Negotiated Finite Field Diffie-Hellman Ephemeral Parameters for TLS
draft-ietf-tls-negotiated-ff-dhe-05

Abstract

Traditional finite-field-based Diffie-Hellman (DH) key exchange during the TLS handshake suffers from a number of security, interoperability, and efficiency shortcomings. These shortcomings arise from lack of clarity about which DH group parameters TLS servers should offer and clients should accept. This document offers a solution to these shortcomings for compatible peers by using a section of the TLS "EC Named Curve Registry" to establish common finite-field DH parameters with known structure and a mechanism for peers to negotiate support for these groups.

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 June 22, 2015.

Copyright Notice

Copyright (c) 2014 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.

Table of Contents

1. Introduction ........................................ 3
   1.1. Requirements Language .......................... 3
   1.2. Vocabulary ...................................... 4
2. Named Group Overview ............................... 4
3. Client Behavior ...................................... 5
4. Server Behavior ...................................... 6
5. Optimizations ........................................ 7
   5.1. Checking the Peer's Public Key ................. 7
   5.2. Short Exponents ................................. 7
   5.3. Table Acceleration .............................. 8
6. Operational Considerations ......................... 8
   6.1. Preference Ordering ............................. 8
7. Acknowledgements .................................... 9
8. IANA Considerations ................................ 9
9. Security Considerations ............................. 9
   9.1. Negotiation resistance to active attacks .... 9
   9.2. Group strength considerations ................ 10
   9.3. Finite-Field DHE only .......................... 11
   9.4. Deprecating weak groups ....................... 11
   9.5. Choice of groups ................................ 11
   9.6. Timing attacks .................................. 12
   9.7. Replay attacks from non-negotiated FFDHE .... 12
   9.8. Forward Secrecy ................................ 12
   9.9. False Start ..................................... 13
10. Privacy Considerations ............................. 13
   10.1. Client fingerprinting ........................... 13
11. References .......................................... 13
   11.1. Normative References .......................... 13
   11.2. Informative References ....................... 14
   11.3. URIs ........................................... 15
Appendix A. Named Group Registry ........................ 15
   A.1. ffdhe2048 ...................................... 16
   A.2. ffdhe3072 ...................................... 17
   A.3. ffdhe4096 ...................................... 18
   A.4. ffdhe8192 ...................................... 20
Author's Address ........................................ 23
1. Introduction

Traditional TLS [RFC5246] offers a Diffie-Hellman ephemeral (DHE) key exchange mode which provides Forward Secrecy for the connection. The client offers a ciphersuite in the ClientHello that includes DHE, and the server offers the client group parameters generator g and modulus p. If the client does not consider the group strong enough (e.g. if p is too small, or if p is not prime, or there are small subgroups), or if it is unable to process the group for other reasons, the client has no recourse but to terminate the connection.

Conversely, when a TLS server receives a suggestion for a DHE ciphersuite from a client, it has no way of knowing what kinds of DH groups the client is capable of handling, or what the client's security requirements are for this key exchange session. For example, some widely-distributed TLS clients are not capable of DH groups where p > 1024 bits. Other TLS clients may by policy wish to use DHE only if the server can offer a stronger group (and are willing to use a non-PFS key-exchange mechanism otherwise). The server has no way of knowing which type of client is connecting, but must select DH parameters with insufficient knowledge.

Additionally, the DH parameters chosen by the server may have a known structure which renders them secure against a small subgroup attack, but a client receiving an arbitrary p and g has no efficient way to verify that the structure of a new group is reasonable for use.

This modification to TLS solves these problems by using a section of the "EC Named Curves" registry to select common DH groups with known structure; defining the use of the "elliptic_curves(10)" extension (described here as "Supported Groups" extension) for clients advertising support for DHE with these groups. This document also provides guidance for compliant peers to take advantage of the additional security, availability, and efficiency offered.

The use of this mechanism by one compliant peer when interacting with a non-compliant peer should have no detrimental effects.

1.1. Requirements Language

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. Vocabulary

The terms "DHE" or "FFDHE" are used in this document to refer to the finite-field-based Diffie-Hellman ephemeral key exchange mechanism in TLS. TLS also supports elliptic-curve-based Diffie-Hellman (ECDHE) ephemeral key exchanges [RFC4492], but this document does not document their use. A registry previously used only by ECHDE-capable implementations is expanded in this document to cover FFDHE groups as well. "FFDHE ciphersuites" is used in this document to refer exclusively to ciphersuites with FFDHE key exchange mechanisms, but note that these suites are typically labeled with a TLS_DHE_ prefix.

