Network Working Group
Request for Comments: 5408
Category: Informational

G. Appenzeller
Stanford University
L. Martin
Voltage Security
M. Schertler
Axway

January 2009

Identity-Based Encryption Architecture and Supporting Data Structures

Status of This Memo

This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Copyright Notice

Abstract

This document describes the security architecture required to implement identity-based encryption, a public-key encryption technology that uses a user's identity as a public key. It also defines data structures that can be used to implement the technology.

Table of Contents

1. Introduction

This document describes the security architecture required to implement identity-based encryption, a public-key encryption technology that uses a user's identity as a public key. It also defines data structures that are required to implement the technology. Objects used in this implementation are defined using ASN.1 [ASN1] and XML [XML].

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [KEY].

2. Identity-Based Encryption

2.1. Overview

Identity-based encryption (IBE) is a public-key encryption technology that allows a public key to be calculated from an identity and a set of public mathematical parameters and that allows for the corresponding private key to be calculated from an identity, a set of public mathematical parameters, and a domain-wide secret value. An IBE public key can be calculated by anyone who has the necessary public parameters; a cryptographic secret is needed to calculate an IBE private key, and the calculation can only be performed by a trusted server that has this secret.

Calculation of both the public and private keys in an IBE system can occur as needed, resulting in just-in-time creation of both public and private keys. This contrasts with other public-key systems [P1363], in which keys are generated randomly and distributed prior to secure communication commencing, and in which private encryption keys need to be securely archived to allow for their recovery if they are lost or destroyed. The ability to calculate a recipient's public key, in particular, eliminates the need for the sender and receiver to interact with each other, either directly or through a proxy such as a directory server, before sending secure messages.

A characteristic of IBE systems that differentiates them from other server-based cryptographic systems is that once a set of public parameters is fetched, encryption is possible with no further communication with a server during the validity period of the public parameters. Other server-based systems may require a connection to a server for each encryption operation.

This document describes an IBE-based messaging system, how the components of such a system work together, and defines data structures that support the operation of such a system. The server components required for such a system are the following:

- A Public Parameter Server (PPS). IBE public parameters include publicly-sharable cryptographic material, known as IBE public parameters, and policy information for an associated PKG. A PPS provides a well-known location for secure distribution of IBE public parameters and policy information that describe the operation of a PKG. Section 5 of this document describes the protocol that a client uses to communicate with a PPS.

- A Private-key Generator (PKG). The PKG stores and uses cryptographic material, known as a master secret, which is used for generating a user's IBE private key. A PKG accepts an IBE user's private key request, and after successfully authenticating them in some way, returns their IBE private key. Section 5 of this document describes the protocol that a client uses to communicate with a PKG.

A logical architecture of such an IBE system would be to have a PKG and PPS per name space, such as a DNS zone. The organization that controls the DNS zone would also control the PKG and PPS and thus the determination of which PKG or PPS to use when creating public and private keys for the organization's members. In this case, the PPS URI/IRI can be uniquely created from a user-friendly name for the form of identity that the PPS supports. This architecture would make it clear which set of public parameters to use and where to retrieve them for a given identity (for example, an RFC 2821 address [SMTP]).

IBE-encrypted messages can use standard message formats, such as the Cryptographic Message Syntax [CMS]. How to use IBE with the CMS to encrypt email messages is defined in [IBECMS].

Note that IBE algorithms are used only for encryption, so if digital signatures are required, they will need to be provided by an additional mechanism.

Section 3 of this document describes the identity format that all PPS and PKG servers MUST support.

2.2. Sending a Message That Is IBE-Encrypted

In order to send an encrypted message, an IBE user must perform the following steps:

1. Obtain the recipient's public parameters

   The public parameters of the recipient's system are needed to perform IBE operations. Once a user obtains these public parameters, he can perform IBE encryption operations. These public parameters may be available at a PPS that is operated by the user's organization, one that is operated by the sender's organization, or by a different organization entirely.

2. Construct and send an IBE-encrypted message

   In addition to the IBE public parameters, all that is needed to construct an IBE-encrypted message is the recipient's identity, the form of which is defined by the public parameters. When this identity is the same as the identity that a message would be addressed to, then no more information is needed from a user to send them an encrypted message than is needed to send them an unencrypted message. This is one of the major benefits of an IBE-based secure messaging system. Examples of identities are individual, group, or role identifiers.

2.2.1. Sender Obtains Public Parameters

The sender of a message obtains the IBE public parameters that he needs from a PPS that is hosted at a well-known URI or IRI. The IBE public parameters contain all of the information that the sender needs to create an IBE-encrypted message except for the identity of the recipient. Section 4 of this document describes the URI [URI] or IRI [IRI] of a PPS, the format of IBE public parameters, and how to obtain them from a PPS. The URI or IRI from which users obtain IBE public parameters MUST be authenticated in some way. PPS servers MUST support TLS 1.2 [TLS] to satisfy this requirement and SHOULD support its successors. This step is shown below in Figure 1.

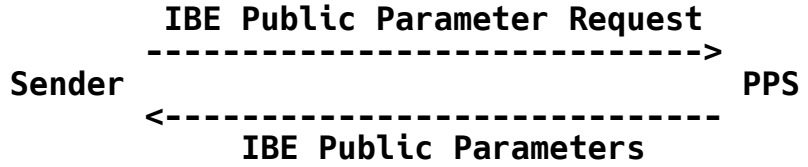

Figure 1: Requesting IBE Public Parameters

The sender of an IBE-encrypted message selects the PPS and corresponding PKG based on his local security policy. Different PPS servers may provide public parameters that specify different IBE algorithms or different key strengths, for example. Or, they may require the use of PKG servers that require different levels of authentication before granting IBE private keys.

