Internet Engineering Task Force (IETF)
Request for Comments: 6187
Category: Standards Track
ISSN: 2070-1721

K. Igoe
National Security Agency
D. Stebila
Queensland University of Technology

March 2011

X.509v3 Certificates for Secure Shell Authentication

Abstract

X.509 public key certificates use a signature by a trusted certification authority to bind a given public key to a given digital identity. This document specifies how to use X.509 version 3 public key certificates in public key algorithms in the Secure Shell protocol.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc6187.

Table of Contents

1. Introduction

There are two Secure Shell (SSH) protocols that use public key cryptography for authentication. The Transport Layer Protocol, described in [RFC4253], requires that a digital signature algorithm (called the "public key algorithm") MUST be used to authenticate the server to the client. Additionally, the User Authentication Protocol described in [RFC4252] allows for the use of a digital signature to authenticate the client to the server ("publickey" authentication).

In both cases, the validity of the authentication depends upon the strength of the linkage between the public signing key and the identity of the signer. Digital certificates, such as those in X.509 version 3 (X.509v3) format [RFC5280], are used in many corporate and government environments to provide identity management. They use a chain of signatures by a trusted root certification authority and its intermediate certificate authorities to bind a given public signing key to a given digital identity.

The following public key authentication algorithms are currently available for use in SSH:

Algorithm	Reference
ssh-dss	[RFC4253]
ssh-rsa	[RFC4253]
pgp-sign-dss	[RFC4253]
pgp-sign-rsa	[RFC4253]
ecdsa-sha2-*	[RFC5656]

Since Pretty Good Privacy (PGP) has its own method for binding a public key to a digital identity, this document focuses solely upon the non-PGP methods. In particular, this document defines the following public key algorithms, which differ from the above solely in their use of X.509v3 certificates to convey the signer's public key.

Algorithm
x509v3-ssh-dss
x509v3-ssh-rsa
x509v3-rsa2048-sha256
x509v3-ecdsa-sha2-*

Public keys conveyed using the x509v3-ecdsa-sha2-* public key algorithms can be used with the ecmqv-sha2 key exchange method.

Implementation of this specification requires familiarity with the Secure Shell protocol [RFC4251] [RFC4253] and X.509v3 certificates [RFC5280]. Data types used in describing protocol messages are defined in Section 5 of [RFC4251].

This document is concerned with SSH implementation details; specification of the underlying cryptographic algorithms and the handling and structure of X.509v3 certificates is left to other

standards documents, particularly [RFC3447], [FIPS-186-3], [FIPS-180-2], [FIPS-180-3], [SEC1], and [RFC5280].

An earlier proposal for the use of X.509v3 certificates in the Secure Shell protocol was introduced by O. Saarenmaa and J. Galbraith; while this document is informed in part by that earlier proposal, it does not maintain strict compatibility.

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 RFC 2119 [RFC2119].

2. Public Key Algorithms Using X.509 Version 3 Certificates

This document defines the following new public key algorithms for use in the Secure Shell protocol: x509v3-ssh-dss, x509v3-ssh-rsa, x509v3-rsa2048-sha256, and the family of algorithms given by x509v3-ecdsa-sha2-*. In these algorithms, a public key is stored in an X.509v3 certificate. This certificate, a chain of certificates leading to a trusted certificate authority, and optional messages giving the revocation status of the certificates are sent as the public key data in the Secure Shell protocol according to the format in this section.

2.1. Public Key Format

The reader is referred to [RFC5280] for a general description of X.509 version 3 certificates. For the purposes of this document, it suffices to know that in X.509 a chain or sequence of certificates (possibly of length one) allows a trusted root certificate authority and its intermediate certificate authorities to cryptographically bind a given public key to a given digital identity using public key signatures.

