INTERNATIONAL JOURNAL OF MATHEMATICAL MODELS AND METHODS IN APPLIED SCIENCES
DOI: 10.46300/9101.2020.14.7 Volume 14, 2020

On the Implementation of Lattice-Based Identification Schemes

Azhar Murzaeva, Sedat Akleylek, Mesut Döngül, Erdal Kılıç

***Abstract*—Identification is one of the important concerns of information security, that is widely used in our daily e-systems to approve authorised users. With the advent of quantum computers, development of quantum secure identification schemes is essential. In this paper, we give the implementation details of quantum secure Kawachi's and Cayrel's identification schemes performed in JavaScript. The hardness of these schemes is based on lattice-based problem SIS in post-quantum cryptography, which requires matrix-vector product operations for its execution. It's important that for efficient implementation choosing an algorithm with low complexity needs more careful. Therefore, in identification schemes chosen for this study, we use algorithms specific to those schemes' parameter properties. Then, we carry out matrix by sparse vector and sparse matrix by vector product operations. We provide experimental results of both standard and property-specific algorithms' execution with their comparison. According to the experimental results, we receive improvements in the specific implementations.**

***Index Terms*—post-quantum cryptography, lattice-based cryptography, identification schemes, software implementation**

I. Introduction

Information security concepts are the main policy for data protection. This policy consists of these objectives: confidentiality, integrity, availability, authentication and non-repudiation. Confidentiality assures that rights to share the private information belong to the authorized user; integrity is about that information must be changed only in an authorized manner; availability is about providing a service to the authorized users; authentication is a corroboration of the user, also known as identification, and non-repudiation is about preventing a user of denying his/her actions [15]. These concepts are crucial for a secure communication and individual access must be controlled. This study concerns the identification area, which is ensured by identification schemes.

Identification is widely used in credit cards, ID cards, passwords to provide an access to the e-services (such as e-goverment, e-payment etc.) in a safe and easy way. It ensures the truth of identity by either accepting or rejecting its proof [1]. Futhermore, identification is a fundamental base for digital signatures. That is why, identification is one of the important aims of security.

With the wide use of low resource devices (in terms of power source and memory size) which have limited computing capacity, identification schemes must be applied in an efficient way. In addition, since the development of quantum computers causes threat for traditional cryptosystems, new post-quantum cryptographic protocols are being researched and developed. Among these developing cryptographic protocols, identification schemes also must be constructed in a way to be quantum resistant.

Current identification schemes are based on computationally hard problems like integer factorization, discrete logarithm problem [2]; however, it was shown that they can be solved in polynomial time by Shor's algorithm which was proposed for a quantum computer [3]. Thus, many researches and developments on post-quantum cryptosystems has emerged. Moreover, as a result of NIST's call for a quantum-resistant algorithms' standardization project [4], these developments continue to increase.

NIST's announcement of request for new ideas in public key post-quantum cryptographic algorithms includes digital signature, public key encryption and key establishment algorithms. Altogether with proposed algorithms, their implementations in C are presented and some of them have developed cryptographic libraries for post-quantum areas depending on relying problems. Practical imlementation and library for Key Encapsulation Mechanism (Frodo) [16]; implementation of secure encryption, key exchange and authenticated key exchange (Kyber) [17]; implementation of post-quantum digital signature algorithm (Falcon) [18] and other implementations of non lattice-based cryptosystems are examples to this.

Originally, digital signatures are just another form of identification schemes. Therefore, quantum secure identification schemes are essential and their implementations are needed. Our main goal is to develop an open source cryptographic library for quantum secure identification schemes and in this study, we introduce efficient execution of few lattice-based identification schemes.

A. Motivation and Contribution

Lattice-based cryptography is one of the most important candidate for the standardization of quantum secure cryptosystems. During this work identification schemes that depend on computational problems based on lattices are reviewed. Practical implementation of these schemes is as important as theoretical proof of their secureness. There are some works on implementation of lattice-based Signature Schemes [11]. Boorghany's implementation of lattice-based identification protocols are performed on smart cards and microcontrollers [12]; Lyubashevsky [13] and Guneysu [14] implemented their signature schemes. There are also several implementation of zero-knowledge identification schemes that depend on code-based cryptosystem [10]. Thus, we perform the implementation of Kawachi's [5] and Cayrel's [6] identification schemes on Javascript.

