_s)r + ks (\mathrm{mod}\, p-1) \qquad (8)$$

Note that eq (8) has a single solution for s because k was chosen such that gcd(*k*,*p*-1)=1. See [SIER88] for supporting theorem.

2) now solve for s and get

$$s \equiv I(h(m) - (A_s)r)(\mathrm{mod}\, p-1) \qquad (9)$$

where *I* is computed such that *k* * *I* ≡ 1 (mod *p*-1).

The ElGamal signature is comparable in size to the corresponding RSA signature.

3) Verification

The recipient receives *Ap*, *m*, *r*, *s*, α, and *p* and computes both sides of eq (4) and then compares the results.

4) Known Constraints on Parameters

The following list of constraints is the result of a search of current literature and may not be complete.

a) *p* must be prime

b) *p* must be large.

Note that Expression (2) can be used to speculate on the level of security afforded by crypto systems based on the discrete log problem. Breaking the ElGamal scheme has not been proven to be equivalent to finding discrete logs, but if we assume equivalence then we can estimate how large p should be for a desired level of security.

For instance,suppose we wanted to use Expression (2) to decide how large *p* should be so that we can be reasonably sure the system cannot be broken (using the best *known* algorithm) in a practical amount of time. To be on the conservative side, we decide we want to protect against a special purpose machine that can perform 10^{15} operations per second. Specifically, we want to know how large *p* should be so that such a machine would take at least one year to break the system.

In one year, the hypothetical machine can perform 3 x 10^{22} operations. To find the size of the desired *p*, solve the following equation for *b*.

$$\exp(\sqrt{cb \ln b}) = 3 \times 10^{22} \tag{10}$$

We get $b \approx 606$. This is the number of bits in the desired *p*. So, the magnitude of the desired *p* is about 2^{606} which is roughly 266 x 10^{180}.

Hence, to be reasonably sure of attaining the desired level of security, we find a prime number greater than 266 x 10^{180} which satisfies all the other criteria listed in this subclause. Our confidence, however, is strictly based on the assumption that breaking ElGamal is as difficult as finding discrete logs and the assumption that the best known algorithm for finding discrete logs is near optimal.

c) *p* should occasionally be changed. This requirement is discussed in [ODLY84] and is related to the discovery of new algorithms for computing discrete logarithms in *GF*(*p*).

d) *p*-1 must have at least one large prime factor. This requirement is discussed in [ODLY84] and is imposed by the Silverman-Pohlig-Hellman algorithm p which computes discrete logarithms in *GF*(*p*) using on the order $\sqrt{r}$ operations and a comparable amount of storage, where *r* is the largest prime factor in *p*-1.

e) *p* should not be the square of any prime. A subexponential-time algorithm for computing discrete logarithms in *GF*(p^2) has been found. See [ELGA85b]for details.

1) Note On subjectPublicKey

The ASN.1 data element **subjectPublicKey**, defined as **BIT STRING** in Annex (G) of Directory Documents, ISO 9594 - Part 8, shall be interpreted in the case of ElGamal as being type:

SEQUENCE{ INTEGER, INTEGER }

Where the first integer is the Arithmetic Modulus and the second is the primitive element for the finite field. The sequence is represented by ASN.1 Basic Encoding Rules.

Implementors should take note that the size of the integers used for these parameters is expected to exceed the pragmatic constraints specified for integers by the upper layers SIG.

2) Note On the ENCRYPTED MACRO

The value associated with the ENCRYPTED MACRO, as defined in Directory Documents, part 8, clause 8.4 shall be interpreted in the case of ElGamal as being type:

SEQUENCE{ INTEGER, INTEGER }

The first integer in the sequence is *r* (see eq (5), 12.2.2.2). The second integer is s (see eq (9), 12.2.2.2).

> **NOTE -** 12.2.3 to 12.4.1, along with portions of annexes b and c, have not appeared in any workshop documentation prior to this time.

FIPS 112 Password Encryption

DES MAC

f) Confidentiality

DES in CBC Mode Encryption

g) ASN.1 Definitions

General Security Algorithms

Refer to Part 13, clause 12.4.1 of the Working Agreements as of March 1991.

ASN.1 for Directory Services Strong Authentication Algorithms

> **Editor's Note -** The following algorithms were originally registered by the Directory Services SIG, hence the Object ID remains dssig(7).

This subclause defines object identifiers assigned to authentication algorithms. The definitions take the form of the ASN.1 module, ‘‘OIWAlgorithmObjectIdentifiers’’.

```
OIWAlgorithmObjectIdentifiers {iso(1) identified-organization(3)
  oiw(14) dssig(7) oIWAlgorithmObjectIdentifiers(1)}
DEFINITIONS ::=
BEGIN

EXPORTS
  md2, md2WithRSA, elGamal, md2WithElGamal;

IMPORTS
  authenticationFramework
    FROM UsefulDefinitions {joint-iso-ccitt ds(5) modules(1)
                            usefulDefinitions(0)}
  ALGORITHM
    FROM AuthenticationFramework authenticationFramework;

-- categories of object identifiers

algorithm OBJECT IDENTIFIER ::= {iso(1) identified-organization(3)
                                 oiw(14) dssig(7) algorithm(2)}

encriptionAlgorithm OBJECT IDENTIFIER ::= {algorithm 1}

hashAlgorthm OBJECT IDENTIFIER        ::= {algorithm 2}

signatureAlgorithm OBJECT IDENTIFIER  ::= {algorithm 3}

-- algorithms

md2 ALGORITHM
    PARAMETER NULL
    ::= {hashAlgorithm 1}

md2WithRsa ALGORITHM
    PARAMETER NULL
    ::= {signatureAlgorithm 1}

elGamal ALGORITHM
    PARAMETER NULL
    ::= {encryptionAlgorithm 1}

Editor’s Note: Refer to the June 1990 Working Agreements for information
regarding why PARAMETER NULL is specified above for the elGamal
encryption algorithm.

md2WithElGamal ALGORITHM
    PARAMETER NULL
    ::= {signatureAlgorithm 2}

END -- of Algorithm Object Identifier Definitions
```

Figure 1 - ASN.1 for directory services strong authentication

Table 1 - SIA part 12 changes

NO. OF ERRATA	TYPE	REFERENCED DOCUMENT	CLAUSE	NOTES
	TECHNICAL	SIA PART - 12	0..12	ADD OUTLINE 2ND LEVEL
	TECHNICAL	SIA PART - 12	9	ADD TEXT
	TECHNICAL	SIA PART - 12	12.1.2	ADD TEXT
	TECHNICAL	SIA PART - 12	12.2.2	ADD TEXT
	TECHNICAL	SIA PART - 12	12.4.1/.2	ADD TEXT

CBibliography