A Number-Theoretic Error-Correcting Code

Eric Brier[1], Jean-Sébastien Coron[2], Rémi Géraud[1,3], Diana Maimuţ[3(✉)], and David Naccache[2,3]

[1] Ingenico, 28-32 boulevard de Grenelle, 75015 Paris, France
{eric.brier,remi.geraud}@ingenico.com
[2] Université du Luxembourg, 6 rue Richard Coudenhove-Kalergi, 1359 Luxembourg, Luxembourg
{jean-sebastien.coron,david.naccache}@uni.lu
[3] Département d'Informatique, École normale supérieure, 45 rue d'Ulm, 75230 Paris Cedex 05, France
{remi.geraud,diana.maimut,david.naccache}@ens.fr

Abstract. In this paper we describe a new error-correcting code (ECC) inspired by the Naccache-Stern cryptosystem. While by far less efficient than Turbo codes, the proposed ECC happens to be more efficient than some established ECCs for certain sets of parameters.

The new ECC adds an appendix to the message. The appendix is the modular product of small primes representing the message bits. The receiver recomputes the product and detects transmission errors using modular division and lattice reduction.

1 Introduction

Error-correcting codes (ECCs) are essential to ensure reliable communication. ECCs work by adding redundancy which enables detecting and correcting mistakes in received data. This extra information is, of course, costly and it is important to keep it to a minimum: there is a trade-off between how much data is added for error correction purposes (bandwidth), and the number of errors that can be corrected (correction capacity).

Shannon showed [13] in 1948 that it is in theory possible to encode messages with a minimal number of extra bits[1]. Two years later, Hamming [7] proposed a construction inspired by parity codes, which provided both error detection and error correction. Subsequent research saw the emergence of more efficient codes, such as Reed-Muller [8,10] and Reed-Solomon [11]. The latest were generalized by Goppa [6]. These codes are known as algebraic-geometric codes.

Convolutional codes were first presented in 1955 [4], while recursive systematic convolutional codes [1] were introduced in 1991. Turbo codes [1] were indeed revolutionary, given their closeness to the channel capacity ("near Shannon limit").

[1] Shannon's theorem states that the best achievable expansion rate is $1 - H_2(p_b)$, where H_2 is binary entropy and p_b is the acceptable error rate.

I. Bica et al. (Eds.): SECITC 2015, LNCS 9522, pp. 25–35, 2015.
DOI: 10.1007/978-3-319-27179-8_2

Results: This paper presents a new error-correcting code, as well as a form of message size improvement based on the hybrid use of two ECCs one of which is inspired by the Naccache-Stern (NS) cryptosystem [2,9]. For some codes and parameter choices, the resulting hybrid codes outperform the two underlying ECCs.

The proposed ECC is unusual because it is based on number theory rather than on binary operations.

2 Preliminaries

2.1 Notations

Let $\mathfrak{P} = \{p_1 = 2, \dots\}$ be the ordered set of prime numbers. Let $\gamma \geq 2$ be an encoding base. For any $m \in \mathbb{N}$ (the "message"), let $\{m_i\}$ be the digits of m in base γ *i.e.*:

$$m = \sum_{i=0}^{k-1} \gamma^i m_i \qquad m_i \in [0, \gamma - 1], \quad k = \lceil \log_\gamma m \rceil$$

We denote by $h(x)$ the Hamming weight of x, *i.e.* the sum of x's digits in base 2, and, by $|y|$ the bit-length of y.

2.2 Error-Correcting Codes

Let $\mathcal{M} = \{0,1\}^k$ be the set of messages, $\mathcal{C} = \{0,1\}^n$ the set of encoded messages. Let $\mathcal{P}$ be a parameter set.

