S/MIME Version 3.1 Message Specification

- Ausarbeitung -

Name:	Thomas Richter
Studienrichtung:	Informatik
Semester:	6
Matrikelnummer:	730 450
eMail:	thomas.richter1@student.fh-nuernberg.de

Studienfach:	Internet Security
Professor:	Prof. Dr. Trommler

Inhaltsverzeichnis

1. Was ist S/MIME

Bei S/MIME, Secure / Multipurpose Internet Mail Extensions, handelt es sich um eine Erweiterung von MIME. Dabei werden folgende Sicherheitsaspekte hinzugefuegt :

- Authentifizierung
- Nachrichtenintegrität
- Fälschungssicherheit des Ursprungs
- Vertraulichkeit der Daten

S/MIME ist nicht nur auf den Emaildienst beschraenkt, bei dem es die Nachrichten um Sicherheitsdienste beim Senden bereichert und angereicherte empfangene Nachrichten entsprechend behandelt, sondern kann auch dort eingesetzt werden, wo auch sonst MIME Objekte versendet werden. Als Beispiel wäre hier der Versand von automatisch generierten Nachrichten zu nennen, wo die Kommunikation ohne menschliches zutun stattfindet, oder das Unterschreiben von automatisch erzeugten Dokumenten.

Dabei setzt S/MIME auf die Objektstruktur von MIME auf. Ein Nachrichtenprogramm muss neben den normalen MIME Blöcken, noch die neuen S/MIME Blöcke verstehen.

2. Übersicht der Spezifikation

In dem Draft wird zum einen beschrieben wie man die Sicherheitsaspekte zu den MIME Daten hinzufügt und zum anderen werden Empfehlungen gegeben wie die Software S/MIME Nachrichten behandeln soll und welche Vorraussetzungen die Software erfüllen muss (Hierbei wird die Terminologie aus [MUSTSHOULD] verwendet um entsprechende Anforderungen oder Vorraussetzungen auszudruecken). Dabei wird oftmals unterschieden, ob es sich um einen Recieving Agent (RA) oder um einen Sending Agent (SA) handelt. Denn es ist nicht immer erforderlich, dass die Software beide Aufgaben löst. Beispielsweise bei automatisierten Systemen ist es möglich, dass RA und SA voneinander getrennte Programme sind.

3. Cryptographic Message Syntax

3.1 CMS Options

CMS erlaubt eine Vielzahl von Optionen um eine groesst moegliche Kompatibilität unter den S/MIME Versionen zu erzielen. Im Folgenden eine tabellarische Übersicht.

	DigestAlgorithm Identifier	**SignatureAlgorithm Identifier**	**KeyEncryption Identifier**
Sending und Receiving agent	MUSS SHA-1 [SHA1] unterstützen	MUSS id-dss [DSS] unterstützen	MUSS rsaEncryption und SOLLTE DH unterstützen
Sending agent		MUSS rsaEncryption unterstützen	SOLLTE rsaEncryption und DH unterstützen
Receiving agent	SOLLTE MD5 [MD5] unterstützen	MUSS rsaEncryption unterstützen	SOLLTE rsaEncryption und DH unterstützen

3.2 Content Type

Es gibt folgende 4 Content Types:

- Data Content Type
- SignedData Content Type
- EnvelopedData Content Type
- CompressedData Content Type

Wird der Data Content Type verwendet, MUSS der SA den "id-data content type identifier" benutzen, um kenntlich zu machen, welchen Inhalt die MIME Daten besitzen.
Der SignedData Content Type wird verwendet, wenn eine Nachricht mit einer digitalen Unterschrift versieht. Damit erhält die Nachricht die Authentifikation, Integrität und Fälschungssicherheit des Absenders.
Um vertrauenswürdige Daten einer Nachricht beizufügen, verwendet man den EnvelopedData Content Type. Man benötigt hierfür alle öffentlichen Schlüssel der Nachrichtenempfänger. Dieser Content Type beinhaltet keine Authentifikation.

Will man eine Nachricht komprimieren, verwendet man den CompressedData Content Type. Dieser fügt allerdings keinerlei Sicherheitsaspekte hinzu, sondern dient lediglich der Reduktion der Datenmenge.

