Rationale

RSA with OAEP padding was used for our key transport protocol while RSA with SHA256 was used for signing.

AES-128 was used for symmetric encryption because there are no currently known practical attacks. AES-128 is also a recommended encryption scheme by NIST.

HmacSHA256 keyed-hashing algorithm was used to generate MAC. This method allows us to hash messages with a key, which makes the hash function even more reliable.

For the configuration that uses both MAC and Encryption, we are using “Encrypt then MAC” method because it is currently the most secure of the options available. We are also able to detect whether MAC fails or not without wasting energy to decrypt.

Specification:

If MAC is utilized, then Alice will initiate the MAC-key-exchange protocol with Bob without prompting for user input. This is to simplify the process for the user.

We are assuming that the user will not input uncommon character/ character sequences that cannot be printed or encoded to Base64.

Have 2 types of messages: “MacExchange” and “NewMessage”

External Libraries:

Bouncy Castle was used to implement RSA with OAEP padding.

Known Problems:

Because we are signing our keys using the SHA256RSA scheme that is vulnerable against padding oracle attacks, we are assuming that we are actively monitoring requests for things that are signed by our RSA key before it reaches our code.

After counter is off, it will remain offset…

If Mallory replays the Mac-key-exchange message, Bob will just conclude that the MAC is off