A low-cost Alternative for OAEP*

Peter Schartner

University of Klagenfurt
Computer Science · System Security
peter.schartner@aau.at

Technical Report TR-syssec-11-02

Abstract

When encryption messages by use of the RSA crypto-system, securely padding these messages is very important for the security of the encryption scheme. One candidate for secure padding is OAEP (Optimal Asymmetric Encryption Padding). OAEP randomizes the plaintext by use of a random parameter (called randomizer) and hash-functions. Unfortunately, when concerning low-cost security hardware and embedded security micro-controllers, OAEP is quite slow, because hash-functions are implemented in software most commonly. The idea is to speed up OAEP by replacing the time consuming calls of the hash-function by symmetric encryption which is implemented in hardware quite often. The paper will first sketch the design principle of the OAEP alternative and will then give an analysis of the performance gain and the security of the new alternative asymmetric encryption padding (AAEP). We will close with a modified scheme, which guarantees that regardless of the input values (which might be equal), all outputs of AAEP will be different within some period of time (or some maximum number of outputs).

1 Introduction

When using RSA [RSA78] for encrypting messages, choosing a suitable padding mechanism is essential for the security of the encryption scheme. Some padding schemes proposed so far include:

- OAEP (Optimal Asymmetric Encryption Padding) [BR94, BR95] und OAEP+ [Sho01a, Sho01b]
- SAEP (Simplified OAEP) und SAEP+ [Bon01]
- REACT (Rapid Enhanced-security Asymmetric Cryptosystem Transform) [OP01]
- 3-round OAEP [PP04]

All these padding mechanisms employ hash-functions, which are – contrary to the encryption algorithms – most commonly implemented in software. Hence these functions are, compared to the encryption functions which are implemented in hardware, quite slow and this causes a substantial overhead and performance losses. In this paper we will propose a replacement for OAEP-like padding mechanisms, which provides comparable security at negligible costs. We will achieve this by replacing the slow software-based hash-algorithms by symmetric encryption algorithms implemented in hardware.

The remainder of this paper is structured as follows. First we will briefly discuss OAEP and OAEP+. After that we will introduce a replacement for OAEP(+) based on the construction

* Updated Version ((07/2011), submitted at "International Workshop on Security and Dependability for Resource Constrained Embedded Systems – SD4RECS 2011"

principle of so called collision-free number generators (see [SSR07] for details) and provide an analysis concerning efficiency and security. Our padding scheme is especially designed to avoid the massive use of hash-functions, which are most commonly quite slow in embedded or low-cost security microcontrollers. Our padding scheme will come in two flavors: the basic construction is designed for maximum speed, whereas the second variant still provides a considerable speedup compared to OAEP(+), but also provides better security. We will close with the discussion of a special padding mode and some open problems, which are the scope of future research.

2 OAEP and OAEP+

In principle OAEP and OAEP+ (and analogous mechanisms) use a randomizer r to modify an input m: $m' = \text{OAEP}(m) = F(m, r)$, where m' is forwarded to the encryption function. In case of RSA, encrypting the message m now results in $c = \text{RSA}_{(e,n)}(\text{OAEP}(m, r))$, where (e, n) is the public key of the receiver. By applying this modification to m, OAEP and OAEP+ provide the following properties:

1. Probabilistic Encryption: By adding randomness, the deterministic encryption scheme (e.g. RSA) is converted into a probabilistic encryption scheme. In other words: identical plaintexts will now result in different ciphertexts.
2. All-or-nothing Security (AONS): The goal of AONS is to prevent partial decryption of ciphertexts (or other information leakage) by ensuring that an adversary cannot recover any portion of the plaintext m without knowing all bits of m'.

Figure 1 shows OAEP and OAEP+. The inputs (plaintexts) m have a length of $n - k_0 - k_1$ bit respectively. The randomizers r (length k_0 bit are chosen internally at random. The value k_1 is determined by the employed hash-function and m is padded to length $n - k_0$ with zeros in case of OAEP and by use of function H′ in case of OAEP+. The output length for both, OAEP and OAEP+, is n bit.

Function G is a so called mask generating function, which expands a seed input to a masking value of suitable length. Let s be the input of G. The output of G is a bit string of bit-length t obtained by concatenating successive hash values $\text{Hash}(s||i)$ with $0 \leq i < \lceil t/l \rceil$ (and deleting any extra rightmost bits if necessary) with l being the output length of the hash-function Hash.

