RSA signatures and Rabin–Williams signatures: the state of the art

Daniel J. Bernstein

Department of Mathematics, Statistics, and Computer Science (M/C 249)
The University of Illinois at Chicago, Chicago, IL 60607–7045
djb@cr.yp.to
Date of this document: 2008.01.31. Permanent ID of this document:
5e92b45abdf8abc4e55ea02607400599.

Abstract. State-of-the-art modular-root signature systems incorporate many useful features that were not present in the original RSA signature system. This paper surveys those features.

1 Introduction

You're a cryptographer. You know the general idea of modular-root signature systems such as "RSA": a signature is an eth root modulo a public key n whose prime factorization is known to the signer. But did you know that signatures can be compressed to 1/2 size? That keys can be compressed to 1/3 size? That signature verification can be much faster than modular cubing—in fact, even faster than modular squaring?

State-of-the-art modular-root signature systems have evolved far beyond the very simple signature system published by Rivest, Shamir, and Adleman thirty years ago. The first few steps in this evolution, the most dramatic improvements in speed and in security, are integrated into typical "RSA" descriptions and are well known; but it seems to me that most cryptographers haven't heard about the most recent improvements and don't realize how small and fast these signature systems can be.

My goal in this paper is to explain, starting from the original 1977/1978 RSA signature system, the changes that have produced today's state-of-the-art signature systems. Specifically:

- Section 2, hashing (1979 Rabin): Messages are scrambled by a public hash function H. A signature of m is an eth root of $H(m)$, not an eth root of m. This is essential for security.
- Section 3, small exponent (1979 Rabin): The exponent e in the signature equation $s^e \equiv \cdots$ is a small fixed integer such as 3, rather than a large integer chosen randomly by the signer. This makes verification *much* faster, and saves space in public keys.
- Section 4, exponent 2 (1979 Rabin): The exponent is 2 rather than 3. This slightly complicates signing, but it speeds up verification, and it improves the signature-compression and signature-expansion features discussed later.

- Section 5, hash randomization (1979 Rabin): The signer inserts a B-bit random string r in front of a message m being signed. A signature of m is a root of $H(r, m)$. Interesting options for B include $B = 0$, $B = 1$, and $B = 128$; each option has advantages and disadvantages.
- Section 6, tweaks (1980 Williams): Square roots s are replaced by tweaked square roots (e, f, s). This speeds up signing.
- Section 7, signature lifting (1997 Bernstein): A square root s of h modulo pq is transmitted as (s, t) satisfying $s^2 - tpq = h$. This option doubles the space taken by signatures but allows extremely fast verification.
- Section 8, message recovery: Hash functions H are designed so that the first C bits of (r, m) can be efficiently computed from $H(r, m)$. This allows users to compress signed messages by C bits with efficient decompression.
- Section 9, lattice signature compression (2004 Bleichenbacher): A square root s is transmitted as the largest under-half-size denominator in the continued fraction for fs/pq. This option requires some computation for compression and decompression, but eliminates 1/2 of the space taken by signatures.
- Section 10, lattice key compression (2003 Coppersmith): Keys are chosen to share the top 2/3 of their bits with a standard key. This option takes extra time in key generation but eliminates 2/3 of the space taken by keys.

Implementors will have to consult other sources to see how state-of-the-art hash functions H are constructed, but all other details of modular-root signature systems should be clear from this paper.

I don't mean to suggest that the only interesting signature systems are modular-root signature systems. In applications where signature length is much more important than verification time, systems of ElGamal–Schnorr–ECDSA type are better choices than modular-root signature systems.

2 Hashing

In state-of-the-art systems, messages are scrambled by a public hash function H. A signature of m is not an eth root of m modulo the public key n; it is an eth root of $H(m)$ modulo the public key n. This is essential for security.

History. The system proposed by Rivest, Shamir, and Adleman did not hash messages; it was trivially breakable. Specifically, [31, page 122, first display] defines a signature S of a message M under a public key B by "$S = D_B(M)$," never mentioning hashing; [31, page 122, third display] defines D as a power of its input. One trivial attack is to forge the message 1 with signature 1.

Rabin's system in [30] did hash messages; it remains unbroken today. The apparent security benefit of hashing was mentioned in [30, page 10, last sentence]: "Actually, this [attack idea] does not seem a serious threat because of the hashing effected by $C(M)$."

