AES S-BOX USING FERMAT'S LITTLE THEOREM FOR THE HIGHLY CONSTRAINED EMBEDDED DEVICES

M. M. Wong [1] *M. L. D. Wong*[2] *A. K. Nandi*[3] *I. Hijazin*[4]

[1]SoECS, Swinburne University of Technology Sarawak, Kuching, Malaysia
[2]Dept. EEE, Xi'an Jiaotong-Liverpool University, Suzhou, China
[3] Dept. EE&E, The University of Liverpool, Liverpool, U.K.
[4] FEIS, Swinburne University of Technology, Victoria, Australia.

ABSTRACT

The recent increase of resource-constrained embedded devices have led to the need of lightweight cryptography. Therefore, the design of secure communication algorithms that fit in this highly constrained environments has become a fundamental issue in cryptographic circuit design. In this paper, we propose an optimization methodology that would efficiently reduces the code size of the S-box, the most expensive operation of the Advanced Encryption Standard (AES). Here, we perform a study on composite field AES S-box constructed using an inversion algorithm based on Fermat's Little Theorem (FLT). Consequently, we derive two AES S-box constructions over the fields $GF((2^4)^2)$ and $GF((2^2)^4)$ respectively. Our methodology results in smaller computational cost compared to the conventional Look-up Table (LUT) method, which is commonly deployed on microcontrollers.

Index Terms— Advanced Encryption Standard (AES), S-box, Fermat's Little Theorem (FLT), microcontroller (MCU), lightweight implementation

1. INTRODUCTION

Small embedded applications such as the smart cards, radio-frequency identification (RFID) tags, sensor nodes are often characterized by their strong low cost constraints in implementation. Therefore, microcontrollers (MCUs) are often used for such applications. As these applications manipulate on potentially sensitive data, a secure authentication protocols are highly required for security and privacy purposes. Therefore, the design of lightweight ciphers, that are suitable for these resource constrained MCU has become a very active research topic over the recent years.

The Advanced Encryption Standard (AES) [1] is currently the most widespread and also the algorithm of choice for symmetric encryption. However, this cipher is claimed to be unsuitable for 4-bit MCUs due to its expensive operations [2]. Furthermore, as AES manipulates on 8-bit data, it is better suited for 8-bit MCUs or other CPUs with larger datapath [3, 4]. These 8-bit architectures are commonly reported in the literatures and are mainly aimed at highly efficient AES implementation [5, 6]. On the contrary, there was no reported work on AES implementation in 4-bit MCUs until the recent work by Kaufmann and Poschmann [4] in 2011.

Instead of focusing on the clock speed optimization, our goal in this study is to investigate the feasibility of the standardized cryptography (specifically the AES) on the ultra-lightweight devices such as 4-bit MCUs. The contributions of this study are two-fold. First, we aim at further reducing the code size of the non-linear transformation, S-box, so that it is feasible and efficient in MCU platform. Rather than using direct employment of LUT, we employ the composite field arithmetic (CFA) that efficiently mapped the arithmetic of the AES S-box from $GF(2^8)$ to its isomorphic fields of the lower orders. With this, the arithmetic cost will be reduced accordingly and therefore, the code size would be minimized as well.

Second, to our best knowledge, this is the first study to present a detailed review of efficient multiplication inversion algorithm based on Fermat's Little Theorem (FLT) for CFA AES. Here, we present two CFA AES S-boxes using FLT over the field $GF((2^4)^2)$ and $GF((2^2)^4)$ respectively. In this study, we proved that there is a substantial gain in our approach, in term of code size reduction, compared to the direct LUT method as suggested in [4].

The rest of this paper is organized as follows: in Section 2, we describe the block cipher AES as well as the justification on code size reduction approach performed in this work. Next, our proposed AES S-boxes, using the FLT-based inversion algorithm, the Itoh-Tsujii inversion (ITI) algorithm are presented in Section 3. In Section 4, our results are discussed and benchmarked analytically with the previous work reported in [4]. The paper is concluded in Section 5.

*M. M. Wong thanks Swinburne Sarawak for supporting her PhD work.

