

---

Stream: Independent Submission  
RFC: [9548](#)  
Category: Informational  
Published: April 2024  
ISSN: 2070-1721  
Author: E. Karelina, Ed.  
*InfoTeCS*

# RFC 9548

## Generating Transport Key Containers (PFX) Using the GOST Algorithms

---

### Abstract

This document specifies how to use "PKCS #12: Personal Information Exchange Syntax v1.1" (RFC 7292) to transport key containers (PFX) for storing keys and certificates in conjunction with the Russian national standard GOST algorithms.

This specification has been developed outside the IETF. The purpose of publication is to facilitate interoperable implementations that wish to support the GOST algorithms. This document does not imply IETF endorsement of the cryptographic algorithms used here.

### Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This is a contribution to the RFC Series, independently of any other RFC stream. The RFC Editor has chosen to publish this document at its discretion and makes no statement about its value for implementation or deployment. Documents approved for publication by the RFC Editor are not candidates for any level of Internet Standard; see Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at <https://www.rfc-editor.org/info/rfc9548>.

### Copyright Notice

Copyright (c) 2024 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

## Table of Contents

1. Introduction	3
2. Conventions Used in This Document	4
3. Basic Terms and Definitions	4
4. PFX	5
4.1. Structure of PFX	5
4.2. AuthenticatedSafe	5
4.2.1. Unencrypted Data	6
4.2.2. Password-Encrypted Data	6
4.3. SafeContents and SafeBag	6
5. GOST R 34.10-2012 Key Representation	7
5.1. Masking GOST R 34.10-2012 Keys	7
5.2. KeyBag Structure for GOST R 34.10-2012 Key	8
5.3. OneAsymmetricKey Structure	9
5.4. EncryptedPrivateKeyInfo Structure for GOST R 34.10-2012 Key	9
6. GOST R 34.10-2012 Certificate Representation	10
7. Security Mechanisms	10
8. Security Considerations	11
9. IANA Considerations	11
10. ASN.1 Modules	12
11. References	12
11.1. Normative References	12
11.2. Informative References	13
Appendix A. Examples	13
A.1. Test Data	13
A.1.1. Test Certificate	13

---

A.1.2. Test Key	14
A.2. Example of a PFX with a Password-Protected Key and Unencrypted Certificate	14
A.2.1. PFX in BASE64 Format	15
A.2.2. PFX in ASN.1 Format	15
A.2.3. Decrypted Key Value in BASE64 Format	19
A.2.4. Decrypted Key Value in ASN.1 Format	19
A.3. Example of a PFX with a Password-Protected Key and a Password-Protected Certificate	19
A.3.1. PFX in BASE64 Format	20
A.3.2. PFX in ASN.1 Format	20
A.3.3. Decrypted Key Value in BASE64 Format	23
A.3.4. Decrypted Key Value in ASN.1 Format	23
Acknowledgments	24
Author's Address	24

## 1. Introduction

This document provides a specification of the usage of GOST algorithms with PKCS #12 v1.1.

PKCS #12 v1.1 describes a syntax for transfer of personal information such as private keys, certificates, and various secrets.

This memo describes the creation of transport key containers (PFX) for keys and certificates using the GOST R 34.10-2012 algorithm. The GOST R 34.11-2012 algorithm is used to ensure the integrity of PFX.

Caution:

This specification is not a standard and does not have IETF community consensus. It makes use of a cryptographic algorithm that is a national standard for Russia. Neither the IETF nor the IRTF has analyzed that algorithm for suitability for any given application, and it may contain either intended or unintended weaknesses.

## 2. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

## 3. Basic Terms and Definitions

Throughout this document, the following notations are used:

P a password encoded as a Unicode UTF-8 string

S a random initializing value

$V_s$  the set of byte strings of length  $s$ , where  $s \geq 0$ ; the string  $b = (b_1, \dots, b_s)$  belongs to the set  $V_s$  if  $b_1, \dots, b_s$  belongs to  $\{0, \dots, 255\}$

$|A|$  the number of components (a length) of the vector  $A$  belonging to  $V_s$  (if  $A$  is an empty string, then  $|A| = 0$ )

$A|C$  a concatenation of two byte strings  $A, C$  from  $V_s$ , i.e., a string from  $V_{|A|+|C|}$ , where the left substring from  $V_{|A|}$  is equal to the string  $A$  and the right substring from  $V_{|C|}$  is equal to the string  $C$ :  $A = (a_1, \dots, a_{n_1})$  in  $V_{n_1}$  and  $C = (c_2, \dots, c_{n_2})$  in  $V_{n_2}$ ,  $res = (a_1, \dots, a_{n_1}, c_2, \dots, c_{n_2})$  in  $V_{n_1+n_2}$

$F_q$  a finite prime field represented as a set of  $q$  integers  $\{0, 1, \dots, q - 1\}$ , where  $q > 3$  - prime number

$b \bmod q$  the minimum non-negative number comparable to  $b$  modulo  $p$

$INT(b)$  integer  $INT(b) = b_1 + b_2 * 256 + \dots + b_s * 256^{s-1}$ , where  $b$  belongs to  $V_s$

This document uses the following terms and abbreviations:

**Signature** one or more data elements resulting from the signature process (Clause 3.12 of [ISO14888-1]). Note: The terms "digital signature", "electronic signature", and "electronic digital signature" are considered equivalent in this document.

**Signature key** set of private data elements specific to an entity and usable only by this entity in the signature process (Clause 3.13 of [ISO14888-1]). Note: Sometimes called a private key.

**Verification key** set of public data elements that is mathematically related to an entity's signature key and is used by the verifier in the verification process (Clause 3.16 of [ISO14888-1]). Note: Sometimes called a public key.

ASN.1 Abstract Syntax Notation One, as defined in [X.680].

BER Basic Encoding Rules, as defined in [X.690].

HMAC\_GOSTR3411 Hash-Based Message Authentication Code. A function for calculating a Message Authentication Code (MAC) based on the GOST R 34.11-2012 hash function (see [RFC6986]) with 512-bit output in accordance with [RFC2104].

## 4. PFX

The PFX (see [RFC7292]) is designed for secure storage and data transfer. The scope of this document is to define how PFX is used for private key and certificate protection with a password when GOST R 34.10-2012 is applied.

### 4.1. Structure of PFX

In accordance with [RFC7292], PFX has the following structure:

```
PFX ::= SEQUENCE
{
  version      INTEGER {v3(3)}(v3, ...),
  authSafe     ContentInfo,
  macData      MacData OPTIONAL
}
```

The fields of the PFX have the following meanings:

- version is the syntax version number; the only allowed value for this specification is 3.
- authSafe contains the data of type ContentInfo. In the case of password integrity mode, the authSafe.content field has a Data type value and contains a BER-encoded value of the AuthenticatedSafe structure.
- macData has a MacData type; in the case of password integrity mode, the macData field should contain information about the algorithm and parameters for password key generation. Integrity control is ensured by using the HMAC\_GOSTR3411\_2012\_512 algorithm: the macData.mac.digestAlgorithm.algorithm field contains the HMAC\_GOSTR3411\_2012\_512 algorithm identifier (see Section 7). When processing PFX, this field should be checked first.

