Public key cryptography based on the clique and learning parity with noise problems for post-quantum cryptography

Péter Hudoba
peter.hudoba@inf.elte.hu

ELTE Eötvös Loránd University
Faculty of Informatics
Budapest, Hungary

Abstract

In this paper we propose a new method for public key encryption. The scheme's security is based on the well-known clique and learning parity with noise problems.

The relevance of this approach is justified by its post-quantum nature: unlike RSA and other cryptosystems based on the hardness of factorization or discrete logarithm which could be broken by a suitable large quantum device, no known quantum attacks are known against our candidate system. We examine the time complexity of our scheme and compare it to RSA.

1 Introduction

As Berstein [Ber09] wrote: "Imagine that it's fifteen years from now and someone announces the successful construction of a large quantum computer. The New York Times runs a front-page article reporting that all of the public-key algorithms used to protect the Internet have been broken. Users panic."

In this apocalyptic "post-quantum" world, we suppose that we already have a big enough quantum machine, hence we can solve the factorization and the discrete logarithm problems in polynomial time. But what does it mean exactly to cryptography?

It is an accepted practice to use hard problems (that presumably cannot be solved in polynomial time) as assumptions and use them to postulate the hardness of a public key encryption scheme. The current widely used methods ("classic methods") like RSA, DSA or ECDSA are based on problems that can be solved by Shor's quantum algorithm [Sho94] in polynomial time (Shor's algorithm needs a bigger quantum machine than what we can build today).

What will we be able to do if that world comes? The classic schemes will be broken, but we can use other types of methods which remain secure against quantum machines. Several such schemes have been proposed but most such schemes are not considered efficient enough for practical purposes, so we have to give novel algorithms and/or optimize the existing ones to reach an acceptable scheme.

In: E. Vatai (ed.): Proceedings of the 11th Joint Conference on Mathematics and Computer Science, Eger, Hungary, 20th – 22nd of May, 2016, published at http://ceur-ws.org

In this paper we follow this idea and approach for public key encryption to extend the possibilities for constructions. Our aim is to open a new way, but admittedly this method is still not efficient enough for real life internet usage.

We note that post-quantum cryptography is not to be confused with quantum cryptography. In post-quantum cryptography the schemes use only classic computers but are assumed to be secure against algorithms run on a quantum machine; whereas in quantum cryptography we use quantum techniques, i.e. physical quantum phenomena for encryption.

In our approach we combine a well-studied lattice problem with a problem from graph theory to obtain a new scheme.

1.1 Overview

In the remaining part of this section we briefly describe the hard problems used.

In Section 2 we describe prior work which led to the current state of the art and define a building block (encryption method) from an already existing public key encryption scheme, finally discuss the correctness and validity.

Section 3 gives the proposal of the new key generation method for the given encryption method and we discuss the security of the key.

Finally in Section 4 we discuss the key security of the new scheme.

1.2 Learning parity with noise (LPN)

A popular area in post-quantum cryptography is lattice-based cryptography, introduced by Ajtai in 1996 [Ajt96]. Here, lattices are used for building an encryption scheme. A widely studied lattice-based problem is the closest vector problem (CVP) where we have a vector (not necessary on the lattice) and a lattice, and we have to find the closest lattice point and determine that with the given basis. In small dimension it is an easy problem, but in higher dimensions it gets hard – NP-hardness results exist proving that random instances of the problem are basically as hard as the worst case instances.

Outside theoretical computer science, the linear code decoding problem is closely related to the CVP, if the lattice is the codeword space and the given vector is the transmitted message with some noise. This problem is also considered hard.

A further closely related problem is the learning with errors (LWE) lattice problem introduced in [Reg09]. In the last few years it has been shown to have nice post-quantum cryptographic uses (summarized by Regev in 2010 [Reg10]) one of which is the following inversion problems which is hard under certain assumptions:

Given $M \in \mathbb{F}_q^{m \times n}$ and $s \in \mathbb{F}_q^n$, a secret value, we have to determine s from $Ms + e$ where e is some small error.

Two of the most promising candidates today for post-quantum cryptography are lattice-based schemes: NTRU [HPS98] and Ring-LWE [LPR10,DXL12,GLP12,Pei14].

In the special case where $q = 2$ we use only binary values in LWE.