2. Named Group Overview

We use previously-unallocated codepoints within the extension currently known as "elliptic_curves" (section 5.1.1. of [RFC4492]) to indicate known finite field groups. The extension's semantics are expanded from "Supported Elliptic Curves" to "Supported Groups". The semantics of the extension's data type (enum NamedCurve) is also expanded from "named curve" to "named group".

Codepoints in the NamedCurve registry with a high byte of 0x01 (that is, between 256 and 511 inclusive) are set aside for FFDHE groups, though only a small number of them are initially defined and we do not expect many other FFDHE groups to be added to this range. No codepoints outside of this range will be allocated to FFDHE groups. The new code points for the NamedCurve registry are:

```c
enum {
    // other already defined elliptic curves (see RFC 4492)
    ffdhe2048(256), ffdhe3072(257), ffdhe4096(258),
    ffdhe8192(259),
    //
    } NamedCurve;
```

These additions to the Named Curve registry are described in detail in Appendix A. They are all safe primes derived from the base of the natural logarithm ("e"), with the high and low 64 bits set to 1 for efficient Montgomery or Barrett reduction.

The use of the base of the natural logarithm here is as a "nothing-up-my-sleeve" number. The goal is to guarantee that the bits in the middle of the modulus are effectively random, while avoiding any suspicion that the primes have secretly been selected to be weak according to some secret criteria. [RFC3526] used pi for this value. See Section 9.5 for reasons that this draft does not reuse pi.
3. Client Behavior

A TLS client that is capable of using strong finite field Diffie-Hellman groups can advertise its capabilities and its preferences for stronger key exchange by using this mechanism.

The compatible client that wants to be able to negotiate strong FFDHE SHOULD send a "Supported Groups" extension (identified by type elliptic_curves(10) in [RFC4492]) in the ClientHello, and include a list of known FFDHE groups in the extension data, ordered from most preferred to least preferred. If the client also supports and wants to offer ECDHE key exchange, it MUST use a single "Supported Groups" extension to include all supported groups (both ECDHE and FFDHE groups). The ordering SHOULD be based on client preference, but see Section 6.1 for more nuance.

A client that offers any of these values in the elliptic_curves extension SHOULD ALSO include at least one FFDHE ciphersuite in the Client Hello.

A client who offers a group MUST be able and willing to perform a DH key exchange using that group.

A client that offers one or more FFDHE groups in the "Supported Groups" extension and an FFDHE ciphersuite, and receives an FFDHE ciphersuite from the server SHOULD take the following steps upon receiving the ServerKeyExchange:

For non-anonymous ciphersuites where the offered Certificate is valid and appropriate for the peer, validate the signature over the ServerDHParams. If not valid, terminate the connection.

If the signature over ServerDHParams is valid, compare the selected dh_p and dh_g with the FFDHE groups offered by the client. If none of the offered groups match, the server is not compatible with this draft. The client MAY decide to continue the connection if the selected group is acceptable under local policy, or it MAY decide to terminate the connection with a fatal insufficient_security(71) alert.

If the selected group matches an offered FFDHE group exactly, the client MUST verify that dh_Ys is in the range 1 < dh_Ys < dh_p - 1. If dh_Ys is not in this range, the client MUST terminate the connection with a fatal handshake_failure(40) alert.

If the selected group matches an offered FFDHE group exactly, and dh_Ys is in range, then the client SHOULD continue with the connection as usual.
4. Server Behavior

If a compatible TLS server receives a Supported Groups extension from a client that includes any FFDHE group (i.e. any codepoint between 256 and 511 inclusive, even if unknown to the server), and if none of the client-proposed FFDHE groups are known and acceptable to the server, then the server SHOULD NOT select an FFDHE ciphersuite. In this case, the server SHOULD select an acceptable non-FFDHE ciphersuite from the client's offered list. If the extension is present with FFDHE groups, none of the client's offered groups are acceptable by the server, and none of the client's proposed non-FFDHE ciphersuites are acceptable to the server, the server SHOULD end the connection with a fatal TLS alert of type insufficient_security(71).

If at least one FFDHE ciphersuite is present in the client ciphersuite list, and the Supported Groups extension is present in the ClientHello, but the extension does not include any FFDHE groups (i.e. no codepoints between 256 and 511 inclusive), then the server knows that the client is not compatible with this document. In this scenario, a server MAY choose to select a non-FFDHE ciphersuite, or MAY choose an FFDHE ciphersuite and offer an FFDHE group of its choice to the client as part of a traditional ServerKeyExchange.

