Lecture Notes in Computer Science 1514
Edited by G. Goos, J. Hartmanis and J. van Leeuwen

Springer
*Berlin*
*Heidelberg*
*New York*
*Barcelona*
*Budapest*
*Hong Kong*
*London*
*Milan*
*Paris*
*Singapore*
*Tokyo*

Kazuo Ohta   Dingyi Pei (Eds.)

# Advances in Cryptology – ASIACRYPT'98

International Conference
on the Theory and Application
of Cryptology and Information Security
Beijing, China, October 18-22, 1998
Proceedings

Springer

Volume Editors

Kazuo Ohta
NTT Information and Communication System Labs. 609A
1-1 Hikarinooka, Yokosuka, Kanagawa 249, Japan
E-mail: ohta@sucaba.isl.ntt.co.jp

Dingyi Pei
SKLOIS, Graduate School of USTC
19A Yu Quan Road, Beijing 100039, P.R. China
E-mail: dypei@sun.ihep.ac.cn

Cataloging-in-Publication data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

**Advances in cryptology** : proceedings / ASIACRYPT '98,
International Conference on the Theory and Application of
Cryptology and Information Security, Beijing, China, October 18 -
22, 1998. Kazuo Ohta ; Dingyi Pei (ed.). - Berlin ; Heidelberg ; New
York ; Barcelona ; Budapest ; Hong Kong ; London ; Milan ; Paris ;
Singapore ; Tokyo : Springer, 1998
(Lecture notes in computer science ; Vol. 1514)
ISBN 3-540-65109-8

CR Subject Classification (1991): E.3, G.2.1, D.4.6, K.6.5, F.2.1-2, C.2, J.1

ISSN 0302-9743
ISBN 3-540-65109-8 Springer-Verlag Berlin Heidelberg New York

Typesetting: Camera-ready by author
SPIN 10692702     06/3142 – 5 4 3 2 1 0     Printed on acid-free paper

# Preface

ASIACRYPT'98, the international conference covering all aspects of theory and application of cryptology and information security, is being held at Beijing Friendship Hotel from October 18 to 22. This is the fourth of the Asiacrypt conferences. ASIACRYPT'98 is sponsored by the State Key Laboratory of Information Security (SKLOIS), University of Science and Technology of China (USTC), and the Asiacrypt Steering Committee (ASC), in cooperation with the International Association for Cryptology Research (IACR).

The 16-member Program Committee organized the scientific program and considered 118 submissions. Of these, 32 were accepted for presentation. The authors' affiliations of the 118 submissions and the 32 accepted papers range over 18 and 13 countries or regions, respectively.

The submitted version of each paper was sent to all members of the Program Committee and was extensively examined by at least three committee members and/or outside experts. The review process was rigorously blinded and the anonymity of each submission are maintained until the selection was completed. We followed the traditional policy that each member of the Program Committee could be an author of at most one accepted paper.

These proceedings contain the revised versions of the 32 contributed talks as well as a short note written by one invited speaker. Comments from the Program Committee were taken into account in the revisions. However, the authors (not the committee) bear full responsibility for the contents of their papers.

We are very grateful to the members of the Program Committee for generously spending so much of their time on the difficult task of selecting the papers. They are: Thomos A. Berson, Colin Boyd, Zongduo Dai, Marc Girault, Xuejia Lai, Tzonelih Hwang, Burt Kaliski, Kwangjo Kim, Kouichi Sakurai, Mitsuru Matsui, Andrew Odlyzko, Guozhen Xiao, Lam Kwok Yan, Yuliang Zheng. We also thank the following outside experts who assisted the Program Committee in evaluating various papers: Masayuki Abe, Kazumaro Aoki, Fabrice Boudot, Dengguo Feng, Atsushi Fujioka, Eiichiro Fujisaki, Henri Gilbert, Louis Goubin, Shaoquan Jiang, Masayuki Kanda, Shiho Moriai, Tatsuaki Okamoto, Haiwen Ou, Jacques Patarin, Philippe Toffin, Jacques Traore, Shigenori Uchiyama, Yujie Zhou, Moti Yung, We apologize for any omission in this list.

We would like to appreciate all who have submitted papers to ASIACRYPT'98 and the authors of accepted papers for their on-time preparation of camera-ready manuscripts.

We are also pleased to thank Shu Chang and Chen Lan for their help with preparation of the various tasks of the program co-chairs.