The learning parity with noise problem is a well-studied problem widely considered appropriate for constructing cryptography schemes. It has the big advantage that the calculation $(Ms + e)$ is a lot cheaper than the classic solutions, but inversion is hard. For some uses and more details, see [Mel13].

1.3 Clique problem

The clique problem is a basic problem from graph theory where given an input graph and an integer k, we have to decide whether the graph contains a complete subgraph (clique) of size k. A classical result is the following (we give quantitative formal hardness results later).

It is hard to find a clique (complete subgraph) of size k in a random graph.

The main advantage of using this problem is that it is an NP-complete problem, so if we can reduce breaking our encryption scheme to this hard problem, the attacker will also fight against the millennium problem "P versus NP".

In the post-quantum world there is a quantum algorithm that solves NP-complete problems: Grover's algorithm [Gro96]. This algorithm gives a reasonable speedup compared to known classical algorithms but still doesn't pose a threat to NP-hard problems in general.

The proposed encryption method is not the first one using the clique problem for constructing a scheme, see e.g. [Kuc91,JP00].

1.4 Some notation

In the following subsection we give some basic notations to avoid confusion with the most widely used notations in graph and probability theory.

- $[n] := \{1, ..., n\} \quad \forall n \in \mathbb{N}$
- a is chosen randomly from A.
- U_n uniform distribution over binary vectors of length n.
- Ber_ε^n distribution binary $0-1$ vectors of length n where each entry is 1 independently with probability ε.
- $G(n,p)$ distribution of n sized random graphs where every two nodes are adjacent with probability p (Erdős–Rényi random graphs).
- $\Pi_G(u)$ set of the neighbors of node u in G.

2 Prior work used directly

In the first section we could see the origin of our main problems, but now we describe the encryption algorithm (based on the LPN) which serves as the base of our scheme. The algorithm was introduced in [ABW10] and we can see it is a one-bit encryption method, but it can be easily extended to longer messages (make sure of the independent randomness of x and e).

2.1 The encryption method

The encryption method

Public key

$M \in \mathbb{F}_2^{m \times n}$ sampled from D. The distribution D is over matrices, in which the last row is a linear combination of $q-1$ other rows. (q must be small compared to m and n)

Private key

A q sized subset $S \subset [m]$ with $\sum_{i \in S} M_i$ where M_i denotes the i-th row of M.

Encryption

Choose a random vector $x \leftarrow U_n$ and a random noise vector $e \leftarrow Ber_\varepsilon^m$, let $b = Mx + e$.

- To encrypt 0, send the vector b.
- To encrypt 1, send the vector b with its last bit flipped .

Decryption

To decrypt $y \in \{0,1\}^n$, output the $\sum_{i \in s} y_i \pmod 2$.

2.2 Validity and correctness

Denote the sent message with $Mx + e + \sigma$, where the σ is zero vector if the message 0, and $\begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$ if 1.

Clearly the explained scheme is valid, because the Mx product is linearly dependent on the position corresponding to S.

$$\sum_{i \in S} y_i = \sum_{i \in S} (Mx + e + \sigma)_i = \sum_{i \in S} (Mx)_i + \sum_{i \in S} e_i + \sum_{i \in S} (\sigma)_i = 0 + \sum_{i \in S} e + \sigma$$

Error occurs in the decryption only with at most

$$\alpha = \frac{1}{2} - \frac{1}{2}(1 - 2\varepsilon)^q < \varepsilon q$$

probability by Lemma 5.3 from [ABW10].

Remark 1. We have to keep q small to achieve a good success ratio for transmission.

2.3 Key generation

Theorem 7.3 from [ABW10] proves the existence of a semantically secure public key encryption scheme under some hardness assumptions using a distribution of d-sparse matrices that is computationally indistinguishable from the uniform distribution of d-sparse matrices.

3 Proposed public key encryption scheme

In this section we propose a public key encryption scheme using the encryption method from [ABW10] described in the preceding section. The scheme has a key generation algorithm based on the NP-complete clique problem.

Our approach uses Erdős–Rényi random graphs for creating a public key – private key pair. Our aim is to inject a small linearly dependent part into the adjacency matrix of the graph and make sure it is hard to find that part.

3.1 Key generation

The key generation parameters:

- n - graph size
- p - edge probability (determines the density of the graph)
- k - planted clique/independent set size
- p_{add} - ratio that determines how to change the outer connection number to even (probability for adding)

Denote the nodes of G graph with V and the edges with E.