### 4.2. AuthenticatedSafe

The AuthenticatedSafe structure is a sequence of ContentInfo values (see [RFC5652]):

```
AuthenticatedSafe ::= SEQUENCE OF ContentInfo
-- Data if unencrypted
-- EncryptedData if password-encrypted
-- EnvelopedData if public key-encrypted
```

### 4.2.1. Unencrypted Data

If the data is not encrypted, then the content field is the BER-encoded value of the SafeContents structure. The contentType field is set to the id-data type.

### 4.2.2. Password-Encrypted Data

When password integrity mode is used, the data is represented as an EncryptedData structure (see [RFC5652]). The encryption algorithm and parameters have the following values:

```
ContentEncryptionAlgorithmIdentifier ::= SEQUENCE
{
  encryptionAlgorithmOID  OBJECT IDENTIFIER,
  parameters               PBES2-params
}
```

The PBES2-params type is defined in [RFC9337]. The content should be encrypted according to the encryption algorithm in the PBES2 scheme, as described in [RFC9337]. The following identifier **MUST** be specified in the EncryptedData.EncryptedContentInfo.contentEncryptionAlgorithm. encryptionAlgorithmOID field:

```
{
  iso(1) member-body(2) us(840) rsadsi(113549)
  pkcs(1) pkcs-5(5) pbes2(13)
}
```

The encrypted content is specified in the EncryptedData.EncryptedContentInfo.encryptedContent field.

## 4.3. SafeContents and SafeBag

In accordance with [RFC7292], the SafeContents structure is a sequence of SafeBag:

```
SafeContents ::= SEQUENCE OF SafeBag
```

where

```
SafeBag ::= SEQUENCE
{
  bagId          BAG-TYPE.&id ({PKCS12BagSet})
  bagValue [0]   EXPLICIT BAG-TYPE.&Type({PKCS12BagSet}@bagId})
  bagAttributes SET OF PKCS12Attribute OPTIONAL
}
```

The fields of SafeBag have the following meanings:

- bagId is an object identifier; it defines the type of object.
- bagValue is the value of an object.
- bagAttributes contains the users' names, the key identifiers, and other additional information. This field is optional.

See [RFC7292], Section 4.2 for the different bag types. This document describes the two object types of the SafeBag structure:

1. pkcs8ShroudedKeyBag
2. certBag

When password integrity mode is used, the private key has the following structure:

```
pkcs8ShroudedKeyBag BAG-TYPE ::=
{
    PKCS8ShroudedKeyBag IDENTIFIED BY {bagtypes 2}
}
```

The bagValue field contains the key and information about the key, in encrypted form, in the EncryptedPrivateKeyInfo structure.

A certBag contains a certificate of a certain type. Object identifiers are used to distinguish between different certificate types.

```
certBag BAG-TYPE ::=
{
    CertBag IDENTIFIED BY { bagtypes 3 }
}
```

If the certificate is not encrypted, the CertBag structure is placed in the Data structure (see [RFC5652]). If the certificate is encrypted, the CertBag structure is placed in the EncryptedData structure (see [RFC5652]).

## 5. GOST R 34.10-2012 Key Representation

This section describes the GOST R 34.10-2012 private key representation for asymmetric key pairs. Masked keys should be used to ensure that private keys are protected from leaking through side channels when reading and performing operations with keys.

### 5.1. Masking GOST R 34.10-2012 Keys

The masking algorithm is defined by the basic cryptographic transformation operation of the algorithm: multiplication in the  $F_q$  field for GOST R 34.10-2012 keys.

Let  $M_1, M_2, \dots, M_k$  be a sequence of  $k$  masks. Let  $M_i()$  denote the operation of applying the  $i$ -th mask and  $M_i^{-1}()$  denote the operation of removing the  $i$ -th mask,  $1 \leq i \leq k$ . Let  $K$  be a key. The masked key  $K_M$  is obtained by applying the masking operation  $k$  times:

$$K_M = M_k (...(M_2(M_1(K))...)).$$

Unmasking is performed by applying the removal operation  $k$  times, but in reverse order:

$$K = M_1^{-1}(...(M_{k-1}^{-1}(M_k^{-1}(K_M))...)).$$

The masked key is represented as the sequence

$$I = K_M || M_1 || M_2 || \dots || M_k.$$

Let the key  $K$  be  $n$  bits in length; then, the sequence  $I$  is represented in memory as a sequence of  $(k + 1) * n$  bits.  $I$  is represented in little-endian format. It is possible to use an unmasked private key (i.e.,  $k = 0$ ,  $K_M = K$ ). For GOST R 34.10-2012 keys, the masking operation is the multiplication of the key by the inverse of the mask:  $\text{INT}(K_M) = \text{INT}(K) * \text{INT}(M)^{-1} \bmod Q$ , where the  $Q$  value is taken from the key parameters. The operation of removing the mask is the multiplication of the masked key by the mask:  $\text{INT}(K) = \text{INT}(K_M) * \text{INT}(M) \bmod Q$ . The public key is specified by a pair of coordinates  $(x, y)$  as defined in GOST R 34.10-2012, presented in the following format:

- a public key corresponding to the GOST R 34.10-2012 algorithm with a key length of 256 bits has the `GostR3410-2012-256-PublicKey` representation. It is specified by a 64-byte string, where the first 32 bytes contain the little-endian representation of the  $x$  coordinate and the last 32 bytes contain the little-endian representation of the  $y$  coordinate.
- a public key corresponding to the GOST R 34.10-2012 algorithm with a key length of 512 bits has the `GostR3410-2012-512-PublicKey` representation. It is specified by a 128-byte string, where the first 64 bytes contain the little-endian representation of the  $x$  coordinate and the last 64 bytes contain the little-endian representation of the  $y$  coordinate.

The public keys `GostR3410-2012-256-PublicKey` and `GostR3410-2012-512-PublicKey` **MUST** be DER encoded as an octet string in accordance with [Section 4.3](#) of [\[RFC9215\]](#):

```
GostR3410-2012-256-PublicKey ::= OCTET STRING (64),
GostR3410-2012-512-PublicKey ::= OCTET STRING (128).
```

## 5.2. KeyBag Structure for GOST R 34.10-2012 Key

In accordance with [\[RFC7292\]](#), a KeyBag is defined as information about a private key represented as the `PrivateKeyInfo` structure:

```
KeyBag ::= PrivateKeyInfo
```

In accordance with [\[RFC5958\]](#), information about a private key is presented in the following form:

```
PrivateKeyInfo ::= OneAsymmetricKey
```

### 5.3. OneAsymmetricKey Structure

In accordance with [\[RFC5958\]](#), OneAsymmetricKey has the following structure:

```
OneAsymmetricKey ::= SEQUENCE
{
    version                Version,
    privateKeyAlgorithm    PrivateKeyAlgorithmIdentifier,
    privateKey             PrivateKey,
    attributes             [0] Attributes OPTIONAL,
    ...,
    [[2:publicKey          [1] PublicKey OPTIONAL]],
    ...
}
Version ::= INTEGER { v1(0), v2(1) } (v1, ..., v2)
PrivateKeyAlgorithmIdentifier ::= AlgorithmIdentifier
PrivateKey ::= OCTET STRING
PublicKey ::= BIT STRING
Attributes ::= SET OF Attribute
```

The fields have the following meanings:

- version identifies the version of OneAsymmetricKey. If publicKey is present, then version is set to 2; else, version is set to 1.
- privateKeyAlgorithm identifies the private key algorithm and optionally contains parameters associated with the asymmetric key pair. For GOST R 34.10-2012 private keys, the identifiers of the corresponding public keys are used; they are defined in [\[RFC9215\]](#). The use of identifiers and public key parameters is defined in [\[RFC9215\]](#).
- privateKey is an OCTET STRING that contains the value of the masked private key I.
- attributes are optional. They contain information corresponding to the public key (e.g., certificates).
- publicKey contains the value of the public key GostR3410-2012-256-PublicKey or GostR3410-2012-512-PublicKey encoded in a BIT STRING. This field is optional.

