Hiding Cliques for Cryptographic Security

Ari Juels* Marcus Peinado†

Abstract

We demonstrate how a well studied combinatorial optimization problem may be introduced as a new cryptographic function. The problem in question is that of finding a "large" clique in a random graph. While the largest clique in a random graph is very likely to be of size about $2\log_2 n$, it is widely conjectured that no polynomial-time algorithm exists which finds a clique of size $\geq (1+\epsilon)\log_2 n$ with significant probability for any constant $\epsilon > 0$. We present a very simple method of exploiting this conjecture by "hiding" large cliques in random graphs. In particular, we show that if the conjecture is true, then when a large clique – of size, say, $(1+2\epsilon)\log_2 n$ – is randomly inserted ("hidden") in a random graph, finding a clique of size $\geq (1+\epsilon)\log_2 n$ remains hard. Our result suggests several cryptographic applications, such as a simple one-way function.

1 Introduction

Many hard graph problems involve finding a subgraph of an input graph $G = (V, E)$ with a certain property (e.g., cliques, Hamiltonian cycles). In some cases, it may be demonstrated that randomly generated input graphs almost always contain a subgraph with the specified property, and yet no polynomial-time algorithm is known which finds such subgraphs with non-negligible probability. This is the case for the clique problem which we consider in this paper. A *clique* of size k in a graph $G = (V, E)$ is a complete subgraph on k nodes, i.e., a set of k nodes such that every pair is connected by an edge. The clique problem is that of finding a clique of size k (for some suitably large k) in a graph G. It is conjectured that finding "large" cliques in a graph G selected uniformly at random is hard.

A number of cryptographic primitives depend on the availability of "solved" hard random problem instances. Most depend on the assumed hardness of only two problems from number theory: factoring and discrete logarithm. The question of whether combinatorial problems might serve as an alternative has been addressed several times, as in [11, 24]. In this paper we describe a straightforward probabilistic method of constructing a hard instance of the clique problem. The method is simple. Our main contribution in this paper is a proof that, under a widely believed conjecture, our method does indeed yield hard problem instances, and can therefore be used in cryptographic applications.

We generate a solved instance of the clique problem as follows. We select a random graph G uniformly at random and randomly embed ("hide") in it a clique K. In particular, we select k nodes at random from G (where k is a suitable size, as explained later), and create a clique on those nodes, i.e., we complete the induced subgraph.

Even though it is conjectured that finding "large" cliques in a graph G selected uniformly at random is hard, it is not obvious that the problem remains hard after a clique is embedded in G. This is because the distribution of graphs induced by embedding a clique is no longer uniform. For example, when $k \geq n^{1/2+\epsilon}$ ($n = |V|, \epsilon > 0$), it is easy to find the hidden clique. (See also [1] in this volume.) Our aim is to show that k can be chosen such that this is not the case. Our main result shows for suitable k that finding the hidden clique – or any other clique in G with k nodes – is at least as hard as finding a clique with k nodes under the uniform distribution. We believe that this is the first rigorously justified application of a graph-based problem to cryptography.

From an asymptotic point of view, the clique problem on random graphs offers only relatively weak security. Under the commonly used random graph distributions each edge exists with probability 1/2, and the size of the largest clique is less than $2\log n$ with high probability. Thus, a clique of size $k < 2\log n$ can be found by exhaustive search, i.e. by considering all sets of k vertices, in time $2^{O(\log^2 n)}$. This growth rate is superpolynomial in n. However, it is small compared to the time taken by even the most efficient attacks on problems like factoring or discrete logarithm ($2^{\Theta(n^\epsilon)}$ for some $\epsilon > 0$).

*RSA Laboratories, 20 Crosby Dr., Bedford, MA 01730, ari@rsa.com. The author performed much of this work at the University of California at Berkeley under a UC Regents Fellowship and NSF Grant CCR-9505448.

†Institute for Algorithms and Scientific Computing, German National Research Center for Information Technology (GMD), 53754 Sankt Augustin, Germany, peinado@gmd.de. Supported in part by DFG grant SFB 408. A portion of this research was conducted while the author was visiting the International Computer Science Institute, Berkeley, CA.

