Published in L. Chen, Y. Mu, and W. Susilo, Eds, *Information Security Practice and Experience (ISPEC 2008)*, vol. 4991 of *Lecture Notes in Computer Science*, pp. 116–130, Springer, 2008.

RSA Moduli with a Predetermined Portion: Techniques and Applications

Marc Joye

Thomson R&D France
Technology Group, Corporate Research, Security Laboratory
1 avenue Belle Fontaine, 35576 Cesson-Sévigné Cedex, France
marc.joye@thomson.net – http://www.geocities.com/MarcJoye/

Abstract. This paper discusses methods for generating RSA moduli with a predetermined portion. Predetermining a portion enables to represent RSA moduli in a compressed way, which gives rise to reduced transmission- and storage requirements. The first method described in this paper achieves the compression rate of known methods but is fully compatible with the fastest prime generation algorithms available on constrained devices. This is useful for devising a key escrow mechanism when RSA keys are generated on-board by tamper-resistant devices like smart cards. The second method in this paper is a compression technique yielding a compression rate of about 2/3 instead of 1/2. This results in higher savings in both transmission and storage of RSA moduli. In a typical application, a 2048-bit RSA modulus can fit on only 86 bytes (instead of 256 bytes for the regular representation). Of independent interest, the methods for prescribing bits in RSA moduli can be used to reduce the computational burden in a variety of cryptosystems.

Keywords. RSA-type cryptosystems, RSA moduli, RSA key lengths, diminished-radix moduli, key compression, key generation, key transport, key storage, key transmission, key escrow, tamper-resistant devices, smart cards, kleptography, setup.

1 Introduction

In 1976, Diffie and Hellman introduced the concept of public-key cryptography [11]. Soon after Rivest, Shamir, and Adleman proposed a concrete realization that works for encryption as well as for digital signatures: the so-called RSA algorithm [26]. RSA has withstood years of extensive cryptanalysis (see e.g. [4]) and is still the most widely deployed and used public-key cryptosystem.

The security of RSA relies on the problem of factoring large numbers, or more exactly, on the problem of computing roots modulo a large composite number. The largest factored RSA modulus is RSA-200 (663 bits), whose factorization was reported by a team of German researchers on May 2005 [28]. Current RSA-based applications typically use 1024-bit RSA moduli but we observe a trend to push for larger moduli like 2048 bits or 4096 bits. It is meaningless to define a threshold value for the key length separating security from insecurity. Security

is defined relatively to a security model: the adversarial goal and the adversary's resources. Further, it is highly dependent on the implementation. There is no need for strong cryptographic algorithms if they are poorly implemented. Clearly, selecting the "appropriate" key length for a given application is a touchy problem. A recent list of recommended key lengths and guidelines is provided in [12].

At first glance, the safe approach would be to increase the key length to (hopefully) increase the security level. Amdahl's law applied to security says that strengthening a secure part (in this case using larger keys) does not help much [16]. A system is as secure as its weakest point and cryptography is rarely the weakest point. More importantly, the use of larger keys comes at a cost. First, it affects the performance in terms of speed. Second, larger keys consume more bandwidth. Third, larger keys imply more memory requirements for their storage. This last point is particularly relevant for constrained devices whose price is mainly dictated by the size of their different memories.

This paper is aimed at mitigating the impacts resulting from the use of larger RSA keys. We develop simple methods to reduce considerably the transmission and storage requirements of RSA moduli. We show how to construct RSA moduli where many bits can be prescribed. Furthermore, such RSA moduli can give rise to substantial speed improvements.

Let n denote the bit length of an RSA modulus and t the number of prescribed bits. If the prescribed portion of the RSA moduli is shared among a group of users, only $(n - t)$ bits are needed to represent the RSA modulus of each user together with a single copy of the t bits used by the entire group. Another scenario is to use t bits of an RSA modulus to represent the user's identity and other publicly available information. In this case, there is no need to store or to transmit the value of those t bits and an RSA modulus can be encoded with only $(n-t)$ bits. Alternatively, one can imagine that a string of t bits is constructed by applying a public function to some seed σ. An RSA modulus is then represented by σ and $(n-t)$ bits; the so-obtained representation is particularly advantageous for short σ's. Yet another application is to make use of t bits of an RSA modulus to convey certain information. This information may appear in clear or in an encrypted form. An example of the first case is presented in [22] where the representation of a DSA prime, $p = \alpha\, q + 1$, embeds the value of prime divisor q and a certificate of proper generation. The second case (namely, encrypted form) can be helpful for key-escrow purposes. For example, t bits of public modulus N can be used to encode the encryption of the corresponding private key or a part thereof.

A modulus N is called a diminished-radix (DR) modulus if it has the special form $N = 2^n - \mu$ for some $\mu < 2^{n-t}$. DR moduli are attractive for implementing the RSA (see, e.g., [21, 24, 34]) because they greatly simplify the modular reductions, which can be computed with only shifts, additions and single-precision multiplications. RSA moduli N can also be constructed under a sparse form, that is, with a reduced Hamming weight. This can be useful for the Paillier cryptosystem [25] and its derivatives where one computes N^{th} powers.

1.1 Related work

In [33], Vanstone and Zuccherato described several methods for generating RSA moduli with a predetermined portion. Typically, for a n-bit RSA modulus, they were able to specify up to $n/2$ bits but in a rather inefficient way. They also proposed a faster method for specifying up to $n/4$ bits, as well as mixed methods of intermediate efficiency for specifying between $n/4$ and $n/2$ bits. A much simpler yet more secure method for specifying up to $n/2$ bits was later presented at ASIACRYPT '98 by Lenstra. This method appears to have being reinvented many times (see [18] and the references therein). It is similar to the method given in [17] for producing RSA moduli with many leading 1-bits and further discussed in [20]. More recently, Shparlinski [29] considered an alternative approach and derived a method with a rigorous analysis for specifying about $n/4$ bits. In [2] (see also [3]), Bernstein reports an unpublished result by Coppersmith for specifying up to $2n/3$ bits using lattice reduction.

