Some Variations on RSA Signatures & their Security

*Wiebren de Jonge & David Chaum**

Subfaculteit Wiskunde en Informatica
Vrije Universiteit Amsterdam the Netherlands

*Centre for Mathematics and Computer Science
Kruislaan 413 1098 SJ Amsterdam the Netherlands

Abstract: The homomorphic structure of RSA signatures can impair security. Variations on a generalization of RSA signatures are considered with the aim of obviating such vulnerabilities. Of these variations, which involve a function of the message in the exponent, several are shown to have potential weaknesses similar to those of RSA.

No attacks have been found for one of the variations. Its security does not depend on redundancy present in or artificially combined with messages. The same holds for a well-known use of RSA that relies on a one-way compression function. A comparison between the schemes is given.

Introduction

The RSA signature function is a homomorphism with respect to multiplication. This multiplicative property can be useful, since it allows various powerful techniques, such as blind signatures [Chaum85]. However, the property also means a potential weakness for RSA signatures used in other applications, since it prevents some redundancy schemes from securing RSA signatures against attacks based on the property [DeJCh85].

One solution would be to find redundancy schemes that are able to resist such attacks. Another solution, which has been well-known in the folklore and offers some advantages, is to make use of a one-way compression function. A third approach is to try to find signature schemes that do not have such unwished for properties. This last approach is the main subject of this paper.

The RSA digital signature scheme is extensively reviewed in section 2. Particularly, that section treats in some detail the aspects relevant to this paper: redundancy, chosen signature attacks, multiplicative attacks, chaining and compression.

In section 3 a generalization of RSA signatures is introduced. This generalization

encompasses, besides RSA, various other digital signature schemes. The properties of one of these schemes are analyzed in section 4 and compared with RSA in section 5.

1. RSA signatures

In the signature scheme of Rivest, Shamir and Adleman [RSA78], now widely known as RSA, each user chooses a large number n, which is a product of two large primes p and q, and a pair of numbers (e,d) such that $e \cdot d$ is congruent with 1 modulo $\phi(n)$. Here, ϕ denotes Euler's totient function and, since n is $p \cdot q$, $\phi(n)$ is equal to $(p-1) \cdot (q-1)$. Each user has to publish his n and e, but should keep secret his d.

In RSA, a user A can construct a signature S_A on a suitable numeric message M by computing $S_A(M) = M^{d_A} \bmod n_A$, where d_A is A's secret and n_A is A's published product.

Since for other persons finding d_A is as difficult as factoring n_A (which is, as far as we know, infeasible for large, appropriately chosen n_A), only user A can compute $S_A(M)$ in practice, so long as he keeps his d_A secret. As a consequence, anyone can substantiate a claim that A signed M if he can come up with $S_A(M)$.

However, if somebody comes up with a number S and claims that it is M signed by A, one should be able to check whether indeed S is equal to $S_A(M)$. Since only A can produce $S_A(M)$, this check cannot be performed directly. But anyone can compute $S^{e_A} \bmod n_A$ using the public numbers e_A and n_A. If the result of this computation is not equal to $M \bmod n_A$, then S cannot have been equal to $S_A(M)$, since

$$(S_A(M))^{e_A} \bmod n_A = M^{d_A \cdot e_A} \bmod n_A = M \bmod n_A.$$

Thus, if somebody comes up with a number S for which $S^{e_A} \bmod n_A$ is equal to M, then one should be convinced that A signed M, otherwise not.

In RSA messages must be restricted to natural numbers less than n. (How to deal with larger messages will be explained in section 2.3.) Without this restriction, messages that are equal modulo n would have the same signature. Since n is publicly known, fraud would be too easy. One should not even use all numbers M with $0 \leq M < n$ for messages to be signed.

1.1 Redundancy to prevent a chosen signature attack

Rivest, Shamir and Adleman recommended that n be about 200 decimal digits long, which amounts to about 664 bits. For concreteness and convenience, while retaining an ample margin of safety, we will assume for the rest of this paper that n is 800 bits long.* Thus, a message can

---

* One cannot choose n arbitrarily large, since the practicability of RSA decreases as n increases, due to the increasing computational cost of exponentiation.

comprise as much as 100 bytes.

