TLS Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 12, 2014

P. Gutmann
University of Auckland

June 10, 2014

Standardised ECC Cipher Suites for TLS
draft-gutmann-tls-eccsuites-06.txt

Abstract

This document describes a set of standard ECC cipher suites for TLS that simplify the complex selection procedure described in the existing ECC RFC, simplifying implementation and easing interoperability problems.

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 12, 2014.

Copyright Notice

Table of Contents

1. Introduction

TLS-ECC [3] provides an extremely flexible, and by extension extremely complex means of specifying a large number of options involving the use of ECC algorithms for TLS [2]. As such the "cipher suites" in TLS-ECC [3] and by extension TLS-ECC-Brainpool [4] aren't suites in the conventional TLS sense but more an indication of intent to negotiate a Chinese menu, with details to be decided on later via various TLS extensions and parameter settings. This makes deciding on a particular suite nondeterministic, since later parameter choices and settings can negate the initial "cipher suite" choice, requiring returning to the suite list to try with another Chinese-menu suite in the hope that later parameter choices allow it to be used.

In practice no currently deployed implementation actually does this, either dropping the connection or aborting the handshake with a handshake-failure if the expected parameters aren't present throughout the various locations in the TLS handshake in which ECC parameters can be specified. This means that establishing a TLS connection using ECC often requires trial-and-error probing to ascertain what the other side is expecting to see before a connection can be established.

Experience with deployed implementations indicates that all of them appear to implement a common subset of fixed ECC parameters that work in all cases (alongside the more obscure options), representing a de facto profile of standard cipher suites rather than Chinese-menu selection options. For example one widely-used implementation didn't send out TLS ECC extensions and yet other implementations had no problems interoperating with it, indicating that what this document

specifies is already a de facto profile of implementations. This document standardises this de facto usage by defining a small number of standard ECC cipher suites with unambiguous parameters and settings.

1.1. Conventions Used in This Document

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 [1].

2. Cipher Suites

The table below defines standard ECC cipher suites with fixed, unambiguous parameters, based on the de facto profiles of suites seen in use in practice. Since the form of these suites match the existing non-ECC suites, they follow the existing suites in the { 0x00, 0xXX } range rather than being placed with the Chinese-menu suites at { 0xC0, 0xXX }.

```
CipherSuite TLS_ECDHE_ECDSA_P256_SHA256_WITH_AES_128_CBC_SHA = { 0x00, 0xXX }
CipherSuite TLS_ECDHE_ECDSA_P256_SHA256_WITH_AES_256_CBC_SHA = { 0x00, 0xXX }
CipherSuite TLS_ECDHE_ECDSA_P384_SHA384_WITH_AES_128_CBC_SHA = { 0x00, 0xXX }
CipherSuite TLS_ECDHE_ECDSA_P384_SHA384_WITH_AES_256_CBC_SHA = { 0x00, 0xXX }
CipherSuite TLS_ECDHE_ECDSA_BRAINPOOLP256_SHA256_WITH_AES_128_CBC_SHA =
  { 0x00, 0xXX }
CipherSuite TLS_ECDHE_ECDSA_BRAINPOOLP256_SHA256_WITH_AES_256_CBC_SHA =
  { 0x00, 0xXX }
CipherSuite TLS_ECDHE_ECDSA_BRAINPOOLP384_SHA384_WITH_AES_128_CBC_SHA =
  { 0x00, 0xXX }
CipherSuite TLS_ECDHE_ECDSA_BRAINPOOLP384_SHA384_WITH_AES_256_CBC_SHA =
  { 0x00, 0xXX }

CipherSuite TLS_ECDHE_ECDSA_P256_SHA256_WITH_AES_128_CBC_SHA256 =
  { 0x00, 0xXX }
CipherSuite TLS_ECDHE_ECDSA_P256_SHA256_WITH_AES_256_CBC_SHA256 =
  { 0x00, 0xXX }
CipherSuite TLS_ECDHE_ECDSA_P384_SHA384_WITH_AES_128_CBC_SHA384 =
  { 0x00, 0xXX }
CipherSuite TLS_ECDHE_ECDSA_P384_SHA384_WITH_AES_256_CBC_SHA384 =
  { 0x00, 0xXX }
CipherSuite TLS_ECDHE_ECDSA_BRAINPOOLP256_SHA256_WITH_AES_128_CBC_SHA256 =
  { 0x00, 0xXX }
CipherSuite TLS_ECDHE_ECDSA_BRAINPOOLP256_SHA256_WITH_AES_256_CBC_SHA256 =
  { 0x00, 0xXX }
CipherSuite TLS_ECDHE_ECDSA_BRAINPOOLP384_SHA384_WITH_AES_128_CBC_SHA384 =
  { 0x00, 0xXX }
CipherSuite TLS_ECDHE_ECDSA_BRAINPOOLP384_SHA384_WITH_AES_256_CBC_SHA384 =
  { 0x00, 0xXX }
```

```
CipherSuite TLS_ECDHE_ECDSA_P256_SHA256_WITH_AES_128_GCM_SHA256 =
  { 0x00, 0xXX }
CipherSuite TLS_ECDHE_ECDSA_P256_SHA256_WITH_AES_256_GCM_SHA256 =
  { 0x00, 0xXX }
CipherSuite TLS_ECDHE_ECDSA_P384_SHA384_WITH_AES_128_GCM_SHA384 =
  { 0x00, 0xXX }
CipherSuite TLS_ECDHE_ECDSA_P384_SHA384_WITH_AES_256_GCM_SHA384 =
  { 0x00, 0xXX }
```

