Internet Engineering Task Force   D. Gillmor
Internet-Draft   ACLU
Intended status: Informational   July 22, 2014
Expires: January 23, 2015

Negotiated Finite Field Diffie-Hellman Ephemeral Parameters for TLS
draft-ietf-tls-negotiated-ff-dhe-00

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 establishing a registry of DH parameters with known structure and a mechanism for peers to indicate support for these groups.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Traditional TLS [RFC5246] offers a Diffie-Hellman ephemeral (DHE) key exchange mode which provides Perfect Forward Secrecy for the connection. The client offers a ciphersuite in the ClientHello that includes DHE, and the server offers the client group parameters g and

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 it for other reasons, it 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. Some widely-distributed TLS clients are not capable of DH groups where p > 1024. 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 DHE parameters with insufficient knowledge.

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

This extension solves these problems with a registry of groups of known reasonable structure, an extension for clients to advertise support for them and servers to select them, and guidance for compliant peers to take advantage of the additional security, availability, and efficiency offered.

The use of this extension 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 term "DHE" is 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 ephemeral key exchanges, but this document does not discuss their use. Mentions of DHE here refer strictly to finite-field-based DHE, and not to ECDHE.

2. 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 client SHOULD send an extension of type "negotiated_ff_dhe_groups" in the ClientHello, indicating a list of known finite field Diffie-Hellman groups, ordered from most preferred to least preferred.

The "extension_data" field of this extension SHALL contain "FiniteFieldDHEGroups" where:

```
      enum {
          ffdhe2432(0), ffdhe3072(1), ffdhe4096(2),
          ffdhe6144(3), ffdhe8192(4), (255)
      } FiniteFieldDHEGroup;

      struct {
          FiniteFieldDHEGroup finite_field_dhe_group_list<1..2^8-1>;
      } FiniteFieldDHEGroups;
```

A client that offers this extension SHOULD include at least one DHE-key-exchange ciphersuite in the Client Hello.

The known groups defined by the FiniteFieldDHEGroup registry are listed in Appendix A. These 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 that they 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 8.4 for reasons that this draft does not reuse pi.

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

3. Server Behavior

A TLS server MUST NOT send the NegotiatedDHParams extension to a client that does not offer it first.

A compatible TLS server that receives this extension from a client SHOULD NOT select a DHE ciphersuite if it is unwilling to use one of the DH groups named by the client. In this case, it SHOULD select an acceptable non-DHE ciphersuite from the client's offered list. If the extension is present, none of the client's offered groups are acceptable by the server, and none of the client's proposed non-DHE ciphersuites are acceptable to the server, the server SHOULD end the connection with a fatal TLS alert of type insufficient_security.

A compatible TLS server that receives this extension from a client and selects a DHE-key-exchange ciphersuite selects one of the offered groups and indicates it to the client in the ServerHello by sending a "negotiated_ff_dhe_groups" extension. The "extension_data" field of this extension on the server side should be a single one-byte value FiniteFieldDHEGroup.

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

If a non-anonymous DHE ciphersuite is chosen, and the TLS client has used this extension to offer a DHE group of comparable or greater strength than the server's public key, the server SHOULD select a DHE 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 ffdhe2432 and ffdhe4096, the server SHOULD select ffdhe4096.

3.1. ServerDHParams changes

When the server sends the "negotiated_ff_dhe_groups" extension in the ServerHello, the ServerDHParams member of the subsequent ServerKeyExchange message should indicate a one-byte zero value (0) in place of dh_g and the identifier of the named group in place of dh_p, represented as a one-byte value. dh_Ys must be transmitted as normal.

This re-purposing of dh_p and dh_g is unambiguous: there are no groups with a generator of 0, and no implementation should accept a modulus of size < 9 bits. This change serves two purposes:

The size of the handshake is reduced (significantly, in the case of a large prime modulus).

The signed struct should not be re-playable in a subsequent key exchange that does not indicate named DH groups.

4. Optimizations

In a successfully negotiated finite field DH group key exchange, 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.

4.1. Checking the Peer's Public Key

Peers should validate the 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 Y against each factor.

4.2. 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*2432 multiplications for a ffdhe2432 handshake, each peer can choose to do 2*2*224 multiplications by choosing their secret exponent in the range [2,2^224] and still keep the approximate 112-bit security level.

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

4.3. Table Acceleration

Peers wishing to further accelerate DHE 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 exponentiation using the peer's public exponent as a base still needs to be done as normal).

