The Advanced Encryption Standard (AES): The Successor of DES

Dr Reinhard Wobst

***Rijndael**, the Belgian cryptographic algorithm is the new cryptographic standard in the USA. The US's adoption of this algorithm was no overnight matter. This paper describes this algorithm as well as the history of some of its predecessors and competitors.*

DES

For many years the well-known cipher DES (Digital Encryption Standard) was the cryptographic standard for unclassified use within the USA. DES has been applied in banks as well as in many software- and hardware products for more than 20 years. One could say that this algorithm is the most widespread cipher at all. Some experts warned that some trapdoor might be built in because the secret service NSA was consulted during the design. Such a trapdoor was never found, although significant theoretical weaknesses which could hardly be exploited in practice were identified shortly after DES was adopted.

The weak point of DES is not its design, but its key size of only 56 bit. This corresponds to about 72,000,000,000,000,000 possible keys. In the seventies (when DES was introduced), this was an astronomical number. Meanwhile hardware has become very fast. In summer 1998 an organisation named EFF built and demonstrated a special computer, *Deep Crack*, that could decrypt a DES-enciphered text within an average time of 4.5 days[1]. Crypto experts were not surprised, but politically it was a shame: As late as in February 1998 an "expert" explained to the U.S. Congress that DES was practically unbreakable.

3DES

The business community did not believe this, however, and instead started to apply the more