For all of the public key algorithms specified in this document, the key format consists of a sequence of one or more X.509v3 certificates followed by a sequence of 0 or more Online Certificate Status Protocol (OCSP) responses as in Section 4.2 of [RFC2560]. Providing OCSP responses directly in this data structure can reduce the number of communication rounds required (saving the implementation from needing to perform OCSP checking out-of-band) and can also allow a client outside of a private network to receive OCSP responses from a server behind a firewall. As with any use of OCSP data, implementations SHOULD check that the production time of the OCSP response is acceptable. It is RECOMMENDED, but not REQUIRED, that implementations reject certificates for which the certificate status is revoked.

The key format has the following specific encoding:

```
  string  "x509v3-ssh-dss" / "x509v3-ssh-rsa" /
          "x509v3-rsa2048-sha256" / "x509v3-ecdsa-sha2-[identifier]"
  uint32  certificate-count
  string  certificate[1..certificate-count]
  uint32  ocsp-response-count
  string  ocsp-response[0..ocsp-response-count]
```

In the figure above, the string [identifier] is the identifier of the elliptic curve domain parameters. The format of this string is specified in Section 6.1 of [RFC5656]. Information on the REQUIRED and RECOMMENDED sets of elliptic curve domain parameters for use with this algorithm can be found in Section 10 of [RFC5656].

Each certificate and ocsp-response MUST be encoded as a string of octets using the Distinguished Encoding Rules (DER) encoding of Abstract Syntax Notation One (ASN.1) [ASN1]. An example of an SSH key exchange involving one of these public key algorithms is given in Appendix A.

Additionally, the following constraints apply:

- The sender’s certificate MUST be the first certificate and the public key conveyed by this certificate MUST be consistent with the public key algorithm being employed to authenticate the sender.

- Each following certificate MUST certify the one preceding it.

- The self-signed certificate specifying the root authority MAY be omitted. All other intermediate certificates in the chain leading to a root authority MUST be included.

- To improve the chances that a peer can verify certificate chains and OCSP responses, individual certificates and OCSP responses SHOULD be signed using only signature algorithms corresponding to public key algorithms supported by the peer, as indicated in the server_host_key_algorithms field of the SSH_MSG_KEXINIT packet (see Section 7.1 of [RFC4253]). However, other algorithms MAY be used. The choice of signature algorithm used by any given certificate or OCSP response is independent of the signature algorithms chosen by other elements in the chain.

- Verifiers MUST be prepared to receive certificate chains and OCSP responses that use algorithms not listed in the server_host_key_algorithms field of the SSH_MSG_KEXINIT packet, including algorithms that potentially have no Secure Shell

equivalent. However, peers sending such chains should recognize that such chains are more likely to be unverifiable than chains that use only algorithms listed in the server_host_key_algorithms field.

- o There is no requirement on the ordering of OCSP responses. The number of OCSP responses MUST NOT exceed the number of certificates.

Upon receipt of a certificate chain, implementations MUST verify the certificate chain according to Section 6.1 of [RFC5280] based on a root of trust configured by the system administrator or user.

Issues associated with the use of certificates (such as expiration of certificates and revocation of compromised certificates) are addressed in [RFC5280] and are outside the scope of this document. However, compliant implementations MUST comply with [RFC5280]. Implementations providing and processing OCSP responses MUST comply with [RFC2560].

When no OCSP responses are provided, it is up to the implementation and system administrator to decide whether or not to accept the certificate. It may be possible for the implementation to retrieve OCSP responses based on the id-ad-ocsp access description in the certificate's Authority Information Access data (Section 4.2.2.1 of [RFC5280]). However, if the id-ad-ocsp access description indicates that the certificate authority employs OCSP, and no OCSP response information is available, it is RECOMMENDED that the certificate be rejected.

[RFC5480] and [RFC5758] describe the structure of X.509v3 certificates to be used with Elliptic Curve Digital Signature Algorithm (ECDSA) public keys. [RFC3279] and [RFC5280] describe the structure of X.509v3 certificates to be used with RSA and Digital Signature Algorithm (DSA) public keys. [RFC5759] provides additional guidance for ECDSA keys in Suite B X.509v3 certificate and certificate revocation list profiles.