Definition 1 (Error-Correcting Code). *An* error-correcting code *is a couple of algorithms:*

- *An algorithm μ, taking as input some message $m \in \mathcal{M}$, as well as some public parameters* $\mathsf{params} \in \mathcal{P}$*, and outputting $c \in \mathcal{C}$.*
- *An algorithm μ^{-1}, taking as input $\tilde{c} \in \mathcal{C}$ as well as parameters* $\mathsf{params} \in \mathcal{P}$*, and outputting $m \in \mathcal{M} \cup \{\perp\}$.*

The $\perp$ symbol indicates that decoding failed.

Definition 2 (Correction Capacity). *Let $(\mu, \mu^{-1}, \mathcal{M}, \mathcal{C}, \mathcal{P})$ be an error-correcting code. There exists an integer $t \geq 0$ and some parameters* $\mathsf{params} \in \mathcal{P}$ *such that, for all $e \in \{0,1\}^n$ such that $h(e) \leq t$,*

$$\mu^{-1}\left(\mu\left(m, \mathsf{params}\right) \oplus e, \mathsf{params}\right) = m, \qquad \forall m \in \mathcal{M}$$

and for all e such that $h(e) > t$,

$$\mu^{-1}\left(\mu\left(m, \mathsf{params}\right) \oplus e, \mathsf{params}\right) \neq m, \qquad \forall m \in \mathcal{M}.$$

t is called the correction capacity *of $(\mu, \mu^{-1}, \mathcal{M}, \mathcal{C}, \mathcal{P})$.*

Definition 3. *A code of message length k, of codeword length n and with a correction capacity t is called an (n, k, t)-code. The ratio $\rho = \frac{n}{k}$ is called the code's* expansion rate.

3 A New Error-Correcting Code

Consider in this section an existing (n, k, t)-code $C = (\mu, \mu^{-1}, \mathcal{M}, \mathcal{C}, \mathcal{P})$. For instance C can be a Reed-Muller code. We describe how the new (n', k, t)-code $C' = (\nu, \nu^{-1}, \mathcal{M}, \mathcal{C}', \mathcal{P}')$ is constructed.

Parameter Generation: To correct t errors in a k-bit message, we generate a prime p such that:

$$2 \cdot p_k^{2t} < p < 4 \cdot p_k^{2t} \tag{1}$$

As we will later see, the size of p is obtained by bounding the worst case in which all errors affect the end of the message. p is a part of $\mathcal{P}'$.

Encoding: Assume we wish to transmit a k-bit message m over a noisy channel. Let $\gamma = 2$ so that m_i denote the i-th bit of m, and define:

$$c(m) := \prod_{i=1}^{k} p_i^{m_i} \bmod p \tag{2}$$

The integer generated by Eq. (2) is encoded using C to yield $\mu(c(m))$. Finally, the encoded message $\nu(m)$ transmitted over the noisy channel is defined as:

$$\mu(m) := m \| \mu(c(m)) \tag{3}$$

Note that, if we were to use C directly, we would have encoded m (and not c). The value c is, in most practical situations, much shorter than m. As is explained in Sect. 3.1, c is smaller than m (except the cases in which m is very small and which are not interesting in practice) and thereby requires fewer extra bits for correction. For appropriate parameter choices, this provides a more efficient encoding, as compared to C.

Decoding: Let α be the received[2] message. Assume that at most t errors occurred during transmission:

$$\alpha = \nu(m) \oplus e = m' \| (\mu(c(m)) \oplus e')$$

where the error vector e is such that $h(e) = h(m' \oplus m) + h(e') \leq t$.

Since $c(m)$ is encoded with a t-error-capacity code, we can recover the correct value of $c(m)$ from $\mu(c(m)) \oplus e'$ and compute the quantity:

$$s = \frac{c(m')}{c(m)} \bmod p \tag{4}$$

Using Eq. (2) s can be written as:

$$s = \frac{a}{b} \bmod p, \quad \begin{cases} a &= \prod\limits_{(m'_i=1)\wedge(m_i=0)} p_i \\ b &= \prod\limits_{(m'_i=0)\wedge(m_i=1)} p_i \end{cases} \tag{5}$$

[2] *i.e.* encoded and potentially corrupted.

