receive_generation field in KeyUpdate

Keith Winstein

July 19, 2016

Mosh (mobile shell) Project and Stanford University
keithw@cs.stanford.edu

Background

"The `KeyUpdate` handshake message is used to indicate that the sender is updating its sending cryptographic keys."

"Upon receiving a `KeyUpdate`, the receiver MUST update their receiving keys and if they have not already updated their sending state up to or past the then current receiving generation MUST send their own `KeyUpdate` prior to sending any other messages. This mechanism allows either side to force an update to the entire connection."

Effect of KeyUpdate

`KeyUpdate` is an instruction to the other side: "**ratchet your receive key forwards and delete the old key**."

"Also, if you haven't already updated your send key to the same generation, do that and send me back a `KeyUpdate`."

Current KeyUpdate message

```
struct {} KeyUpdate;
```

Proposed KeyUpdate message

```
struct {
    uint64 receive_generation;
} KeyUpdate;
```

Proposed explanatory text

`receive_generation`
: The generation of the receive traffic keys in use by the sender of the `KeyUpdate` message. This value equals the number of `KeyUpdate` messages that the sender has received.

An endpoint MAY use the `receive_generation` field in a received `KeyUpdate` message to confirm that the other side has received and acted upon an earlier `KeyUpdate` message.

Same semantics as before

- Still just as asynchronous.
- Still no blocking.
- Endpoints already have to maintain the traffic-key generations.
- On receipt, no obligation to do anything.
- Many endpoints will probably ignore the field.

Okay, but why?

Q: In what cases does an endpoint want this?

A: If the endpoint wants to know **when** the other side has ratcheted its receive key.

Example use cases

- Embedded devices may want to avoid going to sleep until keys have been ratcheted forwards.

- If paranoid, at end of TLS connection, make sure traffic keys are dead: Send KeyUpdate and wait for incremented `receive_generation`.

- Endpoints can permit read-only auditor without MITM by releasing only superseded traffic keys.

Conclusion

KeyUpdate is an instruction to the other side: "**ratchet your receive key forwards and delete the old key**."

Adding this field gives confirmation that instruction has been carried out.

Keith Winstein
keithw@cs.stanford.edu