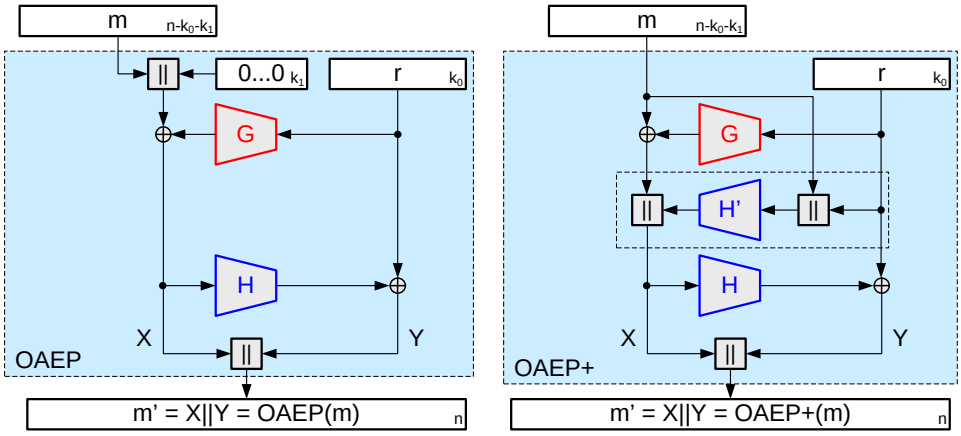

Figure 1: Optimal Asymmetric Encryption Padding: OAEP und OAEP+

In detail, $m' = \text{OAEP}(m)$ employs the following modifications to m:

1. Generate randomizer r: $r \in_R \{0, 1\}^{k_0}$

Table 1: Timings of hash-functions and symmetric encryption

G/H/H'	input length	time HASH
MD5	1024 bit	0,4 ms
	2048 bit	0,7 ms
SHA-1	1024 bit	0,9 ms
	2028 bit	1,6 ms

E	block length	key length	time KL+ENC	time ENC
DES	64 bit	56 bit	6 µs	4 µs
3DES	64 bit	112 bit	10 µs	6 µs

2. $X = (m||00..0) \oplus \mathrm{G}(r)$
3. $Y = \mathrm{H}(X) \oplus r$
4. Return $m' = \mathrm{OAEP}(m) = (X||Y)$

Assume, that the attacker gets hold of some bits of m', say by use of partial decryption. But, due to the all-or-nothing security of OAEP, he needs all bits of X and Y in order to invert OAEP and hence retrieve m. So the only way is to guess the missing bits, which will cause unpredictable errors in the reconstructed values of m, due to the avalanche-effect which is intrinsic to the employed hash-functions. In other words, the attacker has no chance to detect if his guesses were right and hence no way to guess one bit after the other. This finally faces him with a brute force search on all remaining bits, i.e. he has to try all possible combinations of bits he has not gained by partial decryption.

Compared to OAEP, $m' = \mathrm{OAEP} + (m)$ essentially differs by employing an additional function H' (see dashed box in figure **??** containing H'):

1. Generate randomizer r: $r \in_R \{0,1\}^{k_0}$
2. $X = (m \oplus \mathrm{G}(r))||\mathrm{H}'(m||r)$
3. $Y = \mathrm{H}(X) \oplus r$
4. $m' = \mathrm{OAEP} + (m) = (X||Y)$

For our performance analysis it is important to note that most commonly hash-functions like SHA-1 or RIPEMD160 are used to implement the (compressing or expanding) functions G, H and H'.

3 The alternative to OAEP

In [SSR07] we proposed so called collision-free number generators (CFNGs) as a mechanism for generating random but system-wide unique (cryptographic) parameters. Basically these generators disguise a unique (eventually publicly known) parameter by use of a randomizer, what is quite similar to the method, OAEP and OAEP+ use to randomize the message m. Based on the basic construction of collision-free numbers we will propose an alternative to OAEP, which we will call AAEP: alternative asymmetric encryption padding.

In the following, we will implement AAEP by using a symmetric encryption function E in the CBC mode (with ciphertext stealing [MM82]) and a randomly chosen key r. Since the padding process must be invertible without the knowledge of a secret key, we will simply attach the k to the ciphertext, so that the modified message m' results in $m' = \mathrm{AAEP}(m) = X||Y = \mathrm{E}(m,r)||r$ (see figurefig:aaep). Since r is chosen at random for each call of the padding function, we can

omit the initialization value normally used when employing the CBC mode.