Many authors unjustifiably refer to an oversimplified, trivially breakable, non-hashing system as "Rabin's system"; consider, for example, the claim by Goldwasser, Micali, and Rivest in [17, Section 3] that "Rabin's signature scheme

is totally breakable if the enemy uses a directed chosen-message attack." Some authors incorrectly describe hashing as merely a way to handle long messages, rather than as an essential component of the system no matter what the message length might be; see, e.g., [32, Section 7.1]. Modern treatments of "RSA" usually include hashing but usually fail to give Rabin any credit for the idea.

3 Small exponent

In state-of-the-art systems, the exponent e in the signature equation $s^e \equiv \cdots$ is a small fixed integer, rather than a large integer chosen randomly by the signer. This makes verification *much* faster. It also saves space in public keys.

History. The system proposed by Rivest, Shamir, and Adleman used large exponents. See [31, page 123, fifth line]: "You then pick the integer d to be a large, random integer which is relatively prime to $(p-1)*(q-1)$ ... The integer e is ... the 'multiplicative inverse' of d."

Rabin in [30, page 5] pointed out the tremendous speed advantage of small exponents: "computation time for this function is several hundred times faster ... than the corresponding algorithms in [the 1978 RSA paper]."

Most modern descriptions of "RSA," and all serious "RSA" implementations, include small fixed exponents, typically 3 or 17 or 65537. Most of the descriptions fail to give Rabin any credit for the idea, and many of the descriptions actively miscredit small exponents to the RSA papers [16] and [31]. For example, Knuth in [22, pages 386–389] (and again in [23, pages 403–406]) explained an exponent-3 system and unjustifiably called it "RSA." As another example, here's a quote from a well-known algorithms book by Cormen, Leiserson, Rivest, and Stein: "The RSA cryptosystem ... Select a *small* odd integer e ... The RSA cryptosystem was proposed in 1977 by Rivest, Shamir, and Adleman." (Emphasis added.)

Large exponents have inexplicably attracted more attention than small fixed exponents as the topic of security proofs, even though small exponents are just as easy for theoreticians to handle and much more interesting for practitioners. Bellare and Rogaway in [6] analyzed a traditional system that they called "FDH," and a system of their own design called "PSS," in both cases using large exponents. See [6, Section 2.1]; see also Coron's [13, Section 2.3] and [15, Definition 4]. Katz and Wang were exponent-agnostic in [18]: they stated their results for more general "claw-free permutation pairs." The "PRab" claim in [6, Theorem 6.1] used a small exponent, but the proof was merely outlined; [14, Theorem 4] used a small exponent, but no proof was given.

"Strong RSA" proofs require large exponents, but "strong RSA" signature systems do not provide fast verification and do not seem to have attracted any practical interest.

4 Exponent 2

State-of-the-art systems use exponent 2 rather than exponent 3. This speeds up verification, and improves the signature-compression and signature-expansion features discussed in subsequent sections. The signer's secret primes p and q are chosen from $3 + 4\mathbf{Z}$ to simplify signing.

How signers choose square roots. A square h modulo pq usually does not have a unique square root. Which choice does the signer make? One can find three answers in the literature.

Principal square roots: The signer finds the unique square root s of h such that s is itself a square. The most obvious choice of a square root of h modulo p is $h^{(p+1)/4} \bmod p$; the most obvious choice of a square root of h modulo q is $h^{(q+1)/4} \bmod q$; combining these two choices by the Chinese remainder theorem produces the principal square root of h modulo pq. This is the simplest, and the most popular, signing algorithm.

|Principal| square roots: The signer finds the principal square root s of h as above, and then replaces s with $\min\{s, pq - s\}$. The point is that $\min\{s, pq - s\}$ takes a bit less space than s.

Unstructured square roots: The signer chooses s from the uniform distribution among square roots of h. There is a danger here: revealing *two* uniform random square roots of h has a good chance of immediately revealing a factor of n. The easiest way to avoid this danger is to use **fixed** square roots, i.e., to repeat the same square root of h if the same h appears again.