Note that since $h(m' \oplus m) \leq t$, we have that a and b are strictly smaller than $(p_k)^t$. Theorem 1 from [5] shows that given t the receiver can recover a and b efficiently using a variant of Gauss' algorithm [14].

Theorem 1. *Let $a, b \in \mathbb{Z}$ such that $-A \leq a \leq A$ and $0 < b \leq B$. Let p be some prime integer such that $2AB < p$. Let $s = a \cdot b^{-1} \mod p$. Then given A, B, s and p, a and b can be recovered in polynomial time.*

As $0 \leq a \leq A$ and $0 < b \leq B$ where $A = B = (p_k)^t - 1$ and $2AB < p$ from Eq. (1), we can recover a and b from t in polynomial time. Then, by testing the divisibility of a and b with respect to the small primes p_i, the receiver can recover $m' \oplus m$ and eventually m.

A numerical example is given in Appendix A.

Bootstrapping: Note that instead of using an existing code as a sub-contractor for protecting $c(m)$, the sender may also recursively apply the new scheme described above. To do so consider $c(m)$ as a message, and protect $\overline{c} = c(c(\cdots c(c(m))))$, which is a rather small value, against accidental alteration by replicating it $2t+1$ times. The receiver will use a majority vote to detect the errors in $\overline{c}$.

3.1 Performance of the New Error-Correcting Code for $\gamma = 2$

Lemma 1. *The bit-size of $c(m)$ is:*

$$\log_2 p \simeq 2 \cdot t \log_2(k \ln k). \tag{6}$$

Proof. From Eq. (1) and the Prime Number Theorem[3]. □

The total output length of the new error-correcting code is therefore $\log_2 p$, plus the length k of the message m.

C' outperforms the initial error correcting code C if, for equal error capacity t and message length k, it outputs a shorter encoding, which happens if $n' < n$, keeping in mind that both n and n' depend on k.

Corollary 1. *Assume that there exists a constant $\delta > 1$ such that, for k large enough, $n(k) \geq \delta k$. Then for k large enough, $n'(k) \leq n(k)$.*

Proof. Let k be the size of m and k' be the size of $c(m)$. We have $n'(k) = k + n(k')$, therefore

$$n(k) - n'(k) = n(k) - (k + n(k')) \geq (\delta - 1)k - n(k').$$

Now,

$$(\delta - 1)k - n(k') \geq 0 \Leftrightarrow (\delta - 1)k \geq n(k').$$

[3] $p_k \simeq k \ln k$.

But $n(k') \geq \delta k'$, hence

$$(\delta - 1)k \geq \delta k' \Rightarrow k \geq \frac{k'\delta}{(\delta - 1)}.$$

Finally, from Lemma 1, $k' = O(\ln \ln k!)$, which guarantees that there exists a value of k above which $n'(k) \leq n(k)$. □

In other terms, any correcting code whose encoded message size is growing linearly with message size can benefit from the described construction (Fig. 1).

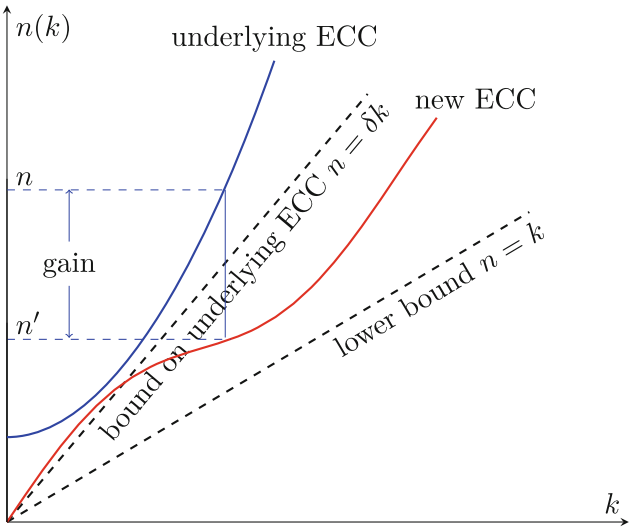

Fig. 1. Illustration of Corollory 1. For large enough values of k, the new ECC uses smaller codewords as compared to the underlying ECC.

