Privacy-Preserving Planarity Testing of Distributed Graphs

Guy Barshap[*,**], **Tamir Tassa**[*]

[*]The Open University of Israel.

[**]Ben Gurion University of the Negev.

E-mail: barshag@post.bgu.ac.il,tamirta@openu.ac.il

Abstract. We study the problem of privacy-preserving planarity testing of distributed graphs. The setting involves several parties that hold private graphs on the same set of vertices, and an external mediator that helps with performing the computations. Their goal is to test whether the union of their private graphs is planar, but in doing so each party wishes to deny from his peers any information on his own private edge set beyond what is implied by the final output of the computation. We present a privacy-preserving protocol for that purpose which is based on the Hanani-Tutte theorem. That theorem enables translating the planarity question into the question of whether a specific system of linear equations over the field $\mathbf{F}_2$ is solvable. Our protocol uses a diverse cryptographic toolkit which includes techniques such as homomorphic encryption, oblivious Gaussian elimination, and private set intersection.

Keywords. secure multiparty computation, privacy-preserving distributed computations, distributed graphs, graph planarity

1 Introduction

A planar graph $G = (V, E)$ is a graph that can be properly embedded in the two-dimensional plane $\mathbf{R}^2$ in the following sense: there exists a bijection φ from the vertex set V to $\mathbf{R}^2$ and a representation of each edge $e = (u, v) \in E$ as a continuous simple curve in $\mathbf{R}^2$ with $\varphi(u)$ and $\varphi(v)$ as its end points, such that no two curves intersect apart possibly at their end points. Letting K_n denote the complete graph over n vertices (namely, the graph that has all $\binom{n}{2}$ edges) and $K_{n,n}$ denote the complete bipartite graph with n vertices in each part, then the graph K_4 is planar, since it has a planar drawing (Figure 1), but K_5 and $K_{3,3}$ are not (Figure 2).

Planar graphs constitute an attractive family of graphs, both in theory and in practice. In many applications where graph structures arise, it is needed to test the planarity of those graphs. A classical example is in the area of integrated circuit (IC) design. An IC consists of electronic modules and the wiring interconnections between them. It can be represented by a graph in which the vertices are the modules and the edges are the wires. An IC can be printed on the surface of a chip iff the graph is planar, because wires must not cross each other. Another setting in which planarity is a natural notion is in road maps. A set of cities and interconnecting roads can be thought of as a graph; the graph vertices are the cities while the edges are connecting roads. Such a map can be constructed with non-crossing

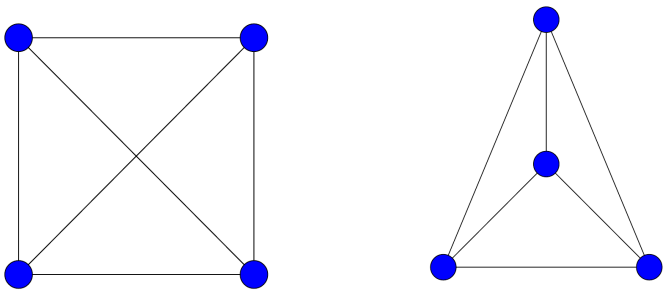

Figure 1: K_4 is a planar graph. The left drawing is non-planar, but the right drawing is planar.

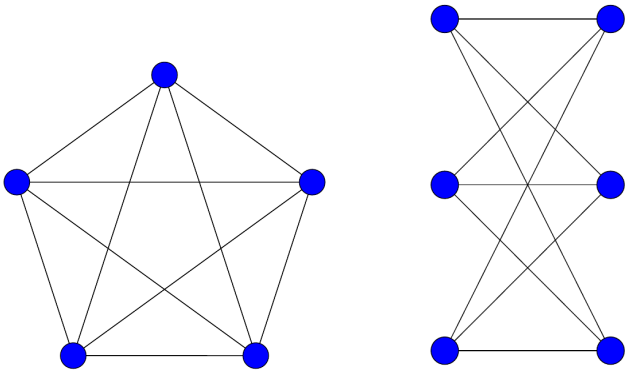

Figure 2: K_5 and $K_{3,3}$ are non-planar. A graph is planar iff it does not include either of them as minors.

roads (in order to avoid constructing bridges or obstructing the traffic flow by stop lights) iff the corresponding graph is planar.

Apart from the above motivating examples, there are cases in which the planarity of a graph can be exploited in order to simplify and expedite the solution of some computational problems. Examples include sub-graph isomorphism [12], maximal clique [32], and maximum cut [19].

In this study we consider a distributed version of the planarity testing problem. In that problem there are several parties, $P_1, \ldots, P_d$, each one holding a private graph on the same set of vertices. Namely, P_h has a graph $G_h = (V, E_h)$ where V is publicly known and shared by all, while E_h is private, $1 \leq h \leq d$. They wish to determine whether the union graph $G = (V, E)$, where $E = \bigcup_{h=1}^{d} E_h$, is planar or not. As the edge sets E_h, $1 \leq h \leq d$, are private, the planarity testing should be carried out in a privacy-preserving manner. Namely, after the conclusion of the computational procedure P_h must not learn anything on E_k, $k \neq h$, beyond what is implied by G_h and the planarity of G. For example, two (or more) companies may wish to check the possibility of printing the ICs which implement their products on the same chip. They prefer not to disclose to each other their own IC design, before they have verified that they could collaborate in that manner. The algorithmic solutions which we propose herein for privacy-preserving planarity testing could be used in that application scenario.

Well known characterizations for planar graphs were proposed by both Wagner [43] and Kuratowski [25]. For example, the Wagner's characterization states that a graph is planar iff it does not have K_5 or $K_{3,3}$ as a minor (see Figure 2). Namely, K_5 or $K_{3,3}$ cannot be obtained from G by a sequence of these operations: contracting edges, deleting edges, and deleting isolated vertices. However, directly applying either Wagner's characterization, or the closely-related Kuratowski's characterization, in order to test the planarity of a given graph, yields exponential-time algorithms [33].

Optimal linear time planarity testing algorithms were proposed in [9, 20]. These algorithms are iterative and use a DFS-subroutine [33]. Alas, those features of these algorithms turn out to be significant obstacles when trying to devise corresponding privacy-preserving variants of these algorithms. Thus, none of the above-mentioned approaches seem to be adequate in order to base on them an efficient privacy-preserving planarity testing protocol.

In this study we propose a privacy-preserving protocol for planarity testing of distributed graphs, which is based on the Hanani-Tutte theorem [34]. Our protocols are protocols of Secure Multiparty Computation (SMC). In the general setting of SMC [44], there are several parties, $P_1, \ldots, P_d$, where each P_h holds a private value x_h. The goal is to compute $f(x_1, \ldots, x_d)$, where f is some publicly known function, so that each party does not learn anything about the private inputs of the other parties, except what is implied by his own input and the output $f(x_1, \ldots, x_d)$. While generic SMC protocols apply in theory to a wide class of functions, their applicability in practice is limited to functions that have a compact representation as a Boolean or arithmetic circuit, due to their high computational and communication complexities. Further studies in this field aim at finding more efficient solutions for specific SMC problems, from a wide range of domains.

Here we consider an SMC problem where the private inputs are graphs. Problems of SMC on distributed graphs are of much interest and importance. Nonetheless, due to their apparent difficulty, very few studies were published so far on such problems. The first such study was by [10] who presented new algorithms for privacy-preserving computation of the all-pairs-shortest-distance and single-source-shortest-distance problems. A more recent study is that of [2] who designed SMC techniques for the shortest path and the maximum

flow problems. Another example is the work of [24] who presented oblivious implementations of several data structures for SMC and then offered a secure computation of Dijkstra's shortest path algorithm on general graphs, where the graph structure is secret. The problem of privacy-preserving computation of the Minimum Spanning Tree (MST) was considered in the recent study by [26]. In that study, Laud shows how the MST-finding algorithm by [5] can be executed without revealing any details about the underlying graph, beside its size. Finally, we mention the work of [3] that proposed SMC protocols for computing vertex centrality in distributed graphs. We note that all of the above mentioned studies present protocols for privately computing *arithmetic* values over graphs, i.e., lengths of paths, values of flows, or vertex centralities. Ours is the first study that privately tests a *geometric* property of a distributed graph.

Our protocol is based on the mediated model that was presented in [1]. In that model, there exists an external mediator T to which the parties may export some computations, but the mediator should not learn information on the private inputs of the parties nor on the final output.

The strict notion of perfect privacy-preservation is sometimes relaxed by allowing some leakage of information, if such a relaxation enables a more efficient computation and if the leakage of information is characterized (in order to decide, in any given application setting, whether the gain in efficiency justifies the reduction in privacy-preservation). There are many examples of studies that relax perfect privacy in order to allow practical solutions, from various domains such as distributed association rule mining [23, 37], anonymization of distributed datasets [22, 38, 40], collaborative filtering [21, 36], distributed graph mining [3], and distributed constraint optimization problems [16, 17, 18, 27, 39, 41, 42]. In similarity to these studies, we too make the common assumption that the interacting parties ($P_1, \ldots, P_d$ and T) are semi-honest. Namely, they follow the protocol correctly, and do not form coalitions, but they try to extract from their view in the protocol information on the private inputs of other parties.

The outline of this work is as follows. In Section 2 we provide the relevant background on planarity testing, while in Section 3 we describe the main cryptographic toolkit that we use in our solution. We overview our solution and the two main stages of which it consists in Section 4. The subsequent Sections 5 and 6 include the detailed description and analysis of each of the two stages in our solution. In Section 7 we describe a variant of our solution with reduced time complexity. Our discussion in Sections 4–7 focuses on the case $d = 2$. In Section 8 we present protocols for any d. In Section 9 we discuss another computational problem on distributed graphs — testing 3-colorability; we devise there an efficient protocol for privacy-preserving testing of 3-colorability of distributed graphs that were already verified to be planar. Finally, in Section 10 we end this study with concluding remarks and outlining future research directions.

A preliminary version of this study appeared in [6]. The main additions in this study with respect to the preliminary version are the contributions described in Sections 7, 8, and 9.

2 Planarity testing using the Hanani-Tutte theorem

In this section we state the Hanani-Tutte theorem and then use it in order to translate the planarity question of a graph to the solvability of a system of linear equations over $\mathbf{F}_2$. To that end, we introduce the following definitions and notations:

– If $e \in E$ we let $a(e)$ and $b(e)$ denote the two vertices that e connects.

– A drawing D of a graph $G = (V, E)$ is an embedding of G in $\mathbf{R}^2$. Namely, it is a mapping

Figure 3: Illustrating $parity_D(e, f)$: In the two left drawings $parity_D(e, f) = 0$ while in the other two $parity_D(e, f) = 1$.

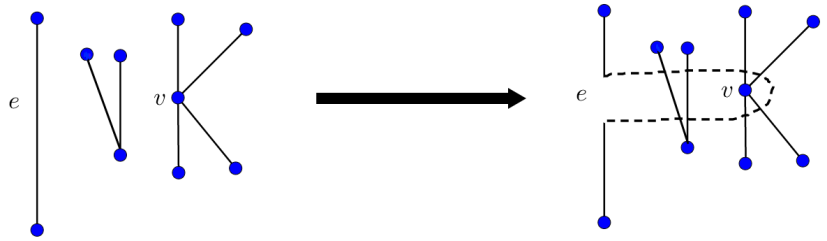

Figure 4: Illustrating an (e, v)-move. As a result of such a move, $parity_D(e, f)$ changes only for the four edges adjacent to v (from 0 to 1); for all other edges f, $parity_D(e, f)$ remains unchanged (0).

$\varphi : V \to \mathbf{R}^2$ together with a representation of each edge $e \in E$ as a continuous simple curve that connects $\varphi(a(e))$ and $\varphi(b(e))$.

– Two edges $e, f \in E$ are called independent if $\{a(e), b(e)\} \cap \{a(f), b(f)\} = \emptyset$.

– The set of all pairs of independent edges in E is denoted E_2^{ind}.

– For a given drawing D of G and $\{e, f\} \in E_2^{ind}$, $parity_D(e, f)$ is the parity of the number of crossings between the curves representing e and f in D (see Figure 3).

Theorem [Hanani-Tutte]. *A graph G is planar iff it has a drawing D in which $parity_D(e, f) = 0$ for all $\{e, f\} \in E_2^{ind}$.*

Let D be a drawing of G, $e \in E$, and $v \in V \setminus \{a(e), b(e)\}$. An (e, v)-move consists of taking a small section of the curve that represents e in D and deforming it in a narrow tunnel to make it pass over v, while not passing over any other vertex. The effect of an (e, v)-move in a drawing D is that $parity_D(e, f)$ changes for all edges f that are adjacent to v, but it remains unchanged for all other edges f (see Figure 4).

