DANIEL LOEBENBERGER & MICHAEL NÜSKEN (2011). Analyzing standards for RSA integers. In AFRICACRYPT *2011*, ABDERRAHMANE NITAJ & DAVID POINTCHEVAL, editors, volume 6737 of *Lecture Notes in Computer Science*, 260–277. Springer. ISBN 978-3-642-21968-9. ISSN 0302-9743. URL https://dx.doi.org/10.1007/978-3-642-21969-6_16.

Analyzing standards for RSA integers

Version 26 April 2011

Daniel Loebenberger and Michael Nüsken

b-it, University of Bonn
{daniel,nuesken}@bit.uni-bonn.de

Abstract. The key-generation algorithm for the RSA cryptosystem is specified in several standards, such as PKCS#1, IEEE 1363-2000, FIPS 186-3, ANSI X9.44, or ISO/IEC 18033-2. All of them substantially differ in their requirements. This indicates that for computing a "secure" RSA modulus it does not matter how exactly one generates RSA integers. In this work we show that this is indeed the case to a large extend: First, we give a theoretical framework that will enable us to easily compute the entropy of the output distribution of the considered standards and show that it is comparatively high. To do so, we compute for each standard the number of integers they define (up to an error of very small order) and discuss different methods of generating integers of a specific form. Second, we show that factoring such integers is hard, provided factoring a product of two primes of similar size is hard.
Keywords: RSA integer, output entropy, reduction. ANSI X9.44, FIPS 186-3, IEEE 1363-2000, ISO/IEC 18033-2, NESSIE, PKCS#1.

1 Introduction

An *RSA integer* is an integer that is suitable as a modulus for the RSA cryptosystem as proposed by Rivest, Shamir & Adleman (1977, 1978):

> "You first compute n as the product of two primes p and q:
>
> $$n = p \cdot q.$$
>
> These primes are very large, 'random' primes. Although you will make n public, the factors p and q will be effectively hidden from everyone else due to the enormous difficulty of factoring n."

Also in earlier literature such as Ellis (1970) or Cocks (1973) one does not find any further restrictions. In subsequent literature people define RSA integers similarly to Rivest, Shamir & Adleman, while sometimes additional safety tests are performed. Real world implementations, however, require *concrete algorithms* that specify in detail how to generate RSA integers. This has led to a variety of standards, notably the standards PKCS#1 (Jonsson & Kaliski 2003), ISO 18033-2 (International Organization for Standards 2006), IEEE 1363-2000 (IEEE working group 2000), ANSI X9.44 (Accredited Standards Committee X9 2007), FIPS 186-3 (Information Technology Laboratory 2009), the standard of the RSA foundation (RSA Laboratories 2000), the standard set

by the German Bundesnetzagentur (Wohlmacher 2009), and the standard resulting from the European NESSIE project (NESSIE working group 2003). All of those standards define more or less precisely how to generate RSA integers and all of them have substantially different requirements. This reflects the intuition that it does not really matter how one selects the prime factors in detail, the resulting RSA modulus will do its job. But what is needed to show that this is really the case?

Following Brandt & Damgård (1993) a quality measure of a generator is the entropy of its output distribution. In abuse of language we will most of the time talk about the *output entropy* of an algorithm. To compute it, we need estimates of the probability that a certain outcome is produced. This in turn needs a thorough analysis of how one generates RSA integers of a specific form. If we can show that the outcome of the algorithm is roughly uniformly distributed, the output entropy is closely related to the count of RSA integers it can produce. It will turn out that in all reasonable setups this count is essentially determined by the desired length of the output. For primality tests there are several results in this direction (see for example Joye & Paillier 2006) but we are not aware of any related work analyzing the output entropy of algorithms for generating RSA integers.

Another requirement for the algorithm is that the output should be 'hard to factor'. Since this statement does not even make sense for a single integer, this means that one has to show that the restrictions on the shape of the integers the algorithm produces do not introduce any further possibilities for an attacker. To prove this, a *reduction* has to be given that reduces the problem of factoring the output to the problem of factoring a product of two primes of similar size, see Section 7. Also there it is necessary to have results on the count of RSA integers of a specific form to make the reduction work. As for the entropy estimations, we do not know any related work on this.

In the following section we will develop a formal framework that can handle all possible definitions for RSA integers. After discussing the necessary number theoretic tools in Section 3, we give explicit formulæ for the count of such integers which will be used later for entropy estimations of the various standards for RSA integers. In Section 4 we show how our general framework can be instantiated, yielding natural definitions for several types of RSA integers (as used later in the standards). Section 5 gives a short overview on generic constructions for fast algorithms that generate such integers almost uniformly. At this point we will have described all necessary techniques to compute the output entropy, which we discuss in Section 6. The following section resolves the second question described above by giving a reduction from factoring special types of RSA integers to factoring a product of two primes of similar size. We finish by applying our results to various standards for RSA integers in Section 8.

We omitted here most of the number theoretic details. For the proofs of those theorems see Loebenberger & Nüsken (2011). Note that for ease of comparison, we have retained the numbering of the extended version.

2 RSA integers in general

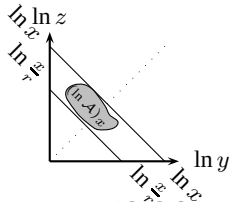
If one generates an RSA integer it is necessary to select for each choice of the security parameter the prime factors from a certain region. This security parameter is typically an

integer k that specifies (roughly) the size of the output. We use a more general definition by asking for integers from the interval $]x/r, x]$, given a *real* bound x and a parameter r (possibly depending on x). Clearly, this can also be used to model the former selection process by setting $x = 2^k - 1$ and $r = 2$. Let us in general introduce a *notion of RSA integers with tolerance* r as a family

$$\mathcal{A} := \langle \mathcal{A}_x \rangle_{x \in \mathbb{R}_{>1}}$$

of subsets of the positive quadrant $\mathbb{R}^2_{>1}$, where for every $x \in \mathbb{R}_{>1}$

$$\mathcal{A}_x \subseteq \left\{ (y,z) \in \mathbb{R}^2_{>1} \,\middle|\, \frac{x}{r} < yz \leq x \right\}.$$



The tolerance r shall always be larger than 1. We allow here that r varies (slightly) with x, which of course includes the case that r is a constant. Typical values used for RSA are $r = 2$ or $r = 4$ which fix the bit-length of the modulus more or less. We can — for a fixed choice of parameters — easily visualize any notion of RSA integers by the corresponding region $\mathcal{A}_x$ in the (y,z)-plane. It is favorable to look at these regions in logarithmic scale. We write $y = e^{\upsilon}$ and $z = e^{\zeta}$ and denote by $(\ln \mathcal{A})_x$ the region in the (υ,ζ)-plane corresponding to the region $\mathcal{A}_x$ in the (y,z)-plane, formally $(\upsilon,\zeta) \in (\ln \mathcal{A})_x :\Leftrightarrow (y,z) \in \mathcal{A}_x$. Now an *$\mathcal{A}$-integer n of size x* — for use as a modulus in RSA — is a product $n = pq$ of a prime pair $(p,q) \in \mathcal{A}_x \cap (\mathbb{P} \times \mathbb{P})$, where $\mathbb{P}$ denotes the set of primes. They are counted by the associated *prime pair counting function* $\#\mathcal{A}$ for the notion $\mathcal{A}$:

$$\#\mathcal{A}\colon \begin{array}{l} \mathbb{R}_{>1} \longrightarrow \mathbb{N}, \\ \quad x \longmapsto \#\left\{ (p,q) \in \mathbb{P} \times \mathbb{P} \,\middle|\, (p,q) \in \mathcal{A}_x \right\}. \end{array}$$

