More on Alerts

When a server detects a problem with what it has received from a client, it should issue an error alert, and close the connection. Sounds simple doesn't it? This situation is quite likely to occur if the client is required to authenticate.

But there is a problem for the server. The alert needs to be encrypted. However after the client sends its finish message it does not expect any further handshake data from the server, so it goes on to calculate the traffic keys and wait for application data.

But what if the server has not sent application data but rather an alert? Recall that we don't know the record type (alert or application) until *after* we have decrypted it. Problem is that the server at the time it detects the problem is still using handshake keys and has not yet calculated the traffic keys. But the client has already moved on to using the traffic keys.

The solution is for the server to delay sending the alert until after it has calculated the traffic keys, rather than send it immediately. Note this from the RFC ...

"Any records following a Finished message MUST be encrypted under the appropriate application traffic key as described in Section 7.2. In particular, this includes any alerts sent by the server in response to client Certificate and CertificateVerify messages"

But what about the Client Finish message? If there is a problem with it, what to do? Recall that the traffic keys are calculated using the transcript hash, and so any disagreement about the transcript means that the traffic keys will be different.

In fact the traffic keys are NOT calculated using the full transcript, only the transcript up to the end of any client CertificateVerify message. Only the resumption master secret (RMS) uses the full transcript – which makes perfect sense as a usable RMS should not be available if the handshake ultimately fails.

Problem solved

So the solution is to delay sending the alert. If the communication takes place flawlessly, but say the client certificate is rejected, then after the handshake completes the first server-to-client message is actually an alert encrypted using the agreed traffic keys, the server drops the connection, and the client knows why.

If an incorrect HMAC is received by the server in the client finish message, then this does not affect the traffic keys (as the client finish message is not involved in the traffic key calculation), and so the server can again send an alert which the client can decrypt.