2.2.2. Construct and Send an IBE-Encrypted Message

To IBE-encrypt a message, the sender chooses a content-encryption key (CEK) and uses it to encrypt his message and then encrypts the CEK with the recipient's IBE public key as described in [CMS]. This operation is shown below in Figure 2. The document [IBCS] describes the algorithms needed to implement two forms of IBE, and [IBECMS] describes how to use the Cryptographic Message Syntax (CMS) to encapsulate the encrypted message along with the IBE information that the recipient needs to decrypt an email message.

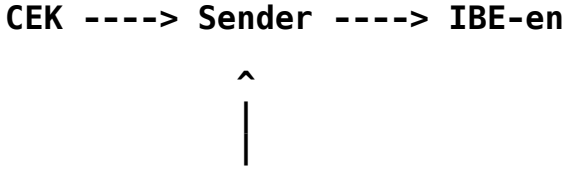

Figure 2: Using an IBE Public-key Algorithm to Encrypt

2.3. Receiving and Viewing an IBE-Encrypted Message

In order to read an IBE-encrypted message, a recipient of such a message parses it to find the URI or IRI he needs in order to obtain the IBE public parameters that are required to perform IBE calculations as well as to obtain a component of the identity that was used to encrypt the message. Next, the recipient carries out the following steps:

1. Obtain the IBE public parameters

   An IBE system's public parameters allow it to uniquely create public and private keys. The recipient of an IBE-encrypted message can decrypt an IBE-encrypted message if he has both the IBE public parameters and the necessary IBE private key. The public parameters also provide the URI or IRI of the PKG where the recipient of an IBE-encrypted message can obtain the IBE private keys.

2. Obtain the IBE private key from the PKG

   To decrypt an IBE-encrypted message, in addition to the IBE public parameters, the recipient needs to obtain the private key that corresponds to the public key that the sender used. The IBE private key is obtained after successfully authenticating to a private key generator (PKG), a trusted third party that calculates private keys for users. The recipient then receives the IBE private key over a secure connection.

3. Decrypt the IBE-encrypted Message

   The IBE private key decrypts the CEK (see Section 2.3.3). The CEK is then used to decrypt the encrypted message.

It may be useful for a PKG to allow users other than the intended recipient to receive some IBE private keys. Giving a mail-filtering appliance permission to obtain IBE private keys on behalf of users, for example, can allow the appliance to decrypt and scan encrypted messages for viruses or other malicious features.

2.3.1. Recipient Obtains Public Parameters

Before he can perform any IBE calculations related to the message that he has received, the recipient of an IBE-encrypted message needs to obtain the IBE public parameters that were used in the encryption operation. This operation is shown below in Figure 3. Because the use of the correct public parameters is vital to the overall security of an IBE system, IBE public parameters MUST be transported to recipients over a secure protocol. PPS servers MUST support TLS 1.2 [TLS] or its successors, using the latest version supported by both parties, for transport of IBE public parameters. In addition, users MUST verify that the subject name in the server certificate matches the URI/IRI of the PPS. The comments in Section 2.2.1 also apply to this operation.

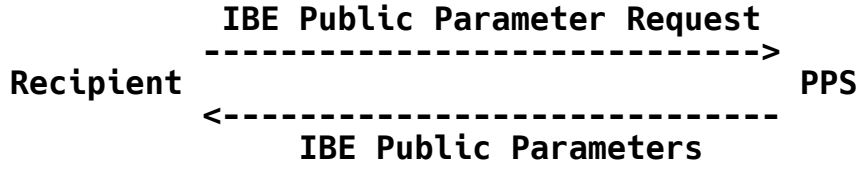

Figure 3: Requesting IBE Public Parameters

2.3.2. Recipient Obtains IBE Private Key

To obtain an IBE private key, the recipient of an IBE-encrypted message provides the IBE public key used to encrypt the message and their authentication credentials to a PKG and requests the private key that corresponds to the IBE public key. Section 5 of this document defines the protocol for communicating with a PKG as well as a minimum interoperable way to authenticate to a PKG that all IBE implementations MUST support. Because the security of IBE private keys is vital to the overall security of an IBE system, IBE private keys MUST be transported to recipients over a secure protocol. PKG servers MUST support TLS 1.2 [TLS] or its successors, using the latest version supported by both parties, for transport of IBE private keys. This operation is shown below in Figure 4.

Figure 4: Obtaining an IBE Private Key

2.3.3. Recipient Decrypts IBE-Encrypted Message

After obtaining the necessary IBE private key, the recipient uses that IBE private key and the corresponding IBE public parameters to decrypt the CEK. This operation is shown below in Figure 5. He then uses the CEK to decrypt the encrypted message content. An example of how to do this with email messages is given in [IBECMS].