A remarkable consequence of this theorem is a planarity testing algorithm. The algorithm starts with an arbitrary drawing D of the input graph G, preferably a drawing in which $parity_D(e, f)$ can be computed efficiently for every pair of independent edges in G. Then, the algorithm tries to find another drawing D', by making a series of (e, v) moves, in which $parity_{D'}(e, f) = 0$ for all $\{e, f\} \in E_2^{ind}$. If it succeeds, then the graph is planar, otherwise it is not (see [34, Lemma 3.3]).

The existence of D' can be determined by considering the following system of linear equations. Define for each $e \in E$ and $v \in V \setminus \{a(e), b(e)\}$ a Boolean variable $x_{e,v}$; that variable equals 1 iff the transition from D to D' includes an (e, v)-move. It follows that for any pair

of independent edges $\{e, f\} \in E_2^{ind}$,

$$parity_{D'}(e, f) = parity_D(e, f) + x_{e,a(f)} + x_{e,b(f)} + x_{f,a(e)} + x_{f,b(e)} \text{ in } \mathbf{F}_2 .$$

Hence, given the drawing D, there exists a drawing D' in which $parity_{D'}(e, f) = 0$ for all $\{e, f\} \in E_2^{ind}$ iff there exists a solution to the following system of linear equations over $\mathbf{F}_2$:

$$parity_D(e, f) + x_{e,a(f)} + x_{e,b(f)} + x_{f,a(e)} + x_{f,b(e)} = 0 \qquad \{e, f\} \in E_2^{ind} . \tag{1}$$

That is the Hanani-Tutte (HT hereinafter) system for the graph $G = (V, E)$ (with respect to the drawing D). It consists of $|E_2^{ind}|$ equations (one for each pair of independent edges) in $|E| \cdot (|V| - 2)$ unknowns ($x_{e,v}$ for all $e \in E$ and $v \in V \setminus \{a(e), b(e)\}$). The graph G is planar iff that system is solvable.

3 Cryptographic toolkit

In this section we provide a birdseye view of the cryptographic primitives and procedures that we shall be using in the main part of this work, Sections 4–7. In the subsequent Section 8 we will use a special homomorphic encryption (due to Boneh, Goh and Nissim [8]), while our solution in Section 9 utilizes the BGW protocol [7]. We postpone the discussion of those two cryptographic tools to the sections were they are used.

3.1 Homomorphic encryption

An encryption function $\mathcal{F}$ is called (additively) *homomorphic* if the domain of plaintexts is a commutative additive group, the domain of ciphertexts is a commutative multiplicative group, and for every two plaintexts, m_1 and m_2, $\mathcal{F}(m_1 + m_2) = \mathcal{F}(m_1) \cdot \mathcal{F}(m_2)$. When the encryption function is randomized (in the sense that $\mathcal{F}(m)$ depends on m as well as on a random string) then $\mathcal{F}$ is called *probabilistic*. Homomorphic encryption functions allow performing arithmetic computations in the ciphertext domain. The property of being probabilistic is essential for getting semantic security.

There are many well known ciphers that are probabilistic and additively homomorphic. A basic example of such a cipher over $\mathbf{F}_2$, which we use in our protocols, is the Goldwasser-Micali cipher [15].

3.2 Deciding the solvability of an encrypted linear system

Nissim and Weinreb [31] presented a method for obliviously deciding whether an encrypted system of linear equations is solvable or not. They considered a setting that involves two parties – T and P. T holds an encrypted matrix $\mathcal{F}(M)$, where M is a matrix of dimensions $k_a \times k_b$, and an encrypted vector $\mathcal{F}(\mathbf{b})$, where $\mathbf{b}$ is a column vector of dimension k_a. Both M and $\mathbf{b}$ are over the field $\mathbf{F} = \mathbf{F}_2$, while $\mathcal{F}$ is an additively homomorphic encryption over that field, for which P holds the private decryption key. Their protocol is Monte Carlo in the sense that its output may be wrong. Specifically, at the conclusion of the protocol T gets $\mathcal{F}(\beta)$ for some bit β. If the system $M\mathbf{x} = \mathbf{b}$ is not solvable then β will always be zero. Otherwise, if the system is solvable, then $\beta = 1$ with probability at least c for some positive constant c. Hence, by performing several independent runs of the protocol, it is possible to decide the solvability of the system with an error probability sufficiently small.

4 Overview of the proposed planarity testing protocol

Our planarity testing protocol has two stages. The first one consists of a preliminary check of the size of the unified edge set E. It is outlined in Section 4.1, and detailed in Section 5. The second stage includes the main protocol, in which the HT system of equations for the unified graph is constructed obliviously, and then its solvability is tested in a privacy-preserving manner. We provide a birdseye view of that stage in Section 4.2, and dive into its details in Section 6.

4.1 Testing the number of edges in the unified graph

Let $V = \{v_1, \ldots, v_n\}$ denote the vertex set in the unified graph $G = (V, E)$. A well-known result (see e.g. [33]) states that G is planar only if

$$|E| \leq 3n - 6\,. \tag{2}$$

Hence, in the first stage, the three parties, P_1, P_2 and T, engage in a secure protocol for checking whether inequality (2) holds or not. If it does not, they know that the unified graph G is not planar. If it does hold, they proceed to the second stage in the verification (Section 4.2).

Running this stage is optional. On one hand, it leaks to the interacting parties information on $|E|$ beyond the required output about the planarity of G. Indeed, if the verification of inequality (2) fails, then the parties will learn that the graph is not planar, and, in addition, that $|E| > 3n - 6$. On the other hand, it may enable the parties to detect non-planarity without running the costly computation of the second stage. Hence, if in the relevant application scenario the information regarding whether inequality (2) holds or not is deemed benign, it is recommended to run this stage.

4.2 A privacy-preserving implementation of the Hanani-Tutte planarity test

The three parties P_1, P_2 and T construct the HT system of linear equations, Eq. (1), for the unified graph G. Towards that end, they begin by constructing the system of linear equations for the complete graph on V, denoted K_V (i.e., K_V is the graph on V that has all $\binom{n}{2}$ edges):

$$x_{e,a(f)} + x_{e,b(f)} + x_{f,a(e)} + x_{f,b(e)} = parity_D(e, f) \qquad \{e, f\} \in K_2^{ind}\,; \tag{3}$$

here, K_2^{ind} is the set of all pairs of independent edges in K_V. The number of equations in that system is

$$N_{K_V} := |K_2^{ind}| = \frac{1}{2} \cdot \binom{n}{2} \cdot \binom{n-2}{2}\,. \tag{4}$$

System 3 can be constructed publicly with no privacy risks, since the vertex set V is known to all, and K_V is the complete graph on V. The main effort is in letting the mediator T extract from the large system in Eq. (3) the subset of equations in Eq. (1). To protect the unified graph data from T, he will get only an encrypted version of the subset of linear equations corresponding to G. The last part of the protocol is dedicated to determining whether that system has a solution or not. The main difficulty here lies in the fact that no party actually sees the relevant system of equations: only T holds that system, but he holds

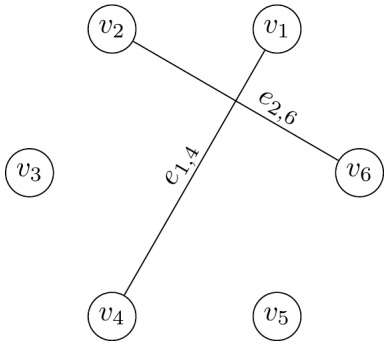

Figure 5: An illustration of the embedding of $n = 6$ vertices, together with two connecting edges.

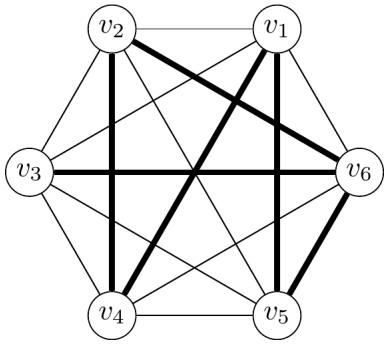

Figure 6: A drawing of $G = (V, E)$ with $E = \{e_{1,4}, e_{1,5}, e_{2,4}, e_{2,6}, e_{3,6}, e_{5,6}\}$ and the corresponding complete graph K_V. The edges in G are marked by thick lines, while all edges in K_V which are not in G are marked by thin lines.

an encryption of that system, where the corresponding decryption key is known only to P_1.

To allow this approach, we must start with some drawing of K_V, which, in turn, induces also a drawing of G. We consider the following embedding of V in $\mathbf{R}^2$. If $V = \{v_1, \ldots, v_n\}$, then v_j is mapped into the point

$$v_j \mapsto \varphi(v_j) := (\cos(2\pi j/n), \sin(2\pi j/n)) \,, \quad 1 \leq j \leq n \,. \tag{5}$$

Namely, the vertices $v_1, \ldots, v_n$ are mapped to equi-distant points on the unit circle, in a counter-clockwise order according to their index. The edge $e_{i,j} = (v_i, v_j)$ is then represented by the straight line segment between $\varphi(v_i)$ and $\varphi(v_j)$. Figure 5 illustrates that basic drawing for a graph over $n = 6$ vertices.

Figure 6 shows a drawing of some graph G, using the above-described embedding of its vertices and edges in $\mathbf{R}^2$, together with the corresponding complete graph. We shall henceforth denote this basic drawing of the graph in question, G, and its corresponding complete graph K_V by D.

Consider now an arbitrary pair of independent edges $e_{i,j}$ and $e_{k,\ell}$; we may assume, without loss of generality, that $i < j$, $k < \ell$, and $i < k$. Then it is easy to see that $parity_D(e_{i,j}, e_{k,\ell}) = 1$ iff $i < k < j < \ell$.

The corresponding HT system (3) can be constructed publicly, by each of P_1, P_2 and T, for this drawing D of the complete graph K_V. As stated earlier, the main problem will be to identify, among those N_{K_V} equations, the $|E_2^{ind}|$ equations that relate to pairs of edges e and f that are both in E. Then, the graph $G = (V, E)$ is planar iff that sub-system of $|E_2^{ind}|$ equations has a solution. In Section 6 we provide the details of that computation.

5 First stage: Testing the size of the unified edge set

Let

$$V_2 := \{(v_i, v_j) : 1 \leq i < j \leq n\} \tag{6}$$

denote the set of all possible $\binom{n}{2}$ edges in G. Since $E = E_1 \cup E_2$, we infer that $E^c = E_1^c \cap E_2^c$, where for any subset $A \subseteq V_2$, $A^c := V_2 \setminus A$ denotes its complement within V_2. Hence, by Eq. (2), the unified graph $G = (V, E)$ is planar only if

$$|E^c| = |E_1^c \cap E_2^c| \geq \binom{n}{2} - (3n - 6)\,. \tag{7}$$

In order to verify the latter inequality it is possible to invoke any of the multitude of protocols for private set intersection. Our solution, Protocol 1, is based on the first private set intersection protocol, proposed in [30], which is based on the Diffie-Hellman problem [11].

Protocol 1 Testing the size of the unified edge set

1: P_1 and P_2 select jointly a large multiplicative group $\mathbf{Z}_p^*$ (p is prime) and a hash function H whose range can be embedded in $\mathbf{Z}_p^*$.
2: P_h selects a secret and random exponent $1 < \alpha_h < p - 1$, $h = 1, 2$.
3: P_1 sends to P_2 a vector $\mathbf{x}_1$ of length $\binom{n}{2}$ where, for each edge $(v_i, v_j) \in E_1^c$, $i < j$, $\mathbf{x}_1$ includes an entry of the form $H(i, j)^{\alpha_1}$, while the remaining $\binom{n}{2} - |E_1^c|$ entries are randomly selected from $\mathbf{Z}_p^*$. The order of $\mathbf{x}_1$'s entries is random.
4: P_2 sends to P_1 a vector $\mathbf{x}_2$ of length $\binom{n}{2}$ where, for each edge $(v_i, v_j) \in E_2^c$, $i < j$, $\mathbf{x}_2$ includes an entry of the form $H(i, j)^{\alpha_2}$, while the remaining $\binom{n}{2} - |E_2^c|$ entries are randomly selected from $\mathbf{Z}_p^*$. The order of $\mathbf{x}_2$'s entries is random.
5: P_1 sends to T the vector $\mathbf{y}_2$, where $\mathbf{y}_2(i) = \mathbf{x}_2(i)^{\alpha_1}$, $1 \leq i \leq \binom{n}{2}$.
6: P_2 sends to T the vector $\mathbf{y}_1$, where $\mathbf{y}_1(i) = \mathbf{x}_1(i)^{\alpha_2}$, $1 \leq i \leq \binom{n}{2}$.
7: T compares the two received vectors and finds out the number z of matching entries in them.
8: If $z < \binom{n}{2} - (3n - 6)$, T notifies P_1 and P_2 that the union graph is not planar.