The relatively low complexity of finding a large clique is a consequence of the fact that the size of the largest clique is only logarithmic in n, which, in turn, is a consequence of the fact that edges exist with constant probability. Clique sizes (and with them the complexity of the attack just described) will increase if the edge probability is increased. Our analysis can be extended to edge probabilities of $1 - 1/f(n)$ where $f(n) = \omega(1)$. Depending on the choice of $f(n)$, cliques will become as large as $\Omega(n^\epsilon)$ $(\epsilon > 0)$. We defer a description of this analysis to the final version and concentrate on the standard case (constant edge probabilities) in this extended abstract.

The remainder of this abstract is organized into four sections. In Section 2, we describe results in the literature relating to the problem of finding large cliques in graphs. In Section 3, we describe and prove the main theorem of this paper, stating that large hidden cliques are as hard to find as large cliques in graphs generated uniformly at random. In Section 4, we discuss some issues surrounding the application of this result to cryptography. We conclude in Section 5 with a brief discussion of possible further avenues of research.

2 Related Work

2.1 Finding large cliques

The problem of determining the size of the largest clique in a graph is one of Karp's original NP-complete problems [22]. In recent years, there has been a sequence of results [3, 4, 6, 7, 5, 21, 20] showing that it is hard to find even an approximate solution. Currently, the strongest results are due to Håstad. Let $\omega(G)$ denote the size of the largest clique in G, and let $n = |V|$. Håstad [21] shows that unless P=NP, no polynomial-time algorithm can find a clique whose size is within a factor of $n^{\frac{1}{3}-\delta}$ of $\omega(G)$ (for any constant $\delta > 0$). Under the assumption NP$\neq$ co-R, the result holds even for a factor of $n^{1-\delta}$ [20].

These results apply to the worst case. Cryptographic applications, however, depend on the average case difficulty of the problem. No comparable hardness results are known in the average case. In general, in contrast to classical (worst case) NP-completeness, only a very small number of problems are known to be average-case complete [25]. Nevertheless, the average-case clique problem has prompted considerable experimental and theoretical interest. Most of the research in this vein focuses on the Erdös-Renyi random graph model $\mathcal{G}_{n,p}$ $(0 \leq p \leq 1)$ over graph instances containing n nodes. A graph G may be drawn from this distribution by inserting each of the $\binom{n}{2}$ possible edges into G independently with probability p. The most frequently considered case is $p = 1/2$, i.e., the uniform distribution. For the overwhelming majority of such graphs, the largest clique is of size $2\log_2 n - O(\log\log n)$ [9]. Smaller cliques exist in abundance: for $k = c\log_2 n$, where $0 < c < 2$ constant, the expected number of cliques of size k is $n^{\Omega(\log n)}$. It is easy to find cliques of size up to $\log_2 n$ in expected polynomial time using a randomized greedy algorithm.

The many attempts at designing polynomial-time algorithms which find larger cliques in random graphs have met with no success. It is now widely conjectured that for any constant $\epsilon > 0$, there does not exist a polynomial-time algorithm capable of finding cliques of size $(1+\epsilon)\log_2 n$ with significant probability in random graphs. Karp [23] first issued the challenge of finding such an algorithm twenty years ago. Jerrum [17] considerably extended this challenge in calling for a randomized, polynomial-time algorithm capable of finding a clique of size $1.01\log_2 n$ with high probability over random graphs containing a clique of size $n^{0.49}$. In support of the difficulty of finding such an algorithm, Jerrum demonstrates the existence of an initial state from which the Metropolis algorithm, a fixed-temperature variant of simulated annealing, cannot find a clique of size $(1+\epsilon)\log_2 n$ for any constant $\epsilon > 0$ in expected polynomial time. He shows, moreover, that this situation holds even when a clique of size as large as $n^{1/2-\delta}$ for constant $\delta > 0$ is randomly inserted ("hidden") in randomly-generated graphs. Similar results have been shown for randomized versions of the algorithm of Boppana and Halldórsson [27, 28]. Moreover, a number of experimental studies seem to confirm the hardness of the problem of finding large cliques in random graphs. A survey of these may be found in [18].

2.2 Embedded graph structures

An application of the clique problem to cryptography has been considered by Kučera [24]. He defines the concept of *generalized encryption scheme* and uses the clique problem (more precisely, the independent set problem) to implement it. The graphs are random graphs from $\mathcal{G}_{n,p}$ with one embedded clique of size $k = n^\kappa$ $(0 < \kappa < 1/2)$. Kučera does not give a rigorous proof of the security of his scheme.

