Lattice attacks on RSA-encrypted IP and TCP

P.A. Crouch & J.H. Davenport*

Department of Mathematical Sciences, University of Bath, Bath BA2 7AY, England
Paul@p-crouch.com J.H.Davenport@bath.ac.uk

Abstract. We introduce a hypothetical situation in which low-exponent RSA is used to encrypt IP packets, TCP segments, or TCP segments carried in IP packets. In this scenario, we explore how the Coppersmith/Howgrave-Graham method can be used, in conjunction with the TCP and IP protocols, to decrypt specific packets when they get re-transmitted (due to a denial-of-service attack on the receiver's side). We draw conclusions on the applicability of the Coppersmith/Howgrave-Graham method, its interaction with "guessing", and the difficulties of building a secure system by combining well-known building blocks.

1 Introduction

We consider a scenario in which there are many, internally secure, TCP/IP networks. These communicate across an insecure internet. To enable secure communication, the firewalls for each secure network take the $\mathrm{IP}_{\mathrm{sec}}$ packet from the secure network, encrypt it with RSA [2] and treat the encrypted packet as data for an $\mathrm{IP}_{\mathrm{insec}}$ packet directed at the firewall of the destination network. This then decrypts the packet and injects it into its secure network, as shown in figure 1.

From the point of view of the $\mathrm{IP}_{\mathrm{sec}}$ layers, the firewalls and the $\mathrm{IP}_{\mathrm{insec}}$ communication all form a (complex) link layer over which the $\mathrm{IP}_{\mathrm{sec}}$ packet travels.

The attacker is assumed to listen to the $\mathrm{IP}_{\mathrm{insec}}$ packets transmitted across the insecure internet from the sender (A) to the receiver (B). By flooding B, or a switch near B, with other traffic, the attacker can (at least with high probability) cause B to miss a transmission from A. IP is inherently an unreliable protocol, so the higher levels above IP (e.g. TCP) will have mechanisms to re-transmit the lost message. What information can the attacker gain in this scenario?

The Coppersmith/Howgrave-Graham method [4, 7, 8, 11] is encapsulated in the following theorem.

Theorem 1. *Let P be a monic polynomial of degree δ in one variable modulo an integer N (of unknown factorisation). Then one can find, in time polynomial in $(\log N, \delta)$, all integers x_0 such that $P(x_0) \equiv 0 \pmod N$ and $|x_0| \le N^{1/\delta}$.*

The method forms an $h\delta \times h\delta$ lattice, where h is the control parameter. To reach $N^{1/\delta}$, one needs h to be arbitrarily large, and the actual bound on x_0 achieved varies with h, as shown in table 1 for the case of $\delta = 3$.

* The authors are grateful to Dr. D. Coppersmith, Dr. N.A. Howgrave-Graham and Mr. A.J. Holt for their contributions to this work.

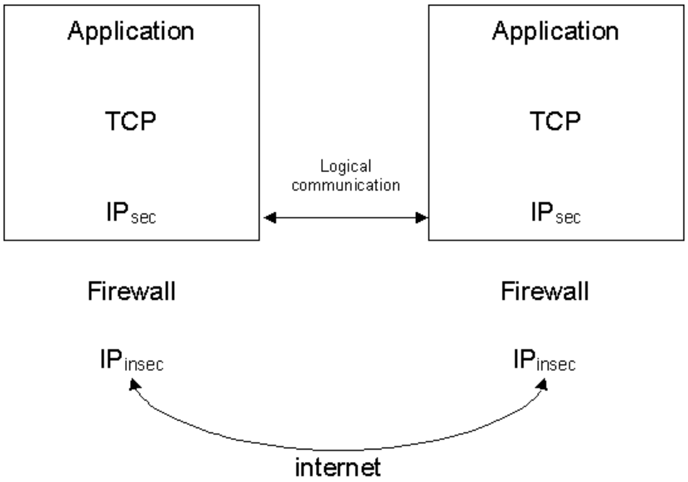

Fig. 1. Protocol layers in the scenario

Table 1. x_0 as a function of h; $\delta = 3$

$h=2$	$h=3$	$h=4$	$h=5$	$h=6$	$h=7$	...	$h=67$
$N^{0.2}$	$N^{0.25}$	$N^{0.27}$	$N^{0.286}$	$N^{0.294}$	$N^{0.3}$	...	$N^{0.33}$

In this paper, we are concerned with recovering bit-fields from IP packets, generally 16-bit fields. Table 1 shows how many bits we can expect to recover for various values of h in various scenarios for the size of the RSA modulus (512, 1024 or 2048 bits), choices of the RSA exponent, and the presence or absence of checksum wrapping (as explained later).