A compatible TLS server that receives the Supported Groups extension with FFDHE codepoints in it, and which selects an FFDHE ciphersuite MUST select one of the client's offered groups. The server indicates the choice of group to the client by sending the group's parameters as usual in the ServerKeyExchange as described in section 7.4.3 of [RFC5246].

A TLS server MUST NOT select a named FFDHE group that was not offered by a compatible client.

A TLS server MUST NOT select an FFDHE ciphersuite if the client did not offer one, even if the client offered an FFDHE group in the Supported Groups extension.

If a non-anonymous FFDHE ciphersuite is chosen, and the TLS client has used this extension to offer an FFDHE group of comparable or greater strength than the server's public key, the server SHOULD select an FFDHE group at least as strong as the server's public key. For example, if the server has a 3072-bit RSA key, and the client offers only ffdhe2048 and ffdhe4096, the server SHOULD select ffdhe4096.

When a compatible server selects an FFDHE group from among a client's Supported Groups, and the client sends a ClientKeyExchange, the server MUST verify that 1 < dh_Yc < dh_p - 1. If it is out of range,
the server MUST terminate the connection with Fatal handshake_failure(40) alert.

5. Optimizations

In a key exchange with a successfully negotiated known FFDHE group, both peers know that the group in question uses a safe prime as a modulus, and that the group in use is of size p-1 or (p-1)/2. This allows at least three optimizations that can be used to improve performance.

5.1. Checking the Peer's Public Key

Peers MUST validate each other's public key Y (dh_Ys offered by the server or dh_Yc offered by the client) by ensuring that 1 < Y < p-1. This simple check ensures that the remote peer is properly behaved and isn't forcing the local system into a small subgroup.

To reach the same assurance with an unknown group, the client would need to verify the primality of the modulus, learn the factors of p-1, and test both the generator g and Y against each factor to avoid small subgroup attacks.

5.2. Short Exponents

Traditional Finite Field Diffie-Hellman has each peer choose their secret exponent from the range [2, p-2]. Using exponentiation by squaring, this means each peer must do roughly 2*log_2(p) multiplications, twice (once for the generator and once for the peer's public key).

Peers concerned with performance may also prefer to choose their secret exponent from a smaller range, doing fewer multiplications, while retaining the same level of overall security. Each named group indicates its approximate security level, and provides a lower-bound on the range of secret exponents that should preserve it. For example, rather than doing 2*2*3072 multiplications for a ffdhe3072 handshake, each peer can choose to do 2*2*250 multiplications by choosing their secret exponent from the range [2^249, 2^250] (that is, a m-bit integer where m is at least 224) and still keep the approximate 125-bit security level.

A similar short-exponent approach is suggested in SSH's Diffie-Hellman key exchange (See section 6.2 of [RFC4419]).
5.3. Table Acceleration

Peers wishing to further accelerate FFDHE key exchange can also pre-compute a table of powers of the generator of a known group. This is a memory vs. time tradeoff, and it only accelerates the first exponentiation of the ephemeral DH exchange (the fixed-base exponentiation). The variable-base exponentiation (using the peer's public exponent as a base) still needs to be calculated as normal.

6. Operational Considerations

6.1. Preference Ordering

The ordering of named groups in the Supported Groups extension may contain some ECDHE groups and some FFDHE groups. These SHOULD be ranked in the order preferred by the client.

However, the ClientHello also contains list of desired ciphersuites, also ranked in preference order. This presents the possibility of conflicted preferences. For example, if the ClientHello contains a CipherSuite with two choices in order
<TLS_DHE_RSA_WITH_AES_128_CBC_SHA, TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA> and the Supported Groups Extension contains two choices in order <secp256r1, ffdhe3072> then there is a clear contradiction. Clients SHOULD NOT present such a contradiction since it does not represent a sensible ordering. A server that encounters such an contradiction when selecting between an ECDHE or FFDHE key exchange mechanism while trying to respect client preferences SHOULD give priority to the Supported Groups extension (in the example case, it should select TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA with secp256r1), but MAY resolve the contradiction any way it sees fit.