2.2. Certificate Extensions

Certificate extensions allow for the specification of additional attributes associated with a public key in an X.509v3 certificate (see Section 4.2 of [RFC5280]). The KeyUsage and ExtendedKeyUsage extensions may be used to restrict the use of X.509v3 certificates in the context of the Secure Shell protocol as specified in the following sections.

2.2.1. KeyUsage

The KeyUsage extension MAY be used to restrict a certificate's use. In accordance with Section 4.2.1.3 of [RFC5280], if the KeyUsage extension is present, then the certificate MUST be used only for one of the purposes indicated. There are two relevant keyUsage identifiers for the certificate corresponding to the public key algorithm in use:

- If the KeyUsage extension is present in a certificate for the x509v3-ssh-dss, x509v3-ssh-rsa, x509v3-rsa2048-sha256, or x509v3-ecdsa-sha2-* public key algorithms, then the digitalSignature bit MUST be set.

- If the KeyUsage extension is present in a certificate for the ecmqv-sha2 key exchange method, then the keyAgreement bit MUST be set.

For the remaining certificates in the certificate chain, implementations MUST comply with existing conventions on KeyUsage identifiers and certificates as in Section 4.2.1.3 of [RFC5280].

2.2.2. ExtendedKeyUsage

This document defines two ExtendedKeyUsage key purpose IDs that MAY be used to restrict a certificate's use: id-kp-secureShellClient, which indicates that the key can be used for a Secure Shell client, and id-kp-secureShellServer, which indicates that the key can be used for a Secure Shell server. In accordance with Section 4.2.1.12 of [RFC5280], if the ExtendedKeyUsage extension is present, then the certificate MUST be used only for one of the purposes indicated. The object identifiers of the two key purpose IDs defined in this document are as follows:

- id-pkix OBJECT IDENTIFIER ::= { iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5) pkix(7) }

- id-kp OBJECT IDENTIFIER ::= { id-pkix 3 } -- extended key purpose identifiers

- id-kp-secureShellClient OBJECT IDENTIFIER ::= { id-kp 21 }

- id-kp-secureShellServer OBJECT IDENTIFIER ::= { id-kp 22 }

3. Signature Encoding

Signing and verifying using the X.509v3-based public key algorithms specified in this document (x509v3-ssh-dss, x509v3-ssh-rsa, x509v3-ecdsa-sha2-*) is done in the analogous way for the corresponding non-X.509v3-based public key algorithms (ssh-dss, ssh-rsa, ecdsa-sha2-*, respectively); the x509v3-rsa2048-sha256 public key algorithm provides a new mechanism, similar to ssh-rsa, but has a different hash function and additional key size constraints. For concreteness, we specify this explicitly below.

3.1. x509v3-ssh-dss

Signing and verifying using the x509v3-ssh-dss key format is done according to the Digital Signature Standard [FIPS-186-3] using the SHA-1 hash [FIPS-180-2].

The resulting signature is encoded as follows:

```
string  "ssh-dss"
string  dss_signature_blob
```

The value for dss_signature_blob is encoded as a string containing r, followed by s (which are fixed-length 160-bit integers, without lengths or padding, unsigned, and in network byte order).

This format is the same as for ssh-dss signatures in Section 6.6 of [RFC4253].

3.2. x509v3-ssh-rsa

Signing and verifying using the x509v3-ssh-rsa key format is performed according to the RSASSA-PKCS1-v1_5 scheme in [RFC3447] using the SHA-1 hash [FIPS-180-2].

The resulting signature is encoded as follows:

```
string  "ssh-rsa"
string  rsa_signature_blob
```

The value for rsa_signature_blob is encoded as a string containing s (which is an integer, without lengths or padding, unsigned, and in network byte order).

This format is the same as for ssh-rsa signatures in Section 6.6 of [RFC4253].

3.3. x509v3-rsa2048-sha256

Signing and verifying using the x509v3-rsa2048-sha256 key format is performed according to the RSASSA-PKCS1-v1_5 scheme in [RFC3447] using the SHA-256 hash [FIPS-180-3]; RSA keys conveyed using this format MUST have a modulus of at least 2048 bits.

