Journal of Modern Technology and Engineering
Vol.5, No.1, 2020, pp.5-17

Jomard Publishing

JAVA LIBRARY FOR LATTICE-BASED IDENTIFICATION SCHEMES

Azhar Murzaeva, Meryem Soysaldi, Sedat Akleylek*

Department of Computer Engineering, Ondokuz Mayıs University, Samsun, Turkey

Abstract. Traditional public key cryptosystems are based on computationally hard problems that will be solved in polynomial time by a quantum computer. Therefore, attentions of researchers are directed on the development of post-quantum cryptosystems and their implementations are needed. There are several examples of implementing quantum secure cryptosystems such as digital signatures, key encapsulation mechanisms on different programming languages like C, Java, Javascript. In this paper, we aim to develop a cryptographic Java library for identification schemes that are insensibly used in our daily life. We introduce our library in details and demonstrate implementations of the selected lattice-based identification schemes.

Keywords: post-quantum cryptography, lattice-based cryptography, identification schemes, java library.
AMS Subject Classification: 94A60, 68P25.
Corresponding author: Sedat Akleylek, Ondokuz Mayis University, Samsun, Turkey,
e-mail: sedat.akleylek@bil.omu.edu.tr
Received: 25 September 2019; Accepted: 15 January 2020; Published: 30 April 2020.

1 Introduction

Due to the Shor's factorization algorithm for quantum computers (Shor, 1997) and NIST's call for a quantum-resistant algorithms' standardization project (Chen et. al., 2016), attentions of researchers are directed on the developments in post-quantum cryptography. Post-quantum cryptography consists of several research areas such as code-based cryptography, hash-based cryptography, lattice-based cryptography, multivariate cryptography and isogeny.

Each of those areas contains some hard problems that are (and become) the basis for different cryptosystems like encryption/decryption, key exchange/encapsulation, digital signatures. In this paper, we outline identification schemes that are the primitives of digital signatures. They are responsible for determining parties in an interactive system, that is quite important for the transmission of data to the trustworthy party. Modern identification schemes rely on today's hard problems (e.g. integer factorization (Guillou & Quisquater, 1998), elliptic curves (Schnorr, 1991)), whereas the new generation identification schemes are expected to be quantum resistant.

Identification is insensibly used in our daily life. It provides a proper access to the electronically controlled services such as e-mails, e-banking and e-government systems. It is responsible to provide a safe and easy communication between two parties by ensuring the truth of identity. Identification is the main basis of the digital signatures that are widely used in those e-services.

Since the development of quantum computers leads to the construction of new post-quantum cryptographic protocols, new quantum resistant identification schemes must be developed, too. While some cryptographic libraries for post-quantum cryptosystems like digital signatures (Prest et. al., 2017; Guneysu et. al., 2012; Lyubashevsky, 2009; Bansarkhani & Buchmann, 2014), public key encryption and key establishment algorithms (Bos et. al., 2018, 2016) have been developed, there is no practical library for lattice-based identification schemes. Therefore, we develop

a cryptographic Java library that will provide a convenience in implementation of identification scheme.

1.1 Motivation and Contribution

Well-known identification schemes depend on computationally hard problems like integer factorization (Guillou & Quisquater, 1998), (elliptic curve) discrete logarithm problem (Fiat & Shamir, 1987; Schnorr, 1991); however, new generation identification schemes based on quantum secure lattice problems are being developed (Cayrel et. al., 2010; Kawachi et. al., 2008; Silva et. al., 2011; Xagawa & Tanaka, 2009). There are some examples of implementations in literature: implementation of lattice-based signature schemes (Bansarkhani & Buchmann, 2014; Lyubashevsky, 2009; Guneysu et. al., 2012), implementation of code-based zero-knowledge identification schemes (Cayrel et. el., 2013), implementation of post-quantum digital signature algorithm (Prest et. al., 2017), implementation of key exchange algorithm (Bos et. al., 2018), implementation of code-based identification and signature schemes (El Yousfi Alaoui et. al., 2013) and attempt to implement lattice-based schemes on constrained devices such as smart cards (Boorghany & Jalili, 2014).

In this paper, we overview some lattice-based identification schemes and develop a common library for their implementations. For the library Java programming language is selected since there is a big need for mobile applications and Java is the official language of Android applications. Readability and modifications of code on that selected language is easier comparing to low-level programming languages.

During the construction of the library, we outline operations needed for the realization of schemes selected for this study and deduce some observations that help us to make a list of regular functions for it. We prepare a framework for the implementation of different schemes by collecting all those functions in one module. All these steps we describe in details and then, we introduce our developed library with its usage example by implementing several identification schemes.

1.2 Organization

The rest content of this paper is organized as follows. Section II presents identification schemes used for this study. Section III introduces details of developed library and demonstrates implementation of identification schemes. In addition, the comparison of those identification schemes by obtained experimental results is given. Section IV states the conclusion and future work.

2 An Overview of Lattice-based Identification Schemes

In this section, definition of identification scheme is recalled. Structures of Silva et. al. (2011); Cayrel et. al. (2010); Kawachi et. al. (2008) and Xagawa & Tanaka (2009) identification schemes are explained.

