ROYAUME DU MAROC
Université Mohammed V - RABAT -
جامعة محمد الخامس – الرباط –
Faculté des sciences
كلية العلوم
CENTRE D'ETUDES DOCTORALES - SCIENCES ET TECHNOLOGIES

N°d'ordre: 3479

THESE

En vue de l'obtention du : ***DOCTORAT***

Centre de recherche *: CEREMAR-FS*
Structure de Recherche *: Laboratoire Mathématiques, Statistique et Applications*
Discipline *: Mathématiques appliquées*
Spécialité *: Cryptographie et sécurité de l'information*

Présentée et soutenue le :24/06/2021 par :

Hamid HAJAJE

Security, Privacy, and Scalability of Path Computation Elements

JURY

Abdelhak ZOGLAT	*PES,Faculté des Sciences, Université Mohammed V-Rabat*	*Président*
Abdelali ZINE EL ABIDINE	*PES, Faculté des Sciences, Université Mohammed V-Rabat*	*Rapporteur / Examinateur*
Charaf BENSOUDA	*PES,Faculté des Sciences, Université Ibn Tofail-Kénitra*	*Rapporteur / Examinateur*
Driss BENNIS	*PH, Faculté des Sciences, Université Mohammed V-Rabat*	*Rapporteur / Examinateur*
Zine El Abidine GUENNOUN	*PES,Faculté des Sciences, Université Mohammed V-Rabat*	*Directeur de thèse*

Année universitaire : 2020 - 2021

✉ Faculté des Sciences, avenue Ibn Battouta, BP. 1014 RP, Rabat –Maroc
☎ 00212(0) 37 77 18 76 🖷 00212(0) 37 77 42 61; http://www.fsr.ac.ma

Remerciements

Les travaux présentés ont été réalisés au Laboratoire Mathématiques, Statistique et Applications (LMSA), Université Mohammed V, Faculté des Sciences de Rabat. Ces travaux ont été s'inscrit dans le cadre des thèmes de recherches de l'CEREMAR-FS « CENTRE DE RECHERCHES MATHEMATIQUES ET APPLICATIONS DE RABAT » sous la direction de Monsieur Zine El Abidine GUENNOUN, Professeur et chef de département de Mathématiques à la Faculté des Sciences de Rabat.

Je tiens à remercier sincèrement Monsieur Zine El Abidine GUENNOUN Professeur à la Faculté des Sciences de Rabat et Directeur de mes travaux de recherches pour son encadrement exemplaire, son assurance, ses aides précieuses, sa bienveillante attention et sa disponibilité permanente, et surtout pour ses amples conseils qu'elle n'a jamais cessé de me prodiguer.

J'exprime ma profonde reconnaissance au Professeur Mr. Abdelhak ZOGLAT, Professeur à la Faculté des Sciences de Rabat d'avoir accepté de juger ce travail et de présider le jury de ma thèse.

J'adresse un remerciement particulier à Monsieur Abdelali ZINE EL ABIDINE, Professeur à la Faculté des Sciences de Rabat, pour avoir accepté d'examiner ce travail et d'enrichir nos résultats par ses remarques et commentaires.

Je remercie vivement Monsieur Charaf BENSOUDA, Professeur à la Faculté des Sciences Ibn Tofail de Kénitra, pour l'intérêt qu'il a porté à mon travail, en acceptant de le juger.

Je tiens tout particulièrement à remercier Monsieur Driss BENNIS, Professeur à la Faculté des Sciences de Rabat, pour avoir examiné ce mémoire. Son expérience accumulée durant son parcours universitaire nous a été profitable.

Mes remerciements vont à Monsieur Mouhcine GUENNOUN, Ingénieur logiciel senior chez Cisco Systèmes, Ottawa, Canada, pour son soutien morale et technique tout au long de la préparation de cette thèse. Merci Docteur Mouhcine !

Remerciements particuliers aux professeurs et étudiants travaillant au Laboratoire de mathématiques, statistiques et applications et à tous les étudiants de la Faculté des Sciences de Rabat pour leurs encouragements, leur aide et leur professionnalisme.

Enfin, je souhaite exprimer ma reconnaissance et mes remerciements à toutes les personnes qui m'ont soutenue et aidée dans l'aboutissement de ce travail.

Abstract

The thesis tackles security, privacy, and scalability issues in Path Computation Elements (PCE). We developed an improved version of the NTRU cryptosystem that is more efficient and robust, this system is much better than NTRU and Matru in both measures, speed, and security. We also proposed a novel method to execute path requests in an oblivious fashion, either with constraints or not. Last, we devise a scalable PCE that can handle large number of requests in a dense network, up to 200,000 nodes.

Keywords: Privacy, Security, Scalability, PCE

Résumé

Ce travail de recherche aborde les failles de sécurité, confidentialité, et d'évolutivité des systèmes de calcul de chemin (PCE). Nous avons développé un nouveau crypto système PMTRU qui est une version améliorée de NTRU, ce système est bien meilleur que NTRU et de Matru dans les deux mesures, la vitesse et la sécurité. Nous avons aussi conçu une méthode originale qui préserve la confidentialité de la requête de calcul de chemin, soit avec des contraintes ou pas. Enfin, on a proposé une architecture évolutive qui permet de traiter un nombre élevé de requêtes dans un réseau dense allons jusqu'à 200 000 nœuds.

Mots-clés: Privacy, Security, Scalability, PCE

Résumé de Thèse

La protection de l'information est l'un des sujets les plus importants de notre époque. De nombreux chercheurs sont engagés soit dans la recherche d'un nouveau système cryptographique, soit dans la démonstration du défaut d'un système existant, soit dans l'utilisation de celui-ci comme outil de protection dans l'un des nombreux domaines, en particulier l'informatique.

Nous avons travaillé dans les deux sens, nous avons créé l'un des nouveaux crypto systèmes appelé PMTRU, c'est une variante du crypto système à clé publique NTRU. Le cryptage et le décryptage sont plus rapides en PMTRU qu'en NTRU ou dans sa variante MaTRU, pour plus de précision, le crypto système PMTRU combine les forces des crypto systèmes NTRU et MaTRU. Pour faire les calculs, on remplace l'anneau commutatif par l'anneau matriciel non commutatif qui est utilisé dans le crypto système MaTRU, mais, en chiffrement ou déchiffrement des messages, on utilise les mêmes procédures NTRU avec une permutation sécurisée, ces changements nous permettent d'étendre le la sécurité contre une attaque par force brute pour avoir donné une clé privée et contre une attaque basée sur un réseau, ainsi que la réduction de l'espace mémoire pour sauvegarder notre clé publique et notre cryptogramme par rapport à ceux utilisés par MaTRU et NTRU.

D'autre part, nous avons utilisé un des types de crypto systèmes dits homomorphes pour protéger la requête entre un Path Computation Client (PCC) et un Path Computation Element (PCE). La demande du chemin le plus court entre deux nœuds vers un PCE est faite en protégeant les informations sensibles portées par la demande ainsi que la topologie globale du réseau. Nous décrivons un nouveau protocole pour calculer le chemin le plus court, entre deux nœuds, une source et une destination, soumis à une contrainte, tout en préservant la

confidentialité du client et du serveur. Dans notre protocole, en mettant en œuvre un crypto système homomorphe sécurisé, le PCE peut calculer aveuglément le chemin tout en ignorant le contenu des requêtes cryptées sans avoir besoin de les décrypter. La réponse PCE est un chemin chiffré qui ne peut être déchiffré que par sa clé secrète d'origine. Dans notre implémentation, l'utilisation du schéma homomorphe sur les entiers de Van Dijk, Gentry, Halevi et Vaikuntanathan (DGHV) montre des résultats prometteurs que nous analysons en détail.

Enfin, le PCE actuel est limité à une mise à l'échelle de centaines à quelques milliers de nœuds, en fonction de l'algorithme de calcul de chemin sous-jacent. En effet, l'ensemble du réseau est géré comme un seul domaine lors de l'application de l'algorithme de calcul de chemin. Il faut environ trois secondes pour calculer tous les chemins entre toutes les paires de sommets dans un graphe 1k, mesuré avec les algorithmes Floyd-Warshall et Dijkstra, ce qui nous permet d'appliquer les avantages de la distribution et du pré-calcul pour optimiser l'élément de calcul du chemin. L'objectif de cette proposition est d'augmenter l'échelle PCE de centaines à centaines de milliers de nœuds. En ce sens, nous proposons une architecture à deux couches avec une capacité prouvée, basée sur des simulations, à développer un réseau relativement important.

Table of Contents

List of Tables

List of Algorithms

List of Figures

Introduction

Background

Almost all of our secure communications are created by the concept of public key cryptography over insecure channels. The idea of public key cryptography was first presented by Diffie and Hellman in 1976 [Das08]. Since the launch day, many public key cryptography systems have been proposed so that their underlying security is founded on solving certain mathematical problems in number theory and group theory problems. such as: the problem of factorization of whole numbers (**IFP**) that was used for the first time in the system of Rivest, Shamir and Adelman (RSA) [Riv78a], the problem of the discrete logarithm (**DLP**) in a certain group finite [Kob04], the theory of algebraic coding [Mce78], the intractability of the elliptic curve and the variants of the Matsumoto-Imai cryptosystems [Mat88]. From a practical point of view, these algorithms cost a very high price in terms of space and computational complexity [Rot05][Tal06]. If these algorithms are displaced by symmetric key cryptosystems, this gives a memory with low processing time, for example, RSA is slower than the standard of data encryption (DES) approximately by 1000 times [Bri89] [Sin13].

With the development of information technology, security has become a crucial problem. The cryptography founded to solve this problem, it is used to link the requirements of data security and network communications, namely confidentiality, integrity, authentication and non-repudiation [Rob82]. The design of high-performance algorithms is highly demanded for success in achieving a new safe and high-performance protection mechanism and is of high and continuous demand. In 1996, the time for a new generation of public key cryptosystems to

arrive, the three mathematicians J. Hoffstein, J. H. Silverman and J. Pipher proposed the public key cryptosystem of the number theory research unit NTRU [Hof98] What is a cryptosystem based on a limited computation of constrained polynomials on polynomial rings whose complexity is of quadratic order $O(N^2)$, this network has studied well with Ajtai and Dwork and GGH systems. The difficult problem underlying this cryptosystem is linked to the search for the shortest, and closest, vector in certain networks linked to the ring $\mathbb{Z}[X]/X^N - I$, which have been illustrated as NP-difficult problems. This very attractive feature includes the NTRU in the industry standard IEEE P1363 for cryptography [IEEE]. It is also often considered the most viable post-quantum public key encryption because of its predicted resistance to attack by quantum computers (see, for example, [Per09]), while conventional systems have proven to be [Sho99] somewhat sure in the presence of quantum computing. NTRU has recently been rated as the fastest public key cryptosystem [Hof98]. Its strengths are the short size of the keys and the speed of encryption and decryption. With these features, NTRU has an advantage over other systems that rely on an **IFP** or **DLP**. These advantages make the NTRU a good choice for many applications. Unfortunately, the NTRU cryptosystem is under attack and the most dangerous attacks are "meet-in-the-middle attack" [Hof98] and "Network attack" [Cop97][May01][Ngu01][Han05] on the public key and the encrypted text uses the LLL algorithm. [Len82]. But an appropriate choice of the size of the parameters makes it possible to avoid these attacks. A correct choice of padding for the NTRU encryption called NEAP was proposed in [How04] with proof of security in the presence of a description failure. In 2005, an algorithm was designed by Howgrave et al. [How05], for a good choice of the size of the parameters compared to the security parameters. Since description failures are missing for NTRU. Dwork et al. have developed in 2007 a method to "Immunizing Encryption Schemes from Decryptions errors" [Dwo04].

NTRU is recommended for applications in the embedded system, portable devices, limited resource devices, mobile phone, smart card [Ahm15] [Che16] [Hu09] [Man14] [Mon08] [She09] [Zha13].

It offers both encryption (NTRU-encrypt) and digital signature (NTRUSign), and just after the adaptation of the NTRU, numerous attempts to generalize its algebraic structure appeared, and several researchers have improved the performance of the NTRU by developing its algebraic structure.

In 2002, William D. Banks and Igor E. Shparlinski get at world [Gab02] by a new variant of NTRU with non-invertible polynomials on the same ring as NTRU. This generalization is more secure against some of the known attacks against conventional NTRUs such as network attacks. But this NTRU is less efficient than the classic NTRU scheme because the length of the public key and the encrypted text are roughly doubled.

In the same year, Gaborit et al. [Gab02] proposed a variant of NTRU called CTRU by replacing the base ring of NTRU with a polynomial ring on a binary field $F_2[x]$. CTRU has removed attacks based on LLL and attacks based on CRT. The complexities of encryption and decryption are the same in both encryption systems, NTRU and CTRU.

In 2005, Kouzmenko [Kou06] showed that CTRU is weak under a time attack and to introduce the GNTRU cryptosystem based on Gaussian integers.

In the same year, Coglianese et al. [Cog05] introduced an analogue of the NTRU cryptosystem called MaTRU. MaTRU is based on a ring of square matrices with polynomial entries. This cryptosystem uses a more efficient linear transformation and offers a level of security comparable to that of NTRU. One of its strongest features of NTRU is speed. It does not offer any additional security against the network or other attacks compared to NTRU.

In 2008, Nitin Vats [Vat08] presented a polynomial time algorithm which breaks CTRU for all recommended parameters, they showed that CTRU does not improve speed compared to NTRU and that it is completely insecure against a linear algebraic attack.

A few months later, Nayak et al. [Nay08] described a matrix formulation of the NTRU cryptosystem and showed that a matrix form of NTRU is much faster than the polynomial form of NTRU.

In 2009, Malekian et al. Introduced the QTRU cryptosystem founded on non-commutative algebra [Mal09], namely quaternary algebra to avoid the network attack discussed by Coppersmith and Shamir [Cop97] in 1997. In QTRU many network reduction algorithms do not work. This makes it much more resistant compared to the NTRU against network attacks.

Afterward, Nitin Vats [Vat09], introduced a scheme, called NNRU, to avoid network attacks proposed by Coppersmith and Shamir [Cop97]. It uses non-commutative operations on the non-commutative ring of $k \times k$ square matrices $M = M_k(\mathbb{Z})[X]/X^N - I_{k\times k}$ where M is a matrix ring of the k × k matrices of polynomials in $\mathbb{Z}[X]/X^N - I$. NNRU is completely secure against network attacks with a significant speed improvement $O(k^{1.624})$ compared to NTRU.

In 2010, Malekian et al. describe the OTRU cryptosystem based on the Octonion algebra [Mal10] which is associative, not commutative. The security of the OTRU cryptosystem is based on the difficulty of SVP in a non-circular modular network.

In 2011, N. Zhao and S. Su [Zha11] improved the algorithm for finding the inverse of the polynomial in NTRU. They also designed a new algorithm to judge whether the polynomial is invertible or not by calculation.

In 2012, Y. Bin Pan and Y. Deng in [Pan12] focused on the technique of hiding the trap door of the NTRU cryptosystem. They therefore presented a general NTRU-type framework. This

framework has built a new network-based public key cryptosystem to find certain specific types of closest vector problems (**CVP**).

In 2013, Jarvis et al. [Jar13] proposed a new variant of NTRU the ETRU cryptosystem based on the ring of integers of Eisenstein. ETRU has a denser network structure than NTRU, i.e. stronger network security than NTRU, requires less storage and is comparable in speed. On the other hand, ETRU has weaker key security. ETRU is faster and has smaller keys for the same or higher level of security than NTRU.

In 2014, A. Nitaj [Nit14] performed an NTRU cryptanalysis with two public keys and compared with a network attack given by Coppersmith and Shamir.

In the same year, P. Gauravaram, H. Narumanchi and N. Emmadi [Gau14] presented an analytical study on the implementation of the NTRU encryption scheme which serves as a guide for security practitioners who are new to network cryptographic implementations.

A few months later, D. Cabarcas, P. Weiden and J. Buchmann in [Cab14] focused on the relationship between two anchoring ideals in geometric space and the shortest vector problem in the main ideal network.

In 2015, S. C. Batson in [Bat15] focused on the relationship between two integration ideals in geometric space and the shortest vector problem in the main ideal network.

In the same year, Alsaidi et al. [Als15] introduced the CQTRU cryptosystem based on the commutative quaternion algebra.

In 2016, Thakur and Tripathi [Tha16] released BTRU, a new NTRU-type cryptosystem which replaces Z with a polynomial ring with a variable on a rational field.

In 2016, Yassein and Alsaidi [Yas16a] presented an analogue of the NTRU cryptosystem called HXDTRU, where operations occur in the specially designed large algebra, called hexadecional algebra.

The same authors presented in 2016 the BITRU encryption [Yas16b] which is a binary version of the public key cryptosystem NTRU via binary algebra.

In May 2016, Daniel Bernstein, Chitchanok Chuengsatiansup, Tanja Lange and Christine van Vredendaal released NTRU Prime, [Ber16] which adds defenses against potential attacks to NTRU by eliminating the algebraic structure they considered disturbing. However, after more than 20 years of examination, no concrete approach to attack the original NTRU exploiting its algebraic structure has been found so far.

In 2017, Sonika Singh and Sahadeo Padhye generalized the idea of A. Nitaj and present NTRU cryptanalysis with n public keys.

In 2018, Reza Ebrahimi Atani, Shahabaddin Ebrahimi Atani and Amir Hassani Karbasi [Ata18] presented A non-commutative and secure variant of the CTRU cryptosystem, they gave the name NETRU, This system works on the non-commutative ring $M = M_k(\mathbb{Z}_p)[T,X]/X^N - I_{k\times k}$, where M is a matrix ring of k by k matrices of polynomials in $R = \mathbb{Z}_p[T,X]/X^N - 1$ where p is a prime number.

In 2018, Mamadou Ghouraissiou Camara, Demba Sow and Djiby Sow [Cam18] presented DTRU encryption, which is to use the ring with zero dividers $D = \mathbb{Z} + \mathcal{E}\mathbb{Z}$, $\varepsilon^2 = 0$ (called the ring of double integers).

In parallel with this erotic development, a type of cryptosystem will be presented by Rivest, Adleman and Dertouzos in 1978 [Riv78b], and built for the first time by Craig Gentry in 2009 [Gen09], it is the homomorphic cryptosystem which allows perform arithmetic operations on

encrypted data without decrypting them, the result of which is the encrypted result of operations carried out on its corresponding plain text. This concept has long been considered an open problem until the revolutionary publication of Gentry [Gen09] which demonstrates the feasibility of calculating any function on encrypted data. Since then, many constructs have appeared involving new mathematical and algorithmic concepts and improving efficiency, and great strides have been made both to base the security of FHE on more standard and well understood security assumptions, and to improve the effectiveness of Gentry's initial solution. Improving the efficiency of Gentry's scheme has received even more attention [Alp13][Alp14][Bra13][Gen11][Gen12a][Gen12b][Gen12c][Gen12d] [Hal14], resulting in improved asymptotic performance, and some reference implementations and libraries [Gen11] [Hal14] which are quite efficient to run on a personal computer and the main open theoretical problem that remains to be solved is how to remove the "circular security" hypothesis formulated in [Gen09]. In homomorphic encryption, messages are encrypted with a noise which increases with each homomorphic evaluation of an elementary operation. In a somewhat encrypted scheme, the number of homomorphic operations is limited, but can be made asymptotically large using the bootstrap [Gen09]. This technical trick introduced by Gentry makes it possible to evaluate arbitrary circuits by essentially evaluating the decryption function on the encrypted secret keys. This step remained very expensive until the recent publication of Ducas and Micciancio [Duc15], which presented a very fast boot procedure operating in approximately 0.69 seconds, making an important step towards a practical FHE for arbitrary NAND circuits. In practice, the homomorphic cryptosystem is very useful worldwide and especially in more secure cloud services that respect everyone's privacy, as well as in electronic voting and homomorphic property provides a tool to calculate the count from the encrypted votes without deciphering individual votes, only the final result is decrypted, also, we find the homomorphic cryptosystem in the context of path calculation. Confidentiality and protection

of information is fundamental in the context of path calculation. When a path calculation client (PCC) requests the shortest path between two nodes to a path calculation element (PCE), it wishes to do so while protecting the sensitive information carried by the request as well as the global topology of the network.

Motivation

Since its invention, public key cryptography has gone from a mathematical curiosity to an essential element of our IT infrastructure. It has been used to verify the authenticity of software and legal documents, to protect financial transactions and to protect the transactions of millions of Internet users on a daily basis.

Although the problem of integer factorization and the problem of the general discrete logarithm are considered difficult in classical computational models, it has been shown that no other problem is difficult in the quantum computational model. It was suggested by Feynman and demonstrated by Deutsch and Jozsa.

Since the publication of the NTRU encryption scheme in 1996, several researchers have tried on the one hand to find the best parameters to properly secure this cryptosystem against all types of attack and on the other hand, to find better or more secure variants quickly to NTRU.

We are motivated by the fact that there are several variants of NTRU and by the fact that it resists quantum attack.

Also, the calculations on encrypted values have attracted valuable attention, and quite a few authors are interested in this type of cryptosystem that is called homomorphic cryptosystem and in the use of this type in the digital world. This type of encryption is an effective approach to data protection.

We are motivated to implement this type of cryptosystems in a usable domain.

Objectives

The objective of this thesis is divided in two directions the first is to develop a new variant of NTRU that it will be more secure and faster to NTRU and to one of these variants Matru, on the other hand the second is a homomorphic cryptoseystem implementation in Path Computation Element. Finally, we increase the scale of the PCE from hundreds to hundreds of thousands of nodes, we provide a two-layer architecture with proven capability, to scale a relatively large network.

Contributions

The main research contributions of this thesis are as follows:

1. We have built an encryption system PMTRU [Haj20b] based on purely mathematical calculations and offers a level of security well comparable with Ntru and Matru. We will demonstrate in detail that our system and faster and more secure at NTRU and Matru, we will present the possible attacks and manage the parameters of our system to diffuse these attacks and protect this system well to have a level of security more than that of NTRU and Matru.
2. The shortest path problem (SP) is the problem of finding one of the shortest possible paths from a node i to a node j in a certain graph. We are trying to develop a protocol to secure requests to search the shortest path with constraint in a certain graph [Haj20a], this protocol uses a homomorphic cryptography system, we write the electronic diagram, and we calculate their complexity in detail.
3. in the last part, Current Path Computation Elements (PCE) are limited to scaling only hundreds to a few thousand nodes, we provide a two-layer architecture with proven capability [Haj20c], based on simulations, to scale a relatively large network.

Thesis Outline

The remainder of this thesis is organized as follows. In Chapter 1, we show the description of NTRU and Matru, and we present in detail our cryptosystem PMTRU [Haj20b] with security analysis. Then, in Chapter 2, we implement the cryptosystem of type homomorphic in Path Computation Elements, we calculate the shortest path with constraint in a secure environment [Haj20a]. In Chapter 3, we shift from Path Computation Elements to Scalable Path Computation Elements [Haj20c]. Finally, Chapter 4 concludes our thesis report and provides future research directions.

List of Publications

Articles in Referred Journals

Hamid Hajaje, Zine El Abidine Guennoun, Mounib Khanafer, Junaid Israr, Mouhcine Guennoun, A Collision-Aware MAC Protocol for Efficient Performance in Wireless Sensor Networks, International Journal of Advanced Computer Science and Applications (IJACSA), Volume 12 No 3, March 2021.

Hajaje, H., Guennoun, Z. D. A., and Guennoun, M. 2020. "*Constraint-Based Privacy Preserving Path Computation Element*," International Journal of Smart Security Technologies, IGI Global, Volume 6, Issue 2.

Articles in Referred Conference Proceedings

Hajaje, H., Guennoun, Z. D. A., and Guennoun, M. 2020. "*PMTRU: another variant of the NTRU public key cryptosystem*," the 2020 IEEE Canadian Conference of Electrical and Computer Engineering, London, Ontario, Canada, April 26-29, 2020.