3.3 Attribute und SignerInfo Type

Der SignerInfo Type erlaubt die Verwendung von unterschriebenen und nicht unterschriebenen Attributen.
Der RA MUSS mit einer oder keiner Instanz eines unterschriebenen Attributs umgehen können. Der SA SOLLTE allerdings von jedem der folgenden Attribute eine Instanz bei einer S/MIME Nachricht erzeugen.

3.3.1 signing Time Attribut

Mit diesem Attribut stellt man den Zeitpunkt der Unterzeichnung der Nachricht sicher. Da allerdings die Zeit von einem Synchronisationsdienst bezogen wird, kann die Zeit nur so genau und somit sicher sein, wie der Dienst, der sie bereitstellt.
Der SA MUSS ein Datum bis 2049 als UTC Zeit darstellen, danach als GeneralizedTime.

3.3.2 S/MIMECapabilities Attribut

Wird dieses Attribut verwendet MUSS es unterschrieben sein. Es legt fest, welche kryptografischen Verfahren verwendet werden. Es ist abzusehnen, dass es in Zukunft noch bessere Verfahren geben wird. Damit es keine Probleme mit den Programmen gibt, hat man dieses Attribut als einen Satz aus Attributen definiert. So kann man neue Verfahren einfach hinzufügen, ohne dass es zu Problemen führt. Ein Client sollte hier allerdings keine vollständige Liste der Dienste anfügen, die er unterstützt, da diese sehr lang wäre und Angreifern ggf. Informationen über mögliche Angriffsstellen offenbart.

3.3.3 S/MIMEEncryptionKeyPreference Attribut

Wird dieses Attribut verwendet MUSS es unterschrieben sein. Es darf nur eine Instanz des Attributs geben.
Es gibt den favorisierten Verschlüsselungsalgorithmus des Unterzeichners an. Dabei MUSS RA und SA die tripleDES Algorithmen unterstützen. Der SA SOLLTE das Zertifikat auf das er sich bezieht mit an die Nachricht anhängen. Der RA kann das Zertifikat überspringen, wenn er es schon früher einmal empfangen hat, er SOLLTE den bevorzugten Algorithmus eines Senders allerdings speichern. Ebenso SOLLTE der RA dem Vorzug des Senders, bei einer Antwort nachkommen und diesen verwenden, wenn er ihn unterstützt.

Ist das Attribut gesetzt, dann sollte das X.509 Zertifikat, auf das sich bezogen wird, verwendet werden.
Ist das Attribut nicht gesetzt, dann sollte nach einem Zertifikat mit dem gleichen Namen wie der des Zertifikats des Unterzeichners gesucht werden, welches dann für den Schlüsselaustausch genutzt werden kann.
Ist das Attribut nicht gesetzt und es wird kein Zertifikat gefunden, dann kann keine Verschlüsselung verwendet werden.

3.4 ContentEncryptionAlgorithmIdentifier

RA und SA MUESSEN tripleDES Ent- und Verschlüsselung beherrschen. Der RA sollte aus Gründen der Rueckwaertskompatibilitaet noch Ent- und Verschlüsselung mit RC2/40 Algorithmus beherrschen, auch wenn dieser als ein "schwacher" Algorithmus gilt. SA und RA SOLLTEN Ent- und Verschlüsselung mit dem AES Algorithmus unterstützen, welcher als "starker" Algorithmus angesehen wird.

3.5 Auswahl der Verschlüsselung

Der SA will eine Verschlüsselte Nachricht erzeugen. Dabei stellt sich die Frage welche Verschlüsselungsmethode er verwenden soll. Falls der SA schon eine Nachricht von dem Empfänger bekommen hat, dem er schreiben will und dieser eine Liste mit unterstützten und bevorzugten Algorithmen mitgesendet hat, soll er diese zur Auswahl des Verschlüsselungsmethode mit einbeziehen.
Hat der RA noch keine Liste erstellt, SOLLTE er eine anlegen, nachdem die Unterschrift verifiziert wurde und der Zeitstempel als gültig angesehen wurde.
Gibt es eine Liste, dann SOLLTE der RA den Zeitstempel der Unterschrift ueberpruefen und den Zeitstempel sowie die Kompatibilitätsliste bei sich in der Liste aktualisieren. Liegt der Zeitstempel zu weit in der Zukunft, DARF er NICHT akzeptiert werden.
Der RA MUSS auch noch entscheiden ob er "schwache" Verschlüsselung zulässt oder nicht. Falls Daten nicht mit "schwacher" Verschlüsselung übermittelt werden dürfen, DARF der RA "schwache" Verschlüsselung NICHT zulassen.
Kennt der RA die Algorithmen die der Empfänger erlaubt, SOLLTE der RA daraus entscheiden welchen er verwendet.
Kennt der RA die Algorithmen und die S/MIME Version des Empfängers nicht, sollte er tripleDES verwenden.
Falls er nicht tripleDES verwendet, SOLLTE er RC2/40 verwenden.
Allerdings sollte der SA bei der Verwendung eines "schwachen" Algorithmus den Benutzer fragen, ob er diesen auch wirklich benutzen will. Denn ein schwacher Algorithmus ist heutzutage relativ einfach zu überwinden.