5. Open Questions

[This section should be removed, and questions resolved, before any formalization of this draft]

5.1. Server Indication of support

Some servers will support this extension, but for whatever reason decide to not negotiate a ciphersuite with DHE key exchange at all. Some possible reasons include:

The client indicated that a server-supported non-DHE ciphersuite was preferred over all DHE ciphersuites, and the server honors that preference.

The server prefers a client-supported non-DHE ciphersuite over all DHE ciphersuites, and selects it unilaterally.

The server would have chosen a DHE ciphersuite, but none of the client's offered groups are acceptable to the server,

Clients will not know that such a server supports the extension.

Should we offer a way for a server to indicate its support for this extension to a compatible client in this case?

Should the server have a way to advertise that it supports this extension even if the client does not offer it?

5.2. Normalizing Weak Groups

Is there any reason to include a weak group in the list of groups? Most DHE-capable peers can already handle 1024-bit DHE, and therefore 1024-bit DHE does not need to be negotiated. Properly-chosen 2432-bit DH groups should be roughly equivalent to 112-bit security. And future implementations should use sizes of at least 3072 bits according to [ENISA].

5.3. Arbitrary Groups

This spec currently doesn't indicate any support for groups other than the named groups. Other DHE specifications have moved away from staticly-named groups with the explicitly-stated rationale of reducing the incentive for precomputation-driven attacks on any specific group (e.g. section 1 of [RFC4419]). However, arbitrary large groups are expensive to transmit over the network and it is computationally infeasible for the client to verify their structure during a key exchange. If we instead allow the server to propose

arbitrary groups, we could make it a MUST that the generated groups use safe prime moduli, while still allowing clients to signal support (and desire) for large groups. This leaves the client in the position of relying on the server to choose a strong modulus, though.

Note that in at least one known attack against TLS [SECURE-RESUMPTION], a malicious server uses a deliberately broken finite field DHE group to impersonate the client to a different server.

6. Acknowledgements

Thanks to Fedor Brunner, Dave Fergemann, Sandy Harris, Watson Ladd, Nikos Mavrogiannopolous, Niels Moeller, Kenny Paterson, and Tom Ritter for their comments and suggestions on this draft. Any mistakes here are not theirs.

7. IANA Considerations

This document defines a new TLS extension, "negotiated_dh_group", assigned a value of XXX from the TLS ExtensionType registry defined in section 12 of [RFC5246]. This value is used as the extension number for the extensions in both the client hello message and the server hello message.

Appendix A defines a TLS Finite Field DHE Named Group Registry. Each entry in this registry 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. This registry may be updated by the addition of new finite field groups, and by reassessments of the security level or utility to TLS of any already present group. Updates are made by IETF Review [RFC5226], and should consider Section 9.1.

8. Security Considerations

8.1. Negotiation resistance to active attacks

Because the contents of this 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-DHE ciphersuite. This is no different than MITM ciphersuite filtering.

Pretend that a compatible server is actually non-compatible by negotiating a DHE ciphersuite and no extension, with an explicit (perhaps weak) group chosen by the server. [XXX what are the worst consequences in this case? What might the client leak before it notices that the handshake fails? XXX]

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

Pretend that a compatible client is not compliant (e.g. by not offering this extension). This could cause the server to negotiate a weaker DHE group during the handshake, but it would fail to complete during the final check of the Finished message.

Pretend that a non-compatible client is compatible. This could cause the server to send what appears to be an extremely odd ServerDHParams (see Section 3.1), and the check in the Finished message would fail. It is not clear how this could be an attack.

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.

8.2. DHE only

Note that this extension 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 cover elliptic curve-based DHE key exchange, which has its own list of named groups.

8.3. 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 therefore compromise the confidentiality and integrity of the TLS session).

Revisions of this extension or updates 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.

8.4. 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.

Other lists of named FF DHE groups are not collected in a single IANA registry, or are mixed with non-FF DHE groups, which makes them inconvenient for re-use in a TLS DHE key exchange context.

8.5. 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).

8.6. Replay attacks from non-negotiated FF DHE

[SECURE-RESUMPTION] shows a malicious peer using a bad FF DHE 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-DHE 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 FF DHE 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 FF DHE 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. Privacy Considerations

9.1. Client fingerprinting

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

10. References

10.1. Normative References

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

10.2. Informative References

[ECRYPTII]
European Network of Excellence in Cryptology II, "ECRYPT II Yearly Report on Algorithms and Keysizes (2011-2012)", September 2012, <http://www.ecrypt.eu.org/documents/D.SPA.20.pdf>.

[ENISA]   European Union Agency for Network and Information Security Agency, "Algorithms, Key Sizes and Parameters Report, version 1.0", October 2013, <http://www.enisa.europa.eu/activities/identity-and-trust/library/deliverables/algorithms-key-sizes-and-parameters-report>.