2. COMPOSITE FIELD ARITHMETIC IN ADVANCED ENCRYPTION STANDARD

The Rijndael AES is the new encryption standard selected by the National Institute of Standards and Technology (NIST) in 2002 as the replacement of the Data Encryption Standard (DES). It is a symmetric block cipher with a constant block size of 128 bits (16 bytes) that supports the key lengths of 128, 192 or 256 bits. It performs encryption and decryption processes on an iterative basis where each iterative step is known as an operation round. The number of the operation round used is determined by the key length (10, 12 and 14 rounds for key lengths 128, 192 and 256 bits respectively).

Each operation round is consisted of four transformations, namely the **SubBytes** (applies non-linear S-box to the bytes of the states), **ShiftRows** (wire crossing), **MixColumns** (a linear diffusion layer) and **AddRoundKey** (a bitwise XOR or the round key). The round keys involved are generated from the secret key through an expansion routine that reuses the S-box in the SubBytes transformation. Previous studies have identified the SubBytes transformation as the bottleneck of the entire encryption process [7]. Structurally, the SubBytes transformation, also known as S-box consists of a multiplicative inverse over $GF(2^8)$ followed by an affine transformation. Note that the multiplicative inverse is the most expensive operation in finite field arithmetic. Consequently, AES appears to be an expensive choice, with a relatively large gate-count and code size when ultra-lightweight devices are involved.

There are two common AES S-box implementation approaches that have been reported in the literature, which is by using direct LUT method and by using combinatorial circuitry through CFA. The first approach is a conventional and straightforward way whereby the outputs for all possible input combinations are pre-tabulated in memory form. While it is simple in nature and often results in high throughput, the advantages are traded off with larger memory size consumption and hence having larger code size. Therefore, this method is rather unpractical for lightweight devices such as the 4-bit MCUs.

On the other hand, the second approach attempts on constructing combinatorial AES S-box circuit through CFA by using the field $GF(((2^2)^2)^2)$ for hardware implementation [8, 9, 10, 11, 12]. In this approach, the computation will be reduced to several algebraic manipulations (multiplications and multiplicative inverse) over its subfields, $GF((2^2)^2)$ and $GF(2^2)$. The approach successfully results in a compact architecture for AES S-box for hardware implementation but it manipulates on 2-bit data and involves complicated crossing and branching signal paths. As a result, the construction may not be best suited for MCU platforms.

Therefore, this leads to the need for the alternative that is both feasible and efficient for AES implementation on MCU. Here, we propose a new approach that employs both the CFA and the LUTs. We adopt CFA in the design such that the arithmetic of the higher field is efficiently reduced to the arithmetic of the lower order field. Meanwhile, in order to avoid the complicated crossing and branching signals, we employ LUT to perform the arithmetic of the lower order field. With this, the LUT required would be relatively smaller and hence reduces the code size required. In this work, we propose to employ the field of $GF((2^4)^2)$ for AES S-box.

It is worth noting that the above mentioned $GF(((2^2)^2)^2)$ CFA AES S-boxes employed the Extended Euclidean Algorithm (EEA) for the multiplicative inverse [8, 9, 10, 11, 12]. Apart from EEA, the FLT is another common tool for computing the multiplicative inverse. In this paper, we employ the multiplication inversion algorithm based on FLT, the Itoh-Tsujii inversion (ITI) algorithm, for our AES S-box design. As this approach was not reported in the previous works of AES, we also present the FLT-based CFA AES in the field $GF((2^2)^4)$ for comparison with the $GF((2^4)^2)$ construction. Overall, two FLT-based CFA AES S-boxes will be presented in this study.

3. THE PROPOSED AES S-BOXES USING FERMAT'S LITTLE THEOREM

FLT stated that for any non zero element A of field $GF(2^m)$, its multiplicative inverse can be computed as $A^{-1} \equiv A^{2^m-2} = A^2.A^{2^2} \cdots A^{2^m-1}$. Several finite field inversions based on FLT have been found in the literatures. One of which was the Itoh-Tsuji inversion algorithm (ITI) which is performed on composite field $GF((2^n)^m)$ in a standard basis representation [13, 14].

Though ITI does not perform a complete inversion, it efficiently reduces the extension field inversion to the inversion in the subfield $GF(q^n)$. We can readily obtain the inverse of $A \in GF((2^n)^m)$ based on Theorem 1.