More subtly, clients MAY interleave preferences between ECDHE and FFDHE groups, for example if stronger groups are preferred regardless of cost, but weaker groups are acceptable, the Supported Groups extension could consist of:
<ffdhe8192, secp384p1, ffdhe3072, secp256r1>. In this example, with the same CipherSuite offered as the previous example, a server configured to respect client preferences and with support for all listed groups SHOULD select TLS_DHE_RSA_WITH_AES_128_CBC_SHA with ffdhe8192. A server configured to respect client preferences and with support for only secp384p1 and ffdhe3072 SHOULD select TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA with secp384p1.
7. Acknowledgements

Thanks to Fedor Brunner, Dave Fergemann, Sandy Harris, Watson Ladd, Nikos Mavrogiannopolous, Niels Moeller, Bodo Moeller, Kenny Paterson, Eric Rescorla, Tom Ritter, Rene Struik, Martin Thomson, Sean Turner, and other members of the TLS Working Group for their comments and suggestions on this draft. Any mistakes here are not theirs.

8. IANA Considerations

IANA maintains the registry currently known as EC Named Curves (originally defined in [RFC4492] and updated by [RFC7027]) at [1].

This document expands the semantics of this registry slightly, to include groups based on finite fields in addition to groups based on elliptic curves. It should add a range designation to that registry, indicating that values from 256-511 (inclusive) are set aside for "Finite Field Diffie-Hellman groups", and that all other entries in the registry are "Elliptic curve groups".

This document allocates five codepoints in the registry, as follows:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>DTLS-OK</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>256</td>
<td>ffdhe2048</td>
<td>Y</td>
<td>[this document]</td>
</tr>
<tr>
<td>257</td>
<td>ffdhe3072</td>
<td>Y</td>
<td>[this document]</td>
</tr>
<tr>
<td>258</td>
<td>ffdhe4096</td>
<td>Y</td>
<td>[this document]</td>
</tr>
<tr>
<td>259</td>
<td>ffdhe8192</td>
<td>Y</td>
<td>[this document]</td>
</tr>
</tbody>
</table>

9. Security Considerations

9.1. Negotiation resistance to active attacks

Because the contents of the Supported Groups extension is hashed in the finished message, an active MITM that tries to filter or omit groups will cause the handshake to fail, but possibly not before getting the peer to do something they would not otherwise have done.

An attacker who impersonates the server can try to do any of the following:

Pretend that a non-compatible server is actually capable of this extension, and select a group from the client's list, causing the client to select a group it is willing to negotiate. It is unclear how this would be an effective attack.
Pretend that a compatible server is actually non-compatible by negotiating a non-FFDHE ciphersuite. This is no different than MITM ciphersuite filtering.

Pretend that a compatible server is actually non-compatible by negotiating a DHE ciphersuite, with a custom (perhaps weak) group chosen by the attacker. This is no worse than the current scenario, and would require the attacker to be able to sign the ServerDHParams, which should not be possible without access to the server's secret key.

An attacker who impersonates the client can try to do the following:

Pretend that a compatible client is not compatible (e.g. by not offering the Supported Groups extension, or by replacing the Supported Groups extension with one that includes no FFDHE groups). This could cause the server to negotiate a weaker DHE group during the handshake, or to select a non-FFDHE ciphersuite, but it would fail to complete during the final check of the Finished message.

Pretend that a non-compatible client is compatible (e.g. by . This could cause the server to select a particular named group in the ServerKeyExchange, or to avoid selecting an FFDHE ciphersuite. The peers would fail to compute the final check of the Finished message.

Change the list of groups offered by the client (e.g. by removing the stronger of the set of groups offered). This could cause the server to negotiate a weaker group than desired, but again should be caught by the check in the Finished message.

### 9.2. Group strength considerations

TLS implementations using FFDHE key exchange should consider the strength of the group they negotiate. The strength of the selected group is one of the factors which defines the connection's resilience against attacks on the session's confidentiality and integrity, since the session keys are derived from the DHE handshake.

While attacks on integrity must generally happen while the session is in progress, attacks against session confidentiality can happen significantly later, if the entire TLS session is stored for offline analysis. Therefore, FFDHE groups should be selected by clients and servers based on confidentiality guarantees they need. Sessions which need extremely long-term confidentiality should prefer stronger groups.
[ENISA] provides rough estimates of group resistance to attack, and recommends that forward-looking implementations ("future systems") should use FFDHE group sizes of at least 3072 bits. ffdhe3072 is intended for use in these implementations.

9.3. Finite-Field DHE only

Note that this document specifically targets only finite field-based Diffie-Hellman ephemeral key exchange mechanisms. It does not cover the non-ephemeral DH key exchange mechanisms, nor does it address elliptic curve DHE (ECDHE) key exchange, which is defined in [RFC4492].