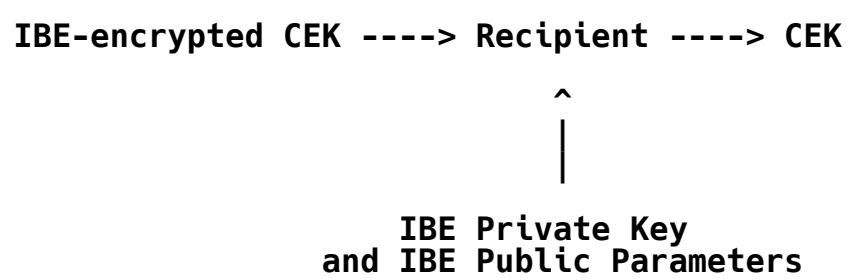

Figure 5: Using an IBE Public-Key Algorithm to Decrypt

3. Identity Format

An IBEIdentityInfo type MUST be used to represent the identity of a recipient. This is defined to be the following:

```
IBEIdentityInfo ::= SEQUENCE {
  district        IA5String,
  serial          INTEGER,
  identityType    OBJECT IDENTIFIER,
  identityData    OCTET STRING
}
```

An IBEIdentityInfo type is used to calculate public and private IBE keys. Because of this, such a structure is typically DER-encoded [DER].

The fields of an IBEIdentityInfo structure have the following meanings.

The district is an IA5String that represents the URI [URI] or IRI [IRI] where the recipient of an IBE-encrypted message can retrieve the IBE public parameters needed to encrypt or decrypt a message encrypted for this identity. Applications MUST support the method described in Section 4 for doing this and MAY support other methods. IRIs MUST be handled according to the procedures specified in Section 7.4 of [PKIX].

The serial is an INTEGER that defines a unique set of IBE public parameters in the event that more than one set of parameters is used by a single district.

identityType is an OBJECT IDENTIFIER that defines the format that the identityData field is encoded with.

An example of a useful IBEIdentityInfo type is given in [IBECMS]. This example is tailored to the use of IBE in encrypting email. Because the information that comprises an identity is very dependent on the application, this document does not define an identityType that all applications MUST support.

4. Public Parameter Lookup

This section specifies how a component of an IBE system can retrieve the public parameters. A sending or receiving client MUST allow configuration of these parameters manually, for example, through editing a configuration file. However, for simplified configuration, a client SHOULD also implement the public parameter URI/IRI request method described in this document to fetch the public parameters

based on a configured URI/IRI. This is especially useful for federating between IBE systems. By specifying a single URI/IRI, a client can be configured to fetch all the relevant parameters for a remote PKG. These public parameters can then be used to encrypt messages to recipients who authenticate to and retrieve private keys from that PKG.

The following section outlines the URI/IRI request method to retrieve a parameter block and describes the structure of the parameter block itself. The technique for fetching IBE public parameters using the process defined in this section is indicated by the OID uriPPSOID, which is defined to be the following:

```
uriPPSOID OBJECT IDENTIFIER ::= {
  joint-iso-itu-t(2) country(16) us(840)
  organization(1) identicrypt(114334)
  pps-schemas(3) ic-schemas(1) pps-uri(1) version(1)
}
```

4.1. Request Method

The configuration URI/IRI SHOULD be an HTTPS URI [HTTP] and MAY additionally support IRIs [IRI] for this purpose. To retrieve the IBE public parameters, the client SHOULD use the HTTP GET method as defined in [HTTP]. The request MUST happen over a secure protocol. The requesting client MUST support TLS 1.2 [TLS] or its successors and SHOULD use the latest version supported by both parties. When requesting the URI/IRI, the client MUST only accept the system parameter block if the server identity was verified successfully by TLS 1.2 [TLS] or its successors.

A successful GET request returns in its body the base64 [B64] encoding of the DER-encoded [DER] IBESysParams structure that is described in the next section. This structure MUST be encoded as an application/ibe-pp-data MIME type.

4.2. Parameter and Policy Format

The IBE public parameters are a structure of the form

```
IBESysParams ::= SEQUENCE {
  version              INTEGER { v2(2) },
  districtName         IA5String,
  districtSerial       INTEGER,
  validity             ValidityPeriod,
  ibePublicParameters  IBEPublicParameters,
  ibeIdentityType      OBJECT IDENTIFIER,
  ibeParamExtensions   IBEParamExtensions OPTIONAL
}
```

The fields of an IBESysParams structure have the following meanings.

The version field specifies the version of the IBESysParams format. For the format described in this document, this MUST be set to 2.

The districtName field is an IA5String that MUST be an encoding of an URI [URI] or IRI [IRI]. IRIs MUST be handled according to the procedures specified in Section 7.4 of [PKIX].

The districtSerial field is an integer that represents a unique set of IBE public parameters that are available at the URI or IRI defined by the districtName. If new parameters are published for a districtName, the districtSerial MUST be increased to a number greater than the previously-used districtSerial.

The validity field defines the lifetime of a specific instance of the IBESysParams and is defined to be the following:

```
ValidityPeriod ::= SEQUENCE {
  notBefore     GeneralizedTime,
  notAfter      GeneralizedTime
}
```

The values of notBefore and notAfter MUST be expressed in Greenwich Mean Time (Zulu), MUST include seconds (i.e., times are always YYYYMMDDHHMMSSZ), even where the number of seconds is equal to zero, and MUST be expressed to the nearest second.

A client MUST verify that the date on which it uses the IBE public parameters falls between the notBefore time and the notAfter time of the IBE public parameters, and it MUST NOT use the parameters for IBE encryption operations if they do not.

IBE public parameters MUST be regenerated and republished whenever the values of ibePublicParameters, ibeIdentityType, or ibeParamExtensions change for a district. A client SHOULD refetch the IBE public parameters at an application-configurable interval to ensure that it has the most current version of the IBE public parameters.

It is possible to create identities for use in IBE that have a time component, as described in [IBECMS], for example. If such an identity is used, the time component of the identity MUST fall between the notBefore time and the notAfter times of the IBE public parameters.