An identification scheme consists of key generation and commitments' computation steps. This scheme contains Prover (P) and Verifier (V) parties that interact with each other through these steps. Key Generation step generates public and secret keys (pk, sk), that are used in the commitments' computations. Then, Verifier (V) checks commitments by comparing them with results of his own computations and then, either accepts or rejects the Prover.

In this study Silva et. al. (2011); Cayrel et. al. (2010); Kawachi et. al. (2008) and Xagawa & Tanaka, (2009) identification schemes are explained and their implementations are performed. Silva et. al. (2011) scheme is based on LWE (Learning With Errors) problem, while Cayrel et. al. (2010) scheme is based on SIS (Short Integer Solution) problem in lattices. Kawachi et. al. (2008) is based on SIS problem and Xagawa & Tanaka (2009) is based on NTRU key recovery problem in lattices.

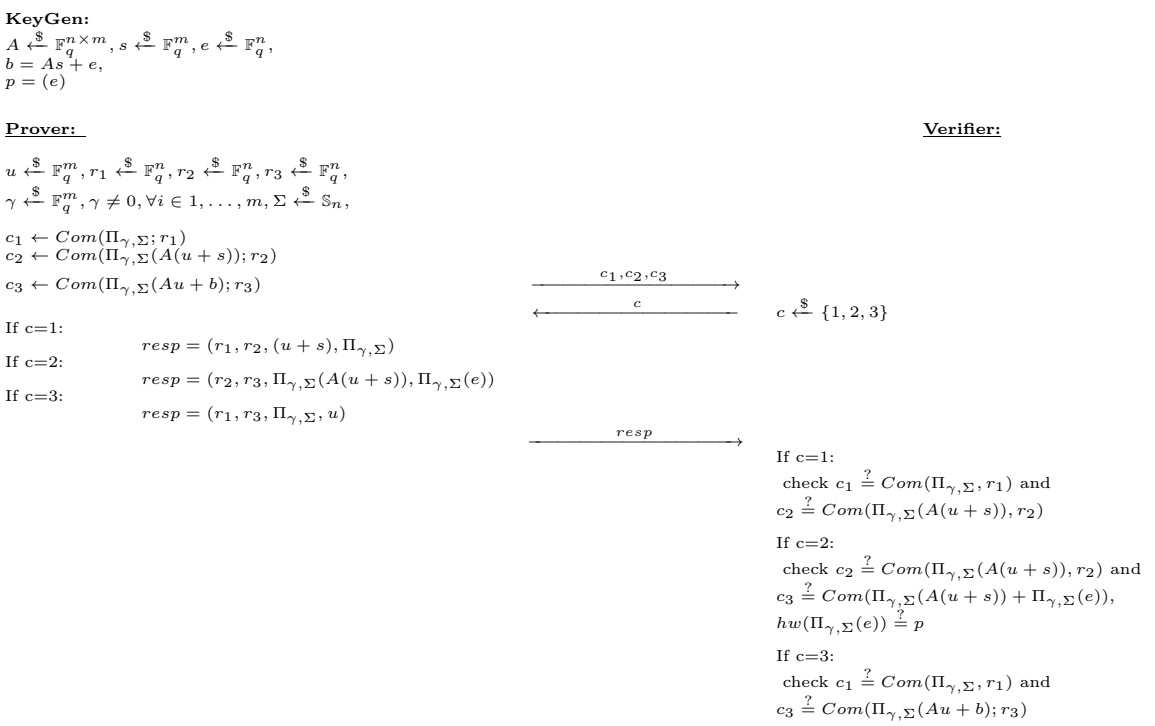


Figure 1: Silva's identification scheme

2.1 Silva's LWE-based Identification scheme

Silva et. al. (2011) proposed a scheme based on LWE (Learning With Errors) problem in lattices. It is a three-pass scheme. Steps of that interactive proof scheme are given in Figure 1. In the Key Generation stage, private and public keys are generated. Later, these keys are used on Prover and Verifier sides for computations. Prover computes and sends commitments to the Verifier. In his turn, Verifier generates a challenge. Depending on this challenge, Prover sends some parameters and Verifier determines the veracity of the statement.

Remark. *For the implementation of this scheme's Key Generation, Computation of commitments and Verification stages, these operations are needed: matrix-vector product and addition of vectors. Implementation of that scheme is explained in details in Section III.*

2.2 CLRS Identification scheme

Cayrel et. al. (2010) proposed a scheme based on SIS (Shortest Integer Solution) problem in lattices. It is composed of five phases as demonstrated in Figure 2. In the Key Generation stage, private and public keys are generated. Later, these keys are used on Prover and Verifier sides for computations. Prover computes and sends commitments to the Verifier. In his turn, Verifier sends α. Using this α, 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 determines the veracity of the statement.