In the above lists, the first set of suites allows use with TLS 1.0 and 1.1, the second set allows use with TLS 1.2, and the third set allows use with Suite B.

For each cipher suite with their ECC parameters denoted 'P256', 'P384', 'Brainpool256' or 'Brainpool384' the ECC parameters are:

- ECDH key agreement in Server Key Exchange/Client Key Exchange message: NIST P-256/X9.62 p256r1/SECG p256r1, NIST P-384/SECG

p384r1, Brainpool P256r1 or Brainpool P384r1 curve with uncompressed points as indicated in the suite name.

- ECDSA signature in Server Key Exchange message: P256, P384, BrainpoolP256 or BrainpoolP384 curve as for ECDH with uncompressed points and SHA256 or SHA384 as indicated in the suite name.
- Client authentication in Certificate Request/Certificate Verify messages: SHA256 or SHA384 as indicated in the suite name.
- (For the non-ECC parameters, namely the symmetric cipher, PRF, and MAC: AES, SHA-1, SHA256, and SHA384 as indicated in the suite name).

If no additional Chinese-menu ECC suites are used, implementations SHOULD NOT send the Supported Elliptic Curves or Supported Point Formats extensions since these parameters are fully specified by the suite choice. If additional Chinese-menu suites are used, implementations MUST send the Supported Elliptic Curves and Supported Point Formats extensions as per TLS-ECC [3]. The parameters specified in these extensions apply only to the Chinese-menu suites, not the fixed suites defined above.

TLS [2] states that if the client doesn't send the signature_algorithms extension then the hash algorithm defaults to

SHA1. This is required in order to provide a fall-back default if no other means of specifying the hash algorithm to be used is available. Since this document makes the use of the hash algorithm explicit in the cipher suite, the fall-back to the SHA1 default SHOULD NOT be triggered.

Note that the suites defined in this document augment, rather than supplant, the existing Chinese-menu suites options. Anyone requiring the use of more unusual ECC parameters and options can use the Chinese-menu capability to specify and select any parameters that they require.

