

INTERNET-DRAFT MIME Encapsulation of Aggregate Applet Objects (mapplet)
[draft-bahreman-mapplet-spec-00.txt](#)

June 13 1996

Expires 13 December 1996

MIME Encapsulation of Aggregate Applet Objects (mapplet)

Alireza Bahreman <bahreman@eit.com>
James Galvin <galvin@commerce.net>
Rajkumar Narayanaswamy <rajkumar@eit.com>
Nick Zhang <zhang@eit.com>

INTERNET-DRAFT MIME Encapsulation of Aggregate Applet Objects (mapplet)

Status of This Document

This document is being circulated for comment. Please send your comments to the authors at <mapplet-authors@eit.com> or to the mapplet mail list <mapplet@eit.com>. You can subscribe to the mailing list by sending an email to majordomo@eit.com with the following line as the only line in the BODY of the message:

```
subscribe mapplet your@email.address
```

To unsubscribe or find out more about using Majordomo, replace the BODY with the line:

```
help
```

and send the message to majordomo@eit.com. If consensus is reached, this document may be submitted to the IESG as a Proposed Standard protocol specification for use with MIME.

This document is an Internet-Draft. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months. Internet-Drafts may be updated, replaced, or obsoleted by other documents at any time. It is not appropriate to use Internet-Drafts as reference material or to cite them other than as a ``working draft'' or ``work in progress.''

To learn the current status of any Internet-Draft, please check the `l1d-abstracts.txt` listing contained in the Internet-Drafts Shadow Directories on `ds.internic.net` (East USA), `ftp.isi.edu` (West USA), `nic.nordu.net` (North Europe), `ftp.nis.garr.it` (South Europe), `munari.oz.au` (Pacific Rim), or `ftp.is.co.za` (Africa).

Abstract

The applets written in the Java programming [[JAVA](#)] language are becoming a household name on the World Wide Web. Currently, web browsers are able to retrieve and display these applets. This document describes a set of guidelines for conforming mail user agents to be able to send and display applets. The applets are transferred within a MIME [[MIME](#)] message. In order to do this, a new MIME subtype, Application/Applet, is being defined in this document. We also discuss how other protocols for adding security to MIME objects could be used to provide additional security.

Bahreman, et. al.

[Page 2]

INTERNET-DRAFT MIME Encapsulation of Aggregate Applet Objects (mapplet)

Table of Contents

Status of This Document.....	2
Abstract.....	2
Table of Contents.....	3
1. Introduction.....	4
2. The MIME Application/Applet Content-Type.....	4
2.1 The Version Parameter.....	5
2.2 The Name Parameter.....	5
2.3 The Site Parameter.....	5
2.4 The Source Parameter.....	6
2.5 Syntax.....	6
3. Sending a single applet.....	6
4. Use of the Content-Type: Multipart/Related.....	6
5. Use in MIME encapsulation of aggregate HTML documents...	7
6. Using MIME security to sign applets.....	8
7. Examples.....	8
7.1 Single self-contained applet.....	9
7.2 Aggregate applet with referred class.....	9
7.3 Applet as data for MHTML.....	11
7.4 MOSS enhanced.....	13
8. Encoding Considerations.....	14
9. Security Considerations.....	14
10. Application usage.....	15
Acknowledgments.....	16
Appendix A - IANA Registration Request.....	16
References.....	16
Authors' Address.....	17
Expiration and File Name.....	18

INTERNET-DRAFT MIME Encapsulation of Aggregate Applet Objects (mapplet)

1. Introduction

Currently, web browsers retrieve data and documents mostly in a "pull" mode. That is, information is presented to the user upon accessing a particular site. On the other hand, email is mostly used in a "push" mode in which information is sent to the user, not necessarily involving their initiative. The ability to extend the information access using the push model expands the application and utility of the Information Infrastructure.

In addition to static or "passive" data on the Net, it is desirable to be able to retrieve and/or send "active" content. As an example of active content, consider the applets written in Java, which are becoming a household name on the World Wide Web. As a matter of fact, the idea of interactive or active messaging has been proposed and experimented in the email community for some time [[ENABLED-MAIL](#)].

Combining the need for active content with the push model for information delivery is exactly what is being advocated here. This document describes a set of guidelines for confirming mail user agents to be able to send and display applets. We describe how to encapsulate an applet object in MIME. In order to do this, a new subtype, Application/Applet, is being defined in this document. We also discuss

how other protocols for adding security to MIME objects could be used to provide additional security.