IBEPublicParameters is a structure containing public parameters that correspond to IBE algorithms that the PKG supports. This structure is defined to be following:

```
IBEPublicParameters ::= SEQUENCE (1..MAX) OF
  IBEPublicParameter

IBEPublicParameter  ::= SEQUENCE {
  ibeAlgorithm          OBJECT IDENTIFIER,
  publicParameterData   OCTET STRING
}
```

The ibeAlgorithm OID specifies an IBE algorithm. The OIDs for two IBE algorithms (the Boneh-Franklin and Boneh-Boyen algorithms) and their publicParameterData structures are defined in [IBCS].

The publicParameterData is a DER-encoded [DER] structure that contains the actual cryptographic parameters. Its specific structure depends on the algorithm.

The IBESysParams of a district MUST contain an OID that identifies at least one algorithm and MAY contain OIDs that identify more than one algorithm. It MUST NOT contain two or more IBEPublicParameter entries with the same algorithm. A client that wants to use IBESysParams can chose any of the algorithms specified in the publicParameterData structure. A client MUST implement at least the Boneh-Franklin algorithm [IBCS] and MAY implement the Boneh-Boyen [IBCS] and other algorithms. If a client does not support any of the supported algorithms, it MUST generate an error message and fail.

ibeIdentityType is an OID that defines the type of identities that are used with this district. The OIDs and the required and optional fields for each OID are application dependent. An example of this is given in [IBECMS], which defines an identity format suitable for use in encrypting email.

IBEParamExtensions is a set of extensions that can be used to define additional parameters that particular implementations may require. This structure is defined as follows:

```
IBEParamExtensions ::= SEQUENCE OF IBEParamExtension

IBEParamExtension ::= SEQUENCE {
  ibeParamExtensionOID     OBJECT IDENTIFIER,
  ibeParamExtensionValue   OCTET STRING
}
```

The contents of the octet string ibeParamExtensionValue are defined by the specific ibeParamExtensionOID. The IBEParamExtensions of a district MAY have any number of extensions, including zero. One example of extensions that have been used in practice is to provide a URI where an encrypted message can be decrypted and viewed by users of webmail systems. Another example is to provide commercial branding information, so that a bank can provide a different user interface for customers of different lines of business.

If a client receives public parameters that contain an extension that it is unable to process, it MUST NOT use the values in the IBESysParams structure for any cryptographic operations. Clients MUST be able to process an IBESysParams structure that contains no IBEParamExtensions.

The pkgURI OID that is defined below defines an IBEParamExtensions structure that contains the URI or IRI of a Private Key Generator. The presence of this OID in an IBEParamExtension indicates that clients MUST use the protocol defined in Section 5 of this document to obtain IBE private keys from the PKG and MUST do so using the URI/IRI that is defined by the ibeParamExtensionValue in the IBEParamExtension.

If the PKG is publicly-accessible, this extension SHOULD be present to allow the automatic retrieval of private keys for recipients of encrypted messages. For this extension, the ibeParamExtensionValue OCTET STRING is an IA5String containing the URI [URI] or IRI [IRI] of the PKG where IBE private keys can be obtained. IRIs MUST be handled according to the procedures specified in Section 7.4 of [PKIX].

```
ibeParamExt OBJECT IDENTIFIER ::= {
  ibcs ibcs3(3) parameter-extensions(2)
}

pkgURI OBJECT IDENTIFIER ::= { ibeParamExt pkgURI(1) }
```

4.3. The application/ibe-pp-data MIME Type

The following summarizes the properties of the application/ibe-pp-data MIME type.

MIME media type name: application

MIME subtype name: ibe-pp-data

Mandatory parameters: none

Optional parameters: none

Encoding considerations: This media type MUST be encoded as 7-bit (US-ASCII text [ASCII]).

Security considerations: The data conveyed as this media type are the public parameters needed for the operation of a cryptographic system. To ensure that the parameters can be trusted, the request for these parameters must take place over a secure protocol, such as TLS 1.2 or its successors. To ensure the validity of the server, the client MUST verify the server certificate and MUST abort the parameter request if the verification of the server certificate of the TLS connection fails. This media type contains no active content and does not use compression.

Interoperability considerations: There are no known interoperability considerations for this media type.

Applications that use this media type: Applications that implement IBE in compliance with this specification will use this media type. The most commonly used of these applications are encrypted email and file encryption.

Additional information: none

Person and email address for further information: Luther Martin, martin@voltage.com.

Intended usage: COMMON

Author/Change controller: Luther Martin, martin@voltage.com.

5. Private Key Request Protocol

5.1. Overview

When requesting a private key, a client has to transmit three parameters:

1. The IBE algorithm for which the key is being requested
2. The identity for which it is requesting a key
3. Authentication credentials for the individual requesting the key

The identity for which a client requests a key may not necessarily be the same as the identity that the authentication credentials validate. This may happen, for example, when a single user has access to multiple aliases. For example, an email user may have access to the keys that correspond to two different email addresses, e.g., bob@example.com and bob.smith@example.com.

This section defines the protocol to request private keys, a minimum user authentication method for interoperability, and how to pass authentication credentials to the server. It assumes that a client has already determined the URI/IRI of the PKG. This can be done from information included in the IBE message format and the public parameters of the IBE system.