### 5.4. EncryptedPrivateKeyInfo Structure for GOST R 34.10-2012 Key

In accordance with [\[RFC7292\]](#), the encrypted information regarding the private key is defined as the PKCS8ShroudedKeyBag structure:

```
PKCS8ShroudedKeyBag ::= EncryptedPrivateKeyInfo
```

In accordance with [\[RFC5958\]](#), EncryptedPrivateKeyInfo has the following structure:

```
EncryptedPrivateKeyInfo ::= SEQUENCE
{
    encryptionAlgorithm EncryptionAlgorithmIdentifier,
    encryptedData        EncryptedData
}
EncryptionAlgorithmIdentifier ::= AlgorithmIdentifier
EncryptedData ::= OCTET STRING
```

The fields have the following meanings:

- encryptionAlgorithm identifies the algorithm under which the private key information is encrypted. Encryption **MUST** use the PBES2 scheme. The algorithm and parameters of this scheme are presented in [\[RFC9337\]](#).
- encryptedData is the DER-encoded PrivateKeyInfo structure.

## 6. GOST R 34.10-2012 Certificate Representation

In accordance with [\[RFC7292\]](#), a CertBag is defined as information about a certificate and has the following structure:

```
CertBag ::= SEQUENCE
{
    certId          BAG-TYPE.&id ({CertTypes}),
    certValue [0] EXPLICIT BAG-TYPE.&Type ({CertTypes}{@certId})
}
```

The fields have the following meanings:

- certId identifies the type of certificate.
- certValue contains the certificate.

## 7. Security Mechanisms

Let the sender and receiver have a previously agreed-upon password P. The sender generates a password key using the PBKDF2 algorithm in accordance with [\[RFC9337\]](#) and uses it to encrypt the transmitted private key. The recipient independently generates a password key using the same PBKDF2 diversification algorithm in accordance with [\[RFC9337\]](#) and uses it to extract the private key from the PFX.

The same password P is used to encrypt different sections of the PFX using a different random initializing value S with a length of 8 to 32 bytes, where S and P are the input parameters of the PBKDF2 function. The password **MUST** be encoded as a Unicode UTF-8 string and fed into the PBKDF2 algorithm as a P parameter.

The integrity of the PFX is ensured by using the HMAC\_GOSTR3411\_2012\_512 algorithm in accordance with [RFC7836]. To check the integrity of the PFX with the HMAC\_GOSTR3411\_2012\_512 algorithm, the key for this algorithm is also generated by using the PBKDF2 algorithm in accordance with [RFC9337], with the same value for the P parameter and a different initializing value S with a length of 8 to 32 bytes. The dkLen parameter for the PBKDF2 algorithm is set to 96 bytes. The key for the HMAC\_GOSTR3411\_2012\_512 algorithm must be the last 32 bytes of the 96-byte sequence generated by the PBKDF2 algorithm. The PBKDF2 algorithm parameters S and c are saved in the macData.Salt and macData.iterations fields, respectively. The HMAC\_GOSTR3411\_2012\_512 function is calculated from the content field of the authSafe structure field. The authSafe structure field is a PFX structure field. The value of the calculated checksum is saved in the macData.mac.digest field. The macData.mac.digestAlgorithm.algorithm field contains the following algorithm identifier:

```
id-tc26-gost3411-12-512 ::= =
{
  iso(1) member-body(2) ru(643) rosstandart(7) tc26(1)
  algorithms(1) digest(2) gost3411-12-512(3)
}
```

The macData.mac.digestAlgorithm.parameters field isn't used and should be omitted.

## 8. Security Considerations

The masked keys **SHOULD** be used to ensure that private keys are protected from leaking through side channels when reading and performing operations with keys. Applications **MUST** use unique values for ukm and S in the PBKDF2 algorithm. It is **RECOMMENDED** that parameter S consist of at least 32 octets of pseudorandom data in order to reduce the probability of collisions of keys generated from the same password. The password **MUST** be encoded as a Unicode UTF-8 string and fed into the PBKDF2 algorithm as a P parameter. For more information, see [RFC9337]. Encryption **MUST** use the PBES2 scheme to encrypt private keys. Public keys **MUST** be DER encoded as an octet string in accordance with [RFC9215]. Passwords **SHOULD** be stored in a secure way. For information on security considerations for generating PFX, see [RFC7292].

## 9. IANA Considerations

This document has no IANA actions.

## 10. ASN.1 Modules

```
PKCS-12RU
{
  iso(1) member-body(2) ru(643) rosstandart(7)
  tc26(1) modules(0) pkcs-12ruSyntax(5)
}
DEFINITIONS EXPLICIT TAGS ::=
BEGIN
IMPORTS
  GostR3410-2012-PublicKey
FROM GostR3410-2012-PKISyntax
{
  iso(1) member-body(2) ru(643) rosstandart(7) tc26(1)
  modules(0) gostR3410-2012-PKISyntax(2)
};
END
```

## 11. References

### 11.1. Normative References

- [RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <<https://www.rfc-editor.org/info/rfc2104>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <<https://www.rfc-editor.org/info/rfc5652>>.
- [RFC5958] Turner, S., "Asymmetric Key Packages", RFC 5958, DOI 10.17487/RFC5958, August 2010, <<https://www.rfc-editor.org/info/rfc5958>>.
- [RFC6986] Dolmatov, V., Ed. and A. Degtyarev, "GOST R 34.11-2012: Hash Function", RFC 6986, DOI 10.17487/RFC6986, August 2013, <<https://www.rfc-editor.org/info/rfc6986>>.
- [RFC7292] Moriarty, K., Ed., Nystrom, M., Parkinson, S., Rusch, A., and M. Scott, "PKCS #12: Personal Information Exchange Syntax v1.1", RFC 7292, DOI 10.17487/RFC7292, July 2014, <<https://www.rfc-editor.org/info/rfc7292>>.
- [RFC7836] Smyshlyaev, S., Ed., Alekseev, E., Oshkin, I., Popov, V., Leontiev, S., Podobaev, V., and D. Belyavsky, "Guidelines on the Cryptographic Algorithms to Accompany the Usage of Standards GOST R 34.10-2012 and GOST R 34.11-2012", RFC 7836, DOI 10.17487/RFC7836, March 2016, <<https://www.rfc-editor.org/info/rfc7836>>.

- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC9215] Baryshkov, D., Ed., Nikolaev, V., and A. Chelpanov, "Using GOST R 34.10-2012 and GOST R 34.11-2012 Algorithms with the Internet X.509 Public Key Infrastructure", RFC 9215, DOI 10.17487/RFC9215, March 2022, <<https://www.rfc-editor.org/info/rfc9215>>.
- [RFC9337] Karelina, E., Ed., "Generating Password-Based Keys Using the GOST Algorithms", RFC 9337, DOI 10.17487/RFC9337, December 2022, <<https://www.rfc-editor.org/info/rfc9337>>.
- [X.680] ITU-T, "Information Technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation", ITU-T Recommendation X.680, ISO/IEC 8824-1:2021, February 2021, <<https://www.itu.int/rec/T-REC-X.680>>.
- [X.690] ITU-T, "Information technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", ITU-T Recommendation X.690, ISO/IEC International Standard 8825-1:2021, February 2021, <<https://www.itu.int/rec/T-REC-X.690>>.

## 11.2. Informative References

- [ISO14888-1] ISO/IEC, "Information technology - Security techniques - Digital signatures with appendix - Part 1: General", ISO/IEC 14888-1, April 2008, <<https://www.iso.org/standard/44226.html>>.

## Appendix A. Examples

This section contains examples of using GOST cryptographic algorithms to create a PFX.

### A.1. Test Data

In all examples, the following data is used.

#### A.1.1. Test Certificate

This section contains a test certificate in BASE64 format.

```
MIICLjCCAdugAwIBAgIEAYy6hDAKBggqhQMHAQEDAjA4MQ0wCwYDVQQKEwRUSzI2
MScwJQYDVQQDEx5DQSBUSzI2OibHT1NUIDM0LjEwLTEyIDI1Ni1iaXQwHhcNMDEw
MTAxMDAwMDAwWhcNNDkxMjMxMDAwMDAwWjA7MQ0wCwYDVQQKEwRUSzI2MSowKAYD
VQQDEyFPuK1HSU5BVE9S0iBHT1NUIDM0LjEwLTEyIDUxMi1iaXQwgaAwFwYIKoUD
BwEBAQIwCwYJKoUBwECAQIBA4GEAASBgLSLtlq8KQ4YZVxioU+1LV9QhE7MHR9g
BEh7S1yVNGlqt7+rNG5VFqmrPM74rbUs0lhV8M+zZKprXdk350z8lSW/n2oIUHZx
ikXIH/SSHj4rv3K/Puvz7hYTSZ1/xPdp78nUmjrEa6d5wfX8biEy2z0dgufFvAk
Mw1Ua4gdXqD0o4GHMIGEMGMA1UdIwRcMFqAFKxsDkxEZqJC1uKfCTs1ZvPLpFMq
oTyk0jA4MQ0wCwYDVQQKEwRUSzI2MScwJQYDVQQDEx5DQSBUSzI2OibHT1NUIDM0
LjEwLTEyIDI1Ni1iaXSCBAGMu0EwHQYDVR00BBYEFH4GVwmYDK1rCKhX7nkAWDrJ
16CkMAoGCCqFAwCBAQMCA0EAC16p8dAbpi9Hk+3mgMyI0WIh17Ir1rSp/mB0F7Zz
Mt8XUD1Dwz3Jrrnxexnfmv0A5BdUJ9hCyDgMVAGs/IcEEA==
```

### A.1.2. Test Key

This section contains test key bytes in hexadecimal.

```
F95A5D44C5245F63F2E7DF8E782C1924EADCB8D06C52D91023179786154CBDB1
561B4DF759D69F67EE1FBD5B68800E134BAA12818DA4F3AC75B0E5E6F9256911
```

## A.2. Example of a PFX with a Password-Protected Key and Unencrypted Certificate

In this example, the PKCS8ShroudedKeybag structure is used to store the key, which is placed in the Data structure. The certBag structure is used to store the certificate, which is placed in the Data structure. The following password is used to encrypt the key and provide integrity control: "Пароль для PFX". The password is in hexadecimal:

```
D09FD0B0D180D0BED0BBD18C20D0B4D0BBD18F20504658
```

The key encryption algorithm identifier:

```
1.2.643.7.1.1.5.2.2
```

### A.2.1. PFX in BASE64 Format

```

MIIFKwIBazCCBMQGSqGSIB3DQEHAaCCBLUEggSxMIIErTCCAswGCSqGSIB3DQEH
AaCCAr0EggK5MIIcTtCCArEGCyqGSIB3DQEMCgEdoIICSjCCAkYGCiqGSIB3DQEJ
FgGgggI2BIICmJCCAi4wggHboAMCAQICBAGMuoQwCgYIKoUDBwEBAwIwODENMASG
A1UEChMEVEsyNjEnMCUGA1UEAxMeQ0EgVEsyNjogR09TVCAzNC4xMC0xMiAyNTYt
Yml0MB4XDTAxMDEwMTAwMDAwMFoXDTQ5MTIzMTAwMDAwMFowOZENMASGA1UEChME
VEsyNjEqMCgGA1UEAxMhT1JJR01OQVRPUjogR09TVCAzNC4xMC0xMiA1MTItYml0
MIGgMBcGCCqFAwcBAQEEMAsGCSqFAwcBAgECAQ0BhAAEgYCOi7davCkOGGvcYqFP
tS1fUIROzB0fYARIE0tclTRpare/qzRuVRapqzz0+K21LDpYVfDPs2Sqa13ZN+Ts
/JUlv59qCFB2cYpFyB/0kh4+K79yvz7r8+4WE0EmZf8T3ae/J1Jo6xGunech1/G4
hMts9HYLnxbwJDMNVGuIhV6gzq0BhzCBhDBjBgNVHSMEXDBagBSsbA5MRGaiQpbi
nwk7JWbzy6RTKqE8pDowODENMASGA1UEChMEVEsyNjEnMCUGA1UEAxMeQ0EgVEsy
NjogR09TVCAzNC4xMC0xMiAyNTYtYml0ggQBJLqBMB0GA1UdDgQWBBR+B1cJmAyt
awioV+55AFg6ydegpDAKBggqhQMHAQEDAgNBAapeqfHQG6YvR5Pt5oDMiNFiiIdey
K5a0qf5gdBe2czLff1A9Q8M9ya658Xl53zLzg0QXVcfYQsg4DFQBrPyHBBAxVDAj
BgkqhkiG9w0BCRUxFgQUeVV0+dS25MICJChpmGc/8AoUwE0wLQYJKoZIhvcNAQkU
MSAeHgBwADEAMgBGAHIAaQBLAG4AZABsAHKATgBhAG0AZTCCAdkGCSqGSIB3DQEH
AaCCAc0EggHGMIIbWjCCAb4GCyqGSIB3DQEMCgEC0IIBVzCCAVMwWQYJKoZIhvcN
AQUNMEwwKQYJKoZIhvcNAQUUMMBWECkf4N7NMwugqAgIIADAMBggqhQMHAQEEAgUA
MB8GCSqFAwcBAQUCAjASBBAlmt2WdfajlsAs0mLKglzBIH1DMvEacbbWRNDVSnX
JLWygYrKoipd0jDA/2HEBZ34uFOLNheUqiKpCPoFpbR2GBiVYVTVK9ibiczgaca
EQYzDXtcS0QCZ0xpKWfteAlbdJLC/SqPurPYyKi0MVRUPROhbisFASDT38HDH1Dh
0dL5f6ga4aPWLrWbbgWERF0o0Pyh4DotlPF37AQ0wiEjsbyyRHq3HgbWiaxQRuAh
eqHOn4QVGY92/HFvJ7u3TcnQdLWhTe/lh1RHLNF3RnXtN9if9zC23laDZOiWZpLU
yLrUiTCbHrtn1RppPDmLFNMt9dJ7KKgCk0i7Zm5nhqPChbywX13wcfYxVDAjBgkq
hkiG9w0BCRUxFgQUeVV0+dS25MICJChpmGc/8AoUwE0wLQYJKoZIhvcNAQkUMSAe
HgBwADEAMgBGAHIAaQBLAG4AZABsAHKATgBhAG0AZTBeME4wCgYIKoUDBwEBAgME
QAkBKw4ihn7pSIYTEhu0bcvTPZjI3WgVxCKUVl0sc80G69EKFE0Tn0bGJGSKJ51U
Kk0sXF0a7+VBZf3BcVVQh9UECIVet0+VpuskAgIIAA==

