Modified Parameter Attacks: Practical Attacks Against CCA2 Secure Cryptosystems, and Countermeasures

Nick Howgrave-Graham, Joe Silverman, Ari Singer, William Whyte

NTRU Cryptosystems

Abstract. We introduce the concept of Modified Parameter Attacks, a natural extension of the idea of Adapative Chosen Ciphertext Attacks (CCA2) under which some CCA2 secure systems can be shown to be insecure. These insecurities can be addressed at the application level, but can also be addressed when cryptographic schemes are being designed. We survey some existing CCA2 secure systems which are vulnerable to this attack and suggest practical countermeasures.

1 Introduction

It is of great interest, when studying cryptosystems, to make statements about their security under certain attack models. The contribution of this paper is to suggest an attack model for public key encryption schemes, stronger than those typically considered at present, within which the security properties of the scheme can be evaluated.

We start by identifying existing concepts used in the study of public key encryption schemes. First, we review security properties of *indistinguishability* and *non-malleability*. Each of these properties can be looked on as the ability to resist a particular type of attack.

- **Indistinguishability**: An attacker submits two plaintexts, x_0 and x_1, to an encryption oracle, which encrypts one of them and outputs the ciphertext y. If the attacker cannot identify, with non-negligible advantage, the plaintext which was used to produce y, the system is said to have the property of *indistinguishability*.
- **Non-Malleability**: An attacker is given a plaintext x and the corresponding ciphertext y. If the attacker can alter y to obtain y', where y' decrypts to a message x' that is related to x in some simple way, then the system is said to be *malleable*. If an attacker cannot perform such an alteration, the system is said to have the property of *non-malleability*.

In general, three significant types of attack are commonly identified [3, 16]:

- **Chosen Plaintext Attack (CPA)**: The attacker may generate as many ciphertexts as she likes, but has no access to a decryption oracle.
- **Non-Adaptive Chosen Ciphertext Attack (CCA1)**: The attacker has access to a decryption oracle before seeing the target ciphertext y, but no access to the oracle after seeing y. This attack is also known as a "lunchtime attack".
- **Adaptive Chosen Ciphertext Attack (CCA2)**: The attacker has access to a decryption oracle both before and after seeing the target ciphertext y. Before seeing y, the attacker may query the decryption oracle with any ciphertext or purported ciphertext. After seeing y, the attacker may query the decryption oracle with any ciphertext or purported ciphertext, except y itself.

Until now, it has been generally believed that the strongest possible security that may be demanded of a cryptosystem is security against attacks of the form CCA2. In this paper, we propose a new, stronger form of attack which we call *Modified Parameter Attacks*. These attacks can break certain systems which are known to be CCA2 secure. We also demonstrate simple countermeasures.

2 Modified Parameter Attacks

2.1 Motivation

In cryptographic research, it has been the custom to look on security against adaptive chosen ciphertext attacks as the property that encryption schemes should aim to have. Many schemes [2, 6, 14] have been proposed that can be proved to have this property. However, each of these schemes is typically parameterizable, and the proofs of security assume that a given key is used only with one parameter set.

This paper looks at vulnerabilities which can be exploited when a given key is used with more than one set of parameters. As an example of how this might happen, consider cryptographic standards such as [5, 10, 11, 15]. Here, the convention is that a public encryption key is associated only with a specific algorithm, not with a specific scheme. For example, in PKIX [11], an RSA public key in a certificate is accompanied by an identifier stating that it is an RSA key. Additional identifiers can restrict its use to (for example) signing only or encryption only, but there is no standard way to restrict its use to a specific encryption scheme. Instead,

the sender of an encrypted message decides the scheme and the scheme parameters to use on encryption. These parameters are sent, typically unprotected and in the clear, along with the ciphertext information, and the recipient cannot necessarily be certain that the parameters used to decrypt a message are the same as the ones that were used to encrypt it. It is certainly an interesting area of study to investigate what might go wrong if those parameters can be altered.

This area has not been closely examined at the cryptographic primitive level in the past. Where the issue has been mentioned [18], it has in general been covered by a statement that a key should have a set of parameters associated with it at key generation time, and that only these parameters should be used. It's not clear how this could be enforced in practice; it would be better to ensure that there are no risks to the system even if this principle isn't followed.