A forger can choose a number S_c, with $S_c < n_A$, and compute $M_c = (S_c)^{e_A} \bmod n_A$ from it. Subsequently, he could claim successfully that S_c is the message M_c signed by A. Since exponentiation modulo n acts as a one-way function when $\phi(n)$ is unknown, this *chosen signature attack* can be used for finding signatures on "random" (i.e., unpredictable) messages only. (In other words, nobody but A can make the signature on a chosen message, but anybody can determine which message corresponds to a chosen signature.)

To prevent such unpredictable messages from having a reasonable chance of being meaningful, it is necessary to have *redundancy* in the messages. Thus, a distinction will be made throughout the paper between *messages* and *valid messages*. All numbers M with $0 \leqslant M < n$ are messages, but only a very small fraction of these are valid messages. For example, if 100 bits of redundancy is used, a chosen signature will have only a chance of 2^{-100} of corresponding to a valid message. Thus, finding a *false signature* (i.e., a signature on a valid message not actually signed by A) will cost 2^{99} trials on the average, which makes a successful chosen signature attack infeasible.

Finally, notice that messages need redundancy against a chosen signature attack because RSA is a *readable signature* scheme; i.e., a scheme whereby anyone can derive the message signed from the signature.

1.2. Multiplicative attacks

The need for redundancy in RSA messages has been established. To prevent a chosen signature attack only the *quantity* of the redundancy present in valid messages is of interest. Still, the *nature* of the redundancy is also important, since RSA signatures have the property of being *multiplicative*.

For example, suppose that person B can construct three valid messages M_1, M_2 and M_3 such that $M_3 = (M_1 \cdot M_2) \bmod n_A$. Then, if he succeeds in getting M_1 and M_2 signed by A, he can form the product (modulo n_A) of these signatures to get a false signature on M_3, since

$$\begin{aligned} S_A(M_3) &= (M_1 \cdot M_2)^{d_A} \bmod n_A \\ &= ((M_1^{d_A} \bmod n_A) \cdot (M_2^{d_A} \bmod n_A)) \bmod n_A \\ &= (S_A(M_1) \cdot S_A(M_2)) \bmod n_A. \end{aligned}$$

B can also use the inverse M^{-1} or the opposite $-M$ of a message M of which he knows the corresponding signed version, as a factor in a product forming a new message, for, $S_A(M^{-1} \bmod n_A) = (S_A(M))^{-1} \bmod n_A$ and $S_A((-M) \bmod n_A) = (-S_A(M)) \bmod n_A$. (This last equation is true, because d_A is known to be odd.)

Thus, if B knows A's signature on one or more valid messages M_i, he can easily forge a signature for any new valid message which he can discover how to rewrite as a product of

message(s) M_i, their opposite(s) $-M_i$, or their inverse(s) M_i^{-1} (all modulo n_A). (Notice that a message and/or its opposite and/or its inverse may occur in such a product more than once.) Therefore, the redundancy should prevent feasibility of discovering such valid messages.

As already mentioned, the protection that a redundancy scheme offers against a chosen signature attack depends only on the amount of redundancy. For protection against multiplicative attacks, however, the nature of the redundancy is also important, because the difficulty of finding valid messages that are products of other valid messages or their inverses can be different for two redundancy schemes, even if both use the same amount of redundancy. Indeed, for some redundancy schemes it seems feasible to find such combinations even though these schemes command a considerable amount of redundancy [DeJCh85].

For example, in two simple redundancy schemes each message must start (respectively end) with a sequence of, say, 100 zero-bits to be accepted as a valid message. Although these two simple and well-known techniques to add redundancy make a chosen signature attack infeasible, they do not generally provide sufficient protection against multiplicative attacks [DeJCh85].

1.3. Chaining or compression

Since our RSA only provides signatures on messages of at most 800 bits, some method is needed to make it also useful for larger messages.

An obvious approach is to split messages up into appropriate parts, and then to sign each part separately. To ensure that the parts are not re-ordered, they should be *chained* in some way; i.e., each part should contain extra information linking it unambiguously to (an)other part(s).

Another solution is to use a suitable, publicly known, one-way, *compression* function F_c (see also section 4.4), which maps a message of *any* size to some 800-bit number, before applying the RSA signing function. Thus, A's signature on a message M then will be $(F_c(M))^{d_A} \bmod n_A$. Note that the function F_c indeed needs the one-way property. Otherwise, a person having acquired A's signature on some message could determine some (or all) other messages having the same signature, and could claim successfully that he got such messages from A.