Theorem 1. [13] *The multiplicative inverse of an element A of the composite field $GF((2^n)^m)$, with $A \neq 0$, can be computed by $A^{-1} = (A^r)^{-1}A^{r-1}$ mod $P(x)$ where $A^r \in GF(2^n)$ and that $r = (2^{nm}-1)/(2^n-1)$.*

The CFA AES S-boxes using ITI algorithm over $GF((2^4)^2)$ and $GF((2^2)^4)$ are presented in Section 3.1 and Section 3.2 respectively.

3.1. Inversion in Field $GF((2^4)^2)$

Let $GF((2^4)^2)$ be constructed using irreducible polynomials $P(x) = x^2 + x + w^{14}$ and $Q(y) = y^4 + y + 1$. The norm w^{14} is an element of $GF(2^4)$ which can be denoted as $\{1001\}_2$. For element $A(x) = \{A_1, A_0\} \in GF((2^4)^2)$, its inversion, A^{-1} is computed using the fours steps as follows.

Step 1: $A^{r-1} = A_i(x) = A^{16} = A^{2^4} = \{A_{i0}, A_{i1}\}$

$$\begin{bmatrix} A_{i0} \\ A_{i1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} A_0 \\ A_1 \end{bmatrix}$$

$$\begin{aligned}A_{i0} &= A_0 + A_1\\ A_{i1} &= A_1\end{aligned}$$

Step 2: $A^r = A^{r-1} \cdot A$

$$\begin{aligned}A^r &= A^{r-1}A\\ &= h_0 + h_2 w^{14}\\ h_0 &= A_0 A_{i0}\\ h_2 &= A_1 A_{i1}\end{aligned}$$

Derivation of h_0 and h_2 involved a multiplication over the subfield, $GF(2^4)$. Instead of performing direct $GF(2^4)$ multiplication, we choose to work on their respective power over the primitive element. Multiplication of finite elements is equivalent to addition of their respective powers over the primitive element.

As all the elements of $GF(2^n)$ form a cyclic group, the element $a_i \in GF(2^n)$ can be expressed as a multiple of a primitive element w, where $w_i = w^i$. All of the pair (w_i, i) can be stored in two tables, log-table sorted on the first component (w_i) and $antilog$-table sorted on the second component (i). Each of the tables took up 2^n of n bits, resulting in a total memory requirement of 64 bits LUT for the field $GF(2^4)$. Using these tables, the product of field elements $w_j, w_k \in GF(2^n)$ can be derived as,

$$w_j w_k = \text{antilog}\,[\log(w_k) + \log(w_k)] \pmod{2^n - 1}. \quad (1)$$

In summary, this technique requires determination of the power with respect to the primitive element (log conversion) that is equivalent to the operands. Next, addition is performed on the powers and the resultant value is mapped back to the field element ($antilog$ conversion). Based on (1), one multiplication over the subfield $GF(2^4)$ would require three LUTs.

Step 3: $(A^r)^{-1}$

In this case, we utilize a new $GF(2^4)$ inversion which is similar to our proposed $GF(2^4)$ multiplication. In general, for $w_x \in GF(2^n)$ the inversion operation can be expressed as,

$$w_x^{-1} = \text{antilog}\,[-\log(w_x)] \pmod{2^n - 1}. \quad (2)$$

Thus, the same log and $antilog$ conversions are required here. Let $B \in GF(2^4)$ denoted as w^x. Its inverse $B^{-1} = w^y$ is then,

$$\begin{aligned}B \cdot B^{-1} &= 1\\ w^x \cdot w^y &= 1\\ &= w^0\\ &\equiv w^{15}\end{aligned}$$

and therefore $y = 15 - x$. After the log conversion, y can be easily determined through subtraction and then followed by an $antilog$ conversion to obtain $(A^r)^{-1}$. Therefore, one inversion over the subfield $GF(2^4)$ would take up two LUTs.

Step 4:$(A^r)^{-1} \cdot A^{r-1}$

Last, two $GF(2^4)$ multiplications are required to multiply $(A^r)^{-1}$ from Step 3 and A^{r-1} from Step 2.

3.2. Inversion in Field $GF((2^2)^4)$

Let $GF((2^2)^4)$ be constructed using irreducible polynomials $P(x) = x^4 + x^3 + x^2 + w$ and $Q(y) = y^2 + y + 1$. The norm w is a primitive element of $GF(2^2)$ of which can be denoted as $\{10\}_2$. For element $A(x) = \{A_3, A_2, A_1, A_0\} \in GF((2^2)^4)$, its inversion, A^{-1} is computed using fours steps as stated in the following.

