Étude Algorithmique du Problème LPN
Thématique : Algorithmique/Cryptologie

Laboratoire : IRISA Rennes
Ville : Rennes
Équipe : Cidre
Directeur de stage : Pierre-Alain Fouque, `Pierre-Alain.Fouque@ens.fr`
Directeur du laboratoire : Jean-Marc Jézéquel

Présentation générale du domaine

Le problème LPN (Learning Parities with Noise) est un problème célèbre en apprentissage et en cryptographie [3, 11]. Ce problème est relié à certains problèmes sur les réseaux euclidiens [11] et de nombreux schémas cryptographiques ont été construits dont la sécurité repose sur la difficulté de résoudre ce problème [11, 7, 4, 8]. Ce problème consiste simplement à résoudre un système linéaire mais où chaque équation est modifiée avec une certaine probabilité.

Description du LPN problème sur $\mathbb{F}_2$. Soit $\boldsymbol{s} \in \{0,1\}^n$ un vecteur de bits de taille n, soit Ber_ϵ la distribution de Bernoulli de paramètre $\epsilon \in [0, 1/2[$, telle que si $\nu \sim \mathsf{Ber}_\epsilon$, alors $\Pr[\nu = 1] = \epsilon$ et $\Pr[\nu = 0] = 1 - \epsilon$, et soit $A_{\boldsymbol{s},\epsilon}$ la distribution définie par

$$\{\boldsymbol{a} \leftarrow \{0,1\}^n; \nu \leftarrow \mathsf{Ber}_\epsilon : (\boldsymbol{a}, \boldsymbol{a} \cdot \boldsymbol{s} \oplus \nu)\}.$$

Supposons que $A_{\boldsymbol{s},\epsilon}$ représente également un oracle qui retourne des échantillons pris selon cette distribution indépendamment les uns des autres entre chaque tirage. Un algorithme M est dit un (t, q, δ)-algorithme pour résoudre le problème LPN_ϵ si

$$\Pr[\boldsymbol{s} \leftarrow \{0,1\}^n : M^{A_{\boldsymbol{s},\epsilon}}(1^n) = \boldsymbol{s}] \geq \delta,$$

et M s'exécute en temps au plus t et pose au plus q questions à l'oracle.

Résoudre le problème LPN consiste à trouver $\boldsymbol{x}$ connaissant $\boldsymbol{a}_i$ et b_i du système suivant où les ν_i sont tirés selon une variable aléatoire ν de Bernouilli de paramètre $\Pr[\nu = 1] = \epsilon$ et indépendant les uns des autres :

$$\begin{cases} \boldsymbol{a}_1 \cdot \boldsymbol{x} + \nu_1 &= b_1 \\ \boldsymbol{a}_2 \cdot \boldsymbol{x} + \nu_2 &= b_2 \\ \qquad\qquad \ldots & \ldots \quad \ldots \\ \boldsymbol{a}_k \cdot \boldsymbol{x} + \nu_k &= b_k \end{cases} \tag{1}$$

Algorithmes par maximum de vraisemblance. La première solution consiste à utiliser un algorithme par maximum de vraisemblance. Si le nombre d'équations

du système 1 est petit, de l'ordre de $O(n)$ équations, la valeur la plus probable de $\boldsymbol{x}$ qui satisfait le plus grand nombre d'équations de ce système est la valeur $\boldsymbol{x} = \boldsymbol{s}$. L'algorithme recherche $\boldsymbol{x}$ exhaustivement et retourne celui qui satisfait le plus grand nombre d'équations. En utilisant l'inégalité de Chernoff et en sommant sur tous les $\boldsymbol{s} \in \mathbb{F}_2^n$, il est possible de montrer que l'algorithme retourne la bonne solution avec une bonne probabilité. Cet algorithme s'exécute en temps $O(n2^n)$.

La deuxième solution consiste à obtenir plusieurs équations qui ne font intervenir qu'un petit nombre de bits de $\boldsymbol{s}$. Supposons que nous ayons des équations du type $x_i = b_j$ pour plusieurs j et x_i représente le i-ième bit de $\boldsymbol{x}$, qui soient biaisées d'un facteur η, c'est-à-dire $|\Pr[\eta = 0] - 1/2| = \eta$. Dans ce cas, nous pouvons retrouver ce bit par un vote par majorité avec bonne probabilité dès que nous avons $C\eta^{-2}$ équations sur le même bit i. Si η est une constante, la complexité est $O(n2^n)$ pour obtenir les équations. En effet, obtenir un vecteur $\boldsymbol{a}_i$ spécifique demandent en moyenne un temps $O(2^n)$, et comme il nous en faut $O(1/\eta^2)$ pour chacun des n bits de $\boldsymbol{s}$, la complexité est en $O(n2^n)$. Contrairement à la technique précédente, le nombre initial d'équations est beaucoup plus grand pour obtenir de telles équations.