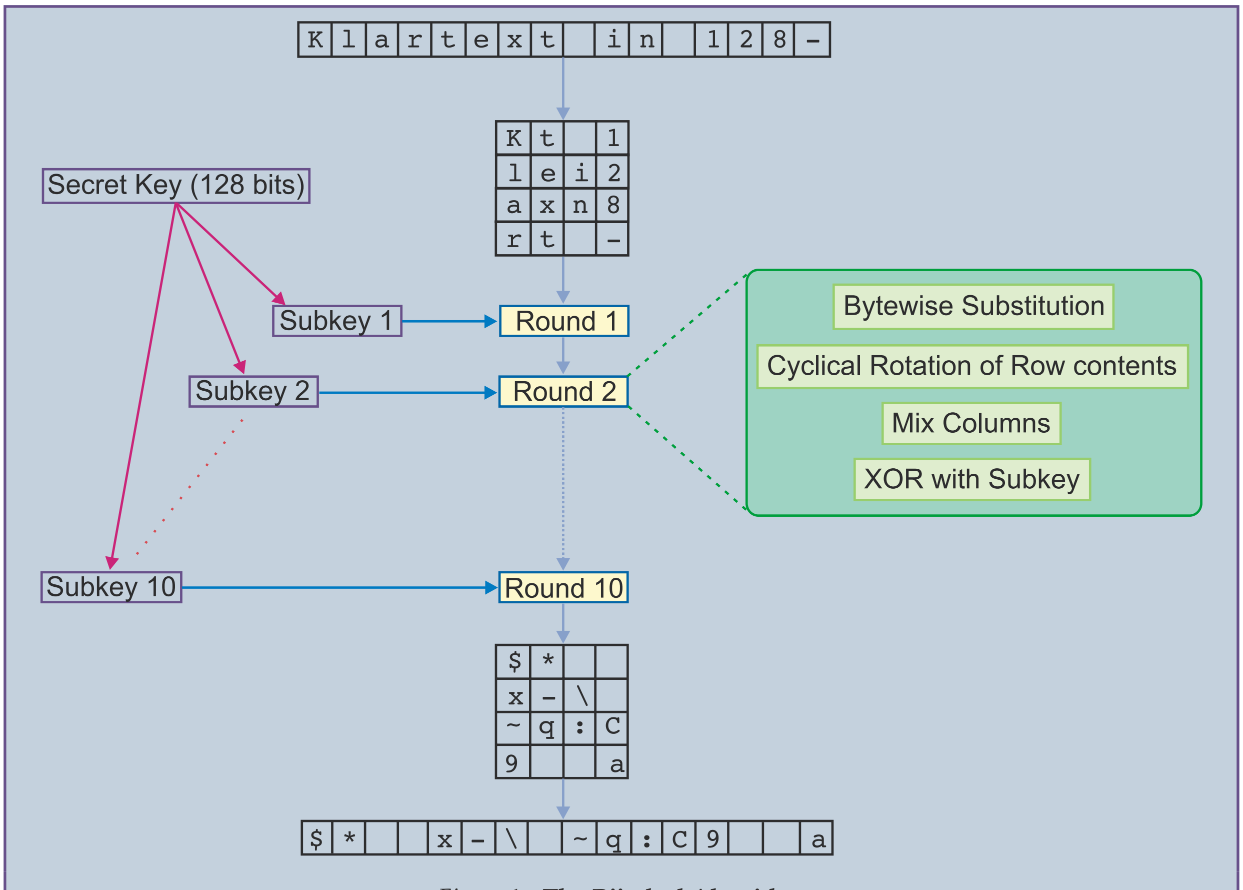

Figure 1 - **The Rijndael Algorithm**

secure Triple-DES (3DES) algorithm. 3DES is a threefold DES encryption in which two 56-bit keys are applied. That means a practical security of 112 bits in strength which, as far we know today, should be reasonably secure. Since 1998 the PIN of ATM cards (at least in Germany) is probably computed by 3DES.

Fundamental Considerations

Some readers might argue that many crypto products claim 1024-bit security. Why then still a discussion about 56- or 112-bit encryption? This type of discussion represents a serious misunderstanding. DES is a symmetric cipher, i.e. encryption and decryption use the same secret key. This type of algorithm is always the base for any secure data exchange! So-called asymmetric algorithms such as RSA are only used for secure exchange of secret keys; they are not applicable to conventional encryption. Such asymmetric algorithms need far longer keys than symmetric ciphers to gain the same security, for instance 512, 1024 or 2048 bits[5, 6].

Most symmetric ciphers are block algorithms. They encrypt plaintext in portions, say 64 bits of plaintext to 64 bits of cipher text. Most block algorithms are so-called *product ciphers*. They execute many rounds, i.e., (almost) the same transformation is applied again and again. Each transformation depends on some secret *subkey* that is derived from the secret cipher key and specific to a given round. Thus, each round uses a different subkey. DES as an example has a 56-bit cipher key, 64-bit blocks and 16 rounds (and thus 16 subkeys).

DES is now outdated. Even 3DES does not fulfill the requirements of a modern algorithm

- it must be suited for implementation in software as well as in hardware. (DES is functionally only hardware-friendly.)
- it must have variable key and block sizes and
- it must be able to resist all known cryptanalytical attacks, even if these can not be used in practice yet.

These facts were also known to the US National Institute of Standards and Technology (NIST), which adopted DES. Until the recent adoption of Rijndael, every 5 years NIST confirmed DES as the standard cipher. In the beginning of 1997 NIST formulated the call for a new algorithm with the name “AES” (Advanced Encryption Standard). It was clear that this process would take a long time: Still more than the *design* of a cipher, its *analysis* is an extraordinary difficult task, except if the cipher is obviously weak. To say it frankly, this is an unsolvable task. One can not prove that some algorithm is secure (with a single exception of almost no practical interest); one can only show that *known* attack methods do not work. Nevertheless, cryptographers have to take this challenge.

Box 1 - AES Requirements

These requirements where formulated by NIST after public discussion, among others during a workshop at 15 April 1997. Here are only some criteria:

It must be a symmetric cipher (the same key is used for encryption and decryption), more precisely a block cipher. (The public cryptanalysis of so-called stream ciphers seems not to be so well developed).

The block size must be at least 128 bits; keys of 128, 192 and 256 bits must be applicable.

AES must be well suited to different purposes, thus is must be easy to implement in both hard and software and perform well in both cases.

AES must be able to resist all known methods of cryptanalysis.

Especially, AES must be resistant against timing and power attacks.

For the use inside smartcards in particular, it must be able to work with limited resource availability (short code, small memory).

There must be no patent right on the algorithm. It must be free to use for everybody.

A remark concerning the demanded key and block sizes is appropriate here. During a public discussion in Germany some years ago, a legal expert declared that trying out all possible keys is only a question of “dedication and industriousness”. The example of a 256-bit key shows exactly how “dedication” is really needed: Suppose, for instance, that you would store all 2^{256} possible keys and only one atom were used for each key (currently nobody knows how to do that!). The required memory would have a mass of about 10^{47} grammes. This is about 10^{14} times the weight of a typical star. As Bruce Schneier has already mentioned, in such cases the storage medium would create a black hole and thus never let out any information.