At a higher level, the idea of studying interactions between different security procedures is well established. The concept of chosen-protocol attacks is introduced in [12]; studies of secure protocols such as SSL [21] have warned about version rollback attacks, in which an attacker can persuade victims to use a different, weaker procedure than the one they would use by choice. The paper [12] demonstrates that (in its own words) "a protocol may be quite secure alone, but may lose its security when another protocol exists that can be carried out with the same key pair", and that "A key should typically have only a small number of closely related uses. There is sometimes a temptation to reuse keys for related applications. This temptation should be avoided wherever possible." The distinction between this paper and theirs is that they concentrate, in general, on the protocol level, and we concentrate on scheme-level issues.

Nearer the cryptographic primitive level, Haber and Pinkas [9] have studied interactions between different secure systems, in particular the question of whether it is safe to use a given key for both a plaintext-aware encryption scheme and an existentially unforgeable signature scheme. Interestingly, their result is that a plaintext aware encryption scheme cannot be used to break any other cryptographic scheme. This paper presents counter-examples to their result; we suggest that the resolution lies in considering exactly what plaintext-awareness means in the context of our attack model. Shoup [18] has observed an attack on the DHAES encryption scheme [1] similar in spirit to the attacks presented here. In DHAES, the sender of an encrypted message may attach a label. If the length of this label is a fixed parameter, the scheme is secure; if the length of the label can vary, the scheme is broken unless the length is explicitly com-

mitted to. In this paper we show similar attacks on other CCA2-secure cryptosystems, and demonstrate that committing to the length of the various building blocks of the system does not prevent all attacks.

In summary, our motivation is this: if a key is to be used with a "secure" system, it should not be possible to make the use of that key break any other "secure" system. We show in this paper that a system that is CCA2-secure may not be secure in the sense just defined, and propose a design philosophy to obtain the desired level of security.

2.2 Model

The concept of the Modified Parameter Attack is simple. Consider a public key cryptosystem $\mathcal{C}$ consisting of the three algorithms $\{\mathcal{K}, \mathcal{E}, \mathcal{D}\}$ and parameterized by the parameter set $\{\mathcal{P}\}$, such that:

- $\mathcal{K}$ takes as input $\mathcal{P}$ and produces as output the keypair k_{pub}, k_{priv}.
- $\mathcal{E}$ takes as input $\mathcal{P}$, k_{pub} and the message x, and produces as output the ciphertext y.
- $\mathcal{D}$ takes as input $\mathcal{P}$, k_{priv}, possibly k_{pub}, and the ciphertext y, and produces either the decrypted message x' or an error message indicating that the decryption has failed.

In a modified parameter attack, the attacker has access to a decryption oracle, and may query it with any y and any $\mathcal{P}$ of their choosing. In other words, whereas in an adaptive chosen ciphertext attack the attacker can query the decryption oracle with any $y' \neq y$ of their choosing, in a modified parameter attack the attacker can query the decryption oracle with any $(y', \mathcal{P}')$ of their choosing, so long as $(y', \mathcal{P}')$ is not exactly equal to $(y, \mathcal{P})$. Another way of looking at this is to say the attacker has access to a large number of decryption oracles $\mathcal{D}_i$, all using the same key but each parameterized by a different parameter set $\mathcal{P}_i$, and that only the oracle whose parameters correspond to those used to encrypt y will refuse to decrypt y.

So, in this model, what counts as a parameter? To help define this, we consider differences between the original definition of RSA-OAEP [2] and two deployed implementations of the scheme [15, 17]. Bellare and Rogaway's original OAEP construction [2], hereafter referred to as "OAEP-BR", is designed for use on l-bit trapdoor permutations f. OAEP-BR is parameterized by k_0 and k_1, which are bitlengths, and by a choice of two hash functions: F, which maps an input string of length k_0 to an output string of length $l - k_0$, and G, which maps an input string of length

$l - k_0$ to an output string of length k_0. The quantities k_0 and k_1 satisfy $k_0 + k_1 < l$; the maximum length of message that can be encrypted is $l - (k_0 + k_1)$.

To encrypt, the sender does the following:

1. Generates r, a random bit string of length k_0.
2. Calculates $x' = F(r) \oplus (x||0^{k_1})$.
3. Calculates $r' = G(x') \oplus r$.
4. Forms the masked message block b as $x'||r'$ and encrypts this with the trapdoor permutation f to get the ciphertext $y = f(b)$.