As a result of Steps 1-6, which are self-explanatory, T receives two vectors, $\mathbf{y}_1$ and $\mathbf{y}_2$, each of which is of length $\binom{n}{2}$. The vector $\mathbf{y}_h$, $h = 1, 2$, includes the hash of all edges in E_h^c, raised to the exponent $\alpha_1\alpha_2$, while the remaining $\binom{n}{2} - |E_h^c|$ entries are random elements in $\mathbf{Z}_p^*$. The number z of matching entries in those two vectors (Step 7) satisfies $z \geq |E^c|$, while with very high probability $z = |E^c|$. Indeed, if $(v_i, v_j) \in E_1^c \cap E_2^c$ then both vectors $\mathbf{y}_1$ and $\mathbf{y}_2$ will include an entry that equals $H(i, j)^{\alpha_1\alpha_2}$; hence, those two entries will be identified by T in Step 7 as matching entries and, consequently, T will increment the counter z by 1. However, we note that T may wrongly increment the counter z due to random false matchings. False matchings can occur if there are collusions in H, namely, if there exist two pairs (i, j) and (i', j') such that $H(i, j) = H(i', j')$, or if P_1 and P_2 selected in Steps 3 and 4

random entries ξ_1 and ξ_2, respectively, so that $\xi_2^{\alpha_1} = \xi_1^{\alpha_2}$. By selecting a secure hash function with a sufficiently large range, the probability of such false matchings is negligible.

The security of Protocol 1 follows from the hardness of the Discrete Log problem. The protocol entails $O(n^2)$ hash function evaluations and exponentiations (for P_1 and P_2) and $O(n^2 \log n)$ comparisons for T. The protocol has only two rounds of communication in which $O(n^2 \log p)$ bits are transmitted.

Protocol 1 reveals to T the size of E. If P_1 and P_2 wish to prevent T from learning that information, they may modify Protocol 1 towards hiding that information. They can choose an integer $K > 0$ and then select at random an integer $k \in [0, K]$. Next, they will select $2K - k$ random and distinct elements from $\mathbf{Z}_p^*$: a_i, $1 \leq i \leq k$, and $b_{h,i}$, $1 \leq i \leq K - k$, $h = 1, 2$. Then, P_1 will add to $\mathbf{y}_2$ additional K entries, at random locations, with the values $a_1, \ldots, a_k, b_{1,1}, \ldots, b_{1,K-k}$, while P_2 will add to $\mathbf{y}_1$ additional K entries, at random locations, with the values $a_1, \ldots, a_k, b_{2,1}, \ldots, b_{2,K-k}$. Given those modifications, T will recover in Step 7 a value z that equals $|E^c| + k$ (with high probability).

Hence, the inequality that needs to be verified now is whether $z < \binom{n}{2} - (3n - 6) + k$. In the latter inequality, T knows the left hand side (z) while P_1 knows the right hand side, and the two parties need to verify the inequality without disclosing to each other information on the compared values beyond the information of whether the inequality holds or not. This is an instance of the celebrated Yao's millionaires' problem [44]. By invoking any of the many available protocols for solving that problem (e.g. [28]), the two parties may find out securely whether $|E^c| < \binom{n}{2} - (3n - 6)$.

Such a modification of Protocol 1 prevents T from getting $|E|$. Higher values of K will imply higher levels of obfuscation, but at the same time also higher communication and computational costs. As implied by [17, Lemma 4], if $K > \binom{n}{2}$ then the probability of T not learning anything on $|E|$ from z is exactly $1 - \binom{n}{2}/(K + 1)$; in all other cases (namely, in probability $\binom{n}{2}/(K + 1)$, T will learn either a lower or an upper bound on $|E|$.

6 Second stage: Private planarity testing

Protocol 2 decides the planarity of the union graph G in a privacy-preserving manner. It begins with P_1 generating a key pair in a probabilistic additively homomorphic cipher, $\mathcal{F}$, over $\mathbf{F}_2$.

Next, the three parties execute a sub-protocol, called CONSTRUCTHTSYSTEM (Step 2). The purpose of that sub-protocol is to construct the HT system for the union graph G, Eq. (1), in an oblivious manner. At the end of that sub-protocol T will hold an (entry-wise) $\mathcal{F}$-encryption of the coefficient matrix of that system. T will actually get an "inflated" version of that system, in the following sense: instead of an encryption of the HT system for G, Eq. (1), he will hold an encryption of the larger HT system for the complete graph K_V, Eq. (3), where all equations that relate to pairs of edges in $K_2^{ind} \setminus E_2^{ind}$ are zeroed.

Then (Step 3), P_1 and T execute a sub-protocol, called DECIDESOLVABILITY, which decides the solvability of the encrypted system that was constructed in the previous step. T holds the encryption of the coefficient matrix and the right hand side vector for that system, while P_1 holds the relevant decryption key. The sub-protocol DECIDESOLVABILITY decides the solvability of the system in a privacy-preserving manner, that is – without decrypting the system. The Boolean flag that DECIDESOLVABILITY returns indicates the solvability of the system. If it is **true** then the system is solvable and, consequently, the union graph G is planar. Otherwise, if it is **false**, and then, with high probability (which may be tuned as desired), the system is not solvable and, hence, the union graph G is not planar.

In the next sub-sections (Sections 6.1 and 6.2) we discuss the implementation of the two main steps in Protocol 2.

Protocol 2 Privacy preserving HT planarity testing

1: P_1 generates a key pair in a probabilistic additively homomorphic cipher, $\mathcal{F}$, over $\mathbf{F}_2$. P_1 notifies P_2 and T of the public encryption key in $\mathcal{F}$.
2: P_1, P_2 and T execute CONSTRUCTHTSYSTEM. At its conclusion, T holds an $\mathcal{F}$-encryption of the HT system for the union graph, $(\mathbf{r}_{e,f} : \{e,f\} \in K_2^{ind})$.
3: P_1 and T execute DECIDESOLVABILITY$(\mathbf{r}_{e,f} : \{e,f\} \in K_2^{ind})$.
4: **if** DECIDESOLVABILITY returns **true then**
5: Output "The union graph is planar".
6: **else**
7: Output "The union graph is non-planar (with high probability)".
8: **end if**

6.1 Constructing an $\mathcal{F}$-encryption of the HT system

Here we discuss the sub-protocol CONSTRUCTHTSYSTEM, which is implemented in Protocol 3. Before starting to do so, we take a look at the HT systems for G and for the complete graph K_V. The HT system for the complete graph K_V, with respect to the drawing D described in Section 4.2, is given in Eq. (3). All parties can construct that system, since K_V is a public graph. The HT system for G, Eq. (1), which determines the planarity of $G = (V, E)$, is a sub-system (subset of equations) of (3). That sub-system includes only the equations relating to pairs $\{e,f\} \in E_2^{ind} \subset K_2^{ind}$ (namely, pairs of independent edges $\{e,f\}$ where both e and f are in E).

Let $V_2 := \{e_{i,j} = (v_i, v_j) : 1 \leq i < j \leq n\}$ denote the edge set in the full graph K_V (consisting of all possible pairs of vertices from V). The set K_2^{ind} consists of all pairs of independent edges in K_V. Its size is N_{K_V} (Eq. (4)), and it consists of all pairs of edges $\{e = (v_i, v_j), f = (v_k, v_\ell)\}$ where all four indices i, j, k, ℓ are distinct (as the two edges are independent), and $i < j$, $k < \ell$, and $i < k$. Our protocol, CONSTRUCTHTSYSTEM, assumes that the set K_2^{ind} is ordered. We assume hereinafter that it is ordered lexicographically by the 4-tuple (i, j, k, ℓ).

For each edge $e \in V_2$ and $h \in \{1, 2\}$, let α_e^h be the Boolean variable denoting whether $e \in E_h$ or not. Then, $e \in E$ iff $\alpha_e^1 \vee \alpha_e^2 = 1$. Consequently, the equation that corresponds to the pair of independent edges $\{e,f\} \in K_2^{ind}$ appears in the HT system for G, Eq. (1), iff

$$\chi_{e,f} := (\alpha_e^1 \vee \alpha_e^2) \wedge (\alpha_f^1 \vee \alpha_f^2) = 1\,. \tag{8}$$

In the first part of CONSTRUCTHTSYSTEM (Steps 1-3), T gets the $\mathcal{F}$-encryption of $\chi_{e,f}$ for all $\{e,f\} \in K_2^{ind}$, where $\chi_{e,f}$ is the Boolean flag indicating whether $\{e,f\} \in E_2^{ind}$ (Eq. (8)), and $\mathcal{F}$ is the cipher that P_1 selected in Step 1 of Protocol 2. Towards that end, we observe that

$$\chi_{e,f} := (\alpha_e^1 \vee \alpha_e^2) \wedge (\alpha_f^1 \vee \alpha_f^2) = (\alpha_e^1 + \alpha_e^2 - \alpha_e^1 \cdot \alpha_e^2) \cdot (\alpha_f^1 + \alpha_f^2 - \alpha_f^1 \cdot \alpha_f^2)\,. \tag{9}$$

Hence, by opening the brackets on the right hand side of Eq. (9) and then applying $\mathcal{F}$ on both sides of that equation, we get, using the homomorphism of $\mathcal{F}$ and simple algebra, that $\mathcal{F}(\chi_{e,f}) = A \cdot B^{-1}$, where

$$A = \alpha^{\alpha_f^2} \cdot \beta^{\alpha_e^2} \cdot \gamma^{1+\alpha_e^2 \cdot \alpha_f^2} \cdot \mathcal{F}(\alpha_e^2 \cdot \alpha_f^2)\,, \quad B = (\alpha \cdot \beta)^{\alpha_e^2 \cdot \alpha_f^2} \cdot \gamma^{\alpha_e^2 + \alpha_f^2}\,, \tag{10}$$

and

$$\alpha := \mathcal{F}(\alpha_e^1)\,,\ \beta := \mathcal{F}(\alpha_f^1)\,,\ \gamma := \mathcal{F}(\alpha_e^1 \cdot \alpha_f^1)\,. \tag{11}$$

In view of the above derivations, P_1 sends to P_2 the three values α, β, and γ, for each of the N_{K_V} pairs $\{e,f\} \in K_2^{ind}$, where the triplets (α,β,γ) are ordered by the lexicographical order over K_2^{ind} (Step 1). P_2 can then use Eq. (10) in order to compute A and B, for each such pair. Note that all powers of α, β and γ in Eq. (10) are determined by Boolean variables owned by P_2, while $\mathcal{F}(\alpha_e^2 \cdot \alpha_f^2)$ can be computed by P_2 since he has the public encryption key of $\mathcal{F}$. P_2 then computes and sends to T a vector $\mathbf{u}$ of length N_{K_V}, in which each entry includes the value $\mathcal{F}(\chi_{e,f}) = A \cdot B^{-1}$ for the relevant pair $\{e,f\} \in K_2^{ind}$ (Steps 2-3).

