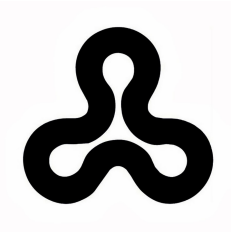


The Open University of Israel
Department of Mathematics and Computer Science

Privacy-preserving planarity testing of distributed graphs

Thesis submitted as partial fulfillment of the requirements
Towards an M.Sc. degree in Computer Science
The Open University of Israel
Computer Science Division

By:
Guy Barshap

Prepared under the supervision of Prof. Tamir Tassa

October 2017

Acknowledgment.

Preparation of this thesis was neither an easy nor a quick task and it would not have been possible without the support of many people. First, I would like to express my sincere appreciation and gratitude to my supervisor, Prof. Tamir Tassa. Tamir taught me diligently how to write an academic paper, with his endless availability and willingness to help me to accomplish this Master degree. I am thankful for absorbing from him a good research attitude and valuable skills that certainly will help me in the future. My gratitude also goes to my family who believe in me and gave me the space that I needed to work on this demanding project. In particular, I would like to thank my father who encouraged me to engage in research, my brother Alon, and my mother for her full support and investment. Last but not least, I wish to thank Talia Szwarc, my girlfriend, who had a significant positive impact on my mental condition during this long-term journey.

Contents

Abstract

In this thesis we present a solution to the problem of privacy preserving graph planarity testing. The setting involves several players 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 in a privacy preserving manner. Namely, each player wishes to protect his private edge set from the other players. We present two privacy preserving algorithms that are based on the Hanani-Tutte (HT) theorem and have polynomial runtime. The HT Theorem translates the planarity question into the question of whether a specific system of linear equations over the binary field, $\mathbb{F}_2$, is solvable. Our algorithms use techniques such as secure rank computation and oblivious Gaussian elimination as subroutines. This is the first time that a solution to this problem is presented.

1 Introduction

A planar graph $G = (V, E)$ is a graph that can be embedded in the two-dimensional plane $\mathbb{R}^2$. Namely, there exists a bijection φ from V to $\mathbb{R}^2$ and a representation of each edge $e = (u, v) \in E$ as a continuous simple curve in $\mathbb{R}^2$ with $\varphi(u)$ and $\varphi(v)$ as its end points, such that no two curves intersect apart possibly for their end points.

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 if and only if 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 highways can be thought of as a graph; the graph vertices are the cities while the edges are connecting highways. Such a map can be constructed with non-crossing highways (in order to avoid constructing bridges or obstructing the traffic flow by stop lights) iff the corresponding graph is planar.

A less known fact is that planar graphs appear in many chemical applications. A graph is called *chemical* if it describes a chemical molecule, where the vertices correspond to atoms and the edges correspond to their chemical bonds. The publicly available NCI chemical dataset[1], which is commonly used as benchmark in graph mining, describes a large group of pharmaceutical compounds. Out of those compounds, 94.3% elements are described by a chemical graph that are outer-planar (a sub-class of planar graphs, that we describe and discuss in Section 5), see [25]. Such graphs are important in computational drug design [25]. Another example of chemical graphs are graphs that represent the $contact$ structure of bi-polymers, where the vertices are the monomers and the edges are the covalent bonds. The DNA and RNA molecules form a special type of $contact$ structure, called secondary structures, and those graphs are also outer-planar [31].

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 [15], maximal clique [37], and maximum cut [22]. The subgraph isomorphism problem is a computational decision problem in which two graphs G and H are given as input, and one must determine whether G contains a subgraph that is isomorphic to H. In case that G is a general graph and H is a fixed graph with k vertices, the running time is known to be polynomial. But when G is planar and H is fixed, the running time of subgraph isomorphism can be reduced to linear time [15]. The problem of finding a maximal clique with the largest possible number of vertices in a general graph is known to be a NP-Complete, but for planar graphs it can be solved in linear time [37]. In the decision maximum cut problem (Max-Cut), defined for a given a graph G and an integer k, it is needed to determine whether there is a cut of size at least k in G. As the Max-Cut Problem is NP-complete, no polynomial-time algorithms are known for solving it in general graphs. However, for planar graphs the computational cost can be reduced to a polynomial time [22].

The problem of planarity testing, namely, deciding whether a given graph is planar or not, is well-studied and well-understood. Optimal linear time planarity testing algorithms were proposed in [24] [11], but it seems unpractical to apply them in a privacy-preserving manner due to theirs inner mechanism.

In this study we consider a distributed version of the planarity testing problem. In that problem

[1] `http://cactus.nci.nih.gov/`

there exist several players, $P_1, \ldots, P_d$, each one holding a private graph on the same set of vertices; namely, P_i has a graph $G_i = (V, E_i)$ where V is publicly known and shared by all, while E_i is private, $1 \leq i \leq d$. They wish to determine whether the union graph $G = (V, E)$, where $E = \bigcup_{i=1}^{d} E_i$, is planar or not. As the edge sets E_i, $1 \leq i \leq d$, are private, that planarity testing should be carried out in a privacy-preserving manner. We propose here secure protocols for that purpose which are based on the Hanani-Tutte theorem [39].