The technique for fetching an IBE private key using the process defined in this section is indicated by the OBJECT IDENTIFIER pkgURI, which is defined to be the following:

```
ibcs OBJECT IDENTIFIER ::= {
  joint-iso-itu-t(2) country(16) us(840)
  organization(1) identicrypt(114334) ibcs(1)
}

ibeParamExt OBJECT IDENTIFIER ::= {
  ibcs ibcs3(3) parameter-extensions(2)
}

pkgURI OBJECT IDENTIFIER ::= { ibeParamExt pkgURI(1) }
```

5.2. Private Key Request

To request a private key, a client MUST perform a HTTP POST method as defined in [HTTP]. The request MUST take place over a secure protocol. The requesting client MUST support TLS 1.2 [TLS] or its

successors, using the latest version supported by both the client and the PKG. When requesting the URI/IRI, the client MUST verify the server certificate [HTTPTLS], and it MUST abort the key request if the server certificate verification of the TLS connection fails. Doing so is critical to protect the authentication credentials and the private key against man-in-the-middle attacks when it is transmitted from the key server to the client.

5.3. Request Structure

The POST method contains in its body the following XML structure that MUST be encoded as an application/ibe-key-request+xml MIME type:

```
<ibe:request xmlns:ibe="urn:ietf:params:xml:ns:ibe">
   <ibe:header>
      <ibe:client version="clientID"/>
   </ibe:header>
   <ibe:body>
   <ibe:keyRequest>
      <ibe:algorithm>
         algorithmOID
      </ibe:algorithm>
      <ibe:id>
         ibeIdentityInfo
      </ibe:id>
   </ibe:keyRequest>
   <ibe:authData>
      ibeAuthData
   </ibe:authData>
   </ibe:body>
</ibe:request>
```

A <ibe:request> SHOULD include an <ibe:client> element in the <ibe:header> part of a key request that contains an ASCII string that identifies the client type and client version.

A key request MUST contain an <ibe:oid> element that contains the base64 [B64] encoding of a DER-encoded [DER] object identifier that identifies the algorithm for which a key is requested. OIDs for the Boneh-Boyen (BB1) and Boneh-Franklin (BF) algorithms are listed in [IBCS].

A key request MUST contain an <ibe:id> element that contains the identity that the private key is being requested for. This identity is the base64 [B64] encoding of a DER-encoded [DER] ASN.1 structure. This structure defines a user's identity in a way appropriate for the application. An example of such a structure that is appropriate for use in encrypting email is defined in [IBECMS].

A key request MAY contain an <ibe:authData> element. This element contains authentication information that the PKG can use to determine whether or not a request for a particular IBE private key should be granted.

A client MAY include optional additional XML elements in the <ibe:body> part of the key request. A PKG MUST be able to process key requests that contain no such optional elements.

5.4. The application/ibe-key-request+xml MIME type

The following summarizes the properties of the application/ibe-key-request+xml MIME type.

MIME media type name: application

MIME subtype name: ibe-key-request+xml

Mandatory parameters: none

Optional parameters: none

Encoding considerations: This media type MUST be encoded as US-ASCII [ASCII].

Security considerations: The data conveyed in this media type may contain authentication credentials. Because of this, its confidentiality and integrity is extremely important. To ensure this, the request for an IBE private key must take place over a secure protocol, such as TLS 1.2 or its successors. To ensure the validity of the server, the client MUST verify the server certificate and MUST abort the key request if the verification of the server certificate of the TLS connection fails. This media type contains no active content and does not use compression.

Interoperability considerations: There are no known interoperability considerations for this media type.

Applications that use this media type: Applications that implement IBE in compliance with this specification will use this media type. The most commonly used of these applications are encrypted email and file encryption.

Additional information: none

Person and email address for further information: Luther Martin, martin@voltage.com.

Intended usage: COMMON

Author/Change controller: Luther Martin, martin@voltage.com.

5.5. Authentication

When a client requests a key from a PKG, the PKG MUST authenticate the client before issuing the key. Authentication may either be done through the key request structure or as part of the secure transport protocol.

A client or server implementing the request protocol MUST support HTTP Basic Auth [AUTH] and SHOULD also support HTTP Digest Auth [AUTH]. Applications MAY also use other means of authentication that are appropriate for the application. An email application, for example, might rely on deployed email infrastructure for this.

For authentication methods that are not done by the transport protocol, a client MAY include additional authentication information in XML elements in the <ibe:authData> element of a key request. If a client does not know how to authenticate to a server, the client MAY send a key request without authentication information. If the key server requires the client to authenticate externally, it MAY reply with an IBE201 responseCode (as defined below) to redirect the client to the correct authentication mechanism. After receiving an authentication credential from this external mechanism, a client can then use the credential to form a key request that contains the additional authentication data.

5.6. Server Response Format

The key server replies to the HTTP request with an HTTP response. If the response contains a client error or server error status code, the client MUST abort the key request and fail.