August 1998

Kazuo Ohta
Dingyi Pei

# ASIACRYPT'98

## Beijing, October 18-22, China

**International Conference on the Theory and Application of Cryptology and Information Security**

Sponsored by
**The State Key Laboratory of Information Security**
**University of Science and Technology of China**

and
**The Asiacrypt Steering Committee**

In cooperation with
**The International Association for Cryptologic Research**

**Gerneral Chair**
Keqin Feng (Vice President of USTC)

**Program Committee**
Thomos A. Berson (Anagram Labs., USA)
Colin Boyd (Queensland Univ. of Tech., Australia)
Zongduo Dai (Chinese Academy of Sciences, China)
Marc Girault (France Telecom, France)
Xuejia Lai (R3 Security Engineering, Switzerland)
Tzonelih Hwang (National Cheng Chung Univ., Taiwan)
Burt Kaliski (RSA Labs., USA)
Kwangjo Kim (Information and Communication Univ., Korea)
Kouichi Sakurai (Kyushu Univ., Japan)
Mitsuru Matsui (Mitsubishi Electric Corp., Japan)
Andrew Odlyzko (AT&T, USA)
Kazuo Ohta (Co-chair, NTT, Japan)
Dingyi Pei (Co-chair, SKLOIS, China)
Guozhen Xiao (Xidian Univ., China)
Lam Kwok Yan (National Univ. of Singapore, Singapore)
Yuliang Zheng (Monash Univ., Australia)

**Organizing Committee**
Qing Chang (National Nature Science Fundation, China)
Guang Hua (Graduate School of USTC, China)
Kan Zhang (Chinese Academy of Sciences, China)
Xinkao Song (Beijing Scientific and Technical Society, China)
Zhansheng Zhao (Chairman, SKLOIS, China)

# Contents

## Public Key Cryptosystems

## Invited Talk

## Elliptic Curve Cryptosystems

## Cryptanalysis 1

## Signature

## Cryptanalysis 2

## Finite Automata

## Authentication Codes

## Electronic Cash

## Steam Ciphers

## Cryptographic Protocols

## Key Escrow

## New Cryptography

## Information Theory

# Generating RSA Moduli with a Predetermined Portion

Arjen K. Lenstra

Citibank, N.A., 4 Sylvan Way, Parsippany, NJ 07054, U.S.A.
arjen.lenstra@citicorp.com

**Abstract.** This paper reviews and generalizes a method to generate RSA moduli with a predetermined portion. The potential advantages of the resulting methods are discussed: both the storage and the computational requirements of the RSA cryptosystem can be considerably reduced. The constructions are as efficient as generation of regular RSA moduli, and the resulting moduli do not seem to offer less security than regular RSA moduli.

## 1 Introduction

In [18] Vanstone and Zuccherato presented several methods to generate RSA moduli that contain a certain predetermined portion. They describe scenarios where such moduli may be useful by reducing the storage requirements of RSA moduli without compromising security. For instance, all members of a group of users may share some fixed number of bits of their RSA moduli, or users may want to include a binary representation of their personal data in their RSA modulus. For DSA keys with a predetermined portion see [13].

For an N-bit RSA modulus Vanstone and Zuccherato are able to specify up to N/2 leading bits. Their method for doing so is, however, quite inefficient. They also present a faster method that allows specification of up to N/4 leading bits, and a compromise scheme of intermediate speed that specifies between N/4 and N/2 leading bits. All these methods are rather cumbersome and require factorization of the number given by the specified leading bits. A more serious disadvantage of the leading bits methods from [18] is that Coppersmith has shown in [12] that the resulting moduli are substantially easier to factor than a general product of two large primes.

Perhaps the most surprising aspect of the Vanstone/Zuccherato method is why they chose such a complicated method and apparently overlooked the obvious and straightforward trick that is reviewed in this paper. Not only is it elementary, it also does not seem to be affected by any known attack. This ‘follow your nose’ approach was known to at least some people, among them Coppersmith (cf. [3]), Quisquater (cf. [12]), and Shamir (cf. [16]), but most people, including the present author, were unaware of it. Allegedly (cf. [12]), it is used in a 1984 French banking standard. Attempts to access the reference [9] to this standard failed. Sakurai pointed out that the method is described in [19] for a different application. Apparently, the trick was independently reinvented many times, which is not so strange given how simple it is.

