Using transport layer security (TLS) with DSA and ECDSA ciphersuites
draft-mavrogiannopoulos-tls-dss-01

Abstract

This memo clarifies the usage of the digital signature algorithm (DSA) with extended key lengths, in the transport layer security (TLS) protocol earlier than 1.2, and makes clarifications for the usage of DSA and its elliptic curves equivalent (ECDSA) in TLS 1.2.

Status of This Memo

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). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on December 10, 2011.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.
1. Introduction

The TLS protocols support the DSA algorithm even from its first incarnation in [RFC2246]. However the latest DSA publication from NIST at [DSS], suggests some changes that do not straightforwardly apply to the TLS protocols.

In this document we describe the differences on the new DSS algorithms [DSS], and define a profile for TLS implementations.

2. Terminology

This document uses the same notation and terminology used in the TLS Protocol specification [RFC5246].

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].

3. DSA in FIPS-186-3

In this section we discuss the differences between the old DSS publication [OLDDSS] and the new one [DSS], that justify the need for a TLS profile.

DSA parameters include a prime modulus p and a prime divisor of p-1 called q. In [OLDDSS] the bit length of p was fixed to 1024 bits, the length of q to 160 bits and the underlying hash algorithm was fixed to SHA-1. However the DSA algorithm in [DSS] allows more lengths for p and q, as well different hash algorithms, than the older version which is currently referred by TLS protocols.

The new document relies on the "bits of security" term defined in [SP800-57], and recommends that security strength of the hash algorithm matches the security strength of other DSA parameters. It is required either the bits of the hash algorithm to match the bits of length of q (N), or if the hash size is larger, only the N leftmost bits of the hash output are being used. The corresponding mappings are shown in Table 1 and Table 2.
<table>
<thead>
<tr>
<th>Hash algorithm</th>
<th>Hash size</th>
<th>Bits of security</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHA-1</td>
<td>160</td>
<td>80</td>
</tr>
<tr>
<td>SHA-224</td>
<td>224</td>
<td>112</td>
</tr>
<tr>
<td>SHA-256</td>
<td>256</td>
<td>128</td>
</tr>
<tr>
<td>SHA-384</td>
<td>384</td>
<td>192</td>
</tr>
<tr>
<td>SHA-512</td>
<td>512</td>
<td>256</td>
</tr>
</tbody>
</table>

Table 1

<table>
<thead>
<tr>
<th>Length of p (L)</th>
<th>Length of q (N)</th>
<th>Bits of security</th>
<th>Matching hash algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td>1024</td>
<td>160</td>
<td>80</td>
<td>SHA-1, SHA-224, SHA-256,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SHA-384, SHA-512</td>
</tr>
<tr>
<td>2048</td>
<td>224</td>
<td>112</td>
<td>SHA-224, SHA-256,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SHA-384, SHA-512</td>
</tr>
<tr>
<td>2048</td>
<td>256</td>
<td>(112,128)</td>
<td>SHA-256, SHA-384, SHA-512</td>
</tr>
<tr>
<td>3072</td>
<td>256</td>
<td>128</td>
<td>SHA-256, SHA-384, SHA-512</td>
</tr>
</tbody>
</table>

Table 2

4. The SSL 3.0, TLS 1.0 and 1.1 protocols

4.1. DSA

The SSL 3.0, TLS 1.0 and 1.1 protocols support ciphersuites that utilize the DSA algorithms for signing. The digital signatures are used for the "Server key exchange" and "Certificate verify" messages. In those messages there is no indication of the signature algorithm used, thus the selection is implicit. The signature contained in both messages is defined, for the DSA algorithm, as:

```c
select (SignatureAlgorithm)
{
    case dsa:
        digitally-signed struct {
            opaque sha_hash[20];
        };
} Signature;
```

This structure refers to the DSA algorithm with L=1024 and N=160, but this is not an explicit requirement of those protocols and several existing implementations are using the SHA-1 algorithm for all DSA
key sizes. For this reason it is RECOMMENDED not to use DSA keys of sizes other than L=1024 and N=160 in combination with those protocols.

If however keys of sizes larger than L=1024 and N=160 have to be used, then the SHA-1 algorithm has to be used.