Hajaje, H., Guennoun, Z. D. A., Israr, J., and Guennoun, M. 2020. "*Scalable Path Computation Element (SPCE)*," the 2020 IEEE Canadian Conference of Electrical and Computer Engineering, London, Ontario, Canada, April 26-29, 2020.

Chapter 1

The Public Key Cryptosystem PMTRU

1.1 Introduction

The asymmetric cryptosystem NTRU [Hof98] operates in the polynomial ring, $Z[X]/(X^N - 1, q)$ where N, q are integers. The encryption-decryption procedures use small integers, which makes the complexity of the scheme in the order of N^2. This characteristic makes NTRU faster than previous public key cryptosystems (e.g. RSA [Riv78a], McEliece [Mce78], ElGamal [Elg85] , ECC [Kob87]), and it is proven to be resistant to quantum algorithms. However, the NTRU cryptosystem can be vulnerable to lattice attacks ([Cop97]), which use LLL algorithms [Len82]. Several variants ([Mal10] [Jar13]) have been developed to improve speed and resistance; for example, Coglianese and Goi [Cog05] developed the MaTRU system, which replaces the commutative ring $Z[X]/(X^n - 1, q)$ with the non-commutative ring of matrix $M_{k\times k}[Z[X]/(X^n - 1, q)]$, and uses different linear transformations in the encryption and decryption procedures. These transformations improved the encryption/decryption speed by a factor of $o(k)$ over NTRU, in return for a somewhat larger public key. Moreover, if Strassen's multiplication [Cha76] is used, the MaTRU speed can also be increased from $o(n^2k^3)$ to $o\left(n^2k^{log_2^7}\right)$ [Cog05]. An analysis by J.E. Song, T.Y. Han and M.L. Lee [Son15] proved it is impossible to generate appropriate key pairs, and the value of parameter q is too small to be achievable.

We propose the PMTRU cryptosystem [Haj20b], which operates in the ring of square matrices of order k with elements in $R_q = Z[X]/(X^n - 1, q)$, and complexity equal to $o\left(n^2k^{log_2^7}\right)$. This

means it is more efficient than an NTRU system. Even though PMTRU has the same complexity as a MaTRU system, it has the advantage of requiring less processing power. Since PMTRU uses the same encryption/decryption procedures as NTRU, we achieved a significant gain in terms of processing time. To generate public keys, we chose specific permutations to increase security and robustness.

The remainder of the chapter is as follows: Section 2.2 discusses the notations used throughout the chapter, Section 2.3 provides a detailed description of the proposed NTRU cryptographic system, Section 2.4 provides a detailed description of the proposed MaTRU cryptographic system, and Section 2.5 provides a detailed description of the proposed PMTRU cryptographic, analysis of the security of the system, and example of a PMTRU instance. In section 2.6 we compare the new variant PMTRU with MATRU and NTRU.

1.2 Notations

We represent each polynomial by a vector that contains the coefficients, and use the following notations:

R denotes the set of elements of $Z[X]/(X^n - 1)$ equipped with the usual addition and convolution product $*$ defined by:

$$\alpha * \beta = \left(\sum_{i+j=k \bmod n}^{n} (\alpha_i . \beta_i) \right)_{0 \leqslant k \leqslant n-1}$$

for all α and β in R.

The ring of square matrices (a_{ij}) of order k, $a_{ij} \in R$ for all $0 \leqslant i \leqslant k-1, 0 \leqslant j \leqslant k-1$ are denoted by $M_{k\times k}(R)$ equipped with the matrix product $\circledast$ and defined by:

$$M \circledast N = \left(\sum_{t=0}^{k-1} (M_{i,t} * N_{t,j}) \right)_{0 \leqslant i,j \leqslant k-1}$$

for all $M = \left(M_{i,j}\right)_{0 \leqslant i,j \leqslant k-1}$ and $N = \left(N_{i,j}\right)_{0 \leqslant i,j \leqslant k-1}$ in $M_{k\times k}(R)$.

For any integer q, R_q denotes all elements of R/q with multiplication modulo q and $M_{k\times k}(R_q)$ represents all square matrices whose coefficients $a_{ij} \in R_q$, for all $0 \leqslant i \leqslant k-1, 0 \leqslant j \leqslant k-1$.

The set of permutations of k elements are denoted S_k, and each permutation σ of S_k, is associated with a matrix $P_\sigma = \left[P_{\sigma_{i,j}}\right]_{0 \leqslant i,j \leqslant k-1}$ defined by:

$$P_{\sigma_{i,j}} = \begin{cases} 1 & if \quad \sigma(i) = j \\ 0 & if \quad \sigma(i) \neq j \end{cases}$$

The matrix P_σ is a product of elementary matrices that operate on the identity matrix by swapping the i^{th} row with the j^{th} column if $\sigma(i) = j$.

For all $M \in M_{k\times k}[R]$ we define R_σ and C_σ with $R_\sigma(M)$, giving a permutation of rows of M and $C_\sigma(M)$ and a permutation of columns of M : $R_\sigma(M) = P_\sigma \circledast M$ and $C_\sigma(M) = M \circledast P_\sigma$.

For all $k \in \{0,\dots,n-1\}$, π_k is a circular permutation of order k from R to R, or for all $\alpha \in R$: $\pi_k\left((\alpha_0,\dots,\alpha_{n-1})\right) = (\alpha_k, \alpha_{k+1 \bmod n}, \dots, \alpha_{k+n-1 \bmod n})$.

From the definition, we obtain the following property: $\forall\, \alpha, \beta \in R$ and $i,j,k \in \{0,\dots,n-1\}$,

if $i + j \equiv k \bmod n$ then $\pi_k(\alpha * \beta) = \pi_i(\alpha) * \pi_j(\beta)$.

We define the width of a matrix $M = \left[M_{i,j}\right]_{0 \leqslant i,j \leqslant k-1} \in M_{k\times k}[R]$ by:

$$|M|_\infty = \left(\max_{0 \leqslant i,j \leqslant k-1}\{coefficient\ of\ M_{i,j}\} - \min_{0 \leqslant i,j \leqslant k-1}\{coefficient\ of\ M_{i,j}\}\right).$$

A matrix $M \in M_{k\times k}[R]$ is called short if and only if $|M|_\infty < p$, and quite short if and only if $p \leqslant |M|_\infty < q$.

To analyze the security cryptosystems, we also define the size of i^{th} row of a matrix $M \in M_{k\times k}[R]$, by: $|M_i| = \sqrt{\sum_{0 \leqslant j \leqslant k-1} |M_{i,j}|^2}$. Likewise, the size of a matrix $M \in M_{k\times k}[R]$,

is:

$$|M| = \sqrt{\sum_{0 \leqslant i \leqslant k-1} |M_i|^2}$$

1.3 Cryptosystem NTRU

The asymmetric cryptosystem NTRU [Hof98] operates in the ring of polynomials $\mathbb{Z}[X]/X^N - 1$, where N is an integer, and uses reductions modulo two prime numbers (or polynomials) between them p and q, it also uses the four constants $d_f, d_g, d_r$and $d_m \in \mathbb{N}$.

These parameters N, p, q, d_f, d_g and d_r play an important role for the security of NTRU against known attacks and for distinct versions of NTRU in connection with the security level.

NTRU function under the ring $\mathrm{Z[X]/X^N} - 1$ provided with the natural sum term to term of the same degrees and uses the product of convolution as a law of multiplication, where for all $f, g \in \mathrm{Z[X]/X^N} - 1$: the convolutional product of f and g is:

$$f * g[X] = \sum_{k=0}^{N-1} \left(\sum_{i+j \equiv k[N]} f_i . g_j \right)_k . X^k.$$

In practice, the complexity of a convolution product has been reduced from $O(N^2)$ to $O(d \times N)$ thanks to an algorithm (See [Zha11]), we find in this chapter also two other important algorithms, such that the first is to test if a polynomial is invertible or not and the second is to calculate the inverse of a polynomial modulo p with less complexity.

Now, suppose that Alice wants to send a message M to Bob. So, in a typical scenario:

1. Bob must generate a public key h and private keys f et f_p .
2. Alice encrypts the message M by public key h and send it to Bob.
3. Bob will decipher the received message and find the clear message M.

1.3.1 Keys generation:

Bob will randomly choose two polynomials $f, g \in \mathrm{Z}_p[\mathrm{X}]/\mathrm{X}^{\mathrm{N}} - 1$, where f must be reversible modulo p and modulo q.

Bob will calculate f_q and f_p so that:

$$\begin{cases} f * f_q \equiv 1[q] \\ f * f_p \equiv 1[p] \end{cases}$$

Finally, the private key will be (f, f_p) and the public key will be: $h = p \cdot f_q * g$

1.3.2 Encryption procedure:

Alice will choose a random polynomial $r \in \mathrm{Z}_p[\mathrm{X}]/\mathrm{X}^{\mathrm{N}} - 1$.

And then it will calculate the encrypt message $e = h * r + m$, and send it to Bob.

In the choice of any polynomial f, g or r, using the values of the constants d_f, d_g ou d_r that they depend first on the type of p (integer or polynomial) and second in our value.

1.3.3 Decryption:

Suppose that Bob has received the message e from Alice and wants to decrypt it using his private key f. To do this efficiently, Bob should have precomputed the polynomial f_p described in keys generation part.

In order to decrypt e, Bob first computes

$$a \equiv f * e \; mod \; q$$

where he chooses the coefficients of a in the interval from $-\frac{q}{2}$ to $\frac{q}{2}$. Now, the clear message is $m \equiv f_p * a \; mod \; p$.

1.3.4 Why Decryption Works:

The polynomial a that Bob computes satisfies

$$a \equiv f * e \equiv p \cdot f * h * r + f * m \equiv p \cdot f * f_q * g * r + f * m \equiv p \cdot g * r + f * m \, mod \, q$$

Then, a equal exactly $p \cdot g * r + f * m$ in $\mathrm{Z}_q[\mathrm{X}]/\mathrm{X}^{\mathrm{N}} - 1$.

Reducing a modulo p then gives him the polynomial $f * m \, mod \, p$, and multiplication by f_p retrieves the message m (mod p).

1.3.5 A Decryption Criterion:

In order for the decryption process to work correctly, it is necessary that

$$|p \cdot g * r + f * m \,|_\infty < q$$

Where $|f|_\infty = \max_{i<N}\{f_i\} - \min_{i<N}\{f_i\}$.

1.3.6 Security Analysis:

1.3.6.1 Brute force attacks

To find a private key by brute force, an attacker must try all possible f and testing if $f * h$ mod q has small entries. Similarly, an attacker can recover a message by trying all possible r and testing if $e - r * h$ mod q has small entries. usually, the security level is given by

$$\binom{Key}{Security} = \frac{1}{d_g!}\sqrt{\frac{N!}{(N-2d_g)!}}$$

$$\binom{Message}{Security} = \frac{1}{d_r!}\sqrt{\frac{N!}{(N-2d_r)!}}$$

Where d_g is the number of 1 and -1 in polynomial g and d_r is the number of 1 and -1 in polynomial r.

1.3.6.2 Meet-in-the-middle attack

it is an attack on the public key h. By writing $f = f_1 + f_2$, we get

$$f_1 * h + f_2 * h \equiv p * g \, mod \, q$$

This attack consists in testing polynomials f_1 and f_2 whose total number of non-zero coefficients equal to 1 is d_f by calculating $f_1 * h + f_2 * h \pmod q$, and by testing whether the polynomial obtained is of the form $p * g$.

1.3.6.3 Multiple transmission attacks

If someone sends a single message m several times using the same public key different randomly r_i.

Suppose that someone transmits $e_i = h * r_i + m$ for $i = 1, \dots, k$. An attacker can compute $(e_i - e_1) * h^{-1} = r_i - r_1$. However, the coefficients of the r_i are so small, and from this he will recover many of the coefficients of r_1 and recover the message m.

1.3.6.4 Lattice based attacks

In NTRU the public key is $h = p \cdot f_q * g$ checked $f * h = p \cdot g$, So there is a polynomial $\mu \in \mathbb{Z}[X]/X^N - 1$ where $f * \frac{h}{p} - q * \mu = g$, and therefore

$$(f, -\mu)\begin{pmatrix} 1 & \frac{h}{p} \\ 0 & q \end{pmatrix} = (f, g)$$

by applying the LLL algorithm on the matrix $\begin{pmatrix} 1 & \frac{h}{p} \\ 0 & q \end{pmatrix}$ we obtain a reduced base in which the first vectors are quite short, and we can thus determine f and g.

1.4 Cryptosystem MaTRU

the MaTRU cryptosystem [Cog05] works under the same general principles as the NTRU cryptosystem, except that it operates in the ring of k by k matrices of polynomials in

$\mathbb{Z}[X]/X^N - 1$, where k and N are integers. Matru used different linear transformation for encryption and decryption, the multiplication and addition of polynomials in $\mathbb{Z}[X]/X^N - 1$ is done in the usual manner like in NTRU, MaTRU also uses the parameters $p, q \in N$. The numbers p and q must be relatively prime but may not be prime. In general, p is much smaller than q.

Now, suppose that Alice wants to send a message M to Bob, where M is a matrix of polynomial. For do this, it takes:

1- Bob creates two keys the first (h, A, B) is public and the second is private.

2- Alice encrypts the message M by the public key h and send it to Bob.

3- Bob decrypts the encrypted message by their private key.

1.4.1 Key's creation:

Bob chooses two k by k matrices A,B where $A^0, A^1, \dots, A^{k-1}$ are linearly independent modulo q like also $B^0, B^1, \dots, B^{k-1}$ are linearly independent modulo q. Next, Bob randomly selects short polynomials $\alpha_0, \alpha_1, \dots, \alpha_{k-1} \in \mathbb{Z}[X]/X^N - 1$ and $\beta_0, \beta_1, \dots, \beta_{k-1} \in \mathbb{Z}[X]/X^N - 1$, Bob then constructs the matrices f, g by taking

$$f = \sum_{i=0}^{k-1} \alpha_i . A^i \text{ and } g = \sum_{i=0}^{k-1} \beta_i . B^i$$

As noted, the matrices f and g must have inverses modulo p and modulo q. We denote the inverses as F_p, F_q and G_p, G_q, where:

$$F_q * f \equiv I \bmod q \quad and \quad F_p * f \equiv I \bmod p$$

$$G_q * g \equiv I \bmod q \quad and \quad G_p * g \equiv I \bmod p$$

Note that I is a k by k identity matrix. So, the private key is (f, g).

For create the public key, Bob selects a random matrix ω and the public key is (h, A, B) where $h = F_q * \omega * G_q$

1.4.2 Encryption:

To encrypt a message M, Alice randomly generates the polynomials $\phi_0, \phi_1, \dots, \phi_{k-1} \in \mathbb{Z}[X]/X^N - 1$ and $\psi_0, \psi_1, \dots, \psi_{k-1} \in \mathbb{Z}[X]/X^N - 1$, Alice then constructs the matrices Φ, Ψ by taking

$$\Phi = \sum_{i=0}^{k-1} \phi_i . A^i \text{ and } \Psi = \sum_{i=0}^{k-1} \psi_i . B^i$$

and computes the encrypted message

$$e \equiv p(\Phi * \mathrm{h} * \Psi) + M \bmod q$$

Alice then sends e to Bob.

1.4.3 Decryption:

to decrypt e, Bob first computes

$$a \equiv f * e * g$$

where he chooses the coefficients of matrix a in the interval from $-\frac{q}{2}$ to $\frac{q}{2}$. Now, the clear message is $M \equiv F_p * a * G_p \ mod \ p$.

1.4.4 Why Decryption Works:

In decryption, Bob has $a \equiv f * e * g \equiv \ p\left(f * \Phi * \left(F_q * \omega * G_q\right) * \Psi * g\right) + f * M * g$

Although matrix multiplication is not generally commutative, f and Φ here do indeed commute:

$$f * \Phi = \left(\sum_{i=0}^{k-1} \alpha_i . A^i\right)\left(\sum_{i=0}^{k-1} \phi_i . A^i\right) = \sum_{i=0}^{k-1} \sum_{i \equiv j+l[k]} \alpha_j . A^j . \phi_l . A^l = \sum_{i=0}^{k-1} \sum_{i \equiv j+l[k]} \phi_l . A^l . \alpha_j . A^j$$

$$= \Phi * f$$

Similarly, $g * \Psi \equiv \Psi * g[q]$. So, Bob has $a \equiv p(\Phi * \omega * \Psi) + f * M * g$.

Reducing a modulo p then gives him the polynomial $f * M * g \; mod \; p$, and multiplication by F_p from the left and by G_p from the right retrieves the message M (mod p).

1.4.5 Security Analysis:

1.4.5.1 Brute force attacks

An attacker can recover the private key by trying all possible values of $\alpha_0, \alpha_1, \ldots, \alpha_{k-1} \in \mathbb{Z}[X]/X^N - 1$ and $\beta_0, \beta_1, \ldots, \beta_{k-1} \in \mathbb{Z}[X]/X^N - 1$ for finding f and g where all integers in $f * h * g$ are smalls. Then, the security level for finding a private key (f, g) is:

$$\binom{Key}{Security} = \left(\frac{n!}{(n-(p-1)d_f)!\,(d_f!)^{p-1}}\right)$$

And also, an attacker can finding the message M by trying all possible values of $\phi_0, \phi_1, \ldots, \phi_{k-1} \in \mathbb{Z}[X]/X^N - 1$ and $\psi_0, \psi_1, \ldots, \psi_{k-1} \in \mathbb{Z}[X]/X^N - 1$, and calculate $e - p(\Phi * \mathrm{h} * \Psi) \bmod q$. So, the security level for finding a message M is also:

$$\binom{Message}{Security} = \left(\frac{n!}{(n-(p-1)d_\Phi)!\,(d_\Phi!)^{p-1}}\right)$$

1.5 Cryptosystem PMTRU

In the following, we use the four non-null integers k, n, p and q, which are relatively prime, and stay with $p = 3$ or $p = 4$. For security reasons, it is preferable to consider k as a prime integer, and n is also a prime integer for defending the attack of Gentry described in [Gen01].

Ultimately, for choice q, it's preferable to take a power of a prime number.

Also, we will use the six sets of matrices L_f, L_g, L_r, L_m, L_w and $L_{w'}$ such that:

$\boldsymbol{L_f}$: is the set of matrices used to create the private key.

$\boldsymbol{L_g}$: is a set of matrices used in the key generation step.

$\boldsymbol{L_r}$: is a set of matrices used to randomize the encryption process.

$\boldsymbol{L_m}$: is the set of possible messages.

$\boldsymbol{L_w}$: is a set of specific matrices used to construct the public and private key; and,

$\boldsymbol{L_{w'}}$: is a set of matrices used to construct the public key.

Like the NTRU cryptosystem, the elements of these sets have small integers encoded as short integers.

When defining the sets of short matrices below, we first define the set $L(d_1, d_2)$ of polynomials from R, such that:

$$\boldsymbol{L(d_1, d_2)} = \left\{ P \in R \middle| \begin{matrix} has\ d_1\ \ coefficients\ equal\ to -1\ and\ d_2\ coefficients\ equal\ to\ 1 \\ with\ the\ rest\ of\ coefficients\ equal\ to\ 0. \end{matrix} \right\}$$

We also define the set of matrices $L_M(d_1, d_2, d_3)$ by:

$$L_M(d_1, d_2, d_3) = \left\{ M \in M_{k\times k}[R] \middle| \begin{matrix} \text{For each row } M_i \text{ of } M \text{ we take one element of } L(d_1, d_3), \\ \text{and the rest of the elements are from } L(d_1, d_2). \end{matrix} \right\}$$

Then, to specify the sets L_f, L_g and L_r we only need to define the parameters d_f, d_g, d_r which are fixed integers. Ideally, we take $L_f = L_M(d_f, d_f, d_f + 1)$, $L_g = L_M(d_g, d_g, d_g)$ and $L_r = L_M(d_r, d_r, d_r)$. To define L_w and $L_{w'}$ we must first define the set of vectors of polynomials $L_R(d_1, d_2)$, such that:

$$\boldsymbol{L_R(d_1, d_2)} = \left\{ L \in R^k \middle| \begin{matrix} \text{has } d_2 \text{ elements equal to } (-1,0,\ldots,0), d_1 \text{ elements} \\ \text{equal to } (1,0,\ldots,0) \text{ and the rest of the elements are null} \end{matrix} \right\}$$

is the set of matrices $L_M(d_1, d_2)$, such that:

$$\boldsymbol{L_M(d_1, d_2)} = \left\{ M \in M_{k\times k}[R] \middle| \begin{matrix} \exists (m, \rho, \sigma_1, \sigma_2) \in (M_{k\times k}[Z_n] \times L_R(d_1, d_2) \times S_k \times S_k) \\ \text{such us } M = R_{\sigma_1}\left(C_{\sigma_2}\left(\left[\pi_{m_{i,j}}\left(\rho_{j-i\ mod\ k} \right) \right]_{0 \leqslant i,j \leqslant k-1} \right) \right) \end{matrix} \right\},$$

$m_{i,j}$ is the element of the i^{th} row and j^{th} column of the matrix m and $\left[\pi_{m_{i,j}}(\rho_{j-i \bmod k})\right]_{0 \leqslant i,j \leqslant k-1}$ is a matrix with coefficients $\pi_{m_{i,j}}(\rho_{j-i \bmod k})$.

The width of M is $|M|_\infty = |\rho|_\infty = 2$ and the size of $L_R(d_1, d_2)$ is $\frac{k!}{(k-d_1-d_2)!d_1!d_2!}$, while the size of $L_M(d_1, d_2)$ is:

$$|L_M(d_1, d_2)| = (k!)^2 (n^{d_1+d_2})^k |L_R(d_1, d_2)| = \frac{n^{k(d_1+d_2)}(k!)^3}{(k-d_1-d_2)!\, d_1!\, d_2!}$$

Thus, L_w and $L_{w'}$ consist of matrices of the form $L_M(d_w, d_w)$ and $L_M(d_{w'}+1, d_{w'})$, respectively.

Notes:

- If the width of $w \in L_w$ $(respectively\ w \in L_{w'})$ is equal to *2*, then it is a short matrix.
- The size of matrix $w \in L_w$ $(respectively\ w \in L_{w'})$ is bounded by $\sqrt{2d_w k}$ $respectively\ by\ \sqrt{(2d_{w'}+1)k}$
- The matrix $w \in L_w$ is not necessarily invertible modulo q.

The set of messages L_m consists of all matrices of polynomials with coefficients of modulo p. We therefore express

$$L_m = \left\{ M \in M_{k\times k}[R] \;\middle|\; \begin{array}{c}\text{Each polynomial in } M \text{ has coefficients} \\ \text{between } \left\lceil \frac{1-p}{2} \right\rceil \text{ and } \left\lfloor \frac{p-1}{2} \right\rfloor. \end{array} \right\}$$

1.5.1 Keys Generation

The key generation process is performed by an encrypted message receiver. It can determine the desired security level by choosing one of k, n, p and q.

The procedure is as follows:

1. Select a matrix $f \in L_f$;
2. Calculate the inverse f_q of f in $M_{k\times k}[R_q]$ (i.e., $f \circledast f_q = I \bmod q$, such that I is matrix

identity in $M_{k\times k}[R]$).

3. Select a matrix $w' \in L_{w'}$;
4. Calculate the inverse ${w'}_q$ of w' in $M_{k\times k}[R_q]$ (i.e., $w \circledast {w'}_q = I \bmod q$;
5. Calculate the inverse F_p of $w' \circledast f$ in $M_{k\times k}[R_p]$ (i.e., $F_p \circledast w' \circledast f = I \bmod p$.
6. Select a matrix g from L_g and w from L_w; and,
7. Calculate the public key:

$h = p.{w'}_q \circledast f_q \circledast g \circledast w$ in $M_{k\times k}[R_q]$.