An alternative approach is to simply send an email which includes the pointer to an applet, and let the recipient retrieve the applet with other means. That method is common practice on the Web today, where an <APPLET> tag in an HTML [\[HTML\]](#) document is defined to be a pointer to the applet. We do not further describe this alternative in this document.

[Section 2](#) of this document will describe the MIME subtype defined to carry applet objects. Sections [3](#) through [5](#) discuss the relationship and interactions with other existing MIME Content-Types. Using MIME security techniques are outlined in [Section 6](#). We include all examples of MIME messages in [Section 7](#). Sections [8](#) and [9](#) of the document enumerate both the encoding and security considerations. An application of this new MIME Content-Type is discussed in [Section 10](#). [Appendix A](#) provides a copy of the completed registration form being sent to IANA for the registration of the Application/Applet Subtype.

[2](#). The MIME Application/Applet Content-Type

We propose a new subtype of the Application Content-Type called, Applet, to encapsulate information needed to transport an applet. The following four parameters are also defined for the Content-Type field of

Bahreman, et. al.

[Page 4]

INTERNET-DRAFT MIME Encapsulation of Aggregate Applet Objects (mapplet)

Application/Applet. The first two parameters, Version and Name, are mandatory. The other parameters, Site and Source, are optional. There are numerous examples in [Section 7](#) for the use of the Application/Applet MIME Content-Type.

[2.1](#) The Version Parameter

A critical parameter that MUST be specified in the Content-Type field for Application/Applet is the version of the system the applet is designed for. This is needed by the receiving end to determine the system (e.g. interpreter) to invoke in order to process the applet. This information is specified with a "version" parameter, as in:

Content-Type: application/applet; version="Java 1.0.2"

Unlike some other parameter values, the values of the version parameter are NOT case sensitive. There is no default for this parameter.

[2.2](#) The Name Parameter

In order to identify the class names for the applet and related classes embedded in the Application/Applet, a Name parameter MUST appear in the Content-Type field, as in:

Content-type: application/applet; name="MyApplet.class"

This will enable applets and related classes to be saved as their real class file names before an applet interpreter is invoked to run the applet. The values of the name parameter are case sensitive.

[2.3](#) The Site Parameter

An optional parameter in the Content-Type field is the site (or sites) to which the applet is to connect. This information is needed to be conveyed to the application processing the applet which determines whether or not to grant such privilege. (See [Section 9](#) for additional security considerations.) Since not all applets may require to connect to other machines, this parameter is optional. The values of the site parameter are NOT case sensitive. The default for this parameter is null (i.e. no host). An example use of this parameter would be:

Content-Type: application/applet;
site="eco.eit.com:80, eco.eit.com:88"

Note that the sheer presence of this parameter does NOT guarantee the applet will be allowed to access those sites. Applet designers SHOULD take this into consideration and design applets that detect this situation and handle failure gracefully.

[2.4](#) The Source Parameter

An optional parameter in the Content-Type field is the Source parameter. This parameter indicates whether or not the message content is the source code of the applet. The only case insensitive values of this parameter are FALSE or TRUE. An example use of this parameter follows:

Content-Type: application/applet; source=TRUE

Default value for this parameter is FALSE. This means that the assumed content of the message is the actual applet executables (encoded binary) and not the source code.

[2.5](#) Syntax

The formal grammar for the subtype Application/Applet Content-Type field is specified by the following BNF:

<<THIS SECTION WILL BE FILLED IN THE NEXT VERSION OF THE DRAFT>>

[3.](#) Sending a single applet

The simplest example of an applet object is one which needs no input data and consists of only one file. In Java terminology, an applet which refers only to the standard class libraries and consists of a single class file. For example, the MyApplet.class file may contain an applet which will display the message "Hello World", once started. This applet can easily be encapsulated in a MIME message using the new Content-Type. See [Section 7.1](#) for an example of a single self-contained applet encapsulated in MIME using the new Application/Applet Content-Type.

[4.](#) Use of the Content-Type: Multipart/Related

For the more complicated applets, consider one with data input parameters which refers to non standard classes. In this case, we need to aggregate several components together. For this purpose, we suggest

the use of the Multipart/Related MIME Content-Type [[RELATED](#)]. This would allow the receiving mail user agent to process the components together and in the appropriate order for display. See the example in [Section 7.2](#) for the use of the Multipart/Related Content-Type.