The one-way property of F_c implies that the signatures have become *unreadable* (which means that the message cannot be derived from the signature), which has two implications. First, the message itself must be delivered together with the particular number which constitutes its signature; i.e., a signed message consists of the pair $(M\ ,\ (F_c(M))^{d_A} \bmod n_A)$. Second, redundancy no longer has to be present in M to prevent the chosen signature attack. (Clearly, a chosen signature attack does not work if it is infeasible to find a message for which the chosen bit pattern is a true signature.) Thus, any bit pattern may represent a valid message, and a signed message needs only 100 bytes more than the *actual* message (i.e., the original message without added redundancy) itself. Therefore, this scheme may be more (storage-)efficient than RSA with chaining, particularly for large messages.

In the case of RSA with chaining, each part of the message must be signed and/or checked separately. Therefore, the costs of signing and/or checking a large message is linear in its size. (Thus, if a message is, for example, three times as long as another large message, signing it costs three times as much as signing the other one.) When RSA is used with compression, after the compression only one part of 800 bits is still involved. Thus, if the computation required for the one-way compression can be made relatively cheap, RSA with compression will be more (time—)efficient than RSA with chaining (which subsequently will also be called: basic RSA).

Although redundancy is no longer required to prevent chosen signature attack, some well-chosen redundancy may still be necessary to prevent a multiplicative attack. For example, this would be the case when the compression function F_c is a homomorphism with respect to multiplication modulo n on a considerable part of its domain; i.e., if for many M_1 and M_2: $F_c(M_1 \cdot M_2) \bmod n = (F_c(M_1) \cdot F_c(M_2)) \bmod n$.

2. Generalized Exponentiation Signatures

As explained above, the multiplicative property of RSA means in particular that one should be very careful in choosing a redundancy scheme. Instead of looking for a suitable redundancy scheme, we will try to solve the problem by finding a signature scheme that does not have such unwished for structure. Thereto, we introduce a generalization of RSA which uses functions of M and n in the base as well as in the exponent, while preserving the idea of choosing n as a public product of secret primes to keep $\phi(n)$ secret. Thus, in this Generalized Exponentiation Signature (GES) scheme, signatures look like:

$$F_1(M,n)^{F_2(M,n)} \bmod n.$$

Since only knowledge of $\phi(n)$ should provide the ability to make signatures corresponding to n, function F_2 is supposed to comprise at least the computation of an inverse in modulo $\phi(n)$ arithmetic.

Obviously, RSA is a special case of GES whereby F_2 is chosen to be a constant function always mapping to $d = e^{-1} \bmod \phi(n)$, and whereby the function F_1 comprises, for example, the redundancy mapping and/or the compression function. In the following, three other signature schemes, which are special cases of GES, will be investigated. The first two of these will be shown to give problems similar to those of RSA. The third variation of GES seems a more promising alternative to RSA.

2.1 A first variation

In our first example, F_1 and F_2 are chosen to be $M \bmod n$ and $(M \cdot d) \bmod \phi(n)$, respectively. (Note that it makes no difference at all whether F_1 is chosen to be M or $M \bmod n$. Similarly, it is equivalent to have just $M \cdot d$ for F_2.) Thus, a signature looks like:

$$M^{M \cdot d} \bmod n.^*$$

In this scheme the combination of a multiplicative property in the base (like in RSA) and an additive property in the exponent may seem to make it more difficult to tamper with signatures to produce a false one. However, the following counterexample shows that this scheme does not offer a significant improvement.

Suppose one has two messages M_1 and M_2 signed by A. By raising the signature on M_1 to the power M_2 (in modulo n_A arithmetic) one gets:

$$({M_1}^{M_1 d} \bmod n_A)^{M_2} \bmod n_A = ({M_1}^{M_1 M_2 d}) \bmod n_A.$$

Similarly, one can raise the signature on M_2 by M_1. Multiplying both results gives A's signature on $M_1 \cdot M_2$, for,

$$({M_1}^{M_1 M_2 d} \bmod n_A) \cdot ({M_2}^{M_1 M_2 d} \bmod n_A) = (M_1 M_2)^{(M_1 M_2)d} \bmod n_A.$$

Thus, if one knows A's signature on one or more valid messages M_i, one can easily forge a signature for any new valid message that one can discover how to write as a product of message(s) M_i. Consequently, the only achievement is that the inverses of these M_i cannot be used as factors in such products, as was the case with RSA (see section 2.2).