Il existe un algorithme pour résoudre ce problème en temps $2^{O(n/\log n)}$ qui utilise deux phases [3]. Pendant la première étape, le système est réduit à $n/\log n$ variables, mais le bruit augmente en utilisant l'algorithme de Wagner [10, 12]. Pendant la deuxième phase, l'algorithme calcule par recherche exhaustive le vecteur le plus probable. J'ai proposé une variante de cet algorithme pour réduire la deuxième phase de l'attaque en exploitant la Transformée de Walsh-Hadamard [9] (en fait, on améliore uniquement la constante dans l'exposant de la complexité exponentielle, mais c'est important en crypto).

Des variantes dans certains modèles de bruit ont aussi été proposées [2] et la complexité devient polynomiale.

Objectifs du stage

L'objectif de ce stage est d'implémenter un algorithme efficace pour résoudre ce problème. D'un point de vue théorique, il pourra être intéressant de remplacer la Transformée de Walsh-Hadamard, par un algorithme qui recherche uniquement les coefficients de Fourier les plus lourds, au lieu de tous les calculer [1]. Cela permettra de réduire la complexité de la deuxième phase, et il pourra être envisagé d'utiliser cette approche plus globalement pour réduire la complexité de l'algorithme. En fonction du niveau de bruit, paramètre ϵ et de n, il pourra aussi être intéressant d'étudier divers algorithmes comme celui qui recherchent un sous-ensemble de taille n des équations qui ne sont pas bruitées. Enfin, de nouveaux algorithmes dont la

complexité dépend du nombre de coefficients de Fourier non nul [6, 5] ont été proposés récemment.

Compétences espérées

Bonne connaissance du cours d'Algorithmique et Programmation, bonne connaissance de la programmation en C et quelques notions d'analyse d'algorithmes probabilistes.

Références

[1] Akavia, A. : Finding significant fourier transform coefficients deterministically and locally. Electronic Colloquium on Computational Complexity (ECCC) **15**(102) (2008)

[2] Arora, S., Ge, R. : New algorithms for learning in presence of errors. In Aceto, L., Henzinger, M., Sgall, J., eds. : ICALP (1). Volume 6755 of Lecture Notes in Computer Science., Springer (2011) 403–415

[3] Blum, A., Kalai, A., Wasserman, H. : Noise-tolerant learning, the parity problem, and the statistical query model. J. ACM **50**(4) (2003) 506–519

[4] Gilbert, H., Robshaw, M.J.B., Seurin, Y. : How to encrypt with the lpn problem. In Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I., eds. : ICALP (2). Volume 5126 of Lecture Notes in Computer Science., Springer (2008) 679–690

[5] Hassanieh, H., Indyk, P., Katabi, D., Price, E. : Nearly optimal sparse fourier transform. In Karloff, H.J., Pitassi, T., eds. : STOC, ACM (2012) 563–578

[6] Hassanieh, H., Indyk, P., Katabi, D., Price, E. : Simple and practical algorithm for sparse fourier transform. In Rabani, Y., ed. : SODA, SIAM (2012) 1183–1194

[7] Hopper, N.J., Blum, M. : Secure human identification protocols. In Boyd, C., ed. : ASIACRYPT. Volume 2248 of Lecture Notes in Computer Science., Springer (2001) 52–66

[8] Juels, A., Weis, S.A. : Authenticating pervasive devices with human protocols. In Shoup, V., ed. : CRYPTO. Volume 3621 of Lecture Notes in Computer Science., Springer (2005) 293–308

[9] Levieil, É., Fouque, P.A. : An improved lpn algorithm. In Prisco, R.D., Yung, M., eds. : Security and Cryptography for Networks, 5th International Conference, SCN 2006. Volume 4116 of Lecture Notes in Computer Science., Springer (2006) 348–359

[10] Minder, L., Sinclair, A. : The extended k-tree algorithm. J. Cryptology **25**(2) (2012) 349–382

[11] Regev, O. : On lattices, learning with errors, random linear codes, and cryptography. J. ACM **56**(6) (2009)

[12] Wagner, D. : A generalized birthday problem. In Yung, M., ed. : Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference. Volume 2442 of Lecture Notes in Computer Science., Springer (2002) 288–303