Remark. *For the implementation of this scheme's Key Generation, Computation of commitments and Verification stages, these operations are required: matrix-vector product, a random permutation function, addition of vectors, matrix inversion. Implementation of that scheme is explained in details in Section III.*

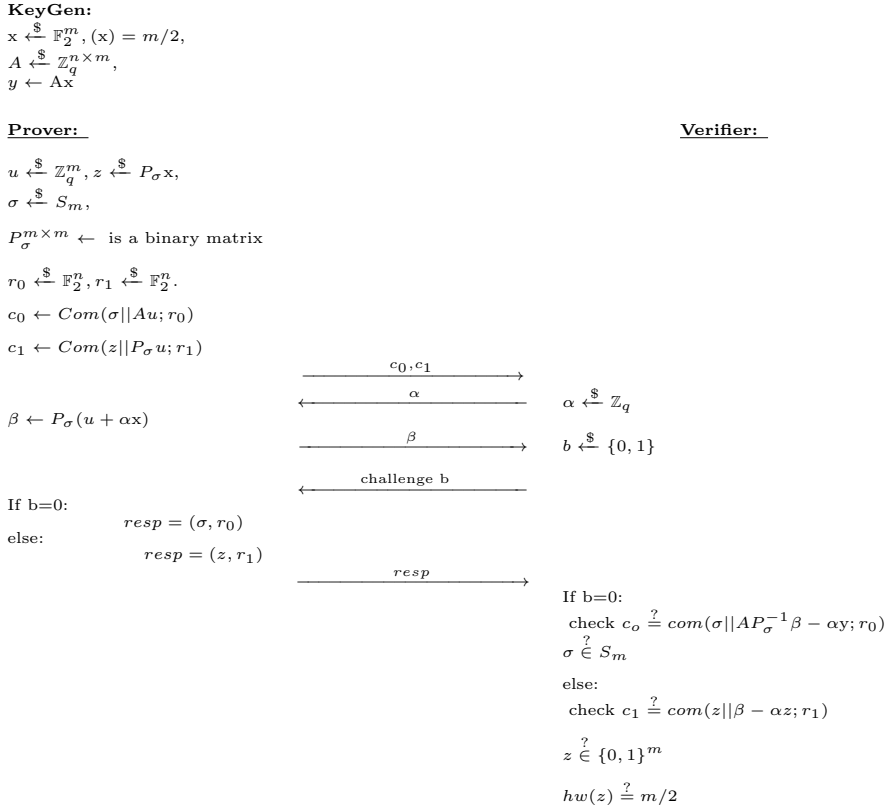


Figure 2: Cayrel's (CLRS) identification scheme

2.3 Kawachi's LWE-based Identification scheme

Kawachi et. al. (2008) proposed a scheme based on SIS (Shortest Integer Solution) problem in lattices. It is a three-pass scheme. Steps of that interactive proof scheme is given in Figure 3. In the Key Generation stage, private and public keys are generated. Later, these keys are used on Prover and Verifier sides for computations. Prover computes and sends commitments to the Verifier. In his turn, Verifier generates a challenge. Depending on this challenge, Prover sends some parameters and Verifier determines the veracity of the statement.

Remark. *For the implementation of this scheme's Key Generation, Computation of commitments and Verification stages, these operations are needed: matrix-vector product and addition of vectors. Implementation of that scheme is explained in details in Section III.*

2.4 Xagawa's NTRU-based Identification scheme

Xagawa & Tanaka (2009) proposed a scheme based on NTRU problem in lattices. It is composed of three phases as demonstrated in Figure 4. In the Key Generation stage, private and public keys are generated. Later, these keys are used on Prover and Verifier sides for computations. Prover computes and sends commitments to the Verifier. In its turn, Verifier generates a challenge. Depending on this challenge, Prover sends some parameters and Verifier determines the veracity of the statement.

Remark. *For the implementation of this scheme's Key Generation, Computation of commitments and Verification stages, these operations are required: vectors product, a random permutation function and addition of vectors. Implementation of that scheme is explained in details in Section III.*

