Nearly Sparse Linear Algebra

and application to the Discrete Logarithm Problem

A. Joux[1], C. Pierrot[2]

[1] *CryptoExperts and Chaire de Cryptologie de la Fondation de l'UPMC, antoine.joux@m4x.org*
[2] *CNRS, DGA, and Sorbonne Universités, LIP6/UPMC, Paris, France, Cecile.Pierrot@lip6.fr*

In this talk, we propose a method to perform linear algebra on a matrix with nearly sparse properties. More precisely, although we require the main part of the matrix to be sparse, we allow some dense columns with possibly large coefficients. We modify Block Wiedemann algorithm and show that the contribution of these heavy columns can be made negligible compared to the one of the sparse part of the matrix. In particular, this eases the computation of discrete logarithms in medium and high characteristic finite fields, where *nearly sparse matrices* naturally appear.

Sparse Linear Algebra. Linear algebra is a widely used tool in both mathematics and computer science. At the boundary of these two disciplines, cryptography is no exception to this rule. Yet, one notable difference is that cryptographers mostly consider linear algebra over finite fields, bringing both drawbacks – the notion of convergence is no longer available – and advantages – no stability problems can occur. As in combinatory analysis or in the course of solving partial differential equations, cryptography also presents the specificity of frequently dealing with sparse matrices.

A sparse matrix is a matrix containing a relatively small number of coefficients that are not equal to zero. It often takes the form of a matrix in which each row (or column) only has a small number of non-zero entries, compared to the dimension of the matrix. With sparse matrices, it is possible to represent in computer memory much larger matrices, by giving for each row (or column) the list of positions containing a non-zero coefficient, together with its value. When dealing with a sparse linear system of equations, using plain Gaussian Elimination is often a bad idea, since it does not consider nor preserve the sparsity of the input matrix. Indeed, each pivoting step during Gaussian Elimination increases the number of entries in the matrix and, after a relatively small number of steps, it overflows the available memory.

Three families of sparse linear algebra algorithms. In order to deal with sparse systems, a different approach is required. Three main families of algorithms have been devised: the first one adapts the ordinary Gaussian Elimination in order to choose pivots that minimize the loss of sparsity and is generally used to reduce the

initial problem to a smaller slightly less sparse problem. The two other algorithm families work in a totally different way. Namely, they do not try to modify the input matrix but aim at directly finding a solution of the sparse linear system by computing only matrix-by-vector multiplications. One of these families consists of Krylov Subspace methods, adapted from numerical analysis, and constructs sequences of mutually orthogonal vectors.

Block Wiedemann algorithm. Throughout this talk, we focus on the third family that contains Wiedemann algorithm and its generalizations. Instead of computing an orthogonal family of vectors, Wiedemann proposed in 1986 [5] to reconstruct the minimal polynomial of the considered matrix. This algorithm computes a sequence of scalars of the form ${}^twA^iv$ where v and w are two vectors and A the sparse matrix of the linear algebra problem. It tries then to extract a recursive relationship that holds for this sequence. Coppersmith and Kaltofen [2, 3] adapted Wiedemann algorithm for parallelization and even distributed computations. The main idea of Coppersmith's Block Wiedemann algorithm is to compute a sequence of matrices of the form ${}^tWA^iV$ where V and W are not vectors as previously but *blocks* of vectors. This step is parallelized by distributing the vectors of the block V to **several processors or CPUs – let us say** c. The asymptotic complexity of extracting the recursive relationships within the sequence of small matrices is in $\tilde{O}(cN^2)$ where N **is the largest dimension of the input matrix**. Finally, a further improvement was proposed by Thomé [4] in 2002: he reduced the complexity of finding the recursive relationships to $\tilde{O}(c^2N)$.

Note that both Krylov Subspace methods and Wiedemann algorithms cost a number of matrix-by-vector multiplications equal to a small multiple of the matrix dimension: for **a matrix containing λ entries per row in average**, the cost of these matrix-by-vector multiplications is $O(\lambda N^2)$. With Block Wiedemann, it is possible to distribute the cost of these products on c machines. In this case, the search for recursive relationships adds an extra cost of the form $\tilde{O}(c^2N)$.

Nearly Sparse Linear Algebra. For a *d-nearly sparse matrix*, which includes d dense columns in addition to its sparse part, the cost of matrix-by-vector multiplications increases. As a consequence, the total complexity becomes:

$$O((\lambda + d)N^2) + \tilde{O}(c^2N) \tag{1}$$

where the second term is an extra cost for Block Wiedemann.

In this talk, we present an algorithm to solve linear algebra problems associated to these special matrices. Our aim is to adapt the Coppersmith's Block Wiedemann

algorithm to improve the cost of linear algebra on matrices that have nearly sparse properties and reduce it to:

$$O(\lambda N^2) + \tilde{O}(\max(c,d)^2 N). \tag{2}$$

We compare our method with preexisting linear algebra techniques and show that it is competitive even with a large number of dense columns. In particular, when the number of dense columns is lower than the number of processors we use for the matrix-by-vector steps, we show that the presence of these unwelcome columns does not affect the complexity of solving linear systems associated to these matrices.

Application to Discrete Logarithm Computations in Finite Fields. In practice, this result precisely applies to the discrete logarithm problem. Indeed, nearly sparse matrices appear in both medium and high characteristic finite fields discrete logarithm computations. To illustrate this claim, we recall the latest record [1] announced in June 2014 for the computation of discrete logarithms in a prime field GF_p, where p is a 180 digit prime number. It uses a matrix containing 7.28M rows and columns with an average weight of 150 non-zero coefficients per row and also presents 4 dense Schirokauer maps columns. These columns precisely give to the matrix the nearly sparse structure we study.

References

[1] Cyril Bouvier and Pierrick Gaudry and Laurent Imbert and Hamza Jeljeli and Emmanuel Thomé, *Discrete logarithms in GF(p) – 180 digits*, Announcement to the NMBRTHRY list, item 003161 (June 2014).

[2] Don Coppersmith, *Solving homogeneous linear equations over GF(2) via block Wiedemann algorithm*, in Mathematics of Computation, **62**, pp. 333–350 (1994).

[3] Erich Kaltofen, *Analysis of Coppersmith's Block Wiedemann Algorithm for the Parallel Solution of Sparse Linear Systems*, in Mathematics of Computation, pp. 777–806 (1995).

[4] Emmanuel Thomé, *Subquadratic Computation of Vector Generating Polynomials and Improvement of the Block Wiedemann Algorithm*, in *J. Symb. Comput.* **33**, 5, pp. 757–775 (2002).

[5] Douglas H. Wiedemann, *Solving sparse linear equations over finite fields*, IEEE Transactions on Information Theory **32**, 1, pp. 54–62 (1986).