Table 2. Values for ϕ for recovering $x_0 \leq 2^{\phi}$

	$e=3$						$e=5$					
	No CS Wrap			CS Wrap			No CS Wrap			CS Wrap		
	512	1024	2048	512	1024	2048	512	1024	2048	512	1024	2048
$h=2$	**34.13**	**68.27**	**136.53**	**30.12**	**60.24**	**120.47**	11.38	**22.76**	**45.51**	10.44	**20.89**	**41.8**
$h=3$	42.66	85.33	170.67	39.38	78.77	157.54	14.63	29.26	58.51	13.84	27.68	55.35
$h=4$	46.54	93.09	186.18	43.89	87.78	175.54	**16.17**	32.34	64.67	15.52	31.03	62.06
$h=5$	48.76	97.52	195.04	46.55	93.09	186.18	17.07	34.14	68.26	**16.52**	33.03	66.06
$h=6$	50.19	100.39	200.78	48.3	96.60	193.20	17.66	35.31	70.62	17.18	34.36	68.72

The complexity of the lattice reduction, which is the dominant step in the Coppersmith/Howgrave-Graham method, is

$$O\left(h^9\delta^6(\log N)^3\right), \tag{1}$$

assuming classical arithmetic. From this and table 1, we see the importance of minimising both h and δ.

Most of the computation for this paper was done in `Maple` but Victor Shoup's `NTL`[12] library for C++ was used for the time-critical lattice reduction part.

2 IP packets

Figure 2 shows a diagram of an IP packet. To understand the attacks on IP datagrams, we need to understand the IP datagram header. A good general reference on TCP and IP is [13].

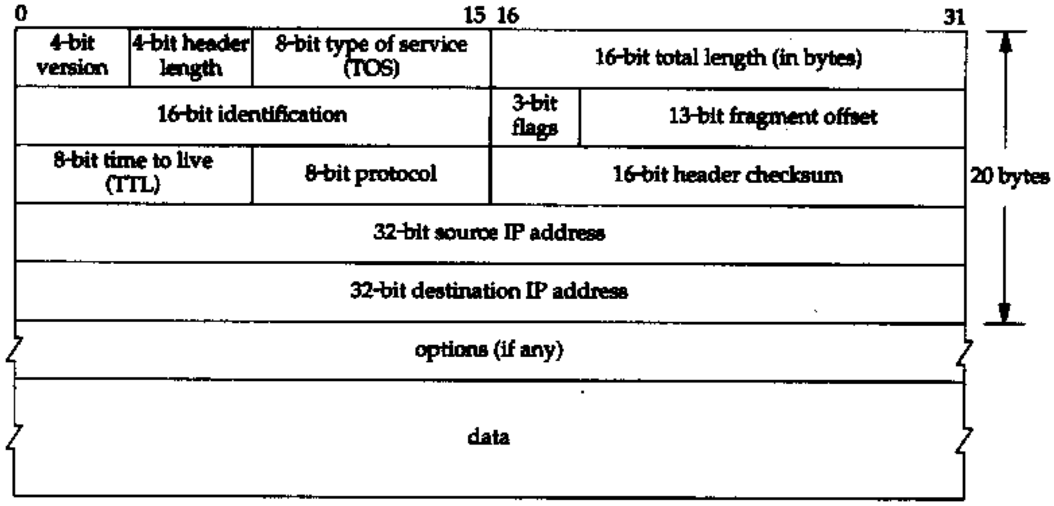

Fig. 2. IP datagram, showing the fields in the IP header

In the denial-of-service scenario mentioned in the introduction, the higher protocols above IP would cause a re-transmission. The IP layer is unaware that this is a re-transmission, and sends this as a new packet. What fields change between this re-transmission and the original transmission? The only two fields that we expect to change are the following.

- The 16-bit identification field. This normally increases by 1 every time the IP layer sends a packet.

– The one's complement sum of the header, stored in the 16-bit checksum field. Every time the 16-bit identification field is incremented, the checksum is decremented. Care must be taken if the checksum exceeds 65535, as it will then wrap around and restart from 1.