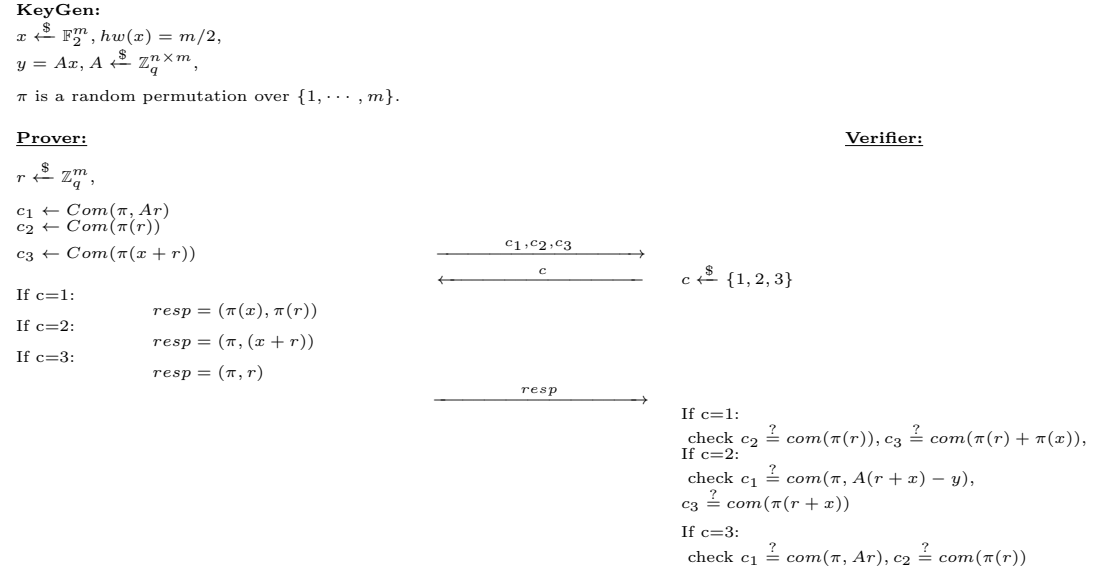
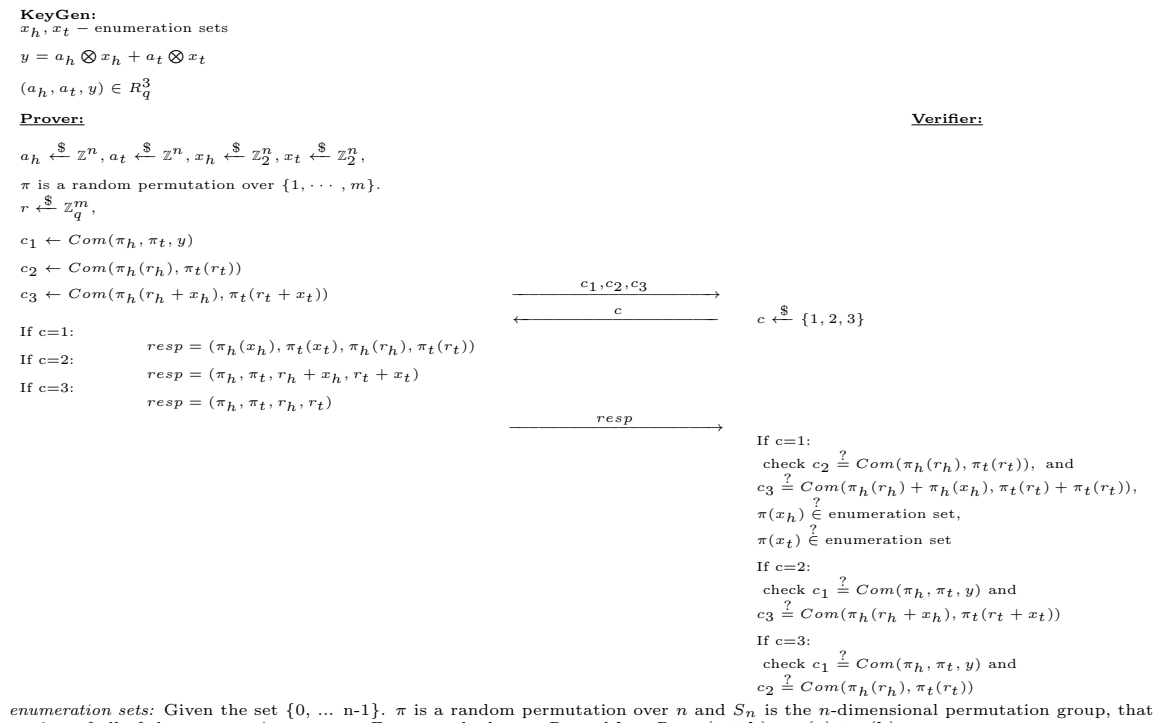


Figure 3: Kawachi's identification scheme



enumeration sets: Given the set {0, ... n-1}. π is a random permutation over n and S_n is the n-dimensional permutation group, that consists of all of the permutations over n. For example, let $\mathbf{a} \in R_q$ and $\mathbf{b} \in R_q$. $\pi(\mathbf{a} + \mathbf{b}) = \pi(\mathbf{a}) + \pi(\mathbf{b})$.

Figure 4: Xagawa's identification scheme

3 Implementation Details and Guideline for the Library

In this section, we present the list of required operations. We introduce the framework and general procedures for the implementation. We give the details of the developed library and as an example of usage, implementations of few identification schemes are demonstrated. The details of the schemes and obtained observations are also described in subsections.

3.1 Framework for the Implementation

Depending on the computational operations needed for the implementation of the concerned scheme, we determine the required functions/modules as in Figure 5. With such blueprints in mind, we collect all required functions in one module. For instance, there is a *knuth_shuffle* function in the module, which performs the permutation of elements in a given vector. Also, in the realization of an identification scheme, as a "commitment function" a one way hash function is used.

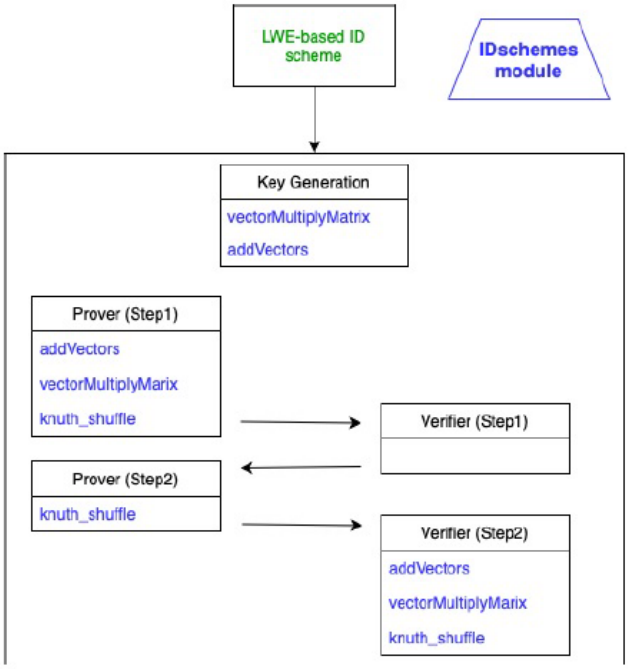


Figure 5: Implementation design of LWE-based scheme

All presented identification schemes need common operations such as matrix-vector product, vector product, addition/subtraction of vectors and a random permutation function. Instead of writing these functions in every step of each scheme's implementation, calling them from one common file provides a convenience and prevents from the code repeat.