```

### A.2.2. PFX in ASN.1 Format

```

0 1323:SEQUENCE:
4 1: INTEGER: 3
7 1220: SEQUENCE:
11 9: OBJECT IDENTIFIER:data [1.2.840.113549.1.7.1]
22 1205: CONTEXT SPECIFIC (0):
26 1201: OCTET STRING:
30 1197: SEQUENCE:
34 716: SEQUENCE:
38 9: OBJECT IDENTIFIER:data [1.2.840.113549.1.7.1]
49 701: CONTEXT SPECIFIC (0):
53 697: OCTET STRING:
57 693: SEQUENCE:
61 689: SEQUENCE:
65 11: OBJECT IDENTIFIER:pkcs-12-certBag
: [1.2.840.113549.1.12.10.1.3]
78 586: CONTEXT SPECIFIC (0):
82 582: SEQUENCE:
86 10: OBJECT IDENTIFIER:x509Certificate
: [1.2.840.113549.1.9.22.1]
98 566: CONTEXT SPECIFIC (0):
102 562: OCTET STRING:

```

```

106 558: SEQUENCE:
110 475: SEQUENCE:
114 3: CONTEXT SPECIFIC (0):
116 1: INTEGER:2
119 4: INTEGER:26000004
125 10: SEQUENCE:
127 8: OBJECT IDENTIFIER:
: [1.2.643.7.1.1.3.2]
137 56: SEQUENCE:
139 13: SET:
141 11: SEQUENCE:
143 3: OBJECT IDENTIFIER:
: organizationName [2.5.4.10]
148 4: PRINTABLE STRING:'TK26'
154 39: SET:
156 37: SEQUENCE:
158 3: OBJECT IDENTIFIER:commonName
: [2.5.4.3]
163 30: PRINTABLE STRING:
: 'CA TK26: GOST 34.10-12 256-bit'
195 30: SEQUENCE:
197 13: UTC TIME:'010101000000Z'
212 13: UTC TIME:'491231000000Z'
227 59: SEQUENCE:
229 13: SET:
231 11: SEQUENCE:
233 3: OBJECT IDENTIFIER:
: organizationName [2.5.4.10]
238 4: PRINTABLE STRING:'TK26'
244 42: SET:
246 40: SEQUENCE:
248 3: OBJECT IDENTIFIER:commonName
: [2.5.4.3]
253 33: PRINTABLE STRING:
: 'ORIGINATOR:
: GOST 34.10-12 512-bit'
288 160: SEQUENCE:
291 23: SEQUENCE:
293 8: OBJECT IDENTIFIER:
: [1.2.643.7.1.1.1.2]
303 11: SEQUENCE:
305 9: OBJECT IDENTIFIER:
: [1.2.643.7.1.2.1.2.1]
316 132: BIT STRING UnusedBits:0:
320 128: OCTET STRING:
: B48BB75ABC290E18655C62A
: 14FB52D5F50844ECC1D1F60
: 04487B4B5C9534696AB7BFA
: B346E5516A9AB3CCEF8ADB5
: 2C3A5855F0CFB364AA6B5DD
: 937E4ECFC9525BF9F6A0850
: 76718A45C81FF4921E3E2BB
: F72BF3EEBF3EE1613412665
: FF13DDA7BF275268EB11AE9
: DE707D7F1B884CB6CF4760B
: 9F16F024330D546B881D5EA0CE
451 135: CONTEXT SPECIFIC (3):
454 132: SEQUENCE:

```

```

457 99: SEQUENCE:
459 3:   OBJECT IDENTIFIER:
      :   authorityKeyIdentifier
      :   [2.5.29.35]
464 92: OCTET STRING:
466 90: SEQUENCE:
468 20:   CONTEXT SPECIFIC (0):
      :   AC6C0E4C4466A24296E2
      :   9F093B2566F3CBA4532A
490 60:   CONTEXT SPECIFIC (1):
492 58:   CONTEXT SPECIFIC (4):
494 56: SEQUENCE:
496 13:   SET:
498 11:     SEQUENCE:
500 3:       OBJECT IDENTIFIER:
      :       organizationName
      :       [2.5.4.10]
505 4:       PRINTABLE STRING:
      :       'TK26'
511 39:     SET:
513 37:       SEQUENCE:
515 3:         OBJECT IDENTIFIER:
      :         commonName
      :         [2.5.4.3]
520 30:         PRINTABLE STRING:
      :         'CA TK26: GOST '
      :         '34.10-12 256-bit'
552 4:       CONTEXT SPECIFIC (2):
      :       018CBA81
558 29: SEQUENCE:
560 3:   OBJECT IDENTIFIER:
      :   subjectKeyIdentifier
      :   [2.5.29.14]
565 22: OCTET STRING:
567 20: OCTET STRING:
      :   7E065709980CAD6B08A8
      :   57EE7900583AC9D7A0A4
589 10: SEQUENCE:
591 8:   OBJECT IDENTIFIER:
      :   [1.2.643.7.1.1.3.2]
601 65: BIT STRING UnusedBits:0:
      :   0A5EA9F1D01BA62F4793EDE680CC88D1
      :   6221D7B22B96B4A9FE607417B67332DF
      :   17503D43C33DC9AEB9F17979DF32F380
      :   E4175427D842C8380C5401ACFC870410
668 84: SET:
670 35: SEQUENCE:
672 9:   OBJECT IDENTIFIER:localKeyID
      :   [1.2.840.113549.1.9.21]
683 22: SET:
685 20:   OCTET STRING:
      :   795574F9D4B6E4C20224
      :   286998673FF00A14C04D
707 45: SEQUENCE:
709 9:   OBJECT IDENTIFIER:friendlyName
      :   [1.2.840.113549.1.9.20]
720 32: SET:
722 30:   BMP STRING:'p12FriendlyName'