Methods for prescribing part of the public key are also found in key-escrow systems: authorities want decryption keys to be escrowed for law enforcement purposes. Pairs of public/private keys are generated using a setup (secretly embedded trapdoor with universal protection) mechanism [35, 36] by trusted third parties or tamper-resistant devices. The notion of setup is related to that of subliminal channel due to Simmons [30] (see also [10]). The setup allows the secure leakage of the private key from the corresponding public key. For RSA cryptosystem, RSA modulus $N = p\,q$ (and/or public exponent e) is used to secretly embed a representation of secret factor p [35] (see also [10]) or private RSA exponent d [9].

1.2 Our contribution

A simple yet efficient method for constructing RSA moduli with a predetermined portion referred to as 'folklore method' is given in [18]. This method enables to specify about half the bits of an RSA modulus. This paper presents other methods for generating such moduli and discusses associated applications.

The first method is a simple variation of the folklore method. It generates an RSA modulus $N = p\,q$ with a predetermined portion (up to its half) by randomly generating prime p and then prime q from a *prescribed interval*. The folklore method proceeds similarly except that prime q is constructed incrementally, which may result in prohibitively too long running times for constrained devices like smart cards. We note that the fastest smart-card implementations of prime generation algorithms [14] (see also [15]) require primes p and q to be chosen in a prescribed interval. Efficient generation of RSA moduli with a predetermined portion on smart cards is particularly relevant for key-escrow purposes as this additional feature is often implemented through tamper resistance.

The second method makes use of an extended version of Euclid's algorithm and enables to generate RSA moduli where about the *two thirds* can be prescribed. Prescribing more than one half is for example very useful to ensure the interoperability of RSA-enabled devices with different key lengths. There are still

programs and/or devices designed in a way such that they cannot accommodate RSA moduli larger than 1024 bits whereas many applications are now requiring at least 2048-bit RSA moduli. Using the folklore method, a 2048-bit RSA modulus can be represented in compact form with 1024 bits *plus* the seed needed to recover the predetermined portion and so will not fit in a 1024-bit memory buffer. If now the two thirds can be predetermined, this is possible since only 683 bits plus the seed are needed to represent a 2048-bit RSA modulus. More generally, achieving a higher compression rate is always useful because this leads to more savings in both storage and bandwidth.

The rest of this paper is organized as follows. In the next section, we provide some necessary definitions and notation. We also review the folklore method for generating RSA moduli with a predetermined portion. In Section 3 and 4, we improve on it and present relevant applications. Section 3 describes an efficient escrow mechanism for RSA keys well suited for constrained tamper-resistant devices like smart cards. Section 4 presents highly compact representations of RSA keys which greatly reduce the storage and transmission requirements. Finally, we conclude in Section 5.

2 Preliminaries and Notation

We first introduce some notation. The concatenation of two bit strings X_0 and X_1 is denoted by $X_0 \| X_1$. To ease the presentation, we do not make the distinction between an integer and its representation. For an integer X, we denote by $|X|_2$ the bit length of X. By an ℓ-bit integer, we mean an integer X such that $2^{\ell-1} \leqslant X < 2^{\ell}$, that is, $|X|_2 = \ell$.

Throughout this paper and unless otherwise specified, we consider a n-bit RSA modulus $N = p\,q$ which is the product of two large primes where p is a $(n - n_0)$-bit prime and q is a n_0-bit prime, for some $1 < n_0 < n$. Without loss of generality, we assume that $|p|_2 \leqslant |q|_2$, or equivalently, that $2n_0 \geqslant n$.

2.1 RSA primitive

For a public exponent e with $\gcd(e, \lambda(N)) = 1$, the corresponding private exponent d satisfies the relation

$$e\,d \equiv 1 \pmod{\lambda(N)},$$

where λ is Carmichael's function. For $N = p\,q$, we have $\lambda(N) = \mathrm{lcm}(p-1, q-1)$. Given $x < N$, the public operation (e.g., message encryption or signature verification) consists in raising x to the e-th power modulo N, i.e., in computing $y = x^e \bmod N$. Then, given y, the corresponding private operation (e.g., decryption of a ciphertext or signature generation) consists in computing $y^d \bmod N$. From the definition of e and d, it readily follows that $y^d \equiv x \pmod{N}$. The private operation can also be carried out at higher speed through Chinese remaindering (CRT mode). Computations are then independently performed modulo p and q and then recombined.

To sum up, a *n-bit RSA modulus* $N = p\,q$ is the product of two large prime integers p and q such that $|p|_2 = n - n_0$, $|q|_2 = n_0$, and $\gcd(p-1, e) = \gcd(q-1, e) = 1$. For security reasons, so-called 'balanced' RSA moduli are generally preferred, which means $n = 2n_0$.

2.2 Folklore method

We review a simple yet efficient method for fixing the leading bits of N. The goal is to construct an RSA modulus N of the form $N = N_H \| N_L$ and where the leading portion, N_H, is predetermined.

Letting t the bit length of N_H, we can write

$$N = N_H\, 2^{n-t} + N_L \quad \text{for some } 0 < N_L < 2^{n-t} \ . \tag{1}$$

We follow the presentation of [18]. For some integer $t' \geqslant t$, pick at random a $(n-t')$-bit prime p such that $\gcd(p-1, e) = 1$. Define $N' = N_H\, 2^{n-t}$ and upper round it to the nearest multiple of p to get $q' = \left\lceil \frac{N'}{p} \right\rceil$. Find the smallest nonnegative integer m such that $q = q' + m$ is prime and $\gcd(q-1, e) = 1$. If $N = p\,q$ satisfies Eq. (1) then return $\{N_L, p, q\}$; otherwise re-iterate the process.

