COMP 4109 - RSA-OAEP

Danen Van De Ven

December 7, 2015

EME-OAEP

Based on Bellare and Rogaway's Optimal Asymmetric Encryption scheme [1], OAEP is an encryption scheme used to in tandem with the RSA encryption protocol in order to increase security. RSA-OAEP is semantically secure against adaptive chosen plain text attacks, provided the mask generating functions (MGF) is viewed as a black box. Following the guidelines laid out in the PKCS #1 v2.1 [2], a label L, and choice of a hash function and mask generating function can be provided to RSA-OAEP. The process of using RSA with OAEP padding is detailed below (Table 1).

Encryption

RSAES-OAEP-Encrypt((n, e), M, L)

Options:	Hash	hash function ($hlen$ denotes the length of in octets the hash function output)
	MGF	mask generation function
Input:	(n, e)	RSA public key (k denotes the length in octets of the RSA modulus n)
	M	message to be encrypted octet string of length $mLen$
	L	optional label to be associated with the message
Output:	C	ciphertext octet string of length k

Table 1: Variables associated with RSA-OAEP encryption.

1. Length checking:
   (a) The label L is too long if $L > 2^{61} - 1$ octets. (Using SHA-1 hash)
   (b) The message is too long if $mLen > k - 2 \cdot hlen - 2$

2. EME-OAEP encoding:

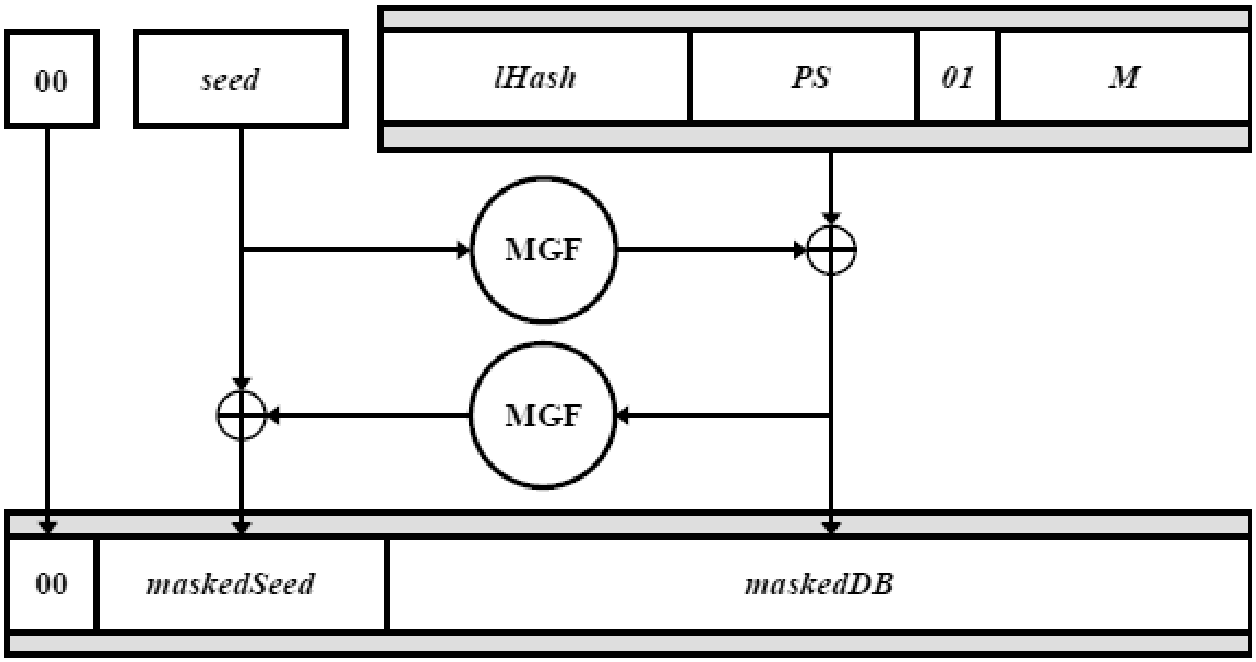

Figure 1: The EME-OAEP encoding process [2].

(a) Let $lHash = Hash(L)$, and octet string of length $hlen$.

(b) PS is an octet string of zeros $k - mLen - 2 \cdot hLen - 2$ in length.

(c) Concatenate $lHash$, PS, 0x01 and the message M.

$$DB = lHash||PS||0\text{x}01||M$$

(d) Generate random octet string $seed$ length $hLen$

(e) $dbMask = MGF(seed, k - hLen - 1)$, same length as DB

(f) $maskedDB = DB \oplus dbMask$

(g) $seedMask = MGF(seed, hLen)$

(h) $maskedSeed = seed \oplus seedMask$

(i) Concatenate one octet of 0x00, $maskedSeed$, and $maskedDB$. This creates an encoded message of length k octets.

$$EM = 0\text{x}00||maskedSeed||maskedDB$$

3. The octet string EM can now be encrypted by converting the value to an integer.

(a) Create message integer representation m.

$$m = OS2IP(EM)$$

(b) Run RSA encryption primitive for integer ciphertext c.

$$c = RSAEP((n, e), m)$$

(c) Return the ciphertext as an octet string C.

$$C = I2OSP(c, k)$$

Decryption

RSAES-OAEP-Decrypt(K, C, L)

Options:	Hash	hash function ($hlen$ denotes the length of in octets the hash function output)
	MGF	mask generation function
Input:	K	RSA private key (k is the length of the octet string representing the modulus n)
	C	ciphertext octet string, with length $k \geq 2 \cdot hLen + 2$
	L	optional label to be associated with the message
Output:	M	the message, an octet string of length $mLen \leq k - 2 \cdot hLen - 2$

Table 2: Variables associated with RSA-OAEP decryption.

1. Length checking:
   (a) The label L is too long if $L > 2^{61} - 1$ octets. (Using SHA-1 hash)
   (b) The length of C must be of length k or there is a decryption error.
   (c) There is also a decryption error if $l < 2 \cdot hLen + 2$.
2. RSA decryption:
   (a) Convert C into its integer representation c.

   $$c = OS2IP(C)$$

   (b) Decrypt c to get the integer representation m using private key K.

   $$m = RSADP(K, c)$$

   (c) Convert to octet string EM of length k octets.

   $$EM = I2OSP(m, k)$$

3. EME-OAEP decoding:

(a) Let $lHash = Hash(L)$, and octet string of length $hlen$.

(b) Separate encoded message EM into its three original parts, a single octet Y, and octet string $maskedSeed$ with length $hLen$, and an octet string $maskedDB$ with length $k - hLen - 1$.

$$EM = Y||maskedSeed||maskedDB$$

(c) $seedMask = MGF(maskedDB, hLen)$

(d) $seed = maskedSeed \oplus seedMask$

(e) $dbMask = MGF(seed, k - hLen - 1)$

(f) $DB = maskedDB \oplus dbMask$

(g) Separate DB into an octet string $lHash'$ with length $hLen$, padded string PS, and M.

$$DB = lHash'||PS||\text{0x01}||M$$

References

[1] M. Bellare and P. Rogaway. Optimal asymmetric encryption. In *Advances in Cryptology—EUROCRYPT'94*, pages 92–111. Springer, 1995.

[2] R. Laboratories. Pkcs #1: Rsa cryptography standard. URL `http://www.emc.com/emc-plus/rsa-labs/standards-initiatives/pkcs-rsa-cryptography-standard.htm`.