Method of Generating Irreducible Polynomials over GF(3) on the Basis of Trinomials

Grzegorz Borowik and Andrzej Paszkiewicz

Institute of Telecommunications,
00-665 Warsaw, Nowowiejska 15/19, Poland
{G.Borowik,anpa}@tele.pw.edu.pl

Abstract. The paper presents a concept design of hardware co-processor that could be used to generate irreducible primitive polynomials with coefficients over GF(3). The process of generating a primitive polynomial is done by replicating the other primitive polynomial which is fixed in the device. The implemented algorithm allows the unit to generate all possible primitive polynomials of the same degree as the stored polynomial. This approach allows us to extend the cryptographic power and capabilities of the existing cryptographic devices.

Keywords: irreducible polynomial, primitive polynomial, trinomial, linear feedback shift register, stream cipher.

1 Introduction

Primitive and irreducible polynomials over the finite fields play an important role in the coding theory [7] and cryptography [4,8]. However, in particular sparse polynomials with only a few non-zero coefficients, e.g. trinomials are most commonly used.

The designers often use a primitive polynomial as a basic unit in constructing stream ciphers for implementing linear feedback shift register (LFSR). Since linear feedback shift registers are described by coefficients of primitive polynomials, these polynomials should be treated as the key information. It would be convenient to be able to generate such information directly into the encryption device, e.g. based on the data which sites transmit to each other during the startup of a call. Such a situation is only possible when the algorithm for generating new LFSRs works efficiently.

Although general methods of generating primitive polynomials are known, they are so complex and laborious that computers have to be used when dealing with polynomials of large degrees. It is also very often not possible to store primitive polynomials in memory.

On the other hand, the popularity of irreducible polynomials over GF(2) in cryptography makes LFSRs be mostly described by sparse polynomials of the characteristic 2. Although they benefit from easy hardware implementation and bit vector representation, it is not always possible to find a primitive trinomial of a given degree.

R. Moreno-Díaz et al. (Eds.): EUROCAST 2011, Part II, LNCS 6928, pp. 335–342, 2012.

Then two solutions are possible: using other sparse polynomials, for example pentanomials, over the same field or choosing a new finite field for which irreducible trinomials exist, e.g. GF(3) instead of GF(2). In the former case it has been observed that irreducible pentanomials over GF(2) exist for all degrees between 4 and 30000 [9]. In the latter a hypothesis based on observations has been made that alongside with the increasing order of a finite field characteristic, the number of degrees for which irreducible trinomials exists increases [10].

In this paper, we focus on generating irreducible polynomials from trinomials over GF(3). The proposed method is not computationally complex and yields primitive polynomials which finally generate the feedback polynomials. Since for the ternary field $-1 = 2$, polynomials over GF(3) can be considered as binary polynomials with both positive and negative coefficients. What we need is the representative of the primitive polynomials of a certain degree to be able to generate all the other primitive polynomials of that degree. We assume that due to the need of memorizing a large number of high-degree polynomials/representatives, we are only interested to store polynomials of a small number of non-zero coefficients, i.e. trinomials.

The following aspects motivated the authors to perform the work on a cryptographic co-processor:

- the reliability of the systems designed by us is greater than the use of computing platforms based on commercial operating systems;
- stream ciphers are convenient for implementation in hardware, which explains their use in encryption/description devices;
- generating cryptographically strong primitive polynomials modulo p is not an easy task, however, replacing them in cryptographic devices frequently is preferred;
- designing stream ciphers is older and better mastered than designing block ciphers and therefore it is easier to design a strong cryptographic stream cipher than a block-cipher;
- cryptographically weak primitive polynomials, i.e. of a small number of non-zero coefficients are convenient to be implemented in software but they decrease the resistance of encryption schemes to the correlation attacks.

2 Basics

A trinomial is a polynomial consisting of three non-zero coefficient monomials of different degrees, e.g. $X^7 + 2X^2 + 1$.