Verschlüsselte Nachrichten machen zurzeit noch einen geringen Teil des Emailverkehrs aus. Daher sind sie für Angreifer wie eine rote Kuh auszumachen. Wenn sie dann lediglich durch einen "schwachen" Algorithmus geschützt sind, ist es fast sinnvoller die Nachricht nicht zu verschlüsseln. Sie somit nur als Teil der Masse, wohl meist unerkannt, mitschwimmen zu lassen.
Ein weiteres Problem sind mehrere Empfänger mit unterschiedlichen Verschlüsselungen. Sollte dies der Fall sein, dann sollte der SA unterschiedliche Nachrichten für die unterschiedlich verwendeten Algorithmen verwenden. Denn so ist sichergestellt, dass er die Nachricht nicht mit "schwacher" und "starker" Verschlüsselung schickt, und so dem Angreifer die Chance bietet eine Nachricht mit "starker" Verschlüsselung zu lesen, indem er die gleiche Nachricht, die mit "schwachen" Verschluesselung versehen wurde knackt.

4 Erstellen einer S/MIME Nachricht

Eine S/MIME Nachricht besteht aus einer Vielzahl von MIME- und CMS-Inhalten. Damit alle Daten eine einheitliche Struktur bekommen, werden alle Daten an eine Funktion gegeben, die daraus CMS Objekte erzeugt. Diese werden dann in MIME Objekte eingebettet. Im Folgenden wird nun beschrieben, wie die unterschiedlich eingebetteten, sicheren S/MIME Nachrichten formatiert werden müssen.
Es gibt mehrere Formate für "signed-only" und "signed and enveloped" Daten, allerdings nur ein Format für "enveloped-only" Daten. Die unterschiedlichen Formate resultieren aus den verwendeten Aufgaben.
[MIME-SPEC] beschreibt, wie die MIME Daten im einzelnen aufbereitet werden.
Es gibt allerdings noch ein paar Punkte die man berücksichtigen muss, damit eine Nachricht auch auf unterschiedlichen Systemen problemlos verarbeitet und gegebenenfalls weitergeleitet werden kann. Jeder MIME Block MUSS in eine kanonische Form gebracht werden, damit die Blöcke auf unterschiedlichen Systemen gleich dargestellt werden. Insbesondere gilt dies für Textblöcke, da beispielsweise der Zeilenumbruch in unterschiedlichen Systemen, anders definiert ist. Das verwendete [CHARSET] sollte angegeben werden, damit der Empfänger eine dementsprechende Darstellung ausgibt.
Für multipart/signed Formate sollte 7 Bit Transfer Encoding (TE) (beispielsweise base64) verwendet werden, solange kein Content-Transfer-Encoding Header vorhanden ist. Da noch nicht alle Systeme eine 8 Bit Codierung verwenden, ist nicht vorhersagbar, was mit einer Nachricht passiert, die mit 8 Bit TE codiert wurde, aber von einem Dienst der nur 7 Bit TE unterstützt verarbeitet wird. Wahrscheinlich wird aber eine Signatur dadurch ungültig.

4.1 application/pkcs7-mime Type

Dieser Typ wird verwendet, wenn man CMS Content Types mit EnvelopedData, SignedData und CompressedData verwendet. Da CMS Content Types meist binäre Daten sind, wird Base-64 TE verwendet. Die MIME Header aller application/pkcs7-mime Objekte SOLLTEN den eigentlich optionalen Parameter "smimetype" beinhalten.