The private key is: $F = f \circledast w$'.

1.5.2 Encryption procedure

Encrypting a message $m \in L_m$ requires the following steps:

1. Select a matrix r from L_r; and,
2. Calculate the encrypted text $e = h \circledast r + m$ in $M_{k\times k}[R_q]$.

1.5.3 Decryption

To decrypt the message e $\in M_{k\times k}[R_q]$, use the secret key F and the following procedure:

1. Calculate $a = F \circledast e$ in $M_{k\times k}[R_q]$ with coefficients in the range $\left[\frac{-q}{2}, \frac{q}{2}\right[$; and,
2. The clear text is then $m = F_p \circledast a$ in $M_{k\times k}[R_p]$.

1.5.4 Decryption correctness

To ensure the accuracy of the decryption, we first calculate $a = F \circledast e$ in $M_{k\times k}[R_q]$, knowing that

$e = h \circledast r + m$ and $h = p.{w'}_q \circledast f_q \circledast g \circledast w$.

Thus, we have:

$a = F \circledast e$

$= F \circledast \left(p.\left({w'}_q \circledast f_q \circledast g \circledast w\right) \circledast r + m\right)$

$= p.f \circledast w \circledast w'_q \circledast f_q \circledast g \circledast w \circledast r + F \circledast m$

$= p.g \circledast w \circledast r + f \circledast w' \circledast m$

The decrypting is exact if $|a'|_\infty < q$, such that $a' = p.g \circledast w \circledast r + f \circledast w' \circledast m$ is the exact calculation in $M_{k\times k}[R]$.

To decrypt a message, we resolve the equation:

$a = F \circledast e = p.(g \circledast w \circledast r) + f \circledast w' \circledast m$ in $M_{k\times k}(R_q)$.

a' denotes the exact calculation of $p.(g \circledast w \circledast r) + f \circledast w' \circledast m$ in $M_{k\times k}(R)$, and the size of a' is always less than q. However, for all

$$(\alpha, \beta) \in \left(R, \left(\bigcup_{i=0}^{n-1}\{\pi_i((-1,0,\dots,0))\}\right) \bigcup \left(\bigcup_{i=0}^{n-1}\{\pi_i((1,0,\dots,0))\}\right)\right),$$

we have: $|\alpha * \beta|_\infty = |\alpha|_\infty$, which means for all i of $\{0,1,\dots,k-1\}$ we have $\left|\sum_{j=0}^{k-1} g_{i,j} * w_{j,i}\right|_\infty \leqslant 4d_w$ Likewise, $\left|\sum_{j=0}^{k-1} f_{i,j} * w'_{j,i}\right|_\infty \leqslant 2(2d_{w'}+1)$and $|g \circledast w|_\infty \leqslant 4d_w$ and $|f \circledast w'|_\infty \leqslant 2(2d_{w'}+1)$and $|g \circledast w \circledast r|_\infty \leqslant 8kd_g d_w$ and$|f \circledast w' \circledast m|_\infty \leqslant 2k(p-1)(2d_{w'}+1)d_f$.

For security reasons, we chose $|g \circledast w \circledast r|_\infty \approx |f \circledast w' \circledast r|_\infty$, then, $2k(p-1)(2d_{w'}+1)d_f \approx 8pkd_g d_w$ is equivalent to $d_f(p-1)(2d_{w'}+1) \approx 4pd_g d_w$.

Typically, to choose $L_{w'}$ and L_w we use $L_w = L_M(\mathrm{d_w}, \mathrm{d_w})$ and $L_{w'} = L_M(\mathrm{d_{w'}}+1, \mathrm{d_{w'}})$ such that $\mathrm{d_w}$ and $\mathrm{d_{w'}}$ are fixed integers. The next section discusses the best choices for $\mathrm{d_w}$ and $\mathrm{d_{w'}}$.

1.5.5 Security Analysis

1.5.5.1 Brute-Force attacks to recover the private key

There are two ways to apply brute-force attacks to find the private key F. The first is to search for all possibilities of the matrix F, and the second is to look for all possibilities of f and w', such as:

$|F \circledast h|_\infty = |f \circledast w' \circledast h|_\infty = |g \circledast w|_\infty \leqslant 4d_w.$

To find F we need to evaluate every possible case for it. Since each coefficient of F is bounded between $-2d_{w'} - 1$ and $2d_{w'} + 1$, the brute force attack is equivalent to finding k elements from $M_{1\times k}(R_{d_{w'}})$, such that the size of $M_{1\times k}(R_{d_{w'}})$ is given by:

$$\#F = (4d_{w'} + 3)^{nk},$$

Similarly, if we want to find f and w' we can identify k elements $f_0, f_1, \ldots, f_{k-1}$ of $M_{1\times k}(R_p)$ and one matrix w', such that for all i of $\{0, \ldots, k-1\} : |f_i \circledast w' \circledast h|_\infty \leqslant 4d_w$.

The size of $M_{1\times k}(R_p)$ is:

$$\#\mathrm{f} = \mathrm{k}\left(\frac{\mathrm{n!}}{(\mathrm{n} - 2\mathrm{d_f} - 1)!\,\mathrm{d_f!}\,(\mathrm{d_f} + 1)!}\right)\left(\frac{\mathrm{n!}}{(\mathrm{n} - 2\mathrm{d_f})!\,(\mathrm{d_f!})^2}\right)^{\mathrm{k-1}}.$$

and the number of possible values for w' is given by:

$$\frac{n^{k(2d_{w'}+1)}\,(k!)^3}{(k - 2d_{w'} - 1)!\,d_{w'}!\,(d_{w'} + 1)!}$$

However, for all permutations σ of $S_k : C_\sigma(f) \circledast R_\sigma(w') = f \circledast w'$, we note that the size of S_k is $k!$. Let m be a matrix in $M_{k\times k}[Z_n]$, $\sigma_1, \sigma_2 \in S_k$ and

$\rho \in L_R(d_{w'} + 1, d_{w'})$, such that:

$$w' = R_{\sigma_1}\left(C_{\sigma_2}\left(\left[\pi_{m_{i,j}}(\rho_{j-i \bmod k})\right]_{0 \leqslant i,j \leqslant k-1}\right)\right)$$

$$= P_{\sigma_1} \circledast \left(\left[\pi_{m_{i,j}}(\rho_{j-i \bmod k})\right]_{0 \leqslant i,j \leqslant k-1}\right) \circledast P_{\sigma_2}$$

$$= \left[\pi_{m_{\sigma_1(i),\sigma_2(j)}}(\rho_{\sigma_2(j)-\sigma_1(i) \bmod k})\right]_{0 \leqslant i,j \leqslant k-1}$$

Then for all rows f_i of f and $(\alpha_0, \ldots, \alpha_{k-1}) \in Z_n^k$

$$\left(\pi_{\alpha_0}(\mathrm{f}_{i,0}), \ldots, \pi_{\alpha_{k-1}}(\mathrm{f}_{i,k-1})\right) \circledast \left[\pi_{-\alpha_i + m_{\sigma_1(i),\sigma_2(j)}}(\rho_{\sigma_2(j)-\sigma_1(i) \bmod k})\right]_{0 \leqslant i,j \leqslant k-1} = f_i \circledast w',$$

thus, the number of all possible values for one row of $f \circledast w'$ is :

$$\#(f, w') = \frac{\#f}{k!\, n^{(2d_{w'}+1)}}\left(\frac{n^{k(2d_{w'}+1)}(k!)^3}{(k-2d_{w'}-1)!\, d_{w'}!\, (d_{w'}+1)!}\right)$$

$$= \#f.\left(\frac{n^{(k-1)(2d_{w'}+1)}(k!)^2}{(k-2d_{w'}-1)!\, d_{w'}!\, (d_{w'}+1)!}\right)$$

Similarly, an attacker can also try to find all possible $G = g_1 \circledast w_1$ such that $g_1 \in L_g$, $w_1 \in L_w$, and test to see if $|G \circledast h^{-1}|_\infty = |g_1 \circledast w_1 \circledast h^{-1}|_\infty = |f \circledast w'|_\infty \leq 2(2d_{w'}+1)$. As with F, the number of possible values for the rows of G is $\#G = (4d_w + 1)^{nk}$, and the number of all possible values for one row of $g \circledast w$ is:

$$\#(g, w) = \#g.\left(\frac{n^{(k-1)(2d_w)}(k!)^2}{(k-2d_w)!\,(d_w!)^2}\right),$$

with:

$$\#g = \left(\frac{n!}{(n-2d_g)!\,(d_g!)^2}\right)^k.$$

In practice, L_g is smaller than L_f, and L_w will be smaller than $L_{w'}$, so the key security is determined by the minimums of $\#(g, w)$ and $\#G$. Typically, for large values of n and k $\#(g, w) \leqslant \#G$.

Similarly, an attacker can use a man-in-the-middle attack to find G by identifying g_1, g_2 and w, such that $|g_1 \circledast w \circledast h^{-1} + g_2 \circledast w \circledast h^{-1}|_\infty \leq 4d_{w'} + 2$. Hence the key security level is given by:

$$\binom{\boldsymbol{Key}}{\boldsymbol{Security}} = \sqrt{\#g}.\left(\frac{n^{(k-1)(2d_w)}(k!)^2}{(k-2d_w)!\,(d_w!)^2}\right)$$

1.5.5.2 The Lattice Attack

As $h = p.H$, with $H = {w'}_q \circledast f_q \circledast g \circledast w$ we have: $f \circledast H \equiv g \circledast w \bmod q$. There is a matrix $\mu \in M_{k\times k}\left[R_q\right]$, such that $F \circledast H - q\mu = f \circledast w' \circledast H - q\mu = g \circledast w$,

which will be interpreted by:

$$(f \circledast w', -\mu) \circledast \left(\begin{array}{c|c} \frac{I_{nk}}{0} & \begin{array}{ccc} C_{0,0} & \cdots & C_{0,k-1} \\ \vdots & \ddots & \vdots \\ C_{k-1,0} & \ldots & C_{k-1,k-1} \end{array} \\ & q.I_{nk} \end{array} \right) = (f, g \circledast w) \qquad (\mathbf{1})$$

$C_{i,j}$ is a circular order matrix of order $n \times n$ containing the coefficients of the polynomial $H_{i,j}$ for all $0 \leqslant i,j \leqslant k-1$.

Letting $F_0, \ldots, F_{k-1}$ be the rows of F, μ_0, ..., μ_{k-1} be the rows of μ and $V_0, \ldots, V_{k-1}$ be the rows of $V = g \circledast w$, we obtain system (1) consisting of k systems containing $2kn$ equations:

$$\begin{cases} (F_0, -\mu_0) \cdot H_p = (F_0, V_0) \\ \vdots \quad \vdots \quad \vdots \\ (F_{k-1}, -\mu_{k-1}) \cdot H_p = (F_{k-1}, V_{k-1}) \end{cases} \qquad (\mathbf{2})$$

such that $H_p = \left(\begin{array}{c|c} \frac{\mathrm{I_{nk}}}{0} & \begin{array}{ccc} \mathrm{C_{0,0}} & \cdots & \mathrm{C_{0,k-1}} \\ \vdots & \ddots & \vdots \\ \mathrm{C_{k-1,0}} & \ldots & \mathrm{C_{k-1,k-1}} \end{array} \\ & \mathrm{q.I_{nk}} \end{array} \right)$ is a square matrix of order $2kn$.

A lattice attack for this system is equivalent to finding the k shortest polynomial of order $2k$. Each component is a short polynomial of degree n which makes the attack very difficult, particularly for large values of k.

On the other hand, since the purpose of a lattice attack is to find the vectors (F_i, V_i) for all $0 \leqslant i \leqslant k-1$, such that F_i is not short but somewhat short because $|F_i|_\infty \leqslant 4d_{w'} + 2$, we also have that V_i is somewhat short because $|V_i|_\infty \leqslant 4d_w$ and the width of this vector is equal to : $|(F_i, V_i)| = \sqrt{|F_i|^2 + |V_i|^2}$, such that if f_i is the i^{th} row of f, and g_i is the i^{th} row of g, we have:

$|f_i| \approx \sqrt{k(p-1)d_f} \approx \sqrt{2kd_f}$ and $|g_i| \approx \sqrt{2kd_g}$.

Moreover, for all $(\alpha, \beta) \in \left(R, \left(\cup_{i=0}^{n-1}\{\pi_i((-1,0,\ldots,0))\}\right) \cup \left(\cup_{i=0}^{n-1}\{\pi_i((1,0,\ldots,0))\}\right)\right)$, we have: $|\alpha * \beta| = |\alpha|$.

Thus, equation (3) becomes:

$|(F_i, V_i)| = \sqrt{|(\mathrm{f} \circledast \mathrm{w}')_i|^2 + |(\mathrm{g} \circledast \mathrm{w})_i|^2} \approx \sqrt{(2d_{w'} + 1)|\mathrm{f}_i|^2 + (2d_w)|\mathrm{g}_i|^2} \approx$

$\sqrt{2k\big((2d_{w'} + 1) + 2d_w\big)d_f}$ for $d_g \approx d_f$.

Since we chose $d_f = \left\lfloor \frac{n}{3} \right\rfloor$ then:

$$|(F_i, V_i)| \approx \sqrt{nk\big((2d_{w'} + 1) + 2d_w\big)\left(1 - \frac{1}{3}\right)} \quad , \forall i \,/\, 0 \leqslant i \leqslant k - 1.$$

The *Gaussian heuristic* predicts that the maximum width of the shortest vector in L_i is given by $|shortest.vector|_{max} = \det(\mathrm{L})^{\frac{1}{\dim(\mathrm{L})}} \sqrt{\frac{\dim(\mathrm{L})}{2\pi \mathrm{e}}}$, such that $L_i = \{(F_i, V_i)\}$ for all i in $\{0,1, \ldots, k - 1\}$ and $dim(L) = dim(L_0) = \ldots = dim(L_{k-1}) = 2nk$ and $det(L) = det(L_0) = \ldots = det(L_{k-1}) = q^{nk}$, then

$$|shortest.vector|_{max} = \sqrt{\frac{qnk}{\pi \mathrm{e}}}.$$

Taken thatc_p equals the ratio of $|(F_i, V_i)|$ to the $|shortest.vector|_{max}$

$c_p = \frac{|(F_i, V_i)|}{|shortest.vector|_{max}} \approx \sqrt{\frac{2\pi \mathrm{e}\left((2d_{w'} + 1) + 2d_w\right)}{3q}}$.

If c_p is close to 0, the width of (F_i, V_i) is much smaller than the widths of other vectors in the lattice, and easier to determine. If c_p is near 1, the widths of many vectors will approach that of (F_i, V_i), and determination of the vector (F_i, V_i) will be more difficult.

A comparison of the ratio c_p with the ratio $c_h \approx \sqrt{\frac{2\pi \mathrm{e}(2\mathrm{n}+3)}{9q}}$ for $p = 3$ of a MATRU cryptosystem allows us to limit the parameters $d_{w'}$ and d_w, and have a c_p ratio greater than c_h. Thus, we have:

$$c_h < c_p \quad \Leftrightarrow \quad 0 < {c_p}^2 - {c_h}^2$$

$$\Leftrightarrow \quad 0 < \frac{6\pi e\left((2d_{w'} + 1) + 2d_w\right) - 2\pi e(2n+3)}{9q}$$

$$\Leftrightarrow \quad 0 < 2\pi e.\frac{\left((2d_{w'} + 1) + 2d_w\right) - (2n+3)}{3q} \quad \Leftrightarrow \quad n + 1 < d_{w'} + d_w$$

1.5.5.3 Attack on the Encrypted Message

There is also a brute-force attack on the scope of the encrypted message $e \equiv h \circledast r + m \bmod q$. By applying $e - h \circledast r \equiv m \bmod q$, the attack tests the polynomials $r \in L_r$ to verify if the polynomial obtained by $e - h \circledast r \bmod q$ is in L_m. Similarly, we can target each line individually, and the total number of possibilities is of the form:

$$|L_r| = \left(\frac{n!}{(n-2d_r)!\,(d_r!)^2}\right)^k$$

An attacker can also use a man-in-the-middle attack to find r by determining r_1 and r_2, such that $|(e - h \circledast r_1)|_\infty = |h \circledast r_2|_\infty \pm 1 \text{ or } 0$. Hence, the message security level is given by:

$$\binom{\boldsymbol{Message}}{\boldsymbol{Security}} = \sqrt{|L_r|} = \left(\sqrt{\frac{n!}{(n-2d_r)!\,(d_r!)^2}}\right)^k$$

1.5.6 Choice of Parameters

The choice of the cryptosystem parameters d_w, $d_{w'}$, k, n, q and p plays a critical role in improving the security of the system.

This section presents some potential choices of parameters, subject to the above conditions. For $p = 3 \text{ or } 4$ and $d_f = \left\lfloor \frac{n}{3} \right\rfloor$ some and $d_g \approx d_r \approx d_f$, in reality, $d_f \gtrsim d_g$ and $d_f \gtrsim d_r$, the integer $d_{w'} + d_w$ is greater than $n + 1$. It is preferable to use $d_f(p-1)\,(2d_{w'} + 1) \approx 4pd_g d_w$.

Table 1 gives important information about the level of system security for each choice of parameters k, n, q and p. For example:

Column 10: The security level of the private key.

Column 11: The security level of messages regarding the difficulty to implement an exhaustive attack to find the clear message.

Column 12: The index of the level of security against a lattice attack.

1 **n**	**2** **k**	**3** **p**	**4** **q**	**5** d_f	**6** d_g	**7** d_r	**8** d_w	**9** $d_{w'}$	**10** **Security of key**	**11** **Security of Message**	**12** c_p
11	5	3	128	3	3	3	1	1	$2^{71.8}$	$2^{32.9}$	0.472
5	11	3	128	1	1	1	1	3	$2^{102.2}$	$2^{23.8}$	0.633
5	19	3	256	1	1	1	2	7	$2^{279.5}$	$2^{41.1}$	0.650
11	11	3	256	3	3	3	1	4	$2^{173.7}$	$2^{125.1}$	0.495
11	23	3	512	3	3	3	2	6	$2^{546.1}$	$2^{151.5}$	0.435
5	11	4	149	1	1	1	1	3	$2^{102.2}$	$2^{23.8}$	0.586
5	23	4	401	1	1	1	2	7	$2^{344.2}$	$2^{49.7}$	0.519
19	19	4	797	6	5	5	2	5	$2^{609.6}$	$2^{232.5}$	0.327

Table 1:Possible choice for parameter of PMTRU

The security of the key is very large compared to the security of the message. Most of the values of c_P in Table 1 are greater than or equal to 0.5, which proves the difficulty of finding a private key with the lattice attack. Finally, the values of q are small, which reduces the size of both the encrypted text and the public key.

1.5.7 Application

This section provides a detailed example of an instance of PMTRU in which we set $n = 5$, $k = 5$, $d_f = d_g = d_r = d_w = d_{w'} = 1$, $p = 3$, $q = 64$, and choose polynomials

f and w' as follows:

$$f = \begin{bmatrix} (0,0,-1,1,1) & (0,1,0,0,-1) & (-1,0,1,0,0) & (-1,0,0,0,1) & (-1,0,0,0,1) \\ (1,0,0,0,-1) & (-1,0,1,1,0) & (0,1,-1,0,0) & (-1,0,1,0,0) & (1,0,-1,0,0) \\ (0,0,0,-1,1) & (0,1,-1,0,0) & (0,1,0,-1,1) & (0,0,0,1,-1) & (0,-1,1,0,0) \\ (-1,0,1,0,0) & (0,-1,0,0,1) & (1,0,0,0,-1) & (1,0,0,-1,1) & (0,0,0,-1,1) \\ (0,0,1,0,-1) & (1,0,0,0,-1) & (0,1,0,-1,0) & (0,0,1,0,-1) & (0,-1,1,0,1) \end{bmatrix}$$

and

$$w' = \begin{bmatrix} (0,0,0,0,0) & (0,0,0,0,0) & (0,-1,0,0,0) & (0,1,0,0,0) & (1,0,0,0,0) \\ (0,0,0,0,1) & (0,0,0,0,0) & (0,0,0,0,0) & (0,0,0,-1,0) & (1,0,0,0,0) \\ (0,0,1,0,0) & (0,0,1,0,0) & (0,0,0,0,0) & (0,0,0,0,0) & (0,0,0,0,-1) \\ (0,-1,0,0,0) & (0,0,0,0,1) & (0,1,0,0,0) & (0,0,0,0,0) & (0,0,0,0,0) \\ (0,0,0,0,0) & (-1,0,0,0,0) & (0,0,0,1,0) & (1,0,0,0,0) & (0,0,0,0,0) \end{bmatrix}$$

Then $F = f \circledast w'$ is:

$$F = \begin{bmatrix} (0,1,-1,-1,1) & (1,0,-1,1,-1) & (0,-1,1,0,-1) & (0,0,1,-1,1) & (0,0,-1,1,1) \\ (0,2,1,0,-2) & (-1,1,1,1,-2) & (0,-2,0,2,0) & (-1,0,-1,1,0) & (-1,1,1,1,-1) \\ (1,0,0,1,-1) & (-1,2,0,0,0) & (-1,0,0,0,1) & (2,-1,1,0,-2) & (-1,1,0,-2,1) \\ (-2,-2,1,1,1) & (0,-1,0,2,0) & (1,1,1,-1,-1) & (0,-1,-1,0,2) & (-1,-1,1,1,0) \\ (0,0,0,-1,1) & (-1,2,-1,0,-1) & (1,0,1,0,-1) & (-1,-1,2,0,1) & (0,0,2,0,-2) \end{bmatrix}$$

And the inverse of f modulo q is:

$$f_q = \begin{bmatrix} (3,2,17,17,26) & (49,15,2,23,39) & (12,23,9,48,36) & (44,4,52,14,14) & (13,55,8,36,16) \\ (59,19,9,9,32) & (22,47,21,28,11) & (47,15,42,42,46) & (7,35,37,38,11) & (4,7,16,9,28) \\ (48,43,31,40,30) & (63,25,2,23,15) & (10,1,47,4,3) & (50,28,19,20,11) & (30,59,1,63,39) \\ (13,11,26,43,35) & (39,53,36,19,45) & (22,30,37,19,20) & (44,3,46,56,44) & (31,31,33,46,51) \\ (47,43,47,27,28) & (51,55,31,19,36) & (60,43,8,18,63) & (36,25,41,19,7) & (24,7,9,58,31) \end{bmatrix}$$

the inverse of w' $modulo$ q is:

$$w'_q = \begin{bmatrix} (56,25,42,50,54) & (45,43,10,57,49) & (39,37,4,52,43) & (53,32,22,49,30) & (20,1,24,3,57) \\ (3,57,20,1,24) & (17,37,48,21,40) & (0,20,30,4,22) & (6,21,60,47,41) & (24,18,47,61,36) \\ (36,24,18,47,61) & (61,22,34,40,12) & (9,33,46,52,23) & (57,49,45,43,10) & (37,4,52,43,39) \\ (49,10,23,29,0) & (47,61,36,24,18) & (18,32,7,59,53) & (25,42,50,54,56) & (37,39,8,24,32) \\ (62,51,14,59,18) & (58,14,25,62,16) & (34,55,1,39,57) & (18,32,7,59,53) & (7,0,29,44,19) \end{bmatrix}.$$

and the inverse of F $modulo$ p is:

$$F_p = \begin{bmatrix} (-1,0,1,0,-1) & (1,0,0,1,0) & (1,-1,1,-1,1) & (-1,-1,1,1,-1) & (0,-1,-1,1,1) \\ (1,-1,0,1,-1) & (0,-1,0,1,-1) & (-1,0,-1,1,0) & (0,0,-1,-1,0) & (0,1,1,1,-1) \\ (0,0,0,0,-1) & (-1,1,-1,1,0) & (0,1,-1,-1,0) & (1,1,-1,1,0) & (0,0,1,-1,1) \\ (-1,1,-1,0,-1) & (0,-1,1,-1,0) & (-1,0,-1,0,-1) & (-1,0,1,1,1) & (1,1,0,0,0) \\ (0,0,1,-1,-1) & (0,0,1,0,0) & (-1,0,-1,1,0) & (1,1,0,0,1) & (-1,-1,-1,-1,0) \end{bmatrix}$$