Protocol 3 CONSTRUCTHTSYSTEM: Constructing an encryption of the HT system

1: P_1 sends to P_2 the vector $((\alpha,\beta,\gamma) : \{e,f\} \in K_2^{ind})$ (see Eq. (11)).
2: P_2 computes the vector $\mathbf{u} := (\mathcal{F}(\chi_{e,f}) : \{e,f\} \in K_2^{ind})$.
3: P_2 sends to T the vector $\mathbf{u}$.
4: **for all** $\{e,f\} \in K_2^{ind}$, where $e = e_{i,j}$, $f = e_{k,\ell}$, $i < j$, $k < \ell$ and $i < k$ **do**
5: T allocates a vector $\mathbf{r}_{e,f}$ of dimension $N+1$ where $N := \binom{n}{2} \cdot (n-2)$.
6: T creates a bijection $\Phi : [N] \to \{(g,v) : g \in V_2, v \in V \setminus \{a(g), b(g)\}\}$.
7: **for** $i \in [N]$ **do**
8: $(g,v) \leftarrow \Phi(i)$
9: **if** ($g = e$ and $v \in \{a(f), b(f)\}$) or ($g = f$ and $v \in \{a(e), b(e)\}$) **then**
10: $\mathbf{r}_{e,f}(i) \leftarrow \mathcal{F}(\chi_{e,f})$
11: **else**
12: $\mathbf{r}_{e,f}(i) \leftarrow \mathcal{F}(0)$
13: **end if**
14: **end for**
15: **if** $i < k < j < \ell$ **then**
16: $\mathbf{r}_{e,f}(N+1) \leftarrow \mathcal{F}(\chi_{e,f})$
17: **else**
18: $\mathbf{r}_{e,f}(N+1) \leftarrow \mathcal{F}(0)$
19: **end if**
20: **end for**

The goal of the main loop in Steps 4-20 is to let T construct an entry-wise $\mathcal{F}$-encryption of the HT system for G, Eq. (1). That system has $|E_2^{ind}|$ equations over $|E| \cdot (n-2)$ unknowns. It is a sub-system of the full system for K_V, Eq. (3), which has N_{K_V} equations over $N := \binom{n}{2} \cdot (n-2)$ unknowns. The encrypted system that T constructs in the loop in Steps 4-20 will be of the same dimensions as the larger system for the full graph, but all rows in it that are not relevant for G will be zeroed. T will remain oblivious to the rows in the full system that he zeroes in this process. We proceed to explain how this is done.

Consider the augmented matrix that describes the HT system for K_V, Eq. (3). It has N_{K_V} rows, one for each pair $\{e,f\} \in K_2^{ind}$. Each row, $\mathbf{r}_{e,f}$, is a Boolean vector of length $N+1$, where $N = \binom{n}{2} \cdot (n-2)$, since it includes the coefficient of each unknown variable (and there are N such variables, one for each coupling of an edge and a non-adjacent vertex) plus the right hand side ($parity_D(e,f)$). In the linear equation corresponding to the pair $\{e,f\}$, the coefficients of all variables are zero, except for four of those variables (see Eq. (3)). Hence, in the inner loop in Steps 7-14, T goes over the first N entries of $\mathbf{r}_{e,f}$; in each of the four entries that should equal 1, T places the value $\mathcal{F}(\chi_{e,f})$ (those are values that T got from P_2 in Step 3), while in all the remaining ones he places the value $\mathcal{F}(0)$ (those are encryptions that T can compute on his own since he has the public encryption key of $\mathcal{F}$). In the last position in $\mathbf{r}_{e,f}$, corresponding to the right hand side of the equation for the pair $\{e,f\}$, T places

the value $\mathcal{F}(\chi_{e,f})$ in case the two edges intersect in the basic drawing D, while otherwise he places the value $\mathcal{F}(0)$ (Steps 15-19). As a result, if $\chi_{e,f} = 0$, T constructs an encryption of the all-zero equation; but if $\chi_{e,f} = 1$, T constructs an encryption of the equation for the pair $\{e, f\}$, as in Eq. (1). In summary, T gets a system of N_{K_V} encrypted equations: $|E_2^{ind}|$ of those equations are an $\mathcal{F}$-encryption of the system (1), while the remaining ones are $\mathcal{F}$-encryptions of the trivial equation (the equation in which all coefficients and right hand side are zero).

6.2 Determining the solvability of an encrypted linear system

The sub-protocol DECIDESOLVABILITY (Protocol 4) decides the solvability of a system of N_{K_V} linear equations over $N = \binom{n}{2} \cdot (n-2)$ unknowns. Let M denote the $N_{K_V} \times N$ matrix of coefficients of that system, and $\mathbf{b}$ denote the right hand side vector (an N_{K_V}-dimensional column vector). The two parties that run the sub-protocol are P_1 and T. P_1 holds the decryption key in $\mathcal{F}$ (see Step 1 in Protocol 2) – a probabilistic additively homomorphic cipher over $\mathbf{F}_2$; T, on the other hand, holds $\mathcal{F}(M)$ and $\mathcal{F}(\mathbf{b})$. DECIDESOLVABILITY determines whether the system $M\mathbf{x} = \mathbf{b}$ has a solution or not. It does so in a privacy-preserving manner, i.e., without revealing to neither of the two parties information on the underlying matrix M and right hand side $\mathbf{b}$.

To do so, the two parties execute the protocol due to Nissim and Weinreb [31] that we discussed in Section 3.2, which enables them to obliviously decide whether an encrypted system of linear equations is solvable or not. We refer to this protocol below by the name SOLVABILITYLINEARSYSTEM. The output of Protocol SOLVABILITYLINEARSYSTEM is $\mathcal{F}(\beta)$, where β is a bit that indicates the existence of a vector $\mathbf{x}$ for which $M\mathbf{x} = \mathbf{b}$.

We recall that the basic protocol due to Nissim and Weinreb is a true-biased Monte Carlo protocol. Namely, a **true** answer (i.e., the system is solvable) is always correct, while a **false** answer may be wrong. We assume herein that the procedure SOLVABILITYLINEARSYSTEM which is invoked in Step 1 of Protocol 4 executes the basic protocol due to Nissim and Weinreb a sufficient number of times so that the probability of a false answer to be incorrect is reduced to below some given desired threshold.

As the protocol SOLVABILITYLINEARSYSTEM is quite involved, we do not describe it here and simply relate to it as a black-box. Interested readers are referred to [31] for a detailed description of it.

Sub-protocol 4 DECIDESOLVABILITY

1: T and P_1 run Protocol SOLVABILITYLINEARSYSTEM with inputs $\mathcal{F}(M)$ and $\mathcal{F}(\mathbf{b})$. The output $\mathcal{F}(\beta)$ goes to T.
2: T sends to P_1 the value $\mathcal{F}(\beta)$.
3: P_1 decrypts and recovers β.
4: **if** $\beta = 1$ **then**
5: return **true**
6: **else**
7: return **false**
8: **end if**

6.3 Privacy analysis

The potential leakages of information to any party is due to messages that he receives from other parties. We proceed to discuss the security of each of the steps in Protocol 2 that involves exchange of messages.

In Step 1 of Protocol 3 (which is invoked by Protocol 2), P_2 receives from P_1 information relating to E_1. Specifically, for each pair of independent edges in K_V, P_2 receives the values $\alpha := \mathcal{F}(\alpha_e^1)$, $\beta := \mathcal{F}(\alpha_f^1)$, and $\gamma := \mathcal{F}(\alpha_e^1 \cdot \alpha_f^1)$. However, as that information is encrypted by $\mathcal{F}$, P_2 cannot extract information on E_1, assuming that the chosen cipher $\mathcal{F}$ is semantically secure (as is the case with the Goldwasser-Micali cipher [15] which we propose to utilize here). Similarly for Step 3 of Protocol 3 in which T receives information on E; as it is encrypted by $\mathcal{F}$, it is protected from T.

Finally, the security of Protocol 4 follows from the security of SOLVABILITYLINEARSYSTEM that was established in [31].

6.4 Computational and communication costs

We begin by assessing the computational and communication costs of the first two steps in Protocol 3. The computational cost of Protocol 3 for P_1 is $2\binom{n}{2}$ encryptions (for computing α and β for all edges in K_V), and, in addition, N_{K_V} (Eq. (4)) encryptions, for computing γ for all pairs of edges in K_2^{ind} (Step 1). The computational cost of Protocol 3 for P_2 is dominated by the need to perform N_{K_V} encryptions ($\mathcal{F}(\alpha_e^2 \cdot \alpha_f^2)$) in order to compute A for all pairs of edges in K_2^{ind}. The remaining operations for computing A and B (see Eq. (10)) in Step 2 are only multiplications (note that all exponents in Eq. (10) are 0, 1, or 2). The communication cost of Steps 1-2 in Protocol 3 is $O(N_{K_V})$ bits.

The computational costs for T due to Protocol 3 are negligible, as it has to perform no new encryptions, other than computing $\mathcal{F}(0)$ once. We note that the value $\mathcal{F}(0)$ appears in many entries in the encrypted HT linear system of equations. However, the procedure SOLVABILITYLINEARSYSTEM, which is executed in the next stage, is designed so that there is no need for T to generate here independent encryptions $\mathcal{F}(0)$ for each such entry.

The main computational bottleneck is Protocol 4. Indeed, as the underlying matrix has $k_a = N_{K_V} = O(n^4)$ rows and $k_b = \binom{n}{2} \cdot (n-2) = O(n^3)$ columns, the computational cost of running an oblivious Gaussian elimination on it is of order $O(k_a \cdot k_b^2) = O(n^{10})$. Such a computational cost severely limits the applicability of our protocol to very small graphs.

The main problem with Protocol 2 is that it runs the oblivious Gaussian elimination over an encrypted matrix that has N_{K_V} rows. We recall that the actual system, Eq. (1), has only $|E_2^{ind}|$ equations. Since $|E| \leq 3n-6$, as verified in the first stage, then $|E_2^{ind}| = O(n^2)$. Hence, one goal is to reduce the number of rows in the encrypted HT system from $N_{K_V} = O(n^4)$ to the exact number of relevant equations $|E_2^{ind}| = O(n^2)$. Moreover, the number of columns in Eq. (1) is $|E| \cdot (n-2) \leq (3n-6) \cdot (n-2) = O(n^2)$. Therefore, another goal is to reduce the number of columns (unknowns) from $O(n^3)$ to $O(n^2)$. In the next section we present a variant of Protocol 2 that achieves those two goals, thus reducing the cost of the oblivious Gaussian elimination to $O(n^6)$. While this time complexity still limits the scalability of the protocol, it allows its execution on graphs with several hundreds of vertices. Such time complexity renders our protocol viable for application settings such as the two motivating examples that were considered in the introduction (IC design and road networks).

7 Reducing the size of the HT system

In this section we present Protocol 5 which is a variant of Protocol 2 that has a reduced time complexity. In that protocol, the HT system that the parties construct (obliviously) has an equation for every pair of edges $\{e, f\} \in E_2^{ind}$, in contrast to Protocol 2 in which the constructed system had an equation for every pair of edges $\{e, f\} \in K_2^{ind}$, where all equations corresponding to $\{e, f\} \in K_2^{ind} \setminus E_2^{ind}$ were obliviously zeroed. As a result, the system of equations whose solvability is tested in Step 3 has reduced dimensions and, consequently, reduced runtime, as explained above.

Protocol 5 Privacy preserving HT planarity testing

1: P_1 generates a key pair in a probabilistic additively homomorphic cipher, $\mathcal{F}$, over $\mathbf{F}_2$. P_1 notifies T of the public encryption key in $\mathcal{F}$.
2: P_1, P_2 and T execute CONSTRUCTREDUCEDHTSYSTEM. At its conclusion, T holds an $\mathcal{F}$-encryption of the HT system for the union graph, $(\mathbf{r}_{e,f} : \{e, f\} \in E_2^{ind})$.
3: P_1 and T execute DECIDESOLVABILITY$(\mathbf{r}_{e,f} : \{e, f\} \in E_2^{ind})$.
4: **if** DECIDESOLVABILITY returns **true then**
5: Output "The union graph is planar".
6: **else**
7: Output "The union graph is non-planar (with high probability)".
8: **end if**

Such economization is achieved by invoking in Step 2 the subroutine CONSTRUCTREDUCEDHTSYSTEM, Protocol 6, instead of CONSTRUCTHTSYSTEM, as done in Protocol 2. We proceed to describe Protocol 6.

It begins with P_1 constructing an $\mathcal{F}$-encryption of the full HT system (Steps 1-17). This construction is identical to the one that T performs in Steps 4-20 of Protocol 3, except that in Steps 7 and 13, P_1 inserts the value $\mathcal{F}(1)$, while in the equivalent steps in Protocol 3 (Steps 10 and 16), T inserted the value $\mathcal{F}(\chi_{e,f})$. Indeed, in Protocol 3 we had to use $\mathcal{F}(\chi_{e,f})$ so that the system that T constructs will have all irrelevant equations zeroed (but still included in the system which is then passed for solvability testing). Here, on the other hand, those equations will be removed from the system before passing on to the solvability testing stage. Hence, as at this stage we only construct an encryption of the full HT system for K_V, we can use here the value $\mathcal{F}(1)$ in those entries.

After completing the construction of the full encrypted system, P_1 sends to T the system after secretly permuting its equations and unknowns (Steps 18-19). Those permutations are needed so that when T removes equations and unknowns, he will not know which equations and unknowns were removed, thus preventing T from learning information on E. (However, T may still learn information on the size of E; we discuss this information leakage later on.)

Next, we proceed to the stage in which we eliminate the irrelevant equations and unknowns. This is done in Steps 20-27. First, T generates a key pair in a probabilistic additively homomorphic cipher, $\mathcal{E}$, over $\mathbf{F}_2$, and he notifies P_1 and P_2 of its encryption key (Step 20).