MIME Type (Dateiname SOLLTE 8.3 Format verwenden)	Dateierweiterung
Application/pkcs7-mime (SignedData, EnvelopedData)	.p7m
Application/pkcs7-mime (degenerate SignedData certificate management message)	.p7c
Application/pkcs7-mime (CompressedData)	.p7z
Application/pkcs7-signature (SignedData)	.p7s

Mit der Dateierweiterung lässt sich der verwendete MIME Typ ableiten, wobei eine gute S/MIME Implementierung den MIME Typ abfragen MUSS, und sich NICHT auf die Dateierweiterung verlassen DARF.

4.2 smime-type Parameter

Name	CMS Type	Inhalt
enveloped-data	EnvelopedData	id-data
signed-data	SignedData	id-data
certs-only	SignedData	none
compressed-data	CompressedData	id-data

Wie oben bereits erwähnt ist dieser Parameter optional, bietet aber die Möglichkeit herauszufinden, ob eine Nachricht unterschrieben oder verschlüsselt ist, ohne in den CMS Block hineinzuschauen.
Für etwaige Erweiterungen sollte die vorgegebene Namenskonvention eingehalten werden.

4.3 Erstellen einer Enveloped-only Nachricht

Diese Nachricht wird nur eingebettet aber nicht signiert. Damit ist die Datenintegrität nicht sichergestellt, da die Nachricht in bestimmter weise modifiziert werden kann, dass sie dennoch als unverändert erkannt wird.
Die einzelnen Schritte zum erzeugen der Nachricht sind bei allen kommenden Nachrichtentypen sehr ähnlich. Daher sollen nur hier die einzelnen Schritte aufgezaehlt werden.

Schritt 1: Aufbereiten des verwendeten MIME Blocks

Schritt 2: Erzeugen eines CMS Objekts vom Typ EnvelopedData

Schritt 3: Einbinden des Objekts aus Schritt 2 in ein CMS ContentInfo Objekt

Schritt 4: Einfuegen des Objekts aus Schritt 3 in ein application/pkcs7-mine MIME Objekt

Ein Beispiel:

```
Content-Type: application/pkcs7-mime; smime-type=enveloped-data; name=smime.p7m
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m
        rfvbnj756tbBghyHhHUujhJhjH77n8HHGT9HG4VQpfyF467GhIGfHfYT6
        7n8HHGghyHhHUujhJh4VQpfyF467GhIGfHfYGTrfvbnjT6jH7756tbB9H
        f8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
        0GhIGfHfQbnj756YT64V
```

4.4 Erstellen einer Signed-only Nachricht

Hier gibt es zwei Formate, "application/pkcs7-mine with" "SignedData" oder "multipart/signed". Es SOLLTEN beide verarbeitet werden können, obwohl nur das multipart/signed Format benutzen werden sollte.
"SignedData" kann nur von S/MIME Clients gelesen werden, "multipart/signed" hingegen von allen Clients.
Der "smime-type parameter" für "application/pkcs7-mine with SignedData" ist "signed-data" mit Dateierweiterung ".p7m".
Bei dem "multipart/signed" Format gibt es noch zwei spezielle Parameter - den Protokoll Parameter und den "micalg" Parameter. Der Protokoll Parameter MUSS "application/pkcs7-signature" sein, der "micalg" Parameter gibt den Algorithmus an mit dem die Checksumme für die Nachrichtenintegrität gebildet wird. Es sollte MD5 und SHA-1 verwendet werden, SHA-256 SHA-384 SHA-512 sind schon vorgesehen, aber sind noch nicht so weit verbreitet, um sie ohne Probleme nutzen zu können.

4.5 Erstellen einer Compressed-only Nachricht

Die Möglichkeit MIME Blöcke zu komprimieren gibt es erst seit S/MIME Version 3.1. Der "smime-type parameter" für "compressed-only" Nachrichten ist "compressed-data" mit der Dateierweiterung ".p7z".

Bei den MIME Formaten "signed-only", "encrypted-only" und "compressed-only" kann eine beliebige Schachtelung vorliegen. Es gibt allerdings Unterschiede bezüglich der Sicherheit und Effizienz bei der Wahl der Verschachtelungsreihenfolge. Beispielsweise ist die Kompressionsrate von unverschlüsseltem bzw. uncodiertem Text viel höher als von verarbeitetem Text.