For the random matrix g, we use:

$$g = \begin{bmatrix} (0,-1,0,1,0) & (0,0,0,-1,1) & (1,-1,0,0,0) & (1,0,0,0,-1) & (-1,0,1,0,0) \\ (1,-1,0,0,0) & (1,0,0,-1,0) & (0,-1,0,0,1) & (0,0,0,-1,1) & (0,1,-1,0,0) \\ (0,-1,0,0,1) & (-1,1,0,0,0) & (0,0,1,-1,0) & (1,0,0,-1,0) & (-1,0,1,0,0) \\ (0,-1,1,0,0) & (-1,0,1,0,0) & (0,0,-1,0,1) & (-1,0,0,0,1) & (0,0,0,-1,1) \\ (0,0,0,1,-1) & (0,0,1,-1,0) & (0,1,0,0,-1) & (1,0,-1,0,0) & (0,0,1,0,-1) \end{bmatrix}.$$

For w we use:

$$w = \begin{bmatrix} (0,0,0,0,0) & (0,0,0,0,0) & (0,0,0,0,0) & (1,0,0,0,0) & (0,0,0,0,-1) \\ (0,0,0,0,0) & (0,0,0,0,0) & (0,0,0,0,1) & (0,0,0,-1,0) & (0,0,0,0,0) \\ (0,0,0,0,0) & (0,0,0,0,1) & (0,0,-1,0,0) & (0,0,0,0,0) & (0,0,0,0,0) \\ (1,0,0,0,0) & (0,0,0,0,-1) & (0,0,0,0,0) & (0,0,0,0,0) & (0,0,0,0,0) \\ (0,0,0,-1,0) & (0,0,0,0,0) & (0,0,0,0,0) & (0,0,0,0,0) & (0,1,0,0,0) \end{bmatrix},$$

and the public key will be:

$$h = \begin{bmatrix} (6,60,55,4,3) & (34,59,16,53,30) & (7,12,22,27,60) & (46,44,11,50,41) & (4,40,40,14,30) \\ (52,9,13,46,8) & (1,37,54,32,4) & (14,5,34,28,47) & (58,44,28,36,26) & (11,57,49,35,40) \\ (19,62,14,16,17) & (26,28,60,13,1) & (43,16,42,42,49) & (13,15,6,44,50) & (32,63,52,58,51) \\ (41,48,13,0,26) & (12,46,22,7,41) & (16,61,47,23,45) & (21,8,2,58,39) & (11,17,2,57,41) \\ (57,60,16,62,61) & (23,44,10,14,37) & (23,46,32,2,25) & (27,11,27,47,16) & (17,42,56,31,46) \end{bmatrix}$$

To encrypt the message:

$$m = \begin{bmatrix} (1,0,-1,-1,0) & (-1,-1,0,-1,1) & (0,1,0,0,0) & (1,0,1,-1,1) & (1,-1,0,-1,1) \\ (-1,1,-1,0,1) & (0,1,-1,1,1) & (-1,-1,0,1,1) & (1,-1,1,1,-1) & (1,0,-1,-1,-1) \\ (0,-1,0,-1,0) & (-1,0,-1,0,1) & (-1,0,1,1,1) & (0,1,1,-1,-1) & (-1,1,1,1,1) \\ (-1,-1,-1,0,-1) & (1,1,1,1,0) & (0,0,-1,1,0) & (0,0,0,1,0) & (-1,-1,-1,-1,-1) \\ (0,0,-1,0,0) & (0,1,1,0,0) & (0,0,1,0,-1) & (1,-1,0,-1,1) & (1,1,-1,-1,-1) \end{bmatrix}$$

select a random matrix:

$$r = \begin{bmatrix} (1,0,0,-1,0) & (0,-1,1,0,0) & (0,0,-1,0,1) & (0,0,-1,0,1) & (-1,0,0,1,0) \\ (-1,0,1,0,0) & (-1,1,0,0,0) & (-1,1,0,0,0) & (1,0,-1,0,0) & (0,-1,0,0,1) \\ (0,0,1,-1,0) & (1,-1,0,0,0) & (1,0,-1,0,0) & (0,1,0,0,-1) & (1,0,-1,0,0) \\ (0,-1,0,0,1) & (1,0,0,-1,0) & (0,0,-1,1,0) & (0,0,1,-1,0) & (1,0,0,0,-1) \\ (0,1,0,-1,0) & (1,0,0,-1,0) & (0,1,0,-1,0) & (0,0,-1,0,1) & (-1,0,0,1,0) \end{bmatrix}$$

and the encrypted text will be:

$$e = \begin{bmatrix} (33,15,32,10,37) & (6,60,43,50,31) & (47,43,61,23,19) & (23,3,45,14,45) & (33,12,36,29,18) \\ (8,24,55,26,15) & (0,6,1,15,44) & (62,40,46,33,11) & (5,16,17,15,12) & (33,42,56,28,31) \\ (20,56,62,48,4) & (18,9,35,24,41) & (46,58,14,47,29) & (7,33,59,50,43) & (40,24,13,3,51) \\ (6,63,37,7,11) & (3,35,56,46,56) & (53,6,7,24,38) & (53,31,35,17,57) & (37,23,51,58,18) \\ (51,0,61,22,57) & (38,46,39,44,27) & (3,36,34,8,47) & (18,59,61,29,25) & (43,48,57,58,49) \end{bmatrix}$$

Finally, to decrypt the encrypted message we first calculate $a = (F \circledast e \bmod q) \bmod p$, such that:

$$a = \begin{bmatrix} (1,-1,1,-1,0) & (0,-1,1,1,0) & (0,0,0,-1,-1) & (0,0,1,0,0) & (1,1,-1,-1,0) \\ (0,0,0,0,-1) & (0,-1,0,1,-1) & (1,-1,0,-1,-1) & (0,-1,0,-1,0) & (1,-1,-1,0,-1) \\ (-1,0,1,0,0) & (0,1,0,1,-1) & (1,1,1,-1,-1) & (0,-1,1,-1,1) & (-1,1,0,1,1) \\ (0,0,0,0,-1) & (1,1,0,-1,-1) & (1,-1,-1,1,1) & (0,0,1,0,1) & (-1,1,0,1,0) \\ (1,0,1,0,1) & (1,-1,-1,1,1) & (0,-1,-1,1,0) & (-1,0,0,-1,-1) & (0,-1,1,1,-1) \end{bmatrix}$$

and the clear text message $m' = F_p \circledast a \bmod p$ is:

$$m' = \begin{bmatrix} (1,0,-1,-1,0) & (-1,-1,0,-1,1) & (0,1,0,0,0) & (0,0,1,0,0) & (1,1,-1,-1,0) \\ (-1,1,-1,0,1) & (0,1,-1,1,1) & (-1,-1,0,1,1) & (1,-1,1,1,-1) & (1,0,-1,-1,-1) \\ (0,-1,0,-1,0) & (-1,0,-1,0,1) & (-1,0,1,1,1) & (0,1,1,-1,-1) & (-1,1,1,1,1) \\ (-1,-1,-1,0,-1) & (1,1,1,1,0) & (0,0,-1,1,0) & (0,0,0,1,0) & (-1,-1,-1,-1,-1) \\ (0,0,-1,0,0) & (0,1,1,0,0) & (0,0,1,0,-1) & (1,-1,0,-1,1) & (1,1,-1,-1,-1) \end{bmatrix}$$

1.6 Comparison with NTRU and MATRU

We first explain the difference between PMTRU [Haj20b] and MaTRU theoretically. According to article [Cog05], MaTRU security against attacks based on lattice reduction depends on the widths of $(\alpha_i * \beta_j)_{0 \leqslant i,j \leqslant k-1}$,, and the ratio of the security level in MaTRU is proportional with the widths. However, for every i, j in $\{0, \ldots, k-1\}$, α_i and β_j contain $\left|\frac{n}{3}\right|$ zero elements, and the convolution product of α_i and β_j is found by the matrix product between α_i and the circular matrix of β_j, in which each column of the matrix can be expressed as a circular permutation of each of the other columns. Thus, each non-zero element in α_i or β_j at minimum $\left|\frac{n}{3}\right|$ will not affect times in the width of $\alpha_i * \beta_j$, as it conducts a decrease in the ratio c_h for MaTRU. On the other hand, the c_p of PMTRU depends on the width of ($f \circledast w', g \circledast w$), with no relationship between the rows of g or the columns of w; likewise, for the rows of f and the columns of w'.

In addition, to maximize c_p and the security level we can assign a large value to d_w and $d_{w'}$, choose w from the set : $\{w \in L_w / |g \circledast w| = \max\{|g \circledast M|\} \text{ such that } M \in L_w\}$, and choose w' from the set : $\{w' \in L_{w'} / |f \circledast w'| = \max\{|f \circledast M|\} \text{ such as } M \in L_{w'}\}$.

Similar to NTRU, PMTRU have a public key and ciphertexts with less elements than parameter

q; therefore, the memory space used to send the public key or the ciphertext is dependent on the value of q, thus it is important to decrease q.

The value of q that should be used in an instance of PMTRU to encrypt a message of size nk^2 with $d_w = d_{w'} = 1$ is equal to the value of q that should be used in an instance of NTRU to encrypt a message of size nk. This is an advantage, and to keep it requires that the value of d_w and $d_{w'}$ to $\frac{k}{2}$ do not increase. Indeed, each rise of d_w or $d_{w'}$ means increasing the value of q and performing a balancing, knowing that an addition of 1 to d_w will be interpreted by an addition of p to $|p.g \circledast w \circledast r + f \circledast w' \circledast m|_\infty$. Thus, it is preferable to increase the value of $d_{w'}$ rather than d_w, which is why we deliberately use $d_{w'} \approx \frac{k}{3}$ and $d_w \approx \frac{k}{3p}$.

We now compare the characteristics of the theoretical operation of PMTRU with that of NTRU and MaTRU, as indicated in ***Table 2***. The properties are listed in terms of the parameters $(n, p, q)$for NTRU, (n, k, p, q) for MaTRU and $(n, k, d_w, d_{w'}, , p, q)$ for PMTRU.

Characteristics	**NTRU[Hof98]**	**MATRU[Cog05]**	**PMTRU[Haj20b]**
Clear Text Size	$N\ log_2\ p$ bits	$nk^2\ log_2\ p$ bits	$nk^2\ log_2\ p$ bits
Encrypted Text Size	$N\ log_2\ q$ bits	$nk^2\ log_2\ q$ bits	$nk^2\ log_2\ q$ bits
Encryption Complexity	$O(N^2)$	$O\left(n^2k^{log_2{}^7}\right)$	$O\left(n^2k^{log_2{}^7}\right)$
Decryption Complexity	$O(N^2)$	$O\left(n^2k^{log_2{}^7}\right)$	$O\left(n^2k^{log_2{}^7}\right)$
Private Key Size	$N\ log_2\ p$ bits	$2nk\ log_2\ p$ bits	$nk^2\ log\ _2(4d_{w'}\ +3)$ bits
Public Key Size	$N\ log_2\ q$ bits	$3nk^2\ log_2\ q$ bits	$nk^2\ log_2\ q$ bits
Security of the Private Key	$\sqrt{\frac{N!}{(d_g!)^2(N-2d_g)!}}$	$\left(\frac{n!}{(d_f!)^2(n-2d_f)!}\right)^k$	$\left(\frac{n^{2d_w(k-1)}(k!)^2}{(k-2d_w)!\,(d_w!)^2}\right)\left(\sqrt{\frac{n!}{(d_f!)^2(n-2d_f)!}}\right)^k$
Security Message	$\sqrt{\frac{N!}{(d_g!)^2(N-2d_g)!}}$	$\left(\frac{n!}{(d_f!)^2(n-2d_f)!}\right)^k$	$\left(\sqrt{\frac{n!}{(d_r!)^2(n-2d_r)!}}\right)^k$
Lattice Security	$2\left(\frac{\pi^2 de^2}{3Nq^2}\right)^{\frac{1}{4}}$	$\frac{1}{3}\sqrt{\frac{2\pi e(2n+3)}{q}}$	$\sqrt{\frac{2\pi e(2(2d_{w'}+1)+2d_w)}{3q}}$

Table 2:Comparaison PMTRU, NTRU and MATRU.

As indicated in ***Table 2***, PMTRU and MaTRU have the same rate of encryption and decryption, and the rate of PMTRU is $k^{4-log_2{}^7}$ times faster than that of NTRU. The memory required to record the public key and ciphertext is less with PMTRU than with MaTRU and NTRU. In addition, the memory used to record the private key linked to the value of $d_{w'}$ is typically greater in PMTRU than in NTRU and MaTRU. Thus, the cost of message security is lower with PMTRU than with NTRU and MaTRU.

As shown in ***Table 2***, the security level of the private key of PMTRU is greater than that of

MaTRU for a high value $d_{w\prime}$. Similarly, the security level of PMTRU against a man-in-the-middle attack is higher than that of MaTRU. In the case of choosing $d_{w\prime}$ and d_w, such as $\frac{n}{6} < d_{w\prime} + d_w{}^2 + d_{w\prime}{}^2$, the lattice security with PMTRU is higher than with MaTRU, which is greater than with NTRU.

Chapter 2

Constraint-Based Privacy Preserving PCE

2.1 Introduction

Path Computation Element (PCE) is a dedicated real time application or router that is responsible for computing paths for Multiprotocol Label Switching (MPLS) Label Switched Path (LSP) from the head end, i.e., source node, to the tail end, i.e., destination node. PCE can use a suitable algorithm to compute paths subject to one or more constraints such as latency, bandwidth, and optimization criteria such as cumulative path cost. The scope of computed path can be intra-area, inter-area, or inter-AS (Autonomous System).

In a typical scenario, a Path Computation Client (PCC) sends a request to PCE with the necessary information for path computation. The request contains the source, destination addresses, metric, and additional constraints to be used in the path computation process. Path Computation Element Protocol (PCEP) is the de-facto IETF standard for PCC-PCE communication. Upon reception of a request, PCE computes the path using the information in its Topology Database to determine the optimal path. The resulting path is sent back to PCC which establishes the LSP [Dho17].

Figure 1 below represents a network containing a number of PCCs and a PCE where the communication between each PCC and the PCE takes place using PCEP.

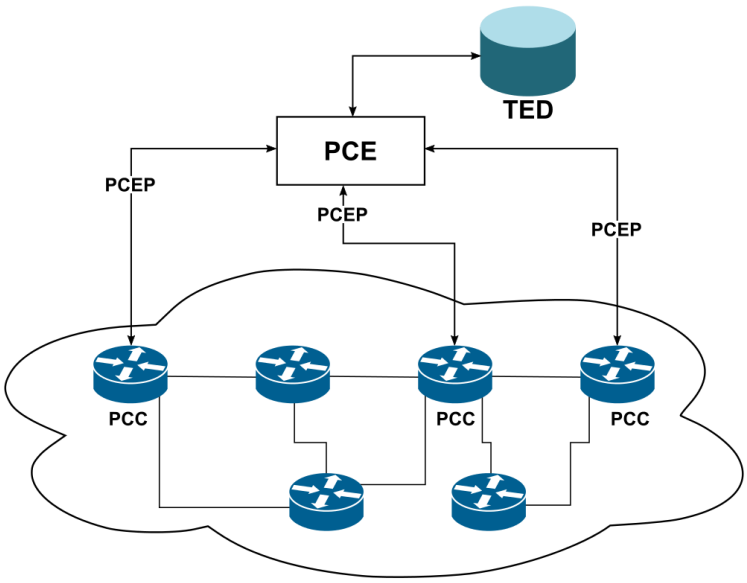

Figure 1:PCC-PCE communication

The current path computation model requires that PCE to have the knowledge about source and destination addresses/identities, metric type, and once computed, to have the knowledge of the path including the path hops (specified in the form of IP address, MPLS label, etc.), as well as the value of the cost associated with the computed path. There are scenarios in which trusted relationship between PCC and PCE does not exist, e.g., PCC and PCE belong to different service providers or administrative domains, in cloud computing services offering path computation as a service. Thus, there is a need for a new type of PCE which is capable of computing paths while preserving the privacy of PCCs, source and destination addresses/identities, as well as computed paths.

The main idea is to build a PCE capable of finding the k shortest paths verifying the required constraint (a minimal bandwidth in our case) from a source to a destination without having the knowledge of the clear text value of neither the source nor the destination. In such a model, PCC encrypts the source and destination of the path using a symmetric cryptosystem with its secret key (sk). PCC sends the resulting encrypted values to PCE. In our model, PCE applies a blind extraction technique that finds the optimal path from the source to the destination without

the need to decrypt the ciphertexts of the source and destination. The resulting path contains encrypted hops under the same secret key sk. PCE sends back the encrypted path to PCC which can use its secret key to decrypt the path hops.

The encryption scheme needed to build the oblivious PCE should have the property of being able to perform addition and multiplication operations on cipher-texts since both these operations are taken advantage of in our protocol calculations. Encryption schemes supporting both addition and multiplication over ciphertexts are referred to in literature as homomorphic encryption schemes. There are two types of homomorphic schemes: somewhat and fully homomorphic. The first type can perform limited number of operations on the ciphertexts, while the latter can perform arbitrary number of operations on ciphertexts, but it has the overhead of a computationally expensive bootstrapping technique needed to refresh the ciphertext.

Our model is more flexible regarding the used cryptosystem in the sense that it does not require a specific cryptosystem by itself. It rather requires the used cryptosystem to have the following properties:

- The set of plaintext messages is the ring Z_2, meaning the set $\{0,1\}$ equipped with addition modulo two and usual multiplication and the set of ciphertext is any ring $(R, +, *)$.
- The cryptosystem is fully homomorphic, or at least somewhat homomorphic but with parameters enabling the execution of the shortest path circuit while keeping the noise sufficiently small to guarantee decryption correctness. This property is known as Leveled homomorphic encryption.

The rest of this chapter is organized as the following:

In section 3.2 we define more precisely the used notion of homomorphic encryption. Next, in section 3.3 we present our model to solve the problem of oblivious PCE using homomorphic encryption. Then, in section 3.4 we present the system parameters in the case of using the

DGHV cryptosystem [Van10] along with the simulation results. In section 3.5 we describe related works that have addressed the problem of privacy preserving path computation and compare our results to the most recent of them. Finally, in section 3.6 we give a conclusion about the present work and some directions for possible future works on the subject.

2.2 Homomorphic Encryption

In this section we present Homomorphic Encryption in a formal way and give some of the first schemes to achieve the homomorphic property.

2.2.1 Homomorphic encryption definition

There is no consensus on a single formal definition of Homomorphic Encryption in the literature, as each author requires for an encryption scheme to have the properties, he/she judges appropriate in order to call it homomorphic. The only common idea between all definitions is that a homomorphic encryption scheme is a scheme that enables one to compute an encryption of the result of some operation on plaintext messages using only encryptions of these plaintext messages.

To formalize this definition let E be an encryption scheme. E is the encrypting procedure and D is the decrypting function.

Let M be the space of plaintext messages while C is the space of ciphertexts.

We say that E is homomorphic if there exist two functions $f: M^n \longmapsto M$and $g: C^n \longmapsto C$ such that for any vector of ciphertexts $(c_1, c_2, \dots, c_n)$ we have:

$$D\big(g(c_1, c_2, \dots, c_n)\big) = f\big(D(c_1), D(c_2), \dots, D(c_n)\big)$$

In other words, there exists a function f for which one can compute an encryption of its result using only encryptions of its arguments. The mean of computing the encryption of the result is here the function g.

In practice, the set M is usually the ring Z_2 which is the set $\{0,1\}$ equipped with its addition and multiplication $(\oplus, .)$ where addition is addition modulo two or XOR and multiplication is the usual integer multiplication or AND. The set C is a ring and its operations are $+$ and $*$ denoting respectively addition and multiplication. In this case we usually say that an encryption scheme defined from M to C is homomorphic if it permits to compute an encryption of the addition or the multiplication of two bits given only encryption of these bits.

$$D(c_1 + c_2) = D(c_1) \oplus D(c_2) \quad (1)$$

or

$$D(c_1 * c_2) = D(c_1).D(c_2) \quad (2)$$

In the first case we say that E is additively homomorphic while in the second case we say that E is multiplicatively homomorphic. If E is not both additively and multiplicatively homomorphic then we say that E is partially homomorphic.

The Partially homomorphic encryption property is not something rare in the literature; famous examples of such schemes are unpadded RSA [Riv78a] and El Gamal [Elg85] which are multiplicatively homomorphic, and Goldwasser-Micali [Gol82], Paillier [Pai99] and Benaloh [Ben94] which are additively homomorphic and can be transformed into a multiplicatively homomorphic scheme.

2.2.2 Fully Homomorphic encryption definition

While homomorphic encryption schemes permit one to perform at least some operations on encrypted data, fully homomorphic encryption schemes can be seen as schemes offering the ability to compute arbitrary functions on encrypted data. The very existence of such powerful yet secure schemes remained an open question for decades. This was due to the fact that the more malleable a scheme is the more likely it is insecure. Constructing the first fully homomorphic encryption scheme was then about constructing the first secure fully homomorphic scheme.

Let E be an encryption scheme. E is the encrypting procedure and D is the decrypting function.

Let M be the space of plaintext messages while C is the space of ciphertexts.

We say that E is fully homomorphic if for every function $f: M^n \longmapsto M$ there exists a function $g: C^n \longmapsto C$ such that for any vector of ciphertexts $(c_1, c_2, \dots, c_n)$ we have:

$$D\big(g(c_1, c_2, \dots, c_n)\big) = f\big(D(c_1), D(c_2), \dots, D(c_n)\big)$$

One can see that fully homomorphic encryption is much more powerful than homomorphic encryption in the sense that it enables one to compute an encryption of the result of any function applied to some plain text messages using only encryptions of these messages.

In the same manner, if we consider that M is the Z_2 ring and that $(C, +, *)$ is a ring then a sufficient condition of an encryption scheme E from M to C to be fully homomorphic is to be both additively and multiplicatively homomorphic.

The first scheme to achieve the remarkable property of being fully homomorphic has been brought by Craig Gentry in his Ph.D. Thesis in 2009 [Gen09]. Even though Gentry's scheme was by no mean practical, it was certainly a breakthrough since it eventually showed that secure

fully homomorphic encryption schemes do exist and opened the door to extensive research on the subject.

2.2.3 Somewhat Homomorphic encryption definition

In his Thesis of 2009 [Gen09], Craig Gentry introduced the concept of somewhat homomorphic encryption scheme. It is an encryption scheme that preserves addition and multiplication like in (1) and (2) but not for any ciphertexts c_1 and c_2. The reason behind this is that the encryption scheme when applied to a plaintext message creates a ciphertext with a noise associated to it. This noise grows if another operation is applied to the resulting ciphertext. Each operation that a ciphertext endures causes its noise to grow more until it reaches a threshold after which decrypting the resulting ciphertext does no longer give necessarily the correct plain text message. This means that the number of operations that can be performed on a message in order to maintain the correctness of the scheme is limited, hence the somewhat qualification.

2.3 Shortest Path Algorithms

The shortest path problem (SP) is the problem of finding one of the possible shortest paths from a node i to a node j in a certain graph. The most famous algorithm to resolve this problem is the Dijkstra's algorithm [Dij59]. The algorithm computes the shortest path from a certain source node to every other node in the graph. The result is a tree having the source node as its root and called the shortest path tree.