Algorithm 1 The key generation

1. Choose a random $G \leftarrow G(n, p)$ graph.
2. Choose a random k sized subset from the nodes of the graph containing the last row. Denote it with $S \subset [n]$.
3. Remove all edges between nodes contained in S: replace E by $(E \setminus \{(u, v) \,|u, v \in S\})$ (plant an independent set to the positions corresponding to S).
4. Iterate through $\{u \in V \setminus S| \,|\Pi_G(u) \cap S| \equiv 1 \pmod 2\}$ with u in random order
   (a) with p_{add} probability add (u, v) for $v \leftarrow S \setminus \Pi_G(u)$ to E,
   (b) else remove (u, v) for $v \leftarrow \Pi_G(u)_{|S}$ from E.

Public key: G
Private key: S

Steps 1 to 3 create a graph with a planted independent set in the position denoted by S. Step 4 adds and removes random edges to until we make sure that every node has an even number of connections to S. This latter property guarantees that this generation algorithm gives the (mod 2) linearly dependent part in the adjacency matrix that we need for the encryption scheme.

3.2 Expected degree

To achieve a graph that "looks" similar to an original $G(n,p)$ instance, we make sure that the expected degree in the generated graph equals to the expected degree of a graph instance from $G(n,p)$. To achieve this, we have to add similar number of random edges as we remove during the key generation. Using this observation we can give a formula for the p_{add} variable.

Theorem 2. *The graph generated by our key generation algorithm has similar expected degree as $G(n,p)$ if and only if*

$$p_{add} = \frac{p\binom{k}{2}}{2 \cdot p_{odd}(n-k)} + \frac{1}{2}$$

where

$$p_{odd} = \sum_{i=0}^{\lceil \frac{k}{2} \rceil - 1} \binom{k}{2i+1} p^{2i+1} (1-p)^{k-2i-1}.$$

Proof. The expected number of removed edges with planting the independent set: $p\binom{k}{2}$.

Denote with p_{odd} the probability that a node from $V \backslash S$ has an odd number of edges to the nodes corresponding in S:

$$p_{odd} = \sum_{i=0}^{\lceil \frac{k}{2} \rceil - 1} \binom{k}{2i+1} p^{2i+1} (1-p)^{k-2i-1}$$

In the key generation algorithm after the planting we check all of the nodes outside the clique ($V \backslash S$) and calculate the parity. If the connection number to the S is odd, we have to add or remove one edge to make the connection number even, otherwise we do not do anything, hence the expected number of edges that are added is

$$p_{odd}(n-k)(p_{add} - (1-p_{add}))$$

To achieve a similar expected degree as a normal random $G_{n,p}$ graph, we have to add as many edges as we removed (in expectation). This gives a formula for p_{add} :

$$p_{odd}(n-k)(p_{add} - (1-p_{add})) - p\binom{k}{2} = 0$$

$$p_{add} = \frac{p\binom{k}{2}}{2 \cdot p_{odd}(n-k)} + \frac{1}{2}$$

□

3.3 Parameter bound

We have a restriction for the parameters because clearly $0 \leq p_{add} \leq 1$ has to hold.

Theorem 3. *For the parameters of a key generation the following has to hold:*

$$\frac{p\binom{k}{2}}{p_{odd}} + k \leq n$$

where p_{odd} is like above in Theorem 2.

Proof. Plug $0 \leq p_{add} \leq 1$ into the formula from Theorem 2.

$$0 \leq \frac{p\binom{k}{2}}{2 \cdot p_{odd}(n-k)} + \frac{1}{2} \leq 1 \Rightarrow -1 \leq \frac{p\binom{k}{2}}{p_{odd}(n-k)} \leq 1 \Rightarrow -(n-k) \leq \frac{p\binom{k}{2}}{p_{odd}} \leq n-k$$

which directly leads to the given bound.

□

4 Cryptanalysis

In this section we discuss the hardness of computing the private key from public key without message sending. In cryptography it is ordinary to prove the security with some deduction to a well-known problem, so first of all we deduct our key attacking problem to the well-known clique problem via Karp reductions. Next we focus on the distribution $G(n,p)$, and finally discuss some attacking algorithms.