```

```

754 473: SEQUENCE:
758 9: OBJECT IDENTIFIER:data [1.2.840.113549.1.7.1]
769 458: CONTEXT SPECIFIC (0):
773 454: OCTET STRING:
777 450: SEQUENCE:
781 446: SEQUENCE:
785 11: OBJECT IDENTIFIER:
: pkcs-12-pkcs-8ShroudedKeyBag
: [1.2.840.113549.1.12.10.1.2]
798 343: CONTEXT SPECIFIC (0):
802 339: SEQUENCE:
806 89: SEQUENCE:
808 9: OBJECT IDENTIFIER:
: [1.2.840.113549.1.5.13]
819 76: SEQUENCE:
821 41: SEQUENCE:
823 9: OBJECT IDENTIFIER:
: [1.2.840.113549.1.5.12]
834 28: SEQUENCE:
836 8: OCTET STRING:'A7F837B34CC2E82A'
846 2: INTEGER:2048
850 12: SEQUENCE:
852 8: OBJECT IDENTIFIER:
: [1.2.643.7.1.1.4.2]
862 0: NULL:
864 31: SEQUENCE:
866 9: OBJECT IDENTIFIER:
: [1.2.643.7.1.1.5.2.2]
877 18: SEQUENCE:
879 16: OCTET STRING:
: 259ADD960DF68F265B00B3498B2A0973
897 245: OCTET STRING:
: 0CCBC469C6DB5913435529D724B5B281
: 8ACAA22A5D3A30C0FF61C49C1677E2E1
: 4E2CD85E52A88AA423E81696D1D86062
: 55855354AF626E273381A71A1106330D
: 7B5C4B440264EC692967ED78095B7492
: C2FD2A8FBAB3D8C8A8B43154543D13A1
: 6E2B050120D3DFC1C31F50E1D1D2F97F
: A81AE1A3D62EB59B6E05844453A838FC
: A1E03A2D94F177EC040EC22123B1BCB2
: 447AB71E06D689AC5046E0217AA1CE9F
: 8415198F76FC716F27BBB74DC9D074B5
: A14DEFE58754472CD1774675ED37D89F
: F730B6DE568364E896669954C8BAD489
: 309B1EBB67D51A693C398B14D32DF5D2
: 7B28A80290E8BB666E6786A3C285BCB0
: 5F5DF071F6
1145 84: SET:
1147 35: SEQUENCE:
1149 9: OBJECT IDENTIFIER:localKeyID
: [1.2.840.113549.1.9.21]
1160 22: SET:
1162 20: OCTET STRING:
: 795574F9D4B6E4C20224
: 286998673FF00A14C04D
1184 45: SEQUENCE:
1186 9: OBJECT IDENTIFIER:friendlyName

```

```

      : [1.2.840.113549.1.9.20]
1197 32: SET:
1199 30: BMP STRING: 'p12FriendlyName'
1231 94: SEQUENCE:
1233 78: SEQUENCE:
1235 10: SEQUENCE:
1237 8: OBJECT IDENTIFIER: [1.2.643.7.1.1.2.3]
1247 64: OCTET STRING:
      : 09012B0E22867EE9488613121BB46DCB
      : D33D98C8DD6815C429145653AC73CD06
      : EBD10A1443939CE6C624648A279D542A
      : 43AC5C5D1AEFE54165FDC171555087D5
1313 8: OCTET STRING: '8544B4EF95A6EB24'
1323 2: INTEGER: 2048

```

### A.2.3. Decrypted Key Value in BASE64 Format

```

MIHiAgEBMBcGCCqFAwcBAQECMAsgCSqFAwcBAgECAQRAEWk1+eblsHws86SngRKq
SxMOgGhbvR/uZ5/WWfdNG1axvUwVhpcXIXDZUmzQuNzqJBkseI7f5/JjXyTFRF1a
+YBGgQG0i7davCk0GGVcYqFPtS1fUIROzB0fYARIE0tclTRpare/qzRuVRapqzz0
+K21LDpYVfDPs2Sqa13ZN+Ts/JUlv59qCFB2cYpFyB/0kh4+K79yvz7r8+4WE0Em
Zf8T3ae/J1Jo6xGunecH1/G4hMts9HYLnxbwJDMNVGuIHV6gzg==

```

### A.2.4. Decrypted Key Value in ASN.1 Format

```

0 226:SEQUENCE:
3 1: INTEGER: 1
6 23: SEQUENCE:
8 8: OBJECT IDENTIFIER: [1.2.643.7.1.1.1.2]
18 11: SEQUENCE:
20 9: OBJECT IDENTIFIER: [1.2.643.7.1.2.1.2.1]
31 64: OCTET STRING:
      : 116925F9E6E5B075ACF3A48D8112AA4B130E80685BBD1FEE679FD6
      : 59F74D1B56B1BD4C158697172310D9526CD0B8DCEA24192C788EDF
      : E7F2635F24C5445D5AF9
97 129: CONTEXT SPECIFIC (1):
      : 01B48BB75ABC290E18655C62A14FB52D5F50844ECC1D1F6004487B
      : 4B5C9534696AB7BFAB346E5516A9AB3CCEF8ADB52C3A5855F0CFB3
      : 64AA6B5DD937E4ECFC9525BF9F6A085076718A45C81FF4921E3E2B
      : BF72BF3EEBF3EE1613412665FF13DDA7BF275268EB11AE9DE707D7
      : F1B884CB6CF4760B9F16F024330D546B881D5EA0CE

```

## A.3. Example of a PFX with a Password-Protected Key and a Password-Protected Certificate

In this example, the PKCS8SHroudedKeybag structure is used to store the key, which is placed in the Data structure (see [RFC5652]). The certBag structure is used to store the certificate, which is placed in the EncryptedData structure (see [RFC5652]). The following password is used to encrypt the key and provide integrity control. The password is in hexadecimal.

```
D09FD0B0D180D0BED0BBD18C20D0B4D0BBD18F20504658
```

