S/MIME Working Group
Internet Draft
expires in six months

D. T. Davis
Curl Corporation

August 2001

Sender Authentication and the Surreptitious Forwarding Attack in CMS and S/MIME

<draft-ietf-smime-sender-auth-00.txt>

Status of this Memo

Copyright Notice

Abstract

By default, a CMS signed-and-encrypted document or message authenticates only the document's originator, and not the person who encrypted the document. This subtle limitation exposes CMS and S/MIME signed-and-encrypted data to "surreptitious forwarding." Secure-messaging standards have treated surreptitious forwarding as an insoluble problem of user carelessness, and have long accepted the risk of this attack. This document discusses easy cryptographic remedies for this attack, suitable for incorporation into the

CMS and S/MIME specifications. This document is an abridgement of [U2001].

1. Introduction

This document describes the repair of CMS's and S/MIME's [RFC2630, RFC2633, RFC2634] vulnerability to "surreptitious forwarding" attacks. In this attack, Alice signs and encrypts data for Bob's eyes, and Bob re-encrypts and forwards Alice's signed data to Charlie, making the document seem to come directly from Alice to Charlie. Charlie won't be fooled if he fully understands CMS's cryptographic semantics, so many document-security workers dismiss this attack as an insignificant user problem, but two points contradict
this dismissal:

* Usability: Most modern users of secure-mail are too unsophisticated about security to detect and reject surreptitiously forwarded documents, or to protect their own documents from this attack;
* Ease of repair: The CMS and S/MIME specifications already provide optional, lightweight machinery that can fix the problem. All that's needed is to stipulate that all sending and receiving clients must address the problem.

Because the security semantics of public-key operations can be subtle, users and programmers prefer to think about all cryptographic security by analogy with familiar symmetric-key "secret codes." In contrast, designers of secure-messaging protocols have relied heavily on simple asymmetric encryption and signing, rather naively combined. Naive sign & encrypt has surprisingly different security semantics from symmetric encryption, but the difference is subtle, perhaps too subtle for non-specialist users and programmers to grasp. Indeed, for senders, sign-and-encrypt guarantees the same security properties as symmetric-key cryptography gives. With both types of crypto, the sender is sure that:

* The recipient knows who wrote the message; and
* Only the recipient can decrypt the message.

The difference appears only in the recipient's security guarantees: the recipient of a symmetric-key ciphertext knows who sent it to him, but a "simple sign & encrypt" recipient knows only who wrote the message, and has no assurance about who encrypted it. This is because naive sign & encrypt is vulnerable to "surreptitious forwarding," but symmetric-key encryption is not. Since users always will assume that sign & encrypt is similar to symmetric-key "secret codes," they will tend to trust naive sign & encrypt too much.

The standards that exist for simple file-encryption, chiefly PKCS#7 [RFC2315] and S/MIME, tend to allow secure Sign & Encrypt implementations (i.e., such as would prevent surreptitious forwarding), but surprisingly, these file-security standards don't require fully-secure implementation and operation. Similarly, some important new security standards, such as the XML security

specifications [XML-Sig,XML-Enc], offer only low-level "toolbox" APIs. Too often, both the established standards and the new ones allow insecure yet compliant implementations. Application programmers need more security guidance than these "toolbox" APIs offer, in order to build effective security into their applications. Without such guidance, programmers tend to suppose incorrectly that simply signing and then encrypting a message or a file will give good security.

1.1 Notation

In this document, our notation pretends that the asymmetric-key cryptosystem behaves similarly to RSA [RSA], so that a signature is a private-key encryption via a modular exponentiation:

* {msg}^x denotes asymmetric encryption or decryption with the key x;
* Upper-case letters denote public keys, and
* Lower-case letters donote private keys,

Thus, "A" is Alice's public key, "a" is her private key, {msg}^a is Alice's signed message, and {msg}^A is an encrypted ciphertext that only Alice can read. This notational device doesn't sacrifice generality, because in this document, we don't actually use RSA's unique properties as a cryptosystem. (Static-Static Diffie-Hellman [RFC 2631] does not conform to this model, because encryption and authentication are not separate operations in that variety of Diffie-Hellman-based messaging. However, S/MIME with Static-Static Diffie-Hellman key-agreement is not vulnerable to the attack we discuss in this document. See sec. 2.1, "Message Context.")