We note that several variations of the previous method can be found in [18].

3 Escrowing RSA Keys

Key escrow allows one to get access to the decryption keys. This can be helpful in certain situations. For example, if an employee leaves her company without returning her private key, a key-escrow mechanism enables to ensure that data intended to this employee is not lost. A key-escrow mechanism can also be used by a manager to read all data of employees within her organization.

In the case of an RSA modulus $N = p\,q$, the knowledge of about half the bits of p suffices to recover the private key using lattice reduction techniques [6] (see also [5, 7, 8]). Therefore if about half the bits of p are encrypted under some secret key K and embedded in the representation of public RSA modulus N then, from N, an 'authority' knowing key K can reconstruct p and thus compute the corresponding private RSA key. Alternative techniques are described in [9].

Remark 1. When RSA key generation is performed on-board in smart cards, it is desirable that secret key K is not shared by all smart cards. It is much better that each employee Id — where Id is a unique identifier (e.g., email address, badge number, ...) — has a different secret key, say K_{Id}, embedded in the tamper-resistant memory of her smart card. To facilitate the key management, the key-escrow authority may hold a master secret key K from which the employees' keys, K_{Id}'s, can be derived.

The folklore method can be adapted to this end. We present a solution in Algorithm 1. It requires a secure length-preserving symmetric cipher $\mathcal{E}$ that on input a plaintext x and a key K_{Id} returns the ciphertext $c = \mathcal{E}_{\mathsf{K}_{\mathsf{Id}}}(x)$.

Algorithm 1. Given key lengths n, n_0 and public exponent e, this algorithm outputs an *escrowed* n-bit RSA modulus $N = p\,q$ with $|p|_2 = n - n_0$ and $|q|_2 = n_0$, and private exponent d.

1. [Prime p] Generate a random prime $p \in [2^{n-n_0-1} + 1, 2^{n-n_0} - 1]$ such that $\gcd(p-1, e) = 1$.
2. [SETUP] Let p_h denote the κ high-order bits of p,

$$p_h = \left\lfloor \frac{p}{2^{n-n_0-\kappa}} \right\rfloor \quad \text{with } \kappa = \lceil (n-1)/4 \rceil \ .$$

   Define

$$N_H = 1 \| \mathcal{E}_{\mathsf{K_{Id}}}(p_h) \quad \text{and} \quad \kappa' = |N_H|_2 = \kappa + 1 \ .$$

3. [Prime q] Generate a random prime

$$q \in \left[\left\lfloor \frac{2^{n-\kappa'} N_H}{p} \right\rfloor + 1, \left\lfloor \frac{2^{n-\kappa'}(N_H+1) - 1}{p} \right\rfloor \right] \tag{2}$$

   such that $\gcd(q-1, e) = 1$.
4. [Output] Return $N = p\,q$ and $d = e^{-1} \bmod (p-1)(q-1)$.

3.1 Analysis

In contrast with the folklore method, the two primes in the RSA key generation of Algorithm 1 lie in a prescribed interval. Their generation can therefore fully benefit from the fast prime generation techniques in [14] (see also [15]).

Suppose we have to generate a prime $q \in [q_{\min}, q_{\max}]$. Basically, define Π as the product of many primes so that $\Pi \leqslant q_{\max} - q_{\min}$ and $\phi(\Pi)/\Pi$ is as small as possible (and thus contains a maximum number of small primes).[1,2] Define also an element $a \in (\mathbb{Z}/\Pi\mathbb{Z})^*$. Next, for a random element $k \in (\mathbb{Z}/\Pi\mathbb{Z})^*$ and a random $T \in [q_{\min}, q_{\max} + 1 - \Pi]$, find the smallest nonnegative integer i such that

$$q = [(a^i\, k - T) \bmod \Pi] + T \tag{3}$$

is prime. This method presents the advantage that all candidates tested for primality are by construction already coprime to Π: $q \equiv a^i\, k \pmod{\Pi} \in (\mathbb{Z}/\Pi\mathbb{Z})^*$. The expected number of candidates to be tried heuristically amounts to

$$n_0 \ln 2 \, \frac{\phi(\Pi)}{\Pi} \tag{4}$$

where $n_0 = |q|_2$ (cf. [14, Section 2.3]).

The next proposition shows that the interval $[q_{\min}, q_{\max}]$ in Step 3 of Algorithm 1 is optimal w.r.t. the above prime generation (Eq. (3)).

[1] ϕ denotes Euler's totient function; $\phi(\Pi) = \#(\mathbb{Z}/\Pi\mathbb{Z})^*$.

[2] In smart card implementations, parameter Π is predetermined before compile time and hard-coded in the prime generation routine. As a consequence, since the values of $q_{\min}$ and $q_{\max}$ depend on p and so change at each execution, parameter Π should be chosen as the largest possible value so that the relation $\Pi \leqslant q_{\max} - q_{\min}$ will always be satisfied.