Expansion Rate: Let k be the length of m and consider the bit-size of the corresponding codeword as in Eq. (6). The expansion rate ρ is:

$$\rho = \frac{|m||\mu(c(m))|}{|m|} = \frac{k + |\mu(c(m))|}{k} = 1 + \frac{|\mu(c(m))|}{k} \tag{7}$$

Reed-Muller Codes. We illustrate the idea with Reed-Muller codes. Reed-Muller (R-M) codes are a family of linear codes. Let $r \geq 0$ be an integer, and $N = \log_2 n$, it can apply to messages of size

$$k = \sum_{i=1}^{r} \binom{N}{i} \tag{8}$$

Table 1. Examples of length n, dimension k, and error capacity t for Reed-Muller code.

n	16	64	128	256	512	2048	8192	32768	131072
k	11	42	99	163	382	1024	5812	9949	65536
t	1	3	3	7	7	31	31	255	255

Such a code can correct up to $t = 2^{N-r-1} - 1$ errors. Some examples of $\{n, k, t\}$ triples are given in Table 1. For instance, a message of size 163 bits can be encoded as a 256-bit string, among which up to 7 errors can be corrected.

To illustrate the benefit of our approach, consider a 5812-bit message, which we wish to protect against up to 31 errors.

A direct use of Reed-Muller would require $n = 8192$ bits as seen in Table 1. Contrast this with our code, which only has to protect $c(m)$, that is 931 bits as shown by Eq. 6, yielding a total size of $5812 + n(931) = 5812 + 2048 = 7860$ bits.

Other parameters for the Reed-Muller primitive are illustrated in Table 2.

Table 2. (n, k, t)-codes generated from Reed-Muller by our construction.

n'	638	7860	98304
k	382	5812	65536
$c(m)$	157	931	9931
RM($c(m)$)	256	2048	32768
t	7	31	255

Table 2 shows that for large message sizes and a small number of errors, our error-correcting code slightly outperforms Reed-Muller code.

3.2 The Case $\gamma > 2$

The difficulty in the case $\gamma > 2$ stems from the fact that a binary error in a γ-base message will in essence scramble all digits preceding the error. As an example,

$$\underline{1220021012202012010011120202}_3 + 2^{30} = \underline{1220021022112000112220110110}_3$$

Hence, unless $\gamma = 2^\Gamma$ for some Γ, a generalization makes sense only for channels over which transmission uses γ symbols. In such cases, we have the following: a k-bit message m is pre-encoded as a γ-base κ-symbol message m'. Here $\kappa = \lceil k/\log_2 \gamma \rceil$. Eq. (1) becomes:

$$2 \cdot p_\kappa^{2t(\gamma-1)} < p < 4 \cdot p_\kappa^{2t(\gamma-1)}$$

Comparison with the binary case is complicated by the fact that here t refers to the number of *any* errors regardless their semiologic meaning. In other words, an error transforming a 0 into a 2 counts exactly as an error transforming 0 into a 1.

Example 1. As a typical example, for $t = 7$, $\kappa = 10^6$ and $\gamma = 3$, $p_\kappa = 15485863$ and p is a 690-bit number.

For the sake of comparison, $t = 7$, $k = 1584963$ (corresponding to $\kappa = 10^6$) and $\gamma = 2$, yield $p_k = 25325609$ and a 346-bit p.

4 Improvement Using Smaller Primes

The construction described in the previous section can be improved by choosing a smaller prime p, but comes at a price; namely decoding becomes only heuristic.