The resulting signature is encoded as follows:

```
string  "rsa2048-sha256"
string  rsa_signature_blob
```

The value for rsa_signature_blob is encoded as a string containing s (which is an integer, without lengths or padding, unsigned, and in network byte order).

Unlike the other public key formats specified in this document, the x509v3-rsa2048-sha256 public key format does not correspond to any previously existing SSH non-certificate public key format. The main purpose of introducing this public key format is to provide an RSA-based public key format that is compatible with current recommendations on key size and hash functions. For example, National Institute of Standards and Technology's (NIST's) draft recommendations on cryptographic algorithms and key lengths [SP-800-131] specify that digital signature generation using an RSA key with modulus less than 2048 bits or with the SHA-1 hash function is acceptable through 2010 and deprecated from 2011 through 2013, whereas an RSA key with modulus at least 2048 bits and SHA-256 is acceptable for the indefinite future. The introduction of other non-certificate-based SSH public key formats compatible with the above recommendations is outside the scope of this document.

3.4. x509v3-ecdsa-sha2-*

Signing and verifying using the x509v3-ecdsa-sha2-* key formats is performed according to the ECDSA algorithm in [FIPS-186-3] using the SHA2 hash function family [FIPS-180-3]. The choice of hash function from the SHA2 hash function family is based on the key size of the ECDSA key as specified in Section 6.2.1 of [RFC5656].

The resulting signature is encoded as follows:

```
string  "ecdsa-sha2-[identifier]"
string  ecdsa_signature_blob
```

The string [identifier] is the identifier of the elliptic curve domain parameters. The format of this string is specified in Section 6.1 of [RFC5656].

The ecdsa_signature_blob value has the following specific encoding:

```
mpint   r
mpint   s
```

The integers r and s are the output of the ECDSA algorithm.

This format is the same as for ecdsa-sha2-* signatures in Section 3.1.2 of [RFC5656].

4. Use in Public Key Algorithms

The public key algorithms and encodings defined in this document SHOULD be accepted any place in the Secure Shell protocol suite where public keys are used, including, but not limited to, the following protocol messages for server authentication and user authentication:

- in the SSH_MSG_USERAUTH_REQUEST message when "publickey" authentication is used [RFC4252]
- in the SSH_MSG_USERAUTH_REQUEST message when "hostbased" authentication is used [RFC4252]
- in the SSH_MSG_KEXDH_REPLY message [RFC4253]
- in the SSH_MSG_KEXRSA_PUBKEY message [RFC4432]
- in the SSH_MSG_KEXGSS_HOSTKEY message [RFC4462]
- in the SSH_MSG_KEX_ECDH_REPLY message [RFC5656]
- in the SSH_MSG_KEX_ECMQV_REPLY message [RFC5656]

When a public key from this specification is included in the input to a hash algorithm, the exact bytes that are transmitted on the wire must be used as input to the hash functions. In particular, implementations MUST NOT omit any of the chain certificates or OCSP responses that were included on the wire, nor change encoding of the certificate or OCSP data. Otherwise, hashes that are meant to be computed in parallel by both peers will have differing values.

For the purposes of user authentication, the mapping between certificates and user names is left as an implementation and configuration issue for implementers and system administrators.

For the purposes of server authentication, it is RECOMMENDED that implementations support the following mechanism mapping host names to certificates. However, local policy MAY disable the mechanism or MAY

impose additional constraints before considering a matching successful. Furthermore, additional mechanisms mapping host names to certificates MAY be used and are left as implementation and configuration issues for implementers and system administrators.