Fixed unstructured square roots might seem to require the signer to remember all previous choices of square roots. However, the signer can produce indistinguishable results without memory, assuming standard conjectures in secret-key cryptography. The trick is simple: the signer replaces the random bits with secret functions of h. This trick was posted to `sci.crypt` in 1997 by Barwood and independently by Wigley; see [3] and [33]. The general idea of replacing a randomly generated array by output of a secret function, so that the array can be regenerated on demand (and can have exponential size or more) without consuming memory, was published by Goldreich in 1986, with credit to Levin; but Goldreich's signatures were not fixed.

Security proofs require different work for principal square roots, |principal| square roots, and unstructured square roots. Unstructured square roots allow the simplest proofs, and seem to allow "tight" proofs in some cases where principal signatures seem to allow only "loose" proofs. See my paper [10] for further discussion.

History. Exponent 2 was introduced by Rabin in [30]. Most writers fail to give credit to Rabin for hashing and small exponents but do give credit to Rabin for exponent 2. I see no reason to use any other exponent; perhaps 2 will eventually become the most popular exponent, and, as a side effect, Rabin will receive more of the recognition that he deserves.

5 Hash randomization: r

In Rabin's system, the signer actually computes a square root of $H(r, m)$, not a square root of $H(m)$. Here r is a string selected randomly by the signer. The number of bits of r is a system parameter B. This randomization was introduced by Rabin in [30]: specifically, Rabin suggested that the signer choose a random 60-bit string r.

One can see, in the literature, five different strategies to choose the parameter B:

- Choose $B = 0$. This means that r is empty and that the H input is not actually randomized. The main argument for this choice is that any larger B means extra effort to generate r, extra effort to include r in the H input, and extra effort to transmit r along with s.
- Choose $B = 1$. The main argument for this choice is that a nonzero B is required for the type of tight security proof introduced by Katz and Wang in [18]. The conventional wisdom is that $B = 0$ does not allow a tight security proof; see the "FDH" analyses in [6] and [13]. On the other hand, my paper [10] proves tight security for fixed unstructured signatures even in the case $B = 0$. Koblitz and Menezes conjecture in [24, Section 3.4] and [25, Section 4.3] that $B = 1$ has the same security as $B = 0$.
- Choose $B = 8$. As a historical matter, Rabin's system was able to produce signatures for only about 1/4 of all choices of r (since only a small fraction of all integers mod pq are squares), and Rabin suggested trying again if the first r failed; having 256 choices of r means that all choices will fail with probability about 2^{-106}. However, the Rabin–Williams system eliminates these failures, as discussed in Section 6. The only remaining argument for $B = 8$ is that it marginally improves the tightness of the Katz–Wang approach.
- Choose B large enough to prevent the attacker from guessing r in advance; for example, $B = 128$. This choice allows a different type of tight security proof that seems to have been rendered obsolete by the idea of "fixed" signatures.
- Choose B large enough to prevent all collisions in r: for example, $B = 256$. This choice allows an older type of tight security proof that seems to have been obsolete for many years.

My impression is that, in practice, the choice $B = 0$ is by far the most popular choice, although I have not done an exhaustive study.

One might wonder, from the above description, why large choices of B would attract any interest, and in particular why Rabin chose $B = 60$ rather than $B = 8$. The answer is that large choices of B are often conjectured to make non-generic attacks, attacks that pay attention to the hash function H, more difficult. For example, MD5-based signatures have been broken for $B = 0$ but not for $B = 128$. Does a larger B allow us to choose a smaller, faster hash function H? Could this compensate for the direct costs of a longer r? Resolving these questions means understanding the cost-security tradeoff for hash functions, obviously not an

easy task. For the moment I recommend that theoreticians and practitioners remain agnostic and continue to investigate all possible B's.

Reader beware: Many authors have failed to give Rabin proper credit for his randomized signatures (r, s). For example, Goldwasser and Bellare have posted lecture notes (1) claiming that Rabin introduced a signature system with neither H nor r; (2) assigning credit to a 1983 paper of Goldwasser, Micali, and Yao for "pioneering" randomized signatures; and then (3) describing a "PSS0" system—randomized (r, s)—as if it were new. Similar comments apply to the "PFDH" system in [15] and [18].