A polynomial is called *irreducible* over a finite field if it cannot be factored into nontrivial polynomials over the same field. In other case polynomial is *reducible*; e.g. $X^7 + 2X^2 + 1$ is irreducible over GF(3), while $X^7 + X^2 + 1$ is reducible because $X^7 + X^2 + 1 = (X + 2)^2(X^2 + X + 2)(X^3 + X^2 + 2)$.

A *primitive polynomial* of degree n with coefficients over the field GF(p) is a polynomial such that the simplest monomial X generates all the elements of the extension field GF(p^n).

Example. Let's consider polynomial $f(X) = X^3 + 2X + 1$ over GF(3). Then $\alpha = X$ is a root of the polynomial $f(X)$. Thus, $\alpha^3 = \alpha + 2$ and similarly $\alpha^4 = \alpha^2 + 2\alpha$. All the elements of the extension field GF(3^n) generated by X are given in Table 1.

Table 1. Elements of GF(3^n) generated by X, where $f(X) = X^3 + 2X + 1$

$\alpha^3 = \alpha + 2$	$\alpha^{12} = \alpha^2 + 2$	$\alpha^{21} = \alpha^2 + 1$
$\alpha^4 = \alpha^2 + 2\alpha$	$\alpha^{13} = 2$	$\alpha^{22} = 2\alpha + 2$
$\alpha^5 = 2\alpha^2 + \alpha + 2$	$\alpha^{14} = 2\alpha$	$\alpha^{23} = 2\alpha^2 + 2\alpha$
$\alpha^6 = \alpha^2 + \alpha + 1$	$\alpha^{15} = 2\alpha^2$	$\alpha^{24} = 2\alpha^2 + 2\alpha + 1$
$\alpha^7 = \alpha^2 + 2\alpha + 2$	$\alpha^{16} = 2\alpha + 1$	$\alpha^{25} = 2\alpha^2 + 1$
$\alpha^8 = 2\alpha^2 + 2$	$\alpha^{17} = 2\alpha^2 + \alpha$	$\alpha^{26} = 1 = \alpha^0$
$\alpha^9 = \alpha + 1$	$\alpha^{18} = \alpha^2 + 2\alpha + 1$	$\alpha^1 = \alpha$
$\alpha^{10} = \alpha^2 + \alpha$	$\alpha^{19} = 2\alpha^2 + 2\alpha + 2$	$\alpha^2 = \alpha^2$
$\alpha^{11} = \alpha^2 + \alpha + 2$	$\alpha^{20} = 2\alpha^2 + \alpha + 1$	$\alpha^3 = \alpha + 2$

3 Proposed Scheme

Let $f(X)$ be a primitive polynomial of degree n over GF(p). Then $\alpha = X$ is a root of the polynomial and $\alpha^{p^1}, \alpha^{p^2}, \ldots, \alpha^{p^{n-1}}$ are the other roots of the polynomial.

Let's assume that $\beta = \alpha^k (\mathrm{mod} f(\alpha))$ and find the minimal polynomial of β a primitive element of the finite extension field GF(p^n), where k is an odd number greater than 1 and $\gcd(k, p^n - 1) = 1$. Then the minimal polynomial is a new primitive polynomial of degree n with the following roots:

$$\begin{aligned} \beta_1 &= \alpha^{k \cdot p^0}, \\ \beta_2 &= \alpha^{k \cdot p^1}, \\ &\vdots \\ \beta_n &= \alpha^{k \cdot p^{n-1}}. \end{aligned}$$

Thus, the minimal polynomial $f_\beta(X)$ of the primitive element β is given by the following formula:

$$f_\beta(X) = (X - \beta_1)(X - \beta_2) \ldots (X - \beta_n). \tag{1}$$

Considering that for the ternary field $-1 = 2$, we have $\beta_i' = -\beta_i = 2\beta_i$, thus

$$f_\beta(X) = (X + \beta_1')(X + \beta_2') \ldots (X + \beta_n'). \tag{2}$$

Expanding the right side of the equation 2 we obtain a sum of monomials:

$$f_\beta(X) = a_n X^n + a_{n-1} X^{n-1} + \ldots + a_1 X + a_0, \quad \text{where } a_n = 1. \tag{3}$$

Coefficients $a_0, a_1, \ldots, a_{n-1}$ can be determined using the following procedure. Let $W_0(X) = 1$, $W_1(X) = X + \beta_1'$, and $W_t(X) = W_{t-1}(X)(X + \beta_t')$, where

$$W_{t-1}(X) = b_{t-1}X^{t-1} + b_{t-2}X^{t-2} + \ldots + b_1X + b_0,$$
$$W_t(X) = a_tX^t + a_{t-1}X^{t-1} + \ldots + a_1X + a_0$$

and $a_t = b_{t-1} = 1$ for $t = 1, 2, \ldots, n$.