If the PKG replies with an HTTP response that has a status code indicating success, the body of the reply MUST contain the following XML structure that MUST be encoded as an application/ibe-pkg-reply+xml MIME type:

```
<ibe:response xmlns:ibe="urn:ietf:params:xml:ns:ibe">
   <ibe:responseType value="responseCode"/>
   <ibe:body>
      bodyTags
   </ibe:body>
</ibe:response>
```

The responseCode attribute contains an ASCII string that describes the type of response from the key server. The list of currently-defined responseCodes and their associated meanings is:

```
IBE100 KEY_FOLLOWS
IBE101 RESERVED
IBE201 FOLLOW_ENROLL_URI
IBE300 SYSTEM_ERROR
IBE301 INVALID_REQUEST
IBE303 CLIENT_OBSOLETE
IBE304 AUTHORIZATION DENIED
```

5.6.1. The IBE100 responseCode

If the key request was successful, the key server responds with a responseCode of IBE100, and the <ibe:body> MUST contain a <ibe:privateKey> element that contains a valid private key. An example of this is shown below.

```
<ibe:response xmlns:ibe="urn:ietf:params:xml:ns:ibe">
   <ibe:responseType value="IBE100"/>
   <ibe:body>
      <ibe:privateKey>
        privateKey
      </ibe:privateKey>
   </ibe:body>
</ibe:response>
```

The privateKey is the base64 [B64] encoding of the DER encoding [DER] of the following structure:

```
IBEPrivateKeyReply ::= SEQUENCE {
  pkgIdentity    IBEIdentityInfo,
  pgkAlgorithm   OBJECT IDENTIFIER,
  pkgKeyData     OCTET STRING,
  pkgOptions     SEQUENCE SIZE (1..MAX) OF PKGOption
}

PKGOption ::=  SEQUENCE  {
  optionID    OBJECT IDENTIFIER,
  optionValue OCTET STRING
}
```

The pkgIdentity is an IBEIdentityInfo structure that MUST be identical to the IBEIdentityInfo structure that was sent in the key request.

The pkgAlgorithm is an OID that identifies the algorithm of the returned private key. The OIDs for the BB1 and BF algorithms are defined in [IBCS].

The pkgKeyData is a structure that contains the actual private key. Private-key formats for the BB1 and BF algorithms are defined in [IBCS].

A server MAY pass back additional information to a client in the pkgOptions structure. A client that receives a IBEPrivateKeyReply from a PKG that contains information in a pkgOptions structure that it is unable process MUST NOT use the IBE private key in the IBEPrivateKeyReply structure for any cryptographic operations. A client MUST be able to process an IBEPrivateKeyReply that contains no PKGOptions structure.

5.6.2. The IBE101 responseCode

The responseCode IBE101 is reserved to ensure interoperability with earlier versions of the protocol described in this document. An example of such a response is shown below. A response with the IBE101 responseCode SHOULD contain no body. If information is contained in the body of such a response, the client receiving the response MUST discard any data that is contained in the body.

```
<ibe:response xmlns:ibe="urn:ietf:params:xml:ns:ibe">
   <ibe:responseType value="IBE101"/>
   <ibe:body>
      This message must be discarded by the recipient
   </ibe:body
</ibe:response>
```

5.6.3. The IBE201 responseCode

A PKG MAY support authenticating users to external authentication mechanisms. If this is the case, the server replies to the client with responseCode IBE201 and the body of the response MUST contain a <ibe:location> element that specifies the URI of the authentication mechanism. An example of such a response is shown below.

```
<ibe:response xmlns:ibe="urn:ietf:params:xml:ns:ibe">
   <ibe:responseType value="IBE201"/>
   <ibe:body>
      <ibe:location
         URI="http://www.example.com/enroll.asp"/>
   </ibe:body>
</ibe:response>
```

The client can now contact the URI returned in such a response using the same mechanisms as defined in Section 5.2 to obtain an authentication credential. Once the client has obtained the credential from the authentication mechanism at this URI, it sends a new key request to the PKG with the correct authentication credentials contained in the request, placing the authentication credential in the <ibe:authData> element of a key request as described in Section 5.5.

5.6.4. The IBE300 responseCode

The IBE300 responseCode indicates that an internal server error has occurred. Information that may help diagnose the error MAY be included in the body of such a response. An example of such a response is shown below. Upon receiving a IBE300 responseCode, the client MUST abort the key request and discard any data that was included in the body of the response.

```
<ibe:response xmlns:ibe="urn:ietf:params:xml:ns:ibe">
   <ibe:responseType value="IBE300"/>
   <ibe:body>
      Widget phlebotomy failure
   </ibe:body>
</ibe:response>
```

5.6.5. The IBE301 responseCode

The IBE303 responseCode indicates that an invalid key request has been received by the server. Information that may help diagnose the error MAY be included in the body of such a response. An example of such a response is shown below. Upon receiving an IBE301 responseCode, the client MUST abort the key request and discard any data that was included in the body of the response.

```
<ibe:response xmlns:ibe="urn:ietf:params:xml:ns:ibe">
   <ibe:responseType value="IBE301"/>
   <ibe:body>
      Some additional stuff
   </ibe:body>
</ibe:response>
```

5.6.6. The IBE303 responseCode

The IBE303 responseCode indicates that the server is unable to correctly process the request because the version of the request is no longer supported by the server. Information that may help diagnose the error MAY be included in the body of such a response.

An example of such a response is shown below. Upon receiving an IBE303 responseCode, the client MUST abort the key request and discard any data that was included in the body of the response.

```
<ibe:response xmlns:ibe="urn:ietf:params:xml:ns:ibe">
   <ibe:responseType value="IBE303"/>
   <ibe:body>
      Version 3.3 or later needed
   </ibe:body>
</ibe:response>
```

5.6.7. The IBE304 responseCode

The IBE304 responseCode indicates that a valid key request has been received by the server, but the authentication credentials provided were invalid. Information that may help diagnose the error MAY be included in the body of such a response. An example of such a response is shown below. Upon receiving an IBE304 responseCode, the client MUST abort the key request and discard any data that was included in the body of the response.

```
<ibe:response xmlns:ibe="urn:ietf:params:xml:ns:ibe">
   <ibe:responseType value="IBE304"/>
   <ibe:body>
      Helpful error message
   </ibe:body>
</ibe:response>
```

5.7. The application/ibe-pkg-reply+xml MIME type

The following summarizes the properties of the application/ibe-pkg-reply+xml MIME type.