6 The tweaks e and f

Recall that Rabin's system needed to try several values of r, on average about 4 values, before finding a square $H(r, m)$ modulo pq. The Rabin–Williams system eliminates this problem by using *tweaked* square roots in place of square roots. A tweaked square root of h modulo pq is a vector (e, f, s) such that $e \in \{-1, 1\}$, $f \in \{1, 2\}$, and $efs^2 - h \in pq\mathbf{Z}$; the signer's secret primes p and q are chosen from $3 + 8\mathbf{Z}$ and $7 + 8\mathbf{Z}$ respectively. Each h has exactly four tweaked square roots, so each choice of r works, speeding up signatures.

Avoiding Jacobi symbols. I've noticed that some programmers fear exponent 2. There appears to be a widespread belief that fast exponent-2 signing requires Euclid-type computations of Jacobi symbols. For example, the "IEEE Standard Specifications for Public-Key Cryptography" claim that, compared to exponent 2, exponent 3 has a "code-size and speed advantage because there is no need to compute the Jacobi symbol."

The simple fact, however, is that Euclid-type computations of Jacobi symbols are *not* required for Rabin–Williams signatures. Here is a straightforward high-speed fault-tolerant algorithm that, given h, p, q, computes a tweaked square root (e, f, s) of h modulo pq:

1. Compute $U \leftarrow h^{(q+1)/8} \bmod q$.
2. If $U^4 - h \bmod q = 0$, set $e = 1$; otherwise set $e = -1$.
3. Compute $V \leftarrow (eh)^{(p-3)/8} \bmod p$.
4. If $(V^4(eh)^2 - eh) \bmod p = 0$, set $f = 1$; otherwise set $f = 2$.
5. Precompute $2^{(3q-5)/8} \bmod q$; compute $W \leftarrow f^{(3q-5)/8}U \bmod q$.
6. Precompute $2^{(9p-11)/8} \bmod p$; compute $X \leftarrow f^{(9p-11)/8}V^3eh \bmod p$.
7. Precompute $q^{p-2} \bmod p$; compute $Y \leftarrow W + q(q^{p-2}(X - W) \bmod p)$.
8. Compute $s \leftarrow Y^2 \bmod pq$.
9. Fault tolerance: If $efs^2 \bmod pq \neq h$, start over. (This never happens if all calculations are carried out correctly.)

The bottlenecks in this algorithm are one exponentiation modulo q in the first step and one exponentiation modulo p in the third step. The information that would be extracted from a Euclid-type Jacobi-symbol computation is trivially extracted from a few multiplications. The output of this algorithm is the **principal** tweaked square root (e, f, s) of h: this means that s is a square modulo

pq, that e is 1 if and only if h is a square modulo q, and that f is 1 if and only if eh is a square modulo p.

History. The tweaks e and f were introduced by Williams in [34].

The same tweaks were republished without credit to Williams as the highlight of the Kurosawa–Ogata paper [27] two decades later. Specifically: [27, Section 3] reviews "Rabin's digital signature scheme" with verification equation "$x^2 = H(M \circ R) \bmod N$"; [27, Section 4] presents "our basic scheme" with verification equation "$x^2 = \alpha_i H(M) \bmod N$" using constants $\alpha_0 = 1, \alpha_1, \alpha_2, \alpha_3 = \alpha_1\alpha_2$ with different quadratic characters; [27, Section 1] advertises the omission of r and the fact that the first square root works ("much faster"); [27, Section 5] presents "our improved scheme" that specifies $\alpha_1 = -1$ and $\alpha_2 = 2$; [27, Section 7] presents a "further improvement" where the choice among $\{-2, -1, 1, 2\}$ is transmitted as part of the signature rather than reconstructed by the verifier.

I posted the comment "The only Jacobi symbols used are $b^{(p-1)/2} \bmod p$, in situations where $b^{(p+1)/4} \bmod p$ is going to be computed in any case" to `sci.crypt` in October 2000, in response to the comment "the Jacobi symbol is hard to grasp by the implementor." See [8].

7 Signature expansion

The Rabin–Williams signature system offers another attractive option for applications where verification speed is much more important than signature length: signatures can be expanded for faster verification. Specifically, the signature (e, f, r, s) satisfying $efs^2 \equiv H(r, m) \pmod{pq}$ can be converted into an expanded signature (e, f, r, s, t) satisfying $efs^2 - pqt = H(r, m)$. The verifier can efficiently check the latter equation modulo a random 128-bit prime (or several smaller primes) with negligible chance of error. The verifier can amortize the prime-generation cost across any number of signatures by keeping the prime (or prime list) secret and reusing it.