Then the polynomial $W_t(X)$ can be expressed by coefficients of the polynomial $W_{t-1}(X)$ as follows:

$$\begin{aligned} W_t(X) &= (b_{t-1}X^{t-1} + b_{t-2}X^{t-2} + \ldots + b_1X + b_0)(X + \beta_t') \\ &= b_{t-1}X^t + (b_{t-2} + \beta_t'b_{t-1})X^{t-1} + (b_{t-3} + \beta_t'b_{t-2})X^{t-2} + \\ &\quad \ldots + (b_1 + \beta_t'b_2)X^2 + (b_0 + \beta_t'b_1)X + b_0\beta_t'. \end{aligned}$$

Thus

$$\begin{aligned} a_0 &= b_0\beta_t', \\ a_1 &= b_0 + \beta_t'b_1, \\ &\vdots \\ a_i &= b_{i-1} + \beta_t'b_i, \quad \text{for } i = 1, 2, \ldots, t-1, \\ a_t &= b_{t-1} \end{aligned}$$

for $t = 1, 2, \ldots, n$.

Polynomial $W_n(X)$ obtained by applying a presented procedure is a new primitive polynomial $f_\beta(X)$ corresponding to the primitive element β.

Example. Let's consider a primitive trinomial $f(X) = X^3 + 2X + 1$ over GF(3) and parameter $k = 5$.

Then $\alpha = X$ is a root of the $f(X)$. Based on the definition of the primitive polynomial, one can easily confirm that α^5 is not a root of $f(X)$ because $5 \neq 3^n$. Let's assume $\beta_1 = \alpha^5$. Since $\gcd(5, 3^3 - 1) = 1$ the minimal polynomial of the element β is primitive.

Based on the Table 1 we have $\beta = \alpha^5 = 2\alpha^2 + \alpha + 2$. Thus, the following are the roots of a new primitive polynomial:

$$\begin{aligned} \beta_1 &= 2\alpha^2 + \alpha + 2, \\ \beta_2 &= \beta_1^3 = \alpha^{15} = 2\alpha^2, \\ \beta_3 &= \beta_2^3 = \alpha^{45} = \alpha^{19} = 2\alpha^2 + 2\alpha + 2. \end{aligned}$$

According to the mentioned procedure of multiplication we have:

$$\begin{aligned} W_1(X) &= X - \beta_1 = X - (2\alpha^2 + \alpha + 2) = X + \alpha^2 + 2\alpha + 1, \\ W_2(X) &= W_1(X)(X - \beta_2) = (X + \alpha^2 + 2\alpha + 1)(X - 2\alpha^2) \\ &= (X + \alpha^2 + 2\alpha + 1)(X + \alpha^2) \end{aligned}$$