The authors are with theOndokuz Mayis University, Bilgisayar Muhendisligi Bolumu, Atakum, Samsun, Turkey

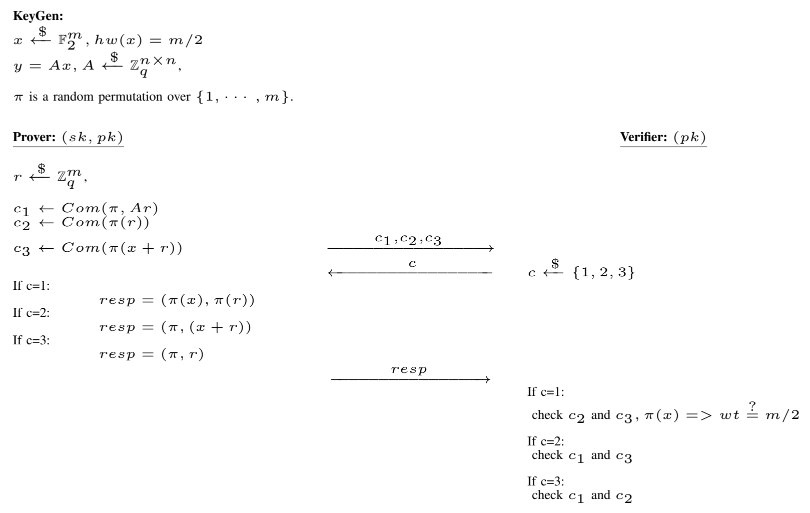

KeyGen:
$x \xleftarrow{\$} \mathbb{F}_2^m, hw(x) = m/2$
$y = Ax, A \xleftarrow{\$} \mathbb{Z}_q^{n \times n},$
π is a random permutation over $\{1, \cdots, m\}$.
Prover: (sk, pk)
Verifier: (pk)
$r \xleftarrow{\$} \mathbb{Z}_q^m,$
$c_1 \leftarrow Com(\pi, Ar)$
$c_2 \leftarrow Com(\pi(r))$
$c_3 \leftarrow Com(\pi(x + r))$
c_1, c_2, c_3
c
$c \xleftarrow{\$} \{1, 2, 3\}$
If c=1: $resp = (\pi(x), \pi(r))$
If c=2: $resp = (\pi, (x + r))$
If c=3: $resp = (\pi, r)$
$resp$
If c=1: check c_2 and c_3, $\pi(x) => wt \overset{?}{=} m/2$
If c=2: check c_1 and c_3
If c=3: check c_1 and c_2

Fig. 1. Kawachi's IDscheme

The main reason of selecting the Javascript programming language is its wide usage in web applications that provides an access for more users through both mobile devices and desktop and it doesn't require any modification of code to execute it on any JavaScript runtime environment. It provides a crossplatform development of applications that will lead to an easy integration of the code to any platform. In addition to this, the use of technologies such as NFC increased utilization rate and the need for mobile applications where JavaScript is used widely.

With this in mind, we implement lattice-based identification schemes by deploying better approaches that depend on the parameters of the schemes' algorithms. These approaches are specific to the identification scheme and integrated to the existing operations.

B. Organisation

The rest content of this paper is organized as follows. Section II presents identification schemes selected for this study. In this section, parameter properties and needed algorithms with their complexity analysis are discussed. Section III introduces implementation details. Section IV concludes this study by presenting and comparing obtained results.

II. Lattice-based Identification Schemes

In this section, we explain the main structure of an identification scheme. Kawachi's and Cayrel's identification schemes are described in details. Required algorthms for their implementation are reviewed and algorithm's complexities are analyzed.

An identification scheme can be considered as an algorithm that is composed of key generation and commitments' computation steps. Prover (P) and Verifier (V) are basic entities that take a main role in this interactive identification proof system. Let us express what the main components, they are: Key Generetaion function generates public and secret keys (pk, sk), while prover and verifier use them (V uses pk, P uses both pk and sk) for the next computational steps to perform an identification process. Later, verifier (V) announces a verification result. We summarize the components of identification schemes as follows:

Key Generation: The public and private keys are obtained.
Prover's computatios: Operations for computing commitments are performed.
Verifier's computatios: Operations for computing some results for comparison of commitments are being executed.