A similar idea applies to exponent-3 "RSA" but requires a double-length t, considerably slowing down verification.

History. I posted this expansion idea to `sci.crypt` in March 1997. See [7].

8 Message recovery

Hash functions H can be, and often are, designed to allow "message recovery." This means that a fixed-length prefix of (r, m), say the first C bits, can be computed efficiently from $H(r, m)$. See [6, Section 5] and [18, Section 4.2] for generically safe methods to construct H from another hash function.

The advantage of "message recovery" is that it allows compression of signed messages: one simply omits the first C bits of (r, m, s). The verifier sees s, computes the alleged $H(r, m)$ by computing $s^2 \bmod pq$, recovers the prefix of (r, m), and then checks that $H(r, m)$ matches.

Message recovery is often viewed as an argument for choosing a large B, such as $B = 128$. The argument is as follows: "Message recovery eliminates the space required by r. The space required by r was the only disadvantage of a large B. Maybe a large B stops attacks." There are several problems with this argument. First, bandwidth is only part of the picture: for example, a larger B means a larger cost to generate r. Second, signed messages are usually uncompressed; one important reason is that uncompressed signatures (and expanded signatures, as discussed in Section 7) allow faster verification. Third, except when (r, m) is extremely short, the alleged savings is a myth. Adding 128 bits to r means pushing 128 bits of m out of the compressed prefix of (r, m), and therefore adding 128 bits to the length of the signed message.

9 Signature compression

Another way to save space—more effective than message recovery when (r, m) is short—is to transmit all of (r, m) but transmit only the top half of the bits of s. The receiver can use Coppersmith's algorithm (see, e.g., my survey [9]) to find a square root of $H(r, m)$ modulo pq given the top half of the bits of the square root. Bleichenbacher in [11] proposed a better approach, allowing the same amount of compression with much faster decompression and verification: s is transmitted as the largest under-half-size denominator in the continued fraction for fs/pq.

A similar compression method applies to exponent-3 "RSA" but saves only 1/3 of the signature bits rather than 1/2.

10 Key compression

Yet another way to save space is to compress public keys pq. It is widely known that RSA/Rabin keys can be compressed to 1/2 size. It is not so widely known that Coppersmith found a method to compress keys to 1/3 size. It is also not widely known that this compression—done properly!—can be proven to preserve security, not just against generic attacks but against all attacks. The critical point is that generating one key p_1q_1 in the conventional way, and then generating another key pq that shares the top 1/2 (or 2/3) of the bits of p_1q_1, produces exactly the same distribution of pq that would have been produced by generating pq in the conventional way.

Key compression has exactly the same benefits for higher-exponent "RSA," so it is orthogonal to a comparison of Rabin–Williams with "RSA." It is, however, relevant to a comparison of Rabin–Williams with signature systems of ElGamal/Schnorr/ECDSA type. For example, a 1024-bit Rabin–Williams signature can be compressed to a 512-bit signature, and a 1024-bit key can be compressed to a 352-bit key. A typical ECDSA variant at the same conjectured security level has a smaller signature (320 bits) and a smaller key (160 bits) but has much slower verification; for many applications, the slowdown in verification outweighs the 192-bit savings in signature length.

Bleichenbacher pointed out a way to further compress keys pq—all the way down to 0 bits!—inside vectors (pq, e, f, s, r, m). The idea is to recover pq as a divisor of $efs^2 - H(r, m)$. The standard compression method (or, as an alternative, Coppersmith's compression method) already reveals the top 1/2 (or 2/3) of the bits of the divisor, and the remaining bits are easily (or very easily) found by lattice-basis-reduction techniques.