The value of the Type parameter or root of the Multipart/Related message MUST be of type Application/Applet or Multipart/Alternative which CAN be resolved to Application/Applet. (We suggest the use of Multipart/Alternative to allow the sender to specify a readable text description of the applet in addition to the applet itself; therefore if the recipient fails to display the applet, it would be able to display the text/plain description instead.)

The Start parameter MUST specify the content-ID of the compound object's root. If not present, the "root" is the first body part in the Multipart/Related entity. The root MUST be of type Application/Applet or Multipart/Alternative which CAN be resolved to Application/Applet.

The Start-Info parameter can provide additional information such as the data input parameters for the applet. The value of the Start-Info parameter MUST be the content-ID of the message component that contains the data for the applet. This may be a message of type Text/Html.

Similar to [[MHTML](#)], we also rely on an additional parameter, Includes, in the Content-Type field of Multipart/Related. The case-insensitive values of this parameter are COMPLETE and INCOMPLETE. They refer to whether the aggregate MIME message includes all referred classes of the applet object or not, respectively. The default value for the Includes parameter is COMPLETE.

[5](#). Use in MIME encapsulation of aggregate HTML documents

Since the applet can be considered a data object within an HTML object, we envision a possible tight coupling between the MIME encapsulation of aggregate HTML documents [[MHTML](#)] work and the work presented here. For example, just like an image data would be included in a MIME encapsulation of an HTML object with a tag, an applet data could be included for an <APPLET> tag.

In order to do this, we recommend encapsulating the applet within a Multipart/Related MIME object with no Start-Info. This is because the HTML object would already convey the data input parameters for the applet in the <APPLET> tag. See the example in [Section 7.3](#) for a sample MIME encapsulated HTML document containing an image and an applet.

INTERNET-DRAFT MIME Encapsulation of Aggregate Applet Objects (mapplet)

[6.](#) Using MIME security to sign applets

The MIME encapsulation of an applet object is vulnerable to the same attacks as any other MIME object. In particular, it is possible for an active eavesdropper to modify the MIME message in transit. Therefore, neither the authenticity of the sender nor the integrity of the applet can be relied upon. The content of the MIME message may be seen by any passive eavesdropper. To combat these security concerns, the sender of the MIME message could sign and optionally encrypt the MIME message sent. The signature can be verified by the recipient (assuming the public key of the sender is available) assuring the authenticity of the sender and the integrity of the message contents. If encryption is performed, the recipient can decrypt and be assured that no intermediary was able to read the content of the message. Encryption provides for confidentiality of the applet during transmit.

The framework within which security services may be applied to MIME body parts is described in [\[RFC1847\]](#). One can use Multipart/Signed and Multipart/Encrypted Content-Types to secure MIME objects. Two mechanisms for achieving security in MIME have been proposed [\[MOSS\]](#) [PGP/MIME], which use this framework. There is another protocol specified [\[S/MIME\]](#) for adding cryptographic signature and/or encryption services to MIME messages. The use of any one of these mechanisms is suggested for providing additional security guarantees to the recipient regarding the authenticity and confidentiality of the MIME encapsulated applet object.

As an example, consider the use of the MOSS mechanism. To sign or encrypt the applet, the applet class files are encapsulated in Multipart/Related object which is in turn signed by MOSS and put into a Multipart/Signed object. When the recipient gets the applet, they should verify the MOSS signature first before invoking any application to run the applet. If the signature verification fails, the recipient mail user agent should prompt the user a warning dialog before proceeding to the next step of invoking and/or executing the applet. [Section 7.4](#) demonstrates a possible signed, MIME-encapsulated applet object using the MOSS security extensions.

[7.](#) Examples

Here, we present example MIME messages for the various scenarios outlined above. These examples are intended to clarify the usage. If there is an inconsistency with the BNF syntax presented in [Section 2.5](#), the BNF syntax is the correct one.