The subsequent Steps 21-23 result in the removal of irrelevant equations. (Those are exactly the same equations that were zeroed in Protocol 3 by using the value $\mathcal{F}(\chi_{e,f})$ in place of all 1-entries.) First (Step 21), P_1 and P_2 create a vector $\mathbf{u}$ which includes an entry for each pair of edges $\{e, f\} \in K_2^{ind}$, where the entry for $\{e, f\}$ is $\mathcal{E}(1)$ if that pair of edges is also in E_2^{ind} and $\mathcal{E}(0)$ otherwise. That computation is done exactly as it was done in Steps 1-2

of Protocol 3, only that the encryption function here is $\mathcal{E}$ (as opposed to $\mathcal{F}$ as was the case in Protocol 3). Then (Step 22), P_2 sends to T a $\pi_{\mathbf{r}}$-permutation of that vector. In Step 23 T proceeds to decrypt that vector and use the recovered binary flags to remove irrelevant rows from the full system that he had obtained earlier (Step 19) from P_1. Note that the same permutation $\pi_{\mathbf{r}}$ was applied by P_1 on the rows of the system (Step 19) and by P_2 on the entries of $\mathbf{u}$ (Step 22).

In Step 24-27, a similar procedure is implemented in order to enable T to remove columns that are irrelevant for the HT system for G. Recall that the full system includes a column for every unknown $x_{e,v}$, where e is any edge in the complete graph (namely, e is any unordered pair of vertices from V_2, see Eq. (6)) while v is any vertex not adjacent to e (namely, v is any vertex from $V \setminus e$). An unknown $x_{e,v}$ should be retained if $e \in E = E_1 \cup E_2$. In order to perform such elimination, P_1 and P_2 compute in Step 24 for each edge $e \in V_2$ the value $\mathcal{E}(\chi_e)$, where χ_e is a binary flag indicating whether $e \in E$ (1) or not (0).

As before, let α_e^h be the Boolean variable indicating whether $e \in E_h$ or not, $h \in \{1, 2\}$. Then

$$\chi_e := \alpha_e^1 \vee \alpha_e^2 = \alpha_e^1 + \alpha_e^2 - \alpha_e^1 \cdot \alpha_e^2 \,.$$

Hence, by the homomorphism of $\mathcal{E}$,

$$\mathcal{E}(\chi_e) = \mathcal{E}(\alpha_e^1) \cdot \mathcal{E}(\alpha_e^2) \cdot \mathcal{E}(-\alpha_e^1 \cdot \alpha_e^2) = \mathcal{E}(\alpha_e^1) \cdot \mathcal{E}(\alpha_e^2) \cdot \mathcal{E}(\alpha_e^1)^{-\alpha_e^2} \,. \tag{12}$$

In view of the above, P_1 sends to P_2 the values $\mathcal{E}(\alpha_e^1)$ for all $e \in V_2$. Then, P_2 proceeds to compute $\mathcal{E}(\chi_e)$ using Eq. (12).

Then, in Step 25, P_2 computes a vector $\mathbf{u}$ that includes the value $\mathcal{E}(\chi_e)$ for every pair (e, v) where $e \in V_2$ and $v \in V \setminus e$. The order of the entries in $\mathbf{u}$ is as determined by $\pi_{\mathbf{c}}$. We note that each encrypted value $\mathcal{E}(\chi_e)$ that was computed in Step 24 will appear in $\mathbf{u}$ exactly $n-2$ times, since there are $n-2$ variables of the form $x_{e,v}$ (since there are $n-2$ vertices v that are not adjacent to e). To prevent T from being able to detect the groups of variables that correspond to the same edge e, P_2 uses the basic encryption $\mathcal{E}(\chi_e)$ obtained in Step 24 for each edge e and he creates from it $n-3$ additional independent encryptions by multiplying it with random encryptions of zero, $\mathcal{E}(0)$. Subsequently, P_2 sends the vector $\mathbf{u}$ to T, who proceeds to decrypt it and use it in order to reduce the number of unknowns (Steps 26-27). After performing those two reductions, T gets an encryption of the system in Eq. (1). That is the system on which the procedure DECIDESOLVABILITY is applied (Step 3 of Protocol 5).

As explained at the end of Section 6.4, Protocol 5 is significantly more efficient than Protocol 2 since the Gaussian elimination that it performs is over the reduced HT system of linear equations and its computational cost is $O(n^6)$ (as opposed to $O(n^{10})$ which was the cost of the Gaussian elimination in Protocol 2). However, Protocol 5 has two disadvantages in comparison to Protocol 2. First, it has larger communication costs, as P_1 needs to transfer to T $O(n^7)$ bits. In addition, Protocol 5 enables T to infer $|E|$ (as the final number of columns equals $|E| \cdot (n-2)$). If the latter value is deemed sensitive, P_1 and P_2 can obfuscate it by sending to T information that will result in keeping unnecessary columns, i.e., columns relating to variables $x_{e,v}$ where $e \notin E$. Such course of action will increase the computational cost, but will prevent T from inferring $|E|$.

Before we conclude our discussion of Protocol 5, we note that the full HT matrix has $k_a = N_{K_V} = O(n^4)$ rows and $k_b + 1 = \binom{n}{2} \cdot (n-2) + 1 = O(n^3)$ columns. By examining the structure of the matrix, see Eq. (3), each of the k_a rows has either four or five 1-entries, while the remaining entries are 0. Hence, in the encrypted matrix that P_1 sends to T in Step 19 of Protocol 6, there will be $O(n^4)$ entries that equal $\mathcal{F}(1)$ and $O(n^7)$ entries that equal $\mathcal{F}(0)$. Since encryptions are a costly operation, it is tempting to compute $\mathcal{F}(0)$ and

Protocol 6 CONSTRUCTREDUCEDHTSYSTEM: Constructing an encryption of the reduced HT system

1: **for all** $\{e, f\} \in K_2^{ind}$, where $e = e_{i,j}$, $f = e_{k,\ell}$, $i < j$, $k < \ell$ and $i < k$ **do**
2: P_1 allocates a vector $\mathbf{r}_{e,f}$ of dimension $N + 1$ where $N := \binom{n}{2} \cdot (n - 2)$.
3: P_1 creates a bijection $\Phi : [N] \to \{(g, v) : g \in V_2, v \in V \setminus \{a(g), b(g)\}\}$.
4: **for** $i \in [N]$ **do**
5: $(g, v) \leftarrow \Phi(i)$
6: **if** ($g = e$ and $v \in \{a(f), b(f)\}$) or ($g = f$ and $v \in \{a(e), b(e)\}$) **then**
7: $\mathbf{r}_{e,f}(i) \leftarrow \mathcal{F}(1)$
8: **else**
9: $\mathbf{r}_{e,f}(i) \leftarrow \mathcal{F}(0)$
10: **end if**
11: **end for**
12: **if** $i < k < j < \ell$ **then**
13: $\mathbf{r}_{e,f}(N + 1) \leftarrow \mathcal{F}(1)$
14: **else**
15: $\mathbf{r}_{e,f}(N + 1) \leftarrow \mathcal{F}(0)$
16: **end if**
17: **end for**
18: P_1 and P_2 agree on random permutations $\pi_{\mathbf{r}}$ and $\pi_{\mathbf{c}}$ on the N_{K_V} equations and $N = \binom{n}{2} \cdot (n - 2)$ unknowns of the full HT system.
19: P_1 sends to T the $\mathcal{F}$-encrypted system that he constructed in the loop above, after permuting its equations and unknowns using $\pi_{\mathbf{r}}$ and $\pi_{\mathbf{c}}$.
20: T generates a key pair in a probabilistic additively homomorphic cipher, $\mathcal{E}$, over $\mathbf{F}_2$. T notifies P_1 and P_2 of the public encryption key in $\mathcal{E}$.
21: P_1 and P_2 compute the vector $\mathbf{u} := (\mathcal{E}(\chi_{e,f}) : \{e, f\} \in K_2^{ind})$, where $\chi_{e,f}$ is a binary flag indicating whether $\{e, f\} \in E_2^{ind}$ (1) or not (0).
22: P_2 permutes the entries of $\mathbf{u}$ using $\pi_{\mathbf{r}}$ and sends the permuted vector to T.
23: T computes $\mathbf{v} := \mathcal{E}^{-1}(\mathbf{u})$ and then he removes from the matrix all rows that correspond to 0-entries in $\mathbf{v}$.
24: P_1 and P_2 compute for each edge $e \in V_2$ in the complete graph K_V the value $\mathcal{E}(\chi_e)$, where χ_e is a binary flag indicating whether $e \in E$ (1) or not (0).
25: P_2 creates a vector $\mathbf{u}$ that includes the value $\mathcal{E}(\chi_e)$ for every pair (e, v) where $e \in V_2$ and $v \in V \setminus e$. The order of the entries in $\mathbf{u}$ is as determined by $\pi_{\mathbf{c}}$.
26: P_2 sends $\mathbf{u}$ to T.
27: T computes $\mathbf{v} := \mathcal{E}^{-1}(\mathbf{u})$ and then he removes from the matrix all columns that correspond to 0-entries in $\mathbf{v}$.

$\mathcal{F}(1)$ just once and reuse those encrypted values wherever necessary. Alas, such a course of action will enable T to fully recover the plaintext matrix. Hence, encrypted values must not be reused. However, it is possible to generate those $O(n^7)$ encrypted entries by performing only $O(n^{3.5})$ encryptions and then rely on the homomorphic property of $\mathcal{F}$ (which implies that $\mathcal{F}(0)\cdot\mathcal{F}(0) = \mathcal{F}(0)$ and $\mathcal{F}(0)\cdot\mathcal{F}(1) = \mathcal{F}(1)$) in order to fill up all entries in the encrypted matrix without reusing encrypted entries.

8 Protocols for a general number of parties

In this section we discuss the changes needed in order to extend our protocols to the case of $d > 2$ parties. Namely, we consider a setting with d parties, $P_1, \ldots, P_d$; P_h has a graph $G_h = (V, E_h)$ where V is publicly known and shared by all, while the edge set E_h is private, $1 \le h \le d$. The goal is to determine whether the union graph $G = (V, E)$, $E = \bigcup_{h=1}^{d} E_h$, is planar or not.

8.1 Testing the size of the unified edge set

In this setting, we have

$$E^c = \left(\bigcup_{h=1}^{d} E_h\right)^c = \bigcap_{h=1}^{d} E_h^c\,,$$

where, as before, for any subset $A \subseteq V_2$, $A^c := V_2 \setminus A$ is its complement within V_2 (Eq. (6)). The unified graph $G = (V, E)$ is planar only if $|E| \le 3n - 6$, namely, only if

$$\left|\bigcap_{h=1}^{d} E_h^c\right| \ge \binom{n}{2} - (3n-6)\,. \tag{13}$$

Protocol 7 is a straightforward generalization of Protocol 1 for the purpose of verifying inequality (13).

Protocol 7 Testing the size of the unified edge set; general d

1: $P_1, \ldots, P_d$ select jointly a large multiplicative group $\mathbf{Z}_p^*$ (p is prime) and a hash function H whose range can be embedded in $\mathbf{Z}_p^*$.
2: P_h selects a secret and random exponent $1 < \alpha_h < p-1$, $1 \le h \le d$.
3: **for** $1 \le h \le d$ **do**
4: P_{h-1} sends to T a vector $\mathbf{y}_h$ of length $\binom{n}{2}$ where, for each edge $(v_i, v_j) \in E_h^c$, $i < j$, $\mathbf{y}_h$ includes an entry of the form $H(i,j)^{\alpha}$, for $\alpha = \prod_{h=1}^{d} \alpha_h$, while the remaining $\binom{n}{2} - |E_h^c|$ entries distribute uniformly over $\mathbf{Z}_p^*$. The order of $\mathbf{y}_h$'s entries is random. Here, when $h = 1$, P_{h-1} is P_d.
5: **end for**
6: T compares the d received vectors, $\mathbf{y}_1, \ldots, \mathbf{y}_d$, and finds out the number z of values that appear in all of them.
7: If $z < \binom{n}{2} - (3n-6)$, T notifies $P_1, \ldots, P_d$ that the union graph is not planar.

The only step that calls for explanation is Step 4. For the sake of simplicity we explain how it is performed when $h = 1$. In similarity to Step 3 in Protocol 1, P_1 sends to P_2 a vector $\mathbf{x}_1$ of length $\binom{n}{2}$ where, for each edge $(v_i, v_j) \in E_1^c$, $i < j$, $\mathbf{x}_1$ includes an entry of the form $H(i,j)^{\alpha_1}$, while the remaining $\binom{n}{2} - |E_1^c|$ entries are randomly selected from $\mathbf{Z}_p^*$.