3 Attacks on IP packets

First we need to take the denial of service attack and look at it in more detail. In the denial of service attack letting

$$c_1 = m^e$$

$$c_2 = (m + x)^e$$

gives us a basic idea of how an attack could be implemented. However we are more specifically interested in what two IP packets would look like when represented as c_1 and c_2, and the $\mathrm{IP_{sec}}$ identification fields differ by α. Using the knowledge about IP packets it is possible to construct a similar general formula, which is shown and explained below.

$$c_1 = m^e$$

$$c_2 = (m + (\overbrace{2^{48}\alpha - \alpha}^{1} \underbrace{\pm 65535}_{2}) \times \overbrace{2^{64+d}}^{3})^e$$

1. This is because the id field is 48 bits from the checksum field and increases by 1 each time a packet is sent. The $-\alpha$ is because when the id field increments by 1, the checksum decrements by 1.
2. The ± 65535 only applies if the $\mathrm{IP_{sec}}$ checksum wraps around, something which cannot be determined from the encrypted packets. If the checksum does not wrap around between the two packets then this value is 0.
3. This 2^{64+d} is derived from the checksum being 64-bits (32 bit source address and 32 bit destination address) from the end of the packet $+d$ which is the number of **bits** of data which are appended to the end of the packet.

Next two RSA encrypted versions of the same (apart from identification and checksum) $\mathrm{IP_{sec}}$ packets are required. Define these $\mathrm{IP_{insec}}$ data fields to be eip_1 and eip_2. These would correspond to the $\mathrm{IP_{sec}}$ packets ip_1 and ip_2, and would have been encrypted by taking $eip_j = ip_j^e \mod N$.

Letting c_1 and c_2 be the symbolic forms of the two encrypted packets eip_1 and eip_2, it is possible to calculate resultants. This is done by taking the resultants of $c_1 - eip_1$ and $c_2 - eip_2$, i.e. of $m^e - eip_1$ and $(m + x)^e - eip_2$, with respect to m.

Taking resultants will give a polynomial in α of degree e^2 (though it may be possible to reduce this to e). This is the polynomial satisfied by the unknown, but comparatively small, α. We use the Coppersmith/Howgrave-Graham method [4, 7, 8] to find α, by solving the appropriate lattice.

Once α is known, we take the greatest common divisor with respect to z of

$$\gcd(z^e - eip_1, (z + \alpha \times (2^{48} - 1) \times 2^{64+d})^e - eip_2) \mod N) = z + ip_1 + \lambda N$$

where z is an indeterminate, and λ represents the fact that we are interested in $ip_1 \pmod N$ (in practice $\lambda = 1$). This gives

$$z - (\gcd(z^e - eip_1, (z + (2^{48}\alpha - \alpha) \times 2^{64+d})^e - eip_2) \mod N) - \lambda N = ip_1$$

Therefore we have recovered ip_1 and broken the RSA encryption on these particular IP packets.

Symbolically, in the absence of checksum wrapping, and ignoring for simplicity the large numerical factors, the equation for α is of the following form.

$$\alpha^9 + 3(c_1 - c_2)\alpha^6 + 3(c_1^2 + 7c_1c_2 + c_2^2)\alpha^3 + (c_1 - c_2)^3 \equiv 0 \pmod N$$

While this is of degree 9 in α, which would imply reducing an 18×18 lattice with $h = 2$, it is in fact a polynomial of degree 3 in α^3, and solving it as such only requires reducing a 6×6 lattice, giving a $3^6 = 729$ theoretical saving[1]. Unfortunately, if the checksum does wrap, this simplification is not possible.

Shown below is a summary of the timings in seconds to perform the lattice reduction phase of this attack for various sizes of the public-key modulus..

Table 3. Times for IP reduction

NTL Timings in seconds to lattice reduce RedHat Linux 6.2 on 1Ghz Pentium III with 500Mb RAM							
Public exponent		e=3			e=5		
RSA-type		512	1024	2048	512	1024	2048
h	(control parameter)	2	2	2	4	2	2
IP	No checksum wrapping	2	9	27	8068*	177	1386
	With checksum wrapping	653	3413	3976	†	793465	§

† Not implemented due to software restrictions. This would in fact have required $h = 5$.

§ Not implemented in this report, due to the running time & resource constraints.

$*$ Taking $\alpha \leq 2^{11}$ allowed $h = 2$, with $e = 5$ this formed a 10x10 matrix which reduced in 19 seconds. Therefore guessing all possibilities for the top five bits of a 16-bit identification field and using the Coppersmith/Howgrave-Graham method to find the bottom 11 bits would take $2^5 * 19 = 608$ seconds: over a factor of 10 faster than direct solving.