Thus every $\mathcal{A}$-integer $n = pq$ is counted once or twice in $\#\mathcal{A}(x)$ depending on whether only $(p,q) \in \mathcal{A}_x$ or also $(q,p) \in \mathcal{A}_x$, respectively. We call a notion *symmetric* if for all choices of the parameters the corresponding area in the (y,z)-plane is symmetric with respect to the main diagonal, i.e. that $(y,z) \in \mathcal{A}_x$ implies also $(z,y) \in \mathcal{A}_x$. If to the contrary $(y,z) \in \mathcal{A}_x$ implies $(z,y) \notin \mathcal{A}_x$ we call the notion *antisymmetric*. If we are only interested in RSA integers we can always require symmetry or antisymmetry, yet many algorithms proceed in an asymmetric way.

Certainly, we will also need restrictions on the shape of the area we are analyzing: If one considers any notion of RSA integers and throws out exactly the prime pairs one would be left with a prime-pair-free region and any approximation for the count of such a notion based on the area would necessarily have a tremendously large error term. However, for practical applications it turns out that it is enough to consider regions of a very specific form. Actually, we will most of the time have regions whose boundary can be described by graphs of certain smooth functions. In the following, we call notions having such boundaries *monotone*. A more detailed explanation of the restrictions we have to impose to make the number-theoretic work sound can be found in the extended version Loebenberger & Nüsken (2011).

For RSA, people usually prefer two prime factors of roughly the same size, where size is understood as bit length. Accordingly, we call a notion of RSA integers $[c_1, c_2]$-*balanced* iff additionally for every $x \in \mathbb{R}_{>1}$

$$\mathcal{A}_x \subseteq \left\{ (y,z) \in \mathbb{R}^2_{>1} \,\middle|\, y, z \in [x^{c_1}, x^{c_2}] \right\},$$

where $0 < c_1 \leq c_2$ can be thought of as constants or — more generally — as smooth functions in x defining the amount of allowed divergence subject to the side condition that x^{c_1} tends to infinity when x grows. If $c_1 > \frac{1}{2}$ then $\mathcal{A}_x$ is empty, so we will usually assume $c_1 \leq \frac{1}{2}$. In order to prevent trial division from being a successful attacker it would be sufficient to require $y, z \in \Omega\left(\ln^k x\right)$ for every $k \in \mathbb{N}$. Our stronger requirement still seems reasonable and indeed equals the condition Maurer (1995) required for secure RSA moduli, as the supposedly most difficult factoring challenges stay within the range of our attention. As a side-effect this greatly simplifies our approximations later. The German Bundesnetzagentur (see Wohlmacher 2009) uses a very similar restriction in their algorithm catalog. There it is additionally required that the primes p and q are not too close to each other. We ignore this issue here, since the probability that two primes are *very close* to each other would be tiny if the notion from which (p, q) was selected is sufficiently large. If necessary, we are able to modify our notions such that also this requirement is met.

Often the considered integers $n = pq$ are also subject to further side conditions, like $\gcd((p-1)(q-1), e) = 1$ for some fixed public RSA exponent e. Most of the number theoretic work below can easily be adapted, but for simplicity of exposition we will often present our results without those further restrictions and just point out when necessary how to incorporate such additional properties.

As we usually deal with balanced notions the considered regions are somewhat centered around the main diagonal. We will show in Section 7 that if factoring products of two primes is hard then it is also hard to factor integers generated from such notions.

3 Toolbox

We will now develop the necessary number theoretic concepts to obtain formulæ for the count of RSA integers that will later help us to estimate the output entropy of the various standards for RSA integers. In related articles, like Decker & Moree (2008) one finds counts for *one particular* definition of RSA integers. We believe that in the work presented here for the first time a sufficiently general theorem is established that allows to compute the number of RSA integers for *all* reasonable definitions.

We assume the Riemann hypothesis throughout the entire paper. The main terms are the same without this assumption, but the error bounds one obtains are then much weaker. We skip intermediate results here and just summarize the number theoretic work (to ease later comparison we have retained the numbering of the extended version Loebenberger & Nüsken 2011). The following lemma covers all the estimation work.

Lemma 3.6 (Two-dimensional prime sum approximation for monotone notions).
Assume that we have a monotone $[c_1, c_2]$-balanced notion $\mathcal{A}$ of RSA integers with tolerance r, where $0 < c_1 \leq c_2$. (The values r, c_1, c_2 are allowed to vary with x.) Then under the Riemann hypothesis there is a value $\widetilde{a}(x) \in \left[\frac{1}{4c_2^2}, \frac{1}{4c_1^2}\right]$ such that

$$\#\mathcal{A}(x) \in \widetilde{a}(x) \cdot \frac{4\,\mathrm{area}(\mathcal{A}_x)}{\ln^2 x} + \mathcal{O}\left(c_1^{-1} x^{\frac{3+c}{4}}\right),$$

where $c = \max(2c_2 - 1, 1 - 2c_1)$. □

Note that the omitted proof gives a precise expression for $\widetilde{a}(x)$, namely

$$\widetilde{a}(x) = \frac{\iint_{\mathcal{A}_x} \frac{1}{\ln p \ln q} \, \mathrm{d}p \, \mathrm{d}q}{4 \iint_{\mathcal{A}_x} \frac{1}{\ln^2 x} \, \mathrm{d}p \, \mathrm{d}q}.$$

It turns out that we can only evaluate $\widetilde{a}(x)$ numerically in our case and so we tend to estimate also this term. Then we often obtain $\widetilde{a}(x) \in 1 + o(1)$. Admittedly, this mostly eats up the advantage obtained by using the Riemann hypothesis. However, we accept this because it still leaves the option of going through that difficult evaluation and obtain a much more precise answer. If we do not use the Riemann hypothesis we need to replace $\mathcal{O}\left(c_1^{-1} x^{\frac{3+c}{4}}\right)$ with $\mathcal{O}\left(\frac{x}{\ln^k x}\right)$ for any $k > 2$ of your choice.