Broder et al. [11] investigate a similar idea for a different problem: Hamiltonian Cycle. They embed a Hamiltonian cycle into $G_{n,p}$ (for appropriate p) and describe an algorithm which finds the embedded cycle in polynomial time, thus showing that their scheme is inappropriate for cryptographic purposes. Hamiltonian Cycle, however, can be solved in linear time on average [16]. In contrast, no average-case polynomial-time

algorithm is known for the clique problem.

Input distributions involving embedded (or hidden) structures have been considered in different contexts for a variety of problems [12, 10, 8, 13, 13, 19, 29, 30].

3 Notation and Proof

3.1 Notation

Recall that our aim is to show that finding a "large" clique in a random graph into which a large clique has been embedded is as hard as finding a "large" clique in a random graph. Let p denote the uniform distribution $\mathcal{G}_{n,1/2}$ over graphs with n nodes. Let p'_k denote the distribution obtained as follows: select a graph $G = (V, E)$ from p, and then form a clique on k nodes selected uniformly at random from V. We refer to a clique formed in this manner as a *hidden* clique. We shall show that when $k \leq (2-\delta)\log_2 n$ for any constant $\delta > 0$, finding a large clique in p'_k is as hard as finding one in p. More precisely, we shall show that if there exists an algorithm A which finds a clique of size $(1+\epsilon)\log_2 n$ in p'_k with probability $\frac{1}{q(n)}$, for some polynomial $q(n)$, then the same algorithm can find a clique of size $(1+\epsilon)\log_2 n$ in p with probability $\frac{1}{q'(n)}$ for some polynomial $q'(n)$. We will use the notation *poly* as an abbreviation for the phrase 'some polynomial in n'.

3.2 Sketch of proof

Given a graph G, let $C_k(G)$ denote the number of distinct (but possibly overlapping) cliques of size k in a specific graph instance G. If G is generated at random from distribution p, $C_k = C_k(G)$ is a random variable. Let $E_k = \mathrm{E}C_k$ denote its expectation, i.e. the expected number of k cliques in G.

Our proof will begin by demonstrating that when $C_k(G)$ is close to E_k, the probability of graph G in the distribution p'_k will be close to that in p. In other words, when the number of cliques in a graph G is close to the expected number E_k, the process of planting a clique of size $(2-\delta)\log_2 n$ in a random graph will yield G with probability similar to that of the process of simply generating a random graph. We shall then show that the variance of C_k is low. This will imply two things: first, that most graphs G are "good", i.e., for most graphs, $p'_k(G)/p(G)$ is less than a relatively small polynomial; second, that "bad" graphs, i.e., those for which $p'_k(G)/p(G)$ is large, will occupy a small fraction Δ of p'_k. In fact, we will be able to make this fraction Δ arbitrarily small. Therefore, an algorithm A which successfully locates a large clique in a $(\frac{1}{poly})$-fraction of graphs in p'_k must be locating such cliques in a set M of good graphs such that $p'_k(M) = \frac{1}{poly} - \Delta = \frac{1}{poly}$. Since graphs in M are good, $p'_k(M) = \frac{1}{poly}$ will imply $p(M) = \frac{1}{poly}$. Thus, A will successfully locate a large clique in a $(\frac{1}{poly})$-fraction of graphs in p, the uniform distribution over graphs.

3.3 Proof of main theorem

LEMMA 3.1.

$$p'_k(G) = \frac{C_k(G)}{E_k} p(G)$$

Proof. Selecting a graph from p'_k may be viewed as the process of selecting a graph G' from p and then planting a clique on a set K of k nodes chosen uniformly at random. In order for the resulting graph to be identical to G, it must be that the nodes K form a clique in G. An appropriate set K will thus be chosen with probability $C_k(G)/\binom{n}{k}$. It must also happen that the edges in G' which lie outside of K correspond exactly to those in G. More precisely, for all edges e not strictly contained in K, we require $e \in G' \iff e \in G$. This will occur with probability $2^{-\binom{n}{2}+\binom{k}{2}}$. Thus,

$$p'_k(G) = \frac{C_k(G)}{\binom{n}{k}} 2^{-\binom{n}{2}+\binom{k}{2}}. \tag{3.1}$$

The expected number of cliques in p is easily seen to be $\binom{n}{k}/2^{\binom{k}{2}}$. The definition of p implies that $p(G) = 2^{-\binom{n}{2}}$ for any graph instance G. Combining these two facts with (3.1) yields the lemma. □

Lemma 3.1 states that when the number of cliques in a graph instance G is close to its expectation over p, then $p'_k(G) \approx p(G)$. Our goal now is to show that for most graphs G, $p'_k(G)$ is only a polynomial factor larger than $p(G)$. For this we need to show that C_k is concentrated tightly around its mean E_k. We shall accomplish this by showing that the variance of C_k is small.