To decrypt, the recipient does the following:

1. Uses the inverse trapdoor permutation f^{-1} to recover the candidate masked message block $B = f^{-1}(y)$.
2. Splits B into the two blocks X' and R', given by the first $l - k_0$ and the last k_0 bits respectively.
3. Calculates $R = G(X') \oplus R'$.
4. Calculates $X = F(R) \oplus X'$.
5. Checks to see if the last k_1 bits of X are equal to 0^{k_1}. If this is the case, the recipient outputs the first $l - (k_0 + k_1)$ bits of X as the message x. Otherwise, the recipient outputs "`error`".

In this context, k_0, k_1, F and G are obviously parameters. However, inspection of OAEP as specified in PKCS#1 versions 2 and higher [15] and in SET [17] leads to a slightly broader definition. In OAEP-BR, the message block before masking looks like

$$[x||\texttt{checkData}||r],$$

where $\texttt{checkData} = 0^k$. In OAEP-PKCS1, the message block looks like

$$[r||\texttt{checkData}||x],$$

where `checkData` is the hash of some information. And in OAEP-SET, the message block looks like

$$[\texttt{checkData}||x||r],$$

with $\texttt{checkData} = 0^k$ again. Both OAEP-PKCS1 and OAEP-SET have the same security properties as the ideal OAEP-BR (modulo some small vulnerabilities discovered by Manger [13]), but the specific implementations are very different.

So we can imagine an attacker who can alter k_0 and k_1, but additionally we can imagine that the attacker has the power to state which of the bits in the unmasked decrypted message belong to x, which to r, and which to `checkData`. Essentially, we can imagine the attacker imposing any interpretation of the bits they like, so long as it doesn't break the IND-CCA2 properites of the system. This means, in particular:

- No parameter indicating a bitlength may be lowered below a level that makes integrity checks effectively meaningless.
- All hash functions used to instantiate random oracles must behave, as much as possible, like random oracles. In other words, an attacker can't persuade someone to encrypt or decrypt with a hash function whose output is (for example) the reverse of the output of SHA-1, or the output of SHA-1 plus 1.

The principle is that both the sender and the recipient are making a good-faith effort to implement a secure cryptosystem and the attacker can make them use any other cryptosystem that, considered in isolation, is at least as secure.

We now survey two CCA2 secure cryptosystems which can be broken by attacks of this type. Their common theme is that, although they perform an integrity check, this check guarantees only that the recipient has decrypted the same *message block* as the sender encrypted, not that the recipient has recovered exactly the same *message*.

3 Modified Parameter Attacks on RSA-OAEP

3.1 Review

OAEP was introduced by Bellare and Rogaway [2], who showed it to be CCA1 secure. Although Shoup [19] showed that OAEP is not CCA2 secure in the general case, it was demonstrated by Fujisaki *et al.* [8] that when OAEP is combined with the RSA function it is, in fact, CCA2 secure. In this section we describe a modified parameter attack that breaks indistinguishability for some forms of OAEP.

OAEP-BR was described in the previous section. Its important characteristic for purposes of this discussion is that the message sender constructs a block of the form

$$[x||\texttt{checkData}||r],$$

where $\texttt{checkData} = 0^{k_1}$, and that the process of hashing and masking uses r separately from $(x||\texttt{checkData})$ but treats $(x||\texttt{checkData})$ as a single unit.

3.2 Modified Parameter Attacks

Now consider an attacker who wants to distinguish between two messages. If they change F or G, decryption will almost certainly fail under the random oracle assumption. Likewise, they cannot change k_0, because this is the length of input to F, and changing the input to F will cause decryption to fail under the random oracle assumption. The attacker instead does the following:

1. Selects two messages x_0 and x_1, such that x_0 ends with a zero byte and x_1 does not.
2. Submits these two messages to the encryption oracle and receives back the ciphertext y encrypted with the parameters $\{k_0,\ k_1,\ F,\ G\}$.
3. Submits the ciphertext y to the decryption oracle with the parameters $\{k_0,\ k_1 + 8,\ F,\ G\}$.
4. If the decryption oracle outputs "`error`", the attacker identifies x_1 as the encrypted message. If the decryption oracle does not output an error, the attacker identifies x_0 as the encrypted message.

Here, the weakness arises because the integrity check that is performed ensures only that the recipient has retrived $(x||0^{k_1})$, not x itself.

3.3 Effect on Deployed Systems, and Countermeasures

Although this attack is effective against OAEP-BR, it does not necessarily apply to deployed systems, and there are simple countermeasures. We now survey existing variants of OAEP for their security against this attack.