[1] The same happens for $e = 5$, and in fact we also save on h, being able to take $h = 4$ rather than $h = 5$, as can be seen from table 1.

Without checksum wrapping, and with $h = 2$, the $e = 3$ examples required a 6×6 lattice, and $e = 5$ required 10×10. With checksum wrapping and $h = 2$, the $e = 3$ examples required a 18×18 lattice, and $e = 5$ required 50×50. In general, we note that doubling the length of the modulus multiplies the running time by about 8, which is what one would expect from the theoretical performance (as given in equation (1)) of the LLL algorithm [9].

4 TCP/IP sessions

TCP (Transmission Control Protocol) is a common protocol to use above IP. It provides a reliable connection-oriented service on top of the unreliable service provided by IP. In the scenario in Figure 1, if an opening (i.e. carrying the

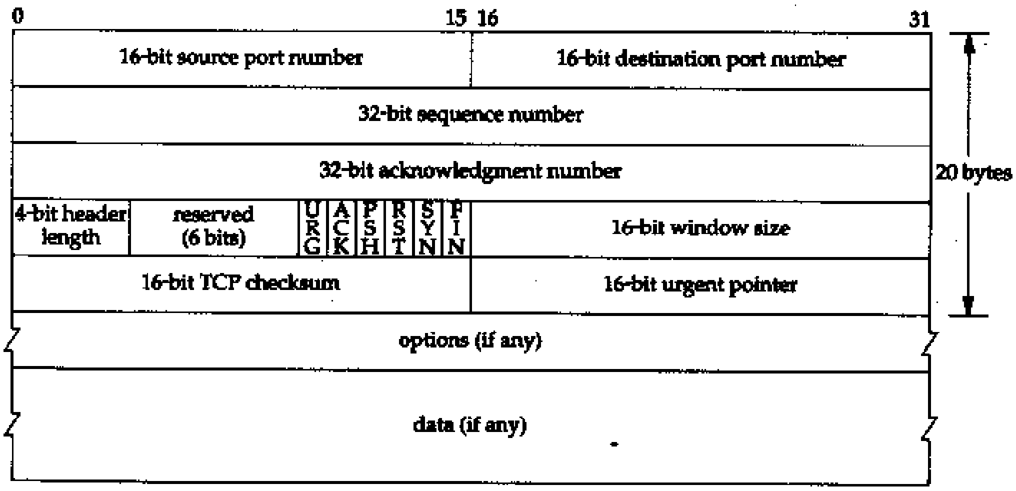

Fig. 3. TCP segment, showing the fields in the TCP header

`SYN` flag) TCP packet is lost, the TCP packet will simply be re-transmitted unchanged. In this case, the attack in the previous session is the only one possible. However, it is possible to deny service for the entire TCP opening sequence, in which case protocols above TCP may well start a new TCP session, which will have its own sequence number.

In a denial of service attack we are interested in what would change in a TCP packet header, shown in figure 3. This is slightly more complex because the TCP checksum is calculated (as in the case of the IP checksum) as a one's complement sum of **16-bit** words. In the case of TCP, the field that changes is the sequence number. Unfortunately this is a 32-bit field. Therefore although we know that the sequence number in many implementations (incorrectly: see [1, 3, 10]) simply increments by 64000 every half second, the effect on the checksum is rather different.

If the 32-bit sequence number increments by 64000β, the high 16-bits in the 32-sequence number normally[2] increase by β and the low 16-bits decrease by 1536β so the overall change in the checksum (as it is the one's complement sum) is $+1535\beta$. This is valid only if $\beta \leq 2^{16}$ (otherwise the 32-bit nature of the field manifests itself). We assume that $\beta \not> 2^{16}$.

5 Attacks on TCP/IP sessions

We consider two kinds of attacks: those where we assume the TCP packet alone is encrypted (i.e. the encryption takes place between TCP and IP, rather than below IP_{sec} as in figure 1) and those where the complete IP_{sec} packet is encrypted, in accordance with figure 1. These are referred to as TCP and TCP/IP in table 5. The TCP attack is similar to the IP attack: we are solving for one variable, β, and the equation to be solved is of degree e^2, which can be reduced to degree e in the event of no checksum wrapping.