For this study, such properties as operation cost, asymptotic complexity, size of received and sent data are taken into consideration during the selection of identification schemes. Subsequently, Cayrel's and Kawachi's identification schemes are selected. Both of those schemes are based on SIS (Short Integer Solution) problem in lattices.

Definition 1: Short Integer Solution ($\mathrm{SIS}_{\beta,q,m}$)
Given a uniform A$\in \mathbb{Z}_q^{m \times n}$, find x $\in \mathbb{Z}^m \backslash 0$ such that:

$$||x|| \leq \beta \text{ and } \mathrm{x}^T \cdot A = 0 mod \mathrm{q}$$

A. Kawachi's Identification scheme

Kawachi et. al. [5] proposed a scheme which consists of functions for Key Generation stage and stage of computing Prover's and Verifier's computations that are given in Figure 1. Firstly, private and public keys are generated in Key Generation stage. Using these keys, Prover and Verifier sides perform some computations in an interactive way: Prover computes commitments and sends them to the Verifier. Verifier generates a challenge and depending on this challenge, Prover sends some parameters. Then, Verifier checks the truthfulness of the statement.

This identification scheme takes hardness of SIS (Shortest Integer Solution) problem [9] and it's composed of three

KeyGen:
$x \xleftarrow{\$} \mathbb{F}_2^m, hw(x) = m/2$
$A \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$,
$y \leftarrow \text{Ax} mod p$

Prover: **Verifier:**

$u \xleftarrow{\$} \mathbb{Z}_q^m, z \xleftarrow{\$} P_\sigma \text{x}$,
$\sigma \xleftarrow{\$} S_m$,
$P_\sigma^{m \times m} \leftarrow$ is a binary matrix
$r_0 \xleftarrow{\$} 0, 1^n, r_1 \xleftarrow{\$} 0, 1^n$.
$c_0 \leftarrow Com(\sigma || Au; r_0)$;
$c_1 \leftarrow Com(z || P_\sigma u; r_1)$;

$\xrightarrow{c_0, c_1}$

$\xleftarrow{\alpha}$ $\alpha \xleftarrow{\$} \mathbb{Z}_q$

$\beta \leftarrow P_\sigma(u + \alpha \text{x})$

$\xrightarrow{\beta}$ $b \xleftarrow{\$} \{0, 1\}$

$\xleftarrow{\text{challenge b}}$

If b=0: $resp = (\sigma, r_0)$
else: $resp = (z, r_1)$

$\xrightarrow{resp}$

If b=0:
check $c_o \overset{?}{=} com(\sigma || A P_\sigma^{-1} \beta - \alpha \text{y}; r_0)$
$\sigma \overset{?}{\in} S_m$
else:
check $c_1 \overset{?}{=} com(z || \beta - \alpha z; r_1)$
$z \overset{?}{\in} \{0, 1\}^m$
$hw(z) \overset{?}{=} m/2$

Fig. 2. CLRS (Cayrel's) IDscheme

passes. The key generation step contains a matrix-vector product and a random permutation function. In subsequent steps matrix-vector product and addition of vectors are required.

B. CLRS Identification scheme

Cayrel et. al. [6] proposed a scheme that is based on SIS (Shortest Integer Solution) problem in lattices. This scheme consists of five phases. As computational functions it contains a sparse matrix-vector multiplication and matrix inversion. It's demonstrated in Figure 2. Firstly, private and public keys are generated. Using these keys, Prover and Verifier sides perform some computations in an interactive way: Prover computes commitments and sends them to the Verifier. Verifier sends α. Then, Prover computes β and sends it to the Verifier. In its turn, Verifier generates a challenge. Depending on this challenge, Prover sends some parameters and Verifier checks the truthfulness of the statement.

C. Required Algorithms

For the realization of these schemes a "commitment function" in demand and for that, a one way hash function is used.

We are concerned here to enhance the efficiency of the identification scheme's implementation on practic and focus on its functions. For instance, we can see that Kawachi's identification scheme contains operations like matrix-vector product, vectors' addition (Figure 1) and Cayrel's identification scheme contains matrix-vector product, vectors' addition, vectors' subtraction, vector's multiplication by scalar, matrix inversion etc. (Figure 2). Depending on these criteria we draw up needed functions for implementing these schemes. Blueprint of designed functions for Kawachi's and Cayrel's identification schemes are given in Figure 3 and Figure 4 respectively.