For 128-bit keys this memory would only have a mass of about 300 tons. It would still store the incredible amount of 10^{24} TB. We did not speak yet about the CPU to perform the necessary calculations. However, for AES such thought experiments are no motivation. More realistic dangers are first that the still hypothetical quantum computers could reach a new quality in parallelizing attacks. Secondly, it is always possible that AES could be attacked more elegantly than with brute force (i.e., trying all possible keys). In this case we need a large safety margin.

By the way, 3DES could never be an AES candidate because of its “short” key size of 112 bits.

Box 2
A Chronology of the NIST AES Initiative

2.1.97: Call for algorithms, accepting proposals until 12.9.97.

5.4.97: During a public AES workshop, the detailed requirements are formulated.

20.8.98: First AES Conference. NIST announces the receipt of 15 algorithms, sent by cryptographers from all over the world. The public evaluation starts.

March 99: Second AES conference. Discussion of obtained results. 28 publications had been submitted before and were put on the NIST homepage to make conference discussions more effective.

15.4.99: End of public evaluation of all candidates. Five candidates (MARS, RC6, Rijndael, Serpent, Twofish) are in the next round. Further work will be concentrated on these algorithms from now.

13./14.4.00: Third AES Conference. The analysis of the final five candidates are presented and discussed.

15.5.00: End of public discussion.

2.10.00: The "winner," Rijndael, is announced.

November 2000: The FIPS standard is published as a draft. Public comments are possible.

February 2001: End of the public discussion of the standard.

April-June 2001: Confirmation of the FIPS standard.

Since cryptanalysis has been strongly developed, particularly in the past decade, we have a rich experience in cipher evaluation today.

How to create an algorithm?

Experts have said it over and over again: *Only a publicly analysed algorithm can be secure*. Ciphers with secret designs are almost always weak. There is an intrinsic example for this: The algorithms A3 and A5 used in GSM mobile phones were disclosed not long after their application. A3 is responsible for authentication. After only one day of analysis it was broken: Mobile phones can be cloned in practice now (i.e., you can create your own SIM card with a false number and use it charging someone else). The A5 analysis took somewhat longer, but now it suffices to intercept and store 2 minutes of talk; then merely one second of computation time on a big PC is needed to crack the code[7].

Even if an algorithm is known it should be clear why it was designed just this way. If this is not clear, users will distrust it. The best example is DES: The modification of the so-called "S-boxes" (presumably by NSA) caused the suspicion that some trapdoor was built in. Some design principles were published years later, but it was too late. However, the AES election procedure was really open: Anybody could propose a cipher, the best cryptologists in the world analyzed it. And the design principles were also published.

AES Requirements

An algorithm is considered to be secure if it can only be cracked by trying all possible keys. This

Box 3 - Submitted algorithms

Round 1:
CAST-256: Entrust Technologies, Inc.

CRYPTON: Future Systems, Inc.

DEAL: Richard Outerbridge, Lars Knudsen

DFC: CNRS Centre National pour la Recherche Scientifique - Ecole Normale Superieure

E2: NTT - Nippon Telegraph and Telephone Corporation

FROG: TecApro Internacional S.A.

HPC: Rich Schroeppel

LOKI97: Lawrie Brown, Josef Pieprzyk, Jennifer Seberry

MAGENTA: Deutsche Telekom AG

MARS: IBM

RC6: RSA Laboratories

RIJNDAEL: Joan Daemen, Vincent Rijmen

SAFER+: Cylink Corporation

SERPENT: Ross Anderson, Eli Biham, Lars Knudsen

TWOFISH: Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, Niels Ferguson

Round 2 finalists:
MARS, RC6, Rijndael, Serpent, Twofish

Selected after Round 3:
Rijndael

is practically impossible for a 256- or even 128-bit long key, as shown in Box 1 in which some of the criteria which must be fulfilled by AES, are listed.) The block size is something very different from key size. The reason that larger block sizes than 64-bits are demanded is the following:

If AES is used in CBC (Cipher Block Chaining) mode, i.e. the n+1-st ciphertext C_{n+1} is generated from the n-th ciphertext and the n-th plaintext P_n according to

$$C_{n+1} = AES(C_n \oplus P_{n+1})$$

($\oplus$ denotes bitwise XOR), and two blocks C_{m+1} and C_{n+1} are coincidentally identical,

it then follows that

$$C_m \oplus P_m = C_n \oplus P_n$$

or

$$P_m \oplus P_n = C_m \oplus C_n.$$

Any attacker can see which ciphertext blocks agree and thus compute the XOR product $P_m \oplus P_n$. So as yet he has not got the plaintext itself, but he has obtained some possibly meaningful information which can be used to draw conclusions about the cleartext. Remember that the death of Julius and Ethel Rosenberg finally was caused by such an "XOR analysis" done under the so-called Venona project of NSA![6] It is a bit paranoid to be afraid of disclosing the XOR product of 64 bits only. But when a new algorithm is designed, the least possible security hole should be closed in advance. In the mean, only 2 of 2^{32} ciphertexts will agree when 64-bit blocks are used. This is a quantity of about 34 GB text. But who will know what data quantities will be encrypted after 20 years? In the case of 128-bit blocks a block is repeated after about 140 million Terabytes - this should suffice for the next 20 years.

The Selection Procedure

The NIST received 15 proposals for algorithms. After thorough investigation, five candidates remained. The best cryptanalysts of the world dealt with them. Now a pat situation arose: All five algorithms were excellent and were also very hard to compare. Each of them could have easily become the new standard; none of them had any significant weak point. Each of them had some advantages over the others - but what properties should be the most important?

During the Third AES conference Don B. Johnson (Certicom) asked in two lectures: *Must there be a best algorithm at all?* Modern software anyway implements some normed crypto API and offers several algorithms in parallel. The five AES candidates are small enough to be all contained in one product. Of course, the situation for hardware is not so convenient. So it would be prudent to prefer different algorithms for different purposes in hardware (smartcards, online enciphering, ...) and to offer all five in software. Such a flexibility would have more advantages than disadvantages.

Products that are fixed on some algorithm would become suddenly insecure if a weakness of this algorithms is discovered in spite of all expectations. Remember the use of DES in banks? It was applied for about 20 years. The switch to 3DES took several years and consumed huge amounts of money.

The Adoption of Rijndael

Ultimately, NIST decided differently from what most people expected - only one algorithm was the winner, Rijndael, the Belgian algorithm developed by Joan Daemen and Viincent Rijmen. The official reasons for this decision (which do not convince everyone) are as follows:

- Should a weakness of Rijndael be discovered, larger key sizes will give more security.
- In the worst case, 3DES could be an alternative for some time (3DES will remain secure in the near future).

Box 4 - Performance of Rijndael in hardware and software

In this box only the speeds for 128-bit blocks and 128-bit keys are given. As mentioned in the text, the decryption speed on 8-bit CPUs may be up to 30% slower than encryption speed.

Hardware:

Intel 8051:
4065 processor cycles (= 48780 clock cycles) per block, 768 bytes of code;

different variant: 3168 processor cycles (= 38016 clock cycles) and 1016 byte of code.

Motorola 68HC08:
8390 processor cycles (= 8390 clock cycles), 36 bytes of RAM, 919 bytes of code

Software (reference platform Pentium II, 200MHz):

ANSI C, optimized:
27 MBit/sec (EGCC 1.0.2, Linux)
70 MBit/sec (Visual C++, Windows)

The big difference is due to a Pentium optimization of the Visual C++ compiler that can execute data rotations in hardware.

Java:
1.1MBit/sec (JDK1.1.1 Java Compiler under Linux)

- It is cheaper to implement one single algorithm (this argument is certainly only true for hardware - R.W.)

- Developers of similar algorithms could claim patent rights. If only one algorithm is selected, costs will be smaller. (In the author's opinion, this reason could be the most important.)

Nevertheless, these critics can not destroy the positive impression we have of the whole procedure: The selection process was open and fair. Don't forget that not long ago in the USA cryptographic algorithms were still classified as "munitions." Now a cipher developed in Belgium and evaluated internationally will become a fundament of national security in the States, although not exclusively, since the US Government will apply it for "sensitive, not classified" information only (and similarly the NSA). But in practice it will dominate - in business, in good crypto products for private use etc. NIST expects AES to be the base for enciphering for the next 20 years or more. There is no reason yet to doubt these assertions.

This year will be the last in which a last public discussion runs. After this, Rijndael will be confirmed as a FIPS standard, probably between April and June 2001.

Rijndael in Details

After explaining why AES was selected and how it was done, let's take a closer look at this algorithm, at least to get some rough idea how it works. The elementary operation behind this cipher is astonishingly simple: bytewise substitution, byte exchange, and XOR. Even non-experts will understand immediately why AES is software- as well as hardware-friendly. There were even critics claiming that AES could be too simple to be really secure! (The cryptanalysis contradicts such opinions, however.) For simplicity, we suppose 128-bit keys and 128-bit blocks in the following (more details can be found in [2]).