[PANOPTICLICK]
Electronic Frontier Foundation, "Panopticlick: How Unique - and Trackable - Is Your Browser?", 2010, <https://panopticlick.eff.org/>.

[RFC3526] Kivinen, T. and M. Kojo, "More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)", RFC 3526, May 2003.

[RFC4419] Friedl, M., Provos, N., and W. Simpson, "Diffie-Hellman Group Exchange for the Secure Shell (SSH) Transport Layer Protocol", RFC 4419, March 2006.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[SECURE-RESUMPTION] Delignat-Lavaud, A., Bhargavan, K., and A. Pironti, "Triple Handshakes Considered Harmful: Breaking and Fixing Authentication over TLS", March 2014, <https://secure-resumption.com/>.

[SESSION-HASH] Bhargavan, K., Delignat-Lavaud, A., Pironti, A., Langley, A., and M. Ray, "Triple Handshakes Considered Harmful: Breaking and Fixing Authentication over TLS", March 2014, <https://secure-resumption.com/draft-bhargavan-tls-session-hash-00.txt>.

[STRONGSWAN-IKE] Brunner, T. and A. Steffen, "Diffie Hellman Groups in IKEv2 Cipher Suites", October 2013, <https://wiki.strongswan.org/projects/strongswan/wiki/IKEv2CipherSuites#Diffie-Hellman-Groups>.

Appendix A. Named Group Registry

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 } * 2^64 - 1
```

New additions 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.

A.1. ffdhe2432

The 2432-bit group has registry value 0, and is calcluated from the following formula:

The modulus is: p = 2^2432 - 2^2368 + {[2^2302 * e] + 2111044} * 2^64 - 1

Its hexadecimal representation is:

```
 FFFFFFFF FFFFFFFF ADF85458 A2BB4A9A AFDC5620 273D3CF1
 D8B9C583 CE2D3695 A9E13641 146433FB CC939DCE 249B3EF9
 7D2FE363 630C75D8 F681B202 AEC4617A D3DF1ED5 D5FD6561
 2433F51F 5F066ED0 85636555 3DED1AF3 B557135E 7F57C935
 984F0C70 E0E68B77 E2A689DA F3EFE872 1DF158A1 36ADE735
 30ACCA4F 483A797A BC0AB182 B324FB61 D108A94B B2C8E3FB
 B96ADAB7 60D7F468 1D4F42A3 DE394DF4 AE56EDE7 6372BB19
 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
 AEFE1309 8533C8B3 FFFFFFFF FFFFFFFF
```

The generator is: g = 2

The group size is (p-1)/2

The estimated symmetric-equivalent strength of this group is 112 bits.

Peers using ffdhe2432 that want to optimize their key exchange with a short exponent (Section 4.2) should choose a secret key of at least 224 bits.

A.2. ffdhe3072

The 3072-bit prime has registry value 1, and is calcluated from the following formula:

p = 2^3072 - 2^3008 + {[2^2942 * e] + 2625351} * 2^64 -1

Its hexadecimal representation is:

```
    FFFFFFFF FFFFFFFF ADF85458 A2BB4A9A AFDC5620 273D3CF1
    D8B9C583 CE2D3695 A9E13641 146433FB CC939DCE 249B3EF9
    7D2FE363 630C75D8 F681B202 AEC4617A D3DF1ED5 D5FD6561
    2433F51F 5F066ED0 85636555 3DED1AF3 B557135E 7F57C935
    984F0C70 E0E68B77 E2A689DA F3EFE872 1DF158A1 36ADE735
    30ACCA4F 483A797A BC0AB182 B324FB61 D108A94B B2C8E3FB
    B96ADAB7 60D7F468 1D4F42A3 DE394DF4 AE56EDE7 6372BB19
    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
    AEFE1309 85139270 B4130C93 BC437944 F4FD4452 E2D74DD3
    64F2E21E 71F54BFF 5CAE82AB 9C9DF69E E86D2BC5 22363A0D
    ABC52197 9B0DEADA 1DBF9A42 D5C4484E 0ABCD06B FA53DDEF
    3C1B20EE 3FD59D7C 25E41D2B 66C62E37 FFFFFFFF FFFFFFFF
```

The generator is: g = 2

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

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 4.2) should choose a secret key of at least 250 bits.

A.3. ffdhe4096

The 4096-bit group has registry value 2, and is calcluated from the following formula:

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

Its hexadecimal representation is:

```
    FFFFFFFF FFFFFFFF ADF85458 A2BB4A9A AFDC5620 273D3CF1
    D8B9C583 CE2D3695 A9E13641 146433FB CC939DCE 249B3EF9
    7D2FE363 630C75D8 F681B202 AEC4617A D3DF1ED5 D5FD6561
    2433F51F 5F066ED0 85636555 3DED1AF3 B557135E 7F57C935
    984F0C70 E0E68B77 E2A689DA F3EFE872 1DF158A1 36ADE735
    30ACCA4F 483A797A BC0AB182 B324FB61 D108A94B B2C8E3FB
    B96ADAB7 60D7F468 1D4F42A3 DE394DF4 AE56EDE7 6372BB19
    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
    AEFE1309 85139270 B4130C93 BC437944 F4FD4452 E2D74DD3
    64F2E21E 71F54BFF 5CAE82AB 9C9DF69E E86D2BC5 22363A0D
    ABC52197 9B0DEADA 1DBF9A42 D5C4484E 0ABCD06B FA53DDEF
    3C1B20EE 3FD59D7C 25E41D2B 669E1EF1 6E6F52C3 164DF4FB
    7930E9E4 E58857B6 AC7D5F42 D69F6D18 7763CF1D 55034004
    87F55BA5 7E31CC7A 7135C886 EFB4318A ED6A1E01 2D9E6832
    A907600A 918130C4 6DC778F9 71AD0038 092999A3 33CB8B7A
    1A1DB93D 7140003C 2A4ECEA9 F98D0ACC 0A8291CD CEC97DCF
    8EC9B55A 7F88A46B 4DB5A851 F44182E1 C68A007E 5E655F6A
    FFFFFFFF FFFFFFFF
```

The base is: g = 2

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

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 4.2) should choose a secret key of at least 300 bits.

A.4. ffdhe6144

The 6144-bit group has registry value 3, and is calcluated from the following formula:

The modulus is: p = 2^6144 - 2^6080 + {[2^6014 * e] + 15705020} * 2^64 - 1

Its hexadecimal representation is:

```
    FFFFFFFF FFFFFFFF ADF85458 A2BB4A9A AFDC5620 273D3CF1
    D8B9C583 CE2D3695 A9E13641 146433FB CC939DCE 249B3EF9
    7D2FE363 630C75D8 F681B202 AEC4617A D3DF1ED5 D5FD6561
    2433F51F 5F066ED0 85636555 3DED1AF3 B557135E 7F57C935
    984F0C70 E0E68B77 E2A689DA F3EFE872 1DF158A1 36ADE735
    30ACCA4F 483A797A BC0AB182 B324FB61 D108A94B B2C8E3FB
    B96ADAB7 60D7F468 1D4F42A3 DE394DF4 AE56EDE7 6372BB19
    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
    AEFE1309 85139270 B4130C93 BC437944 F4FD4452 E2D74DD3
    64F2E21E 71F54BFF 5CAE82AB 9C9DF69E E86D2BC5 22363A0D
    ABC52197 9B0DEADA 1DBF9A42 D5C4484E 0ABCD06B FA53DDEF
    3C1B20EE 3FD59D7C 25E41D2B 669E1EF1 6E6F52C3 164DF4FB
    7930E9E4 E58857B6 AC7D5F42 D69F6D18 7763CF1D 55034004
    87F55BA5 7E31CC7A 7135C886 EFB4318A ED6A1E01 2D9E6832
    A907600A 918130C4 6DC778F9 71AD0038 092999A3 33CB8B7A
    1A1DB93D 7140003C 2A4ECEA9 F98D0ACC 0A8291CD CEC97DCF
    8EC9B55A 7F88A46B 4DB5A851 F44182E1 C68A007E 5E0DD902
    0BFD64B6 45036C7A 4E677D2C 38532A3A 23BA4442 CAF53EA6
    3BB45432 9B7624C8 917BDD64 B1C0FD4C B38E8C33 4C701C3A
    CDAD0657 FCCFEC71 9B1F5C3E 4E46041F 388147FB 4CFDB477
    A52471F7 A9A96910 B855322E DB6340D8 A00EF092 350511E3
    0ABEC1FF F9E3A26E 7FB29F8C 183023C3 587E38DA 0077D9B4
    763E4E4B 94B2BBC1 94C6651E 77CAF992 EEAAC023 2A281BF6
    B3A739C1 22611682 0AE8DB58 47A67CBE F9C9091B 462D538C
    D72B0374 6AE77F5E 62292C31 1562A846 505DC82D B854338A
    E49F5235 C95B9117 8CCF2DD5 CACEF403 EC9D1810 C6272B04
    5B3B71F9 DC6B80D6 3FDD4A8E 9ADB1E69 62A69526 D43161C1
    A41D570D 7938DAD4 A40E329C D0E40E65 FFFFFFFF FFFFFFFF
```

The generator is: 2

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

The estimated symmetric-equivalent strength of this group is 175 bits.

Peers using ffdhe6144 that want to optimize their key exchange with a short exponent (Section 4.2) should choose a secret key of at least 350 bits.

A.5. ffdhe8192

The 8192-bit group has registry value 4, and is calcluated from the following formula:

The modulus is: p = 2^8192 - 2^8128 + {[2^8062 * e] + 10965728} * 2^64 - 1

Its hexadecimal representation is:

```
    FFFFFFFF FFFFFFFF ADF85458 A2BB4A9A AFDC5620 273D3CF1
    D8B9C583 CE2D3695 A9E13641 146433FB CC939DCE 249B3EF9
    7D2FE363 630C75D8 F681B202 AEC4617A D3DF1ED5 D5FD6561
    2433F51F 5F066ED0 85636555 3DED1AF3 B557135E 7F57C935
    984F0C70 E0E68B77 E2A689DA F3EFE872 1DF158A1 36ADE735
    30ACCA4F 483A797A BC0AB182 B324FB61 D108A94B B2C8E3FB
    B96ADAB7 60D7F468 1D4F42A3 DE394DF4 AE56EDE7 6372BB19
    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
    AEFE1309 85139270 B4130C93 BC437944 F4FD4452 E2D74DD3
    64F2E21E 71F54BFF 5CAE82AB 9C9DF69E E86D2BC5 22363A0D
    ABC52197 9B0DEADA 1DBF9A42 D5C4484E 0ABCD06B FA53DDEF
    3C1B20EE 3FD59D7C 25E41D2B 669E1EF1 6E6F52C3 164DF4FB
    7930E9E4 E58857B6 AC7D5F42 D69F6D18 7763CF1D 55034004
    87F55BA5 7E31CC7A 7135C886 EFB4318A ED6A1E01 2D9E6832
    A907600A 918130C4 6DC778F9 71AD0038 092999A3 33CB8B7A
    1A1DB93D 7140003C 2A4ECEA9 F98D0ACC 0A8291CD CEC97DCF
    8EC9B55A 7F88A46B 4DB5A851 F44182E1 C68A007E 5E0DD902
    0BFD64B6 45036C7A 4E677D2C 38532A3A 23BA4442 CAF53EA6
    3BB45432 9B7624C8 917BDD64 B1C0FD4C B38E8C33 4C701C3A
    CDAD0657 FCCFEC71 9B1F5C3E 4E46041F 388147FB 4CFDB477
    A52471F7 A9A96910 B855322E DB6340D8 A00EF092 350511E3
    0ABEC1FF F9E3A26E 7FB29F8C 183023C3 587E38DA 0077D9B4
    763E4E4B 94B2BBC1 94C6651E 77CAF992 EEAAC023 2A281BF6
    B3A739C1 22611682 0AE8DB58 47A67CBE F9C9091B 462D538C
    D72B0374 6AE77F5E 62292C31 1562A846 505DC82D B854338A
    E49F5235 C95B9117 8CCF2DD5 CACEF403 EC9D1810 C6272B04
    5B3B71F9 DC6B80D6 3FDD4A8E 9ADB1E69 62A69526 D43161C1
    A41D570D 7938DAD4 A40E329C CFF46AAA 36AD004C F600C838
    1E425A31 D951AE64 FDB23FCE C9509D43 687FEB69 EDD1CC5E
    0B8CC3BD F64B10EF 86B63142 A3AB8829 555B2F74 7C932665
    CB2C0F1C C01BD702 29388839 D2AF05E4 54504AC7 8B758282
    2846C0BA 35C35F5C 59160CC0 46FD8251 541FC68C 9C86B022
    BB709987 6A460E74 51A8A931 09703FEE 1C217E6C 3826E52C
    51AA691E 0E423CFC 99E9E316 50C1217B 624816CD AD9A95F9
    D5B80194 88D9C0A0 A1FE3075 A577E231 83F81D4A 3F2FA457
    1EFC8CE0 BA8A4FE8 B6855DFE 72B0A66E DED2FBAB FBE58A30
    FAFABE1C 5D71A87E 2F741EF8 C1FE86FE A6BBFDE5 30677F0D
    97D11D49 F7A8443D 0822E506 A9F4614E 011E2A94 838FF88C
    D68C8BB7 C5C6424C FFFFFFFF FFFFFFFF
```

The base is: g = 2

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

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 4.2) 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