The RECOMMENDED server authentication mechanism is as follows. The subjectAlternativeName X.509v3 extension, as described in Section 4.2.1.6 of [RFC5280], SHOULD be used to convey the server host name, using either dNSName entries or iPAddress entries to convey domain names or IP addresses as appropriate. Multiple entries MAY be specified. The following rules apply:

- If the client's reference identifier (e.g., the host name typed by the client) is a DNS domain name, the server's identity SHOULD be checked using the rules specified in [RFC6125]. Support for the DNS-ID identifier type is RECOMMENDED in client and server software implementations. Certification authorities that issue certificates for use by Secure Shell servers SHOULD support the DNS-ID identifier type. Service providers SHOULD include the DNS-ID identifier type in certificate requests. The DNS-ID MAY contain the wildcard character '*' as the complete left-most label within the identifier.

- If the client's reference identifier is an IP address as defined by [RFC0791] or [RFC2460], the client SHOULD convert that address to the "network byte order" octet string representation and compare it against a subjectAltName entry of type iPAddress. A match occurs if the octet strings are identical for the reference identifier and any presented identifier.

5. Security Considerations

This document provides new public key algorithms for the Secure Shell protocol that convey public keys using X.509v3 certificates. For the most part, the security considerations involved in using the Secure Shell protocol apply, since all of the public key algorithms introduced in this document are based on existing algorithms in the Secure Shell protocol. However, implementers should be aware of security considerations specific to the use of X.509v3 certificates in a public key infrastructure, including considerations related to expired certificates and certificate revocation lists.

The reader is directed to the security considerations sections of [RFC5280] for the use of X.509v3 certificates, [RFC2560] for the use of OCSP response, [RFC4253] for server authentication, and [RFC4252] for user authentication. Implementations SHOULD NOT use revoked certificates because many causes of certificate revocation mean that the critical authentication properties needed are no longer true.

For example, compromise of a certificate's private key or issuance of a certificate to the wrong party are common reasons to revoke a certificate.

If a party to the SSH exchange attempts to use a revoked X.509v3 certificate, this attempt along with the date, time, certificate identity, and apparent origin IP address of the attempt SHOULD be logged as a security event in the system's audit logs or the system's general event logs. Similarly, if a certificate indicates that OCSP is used and there is no response to the OCSP query, the absence of a response along with the details of the attempted certificate use (as before) SHOULD be logged.

As with all specifications involving cryptographic algorithms, the quality of security provided by this specification depends on the strength of the cryptographic algorithms in use, the security of the keys, the correctness of the implementation, and the security of the public key infrastructure and the certificate authorities. Accordingly, implementers are encouraged to use high-assurance methods when implementing this specification and other parts of the Secure Shell protocol suite.

6. IANA Considerations

Consistent with Section 8 of [RFC4251] and Section 4.6 of [RFC4250], this document makes the following registrations:

In the Public Key Algorithm Names registry:

- The SSH public key algorithm "x509v3-ssh-dss".
- The SSH public key algorithm "x509v3-ssh-rsa".
- The SSH public key algorithm "x509v3-rsa2048-sha256".
- The family of SSH public key algorithm names beginning with "x509v3-ecdsa-sha2-" and not containing the at-sign ('@').

The two object identifiers used in Section 2.2.2 were assigned from an arc delegated by IANA to the PKIX Working Group.

7. References

7.1. Normative References

[ASN1] International Telecommunications Union, "Abstract Syntax Notation One (ASN.1): Specification of basic notation", X.680, July 2002.

[FIPS-180-2] National Institute of Standards and Technology, "Secure Hash Standard", FIPS 180-2, August 2002.

[FIPS-180-3] National Institute of Standards and Technology, "Secure Hash Standard", FIPS 180-3, October 2008.

[FIPS-186-3] National Institute of Standards and Technology, "Digital Signature Standard (DSS)", FIPS 186-3, June 2009.

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[RFC2560] Myers, M., Ankney, R., Malpani, A., Galperin, S., and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", RFC 2560, June 1999.

[RFC3279] Bassham, L., Polk, W., and R. Housley, "Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3279, April 2002.

[RFC3447] Jonsson, J. and B. Kaliski, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", RFC 3447, February 2003.

