Internet Engineering Task Force (IETF)
Request for Comments: 8332
Updates: 4252, 4253
Category: Standards Track
ISSN: 2070-1721

D. Bider
Bitvise Limited

March 2018

Use of RSA Keys with SHA-256 and SHA-512 in the Secure Shell (SSH) Protocol

Abstract

This memo updates RFCs 4252 and 4253 to define new public key algorithms for use of RSA keys with SHA-256 and SHA-512 for server and client authentication in SSH connections.

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 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc8332.

Table of Contents

1. Overview and Rationale

Secure Shell (SSH) is a common protocol for secure communication on the Internet. In [RFC4253], SSH originally defined the public key algorithms "ssh-rsa" for server and client authentication using RSA with SHA-1, and "ssh-dss" using 1024-bit DSA and SHA-1. These algorithms are now considered deficient. For US government use, NIST has disallowed 1024-bit RSA and DSA, and use of SHA-1 for signing [NIST.800-131A].

This memo updates RFCs 4252 and 4253 to define new public key algorithms allowing for interoperable use of existing and new RSA keys with SHA-256 and SHA-512.

1.1. Requirements Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.2. Wire Encoding Terminology

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

2. Public Key Format vs. 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; and
C. procedures used to calculate, encode, and verify a signature.

This document uses the term "public key format" to identify only A and B in isolation. The term "public key algorithm" continues to identify all three aspects -- A, B, and C.

3. New RSA Public Key Algorithms

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

The following new public key 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 public key 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 public key algorithms, 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 [RFC8017] using SHA-2 [SHS] as hash.

For the algorithm "rsa-sha2-256", the hash used is SHA-256.
For the algorithm "rsa-sha2-512", the hash used is SHA-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 that contains an octet string S (which is the output of RSASSA-PKCS1-v1_5) and that has the same length (in octets) as the RSA modulus. When S contains leading zeros, there exist signers that will send a shorter encoding of S that omits them. A verifier MAY accept shorter encodings of S with one or more leading zeros omitted.

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" public key format.

For example, as defined in [RFC4252] and [RFC4253], 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
```

If the client includes the signature field, the client MUST encode the same algorithm name in the signature as in SSH_MSG_USERAUTH_REQUEST -- either "rsa-sha2-256" or "rsa-sha2-512". If a server receives a mismatching request, it MAY apply arbitrary authentication penalties, including but not limited to authentication failure or disconnect.

OpenSSH 7.2 (but not 7.2p2) incorrectly encodes the algorithm in the signature as "ssh-rsa" when the algorithm in SSH_MSG_USERAUTH_REQUEST is "rsa-sha2-256" or "rsa-sha2-512". In this case, the signature does actually use either SHA-256 or SHA-512. A server MAY, but is not required to, accept this variant or another variant that corresponds to a good-faith implementation and is considered safe to accept.

3.3. Discovery of Public Key Algorithms Supported by Servers

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

Servers that accept rsa-sha2-* signatures for client authentication SHOULD implement the extension negotiation mechanism defined in [RFC8308], 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. When the new rsa-sha2-* algorithms have been sufficiently widely adopted to warrant disabling "ssh-rsa", clients MAY default to one of the new algorithms.

4. IANA Considerations

IANA has updated the "Secure Shell (SSH) Protocol Parameters" registry, established with [RFC4250], to extend the table "Public Key Algorithm Names" [IANA-PKA] as follows.

- To the immediate right of the column "Public Key Algorithm Name", a new column has been added, titled "Public Key Format". For existing entries, the column "Public Key Format" has been assigned the same value as under "Public Key Algorithm Name".

- Immediately following the existing entry for "ssh-rsa", two sibling entries have been added:

P. K. Alg. Name	P. K. Format	Reference	Note
rsa-sha2-256	ssh-rsa	RFC 8332	Section 3
rsa-sha2-512	ssh-rsa	RFC 8332	Section 3

5. Security Considerations

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

5.1. Key Size and Signature Hash

The National Institute of Standards and Technology (NIST) Special Publication 800-131A, Revision 1 [NIST.800-131A] disallows RSA and DSA keys shorter than 2048 bits for US government use. The same document disallows the SHA-1 hash function for digital signature generation, except under NIST's protocol-specific guidance.

It is prudent to follow this advice also outside of US government use.

5.2. Transition

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 public key 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 the implementers believe that new RSA signature algorithms have been widely adopted.

5.3. PKCS #1 v1.5 Padding and Signature Verification

This document prescribes RSASSA-PKCS1-v1_5 signature padding because:

(1) RSASSA-PSS is not universally available to all implementations;
(2) PKCS #1 v1.5 is widely supported in existing SSH implementations;
(3) PKCS #1 v1.5 is not known to be insecure for use in this scheme.

Implementers are advised that a signature with RSASSA-PKCS1-v1_5 padding MUST NOT be verified by applying the RSA key to the signature, and then parsing the output to extract the hash. This may give an attacker opportunities to exploit flaws in the parsing and vary the encoding. Verifiers MUST instead apply RSASSA-PKCS1-v1_5 padding to the expected hash, then compare the encoded bytes with the output of the RSA operation.

6. References

6.1. Normative References

[SHS] NIST, "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, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC4251] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Protocol Architecture", RFC 4251, DOI 10.17487/RFC4251, January 2006, <https://www.rfc-editor.org/info/rfc4251>.

[RFC4252] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Authentication Protocol", RFC 4252, DOI 10.17487/RFC4252, January 2006, <https://www.rfc-editor.org/info/rfc4252>.

[RFC4253] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Transport Layer Protocol", RFC 4253, DOI 10.17487/RFC4253, January 2006, <https://www.rfc-editor.org/info/rfc4253>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8308] Bider, D., "Extension Negotiation in the Secure Shell (SSH) Protocol", RFC 8308, DOI 10.17487/RFC8308, March 2018, <https://www.rfc-editor.org/info/rfc8308>.

6.2. Informative References

[NIST.800-131A] NIST, "Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths", NIST Special Publication 800-131A, Revision 1, DOI 10.6028/NIST.SP.800-131Ar1, November 2015, <http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-131Ar1.pdf>.

[RFC4250] Lehtinen, S. and C. Lonvick, Ed., "The Secure Shell (SSH) Protocol Assigned Numbers", RFC 4250, DOI 10.17487/RFC4250, January 2006, <https://www.rfc-editor.org/info/rfc4250>.

[RFC8017] Moriarty, K., Ed., Kaliski, B., Jonsson, J., and A. Rusch, "PKCS #1: RSA Cryptography Specifications Version 2.2", RFC 8017, DOI 10.17487/RFC8017, November 2016, <https://www.rfc-editor.org/info/rfc8017>.

[IANA-PKA] IANA, "Secure Shell (SSH) Protocol Parameters", <https://www.iana.org/assignments/ssh-parameters/>.

Acknowledgments

Thanks to Jon Bright, Niels Moeller, Stephen Farrell, Mark D. Baushke, Jeffrey Hutzelman, Hanno Boeck, Peter Gutmann, Damien Miller, Mat Berchtold, Roumen Petrov, Daniel Migault, Eric Rescorla, Russ Housley, Alissa Cooper, Adam Roach, and Ben Campbell for reviews, comments, and suggestions.

Author’s Address

Denis Bider
Bitvise Limited
4105 Lombardy Court
Colleyville, Texas 76034
United States of America

Email: ietf-ssh3@denisbider.com
URI: https://www.bitvise.com/