2.2 A second variation

In a second variation of GES, F_1 and F_2 are chosen to map to a constant number C and to the inverse of M in modulo $\phi(n)$ arithmetic, respectively. In this variation A's signature function is:

$$S_A(M) = C^{M^{-1} \bmod \phi(n_A)} \bmod n_A.$$

(How to guarantee that the inverse needed in the exponent does in fact exist will be treated shortly.)

At first sight this signature scheme might seem to offer excellent protection against tampering, since the additive property in the exponent does no harm, because the inverse of a sum is, in general, not equal to the sum of the inverses. Thus, the following inequality usually holds:

$$S_A(M_1) \cdot S_A(M_2) \neq S_A(M_1 + M_2).$$

However, this signature scheme is open to the following attack. Suppose one has A's signature on a valid message M which can be written as the normal integer product of some factors $m_1, m_2, \ldots, m_k$. Thus,

---

* This first variation of GES can also be considered to be an "RSA with compression" signature using the compression function $F_c(M) = M^M \bmod n$.

$$S_A(M) = C^{(m_1 m_2 \ldots\ldots m_k)^{-1} \bmod \phi(n_A)} \bmod n_A.$$

By raising $S_A(M)$ to the power m_1 in modulo n_A arithmetic, one gets $S_A(m_2 m_3 \ldots\ldots m_k)$. Similarly, one is able to get a false signature on any other valid message that can be obtained from M by erasing one or more of its factors.

Another point to consider with this signature scheme, already mentioned, was how to guarantee that the inverse modulo $\phi(n)$, which is used as exponent, does in fact exist. For any number x, its inverse modulo $\phi(n)$ exists only if x is co-prime with $\phi(n)$; i.e., if $gcd(x, \phi(n)) = 1$. However, this restriction poses no serious problems, even though $\phi(n)$ is known to always contain the factor 4.

For example, A can choose his public product n_A as follows. First he searches for two large primes p'_A and q'_A (roughly 400 bits each) such that $2 \cdot p'_A + 1$ and $2 \cdot q'_A + 1$ are also prime. Then A takes $n_A = p_A \cdot q_A$ with $p_A = 2 \cdot p'_A + 1$ and $q_A = 2 \cdot q'_A + 1$. As a consequence, $\phi(n_A)$ will be $4 \cdot p'_A \cdot q'_A$, and thus almost all odd numbers will be co-prime with $\phi(n_A)$. To be more precise, the chance that an odd number is not co-prime with $\phi(n_A)$ is $(p'_A + q'_A - 1) \; / \; (p'_A \cdot q'_A)$, which is negligibly small (roughly 2^{-400}). (Acceptably small probabilities might also be achieved when p'_A and q'_A have only large factors.) To get the number to be inverted to be odd, one could append a 1-bit to M before computing its inverse modulo $\phi(n_A)$. Note that not only $2M + 1$ is guaranteed to be odd, but also $(2M + 1) \bmod \phi(n_A)$, since $\phi(n_A)$ is even.

The change of A's signature function to $S_A(M) = C^{(2M+1)^{-1} \bmod \phi(n_A)} \bmod n_A$ does not result in much more security against tampering. Suppose that $2M + 1$ is known to be a product of some factors $m_1, m_2, \ldots\ldots, m_k$. The product of any subset of these factors then will be odd too, and thus will correspond to some other message M'. For example, if $k > 2$ one is able to get a false signature on the message $M' = (m_1 m_2 - 1) \; / \; 2$.

A safer signature scheme results if a one-way function F_o is used to change the signature function to $S_A(M) = C^{(2 \cdot F_o(M)+1)^{-1} \bmod \phi(n_A)} \bmod n_A$. Making the base number also depend on the message to be signed seems to be another way to improve safety. This approach will be investigated below.