As mentioned before, in many standards the selection of the primes p and q is additionally subject to the side condition that $\gcd((p-1)(q-1), e) = 1$ for some fixed public exponent e of the RSA cryptosystem. To handle these restrictions, we state a theorem from the extended version

Theorem 3.11. *Let $e \in \mathbb{N}_{>2}$ be a public RSA exponent and $x \in \mathbb{R}$. Then we have for the number $\pi_e(x)$ of primes $p \leq x$ with $\gcd(p-1, e) = 1$ that*

$$\pi_e(x) \in \frac{\varphi_1(e)}{\varphi(e)} \cdot \mathrm{Li}(x) + \mathcal{O}\left(\sqrt{x} \ln x\right),$$

where $\mathrm{Li}(x) = \int_2^x \frac{1}{\ln t} \, \mathrm{d}t$ is the integral logarithm, $\varphi(e)$ is Euler's totient function and

$$\frac{\varphi_1(e)}{\varphi(e)} = \prod_{\substack{\ell \mid e \\ \ell \text{ prime}}} \left(1 - \frac{1}{\ell - 1}\right). \tag{3.12}$$

□

This theorem shows that the prime pair approximation in Lemma 3.6 can be easily adapted to RSA integers whose prime factors satisfy the conditions of Theorem 3.11, since the density of such primes differs for every fixed e just by a constant.

4 Some common definitions for RSA integers

We will now give formal definitions of two specific notions of RSA integers. In particular, we consider the following example definitions within our framework:

– The simple construction given by just choosing two primes in given intervals. This construction occurs in several standards, like the standard of the RSA foundation (RSA Laboratories 2000), the standard resulting from the European NESSIE project (NESSIE working group 2003) and the FIPS 186-3 standard (Information Technology Laboratory 2009). Also open source implementations of `OpenSSL` (Cox *et al.* 2009), `GnuPG` (Skala *et al.* 2009) and the GNU crypto library `GNU Crypto` (Free Software Foundation 2009) use some variant of this construction.

– An algorithmically inspired construction which allows one prime being chosen arbitrarily and the second is chosen such that the product is in the desired interval. This was for example specified as the IEEE standard 1363 (IEEE working group 2000), Annex A.16.11. However, we could not find any implementations following this standard.

4.1 A fixed bound notion

We consider the number of integers smaller than a real positive bound x that have exactly two prime factors p and q, both lying in a fixed interval $]B, C]$, in formulæ:

$$\pi^2_{B,C}(x) := \#\left\{ n \in \mathbb{N} \;\middle|\; \begin{array}{c} \exists p, q \in \mathbb{P} \cap]B, C]: \\ n = pq \;\wedge\; n \leq x \end{array} \right\}.$$

To avoid problems with rare prime squares, which are also not interesting when talking about RSA integers, we instead count

$$\kappa^2_{B,C}(x) := \#\left\{ (p, q) \in (\mathbb{P} \cap]B, C])^2 \;\middle|\; pq \leq x \right\}.$$

Such functions are treated in Loebenberger & Nüsken (2010) . In the context of RSA integers we consider the notion

$$\mathcal{A}^{\mathrm{FB}(r,\sigma)} := \left\langle \left\{ (y, z) \in \mathbb{R}^2_{>1} \;\middle|\; \sqrt{\frac{x}{r}} < y, z \leq \sqrt{r^\sigma x} \;\wedge\; yz \leq x \right\} \right\rangle_{x \in \mathbb{R}_{>1}}$$

with $\sigma \in [0, 1]$. The parameter σ describes the (relative) distance of the restriction $yz \leq x$ to the center of the rectangle in which y and z are allowed. The next theorem follows directly from Loebenberger & Nüsken (2010) but we can also derive it from Lemma 3.6:

Theorem 4.6. *We have for* $\ln r \in o(\ln x)$ *under the Riemann hypothesis*

$$\#\mathcal{A}^{\mathrm{FB}(r,\sigma)}(x) \in \widetilde{a}(x) \frac{4x}{\ln^2 x} \left(\sigma \ln r + 1 - \frac{2}{r^{\frac{1-\sigma}{2}}} + \frac{1}{r} \right) + \mathcal{O}\left(x^{\frac{3}{4}} r^{\frac{1}{4}} \right)$$

with $\widetilde{a}(x) \in \left[\left(1 - \frac{\sigma \ln r}{\ln x + \sigma \ln r}\right)^2, \left(1 + \frac{\ln r}{\ln x - \ln r}\right)^2 \right] \subseteq 1 + o(1).$

□

4.2 An algorithmically inspired notion

A second option to define RSA integers is the following notion: Assume you wish to generate an RSA integer between $\frac{x}{r}$ and x, which has two prime factors of roughly equal size. Then algorithmically we might first generate the prime p and afterward select the prime q such that the product is in the correct interval. As we will see later, this procedure does — however — not produce every number with the same probability, see Section 5. Formally, we consider the notion

$$\mathcal{A}^{\mathrm{ALG}(r,\sigma)}(x) := \left\langle \left\{ (y,z) \in \mathbb{R}^2_{>1} \;\middle|\; \begin{array}{c} r^{\sigma-1}\sqrt{x} < y \le r^\sigma\sqrt{x} \;\wedge\; \frac{x}{ry} < z \le \frac{x}{y} \\ \frac{x}{r} < yz \le x \end{array} \right\} \right\rangle_{x\in\mathbb{R}_{>1}},$$

with $\sigma \in [0,1]$. The parameter σ describes here the (relative) position of the defining area of the notion with respect to the diagonal. Write $\sigma' := \max(\sigma, 1-\sigma)$. Similar to the theorem above we obtain

Theorem 4.11. *Assuming* $\ln r \in o(\ln x)$ *we have under the Riemann hypothesis*

$$\#\mathcal{A}^{\mathrm{ALG}(r,\sigma)}(x) \in \widetilde{a}(x)\frac{4x}{\ln^2 x}\left(\ln r - \frac{\ln r}{r}\right) + \mathcal{O}\left(x^{\frac{3}{4}} r^{\frac{1}{4}}\right),$$

with $\widetilde{a}(x) \in \left[\left(1 - \frac{2\sigma'\ln r}{\ln x + 2\sigma'\ln r}\right)^2, \left(1 + \frac{2(1+\sigma)\ln r}{\ln x - 2(1+\sigma)\ln r}\right)^2\right] \subseteq 1 + o(1).$ □

As we see both notions open a slightly different view. However the outcome is not that different, at least the numbers of described RSA integers are quite close to each other. The proof that this is the case for *all* reasonable notions can be found in the extended version Loebenberger & Nüsken (2011).

Current standards and implementations of various crypto packages mostly use the notions $\mathcal{A}^{\mathrm{FB}(4,0)}$, $\mathcal{A}^{\mathrm{FB}(4,1)}$, $\mathcal{A}^{\mathrm{FB}(2,0)}$ or $\mathcal{A}^{\mathrm{ALG}(2,1/2)}$. For details see Section 8.

5 Generating RSA integers properly

In this section we analyze how to generate RSA integers properly. It completes the picture and we found several implementations overlooking this kind of arguments.
We wish that all the algorithms generate integers with the following properties:

- If we fix x we should with overwhelming probability generate integers that are a product of a prime pair in $\mathcal{A}_x$.
- These integers (not the pairs) should be selected roughly uniformly at random.
- The algorithm should be efficient. In particular, it should need only few primality tests.

5.1 Rejection sampling

Assume that $\mathcal{A}$ is a $[c_1, c_2]$-balanced notion of RSA integers with tolerance r. The easiest approach for generating a pair from $\mathcal{A}$ is based on von Neumann's rejection sampling method. Let $\mathcal{B}_x := x^{[c_1,c_2]} \times x^{[c_1,c_2]}$. There may be better ways for choosing $\mathcal{B}_x \supseteq \mathcal{A}_x$, but we skip this here. We obtain the following straightforward Las Vegas algorithm:

Algorithm 5.2. Generating an RSA integer (Las Vegas version).
Input: A notion $\mathcal{A}$, a bound $x \in \mathbb{R}_{>1}$.
Output: An integer $n = pq$ with $(p, q) \in \mathcal{A}_x$.