Table 4. Times for TCP reduction

NTL Timings in seconds to lattice reduce RedHat Linux 6.2 on 1Ghz Pentium III with 500Mb RAM							
Public exponent		e=3			e=5		
RSA-type		512	1024	2048	512	1024	2048
h	(control parameter)	2	2	2	4	2	2
TCP	No checksum wrapping	1	9	27	9456	167	1384
	With checksum wrapping	702	4631	5129	†	§	§
TCP/IP	No checksum wrapping	Ψ	§	§	§	§	§

† Not implemented due to software restrictions. This would in fact have required $h = 5$.
Ψ Computation aborted after one month.
§ Not implemented due to the running time & resource constraints.

The TCP/IP attack is more complicated, as we are solving for two variables, α (the change in IP identification field) and β. From a polynomial point of view, this forms a bivariate modular equation. As pointed out by Coppersmith [4] and Howgrave-Graham [7, 8], there is a heuristic extension of their method to bivariates. Let us assume that the IP_{sec} packets are $tcpip_1$ and $tcpip_2$, with one byte of TCP data. Taking $etcpip_1$ and $etcpip_2$ to be $tcpip_1$ and $tcpip_2$ encrypted

[2] That is to say, when the low-order part of the sequence number is greater than 1536β, which one might expect to occur $1 - \frac{1536\beta}{65536}$ of the time.

with RSA, resultants were taken using them and c_1 and c_2 shown below, which formed a bivariate modular polynomial.

$$c_1 = m^3$$

$$c_2 = (m + \underbrace{\underbrace{((2^{48}-1)\alpha\, \overbrace{2^{232}}^{\Delta})}_{\text{IP packet}} + \underbrace{((2^{80} \times (64000\beta) + (1535\beta)) \times 2^{24})}_{\text{TCP packet}}}_{\text{TCP/IP packet}})^3$$

Δ - The IP checksum field is now 232 **bits** from the end of the packet as it is 64 bits from the end of the IP header, 160 bits (20 bytes) of TCP header and 8 bits of data. 64+160+8=232

As this is an $e = 3$ case we have already attacked IP and TCP with $e = 3$ so it is safe to conclude that $h = 2$. Therefore the polynomial is never raised to a greater degree than 1.

We obtained a polynomial in α and β where the maximum degree of β is 9. This polynomial was made monic with respect to β^9. Then using the formula to calculate the dimensions of the matrix where $j = 9$, showed that a 190x190 matrix was required.

$$dim = 2j^2 + 3j + 1 = 2 \times 9^2 + 3 \times 9 + 1 = 162 + 27 + 1 = 190$$

It is possible to calculate the number of rows which must contain N and the number of rows which must contain the coefficients of $p(x)$,

$$190 - \frac{j^2 + 3j + 2}{2} = 190 - \frac{110}{2} = 135$$

So 135 rows of the matrix contain N's (scaled by the bounds for α and β) on the diagonal (and zeros elsewhere) and the remaining 55 rows contain the coefficients of the polynomial multiplied by the different combinations of the two variables α and β and scaled by the bounds for α and β, so that the leading term is on the diagonal.

The lattice reduction is extremely costly in time and resources, and at the time of finishing [5], the 190x190 lattice reduction process had been running for in excess of 750 hours.

In theory though, if the LLL reduction had been successful on this large matrix, we would have taken the first two short vectors of the LLL reduced matrix, divided them both by the numeric vector (of upper bounds), and formed two simultaneous equations. These two simultaneous equations would be solved to return α and β.

Next with α and β in hand, and $etcpip_1$ and $etcpip_2$ corresponding to the two encrypted messages, we would calculate the linear polynomial

$$\gcd(z^3 - ep_1, (z + (2^{48}-1)\alpha 2^{232} + (2^{80} \times 64000 + 1535)\beta \times 2^{24})^3 - ep_2) \mod N)$$

to recover $tcpip_1$.

6 TCP/IP attacks revisited

A better way to attack TCP/IP packets is by "'guessing" β. α represents the change in the IP identification field, as this increments by 1 every time a packet is sent from the original sender (including packets internal to the sender's $\mathrm{IP}_{\mathrm{sec}}$ network, which cannot be detected outside), guessing this would be essentially impossible due to the number of packets sent. However β increments by 1 every half second. Remembering that we are performing a denial of service attack, if we, for example, sniffed two packets in the space of 4 seconds, there would be only eight β's to guess.

After experimentation with $e = 3$ and substituting $\beta = \{1, 2, 3, 4, 5, 6, 7, 8\}$ it was clear to see from the resultant polynomial that this was a univariate polynomial in α with maximum degree 9. Unfortunately this could not be solved as a degree 3 polynomial in α^3, but running eight 18x18 lattice reductions takes significantly less time ($8 \times 680 = 5440$ seconds) than attempting to LLL reduce the 190x190 matrix.