Measured by computational cost to the TLS peers, ECDHE appears today to offer much a stronger key exchange than FFDHE.

9.4. Deprecating weak groups

Advances in hardware or in finite field cryptanalysis may cause some of the negotiated groups to not provide the desired security margins, as indicated by the estimated work factor of an adversary to discover the premaster secret (and may therefore compromise the confidentiality and integrity of the TLS session).

Revisions of this document should mark known-weak groups as explicitly deprecated for use in TLS, and should update the estimated work factor needed to break the group, if the cryptanalysis has changed. Implementations that require strong confidentiality and integrity guarantees should avoid using deprecated groups and should be updated when the estimated security margins are updated.

9.5. Choice of groups

Other lists of named finite field Diffie-Hellman groups [STRONGSWAN-IKE] exist. This draft chooses to not reuse them for several reasons:

Using the same groups in multiple protocols increases the value for an attacker with the resources to crack any single group.

The IKE groups include weak groups like MODP768 which are unacceptable for secure TLS traffic.

Mixing group parameters across multiple implementations leaves open the possibility of some sort of cross-protocol attack. This shouldn't be relevant for ephemeral scenarios, and even with non-ephemeral keying, services shouldn't share keys; however, using different groups avoids these failure modes entirely.
9.6. Timing attacks

Any implementation of finite field Diffie-Hellman key exchange should use constant-time modular-exponentiation implementations. This is particularly true for those implementations that ever re-use DHE secret keys (so-called "semi-static" ephemeral keying) or share DHE secret keys across a multiple machines (e.g. in a load-balancer situation).

9.7. Replay attacks from non-negotiated FFDHE

[SECURE-RESUMPTION], [CROSS-PROTOCOL], and [SSL3-ANALYSIS] all show a malicious peer using a bad FFDHE group to maneuver a client into selecting a pre-master secret of the peer's choice, which can be replayed to another server using a non-FFDHE key exchange, and can then be bootstrapped to replay client authentication.

To prevent this attack (barring the fixes proposed in [SESSION-HASH]), a client would need not only to implement this draft, but also to reject non-negotiated FFDHE ciphersuites whose group structure it cannot afford to verify. Such a client would need to abort the initial handshake and reconnect to the server in question without listing any FFDHE ciphersuites on the subsequent connection.

This tradeoff may be too costly for most TLS clients today, but may be a reasonable choice for clients performing client certificate authentication, or who have other reason to be concerned about server-controlled pre-master secrets.

9.8. Forward Secrecy

One of the main reasons to prefer FFDHE ciphersuites is Forward Secrecy, the ability to resist decryption even if when the endpoint's long-term secret key (usually RSA) is revealed in the future.

This property depends on both sides of the connection discarding their ephemeral keys promptly. Implementations should wipe their FFDHE secret key material from memory as soon as it is no longer needed, and should never store it in persistent storage.

Forward secrecy also depends on the strength of the Diffie-Hellman group; using a very strong symmetric cipher like AES256 with a forward-secret ciphersuite, but generating the keys with a much weaker group like dhe2048 simply moves the adversary's cost from attacking the symmetric cipher to attacking the dh_Ys or dh_Yc ephemeral keyshares.
If the goal is to provide forward secrecy, attention should be paid to all parts of the ciphersuite selection process, both key exchange and symmetric cipher choice.

### 9.9. False Start

Clients capable of TLS False Start [FALSE-START] may receive a proposed FFDHE group from a server that is attacker-controlled. In particular, the attacker can modify the ClientHello to strip the proposed FFDHE groups, which may cause the server to offer a weaker FFDHE group than it should, and this will not be detected until receipt of the server's Finished message. This could cause the a client using the False Start protocol modification to send data encrypted under a weak key agreement.

Clients should have their own classification of FFDHE groups that are "cryptographically strong" in the same sense described in the description of symmetric ciphers in [FALSE-START], and MUST offer at least one of these in the initial handshake if they contemplate using the False Start protocol modification.

Compatible clients performing a full handshake MUST NOT use the False Start protocol modification if the server selects an FFDHE ciphersuite but sends a group that is not cryptographically strong from the client's perspective.

### 10. Privacy Considerations

#### 10.1. Client fingerprinting

This extension provides a few additional bits of information to distinguish between classes of TLS clients (see e.g. [PANOPTICCLICK]). To minimize this sort of fingerprinting, clients SHOULD support all named groups at or above their minimum security threshold. New named groups SHOULD NOT be added to the registry without consideration of the cost of browser fingerprinting.