1. Before encrypting, the 128 bit key is used to generate 10 subkeys of 128 bits (16 bytes) each. The bytes of each subkey are written column-wise into a 4x4 matrix, giving ten such matrices. (We are not going to step through this in huge detail but Figure 1 illustrates the method).

2. Analogously, a plaintext block of 128 bits (16 bytes) is written row-wise into a 4x4 matrix, called a *state* by Daemen and Rijmen. Each round generates a new state from the old; the state after the 10th round then contains the cipher text (also column-wise). After this, the procedure restarts with the next 128 bits of plaintext.

3. Each round of Rijndael executes sequentially the following steps:
   - ByteSub: The individual bytes in a state matrix are substituted according to a fixed scheme, i.e. they are replaced by other bytes (with the help of a "codebook"). This is a fixed transformation, there is still no encryption.
   - ShiftRow: The lines in the state matrix are rotated cyclically to the left, namely the 1./2./3./4. line by 0/1/2/3 bytes, respectively. For instance, if we have in line 2 of the state matrix

     a b c d

     then after ShiftRow it reads:

     b c d a.

     Also this transformation is obviously fixed (deterministic) and encrypts nothing.
   - MixColumn: The state columns are shuffled by a complicated, nevertheless fixed scheme.
   - AddRoundKey: The round key (also a 4x4 matrix!) will be bitwise XORed with the state. Only this transformation puts some secret into the state; we can thus now speak of an encryption.

4. Before the first round, an AddRoundKey transformation is done. In the last round the MixColumn transformation is left out.

Rijndael can work with 192- and 256-bit blocks as well. Then it uses 4x6 resp. 8x6 matrices as states and subkeys. The 128-bit algorithm executes 10 rounds; for longer keys, this number is increased to 12 and 14 rounds, respectively.

The security of Rijndael is essentially based on the number of rounds. Cracking of Rijndael with one round only would be a simple exercise for a cryptanalyst: Since ByteSub, ShiftRow and MixColumn are fixed, invertible transformations, the security would be the same as that on a so-called 128-bit Vernam encryption, that is, XOR'ing a 128-bit long key with 128-bit long pieces of cleartext. Such an encryption can be cracked within milliseconds; software can be found in [6]. But Rijndael is a product algorithm, i.e., similar transformations (differing only by the subkeys used in the AddRoundKey step) are applied again and again, each on the result of the former transformation. DES cryptanalysis teaches that security increases in steps, i.e. more than exponentially, with the number of rounds. The (so far) known cryptanalysis of Rijndael in Box 5 shows this impressively.

The transformations ByteSub, ShiftRow and MixColumn have been chosen such that they are as simple as possible (and thus easy to analyze) and also so that in spite of this simplicity all known methods of cryptanalysis do not work.

Box 5 - Cryptanalysis of Rijndael

We mention here the results of [4]. No practically meaningful attack on Rijndael is known, nor is such an attack to be expected in the future.

An algorithm is considered to be secure if it can be broken only by testing all possible keys (in the mean thus 2^{127} tests are necessary for 128-bit keys). This method is also called "brute force" and remains unrealistic for such key sizes (see also Box 1).

In contrast to brute force so-called "shortcut" attacks in theory allow to apply fewer tests. Often they require an exorbitant computing power and still far too much tries. If such a shortcut attack is not found, one tries to crack the algorithm with fewer rounds. This is of academic interest; however; it could be the start to a more effective cryptanalysis of the unchanged algorithm.

For Rijndael with reduced round number, the following shortcut attacks are known (also valid for 192- and 256-bit keys!):

6 rounds:

By encrypting $6*2^{32}$ chosen plaintext blocks and analyzing the generated ciphertext blocks with the help of 2^{44} complex operations (about 17 trillions), the key can be computed. In practice this means that about 384 GB plaintexts have to be "smuggled in," and 384 GB of resulting ciphertext have to be protocoled by an eavesdropper. If such a complex operation lasts only one microsecond, it takes about 200 days to compute the key (the time will be reduced to five hours if an operation takes a nanosecond).