[RFC4250] Lehtinen, S. and C. Lonvick, "The Secure Shell (SSH) Protocol Assigned Numbers", RFC 4250, January 2006.

[RFC4251] Ylonen, T. and C. Lonvick, "The Secure Shell (SSH) Protocol Architecture", RFC 4251, January 2006.

[RFC4252] Ylonen, T. and C. Lonvick, "The Secure Shell (SSH) Authentication Protocol", RFC 4252, January 2006.

[RFC4253] Ylonen, T. and C. Lonvick, "The Secure Shell (SSH) Transport Layer Protocol", RFC 4253, January 2006.

[RFC5280] 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.

[RFC5480] Turner, S., Brown, D., Yiu, K., Housley, R., and T. Polk, "Elliptic Curve Cryptography Subject Public Key Information", RFC 5480, March 2009.

[RFC5656] Stebila, D. and J. Green, "Elliptic Curve Algorithm Integration in the Secure Shell Transport Layer", RFC 5656, December 2009.

[RFC5758] Dang, Q., Santesson, S., Moriarty, K., Brown, D., and T. Polk, "Internet X.509 Public Key Infrastructure: Additional Algorithms and Identifiers for DSA and ECDSA", RFC 5758, January 2010.

[RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, March 2011.

[SEC1] Standards for Efficient Cryptography Group, "Elliptic Curve Cryptography", SEC 1, September 2000, <http://www.secg.org/download/aid-780/sec1-v2.pdf>.

7.2. Informative References

[RFC4432] Harris, B., "RSA Key Exchange for the Secure Shell (SSH) Transport Layer Protocol", RFC 4432, March 2006.

[RFC4462] Hutzelman, J., Salowey, J., Galbraith, J., and V. Welch, "Generic Security Service Application Program Interface (GSS-API) Authentication and Key Exchange for the Secure Shell (SSH) Protocol", RFC 4462, May 2006.

[RFC5759] Solinas, J. and L. Zieglar, "Suite B Certificate and Certificate Revocation List (CRL) Profile", RFC 5759, January 2010.

[SP-800-131] Barker, E. and A. Roginsky, "DRAFT Recommendation for the Transitioning of Cryptographic Algorithms and Key Lengths", NIST Special Publication 800-131, June 2010.

Appendix A. Example

The following example illustrates the use of an X.509v3 certificate for a public key for the Digital Signature Algorithm when used in a Diffie-Hellman key exchange method. In the example, there is a chain of certificates of length 2, and a single OCSP response is provided.

```
  byte    SSH_MSG_KEXDH_REPLY
  string  0x00 0x00 0xXX 0xXX  -- length of the remaining data in
                                  this string
          0x00 0x00 0x00 0x0D  -- length of string "x509v3-ssh-dss"
          "x509v3-ssh-dss"
          0x00 0x00 0x00 0x02  -- there are 2 certificates
          0x00 0x00 0xXX 0xXX  -- length of sender certificate
          DER-encoded sender certificate
          0x00 0x00 0xXX 0xXX  -- length of issuer certificate
          DER-encoded issuer certificate
          0x00 0x00 0x00 0x01  -- there is 1 OCSP response
          0x00 0x00 0xXX 0xXX  -- length of OCSP response
          DER-encoded OCSP response
  mpint   f
  string  signature of H
```

Appendix B. Acknowledgements

The authors gratefully acknowledge helpful comments from Ran Atkinson, Samuel Edoho-Eket, Joseph Galbraith, Russ Housley, Jeffrey Hutzelman, Jan Pechanec, Peter Saint-Andre, Sean Turner, and Nicolas Williams.

O. Saarenmaa and J. Galbraith previously drafted a document on a similar topic.

Authors' Addresses

Kevin M. Igoe
National Security Agency
NSA/CSS Commercial Solutions Center
United States of America

EMail: kmigoe@nsa.gov

Douglas Stebila
Queensland University of Technology
Information Security Institute
Level 7, 126 Margaret St
Brisbane, Queensland 4000
Australia

EMail: douglas@stebila.ca