1. Repeat 2–4
2. Repeat
3. Select (y, z) at random from $\mathcal{B}_x \cap \mathbb{N}^2$.
4. Until $(y, z) \in \mathcal{A}_x$.
5. Until y prime and z prime.
6. $p \leftarrow y$, $q \leftarrow z$.
7. Return pq.

The expected repetition count of the inner loop is roughly $\frac{\text{area}(\mathcal{B}_x)}{\text{area}(\mathcal{A}_x)}$. The expected number of primality tests is about $\frac{\text{area}(\mathcal{A}_x)}{\#\mathcal{A}(x)}$. This is for many notions in $\mathcal{O}\left(\ln^2 x\right)$. We have seen implementations (for example the one of GnuPG) where the inner and outer loop have been exchanged. This increases the number of primality tests by the repetition count of the inner loop. Also easily checkable additional conditions, like $\gcd((p-1)(q-1), e) = 1$, should be checked before the primality tests to improve the efficiency.

5.2 Inverse transform sampling

Actually we would like to avoid generating out-of-bound pairs completely. To retain uniform selection, we need to select the primes p non-uniformly with the following distribution:

Definition 5.4. *Let $\mathcal{A}$ be a notion of RSA integers with tolerance r. For every $x \in \mathbb{R}_{>1}$ the associated* cumulative distribution function *of $\mathcal{A}_x$ is defined as*

$$F_{\mathcal{A}_x} : \begin{array}{c} \mathbb{R} \longrightarrow [0,1], \\ y \longmapsto \frac{\text{area}(\mathcal{A}_x \cap ([1,y] \times \mathbb{R}))}{\text{area}(\mathcal{A}_x)}. \end{array}$$

In fact we should use the function $G_{\mathcal{A}_x} : \mathbb{R} \to [0,1],\ y \mapsto \frac{\#(\mathcal{A}_x \cap (([1,y] \cap \mathbb{P}) \times \mathbb{P}))}{\#\mathcal{A}_x}$, in order to compute the density but computing $G_{\mathcal{A}_x}$ (or its inverse) is tremendously expensive. Fortunately, by virtue of Lemma 3.6 we know that $F_{\mathcal{A}_x}$ approximates $G_{\mathcal{A}_x}$ for monotone, $[c_1, c_2]$-balanced notions $\mathcal{A}$ quite well. So we use the function $F_{\mathcal{A}_x}$ to capture the distribution properties of a given notion of RSA integers. As can be seen by inspection, in practically relevant examples this function is sufficiently easy to handle. We obtain the following algorithm:

Algorithm 5.5. Generating an RSA integer.
Input: A notion $\mathcal{A}$, a bound $x \in \mathbb{R}_{>1}$.
Output: An integer $n = pq$ with $(p, q) \in \mathcal{A}_x$.

1. Repeat
2. Select y with distribution $F_{\mathcal{A}_x}$ from $\{y \in \mathbb{R} \mid \exists z\colon (y, z) \in \mathcal{A}_x\} \cap \mathbb{N}$.
3. Until y prime.

4. $p \leftarrow y$.
5. Repeat
6. Select z uniformly at random from $\{z \in \mathbb{R} \,|\, (p, z) \in \mathcal{A}_x\} \cap \mathbb{N}$.
7. Until z prime.
8. $q \leftarrow z$.
9. Return pq.

As desired, this algorithm generates any pair $(p, q) \in \mathcal{A}_x \cap (\mathbb{P} \times \mathbb{P})$ with almost the same probability. In order to generate y with distribution $F_{\mathcal{A}_x}$ one can use inverse transform sampling, see for example Knuth (1998). The expected number of primality tests now is in $\mathcal{O}(\ln x)$. Of course we have to take into account that for each trial y the inverse $F^{-1}_{\mathcal{A}_x}(y)$ has to be computed — at least approximately —, yet this cost is usually negligible compared to a primality test.

5.3 Other constructions

There are variants around, where the primes are selected differently: Take an integer randomly from a suitable interval and increase the result until the first prime is found. This has the advantage that the amount of randomness needed is considerably lower and by optimizing the resulting algorithm can also be made much faster. The price one has to pay is that the produced primes will not be selected uniformly at random: Primes p for which $p - 2$ is also prime will be selected with a much lower probability than randomly selected primes of a given length. As shown in Brandt & Damgård (1993) the output entropy of such algorithms is still almost maximal and also generators based on these kind of prime-generators might be used in practice.

5.4 Summary

We have seen that Algorithm 5.2 and Algorithm 5.5 are practical uniform generators for any symmetric or antisymmetric notion.

Note that Algorithm 5.2 and Algorithm 5.5 may, however, still produce numbers in a non-uniform fashion: In the last step of both algorithms a product is computed that corresponds to either one pair or two pairs in $\mathcal{A}_x$. To solve this problem we have two choices: Either we replace $\mathcal{A}$ by its symmetric version $\mathcal{S}$ which we define as $\mathcal{S}_x := \left\{(y, z) \in \mathbb{R}^2_{>1} \,\middle|\, (y, z) \in \mathcal{A}_x \vee (z, y) \in \mathcal{A}_x\right\}$, or by its, say, top half $\mathcal{T}$ given by $\mathcal{T}_x := \{(y, z) \in \mathcal{S}_x \,|\, z \geq y\}$ before anything else.

6 Output entropy

The entropy of the output distribution is one important quality measure of a generator. For primality tests several analyses where performed, see for example Brandt & Damgård (1993) or Joye & Paillier (2006). For generators of RSA integers we are not aware of any work in this direction.

Let $\mathcal{A}_x$ be any monotone notion. Consider a generator G_ϱ that produces a pair of primes $(p, q) \in \mathcal{A}_x$ with distribution ϱ. Seen as random variables, G_ϱ induces two

random variables P and Q by its first and the second coordinate, respectively. The entropy of the generator G_ϱ is given by

$$H(G_\varrho) = H(P \times Q) = H(P) + H\left(Q \,\middle|\, P\right),$$

where H denotes the binary entropy and $H\left(Q \,\middle|\, P\right)$ denotes the conditional entropy. If ϱ is the uniform distribution U we obtain by Lemma 3.6 maximal entropy

$$H(G_U) = \log_2(\#\mathcal{A}\,(x)) \approx \log_2(\text{area}(\mathcal{A}_x)) - \log_2(\ln x) + 1,$$

with an error of very small order. The algorithms from Section 5, however, return the product $P \cdot Q$. The entropy of this random variable can be estimated as

$$\begin{aligned} H(P \cdot Q) &= -\sum_{\substack{n=pq\in\mathbb{N}\\(p,q)\in\mathcal{A}_x}} \text{prob}(P \cdot Q = n) \log_2(\text{prob}(P \cdot Q = n)) \\ &\geq -\sum_{(p,q)\in\mathcal{A}_x} \text{prob}(P \times Q = (p,q)) \log_2(2\,\text{prob}(P \times Q = (p,q))) \\ &= H(P \times Q) - 1. \end{aligned}$$

Some of the standards and implementations in Section 8 (like the standard IEEE 1363-2000 or the implementation of `GNU Crypto`) *do not* generate every possible outcome with the same probability. All of them have in common that the prime p is selected uniformly at random and afterwards the prime q is selected uniformly at random from an appropriate interval. This is a non-uniform selection process since for some choices of p there might be less choices for q.