LEMMA 3.2. *Let $k = (2-\delta)\log_2 n$ for some constant $\delta > 0$. Then*

$$\mathrm{Var}[C_k] < n^6 E_k^2 \quad .$$

Proof. We employ the method of [9], Chapter XI, and consider pairs of cliques in G. This gives us

$$\mathrm{E}[C_k^2] = \sum_{i=0}^{k} \binom{n}{k}\binom{k}{i}\binom{n-k}{k-i} 2^{-2\binom{k}{2}+\binom{i}{2}} \quad , \tag{3.2}$$

and thus,

$$\frac{\mathrm{E}[C_k^2]}{\mathrm{E}^2[C_k]} = \sum_{i=0}^{k} \binom{n}{k}^{-1} \binom{k}{i} \binom{n-k}{k-i} 2^{\binom{i}{2}} \quad . \tag{3.3}$$

Let us denote the i^{th} term in the above sum by f_i. Clearly $f_0 < 1$. By employing the well-known bounds $\binom{n}{k} \leq (\frac{n\mathrm{e}}{k})^k$ and $\binom{n}{k} \geq (\frac{n}{k})^k$, we obtain for $i > 0$ the inequality

$$f_i \leq \left(\frac{k\mathrm{e}}{i}\right)^i \left(\frac{k}{n}\right)^k \left(\frac{(n-k)\mathrm{e}}{k-i}\right)^{k-i} 2^{\binom{i}{2}} \quad . \tag{3.4}$$

Algebraic manipulation shows that the above is equal to

$$\left(\frac{k^2}{i}\right)^i \left(\frac{(n-k)k}{k-i}\right)^{k-i} \mathrm{e}^k n^{-k} 2^{\binom{i}{2}} \quad ,$$

which is less than

$$\left(\frac{k}{k-i}\right)^{k-i} \mathrm{e}^k \left(\frac{k^2}{n}\right)^i 2^{\frac{i^2}{2}} \quad . \tag{3.5}$$

Let us first consider the quantity $(\frac{k}{k-i})^{k-i}$. This is equal to $(1 + \frac{i}{k-i})^{k-i} \leq \mathrm{e}^i$. Since $i \leq k = (2-\delta)\log_2 n$ for some constant $\delta > 0$, it follows that $(\frac{k}{k-i})^{k-i} = n^{2\log_2 \mathrm{e}} < n^{2.9}$. Similarly, it is also the case that $e^k < n^{2.9}$.

Now let us consider the quantity $\beta = (\frac{k^2}{n})^i 2^{\frac{i^2}{2}}$. Clearly, $\log_2 \beta = -i\log_2 n + i^2/2 + 2i\log_2 k$. Since $k = O(\log n)$, it follows that $\log_2 \beta < 0$ if $i < 2\log_2 n$. Since $i \leq k = (2-\delta)\log_2 n$ for some constant $\delta > 0$, it follows that $\beta < 1$ for all values of i. Tying together all of the above, we see that $f_i < n^{5.8}$ for all i, and therefore

$$\sum_{i=0}^{k} f_i = \frac{\mathrm{E}[C_k^2]}{\mathrm{E}^2[C_k]} < n^6 \tag{3.6}$$

for sufficiently large n. □

Remark: The bound on the variance is far from tight. A more detailed analysis shows that $\mathrm{Var}[C_k] \leq c^{\log^2 n} E_k^2$ for some c such that $0 < c < 1$, which depends only on k. This bound can be used to obtain a 'more efficient' version of Theorem 3.1. We omit further details in this extended abstract.

From the above lemma, it follows by Chebyshev's inequality that "bad" graphs, i.e., those graphs G such that $C_k(G) \gg E_k$, constitute a small fraction of p. As we see in the next lemma, when $k = (2-\delta)\log_2 n$ for some constant $\delta > 0$, such graphs also occupy a small fraction of p'_k.