1.2 Surreptitious Forwarding Attack

Why is naive Sign & Encrypt insecure? Most simply, S&E is vulnerable to "surreptitious forwarding:" Alice signs & encrypts for Bob's eyes, but Bob re-encrypts Alice's signed message for Charlie to see. In the end, Charlie believes Alice wrote to him directly, and can't detect Bob's subterfuge. Bob might do this just to embarass Alice, or Charlie, or both. For example, suppose Alice wants to have dinner with Bob. She writes a note, "Meet me at the Iron Gate at 7:00 tonight", signs it, encrypts it for Bob, and sends it to Bob. Bob then decrypts the message, validates the signature, and reads the message. Bob does not want to have dinner with Alice tonight, so for fun he encrypts the signed message for Charlie, and sends it to Charlie:

```
A ---> B   :  {{"Meet me at 7"}^a }^B          (1)

B ---> C   :  {{"Meet me at 7"}^a }^C          (2)
```

Charlie then decrypts the message, validates Alice's signature, and reads her message. Charlie and Alice meet for dinner, but both are surprised by the resulting confusion; because each believed that Alice's message was "secure." Thus, even if Alice doesn't care whether Bob divulges the message,

she and others may be misled if Bob is able to forward her signature surreptitiously.

The problem is that Alice and Charlie both tacitly and incorrectly assumed that to sign-then-encrypt a message is an irreducible or atomic operation. Thus, in effect, both supposed that Alice's encryption not only kept her message private, but that it also authenticated Alice's recipient-list. Neither knew that Alice needed to sign her intended recipient's name inside her message, because both assumed that the encryption layer expressed Alice's intent. But, if Alice had simply said, "Bob, please meet me at the Iron Gate at 7:00 tonight," naive S&E would have blocked Bob's replay. If every user could be relied upon to take proper care when he constructs and interprets signed and encrypted messages, then the current naive Sign & Encrypt standards would need no repair. But among today's nontechnical network users, when Charlie gets Alice's message via Bob, Charlie very likely will assume that Alice sent it directly.

2. Problem Scope

Why is this old and easy problem worth discussing at this late date? Though designing a secure Sign & Encrypt protocol is easy for cryptographers, it's a different class of engineer who faces this problem nowadays. Application programmers have to rely on crypto vendors and crypto standards, in order to learn how to write crypto applications. Unfortunately, the vendors and standards have left untended a big gap in their support for application programmers. Current security standards don't give application programmers a simple recipe for file-encryption problems.

2.1 Message Context

Which CMS applications are vulnerable to surreptitious forwarding? Message-bodies sometimes include or bind the recipents' names by default, for various reasons, and so are protected from this attack:

Protected:

* Personal messages that bear a "Dear Bob" salutation are automatically and naturally protected from surreptitious forwarding.
* Similarly, but less reliably, a personal message is protected if it refers to privately-shared context, such as a previous communication or experience: "...When Mom gave you the lamp,...".
* Some secure-mail application protocols, such as MIME-encapsulated EDI [RFC1767], are well-protected against this attack, because the message-formats are rich in redundant names, and already tend to carry all participants' identifiers in each message.
* S/MIME with Static-Static Diffie-Hellman key-management [RFC 2631] protects users from this attack, because long-lived certificates tie both names, sender's and recipient's, to the symmetric key that protects the message-body.

Vulnerable:

* Surreptitious forwarding works readily with personal text messages, because senders commonly omit "Dear Bob" salutations nowadays.
* Similarly, if Alice forwards a MIME attachment without an accompanying textual message-body, her signature often won't encompass any reference to her intended audience.
* If an automated application protocol uses S/MIME for secure transport, if the application's message-format omits the recipient's identifier, and if some participants can act as both client and server, then such a dual-role participant may be able to perform surreptitious forwarding.
* The most commonly-deployed S/MIME key-management techniques, such as RSA and Ephemeral-Static Diffie-Hellman key-agreement [RFC2631], afford no protection against this attack.

Surreptitious forwarding is only one of a class of similar "message context" attacks on S/MIME, in which an interloper can replace various unsigned mail-header contents. Surreptitious forwarding is more easily blocked than most of these other attacks, though. For example, just as recipients shouldn't rely on unsigned To-lists, they also shouldn't trust a message's "Date Sent" timestamp, even if the sender has signed the timestamp. In this case, though, the recipient needs to see that a trusted third-party has signed the timestamp, which might unduly complicate a secure-mail protocol.