K. Ohta and D. Pei (Eds.): ASIACRYPT’98, LNCS 1514, pp. 1-10, 1998.

The method presented in this paper allows generation of RSA moduli with any number of predetermined leading bits, with the fraction of specified bits only limited by security considerations. The basic method is as efficient as regular generation of RSA moduli. Several generalizations are described as well: a slower version that allows specification of slightly more bits, a method to specify any number of trailing bits, and combined methods where the specified bits are split among the leading and trailing bits of the modulus. In all methods ‘bits’ may be replaced by digits with respect to any radix. The method to specify trailing bits is a simple modification of the basic method. It was already described in [18] and is included for completeness.

Coppersmith’s attack does not affect any of the methods presented here (and therefore neither the security of the Vanstone/Zuccherato trailing bits method). Neither does any other known attack seem to affect the security of the moduli as generated by the methods presented here. Obviously that does not imply a proof of security. It can be proved that for a randomly selected predetermined portion the resulting moduli cannot be distinguished from regular RSA moduli. This is about the strongest security result one may hope for in this context. Proving absolute security of the schemes themselves is an entirely different matter. Such a proof is unlikely. Some confidence in the strength of the methods may be provided by the fact that several eminent cryptanalists have been aware of the basic method for many years without being able to break it.

More or less the opposite approach to randomly selecting a predetermined portion is to select it in such a way that the resulting moduli are relatively close to a power of 2. According to [12] both Quisquater and the French banking standard focussed on this particular application, because it allows entirely division-free and thus much faster modular multiplication. Intuitively it sounds like a bad idea, but when a few straightforward precautions are taken no published factoring method can take substantial advantage of the special form of the modulus. Both in the May 1998 draft of the forthcoming ANSI X9.31 standard (cf. [1]) and in [17] it is mentioned that RSA moduli of the form $2^{64x} \pm c$ should not be used because they would be ‘readily susceptible’ to the special version of the number field sieve integer factoring algorithm. Neither [1] nor [17] specify how such moduli are generated, but if one of the methods presented in this paper is used, then they are not vulnerable to such an attack.

The X9.31 standard contains a number of criteria to be satisfied by primes dividing an RSA modulus. Some of these criteria make sense and can easily be satisfied, either by construction or by rejecting the (sufficiently small) fraction of moduli that violate one of the criteria. Other criteria are meant to protect against certain attacks but do, in fact, not offer any additional protection and provide only a false and misleading sense of security. Attempts to satisfy these latter criteria simply do not make sense, as argued in [15] as well. Therefore, the X9.31 criteria have not been incorporated in the methods presented in this paper. It should be kept in mind, however, that incorporation is possible and that in practical circumstances some of the criteria will have to be taken into account.

This paper is organized as follows. In Section 2 the basic method and its generalizations are presented. Section 3 comments on the security of the proposed methods, and in Section 4 RSA moduli that are relatively close to a power of 2 are discussed.

## 2 Generating RSA Moduli with a Predetermined Portion

Throughout this section $d \in \mathbf{Z}_{>1}$ denotes a fixed radix. A *digit* refers to a digit in the radix d representation. For a positive integer r its *length* |r| refers to the length of r’s radix d representation with non-zero leading digit.

The length of the RSA modulus n to be constructed is denoted by N, the length of the predetermined portion s of n is denoted by K with K < N, and L = N–K. The concatenation of two arrays of digits $a_1$ and $a_0$ is denoted $a_1||a_0$. In this section methods are presented to construct RSA moduli with radix d representation s||r, r||s, and $s_1||r||s_0$, where r is an array of L digits and $s = s_1||s_0$. Throughout this section it is assumed that N, K, and L are sufficiently large.

**(2.1) Fixing the leading digits of n.** Let s be a number of length K. First compute the number $n' = s*d^L$ of length N. Next, pick a random prime p of length at most L, round n' up to the nearest multiple of p, and let q’ be the integer such that n' = p∗q'. Finally, find the smallest non-negative integer m such that q = q' + m is prime. If the resulting n = p∗q is of the form s||r, then return n, p, and q, and terminate; otherwise start all over again with the same s.

**Remarks.** Note that $[p*q'/d^L] = s$ and that n = s||r, i.e., $[n/d^L] = [p*(q'+m)/d^L] = s$, holds if $p-1+m*p < d^L$ (where the ‘p–1’ results from the rounding up). Because of the Prime Number Theorem, m may on average be expected to be of order ln(N–|p|). It follows that if p is chosen such that |p| is approximately equal to L–ln(K), then Algorithm (2.1) may be expected to find an RSA modulus in time O(ln(L)+ln(K)): O(ln(L)) steps to find a random p and O(ln(K)) to find q given q'. This is the same as the expected time needed for regular generation of an RSA modulus of length N consisting of the product of a length L and a length K prime.