In addition, we pay attention to the properties of defined parameters. To implement Kawachi's scheme (Figure 3), vector-matrix product is needed, where vector x is uniformly random at F_2^m (Figure 1). Something similar is observed in Cayrel's scheme: one of expensive functions for implementation of Cayrel's identification scheme is the vector matrix product in Prover's (Step 2) comptutations (Figure 4), where one of parameters is defined as a binary matrix (P_σ) in Figure 2. Due to the proper selection of algorithms considering these mentioned requirements, we obtain more efficient results.

D. Complexity Analysis

For a long time, different efficient techniques for the algorithms of computational problems have been developed. Schoolbook and Karatsuba-Ofman [7] algorithms are well known examples to this. Schoolbook (a.k.a. standart multiplication) algorithm is for a polynomials' multiplication with a complexity cost $O(n^2)$, whilst the Karatsuba algorithm is used for both polynomials' multiplication and a fast multiplication of big numbers with a better complexity $O(n^{log_2 3})$.

Looking back to such examples, we see that delpoyment of a better algorithm saves time and demonstrates a better performance in practical usage. Depending on the parameter properties used in those schemes, we run matrix-vector product using Hamming Weight (HW) additions that sped up the computation process. In Kawachi's scheme multiplication of

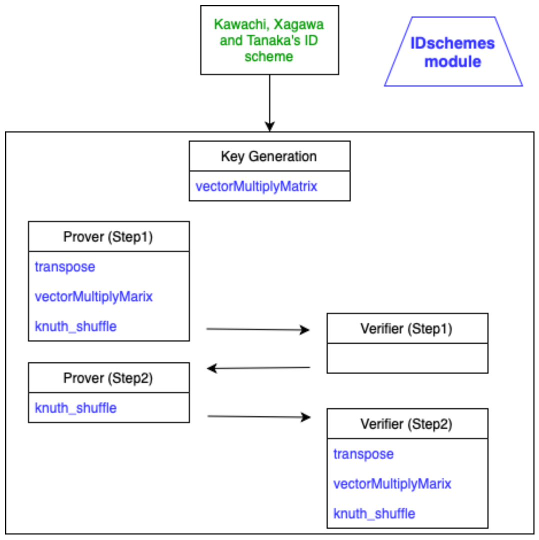


Fig. 3. Kawachi's implementation design

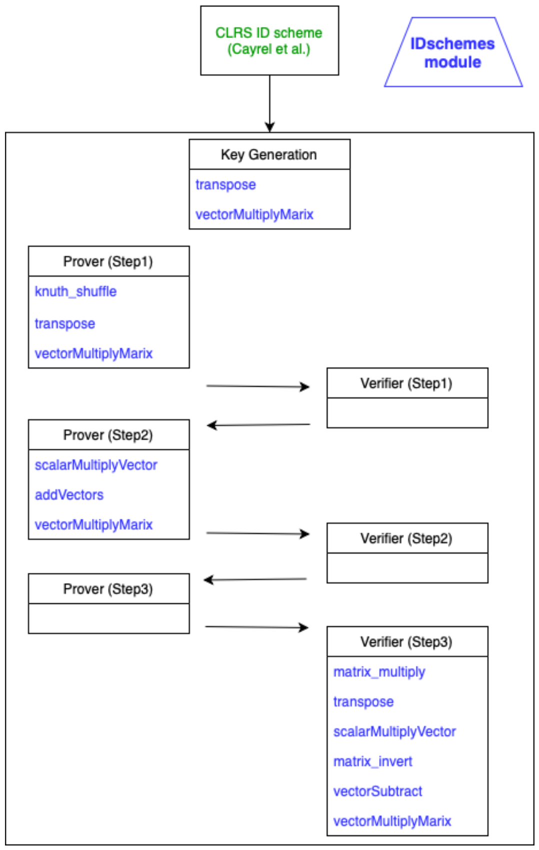


Fig. 4. Cayrel's implementation design

matrix-vector is required. In case of using the standart method (multiplying two dimensional array by one dimensional) for its imlementation, it performs nxn multiplications and n additions. Thus its time complexity is O(n^2). However, in this work implementation of algorithm depending on properties of a system is proposed and it performs HW(x) or n/2 additions n times.