Proposition 1. *The integer interval q is chosen from is maximal and contains (at least) $2^{n_0-\kappa'}$ elements.*

Proof. Define $q_{\min} = \left\lfloor \frac{2^{n-\kappa'} N_H}{p} \right\rfloor + 1$ and $q_{\max} = \left\lfloor \frac{2^{n-\kappa'}(N_H+1)-1}{p} \right\rfloor$. We have

$$\begin{aligned} p\, q_{\min} &= p \left\lfloor \frac{2^{n-\kappa'} N_H}{p} \right\rfloor + p \\ &= 2^{n-\kappa'} N_H + p - (2^{n-\kappa'} N_H \bmod p) = N_H \| N_L^{(\min)} \end{aligned}$$

with $1 \leqslant N_L^{(\min)} = p - (2^{n-\kappa'} N_H \bmod p) \leqslant p$. Therefore, since the least significant bit of N_L must be 1, we see that $p\,(q_{\min} - 1)$ cannot be of the required form. Likewise, we have

$$\begin{aligned} p\, q_{\max} &= p \left\lfloor \frac{2^{n-\kappa'}(N_H+1)-1}{p} \right\rfloor \\ &= 2^{n-\kappa'} N_H + 2^{n-\kappa'} - 1 - \left([2^{n-\kappa'}(N_H+1)-1] \bmod p\right) = N_H \| N_L^{(\max)} \end{aligned}$$

with $2^{n-\kappa'} - p \leqslant N_L^{(\max)} = 2^{n-\kappa'} - 1 - \left([2^{n-\kappa'}(N_H+1)-1] \bmod p\right) \leqslant 2^{n-\kappa'} - 1$, and $p\,(q_{\max} + 1)$ cannot be of the required form. Consequently, the interval is maximal.

For the second part of the proposition, we have

$$q_{\max} = \left\lfloor \frac{2^{n-\kappa'}(N_H+1)-1}{p} \right\rfloor \geqslant \left\lfloor \frac{2^{n-\kappa'} N_H}{p} \right\rfloor + \left\lfloor \frac{2^{n-\kappa'}-1}{p} \right\rfloor .$$

So, we get

$$q_{\max} - q_{\min} \geqslant \left\lfloor \frac{2^{n-\kappa'}-1}{p} \right\rfloor - 1 \geqslant \left\lfloor \frac{2^{n-\kappa'}-1}{2^{n-n_0}-1} \right\rfloor - 1 \geqslant 2^{n_0-\kappa'} - 1$$

since $n - \kappa' = n - (1 + \lceil (n-1)/4 \rceil) \geqslant n - (1 + \lceil (2n_0 - 1)/4 \rceil) \geqslant n - n_0$, which concludes the proof. □

The worst case for the generation of q appears for balanced RSA moduli (i.e., $n = 2n_0$), in which case the length of Π is halved compared to the regular prime generation (i.e., without key escrow). Hence, from Eq. (4) and using the figures given in [15, Fig. 7], we get:

Table 1. Heuristic expected number of primality tests to generate a balanced n-bit RSA moduli with and without key escrow.

n	1024	1536	2048
With key escrow	70.73	99.14	126.56
Without key escrow	66.58	93.80	119.96

From this, we conclude that embedding the key-escrow mechanism described in this section has little impact on the overall performance of the RSA key generation. We refer the reader to [18] for security considerations.

3.2 Recovering the private key

It remains to explain how to recover a private RSA key corresponding to a given RSA modulus. The technique is based on a powerful result making use of the LLL reduction algorithm [19].

Theorem 1 (Coppersmith). *Let $\mathscr{P}(x, y)$ be an irreducible polynomial in two variables over $\mathbb{Z}$, of maximum degree δ in each variable separately. Let X, Y be bounds on the desired solutions x_0, y_0. Define $\widetilde{\mathscr{P}}(x, y) = \mathscr{P}(xX, yY)$ and let W be the absolute value of the largest coefficient of $\widetilde{\mathscr{P}}$. If*

$$X \cdot Y \leqslant W^{2/(3\delta)}$$

then in time polynomial in $(\log W, 2^{\delta})$, we can find all integer pairs (x_0, y_0) with $\mathscr{P}(x_0, y_0) = 0$, $|x_0| < X$, and $|y_0| < Y$.

Proof. See [6, Corollary 2]. □

Let N_H denote the κ' leading bits of n-bit RSA modulus N and let N'_H denote N_H without its most significant bit: $N'_H = N_H \bmod 2^{\kappa'-1}$. Using corresponding secret key $\mathsf{K_{Id}}$, the key-escrow authority can recover $p_h = {\mathcal{E}_{\mathsf{K_{Id}}}}^{-1}(N'_H)$. For completeness, we show below that if $|p_h|_2 \geqslant \lceil (n-1)/4 \rceil$ then the knowledge of p_h suffices to recover the whole value of p (and thus the corresponding private key). This is an application of Coppersmith's theorem.

Write

$$p = \bar{p} + x_0 \quad \text{and} \quad q = \bar{q} + y_0$$

for some unknown integers x_0 and y_0, and where $\bar{p}$ and $\bar{q}$ are defined by

$$\bar{p} = 2^{n-n_0-\kappa-1}(2p_h + 1) \quad \text{and} \quad \bar{q} = \left\lfloor \frac{N}{\bar{p}} \right\rfloor .$$

It is easily verified that respective bounds X and Y on $|x_0|$ and $|y_0|$ are given by

$$|x_0| < 2^{n-n_0-\kappa-1} = X \quad \text{and} \quad |y_0| < 2^{n_0-\kappa} = Y .$$

Now, define the bivariate polynomial

$$\mathscr{P}(x, y) = (\bar{p} + x)(\bar{q} + y) - N = \bar{q}\, x + \bar{p}\, y + x\, y + (\bar{p}\, \bar{q} - N) ,$$

an integer solution of which is (x_0, y_0): $\mathscr{P}(x_0, y_0) = 0$. Corresponding polynomial $\widetilde{\mathscr{P}}$ is given by $\widetilde{\mathscr{P}}(x, y) = \bar{q}X\, x + \bar{p}\, Y\, y + XY\, x\, y + (\bar{p}\, \bar{q} - N)$. Hence, it follows that

$$\begin{aligned} W &:= \max\{\bar{q}X, \bar{p}\, Y, XY, |\bar{p}\, \bar{q} - N|\} = \bar{p}\, Y = 2^{n-2\kappa-1}(2p_h + 1) \\ &\geqslant 2^{n-2\kappa-1}(2^{\kappa} + 1) > 2^{n-\kappa-1} . \end{aligned}$$

Noting that $n - 1 \leqslant 4\kappa$, this yields

$$X \cdot Y = 2^{n-2\kappa-1} \leqslant (2^{n-\kappa-1})^{2/3} < W^{2/3} .$$

Consequently, the conditions of Theorem 1 are satisfied and so the key-escrow authority can recover (x_0, y_0) and thus the two secret factors of RSA modulus N, $p = \bar{p} + x_0$ and $q = \bar{q} + y_0$. The private RSA exponent is then recovered as $d = e^{-1} \bmod (p-1)(q-1)$.