4.6 Erstellen einer Certificate Management Nachricht

Diese Nachricht wird für das Übermitteln von Zertifikaten und/oder von so genannte Certificate Revocation Lists (CRLs) benötigt und den "mime-type parameter" für diese Nachricht ist "certs-only" und die Dateierweiterung lautet ".p7c".
Der SA, der Nachrichten unterschreibt, MUSS ein X.509 Zertifikat für die Unterschrift besitzen, damit der RA dies verifizieren kann. Wie ein Zertifikat bezogen wird, wird in S/MIME Version 3.1 nicht weiter beschrieben, in S/MIME Version 2 gab es noch ein bestimmtes Verfahren.
Der RA MUSS allerdings in irgendeiner weise Zertifikate beziehen können, um sie digitalen Umschlägen verwenden zu können. Beschrieben wird dies in [CERT31].

4.7 Erzeugen eines Schlüsselpaares

Wie auch bei anderen Aufgaben, hat man auch hier das Problem Zufallszahlen zu erzeugen. Das Problem wird unter [RANDOM] genauer beschrieben. Soll ein RSA Schlüsselpaar erzeugt werden, SOLLTE der Client einen RSA Schlüssel mit minimaler Schlüssellänge von 768 bit erzeugen. Er DARF KEINE Schlüssel mit Schluessellaenge unter 512 bit erzeugen.
Zu Problemen kann es bei Schlüssellängen über 1024 bit kommen, da ältere RAs die Signaturen nicht verifizieren können. Ein RA SOLLTE in der Lage sein mit Schlüssellängen über 512 bit jede Signatur zu verifizieren.

5. Identifikation einer S/MIME Nachricht

Eine Nachricht wird als eine S/MIME Nachricht interpretiert wenn eine der folgenden Bedingungen zutrifft.

MIME type: application/pkcs7-mime
Parameter: *
Dateierweiterung: *

MIME type: multipart/signed
Parameter: protocol="application/pkcs7-signature"
Dateierweiterung: *

MIME type: application/octet-stream
Parameter: *
Dateierweiterung:p7m p7s p7c p7z

6. Unterschiede zu früheren S/MIME Versionen

S/MIME Version 3.1 sollte grösstmögliche Kompatibilität bieten, aus diesem Grund sind einige Parameter nur noch erlaubt um Kompatibilität mit früheren Versionen zu gewährleisten.

- RSA MUSS seit Version 3.0, Diffie-Hellmann SOLLTE weiterhin als Public Key Algorithmus implementiert sein
- AES Verschlüsselung (symmetrisch) SOLLTE implementiert seit
- RSA SOLLTE als Public Key Algorithmus für digitale Unterschriften verwendet werden
- CompressedData CMS Typ ist erlaubt

S/MIME Version 2 ist in RFC 2311 bis RFC 2315 beschrieben.

S/MIME Version 3 ist in RFC 2630 bis RFC 2634 beschrieben.

7. Literaturliste

[CERT31] "S/MIME Version 3.1 Certificate Handling", Internet Draft draft-ietf-smime-rfc2632bis

[CHARSETS] Character sets assigned by IANA. See <ftp://ftp.isi.edu/in-notes/iana/assignments/character-sets>.

[CMS] "Cryptographic Message Syntax", RFC 3369

[CMSAES] "Use of the AES Encryption Algorithm in CMS", RFC 3565

[CMSALG] "Cryptographic Message Syntax (CMS) Algorithms", RFC 3370

[ESS] "Enhanced Security Services for S/MIME", RFC 2634

[MIME-SPEC] The primary definition of MIME. "MIME Part 1: Format of Internet Message Bodies", RFC 2045;
"MIME Part 2: Media Types", RFC 2046;
"MIME Part 3: Message Header Extensions for Non-ASCII Text", RFC 2047;
"MIME Part 4: Registration Procedures", RFC 2048;
"MIME Part 5: Conformance Criteria and Examples", RFC 2049

[MIME-SECURE] "Security Multiparts for MIME: Multipart/Signed and Multipart/Encrypted", RFC 1847

[MUSTSHOULD] "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119

[X.509-88] CCITT. Recommendation X.509: The Directory – Authentication Framework. 1988.

[PKCS-7] "PKCS #7: Cryptographic Message Syntax Version 1.5", RFC 2315

[RANDOM] "Randomness Recommendations for Security", RFC 1750

[SMIMEV2] "S/MIME Version 2 Message Specification", RFC 2311

[X.509] ITU-T Recommendation X.509 (1997 E): Information Technology –
Open Systems Interconnection – The Directory: Authentication Framework, June 1997.