Step 1: $A^{r-1} = A_j(x) = A^{84} = A^{2^2} \cdot A^{2^4} \cdot A^{2^6}$

$$\begin{aligned}A^{2^2} &= A_I(x)\\ &= \sum_{i=0}^{3} A_i x^{i4}\\ &= A_0 + A_1x^4 + A_2x^8 + A_3x^{12}\end{aligned}$$

$$\begin{bmatrix}A_{I0}\\A_{I1}\\A_{I2}\\A_{I3}\end{bmatrix} = \begin{bmatrix}1 & \text{w} & 1 & 1\\ 0 & 0 & \text{w} & \text{w}\\ 0 & 1 & \text{w}^2 & \text{w}\\ 0 & 1 & 0 & \text{w}\end{bmatrix} \times \begin{bmatrix}A_0\\A_1\\A_2\\A_3\end{bmatrix}$$

$$\begin{aligned}A_{I0} &= A_0 + wA_1 + A_2 + A_3\\ A_{I1} &= wA_2 + wA_3\\ A_{I2} &= A_1 + w^2A_2 + wA_3\\ A_{I3} &= A_1 + wA_3\end{aligned}$$

$$\begin{aligned}A^{2^4} &= A_{II}(x)\\ &= \sum_{i=0}^{3} A_i x^{i16}\\ &= A_0 + A_1x^{16} + A_2x^{32} + A_3x^{48}\end{aligned}$$

$$\begin{bmatrix}A_{II0}\\A_{II1}\\A_{II2}\\A_{II3}\end{bmatrix} = \begin{bmatrix}1 & \text{w} & 1 & \text{w}\\ 0 & 0 & 1 & 0\\ 0 & 1 & 0 & 0\\ 0 & \text{w} & \text{w} & 1\end{bmatrix} \times \begin{bmatrix}A_0\\A_1\\A_2\\A_3\end{bmatrix}$$

$$\begin{aligned}A_{II0} &= A_0 + wA_1 + A_2 + wA_3\\ A_{II1} &= A_2\\ A_{II2} &= A_1\\ A_{II3} &= wA_1 + wA_2 + A_3\end{aligned}$$

$$\begin{aligned} A^{2^6} &= A_{III}(x) \\ &= \sum_{i=0}^{3} A_i x^{i64} \\ &= A_0 + A_1 x^{64} + A_2 x^{128} + A_3 x^{192} \end{aligned}$$

$$\begin{bmatrix} A_{III0} \\ A_{III1} \\ A_{III2} \\ A_{III3} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \mathrm{w}^2 \\ 0 & 1 & \mathrm{w}^2 & \mathrm{w} \\ 0 & 0 & \mathrm{w} & \mathrm{w} \\ 0 & \mathrm{w}^2 & \mathrm{w} & \mathrm{w} \end{bmatrix} \times \begin{bmatrix} A_0 \\ A_1 \\ A_2 \\ A_3 \end{bmatrix}$$

$$\begin{aligned} A_{III0} &= A_0 + A_1 + A_2 + w^2 A_3 \\ A_{III1} &= A_1 + w^2 A_2 + w A_3 \\ A_{III2} &= w A_2 + w A_3 \\ A_{III3} &= w^2 A_1 + w A_2 + w A_3 \end{aligned}$$

Eventually, we utilized Karatsuba's algorithm in [15] to perform multiplications of $A^{r-1} = A^{2^2} \cdot A^{2^4} \cdot A^{2^6}$.