2.2 Cryptographic Protocols

Secure session protocols have attracted a lot of research attention, and several effective session-security protocols have been standardized. Naive Sign & Encrypt is not a problem in session security, because Session-security standards, like Kerberos [RFC1510], TLS [RFC2246], and SET [SET], give straightforward, out-of-the-box solutions. For files and one-way messaging, though, current security standards give developers only a kind of "toolbox" support, with a variety of security options, but with no clear or firm guidance about how to combine the options to make Sign & Encrypt an effective security solution. Providing only toolbox-style cryptographic protocols is appropriate for a low-level mechanism like IPSEC [IPSEC], but for user-visible applications like secure e-mail, programmers need "turnkey" cryptography, not only cryptographic toolkits. The demand for simple file-security and message-security is big and growing, so widespread use of these naive Sign & Encrypt security models may lead to widespread exposures.

2.3 Social Scope

Increasingly, secure applications are being designed and built by application programmers, not by cryptographers. Several factors have obliged mainstream application programmers to undertake public-key protocol design:

* Commercial PKI is in widespread deployment;
* Secure networking standards don't address file-encryption;
* Demand for cryptographers greatly exceeds the supply.

So, when application programmers need file-encryption help, they can seek help from crypto vendors and from crypto standards. Unfortunately, the vendors and the standards both offer either high-level secure connections, or low-level "toolkit" mechanisms. Neither offering makes file-encryption easy. The available standards specifications for file-encryption intend to support security applications, but the specifications tend to standardize only low-level APIs for cryptographic primitives, so as to leave designers as much flexibility as possible.

3. Method of Protection

S/MIME is flexible enough to allow the Sign & Encrypt defect to be repaired, but the repair is currently only an option. Any protection against surreptitious forwarding will only be effective, if all sending clients are required to include such protection, and if all recipient clients demand such protection, on every file or message they handle.

Like PKCS#7, CMS provides for "signed attributes," which offer a way to prevent crypto-layer alterations:

* Every sender should include a signed attribute, which includes the contents of the "To:", and "Cc:" lists. (Protecting the privacy of a "Bcc:" list is more complicated).
* Every recipient's cryptographic software has to know how to process and interpret such an attribute automatically,
* Every recipient's software must recognize that when a message lacks the signed attribute listing the intended recipients, this lack consitutes a security failure.

(For secure e-mail applications, the processing of mailing-lists entails some extra cryptographic details, which this document will not describe.)

So, upon decryption of a signed and encrypted document, the recipient's decryption software should:

* Validate the signed content, including the signed attribute listing the intended recipients;
* Compare the recipient's name (as found in the recipient's own public-key certificate) to the signed attribute listing the intended recipients;
* If the recipient's name doesn't appear in the signed attribute listing the intended recipients, then the recipient's software must inform the recipient that there is a security problem: the author apparently didn't intend the document for this recipient's eyes.
* For human recipients, the software might raise this alarm as a console error, as a "security alert" dialog-box, or by other GUI signifiers.

* For automated recipients, the software might log an error-message, or mark the received data as unreliable, or raise a security exception-condition of some sort.

In some such way, the recipient's decryption software must inform the recipient whether the "secure" document truly carried a satisfactorily consistent signed-and-encrypted envelope. Once the end-user can determine whether the document's encryption-layer has been altered, no attacker can profit by replacing the outer crypto layers.

The CMS, S/MIME, and PKCS specifications do not yet stipulate or suggest such a signed attribute listing the intended recipients, though the CMS specification does suggest other signed attributes.

3.1 Sender Authentication and List Servers

In S/MIME applications, secure mail-list servers sometimes mediate between sender and receiver. This subsection sketches how clients and servers should process the sender's signed attribute listing the intended recipients, so as to prevent surreptitious forwarding attacks.

When Alice sends her message to a secure mail-list, a few complications arise, but these interact in a straightforward way with Alice's signed attribute listing the intended recipients:

* The mail-list server must look for its own name in Alice's signed attribute listing her intended recipients;
* The server will replace Alice's encryption envelope with another envelope, using Bob's public key;
* The server adds to Alice's message a "triple-wrapped" signature, which includes a mail list expansion history [RFC2634];
* Thus, Bob's secure-mail client will find a superficial mismatch between his name and Alice's signed attribute listing the intended recipients.