MIME media type name: application

MIME subtype name: ibe-pkg-reply+xml

Mandatory parameters: none

Optional parameters: none

Encoding considerations: This media type MUST be encoded as US-ASCII [ASCII].

Security considerations: The data conveyed as this media type is an IBE private key, so its confidentiality and integrity are extremely important. To ensure this, the response from the server that contains an IBE private key must take place over a secure

protocol, such as TLS 1.2 or its successors. To ensure the validity of the server, the client MUST verify the server certificate and MUST abort the key request if the verification of the server certificate of the TLS connection fails. This media type contains no active content and does not use compression.

Interoperability considerations: There are no known interoperability considerations for this media type.

Applications that use this media type: Applications that implement IBE in compliance with this specification will use this media type. The most commonly used of these applications are encrypted email and file encryption.

Additional information: none

Person and email address for further information: Luther Martin, martin@voltage.com.

Intended usage: COMMON

Author/Change controller: Luther Martin, martin@voltage.com.

6. ASN.1 Module

The following ASN.1 module summarizes the ASN.1 definitions discussed in this document.

```
IBEARCH-module { joint-iso-itu-t(2) country(16) us(840)
  organization(1) identicrypt(114334) ibcs(1) ibearch(5)
  module(5) version(1)
}

DEFINITIONS IMPLICIT TAGS ::= BEGIN

IBESysParams ::= SEQUENCE {
  version               INTEGER { v2(2) },
  districtName          IA5String,
  districtSerial        INTEGER,
  validity              ValidityPeriod,
  ibePublicParameters   IBEPublicParameters,
  ibeIdentityType       OBJECT IDENTIFIER,
  ibeParamExtensions    IBEParamExtensions OPTIONAL
}
```

```
   ValidityPeriod ::= SEQUENCE {
     notBefore     GeneralizedTime,
     notAfter      GeneralizedTime
   }

   IBEPublicParameters ::= SEQUENCE (1..MAX) OF
     IBEPublicParameter

   IBEPublicParameter  ::= SEQUENCE {
     ibeAlgorithm          OBJECT IDENTIFIER,
     publicParameterData   OCTET STRING
   }

   IBEParamExtensions ::= SEQUENCE OF IBEParamExtension

   IBEParamExtension ::= SEQUENCE {
     ibeParamExtensionOID     OBJECT IDENTIFIER,
     ibeParamExtensionValue   OCTET STRING
   }

   ibcs OBJECT IDENTIFIER ::= {
     joint-iso-itu-t(2) country(16) us(840)
     organization(1) identicrypt(114334) ibcs(1)
   }

   ibeParamExt OBJECT IDENTIFIER ::= {
     ibcs ibcs3(3) parameter-extensions(2)
   }

   pkgURI OBJECT IDENTIFIER ::= { ibeParamExt pkgURI(1) }

   IBEPrivateKeyReply ::= SEQUENCE {
     pkgIdentity  IBEIdentityInfo,
     pgkAlgorithm OBJECT IDENTIFIER,
     pkgKeyData   OCTET STRING,
     pkgOptions   SEQUENCE SIZE (1..MAX) OF PKGOption
   }

   PKGOption  ::=  SEQUENCE  {
     optionID    OBJECT IDENTIFIER,
     optionValue OCTET STRING
   }

   uriPPSOID OBJECT IDENTIFIER ::= {
     joint-iso-itu-t(2) country(16) us(840)
     organization(1) identicrypt(114334)
     pps-schemas(3) ic-schemas(1) pps-uri(1) version(1)
   }
```

```
   IBEIdentityInfo ::= SEQUENCE {
     district        IA5String,
     serial          INTEGER,
     identityType    OBJECT IDENTIFIER,
     identityData    OCTET STRING
   }

   END
```

7. Security Considerations

7.1. Attacks outside the Scope of This Document

Attacks on the cryptographic algorithms that are used to implement IBE are outside the scope of this document. Such attacks are detailed in [IBCS], which defines parameters that give 80-bit, 112-bit, and 128-bit encryption strength. We assume that capable administrators of an IBE system will select parameters that provide a sufficient resistance to cryptanalytic attacks by adversaries.

Attacks that give an adversary the ability to access or change the information on a PPS or PKG, especially the cryptographic material (referred to in this document as the master secret), will defeat the security of an IBE system. In particular, if the cryptographic material is compromised, the adversary will have the ability to recreate any user's private key and therefore decrypt all messages protected with the corresponding public key. To address this concern, it is highly RECOMMENDED that best practices for physical and operational security for PPS and PKG servers be followed and that these servers be configured (sometimes known as hardened) in accordance with best current practices [NIST]. An IBE system SHOULD be operated in an environment where illicit access is infeasible for attackers to obtain.

Attacks that require administrative access or IBE-user-equivalent access to machines used by either the client or the server components defined in this document are also outside the scope of this document.

We also assume that all administrators of a system implementing the protocols that are defined in this document are trustworthy and will not abuse their authority to bypass the security provided by an IBE system. Similarly, we assume that users of an IBE system will behave responsibly, not sharing their authentication credentials with others. Thus, attacks that require such assumptions are outside the scope of this document.