For Cayrel's identification scheme multiplication of a binary matrix by vector is required. Using a standart schoolbook method performs nxn multiplications and n additions. Its time complexity is O(n^2). In this work, an algorithm specific to the binary matrix is proposed and it performs nx(n/16) additions. We see the decrease from the quadratic time complexity to the linear, that is a desired result for execution of code. A pseudocode of proposed sparse matrix-vector product is given in Algorithm 1.

Algorithm 1 sparseMatrixVectorProduct(B, a, ids)

```
Require: $P_\sigma$ is a binary matrix with dimension $m \times m$
Ensure: $t$ vector with length $m$
  for $i = 0$ to $M$ do
    $t[i] \leftarrow 0$
  end for
  for $i = 0$ to $M$ do
    $temp = 0$
    for $j = 0$ to $ids.length$ do
      $temp+ = a[ids[i][j]]$
    end for
    $t[i]+ = temp$
  end for
  return $t$
```

III. Implementation Details

In this section, details for each identification scheme with its parameter properties are given. In Kawachi's and Cayrel's identification schemes matrix-sparse vector product and sparse matrix-vector product operations are used. These operations are executed in the Key Generation and Prover's commitment computation steps. Prior condition for the efficiency of algorithms proposed for those operations in this study is, there must be a matrix with dimensions $m \times m$ filled with random values that consist of $(m/16)$ 1s for the sparseMatrixVectorProduct and a vector x with $m/2$ 1s for the sparseVectorMultilyMatrix functions. Details of each scheme are described in subsections.

A. *Kawachi's Identification scheme*

Parameter properties of Kawachi's scheme are defined in its Key Generation step. They are the matrix A with dimensions $n \times m$ filled with random values in modulo q and a vector x with length m that should contain $(m/2)$ 1s $(HW(x) = m/2)$. Then, the matrix A and vector x's multiplication must be computed.

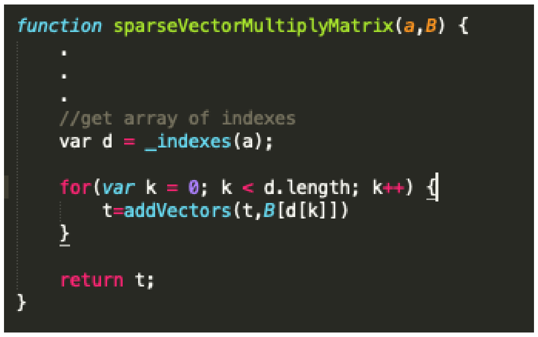
Instead of multiplying n^2 times to compute A matrix-vector product x (or Ax), we propose to carry out this multiplication depending on its properties. x is a sparse vector with lentgh m, where $m/2$ elements are 1s and the rest of elements in x are 0. Since result of multiplying by 0 is 0, we propose to use values corresponding to the indexes of elements with value 1.

Our function for a sparse vector-matrix product is given below. This function is used in Key Generation step of the scheme to obtain a public key y (Figure

1). The main organization of functions and place of the *sparseVectorMatrixProduct* function in it are given:

```
KeyGeneration:
        generateXvector(); //secret key
        generateAmatrixNxM(); //public key
        computeY() { //public key
            sparseVectorMatrixProduct();
        }
ProverStep1:
        generateRvector();
        computeC1();
        computeC2();
        computeC3();
VerifierStep1:
        generateCH();
ProverStep2:
        sendRespond();
VerifierStep2:
        check();
```

Complexity of this computation will lessen to $hw(m/2)$ additions. The code is given in Figure 5 as follows.



```
function sparseVectorMultiplyMatrix(a,B) {
    .
    .
    .
    //get array of indexes
    var d = _indexes(a);

    for(var k = 0; k < d.length; k++) {
        t=addVectors(t,B[d[k]])
    }

    return t;
}
```

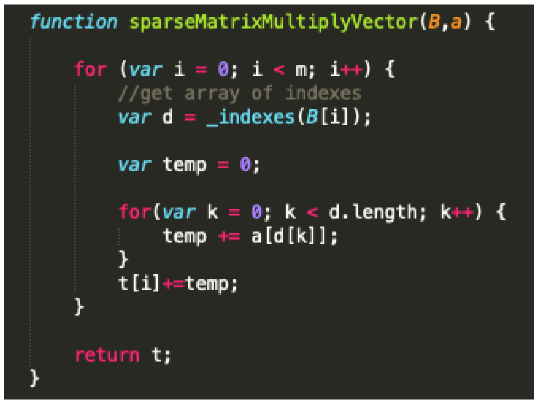
Fig. 5. Sparse Vector-Matrix Product

In that piece of code, d is an array of indexes, that address to the elements in x with value 1. Then, the addition of corresponding elements is performed. Thus, comparing to walking through all elements, using just needful elements saves the time.