A simple countermeasure would be to have F take as input a string of length $(k_0 + k_1)$, specifically $(0^{k_1}||r)$, and to have G take as input a string of length $l - (k_0 + k_1)$. This way, any alteration to k_0 or k_1 will alter the input to F; under the random oracle assumption, this will make successful decryption negligibly unlikely. This countermeasure is effective, but not particularly generic: the countermeasures below are preferred.

OAEP-PKCS1 [15] uses a slightly stronger countermeasure. In this version of the scheme, we require $k_0 = k_1$ (they are both defined as the output length of an appropriate hash function), and instead of performing the integrity check by looking for a string of zeroes in a specified location, we look for the hash of a known, public string, called the "encoding parameters". These encoding parameters, however, are optional and are not linked directly to the message, and the approach seems like an engineering countermeasure to the attack rather than a more fundamental cryptographic approach.

The form of RSA-OAEP specified in the SET documents [17] is not characterized by variable parameters. The specification requires 1024-bit keys, SHA-1, and an eight-byte `checkData` equal to `03 00 ... 00`. We make two comments about this. First, if in a specific implementation the length of `checkData` was variable, rather than fixed, an equivalent attack to the one described against OAEP-BR would work against OAEP-SET. It is easy to imagine an engineer allowing this flexibility at some API level, anticipating later versions of the protocol with longer check strings. Additionally, there is a rather more interesting attack which uses OAEP-BR to break OAEP-SET, and vice versa. Recall that in OAEP-BR the message block, before masking, looks like $[x||\texttt{checkData}||r]$, while in OAEP-SET it looks like $[\texttt{checkData}||x||r]$. The attack does the following:

1. Selects two messages x_0 and x_1, such that x_0 begins with `03 00 ... 00` and x_1 does not.
2. Submits these two messages to an encryption oracle using OAEP-BR and receives back the ciphertext y.
3. Submits the ciphertext y to a decryption oracle using OAEP-SET.
4. If the decryption oracle outputs "`error`", the attacker identifies x_1 as the encrypted message. If the decryption oracle does not output an error, the attacker identifies x_0 as the encrypted message.

We note that this attack does not apply to OAEP-PKCS1. This is because OAEP-PKCS1 places r at the beginning, not the end, of the message block.

In the SAEP+ construction due to Boneh [4], the sender does not use the string $(x||0^{k_1})$ in step 2 of the encryption. Instead, using the hash functions G, H, the sender constructs:

$$\begin{aligned} s &= H(r) \oplus (x||G(x||r)) \\ w &= s||r, \end{aligned}$$

and encrypts w. On decryption, the recipient recovers and checks $G(x||r)$. It appears, although we do not prove it here, that this prevents all modified parameter attacks: the use of $H(r)$ guarantees that r has been recovered correctly, and the use of $G(x||r)$, given that r has been recovered, guarantees that x has been recovered correctly.

The OAEP+ construction due to Shoup [19], which inspired SAEP+, takes a similar approach. In OAEP+, using an additional hash function H', the sender constructs:

$$\begin{aligned} s &= (G(r) \oplus x)||H'(r||x), \\ t &= H(s) \oplus r, \\ w &= s||t, \end{aligned}$$

and encrypts w. This construction also appears immune to parameter altering attacks: $G(r)$ guarantees that r has been recovered, and $H'(r||x)$ guarantees that x has been recovered. Our recommendation would be that any future standards that adopt an OAEP-based system seriously consider the use of OAEP+ or SAEP+.

3.4 MPA Attacks on RSA-OAEP: Conclusions

In this section, we have surveyed different implementations of RSA-OAEP, and demonstrated that some are more vulnerable than others to parameter altering attacks. In particular, we have shown an interaction between OAEP-SET and OAEP-BR which allows a decryption oracle for one to be used to break the other. In comparing those constructions which resist MPA to those which fail against it, we draw the following design conclusion: **Integrity checks should guarantee that the decrypter has recovered the correct message, not the correct message block**. This principle will be further endorsed by our study of another CCA2 secure scheme, that due to Fujisaki and Okamoto [6, 7].