The key encryption algorithm identifier:

```
1.2.643.7.1.1.5.1.1
```

The certificate encryption algorithm identifier:

```
1.2.643.7.1.1.5.1.2
```

### A.3.1. PFX in BASE64 Format

```
MIIFjAIBAzCCBSUGCSqGSIB3DQEHAaCCBRYEggUSMIIFDjCCA0EGCSqGSIB3DQEH
BqCCAziWggMuAgEAMIIDJwYJKoZIHvcNAQcBMFUGCSqGSIB3DQEFDTBIMCKGCSqG
SIb3DQEFDDAcBAguUsvGSwGjQICCAAwDAYIKoUDBwEBBAIFADAbBgkqhQMHAQEF
AQIwDgQM9HK3dagtS48+G/x+gIICwWGPqxxN+sTrKbruRf9R5Ya9cf5At01frqMn
f1eULfmZmTg/BdE51QQ+Vbnh3v1kmspr6h2+e4Wli+ndEeCWG6A6X/G22h/RAHW2
YrVmf6cCWxW+YrqzT4h/8RQL/9haunD5LmHPLVsYrEai0wbGxayDSwARVJQLQYq
sLNmZK5ViN+frIS5wszVJ3AtVq8EuPt41aQEKwPy2gmH4S6WmnQRC6W7aoqmIifF
PJENJNn5K2M1J6zNess6bFtYnkMArNqtvv3rioY6eAaaLy6AV6ljsekmqodHmQjv
Y4eEioJs0xhpXhZY69PXT+ZBeHv6MSheBhwXqxAd1DqtPTafmJnk8rqKCap9TtPG
vONvo5W9dgwegxRRQz1um8dzV4m1W9Aq4W7t8/UcxDWRz3k6ijFPLGaA9+8ZMTE0
RHhBRvM60Y2/VNNxbgxWfGYuPxpSi3YnCZIPmBEe51U/Xv7KjzFusGM38F8YR61k
4/QNpKI1QUv714YKfaUQznshGGzILv1NGID62p11+JI3vuawi2mDmrmkuM9QFU9v
/kRp+c2uBHdu0GEUUSNhF08p7+w3vxp1atGWXh9fmIsPBdk2f3wkn+rwoqrEuijM
I/bcAy1U/M0DMKha09j31UYSZdi4fsfRWYDJMq/8FPn96tuo+oCpbqv3NUwpZM/8
Li4xqgTHtYw/+fRG0/P6XadNEiII/TYjenLfvHXjAH0VJsVeCu/t3EsMYHQddNCh
rFk/Ic2PdIQ0yB4/enpW0qrKegSbyZNUf1WI4z14mI89L8dTQBukhy45yQXZ1DD8
k1ErYtdtEsPtz/4zuSpbnmwCEIRo0uSxtGuJP+tbcWEXRKM2UBgi3qBjpn7DU18M
tsrRM9pDdad18mT/Vfh9+B8dZBZVxgQu701MPEGexbUkYHuFCCny9J0V92StbIz
Elx1a1VebjCCAcUGCSqGSIB3DQEHAaCCAbYEggGyMIIBrjCCAaoGCyqGSIB3DQEM
CGCoIIBQzCCAT8wVQYJKoZIHvcNAQUNMEgwKQYJKoZIHvcNAQUMMBwECP0EQk00
1twvAgIIADAMBggqhQMHAQEEAgUAMBSGCSqFAwCBAQUBATA0BAzwxSggAAAAA
AAAEgeUqj9mI3RdfK5hMd0EeYws7foZK/5ANr2wUhp5qnDjAZgn76lExJ+wuvlnS
9PChfWVugvd1/9XJgQvvr9Cu4p0h4ICXp1chcy0dGk/MzItHRVC5wK2nTxwQ4kKT
kG9xhLFzoD16dhtqX0+/dQg9G8pE5EzCBIYRXLm1Arcz9k7KVSTJuNmjFr7EQuu
Tr80ATSQ0tsq50zpfYrpznVPGCrOdIjpyMZxNdvw48bZxqTtRVDxYATOGqz0pwH
ClWULHD9LIajLMB2GhBkyQw6ujI1ltJs0T+WNdx/AT2FLi1LFSS3+Cj9MVQwIwYJ
KoZIHvcNAQKVMRYEFH1VdPnUtutCAiQoaZhnP/AKFMBNMC0GCSqGSIB3DQEFJDEg
Hh4AcAAxADIARgByAGkAZQBAGQAbAB5AE4AYQBtAGUwXjBOMAoGCCqFAwCBAQID
BEDp4e22JmXdnvR0xA99yQuzQuJ8pxBe0psLm2dZQqt3Fje5zqW1uk/7V0cfv5r2
bK8nsL0s2rPT8hB0oeAZvOIBAjGIUhw6IjG2QICCAA=
```

### A.3.2. PFX in ASN.1 Format

```
0 1420: SEQUENCE :
  4   1:  INTEGER:3
  7 1317: SEQUENCE :
  11   9:  OBJECT IDENTIFIER:data [1.2.840.113549.1.7.1]
  22 1302:  CONTEXT SPECIFIC (0):
```

```

26 1298:   OCTET STRING:
30 1294:   SEQUENCE:
34 833:   SEQUENCE:
38 9:     OBJECT IDENTIFIER:
      :     encryptedData [1.2.840.113549.1.7.6]
49 818:   CONTEXT SPECIFIC (0):
53 814:   SEQUENCE:
57 1:     INTEGER:0
60 807:   SEQUENCE:
64 9:     OBJECT IDENTIFIER:data [1.2.840.113549.1.7.1]
75 85:     SEQUENCE:
77 9:     OBJECT IDENTIFIER:[1.2.840.113549.1.5.13]
88 72:     SEQUENCE:
90 41:     SEQUENCE:
92 9:     OBJECT IDENTIFIER:[1.2.840.113549.1.5.12]
103 28:    SEQUENCE:
105 8:    OCTET STRING:'14B92546B12C068D'
115 2:    INTEGER:2048
119 12:    SEQUENCE:
121 8:    OBJECT IDENTIFIER:[1.2.643.7.1.1.4.2]
131 0:    NULL:
133 27:    SEQUENCE:
135 9:    OBJECT IDENTIFIER:[1.2.643.7.1.1.5.1.2]
146 14:    SEQUENCE:
148 12:    OCTET STRING:
      :     F4793775A82D4B8F3E1BFC7E
162 705:  CONTEXT SPECIFIC (0):
      :     618FAB1C4DFAC4EB29BAEE45FF51E586BD7
      :     1FE40B4ED5FAEA3277F57942DF99999383F
      :     05D139D5043E55B9E1DEFD649ACA6BEA1DB
      :     E7B85A58BE9DD11E0961BA03A5FF1B6DA1F
      :     D10075B662B5667FA7025B15BE62BAB34F8
      :     87FF1140BFFD85ABA70F92E61CF2D5B18AC
      :     46A2D0EC1B8176B20D2C004552502D062AB
      :     0B36664AE5588DF9F4624B9C2CCD527702D
      :     56AF04B8FB78D5A4042B03F2DA0987E12E9
      :     69A74110BA5BB6A8AA62227C53C910D24D9
      :     F92B633527ACCD112B3A6C5B5834A300ACD
      :     AADBEFDEB8A863A78069A2F2E8057A963B1
      :     E926AA87479908EF6387848A826CD318695
      :     E1658EBD3D74FE641787BFA31285E061C17
      :     AB101DD43AAD3D369F32334AF2BA8A09AA7
      :     D4ED3C6BCE36FA395BD760C1E8314514339
      :     6E9BC7735789B55BD02AE16EEDF3F51CC43
      :     591CF793A8A314F946680F7EF1931310E44
      :     784146F33A398DBF54D3716E0C567C662E3
      :     F1A528B762709920F98111EE6553F5EFECA
      :     8F316EB06337F05F1847AD64E3F40DA4A23
      :     5414BFBD7860A7DA510CE7B21186CC82EFD
      :     4D1880FADA9975F89237BEE6B08B698332B
      :     9A4B8CF50154F6FFE444FF9CDAE0470EE38
      :     6114512361174F29EFEC37BF1A656AD1965
      :     C7F5F988B0F05D9367F7C249FEAF0A2AAC4
      :     BA28CC23F6C2032954FCCD0330A840A3D8F
      :     7D5461265D8B87EC7D15980C932AFFC14F9
      :     FDEADBA8FA80A96EABF7354C2964CFFC2E2
      :     E31AA04C7B58C3FF9F446D3F3FA5DA74D12
      :     2208FD36237A72DF5475E300739526C55E0