The k shortest paths problem (KSP) is a generalization of the SP problem as it addresses the case when one would like to compute not only the shortest path but the k shortest paths from a source node i to a destination node j when k is a positive integer.

Many versions of KSP have been considered in the literature [Fox73][Shi76][Fox78][Mar84][Mia91][Aze93][Aze94]. The best algorithm, in terms of the worst case complexity, to resolve this problem is the Eppstein's algorithm (EA) [Epp94][Epp98] which has a complexity in $O(M + N\ log\ N + k)$ where N is the number of nodes and M is the number of edges in the considered graph. Martins and Dos Santos have proposed another algorithm in [Mar00] called MSA. The authors claim that their algorithm has better practical results than EA. Another algorithm presented by Jiménez et Marzal in [Jim99] is a variation of EA called recursive enumeration algorithm (REA) and has showed, like MSA, better experimental results than EA.

In our case we have chosen to use REA as it is the fastest algorithm known till the moment of writing this paper.

2.4 Oblivious Path Computation Element

Taking advantage of the powerful tool of homomorphic encryption, we present in this section our solution to the problem of computing a path by the PCE, subject to a constraint represented in our case by a minimal bandwidth to be achieved, while preserving the privacy of the PCC.

2.4.1 Threat Model

The security threats against which our protocol is designed to be protected are described by the following conditions:

- The PCE has the complete topology of the network and keeps it private.
- The PCC has a source node address S a destination node address D and a required minimal bandwidth W and keeps all these parameters secret.

- The PCC asks the PCE for the shortest path between S and D without revealing the plain values of S and D to the PCE.
- The PCE can compute the shortest path between S and D using only encryptions of S and D

At the end of the protocol, the overall topology that the PCE has is kept private and the whole shortest path between S and D is not revealed to the PCE, including S,D and the cost associated with the path.

2.4.2 Notation

The symbols used in the rest of the paper are described in Table **3**.

Symbol	Description
i, j, t and k	Indexes
S	Source IP address
D	Destination IP address
N	Number of nodes
a_{ij}	Cost of going directly from node i to node j (can be infinite)
b_{ij}	Shortest path from node i to node j
w_{ij}	Bandwidth associated to b_{ij} (can be zero)
W	Minimal bandwidth required
m_{ij}	Number of nodes in b_{ij}

b_{ijtk}	k^{th} bit of the IP address of the t^{th} node in b_{ij}
s	index of the source node (unknown by the PCE)
s_i	i^{th} bit of the source IP address
d	index of the destination node (unknown by the PCE)
d_i	i^{th} bit of the destination IP address
x	A bit: zero or 1
E	Encryption procedure
$E(x)$	An encryption of the bit $x$under the private key p. Notice that the used cryptosystem is probabilistic and thus $E(x)$ is only one of the many possible encryptions of x. The private key $p$and the random parameter s which uniquely define one single encryption of$x$ have been omitted for the sake of notation clarity.
D	Decryption function
I_{ij}	Localizer of b_{ij}. Determines if b_{ij} is the requested shortest path.
$\oplus$	Addition modulo 2 or XOR
b_{ijk}	k^{th} bit in the shortest path between nodes i and j in the simplified notation
m	Number of bits in the longest possible shortest path
P_{ijk}	$D(P_{ijk}) = b_{ijk}$ if (i, j) is the matching pair, $D(P_{ijk}) = 0$ otherwise

P_k	$D(P_k) = b_{sdk}$
c_{ijk}	k^{th} bit of the cost of the path b_{ij}
C_{ijk}	$D(C_{ijk}) = c_{ijk}$ if (i,j) is the matching pair, $D\ (C_{ijk}) = 0$ otherwise
C_k	$D\ (C_k) = c_{sdk}$
Z_2	Ring {0,1} equiped with addition modulo two and multiplication
c	A ciphertext
r and q	Random integers used in the DGHV scheme.
p	Secret key (also called private key)
Q	Fixed random integer used to produce the reduction modulus
M	Reduction modulus: M=p.Q
Λ	Security parameter
$N(c)$	Noise associated with the ciphertext c

Table 3:List of symbols used in the rest of the paper

2.4.3 Protocol overview

Figure 2 below summarizes the interactions between the PCC and the PCE, and the processes performed by each one of them.

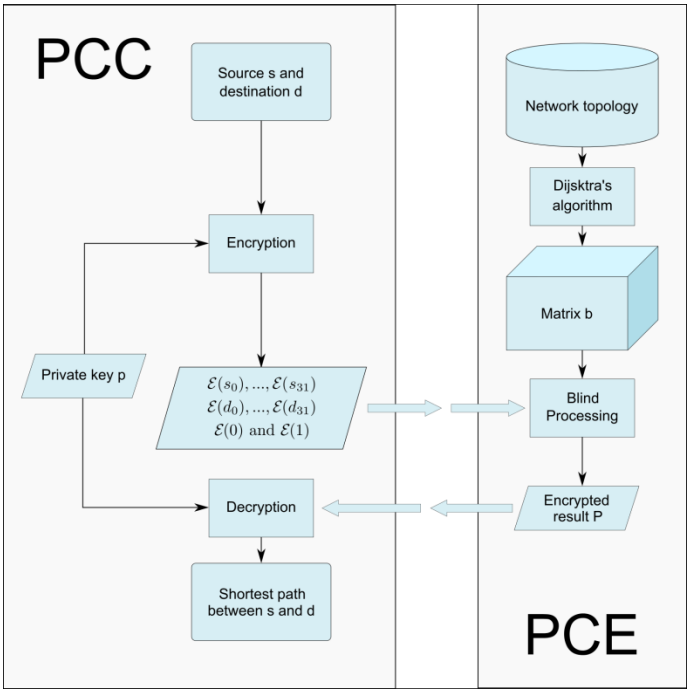

Figure 2:PCC and PCE interactions

Recall that the goal of our protocol is to enable a PCC, to obtain the shortest path with a bandwidth at least equal to W between two nodes with IP addresses S and D from a PCE without compromising the privacy of neither the PCC, nor the PCE. More specifically, the PCC sends the addresses S and D and the minimal bandwidth value Win an encrypted form to the PCE. But since the used cryptosystem is homomorphic, this enables the PCE to blindly perform calculations on S and D and obtain an encrypted shortest path between S and D that respects the bandwidth constraint. All of this without knowing what is S, D, W or the resulting shortest path between S and D. Upon completion, the resulting encrypted shortest path is sent back to the PCC who possesses the decryption key enabling it to get the shortest path in the clear form. Thus, the PCC has preserved the true values of S, $D$and W from being discovered by the PCE (or a third party adversary), but the PCE has also preserved its information about the network topology so that at any time t the PCE is the only one who has the detailed and up-to-date big picture of the network.

2.4.4 Data preparation

We consider a network constituted of N nodes. In our model the PCE has the complete topology of the network in the form of a square matrix $(a_{ij}) \in R^{N\times N}$ where $\forall\ 0 \le i,j \le N-1$, a_{ij} is the cost of going from node i to node j.

Having this information at its disposal, the PCE goes ahead and pre-computes the k shortest paths from each node i to each node j a long with their corresponding bandwidths using Algorithm 1 which takes advantage of the REA algorithm [Jim99].

```
1  Procedure: DP
   Input: (a_ij), N, d
   Output: (b_ij), (w_ij)
2  foreach i ∈ [[1, N]] do
3      foreach j ∈ [[1, N]] do
4          k=d;
5          while length(b_ij) < d do
6              tmp = REA((a_{i,j}), i, j, k);
7              l = (floor(k/d) − 1) * d;
8              while l < k and length(b_ij) < d do
9                  if length(w_ij) = 0 or length(w_{i,j,length(w_ij)}) ≤ bandwidth(tmp_l) then
10                     add(b_ij, tmp_l);
11                     add(w_ij, bandwidth(tmp_l));
12                 end
13                 l++;
14             end
15         end
16     end
17 end
```

Algorithm 1:Data preparation

The result of this data preparation is two matrixes $(b_{ij}) \in C[k]^{N\times N}$ and $(w_{ij}) \in N[k]^{N\times N}$.

In matrix (b_{ij}) each element b_{ij} is an array of the k shortest paths from node i to node j denoted by $b_{ij} = (b_{ij1}, b_{ij2}, \cdots, b_{ijk})$. In the same manner we have $w_{ij} = (w_{ij1}, w_{ij2}, \cdots, w_{ijk})$ where $\forall\ i,j \in [\![1,N]\!]\ and\ l \in [\![1,k]\!]\ w_{ijl}$ is the bandwidth corresponding to b_{ijl} . These two arrays are both in increasing order of bandwidth which means that $\forall\ \alpha,\beta \in [\![1,k]\!]\ \alpha < \beta \implies w_{ij\alpha} < w_{ij\beta}$

2.4.5 Integer Comparison

Since the PCE has to send, eventually, only the shortest path with a bandwidth that is greater than or equal to W, this means that the PCE needs a mean to blindly compare two integers. This means that we need a procedure $Compare$ that accepts as input two encrypted integers a and $b$and outputs two encrypted bits (zero or one) indicating $a < b$, $a > b$ or $a = b$

An ordinary comparison, between plaintext integers, is a recursive procedure presented in Figure 3. The comparison starts from the left to the right and supposes that both compared numbers have the same size (the same number of bits). For example, if we were working on 8 bits then the number 25 would be reprsented by 00011001.

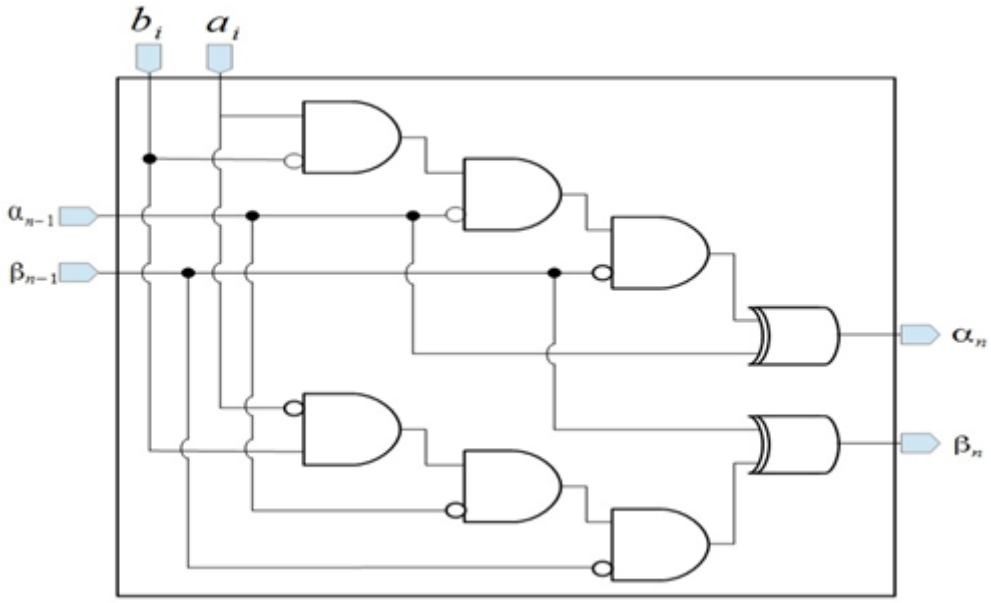

Figure 3:Circuit of the comparison between an integer a and an integer b

As shown in Figure 3 the comparison procedure is recursive where at each iteration the α_i and $\beta_i$depend on the previous values (α_{i-1} and β_{i-1})

The comparison formulas are given by the following equations:

$$\{\alpha_i = a_{i.}(b_i \oplus 1).(\alpha_{i-1} \oplus 1).(\beta_{i-1} \oplus 1) \oplus \alpha_{i-1} = a_i\overline{b_i}\overline{\alpha_{i-1}}\overline{\beta_{i-1}} \oplus \alpha_{i-1}$$

$$= b_{i.}(a_i \oplus 1).(\alpha_{i-1} \oplus 1).(\beta_{i-1} \oplus 1) \oplus \beta_{i-1}$$

$$= b_i\overline{a_i}\overline{\alpha_{i-1}}\overline{\beta_{i-1}} \oplus \beta_{i-1}$$

If the common size of a and b is n then $\alpha_n = \beta_n = 0$.

At any iteration if $\alpha_i = 1$ then $a > b$, otherwise if $\beta_i = 1$ then $a < b$. and finally if $\alpha_i = \beta_i = 0$ then all the bits seen so far are equal and the procedure must go further in order to determine which number is greater than the other or if the two numbers are equal.

The problem with this procedure, as remarked by Gahi et al. in [Gah12] is the fact that the noise quickly grows during the procedure due to fact that three multiplication operations are performed at each iteration. This problem has been resolved by the authors in [Gah12] by dividing each one of the two compared numbers into two parts (a high part and a low part) and comparing them separately using subtractions. As shown in Figure 4 from [Gah12].

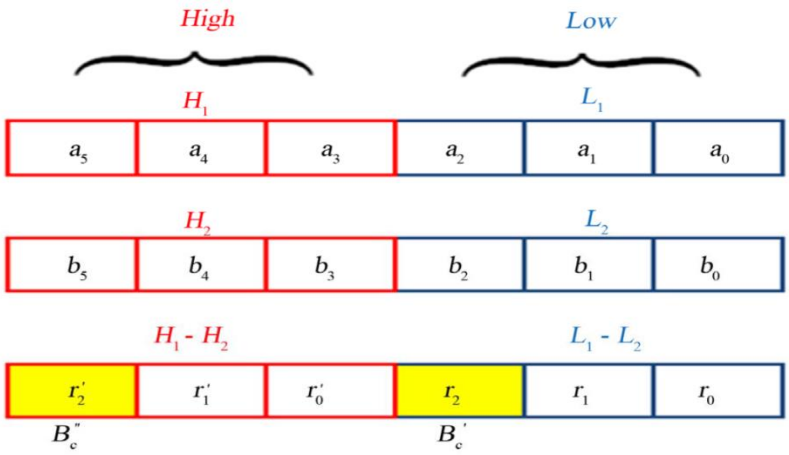

Figure 4:Comparison procedure proposed in [Gah12]

From the above figure the comparison formula is given by:

$$B_c = B_c'' + B_c' * \overline{B_c''}$$

Where B_c' and B_c'' are the comparison results of, respectively, the low part and the high part).

The overall comparison is deduced from the value of B_c as if $B_c = 1$ then $a \geq b$, otherwise $a < b$.

The noise produced by this technique is lower than in Equation (3). If we denote the noise function by N then the noise associated to B_c is given by the following equation that we will use later in the simulation section:

$$N(B_c) = N(B_c'') + N(B_c') * (1 + N(B_c'')) \quad (3)$$

In our case we need a procedure that takes two encrypted integers and outputs an encryption of 1 if and only if the first operand is greater than or equal to the second one. This procedure is presented in Algorithm 2:

1 **Procedure:** Compare
Input: Integer a, Integer b
Output: Integer c
2 $n = \frac{size(a)}{2} - 1;$
3 **return** $r'_n + r_n * (1 + r'_n);$

Algorithm 2:Comparing two integers

2.4.6 Request processing

We recall that $b_{i,j,0}$ is the first path in the array $b_{i,j}$. Let's suppose that each $b_{i,j,0}$is composed of m_{ij}IP address representing the corresponding nodes (where$2 \leq m_{ij} \leq N$)

Let's note:

$$b_{i,j,0,t,k} \mid 0 \leq i,j \leq N-1 \; and \; 0 \leq t \leq m_{ij} - 1 \; and \; 0 \leq k \leq 31$$

Where $b_{i,j,0,t,k}$ is the k^{th} bit of the IP address of the t^{th} node in $b_{i,j,0}$.

Now that the shortest paths matrix is computed, the PCE is ready to receive path computation requests from PCCs. Such a request comes in an encrypted form consisting of the encryption

of each one of the 32 bits ($s_i \mid 0 \leq i \leq 31$) of the source IP address and each one of the 32 bits ($d_i \mid 0 \leq i \leq 31$) of the destination IP address:

Encrypted source IP address:

$$E(s_i) \mid 0 \leq i \leq 31$$

Encrypted destination IP address:

$$E(d_i) \mid 0 \leq i \leq 31$$

The PCC also sends and encryption of the required minimal bandwidth W as the following:

$$E(w_i) \mid 0 \leq i \leq size(W)$$

Two additional ciphertexts are also sent: an encryption of zero $E(0)$ and an encryption of one $E(1)$. These two ciphertexts are used by the PCE to compute an encryption of any bit when needed.

Using the ciphertexts $E(0)$, $E(1)$, $E(s_i)$ and $E(d_i)$ where $0 \leq i \leq 31$, the PCE computes the localizer of each pair (i, j)as the following:

$$I_{ij} = \prod_{k=0}^{31} \left(E(1) + E(s_k) + E\left(b_{i,j,0,0,k}\right)\right) * \prod_{k=0}^{31} \left(E(1) + E(d_k) + E\left(b_{i,j,0,(m_{ij}-1),k}\right)\right) \quad (3)$$

In the above formula, the terms $E\left(b_{i,j,0,0,k}\right)$ and $E\left(b_{i,j,0,(m_{ij}-1),k}\right)$ are obtained thanks to the two encryptions $E(0)$ and $E(1)$ received from the PCC. With these two ciphertexts the PCE is able to compute an encryption of any bit in the $\left(b_{ij}\right)$ matrix.

The formula is further explained in Figure 5 below.

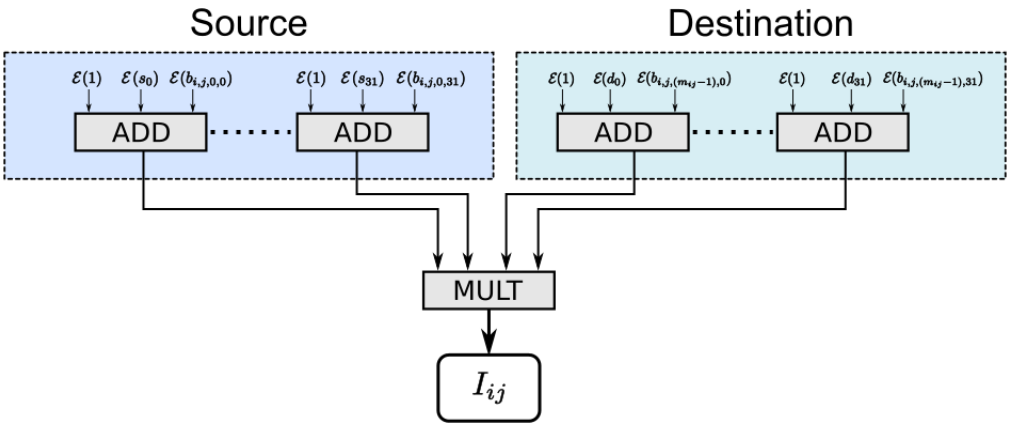

Figure 5:Calculation of the localizer I_{ij}

2.4.6.1 Proposition:

Let's note D the decryption function of the used cryptosystem.

$$\forall\ 0 \leq i, j \leq N-1$$

$D(I_{ij}) = 1 \Leftrightarrow i$ is the requested source and j is the requested destination

We call I_{ij} the localizer of the pair of nodes (i, j) because it is equal to an encryption of 1 if and only if i is the requested source and j is the requested destination. This means that only one localizer is equal to an encryption of one while all the others are encryptions of zero.

2.4.6.2 Proof

- ***Proof of the direct statement:***

Let's suppose that $D(I_{ij}) = 1$ for a certain pair (i, j) this means that

$$D\left(\prod_{k=0}^{31}\left(E(1) + E(s_k) + E\left(b_{i,j,0,0,k}\right)\right) * \prod_{k=0}^{31}\left(E(1) + E(d_k) + E\left(b_{i,j,0,(m_{ij}-1),k}\right)\right)\right) = 1$$

The fact that cryptosystem is multiplicatively homomorphic implies that:

$$\prod_{k=0}^{31} D\left(E(1)+E(s_k)+E\left(b_{i,j,0,0,k}\right)\right).\prod_{k=0}^{31} D\left(\left(E(1)+E(d_k)+E\left(b_{i,j,0,(m_{ij}-1),k}\right)\right)\right)=1$$

In the same manner, the fact that the cryptosystem is additively homomorphic implies that

$$\prod_{k=0}^{31}\left(D\left(E(1)\right)\oplus D\left(E(s_k)\right)\oplus D\left(E\left(b_{i,j,0,0,k}\right)\right)\right).\prod_{k=0}^{31}\left(D\left(E(1)\right)\oplus D\left(E(d_k)\right)\oplus D\left(E\left(b_{i,j,0,(m_{ij}-1),k}\right)\right)\right)$$

$$=1$$

Using the fact that the decryption of an encryption of an element is the element itself we have

$$\prod_{k=0}^{31}\left(1\oplus s_k\oplus b_{i,j,0,0,k}\right).\prod_{k=0}^{31}\left(1\oplus d_k\oplus b_{i,j,0,(m_{ij}-1),k}\right)=1$$

Which implies that

$$\{\prod_{k=0}^{31}\left(1\oplus s_k\oplus b_{i,j,0,0,k}\right)=1\ \prod_{k=0}^{31}\left(1\oplus d_k\oplus b_{i,j,0,(m_{ij}-1),k}\right)=1$$

This means that

$$\forall\ 0\le k\le 31\ \{1\oplus s_k\oplus b_{i,j,0,0,k}=1\ 1\oplus d_k\oplus b_{i,j,0,(m_{ij}-1),k}=1$$

Using $\oplus$ associative property and knowing that $\forall\ x\ \epsilon\ \{0,1\}\ 1\oplus x=1-x$ (usual subtraction because $1-0=1$ and $1-1=0$) we have:

$$\forall\ 0\le k\le 31\ \{1-\left(s_k\oplus b_{i,j,0,0,k}\right)=1\ 1-\left(d_k\oplus b_{i,j,0,(m_{ij}-1),k}\right)=1$$

Which means that

$$\forall\ 0\le k\le 31\ \{s_k\oplus b_{i,j,0,0,k}=0\ d_k\oplus b_{i,j,0,(m_{ij}-1),k}=0$$

Again, knowing that $\forall\ x,y\epsilon\{0,1\}\ x\oplus y=0\ \Leftrightarrow x=y$ we deduce that:

$$\forall\, 0 \leq k \leq 31\ \{s_k = b_{i,j,0,0,k}\ d_k = b_{i,j,0,(m_{ij}-1),k}$$

This means that i is the requested source and j is the requested destination because:

- $\forall\, 0 \leq k \leq 31$ the k^{th} bit of the **source** IP address is equal to the k^{th} bit of the first element (of index **zero**) of a path from i to j
- $\forall\, 0 \leq k \leq 31$ the k^{th} bit of the **destination** IP address is equal to the k^{th} bit of the last element (of index $(m_{ij} - 1)$**)** of a path from i to j)

Which ends the direct statement proof.