4 Modified Parameter Attacks on the Fujisaki-Okamoto Scheme

4.1 Review

The Fujisaki-Okamoto scheme [6, 7] provably converts a probabilistic public-key encryption scheme that is secure in the sense of IND-CPA to one that is secure in the sense of IND-CCA2 (and therefore NM-CCA2). The construction is simple. Let the original probabilistic algorithm be represented by

$$\mathcal{E}_{pk}(x,\ r),$$

where x is the message of length l to be encrypted, and r is a set of coin tosses of appropriate length. Then the encryption under the FO scheme is

$$\bar{\mathcal{E}}_{pk}^{H,k}(x,\ r) = \mathcal{E}_{pk}(x||r,\ H(x||r)),$$

where r is now a set of coin tosses of length k, and x is restricted to being of length $l-k$. On decrypting a ciphertext y, the recipient recovers $(x||r)$ and checks that $\mathcal{E}_{pk}(x||r,\ H(x||r)) = y$; if the check succeeds, the recipient outputs x, and if it fails they output the error string.

4.2 MPA Attacks on Fujisaki-Okamoto

We demonstrate two attacks, one of which depends on altering the parameter k, the other of which is more sophisticated.

First, consider an attacker who can alter the parameter k. As in the OAEP case above, there is a simple attack which breaks indistinguishability, as follows. The attacker:

1. Selects two messages x_0 and x_1, which differ in at least one byte other than the last byte.
2. Submits these two messages to the encryption oracle and receives back the ciphertext y encrypted with the parameters $\{k, H\}$.
3. Submits the ciphertext y to the decryption oracle with the parameters $\{k + 8, H\}$.
4. The decryption will succeed, and will output all but the last byte of the x_i which was encrypted.

The attacker can thus distinguish between the two messages.

Second, consider the possibility that there might be two different implementations of FO in common use: one which encrypts $(x||r,\ H(x||r))$ and one which encrypts $(r||x,\ H(r||x))$. This is exactly analogous to the difference between OAEP-BR and OAEP-SET exploited in the previous section, and enables a similar attack. The attacker:

1. Selects two messages x_0 and x_1, which differ in at least one bit in their first $l - 2k$ bits.
2. Submits these two messages to the encryption oracle to be encrypted in the mode $(r||x)$, and receives the ciphertext y.
3. Submits the ciphertext y to the decryption oracle, claiming that it was encrypted in the mode $(x||r)$.
4. The decryption will succeed and return a "plaintext" x'. The attacker takes x', discards the first k bits, and compares the remaining bits to the first $l - 2k$ bits of x_0, x_1.

4.3 Countermeasures and Suggestions

It would clearly be useful to redesign the FO construction so that it requires the recipient to distinguish between the message x and the randomness r, and to confirm that the message they received is the same as the message sent. We discuss three countermeasures: first, one which turns out to be unsatisfactory, and then two that appear to solve the problem.

First, consider the following construction:

$$\bar{\mathcal{E}}'^{\,H,k}_{pk}(x,r) = \mathcal{E}_{pk}(x||r, H(len(x)||x||len(r)||r)),$$

where $len(x)$ and $len(r)$ denote the length in bits of x and r respectively (these will, of course, be $l-k$ and k). This construction defeats the first attack outlined above, which relied on an attacker being able to alter k at will; now, any alteration to k will alter $len(x)$ and prevent the check from succeeding. However, it fails when there are two different implementations, one (say $\mathcal{E}'_1$) which encrypts using $H(len(x)||x||len(r)||r))$ and one (say $\mathcal{E}'_2$) which encrypts using $H(len(r)||r||len(x)||x))$. In this case, the attacker proceeds as follows:

1. Selects two messages x_0 and x_1.
2. Submits these two messages to the encryption oracle to be encrypted by $\mathcal{E}'_1$ with the length of r set to k.
3. Submits the ciphertext y to the decryption oracle, claiming that it was encrypted by $\mathcal{E}'_2$ with the length of the randomness set to $l-k$.
4. The decryption will succeed and return a "plaintext" x' which is exactly the r used in encryption. The attacker then encrypts x_0 and x_1 using this r, and selects the x_i which recovered y.