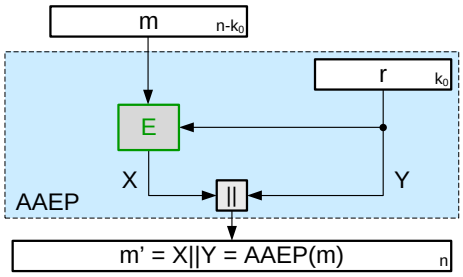

Figure 2: AAEP – a first alternative for OAEP(+)

Our proposal AAEP modifies the message m in the following way::

1. Generate randomizer r: $r \in_R \{0,1\}^{k_0}$
2. $X = \mathrm{E}_r(m)$
3. $Y = r$
4. $m' = \mathrm{AAEP}(m) = (X||Y)$

3.1 Analysis of Performance

Analyzing OAEP(+) results in the fact that performing the mask generating function, G is the bottleneck: in principle G concatenates outputs of a hash-function like MD5 (128 bit outputs) or SHA-1 and RIPEMD160 (160 bit outputs) as long as the mask value has the appropriate length. In case of an 2048 or 4096 bit input, this process results in about 13 to 32 calls of the original hash-function.

In contrast to TPMs, smartcards most commonly only provide software-based implementations of hash-functions like SHA-1 and MD-5. Since the available datasheets on smartcard microprocessors do not contain precise timings for hash-functions, we conducted tests on a Sm@rtCafé Expert 4.0 JavaCard from Giesecke & Devrient. Table 1 shows the time needed to hash 1024 bit and 2048 bit blocks by use of MD5 and SHA-1 respectively.

In order to compare these values with timings of symmetric encryption (DES and 3DES), we additionally used datasheets of the Infineon SLE66-family microprocessors. Since RSA blocks are much larger than DES blocks, we had to apply some mode of operation in order to encrypt the complete message. We decided to use the CBC mode for first tests. Note that due to performance reasons the round keys will only be derived at the first block and cached for later usage. This key derivation is commonly named key load (KL) and results in a computational overhead of about 50 % for both, DES and 3DES!

Table 2 summarizes the computational overhead when using AAEP of the form $m' = \mathrm{E}_r(m)||r$ with the block ciphers DES and 3DES both in CBC mode with ciphertext stealing.

When analyzing OAEP(+) one immediately sees, that performing the mask generating function G is the bottleneck: in principle G concatenates outputs of a hash-function like MD5 (128 bit outputs) or SHA-1 and RIPEMD160 (160 bit outputs) as long as the mask value has the appropriate length. In case of an 2048 or 4096 bit input, this process results in 13 to 32 calls of the original hash-function. Hence table 2 shows the overhead of OAEP considering the 13 to 32 calls of the hash-function by the mask generating function G and additionally 1 call for H and H' each.

Table 2: Performance of AAEP and OAEP+

	RSA modulus	time RSA	E	payload length	blocks	time AAEP	Overhead
AAEP	1024 bit	14.000 µs	DES	960 bit (94 %)	1 + 14	6+14·4 = 66 µs	0,47 %
			3DES	896 bit (88 %)	1 + 13	10+13·6 = 84 µs	0,60 %
	2048 bit	58.000 µs	DES	1984 bit (97 %)	1 + 30	6+30·4 = 130 µs	0,22 %
			3DES	1920 bit (94 %)	1 + 29	10+29·6 = 190 µs	0,33 %

	RSA modulus	time RSA	G/H/H'	payload length	time OAEP+	Overhead
OAEP+	1024 bit	14 ms	MD5	896 bit (88 %)	15·0,4 = 6,0 ms	42,85 %
			SHA-1	864 bit (84 %)	15·0,9 = 13,5 ms	96,42 %
	2048 bit	58 ms	MD5	1920 bit (94 %)	34·0,7 = 23,8 ms	41,03 %
			SHA-1	1888 bit (92 %)	34·1,6 = 54,4 ms	93,79 %

Table 3: Bits needed for partial decryption of m'

E	block length	key length	1^{st} block	other blocks
DES	64 bit	56 bit	64 + 56 = 120 bit	2· 64 + 56 = 184 bit
Skipjack	64 bit	80 bit	64 + 80 = 144 bit	2· 64 + 80 = 208 bit
3DES	64 bit	112 bit	64 + 112 = 176 bit	2· 64 + 112 = 240 bit
AES	128 bit	128 bit	128 + 128 = 256 bit	2·128 + 128 = 384 bit
		192 bit	128 + 192 = 320 bit	2·128 + 192 = 448 bit
		256 bit	128 + 256 = 384 bit	2·128 + 256 = 512 bit

Of course the overheads shown above are only valid for our test scenario. Other smartcards, especially smartcards with hash-functions implemented in hardware, may come up with less overhead for OAEP. Eventually the performance gain of AAEP over OAEP will not be given when using high end security tokens. In this case, using the well known OAEP will be the better choice.