3.2 Guideline for the Library

Traditionally, before performing an implementation, a pseudocode of the system or its flowchart is prepared. Such a general pseudocode for an identification scheme is prepared as in Figure 6.

Following the procedures from the pseudocode and adding small modifications depending on the properties of a scheme, it can be organized and implemented using any language in a proper way.

3.2.1 Observations

To construct the library more efficiently, we determine the required arithmetic operations and functions. To achieve this goal, we have the following observations:

- Silva et. al. (2011) and Kawachi et. al. (2008) schemes need a matrix-vector product and vectors addition/subtraction operations.

```
function KeyGeneration:
        generateParam1(); // compute parameters
        generateParam2();
        computeSecretKey(Param1); // compute secret key
        computePublicKey(Param2); // compute public key
function P1 (Prover's Step1):
        generateParam1(); // compute parameters
        generateParam2();
        computeC1(); // compute commitments
        computeC2();
        computeC3();
function V1 (Verifier's Step1):
        generateCH(); // compute challenge
function P2 (Prover's Step2):
        sendRespond(); // send some parameters
function V2 (Verifier's Step2):
        check(); // verify a prover
```

Figure 6: General pseudocode of an identification scheme

- Cayrel et. al. (2010) scheme needs a sparse matrix-vector product and matrix inversion operations.
- Xagawa & Tanaka (2009) scheme needs to compute the product of two short polynomials.

Considering the information gained from these observations, we noticed common computational operations that can be called from one module and can be used in any scheme's corresponding step. Additionally, the number of prover's and verifier's steps (given in the pseudocode) varies with the construction of a scheme. For instance, while Silva's scheme contains three phases, Cayrel's scheme has five phases. However, most of the operations used in computations on each of those steps are similar. Both of those schemes are lattice-based identification schemes that are based on hard problems in lattices. Thus, we collect all these regular operations in one common library.

The list of main regular operations in the library is as follows: scalarMultiplyVector, generateIdentityMatrix, initMatrixRandom, vectorMultiplyMatrix, transpose, hw (computes hamming weight), padding, sha256, knuth _shuffle, matrix _invert, addVectors, subtractVectors, multiplyVectors, randInt, com. These operations are in the *idscheme _lib.jar* file. After adding this file to a project, those functions can be called from the existing module (IDscheme module).

Silva's LWE-based scheme is a 3-pass scheme and for its implementation, needed functions are demonstrated in Figure 7. Inside of these main functions, functions from our built library are called and used. In Figure 8, usage examples of some functions are demonstrated.

```
keyGeneration(n, m, q);
p1();
int ch;
ch = v1();
Object[] response = p2(ch);
v2(ch,response);
```

Figure 7: Main functions for the implementation of Silva et. al. (2011) scheme

In the piece of code given in Figure 8, the Key Generation step is presented. Required initMatrixRandom, vectorMultiplyMatrix, addVectors and hw functions are called from the

```
// b = As + e
int[] y = IDscheme.vectorMultiplyMatrix(s, amatrix);
int[] b = IDscheme.addVectors(y,e);
int p = IDscheme.hw(e);
```

Figure 8: Calling some functions from IDscheme library

IDscheme library. An example for computing commitments c_1, c_2 and c_3 in Silva's scheme by calling a *com* function is given in Figure 9.

```
// com(Pi_gamma,_sigma;r1)
c1 = IDscheme.com(Arrays.toString(perm1)+Arrays.toString(r1));
// com(Pi_gamma,_sigma(A(u+s));r2)
c2 = IDscheme.com(Arrays.toString(perm2)+Arrays.toString(r2));
// com(Pi_gamma,_sigma(Au+b);r3)
c3 = IDscheme.com(Arrays.toString(perm3)+Arrays.toString(r3));
```

Figure 9: Calling a commitment function from IDscheme library

```
keyGeneration(n, m, q);
p1();
double alpha = v1();
double[] beta = p2(alpha);
int b = v2();
Object[] response_p3 = p3(b);
v3(b,response_p3, beta);
```

Figure 10: Main functions for the implementation of Cayrel et. al. (2010) scheme

Main functions for the implementation of Cayrel's scheme are listed in Figure 10. Inside of these main functions for Cayrel's scheme, functions such as transpose, vectorMultiplyMatrix, scalarMultiplyVector, addVectors are called from the developed library. Usage examples are demonstrated in Figure 11.

```
double[] alphax_vector = IDscheme.scalarMultiplyVector(alpha,x_vector);
u_alphax = IDscheme.addVectors(u_vector,alphax_vector); // u + alpha.x
double[][] p_transpose = IDscheme.transpose(p_sigma_matrix);
betha = IDscheme.vectorMultiplyMatrix(u_alphax, p_transpose);
```

Figure 11: Calling functions from IDscheme library for Cayrel's challenge computation step (computing β)

Cayrel's scheme consists of five phases as illustrated in Figure 2. Differently from the previous LWE-based scheme, this scheme needs functions called in Figure 12. In that piece of code, the verification step is presented. These matrixInvert, transpose, multiply, vectorMultiplyMatrix, scalarMultiplyVector and subtractVectors functions are called from the IDscheme module.