7 rounds:

In this case almost 2^{128} plaintexts are needed (abut $5*10^{39}$ byte), the computing extends to 2^{120} operations. Supposing 1 nanosecond per operation, the five hours of the 6-round-variant now become $4*10^{19}$ years (40 billion billions of years). This attack even theoretically makes sense only for keys longer than 128 bits - otherwise, brute force would be almost as fast.

Note how strong security increases by adding merely one round! However, Rijndael executes at least 10 rounds (depending on the key size). Some theoretical weaknesses in round key generation were discovered, also of more or less academic interest. Practical implications are not expected.

The motivation for their design is explained by the authors in detail[1]. Some knowledge of algebra is useful when reading this literature - especially operations on Galois fields are important there. But some features are also understandable without mathematics. Namely, employing strong diffusion and confusion in each round is essential here:

Diffusion means that changing even a single state (or subkey) bit will influence many, preferably all, state bits after a few rounds. In Rijndael, ShiftRow and MixColumn are responsible for diffusion.

Confusion means that "structure" is destroyed - from the result of a round you should not be able to conclude on its input, in any manner. This task is mainly done by ByteSub and AddRoundKey.

Practically, each round is only a fixed transformation with XOR addition of the secret subkey after it. But the repeated execution of such transformation groups (in 10 rounds) creates a problem that cryptanalysts can not solve yet (hopefully this remains the case in the future). Neither differential nor linear cryptanalysis (the last works even for DES, at least in theory!) nor interpolation attacks work in this case. There are no weak keys, for which cryptanalysis would be easier. Nor are there attacks with related keys (which are applicable for smartcards) that have some effect, quite differently from DES.

Nevertheless some experts announced doubts that such a simple algorithm can be so secure. But there is no rational reason to be not sure. Complicated, "non-intrinsic" procedures (like DES) are more difficult to attack in theory. However, computers become faster and faster and can even work with formulas much larger than any human can handle. So it is better to have some algorithm we do understand; "security by obscurity" has never been a good recipe in cryptography. In work [2] the authors explain in detail the whole background for their design. So we can trust that there is no trapdoor in the algorithm. It seems strikingly secure for its simplicity.

Application Aspects

As has been mentioned, Rijndael can be implemented in hardware as well as in software (see Box 4). Even as an optimized C program it is shorter than 500 lines of code. And Rijndael can be implemented in hardware in such a way that neither timing nor power attacks work. These are malicious methods on smartcards containing some secret, hidden key. The attacker measures execution times and power consumption, respectively, during encryption of known (or given) plaintexts and so can reconstruct the secret key.

In 8 bit CPUs on smartcards decryption is up to 30% slower than encryption, however. In the software version there are small time differences since round keys are computed in another manner during decryption. For many algorithms decryption and encryption speed are the same, and the procedures differ very little. In contrast to this the decryption process of Rijndael uses the encryption hardware modules only partially, and in software other tables and other code are used. But decryption is not necessary if CFB (Cipher Feed-back) or OFB (Output Feed-back) modes are implemented[5, 6].

As could be expected, many products have already implemented the forthcoming standard. Also freeware was updated; the GNU Privacy Guard (gpg) gives a popular example[8].

References

[1] R. Wobst, *Offenes Geheimnis*,
UNIX/open 11/98, S.32-36

[2] www.nist.gov/aes

[3] R. Wobst, *Auf Nummer Sicher sein*,
Computer Zeitung 25/00 (23.6.00), S.20

[4] www.counterpane.com/rijndael.html

[5] B. Schneier, *Arpplied Cyptography*,
Addison-Wesley 1996

[6] R. Wobst, *Abenteuer Kryptologie*,
Addison-Wesley 1998

[7] R. Wobst, *Lauscher am Handy*,
Computer Zeitung 16/00 (20.4.00), S.20

[8] www.demcom.com/deutsch/steganos/

Reinhard Wobst studied mathematics at Technical University of Dresden and obtained his MS in the field of stochastic processes in 1980. About at this time he got infected by his first contact with interactive computers (he hated punchcards, as he hates Windows now). The computer disease has not vanished Now he programs scientific and industrial applications in C under UNIX and writes technical articles about this subject. Also about 1980 he made his first computer experiments with cryptographic algorithms. This hobby finally led to many articles about cryptography, security consulting and to the book “Abenteuer Kryptologie” (Addison-Wesley/Pearson Education) which is popular in Germany and - he hopes so - soonly also in Poland. The author can be reached at r.wobst@gmx.de