In notation:

A --> S: {{"To: M-L", "msg"}^a }^S

S --> B: {{{"To: M-L", "msg"}^a }^B , "S served M-L to B"}^s

In order to process incoming mail-list messages properly, Bob's secure-mail client checks all of the message's names and signatures, to verify that:

* Alice originally sent her msg to the M-L list, and that
* The M-L list-server S affirms that it has translated Alice's encryption, for Bob's eyes;

Alice implicitly affirms, by signing the name "M-L" into her signed attribute listing her intended recipients, that she trusts that list's server S to translate her encryption envelope. Bob's secure-mail client

searches the message for a chain of signatures and names, connecting Alice's name to Bob's:

A --> M-L --> S --> B

If Bob's secure-mail client doesn't find a such correct chain of names and signatures, the client's user-interface should flag an authentication error, for Bob to see.

3.2 Lack of Protection in CMS and S/MIME

The S/MIME specification acknowledges some shortcomings of the Sign & Encrypt construct, but the S/MIME specification fails to discuss the main defect. Further, the document tells implementors nothing about how to shore up Sign & Encrypt. Instead, the S/MIME specification merely cautions users and implementors not to over-rely on a message's security:

> 1. "An S/MIME implementation MUST be able to receive and process arbitrarily nested S/MIME within reasonable resource limits of the recipient computer.
>
> 2. "It is possible to either sign a message first, or to envelope the message first. It is up to the implementor and the user to choose. When signing first, the signatories are then securely obscured by the enveloping. When enveloping first, the signatories are exposed, but it is possible to verify signatures without removing the enveloping. This may be useful in an environment where automatic signature verification is desired, as no private key material is required to verify a signature.
>
> 3. "There are security ramifications to choosing whether to sign first or to encrypt first. A recipient of a message that is encrypted and then signed can validate that the encrypted block was unaltered, but cannot determine any relationship between the signer and the unencrypted contents of the message. A recipient of a message that is signed-then-encrypted can assume that the signed message itself has not been altered, but that a careful attacker may have changed the unauthenticated portion of the encrypted message" [sic].
>
> - [RFC2633] Sec. 3.5, "Signing and Encrypting."

This excerpt is the S/MIME specification's only discussion of Sign & Encrypt's limitations. Several features in the excerpt deserve comment:

* Paragraph 2 presents the security issues as a tradeoff between confidentiality and ease of verification;
* Paragraph 3 hints that an attacker can replace the external signature in an encrypted-then-signed message,
* But there's no mention that sign-then-encrypt is vulnerable to surreptitious forwarding, by replacement of the outermost encryption

layer. (In paragraph 3, "unauthenticated portion" seems to refer not to the unauthenticated ciphertext, but to unauthenticated plaintext.)
* The excerpt presents only the choice between signing first and encrypting first. There's no mention of repairing either option's defects.

4. Analysis

We propose that users of file-security and mail-security need simple security semantics, and that symmetric-key semantics are sufficient for most users and most applications' needs. Further, symmetric-key semantics are natural and easy for unsophisticated users to understand.

In this section, we present three overlapping views of what's wrong with naive Sign & Encrypt. Then, we summarize and discuss several arguments in defense of the naive Sign & Encrypt standards.

4.1 Asymmetric Security Guarantees

At first glance, naive Sign & Encrypt seems quite secure, because message-sender Alice gets the security guarantees she needs: her signature proves her authorship, and she knows who can read the message. The message-receiver, Bob, doesn't get the same guarantees, though. He knows who wrote the message, but he doesn't know who encrypted it, and therefore doesn't know who else besides Alice has read the message. Note the asymmetry:

* When A sends B a signed & encrypted message, A knows that only B can read it, because A trusts B not to divulge the message, but -
* When B receives A's signed & encrypted message, B can't know how many hands it has passed through, even if B trusts A to be careful.

Seen this way, the flaw in naive Sign & Encrypt is that B gets no proof that it was A who encrypted the message. In hindsight, this is obvious: public key algorithms usually don't automatically authenticate the encryptor of a message.

Certainly, in some applications, it's neither necessary nor feasible to give a recipient any assurance that only the sender has seen the message-plaintext. Thus, for example, mail-security applications do need the flexibility to waive full end-to-end symmetric-key semantics. But, whenever possible, and by default, mail- and file-security applications should give end-users easy-to-understand security guarantees.

4.2 Symmetric-Key Semantics

Users tacitly expect public-key file-encryption to offer the same security semantics that a symmetric key offers. Thus, another way to describe the Sign & Encrypt problem is that whether signing or encryption is applied first, naive Sign & Encrypt fails to duplicate the security meaning of a

symmetric-key ciphertext. When B receives a symmetric-key ciphertext from A, B can safely assume that:

* A sent the message,
* No-one else has seen the plaintext,
* A intended B to receive the plaintext.