References

1. Vijay Atluri (program chair), Trent Jaeger (program chair), *Proceedings of the 10th ACM conference on Computer and communications security*, ACM Press, 2003. ISBN 1–58113–738–9. See [18].
2. Rana Barua, Tanja Lange (editors), *Progress in Cryptology—INDOCRYPT 2006, 7th International Conference on Cryptology in India, Kolkata, India, December 11–13, 2006, Proceedings*, Lecture Notes in Computer Science, 4329, Springer, 2006. ISBN 3–540–49767–6. See [25].
3. George Barwood, *Digital signatures using elliptic curves*, message `32f519ad.19609226@news.dial.pipex.com` posted to `sci.crypt` (1997). URL: `http://groups.google.com/group/sci.crypt/msg/b28aba37180dd6c6`. Citations in this document: §4.
4. Mihir Bellare (editor), *Advances in cryptology—CRYPTO 2000: proceedings of the 20th Annual International Cryptology Conference held in Santa Barbara, CA, August 20–24, 2000*, Lecture Notes in Computer Science, 1880, Springer-Verlag, Berlin, 2000. ISBN 3–540–67907–3. MR 2002c:94002. See [13].
5. Mihir Bellare, Phillip Rogaway, *The exact security of digital signatures: how to sign with RSA and Rabin*, in [28] (1996), 399–416; see also newer version [6].
6. Mihir Bellare, Phillip Rogaway, *The exact security of digital signatures: how to sign with RSA and Rabin* (1996); see also older version [5]. URL: `http://www-cse.ucsd.edu/~mihir/papers/exactsigs.html`. Citations in this document: §3, §3, §3, §5, §8.
7. Daniel J. Bernstein, *The world's fastest digital signature system*, message `1997Mar1104.27.46.12488@koobera.math.uic.edu` posted to `sci.crypt` (1997). URL: `http://groups.google.com/group/sci.crypt/msg/840e777ec0fc5679`. Citations in this document: §7.
8. Daniel J. Bernstein, *Choice of public exponent in RSA signatures*, message `20000ct208.43.07.252@cr.yp.to` posted to `sci.crypt` (2000). URL: `http://groups.google.com/group/sci.crypt/msg/6b2ecd514293b33e`. Citations in this document: §6.
9. Daniel J. Bernstein, *Reducing lattice bases to find small-height values of univariate polynomials*, in [12] (2007). URL: `http://cr.yp.to/papers.html#smallheight`. Citations in this document: §9.
10. Daniel J. Bernstein, *Proving tight security for Rabin–Williams signatures* (2008). URL: `http://cr.yp.to/papers.html#rwtight`. Citations in this document: §4, §5.
11. Daniel Bleichenbacher, *Compressing Rabin signatures*, in [29] (2004), 126–128. Citations in this document: §9.
12. Joe P. Buhler, Peter Stevenhagen (editors), *Surveys in algorithmic number theory*, Mathematical Sciences Research Institute Publications, 44, to appear, Cambridge University Press, New York, 2007. See [9].