7.2. Attacks within the Scope of This Document

Attacks within the scope of this document are those that allow an adversary to:

- passively monitor information transmitted between users of an IBE system and the PPS and PKG
- masquerade as a PPS or PKG
- perform a denial-of-service (DoS) attack on a PPS or PKG
- easily guess an IBE users authentication credential

7.2.1. Attacks on the Protocols Defined in This Document

All communications between users of an IBE system and the PPS or PKG are protected using TLS 1.2 [TLS]. The IBE system defined in this document provides no additional security protections for the communications between IBE users and the PPS or PKG. Therefore, the described IBE system is completely dependent on the TLS security mechanisms for authentication of the PKG or PPS server and for confidentiality and integrity of the communications. Should there be a compromise of the TLS security mechanisms, the integrity of all communications between an IBE user and the PPS or PKG will be suspect.

The protocols defined in this document do not explicitly defend against an attacker masquerading as a legitimate IBE PPS or PKG. The protocols rely on the server authentication mechanism of TLS [TLS]. In addition to the TLS server authentication mechanism, IBE client software can provide protection against this possibility by providing user interface capabilities that allow users to visually determine that a connection to PPS and PKG servers is legitimate. This additional capability can help ensure that users cannot easily be tricked into providing valid authorization credentials to an attacker.

The protocols defined in this document are also vulnerable to attacks against an IBE PPS or PKG. Denial-of-service attacks against either component can result in users' being unable to encrypt or decrypt using IBE, and users of an IBE system SHOULD take the appropriate countermeasures [DOS, BGPDOS] that their use of IBE requires.

The IBE user authentication method selected by an IBE PKG SHOULD be of sufficient strength to prevent attackers from easily guessing the IBE user's authentication credentials through trial and error.

8. IANA Considerations

8.1. Media Types

With this specification, IANA has registered three media types in the standard registration tree. These are application/ibe-pp-data, application/ibe-key-request+xml, and application/ibe-pkg-reply+xml. The media type application/ibe-pp-data is defined in Section 4.3 of this document. The media type application/ibe-key-request+xml is defined in Section 5.4 of this document. The media type application/ibe-pkg-reply+xml is defined in Section 5.7 of this document.

8.2. XML Namespace

The IANA is requested to register the following namespace identifier:

urn:ietf:params:xml:ns:ibe

Registrant Contact:

Luther Martin
Voltage Security
1070 Arastradero Rd Suite 100
Palo Alto, CA 94304

Phone: +1 650 543 1280
Email: martin@voltage.com

XML:

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C/DTD XHTML Basic 1.0//EN"
          "http://www.w3.org/TR/xhtml-basic/xhtml-basic10.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="content-type"
          content="text/html;charset=iso-8859-1"/>
    <title>Identity-Based Encryption</title>
  </head>
  <body>
    <h1>Namespace for Identity-Based Encryption</h1>
    <h2>urn:ietf:params:xml:ns:ibe</h2>
    <p>
<a href="http://www.rfc-editor.org/rfc/rfc5408.txt">RFC5408</a>.
    </p>
  </body>
</html>
```

9. References

9.1. Normative References

[ASCII] ISO/IEC 646:1991 - Information Technology - ISO 7-bit Coded Character Set for Information Exchange.

[ASN1] ITU-T Recommendation X.680: Information Technology - Abstract Syntax Notation One, 1997.

[AUTH] Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., and L. Stewart, "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, June 1999.

[B64] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.

[CMS] Housley, R., "Cryptographic Message Syntax (CMS)", RFC 3852, July 2004.

[DER] ITU-T Recommendation X.690: OSI Networking and System Aspects: Abstract Syntax Notation One (ASN.1), July 2002.

[DOS] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, May 2000.

[HTTP] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[HTTPTLS] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.

[IBCS] Boyen, X. and L. Martin, "Identity-Based Cryptography Standard (IBCS) #1: Supersingular Curve Implementations of the BF and BB1 Cryptosystems", RFC 5091, December 2007.

[IRI] Duerst, M. and M. Suignard, "Internationalized Resource Identifiers (IRIs)", RFC 3987, January 2005.

[KEY] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[PKIX] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.

[SMTP] Klensin, J., "Simple Mail Transfer Protocol", RFC 5321, October 2008.

[TLS] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[URI] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[XML] W3C, Extensible Markup Language (XML) 1.0 (Fourth Edition), September 2006.

9.2. Informative References

[BGPDOS] Turk, D., "Configuring BGP to Block Denial-of-Service Attacks", RFC 3882, September 2004.

[IBECMS] Martin, L. and M. Schertler, "Using the Boneh-Franklin Identity-Based Encryption Algorithm with the Cryptographic Message Syntax (CMS)", RFC 5409, January 2009.

[NIST] M. Souppaya, J. Wack and K. Kent, "Security Configuration Checklist Program for IT Products - Guidance for Checklist Users and Developers", NIST Special Publication SP 800-70, May 2005.

[P1363] IEEE P1363, "Standard Specifications for Public-Key Cryptography", 2001.

Authors' Addresses

Guido Appenzeller
Stanford University
Gates Building 3A
Stanford, CA 94305

Phone: +1 650 732 2273
EMail: appenz@cs.stanford.edu

Luther Martin
Voltage Security
1070 Arastradero Rd, Suite 100
Palo Alto, CA 94304
USA

Phone: +1 650 543 1280
EMail: martin@voltage.com

Mark Schertler
Axway
1600 Seaport Blvd, Suite 400
Redwood City, CA 94063
USA

Phone: +1 650 216 2039
EMail: mschertler@us.axway.com