With naive Sign & Encrypt, these assumptions can break down, because the recipient may have to rely on the crypto layer to supply the intended recipients' names. That is, the problem arises when:

* The message plaintexts don't mention the sender's and recipient's names;
* The sender's and recipient's names are important for understanding the message or its security import;
* The recipient assumes that the signer encrypted the message.

Under these conditions, an attacker can successfully and surreptitiously forward a naively signed and encrypted message.

4.3 Sign & Encrypt Must Cross-Refer

It is a common mistake to treat public-key encryption and digital signatures as if they are fully independent operations. This independence assumption is convenient for writing standards and for writing software, but it is cryptographically incorrect. When independent operations are applied one on top of another, then the outermost crypto layer can undetectably be replaced, and security is weakened. Though signing and encryption are not independent of one another, the CMS and S/MIME standards treat crypto operations as independent content-transformations, converting "content" to "content." Conceptually, this makes it easy for users and programmers to layer crypto operations in arbitrary depth and in arbitrary order. By this device, the standards authors sought to avoid constraining application developers' designs.

With such independent operations, though, it's hard to fulfill the recipient's security expectations. In order to work properly together, the signature layer and the encryption layer actually must refer to one another, so as to achieve basic symmetric-key security guarantees that users expect. The recipient needs proof that the signer and the encryptor were the same person, which necessarily entails either signing the recipient's identifier (in Sign & Encrypt), or encrypting the signer's identifier (in Encrypt & Sign). Once such cross-references are in place, an attacker can't remove and replace the outermost layer, because the inner layer's reference will reveal the alteration. For example, one repair for Sign & Encrypt puts the decrypting recipient's name inside the signed plaintext message:

```
A ---> B  :   {{"To: Bob", msg}^a }^B
```

This repair is straightforward for a user or an implementor to do, but it's hard for a standards specification to stipulate that different crypto

operations must be tied together like this, without breaking the full generality of the content-transformation model.

4.4 Trust and Risk

A common defense of naive Sign & Encrypt is that users have to be careful about whom they trust, or equivalently, that users should carefully assess risk when putting sensitive material under cryptographic protection. In this view, the recipient of a signed and encrypted message should not invest more trust in the message than the technology and the sender's reputation can support. This argument seems very plausible, but it turns out not to address the problems with naive Sign & Encrypt.

B has no way to gauge the risk that the message has been divulged to people unknown to A and B. To gauge the risk, B would have to know how trustworthy are the people who have surreptitiously forwarded the message along from A towards B. Thus, in general, one can't assess the privacy of a decrypted plaintext, and shouldn't trust its privacy, unless one knows who encrypted it. In sum: if we accept the Trust and Risk argument, then the encryption step of Sign & Encrypt is quite pointless from the receiver's point-of-view.

4.5 Security and Ease-of-Use

Another common defense of S/MIME's naive Sign & Encrypt is that "Users shouldn't trust unsigned information" about the signer's intended recipients. This argument misses the point of S/MIME's weakness, by supposing that users are over-relying on the unsigned SMTP header to identify the sender's intended recipients. The users' mistake is more subtle, though; they're over-relying on the encrypting-key's certificate, as if it were a secure record of the sender's intended recipient.

It's unrealistic to expect today's users to catch such a subtle point. When **X.509 [X.509], PEM [RFC1421], and S/MIME [RFC2633] were designed, PKI users** were expected to be system administrators and other fairly sophisticated users; now, though, with the modern Internet and with electronic commerce in play,we can't expect most users to understand any cryptographic nuances at all.

A similar defense of the defective secure mail standards is that the specifications aren't actually broken, because "Applications can and should put names into the content, if that's what they want." This argument assumes that application programmers shouldn't try to incorporate cryptographic
security into programs in the first place, unless they understand security and cryptography well enough to design security protocols. Further, the argument insists that no security standard can be so complete as to prevent ignorant programmers from "shooting themselvers in the foot."

A ready answer to this argument is "SSL." The SSL specification gives fairly complete security, out-of-the-box. Further, non-specialist

programmers are able to set up secure SSL connections for their applications, without having to patch the SSL protocol on their own.

5. Security Considerations

This entire document addresses security considerations in the implementation of Cryptographic Message Syntax.