4.1 NP-hardness of finding the secret

In this subsection we discuss the hardness of finding an independent set S in a graph to which every node has an even number of connections. If one could find such sets S efficiently, this would enable breaking the encryption scheme above.

Definition 4. The independent set problem consists of deciding the following property
Input: graph G, positive integer k.
Property: G has k nodes which do not have edges between each other.

Lemma 5. *The independent set problem is NP-complete.*

Definition 6. Even neighbor independent set problem
Input: graph G, positive integer k.
Property:
1) G has a set of k nodes (denoted by S) which don't have edges between them.
2) Each node outside S has an even number of neighbors from S.

Theorem 7. *The even neighbor independent set problem is NP-hard.*

Proof. Using Karp reduction to the independent set problem:
Let $(V,E) := G$
$V' = G \times \{0,1\}$
$E' = \{((u,0),(v,0)) \mid (u,v) \in E\} \cup \{((u,1),(v,1)) \mid (u,v) \in E\} \cup \{((u,0),(v,1)) \mid (u,v) \in E\}$
$G' := (V', E')$
$k' = 2k$

□

If one had an efficient algorithm for the search problem version of the even neighbor independent set problem, one could also solve the search problem version of the clique problem.

1) We have a k-sized independent set problem with G graph and $k, n = |G|$.

2) Use the transformation in Theorem 7 to get G'.

3) Use the oracle on G' which solves even neighbor independent set problem in polynomial time and gets solution S.

4) The original independent set position is $\{s' | s' \equiv s \bmod n, s \in S\}$

4.2 Hardness of finding clique in $G(n,p)$

In the previous subsection we only argued that we cannot expect to be able to efficiently solve the even neighbor independent set problem in the worst-case. In order to formally prove the security of the proposed scheme, one would have to prove that finding cliques in graphs obtained as the public key of this scheme is at least as hard as finding cliques in general random graphs. We were not (yet) able to prove this, but below we give heuristic arguments about why this might be the case. The analysis below helps us find the optimal parameters for our encryption, therefore we discuss the potential attacking algorithms and related studies.

The simple brute force approach needs to check all the combinations of k nodes and check whether every node adjacent with each other. In worst case after we found an independent set we have to check the even connections from outside property.

The second intuitive approach is the greedy one, which has a lot of good improvement and all of them are based on the following two base steps:

1. Choose a random node.
2. Choose a new node which is adjacent with all of the nodes selected before.

We revise some relevant results from the literature on the clique problem.

- The following modified version of the clique problem was investigated by Karp [Kar72], [Kar76]: Is there any polynomial time algorithm that finds a $k = (1+\varepsilon) \cdot lg(n)$ sized clique for any constant $\varepsilon > 0$?
- In [Mat72], it is proved that the expected size of the maximal clique is $2 \cdot log_{1/p}(n) - 2 \cdot log_{1/p}\left(log_{1/p}(n)\right) + 2 \cdot log_{1/p}\left(\frac{1}{2}e\right) + 1$ in the Erdős–Rényi model with parameter p. Many papers consider the $p = 1/2$ case and the approximation $2 \cdot lg(n)$.
- In [GM75,Kar76], it is proved that in a uniformly generated graph, the greedy algorithm outputs a clique of size $lg(n)$ with high probability. By running the greedy algorithm for $n^{k/4+o(1)}$ times, we find a $\left(\frac{1}{2}+\varepsilon\right)k$ sized clique also with high probability.
- The original question hardened by Jerrum [Jer92]: Is there a polynomial time algorithm that finds a clique with $k \geq 1.01 \cdot log(n)$ in a random graph which contains a clique of size at least $n^{0.49}$? In this paper they showed that some search techniques fail to efficiently find a clique of size $(1+\varepsilon) \cdot log(n)$ in a graph $G\left(n, \frac{1}{2}\right)$ even if it is known to contain a clique of size $n^{\frac{1}{2}-\delta}$.
- The $G(n,p)$ Erdős-Rényi random graphs are well-studied (e.g. [Bol98,Wes01]) especially the uniform case $G\left(n, \frac{1}{2}\right)$.
- [AKS98] gave an algorithm which solves the planted clique problem in polynomial time with high probability for $k > cn^{0.5}$ cases which have some improvements [DGP14,FR10], but only the probability is improved, the lower bound not.
- In [NP85] the runtime is $n^{(\omega/3)k+o(1)}$, where $n^{\omega+o(1)}$ is the runtime of the matrix multiplication. The lowest possible value (we have not reached that yet) of ω is 2, so if we have the theoretically best matrix multiplication algorithm, this algorithm gives a $n^{(2/3)k+o(1)}$ runtime. In practice this algorithm is inapplicable for our cases, because asymptotically fast matrix multiplication algorithms are not practical for the problem sizes we consider.
- [Vas09] gave an algorithm with $O\left(n^k / \left(\varepsilon \cdot log(n)^{k-1}\right)\right)$ runtime and $O\left(n^{\varepsilon}\right)$ space requirement.
- Several slight improvements appear in the following papers: [Rob01,BSK10,Ros14].
- Rossman [Ros14] analyzed Boolean circuits bounds for average case clique complexity and prooved that the boolean circuits of size $O\left(n^{k/4}\right)$ and depth at most $k^{-2} \cdot log(n) / log(log(n))$ cannot solve the k-clique problem with high probability on $G(n,p)$.