3.2 Analysis of Security

In contrast to OAEP, the basic construction of AAEP does not provide "All-or-nothing-Security". This is quite obvious, because when applying the CBC mode, we need the key and two succeeding cipher blocks in order to partially decrypt. In case of decrypting the first cipher block, there is no preceding block and we need the initialization value (which is not used in our approach).

Table 3 summarizes the bits needed in order to be able to decrypt. Note that this must not be arbitrary bits, but bits representing the key and one (or two succeeding) cipher blocks. Hence, the scheme provides a nice security-speed-tradeoff.

4 Variants of the Scheme

4.1 AONS and high(er) Performance

Our performance analysis above showed that performing the mask generating function G is the bottleneck. On the other hand our security analysis showed that AAEP does not provide AONS. So the idea is to replace the slow subroutine G by symmetric encryption (CBC mode with ciphertext stealing) while keeping the hash-function H. This improves performance without reducing security.

Compared to OAEP we replace 13 to 32 calls (in case of OAEP+ 14 to 33 calls) of the slow hash-function by 16 to 31 calls of the much faster block cipher and keep hash-function H to

Table 4: Performance of AAEP+

	RSA modulus	time RSA	E/H	payload length	blocks	time AAEP+	Overhead
AAEP+	1024 bit	14.000 µs	DES MD5	896 bit (88 %)	1 + 13	6+13·4+ 400 = = 458 µs	3,27 %
			3DES SHA-1	864 bit (84 %)	1 + 13	10+13·6+ 900 = = 988 µs	7,06 %
	2048 bit	58.000 µs	DES MD5	1920 bit (94 %)	1 + 29	6+29·4+ 700 = = 822 µs	1,42 %
			3DES SHA-1	1888 bit (92 %)	1 + 29	10+29·6+1600 = = 1784 µs	3,08 %

protect the randomizer r (see figure 3). Concerning security the attack must know all bits of X before he can "decrypt" Y in order to get hold of the randomizer r. Of course, the attacker is free to guess the missing bits of X (or Y respectively), but this is the same with OAEP(+).

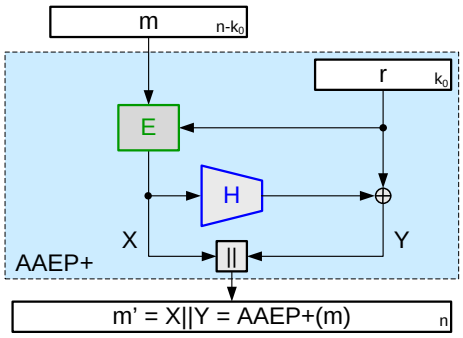

Figure 3: AAEP+ – an alternative for OAEP(+)

If we compare the overhead of AAEP+ (see table 4) to the overhead of OAEP+ (see table 2), we get can reduce the overhead from 42% to 93% in case of OAEP+ to 3% to 7% in case of AAEP+ without reducing security.

In detail, $m' = \text{AAEP} + (m)$ employs the following modifications to m:

1. Generate randomizer r: $r \in_R \{0,1\}^{k_0}$
2. $X = \text{E}_r(m)$
3. $Y = \text{H}(X) \oplus r$
4. Return $m' = \text{AAEP} + (m) = (X||Y)$

4.2 Other Modes of Operation

Revisiting the security drawbacks of AAEP, other modes of operation like the propagating cipher block chaining mode (PCBC) might be quite useful. As figure 4 shows, the PCBC mode only uses on additional XOR per block, which is negligible compared to the time needed to encrypt a block. But now, in order to decrypt a specific cipher block, we need all preceding cipher blocks. Of course this still does not provide AONS, but this variant is more secure than AAEP and much faster than AAEP+.

