Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: September 29, 2014

D. Gillmor
ACLU

March 28, 2014

# Negotiated Discrete Log Diffie-Hellman Ephemeral Parameters for TLS
## draft-gillmor-tls-negotiated-dl-dhe-00

Abstract

Traditional discrete logarithm-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.

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].

## 2. Client Behavior

A TLS client that is capable of using strong discrete log 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_dl_dhe_groups" in the ClientHello, indicating an list of

known discrete log Diffie-Hellman groups, ordered from most preferred to least preferred.

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

```
enum {
    dldhe2432(0), dldhe3072(1), dldhe4096(2),
    dldhe6144(3), dldhe8192(4), (255)
} DiscreteLogDHEGroup;

struct {
    DiscreteLogDHEGroup discrete_log_dhe_group_list<1..2^8-1>;
} DiscreteLogDHEGroups;
```

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 DiscreteLogDHEGroup registry are listed in Appendix A. These are all safe primes, designed to be sparse, and with the high and low 64 bits set to 1 for efficient Montgomery or Barrett reduction.

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_dl_dhe_groups" extension. The "extension_data" field of this extension on the server side should be a single one-byte value DiscreteLogDHEGroup.

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

When the server sends the "negotiated_dl_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 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 discrete log 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 two 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 its factors, and test Y against each of its factors.

### 4.2. Short Exponents

Traditional Discrete Log 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*2048 multiplications for a dldhe2048 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 used 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 discrete log DHE group to impersonate the client to a different server.

## 6. Acknowledgements

Thanks to Tom Ritter and Nikos Mavrogiannopolous and Niels Moeller and Kenny Paterson for their comments and suggestions on the idea for 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.

This extension also defines a registry of TLS named Discrete Log DH groups, derived initially from some of the IKE DH groups [RFC3526], indicating the advised strength of each group and whether it is recommended for use in TLS. These recommendations may be updated by future revisions.

## 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 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. 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 discrete log-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. 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.

### 8.4. Deprecating weak groups

Advances in hardware or in discrete log cryptanalysis may cause some of the negotiated groups to not provide the desired security margins, as indicated by number of years to protect the premaster secret (and therefore the confidentiality and integrity of the TLS session) against a powerful adversary.

Revisions of this extension or updates should mark known-weak groups as explicitly deprecated, and implementations that require strong confidentiality and integrity guarantees should avoid using any deprecated groups.

### 8.5. Choice of groups

Other lists of named discrete log 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.

The IKE groups do not have sparse moduli, which makes modular exponentiation less efficient.

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 reuse keys; however, using different groups avoids these failure modes entirely.

The DL DHE groups are not collected in a single IANA registry, or are mixed with non-DL DHE groups, which makes them inconvenient for re-use in TLS.

### 8.6. Timing attacks

Any implementation of discrete log Diffie-Hellman key exchange should use constant-time modular-exponentiation implementations. This is particularly true for those implementations that ever re-use DHE parameters (so-called "semi-static" ephemeral keying).

### 8.7. Replay attacks from non-negotiated DL DHE

[SECURE-RESUMPTION] shows a malicious peer using a bad DL 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 DL 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 DL 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. References

### 9.1. Normative References

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

### 9.2. Informative References

[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.

[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

### A.1. dldhe2432

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

The modulus is: p = 2^2432 - 2^2368 + 2332920 * 2^64 - 1

Its hexadecimal representation is:

```
FFFFFFFF FFFFFFFF 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000
```

```
00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000
00000000 002398F7 FFFFFFFF FFFFFFFF
```

The generator is: g = 2

The group size is (p-1)/2

Peers using dldhe2432 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. dldhe3072

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

p = 2^3072 - 2^3008 + 425754 * 2^64 -1

Its hexadecimal representation is:

```
FFFFFFFF FFFFFFFF 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00067F19 FFFFFFFF FFFFFFFF
```

The generator is: g = 2

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

Peers using dldhe3072 that want to optimize their key exchange with a short exponent (Section 4.2) should choose a secret key of at least 256 bits.

### A.3. dldhe4096

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

The modulus is: p = 2^4096 - 2^4032 + 341664 * 2^64 - 1

Its hexadecimal representation is:

```
FFFFFFFF FFFFFFFF 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 0005369F
FFFFFFFF FFFFFFFF
```

The base is: g = 2

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

Peers using dldhe4096 that want to optimize their key exchange with a short exponent (Section 4.2) should choose a secret key of at least XXX bits.

### A.4. dldhe6144

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

The modulus is: p = 2^6144 - 2^6080 + XXX * 2^64 - 1

Its hexadecimal representation is:

XXX ...still calculating... XXX

The generator is: 2

Peers using dldhe6144 that want to optimize their key exchange with a short exponent (Section 4.2) should choose a secret key of at least XXX bits.

### A.5. dldhe8192

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

The modulus is: p = 2^8192 - 2^8128 + XXX * 2^64 - 1

Its hexadecimal representation is:

XXX ...still calculating... XXX

The base is: g = 2

Peers using dldhe8192 that want to optimize their key exchange with a short exponent (Section 4.2) should choose a secret key of at least XXX bits.

Author's Address

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

Email: dkg@fifthhorseman.net