It is clear that even a construction which ensures that the recipient has a message of the correct length is vulnerable to attack. What is necessary is that the recipient unambiguously identifies what part of the decrypted block is the data, and what part is the randomness

Second, consider a version of FO that takes the following form:

$$\bar{\mathcal{E}}''^{\,H,k}_{pk}(x,r) = \mathcal{E}_{pk}(x||r, H(G(\text{“}\texttt{message}\text{”}||x)||r)).$$

Here, G is an additional hash function, and "`message`" is the ASCII string "message". The use of G on x but not r guarantees that the recipient has distinguished between x and r; the use of the "message" string prevents an attack that switches the roles of x and r (we assume that, although the attacker is at liberty to define infinitely many CCA2-secure systems, she can't alter the English language). So long as k is sufficiently long, and the output of G is sufficiently long, this construction appears both CCA2 secure and immune to modified parameter attacks. Other constructions with similar properties are possible, such as $\bar{\mathcal{E}}^{H,k}_{pk}(x,r) = \mathcal{E}_{pk}(x \oplus G(\text{“}\texttt{randomness}\text{”}\text{”}||r)||r, H(x||r))$.

Finally, consider the following version of FO:

$$\bar{\mathcal{E}}'''^{\,H,k}_{pk}(x,r) = \mathcal{E}_{pk}(x||r, H(ID||x||r)).$$

Here, ID is an identifier for the parameter set. Its inclusion amounts to having the sender and recipient attest to the parameter set that they are using, such that decryption will fail unless both parties are using the same parameters.

This proposal represents a third way between the two schemes above. So long as all parameter sets include the ID first, not last, in the input to H, this scheme appears secure. However, if some schemes are in use that input the ID last, some attacks similar to those described earlier will clearly be possible. We recommend that all new public key schemes adopt this approach, and **ensure that the ID is always the first data input into H.**

5 Conclusions

We have introduced the concept of Modified Parameter Attacks and shown how they may be used to break CCA2 secure encryption schemes.

This paper leaves open several areas of possible future investigation. First, and most fundamentally, the question is raised: what is the strongest possible attack model in which we can make statements about the provable security of encryption schemes? If such a model could be formalised, it would naturally replace CCA2 as the "right" notion of security. Part of this research would be to come up with a definition of plaintext awareness reconciling our results and those of [9].

Second, we have suggested countermeasures but not proved their effectiveness. This is an obvious area for future research. One potential countermeasure is to take a specification of the parameters as one of the inputs to the system. In this paper, however, we have focussed on proving knowledge of the exact message sent. This is because of the difficulty of unambiguously and completely specifying parameters, when our definition of "parameters" includes the specification of how the individual bits in the message block are to be interpreted. Countermeasures built on committing to the message itself seem both more satisfying and easier to specify correctly.

Finally, we note that both OAEP and Fujisaki-Okamoto, as originally specified, imply that the length of a message is fixed. Where we have talked about a guarantee that the recipient has recovered the message, not the message block, it has been this fixed-length message that we've been talking about. In practice, however, the actual message to be transmitted is typically shorter than this and must be padded to the appropriate length using padding that can be unambiguously removed. Recent

work by Vaudenay in the symmetric context [20] has demonstrated that there may be subtle issues concerned with performing this padding and unpadding. It would be interesting to extend the investigations of this paper to encompass the use of unambiguous padding methods.

Constructing a secure encryption scheme is a difficult problem, and made all the more difficult by the fact that the recipient of a message cannot rely on being 100% certain of the parameters used to encrypt that message. In this paper, we have posed an interesting challenge to developers of secure encryption schemes: the scheme should have security properties which hold not only against ciphertexts generated with that scheme, but against ciphertexts generated with *any* encryption scheme.