Similar to the Silva's and Cayrel's identification schemes, Kawachi's scheme and Xagawa's schemes use the common vectorMultiplyMatrix, addVectors, subtractVectors and other functions. Differently from these functions, in Xagawa's scheme the multiplyVectors function is needed. Those functions are called from the IDscheme module.

```
double[][] p_matrix = new double[glob_m][glob_m];
p_matrix = IDscheme.inverse(pmatrix);

double[][]  ap_sigma = IDscheme.multiply(amatrix,p_matrix); //A.P_sigma
double[][]  ap_transpose = IDscheme.transpose(ap_sigma);

String sigma_str = Integer.toString(sigma);
double[] op1 = IDscheme.vectorMultiplyMatrix(betha, ap_transpose);

double[]  op2 = IDscheme.scalarMultiplyVector(alpha,y);
double[]  op3 = IDscheme.subtractVectors(op1,op2);
comp1 = IDscheme.com(sigma_str+Arrays.toString(op3)+Arrays.toString(r0));
```

Figure 12: Calling functions from IDscheme library for Cayrel's verification step

3.3 Experimental Results

Parameter set values for implementation of Cayrel's, Silva's, Kawachi's and Xagawa's identification schemes are given in Table 1. Parameter sets for the Cayrel's scheme are recommended in Cayrel et. al. (2010). Kawachi et. al. (2008) did not provide parameter values, so the values listed in Table 1 are extracted from Cayrel et. al. (2010). Parameter set values

Table 1: System parameters for identification schemes

Parameter	Cayrel's Scheme	Silva's Scheme	Kawachi's Scheme	Xagawa's Scheme
n	512	608	512	677
m	2048	960	2048	-
q	257	1024	257	2048
Commitment length	256-bits	256-bits	256-bits	256-bits

for the Xagawa & Tanaka (2009) scheme are proposed in Hirschhorn et. al. (2009). For the Kawachi et. al. (2008) scheme, parameter values were not given in their original work, thus, parameters suggested for the LWE in Cheon et. al. (2016) are used. These parameter values provide a 80-bit security level for the Cayrel's, Kawachi's, Xagawa's schemes and 128-bit security level for the Silva's identification scheme. Implementation of those schemes is performed on macOS environment (Mojavi version 10.14). The experimental results of Silva's execution is shown in Figure 13. There, results of commitments (on the Prover's step) and results of computed commitments (on the Verifier's step) are printed on the screen to show that scheme works properly as desired.

```
LWE-based ID Scheme:
Parameters Set:
n = 608
m = 960
q = 1024
Prover: compute commitments c1, c2 and c3:
c1:c5381a079903674a0081c36b6a591435d108fcec7d5495d946f7b4f9b7bd4526
c2:103bfeb70176442531c71d36d05a976a65d542e1d730aa709a143a6b231bf005
c3:a4160e3a64ec71e0c2a85447bc9ed23b8a658741a5c9625669ecfdce9f5c0ce2
Verifier: sends a challenge: 1
Prover: reveals some parameters depending on challenge.(Current challenge is: 1)
Verifier: checks commitment correctness from information revealed by prover and accept in case of success.
computed c1:c5381a079903674a0081c36b6a591435d108fcec7d5495d946f7b4f9b7bd4526
computed c3:a4160e3a64ec71e0c2a85447bc9ed23b8a658741a5c9625669ecfdce9f5c0ce2
Success!
```

Figure 13: Output of Silva's scheme

On the current execution (in Figure 13), as a challenge 1 is generated. Depending on this challenge value, c_1 and c_3 commitments are computed and compared with c_1, c_3 commitments, that were computed on Prover's side.

Similarly, output of Cayrel's execution is shown in Figure 14. Results of commitments (on the Prover's step) and results of computed commitments (on the Verifier's step) are printed on the screen, too. This shows that scheme works properly as desired.

```
CLRS ID Scheme:
Parameters Set:
n = 512
m = 2048
q = 257
Prover: compute commitments c1, c2 and c3:
c1:cf4756049d1557b4ab2e3ecf8d643e784c617fe90f50aa9d196d16a8618032da
c2:b657ec5a3429093771480164ca57b9a7e8d8fedab4c49bb90574f5e206514d55
Verifier: sends a challenge: 1
Prover: reveals some parameters depending on challenge.(Current challenge is: 1)
Verifier: checks commitment correctness from information revealed by prover and accept in case of success.
computed c1:cf4756049d1557b4ab2e3ecf8d643e784c617fe90f50aa9d196d16a8618032da
com1:cf4756049d1557b4ab2e3ecf8d643e784c617fe90f50aa9d196d16a8618032da
Success!
```