### 11. References

#### 11.1. Normative References

[FALSE-START]
Langley, A., Modadugu, N., and B. Moeller, "Transport Layer Security (TLS) False Start", Work in Progress, 
draft-bmoeller-tls-falsestart-01, November 2014.

11.2. Informative References

[CROSS-PROTOCOL]

[ECRYPTII]

[ENISA]

[PANOPTICLICK]


[SECURE-RESUMPTION]

[SESSION-HASH]

[SSL3-ANALYSIS]

[STRONGSWAN-IKE]

11.3. URIs

[1] https://www.iana.org/assignments/tls-parameters/tls-parameters.xhtml#tls-parameters-8

Appendix A. Named Group Registry

Each description below indicates the group itself, its derivation, its expected strength (estimated roughly from guidelines in [ECRYPTII]), and whether it is recommended for use in TLS key exchange at the given security level. It is not recommended to add further finite field groups to the NamedCurves registry; any attempt to do so should consider Section 10.1.

The primes in these finite field groups are all safe primes, that is, a prime p is a safe prime when q = (p-1)/2 is also prime. Where e is the base of the natural logarithm, and square brackets denote the floor operation, the groups which initially populate this registry are derived for a given bitlength b by finding the lowest positive integer X that creates a safe prime p where:

\[ p = 2^b - 2^{b-64} + \{2^{b-130} e\} + X \times 2^{64} - 1 \]

New additions of FFDHE groups to this registry may use this same derivation (e.g. with different bitlengths) or may choose their
parameters in a different way, but must be clear about how the parameters were derived.

New additions of FFDHE groups MUST use a safe prime as the modulus to enable the inexpensive peer verification described in Section 5.1.

A.1. ffdhe2048

The 2048-bit group has registry value 256, and is calculated from the following formula:

The modulus is: $p = 2^{2048} - 2^{1984} + \{2^{1918} \times e \} + 560315 \times 2^{64} - 1$

The hexadecimal representation of $p$ is:

```
60C7583 E2D36965 A9E13641 46433FB CC939DE 249B3E9
7D2FE363 630C75D8 F681B202 AEC4617A D3DF1ED5 D5FD6561
2433F11F 5F666E0D B557135E 7F57C935
984F0C70 E0E6B877 E2A689DA F3FE872 1DF158A1 36ADE735
30ACCA4F 483A797A BC0AB182 B324FB61 D108A94B B2C8E3FB
B96ADAB7 60DF468 1D4F42A3 DE394DF4 AE56EDE7 6372BB19
0B07A7C8 EE0A6D70 9E02FCE1 CDF7E2EC C03404CD 28342F61
9172F39C E9583FF 8E4F1232 EEF2B183 C3F36B1B 4C6FAD73
3BB5F4BC 2EC22005 C58EF183 7D16B3B 6E3F3426 C1B2EFFA
886B4238 61285C97 F0000000
```

The generator is: $g = 2$

The group size is: $q = (p-1)/2$

The hexadecimal representation of $q$ is:

```
7F58458 A2BB4A9A AFDC5620 273D3CF1
D8B9C583 CE2D36965 A9E13641 146433FB CC939DE 249B3E9
7D2FE363 630C75D8 F681B202 AEC4617A D3DF1ED5 D5FD6561
2433F11F 5F666E0D B557135E 7F57C935
984F0C70 E0E6B877 E2A689DA F3FE872 1DF158A1 36ADE735
30ACCA4F 483A797A BC0AB182 B324FB61 D108A94B B2C8E3FB
B96ADAB7 60DF468 1D4F42A3 DE394DF4 AE56EDE7 6372BB19
0B07A7C8 EE0A6D70 9E02FCE1 CDF7E2EC C03404CD 28342F61
9172F39C E9583FF 8E4F1232 EEF2B183 C3F36B1B 4C6FAD73
3BB5F4BC 2EC22005 C58EF183 7D16B3B 6E3F3426 C1B2EFFA
886B4238 61285C97 F0000000
```

The estimated symmetric-equivalent strength of this group is 103 bits.
Peers using ffdhe2048 that want to optimize their key exchange with a short exponent (Section 5.2) should choose a secret key of at least 206 bits.