Taking

$$c2 := (m + ((2^{48}\alpha - \alpha) \times 2^{232}) + ((2^{80} \times (64000\beta) + (1535\beta)) \times 2^{24}))^3$$

But then for example guessing $\beta = 1$ gives:

$$c2 := (m + ((2^{48}\alpha - \alpha) \times 2^{232}) + 1329207713375312221233383113029058560)^3$$

We observe empirically that **lattice reducing a matrix with an unsuccessful guess of β takes no longer than to reduce a successful guess of β.**

A polynomial is constructed and solved using LLL reduction as in the IP case with checksum wraparound to resolve α, and now with α calculated and β guessed,

$$\begin{aligned}\gcd(\ & z^3 - etcpip_1, \\ & (z + (2^{48} - 1)\alpha 2^{232} + (2^{80} \times 64000 + 1535)\beta \times 2^{24})^3 - etcpip_2) \mod N)\end{aligned}$$

is calculated, which is equal to $z - tcpip_1 - \lambda N$, hence RSA encryption on these TCP/IP packets is broken.

7 Conclusions

Table 1 shows that the Coppersmith/Howgrave-Graham method is eminently applicable for the decoding of low-exponent RSA-encrypted IP packets. For $e = 5$ we also have the paradoxical result that increasing the key length from 512 bits to 1024 actually reduces the time by a factor of 45, since a smaller lattice (10×10 rather than 20×20, with entries modulo N rather than N^3) is involved.

Extrapolating from table 3 shows that the key size would have to rise to 4096 bits before security is improved.

Yet again, we note that the common, but flawed, implementation of TCP initial sequence numbers is a security loophole [1, 3, 10]. In our case, the fact that $64000 \approx 2^{16}$ is an added weakness.

We also note the power of combining guessing some bits with the Coppersmith/Howgrave-Graham method for finding others: see note (*) in IP attacks and section 6.

More generally, we have shown that a cryptosystem built according to standard principles of protocol layering with "standard" components displays unexpected, and in some cases computationally trivial, vulnerabilities.

References

1. Bellovin, S., Defending Against Sequence Number Attacks. Internet RFC 1948, May 1996.
2. Boneh, D., Twenty Years of Attacks on the RSA Cryptosystem. *Notices of the AMS* **46** (1998) pp. 203–213. `http://crypto.stanford.edu/~dabo/papers/RSA.ps`
3. Braden, R. (Ed.), Requirements for Internet Hosts — Communication Layers. Internet RFC 1122, October 1989.
4. Coppersmith, D., Small solutions to polynomial equations, and low exponent RSA vulnerabilities. *J. Cryptology* **10** (1997) pp. 233–260.
5. Crouch, P.A., *A small public exponent RSA attack on TCP/IP packets*. Project, University of Bath Department of Mathematical Sciences, May 2001. `http://www.p-crouch.com/rsa-tcpip`.
6. Davenport, J.H., Lecture notes at LMS Durham Symposium. `http://www.bath.ac.uk/~masjhd/Durham.{dvi,ps,pdf}`
7. Howgrave-Graham, N.A., Finding Small Roots of Univariate Modular Equations Revisited. *Cryptography and Coding* (Ed. M. Darnell), Springer Lecture Notes in Computer Science 1355, 1997, pp. 131–142.
8. Howgrave-Graham, N.A., *Computational Mathematics inspired by RSA*. Ph.D. Thesis, University of Bath, 1998.
9. Lenstra, A. Lenstra, H. Lovász. Factoring Polynomials with Rational Coefficients. *Mathematische Annalen* **261** (1982) pp. 515–534. Zbl. 488.12001. MR 84a:12002.
10. Morris, R.T., A Weakness in the 4.2BSD Unix TCP/IP Software. Computing Science Technical Report 117, AT&T Bell Laboratories, Murray Hill, New Jersey, 1985.
11. Nguyen, S. and Stern, J., Lattice Reduction in Cryptography: An update. Proc. ANTS-IV (ed. W. Bosma), Springer Lecture Notes in Computer Science 1838, Springer-Verlag, 2000, pp. 85–112. Updated at `http://www.di.ens.fr/~pnguyen/pub`.
12. Shoup, V. NTL (Number Theory Library) for C++. `http://www.shoup.net`.
13. Stevens, W.R., *TCP/IP Illustrated, Volume 1*. Addison Wesley, 1994.