INTERNET-DRAFT MIME Encapsulation of Aggregate Applet Objects (mapplet)

[7.1](#) Single self-contained applet

This example demonstrates the use of the Application/Applet Content-Type. The parameters specify the environment this applet runs in (Java 1.0.2), the original name it was saved under (MyApplet.class), and that the body of this MIME message includes the executable and NOT the source file for this applet (source=False). The body of the message then contains the base64 encoded applet class file.

```
From: bahreman@eit.com
To: zhang@eit.com
Subject: The Hello World Applet
Content-Type: application/applet;
             version="Java 1.0.2";
             name="MyApplet.class";
             source="False"
Content-Transfer-Encoding: base64

<base64 encoded class file>
```

[7.2](#) Aggregate applet with referred class

In this example, we are going to encapsulate an applet which has data input and refers to another class. The Multipart/Related Content-Type is used to aggregate all the parts together. The start and start-info parameters of the Multipart/Related Content-Type indicate the unique message id representing the part of the MIME message corresponding to the main applet class file and its data input, respectively. The include parameter specifies that all referred classes are included in this message; obviously, non of the standard classes are included. The

message part corresponding to unique-id-one@foo.com is the main applet class and specifies the request to connect to eco.eit.com:80 host using the site parameter. (As mentioned before, the receiving system has the right to grant or deny such access.) In addition to the executables and the input data, this message contains the source code for the applet and its referred class as indicated by the source=True parameter value in the last two parts of this aggregate MIME object.

From: bahreman@eit.com
To: zhang@eit.com
Subject: A More Complicated Applet
Content-Type: multipart/related; boundary="boundary.1";
 type="application/applet";
 start="unique-id-one@foo.com";
 start-info="unique-id-two@foo.com";

Bahreman, et. al.

[Page 9]

INTERNET-DRAFT MIME Encapsulation of Aggregate Applet Objects (mapplet)

 includes="complete"

--boundary.1
Content-Type: text/html
Content-ID: "unique-id-two@foo.com"

```
<html>
<head></head>
<body>
<center>
<applet code=MyApplet.class width=150 height=35>
<param name=name1 value="value1">
<param name=name2 value="value2">
</applet>
</center>
</body>
</html>
```

--boundary.1
Content-Type: application/applet;
 version="Java 1.0.2";
 name="MyApplet.class";
 site="eco.eit.com:80"
Content-ID: <unique-id-one@foo.com>
Content-Transfer-Encoding: base64

<base64 encoded class file>

--boundary.1

Content-Type: application/applet;
version="Java 1.0.2";
name="ReferredClass.class"
Content-Transfer-Encoding: base64

<base64 encoded class file>

--boundary.1

Content-Type: application/applet;
version="Java 1.0.2";
name="MyApplet.java";
source="True"
Content-Transfer-Encoding: 7Bit

<7Bit encoded source code for the applet class file>

--boundary.1

Content-Type: application/applet;
version="Java 1.0.2";
name="ReferredClass.java";
source="True"

Bahreman, et. al.

[Page 10]

INTERNET-DRAFT MIME Encapsulation of Aggregate Applet Objects (mapplet)

Content-Transfer-Encoding: 7Bit

<7Bit encoded source code for the referred class file>

--boundary.1

[7.3](#) Applet as data for MHTML

This example demonstrates encapsulating an applet object within an aggregate HTML document encapsulated in MIME. Basically, the applet object and its related information (except the input data) are presented as a Multipart/Related object. This object is then encapsulated inside the Multipart/Related MIME message conveying the HTML document. In this example, the HTML document is a relatively simple file which only points to one applet and one image. Both the image data and the applet data are

therefore encapsulated in the outermost Multipart/Related Content-Type for the aggregate HTML object. The applet aggregate object includes the main class, its only referred class, and source code for both. No input data for the applet is needed as that information (the <applet> tag) is conveyed in the HTML document itself.

From: bahreman@eit.com
To: zhang@eit.com
Subject: A Complicated Applet Inside an HTML Document
Content-Base: "http://www.ietf.cnri.reston.va.us"
Content-Type: multipart/related; boundary="boundary.0";
 type="text/html"

--boundary.0
Content-Type: text/html

```
<html>
<head></head>
<body>
<center>
<applet code=MyApplet.class width=150 height=35>
<param name=name1 value="value1">
<param name=name2 value="value2">
</applet>
</center>
<IMG SRC="/images/ietflogo.gif" ALT="IETF Logo">
</body>
</html>
```

--boundary.0
Content-Location: "/images/ietflogo.gif"

Bahreman, et. al.