2.3. A third variation

Trying to prevent the attack that appeared to be possible in the previous section, we now choose $F_1(M,n) = M \bmod n$. For the exponent we use again $F_2(M,n) = (2M + 1)^{-1} \bmod \phi(n)$, assuming that n is chosen appropriately as described in the previous section. This time, the F_2 chosen is not only suited for solving the problems resulting from the fact that not every M has an inverse in *mod* $\phi(n)$ arithmetic, but also prevents the following attack that still (!) would be possible in case the signature function would be just $M^{M^{-1} \bmod \phi(n)} \bmod n$.

Suppose one would like to get a false signature on the message M. First, one uses some P and Q to construct three messages $M_1 = MP$, $M_2 = MPQ$ and $M_3 = M^2 PQ$. If one succeeds in getting A to sign M_1, M_2 and M_3, one can forge A's signature on M as follows.

Computing $(S_A(M_2))^Q \bmod n_A$ gives $(MPQ)^{(MP)^{-1}}$. Multiplying this with the mod n_A inverse of $S_A(M_1)$ gives $Q^{(MP)^{-1}}$. In a similar way, one can compute $(MQ)^{(MP)^{-1}}$ from $S_A(M_1)$ and $S_A(M_3)$. Multiplying $(MQ)^{(MP)^{-1}}$ with the mod n_A inverse of $Q^{(MP)^{-1}}$ gives $M^{(MP)^{-1}}$. This last number exponentiated with P gives $S_A(M) = M^{M^{-1}}$.

In the next section we will examine in some detail the properties of the more promising variation which uses $F_1(M,n) = M \bmod n$ and $F_2(M,n) = (2M+1)^{-1} \bmod \phi(n)$. For convenience, this last scheme will be called DJ.

3. Some properties of DJ signatures

3.1. Fixed storage costs

Naturally any GES scheme that uses a function F_2 which is really dependent on M is unreadable. Thus, DJ signatures are unreadable. Therefore, to give a person a message signed by A, one has to send him both the message M and A's signature on it; i.e., one has to send the pair (M, $M^{(2M+1)^{-1} \bmod \phi(n_A)} \bmod n_A$). As a consequence, a signed message requires only 100 bytes more than the message itself (independent of the size of the message, as explained below). This is the same as for RSA with compression.

3.2. No need for redundancy in messages

DJ signatures being unreadable also implies that messages need no redundancy to protect against a chosen signature attack. Since DJ signatures are not multiplicative, messages do not need some well-chosen redundancy to prevent a multiplicative attack either. DJ appears to have no other unwished for properties such as, for example, being additive. And so, it currently seems to be secure against other, similar attacks.

3.3. No chaining required for large messages

With DJ it is not necessary to restrict M to, for example, numbers less than n, as in case of RSA. Of course, all messages that are congruent modulo n and modulo $\phi(n)$ will have the same signature. But this gives no problem, since $\phi(n)$ is supposed to be secret. So, the *implicit* compression that results from the reduction modulo $\phi(n)$, has the required one-way property. As a consequence, it is not necessary with the DJ scheme to use chaining or a *separate* compression function for signing large messages. Furthermore, it is likely that implicit compression is much easier (cheaper) to perform than separate compression, since a separate compression function is likely to involve a much more complex computation than just a reduction modulo $\phi(n)$ of $2M+1$.

3.4. Computational costs

Because of the above mentioned implicit compression, forming a signature costs one exponentiation in modulo n arithmetic to an exponent of at most 800 bits. Thus, signing large messages is hardly more expensive than signing short ones. However, the same is not true for checking a signature. Since only the signer knows his $\phi(n)$, checking a signature has to be done by first raising (in modulo n arithmetic) the given signature to the full $2M+1$ and then checking whether the result is indeed equal to M mod n. Thus, the cost of checking a signature is linear in the size of the message. (In the sense that checking a twice as large message cost twice as much work.) Recall that when RSA is used with chaining, the costs of signing and of checking are *both* linear in the size of the message, since each part of the message must be signed or checked separately.

Of course, as with RSA, it is also possible to use a separate, publicly known, one-way, compression function F_c. Then A's signature on a message M will be $M^{(2\cdot F_c(M)+1)^{-1} \bmod \phi(n_A)} \bmod n_A$.*
If the computation required for such a separate compression can be made relatively inexpensive, it is advantageous to use it for DJ as well, since the second phase of checking then also consists of only one exponentiation to a number of at most 800 bits. (The first phase encompasses the computation of $F_c(M)$.)

