
Key Share Prediction

— draft-davidben-tls-key-share-prediction —

David Benjamin

(EC)DHE* negotiation

Client sends *two* extensions

supported_groups — list of preferred NamedGroups

key_share — keys for some subset of supported_groups

Server picks some supported group

Sends ServerHello if the group is in key_share

Sends HelloRetryRequest otherwise (extra round-trip)

Policies left up to implementation...

...but normally semantics of fields are specified

* Now also postquantum KEMs

(EC)DHE* negotiation

Client sends *two* extensions

supported_groups — list of preferred NamedGroups

key_share — keys for some subset of supported_groups

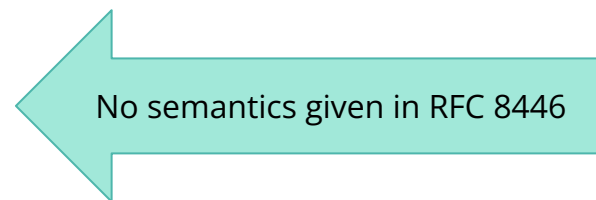
Server picks some supported group

Sends ServerHello if the group is in key_share

Sends HelloRetryRequest otherwise (extra round-trip)

Policies left up to implementation...

...but normally semantics of fields are specified



* Now also postquantum KEMs

Is this server behavior okay?

"key_share first"

1. Pick a group out of key_shares, send ServerHello
2. Otherwise, pick a group out of supported_groups, send HelloRetryRequest

Compare with

"supported_groups first"

1. Pick a group out of supported_groups
2. If group is in key_shares, ServerHello. Otherwise, HelloRetryRequest

Clients always predict
their most preferred
groups, right?

Imagine the *next* postquantum transition

Consider three groups: AwesomeNewKEM, Kyber, X25519

AwesomeNewKEM and Kyber are postquantum, X25519 is classical

Sending two PQ KEMs is expensive

Client will not predict both

Server might support AwesomeNewKEM or Kyber

Client needs prior knowledge to guess — how?

Uncommon groups

Client predicts AwesomeNewKEM over Kyber because it's more common

Server only supports Kyber

Client

supported_groups

{AwesomeNewKEM, Kyber, X25519}

key_share

{AwesomeNewKEM, X25519}

Server

supported_groups

{Kyber, X25519}

Result

X25519 🙄

Out-of-band prediction

Fix this by adding DNS or other key share hint

Attacker hints X25519, but actually server supports Kyber

Client

supported_groups

{Kyber, X25519}

key_share

{X25519}

Server

supported_groups

{Kyber, X25519}

Result

X25519 🙄

Compatibility hacks*

Client implements compatibility hack by sometimes not predicting Kyber

Attacker triggers this path even though server supports Kyber

Client

supported_groups
 {Kyber, X25519}

key_share
 {X25519}

Server

supported_groups
 {Kyber, X25519}

Result
 X25519 🙄

* Not recommended; let's please avoid this if we can

Equipreference

Server has no security preference between X25519 and P-256

Picking the one without HelloRetryRequest is faster

Client

```
supported_groups
  {X25519, P-256}
key_share
  {X25519}
```

Client

```
supported_groups
  {X25519, P-256}
key_share
  {P-256}
```

Server

```
supported_groups
  {X25519, P-256}
Result
  X25519 🙌
```

Server

```
supported_groups
  {X25519, P-256}
Result
  P-256 🙌
```

draft-davidben-tls-key-share-prediction

Clarify protocol semantics:

- Client key shares are predictions, not preferences

- Servers cannot assume key share list reflects preferences

Introduce SvcParamKey to hint server preferences in DNS

Backwards compatibility

Older TLS 1.3 servers exist

Define “prediction-safe” vs “prediction-unsafe” groups

- Existing groups are prediction-unsafe

- Newer groups are prediction-safe, require the new server behavior

Prediction tricks only apply to prediction-safe groups

- Prediction-unsafe key shares must reflect preferences

- Limit DNS hint to prediction-safe groups

Make all postquantum KEMs prediction-safe

Is this server behavior okay?

1. Pick a group out of `key_shares`, send `ServerHello`
2. Otherwise, pick a group out of `supported_groups`, send `HelloRetryRequest`

Answer

Only if you have *no preference* between *any* of your supported groups

Definitely not okay if you support both postquantum and classical options

Questions?

<https://datatracker.ietf.org/doc/draft-davidben-tls-key-share-prediction/>

<https://github.com/davidben/tls-key-share-prediction>