B. *Cayrel's Identification scheme*

Parameter properties of Cayrel's scheme are defined in its Key Generation step and in Prover's step (see Figure 1). They are the matrix A with dimensions $n \times m$ filled with random values in modulo q and a vector x with length m that should contain $(m/2)$ 1s $(HW(x) = m/2)$. Then, the matrix A and vector x's multiplication must be computed. In the Prover's step $P_\sigma x$ must be computed. The parameter P_σ is a binary matrix with dimension $m \times m$, where each row contains $n/16$ 1s.

To compute Ax, we use the function sparseVectorMatrixProduct that is mentioned in Kawachi's implementation and



```
function sparseMatrixMultiplyVector(B,a) {

    for (var i = 0; i < m; i++) {
        //get array of indexes
        var d = _indexes(B[i]);

        var temp = 0;

        for(var k = 0; k < d.length; k++) {
            temp += a[d[k]];
        }
        t[i]+=temp;
    }

    return t;
}
```

Fig. 6. Sparse Matrix-Vector Product

to compute $P_\sigma x$ multiplication the function given in Figure 6 is used.

P_σ is a binary matrix. Thus, to multiply P_σ by x vector $(P_\sigma x)$, we propose to use elements of matrix with value 1. Indexes of elements with value 1 from a each row are keeped in array and then, the addition of those elements is performed. Array of these sums is our desired multiplication result. The main organization of functions and place of the *sparseMatrixVectorProduct* function in it is shown below:

```
KeyGeneration:
        generateXvector(); //secret key
        generateAmatrixNxM(); //public key
        computeY() { //public key
            sparseVectorMatrixProduct();
        }
ProverStep1:
        generateUvector();
        generateRvectorx();
        computeZ();{ // z ← $P_\sigma$x
            sparseMatrixVectorProduct();
        }
        computeC0();
        computeC1();
VerifierStep1:
        generateAlpha();
ProverStep2:
        computeBeta();
VerifierStep2:
        generateCH();
ProverStep3:
        sendRespond();
VerifierStep3:
        check();
```

C. *Experimental Results*

Parameter set values for implementation of Kawachi's and Cayrel's identification schemes are given in Table I. These parameter values provide a 80-bit security level. Implementation was run on macOS environment, Mojavi version 10.14.Implementation results are discussed in Table III-C.

TABLE I
SYSTEM PARAMETERS FOR BOTH KAWACHI'S AND CAYREL'S IDENTIFICATION SCHEMES

Parameter	Value
n	512
m	2048
q	257
Commitment length	256-bits

Proceedingly, we entegrated our *sparseVectorMatrixProduct* and *sparseMatrixVectorProduct* functions that are mentioned in Implementation Details section. These functions were implemented to save time without wasting it for multiplying all elements in a matrix, whilst there is a chance to use just corresponding elements. Experimentaln results are demonstrated in Table II (running time is measured in milliseconds(ms)).

TABLE II
IMPLEMENTATION RESULTS AFTER OUR CONTRIBUTION

Functions	Kawachi's IDscheme (ms)	Cayrel's IDscheme (ms)	Improvement (%)
vectorbyMatrix (standard)	9	-	
sparseVectorbyMatrix (proposed)	10	-	-
matrixbyVector (standard)	-	19	
sparseMatrixbyVector (proposed)	-	10	52.6%

The implementation of identification schemes presented in this study is available at https://github.com/msAzhar/pqc-id_schemes/

IV. CONCLUSION

In this study, we present a sparse vector by matrix multiplication and binary matrix by vector multiplication for Kawachi's and Cayrel's identification schemes. Firstly, we run those schemes using traditional ways of computing matrix and vector multiplication. As a traditional way we first implement the schoolbook method. For efficient implementation we consider a property-specific algorithm.