To show the different demands that RSA and DJ make upon the one-way function, we will digress now somewhat into the subject of one-way functions. A usual definition is that a function is called one-way if it is not generally feasible to find, for a given y, an x such that $F(x)=y$. However, in the context of cryptographic applications one often adds the requirement that, given some x, it should also be infeasible to find an x' with $x' \neq x$ such that $F(x')=F(x)$.

Since we have chosen to use the function F_c only in the exponent, the only requirement to be imposed on the compression function is that, when knowing some pair (x,y) for which $F_c(x)=y$, it must be infeasible to find an x' for which $x' \neq x$, $x' \bmod n = x \bmod n$ and $F_c(x')=y$. Thus, with DJ the compression function has to be "one-way" only in a more restricted sense. As a consequence, it may be much easier to find suitable compression functions for DJ than for RSA.

4. Comparison of DJ with RSA

Both the RSA and the DJ digital signature scheme are a special case of a GES scheme. The basic form of RSA has several unpleasant properties. Since it only works for messages of less than $\log_2(n)$ bits, large messages must be divided into small parts. To prevent forgeries, it is necessary to include in each of these parts some bits comprising chaining information and redundancy, to prevent a chosen signature attack. Furthermore, since RSA signatures are

---

* We prefer not to use $F_c(M)$ in the base as well.

multiplicative, the redundancy scheme to be used should be chosen very carefully to prevent successful multiplicative attacks. Thus, the security of basic RSA signatures (i.e., RSA with chaining) depends heavily on the quality of the redundancy scheme chosen.

The following properties of DJ signatures give them considerable advantages over basic RSA signatures:

- Messages and signatures are kept separate.
- Since DJ signatures are unreadable and not multiplicative, messages need no redundancy to prevent a multiplicative attack or a chosen signature attack.
- DJ signatures cost $\log_2(n)$ bits beyond the size of the message. In practice, this means less than 100 bytes for a signature.
- The costs for signing a message are more or less constant, because the implicit compression means that only one exponentiation to a number of at most $\log_2 \phi(n)$ bits has to be performed, even for very large messages. (Thus, signing large messages is much cheaper with DJ than with basic RSA, since in the latter case such an exponentiation has to be performed for each part of the message.)
- Every person has to publish only his product n. (The same holds for RSA if one agrees on everybody using the same public exponent e.)

Another way to circumvent the disadvantages of basic RSA signatures is to use RSA with compression. This requires a publicly known, one-way compression function that also destroys the multiplicative structure. With respect to RSA with compression, DJ has only two advantages. The first is that DJ even works without using any compression function. (With DJ, compression s only useful for bounding the costs for checking signatures on large messages.) Second, a ompression function has to meet less requirements in case of DJ than in case of RSA; for example, it is not crucial for DJ whether the compression function destroys multiplicative properties or not. On the other hand, RSA with compression has the advantage that signatures can be checked more economically if everybody uses a relatively small number e as the public exponent. It may be concluded that RSA with compression and DJ both seem to be good digital signature schemes, and that both are better than RSA with chaining.

Summary

It has been shown in [DeJCh85] that RSA signatures may be vulnerable to so-called multiplicative attacks. In this paper we have shown a similar potential weakness in the special cases of the generalizations of RSA presented that use the functions $M^{M \cdot d} \bmod n$, $C^{(2M+1)^{-1} \bmod \phi(n)} \bmod n$, and $M^{M^{-1} \bmod \phi(n)} \bmod n$, respectively.

A further variation is presented that, although it does not rely on the use of a one-way function, does not seem to be vulnerable to multiplicative or other attacks known to the authors.

References

[Chaum85] Chaum, D., Security Without Identification: Transaction Systems to Make Big Brother Obsolete, Communications of the ACM, Vol. 28, No. 10, October 1985, pp. 1030-1044.

[DeJCh85] de Jonge, W. and Chaum, D., Attacks on some RSA Signatures, in: Hugh C. Wiiliams (ed.), Advances in Cryptology - Crypto '85, Lecture Notes in Computer Science, No. 218, Springer Verlag, pp. 18-27.

[RSA78] Rivest, R.L., Shamir, A., and Adleman, L., A Method for Obtaining Digital Signatures and Public-Key Cryptosystems, Communications of the ACM, Vol. 21, No. 2, February 1978, pp. 120-126.