- ***Proof of the converse:***

Let's suppose that i is the requested source and that j is the requested destination. This can be written as:

$$\forall\, 0 \leq k \leq 31\ \{s_k = b_{i,j,0,0,k}\ d_k = b_{i,j,0,(m_{ij}-1),k}$$

Let's then compute $D(I_{ij})$:

$$D(I_{ij}) = D\left(\prod_{k=0}^{31}\left(E(1) + E(s_k) + E\left(b_{i,j,0,0,k}\right)\right)\right.$$
$$\left. * \prod_{k=0}^{31}\left(E(1) + E(d_k) + E\left(b_{i,j,0,(m_{ij}-1),k}\right)\right)\right)$$

This means that:

$$D(I_{ij}) = D\left(\prod_{k=0}^{31}\left(E(1) + E(s_k) + E(s_k)\right) * \prod_{k=0}^{31}\left(E(1) + E(d_k) + E(d_k)\right)\right)$$

Using the multiplicative property of the cryptosystem we deduce that:

$$D(I_{ij}) = \prod_{k=0}^{31} D\big(E(1) + E(s_k) + E(s_k)\big) . \prod_{k=0}^{31} D\big(E(1) + E(d_k) + E(d_k)\big)$$

Using the additive property of the cryptosystem we deduce that:

$$D(I_{ij}) = \prod_{k=0}^{31} \Big(D\big(E(1)\big) \oplus D\big(E(s_k)\big) \oplus D\big(E(s_k)\big)\Big) . \prod_{k=0}^{31} \Big(D\big(E(1)\big) \oplus D\big(E(d_k)\big) \oplus D\big(E(d_k)\big)\Big)$$

Which means that:

$$D(I_{ij}) = \prod_{k=0}^{31} (1 \oplus s_k \oplus s_k) . \prod_{k=0}^{31} (1 \oplus d_k \oplus d_k)$$

Using the associative property and the fact that $\forall\, x \,\epsilon\, \{0,1\}\; x \oplus x = 0$ we obtain that:

$$D(I_{ij}) = \prod_{k=0}^{31} 1 . \prod_{k=0}^{31} 1$$

Or simply:

$$D(I_{ij}) = 1$$

Which ends the proof.

Using the localizer I_{ij} we are able to target the correct source i and correct destination j but this gives us k possible paths to choose from. This is where the desired minimal bandwidth W comes into play.

For each pair of nodes (i, j) the bandwidth localizer J_{ij} is an array of k elements defined as the following:

$$J_{i,j,k} = Compare\left(C_{i,j,k}, E(W)\right)$$

This means that $D\left(J_{i,j,k}\right) = 1$ if and only if $C_{i,j,k} \geq W$

Finally, and in the same manner, we define the final localizer $L_{i,j}$ for each pair of nodes (i,j) as an array of d elements like the following:

$$L_{i,j,k} = \begin{cases} J_{i,j,0} * I_{i,j} & if \quad k = 0 \\ \prod_{l=0}^{k-1} \left(J_{i,j,l} + E(1)\right) * J_{i,j,k} * I_{i,j} & otherwise \end{cases}$$

2.4.6.3 Proposition

$b_{i,j,k}$ is the shortest path from the requested source to the requested destination such that $C_{i,j,k} \geq W$ if and only if $D(L_{i,j,k}) = 1$

2.4.6.4 Proof

- ***Proof of the direct statement:***

$B_{i,j,k}$ is the shortest path from the requested source i to the requested destination j such that $C_{i,j,k} \geq W$ then we have:

For $k = 0 D(I_{i,j}) = 1$ and $D\left(J_{i,j,0}\right) = 1$ which means that $D(I_{i,j}).D(J_{i,j,0}) = 1$

This means that $D(I_{i,j} * J_{i,j,0}) = 1$ or in other words that $D\left(L_{i,j,0}\right) = 1$ which had to be proven.

For $k \neq 0$ we have $\forall\ l < k C_{i,j,l} < W$ and $C_{i,j,k} \geq W$ and $D\left(I_{i,j}\right) = 1$.

This means that $\forall\ l < k: D(J_{i,j,l}) = 0$ and $D\left(J_{i,j,k}\right) = 1$

Which means that $\forall\ 0 \leq l < k, D(J_{i,j,l}) \oplus D\left(E(1)\right) = 1$

Which can be written as:

$$\prod_{l=0}^{k-1} D\left(J_{i,j,l} + E(1)\right) = 1$$

If we add that $D(J_{i,j,k}) = 1$ and that $D(I_{i,j}) = 1$ we obtain:

$$\prod_{l=0}^{k-1} D\left(J_{i,j,l} + E(1)\right) * D(J_{i,j,k}) * D(I_{i,j}) = 1$$

This means that:

$$D\left(\prod_{l=0}^{k-1} \left(J_{i,j,l} + E(1)\right) * J_{i,j,k} * I_{i,j}\right) = 1$$

Or in other words:

$$D(L_{i,j,k}) = 1$$

This ends the proof of the direct statement.

- ***Proof of the converse:***

If we have $D(L_{i,j,k}) = 1$

For $k = 0$ we have then $D(L_{i,j,0}) = 1$ which means that $D(J_{i,j,0} * I_{i,j}) = 1$

This means that $D(J_{i,j,0}) = D(I_{i,j}) = 1$ which is equivalent to the fact that i is the requested source and j is the requested destination and $C_{i,j,0} \geq W$. This means that $b_{i,j,0}$ is a shortest path from the requested source to the requested destination satisfying the bandwidth constraint.

For $k \neq 1$ we have:

$$D\left(\prod_{l=0}^{k-1} \left(J_{i,j,l} + E(1)\right) * J_{i,j,k} * I_{i,j}\right) = 1$$

This is equivalent to:

$$\prod_{l=0}^{k-1} D\left(J_{i,j,l} + E(1)\right) * D\left(J_{i,j,k}\right) * D\left(I_{i,j}\right) = 1$$

This means that $D\left(J_{i,j,k}\right) = D\left(I_{i,j}\right) = 1$ and that $\forall 0 \leq l < k\ D\left(J_{i,j,l} + E(1)\right) = 1$

Thus, we have: $D\left(J_{i,j,k}\right) = D\left(I_{i,j}\right) = 1$ and $\forall 0 \leq l < k,\ D\left(J_{i,j,l}\right) = 0$

Which leads to $C_{i,j,k} \geq W$, i is the requested source, j is the requested destination and $\forall 0 \leq l < k, C_{i,j,l} < W$.

This ends the proof.

To simplify the next calculation, we consider that all paths $b_{i,j,k}$ are simply a sequence of bits, so that instead of writing:

$$b_{i,j,l,t,k} \mid 0 \leq i,j < N\ and\ 0 \leq l < d\ and\ 0 \leq t \leq m_{ij} - 1\ and\ 0 \leq k \leq 31$$

We use the notation:

$$b_{i,j,l,k} \mid 0 \leq i,j < N\ and\ 0 \leq k \leq 31 * (m_{i,j,l} - 1)$$

Let m be the number of bits in the longest path $b_{i,j,l}$ and let's consider that all paths have the same length by adding zeros to the end of every path that contains less than m bits. We can now describe $b_{i,j,l}$ like the following:

$$b_{i,j,l,k} \mid 0 \leq i,j < N\ and\ 0 \leq l < d\ \ and\ 0 \leq k \leq m - 1$$

Now for every triplet (i, j, l) the PCE computes $P_{i,j,l,k}$ as the following:

$$P_{i,j,l,k} = L_{i,j,l} * E(b_{i,j,l,k})$$

If i and j are not the requested source and destination, or if the bandwidth constraint is not satisfied then there exists an encryption of zero$E(0)$such that:

$$D(P_{i,j,l,k}) = D\left(E(0) * E(b_{i,j,l,k})\right)$$

This means that:

$$D(P_{i,j,l,k}) = D(E(0)).D\left(E(b_{i,j,l,k})\right)$$

$$D(P_{i,j,l,k}) = 0.b_{i,j,l,k}$$

$$D(P_{i,j,l,k}) = 0$$

But if i and j are the requested source and destination and the bandwidth constraint is satisfied then there exists an encryption of one $E(1)$ such that:

$$D(P_{i,j,l,k}) = D\left(E(1) * E(b_{i,j,l,k})\right)$$

This means that:

$$D(P_{i,j,l,k}) = D(E(1)).D\left(E(b_{i,j,l,k})\right)$$

$$D(P_{i,j,l,k}) = 1.b_{i,j,l,k}$$

$$D(P_{i,j,l,k}) = b_{i,j,l,k}$$

We deduce that $P_{i,j,l,k}$ decrypts to the k^{th} bit of the path $b_{i,j,l}$ when (i, j) is the matching pair and $b_{i,j,l}$ satisfies the bandwidth constraint, and decrypts to zero otherwise. This means that, for

certain (l, k) there is exactly one $P_{i,j,l,k}$ that decrypts to the k^{th} bit of the matching path. All the others are encryptions of zero.

Now the PCE computes the following sum P_k for every $0 \leq k \leq m-1$

$$P_k = \sum_{0 \leq i,j \leq N-1, 0 \leq l < d} P_{i,j,l,k}$$

As shown just earlier, all the operands of this sum are encryptions of zero except one that is an encryption of the k^{th} bit of the shortest matching path satisfying the constraint. Let (s, d) be the pair of indexes of the matching source and destination. And let e be the index of the first path in $b_{i,j}$ that satisfies the bandwidth constraint.

We have then:

$$D(P_k) = D\left(\sum_{0 \leq i,j \leq N-1, 0 \leq l < d} P_{i,j,l,k} \right)$$

$$D(P_k) = \sum_{0 \leq i,j \leq N-1, 0 \leq l < d} D\left(P_{i,j,l,k}\right)$$

$$D(P_k) = D\left(P_{s,d,e,k}\right) + \sum_{0 \leq i,j \leq N-1, 0 \leq l < d \ (i,j,l) \neq (s,d,e)} D\left(P_{i,j,l,k}\right)$$

$$D(P_k) = b_{s,d,e,k} + \sum_{0 \leq i,j \leq N-1, 0 \leq l < d \ (i,j,l) \neq (s,d,e)} 0$$

$$D(P_k) = b_{s,d,e,k}$$

This proves that the sum computed by the PCE decrypts to nothing but the k^{th} bit of the requested path. Notice how the PCE was able to compute an encryption of this bit without knowing the cleartext values of the requested source or destination.

The PCE sends all the P_k $(0 \leq k \leq m-1)$ to the PCC for decryption. Note that the obtained path may contain in its end a series of IP addresses of the form 0.0.0.0. These recognizable IP addresses are simply eliminated by the PCC.

In the above calculations we assumed the existence of a matching pair $(s, d)$and the existence of a bandwidth $C_{s,t,e} \geq W$ but it is easy to see that if there is no match then the resulting path contains only zeros. So, whenever the PCC obtains a path that only contains zeros it deduces that there is no route from the requested source to the requested destination satisfying the requested bandwidth constraint.

Using the same idea of P_k the PCE can compute an encryption of the k^{th} bit of the cost of the matching path like the following:

$$K_{i,j,l,k} = I_{ij} * E\left(c_{i,j,l,k}\right)$$

Where $c_{i,j,l,k}$ is the k^{th} bit of the cost of the path $b_{i,j,l}$.

Therefore,

$$K_k = \sum_{0 \leq i,j \leq N-1, 0 \leq l < d} K_{i,j,l,k}$$

Where K_k is an encryption of the k^{th} bit of the cost of the shortest matching path satisfying the bandwidth constraint.

2.5 Simulation

In this section we give the results of the implementation of our protocol using one of the available homomorphic schemes.

The scheme we have chosen to implement our protocol is DGHV [Van10] which has the merit of being one of the conceptually simplest homomorphic cryptosystems in the literature. Furthermore, in our case we only need a further simplified symmetric somewhat homomorphic version of the scheme presented as the following:

- The ring of plaintexts is Z_2 equipped with addition modulo two $(\oplus)$ and the usual multiplication $(.)$
- The ring of ciphertexts is the ring$(Z_M, +, *)$ equipped with addition and multiplication modulo$M$, where M is a big exact multiple of the secret key as described below.
- The encryption process E gives an encryption to a bit $b \in \{0,1\}$ as the following:

$$E(x) = x + 2r + pq$$

In the above formula, r and q are random integers while p is a prime number used as the secret key.

- All additions and multiplications on ciphertexts are done modulo the integer M such that $M = p.Q$ where Q is a random prime number and p is the secret key.

According to the DGHV[1] scheme in [Van10], these three integers must be under the following security constraints:

[1]We added two constraints stating that p and Q are prime numbers so that the factorization of M=p.Q becomes intractable like in an RSA modulus [Gol82]. Notice that for any integer number N≥2 the interval ⟦N,2N⟧ is guaranteed to contain at least one prime number as proven by Tchebichef in [Jim99]. Furthermore, the Prime Number Theorem, proven independently by Hadamard [Fox78] and de la Valée-

$$0 \leq r < 2^{\lambda} \quad (4)$$

p is a prime number and $2^{\lambda^3-1} \leq p < 2^{\lambda^3}$ (5)

$$0 \leq q < 2^{\lambda^7-\lambda^3} \quad (6)$$

Q is a prime number and $2^{\lambda^7-\lambda^3-1} \leq Q < 2^{\lambda^7-\lambda^3}$ (7)

Where λ is an integer parameter indicating the security of the scheme: the highest λ is, the more secure is the cryptosystem, and the more impractical too because of the calculations on huge integers.

With these parameters, the complexity of the scheme in the number of nodes n and the security parameter λ is $O(N^3\lambda^7\ log\ log\ (\lambda)\ log\ log\ (log\ log\ (\lambda)))$ when the Schönhage-Strassen multiplication algorithm is used.

Notice that constraints (6) and (7) are a consequence of the choice of λ^3 as the bit length of the secret key $p$in (5). The motivation behind this choice is to be able to perform all the necessary operations without making any ciphertext noise exceed the threshold represented by the secret key$p$. In our case the recommended λ^2 secret key bit length cannot be used unless λ is unreasonably large.

- The decryption function D is as follows for an integer ciphertext c:

$$D(c) = (c\ mod\ p) mod\ 2$$

Poussin [Kus97], implies that when N tends to infinity then the number of prime numbers between N and 2N becomes approximately N/log(N).

The quantity $x + 2r$ is called the noise associated to the ciphertext $c = x + 2r + pq$. We note the noise of a ciphertext c in the rest of the document as $N(c)$. This quantity is subject to the following correctness constraint:

$$N(c) = x + 2r < p$$

This means that if the noise constraint is not satisfied, then there is no guarantee that the decryption of a ciphertext $x + 2r + pq$ will result in the plaintext x.

As described above, the DGHV scheme can be used in our protocol only if the security parameter λ is set so that in all the calculations in our protocol, no noise exceeds the threshold consisting in the secret key p. But since throughout any addition or multiplication the noise can only increase, then an equivalent condition to the noises not exceeding p is for our last result P_k to have a noise lower than p for all values of k ($0 \leq k \leq m - 1$)

$$\forall\, k \in [\![0, m-1]\!]\; N(P_k) < p$$

Where N means "noise". This means that

$$\forall\, k \in [\![0, m-1]\!]\; N\left(\sum_{0 \leq i,j \leq N-1, 0 \leq l < d} P_{i,j,l,k}\right) < 2^{\lambda^3 - 1}$$

We have seen earlier in this paper that in the context of DGHV the noise of a sum (respectively a product) of two ciphertexts is the sum (respectively the product) of the noises of the two ciphertexts. This is true for the sum or the product of any number of ciphertexts.

Thus, we have:

$$\forall\, k \in [\![0, m-1]\!] \sum_{0 \leq i,j \leq N-1, 0 \leq l < d} N\left(P_{i,j,l,k}\right) < 2^{\lambda^3 - 1}$$

Enough condition is then:

$$\sum_{0\leq i,j\leq N-1,0\leq l<d} Max_{(0\leq i,j<N,0\leq l<d\ and\ 0\leq k<m)}\left(N\left(P_{i,j,l,k}\right)\right) < 2^{\lambda^3-1}$$

This means that:

$$d * N^2 * Max_{(0\leq i,j<N,0\leq l<d\ and\ 0\leq k<m)}\left(N\left(P_{i,j,l,k}\right)\right) < 2^{\lambda^3-1}$$

Or:

$$d * N^2 * Max_{(0\leq i,j\leq N-1,0\leq l<d\ and\ 0\leq k\leq m-1)}\left(N\left(L_{i,j,l} * E\left(b_{i,j,l,k}\right)\right)\right) < 2^{\lambda^3-1}$$

In the context of DGHV we can write that $E\left(b_{i,j,l,k}\right) = b_{i,j,l,k} + 2r + pq$ for a certain pair of parameters (r, q).Now we have:

$$d * N^2 * Max_{(0\leq i,j<N,0\leq l<d\ and\ 0\leq k<m)}\left(N\left(L_{i,j,l}\right) * \left(b_{i,j,l,k} + 2r\right)\right) < 2^{\lambda^3-1}$$

Always in the context of the DGHV construction, the ciphertext $E\left(b_{ijk}\right) = b_{ijk} + 2r + pq$ can be sent to the PCC with parameters $r = 0$ and $q = 0$ without compromising the security of the cryptosystem. This is because 0 and 1 can be used as valid encryptions of, respectively, 0 and 1 whatever the private key p is.

Thus, we can write:

$$d * N^2 * Max_{(0\leq i,j<N,0\leq l<d\ and\ 0\leq k<m)}\left(N\left(L_{i,j,l}\right) * b_{i,j,l,k}\right) < 2^{\lambda^3-1}$$

Let u and v integers from $[\![0, N-1]\!]$, y ad integer from $[\![0, d-1]\!]$and z and integer from $[\![0, m-1]\!]$so that:

$$N(L_{u,v,y}) * b_{u,v,y,z} = Max_{(0\leq i,j<N, 0\leq l<d \text{ and } 0\leq k<m)}(N(L_{i,j,l}) * b_{i,j,l,k})$$

We have then:

$$d * N^2 * N(L_{u,v,y}) * b_{u,v,y,z} < 2^{\lambda^3-1}$$

If $b_{u,v,y,z} = 0$then the inequation is already satisfied. Now let's address the other case where $b_{u,v,y,z} = 1$

After developing $N(L_{u,v,y})$ we obtain:

$$d * N^2 * \prod_{k=0}^{y-1}(N(J_{u,v,k}) + 1) * N(J_{u,v,y}) * N(I_{u,v}) < 2^{\lambda^3-1}$$

This means that:

$$d * N^2 * \prod_{k=0}^{y-1}\left(N\left(Compare(C_{u,v,k}, W)\right) + 1\right) * N\left(Compare(C_{u,v,y}, W)\right) * N(I_{u,v}) < 2^{\lambda^3-1}$$

And since we have:

$N(Compare(C_{i,j,k}, W)) = N(B_c) = N(B_c'') + N(B_c') * (1 + N(B_c''))$,

And knowing that:

$N(B_c') \leq 2 * \frac{lnln\,(w_{max})}{lnln\,(2)} * (2^{\lambda} - 1) + 1$ and $N(B_c'') \leq 2 * \frac{lnln\,(wb_{max})}{lnln\,(2)} * (2^{\lambda} - 1) + 1$

Where w_{max} is the maximal value of bandwidths among all the values in the (w_{ij}) matrix as well as the requested bandwidth W.

Thus, we can write that:

$$N\left(Compare\left(C_{i,j,k},b\right)\right) < 4 * \left(\frac{ln\ (w_{max})}{ln\ ln\ (2)}\right)^2 * \left(2^{\lambda}-1\right)^2 + 8 * \left(\frac{ln\ (w_{max})}{ln\ ln\ (2)}\right) * (2^{\lambda}-1) + 3$$

If we take, $w_{max} = 256$ then $\frac{ln\ (w_{max})}{lnln\ (2)} = 8$ this gives us:

$$N\left(Compare\left(C_{i,j,k},W\right)\right) < 256 * \left(2^{\lambda}-1\right)^2 + 64 * \left(2^{\lambda}-1\right) + 3$$

Or:

$$N\left(Compare\left(C_{i,j,k},b\right)\right) < 2^{2\lambda+8} + 9 * 2^{\lambda+6} + 195$$

Now we can write that:

$$d * y * N^2 * \left(2^{4\lambda+16} - 7 * 2^{3\lambda+15} + 1135 * 2^{2\lambda+8} - 2457 * 2^{\lambda+6} + 175 * 176\right) * N\left(I_{u,v}\right) < 2^{\lambda^3-1}$$

And since the maximal value of y is d we obtain:

$$d^2 * N^2 * \left(2^{4\lambda+16} - 7 * 2^{3\lambda+15} + 1135 * 2^{2\lambda+8} - 2457 * 2^{\lambda+6} + 175 * 176\right) * N\left(I_{u,v}\right) < 2^{\lambda^3-1}$$

Developing $N\left(I_{u,v}\right)$ we have:

$$N(I_{u,v}) = N\left(\prod_{k=0}^{31}\left(E(1) + s_k + E\left(B_{u,v,0,0,k}\right)\right) * \prod_{k=0}^{31}\left(E(1) + d_k + E\left(B_{u,v,0,m_{i,j}-1,k}\right)\right)\right)$$

$$N(I_{u,v}) = \prod_{k=0}^{31}\left(N\left(E(1)\right) + N(s_k) + N\left(E\left(B_{u,v,0,0,k}\right)\right)\right) * \prod_{k=0}^{31}\left(N\left(E(1)\right) + N(d_k) + N\left(E\left(B_{u,v,0,m_{i,j}-1,k}\right)\right)\right)$$

And if we take $E(m) = m$ for every bit then we have:

$$N(I_{u,v}) = \prod_{k=0}^{31}(N(1) + N(s_k) + N(B_{u,v,0,0,k})) * \prod_{k=0}^{31}(N(1) + N(d_k) + N(B_{u,v,0,m_{i,j}-1,k}))$$

Notice that $E(s_k) = s_k + 2r_1 + pq_1$ and that $E(d_k) = d_k + 2r_2 + pq_2$ for two given pairs (r_1, q_1) and (r_2, q_2) that are unknown to the PCE. This leads us to the following equation:

$$N(I_{u,v}) = \prod_{k=0}^{31}(1 + s_k + 2r_1 + B_{u,v,0,0,k}) * \prod_{k=0}^{31}(1 + d_k + 2r_2 + B_{u,v,0,m_{i,j}-1,k})$$

In the maximal case we have $s_k = d_k = B_{u,v,0,0,k} = B_{u,v,0,m_{i,j}-1,k} = 1$ and $r_1 = r_2 = 2^{\lambda}$

With these values $N(I_{u,v})$ becomes:

$$N(I_{u,v}) = (3 + 2^{\lambda+1})^{64}$$

Our sufficient condition becomes then:

$$d^2 * N^2 * (2^{4\lambda+16} - 7 * 2^{3\lambda+15} + 1135 * 2^{2\lambda+8} - 2457 * 2^{\lambda+6} + 175 * 176) * (3 + 2^{\lambda+1})^{64} < 2^{\lambda^3-1}$$

$$N < \frac{2^{\frac{\lambda^3-1}{2}}}{d * (3 + 2^{\lambda+1})^{32}} * \frac{1}{\sqrt{(2^{4\lambda+16} - 7 * 2^{3\lambda+15} + 1135 * 2^{2\lambda+8} - 2457 * 2^{\lambda+6} + 175 * 176)}}$$

Table 4 below shows that the first value of λ that permits a number of nodes $N > 0$ is 9

d λ	5	10	15	20
9	53561	25249	16529	12287
10	$2.0* 10^{35}$	$9.4* 10^{34}$	$6.1* 10^{34}$	$4.6* 10^{34}$
11	$7.9* 10^{74}$	$3.7* 10^{74}$	$2.4* 10^{74}$	$1.8* 10^{74}$
12	$2.6* 10^{124}$	$1.2* 10^{124}$	$8.1* 10^{123}$	$6.0* 10^{123}$

Table 4:Maximal number of nodes depending on λ

In our simulation we used the following choice of parameters:

Integer Parameter	Description	Value or range
λ	Security parameter. The highest is λ the more secure is the scheme, but at the cost of slower operations.	9
w_{max}	Maximal bandwidth that can be requested	256

r	Random noise parameter used to prevent an attacker from computing the GCD of two ciphertexts.	$[\![0, 2^{\lambda}[\![$
p	Prime number used as the secret key.	$[\![2^{\lambda^3-1}, 2^{\lambda^3}[\![$
q	Random parameter used to get a multiple of the secret key before adding the noise.	$[\![0, 2^{\lambda^7-\lambda^3}[\![$
Q	Random prime number used to compute an exact multiple of the secret key used as a modulus.	$[\![2^{\lambda^7-\lambda^3-1}, 2^{\lambda^7-\lambda^3}[\![$
M	Modulus. Used to reduce the size of the ciphertext resulting from one or more operations.	$p.Q$

Table 5:Simulation parameters choice

With these parameters the complexity of the protocol is O(n^3d^2).