$$
\begin{aligned}
&= X^2 + \alpha^2 X + 2\alpha X + X + X\alpha^2 + \alpha^4 + 2\alpha^3 + \alpha^2 \\
&= X^2 + (2\alpha^2 + 2\alpha + 1)X + \alpha^2 + 2\alpha + 2\alpha + 1 + \alpha^2, \\
&= X^2 + (2\alpha^2 + 2\alpha + 1)X + 2\alpha^2 + \alpha + 1, \\
W_3(X) &= W_2(X)(X - \beta_3) \\
&= (X^2 + (2\alpha^2 + 2\alpha + 1)X + 2\alpha^2 + \alpha + 1)(X - (2\alpha^2 + 2\alpha + 2)) \\
&= (X^2 + (2\alpha^2 + 2\alpha + 1)X + 2\alpha^2 + \alpha + 1)(X + \alpha^2 + \alpha + 1) \\
&= X^3 + (2\alpha^2 + 2\alpha + 1)X^2 + (2\alpha^2 + \alpha + 1)X \\
&\quad + X^2\alpha^2 + (2\alpha^4 + 2\alpha^3 + \alpha^2)X + 2\alpha^4 + \alpha^3 + \alpha^2 \\
&\quad + X^2\alpha + (2\alpha^3 + 2\alpha^2 + \alpha)X + 2\alpha^3 + \alpha^2 + \alpha \\
&\quad + X^2 + (2\alpha^2 + 2\alpha + 1)X + 2\alpha^2 + \alpha + 1 \\
&= X^3 + (2\alpha^2 + 2\alpha + 1)X^2 + (2\alpha^2 + \alpha + 1)X \\
&\quad + X^2\alpha^2 + (2\alpha^2 + \alpha + 2\alpha + 1 + \alpha^2)X + 2\alpha^2 + \alpha + \alpha + 2 + \alpha^2 \\
&\quad + X^2\alpha + (2\alpha + 1 + 2\alpha^2 + \alpha)X + 2\alpha + 1 + \alpha^2 + \alpha \\
&\quad + X^2 + (2\alpha^2 + 2\alpha + 1)X + 2\alpha^2 + \alpha + 1 \\
&= X^3 + (2\alpha^2 + 2\alpha + 1)X^2 + (2\alpha^2 + \alpha + 1)X \\
&\quad + X^2\alpha^2 + X + 2\alpha + 2 \\
&\quad + X^2\alpha + (2\alpha^2 + 1)X + \alpha^2 + 1 \\
&\quad + X^2 + (2\alpha^2 + 2\alpha + 1)X + 2\alpha^2 + \alpha + 1 \\
&= X^3 + 2X^2 + X + 1.
\end{aligned}
$$

Thus, a primitive polynomial corresponding to the element β is a polynomial

$$f_\beta(X) = X^3 + 2X^2 + X + 1.$$

More examples of generating primitive polynomials on the basis of trinomials are given in Table 2.

4 Practical Application

A sample application is discussed by the modification of A5/1 algorithm which is used to encrypt data in GSM systems. Until recently the strength of the algorithm has been analyzed through several studies [1,2,3,5,6] and a number of serious weaknesses in the cipher have been identified. Currently known attacks can retrieve the data sequence in real-time basing only on the ciphertext. It is therefore possible to tap GSM mobile phone conversations and decrypt them.

A5/1 is based on a combination of three linear feedback shift registers with irregular clocking. In Figure 1 the internal structure of the algorithm was shown. The three shift registers are specified by fixed primitive polynomials over GF(2), as follows:

$$
\begin{aligned}
&\text{R1}: \ X^{19} + X^{18} + X^{17} + X^{14} + 1, \\
&\text{R2}: \ X^{22} + X^{21} + 1, \\
&\text{R3}: \ X^{23} + X^{22} + X^{21} + X^8 + 1.
\end{aligned}
$$