If |p| is chosen closer to L (or, equivalently, if the length of s is chosen larger while keeping |p| fixed) then Algorithm (2.1) may be expected to require more iterations before it is successful. The largest |p| (or, equivalently, largest |s|) would require q' to be prime, in which case Algorithm (2.1) may be expected to find an RSA modulus in time O(ln(L)∗ln(K)). This is of course substantially slower than regular RSA modulus generation, but maximizes the length of the predetermined portion as N–|p|.

If L–|p| is chosen larger, then q can be chosen at random from among the primes in a much wider range above q’. Obviously, for random s, this makes q much closer to a random prime than is the case in Algorithm (2.1).

The number n' may also be defined as $s*d^L+d^L-1$, rounded down instead of up, after which the smallest non-negative m such that q'–m is prime should be determined. Or the last L digits of n' may be randomized in either version. Also, truly random digits may be appended to s, or p may be the product of several (sufficiently large) primes. Or any of numerous other minor modifications may be applied to Algorithm (2.1). Note that Algorithm (2.1) does not impose any size restrictions on p and q in addition to the standard size restrictions for factors of RSA moduli.

A similar straightforward construction can be used to fix the trailing digits of n. Instead of dividing the (shifted) pattern by a random prime and appropriately changing the trailing digits, the pattern may be divided by a random prime modulo a power of the radix, after which the leading digits are changed appropriately. The resulting method is identical to the method in Section 7 of [18]:

**(2.2) Fixing the trailing digits of n.** Let s be an array of K digits that corresponds to an odd number (where s may have leading digits zero). First pick a random prime p of length at most L, and a random number x of length $L-|p|$. Next, let $q' = x*d^K + ((s/p) \bmod d^K)$ and let $n' = p*q'$. Finally, find the smallest non-negative integer m such that $q' + m*d^K$ is prime, and let q be $q' + m*d^K$. If the resulting $n = p*q$ has length N, then return n, p, and q, and terminate; otherwise start all over again with the same s.

**Remarks.** The inverse of p modulo $d^K$ exists since p is prime and d may be assumed to be much smaller than p. Furthermore, $|n'| \approx K+L-|p|+|p| = N$ and $n \bmod d^K = p*(q'+m*d^K) \bmod d^K = p*(x*d^K + ((s/p) \bmod d^K)) \bmod d^K = s \bmod d^K = s$. The resulting n has length at most N as long as m is not too large. Combined with the fact that the Prime Number Theorem also holds in arithmetic progressions it follows that the run time analysis of Algorithm (2.2) is similar to the run time analysis of Algorithm (2.1). Also, more or less the same modifications can be applied.

Algorithms (2.1) and (2.2) can be combined into at least two different methods to predetermine leading and trailing portions of the digits of an RSA modulus. The conceptual ideas of the two methods are presented below. Let $s = s_1||s_0$ with $|s_1| = K_1$, $|s_0| = K_0$, $K = K_1+K_0$, $s_0$ odd if $K_0 > 0$, and assume that $N \approx 2K$. The constructions immediately generalize to any $N > 2K$. In general $N < 2K$ cannot be achieved, i.e., at most half the bits of the resulting modulus can be predetermined.

**(2.3) Fixing the leading and trailing digits of n.** Pick a random prime p of length K, and write $p = p_1||p_0$ with $|p_1| = K_1$ and $|p_0| = K_0$. As in Algorithm (2.1) divide $s_1$ by $p_1$ to get $q_1$, the leading $K_1$ digits of q. As in Algorithm (2.2) divide $s_0$ by $p_0$ modulo $d^{K_0}$ to get $q_0$, the trailing $K_0$ digits of q. Let $q' = q_1||q_0$. Find the smallest non-negative integer m such that $q'+m*d^{K_0}$ is prime, and let $q = q'+m*d^{K_0}$.