Cryptosystems:

- Kucera [Kuc91] started to use the clique problem for cryptographic purposes in 1991, they used $k = n^{\kappa}$ clique size, where $0 < \kappa < \frac{1}{2}$.
- [JP00] used the planted clique problem to construct an encryption scheme, but none of them proposed a public key encryption scheme. They proved it is hard to claim a planted clique from an uniform Erdős-Rényi graph $(G\left(n, \frac{1}{2}\right))$

[Ros10] raised a question in 1.3: Is there an $O\left(n^{(1-\delta)k/4}\right)$ time algorithm which solves the k-clique problem with high probability on $G(n,p)$? If we have a negative answer for it, that would imply $P \neq NP$. This question is still not answered.

We can find a lot of algorithms solving the k-clique or maximal clique problems but still there are not any efficient algorithms even for Karp's question on a clique of size slightly above $\lg n$.

Remark 8. The most frequently considered case is the uniform $G\left(n, \frac{1}{2}\right)$ distribution, and the interval considered hard for the clique size $log(n) < k < \sqrt{n}$. If we use this interval as a boundary for the key generation parameters, the attackers do not have any publicly known efficient algorithms.

4.3 Independent set size

We saw in the previous section that the hardness of the decryption strongly depends on the size of the independent set. It is clear, we have to use at least a $log(n)$ sized independent set as a secret key, but that is an open question which upper bound would be better for us.

It was shown that the expected maximum clique size is $k \approx 2 \cdot log(n)$, so in the remaining part of this paper we investigate only graphs with $k = 1.95 \cdot log(n)$ sized injected independent sets.

4.4 Distribution

We could not formally prove that the distribution of graphs obtained as a public key of our scheme is computationally indistinguishable from the set of all Erdős–Rényi graphs. To compare the distribution of our key generation to the uniform $G(n,p)$ we used the Wilcoxon rank-sum to measure the difference between the two distribution and it shown a really good p-value for the typical properties like diameter, degree, average maximum clique size etc.

4.5 Experimental result

As part of the research we implemented our cryptosystem in sage and compared to the classical RSA security level with some experimental tests. Our aim was to break the cryptosystems with a publicly available algorithm. We tried many implementations of the best algorithms against both of problems (RSA: Pollard Rho, Pollard p-1, field sieve , Clique/independent set: [Akk73,BK73,TIAS77,CN85,MU04,TTT06]) and the final choice against the RSA was the Pari/GP computer algebra system (It was faster than the publicly available sieve based on gmp in c++, polard, etc.), and the Tomita algorithm [TTT06] against our cryptosystem (which was also faster than the famous Bron–Kerbosch algorithm in our cases).

In our measurements we used the $k = 1.95 \cdot log(n)$ clique size, because naturally the expected maximum clique size is $2 \cdot log(n)$, so our graph will be more similar to the uniform distribution.