4.3 Unique outputs m'

It is worth mentioning that with little modifications, the proposed padding scheme can guarantee that regardless of the input values m all outputs m' will be different within some period of time

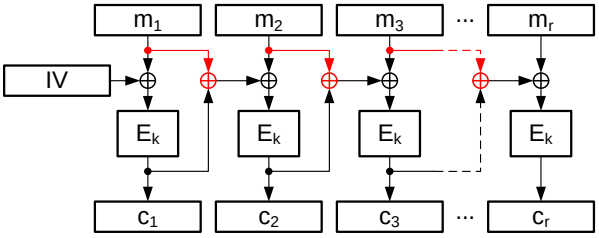

Figure 4: Propagating Cipher Block Chaining Mode (PCBC)

(or some maximum number of outputs). To achieve this, simply append a counter *cnt* of appropriate length to m: $m' = \text{AAEP}(m) = E(m||cnt, r)||r = c||r$. In order to guarantee this uniqueness, we have to fulfill two requirements:

1. The counter *cnt* has to be incremented at each call of AAEP.
2. The counter must not have an overflow within it's time of usage.

Note that the second requirement does not imply that duplicate inputs result in duplicate outputs in case of an overflow of the counter. This only accrues in the case, that the same randomizer is used for the duplicate inputs. If different randomizers are used for equal inputs, the resulting outputs will stell be different. So the only thing with overflows of the counter is, that we can no longer guarantee the uniqueness of the outputs.

For a detailed discussion and extensions (e.g. uniqueness for different instances of AAEP) we refer the reader to [SSR07]. Proofing this postulated uniqueness is quite easy. Concerning two outputs $m'_1 = \text{AAEP}(m_1) = \text{E}(m_1||cnt_1, r_1)||r_1$ and $m'_2 = \text{AAEP}(m_2) = \text{E}(m_2||cnt_2, r_2)||r_2$, we will consider the two halves E() and r separately:

1. $r_1 \neq r_2$: this directly means that $m'_1 \neq m'_2$ and we are done.
2. $r_1 = r_2 = r$: It is a fact that symmetric encryption functions are bijective for an arbitrary but fixed key. Hence the outputs m'_1 and m'_2 will differ in at least one bit, because incrementing the counter at each call guarantees that $cnt_1 \neq cnt_2$.

Note that we are aware, that this property is not necessary for standard RSA encryption, but it might be useful in other, not yet identified, application scenarios.

5 Conclusion

In this paper we presented an efficient and secure alternative for OAEP and analogous padding mechanisms for asymmetric encryption schemes especially designed for security tokens equipped with hardware-accelerated encryption functions, but software-based hash-functions. We first described the basic construction and analyzed the performance (gain) and security (loss). Based on this analysis we proposed several variants which either further improved performance or improved security. The last variant proposed a scheme which provides guaranteed unique cipher texts during the live time of the system. Current research includes identifying application scenarios for the last variant and further improvements of the core concept.

References

[Bon01] D. Boneh. Simplified OAEP for the RSA and Rabin Functions. In *Crypto '01*, volume 2139 of *LNCS*, pages 275–291. Springer-Verlag, Berlin, 2001.

[BR94] M. Bellare and P. Rogaway. Optimal Asymmetric Encryption. In *Proceedings of EROCRYPT 94*, pages 92–111, 1994.

[BR95] M. Bellare and P. Rogaway. Optimal Asymmetric Encryption – How to Encrypt with RSA, 1995.

[MM82] C.G. Meyer and S.M. Matyas. *Cryptography: A New Dimension in Computer Data Security*. John Wiley & Sons Inc, 1982.

[OP01] T. Okamoto and D. Pointcheval. REACT: Rapid Enhanced-security Asymmetric Cryptosystem Transform. In *CT RSA '01*, number 2020 in LNCS, pages 159–175. Springer-Verlag, Berlin, 2001.

[PP04] D.H. Phan and D. Pointcheval. OAEP 3-round: A generic and secure asymmetric encryption padding. In P.J. Lee, editor, *Advances in Cryptology – ASIACRYPT'04*, volume 3329 of *Lecture Notes in Computer Science*, pages 63–77. Springer Verlag, 2004.

[RSA78] R.L. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21:120–126, February 1978.

[Sho01a] V. Shoup. OAEP Reconsidered. In *Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '01, pages 239–259, London, UK, 2001. Springer-Verlag.

[Sho01b] V. Shoup. OAEP Reconsidered – Extended Version, 2001.

[SSR07] M. Schaffer, P. Schartner, and S. Rass. Universally Unique Identifiers: How To Ensure Uniqueness While Protecting The Issuer's Privacy. In S. Aissi and H.R. Arabnia, editors, *Security and Management*, pages 198–204. CSREA Press, 2007.