**(2.4) Alternative (slow) method of fixing the leading and trailing digits of n.** Pick a number $p_1$ at random with $|p_1| = K_1$, the leading $K_1$ digits of p. As in Algorithm (2.1) divide $s_1$ by $p_1$ to get $q_1$, the leading $K_1$ digits of q. Pick an array $q_0$ of $K_0$ random digits, the trailing $K_0$ digits of q. As in Algorithm (2.2) divide $s_0$ by $q_0$ modulo $d^{K_0}$ to get $p_0$, the trailing $K_0$ digits of p. Iterate choice of $q_0$ (or add 1 to $q_0$ and adapt $p_0$ accordingly) until $p = p_1||p_0$ and $q = q_1||q_0$ are prime.

**Remarks.** In both (2.3) and (2.4) the resulting $n = p*q$ has trailing $K_0$ digits equal to $s_0$ and leading $K_1$ digits close to $s_1$. The leading $K_1$ digits can be made equal to $s_1$ by including breathing space after the $K_1$-th but before the $K_0$-th digit, similar to the construction explained in the analysis of Algorithm (2.1). The details can be filled in easily.

Algorithm (2.3) runs in expected time O(ln(K)): p is selected first after which q follows, as in Algorithms (2.1) and (2.2). In Algorithm (2.4) parts of p and q are selected at random, after which the complementary parts follow. Because the primes are constructed simultaneously Algorithm (2.4) runs in expected time $O(\ln(K)^2)$. It is unclear if the approach of spreading the randomness between the two factors as in Algorithm (2.4) has any advantages compared to the more direct approach of Algorithm (2.3). With $K_0 = 0$ Algorithm (2.3) generalizes to Algorithm (2.1) with $N \approx 2K$, and with $K_1 = 0$ Algorithm (2.3) generalizes to Algorithm (2.2) with $N \approx 2K$.

The lengths of p and q do not have to be the same, as follows from the following generalized version of Algorithm (2.3).

> Pick a random prime p of length L ($L = N-K$), and write $p = p_1||p_0$ with $|p_1| = L_1$ and $|p_0| = L_0$. As in Algorithm (2.1) divide $s_1d^{L_1}$ by $p_1$ to get $q_1$, the leading $K_1$ digits of q. As in Algorithm (2.2) divide $s_0$ by $p_0$ modulo $d^{K_0}$ to get $q_0$, the trailing $K_0$ digits of q. Let $q = q_1||q_0$. Keep adding $d^{K_0}$ to q until q is prime.

A similar change applies to Algorithm (2.4). In both generalizations at most half the bits of the resulting modulus can in general be predetermined.

## 3 Security Considerations

During regular generation of RSA moduli two primes, say p and q, are randomly and independently selected, and their product, say n, is made public. Despite the independence of p and q, however, the prime q is determined by p and a complementary portion of only $[\log_d(n)-\log_d(p)]$ leading and/or trailing radix d digits of n, for any $d > 1$. In Algorithms (2.1), (2.2), and (2.3) the prime factor q is, by construction, determined by the choice of p and the predetermined portion s of complementary length. It follows that this situation is identical to regular RSA moduli, as long as the complementary portion s of the modulus is randomly selected and as long as only $m = 0$ is allowed in Algorithms (2.1), (2.2), and (2.3). (Instead of requiring $m = 0$, a much larger range of m's may be allowed than in Algorithms (2.1), (2.2), and (2.3), to deal with the 'unfair' advantage of primes ending a long arithmetic progression of composites. Note, however, that many efficient prime generation methods used in RSA moduli generation have the same bias.) A slightly more involved argument applies if portions of p and q are randomly selected (and s is random), as in Algorithm (2.4). Thus, if the predetermined portion is randomly selected then the moduli as constructed by the proper variations of the methods from Section 2 cannot be distinguished from regular RSA moduli. A similar argument appeared in [19].

Even if the predetermined portion is not randomly selected it is in general unclear how to distinguish any particular modulus constructed as in Section 2 from a regular RSA modulus. A predetermined portion that looks random to an unsuspecting outsider may consist of some contrived encoding of useful information, such as a key merged with a block cipher encryption using that key. An insider who knows the encoding scheme for the predetermined portion has an advantage if no precautions are taken to hide the length of the factors. How this advantage may be used to factor the modulus is unclear, as long as none of the factors is chosen too small and p does not depend on the predetermined portion (as in [19]). Hiding the length of the factors may for instance be done by adding truly random bits to the predetermined portion s, as mentioned in Section 2, or by forcing m as in Section 2 to be such that s gets extended, a modification not explicitly mentioned in Section 2.