Parameter Generation: The idea consists in generating a prime p smaller than before. Namely, we generate a p satisfying :

$$2^u \cdot p_k^t < p < 2^{u+1} \cdot p_k^t \tag{9}$$

for some small integer $u \geq 1$.

Encoding and Decoding: Encoding remains as previously. The redundancy $c(m)$ being approximately half as small as the previous section's one, we have :

$$s = \frac{a}{b} \mod p, \quad \begin{cases} a &= \prod_{(m'_i=1)\wedge(m_i=0)} p_i \\ b &= \prod_{(m'_i=0)\wedge(m_i=1)} p_i \end{cases} \tag{10}$$

and since there are at most t errors, we must have :

$$a \cdot b \leq (p_k)^t \tag{11}$$

We define a finite sequence $\{A_i, B_i\}$ of integers such that $A_i = 2^{u \cdot i}$ and $B_i = \lfloor 2p/A_i \rfloor$. From Eqs. (9) and (11) there must be at least one index i such that $0 \leq a \leq A_i$ and $0 < b \leq B_i$. Then using Theorem 1, given A_i, B_i, p and s, the receiver can recover a and b, and eventually m.

The problem with that approach is that we lost the guarantee that $\{a, b\}$ is unique. Namely we may find another $\{a', b'\}$ satisfying Eq. (10) for some other index i'. We expect this to happen with negligible probability for large enough u, but this makes the modified code heuristic (while perfectly implementable for all practical purposes).

4.1 Performance

Lemma 2. *The bit-size of $c(m)$ is:*

$$\log_2 p \simeq u + t \log_2(k \ln k). \tag{12}$$

Proof. Using Eq. (9) and the Prime Number Theorem. □

Thus, the smaller prime variant has a shorter $c(m)$.

As u is a small integer (*e.g.* $u = 50$), it follows immediately from Eq. (1) that, for large n and t, the size of the new prime p will be approximately half the size of the prime p generated in the preceding section.

This brings down the minimum message size k above which our construction provides an improvement over the bare underlying correcting code.

Note: In the case of Reed-Muller codes, this variant provides no improvement over the technique described in Sect. 3 for the following reasons: (1) by design, Reed-Muller codewords are powers of 2; and (2) Eq. (12) cannot yield a twofold reduction in p. Therefore we cannot hope to reduce p enough to get a smaller codeword.

That doesn't preclude other codes to show benefits, but the authors did not look for such codes.

5 Prime Packing Encoding

It is interesting to see whether the optimization technique of [2] yields more efficient ECCs. Recall that in [2], the p_is are distributed amongst κ packs. Information is encoded by picking one p_i per pack. This has an immediate impact on decoding: when an error occurs and a symbol σ is replaced by a symbol σ', both the numerator and the denominator of s are affected by *additional* prime factors.

Let $C = (\mu, \mu^{-1}, \mathcal{M}, \mathcal{C}, \mathcal{P})$ be a t-error capacity code, such that it is possible to efficiently recover c from $\mu(c) \oplus e$ for any c and any e, where $h(e) \leq t$. Let $\gamma \geq 2$ be a positive integer.

Before we proceed, we define $\kappa := \lceil k/\log_2 \gamma \rceil$ and

$$f := f(\gamma, \kappa, t) = \prod_{i=k-t}^{k} p_{\gamma i}.$$

Parameter Generation: Let p be a prime number such that:

$$2 \cdot f^2 < p < 4 \cdot f^2 \tag{13}$$

Let $\hat{\mathcal{C}} = \mathcal{M} \times \mathbb{Z}_p$ and $\hat{\mathcal{P}} = (\mathcal{P} \cup \mathfrak{P}) \times \mathbb{N}$. We now construct a variant of the ECC presented in Sect. 3 from C and denote it

$$\hat{C} = \left(\nu, \nu^{-1}, \mathcal{M}, \hat{\mathcal{C}}, \hat{\mathcal{P}}\right).$$