The order of $\mathbf{x}_1$'s entries is random. Then, P_2 raises each entry in the received vector $\mathbf{x}_1$ to the power α_2 and he sends the resulting vector to P_3. This loop proceeds until the vector reaches P_d that adds his own exponentiation layer by his secret exponent α_d. After doing this, P_d sends the resulting vector, denoted $\mathbf{y}_1$, to T. Note that each entry in that vector that corresponds to an edge $(v_i, v_j) \in E_1^c$ will store the value $H(i,j)^\alpha$, for $\alpha = \prod_{h=1}^{d} \alpha_h$. On the other hand, every other entry distributes uniformly at random over $\mathbf{Z}_p^*$ since its initial setting was to a random element in $\mathbf{Z}_p^*$, and exponentiations by the intermediate powers $1 < \alpha_h < p-1$, $2 \leq h \leq d$, maintain the uniform distribution.

The security of Protocol 7 follows from the hardness of the Discrete Log problem. The protocol entails a total of $O(dn^2)$ hash function evaluations and exponentiations (for $P_1, \ldots, P_d$) and $O(dn^2 \log n)$ comparisons for T. The protocol has d rounds of communication in which $O(dn^2 \log p)$ bits are transmitted.

In similarity to Protocol 1, also Protocol 7 reveals to T the size of E. To mitigate this information leakage, $P_1, \ldots, P_d$ may add random entries to their vectors, as discussed in Section 5.

8.2 Private planarity testing

Here we discuss the extension of our more efficient protocol, Protocol 5, to the case of a general number of parties. The extension of Protocol 2 to any number of parties is similar and is therefore omitted.

Protocol 5 remains unchanged, except that CONSTRUCTREDUCEDHTSYSTEM (the procedure that it invokes in Step 2) is now executed by all $d+1$ parties, $P_1, \ldots, P_d$ and T. We proceed to discuss the needed modifications in that procedure, as implemented in Protocol 6. Before doing so, we note that the second procedure in Protocol 5 – DECIDESOLVABILITY (invoked in Step 3), is indifferent to d and also in the case of a general d it will be executed by P_1 and T.

The loop in Steps 1-17 in Protocol 6 is indifferent to d and, hence, remains unchanged. Also Steps 18-19 remain the same: $P_1, \ldots, P_d$ select jointly the row and column permutations, and then P_1 sends the doubly-permuted encrypted matrix to T.

The purpose of the remaining computations (Steps 20-27) is to enable T to know which rows and which columns from the encrypted matrix that he had received from P_1 he should get rid of. Towards that goal, T generates a homomorphic cipher $\mathcal{E}$ and then the main computation goals of $P_1, \ldots, P_d$ in Steps 20-27 are:

1. For each $e \in V_2$, Eq. (6), compute $\mathcal{E}(\chi_e)$, where $\chi_e := \bigvee_{h=1}^{d} \alpha_e^h$, and α_e^h is the Boolean variable denoting whether $e \in E_h$ or not.
2. For each $\{e, f\} \in K_2^{ind}$, compute $\mathcal{E}(\chi_{e,f})$, where $\chi_{e,f} = \chi_e \cdot \chi_f$.

When $d = 2$, we were able to translate the Boolean expressions that define χ_e and $\chi_{e,f}$ into arithmetic expressions in the private Boolean variables, α_e^h, $h = 1, 2$, $e \in V_2$. We could do the same also for higher values of d, but then the arithmetic expressions become more complex and less manageable. In order to simplify those computations, we suggest to use here the homomorphic encryption due to Boneh, Goh and Nissim [8] (BGN hereinafter). Using that special homomorphic encryption, it is possible to perform any number of additions in the ciphertext domain, as is the case with any additively homomorphic cipher; but, in addition, it is possible to perform also a single multiplication in the ciphertext domain.

In the BGN cipher there are two cyclic groups, $\mathbf{G}$ and $\mathbf{G}_1$, and a bilinear map between them, $\theta : \mathbf{G} \times \mathbf{G} \mapsto \mathbf{G}_1$ (i.e., for all $u, v \in \mathbf{G}$ and $a, b \in \mathbf{Z}$, we have $e(u^a, v^b) = e(u, v)^{ab}$).

Both groups are of order $\nu = q_1 q_2$, where q_1 and q_2 are two large primes. There are two encryption functions, one in $\mathbf{G}$ and one in $\mathbf{G}_1$, as described below:

- The encryption function $\mathcal{E} : \mathbf{Z}_\nu \to \mathbf{G}$. Let g be a generator of $\mathbf{G}$ and let h be an element of order q_1 in $\mathbf{G}$. Then, for any plaintext $m \in \mathbf{Z}_\nu$, its $\mathcal{E}$-encryption is given by $\mathcal{E}(m) = g^m h^r$, for some random $r \in \mathbf{F}_{q_1}$.
- The encryption function $\mathcal{E}_1 : \mathbf{Z}_\nu \to \mathbf{G}_1$. Define $g_1 = \theta(g, g)$ and $h_1 = \theta(g, h)$. It is easy to see that g_1 is a generator of $\mathbf{G}_1$ and h_1 is an element of order q_1 in $\mathbf{G}_1$. Then, for any plaintext $m \in \mathbf{Z}_\nu$, its $\mathcal{E}_1$-encryption is given by $\mathcal{E}_1(m) = g_1^m h_1^r$, for some random $r \in \mathbf{F}_{q_1}$.

Hence, the public key consists of ν, $\mathbf{G}$, $\mathbf{G}_1$, g, h, g_1 and h_1. The private key, on the other hand, is q_1. In order to decrypt $c := \mathcal{E}(m) = g^m h^r$, the owner of the private key q_1 computes c^{q_1}. Since h is of order q_1, we get $c^{q_1} = (g^{q_1})^m$. To recover m from c^{q_1} it is necessary to compute the discrete logarithm of c^{q_1} with respect to the base g^{q_1}. Such a computation is feasible only when the plaintext domain is sufficiently small to allow an efficient solution of that problem (say, by holding a pre-computed table of $(g^{q_1})^m$ for all possible values of m). Similarly, to decrypt $c_1 := \mathcal{E}_1(m) = g_1^m h_1^r$, it is needed to find the value m for which $c_1^{q_1} = (g_1^{q_1})^m$.

In our case, as we shall see below, the range of possible plaintexts is large, in which case decryption is infeasible. However, in our implementation of the BGN cipher, the owner of the private key needs only to determine if the ciphertext corresponds to the plaintext $m = 0$ or not. Such a simplified question is easy to answer. The plaintext m equals zero if and only if $c = \mathcal{E}(m)$ (or $c_1 = \mathcal{E}_1(m)$) satisfy $c^{q_1} = 1$ (or $c_1^{q_1} = 1$).

We are now ready to describe the manner which we suggest to perform the computation that was previously done in Steps 20-27 of Protocol 6. We suggest to replace Steps 20-27 in Protocol 6 with the procedure that is described in Protocol 8. The main difference between Protocol 8 and Steps 20-27 in Protocol 6 is as follows: in Protocol 6 (for $d = 2$) we computed encryptions of the Boolean flags $\chi_{e,f}$ and χ_e that indicated the relevance of equations and unknowns, respectively, in the linear system; in Protocol 8 (for general d), on the other hand, we compute encryptions of integers that represent those Boolean flags.

Specifically, in Steps 2-4 of Protocol 8, $P_1, \ldots, P_d$ compute an $\mathcal{E}$-encryption of the integer $r_e \alpha_e$, where r_e is any random multiplier from $\mathbf{F}_\nu^*$ while $\alpha_e = \sum_{h=1}^d \alpha_e^h$ is the sum of the private Boolean flags. Clearly, if $\chi_e = \bigvee_{h=1}^d \alpha_e^h = 0$ then $r_e \alpha_e = 0$. On the other hand, if $\chi_e = 1$ then $\alpha_e \in [1, d]$. Since the modulus $\nu = q_1 q_2$ is a product of two large primes, then $d < q_1, q_2$ and, therefore, $\alpha_e \in \mathbf{F}_\nu^*$. Since r_e is selected uniformly at random from $\mathbf{F}_\nu^*$, it follows that also $r_e \alpha_e$ distributes uniformly in $\mathbf{F}_\nu^*$. Hence, the value $r_e \alpha_e$ discloses the value of the Boolean flag χ_e, but nothing beyond that. If $r_e \alpha_e = 0$ then $\chi_e = 0$; if $r_e \alpha_e \neq 0$ then $\chi_e = 1$, but the value of $r_e \alpha_e$ in that case does not reveal any further information.

Similarly, in Step 5 we compute an $\mathcal{E}_1$-encryption of the integer $r_e r_f \alpha_e \alpha_f$. As argued above, if $\chi_{e,f} = \alpha_e \cdot \alpha_f = 0$ then also $r_e r_f \alpha_e \alpha_f = 0$, while otherwise $r_e r_f \alpha_e \alpha_f$ distributes uniformly in $\mathbf{F}_\nu^*$. Hence, the value $r_e r_f \alpha_e \alpha_f$ discloses the value of the Boolean flag $\chi_{e,f}$, but nothing beyond that.

Therefore, the encrypted vector $\mathbf{u}$ that P_d sends to T in Step 6 enables T to identify from it the rows that can be removed from the matrix; those are the rows that correspond to entries in $\mathbf{v} := \mathbf{u}^{q_1}$ which equal 1, as such values in $\mathbf{v}$ indicate that the plaintext corresponding to those entries in $\mathbf{u}$ was zero (Step 7). Similarly, the encrypted vector $\mathbf{u}$ that P_d computes in Step 8 and sends to T in Step 9 enables T to detect the columns that can be removed from the matrix (Step 10). We note that the vector $\mathbf{u}$ that is computed in Step 8 has repeated

values, since each entry of the form $\mathcal{E}(r_e\alpha_e)$ appears in $n-2$ entries in the vector. To prevent T from identifying which $n-2$ entries correspond to the same edge, it is possible to use the basic encryption $\mathcal{E}(r_e\alpha_e)$ that was generated in Step 4 in order to create $n-3$ additional $\mathcal{E}$-encryptions of $r_e\alpha_e$ by using homomorphic additions with random encryptions of zero, $\mathcal{E}(0)$.

Protocol 8 Eliminating irrelevant equations and unknowns from the full HT system; general d

1: T generates a BGN cipher and he notifies $P_1, \ldots, P_d$ of the corresponding encryption keys.
2: P_h, $1 \leq h \leq d$, sends to P_d the values $\mathcal{E}(\alpha_e^h)$ for all $e \in V_2$.
3: P_d performs homomorphic addition and computes $\mathcal{E}(\alpha_e)$ for all $e \in V_2$, where $\alpha_e = \sum_{h=1}^{d} \alpha_e^h$.
4: P_d selects for each $e \in V_2$ a random multiplier $r_e \in \mathbf{F}_\nu^*$ and computes $\mathcal{E}(\alpha_e)^{r_e} = \mathcal{E}(r_e\alpha_e)$.
5: For each pair of independent edges $\{e, f\} \in K_2^{ind}$, P_d performs homomorphic multiplication of $\mathcal{E}(r_e\alpha_e)$ and $\mathcal{E}(r_f\alpha_f)$ in order to get $\mathcal{E}_1(r_e r_f \alpha_e \alpha_f)$.
6: P_d sends to T the vector $\mathbf{u} := (\mathcal{E}_1(r_e r_f \alpha_e \alpha_f) : \{e, f\} \in K_2^{ind})$, where the entries are ordered by $\pi_{\mathbf{r}}$.
7: T computes $\mathbf{v} := \mathbf{u}^{q_1}$ and then he removes from the matrix all rows that correspond to 1-entries in $\mathbf{v}$.
8: P_d creates a vector $\mathbf{u}$ that includes the value $\mathcal{E}(r_e\alpha_e)$ for every pair (e, v) where $e \in V_2$ and $v \in V \setminus e$. The order of the entries in $\mathbf{u}$ is as determined by $\pi_{\mathbf{C}}$.
9: P_d sends $\mathbf{u}$ to T.
10: T computes $\mathbf{v} := \mathbf{u}^{q_1}$ and then he removes from the matrix all columns that correspond to 1-entries in $\mathbf{v}$.