Step 2: $A^r = A^{r-1} \cdot A$

$$\begin{aligned} A^r &= A^{r-1} A \\ &= h_0 + (h_4 + h_5) w \\ h_0 &= A_0 A_{j0} \\ h_4 &= A_1 A_{j3} + A_2 A_{j2} + A_3 t A_{j1} \\ h_5 &= A_2 A_{j3} + A_3 A_{j2} \end{aligned}$$

Derivation of h_0, h_4 and h_5 involved multiplication over subfield, $GF(2^2)$. In this case, we use direct multiplication since $GF(2^2)$ is a relatively small field. Let $h(x), g(x) \in GF(2^2)$ and that $f(x) = h(x) \cdot g(x)$ is defined as,

$$\begin{aligned} f_1 &= (g_1 + g_0)(d_1 + d_0) + (g_0 d_0) \\ f_0 &= g_1 d_1 + g_0 d_0 \end{aligned}$$

Step 3: $(A^r)^{-1}$

Here we utilized EEA for $GF(2^2)$ inversion. For $A^r = \{g_1, g_0\}$, its inversion, $(A^r)^{-1} = \{d_1, d_0\}$ is computed as,

$$\begin{aligned} d_1 &= g_1 \\ d_0 &= g_1 + g_0 \end{aligned}$$

Step 4:$(A^r)^{-1} \cdot A^{r-1}$

Last but not the least, two $GF(2^2)$ multiplication is required to multiply $(A^r)^{-1}$ from Step 3 and A^{r-1} from Step 2.

4. RESULTS AND DISCUSSION

In this section we are going to review and analyze the complexity of the proposed two AES S-box constructions. In architectural level, both of the constructions are as depicted in Figure 1 and 2. Our implementations are benchmarked with the direct LUT approach reported in [4] and the analytical results are summarized in Table 1.

The direct LUT approach is the conventional way of AES S-box implementation whereby all the 256 of 8-bit data are hard-coded in memory form. This approach would therefore require a total memory allocation of $2,048$-bit for the S-box, which is considerably large for ultra-lightweight devices. On the other hand, we proposed a methodology that employed CFA, of which the arithmetic are reduced to the field of $GF(2^4)$. Due to the utilized FLT-based algorithm, the generalized ITI algorithm, the construction requires only a total of 14 64-bits LUT. Therefore, only a total of 896-bit memory is required, which less than half of what is required in the direct LUT approach. Apart from the LUTs, the operation also requires 8 XORs and 6 additions/substractions. Therefore, our approach has successfully minimizes the computational costs and this would reduce the code size required in the implementation.

AES S-box of the field $GF(2^8)$ requires large LUT in hardware implementation. Meanwhile, the pure combinatorial CFA over $GF(((2^2)^2)^2)$ is complicated and manipulates on 2-bit data. Therefore both of the fields are deemed to be inefficient for MCU platform. On the other hand, we proposed the use of $GF((2^4)^2)$ that would result in small code size, efficient and feasible for 4-bit MCU implementation.

Table 1. Complexity of AES S-box using direct LUT approach and our proposed FLT-based inversion

Work	Total Gate				
	AND	XOR	Adder/Subtractor	LUT (64 bits)	LUT (2,048 bits)
LUT approach [4]	-	-	-	-	1
Our $GF((2^4)^2)$	-	8	6	13	-
Our $GF((2^2)^4)$	59	98	-	-	-

5. CONCLUSION

In this study, we present an efficient and compact AES S-box that is feasible for MCU implementations. Our work employed the CFA that maps the original $GF(2^8)$ to its isomorphic field of $GF((2^4)^2)$. By further employing the FLT based inversion algorithm, the total size of the LUT required can be reduced efficiently from $2,048$ bit to 896-bit. In addition to that, the arithmetic (in the field $GF(2^4)$) can then be efficiently manipulated on the 4-bit MCU platform as well. To our best knowledge, this is the second work that studies the optimization of AES on 4-bit MCU besides [4]. We have theoretically reviewed that our optimization managed to promote

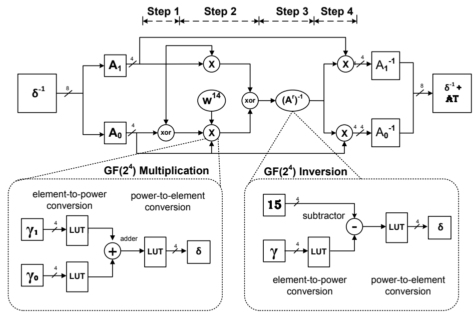
Step 1
Step 2
Step 3
Step 4
GF(2⁴) Multiplication
element-to-power conversion
power-to-element conversion
adder
LUT
GF(2⁴) Inversion
subtractor
element-to-power conversion
power-to-element conversion

