Internet-Draft
Updates: 4252, 4253 (if approved)
Intended status: Standards Track
Expires: September 29, 2017

D. Bider
Bitvise Limited

March 29, 2017

Use of RSA Keys with SHA-2 256 and 512 in Secure Shell (SSH)
draft-ietf-curdle-rsa-sha2-04.txt

Abstract

This memo defines an algorithm name, public key format, and signature format for use of RSA keys with SHA-2 hashing for server and client authentication in SSH connections.

Status

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

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 and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/1id-abstracts.html

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

1. Overview and Rationale

Secure Shell (SSH) is a common protocol for secure communication on the Internet. In [RFC4253], SSH originally defined the signature methods "ssh-rsa" for server and client authentication using RSA with SHA-1, and "ssh-dss" using 1024-bit DSA and SHA-1.

A decade later, these signature methods are considered deficient. For US government use, NIST has disallowed 1024-bit RSA and DSA, and use of SHA-1 for signing [800-131A].

This memo introduces a distinction between public key and signature algorithms in SSH, and defines new signature algorithm names allowing for interoperable use of existing and new RSA keys with SHA-2 hashing.

1.1. Requirements Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

1.2. Wire Encoding Terminology

The wire encoding types in this document - "boolean", "byte", "string", "mpint" - have meanings as described in [RFC4251].

2. Signature Algorithm as Distinct Aspect of Public Key Algorithm

In [RFC4252], the concept "public key algorithm" is used to establish a relationship between one algorithm name, and:

A. Procedures used to generate and validate a private/public keypair.
B. A format used to encode a public key.
C. Procedures used to calculate, encode, and verify a signature.

This document narrows the term "public key algorithm" to mean A and B, though it can still potentially imply C when a public key algorithm is associated with only one signature algorithm. A new term, "signature algorithm", is introduced to refer specifically to C.

This affects the meaning of the field "server_host_key_algorithms" in the message SSH_MSG_KEXINIT ([RFC4253]). With this document, this field now refers specifically to signature, not public key algorithms.

This also affects the message SSH_MSG_USERAUTH_REQUEST when used with the "publickey" authentication method as defined in [RFC4252]. With this document, the definition of this message is updated as follows:

```
byte      SSH_MSG_USERAUTH_REQUEST
string    user name in ISO-10646 UTF-8 encoding [RFC3629]
string    service name in US-ASCII
string    "publickey"
boolean   FALSE
string    signature algorithm name
string    public key blob
```

The format of the message remains unchanged. The change is in the line which now reads "signature algorithm name". This used to read "public key algorithm name".

These changes do not affect key types other than RSA. Other public key algorithms continue to use one signature algorithm of the same name.

There is no impact on existing implementations that support RSA keys only as "ssh-rsa". Such implementations continue to use the public key algorithm "ssh-rsa", and the signature algorithm of the same name.

3. New RSA Signature Algorithms

This memo adopts the style and conventions of [RFC4253] in specifying how use of a signature algorithm is indicated in SSH.

The following new signature algorithms are defined:

rsa-sha2-256	RECOMMENDED	sign	Raw RSA key
rsa-sha2-512	OPTIONAL	sign	Raw RSA key

These algorithms are suitable for use both in the SSH transport layer [RFC4253] for server authentication, and in the authentication layer [RFC4252] for client authentication.

Since RSA keys are not dependent on the choice of hash function, the new signature algorithms are defined as aspects of the existing "ssh-rsa" public key algorithm. This means the new algorithms reuse the "ssh-rsa" public key format as defined in [RFC4253]:

```
string    "ssh-rsa"
mpint     e
mpint     n
```

All aspects of the "ssh-rsa" format are kept, including the encoded string "ssh-rsa". This allows existing RSA keys to be used with the new signature formats, without requiring re-encoding, or affecting already trusted key fingerprints.

Signing and verifying using these algorithms is performed according to

the RSASSA-PKCS1-v1_5 scheme in [RFC3447] using SHA-2 [FIPS-180-4] as hash; MGF1 as mask function; and salt length equal to hash size.

For the algorithm "rsa-sha2-256", the hash used is SHA-2 256.
For the algorithm "rsa-sha2-512", the hash used is SHA-2 512.

The resulting signature is encoded as follows:

```
string    "rsa-sha2-256" / "rsa-sha2-512"
string    rsa_signature_blob
```

The value for 'rsa_signature_blob' is encoded as a string containing S - an octet string which is the output of RSASSA-PKCS1-v1_5, of length equal to the length in octets of the RSA modulus.

3.1. Use for server authentication

To express support and preference for one or both of these algorithms for server authentication, the SSH client or server includes one or both algorithm names, "rsa-sha2-256" and/or "rsa-sha2-512", in the name-list field "server_host_key_algorithms" in the SSH_MSG_KEXINIT packet [RFC4253]. If one of the two host key algorithms is negotiated, the server sends an "ssh-rsa" public key as part of the negotiated key exchange method (e.g. in SSH_MSG_KEXDH_REPLY), and encodes a signature with the appropriate signature algorithm name - either "rsa-sha2-256", or "rsa-sha2-512".