Define the set Z of bad graphs to include those graphs G such that $C_k(G) > n^{2h} E_k$ for some constant $h > 0$. In other words, let

$$Z = \{\ G \mid C_k(G) > n^{2h} E_k\ \} \quad .$$

LEMMA 3.3.

$$p'_k(Z) = O(n^{-h+6})$$

Proof. We shall determine the probability $p'_k(Z)$ of the set of bad graphs by partitioning it into disjoint sets Z_j whose probability is more easily estimated. Let

$$Z_j = \{\ G \mid n^{jh} E_k < C_k(G) \leq n^{(j+1)h} E_k\ \} \quad .$$

Clearly, $Z = \bigcup_{j=2}^{\infty} Z_j$. By Lemma 3.2 (or eq. 3.6) and Chebyshev's inequality, $p(Z_j) < n^{-2jh+6}$. By Lemma 3.1 then,

$$p'_k(Z_j) < n^{(j+1)h} \left(\frac{1}{n^{2jh-6}}\right). \tag{3.7}$$

Since $Z = \bigcup_{j=2}^{\infty} Z_j$, it follows that

$$\begin{aligned} p'_k(Z) &= \sum_{j=2}^{\infty} p'_k(Z_j) \\ &< \sum_{j=2}^{\infty} \frac{n^{(j+1)h}}{n^{2jh-6}} \\ &= \sum_{j=2}^{\infty} n^{-jh+h+6} \\ &= n^{-h+6} \sum_{j=0}^{\infty} n^{-jh} \\ &= n^{-h+6} O(1) \\ &= O(n^{-h+6}) \quad , \end{aligned}$$

which proves the lemma. □

By making the constant h large enough – in other words, by making the graphs in Z sufficiently "bad" – we may make the set Z arbitrarily small. By making Z small, we ensure that an algorithm A which successfully finds cliques in p'_k does so principally on good graphs. These graphs will constitute a $(\frac{1}{poly})$-fraction of graphs in p, implying that A successfully finds cliques in p with probability $\frac{1}{poly}$.

THEOREM 3.1. *Suppose that $k \leq (2-\delta)\log_2 n$ for some $\delta > 0$. Suppose then that there exists a deterministic,*

polynomial-time algorithm A which finds a k-clique in graphs drawn from p'_k with probability $\frac{1}{q(n)}$, for some polynomial $q(n)$. Then there exists a polynomial $q'(n)$ such that A finds a k-clique in graphs drawn from p with probability $\frac{1}{q'(n)}$.

Proof. Suppose $\frac{1}{q(n)} = \Omega(n^{-j})$ for some constant j. Let Z be the set of graphs such that $C_k(G) > (n^{2(j+6+\epsilon)})E_k$ for some $\epsilon > 0$. By Lemma 3.3, $p'_k(Z) = O(n^{-j-\epsilon})$. Let Q denote the set of graphs G not in Z on which A finds a k-clique. Clearly, $p'_k(Q) = \Omega(n^{-j}) - p'_k(Z) = \Omega(n^{-j}) - O(n^{-j-\epsilon}) = \Omega(n^{-j})$. Therefore, by Lemma 3.1, $p(Q) = \Omega(n^{-j})(n^{-2(j+6+\epsilon)}) = \Omega(n^{-3j-12-2\epsilon})$. This proves the theorem. □

Remarks.

Observe that this theorem may be extended in a suitable fashion to randomized algorithms A. In particular, if A is a randomized algorithm which finds cliques in p'_k with probability $\frac{1}{poly}$ in expected polynomial time, then A also finds cliques in p with probability $\frac{1}{poly}$ in expected polynomial time.

The theorem also applies for random graphs generated with different edge densities, i.e., graphs drawn from $G_{n,p}$ for constant p. In general, the size of the large clique in a graph drawn from $G_{n,p}$ will be of size about $2\log_{1/p} n$.

Finally, Theorem 3.1 holds also for distributions p^c_k, where a graph from p^c_k is generated by randomly inserting any constant number of cliques $K_1, K_2, \ldots, K_c$ of size k in a random graph. In other words, it is possible to hide at least a constant number of large cliques in a random graph.

4 Cryptographic Applications

Assuming the conjectured hardness of finding large cliques in random graphs, Theorem 3.1 states that when a clique K of sufficiently large size – say, $3/2\log_2 n$ – is randomly inserted into a random graph G, yielding graph G', finding any large clique in G' is still hard. This result suggests several cryptographic applications.