Once a predetermined portion has been recognized, for instance because it is shared by many moduli, it is hard to imagine how it would help to factor any of them, since anyone can generate such moduli. Furthermore, it is very unlikely that the security of a regular RSA modulus is affected by using one of the algorithms from Section 2 to generate a modulus having a large portion in common with it. Note that shared leading digits can easily be recognized, mostly irrespective of the d used during construction and the radix used for representation of the moduli. A common trailing portion may be harder to recognize if different radices are used during construction and representation. One very minor problem with a shared portion is that there is a larger probability that two participants end up with the same modulus, but this probability is of the same order of magnitude that someone guesses one of the factors of an RSA modulus, and may thus be neglected.

Predetermined portions that lead to special computational properties of the resulting moduli are discussed in the next section.

## 4 Moduli of a Special Form

Algorithm (2.1) can be used to generate RSA moduli of the form $d^N \pm t$ for positive t’s that are substantially smaller than $d^N$. On computers where numbers are internally represented using radix d numbers, arithmetic operations modulo such RSA moduli can be carried out very efficiently because divisions can entirely be avoided. Let the radix d representation of t have N/c digits for some c > 1. Then reduction modulo $d^N \pm t$ of a product of approximately 2N radix d digits can be done in approximately N/(N−N/c) multiplications of two numbers of N–N/c and N/c radix d digits, plus some additions. In standard arithmetic this amounts to approximately $N^2/c$ multiplications, which is c times faster than ordinary or Montgomery reduction (cf. [10]) modulo numbers of the same size. Moreover, ordinary reduction requires on the order of N low level divisions. The above division-free reduction can be expected to make modular exponentiation approximately 5c/(2c+3) times faster, a speed-up that can be increased by using Karatsuba multiplication during the reduction process.

Algorithms (2.3) or (2.4) or their generalizations can be used to generate RSA moduli of the form $d^N \pm td^M \pm 1$ with M < N/2 and |t| ≈ N/2. As above, reduction modulo $d^N \pm td^M \pm 1$ of a product of approximately 2N radix d digits can be done in approximately $N^2/2$ multiplications. Algorithm (2.2) can be used to generate N-digit RSA moduli of the form $td^M \pm 1$. It is easy to see that such moduli lead to similar speed-ups, not only when using standard arithmetic but also when Montgomery arithmetic (cf. [10]) is used.

It is suggested in [1] and [17] that such special form moduli are easier to factor than regular RSA moduli. If that is indeed the case, then using them cannot be recommended. Below some characteristics of the currently known factoring algorithms are discussed and how their speed may be affected by factoring the above N-digit special composites as opposed to regular RSA moduli having N radix d digits. Without loss of generality it is assumed that d = 2 and that regular 1024-bit RSA moduli may be considered to be sufficiently secure against factoring attacks. The question addressed is how small t may be chosen so that the factorization of $2^{1024}\pm t$ does not become substantially easier than the factorization of a regular 1024-bit RSA modulus.

**Elliptic curve method.** The elliptic curve method (cf. [8]) is good at finding small factors, with only polynomial dependence on the size of the number being factored. Given ample practical experience with this method it may safely be assumed that $2^{1024}\pm t$ cannot be factored by the elliptic curve method as long as the smallest factor is at least $2^{300}$, even if the implementation takes advantage of the special arithmetic properties of the number $2^{1024}\pm t$ being factored. Because in Algorithm (2.1) the size of t corresponds to the size of one of the factors it follows that t should be at least $2^{300}$ and at most $2^{724}$.

**Other special purpose methods.** With a smallest prime factor that is already at least $2^{300}$ it may safely be assumed that $2^{1024}\pm t$ is secure against trial division and Pollard's rho method, and may be expected to be secure against Pollard's p–1 method and variations thereof (cf. [6]). Since the size of t corresponds to the size of the factor p that is randomly selected at the beginning of Algorithm (2.1), explicit protection against the latter methods may even be included (despite [15] and the author's reservations about such protections).

**Number field sieve.** If $2^{1024}\pm t$ can be written as f(m) for an integer m and integral polynomial f of reasonably low degree d, say between 3 and 10, and with coefficients substantially smaller than the d-th or (d+1)-st root of n, then the number field sieve (cf. [7]) runs substantially faster than for general 1024-bit numbers. Actually, if the coefficients are bounded by constants the much faster 'special' number field sieve applies. Note that **all** coefficients need to be small to get a substantial speed-up – as long as even one of them is close to the d-th or (d+1)-st root of n hardly any speed-up will be obtained. For $2^{1024}\pm t$ with a t that may be expected to behave as a random number of at least 300 bits the probability is negligible that such a polynomial f with unusually small coefficients exists. Thus the number field sieve cannot be expected to factor numbers of the form $2^{1024}\pm t$ (as generated by Algorithm (2.1)) faster than regular 1024-bit RSA moduli, if the number $2^{1024}\pm t$ is already properly protected against the elliptic curve method.