3.3 Variants

Analogously to [33, 18], it is possible to fix the trailing bits of modulus N rather than the leading bits. The SETUP phase then defines $N_L = \mathcal{E}_{\mathsf{K_{ld}}}(p_h)\|1$ and prime q is generated as $q = C + q'\,2^{\kappa'}$ with $C = N_L/p \bmod 2^{\kappa'}$ for some random q' in

$$\left[\left\lceil\frac{2^{n-1}+1-Cp}{2^{\kappa'}p}\right\rceil, \left\lfloor\frac{2^n-Cp}{2^{\kappa'}p}\right\rfloor\right] .$$

More generally, it is possible to fix some leading bits and some trailing bits of N.

The proposed method can also be adapted to support RSA moduli that are made of more than two factors, for example, 3-prime RSA moduli or RSA moduli of the form $N = p^r\,q$ [31].

4 Compressing RSA Keys

Compressing RSA moduli leads to substantial reductions in storage and transmission requirements. This is even more true as standard bodies and organizations are pushing for increasingly longer RSA keys. In particular, storage requirements can be critical for constrained devices for which RSA moduli are typically stored in EEPROM-like memory, which is expensive. Halving the representation of a 2048-bit modulus already frees 128 bytes in memory. In this section, we will present a method that enables to compress a 2048-bit on only 86 bytes.

As exemplified in the introduction, the techniques of this section can also used to reduce the computational requirements.

We require a mask generating function (MGF). A practical implementation can be found in [1, Appendix A]. The MGF takes on input a seed s_0 and expands it into a binary string of κ' bits. Moreover, we force the leading to 1 so as to obtain a κ'-bit integer, $N_H = 2^{\kappa'-1} \vee \mathtt{MGF}(s_0)$. Primes p and q are then generated so that the κ' leading bits of $N = p\,q$ represent N_H. The compressed RSA modulus is given by the $(n-\kappa')$ bits of N, N_L. If seed s_0 is not public (or cannot be publicly recovered), it should be returned together with N_L.

Here is the detailed algorithm.

Algorithm 2. Given key lengths n, n_0, public exponent e and compression parameter κ', this algorithm outputs (the representation of) a *compressed* n-bit RSA modulus $N = p\,q$ with $|p|_2 = n - n_0$ and $|q|_2 = n_0$, $\widetilde{N}$, and private exponent d.

1. [Fixing N_H] Produce a κ'-bit integer N_H from a seed s_0:

$$N_H := 2^{\kappa'-1} \vee \mathtt{MGF}(s_0) \in [2^{\kappa'-1}, 2^{\kappa'}-1] .$$

2. [Primes p and q] Generate random primes p and q with $\gcd(p-1,e) = \gcd(q-1,e) = 1$, $|p|_2 = n - n_0$, $|q|_2 = n_0$, and such that

$$p\,q = N_H\|N_L \quad \text{for some } 1 \leqslant N_L < 2^{n-\kappa'} .$$

3. [Output] Return $\widetilde{N} = \{N_L\ [, s_0]\}$ and $d = e^{-1} \bmod (p-1)(q-1)$.

Given the representation $\widetilde{N} = \{N_L\ [, s_0]\}$, anyone can recover the corresponding n-bit RSA modulus N as $N_H \| N_L$ with $N_H = 2^{\kappa'-1} \vee \mathtt{MGF}(s_0)$. We note that decompressing $\widetilde{N}$ to recover N is very fast.

Remark 2. Clearly, Algorithm 1 can be used to generate primes p and q in the previous algorithm. From Eq. (2), we see however that if $2^{n-\kappa'} \leqslant p$ then the interval q is chosen from may be empty and thus κ' should be at most n_0. Therefore, the method of Algorithm 1 can compress at best n-bit RSA moduli up to n_0 bits. For balanced RSA moduli (i.e., $n = 2n_0$), this yields a compression rate of $1/2$.

4.1 Beyond the 1/2 compression rate

We consider now higher compression rates, that is, $\kappa' \geqslant n_0 + 1$. From the description of Algorithm 2, we have $N = p\,q = N_H \| N_L = N_H\, 2^{n-\kappa'} + N_L$, which implies that $N_L = -N_H\, 2^{n-\kappa'} \bmod p$ since $N_L < 2^{n-\kappa'} \leqslant 2^{n-n_0-1} < p$.

Consequently, achieving higher compression rates translates into the problem of finding an $(n-n_0)$-bit prime p such that

$$(-N_H\, 2^{n-\kappa'} \bmod p) < 2^{n-\kappa'}\ .$$

Indeed, letting

$$q = \left\lfloor \frac{N_H\, 2^{n-\kappa'}}{p} \right\rfloor + 1$$

and, provided that it is prime, we obtain

$$p\,q = N_H\, 2^{n-\kappa'} + \underbrace{p - (N_H\, 2^{n-\kappa'} \bmod p)}_{=N_L < 2^{n-\kappa'}}$$

as required.

The following algorithm derived from Okamoto-Shiraishi's paper [23] (see also [13]) seeks a solution to the above problem.

Algorithm 3. Given key lengths n, n_0, compression parameter κ' and κ'-bit predetermined portion N_H, this algorithm outputs the lower portion N_L of n-bit RSA modulus $N = N_H \| N_L = p\,q$ with $|p|_2 = n - n_0$ and $|q|_2 = n_0$.