A new one-way function: A *one-way function* is, informally, a function which is easy to compute on all elements in its domain, but with high probability hard to invert on a randomly selected element in its range [26]. Theorem 4 shows that these criteria are met by a one-way function f which simply hides a clique in a graph. More formally, the function f may be defined as follows: $f : \mathcal{G} \times \mathcal{K} \rightarrow \mathcal{G}$, where $\mathcal{G}$ is the set of graphs on n nodes, and $\mathcal{K}$ is the collection of all sets of k vertices (subsets of $\{1, \ldots, n\}$); $f(G, K)$ is the graph G altered so that the subgraph induced by K is complete.

A zero-knowledge proof of knowledge: Informally, a *zero-knowledge proof of knowledge* [15, 14] is a protocol by which a party A holding a secret s demonstrates its possession of s to some party B in such a way that B learns nothing about s. A *computational* zero-knowledge proof is one in which learning information about s is computationally intractable. (As opposed to a perfect zero-knowledge proof, in which s is concealed in an information theoretically secure fashion.)

Suppose Alice generates a random graph G and randomly plants a clique K of size $k = 3/2\log_2 n$ in it. After sending the resulting graph, G' to Bob, Alice may prove knowledge of K in computational zero knowledge by use of the following protocol (repeated suitably many times): (1) Alice applies a random permutation π to the graph G', yielding graph G''. She sends commitments of the edges of G'' to Bob; (2) Bob flips a coin, and sends the result (heads or tails) to Alice; (3) If heads, Alice sends Bob decommitments of all of the edges in G'', along with the permutation π. If tails, Alice decommits the edges corresponding to the clique K in G'' (i.e., those in $\pi(K)$); (4) If heads, Bob accepts the proof if the decommitment of G'' corresponds to $\pi(G')$. If tails, Bob accepts the proof if the decommitted edges form a clique of size k. Otherwise, Bob rejects the proof.

Hierarchical key creation: As mentioned above, our main theorem holds for any constant number of cliques. In other words, it is possible to have a set of multiple cliques $K_1, K_2, \ldots, K_c$ hidden in a single graph G'. If we regard the cliques $\{K_i\}$ as private keys, and G' as a public key, this suggests the ability to create private keys "hierarchically".

Suppose that G is the random graph into which a clique K of suitable size is hidden, and $G' = (V, E')$ is the graph resulting from this implantation. A party with knowledge of G is likely to be able to extract K. In particular, the set $E' - E$ will contain half of the edges of K on average; with this information, K can be easily determined with high probability. Consider, therefore, the following protocol. Party P_1 generates a random graph G and randomly inserts into it a clique K_1, yielding graph G_1. Party P_1 then passes G_1 to P_2, who randomly inserts a clique K_2, yielding G_2. This process is continued through to party P_c, who then publishes the public key G_c. Observe that in a suitably formulated system, P_1 can use its knowledge of G_1 to extract the private keys of $P_2, P_3, \ldots P_c$ (although it should be observed that P_1 cannot determine which key belongs to which party). Parties $P_2, P_3, \ldots, P_c$, however, cannot extract the private key of P_1. In general, party P_i can extract the private keys of all

parties P_j for $j > i$, while the reverse would require the ability to find a large clique, and is therefore presumed infeasible.

5 Further research

We have shown how hidden cliques provide an elementary means of creating cryptographically secure primitives. Many possible extensions to the work in this paper suggest themselves. It would be desirable, for instance, to strengthen Theorem 3.1 so that the result holds when more than a constant number of cliques, or when cliques of a size greater than $2\log_2 n$ are hidden. Does the security improve for non-constant edge probabilities (e.g. $1 - n^{-\epsilon}$ for $0 < \epsilon < 1$)? As a graph-based cryptographic tool, hidden cliques have some unusual properties, such as the ability to create private keys hierarchically. A good practical application of such properties would be interesting. Similarly interesting would be the creation of a public key cryptosystem based on cliques. Whether clique-based cryptographic primitives can be made efficient and practical remains an open question.

References

[1] N. Alon, M. Krivelevich, and B. Sudakov. Finding a large hidden clique in a random graph. In *Proc. of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, 1998.

[2] N. Alon and J. Spencer. *The Probabilistic Method*. Wiley, 1992.

[3] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. In *Proceedings 33rd IEEE Symposium on the Foundations of Computer Science*, pages 14–23, Los Angeles, CA, 1992. IEEE Computer Society.