**Quadratic sieve.** In the quadratic sieve factoring method (cf. [14]) one attempts to find many smooth numbers close to the square root of the number being factored, where a number is smooth when all its prime factors are smaller than some specified bound. Given the way the numbers that are inspected for smoothness are generated, their size or smoothness probability is not affected by the special form of $2^{1024}\pm t$. Thus quadratic sieve cannot be expected to factor numbers of the form $2^{1024}\pm t$ faster than regular 1024-bit RSA moduli.

**Continued fraction method.** The continued fraction method (cf. [11]) uses the continued fraction expansion of the squareroot of the number n being factored to generate numbers that are inspected for smoothness. In general these numbers are at most $2\sqrt{n}$. The recursion used to generate them does not produce numbers of significantly different size if n is of the form $2^{1024}\pm t$. The continued fraction method can therefore not be expected to be able to take advantage of the special form of $2^{1024}\pm t$.

**Cubic sieve.** If integers a, b, c, can be found such that $b^3 \neq a^2c$ and $b^3 \equiv a^2c$ modulo $2^{1024}\pm t$ (or if a similar identity holds), then $2^{1024}\pm t$ can be factored by means of the cubic sieve (cf. [4]) in approximately the same time it would take quadratic sieve to factor a general number of the same order of magnitude as $(\max(a,b,c))^2$. Thus, the identity $2*(2^{341})^3 \equiv \pm t$ modulo $2^{1024}\pm t$ may make it possible to factor $2^{1024}\pm t$ in the same time it would take quadratic sieve to factor a number of order $\max(2^{682},t^2)$, assuming that t is square-free. It should be noted that this estimate is probably rather optimistic as far as the speed of the cubic sieve is concerned, because computational experience with the cubic sieve is very limited. For a conservative estimate of the security of $2^{1024}\pm t$ this optimism is justifiable.

The number field sieve factorization of a number $\approx 10^{130}$ could have been completed in one fifth of the time of the quadratic sieve factorization of a number $\approx 10^{129}$ (cf. [5]). Combining this conservative estimate with the asymptotic run times of the number field sieve and quadratic sieve, it follows that a 780-bit number offers approximately the same amount of security against a quadratic sieve attack as a 1024-bit number offers against a number field sieve attack. Thus, for square-free t, the security of $2^{1024}\pm t$ is not affected by the cubic sieve if $\max(2^{682},t^2) \approx 2^{780}$. It follows that t should be of the form $a^2t'$ where t' has at least approximately 390 bits and is square-free. This condition either can be satisfied by factoring (and possibly rejecting) a 390-bit t as found by Algorithm (2.1), a rather impractical and cumbersome process, or it may safely be assumed to hold on probabilistic grounds by using a larger t of, say, 500 bits.

**Zhang's method.** If $2^{1024}\pm t$ can be written as $m^3+c_2m^2+c_1m+c_0$, then the method from [20] factors $2^{1024}\pm t$ in approximately the same time it would take quadratic sieve to factor a general number of the same order of magnitude as $2^{682}*\max_{0\le i\le 2}c_i$. Because vulnerability to this method implies vulnerability to the number field sieve, no speed-up can be expected if the number $2^{1024}\pm t$ is already properly protected against the elliptic curve method.

**Other general purpose factoring methods.** There do not seem to be any special properties of any other published general purpose factoring algorithms, such as Dixon's random squares method or the various methods using class groups, that may affect the security of numbers of the form $2^{1024}\pm t$ with, say, $2^{500} < t < 2^{700}$ as constructed by Algorithm (2.1).

**Conclusion.** It follows from the above brief factoring survey that numbers of the form $2^{1024}$±t as constructed by Algorithm (2.1) offer regular 1024-bit RSA security, as long as t is not much smaller than $2^{500}$, and that square-free t's as small as 400 bits may even be used. Furthermore, t should not be much bigger than $2^{700}$. Thus RSA operations could be made at least 30% faster (using the analysis presented above with c = 2), while at the same time considerably simplifying the division code and saving storage space for RSA moduli. These are a rather minor advantages compared to the enormous disadvantage of a security breach when the above conclusion happens to be incorrect.