3.2. Use for client authentication

To use this algorithm for client authentication, the SSH client sends an SSH_MSG_USERAUTH_REQUEST message [RFC4252] encoding the "publickey" method, and encoding the string field "public key algorithm name" with the value "rsa-sha2-256" or "rsa-sha2-512". The "public key blob" field encodes the RSA public key using the "ssh-rsa" algorithm name. The signature field, if present, encodes a signature using an algorithm name that MUST match the SSH authentication request - either "rsa-sha2-256", or "rsa-sha2-512".

For example, an SSH "publickey" authentication request using an "rsa-sha2-512" signature would be properly encoded as follows:

```
byte      SSH_MSG_USERAUTH_REQUEST
string    user name
string    service name
string    "publickey"
boolean   TRUE
string    "rsa-sha2-512"
string    public key blob:
    string    "ssh-rsa"
    mpint     e
    mpint     n
string    signature:
```

```
        string    "rsa-sha2-512"
        string    rsa_signature_blob
```

3.3. Discovery of signature algorithms supported by servers

Implementation experience has shown that there are servers which apply authentication penalties to clients attempting signature algorithms which the SSH server does not support.

Servers that accept rsa-sha2-* signatures for client authentication SHOULD implement the extension negotiation mechanism defined in [SSH-EXT-INFO], including especially the "server-sig-algs" extension.

When authenticating with an RSA key against a server that does not implement the "server-sig-algs" extension, clients MAY default to an ssh-rsa signature to avoid authentication penalties.

4. IANA Considerations

IANA is requested to update the "Secure Shell (SSH) Protocol Parameters" registry, to extend the table Public Key Algorithm Names:

- To the immediate right of the column Public Key Algorithm Name, a new column is to be added, titled Signature Algorithm Name. For existing entries, the column Signature Algorithm Name should be assigned the same value found under Public Key Algorithm Name.

- Immediately following the existing entry for "ssh-rsa", two sibling entries are to be added:

P. K. Alg. Name	Sig. Alg. Name	Reference	Note
ssh-rsa	rsa-sha2-256	[this document]	Section 3
ssh-rsa	rsa-sha2-512	[this document]	Section 3

5. Security Considerations

The security considerations of [RFC4251] apply to this document.

The National Institute of Standards and Technology (NIST) Special Publication 800-131A [800-131A] disallows the use of RSA and DSA keys shorter than 2048 bits for US government use after 2013. The same document disallows the SHA-1 hash function, as used in the "ssh-rsa" and "ssh-dss" algorithms, for digital signature generation after 2013.

This document is based on the premise that RSA is used in environments where a gradual, compatible transition to improved algorithms will be better received than one that is abrupt and incompatible. It advises that SSH implementations add support for new RSA signature algorithms along with SSH_MSG_EXT_INFO and the "server-sig-algs" extension to allow coexistence of new deployments with older versions that support

only "ssh-rsa". Nevertheless, implementations SHOULD start to disable "ssh-rsa" in their default configurations as soon as they have reason to believe that new RSA signature algorithms have been widely adopted.

6. Why no DSA?

A draft version of this memo also defined an algorithm name for use of 2048-bit and 3072-bit DSA keys with a 256-bit subgroup and SHA-2 256 hashing. It is possible to implement DSA securely by generating "k" deterministically as per [RFC6979]. However, a plurality of reviewers were concerned that implementers would continue to use libraries that

generate "k" randomly. This is vulnerable to biased "k" generation, and extremely vulnerable to "k" reuse. This document therefore disrecommends DSA, in favor of RSA and elliptic curve cryptography.

7. References

7.1. Normative References

[FIPS-180-4] National Institute of Standards and Technology (NIST), United States of America, "Secure Hash Standard (SHS)", FIPS Publication 180-4, August 2015, <http://dx.doi.org/10.6028/NIST.FIPS.180-4>.

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

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

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

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

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

7.2. Informative References

[800-131A] National Institute of Standards and Technology (NIST), "Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths", NIST Special Publication 800-131A, January 2011, <http://csrc.nist.gov/publications/nistpubs/800-131A/sp800-131A.pdf>.

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

[RFC6979] Pornin, T., "Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA)", RFC 6979, August 2013.

[SSH-EXT-INFO] Bider, D., "Extension Negotiation in Secure Shell (SSH)", draft-ietf-curdle-ssh-ext-info-03.txt, March 2017,

<https://tools.ietf.org/html/
draft-ietf-curdle-ssh-ext-info-03>.

Author's Address

Denis Bider
Bitvise Limited
Suites 41/42, Victoria House
26 Main Street
GI

Phone: +506 8315 6519
EMail: ietf-ssh3@denisbider.com
URI: https://www.bitvise.com/

Acknowledgments

Thanks to Jon Bright, Niels Moeller, Stephen Farrell, Mark D. Baushke, Jeffrey Hutzelman, Hanno Boeck, Peter Gutmann, Damien Miller, Mat Berchtold, and Roumen Petrov for reviews, comments, and suggestions.