In our simulation using Intel® Core™ i5 2.40 GHz laptop we developed a Java ™ server and a client communicating through sockets. We computed the time needed by the server to compute a path satisfying a bandwidth constraint for a certain number of nodes $N$and number of paths d.

The obtained values are presented in Table 6.

d N	5	10	15	20
20	1.7	2.1	2.5	2.9

40	7.5	9.3	11.3	13.1
80	33.6	44.6	56.3	67.4
160	159.8	231.5	311.2	492.3

Table 6:PCE processing time depending on N and d in seconds

Figure 6 plots the obtained values, which shows how quickly the processing time increases each time the number of nodes is doubled.

number of nodes is doubled.

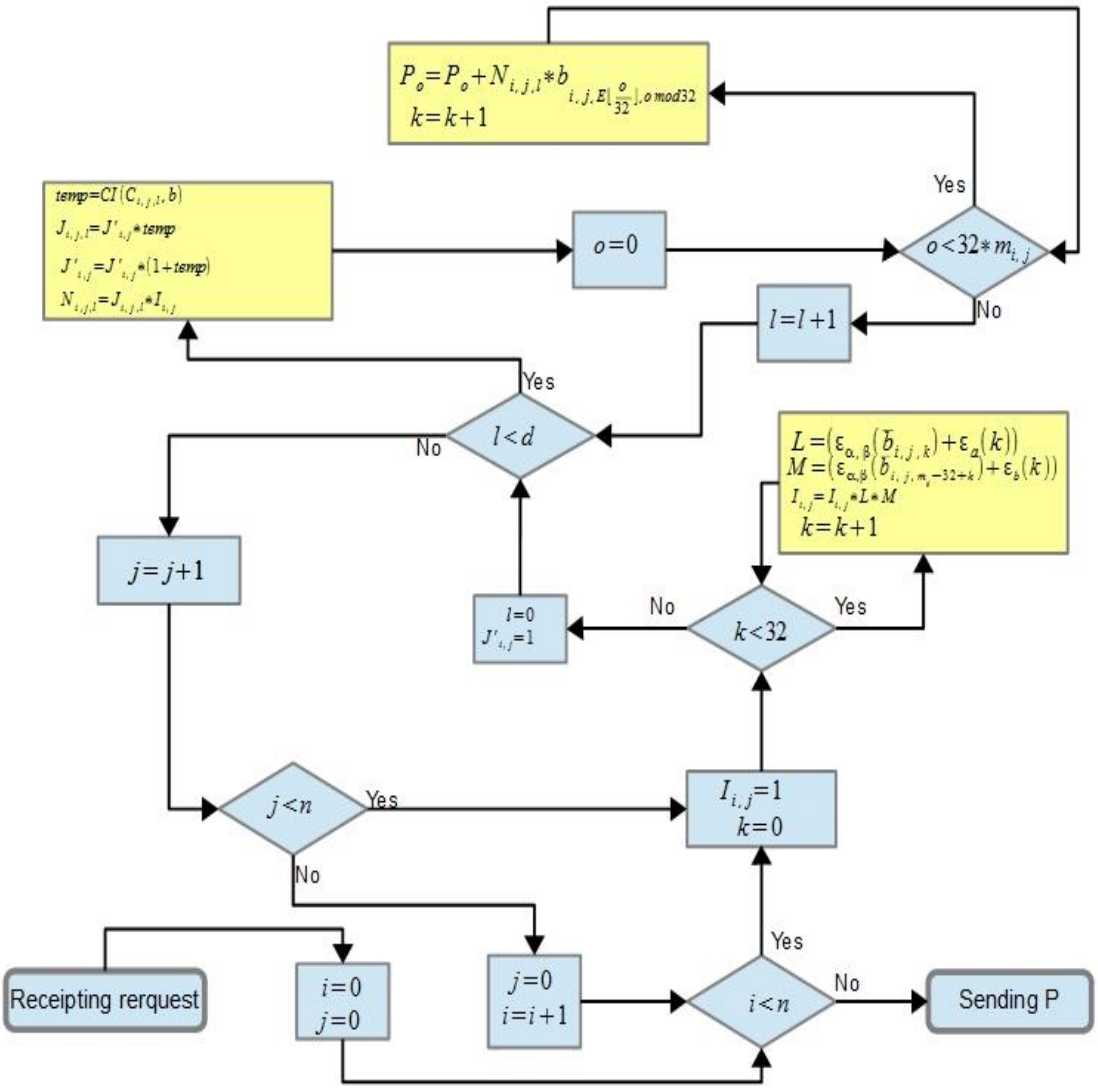

Figure 6:Protocol implementation

2.6 Related Work

After that we presented our protocol, we present in this section, some of the works that tackled directly or indirectly the problem of computing a shortest path while preserving the privacy of the client. At the end of the section, we compare the most recent of these works to our contribution.

One of these works was in 2008 by Das et al [Du01] at a geometrical level. The chapter addresses the problem of a client A desiring to compute the shortest path between two points S and D on the Cartesian plane while there are some obstacles, modeled as convex polygons, that the client A does not know about. Therefore, the client A needs the server B who has the exhaustive list of these obstacles to compute the shortest path between S and D for him. The privacy constraint imposed by the authors is that at the end of the protocol the client A does not know about any obstacle other than the obstacles that may occur on the resulting shortest path and the server B must not get any information about the resulting shortest path including, and especially, the source S and the destination D. To be able to do the different oblivious calculations between the two parties A and B, the main used technique by the authors is the privacy-preserving scalar product protocol described in [Ata01] and [Du01]. Briefly, this technique permits to Alice who has a secret vector X to compute $u = X.Y + v$ where Y is Bob's secret vector and v is Bob's secret number.

Another work by Gahi et al. in [Gah12] used homomorphic encryption to provide a service to a user based on his location. In a nutshell, the protocol consists of a client who sends its geographical location along with a search criterion (for example School or Restaurant) in encrypted form to the server. The server then blindly processes its database to find the nearest, service to the received location according to the Manhattan distance [Cra10]. The result is then sent back to the client for decryption.

The protocols presented by Brickell and Shmatikov in [Bri05] enables two parties in possession of the same graph but with different weights to compute some graph algorithms like the shortest path distances between all pairs of nodes if the common graph had on each one of its edges the minimum of the two weights from the two parties.

One of the latest works on the subject was in 2016 by Wu et al. In this chapter the authors addressed the problem of computing the next hop in the shortest path between a source and a destination in the context of a city map while keeping the source and destination unknown to the server, and the importance of each road (the estimated time to traverse it depending on traffic conditions) unknown to the client. The protocol presented by the authors is as the following:

- The city map is assumed to be aligned with the cardinal directions (North, East, South and West) and is modeled as a directed graph where nodes are the crossroads and streets are the edges.
- The graph is transformed by adding some fictive nodes so that, at the end, each node has at most 4 neighbors. Each one of these neighboring nodes has an associated direction N, E, S or W encoded on two bits (giving respectively 00,01,10 and 11)
- Dijkstra's algorithm [Dij59] is used to compute the shortest path for all pairs of nodes.
- Two matrices M^{NE} and M^{NW} from $\{0,1\}^{n\times n}$ are then defined so that for each pair of nodes (i,j) the bits $M_{ij}^{NE}$and M_{ij}^{NW} encode the direction to take to get to the next node in the shortest path between i and j.
- The authors noticed that each one of the two matrices M^{NE} and M^{NW} is a matrix M from $\{0,1\}^{n\times n}$ who often has a compressible structure. The authors compressed such a matrix M by first transforming all its zeros to -1s and then finding two matrices A and B from $Z^{n\times d}$ such that $sign(AB^T) = M$ and $d < n$ where the $sign$ function is a function that transforms the elements of a matrix to 1s and -1s depending on their signs. One can see that with this compression, instead of having one matrix containing n^2 bits, the two matrices A and B only contain a total of $2dn$ integers. The authors found empirically that in the case of the four cities they studied, they can achieve a compression ratio between 7.63 and 11.23 depending on the number of nodes in each city.

- The computation of the next direction to be taken by the client in order to proceed further on the shortest path is reduced to computing the sign of inner products between two rows A_s and B_t of matrices A and B such that the server does not know the value of the source and destination indexes s and t, and the client does not know the exact value of the inner product. This is done by the authors using Yao's garbled circuits [Yao86], garbled arithmetic circuits [App14], private information retrieval (PIR), which is a protocol requiring a partially homomorphic encryption scheme like Paillier [Pai99] and oblivious transfer using the One-Sided Simulation described in [Haz10].

Compared to our protocol, the work of Wu et al. [Wu16] has the advantage of reducing the needed bandwidth for communication between the client and the server thanks to the city map compression technique, but our protocol has the advantage of protecting the network topology of the server, meaning that the PCC does not have the big picture of the network. Our protocol has also the merit of being easily generalizable to other formats of addresses like IPv6 addresses or simply indexes of nodes, while the work in [Wu16] is tailored to roads on cities maps. Another difference between the two protocols is that the work of Wu et al. [Wu16] only returns the next direction to be taken by the client while our protocol gives the whole shortest path at once.

A brief comparison of the two works is shown in Table 7 below.

Criterion	Wu et al. protocol[Wu16]	Our protocol
Preserves the privacy of the client	Yes	Yes
Preserves the privacy of the topology	No	Yes

Number of requests to get the shortest path	R requests	One request
Complexity of the PCE	$O(N^2.d)$	$O(N^3)$
Constraint Support	No	Yes
Uses	City maps	Supports IPV6 addresses and MPLS Labels

Table 7:Comparison with Wu et al. protocol

Chapter 3

Scalable Path Computation Element (SPCE)

3.1 Introduction

A Path Computation Client (PCC) sends a request to PCE with the information necessary to compute a path. The request contains the source and destination addresses, metrics and additional constraints required for the computation process. Path Computation Element Protocol (PCEP) is the IETF standard for PCC-PCE communication. Up on receiving a request, PCE computes the optimal path using the information in its Topology Database, then sends it back to PCC to establish the LSP.

Flexible Algorithm (flex-algo) is a routing protocol that enables Interior Gateway Protocols (IGPs) to compute intra-domain paths over a network based on user-defined constraints and metrics. The computation is performed by routers participating in the specific network in a distributed manner using a Flex Algorithm definition . This definition is provisioned on one or more routers and propagated, via OSPF and IS-IS flooding, through the network [Pse19].

Scalability is an issue when designing a path computation element, since the PCE represents the entire network as one domain when it computes inter-domain paths spanning multiple domains [Per10]. This can be avoided by splitting the network into individual domains and assigning each to a controller, which precomputes all the pairs of paths between the vertices

with in the domain.The results can then be sent to an inter-domain controller that performs the end-to-end path computations on the Area Border Routers(ABRs).

The rest of this chapter is organized as follows. Section 4.2 presents a Scalable Path Computation Element (SPCE) architecture that can scale to a large number of nodes. The architecture takes advantage of the distribution and pre-computations to scale a few hundred domains. Section 4.3 discusses different scenarios of path computation requests. Depending on the location of the source and destination nodes, SPCE uses applicable logic to compute the end-to-end path. Section 4.4 as sesses an example of a Scalable Path Computation Element, and Section 4.5 presents the simulation results that measure the performance of the SPCE.

3.2 Scalable Path Computation Element

SPCE is a distributed path computation engine comprised of two types of controllers. The Domain Controller (DC) manages the nodes of a single IGP domain for a specific flex-algo, while the Inter-Domain Controller (IDC), which run son a standalone server, manages a topology composed exclusively of Area Border Routers (ABRs). Figure 7 illustrates the SPCE architecture. The engines can run as regular processes within an execution environment.

• Assuming 1% of the nodes in a certain domain are ABRs, and one ABR belongs to at least two domains, the total number of nodes supported by the SPCE can be up to 200K (i.e. one IDC can efficiently manage 200 DCs).

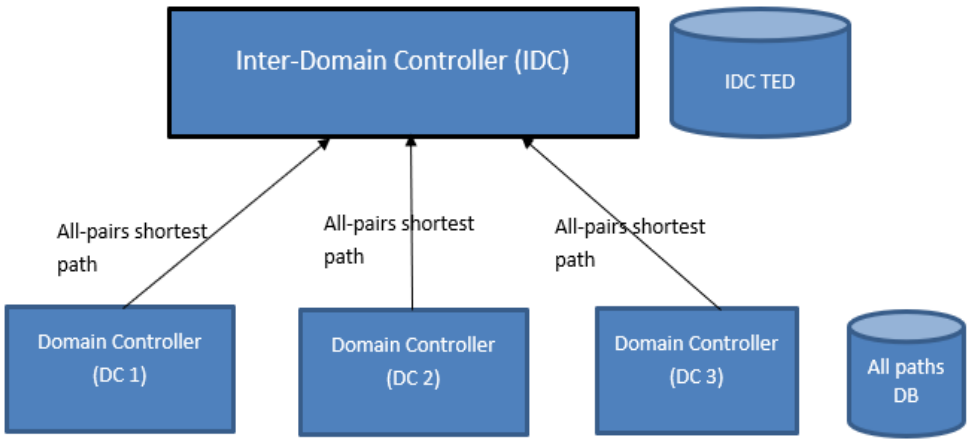

Figure 7:Scalable Path Computation Element

The DC and IDC reside in a cloud, and share the information using shared memory allocation for fast distribution and information access.

We note the following for the Domain Controller (DC):

- It plays the role of a Path Computation Engine for an IGP domain.
- It maintains a Traffic Engineering Database (TED) of nodes be longing to the same domain.
- There is one DC perflex-algo.
- Upon a topological change, DC computes all pairs' shortest paths between two nodes belonging to the domain managed by the DC. It then forwards the paths database to the Inter-Domain Controller (IDC).
- The cost of an inter-ABR path, which connects two ABRs belonging to the same do main, is computed by the DC managing the domain. This information is forwarded to the IDC with the paths database, as described in the previous note.
- The DC can easily scale to 1k nodes with current underlying PCE implementations.

Similarly, we note the following for the Inter-Domain Controller (IDC):

- It plays the role of a Path Computation Engine of a graph connecting only ABRs.
- With ABRs belonging to the same domain, the IDC ascertains the cost of the inter-ABR path from the DC managing the domain.
- IDC can easily manage 1k ABRs.

3.3 Path Computation Scenarios

In this section, we examine different path computation scenarios to illustrate diverse cases of end-to-end path computation.

1) Inter-Domain Path Computation in which the source and destination are ABRs: The IDC performs the path computation and sends the path back to the PCC (note that the Traffic Engineering Database (TED) size does not exceed 1K nodes). We verified that the existing path computation algorithms are efficient at this scale. The size of the TED allows pre-computing of all paths between all pairs of ABR nodes.

2) Source and Destination belonging to the same domain:
This is an intra-domain path scenario with the computation conducted by the DC managing the domain of the source and destination. The computed paths are then for warded to the IDC in advance.

3) Path Source is a non-ABR node: The IDC fetches the paths between the source and all A BRs be longing to its domain, then determines the shortest path between the source and the destination (minimum cost of around 10 precomputed paths).

4) Path Destination is a non-ABR node: Same as path computation scenario 3.

5) Source and Destination are non ABR nodes: Same as scenarios 3 and 4: We note that the IDC must find the minimum of 100 precomputed paths.

3.4 Scalable PCE Example

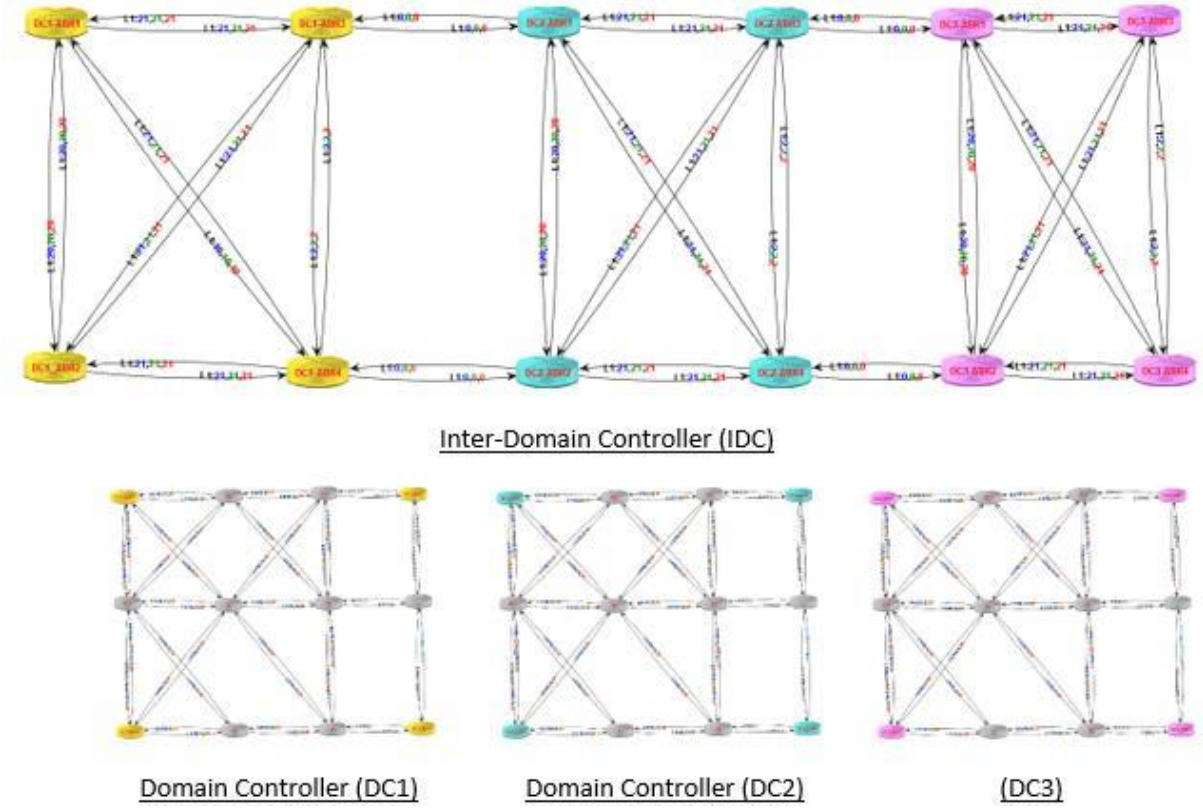

Figure 8:Example of a Scalable Path Computation Element

The scalable path computation element in Figure 8, is comprised of three domain controllers and one inter-domain controller. We note the following:

1. Each domain controller has a traffic engineering database of 12 nodes, four of which are ABRs (colored nodes).
2. The IDC manages a TED of eight ABR nodes. We note that some ABRs are part of the same physical router. For example, DC1-ABR3 and DC2-ABR1 belong to thesamephysicalnode.

3. Each DC computes all the path databases (all paths between each two pairs of nodes), and forwards them totheIDC.
4. For example, the cost of link DC1-ABR1 – DC1- ABR4 (21 in Figure 3) is computed by DC1 and forwarded to the IDC, which can build its own Inter-ABR Traffic Engineering Database.

3.5 Simulation Results

We simulated the SPCE on a topology that is represented by a honeycomb of 200 domains. As shown in Figure 9, each domain is represented by a hexagon, and the topology comprises 200 DCs each of which is represented by a hexagon with six vertices. DC manages an intra-domain of 1,000 nodes. The IDC is running an inter-domain topology of 400 (200*2) nodes, and each vertexis an ABR; note that a vertexis shared between 3 DCs.We used Floyd-Warshall [Flo62] algorithm as the intra-domain path computation to compute all paths between all pairs of nodes. The inter-domain path computation was performed using Dijkstra [Cor01] algorithm to compute the paths between the ABR nodes. The average degree of the intra-domain topology is 5, whereas,the degree oftheinter-domain topologyis 3.

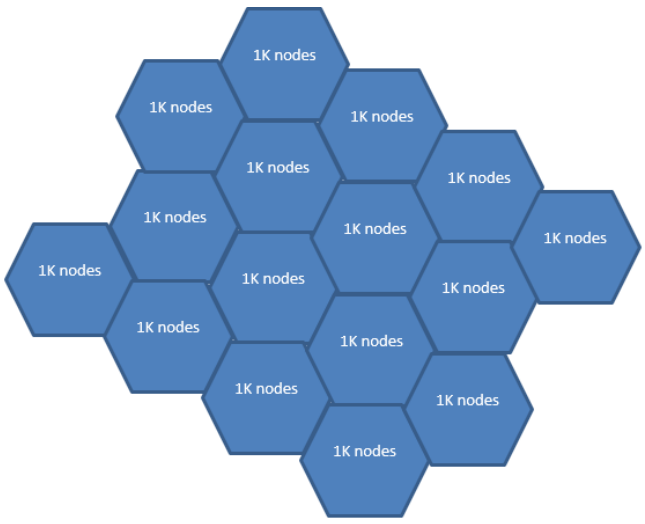

Figure 9:Simulation topology represented as a honeycomb

Simulation Results:

1. DC computed all pair paths in 2.14 seconds
2. IDC computed all inter-ABRs paths in 4.17 seconds
3. IDC computed 200,000 paths in 5.82 seconds

Thus, the SPCE can reoptimize 200,000 LSPs in a 200,000 node topology inless than 12 seconds.

Chapter 4

Conclusions and Future Research

The objective of this thesis divided in two directions, first, we present the new variant of NTRU cryptosystem, PMTRU cryptosystem, which combines the advantages of the NTRU and MaTRU cryptosystems. We have proved analytically that PMTRU has a level of security comparable to that of NTRU and MaTRU, is faster than NTRU, and has the same complexity as MaTRU. It also has greater resistance to well-known attacks, since our public key construction increases security against brute force and lattice reductions-based attempts to identify the private key.

In the second direction, we are trying to implement one of the types of cryptosystem called homomorphic in a path calculation protocol. The need of a PCC to obtain the shortest path between two nodes by sending requests to a PCE is problematic when it comes to preserving the privacy of the PCC. In fact, these requests carry sensitive data that must not be divulged. This problem has been addressed in this thesis with a novel PCE protocol that has been presented. The protocol takes advantage of a symmetric leveled homomorphic encryption scheme to enable the PCE to blindly compute the shortest path from a source node to a destination node and send the encrypted path to the PCC, which is the only one capable of retrieving the resulting path, thanks to its secret key. The protocol thus protects the privacy of the PCC against any curious PCE. On the other hand, it also protects to a certain extent the PCE privacy by keeping the overall topology of the network secret. The PCC only sends queries with addresses of nodes that may or may not exist and has in return a shortest path between the two desired nodes. Notice that even if the PCC sends a lot of queries to the server in a hope of

discovering the network topology, the obtained picture will never be complete and up-to-date, and thus will never be sufficient to enable the PCC to calculate the shortest path without consulting the PCE who has the up-to-date topology.

We implemented the protocol using a conceptually simple homomorphic encryption scheme [Van10] and gave the results of our simulation giving an idea of the still impractical aspect of all homomorphic encryption schemes.

Scalability is an issue in current Path Computation Element implementations. The traditional architecture, where the entire network is managed by one controller, can only scale a few thousand nodes,where as in this paper, we presented a SPCE that can scale to hundreds of thousand nodes or more.The key factor that distinguishes our PCE from current solutions, is it takes advantage of precomputation and distribution to achieve the high scale.