References

1. M. Abdalla, M. Belalre, P. Rogaway, *DHIES: An encryption scheme based on the Diffie-Hellman Problem*, available from http://www.cs.ucsd.edu/users/mihir/crypto-research-papers.html. Extended abstract, entitled "The Oracle Diffie-Hellman Assumptions and an Analysis of DHIES", was in Topics in Cryptology - CT-RSA 01, D. Naccache, Ed., Lecture Notes in Computer Science 2020, Springer-Verlag, pp. 143-158, 2001.
2. M. Bellare, P. Rogaway, *Optimal asymmetric encryption*, in Proc. Eurocrypt 1994, A. De Santis, Ed., Lecture Notes in Computer Science 950, Springer-Verlag, pp. 92-111, 1994.
3. M. Bellare, A. Desai, D. Pointcheval, P. Rogaway, *Relations between notions of security for public-key encryption schemes*, in Proc. Crypto 1998, H. Krawwczyk, Ed., Lecture Notes in Computer Science 1462, Springer-Verlag, pp. 26-45, 1998.
4. D. Boneh, *Simplified OAEP for the RSA and Rabin functions*, in Proc. Crypto 2001, J. Killian, Ed., Lecture Notes in Computer Science 2139, Springer-Verlag, pp. 275-291, 2001.
5. D. Eastlake, J. Reagle (editors), *XML Encryption Syntax and Processing, W3C Candidate Recommendation*, W3C, 2002. Draft available from http://www.w3.org/Encryption/2001/Drafts/xmlenc-core/.
6. E. Fujisaki, T. Okamoto, *How to enhance the security of public-key encryption at minimum cost*, in Proc. PKC '99, H. Imai, Y. Zheng, Eds., Lecture Notes in Computer Science 1560, Springer-Verlag, pp. 53-68, 1999.
7. E. Fujisaki, T. Okamoto, *How to enhance the security of public-key encryption at minimum cost*, IEICE Trans. Fundamentals, Vol E83-A, Number 1 pp 24-32, 2000.
8. E. Fujisaki, T. Okamoto, D. Pointcheval, J. Stern, *RSA-OAEP is secure under the RSA assumption"*, in Proc. Crypto 2001, J. Killian, Ed., Lecture Notes in Computer Science 2139, Springer-Verlag, pp. 260-273, 2001.
9. S. Haber, B. Pinkas, *Combining Public Key Cryptosystems*, in Proc. ACM Computer and Security Conference, November 2001, ACM Press, pp. 215-224, 2001
10. R. Housley, RFC 2630: Cryptographic Message Syntax, Internet Activities Board, 1999. Available at http://www.rfc-editor.org/. [CHECK ME!]

11. R. Housley, W. Ford, W. Polk, D. Solo, RFC 2459: Internet X.509 Public Key Infrastructure Certificate and CRL Profile, Internet Activities Board,1999. Available at http://www.rfc-editor.org/.
12. J. Kelsey, B. Schneier, D. Wagner, *Protocol Interactions and the Chosen Protocol Attack*, in Proc. Security Protocols - 5th International Workshop, B. Christianson, B. Crispo, T. M. A. Lomas, M. Roe, Eds., Lecture Notes in Computer Science 1361, Springer-Verlag, pp. 91-104, 1997
13. J. Manger, *A chosen ciphertext attack on RSA Optimal Asymmetric Encryption Padding (OAEP) as standardized in PKCS#1*, in Proc. Crypto 2001, J. Killian, Ed., Lecture Notes in Computer Science 2139, Springer-Verlag, pp. 230-238, 2001.
14. T. Okamoto, D. Pointcheval, *REACT: Rapid Enhanced-Security Asymmetric Cryptosystem Transform*, in Proc. CT-RSA 2001, D. Naccache, Ed., Lecture Notes in Computer Science 2020, Springer-Verlag, pp. 159-175, 2001.
15. Public Key Cryptography Standards (PKCS), *PKCS#1 v2.1: RSA Cryptography Standard*, Draft 2, 2001. Available at http://www.rsasecurity.com/rsalabs/pkcs/.
16. C. Rackoff, D. Simon, *Noninteractive zero-knowledge proof of knowledge and chosen ciphertext attack*, in Proc. Crypto 1991, J. Feigenbaum, Ed., Lecture Notes in Computer Science 576, Springer-Verlag, pp. 433-444, 1992.
17. SET Secure Electronic Transaction LLC, *SET Secure Electronic Transaction Specification Book 3: Formal Protocol Definition*, 1997. Available at http://www.setco.org/download.html
18. V. Shoup, *A Proposal for an ISO Standard for Public Key Encryption (version 2.1)*, preprint, available from http://www.shoup.net or http://eprint.iacr.org/2001/112.pdf.
19. V. Shoup, *OAEP reconsidered*, in Proc. Crypto 2001, J. Killian, Ed., Lecture Notes in Computer Science 2139, Springer-Verlag, pp. 239-259, 2001.
20. S. Vaudenay, *Security Flaws Induced by CBC Padding - Applications to SSL, IPSec, WTLS, ...* in Proc. Eurocrypt 2002, L. Knudsen, Ed., Lecture Notes in Computer Science 2332, Springer-Verlag, pp. 534-546, 2002.
21. D. Wagner, B. Schneier, *Analysis of the SSL 3.0 protocol*, in Proc. Second USENIX Workshop on Electronic Commerce, USENIX Press pp. 29–40, 1996. Available at http://www.counterpane.com/ssl.html