[4] S. Arora and S. Safra. Approximating clique is NP-complete. In *Proceedings 33rd IEEE Symposium on the Foundations of Computer Science*, 1992.

[5] M. Bellare, O. Goldreich, and M. Sudan. Free bits, PCPs and non-approximability – towards tight results. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, pages 422–431, 1995.

[6] M. Bellare, S. Goldwasser, C. Lund, and A. Russel. Efficient probabilistically checkable proofs and applications to approximation. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing*, 1993.

[7] M. Bellare and M. Sudan. Improved non-approximability results. In *Proceedings of the 26th ACM Symposium on the Theory of Computing*, pages 184–193, Montreal, 1994. ACM Press.

[8] A. Blum and J. Spencer. Coloring random and semi-random k-colorable graphs. *Journal of Algorithms*, 19(2):204–234, 1995.

[9] B. Bollobás. *Random Graphs*. Academic Press, 1985.

[10] R. Boppana. Eigenvalues and graph bisection: An average-case analysis. In *Proceedings of the 28th Annual Symposium on Foundations of Computer Science*, pages 280–285, 1987.

[11] A. Z. Broder, A. M. Frieze, and E. Shamir. Finding hidden hamiltonian cycles. In *Proceedings of the 23rd Annual ACM Symposium on the Theory of Computing*, pages 182–189, 1991.

[12] T. Bui, S. Chaudhuri, T. Leighton, and M. Sipser. Graph bisection algorithms with good average-case behavior. *Combinatorica*, 6, 1986.

[13] M. E. Dyer and A. Frieze. Fast algorithms for some random NP-hard problems. *Journal of Algorithms*, 10:451–489, 1989.

[14] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design. In *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*, pages 174–187. IEEE, 1986.

[15] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. In *Proceedings of the 17th Annual ACM Symposium on the Theory of Computing*, pages 291–304, 1985.

[16] Y. Gurevich and S. Shelah. Expected computation time for Hamiltonian path problem. *SIAM Journal on Computing*, 16(3):486–502, 1987.

[17] M. Jerrum. Large cliques elude the Metropolis process. *Random Structures and Algorithms*, 3(4):347–360, 1992.

[18] D. S. Johnson and M. Trick, editors. *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge*. American Mathematical Society, 1996. DIMACS Series in Discrete Mathematics and Theoretical Computer Science.

[19] A. Juels. *Topics in Black-box Combinatorial Optimization*. PhD thesis, University of California, Berkeley, 1996.

[20] J. Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, pages 627–636, 1996.

[21] J. Håstad. Testing of the long code and hardness for clique. In *Proceedings of the 28th ACM Symposium on the Theory of Computing*, pages 11–19. ACM Press, 1996.

[22] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[23] R. M. Karp. Probabilistic analysis of some combinatorial search problems. In J. F. Traub, editor, *Algorithms and Complexity: New Directions and Recent Results*. Academic Press, 1976.

[24] L. Kučera. A generalized encryption scheme based on random graphs. In *Graph-Theoretic Concepts in Computer Science, WG'91*, Lecture Notes in Computer Science 570, pages 180–186. Springer-Verlag, 1991.

[25] L. A. Levin. Average-case complete problems. *SIAM Journal on Computing*, 15:285–286, 1986.

[26] M. Luby. *Pseudorandomness and Cryptographic Applications*. Princeton University Press, 1996.

[27] M. Peinado. Hard graphs for the randomized Boppana-Halldórsson algorithm for maxclique. *Nordic Journal of Computing*, 1:493–515, 1994. Preliminary version in *Proceedings of the 4th Scandinavian Workshop on Algorithm Theory*, Århus, Denmark. pages 278–289. Springer-Verlag, 1994.

[28] M. Peinado. Improved lower bounds for the randomized Boppana-Halldórsson algorithm for MAX-CLIQUE. In *Proceedings of the First Annual Computing and Combinatorics Conference*. Springer-Verlag, 1995.

[29] J. S. Turner. On the probable performance of heuristics for bandwidth minimization. *SIAM Journal on Computing*, 15(2):561–580, 1986.

[30] R. Venkatesan and L. Levin. Random instances of a graph coloring problem are hard. In *Proceedings of the 20th Annual ACM Symposium on the Theory of Computing*, pages 217–222, 1988.