6 Acknowledgements

I have had profitable discussions about these ideas with many expert critics: Martin Abadi, Ross Anderson, Marc Branchaud, Dave Carver, Dan Geer, Peter Gutmann, Philip Hallam-Baker, Paul Hoffman, Russ Housley, Steve Kent, Norbert Leser, John Linn, Ellen McDermott, Joseph Reagle, Ed Simon, Win Treese, Charlie Reitzel, Ralph Swick, and Henry Tumblin. I thank all of these people for their patient attention.

This document is an abridgement of [U2001]. Excerpts from that publication are reprinted here with the permission of the USENIX Association.

7. References

[IPSEC] B. Fraser, T.Y. Ts'o, J. Schiller, M. Leech, "IP Security Protocol (IPSEC) Charter," http://www.ietf.org/html.charters/ipsec-charter.html .

[RFC2315] B. Kaliski, "PKCS#7: Cryptographic Message Syntax," Version 1.5, March 1998. http://www.imc.org/rfc2315

[RFC1421] J. Linn, Internet RFC 1421, "Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication Procedures", February 1993. ftp://ftp.isi.edu/in-notes/rfc1421.txt

[RFC1510] C. Neuman and J. Kohl, Internet RFC 1510 "The Kerberos Network Authentication Service (V5)", September 1993. ftp://ftp.isi.edu/in-notes/rfc1510.txt

[RFC1767] D. Crocker, Internet RFC 1767 "MIME Encapsulation of EDI Objects", March 1995. ftp://ftp.isi.edu/in-notes/rfc1767.txt

[RFC2246] T. Dierks and C. Allen, Internet RFC 2246 "The TLS Protocol Version 1.0," January 1999. ftp://ftp.isi.edu/in-notes/rfc2246.txt

[RFC2630] R. Housley, Internet RFC 2630 "Cryptographic Message Syntax," June 1999. ftp://ftp.isi.edu/in-notes/rfc2630.txt

[RFC2631] E. Rescorla, Internet RFC 2631 "Diffie-Hellman Key Agreement Method," June 1999. ftp://ftp.isi.edu/in-notes/rfc2631.txt

[RFC2633] B. Ramsdell, Internet RFC 2633 "S/MIME Version 3 Message Specification," June 1999. ftp://ftp.isi.edu/in-notes/rfc2633.txt

[RFC2634] P. Hoffman, Internet RFC 2634 "Enhanced Security Services for S/MIME," June 1999. ftp://ftp.isi.edu/in-notes/rfc2634.txt

[RSA] R. Rivest, A, Shamir, L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," Comm. ACM, v. 21, 2, Feb. '78, pp. 120-126.

[SET] Visa International and MasterCard, "Secure Electronic Transactions Protocol Specification," http://www.setco.org/set_specifications.html .

[U2001] Don Davis, "Defective Sign & Encrypt in S/MIME, PKCS#7, MOSS, PEM, PGP, and XML." Proc. Usenix Tech. Conf. 2001, edited by Yoonho Park (Boston, June 25-30, 2001). http://world.std.com/~dtd/sign_encrypt/sign_encrypt7.html

[X.509] International Telegraph and Telephone Consultative Committee (CCITT). Recommendation X.509: The Directory - Authentication Framework. In Data Communications Network Directory, Recommendations X.500-X.521, pp. 48-81. Vol. 8, Fascicle 8.8 of CCITT Blue Book. Geneva: International Telecommunication Union, 1989.
R. Housley, W. Ford, W. Polk, D. Solo, Internet RFC 2459, "Internet X.509 Public Key Infrastructure Certificate and CRL Profile," January 1999. http://www.imc.org/rfc2459

[XML-Enc] J. Reagle, "XML Encryption Requirements," W3C Working Draft 2001-April-20, http://www.w3.org/TR/2001/WD-xml-encryption-req-20010420 .

[XML-Sig] D. Eastlake, J. Reagle, and D. Solo (Editors), "XML-Signature Syntax and Processing: W3C Working Draft 18-September-2000," http://www.w3.org/TR/xmldsig-core/

8. Author's Address

Donald T. Davis, Jr.
Curl Corp.
400 Tech Square
Cambridge, MA 02139

Email: dtd@world.std.com, ddavis@curl.com

9. Intellectual Property Rights

The IETF takes no position regarding the validity or scope of any

intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on the IETF's procedures with respect to rights in standards-track and standards-related documentation can be found in BCP-11. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementors or users of this specification can be obtained from the IETF Secretariat.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this standard. Please address the information to the IETF Executive Director.

10. Full Copyright Statement