2.1. Discussion

The issue that this document is intended to address may be more easily seen by considering how the parts of the Client Hello are processed. For standard cipher suites the server iterates through a list of suites proposed by the client and selects the most cromulent one. For example a server may have a list of suite IDs and parameters sorted in order of preference and select the lowest-ranked suite in the list from the ones proposed by the client.

For the Chinese-menu suites on the other hand, the server sees a Chinese menu selector sent by the client and then has to skip the remaining suites and other parts of the hello and process the extensions to see whether what's in there matches up with that the

Chinese-menu selector requested. For example if the Chinese menu said TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 but the supported-curves extension says P256 then the server has to either hope that the other side does the special-case X9.62 handling for hash truncation and gets it right (experience with current implementations indicates that they don't even support this capability, let alone get it right), or not take the gamble and go back to the cipher suites and look for another Chinese-menu option, and then skip the rest of the hello and process the extensions again to see if things work out this time, and if that doesn't work either then go back ...

In practice with currently-deployed implementations it's hard enough just trying to figure out which basic combinations of parameters they support (the usual response is a dropped connection or aborted handshake, requiring the use of trial-and-error probing to find out

what's possible), and even getting to the point of being able to interopability-test any of the more exotic combinations like hash truncation becomes more or less impossible. So the purpose of this document is to try and identify the common combinations of parameters that everyone seems to implement anyway and list them as conventional cipher suites, with no further parameterisation required.

At least one major implementation, Microsoft's SChannel, already does this, listing 'suites' like TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256_P256 and TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256_P256, see [1]. The choices given in Section 2 coincide with the Microsoft ones not because of any explicit attempt to copy them but because they represent the obvious, logical choices.

An additional problem with the Chinese-menu selection process is the fact that although it allows the specification of arbitrary numbers of handshake parameters, it never nails down how and where these parameters should be applied. Practical experience with implementations indicates that only the most straightforward combinations of algorithm parameters are likely to work. For example although it's possible to specify both P256 and P384 as acceptable curves, what this tends to mean in practice is that { ECDH P256 + ECDSA P256 } or { ECDH P384 + ECDSA P384 } are acceptable but { ECDH P256 + ECDSA P384 } or { ECDH P384 + ECDSA P256 } aren't. In the interests of interoperability it's recommended that, despite the apparent flexibility implied by the Chinese menu, implementations stick to the most straightforward application of algorithm parameters, using the same algorithm or parameters throughout the handshake even if it's implied by the Chinese-menu that mix-and-match combinations are possible. For example if the overall cipher suite is TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 then use SHA256 everywhere a hash function is used; if the curve types are P256 or P384 then use

either P256 everywhere or P384 everywhere. This design principle is captured in the requirements given in Section 2.

The term "Chinese menu" comes from the US (although the same usage, going back at least half a century, exists in the UK as well), where Chinese restaurants traditionally had columns for ordering food, and orders were put together in a mix-and-match manner by ordering an item from column A, two from column B, and so on. Any process that

involves picking a selection from different columns has become described as a "Chinese menu system".

3. Security Considerations

This document is a profile of, and simplifcation of, TLS-ECC [3]. No further security considerations are introduced beyond those present in TLS-ECC [3].

4. IANA Considerations

This document defines new cipher suites for TLS [to be allocated in the currently unallocated range { 0x00, 0xC6 } - { 0x00, 0xD1 }].

5. References

5.1. Normative References

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

[2] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[3] Blake-Wilson, S., Bolyard, N., Gupta, V., and C. Hawk, "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)", RFC 4492, May 2006.

[4] Merkle, J. and M. Lochter, "Elliptic Curve Cryptography (ECC) Brainpool Curves for Transport Layer Security (TLS)", RFC 7027, October 2013.

5.2. URIs

[1] http://msdn.microsoft.com/en-us/library/aa374757%28v=vs.85%29.aspx

Author's Address

Peter Gutmann
University of Auckland
Department of Computer Science
University of Auckland
New Zealand

Email: pgut001@cs.auckland.ac.nz