Encoding: We define the "redundancy" of a k-bit message $m \in \mathcal{M}$ (represented as κ digits in base γ) by:

$$\hat{c}(m) := \prod_{i=0}^{\kappa-1} p_{i\gamma+m_i+1} \bmod p$$

A message m is encoded as follows:

$$\nu(m) := m \| \mu\left(\hat{c}\,(m)\right)$$

Decoding: The received information α differs from $\nu(m)$ by a certain number of bits. Again, we assume that the number of these differing bits is at most t. Therefore $\alpha = \nu(m) \oplus e$, where $h(e) \leq t$. Write $e = e_m \| e_{\hat{c}}$ such that

$$\alpha = \nu(m) \oplus e = m \oplus e_m \| \mu(\hat{c}(m)) \oplus e_{\hat{c}} = m' \| \mu(\hat{c}(m)) \oplus e_{\hat{c}}.$$

Since $h(e) = h(e_m) + h(e_{\hat{c}}) \leq t$, the receiver can recover efficiently $\hat{c}(m)$ from α. It is then possible to compute

$$s := \frac{\hat{c}(m')}{\hat{c}(m)} \bmod p = \frac{\prod_{i=0}^{\kappa-1} p_{i\gamma+m'_i+1}}{\prod_{i=0}^{\kappa-1} p_{i\gamma+m_i+1}} \bmod p.$$

$$s = \frac{a}{b} \bmod p, \quad \begin{cases} a &= \prod_{m'_i \neq m_i} p_{i\gamma+m'_i+1} \\ b &= \prod_{m_i \neq m'_i} p_{i\gamma+m_i+1} \end{cases} \tag{14}$$

As $h(e) = h(e_m) + h(e_{\hat{c}}) \leq t$, we have that a and b are strictly smaller than $f(\gamma,\kappa)^{2t}$. As $A = B = f(\gamma,\kappa)^{2t} - 1$, we observe from Eq. (13) that $2AB < p$. We are now able to recover a, b, $\gcd(a,b) = 1$ such that $s = a/b \bmod p$ using lattice reduction [14].

Testing the divisibility of a and b by $p_1, \ldots, p_{\kappa\gamma}$ the receiver can recover $e_m = m' \oplus m$, and from that get $m = m' \oplus e_m$. Note that by construction only one prime amongst γ is used per "pack": the receiver can therefore skip on average $\gamma/2$ primes in the divisibility testing phase.

5.1 Performance

Rosser's theorem [3,12] states that for $n \geq 6$,

$$\ln n + \ln\ln n - 1 < \frac{p_n}{n} < \ln n + \ln\ln n$$

i.e. $p_n < n(\ln n + \ln\ln n)$. Hence a crude upper bound of p is

$$\begin{aligned} p &< 4f(\kappa,\gamma,t)^2 \\ &= 4\left(\prod_{i=\kappa-t}^{\kappa} p_{\gamma i}\right)^2 \\ &\leq 4\prod_{i=\kappa-t}^{\kappa} \left(i\gamma(\ln i\gamma + \ln\ln(i\gamma))\right)^2 \\ &\leq 4\gamma^{2t}\left(\frac{\kappa!}{(\kappa-t-1)!}\right)^2 (\ln\kappa\gamma + \ln\ln\kappa\gamma)^{2t} \end{aligned}$$

Again, the total output length of the new error-correcting code is $n' = k + |p|$.

Plugging $\gamma = 3$, $\kappa = 10^6$ and $t = 7$ into Eq. (13) we get a 410-bit p. This improves over Example 1 where p was 690 bits long.