1. [Initialization] Choose a random $(n-n_0)$-bit integer p_0 and define

$$q_0 = \left\lfloor \frac{N_H\, 2^{n-\kappa'}}{p_0} \right\rfloor\ .$$

2. [Euclidean step] Form the list $\mathcal{L}$ of pairs (x_i, y_i) verifying

$$|z_i - x_i\, y_i| < 2^{n-\kappa'-1} \tag{5}$$

where (x_i, y_i, z_i) are defined as

$$\begin{cases} (z_0, x_0, y_0) = \big((N_H 2^{n-\kappa'} \bmod p_0) + 2^{n-\kappa'-1},\ 0,\ 0\big) \\ (z_i, x_i, y_i) = \Big(z_{i-1} \bmod d_i,\ x_{i-1} + \Big\lfloor \frac{z_{i-1}}{d_i} \Big\rfloor u_i,\ y_{i-1} + \Big\lfloor \frac{z_{i-1}}{d_i} \Big\rfloor v_i\Big) \end{cases}$$

and

$$\begin{cases} (d_0, u_0, v_0) = \big(p_0,\ 0,\ 1\big),\ \ (d_{-1}, u_{-1}, v_{-1}) = \big(q_0,\ 1,\ 0\big) \\ (d_i, u_i, v_i) = \Big(d_{i-2} \bmod d_{i-1},\ u_{i-2} - \Big\lfloor \frac{d_{i-2}}{d_{i-1}} \Big\rfloor u_{i-1},\ v_{i-2} - \Big\lfloor \frac{d_{i-2}}{d_{i-1}} \Big\rfloor v_{i-1}\Big) \end{cases} .$$

3. [Finished?] Find a pair $(x_i, y_i) \in \mathcal{L}$ such that $p = p_0 + x_i$ and $q = q_0 + y_i$ are prime. If no such pair is found, go to Step 1.
4. [Output] Compute $N = (p_0 + x_i)(q_0 + y_i)$ and return $N_L = N \bmod 2^{n-\kappa'}$.

The next proposition shows that a solution returned by the algorithm is of the correct form.

Proposition 2. *Using the notations of Algorithm 2 and provided that* $|z_i - x_i\, y_i| < 2^{n-\kappa'-1}$*, any pair* (x_i, y_i) *verifies*

$$(p_0 + x_i)(q_0 + y_i) = N_H\, 2^{n-\kappa'} + N_L$$

for some $1 \leqslant N_L < 2^{n-\kappa'}$.

Proof. Let $N_L := 2^{n-\kappa'-1} + x_i\, y_i - z_i$. Since at Step 4, $(x_i, y_i) \in \mathcal{L}$, it follows that (z_i, x_i, y_i) satisfies Eq. (5) and thus $1 \leqslant N_L \leqslant 2^{n-\kappa'} - 1$, as desired.

From this definition of N_L, we also get $z_i = x_i\, y_i + 2^{n-\kappa'-1} - N_L$, which in turn implies

$$\begin{aligned} q_0\, x_i + p_0\, y_i + z_i &= q_0\, x_i + p_0\, y_i + x_i\, y_i + 2^{n-\kappa'-1} - N_L \\ &= (p_0 + x_i)(q_0 + y_i) - p_0\, q_0 + 2^{n-\kappa'-1} - N_L \\ &= N_H\, 2^{n-\kappa'} - p_0\, q_0 + 2^{n-\kappa'-1} \\ &= (N_H\, 2^{n-\kappa'} \bmod p_0) + 2^{n-\kappa'-1} \qquad (\dagger) \end{aligned}$$

by noting that $p_0\, q_0 = p_0 \lfloor N_H\, 2^{n-\kappa'} / p_0 \rfloor = N_H\, 2^{n-\kappa'} - (N_H\, 2^{n-\kappa'} \bmod p_0)$. Moreover, any tuple of (z_i, x_i, y_i) verifies the property

$$q_0\, x_i + p_0\, y_i + z_i = z_0 \ .$$

This easily follows by construction:

$$\begin{aligned} q_0\, x_i + p_0\, y_i + z_i &= q_0\, x_{i-1} + p_0\, y_{i-1} + \lfloor \tfrac{z_{i-1}}{d_i} \rfloor (q_0\, u_i + p_0\, v_i) + z_i \\ &= q_0\, x_{i-1} + p_0\, y_{i-1} + \lfloor \tfrac{z_{i-1}}{d_i} \rfloor d_i + z_i \\ &= q_0\, x_{i-1} + p_0\, y_{i-1} + z_{i-1} = z_0\,. \end{aligned}$$

Comparing with (†), we see that $\big((N_H\, 2^{n-\kappa'} \bmod p_0) + 2^{n-\kappa'-1}, 0, 0\big)$ is a valid choice for (z_0, x_0, y_0). □

4.2 Analysis

Clearly, the sequences $\{u_i\}$, $\{v_i\}$ and $\{d_i\}$ defined in Algorithm 3 are those given by extended Euclid's algorithm: $q_0\, u_i + p_0\, v_i = d_i$. The sequence $|z_i - x_i\, y_i|$ is decreasing and then increasing. This gives a condition break in the construction of list $\mathcal{L}$ in Algorithm 3. For small compression parameters κ', the algorithm finds many pairs $(x_i, y_i) \in \mathcal{L}$. When κ' increases, the number of pairs decreases.