From Table II, we can see that implementation results of Kawachi's identification scheme are same, however, there are some enhancements in implementation of Cayrel's identification scheme. By using a *sparseMatrixVectorProduct* function, Cayrel's scheme's execution performance got better. Computation is accelerated approximately for 52.6%.

ACKNOWLEDGMENT

This research was partially supported by TUBITAK under grant no.EEEAG-117E636.

REFERENCES

[1] U. Feige, A. Fiat, A. Shamir, "Zero-knowledge proofs of identity," J. Cryptology 1988; pp. 77–94.
[2] A. Fiat, A. Shamir, "How to prove yourself: practical solutions to identification and signature problems," Advances in Cryptology — CRYPTO'86; 1987; pp. 186-194.
[3] P.W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," SIAM J Comput 1997; 26: 1484-1509.
[4] L. Chen, S. Jordan, Y.K. Liu, et al., "Report on post-quantum cryptography," National Institute of Standards and Technology; 2016.
[5] A. Kawachi, K. Tanaka, and K. Xagawa, "Concurrently Secure Identification Schemes Based on the Worst-Case Hardness of Lattice Problems," J. Advances in Cryptology - ASIACRYPT 2008; pp. 372–389.
[6] P.-L. Cayrel, R. Lindner, M. Ruckert, and R. Silva, "Improved Zero-Knowledge Identification with Lattices," J. Tatra Mountains Mathematical Publications, vol: 53, 2010; pp. 1-17.
[7] A. Karatsuba and Y. Ofman, "Multiplication of multidigit numbers on automata," J. Soviet physics doklady, vol: 7, 1963; p. 595.
[8] S. Winograd, "Arithmetic Complexity of Computations," CBMS-NSF Regional Conference Series in Applied Mathematics, 1980; p. 5. doi:10.1137/1.9781611970364
[9] M. Ajtai, "Generating Hard Instances of Lattice Problems", J. Electronic Colloquium on Computational Complexity (ECCC), vol:3, 1996.
[10] P.-L. Cayrel, S.M.E. Yousfi Alaoui, F. Gunther, G. Hoffmann and H. Rother, "Efficient implementation of code-based identification schemes," In: Security Engineering and Intelligence Informatics. CD-ARES 2013. Lecture Notes in Computer Science, vol: 8128. Springer, Berlin, Heidelberg; pp. 122-136.
[11] R. E. Bansarkhani and J. A. Buchmann, "Improvement and Efficient Implementation of a Lattice-Based Signature Scheme," Selected Areas in Cryptography 2014; pp. 48-67. doi: 10.1007/978-3-662-43414-7
[12] A. Boorghany and R. Jalili, "Implementation and Comparison of Lattice-based Identification Protocols on Smart Cards and Microcontrollers", 2014.
[13] V. Lyubashevsky. F. Shamir, "Applications to lattice and factoring-based signatures," Advances in Cryptology –ASIACRYPT 2009, Lecture Notes in Computer Science; Springer Berlin Heidelberg, 2009; pp. 598–616.
[14] T. Guneysu, V. Lyubashevsky and T.Poppelmann, "Practical lattice-based cryptography: A signature scheme for embedded systems," Cryptographic Hardware and Embedded Systems –CHES 2012, Lecture Notes in Computer Science; Springer Berlin Heidelberg, 2012; pp. 530–547.
[15] J. Menezes, P. C. Oorschot, S. A. Vanstone, (1996), Handbook of Applied Cryptography, http://www.cacr.math.uwaterloo.ca/hac/
[16] J. Bos, C. Costello, L. Ducas, I. Mironov, M. Naehrig, V. Nikolaenko, A. Raghunathan, D. Stebila, "Frodo: Take off the ring! Practical, quantum-secure key exchange from LWE," In ACM Conference on Computer and Communications Security (CCS) 2016. doi:10.1145/2976749.2978425, eprint: http://eprint.iacr.org/2016/659.
[17] J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, D. Stehlé, "CRYSTALS – Kyber: a CCA-secure module-lattice-based KEM," In IEEE European Symposium on Security and Privacy, 2018, doi:10.1109/EuroSP.2018.00032
[18] T.Prest, P. A. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Ricosset, G. Seiler, W. Whyte and Z. Zhang, "Falcon," Technical report, National Institute of Standards and Technology, 2017, https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/ round-2-submissions