13. Jean-Sébastien Coron, *On the exact security of Full Domain Hash*, in [4] (2000), 229–235. MR 2002e:94109. URL: `http://www.eleves.ens.fr/home/coron/publications/publications.html`. Citations in this document: §3, §5.
14. Jean-Sébastien Coron, *Security proof for partial-domain hash signature schemes*, in [35] (2002), 613–626. URL: `http://www.gemplus.com/smart/r_d/publications/`. Citations in this document: §3.
15. Jean-Sébastien Coron, *Optimal security proofs for PSS and other signature schemes*, in [19] (2002), 272–287. URL: `http://www.eleves.ens.fr/home/coron/publications/publications.html`. Citations in this document: §3, §5.
16. Martin Gardner, *A new kind of cipher that would take millions of years to break*, Scientific American (1977), 120–124. Citations in this document: §3.
17. Shafi Goldwasser, Silvio Micali, Ronald L. Rivest, *A digital signature scheme secure against adaptive chosen-message attacks*, SIAM Journal on Computing **17** (1988), 281–308. ISSN 0097–5397. MR 89e:94009. URL: `http://theory.lcs.mit.edu/~rivest/publications.html`. Citations in this document: §2.
18. Jonathan Katz, Nan Wang, *Efficiency improvements for signature schemes with tight security reductions*, in [1] (2003), 155–164. URL: `http://www.cs.umd.edu/~jkatz/papers.html`. Citations in this document: §3, §5, §5, §8.
19. Lars Knudsen (editor), *Advances in cryptology—EUROCRYPT 2002: proceedings of the 21st International Annual Conference on the Theory and Applications of Cryptographic Techniques held in Amsterdam, April 28–May 2, 2002*, Lecture Notes in Computer Science, 2332, Springer-Verlag, Berlin, 2002. ISBN 3–540–43553–0. See [15].
20. Donald E. Knuth, *The art of computer programming, volume 2: seminumerical algorithms*, 1st edition, 1st printing, Addison-Wesley, Reading, 1969; see also newer version [21]. MR 44:3531.
21. Donald E. Knuth, *The art of computer programming, volume 2: seminumerical algorithms*, 1st edition, 2nd printing, Addison-Wesley, Reading, 1971; see also older version [20]; see also newer version [22]. MR 44:3531.
22. Donald E. Knuth, *The art of computer programming, volume 2: seminumerical algorithms*, 2nd edition, Addison-Wesley, Reading, 1981; see also older version [21]; see also newer version [23]. ISBN 0–201–03822–6. MR 83i:68003. Citations in this document: §3.
23. Donald E. Knuth, *The art of computer programming, volume 2: seminumerical algorithms*, 3rd edition, Addison-Wesley, Reading, 1997; see also older version [22]. ISBN 0–201–89684–2. Citations in this document: §3.
24. Neal Koblitz, Alfred J. Menezes, *Another look at "provable security"*, revised 4 May 2005 (2005); see also newer version [26]. URL: `http://eprint.iacr.org/2004/152/`. Citations in this document: §5.
25. Neal Koblitz, Alfred J. Menezes, *Another look at "provable security". II*, in [2] (2006), 148–175. URL: `http://eprint.iacr.org/2006/229`. Citations in this document: §5.
26. Neal Koblitz, Alfred J. Menezes, *Another look at "provable security"*, Journal of Cryptology **20** (2007), 3–37; see also older version [24]. ISSN 0933–2790.
27. Kaoru Kurosawa, Wakaha Ogata, *Efficient Rabin-type digital signature scheme*, Designs, Codes and Cryptography **16** (1999), 53–64. ISSN 0925–1022. Citations in this document: §6, §6, §6, §6, §6, §6.
28. Ueli M. Maurer (editor), *Advances in cryptology—EUROCRYPT '96: Proceedings of the Fifteenth International Conference on the Theory and Application of Cryptographic Techniques held in Saragossa, May 12–16, 1996*, Lecture Notes in

Computer Science, 1070, Springer-Verlag, Berlin, 1996. ISBN 3–540–61186–X. MR 97g:94002. See [5].

29. Tatsuaki Okamoto (editor), *Topics in cryptology—CT-RSA 2004: the cryptographers' track at the RSA Conference 2004, San Francisco, CA, USA, February 23–27, 2004, proceedings*, Lecture Notes in Computer Science, Springer, Berlin, 2004. ISBN 3–540–20996–4. MR 2005d:94157. See [11].
30. Michael O. Rabin, *Digitalized signatures and public-key functions as intractable as factorization*, Technical Report 212, MIT Laboratory for Computer Science, 1979. URL: `http://ncstrl.mit.edu/Dienst/UI/2.0/Describe/ncstrl.mit_lcs/MIT/LCS/TR-212`. Citations in this document: §2, §2, §3, §4, §5.
31. Ronald L. Rivest, Adi Shamir, Leonard M. Adleman, *A method for obtaining digital signatures and public-key cryptosystems*, Communications of the ACM **21** (1978), 120–126. ISSN 0001–0782. URL: `http://cr.yp.to/bib/entries.html#1978/rivest`. Citations in this document: §2, §2, §3, §3.
32. Douglas R. Stinson, *Cryptography: theory and practice*, CRC Press, Boca Raton, Florida, 1995. ISBN 0–8493–8521–0. MR 96k:94015. Citations in this document: §2.
33. John Wigley, *Removing need for rng in signatures*, message `5gov5d$pad@wapping.ecs.soton.ac.uk` posted to `sci.crypt` (1997). URL: `http://groups.google.com/group/sci.crypt/msg/238e1fbc2ea66e6b`. Citations in this document: §4.
34. Hugh C. Williams, *A modification of the RSA public-key encryption procedure*, IEEE Transactions on Information Theory **26** (1980), 726–729. ISSN 0018–9448. URL: `http://cr.yp.to/bib/entries.html#1980/williams`. Citations in this document: §6.
35. Moti Yung (editor), *Advances in cryptology—CRYPTO 2002: 22nd annual international cryptology conference, Santa Barbara, California, USA, August 2002, proceedings*, Lecture Notes in Computer Science, 2442, Springer-Verlag, Berlin, 2002. ISBN 3–540–44050–X. See [14].