After some straightforward modifications the same factoring analysis (and thus the same conclusion) applies to the special form moduli generated using Algorithms (2.2), (2.3), or (2.4).

**Acknowledgments**. Acknowledgments are due to Don Coppersmith, David Naccache, Kouichi Sakurai, Adi Shamir, and, indirectly, Moti Yung, for their help in tracking down 'prior art', and to the Asiacrypt'98 program committee for helpful remarks.

## References

1. ANSI X9.31: American national standard for financial services – Digital signatures using reversible public key cryptography for the financial services industry (rDSA). Working draft (May 1998)
2. Coppersmith, D.: Finding a small root of a bivariate integer equation; factoring with high bits known. In: Maurer, U. (ed.): Advances in Cryptology – Eurocrypt'96. Lecture Notes in Computer Science, Vol. 1070. Springer-Verlag, Berlin Heidelberg New York (1996) 178-189
3. Coppersmith, D.: Personal communication (1998)
4. Coppersmith, D., Odlyzko, A.M., Schroeppel, R.: Discrete logarithms in GF(p). Algorithmica 1 (1986) 1-15
5. Cowie, J., Dodson, B., Elkenbracht-Huizing, R.M., Lenstra, A.K., Montgomery, P.L., Zayer, J.: A world wide number field sieve factoring record: on to 512 bits. In: Kim, K., Matsumoto, T. (eds.): Advances in Cryptology – Asiacrypt'96. Lecture Notes in Computer Science, Vol. 1163. Springer-Verlag, Berlin Heidelberg New York (1996) 382-394
6. Knuth, D.E.: The art of computer Programming, Vol. 2, Seminumerical algorithms, 2nd edn. Addison Wesley (1981)
7. Lenstra, A.K., Lenstra, Jr., H.W. (eds.): The development of the number field sieve. Lecture Notes in Mathematics, Vol. 1554. Springer-Verlag, Berlin Heidelberg New York (1993)
8. Lenstra, Jr., H.W.: Factoring integers using elliptic curves. Ann. of Math. 126 (1987) 649-673
9. Les Specifications et les Normes de la Carte à Memoire Bancaire, Groupement Carte à Memoire, 96 rue de la Victoire, 75009, Paris, Janvier 1984, chapitre 6: La Securité.
10. Montgomery, P.L.: Modular multiplication without trial division. Math. Comp. 44 (1985) 519-521

11. Morrison, M.A., Brillhart, J.: A method of factoring and the factorization of $F_7$. Math. Comp. 29 (1975) 183-205
12. Naccache, D.: Personal communication (1998)
13. Naccache, D., M'Raïhi, A., Vaudenay, S., Raphaeli, D.: Can D.S.A. be improved? – Complexity trade-offs with the digital signature standard. In: DeSantis, A. (ed.): Advances in Cryptology – Eurocrypt'94. Lecture Notes in Computer Science, Vol. 950. Springer-Verlag, Berlin Heidelberg New York (1995) 77-85
14. Pomerance, C.: Analysis and comparison of some integer factoring algorithms. In: Lenstra, Jr., H.W., Tijdeman, R. (eds.): Computational methods in number theory. Math. Centre tracts 154/155. Math. Centrum, Amsterdam (1982) 89-139
15. Rivest, R.L., Silverman, R.D.: Are 'strong' primes needed for RSA?. In: RSA'98 conference binder.
16. Shamir, A.: Personal communication (1998)
17. Silverman, R.D.: Fast generation of random, strong RSA primes. RSA Laboratories' Cryptobytes 3, number 1, (1997) 9-13
18. Vanstone, S.A., Zuccherato, R.J.: Short RSA keys and their generation. Journal of Cryptology 8 (1995) 101-114
19. Young, A. Yung, M.: The dark side of „Black-Box“ cryptography, or: should we trust Capstone? In: Koblitz, N. (ed.): Advances in Cryptology – Crypto'96. Lecture Notes in Computer Science, Vol. 1109. Springer-Verlag, Berlin Heidelberg New York (1996) 89-103
20. Zhang, M.: Factorization of the numbers of the form $m^3+c_2m^2+c_1m+c_0$. In: Buhler, J.P. (ed.): Algorithmic Number Theory – ANTS-III. Lecture Notes in Computer Science, Vol. 1423. Springer-Verlag, Berlin Heidelberg New York (1998) 131-136