Table 2. Sample primitive polynomials generated on the basis of trinomials

k	$f(X) = X^{23} + 2X^3 + 1$
5	$X^{23} + X^{19} + X^{15} + 2X^{11} + 2X^7 + 2X^3 + 1$
7	$X^{23} + X^{14} + 2X^7 + X^5 + 2X^3 + 1$
11	$X^{23} + X^{17} + 2X^{10} + 2X^5 + 2X^4 + 2X^3 + 1$
13	$X^{23} + X^{12} + X^{10} + 2X^8 + 2X^4 + 2X^3 + X^2 + 1$
17	$X^{23} + X^{11} + X^7 + X^6 + X^5 + 2X^4 + 2X^3 + 1$
19	$X^{23} + X^{15} + 2X^{10} + 2X^7 + X^6 + 2X^5 + X^4 + 2X^3 + X^2 + 1$
k	$f(X) = X^{29} + 2X^4 + 1$
5	$X^{29} + X^{24} + X^{19} + 2X^{14} + 2X^9 + 2X^4 + 1$
7	$X^{29} + X^{13} + X^{10} + X^7 + 2X^4 + 1$
11	$X^{29} + 2X^{15} + X^{12} + 2X^9 + 2X^6 + 2X^4 + 2X^3 + 1$
13	$X^{29} + X^{21} + X^7 + X^5 + 2X^4 + 1$
17	$X^{29} + 2X^{19} + X^{18} + 2X^{17} + 2X^8 + 2X^5 + 2X^4 + 1$
19	$X^{29} + X^{22} + X^{17} + 2X^{15} + X^{13} + 2X^{10} + X^8 + 2X^6 + 2X^5 + 2X^4 + X^3 + 1$
k	$f(X) = X^{31} + 2X^5 + 1$
5	$X^{31} + 2X^5 + 2X^4 + 2X^3 + X^2 + X + 1$
7	$X^{31} + X^{14} + X^{11} + X^8 + 2X^5 + 1$
11	$X^{31} + X^{23} + 2X^{14} + 2X^7 + 2X^6 + 2X^5 + 1$
13	$X^{31} + 2X^{29} + 2X^{25} + X^{23} + 2X^{13} + X^{11} + X^7 + 2X^5 + 1$
17	$X^{31} + 2X^{24} + 2X^{20} + X^{17} + X^{13} + 2X^{10} + X^9 + 2X^6 + 2X^5 + 2X^3 + 1$
19	$X^{31} + X^{28} + 2X^{25} + 2X^{22} + X^{19} + 2X^{16} + 2X^5 + X^4 + 1$

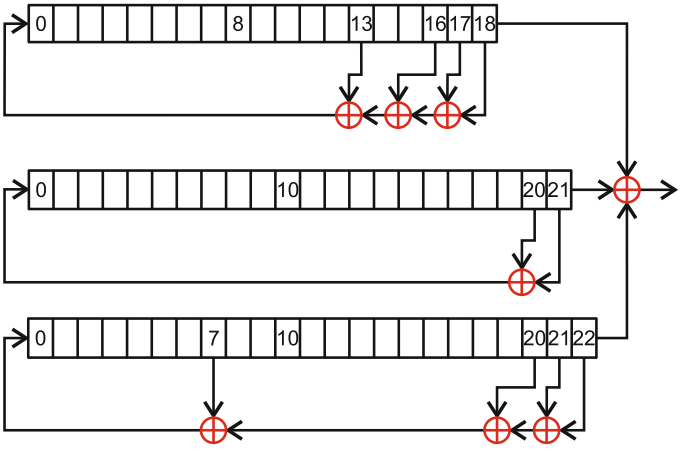

Fig. 1. Architecture of A5/1 algorithm

In [12], a modification of A5/1 algorithm has been proposed. The idea of improving the strength of the algorithm is a dynamic setting of linear feedback shift registers. This method is based on generating primitive polynomials over GF(2) and allows designers to eliminate many of the methods used to attack on the original A5/1 algorithm.

LFSRs in the structure of A5/1 are designed over binary arithmetics. However, such registers can also be considered over finite fields of a larger order, e.g. over GF(3). In this case we need two bits for each state and the addition modulo 3 feedback operation. Similarly to a binary field, some conditions of LFSR, such as period and linear complexity, depend on the feedback polynomial. In the case when the feedback polynomial is primitive, the register generates all possible states except the zero-state. It yields a period of $2^n - 1$ for a binary field and $3^n - 1$ for a ternary field. For $n = 32$, the maximal LFSR period over a binary field is 0,000231% of the maximal LSFR period over GF(3). Moreover, for the length of 32 the number of feedback polynomials which ensure the maximal register period equals 67108864 for GF(2) and for GF(3) 21158323814400, which is 0,000317% of primitive polynomials of degree 32 over a ternary field [11]. These are the arguments behind applying GF(3) instead of GF(2).

The original A5/1 algorithm controls shifting registers in a special way: the majority bit is calculated using selected bits of the registers, and subsequently the registers yielding compatible value of the selected bits to the value of the majority bit are shifted.