In the balanced case (that is, the worst case), we observed that list $\mathcal{L}$ is always nonempty for $\kappa' \lessapprox 2n/3$. We conducted numerous experiments to assess this. For each tested predetermined portion N_H, a pair of matching primes p and q was found. We give below one such pair of primes (p, q) when N_H was set as the 1360 leading bits of the RSA-2048 challenge [27].

$$
\begin{aligned}
\text{RSA2048} &= \texttt{c7970ceedcc3b075 4490201a7aa613cd 73911081c790f5f1 a8726f463550bb5b}\\
&\quad \texttt{7ff0db8e1ea1189e c72f93d1650011bd 721aeeacc2acde32 a04107f0648c2813}\\
&\quad \texttt{a31f5b0b7765ff8b 44b4b6ffc93384b6 46eb09c7cf5e8592 d40ea33c80039f35}\\
&\quad \texttt{b4f14a04b51f7bfd 781be4d1673164ba 8eb991c2c4d730bb be35f592bdef524a}\\
&\quad \texttt{f7e8daefd26c66fc 02c479af89d64d37 3f442709439de66c eb955f3ea37d5159}\\
&\quad \texttt{f6135809f85334b5 cb1813addc80cd05 609f10ac6a95ad65 872c909525bdad32}\\
&\quad \texttt{bc729592642920f2 4c61dc5b3c3b7923 e56b16a4d9d373d8 721f24a3fc0f1b31}\\
&\quad \texttt{31f55615172866bc cc30f95054c824e7 33a5eb6817f7bc16 399d48c6361cc7e5}\\
p &= \texttt{f2cbf408c0712b00 bb40d1ff5ef0d42b 981ba43a174da647 a474918aea017483}\\
&\quad \texttt{e8406e140d522a09 da65cb960a912c3f 5bf031af675b7907 5f5eb2151ad9c0ff}\\
&\quad \texttt{7bd518dd0f01bdc2 ac68f8b8edd30426 7b58d3317ab47072 4a04313d85be7d88}\\
&\quad \texttt{f63fe405a7628b12 3b34217ca4d45e42 97b5d728c2f74cd9 7fa673e6483804f5}\\
q &= \texttt{d2719ab388ad9e05 e9f46df9ce8c822e de61d36a7ce61b6c a4d3b4a3b41a8f42}\\
&\quad \texttt{0935eec7ce1a57c1 2e15bfaf4873e2d2 095297c6fd8d49d9 8ef44b955a983ba9}\\
&\quad \texttt{75f9f1be4f1730fb 2e9834e075988ed0 9229cb1514998172 7c1b58d2e20932d5}\\
&\quad \texttt{25a06de1111311af 5a88ae25f3e27e3d c2c44e9b51ffae50 2ed18dd903907931}\\
p \cdot q &= \underline{\texttt{c7970ceedcc3b075}}\ \underline{\texttt{4490201a7aa613cd}}\ \underline{\texttt{73911081c790f5f1}}\ \underline{\texttt{a8726f463550bb5b}}\\
&\quad \underline{\texttt{7ff0db8e1ea1189e}}\ \underline{\texttt{c72f93d1650011bd}}\ \underline{\texttt{721aeeacc2acde32}}\ \underline{\texttt{a04107f0648c2813}}\\
&\quad \underline{\texttt{a31f5b0b7765ff8b}}\ \underline{\texttt{44b4b6ffc93384b6}}\ \underline{\texttt{46eb09c7cf5e8592}}\ \underline{\texttt{d40ea33c80039f35}}\\
&\quad \underline{\texttt{b4f14a04b51f7bfd}}\ \underline{\texttt{781be4d1673164ba}}\ \underline{\texttt{8eb991c2c4d730bb}}\ \underline{\texttt{be35f592bdef524a}}\\
&\quad \underline{\texttt{f7e8daefd26c66fc}}\ \underline{\texttt{02c479af89d64d37}}\ \underline{\texttt{3f442709439de66c}}\ \underline{\texttt{eb955f3ea37d5159}}\\
&\quad \underline{\texttt{f6135809f85334b5}}\ \underline{\texttt{cb18}}\texttt{fd5b056dbd78 01de0bb2fd1e7b5a b4d33205ac4c9a71}\\
&\quad \texttt{800cbfe76ac1424a 90121c232d945edc 6e7d34038e271b4c 92b620e5ddf836da}\\
&\quad \texttt{62eede67b33ab2b4 ae58d531cf27f090 6034201bf753711e 24417374f5e0bfe5}
\end{aligned}
$$

4.3 Variants

As in Section 3, the methods of this section are subject to numerous variants. For example, it is possible to fix the trailing bits of N, or some leading bits and some trailing bits of N.

To achieve higher compression rates, multi-prime RSA moduli can be used (as suggested in [32]) or RSA moduli of the form $p^r q$ [31]. Unbalanced RSA moduli (i.e., $n \lneqq 2n_0$) always offer better compression rates.

5 Conclusions

We have presented enhancements to the folklore method for generating RSA moduli with a predetermined portion and pointed out relevant applications. We have shown how to efficiently implement a key escrow mechanism. Special care was taken to make it compatible with the fastest smart-card prime generation algorithms and to optimally benefit from them to only mildly affect the global performance. We have also presented a compression technique that enables to represent RSA moduli with about three times fewer bits.

Acknowledgments I thank Igor Shparlinski for sending a copy of [29], and Alain Durand and Mohamed Karroumi for stimulating discussions. I also thank the anonymous referees for useful comments.