An extension of the current work would be, instead of returning a single path, to blindly return two or more disjoint paths. Disjoint paths are paths that do not share any link or node so that the PCC can have an alternative path in the case where the first path is in failure. In addition to this, these disjoint paths may not share the same status, or more precisely they may not belong to the same Shared Risk Link Group (SRLG). An SRLG can be understood as a set of links that, despite of the fact they appear to be different by just looking at the network topology, are condemned to the same fate because they are, for example, using the same underground conduit. This means that all links in an SRLG are either all up or all down: their status is the same. Therefore, the PCE could take the SRLG information into consideration so that the PCC has more robust alternative paths in case of failure of the first or second path. However, the challenge is for the PCE to be able to blindly retrieve the paths with these characteristics while keeping the computations as simple as possible.

Another work, we plan to develop an analytical model to confirm and verify the simulation results. We are also planning to generate the simulation data using a realistic customer topology. Last, we will present results for different path computation algorithms.

References

[Ahm15] Ahmad, B., Siavash Bayat, S., and rasool, J. 2015. "*On constrained implementation of Lattice-based cryptographic primitives and schemes on smart cards*." ACM Transactions on Embedded Computing Systems; **14**(3): 1–25, Article 42.

[Alp13] Alperin-Sheriff, J., and Peikert, C. 2013. "*Practical bootstrapping in quasilinear time*." In R. Canetti and J. A. Garay, editors, CRYPTO 2013, Part I, volume 8042 of LNCS, pages 1–20. Springer.

[Alp14] Alperin-Sheriff, J., and Peikert, C. 2014. "*Faster bootstrapping with polynomial error*." In J. A. Garay and R. Gennaro, editors, CRYPTO 2014, Part I, volume 8616 of LNCS, pages 297–314. Springer.

[Als15] Alsaidi, N., Said, M., Sadiq, A., and Majeed, A. 2015. "*An improved NTRU cryptosystem via commutative quaternions algebra*," Int. Conf. Security and Management, SAM'15, pp.198-203.

[App14] Applebaum, B., Ishai, Y., and Kushilevitz, E. 2014. "*How to garble arithmetic circuits*," SIAM Journal on Computing, vol. 43, no. 2, pp. 905-929.

[Ata01] Atallah, M., and Du, W. 2001. "*Secure multi-party computational geometry*," Algorithms and Data Structures, pp. 165-179.

[Ata18] Atani, R. E., Atani, S. E., and Karbasi, H. A. 2018 "*NETRU: A Non-commutative and Secure Variant of CTRU Cryptosystem*" The ISC Int'l Journal of Information Security Volume 10, Number 1 (pp. 45–53).

[Aze93] Azevedo, J., Costa, M. E. O. S., Madeira, J. J. E. S., and Martins, E. Q. V. 1993. "*An algorithm for the ranking of shortest paths*," European Journal of Operational Research, vol. 69, no. 1, pp. 97-106.

[Aze94] Azevedo, J.A., Madeira, J., Costa, M., Martins, E.Q.V., Pires, F., 1994. "*A computational improvement for a shortest paths ranking algorithm*." European Journal of Operational Research 73, 188–191.

[Ban02] Banks, W. D., and Shparlinski, I. 2002. "*A Variant of NTRU with Non-Invertible Polynomials,"* INDOCRYPT.

[Bat15] Batson, S. C. 2015. "*On the Relationship between Two Embeddings of Ideals into Geometric Space and the Shortest Vector Problem in Principal Ideal Lattices*" Ph.D. thesis, North Carolina State University.

[Ben94] Benaloh, J. 1994. "*Dense probabilistic encryption*," in Proceedings of the workshop on selected areas of cryptography, pp. 120-128.

[Ber16] Bernstein, D. J., Chuengsatiansup, C., Lange, T., Vredendaal, C. V. 2016 "*NTRU prime*" https://ntruprime.cr.yp.to/ntruprime-20160511.pdf.

[Bra13] Brakerski, Z., Gentry, C. and Halevi, S. 2013. "*Packed ciphertexts in LWE-based homomorphic encryption*." In K. Kuro-sawa and G. Hanaoka, editors, PKC 2013, volume 7778 of LNCS, pages 1–13. Springer.

[Bri05] Brickell, J., and Shmatikov, V. 2005. "*Privacy-preserving graph algorithms in the semi-honest model,"* in International Conference on the Theory and Application of Cryptology and Information Security. Springer, pp. 236-252.

[Bri89] Brickell, E. F. 1989. "*Survey of hardware implementations of RSA*." In Proceeding of CRYPTO '89, LNCS. Springer: NewYork, USA; 368–370.

[Cab14] Cabarcas, D., Weiden, P., and Buchmann, J. 2014. "*On the efficiency of provably secure NTRU*." International Workshop on Post-Quantum Cryptography, 22-39.

[Cam18] Camara, M. G., Demba, S., and Djiby, S. 2018. "*DTRU1: First Generalization of NTRU Using Dual Integers*" International Journal of Algebra, Vol. 12, no. 7, 257 – 271.

[Cha76] Chandra, A. 1976. "*Maximal parallelism in matrix multiplication*," Report RC 6193, I.B.M. Watson Research Center, Yorktown Heights, N.Y.

[Che16] Chehelcheshmeh, S. B., and Hosseinzadeh, M. 2016. "*Quantum- resistance authentication in centralized cognitive radio networks*." Security and Communication Network; **9**(10): 1158–1172.

[Cog05] Coglianese, M., and Goi, B. M. 2005. "*MaTRU a new NTRU-based cryptosystem*." In *Proceeding of INDOCRYPT 2005, LNCS*, Vol. 3797. Springer: Dalian, China; 232–243.

[Cop97] Coppersmith, D., and Shamir, A. 1997. "*Lattice attacks on NTRU,*" In International Conference on the Theory and Applications of Cryptographic Techniques, Vol. 1233, Springer, Berlin, 52-61. https://doi.org/10.1007/3-540-69053-0 5.

[Cor01] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and, Stein, C. 2001. "*Section 24.3: Dijkstra's algorithm*". Introduction to Algorithms (Second ed.). MIT Press and McGraw–Hill. pp. 595–601. ISBN 0-262-03293-7.

[Cra10] Craw, S. 2010. "*Manhattan distance*," in Encyclopedia of Machine Learning, C. Sammut and G. I. Webb, Eds. Boston, MA: Springer US, pp. 639-639. [Online]. Available: https://doi.org/10.1007/978-0-387-30164-8 506.

[Cra17a] Crabbe, E., Minei, I., Medved, J., and R. Varga, 2017. "*Path Computation Element Communication Protocol (PCEP) Extensions for Stateful PCE*", RFC 8231, DOI 10.17487/RFC8231, September 2017, <https://www.rfc-editor.org/info/rfc8231>.

[Cra17b] Crabbe, E., Minei, I., Medved, J., Varga, R., Zhang, X., and D. Dhody, "*Optimizations of Label Switched Path State Synchronization Procedures for a Stateful PCE"*, RFC 8232, DOI 10.17487/RFC8232, September 2017, <https://www.rfc-editor.org/info/rfc8232>.

[Das08] Das, A. S., Keshri, J. K., Srinathan, K., and Srivastava, V. 2008. "*Privacy preserving shortest path computation in presence of convex polygonal obstacles*," in Availability, Reliability and Security. ARES 08. Third International Conference on. IEEE, pp. 446-451. 23.

[Dho17] Dhody, D., and Zhang, X. 2017. "*Stateful Path Computation Element (PCE) Interdomain Considerations*," Internet Engineering Task Force, Internet-Draft draft-dhodypce- stateful-pce-interdomain-05, Sep. work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-dhody-pce-stateful-pce-interdomain-05.

[Dif76] Diffie, W., and Hellman, M. 1976. "*New directions in cryptography*," IEEE Transactions On information theory, vol. 22, no.6, pp.644-654.

[Dij59] Dijkstra, E. W. 1959. "*A note on two problems in connexion with graphs*," Numerische mathematik, vol. 1, no. 1, pp. 269-271.

[Du01] Du, W., Atallah, M. J., and Spafford, E. H. 2001. "*A study of several specific secure two-party computation problems*," Purdue University, West Lafayette, IN.

[Duc15] Ducas, L. and Micciancio, D. 2015. "*FHEW: Bootstrapping homomorphic encryption in less than a second*." In Eurocrypt, pages 617–640.

[Dwo04] Dwork, C., Naor, M., and Reingold, O. 2004. "*Immunizing Encryption Schemes from Decryptions errors*." EUROCRYPT 2004: Advances in Cryptogray – EUROCRYPT 2004 pp 342-360.

[Elg85] ElGamal, T. 1985. "*A public key cryptosystem and a signature scheme based on discrete logarithms*," IEEE Transactions on Information Theory, 31, 469-472.

[Epp94] Eppstein, D. 1994. "*Finding the k shortest paths*," in Proceedings of the 35th Annual Symposium on Foundations of Computer Science, ser. SFCS '94. Washington, DC, USA: IEEE Computer Society, pp. 154-165. [Online]. Available: https://doi.org/10.1109/SFCS.1994.365697.

[Epp98] Eppstein, D. 1998. "*Finding the k shortest paths*," SIAM Journal on computing, vol. 28, no. 2, pp. 652-673.

[Flo62] Floyd, R. W. 1962. "*Algorithm 97*." Comm. ACM 5-6, 345.

[Fox73] Fox, B. L. 1973. "*Calculating k th shortest paths*," INFOR: Information Systems and Operational Research, vol. 11, no. 1, pp. 66-70.

[Fox78] Fox, B. L. 1978. "*Data structures and computer science techniques in operations research*," Operations Research, vol. 26, no. 5, pp. 686-717.

[Gab02] Gaborit, P., Ohler, J., and Soli, P. 2002. "*CTRU, a polynomial analogue of NTRU*," INRIA. Rapport de recherche, N. 4621.

[Gah12] Gahi, Y., Guennoun, M., Guennoun, Z., and El-Khatib, K. 2012. "*Privacy preserving scheme for location based services*," Journal of Information Security, vol. 3, no. 02, p. 105.

[Gau14] Gauravaram, P., Narumanchi, H., and Emmadi, N. 2014. "*Analytical study of Implementation issues of NTRU*", International Conference on Advances in Computing, Communications and Informatics, IEEE, New Delhi, India, pp. 700-707.

[Gen01] Gentry, C. 2001. "*Key Recovery and Message Attacks on NTRU-Composite*," Advances in Cryptology - EUROCRYPT, LNCS 2045, Springer-Verlag.

[Gen09] Gentry, C. 2009. "*Fully homomorphic encryption using ideal lattices*." In 41st ACM STOC, pages 169–178.

[Gen11] Gentry, C. and Halevi, S. 2011. "*Implementing Gentry's fully-homomorphic encryption scheme*." In K. G. Paterson, editor, EUROCRYPT 2011, volume 6632 of LNCS, pages 129–148. Springer.

[Gen12a] Gentry, C., Halevi, S., Peikert, C., and Smart, N. P. 2012. "*Ring switching in BGV-style homomorphic encryption*." In I. Visconti and R. D. Prisco, editors, SCN 12, volume 7485 of LNCS, pages 19–37. Springer.

[Gen12b] Gentry, C., Halevi, S., and Smart, N. P. 2012. "*Better bootstrapping in fully homomorphic encryption*." In M. Fischlin, J. Buchmann, and M. Manulis, editors, PKC 2012, volume 7293 of LNCS, pages 1–16. Springer.

[Gen12c] Gentry, C., Halevi, S. and Smart, N. P. 2012. "*Fully homomorphic encryption with polylog overhead*." In D. Pointcheval and T. Johansson, editors, EUROCRYPT 2012, volume 7237 of LNCS, pages 465–482. Springer.

[Gen12d] Gentry, C., Halevi, S. and Smart, N. P. 2012. "*Homomorphic evaluation of the AES circuit*." In R. Safavi-Naini and R. Canetti, editors, CRYPTO 2012, volume 7417 of LNCS, pages 850–867. Springer, Aug.

[Gol82] Goldwasser, S., and Micali, S. 1982. "*Probabilistic encryption & how to play mental poker keeping secret all partial information*," in Proceedings of the fourteenth annual ACM symposium on Theory of computing. ACM, pp. 365-377.

[Haj20a] Hajaje, H., Guennoun, Z. D. A., and Guennoun, M. 2020. "*Constraint-Based Privacy Preserving Path Computation Element*" , International Journal of Smart Security Technologies, IGI Global, Volume 6, Issue 2.

[Haj20b] Hajaje, H., Guennoun, Z. D. A., and Guennoun, M. 2020. "*PMTRU: another variant of the NTRU public key cryptosystem*" the 2020 IEEE Canadian Conference of Electrical and Computer Engineering, London, Ontario, Canada, April 26-29, 2020.

[Haj20c] Hajaje, H., Guennoun, Z. D. A., Israr, J., and Guennoun, M. 2020. "*Scalable Path Computation Element (SPCE)"* the 2020 IEEE Canadian Conference of Electrical and Computer Engineering, London, Ontario, Canada, April 26-29, 2020.

[Hal14] Halevi, S., and Shoup, V. 2014. "*Algorithms in HElib*." In J. A. Garay and R. Gennaro, editors, CRYPTO 2014, Part I, volume 8616 of LNCS, pages 554–571. Springer, Aug.

[Han05] Han, D. 2005. "*A new lattice attack on NTRU cryptosystem*," Trends in Mathematics Information Center for Mathematical Sciences, 8, no. 1, 197-205.

[Haz10] Hazay, C., and Lindell, Y. 2010. "*Efficient secure two-party protocols: Techniques and constructions*." Springer Science & Business Media.

[Hof98] Hoffestein, J., Pipher, J., and Silverman, J. H. 1998. "*NTRU: A Ring-Based Public Key Cryptosystem*." In Proceeding of ANTS III, LNCSA, vol. 1423, Springer- Verlag, pp. 267- 288.

[How04] Howgrave-Graham, N., Silverman, J., Singer, A., and Whyte, W. 2005. "*NAEP: Provable Security in the Presence of Decryption Failures.*" NTRU Cryptosystems. available at: http://www.ntru.com.

[How05] Howgrave-Graham, N., Silverman, J., and Whyte, M. 2005. "*Choosing Parameter Sets for NTRUEncrypt with NAEP and SVES-3*." available at http://www.ntru.com.

[Hu09] Hu, F., Wilhelm, K., Schab, M., Lukowiak, M., Radzis-zowski, S., and Xiao, Y. 2009. "*NTRU-based sensor network security: a low-power hardware implementation perspective.*" Security and Communication Network; **2**(1): 71–81.

[IEEE] IEEE P1363, Standard Specifications For Public-Key Cryptography, http://grouper. ieee.org/groups/1363/.

[Jar13] Jarvis, K., and Nevins, M. 2013. "*ETRU: NTRU over the Eisenstein integers*," Springer Science +Business Media New York.

[Jim99] Jiménez, V. M., and Marzal, A. 1999. "*Computing the k shortest paths: A new algorithm and an experimental comparison*," in International Workshop on Algorithm Engineering. Springer, pp. 15-29. 24.

[Kob04] Koblitz, N., and Menezes, A. J. 2004. "*A Survey of Public Key Cryptosystems*," SIAM Review, Vol. 46, No. 4, pages 599-634.

[Kob87] Koblitz, N. 1987. "*Elliptic curves cryptosystems*." Math of Comp. vol. 48, pp. 203-209.

[Kou06] Kouzmenko, R. 2006. "*Generalizations of the NTRU cryptosystem*," Preprints.

[Kus97] Kushilevitz, E., and Ostrovsky, R. 1997. "*Replication is not needed: Single database, computationally private information retrieval*," in Foundations of Computer Science. Proceedings., 38th Annual Symposium on. IEEE, pp. 364-373.

[Len82] Lenstra, A.K., Lenstra, H.W., and Lovasz, L. 1982. "*Factoring polynomials with rational coefficients*," Mathematische Annalen, Vol. 261, 513-534.

[Lop17] Lopez, D., Gonzalez de Dios, O., Wu, Q., and D. Dhody, "*PCEPS: Usage of TLS to Providea Secure Transport for the Path Computation Element Communication Protocol (PCEP)",* RFC 8253, DOI 10.17487/RFC8253, October 2017,<https://www.rfc-editor.org/info/rfc8253>.

[Mal09] Malekian, E., Zakerolhosseini, A., and Mashatan, A. 2009. "*QTRU: a lattice attack resistant version of NTRU*." https://eprint. iacr.org/2009/386.pdf.

[Mal10] Malekian, E., and Zakerolhosseini, A. 2010. "*OTRU: a non-associative and high speed public key cryptosystem*." In 15th CSI International Symposium on Com- puter Architecture and Digital Systems, 23-24 Sept.

[Man14] Manifavas, C., Hatzivasilis, G., Fysarakis, K., and Rantos, K. 2014. "*Lightweight cryptography for embedded systems - a comparative analysis. DPM 2013 and SETOP 2013, LNCS 8247*," Springer, NewYork, USA; 333–349.

[Mar00] Martins, E. d. Q. V., and Dos Santos, J. L. E. 2000. "*A new shortest paths ranking algorithm*," Investigação Operacional, vol. 20, no. 1, pp. 47-62.

[Mar84] Martins, E. d. Q. V. 1984. "*An algorithm for ranking paths that may contain cycles*," European Journal of Operational Research, vol. 18, no. 1, pp. 123-130.

[Mat88] Matsumoto, T., and Imai, H. 1988. "*Public quadratic polynomial-tuples for efficient signature-verification and message-encryption*," In Proceed- ing of Eurocrypt '88, LNCS of vol. 330, Springer-Verlag, pages 419-453.

[May01] May, A. and Silverman, J. H. 2001. "*Dimension Reduction Methods for Convolution Modular Lattices*," in International Cryptography and Lattices Conference, Vol. 2146, Springer-Verlag, 110-125. https://doi.org/10.1007/3-540-44670-2 10

[Mce78] McEliece, R. J. 1978. "*A public key cryptosystem based on algebraic coding theory,"* JPL DSN Progress Report, No. 42-44, pages 114-116.

[Mia91] Miaou, S. P., and Chin, S. M. 1991. "*Computing k-shortest path for nuclear spent fuel highway transportation*," European Journal of Operational Research, vol. 53, no. 1, pp. 64-80.

[Mon08] Monteverde, M. 2008. "*NTRU software implementation for constrained devices*." Master Thesis, KATHOLIEKE UNIVERSITEIT LEUVEN.

[Nay08] Nayak, R., Sastry, C. V., and Pradhan, J. 2008. "*A matrix formula- tion for NTRU cryptosystem*." In Proceeding of ICON, LNCS. Springer: New Delhi, India.

[Ngu01] Nguyen, P. Q., and Stern, J. 2001. "*The Two Faces of Lattices in Cryptology*," in International Cryptography and Lattices Conference, Vol. 2146, Springer-Verlag, 148-180. https://doi.org/10.1007/3-540-44670-2 12

[Nit14] Nitaj A. 2014. "*Cryptanalysis of NTRU with two public keys*," International Journal of Network Security, 16(2): 112-117.

[Pai99] Paillier, P. 1999. "*Public-key cryptosystems based on composite degree residuosity classes*," in Eurocrypt, vol. 99. Springer, pp. 223-238.

[Pan12] Pan, Y. and Deng, Y. 2012. "*A General NTRU-Like Framework for Constructing Lattice Based Public Key Cryptosystems*," Springer-Verlag Berlin Heidelberg, p.p. 109–120.

[Per09] Perlner, R. A., and Cooper, D. A. 2009. "*Quantum resistant public key cryptography*," a survey, in: Proc. of ACM, 85-93.

[Per10] Perelló, J., Hernández‐Sola, G., Agraz, F., Spadaro, S. and Comellas, J. (2010), "*Scalable Path Computation Flooding Approach for PCE‐Based Multi-domain Networks."* ETRI Journal, 32: 622-625. doi:10.4218/etrij.10.0210.0063.

[Pse19] Psenak, P., Hegde, S., Filsfils, C., Talaulikar, K., and Gulko, A. 2019. "*IGP F lexible Algorithm*", draft -ietf-lsr- flex- algo-04 (work in progress).

[Riv78a] Rivest, R., Shamir, A., and Adleman, L. 1978. "*A method for obtaining digital signature and public key cryptosystems*," Communications of the ACM, vol. 21, no. 2, pp.120-126.

[Riv78b] Rivest, R., Adleman, L., and Dertouzos, M. 1978. "*On data banks and privacy homomorphisms*." In Foundations of Secure Computation, pages 169–180.

[Rob82] Robling, D. 1982. "*Cryptography and data security*," Addison – Wisely Publishing Company.

[Rot05] Rothe, JRG. 2005. "*Complexity Theory and Cryptology*." Springer: Berlin.

[She09] Shen, X., Du, Z., Chen, R. *2009. "Research on NTRU algorithm for mobile java security*." International Conference on Scalable Computing and Communications;The Eighth International Conference on Embedded Computing, SCALCOM- EMBEDDED- COM'09, Dalian, China, 2009; 366–369.

[Shi76] Shier, D. R. 1976. "*Iterative methods for determining the k shortest paths in a network*," Networks, vol. 6, no. 3, pp. 205-229.

[Sho99] Shor, P. W. 1999. "*Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*," SIAM Review 41 (), 303-332.

[Sin13] Singh, G., and Supriya, A. 2013. "*study of encryption algorithm (RSA, DES, 3DES and AES) for information security*." International Journal of Computer Applications; **67**: 33–38.

[Son15] Song, J. E., Han, T. Y., and Lee, M. L. 2015. "*Analysis and Improvement of MaTRU Public Key Crypto-system* ," IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences, E98.A(4), pp. 982 991.

[Tal06] Talbot, J., Welsh, D. 2006. "*Complexity and Cryptography*." Cambridge University Press: Cambridge.

[Tha16] Thakur, K., Tripathi, B. P. 2016. "*BTRU, A Rational Polynomial Analogue of NTRU Cryptosystem*" International Journal of Computer Applications (0975 – 8887) Volume 145 – No.

[Van10] Van Dijk, M., Gentry, C., Halevi, S., and Vaikuntanathan, V. 2010. "*Fully homomorphic encryption over the integers*," in Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer, pp. 24-43.

[Vas09] Vasseur, JP., Ed. and JL. Le Roux, Ed., "*Path Computation Element (PCE) Communication Protocol (PCEP)*", RFC 5440, DOI 10.17487/RFC5440, March 2009, <https://www.rfc-editor.org/info/rfc5440>.

[Vat08] Vats, N. 2008. "*Algebraic cryptanalysis of CTRU cryptosystem*." In *COCOON 2008, Proceeding of LNCS*, Vol. 5092. Springer: Dalian, China; 235–244.

[Vat09] Vats, N. 2009. "*NNRU. A non-commutative analogue of NTRU*," arXiv:0902.1891v1.[cs.CR].

[Wu16] Wu, D. J., Zimmerman, J., Planul, J., and Mitchell, J. C. 2016. "*Privacy-preserving shortest path computation*," arXiv preprint arXiv:1601.02281.

[Yao86] Yao, A. C. C. 1986. "*How to generate and exchange secrets*," in Foundations of Computer Science., 27th Annual Symposium on. IEEE, pp. 162-167.

[Yas16a] Yassein, H.R., and AlSaidi, N. 2016. "*HXDTRU Cryptosystem Based On Hexadecnion Algebra*," 5th International Cryptology and Information Security Conference.

[Yas16b] Yassein, H.R. and Alsaidi, N. M. G. 2016. "*BITRU: Binary Version of the NTRU Public Key Cryptosystem via Binary Algebra*" International Journal of Advanced Computer Science and Applications, Vol. 7, No. 11.

[Zha11] Zhao, N., and Su, S. 2011. "*An improvement and a new design of Algorithms for Seeking the Inverse of NTRU polynomial*," EEE Computer Society, Washington.

[Zha13] Zhan, X., Zhang, R., Xiong, Z., Zheng, Z., and Liu, Z. 2013. "*Efficient implementations of NTRU*." The 9th international conference on Wireless Communications, Networking and Mobile Computing(WICOM2013), Beijing, China; 485–492.