[Page 11]

INTERNET-DRAFT MIME Encapsulation of Aggregate Applet Objects (mapplet)

Content-Type: image/gif; name="ietflogo.gif"
Content-Transfer-Encoding: base64

<base64 encoded gif image of the IETF logo>

--boundary.0
Content-Type: multipart/related; boundary="boundary.1";
 type="application/applet";
 start="unique-id-one@foo.com"

```
--boundary.1
Content-Type: application/applet;
    version="Java 1.0.2";
    name="MyApplet.class";
    site="eco.eit.com:80"
Content-ID: <unique-id-one@foo.com>
Content-Transfer-Encoding: base64

<base64 encoded class file>

--boundary.1
Content-Type: application/applet;
    version="Java 1.0.2";
    name="ReferredClass.class"
Content-Transfer-Encoding: base64

<base64 encoded class file>

--boundary.1
Content-Type: application/applet;
    version="Java 1.0.2";
    name="MyApplet.java";
    source="True"
Content-Transfer-Encoding: 7Bit

<7Bit encoded source code for the applet class file>

--boundary.1
Content-Type: application/applet;
    version="Java 1.0.2";
    name="ReferredClass.java";
    source="True"
Content-Transfer-Encoding: 7Bit

<7Bit encoded source code for the referred class file>

--boundary.1

--boundary.0
```

This example shows how to use MOSS [[MOSS](#)] to enhance the security of the applet by signing it. This conforms to the security framework specified by [[RFC1847](#)] for MIME messages. Other mechanisms such as that advocated by [S/MIME] could also be used. In MOSS, the aggregate applet MIME object is encapsulated by a message of type Multipart/Signed Content-Type. The first part is the applet and the second part is the signature for that object.

```
From: bahreman@eit.com
To: zhang@eit.com
Subject: Example of a Signed, Complicated Applet
Content-Type: multipart/signed; protocol="application/moss-signature";
            boundary="boundary.0"
```

```
--boundary.0
Content-Type: multipart/related; boundary="boundary.1";
            type="application/applet";
            start="unique-id-one@foo.com";
            start-info="unique-id-two@foo.com"
```

```
--boundary.1
Content-Type: text/html
Content-ID: "unique-id-two@foo.com"
```

```
<html>
<head></head>
<body>
<center>
<applet code=MyApplet.class width=150 height=35>
<param name=name1 value="value1">
<param name=name2 value="value2">
</applet>
</center>
</body>
</html>
```

```
--boundary.1
Content-Type: application/applet;
            version="Java 1.0.2";
            name="MyApplet.class";
            site="eco.eit.com:80"
Content-ID: <unique-id-one@foo.com>
Content-Transfer-Encoding: base64
```

```
<base64 encoded class file>
```

```
--boundary.1
```

```
Content-Type: application/applet;
           version="Java 1.0.2";
           name="ReferredClass.class"
Content-Transfer-Encoding: base64

<base64 encoded class file>

--boundary.1

--boundary.0
Content-Type: Application/Moss-signature

<moss signature>

--boundary.0
```

[8. Encoding Considerations](#)

Base64 is recommended as the content transfer encoding mechanism for applet and their referred classes embedded in an MIME message. For the source code(s), the 7Bit encoding is recommended. The Multipart/Related Content-Type should NOT be encoded.

[9. Security Considerations](#)

User beware that even a signed applet could be malicious! The security enhancements suggested in this document will not protect the recipient from an applet who's code is malicious. The only guarantee the signing of a MIME encapsulated applet object provides is the authenticity of the origin. The user must configure their mail user agents with extreme caution in order to only run applets whose source is trusted. For example, if the user trusts a company and regularly downloads applications from their public servers and executes them, this user can also receive signed applets from that company and execute them; the user carries the same amount of risk in both cases. One might even argue that we have increased the potential for abuse, by simply allowing the sending of executables.

A key implementation guideline for temporary storage and execution of the MIME encapsulated objects is to store the applet and referred class files in a directory other than the local environment. Using Java

terminology, the Java applet and related classes should not be saved into a directory which is specified in the CLASSPATH environment variable. Collectively they should be saved into a temporary directory for viewing. The reason for this is that some applet viewers or players

Bahreman, et. al.

[Page 14]

INTERNET-DRAFT MIME Encapsulation of Aggregate Applet Objects (mapplet)

treat all classes under CLASSPATH as local classes and therefore, some security checks may be turned off.

Another security consideration to mention is the restrictions imposed on the applet by the environment in which it is executed in. One of these restrictions is to prevent the applet from opening a communication channel to other sites. Recall that the Site parameter of the Application/Applet Content-Type, if present, implies that the applet requires communication to one or more sites. The policy of allowing or denying this request MUST rest on the local environment. The host addresses in the value of the Site parameter are informational and will be used at the discretion of the confirming mail user agent and the applet execution environment.