Figure 14: Output of Cayrel's scheme

On the execution of Cayrel's scheme (in Figure 14), as a challenge 1 is generated. Depending on this challenge value, c_1 commitments is computed by Verifier and compared with c_1 commitment, that was computed by Prover.

```
Kawachi, Xagawa and Tanaka's ID Scheme:
Parameters Set:
n = 512
m = 2048
q = 257
Prover: compute commitments c1, c2 and c3:
c1:c1b778b97de26793b3b1479bbd554faa2313c0d8fa126604110d56940af0e400
c2:203b49374603943cf94ddff9436dfa5b530ef666ee9d2051a79443a771bb932
c3:acde3958af5c08242cdad4b89fa1b35649e4ee3beb56ad8f35d075dd8809d60e
Verifier: sends a challenge: 2
Prover: reveals some parameters depending on challenge.(Current challenge is: 2)
Verifier: checks commitment correctness from information revealed by prover and accept in case of success.
computed c1:c1b778b97de26793b3b1479bbd554faa2313c0d8fa126604110d56940af0e400
computed c3:acde3958af5c08242cdad4b89fa1b35649e4ee3beb56ad8f35d075dd8809d60e
Success!
```

Figure 15: Output of Kawachi's scheme

In addition to these two schemes, Kawachi et. al. (2008) and Xagawa & Tanaka (2009) 3-pass schemes are implemented. Kawachi's scheme is based on SIS (Short Integer Solution) problem, likewise Cayrel's scheme. Output of Kawachi's scheme is given in Figure 15. Depending on the generated challenge value 2, comparison of commitments c_1 and c_3 is performed.

The execution output of Xagawa's scheme, that is based on NTRU problem, is given in Figure 16. Generated value of challenge is 3 and depending on this, c_1 and c_2 commitments are computed and compared.

```
Test Xagawa and Tanaka's ID Scheme (NTRU-based):
Parameters Set:
n = 677
q = 2048
Prover: compute commitments c1, c2 and c3:
c1:f2f9686304ce93301768aa69550efbc636f7f1452536b29047a3c020dd0f7972
c2:5b11fe88e07ed8d37dd690ea644df81d6ca462080c49647685fa1987b562f564
c3:f37d88ac2ce64449bbb5b921a78a2dfe67df8c8046de36fbb68dfdbc2c0e1f97
Verifier: sends a challenge: 3
Prover: reveals some parameters depending on challenge.(Current challenge is: 3)
Verifier: checks commitment correctness from information revealed by prover and accept in case of success.
computed c1:f2f9686304ce93301768aa69550efbc636f7f1452536b29047a3c020dd0f7972
computed c2:5b11fe88e07ed8d37dd690ea644df81d6ca462080c49647685fa1987b562f564
Success!
```

Figure 16: Output of Xagawa's scheme

The performance results of Silva et. al. (2011), Cayrel et. al. (2010), Kawachi et. al. (2008) and Xagawa & Tanaka (2009) schemes' execution are demonstrated in Table 2. As expected, performance results vary depending on related hard problems of schemes. Cayrel et. al. (2010) scheme that depends on SIS problem, where computational operations such as matrix-matrix product, matrix-vector product and matrix inversion are needed, requires the most amount of time for the implementation. Those used operations consist of nested loops (one loop is used

Table 2: Implementation results of identification schemes

Time (ms)	Cayrel's Scheme	Silva's Scheme	Kawachi's Scheme	Xagawa's Scheme
Key Generation	76.185	33.114	47.013	3.508
Computation of Commitments	437.677	38.303	49.09	25.279
Verification	80,319.459	5.023	4.158	1.37
Total	80,833.321	76.44	100.261	30.157

inside another loop) and use multiplication/addition/subtraction operations. For instance, in matrix-matrix product operation one nested loop is needed, while in matrix inversion operation number of nested loops is 3 (one of those loops contains 2 loops inside). Therefore, for matrix A and B with dimensions $n \times m$ and $m \times p$, m multiplications and m additions are required. Besides, there are np elements to compute, thus, matrix-matrix product operations takes $O(nmp)$ time for computations. For the implementation of Xagawa & Tanaka (2009) scheme, computationally easier operations such as vectors product, a random permutation function and addition of vectors are used. Consequently, it takes the least time. Despite the Silva et. al. (2011) and Kawachi et. al. (2008) schemes are based on different problems, operations required for their implementation are similar. But the number of those used operation for the implementation of Silva et. al. (2011) is less comparing to the number of used operations for the Kawachi et. al. (2008) scheme. Therefore, there is a small difference between their results.

Presented library and the implementation of identification schemes demonstrated in this study are available at https://github.com/msAzhar/pqc-id_schemes/.

4 Conclusion

Many quantum secure cryptosystems are being developed. With the importance of their realization, in this study, we are concerned to develop a Java library for lattice-based identification schemes. We review the main structures of Silva et. al. (2011), Cayrel et. al. (2010), Kawachi et. al. (2008) and Xagawa & Tanaka (2009) identification schemes. We figure out the useful functions and computational operations needed for their implementations. The obtained list of those regular functions is applied to the library. Therefore, if one wants to implement his/her developed identification scheme to test its performance, he/she can make it easily by using our library regardless of the complexity of the structure of the functions. We introduce our developed library and demonstrate implementations of identification schemes summarized during this work. Implementation of the selected identification schemes is performed on macOS environment (Mojavi version 10.14). The obtained results of implementation show that Cayrel et. al. (2010) identification scheme takes the most amount of time, while the Xagawa & Tanaka (2009) scheme' execution takes the least time. The reason is that, Cayrel et. al. (2010) identification scheme uses matrix-matrix product, matrix-vector product and matrix inversion operations which require more time for computations comparing to the remaining identification schemes. As a future work, more efficient algorithms and better approaches (e.g. argument specific algorithms) for the existing functions can be investigated, found and later, applied to the developed library.