If in general the probability distribution ϱ is close to the uniform distribution, say $\varrho(p,q) \in [2^{-\varepsilon}, 2^{\varepsilon}]\frac{1}{\#\mathcal{A}(x)}$ for some fixed $\varepsilon \in \mathbb{R}_{>0}$, then the entropy of the resulting generator G_ϱ can be estimated as

$$H(G_U) - \varepsilon \leq H(G_\varrho).$$

7 Complexity theoretic considerations

We are about to reduce factoring products of two comparatively equally sized primes to the problem of factoring integers generated from a sufficiently large notion. As far as we know there are no similar reductions in the literature.

We consider finite sets $M \subset \mathbb{N} \times \mathbb{N}$, in our situation we actually have only prime pairs. The multiplication map μ_M is defined on M and merely multiplies, that is, $\mu_M \colon M \to \mathbb{N},\ (y,z) \mapsto y \cdot z$. The random variable U_M outputs uniformly distributed values from M. An attacking algorithm F gets a natural number $\mu_M(U_M)$ and attempts to find factors inside M. Its success probability

$$\text{succ}_F(M) = \text{prob}\left(\quad F(\mu_M(U_M)) \ \in\ \mu_M^{-1}(\mu_M(U_M)) \quad\right) \tag{7.1}$$

measures its quality in any fixed-size scenario. Integers generated from a notion $\mathcal{A}$ are *hard to factor* iff for all probabilistic polynomial time machines F, all $s \in \mathbb{N}$, there exists a value $x_0 \in \mathbb{R}_{>1}$ such that for any $x > x_0$ we have $\text{succ}_F(\mathcal{A}_x) \leq \ln^{-s} x$.

For any polynomial f we define the set $R_f = \{(m,n) \in \mathbb{N} \,\big|\, m \le f(n) \wedge n \le f(m)\}$ of f-related positive integer pairs. Denote by $\mathbb{P}^{(m)}$ the set of m-bit primes. We can now formulate the basic assumption:

Assumption 7.2 (Intractability of factoring). *For any unbounded positive polynomial f integers from the f-related prime pair family $(\mathbb{P}^{(m)} \times \mathbb{P}^{(n)})_{(m,n)\in R_f}$ are hard to factor.*

This is exactly the definition given by Goldreich (2001). Note that this assumption implies that factoring in general is hard, and it covers the supposedly hardest factoring instances. Now we are ready to state that integers from all relevant notions are hard to factor.

Theorem 7.3. *Let $\ln r \in \Omega\left(\frac{1-2c_1}{\ln^\ell x}\right)$ and $\mathcal{A}$ be a monotone, $[c_1,c_2]$-balanced notion for RSA integers of tolerance r with large area, namely, for some k and large x we have* area $\mathcal{A}_x \ge \frac{x}{\ln^k x}$. *Assume that factoring is difficult in the sense of Assumption 7.2 (or if only integers from the family of linearly related prime pairs are hard to factor). Then integers from the notion $\mathcal{A}$ are hard to factor.* □

Proof. Assume that we have an algorithm F that factors integers generated uniformly from the notion $\mathcal{A}$. Our goal is to prove that this algorithm also factors polynomially related prime pairs successfully. In other words: its existence contradicts the assumption that factoring in the form of Assumption 7.2 is difficult.

By assumption, there is an exponent s so that for any x_0 there is $x > x_0$ such that the assumed algorithm F has success probability $\operatorname{succ}_F(\mathcal{A}_x) \ge \ln^{-s} x$ on inputs from $\mathcal{A}_x$. We are going to prove that for each such x there exists a pair (m_0,n_0), both in the interval $[c_1 \ln x - \ln 2, c_2 \ln x + \ln 2]$, such that F executed with an input from image $\mu_{\mathbb{P}^{m_0},\mathbb{P}^{n_0}}$ still has success probability at least $\ln^{-(s+k)} x$. By the interval restriction, m_0 and n_0 are polynomially (even linearly) related, namely $m_0 < \frac{2c_2}{c_1} n_0$ and $n_0 < \frac{2c_2}{c_1} m_0$ for large x. So that contradicts Assumption 7.2.

First, we cover the set $\mathcal{A}_x$ with small rectangles. Let $S_{m,n} := \mathbb{P}^{(m)} \times \mathbb{P}^{(n)}$ and $I_x := \left\{(m,n) \in \mathbb{N}^2 \,\middle|\, S_{m,n} \cap \mathcal{A}_x \ne \emptyset\right\}$ then

$$\mathcal{A}_x \cap \mathbb{P}^2 \subseteq \biguplus_{(m,n)\in I_x} S_{m,n} =: S_x. \tag{7.4}$$

Next we give an upper bound on the number $\#S_x$ of prime pairs in the set S_x in terms of the number $\#\mathcal{A}(x)$ of prime pairs in the original notion: First, since each rectangle $S_{m,n}$ extends by a factor 2 along each axis we overshoot by at most that factor in each direction, that is, we have for $c_1' = c_1 - (1+2c_1)\frac{\ln 2}{\ln x}$ and all $x \in \mathbb{R}_{>1}$

$$S_x \subset \mathcal{M}_{4x}^{16r,c_1'} = \left\{(y,z) \in \mathbb{R}^2 \,\middle|\, y,z \ge \frac{1}{2}x^{c_1} \;\wedge\; \frac{x}{4r} < yz \le 4x\right\}.$$

Provided x is large enough we can guarantee by Theorem 5.2 from the extended version (similar to Lemma 3.6) that

$$\#S_x \le \#\mathcal{M}^{16r,c_1'}(4x) \le \frac{8x}{c_1'^2 \ln x}.$$

On the other hand side we apply Lemma 3.6 for the notion $\mathcal{A}_x$ and use that $\mathcal{A}_x$ is large by assumption. Let $c = \max(2c_2 - 1, 1 - 2c_1)$. Then we obtain for large x with some $e_{\mathcal{A}}(x) \in \mathcal{O}\left(x^{\frac{3+c}{4}}\right)$.

$$\#\mathcal{A}(x) \geq \frac{\text{area}(\mathcal{A}_x)}{c_2^2 \ln^2 x} - e_{\mathcal{A}}(x) \geq \frac{x}{2c_2^2 \ln^{k+2} x}.$$

Together we obtain

$$\frac{\#\mathcal{A}(x)}{\#S_x} \geq \frac{c_1'^2}{16c_2^2 \ln^{k+1} x} \geq \ln^{-(k+2)} x \tag{7.5}$$

By assumption we have $\text{succ}_F(\mathcal{A}_x) \geq \ln^{-s} x$ for infinitely many values x. Thus F on an input from S_x still has large success even if we ignore that F might be successful for elements on $S_x \setminus \mathcal{A}_x$,

$$\text{succ}_F(S_x) \geq \text{succ}_F(\mathcal{A}_x)\frac{\#\mathcal{A}(x)}{\#S_x} \geq \ln^{-(k+s+2)} x.$$

Finally choose $(m_0, n_0) \in I_x$ for which the success of F on S_{m_0,n_0} is maximal. Then $\text{succ}_F(S_{m_0,n_0}) \geq \text{succ}_F(S_x)$. Combining with the previous we obtain that for infinitely many x there is a pair (m_0, n_0) where the success $\text{succ}_F(S_{m_0,n_0})$ of F on inputs from S_{m_0,n_0} is still larger than inverse polynomial: $\text{succ}_F(S_{m_0,n_0}) \geq \ln^{-(k+s+2)} x$.