A new idea proposed by the authors is exchanging the majority function with the selector based on primitive polynomial over GF(3) which can be dynamically set in the device. It allows designers to extend the cryptographic power and capabilities of the old A5/1 algorithm. The new concept is presented in Figure 2.

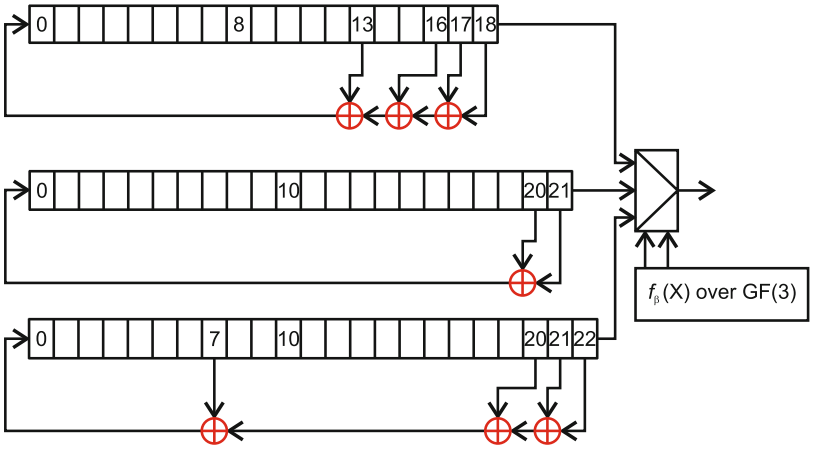

Fig. 2. Modified architecture of A5/1 algorithm

5 Conclusion

The specialized cryptographic system differs from the traditional one. It is characterized by a high security level. However, the implementation is relatively expensive and that is why one has to compromise these facts.

In this paper, we decrease the possibility of breaking into the system by limiting the knowledge of a cryptographic function. It is possible thanks to the replacement of the basic irreducible polynomial generating arithmetic of a finite field embedded in a cryptographic scheme.

Due to the fact that for a number field larger than GF(2), the probability of finding an irreducible lacunary polynomial grows, it can be assumed that fields GF(3), GF(5) and GF(7) will be applied in the near future.

References

1. Anderson, R., Roe, M.: A5 (1994), http://jya.com/crack-a5.htm
2. Babbage, S.: A space/time trade-off in exhaustive search attacks on stream ciphers. In: European Convention on Security and Detection. IEE Conference Publication 408 (May 1995)
3. Briceno, M., Goldberg, I., Wagner, D.: A pedagogical implementation of A5/1 (1999), http://cryptome.org/jya/a51-pi.htm#PI
4. von zur Gathen, J.: Irreducible trinomials over finite fields. In: Proc. of the 2001 International Symposium on Symbolic and Algebraic Computation, pp. 332–336. ACM, New York (2001)
5. Golić, J.D.: Cryptanalysis of alleged A5 stream cipher. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 239–255. Springer, Heidelberg (1997)
6. Hellman, M.E.: A cryptoanalytic time-memory trade-off. IEEE Transactions on Information Theory IT-26(4), 401–406 (1980)
7. MacWilliams, F.J., Sloane, N.J.A.: The Theory of Error-Correcting Codes. North-Holland Mathematical Library. North-Holland, Amsterdam (1988)
8. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press, Boca Raton (2001), http://www.cacr.math.uwaterloo.ca/hac/
9. Paszkiewicz, A.: Irreducible pentanomials and their applications to effective implementations of arithmetic in binary fields. Electronics and Telecommunications Quarterly 55(2), 363–375 (2009)
10. Paszkiewicz, A.: On some properties of irreducible polynomials over small number fields. Telecommunications Review and Telecommunication News 4, 129–135 (2009) (in Polish)
11. Paszkiewicz, A.: Trinomials which are irreducible over the number field GF(3). Telecommunications Review and Telecommunication News 8-9, 1767–1774 (2009)
12. Paszkiewicz, A., Stolarek, P.: A modification of A5/1 algorithm. Telecommunication Review and Telecommunication News 12, 1073–1075 (2008) (in Polish)