5 Acknowledgement

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

References

Bansarkhani, R.E., Buchmann, J.A. (2014). Improvement and Efficient Implementation of a Lattice-Based Signature Scheme, *Selected Areas in Cryptography 2014*, 48-67. doi: 10.1007/978-3-662-43414-7

Boorghany, A., Jalili, R. (2014). Implementation and Comparison of Lattice-based Identification Protocols on Smart Cards and Microcontrollers, *Cryptology ePrint Archive, Report 2014/078*, https://eprint.iacr.org/2014/078

Bos, J., Costello, C., Ducas, L., Mironov, I., Naehrig, M., Nikolaenko, V., Raghunathan, A., Stebila, D. (2016). Frodo: Take off the ring! Practical, quantum-secure key exchange from LWE, *ACM Conference on Computer and Communications Security (CCS)*, doi:10.1145/2976749.2978425, eprint: http://eprint.iacr.org/2016/659.

Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Seiler, G., Stehlé D. (2018). CRYSTALS – Kyber: a CCA-secure module-lattice-based KEM, *IEEE European Symposium on Security and Privacy*, doi:10.1109/EuroSP.2018.00032

Cayrel, P.L., Lindner, R., Ruckert, M., & Silva R. (2010). Improved Zero-knowledge identification with lattices, *J. Tatra Mountains Mathematical Publications*, *53*, 1-17.

Cayrel, P.L., Yousfi Alaoui, S.M.E., Gunther, F., Hoffmann, G., & Rother, H. (2013). Efficient implementation of code-based identification schemes, *Security Engineering and Intelligence Informatics. CD-ARES 2013. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg*, *8128*, 122-136.

Chen, L., Jordan, S., Liu, Y.K. et al. (2016). Report on post-quantum cryptography, *National Institute of Standards and Technology.*

Cheon, J.H., Kim, D., Lee, J., & Song Y. (2016). Lizard: Cut off the Tail! Practical Post-Quantum Public-Key Encryption from LWE and LWR. *Cryptology ePrint Archive Report 2016/1126.*

El Yousfi Alaoui, S.M., Cayrel, P.L., El Bansarkhani, R., Hoffmann, G. (2013). Code-Based Identification and Signature Schemes in Software. *Security Engineering and Intelligence Informatics. CD-ARES 2013. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg 8128.*

Fiat, A., Shamir, A. (1986, August). How to prove yourself: Practical solutions to identification and signature problems. In *Conference on the Theory and Application of Cryptographic Techniques* (pp. 186-194). Springer, Berlin, Heidelberg.

Guillou, L., Quisquater, J.J. (1988). A "paradoxical" identity-based signature scheme resulting from zero-knowledge. In *Conference on the Theory and Application of Cryptography* (pp. 216-231). Springer, New York, NY.

Güneysu, T., Lyubashevsky, V., & Pöppelmann, T. (2012, September). Practical lattice-based cryptography: A signature scheme for embedded systems. In *International Workshop on Cryptographic Hardware and Embedded Systems* (pp. 530-547). Springer, Berlin, Heidelberg.

Hirschhorn, P.S., Hoffstein, J., Howgrave-Graham, N., & Whyte, W. (2009, June). Choosing NTRUEncrypt parameters in light of combined lattice reduction and MITM approaches. In *International Conference on Applied Cryptography and Network Security* (pp. 437-455). Springer, Berlin, Heidelberg.

Kawachi, A., Tanaka, K., & Xagawa, K. (2008). Concurrently Secure Identification Schemes Based on the Worst-Case Hardness of Lattice Problems, *J. Advances in Cryptology - ASIACRYPT 2008*, 372–389.

Lyubashevsky, V. (2009). Fiat-Shamir With Aborts: Applications to lattice and factoring-based signatures, *Advances in Cryptology –ASIACRYPT 2009, Lecture Notes in Computer Science; Springer Berlin Heidelberg*, 598-616.

Menezes, J., Oorschot, P.C., Vanstone S.A. (1996). *Handbook of Applied Cryptography*, http://www.cacr.math.uwaterloo.ca/hac/

Hoffstein J., Pipher J., & Silverman J.H. (1998). NTRU: A ring-based public key cryptosystem. *Algorithmic Number Theory; Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 1423.*

Prest T., Fouque P.A., Hoffstein J., Kirchner P., Lyubashevsky V., Pornin T., Ricosset T., Seiler G., Whyte W. & Zhang Z. (2017). Falcon. *Technical report, National Institute of Standards and Technology*, https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/ round-2-submissions

Schnorr, C.P. (1991). Efficient signature generation by smart cards. *Journal of Cryptology, 4(3)*,161-174.

Shor, P.W. (1997). Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, *SIAM J Comput.*, *26*(5), 1484-1509.

Silva, R., Campello, A. & Dahab R. (2011). LWE-based identification schemes,*CoRR*, abs/1109.0631.

Xagawa K., Tanaka, K. (2009). Zero-Knowledge Protocols for NTRU: Application to Identification and Proof of Plaintext Knowledge, *Lecture Notes in Computer Science*, 198-213.