For these infinitely many pairs (m_0, n_0) the success probability of the algorithm F on S_{m_0,n_0} is at least $\ln^{-(k+s+2)} x$ contradicting the hypothesis. □

All the specific notions that we have found in the literature fulfill the criterion of Theorem 7.3. Thus if factoring is difficult in the stated sense then each of them is invulnerable to factoring attacks. Note that the above reduction still works if the primes p, q are due to the side condition $\gcd((p-1)(q-1), e) = 1$ for a fixed integer e (see Theorem 3.11). We suspect that this is also the case if p and q are strong primes. Yet, this needs further investigation.

8 Impact on standards and implementations

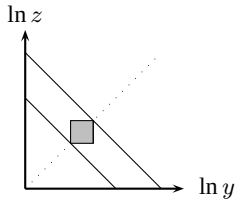
In order to get an understanding of the common implementations, it is necessary to consult the main standard on RSA integers, namely the standard PKCS#1 (Jonsson & Kaliski 2003). However, one cannot find *any* requirements on the shape of RSA integers. Interestingly, they even allow more than two factors for an RSA modulus. Also the standard ISO 18033-2 (International Organization for Standards 2006) does not give any details besides the fact that it requires the RSA integer to be a product of two different primes of similar length.

8.1 RSA-OAEP

The RSA Laboratories (2000) describe the following variant:

Algorithm 8.1. Generating an RSA number for RSA-OAEP and variants.
Input: A number of bits k, the public exponent e.
Output: A number $n = pq$.

1. Pick p from $\left[\left\lfloor 2^{(k-1)/2}\right\rfloor + 1, \left\lceil 2^{k/2}\right\rceil - 1\right] \cap \mathbb{P}$ such that $\gcd(e, p-1) = 1$.
2. Pick q from $\left[\left\lfloor 2^{(k-1)/2}\right\rfloor + 1, \left\lceil 2^{k/2}\right\rceil - 1\right] \cap \mathbb{P}$ such that $\gcd(e, q-1) = 1$.
3. Return pq.



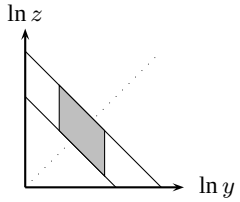
This will produce uniformly at random a number from the interval $[2^{k-1}+1, 2^k-1]$ and no cutting off. The output entropy is thus maximal. So this corresponds to the notion $\mathcal{A}^{\mathrm{FB}(2,0)}$ generated by Algorithm 5.5. The standard requires an expected number of $k \ln 2$ primality tests if the gcd condition is checked first. Otherwise the expected number of primality tests increases to $\frac{\varphi(e)}{\varphi_1(e)} \cdot k \ln 2$ (see (3.12)). We will in the following always mean by the above notation that the second condition is checked first and afterwards the number is tested for primality. For the security Theorem 7.3 applies.

8.2 IEEE

IEEE standard 1363-2000, Annex A.16.11 (IEEE working group 2000) introduces our algorithmic proposal:

Algorithm 8.2. Generating an RSA number, IEEE 1363-2000.
Input: A number of bits k, the odd public exponent e.
Output: A number $n = pq$.

1. Pick p from $\left[2^{\lfloor \frac{k-1}{2} \rfloor}, 2^{\lfloor \frac{k+1}{2} \rfloor} - 1\right] \cap \mathbb{P}$ such that $\gcd(e, p-1) = 1$.
2. Pick q from $\left[\left\lfloor \frac{2^{k-1}}{p} + 1 \right\rfloor, \left\lfloor \frac{2^k}{p} \right\rfloor\right] \cap \mathbb{P}$ such that $\gcd(e, q-1) = 1$.
3. Return pq.



Since the resulting integers are in the interval $[2^{k-1}, 2^k - 1]$ this standard follows $\mathcal{A}^{\mathrm{ALG}(2,1/2)}$ generated by a corrupted variant of Algorithm 5.5 using an expected number of $k \ln 2$ primality tests like the RSA-OAEP standard. The notion it implements is neither symmetric nor antisymmetric. The selection of the integers is *not* done in a uniform way, since the number of possible q for the largest possible p is roughly half of the corresponding number for the smallest possible p. Since the distribution of the outputs is close to uniform, we can use the techniques from Section 6 to estimate the output entropy to find that the entropy-loss is less than 0.69 bit. The (numerically approximated) values in Table 8.1 gave an actual entropy-loss of approximately 0.03 bit.

8.3 NIST

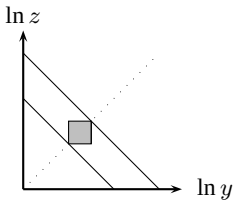
We will now analyze the standard FIPS 186-3 Information Technology Laboratory (2009). In Appendix B.3.1 of the standard one finds the following algorithm:

Algorithm 8.3. Generating an RSA number, FIPS186-3.

Input: A number of bits k, a number of bits $\ell < k$, the odd public exponent $2^{16} < e < 2^{256}$.

Output: A number $n = pq$.

1. Pick p from $\left[\sqrt{2}2^{k/2-1}, 2^{k/2} - 1\right] \cap \mathbb{P}$ such that $\gcd(e, p-1) = 1$ and $p \pm 1$ has a prime factor with at least ℓ bits.
2. Pick q from $\left[\sqrt{2}2^{k/2-1}, 2^{k/2} - 1\right] \cap \mathbb{P}$ such that $\gcd(e, p-1) = 1$ and $q \pm 1$ has a prime factor with at least ℓ bits and $|p - q| > 2^{k/2-100}$.
3. Return pq.



In the standard it is required that the primes p and q shall be either provable prime or at least probable primes. The (at least ℓ-bit) prime factors of $p \pm 1$ and $q \pm 1$ have to be provable primes. We observe that also in this standard a variant of the notion $\mathcal{A}^{\mathrm{FB}(2,0)}$ generated by Algorithm 5.5 is used. The output entropy is thus maximal. However, we do not have any restriction on the parity of k, such that the value $k/2$ is not necessarily an integer. Another interesting point is the restriction on the prime factors of $p \pm 1, q \pm 1$. Our notions cannot directly handle such requirements, but we are confident that this can be achieved by appropriately modifying the densities in Lemma 3.6.