References

1. Mihir Bellare and Phillip Rogaway. The exact security of digital signatures. In *Advances in Cryptology – EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 399–416. Springer, 1996.
2. Daniel J. Bernstein. Stop overestimating RSA bandwidth! Rump session, CRYPTO 2004, Santa Barbara, CA, USA, August 17, 2004. Slides available at URL `http://cr.yp.to/talks/2004.08.17/slides.pdf`.
3. Daniel J. Bernstein. Compressing RSA/Rabin keys. Invited talk, Number Theory Inspired By Cryptography (NTIBC 2005), Bannf Centre, Alberta, Canada, November 6, 2005. Slides available at URL `http://cr.yp.to/talks/2005.11.06/slides.pdf`.
4. Dan Boneh. Twenty years of attacks on the RSA cryptosystem. *Notices of the American Mathematical Society (AMS)*, 46(2):203–213, 1999.
5. Don Coppersmith. Finding a small root of a bivariate integer equation; factoring with high bits known. In *Advances in Cryptology – EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 155–165. Springer, 1996.
6. Don Coppersmith. Small solutions to polynomial equations, and low exponent RSA vulnerabilities. *Journal of Cryptology*, 10(4):233–260, 1997.
7. Jean-Sébastien Coron. Finding small roots of bivariate integer polynomial equations revisited. In *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 492–505. Springer, 2004.
8. Jean-Sébastien Coron. Finding small roots of bivariate integer polynomial equations: A direct approach. In *Advances in Cryptology – CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 379–394. Springer, 2007.
9. Claude Crépeau and Alain Slakmon. Simple backdoors for RSA key generation. In *Topics in Cryptology – CT-RSA 2003*, volume 2612 of *Lecture Notes in Computer Science*, pages 403–416. Springer, 2003.

10. Yvo Desmedt. Abuses in cryptography and how to fight them. In *Advances in Cryptology – CRYPTO '88*, volume 403 of *Lecture Notes in Computer Science*, pages 375–389. Springer, 1990.
11. Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.
12. Christian Gehrmann and Mats Näslund, editors. ECRYPT yearly report on algorithms and keysizes. ECRYPT Report, D.SPA.16, Revision 1.0, January 2006. Available at URL `http://www.ecrypt.eu.org/documents/D.SPA.16-1.0.pdf`.
13. Marc Girault and Jean-François Misarski. Selective forgery of RSA signatures using redundancy. In *Advances in Cryptology – EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 495–507. Springer, 1997.
14. Marc Joye and Pascal Paillier. Fast generation of prime numbers on portable devices: An update. In *Cryptographic Hardware and Embedded Systems – CHES 2006*, volume 4249 of *Lecture Notes in Computer Science*, pages 160–173. Springer, 2006.
15. Marc Joye, Pascal Paillier, and Serge Vaudenay. Efficient generation of prime numbers. In *Cryptographic Hardware and Embedded Systems – CHES 2000*, volume 1965 of *Lecture Notes in Computer Science*, pages 340–354. Springer, 2000.
16. Ari Juels. Provable security: Some caveats. Panel presentation, 6th ACM Conference on Computer and Communications Security (ACM CCS '99), Singapore, November 1-4, 1999.
17. Hans-Joachim Knobloch. A smart card implementation of the Fiat-Shamir identification scheme. In *Advances in Cryptology – EUROCRYPT '88*, volume 330 of *Lecture Notes in Computer Science*, pages 87–95. Springer, 1988.
18. Arjen K. Lenstra. Generating RSA moduli with a predetermined portion. In *Advances in Cryptology – ASIACRYPT '98*, volume 1514 of *Lecture Notes in Computer Science*, pages 1–10. Springer, 1998.
19. Arjen K. Lenstra, Hendrik W. Lenstra, Jr., and László Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:515–534, 1982.
20. Gisela Meister. On an implementation of the Mohan-Adiga algorithm. In *Advances in Cryptology – EUROCRYPT '90*, volume 473 of *Lecture Notes in Computer Science*, pages 496–500. Springer, 1991.
21. S.B. Mohan and B.S. Adiga. Fast algorithms for implementing RSA public key cryptosystems. *Electronics Letters*, 21(7):761, 1985.
22. David Naccache, David M'Raïhi, Serge Vaudenay, and Dan Raphaeli. Can D.S.A. be improved? Complexity trade-offs with the digital signature standard. In *Advances in Cryptology – EUROCRYPT '94*, volume 950 of *Lecture Notes in Computer Science*, pages 77–85. Springer, 1995.
23. Tatsuaki Okamoto and Akira Shiraishi. A fast signature scheme based on quadratic inequalities. In *1985 IEEE Symposium on Security and Privacy*, pages 123–133. IEEE Press, 1985.
24. Glenn Orton, Lloyd Peppard, and Stafford Tavares. A design of a fast pipelined modular multiplier based on a diminished-radix algorithm. *Journal of Cryptology*, 6(4):183–208, 1993.
25. Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology – EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 1999.
26. Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

27. RSA Laboratories. The RSA challenge numbers. URL: `http://www.rsa.com/rsalabs/node.asp?id=2093`.
28. RSA Laboratories. RSA-200 is factored! URL: `http://www.rsa.com/rsalabs/node.asp?id=2879`, May 2005.
29. Igor E. Shparlinski. On RSA moduli with prescribed bit patterns. *Designs, Codes and Cryptography*, 39(1):113–122, 2006.
30. Gustavus J. Simmons. The subliminal channel and digital signatures. In *Advances in Cryptology – EUROCRYPT '84*, volume 209 of *Lecture Notes in Computer Science*, pages 364–368. Springer, 1985.
31. Tsuyoshi Takagi. Fast RSA-type cryptosystem modulo $p^k q$. In *Advances in Cryptology – CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 318–326. Springer, 1998.
32. Scott A. Vanstone and Robert J. Zuccherato. Using four-prime RSA in which some of the bits are specified. *Electronics Letters*, 30(25):2118–2119, 1994.
33. Scott A. Vanstone and Robert J. Zuccherato. Short RSA keys and their generation. *Journal of Cryptology*, 8(2):101–114, 1995.
34. Colin D. Walter. Faster modular multiplication by operand scaling. In *Advances in Cryptology – CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 313–323. Springer, 1992.
35. Adam Young and Moti Yung. The dark side of "black-box" cryptography, or: Should we trust Capstone? In *Advances in Cryptology – CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 89–103. Springer, 1996.
36. Adam Young and Moti Yung. Kleptography: Using cryptography against cryptography. In *Advances in Cryptology – EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 62–74. Springer, 1997.