The impact of TLS ciphers on an MQ for z/OS channel

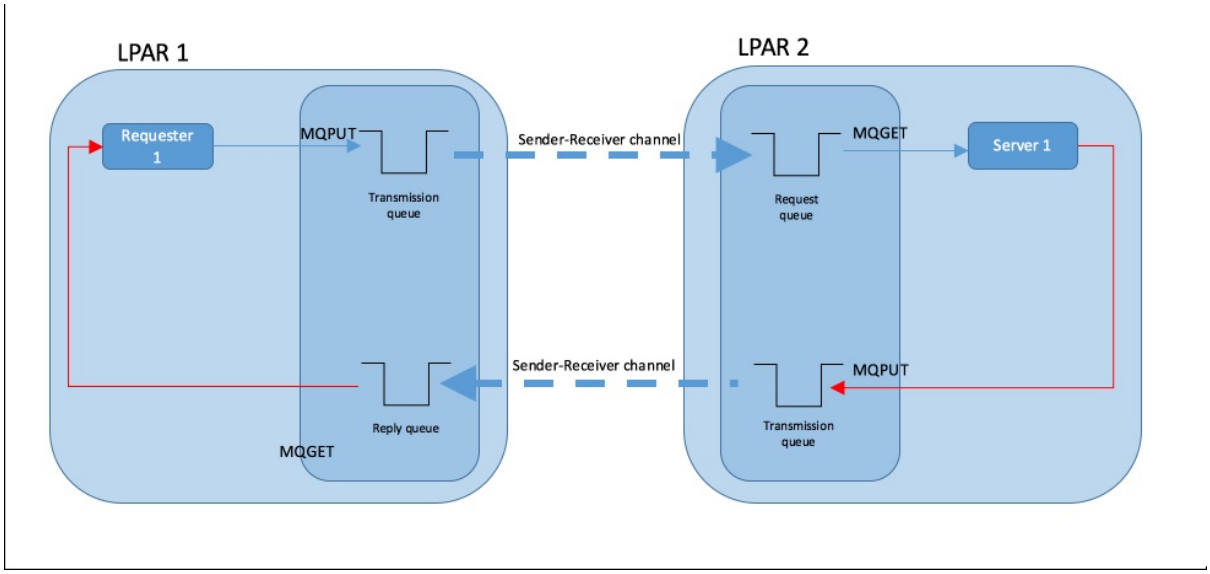
tonySharkey
Published on 04/09/2018

Rightly so, there is a focus on ensuring data flowing over MQ is protected, and there are a number of options for protecting your data such as:

1. TLS cipher specification on the MQ channels
2. AMS policy protection at the time of the MQPUT, whether:
    - Integrity (message signing)
    - Privacy (message signing and encryption)
    - Confidentiality (encryption)
3. Combination of both TLS ciphers and AMS policy protection.

This blog takes a brief look at the impact of applying TLS ciphers to a request/reply workload between 2 z/OS queue managers.

The workload used for the measurements in this blog uses a simple request/reply format as demonstrated in the following diagram.



Request/Reply model using 2 z/OS queue managers

For simplicity, the workload uses 1 requester task and 1 server task – this means that there is only 1 message on the system at any time. This also means that there is no waiting for any of the channel initiator tasks namely adaptor, dispatcher or SSL tasks.

When using ciphers to protect the MQ messages, the maximum size of data that can be send in a single TCP send call is 16KB, therefore we show the impact to the round-trip times for messages of:

- 8KB (single TCP/IP send).
- 32KB (multiple TCP/IP sends – 2 sends for the baseline and 3 sends for the protected measurements).