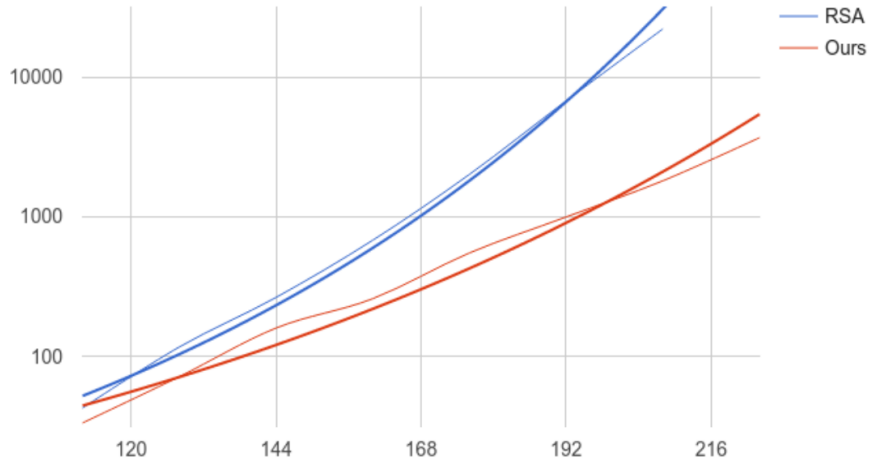

Figure 1: Compare RSA and clique on logarithmic scale.

Our experimental results show that we can use nearly similar n parameter (needed little bit bigger) as used for the RSA, so we can claim an acceptable encryption if we use the following parameter set.

$n \geq 1024$
$k = 1.95 \cdot log\,(n) \approx 14$
$p = 0.5$
$p_{add} = 0.545$

5 Result and conclusion

We proposed a public key encryption scheme whose security is based on the difficulty of the clique problem and we expect it to be secure against quantum algorithms as well. Yet there are several improvements to be made.

Currently, our key size is $\frac{n^2}{2}$, and the encrypted size / message size ratio is n, so during further development we plan to improve the algorithm to get a better ratio, because now it equals to the graph size, which is a really huge overhead to the encrypted message. The big key size can be negligible in practice, because it has to be transfered only once and can be cached in middle points on the internet.

By research of Jerrum [Jer92] we will investigate the $2 \cdot log\,(n) < k < \sqrt{n}$ interval for secret sizes, because as we can see in previous sections, the hardness of finding an independent set exponentially grows together with the size of the independent set. We expect to supplement present work with additional formal proofs of security and perform more extensive experiments in the future.

References

[ABW10] B. Applebaum, B. Barak, A. Wigderson. Public-key cryptography from different assumptions, *In Proceedings of the forty-second ACM symposium on Theory of computing* 171–180, 2010.

[Ajt96] M. Ajtai. Generating hard instances of lattice problems, *In Proceedings of the twenty-eighth annual ACM symposium on Theory of computing* 99–108, 1996.

[Akk73] E. Akkoyunlu. The enumeration of maximal cliques of large graphs, *In SIAM Journal on Computing* 2(1):1–6, SIAM, 1973.

[AKS98] N. Alon, M. Krivelevich, B. Sudakov. Finding a large hidden clique in a random graph, *In Random Structures and Algorithms* 13(3-4):457–466, 1998.

[Ber09] D. J. Bernstein. Introduction to post-quantum cryptography, *Chapter in Post-quantum cryptography* 1–14, Springer, 2009.

[BK73] C. Bron, J. Kerbosch. Algorithm 457: finding all cliques of an undirected graph, *In Communications of the ACM* 16(9):575–577, ACM, 1973.

[Bol98] B. Bollobás. Random graphs, *Chapter in Modern Graph Theory* 215–252, Springer, 1998.

[BSK10] S. Balaji, V. Swaminathan, K. Kannan. A simple algorithm to optimize maximum independent set, *In Advanced Modeling and Optimization* 12(1):107–118, 2010.

[CN85] N. Chiba, T. Nishizeki. Arboricity and subgraph listing algorithms, *In SIAM Journal on Computing* 14(1):210–223, SIAM, 1985.

[DGP14] Y. Dekel, O. Gurel-Gurevich, Y. Peres. Finding hidden cliques in linear time with high probability, *In Combinatorics, Probability and Computing* 23(01):29–49, Cambridge Univ Press, 2014.

[DXL12] J. Ding, X. Xie, X. Lin. A Simple Provably Secure Key Exchange Scheme Based on the Learning with Errors Problem., *In IACR Cryptology ePrint Archive* 2012:688, 2012.

[FR10] U. Feige, D. Ron. Finding hidden cliques in linear time, *In 21st International Meeting on Probabilistic, Combinatorial, and Asymptotic Methods in the Analysis of Algorithms (AofA'10)* 189–204, 2010.