A.2.  ffdhe3072

The 3072-bit prime has registry value 257, and is calculated from the following formula:

The modulus is: \( p = 2^{3072} - 2^{3008} + \left\{ \left[ 2^{2942} \times e \right] + 2625351 \right\} \times 2^{64} - 1 \)

The hexadecimal representation of \( p \) is:

```
0B07A7C8 EE0A6D70 9E02FCE1 CDF7E2EC C03404CD 28342F61
9172FE9C E98583FF 8E4F1232 EEF28183 C3FE3B1B 4C6FAD73
3BB5FCBC 2EC22005 C58EF183 7D1683B2 C6F34A26 C1B2EFFA
886B4238 611FCFDC DE355B3B 6519035B BC34F4DE F99C0238
61B46FC9 D6E6C907 7AD91D26 91F7F7EE 598CB0FA C186D91C
AEE1309 85139270 B4130C93 BC437944 F4FD4452 E2D74DD3
64F2E21E 71F54BFF 5CAE82AB 9C9DF69E E86D2BC5 22363A0D
ABC52197 9B0DEADA 1DBF9A42 D5C4484E 0ABC0D6B FA53DDEF
3C1B20EE 3FD59D7C 25E41D2B 66C62E37 FFFFFFFF FFFFFFFF
```

The generator is: \( g = 2 \)

The group size is: \( q = (p-1)/2 \)

The hexadecimal representation of \( q \) is:
The estimated symmetric-equivalent strength of this group is 125 bits.

Peers using ffdhe3072 that want to optimize their key exchange with a short exponent (Section 5.2) should choose a secret key of at least 250 bits.

A.3. ffdhe4096

The 4096-bit group has registry value 258, and is calculated from the following formula:

The modulus is: $p = 2^{4096} - 2^{4032} + \{[2^{3966} \times e] + 5736041\} \times 2^{64} - 1$

The hexadecimal representation of $p$ is:
The generator is: \( g = 2 \)

The group size is: \( q = (p-1)/2 \)

The hexadecimal representation of \( q \) is:
The estimated symmetric-equivalent strength of this group is 150 bits.

Peers using ffdhe4096 that want to optimize their key exchange with a short exponent (Section 5.2) should choose a secret key of at least 300 bits.

A.4. ffdhe8192

The 8192-bit group has registry value 259, and is calculated from the following formula:

The modulus is: $p = 2^{8192} - 2^{8128} + \{2^{8062} \cdot e \} + 10965728 \cdot 2^{64} - 1$

The hexadecimal representation of $p$ is:

7FFFFFFF FFFFFFFF D6FC2A2C 515DA54D 57EE2B10 139E9E78 EC5CEC41 E7169B4A D4F09B20 8A3219FD E649CEE7 124D9F7C BE97F1B1 B1863AEC 7B409B01 576230BD 69EF8F6A EAFEB2B0 9219FA8F AF833768 42B1B2AA 9EF68D79 DAAB89AF 3FABE49A CC278638 707345BB F15344ED 79F7F439 0EF8AC50 9B56F39A 96C5CE2C 67169B4A D4F09B20 8A3219FD E649CEE7 124D9F7C DB56D5B 806BFA34 0EA7A151 EF1CA6FA 572B76F3 B1B95D8C 8583D3E4 17611002 717778C1 BE8B41D9 9D79A513 0D977FD 4435A11C 308FE7EE 6F1AAD9D B28C81AD DE1A7A6F 7CCE011C 30DA37E4 EB7364B3 BD6C8E93 48FBBF7F7 2CC6587D 60C36C8E 577F0984 C289C93B 5A09B649 DE21BCA2 7A7EA229 716BA6E9 B279710F 38FAA5FF AE574155 CE4FB4F 743695E2 911B1D06 D5E290CB CD86F56D 9EDFC2D1 6AE22427 0566E835 FD29EEF7 9E0D9077 1FEACEBE 12F20E95 B34F0F78 B737A961 8B26FA7D BC9874F2 72C42BBB 563EFAFA 6B4FB68C 3BB1E78E AA81A002 43FAAAD2 BF18E63D 389AE443 77DA18C5 76B50F00 96CF3419 5483B005 48C09862 36E3BC7C B8D6801C 0494CD01 99E5C5BD 0D06C9E B8A00E15 15277554 FCC65856 054148E6 764BEE7 C764D9AD 3FC45235 A6DAD428 FA2D0C170 E345003F 2F32AFB5 7FFFFFFF FFFFFFFF
The generator is: g = 2