Fig. 1. AES S-box over $GF((2^4)^2)$ and using generalized ITI algorithm

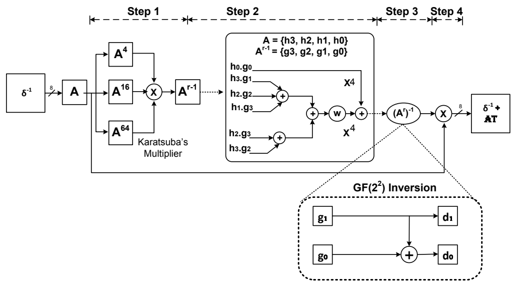

Fig. 2. AES S-box over $GF((2^2)^4)$ and using generalized ITI algorithm

further code size reduction compared to the method proposed in [4]. For future work, we would like to extend this work by implementing our designs on a chosen 4-bit MCU platform.

6. REFERENCES

[1] *Advanced Encryption Standard (AES), FIPS PUB 197*, Federal Information Processing Standard Publication 197. Nov.26, 2001.

[2] M. Krause and M. Hamann, "The cryptographic power of random selection.," *IACR Cryptology ePrint Archive*.

[3] M. Vogt, A. Poschmann, and C. Paar, "Cryptography is feasible on 4-bit microcontrollers - a proof of concept," in *RFID, 2009 IEEE International Conference on*, april 2009, pp. 241 –248.

[4] T. Kaufmann and A. Poschmann, "Enabling standardized cryptography on ultra-constrained 4-bit microcontrollers," in *ECRYPT Workshop on Lightweight Cryptography, Louvain-la-Neuve, Belgium*, November 28-29, 2011, pp. 255–276.

[5] S. Rinne, T. Eisenbarth, and C. Paar, "Performance Analysis of Contemporary Light-Weight Block Ciphers on 8-bit Microcontrollers," in *ecrypt workshop SPEED - Software Performance Enhancement for Encryption and Decryption*, 2007, to appear.

[6] *AN821: Advanced Encryption Standard Using the PIC16XXX*, Microchip Technology Inc. 2002.

[7] S. Mathew, F. Sheikh, A. Agarwal, M. Kounavis, S. Hsu, H. Kaul, M. Anders, and R. Krishnamurthy, "53Gbps native $GF(2^4)^2$ composite-field AES-encrypt/decrypt accelerator for content-protection in 45nm high-performance microprocessors," in *VLSI Circuits (VLSIC), 2010 IEEE Symposium on*, 2010, pp. 169 –170.

[8] A. Satoh, S. Morioka, K. Takano, and S. Munetoh, "A compact Rijndael hardware architecture with S-box optimization," in *Proc. ASIACRYPT*, Gold Coast, Australia, Dec. 2000, pp. 239–245, Springer-Verlag.

[9] D. Canright, "A very compact Rijndael S-box," Tech. Rep. NPS-MA-04-001, Naval Postgraduate School, 2005.

[10] X. Zhang and K. K. Parhi, "On the optimum constructions of composite field for the AES algorithm," vol. 53, no. 10, pp. 1153–1157, 2006.

[11] J. Boyar and R. Peralta, "A new combinational logic minimization technique with applications to cryptology," in *Experimental Algorithms*, Paola Festa, Ed., vol. 6049 of *Lecture Notes in Computer Science*, pp. 178–189. Springer Berlin / Heidelberg.

[12] M. M. Wong, M. L. D. Wong, A. K. Nandi, and I. Hijazin, "Construction of optimum composite field architecture for compact high-throughput aes s-boxes," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. PP, no. 99, pp. 1 –5, 2011.

[13] J. Guajardo and C. Paar, "Efficient algorithms for elliptic curve cryptosystems," *In B. Kaliski, (ed.) Advances in Cryptology- CRYPTO 97, Lecture Notes in Computer Science*, vol. 1924 Berlin, pp. 342–356, 1997.

[14] J. Guajardo and C. Paar, "Itoh-Tsujii inversion in standard basis and its application in cryptography and codes," *Designs, Codes and Cryptography*, vol. 25, pp. 207–216, 2002, 10.1023/A:1013860532636.

[15] A. Karatsuba and Yu Ofman, "Multiplication of many-digital numbers by automatic computers," *Doklady Akad. Nauk SSSR*, vol. 145, no. 293-294, pp. 85, 1962.