9 Privacy-preserving testing of 3-colorability of distributed planar graphs

The 3-colorability decision problem is an NP-complete problem, even for special graph classes, e.g. for triangle-free or $K_{1,5}$-free graphs [29]. However, by Grötzsch Theorem [45], planar graphs that are triangle-free are 3-colorable. (Note that triangle-freeness alone does not imply 3-colorability, as demonstrated by the so-called Grötzsch Graph, see Figure 7.)

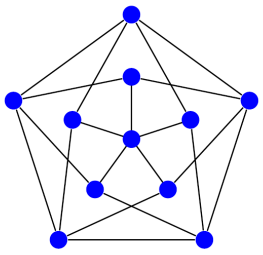

Figure 7: The Grötzsch Graph is a triangle-free graph which is not planar. It is not 3-colorable.

Assume that $P_1, \ldots, P_d$, who hold private graphs $G_h = (V, E_h)$, $1 \leq h \leq d$, already found that the union graph $G = (V, E)$, $E = \bigcup_{h=1}^{d} E_h$, is planar, using Protocol 2 or 5. Assume

that they further need to decide whether it is 3-colorable. Then, by Grötzsch Theorem, it suffices to check whether it is triangle-free.

To this end, let us denote the adjacency matrix of the private graph G_h by A_h, $1 \leq h \leq d$; i.e., $A_h : V^2 \to \{0,1\}$ and $A_h(u,v) = 1$ iff E_h has the edge (u,v). The adjacency matrix of the union graph $G = (V,E)$ is given by $A = \bigvee_{h=1}^{d} A_h$. The matrix $B = \sum_{h=1}^{d} A_h$, on the other hand, gives for any pair of vertices (u,v) the number of private graphs that have the edge (u,v).

Theorem 1. *The graph $G = (V,E)$, where $E = \bigcup_{h=1}^{d} E_h$, is triangle-free iff $tr(A^3) = tr(B^3) = 0$.*

Before proving Theorem 1, we define the notions of paths and path scenarios:

Definition 2. If $(u,v) \in V^2$ then a path from u to v in the union graph $G = (V,E)$ is a sequence of vertices $\langle w_0 = u, w_1, \ldots, w_{s-1}, w_s = v \rangle$ such that $(w_{i-1}, w_i) \in E$ for all $1 \leq i \leq s$; in that case s is the length of the path. A path scenario is a path as defined above together with a sequence of indices $1 \leq h_1, \ldots, h_s \leq d$ such that $(w_{i-1}, w_i) \in E_{h_i}$ for all $1 \leq i \leq s$.

Lemma 3. *For any integer $k \geq 1$, $A^k(u,v)$ equals the number of paths of length k from u to v, while $B^k(u,v)$ equals the number of path scenarios of length k from u to v.*

Proof of Lemma 3. As both claims of the lemma are clearly true for $k = 1$, we proceed by induction. We start with the first claim regarding $A^k(u,v)$. We have

$$A^k(u,v) = \sum_{w \in V} A(u,w) A^{k-1}(w,v) \,.$$

By induction, $A(u,w)$ equals 1 if there is an edge from u to w in G and 0 otherwise, while $A^{k-1}(w,v)$ is the number of paths of length $k-1$ from w to v in G. Hence, $A(u,w)A^{k-1}(w,v)$ equals the number of paths of length k from u to v in G that start with the edge (u,w). Therefore, by summing those numbers over all $w \in V$ we get the total number of paths of length k from u to v. Similar argumentation applies also for our second claim regarding $B^k(u,v)$. □

Proof of Theorem 1. A graph is triangle-free iff it includes no cycles of length 3, that is, no paths of length 3 from a vertex to itself. Hence, a graph is triangle-free iff all entries on the diagonal of A^3 are zero. Therefore, as all entries in A^3 are non-negative, a graph is triangle-free iff the trace of A^3 is zero. The second claim, that a graph is triangle-free iff $\mathrm{tr}(B^3) = 0$, follows immediately because a path between any given pair of vertices exists iff a path scenario between those two vertices exists. □

In view of the above discussion, the parties $P_1, \ldots, P_d$ need only to verify, in a privacy-preserving manner, whether $\mathrm{tr}(A^3) = 0$ or, equivalently, whether $\mathrm{tr}(B^3) = 0$. We prefer to use the latter condition, since $\mathrm{tr}(B^3)$ can be expressed as a polynomial in private values that $P_1, \ldots, P_d$ hold (namely, the entries of their private adjacency matrices A_h, $1 \leq h \leq d$). To that end, the parties may verify that condition by invoking the BGW protocol [7]. (The equivalent condition that involves the matrix A is harder for verification since the entries of the matrix A are Boolean functions of the private inputs.)

9.1 A birdseye view of the BGW protocol

Consider a setting in which $d > 2$ parties, $P_1, \ldots, P_d$, wish to compute an arithmetic function of private inputs that they hold. Party P_h holds a private vector (or, equivalently, a

matrix) x_i over some finite field $\mathbf{F}$, $1 \leq h \leq d$, and the goal is to compute $y = f(x_1, \ldots, x_d)$, where the function f is a publicly known polynomial in the entries of $x_1, \ldots, x_d$, without disclosing to each other the private inputs. The Ben-Or–Goldwasser–Wigderson (BGW) protocol [7] was designed for this purpose. We proceed to provide a birdseye view of that protocol, with the efficiency improvement of [14]. The security of the protocol was proven in [4].

Let C be some arithmetic circuit that computes f over the finite field $\mathbf{F}$. The field's size must be greater than d as well as greater than the a-priori bound on the input and output values. The circuit consists of two different types of gates: addition gates, and multiplication gates. Let $\alpha_1, \ldots, \alpha_d$ be distinct non-zero elements in $\mathbf{F}$; then α_h will be used as a public identifier of P_h, $1 \leq h \leq d$. The parties preserve the following invariant during the computation: the value of each wire of the circuit is secret-shared using a Shamir's $(t+1)$-out-of-d secret sharing scheme (see [35]), with $t < d/2$. The protocol consists of the following three stages: input sharing phase, circuit emulation phase, and output reconstruction phase.

The input sharing phase. In this phase, each party P_h, $1 \leq h \leq d$, shares his input x_h with all parties: for each scalar entry $x_h(\cdot)$ in his private input x_h he chooses a random polynomial g_h of degree t such that $g_h(0) = x_h(\cdot)$, and he then sends to each party P_k the value $g_h(\alpha_k)$, $1 \leq k \leq d$.

The circuit emulation phase. In this phase, the parties emulate the computation of the circuit C on the inputs $x_1, \ldots, x_d$, where in each gate, the parties compute shares of the value of the output wire using their shares of the input wires by invoking a secure protocol. There are two types of gates to consider: addition gates and multiplication gates.

Addition gates. The computation of the output shares can be performed locally and without any interaction, since if $f_1(\alpha_h)$ and $f_2(\alpha_h)$ are the shares that P_h holds for the two input wires to an addition gate, then $f(\alpha_h) = f_1(\alpha_h) + f_2(\alpha_h)$ is a valid sharing of the output wire. Indeed, if the polynomials $f_1(x)$ and $f_2(x)$ are of degree at most t, then so is their sum $f(x) := f_1(x) + f_2(x)$; and, in addition, $f(0) = f_1(0) + f_2(0)$.

Multiplication gates. The case of multiplication gates is more involved as it requires interaction among the parties. In particular, given shares $f_1(\alpha_h)$ and $f_2(\alpha_h)$ for the two input wires of a multiplication gate, $1 \leq h \leq d$, then $f(\alpha_h) := f_1(\alpha_h) \cdot f_2(\alpha_h)$ are shares of a polynomial $f(x)$ with the correct constant term $f(0) = f_1(0) \cdot f_2(0)$, as required, but its degree could be as high as $2t$, while we need the sharing polynomial to be of degree at most t. Hence, the players must interact in order to reduce the degree of that polynomial. The degree reduction procedure can be done using the method of [14], which is based on the fact that if f is a polynomial of degree at most $d-1$ and $\alpha_1, \ldots, \alpha_d$ are d distinct non-zero points in the field, then the constant term $f(0)$ is a linear combination of the other points on that polynomial. That is, $f_1(0) \cdot f_2(0) = f(0) = \sum_{h=1}^{d} \lambda_h \cdot f(\alpha_h)$, where $\lambda_h := \prod_{k \neq h} \alpha_k / (\alpha_h - \alpha_k)$ are the Lagrange coefficients.

The multiplication sub-protocol proceeds as follows. Given the shares $f_1(\alpha_h), f_2(\alpha_h)$ of the party P_h, the party P_h locally multiplies these two shares and gets the value $f(\alpha_h)$. Then, he chooses a polynomial $g_h(x)$ of degree at most t such that $g_h(0) = f(\alpha_h) = f_1(\alpha_h) \cdot f_2(\alpha_h)$. He then shares the polynomial g_h with all parties, so that each party P_k receives the share $g_h(\alpha_k)$. At the end of this stage, each party P_k holds the shares $g_1(\alpha_k), \ldots, g_d(\alpha_k)$. Next, let us define the polynomial $p(x) := \sum_{h=1}^{d} \lambda_h \cdot g_h(x)$, which has a degree at most t. Each party P_k locally computes the linear combination $\sum_{h=1}^{d} \lambda_h \cdot g_h(\alpha_k) = p(\alpha_k)$, which is his share in the implicitly defined polynomial $p(x)$. Note that $p(x)$ is a polynomial of degree at most t, and $p(0) = \sum_{h=1}^{d} \lambda_h \cdot g_h(0) = \sum_{h=1}^{d} \lambda_h \cdot f(\alpha_h) = f_1(0) \cdot f_2(0)$.

Output reconstruction phase. In this phase, each party P_h receives all the shares of the output wire and he then reconstructs the final output y.

9.2 A protocol for checking whether the trace of B^3 is zero

Protocol 9 performs a privacy-preserving check of the trace of B^3. It starts (Step 1) by selecting the finite field $\mathbf{F}$ over which all computations will take place. The size of the field must be greater than d^3n^3, since that is the upper bound on $\text{tr}(B^3)$. The latter bound follows directly from Lemma 4 below, that can be easily proven by induction.

Lemma 4. *$0 \leq B^k(u,v) \leq d^k n^{k-1}$ for every $u, v \in V$ and $k \geq 1$.*

Proof. The lemma is obviously correct for $k = 1$ since $B(u,v)$ equals the number of private graphs that have the edge (u,v) and that number is within the range $[0,d]$. The proof proceeds by induction:

$$B^k(u,v) = \sum_{w \in V} B^{k-1}(u,w) \cdot B(w,v) \leq \sum_{w \in V} d^{k-1}n^{k-2} \cdot d = d^k n^{k-1} .$$

□

Next (Step 2), each party distributes to all parties (including himself) shares in a Shamir secret sharing scheme in each entry of his private adjacency matrix. The degree t of all interpolation polynomials are selected by all parties upfront. (Note that since all graphs are non-directional, then all adjacency matrices are symmetric and hence it suffices to share and compute only the entries on the diagonal and above it.) After each party collects his respective shares in all adjacency matrices, he proceeds to add them in order to receive a share in each entry of the matrix B (Step 3). In Steps 4-5, the parties proceed to translate their shares into shares in the entries of B^2 and B^3. Since, for any $(u,v) \in V^2$ and $k \geq 2$,

$$B^k(u,v) = \sum_{w \in V} B^{k-1}(u,w)B(w,v)$$

the parties can use the BGW multiplication procedure in order to compute shares in the product $B^{k-1}(u,w)B(w,v)$ (from the shares in $B^{k-1}(u,w)$ and $B(w,v)$ which they already have) and, subsequently, get shares in $B^k(u,v)$. Then the parties compute their share in $\text{tr}(B^3)$ by simple addition (Step 6).

It is left only to reconstruct $\text{tr}(B^3)$ and check whether it is zero or not. Such a conclusion of the protocol is possible, but then the parties would reveal $\text{tr}(B^3)$, which equals the number of cycles of length 3 in G, multiplied by 3 (as each such cycle contributes 1 to three entries on the diagonal of B^3). If such information leakage is considered benign then the protocol could end by such reconstruction among the d interacting parties. If, however, such information leakage is to be avoided, the parties can decide on a random multiplier $q \in \mathbf{F}_p^*$ and then send to the mediator T their shares multiplied by q. T may then interpolate the received shares in order to get $w := q \cdot \text{tr}(B^3)$. Such a value reveals to T (who does not know q) only whether $\text{tr}(B^3) = 0$ or not, but nothing beyond that (Steps 7-10).