The group size is: q = (p-1)/2
The hexadecimal representation of q is:

```
7FFFFFFF FFFFFFFF D6FC2A2C 515DA54D 57EE2B10 139E9E78
EC5CE2C1 E716984A D4F09B20 8A3219FD E649CEE7 124D9F7C
BE97F1B1 B1863AEC 7B40D901 576230BD 69EF8F6A EAFEB2B0
9219FA8F AF33376B 42B1B2AA 9EF68D79 DAAB89AF 3FABE49A
CC27863B 707345BB F15344ED 79F7F439 0EF8AC50 B956F39A
98566527 A413C3BD 5E0558C1 59927DB0 E88454A5 D96471FD
DCB56D5B B06BFA34 0E7A1511 EF1CA6FA 57B27663 B1B95D8C
8583D3E4 770536B8 4F01F7E0 E6FBF176 601A0266 941A17B0
CB97F4E 74C2C1FF C7278919 777940C1 E1FF1D8D A637D6B9
9DDAFE5E 17611802 E2C778C1 BE8B41D9 6379A513 60D977FD
4435A11C 308FE7EE 6F1AAD9D B28C81AD DE1A7AEF 7CCE011C
30DA3E7E EB736483 BD6C8E93 48FFBF7F 2CC6587D 60C6368E
57FF0984 C289C938 5A998649 DE21B8A2 7A7EA229 716BA6E9
B279710F 38FAA5FF AE574155 CE4EB4F 743695E2 911B1D06
5E2E90CB CD8F656D 0EDFCD21 6AA22427 055E6835 FD29EEF7
9E0D9077 1F6ACEBE 12F20E95 B34F0F78 B737A961 8B26FA7D
BC9874F2 72C42DBB 56E3EAF1 6B4F6B6C 3B1E78E AA81A002
43FAAADD2 BF18E63D 389AE443 7DDA18C5 76B50F00 96CF3419
5483B005 48C09862 36E3BC7C BB68001C 0494CCD1 99E55C6D
0D0E9DC9 BB9A001E 15276754 FCC68566 054148E6 E764BEE7
C764DAA0 3FC45235 A60DA428 FA20C170 E345003F 2F06EC81
0F5EB25B 22B1B63D 27338E96 1C29951D 11DD2221 657A9F53
1D2A2A19 4DBB1264 48BDEE2B 58E07EA6 59C74619 A6380E1D
66D6382B FE67F638 CD8F8EEF 2723020F 9C40A3FD A67EDA3B
D29238FB D4DB488 5C2A9917 6DB1A06C 50077849 1A8288F1
855F60FF FCF1D137 3FD94FC6 0C1811E1 AC3F1C6D 003BECDA
3B1F2725 CA595DE0 CA63328F 3BE577C9 77556011 95140D8F
59D39CE0 91308B41 05746D4C 23D33E5F 7CE4848D A316A9C6
6B9581BA 3573BFAF 31149618 8AB15423 2B3EE416 DC2A19C5
724FA91A E4ADCC8B C66796EA E5677A01 64E8C098 63139582
2D9DBBFC EE35C06B 5D79A0E7 4D6B8F34 B1534AA3 6A18B0E0
D20EA8B6 BC9C6D6A 5207194E 67FA3555 1B586026 780641C
0F212D18 ECA8B732 7ED91FE7 64A84EAE 1B43FF5B4 F6E8E62F
05C661DE FB258877 C35B18A1 51D5C41A AAA0975A 3E499332
E596078E 600E8814 149C441C E95782F2 2A282563 C7BAC141
1423605D 1AE1F9AE 2CB66060 237EC128 AA0E3436 4E435811
5DBA4CC3 B523073A 28D45498 84B81FF7 0B10FB3F 1C137296
28D5348B 0721117E 4CF4F18B 286090BD B124B066 D6CD4AFC
EADCO0CA 446CE050 50FF1B53 D2BBF118 C1F0CE2A 1F97D22B
8F7E4670 5D45274F 5B42AEEF 39585337 6F697DD5 FDF2C518
7D7D5F0E 2EB8D43F 178A07FC 60FF437F 53DF2E29 9833BF86
CBE88EA4 FBD4221E 84117283 54FA30A7 008F154A 41C7FC46
6B4645DB E232126 7FFFFFFFF FFFFFFFF
```

The estimated symmetric-equivalent strength of this group is 192 bits.
Peers using ffdhe8192 that want to optimize their key exchange with a short exponent (Section 5.2) should choose a secret key of at least 384 bits.

Author's Address

Daniel Kahn Gillmor
ACLU
125 Broad Street, 18th Floor
New York, NY 10004
USA

Email: dkg@fifthhorseman.net