[GLP12] T. Güneysu, V. Lyubashevsky, T. Pöppelmann. Practical lattice-based cryptography: A signature scheme for embedded systems, *In International Workshop on Cryptographic Hardware and Embedded Systems* 530–547, 2012.

[GM75] G. R. Grimmett, C. J. McDiarmid. On colouring random graphs, *In Mathematical Proceedings of the Cambridge Philosophical Society* 77(02):313–324, 1975.

[Gro96] L. K. Grover. A fast quantum mechanical algorithm for database search, *In Proceedings of the twenty-eighth annual ACM symposium on Theory of computing* 212–219, 1996.

[HPS98] J. Hoffstein, J. Pipher, J. H. Silverman. NTRU: A ring-based public key cryptosystem, *In International Algorithmic Number Theory Symposium* 267–288, 1998.

[Jer92] M. Jerrum. Large cliques elude the Metropolis process, *In Random Structures & Algorithms* 3(4):347–359, Wiley Online Library, 1992.

[JP00] A. Juels, M. Peinado. Hiding cliques for cryptographic security, *In Designs, Codes and Cryptography* 20(3):269–280, Springer, 2000.

[Kar72] R. M. Karp. Reducibility among combinatorial problems, *Chapter in Complexity of computer computations* 85–103, Springer, 1972.

[Kar76] R. M. Karp. The probabilistic analysis of some combinatorial search algorithms, *In Algorithms and complexity: New directions and recent results* 1:19, Academic Press, New York, 1976.

[Kuc91] L. Kučera. A generalized encryption scheme based on random graphs, *In International Workshop on Graph-Theoretic Concepts in Computer Science* 180–186, 1991.

[LPR10] V. Lyubashevsky, C. Peikert, O. Regev. On ideal lattices and learning with errors over rings, *In Annual International Conference on the Theory and Applications of Cryptographic Techniques* 1–23, 2010.

[Mat72] D. W. Matula. Employee party problem, *In Notices of the American Mathematical Society* 19(2):A382–A382, 1972.

[Mel13] L. Melis. On the Learning Parity with Noise Problem, *In* , 2013.

[MU04] K. Makino, T. Uno. New algorithms for enumerating all maximal cliques, *In Scandinavian Workshop on Algorithm Theory* 260–272, 2004.

[NP85] J. Nešetřil, S. Poljak. On the complexity of the subgraph problem, *In Commentationes Mathematicae Universitatis Carolinae* 26(2):415–419, Charles University in Prague, Faculty of Mathematics and Physics, 1985.

[Pei14] C. Peikert. Lattice cryptography for the internet, *In International Workshop on Post-Quantum Cryptography* 197–219, 2014.

[Reg09] O. Regev. On lattices, learning with errors, random linear codes, and cryptography, *In Journal of the ACM (JACM)* 56(6):34, ACM, 2009.

[Reg10] O. Regev. The learning with errors problem, *In Invited survey in CCC*, 2010.

[Rob01] J. M. Robson. Finding a maximum independent set in time O (2n/4), , Technical report, Technical Report 1251-01, LaBRI, Université Bordeaux I, 2001.

[Ros10] B. Rossman. Average-Case Complexity of Detecting Cliques, ProQuest LLC, *In Ann Arbor, MI*, 2010.

[Ros14] B. Rossman. The monotone complexity of k-clique on random graphs, *In SIAM Journal on Computing* 43(1):256–279, SIAM, 2014.

[Sho94] P. W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring, *In Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on* 124–134, 1994.

[TIAS77] S. Tsukiyama, M. Ide, H. Ariyoshi, I. Shirakawa. A new algorithm for generating all the maximal independent sets, *In SIAM Journal on Computing* 6(3):505–517, SIAM, 1977.

[TTT06] E. Tomita, A. Tanaka, H. Takahashi. The worst-case time complexity for generating all maximal cliques and computational experiments, *In Theoretical Computer Science* 363(1):28–42, Elsevier, 2006.

[Vas09] V. Vassilevska. Efficient algorithms for clique problems, *In Information Processing Letters* 109(4):254–257, Elsevier, 2009.

[Wes01] D. B. West. Introduction to graph theory, 2, Prentice hall Upper Saddle River, 2001.