4.2. ECDSA

For TLS negotiation to proceed smoothly when an ECDSA enabled ciphersuite is negotiated both parties must agree to a curve. However given that [RFC4492] lists a very large number of curves but doesn’t recommend any, it is unclear which curves should be used in certificates for TLS.

To improve interoperability implementations SHOULD use certificates with curves restricted to the recommended by [RFC5480]. Those are summarized in Table 3.

+-----------+
| Curve     |
+-----------+
| secp224r1 |
| secp256r1 |
| secp384r1 |
| secp521r1 |
+-----------+

Table 3

5. The TLS protocol 1.2

This version of the protocol also requires signatures for the "Server key exchange" and "Certificate verify" messages. However in this version signature algorithm negotiation is explicit via the "Signature algorithms" extension. The signature used is as below:

```
struct {
    SignatureAndHashAlgorithm algorithm;
    opaque signature<0..2^16-1>;
} DigitallySigned;
```

It is however desirable for interoperability reasons to restrict the available options. This would allow constrained clients to support only the required algorithms, and servers that do not cache all messages up to "Certificate verify" in order to calculate the signature, to carry a single hash state instead.
5.1. DSA

In this case a signature algorithm should be selected that matches the requirements as in Table 4. Implementations SHOULD select the algorithms shown on that table.

<table>
<thead>
<tr>
<th>Length of p in bits</th>
<th>Length of q in bits</th>
<th>Hash algorithm</th>
<th>Hash size</th>
<th>Truncated hash size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1024</td>
<td>160</td>
<td>SHA-1</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>2048</td>
<td>224</td>
<td>SHA-256</td>
<td>32</td>
<td>28</td>
</tr>
<tr>
<td>2048</td>
<td>256</td>
<td>SHA-256</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>3072</td>
<td>256</td>
<td>SHA-256</td>
<td>32</td>
<td>32</td>
</tr>
</tbody>
</table>

Table 4

Note: When the hash size does not match the length of q, then only the leftmost bytes of the hash, that match the length of q, are used. This is indicated in the "Truncated hash size" column of the table.

5.1.1. Parameters not allowed by DSS

TLS implementations MUST NOT support parameter lengths not allowed by [DSS]. If illegal parameters are encountered, the handshake should be aborted using an "illegal_parameter" alert.

5.2. ECDSA

The signature hash algorithm SHOULD be selected in way that matches the requirements of Table 5. Also implementations SHOULD use certificates with curves restricted to the recommended by [RFC5480]. Those are summarized in Table 3.
### Table 5

<table>
<thead>
<tr>
<th>ECDSA key size</th>
<th>Hash algorithm</th>
<th>Hash size</th>
<th>Truncated hash size</th>
</tr>
</thead>
<tbody>
<tr>
<td>192</td>
<td>SHA-256</td>
<td>32</td>
<td>24</td>
</tr>
<tr>
<td>224</td>
<td>SHA-256</td>
<td>32</td>
<td>28</td>
</tr>
<tr>
<td>256</td>
<td>SHA-256</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>384</td>
<td>SHA-384</td>
<td>48</td>
<td>48</td>
</tr>
<tr>
<td>512</td>
<td>SHA-512</td>
<td>64</td>
<td>64</td>
</tr>
</tbody>
</table>

Note: As with DSA, when the hash size does not match the curve key size, only the leftmost bytes of the hash are used. This size is shown in the "Truncated hash size" column of the table.

### 6. Security Considerations

When DSA keys are being used in connections that involve the SSL 3.0, TLS 1.0 or TLS 1.1 protocols then the entire connection security depends on the SHA-1 algorithm. This is about 80-bits of security irrespective of the sizes of the DSA keys.

All security considerations discussed in [RFC5246], apply to this document.

### 7. References

#### 7.1. Normative References


7.2. Informative References


Author’s Address

Nikos Mavrogianiopoulos
ESAT/COSIC Katholieke Universiteit Leuven
Kasteelpark Arenberg 10, bus 2446
Leuven-Heverlee, B-3001
Belgium

EMail: nikos.mavrogianiopoulos@esat.kuleuven.be