The standard requires an expected number of slightly more than $k \ln 2$ primality tests. It is thus slightly less efficient than the RSA-OAEP standard. For the security the remarks from the end of Section 7 apply.

8.4 ANSI

The ANSI X9.44 standard (Accredited Standards Committee X9 2007), formerly part of ANSI X9.31, requires strong primes for an RSA modulus. Unfortunately, we could not access ANSI X9.44 directly and are therefore referring to ANSI X9.31-1998. Section 4.1.2 of the standard requires that

- $p - 1$, $p + 1$, $q - 1$, $q + 1$ each should have prime factors p_1, p_2, q_1, q_2 that are randomly selected primes in the range 2^{100} to 2^{120},
- p and q shall be the first primes that meet the above, found in an appropriate interval, starting from a random point,
- p and q shall be different in at least one of their first 100 bits.

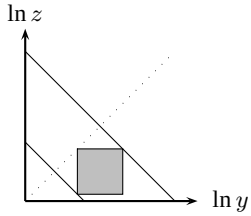
The additional restrictions are similar to the ones required by NIST. This procedure will have an output entropy that is close to maximal (see Section 6).

8.5 OpenSSL

We now turn to implementations: For `OpenSSL` (Cox *et al.* 2009), we refer to the file `rsa_gen.c`. Note that in the configuration the routine used for RSA integer generation can be changed, while the algorithm given below is the standard one. `OpenSSH` (de Raadt *et al.* 2009) uses the same library. Refer to the file `rsa.c`. We have the following algorithm:

Algorithm 8.5. Generating an RSA number in `OpenSSL`.
Input: A number of bits k.
Output: A number $n = pq$.

1. Pick p from $\left[2^{\lfloor \frac{k-1}{2} \rfloor}, 2^{\lfloor \frac{k+1}{2} \rfloor} - 1\right] \cap \mathbb{P}$.
2. Pick q from $\left[2^{\lfloor \frac{k-3}{2} \rfloor}, 2^{\lfloor \frac{k-1}{2} \rfloor} - 1\right] \cap \mathbb{P}$.
3. Return pq.



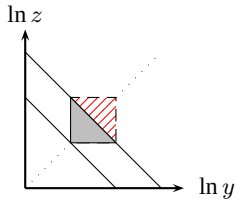
This is nothing but a rejection-sampling method of a notion similar to the fixed-bound notion $\mathcal{A}^{\mathrm{FB}(4,0)}$ generated by Algorithm 5.2. The output entropy is thus maximal. The result the algorithm produces is always in $[2^{k-2}, 2^k - 1]$. It is clear that this notion is antisymmetric and the factors are on average a factor 2 apart of each other. The implementation runs in an expected number of $k \ln 2$ primality tests. The public exponent e is afterwards selected such that $\gcd((p-1)(q-1), e) = 1$. It is thus slightly more efficient than the RSA-OAEP standard. For the security Theorem 7.3 applies.

8.6 GnuPG

Also `GnuPG` (Skala *et al.* 2009) uses rejection-sampling of the fixed-bound notion $\mathcal{A}^{\mathrm{FB}(2,1)}$ generated by a variant of Algorithm 5.2, implying that the entropy of its output distribution is maximal.

Algorithm 8.7. Generating an RSA number in `GnuPG`.
Input: A number of bits k.
Output: A number $n = pq$.

1. Repeat 2–3
2. Pick p from $\left[2^{\lfloor \frac{k-1}{2} \rfloor}, 2^{\lfloor \frac{k+1}{2} \rfloor} - 1\right] \cap \mathbb{P}$.
3. Pick q from $\left[2^{\lfloor \frac{k-1}{2} \rfloor}, 2^{\lfloor \frac{k+1}{2} \rfloor} - 1\right] \cap \mathbb{P}$.
4. Until $\mathrm{len}(pq) = 2\lceil k/2 \rceil$
5. Return pq.



The hatched region in the picture above shows the possible outcomes that are discarded. We refer here to the file `rsa.c`. The algorithm is given in the function `generate_std` and produces always numbers with either k or $k+1$ bits depending on the parity of k. Note that the generation procedure indeed first selects primes before checking the validity of the range. This is of course a waste of resources, see Section 5.

Standard / Implementation	Notion	Entropy for specific k			Remarks
		768	1024	2048	
PKCS#1 ISO 18033-2 ANSI X9.44	Undefined	—	—	—	– – –
FIPS 186-3	$\mathcal{A}^{\mathrm{FB}(2,0)}$	$\lesssim 747.34$	$\lesssim 1002.51$	$\lesssim 2024.51$	strong primes
RSA-OAEP	$\mathcal{A}^{\mathrm{FB}(2,0)}$	747.34	1002.51	2024.51	—
IEEE 1363-2000	$\mathcal{A}^{\mathrm{ALG}(2,\frac{1}{2})}$	749.33	1004.50	2026.50	non-uniform
`GNU Crypto`	$\mathcal{A}^{\mathrm{FB}(2,1)}$	747.89	1003.06	2025.06	non-uniform
`GnuPG`	$\mathcal{A}^{\mathrm{FB}(2,1)}$	748.52	1003.69	2025.69	—
`OpenSSL`	$\cong \mathcal{A}^{\mathrm{FB}(4,0)}$	749.89	1005.06	2027.06	—

Table 8.1. Overview of various standards and implementations. The entropies given there are always above 99.89% of the maximal entropy. As explained in the text, the entropy of the standards is sightly smaller than the values given due to the fixed public exponent e. Additionally there is a small entropy loss for the standard FIPS 186-3 due to the fact that it requires strong primes.

The implementation runs in an expected number of roughly $2.589 \cdot (k+1)\ln 2$ primality tests. It is thus less efficient than the RSA OAEP standards. Like in the other so far considered implementations, the public exponent e is afterwards selected such that $\gcd((p-1)(q-1), e) = 1$. For the security Theorem 7.3 applies.

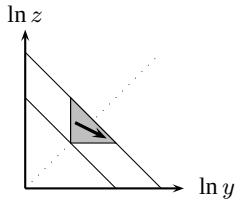
8.7 GNU Crypto

The `GNU Crypto` library (Free Software Foundation 2009) generates RSA integers the following way. Refer here in the file `RSAKeyPairGenerator.java` to the function `generate`.

Algorithm 8.8. Generating an RSA number in `GNU Crypto`.

Input: A number of bits k.
Output: A number $n = pq$.

1. Pick p from $\left[2^{\lfloor \frac{k-1}{2} \rfloor}, 2^{\lfloor \frac{k+1}{2} \rfloor} - 1\right] \cap \mathbb{P}$.
2. Repeat
3. Pick q from $\left[2^{\lfloor \frac{k-1}{2} \rfloor}, 2^{\lfloor \frac{k+1}{2} \rfloor} - 1\right]$.
4. Until $\mathrm{len}(pq) = k$ and $q \in \mathbb{P}$.
5. Return pq.



The arrow in the picture above points to the results that will occur with higher probability. Also here the notion $\mathcal{A}^{\mathrm{FB}(2,1)}$ is used, but the generated numbers will not be uniformly distributed, since for a larger p we have much less choices for q. Since the distribution of the outputs is not close to uniform, we could only compute the entropy for real-world parameter choices numerically (see Table 8.1). For all choices the loss was less than 0.63 bit. The implementation is as efficient as the RSA-OAEP standard.