```

```

      : AEFEDDC4B0C60741D74D0A1AC593F21CD8F
      : 74840EC81E3F7A7A56D2AACA7A049BC9936
      : E175588E33978988F3D2FC753401524872E
      : 39C905D99430FC93512B61DB5D12C3EDCFF
      : E33B92A5B9E6C021084683AE497B46B893F
      : EB5B71611744A336501822DEA063A67EC35
      : 35F0CB6CAD133DA4375A765F264FF55F87D
      : F81F1D641655C6042EEF494C3C419EC5B52
      : 4607B850829F28BD27457DD92B5B233125C
      : 656B555E6E
871 453: SEQUENCE:
875 9: OBJECT IDENTIFIER:data [1.2.840.113549.1.7.1]
886 438: CONTEXT SPECIFIC (0):
890 434: OCTET STRING:
894 430: SEQUENCE:
898 426: SEQUENCE:
902 11: OBJECT IDENTIFIER:
      : pkcs-12-pkcs-8ShroudedKeyBag
      : [1.2.840.113549.1.12.10.1.2]
915 323: CONTEXT SPECIFIC (0):
919 319: SEQUENCE:
923 85: SEQUENCE:
925 9: OBJECT IDENTIFIER:
      : [1.2.840.113549.1.5.13]
936 72: SEQUENCE:
938 41: SEQUENCE:
940 9: OBJECT IDENTIFIER:
      : [1.2.840.113549.1.5.12]
951 28: SEQUENCE:
953 8: OCTET STRING:
      : FD04424D0ED6DC2F
963 2: INTEGER:2048
967 12: SEQUENCE:
969 8: OBJECT IDENTIFIER:
      : [1.2.643.7.1.1.4.2]
979 0: NULL:
981 27: SEQUENCE:
983 9: OBJECT IDENTIFIER:
      : [1.2.643.7.1.1.5.1.1]
994 14: SEQUENCE:
996 12: OCTET STRING:
      : F0C52AA0000000000000000000000000
1010 229: OCTET STRING:
      : 2A8FD988DD10DF2B984C77411E630B3B
      : 7E864AFF900DAF6C1484FE6A9C38C066
      : 09FBEA513127EC2EBE59D2F4F0A17D65
      : 6E82F765FFD5C9810BEFAFD0AEE293A1
      : E08097A65721732D1D1A4FCCCC8B4745
      : 50B9C0ADA74F1C10E24293906F7184B1
      : 73A03D7A761B6A5F4FBF75083D1BCA44
      : E44CC20486115CB9B502B733F64ECA56
      : C4C9B8D32316BAFB110BAE4EBF340134
      : 903ADB2AE74CE9172AE9CE754F182ACE
      : 7488E9CA667135DBF0E3C6D9C6A4ED45
      : 50F1098013386AB3D29C070A55942C70
      : FD2C86A32CC0761A104AC90C3ABA3225
      : 96D26CD13F9635D5FF013D852E2D4B15
      : 24B7F828FD

```

```

1242 84:          SET:
1244 35:          SEQUENCE:
1246  9:          OBJECT IDENTIFIER:localKeyID
      :          [1.2.840.113549.1.9.21]
1257 22:          SET:
1259 20:          OCTET STRING:
      :          795574F9D4B6E4C20224
      :          286998673FF00A14C04D
1281 45:          SEQUENCE:
1283  9:          OBJECT IDENTIFIER:
      :          friendlyName [1.2.840.113549.1.9.20]
1294 32:          SET:
1296 30:          BMP STRING:'p12FriendlyName'
1328 94: SEQUENCE:
1330 78: SEQUENCE:
1332 10: SEQUENCE:
1334  8:   OBJECT IDENTIFIER:[1.2.643.7.1.1.2.3]
1344 64:   OCTET STRING:
      :   E9E1EDB62665DD9EF474C40F7DC90BB3
      :   42E27CA7105E3A9B0B9B675942AB7716
      :   37B9CEA5B5BA4FFB54E71F579AF66CA9
      :   BC9EC2CEB36ACF4FC8413A878066F388
1410  8: OCTET STRING:'C62141F0E888C6D9'
1420  2: INTEGER:2048

```

### A.3.3. Decrypted Key Value in BASE64 Format

```

MIHiAgEBMBcGCCqFAwcBAQECMAsgCSqFAwcBAgECAQRAEWk1+eblsHWs86SNgRKq
SxM0gGhbvR/uZ5/WWfdNG1axvUwVhpcXIxDZUmzQuNzqJBkseI7f5/JjXyTFRF1a
+YGBgQG0i7davCk0GGVcYqFPtS1fUIROzB0fYARIE0tclTRpare/qzRuVRapqzz0
+K21LDpYVfDPs2Sqa13ZN+Ts/JUlv59qCFB2cYpFyB/0kh4+K79yvz7r8+4WE0Em
Zf8T3ae/J1Jo6xGunecH1/G4hMts9HYLnxbwJDMNVGuIHV6gzg==

```

### A.3.4. Decrypted Key Value in ASN.1 Format

```

0 226:SEQUENCE:
3  1:  INTEGER: 1
6 23:  SEQUENCE:
8  8:  OBJECT IDENTIFIER: [1.2.643.7.1.1.1.2]
18 11: SEQUENCE:
20  9:  OBJECT IDENTIFIER: [1.2.643.7.1.2.1.2.1]
31 64:  OCTET STRING:
      :  116925F9E6E5B075ACF3A48D8112AA4B130E80685BBD1FEE679FD6
      :  59F74D1B56B1BD4C158697172310D9526CD0B8DCEA24192C788EDF
      :  E7F2635F24C5445D5AF9
97 129: CONTEXT SPECIFIC (1):
      :  01B48BB75ABC290E18655C62A14FB52D5F50844ECC1D1F6004487B
      :  4B5C9534696AB7BFAB346E5516A9AB3CCEF8ADB52C3A5855F0CFB3
      :  64AA6B5DD937E4ECFC9525BF9F6A085076718A45C81FF4921E3E2B
      :  BF72BF3EEBF3EE1613412665FF13DDA7BF275268EB11AE9DE707D7
      :  F1B884CB6CF4760B9F16F024330D546B881D5EA0CE

```

## Acknowledgments

The author thanks Alexander Potashnikov, Semen Pianov, and Valery Smyslov for their careful readings and useful comments, and Alexander Chelpanov for his help with the registration of identifiers.

## Author's Address

**Ekaterina Karelina (EDITOR)**

InfoTeCS

2B stroenie 1, ul. Otradnaya

Moscow

127273

Russian Federation

Email: [Ekaterina.Karelina@infotecs.ru](mailto:Ekaterina.Karelina@infotecs.ru)