The measurements shown use:

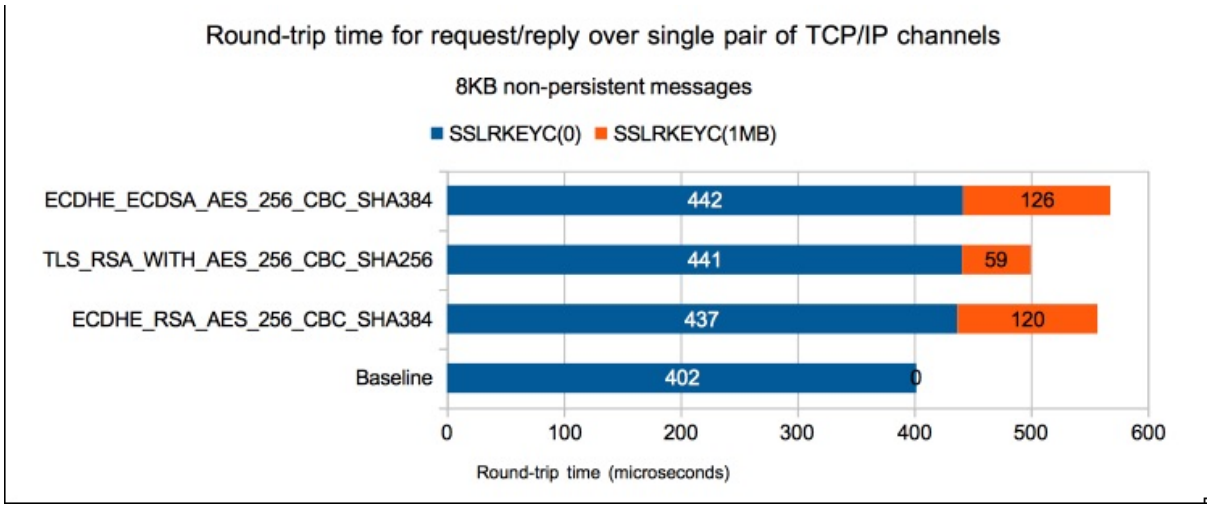
- No cipher (baseline)
- ECDHE_RSA_AES_256_CBC_SHA384
- TLS_RSA_WITH_AES_256_CBC_SHA256
- ECDHE_ECDSA_AES_256_CBC_SHA384

There are 2 configurations, firstly negotiating the secret key only at channel start, and secondly negotiating the secret key every 1MB, i.e. SSLRKEYC(0) and SSLRKEYC(1048576) respectively.

- When using a message of size 8KB, the negotiation occurs approximately every 122 messages.
- When using a message of size 32KB, the negotiation occurs approximately every 32 messages.

8KB Messages

The following chart shows the **round-trip** time for the configurations of SSLRKEYC(0) and SSLRKEYC(1MB):



Round-trip times for 8KB workload

The blue bars indicate the round-trip time when negotiating the secret key only at channel start and the channel being long running.

The act of adding a cipher spec to a request/reply workload with an 8KB workload on low-latency systems added between 35 to 40 microseconds to the round-trip time, i.e.

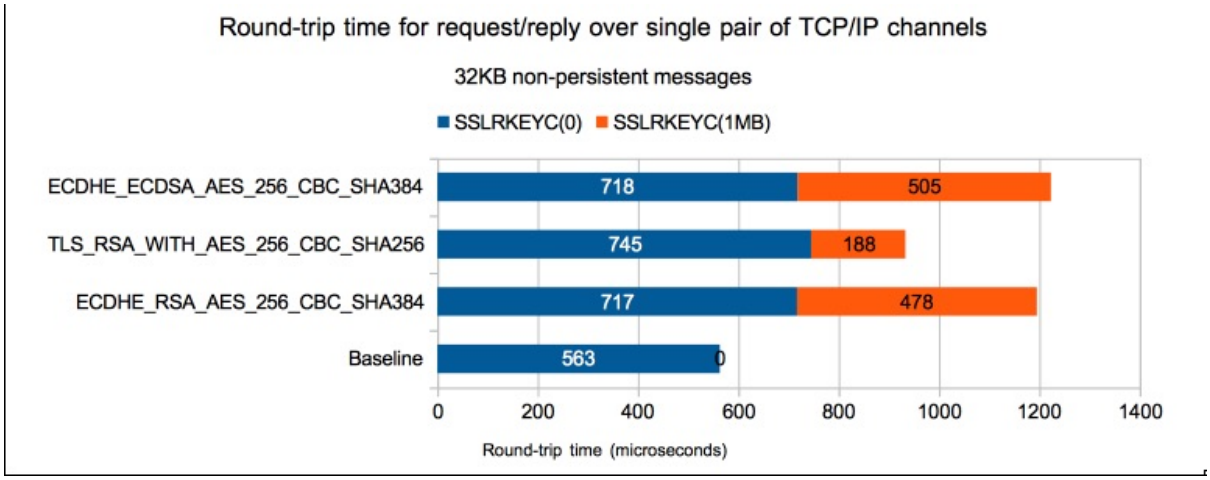
- No cipher took 402 microseconds.
- ECDHE_RSA_AES_256_CBC_SHA384 took 437 microseconds.