9 Conclusion

We have seen that there are various definitions for RSA integers, which result in substantially differing standards. We have shown that the concrete specification does not essentially affect the (cryptographic) properties of the generated integers: The entropy of the output distribution is always almost maximal, generating those integers can be done efficiently, and the outputs are hard to factor if factoring in general is hard. It remains open to incorporate strong primes into our model. Also a tight bound for the entropy of non-uniform selection is missing if the distribution is not close to uniform.

Acknowledgements

This work was funded by the B-IT foundation and the state of North Rhine-Westphalia.

References

1. ACCREDITED STANDARDS COMMITTEE X9 (2007). ANSI X9.44-2007: Public Key Cryptography Using Reversible Algorithms for the Financial Services Industry: Transport of Symmetric Algorithm Keys Using RSA. Technical report, American National Standards Institute, American Bankers Association.
2. JØRGEN BRANDT & IVAN DAMGÅRD (1993). On Generation of Probable Primes by Incremental Search. In *Advances in Cryptology: Proceedings of CRYPTO '92*, Santa Barbara, CA, ERNEST BRICKELL, editor, volume 740 of *Lecture Notes in Computer Science*, 358–370. Springer-Verlag, Berlin, Heidelberg. ISBN 978-3-540-57340-1. ISSN 0302-9743. URL `http://dx.doi.org/10.1007/3-540-48071-4_26`.
3. CLIFFORD C. COCKS (1973). A note on 'non-secret encryption'. CESG Memo. URL `http://www.cesg.gov.uk/publications/media/notense.pdf`. Last download 12 May 2009.
4. MARK J. COX, RALF ENGELSCHALL, STEPHEN HENSON & BEN LAURIE (2009). OpenSSL 0.9.8j. Open source implementation. URL `http://www.openssl.org/`. Refer to `openssl-0.9.8j.tar.gz`. Last download 21 April 2009.
5. ANDREAS DECKER & PIETER MOREE (2008). Counting RSA-integers. *Results in Mathematics* **52**, 35–39. URL `http://dx.doi.org/10.1007/s00025-008-0285-5`.
6. JAMES H. ELLIS (1970). The possibility of secure non-secret digital encryption. URL `http://cryptocellar.web.cern.ch/cryptocellar/cesg/possnse.pdf`. Last download 12 May 2009.
7. FREE SOFTWARE FOUNDATION (2009). GNU Crypto. Open source implementation. URL `http://www.gnu.org/software/gnu-crypto/`. Refer to `gnu-crypto-2.0.1.tar.bz2`. Last download 21 April 2009.
8. ODED GOLDREICH (2001). *Foundations of Cryptography*, volume I: Basic Tools. Cambridge University Press, Cambridge. ISBN 0-521-79172-3.
9. IEEE WORKING GROUP (2000). IEEE 1363-2000: Standard Specifications For Public Key Cryptography. IEEE standard, IEEE, New York, NY 10017, USa. URL `http://grouper.ieee.org/groups/1363/P1363/`.
10. INFORMATION TECHNOLOGY LABORATORY (2009). FIPS 186-3: Digital Signature Standard (DSS). Technical report, National Institute of Standards and Technology.

11. INTERNATIONAL ORGANIZATION FOR STANDARDS (2006). ISO/IEC 18033-2, Encryption algorithms — Part 2: Asymmetric ciphers. Technical report, International Organization for Standards.
12. JAKOB JONSSON & BURT KALISKI (2003). Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1. URL `http://tools.ietf.org/html/rfc3447`. RFC 3447.
13. MARC JOYE & PASCAL PAILLIER (2006). Fast Generation of Prime Numbers on Portable Devices: An Update. In *Cryptographic Hardware and Embedded Systems, Workshop, CHES'06*, Yokohama, Japan, LOUIS GOUBIN & MITSURU MATSUI, editors, volume 4249 of *Lecture Notes in Computer Science*, 160–173. Springer-Verlag, Berlin, Heidelberg. ISBN 978-3-540-46559-1. ISSN 0302-9743. URL `http://dx.doi.org/10.1007/11894063_13`.
14. DONALD E. KNUTH (1998). *The Art of Computer Programming, vol. 2, Seminumerical Algorithms*. Addison-Wesley, Reading MA, 3rd edition. ISBN 0-201-89684-2. First edition 1969.
15. DANIEL LOEBENBERGER & MICHAEL NÜSKEN (2010). Coarse-grained integers. *e-print arXiv:1003.2165v1* URL `http://arxiv.org/abs/1003.2165`.
16. DANIEL LOEBENBERGER & MICHAEL NÜSKEN (2011). Analyzing standards for RSA integers – extended version. *e-print arXiv:1104.4356v2* URL `http://arxiv.org/abs/1104.4356`.
17. UELI M. MAURER (1995). Fast Generation of Prime Numbers and Secure Public-Key Cryptographic Parameters. *Journal of Cryptology* **8**(3), 123–155. URL `http://dx.doi.org/10.1007/BF00202269`.
18. NESSIE WORKING GROUP (2003). NESSIE D20 - NESSIE security report. Technical report, NESSIE.
19. JOHN VON NEUMANN (1951). Various techniques used in connection with random digits. Monte Carlo methods. *National Bureau of Standards, Applied Mathematics Series* **12**, 36–38.
20. THEO DE RAADT, NIELS PROVOS, MARKUS FRIEDL, BOB BECK, AARON CAMPBELL & DUG SONG (2009). OpenSSH 2.1.1. Open source implementation. URL `http://www.openssh.org/`. Refer to `openssh-2.1.1p4.tar.gz`. Last download 21 April 2009.
21. RONALD L. RIVEST, ADI SHAMIR & LEONARD M. ADLEMAN (1977). A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Technical Report MIT/LCS/TM-82, Massachusetts Institute of Technology, Laboratory for Computer Science, Cambridge, Massachusetts.
22. RONALD L. RIVEST, ADI SHAMIR & LEONARD M. ADLEMAN (1978). A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM* **21**(2), 120–126.
23. RSA LABORATORIES (2000). RSAES-OAEP Encryption Scheme. Algorithm specification and supporting documentation, RSA Security Inc., Bedford, MA 01730 USA. URL `ftp://ftp.rsasecurity.com/pub/rsalabs/rsa_algorithm/rsa-oaep_spec.pdf`.
24. MATTHEW SKALA, MICHAEL ROTH, NIKLAS HERNAEUS, RÉMI GUYOMARCH & WERNER KOCH (2009). GnuPG. Open source implementation. URL `http://www.gnupg.org/`. Refer to `gnupg-2.0.9.tar.bz2`. Last download 21 April 2009.
25. PETRA WOHLMACHER (2009). Bekanntmachung zur elektronischen Signatur nach dem Signaturgesetz und der Signaturverordnung (Übersicht über geeignete Algorithmen). *Bundesanzeiger* **2009**(13), 346–350. ISSN 0344-7634. Preprint at `http://www.bundesnetzagentur.de/media/archive/14953.pdf`.