A Toy Example

Let m be the 10-bit message 1100100111. For $t = 2$, we let p be the smallest prime number greater than $2 \cdot 29^4$, *i.e.* $p = 707293$. We generate the redundancy:

$$c(m) = 2^1 \cdot 3^1 \cdot 5^0 \cdot 7^0 \cdot 11^1 \cdot 13^0 \cdot 17^0 \cdot 19^1 \cdot 23^1 \cdot 29^1 \bmod 707293$$

$$\Rightarrow c(m) = 836418 \bmod 707293 = 129125.$$

As we focus on the new error-correcting code we simply omit the Reed-Muller component. The encoded message is

$$\nu(m) = \mathtt{1100100111}_2 \| \mathtt{129125}_{10}.$$

Let the received encoded message be $\alpha = \mathtt{1100101011}_2 \| \mathtt{129125}_{10}$. Thus,

$$c(m') = 2^1 \cdot 3^1 \cdot 5^0 \cdot 7^0 \cdot 11^1 \cdot 13^0 \cdot 17^1 \cdot 19^0 \cdot 23^1 \cdot 29^1 \bmod p$$

$$\Rightarrow c(m') = 748374 \bmod 707293 = 41081.$$

Dividing by $c(m)$ we get

$$s = \frac{c(m')}{c(m)} = \frac{41081}{129125} \bmod 707293 = 632842$$

Applying the rationalize and factor technique we obtain $s = \dfrac{17}{19} \bmod 707293$. It follows that $m' \oplus m = \mathtt{0000001100}$. Flipping the bits retrieved by this calculation, we recover m.

References

1. Berrou, C., Glavieux, A., Thitimajshima, P.: Near Shannon limit error-correcting coding and decoding: turbo-codes. In: IEEE International Conference on Communications - ICC 1993, vol. 2, pp. 1064–1070, May 1993
2. Chevallier-Mames, B., Naccache, D., Stern, J.: Linear bandwidth naccache-stern encryption. In: Ostrovsky, R., De Prisco, R., Visconti, I. (eds.) SCN 2008. LNCS, vol. 5229, pp. 327–339. Springer, Heidelberg (2008)
3. Dusart, P.: The kth prime is greater than $k(\ln k + \ln \ln k - 1)$ for $k \geq 2$. Math. Comput. **68**, 411–415 (1999)
4. Elias, P.: Coding for noisy channels. In: IRE Convention Record, pp. 37–46 (1955)
5. Fouque, P.-A., Stern, J., Wackers, G.-J.: CryptoComputing with rationals. In: Blaze, M. (ed.) FC 2002. LNCS, vol. 2357, pp. 136–146. Springer, Heidelberg (2003)
6. Goppa, V.D.: Codes on algebraic curves. Sov. Math. Dokl. **24**, 170–172 (1981)
7. Hamming, R.W.: Error detecting and error correcting codes. Bell Syst. Tech. J. **29**(2), 147–160 (1950)
8. Muller, D.E.: Application of boolean algebra to switching circuit design and to error detection. IRE Trans. Inf. Theory **3**, 6–12 (1954)

9. Naccache, D., Stern, J.: A new public-key cryptosystem. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 27–36. Springer, Heidelberg (1997)
10. Reed, I.: A class of multiple-error-correcting codes and the decoding scheme. IRE Trans. Inf. Theory **4**, 38–49 (1954)
11. Reed, I.S., Solomon, G.: Polynomial codes over certain finite fields. J. Soc. Ind. Appl. Math. **8**(2), 300–304 (1960)
12. Rosser, J.B.: The n-th prime is greater than $n \ln n$. Proc. Lond. Math. Soc. **45**, 21–44 (1938)
13. Shannon, C.: A mathematical theory of communication. Bell Syst. Tech. J. **27**(379–423), 623–656 (1948)
14. Vallée, B.: Gauss' algorithm revisited. J. Algorithms **12**(4), 556–572 (1991)

http://www.springer.com/978-3-319-27178-1

Innovative Security Solutions for Information Technology and Communications
8th International Conference, SECITC 2015, Bucharest, Romania, June 11-12, 2015. Revised Selected Papers
Bica, I.; Naccache, D.; Simion, E. (Eds.)
2015, XXIV, 281 p. 88 illus., Softcover
ISBN: 978-3-319-27178-1