Protocol 9 Privacy-preserving testing of the triangle-freeness of a distributed planar graph

1: $P_1, \ldots, P_d$ choose a finite field $\mathbf{F}$ where $|\mathbf{F}| > d^3n^2$.
2: Each P_h, $1 \leq h \leq d$, shares each entry in his own adjacency matrix A_h with all parties.
3: Each P_h computes from the shares he received his own share in each entry of the matrix $B = \sum_{h=1}^{d} A_h$.
4: $P_1, \ldots, P_d$ invoke the BGW protocol to compute shares in each of the entries in B^2.
5: $P_1, \ldots, P_d$ invoke the BGW protocol to compute shares in each of the entries on the diagonal of B^3.
6: Each P_h adds up his shares in the diagonal entries of B^3 in order to get his share s_h in $\mathrm{tr}(B^3)$.
7: $P_1, \ldots, P_d$ select a random nonzero term $q \in \mathbf{F}_p^*$.
8: Each P_h sends to T the value $q \cdot s_h$, $1 \leq h \leq d$.
9: T recovers from the received shares the value $w := q \cdot \mathrm{tr}(B^3)$.
10: T outputs "**The graph is triangle-free**" if $w \neq 0$ and "**The graph is not triangle-free**" otherwise.

Such a protocol does not leak anything to the interacting parties $P_1, \ldots, P_d$, as implied by the security of the BGW protocol [4], and not to T as argued above. As for the computational costs, the main bottleneck is due to the BGW multiplication procedure. In computing shares in B^2's entries there are $O(n^3)$ multiplication procedures (that could be parallelized), while the computation of the diagonal entries of B^3 involves additional $O(n^2)$ multiplication procedures (which could also be parallelized). As the complexity of our planarity-testing protocols is higher than that, then this subsequent computation does not increase the overall complexity.

10 Conclusions

We introduced the problem of privacy-preserving planarity testing of distributed graphs. We presented a protocol that solves this problem. Our protocol, based on the Hanani–Tutte theorem, protects the private edge sets of each of the parties, under the assumption that the parties are semi-honest and do not collude. Our study presents, for the first time, an SMC protocol that privately tests a *geometric* property of distributed graphs.

A related property of graphs is that of outer-planarity: a graph is called *outer-planar* if it has a planar drawing for which all the vertices belong to the outer face of the drawing. It is known [13] that a graph is outer-planar iff the graph that is obtained from it by adding a new vertex and adding from it edges to all original vertices is planar. Hence, our privacy-preserving planarity testing protocols can be easily modified for testing outer-planarity. All that is needed is to augment the vertex set $V = \{v_1, \ldots, v_n\}$ with a new vertex, $V \mapsto V' = V \cup \{v_{n+1}\}$, and that one of the parties, say P_1, adds to his edge set, E_1, the n edges (v_i, v_{n+1}), $1 \leq i \leq n$. Then the parties may proceed by testing the planarity of the new augmented distributed graph on V'.

This study raises the following problems for future research:

(a) Improving scalability, either by devising more efficient ways to test the solvability of the HT system, or by designing a privacy-preserving version of another planarity testing algorithm.

(b) Devising privacy-preserving protocols for solving graph problems, which are known to have an efficient solution in cases where the underlying graph is planar, as we did for 3-colorability. The sub-graph isomorphism and the maximal clique problems are examples of such problems.

(c) Enhancing the resiliency of the protocol to stronger adversarial models (i.e., malicious parties).

(d) Implement mechanisms to prevent coalitions between the interacting parties. In the solutions that we presented herein, one of the parties $P_1, \ldots, P_d$ may collude with the mediator T in order to disclose information on the union graph beyond its planarity. One way to prevent such a coalition is to split the role of the mediator T into k distinct and independent mediators, $T_1, \ldots, T_k$, and enhance our protocols by secret sharing techniques, so that the recovery of information on the union graph requires a collaboration of all k mediators, or of k' mediators, where $1 < k' < k$ (the latter option increases the robustness of the system).

(e) Devising privacy-preserving protocols for computing planar drawings of distributed graphs. Assume that $P_1, \ldots, P_d$ ran the protocols described in the present study and found that the distributed graph which they hold is planar. In such a case a new problem arises: finding a planar drawing of the distributed graph. Namely, the parties should arrive at a bijection φ from the vertex set V to $\mathbf{R}^2$ and a representation of each edge $e = (u, v) \in E$ as a continuous simple curve in $\mathbf{R}^2$ with $\varphi(u)$ and $\varphi(v)$ as its end points, such that no two curves intersect apart possibly at their end points. The bijection φ can be public, as V is public and shared by all parties. However, the curve representing any given edge should be revealed only to parties that have that edge in their private edge set. This is a problem of secure multiparty computation of a new type, as the output consists of continuous curves. We are not aware of any prior art that pushed the envelope of secure multiparty computation to include such types of computation.

References

[1] Joël Alwen, Abhi Shelat, and Ivan Visconti. Collusion-free protocols in the mediated model. In *Advances in Cryptology - CRYPTO, 28th Annual International Cryptology Conference*, pages 497–514, 2008.

[2] Abdelrahaman Aly, Edouard Cuvelier, Sophie Mawet, Olivier Pereira, and Mathieu Van Vyve. Securely solving simple combinatorial graph problems. In *Financial Cryptography and Data Security - 17th International Conference*, pages 239–257, 2013.

[3] Gilad Asharov, Francesco Bonchi, David García-Soriano, and Tamir Tassa. Secure centrality computation over multiple networks. In *Proceedings of the 26th International Conference on World Wide Web*, pages 957–966, 2017.

[4] Gilad Asharov and Yehuda Lindell. A full proof of the BGW protocol for perfectly secure multiparty computation. *J. Cryptology*, 30(1):58–151, 2017.

[5] Baruch Awerbuch and Yossi Shiloach. New connectivity and msf algorithms for shuffle-exchange network and pram. *IEEE Transactions on Computers*, 36(10):1258–1263, 1987.

[6] Guy Barshap and Tamir Tassa. Privacy-preserving planarity testing of distributed graphs. In *Proceedings of Data and Applications Security and Privacy XXXII - 32nd Annual IFIP WG 11.3 Conference (DBSec)*, pages 131–147, 2018.

[7] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 1–10, 1988.

[8] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In *Theory of Cryptography, Second Theory of Cryptography Conference*, pages 325–341, 2005.

[9] John M. Boyer and Wendy J. Myrvold. On the cutting edge: Simplified O(n) planarity by edge addition. *Journal of Graph Algorithms and Applications*, 8(2):241–273, 2004.

[10] Justin Brickell and Vitaly Shmatikov. Privacy-preserving graph algorithms in the semi-honest model. In *Advances in Cryptology - ASIACRYPT, 11th International Conference on the Theory and Application of Cryptology and Information Security*, pages 236–252, 2005.

[11] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22:644–654, 1976.

[12] David Eppstein. Subgraph isomorphism in planar graphs and related problems. *Journal of Graph Algorithms and Applications*, 3(3):1–27, 1999.

[13] Stefan Felsner. Geometric graphs and arrangements: Some chapters from combinatorial geometry. *Combinatorics, Probability & Computing*, 15(6):941–942, 2006.

[14] Rosario Gennaro, Michael O. Rabin, and Tal Rabin. Simplified VSS and fact-track multiparty computations with applications to threshold cryptography. In *Proceedings of the Seventeenth Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 101–111, 1998.

[15] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *Proceedings of the 14th Annual ACM Symposium on Theory of Computing*, pages 365–377, 1982.

[16] Tal Grinshpoun and Tamir Tassa. A privacy-preserving algorithm for distributed constraint optimization. In *International conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 909–916, 2014.

[17] Tal Grinshpoun and Tamir Tassa. P-SyncBB: A privacy preserving branch and bound DCOP algorithm. *Journal of Artificial Intelligence Research*, 57:621–660, 2016.

[18] Tal Grinshpoun, Tamir Tassa, Vadim Levit, and Roie Zivan. Privacy preserving region optimal algorithms for symmetric and asymmetric dcops. *Artificial Intelligence*, 266:27–50, 2019.

[19] Frank Hadlock. Finding a maximum cut of a planar graph in polynomial time. *The SIAM Journal on Computing*, 4(3):221–225, 1975.

[20] John E. Hopcroft and Robert E. Tarjan. Efficient planarity testing. *Journal of the ACM*, 21(4):549–568, 1974.

[21] Arjan Jeckmans, Qiang Tang, and Pieter H. Hartel. Privacy-preserving collaborative filtering based on horizontally partitioned dataset. In *International Conference on Collaboration Technologies and Systems (CTS)*, pages 439–446, 2012.

[22] Wei Jiang and Chris Clifton. A secure distributed framework for achieving k-anonymity. *The VLDB Journal*, 15:316–333, 2006.

[23] Murat Kantarcioglu and Chris Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. *IEEE Transactions on Knowledge and Data Engineering*, 16:1026–1037, 2004.

[24] Marcel Keller and Peter Scholl. Efficient, oblivious data structures for MPC. In *Advances in Cryptology - ASIACRYPT - 20th International Conference on the Theory and Application of Cryptology and Information Security*, pages 506–525, 2014.

[25] Kazimierz Kuratowski. Sur le problème des courbes gauches en topologie. *Fundamenta Mathematicae*, 15:271–283, 1930.

[26] Peeter Laud. Parallel oblivious array access for secure multiparty computation and privacy-preserving minimum spanning trees. *Proceedings on Privacy Enhancing Technologies (PoPETs)*, 2015(2):188–205, 2015.

[27] Thomas Léauté and Boi Faltings. Protecting privacy through distributed computation in multi-agent decision making. *Journal of Artificial Intelligence Research*, 47:649–695, 2013.

[28] Hsiao-Ying Lin and Wen-Guey Tzeng. An efficient solution to the millionaires' problem based on homomorphic encryption. In *Applied Cryptography and Network Security, Third International Conference (ACNS)*, pages 456–466, 2005.

[29] Frédéric Maffray and Myriam Preissmann. On the np-completeness of the k-colorability prob-

lem for triangle-free graphs. *Discrete Mathematics*, 162(1-3):313–317, 1996.

[30] Catherine A. Meadows. A more efficient cryptographic matchmaking protocol for use in the absence of a continuously available third party. In *IEEE Symposium on Security and Privacy*, pages 134–137, 1986.

[31] Kobbi Nissim and Enav Weinreb. Communication efficient secure linear algebra. In *Theory of Cryptography, Third Theory of Cryptography Conference (TCC)*, pages 522–541, 2006.

[32] Christos H. Papadimitriou and Mihalis Yannakakis. The clique problem for planar graphs. *Information Processing Letters*, 13(4/5):131–133, 1981.

[33] Maurizio Patrignani. Planarity testing and embedding. In *Handbook on Graph Drawing and Visualization*, pages 1–42. 2013.

[34] Marcus Schaefer. Toward a theory of planarity: Hanani-tutte and planarity variants. *Journal of Graph Algorithms and Applications*, 17(4):367–440, 2013.

[35] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[36] Erez Shmueli and Tamir Tassa. Secure multi-party protocols for item-based collaborative filtering. In *Proceedings of the Eleventh ACM Conference on Recommender Systems (RecSys)*, pages 89–97, 2017.

[37] Tamir Tassa. Secure mining of association rules in horizontally distributed databases. *IEEE Transactions on Knowledge and Data Engineering*, 26:970–983, 2014.

[38] Tamir Tassa and Dror J. Cohen. Anonymization of centralized and distributed social networks by sequential clustering. *IEEE Transactions on Knowledge and Data Engineering*, 25:311–324, 2013.

[39] Tamir Tassa, Tal Grinshpoun, and Roie Zivan. Privacy preserving implementation of the max-sum algorithm and its variants. *Journal of Artificial Intelligence Research*, 59:311–349, 2017.

[40] Tamir Tassa and Ehud Gudes. Secure distributed computation of anonymized views of shared databases. *Transactions on Database Systems*, 37, Article 11, 2012.

[41] Tamir Tassa, Roie Zivan, and Tal Grinshpoun. Max-sum goes private. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 425–431, 2015.

[42] Tamir Tassa, Roie Zivan, and Tal Grinshpoun. Preserving privacy in region optimal DCOP algorithms. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 496–502, 2016.

[43] Klaus Wagner. Über eine eigenschaft der ebenen komplexe. *Mathematische Annalen*, 114:570–590, 1937.

[44] Andrew C. Yao. Protocols for secure computations (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science*, pages 160–164, 1982.

[45] Dvořák Zdeněk, Kawarabayashi Ken-Ichi, and Thomas Robin. Three-coloring triangle-free planar graphs in linear time. *The ACM Transactions on Algorithms*, 7(4):41:1–41:14, 2011.