10. Application usage

<<THIS SECTION WILL INCLUDE EXPLANATION ON WHY THIS IS USEFUL,>>
<<REPORT ON THE IMPLEMENTATION OR IMPLEMENTATION GUIDELINES, AND>>
<<EXAMPLE USE AS PAYMENT PLUG-IN MECHANISM>>

INTERNET-DRAFT MIME Encapsulation of Aggregate Applet Objects (mapplet)

Acknowledgments

The contributions of the following people to early drafts of this document are greatly appreciated:

<<INCLUDE LIST OF ANY CONTRIBUTOR HERE>>

Appendix A - IANA Registration Request

In order to register, this appendix captures the filled email template for the registration of new values with IANA. This email message will be sent to IANA.

To: IANA@isi.edu

Subject: Registration of new MIME content-type/subtype

MIME type name: Application

MIME subtype name: Applet

Required parameters: Version, Name

Optional parameters: Site, Source

Encoding considerations: base64 encoded content recommended for Applet executables. For source, 7Bit is recommended.

Security considerations: In the pure MIME encapsulation of the aggregate applet objects, no security is provided. The MIME objects could be signed in conventional ways to provide a higher level of security such as integrity and authenticity. However, users are cautioned that none of these security measures ensures that the executable applet is not malicious.

Published specification: [draft-bahreman-mapplet-spec-00.txt](#)

Person & email address to contact for further information: mapplet-authors@eit.com

References

NOTE: This list contains some references to Internet drafts. It is anticipated that these Internet Drafts will become RFC-s before this memo. The references will then be changed to refer to the corresponding

Bahreman, et. al.

[Page 16]

INTERNET-DRAFT MIME Encapsulation of Aggregate Applet Objects (mapplet)

RFC instead.

[JAVA] James Gosling & Henry McGilton: "The Java(TM) Language Environment: A White Paper",
<http://www.javasoft.com/java.sun.com/whitePaper/java-whitepaper-1.html>.

[MIME] N. Borenstein & N. Freed: "MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies", [RFC 1521](#), September 1993.

[ENABLED-MAIL] N. Borenstein & M.T. Rose: "MIME Extensions for Mail-Enabled Applications: application/Safe-Tcl and multipart/enabled-mail", Draft in preparation, First Virtual Holdings, Dover Beach Consulting, September 1993.

[HTML] T. Berners-Lee & D. Connolly: "Hypertext Markup Language - 2.0", [RFC 1866](#), November 1995.

[RELATED] Harald Tveit Alvestrand & Edward Levinson: "The Mime Multipart/Related Content-Type", <[draft-ietf-mhtml-related-00.txt](#)>, May 1996.

[MHTML] Jacob Palme & Alexander Hopmann: "Mime E-mail Encapsulation of Aggregate HTML Documents (MHTML)", <[draft-ietf-mhtml-spec-00.txt](#)>, April 1996.

[RFC1847] J. Galvin, S. Murphy, S. Crocker, & N. Freed: "Security Multiparts for MIME: Multipart/Signed and Multipart/Encrypted", October 1995.

[MOSS] S. Crocker, N. Freed, J. Galvin, & S. Murphy: "MIME Object Security Services", [RFC 1848](#), October 1995.

[PGP/MIME] M. Elkins: "MIME Security with Pretty Good Privacy (PGP)", <[draft-elkins-pem-pgp-03.txt](#)>, March 1996.

[S/MIME] RSA Data Security, [ftp://ftp.rsa.com](#).

Authors' Address

Alireza Bahreman, Rajkumar Narayanaswamy, Nick Zhang
EIT/VeriFone
[800](#) El Camino Real
Menlo Park, CA 94025 USA

Telephone: +1 415 617 8000
FAX: +1 415 617 8019
EMail: {bahreman, rajkumar, zhang}@eit.com

Bahreman, et. al.

[Page 17]

INTERNET-DRAFT MIME Encapsulation of Aggregate Applet Objects (mapplet)

mapplet-authors@eit.com

James Galvin
CommerceNet
[4005](#) Miranda Ave., Suite 175
Palo Alto, CA 94304

Telephone: +1 415 858 1930x226
Fax: +1 415 858 1936
EMail: galvin@commerce.net
mapplet-authors@eit.com

Expiration and File Name

This draft expires 13 December 1996.

Its file name is [draft-bahreman-mapplet-spec-00.txt](#).