A clarification on the multithreading constraints of the EncryptMessage function

devblogs.microsoft.com/oldnewthing/20231109-00

November 9, 2023

Raymond Chen

In the documentation for the `EncryptMessage` function, there is a sentence that goes

> If more than one thread is encrypting, or more than one thread is decrypting, each thread should obtain a unique context.

A customer was looking for clarification on this constraint. Is it saying that if I have two threads, and both of them want to (say) encrypt, then each one should get a separate context, even if the two encryption operations are not simultaneous? Does this mean that each encryption context can be used for encryption on the thread that performed the first encryption? Like, maybe the first call remembers some per-thread state that the second call relies on?

No, that's not what the sentence is saying. Let's look at the sentence in context. It's the second half of a paragraph, and it's following up on a topic introduced by the first half.

> **EncryptMessage** and **DecryptMessage** can be called at the same time from two different threads in a single Security Support Provider Interface (SSPI) context if one thread is encrypting and the other is decrypting. If more than one thread is encrypting, or more than one thread is decrypting, each thread should obtain a unique context.

The topic of the paragraph is simultaneous usage of an SSPI context from multiple threads. It says that you can perform a single encryption concurrent with a single decryption, but you cannot have two concurrent encryptions or two concurrent decryptions.

The text omitted the word "concurrently" because that was implied by the previous sentence. The fully-qualified sentence would be

> If more than one thread is encrypting concurrently, or more than one thread is decrypting concurrently, each thread should obtain a unique context.