In terms of cost, there is additional cost, primarily in the MQ channel initiator SSL tasks, totaling approximately 40 CPU microseconds per round-trip.

Negotiating the secret key adds both cost and latency, as indicated by the red bars on the diagram. These show the impact of the negotiation on the round-trip time, spread over each transaction, i.e. for TLS_RSA_WITH_AES_256_CBC_SHA256 the average increase over 122 transactions is 59 microseconds/transaction, or 3.6 milliseconds per negotiation.
Note: *negotiations are required on both pairs of sender-receiver channels, so the total time spent negotiating the secret key for every 122 transactions is 7.2 milliseconds.*

32KB Messages

The following chart shows the **round-trip** time for the configurations of SSLRKEYC(0) and SSLRKEYC(1MB):



Round-trip times for 32KB workload

Again, the blue bars indicate the round-trip time when negotiating the secret key at channel start and the channel being long running.

The act of adding a cipher spec to a request/reply workload with a 32KB workload on low-latency systems added between 188 to 505 microseconds to the round-trip time, i.e.

- No cipher took 563 microseconds
- ECDHE_RSA_AES_256_CBC_SHA384 took 717 microseconds.

Negotiating the secret key adds both cost and latency, as indicated by the red bars on the diagram. These show the impact of the negotiation on the round-trip time, spread over each transaction, i.e. for TLS_RSA_WITH_AES_256_CBC_SHA256 the average increase over 32 transactions is 188 microseconds/transaction, or 3 milliseconds per negotiation.

Note: *negotiations are required on both pairs of sender-receiver channels, so the total time spent negotiating the secret key for every 32 transactions is 6 milliseconds, which is 20% lower than calculated when using 8KB messages. The higher frequency of negotiation with the 32KB workload will make some difference in keeping the CryptoExpress6S hardware in a 'warmer' state, which may explain some of the reduction in cost.*

Latency added when negotiating the secret key

The time spent performing the secret key negotiation is not affected particularly by the message size. In our measurements, the time spent switching to the CryptoExpress6S card and performing the work on the crypto hardware adds between 6 and 10 milliseconds to the round-trip time **per negotiation**.

Cipher	Elapsed time per negotiation on z14 with CryptoExpress6S
TLS_RSA_WITH_AES_256_CBC_SHA256	6 milliseconds
ECDHE_RSA_AES_256_CBC_SHA256	10 milliseconds
ECDHE_ECDSA_AES_256_CBC_SHA256	10 milliseconds

Conclusions

Applying ciphers to the MQ channels does impact the round-trip and cost of transporting the messages across the MQ network. The impact of encryption can be relatively small (10%) depending on the message size.

Our measurements use a relatively small SSLRKEYC value to drive frequent re-negotiations. If the frequency of secret key negotiation is reduced, the impact of the key negotiation per transaction can be reduced. This will bring the round-trip time closer to that shown by the blue bars in the charts.

Assumptions

These measurements assume:

- Sufficient CPU.
- Sufficient MQ tasks, namely SSLTASK, CHIDISP, CHIADAP.
- CryptoExpress6S hardware is available and not constrained.
- Key size of 2048 (default on z/OS v2r3) – larger keys may add cost and latency.
- Minimal queuing for resource.

Further reference

For further reference on encrypted channels:

- MP16 has a whole section on SSL and TLS.
- MQ for z/OS on z14 discusses the Crypto improvements on z14 and CryptoExpress6S.