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_i holds a private value x_i. 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 its 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 secure multi-party computations 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 [12] 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 secure multi-party computation techniques for the shortest path and the maximum flow problems. Another example is the work of [27] who presented oblivious implementations of several data structures for secure multi-party computation 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 [30]. 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 focus here on the case of $d = 2$ players; the extension to any number d of players is straightforward. Our protocols are in the mediated model that was presented in [1]. In that model, there exists an external mediator T to which the players may export some computations, but the mediator should not learn information on the private inputs of the players or the final output. We assume that all interacting parties (the players as well as the mediator) are semi-honest. Namely, they follow the protocol specification, but they try to extract from their view in the protocol information on the private inputs of other players.

The outline of this work is as follows. In Section 2 we provide the relevant background on planarity testing. Then, in Section 3 we cover the relevant cryptographic background: Oblivious transfer, Yao's Millionaires' problem, Yao's garbled circuit protocol, the BGW protocol, and homomorphic encryption. Apart from those general purpose tools, we also describe the work of [35] about oblivious testing of the solvability of an encrypted linear system of equations, and the work of [28] about computing the rank of an encrypted matrix. Section 4 holds the main part of this work — our novel privacy-preserving planarity testing protocols. In Section 5 we describe further results, regarding graph coloring and testing outer-planarity, that rely on the protocols described in the previous section. We conclude in Section 6.

2 Background: Planarity testing

A graph $G = (V, E)$ is called *planar* if it is possible to draw it in the Euclidean plane with no crossings among the edges. Well known characterizations for planar graphs were proposed by both Wagner [43] and Kuratowski [29]. For example, the Wagner's characterization states that a graph is planar iff it does not have the graphs K_5 or $K_{3,3}$ as a minor (see Figure1). 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 or Kuratowski's characterizations in order to test the planarity of a given graph yield exponential-time algorithms [38]. Moreover, there exist linear-time algorithms for testing planarity, but they are rather iterative or using a DFS-subroutine [38]. However, 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 algorithm.

Figure 1: The minors that cannot appear in a planar graph — K_5 and $K_{3,3}$.

In this paper we focus on testing the planarity via Hanani-Tutte theorem. Before we introduce the theorem, we give some preliminary definitions.

- Every graph $G = (V, E)$ has many possible embeddings in $\mathbb{R}^2$; each such embedding is called a drawing and is denoted by D.
- Independent edges: edges that have no vertex in common. We let E_2^{ind} denote hereinafter the set of all pairs of independent edges (e, f) in E.
- $iocr(D)$: the number of pairs of *independent* edges of G that cross in D an odd number of times.
- $iocr(G)$ is the minimum of $iocr(D)$ over all drawings D of G.
- If $e \in E$ we let $a(e)$ and $b(e)$ denote the two vertices that it connects.
- For a drawing D and a pair of edges e and f, $parity_D(e, f)$ denotes the parity of the number of crossings between e and f in D.
- Let D be a drawing of G. Then for a given edge e and a vertex v, an (e, v)-move consists of taking a small section of e and deforming it in a narrow tunnel to make it pass over v (while avoiding passing over vertices other than v). 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 and remain unchanged for all other edges f (see Figure 2).

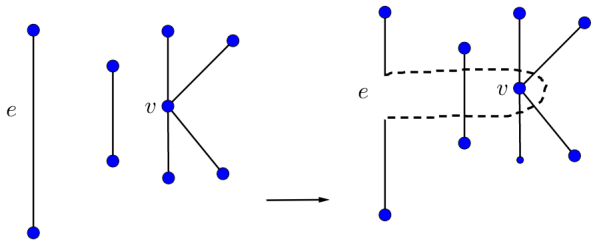


Figure 2: Performing an (e, v)-move.

Theorem 2.1 (Hanani-Tutte)**.** *A graph G is planar iff $iocr(G) = 0$. Namely, G is planar iff it has a drawing in which every two independent edges cross evenly.*

The Hanani-Tutte Theorem gives rise to the following planarity testing algorithm. It 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, such that $iocr(D') = 0$. If it succeeds, then the graph is planar, otherwise it is not. WLOG, it is assumed that the position of all vertices remains unchanged; the only changes in D' with respect to D is in the curves that represent the edges.

The existence of D' can be determined by considering the following system of linear equations. Define the following $|E| \cdot (|V| - 2)$ binary variables

$$\{x_{e,v} : e \in E, v \in V \setminus \{a(e), b(e)\}\}\,. \tag{1}$$

The binary variable $x_{e,v}$ will equal 1 iff the transition from D to D' included an (e, v)-move. It is now clear that for any pair $(e, f) \in E_2^{ind}$,

$$parity_{D'}(e, f) = parity_D(e, f) + x_{e,a(f)} + x_{e,b(f)} + x_{f,a(e)} + x_{f,b(e)} \mod 2\,.$$

Hence, given the drawing D, there exists a drawing D' with $iocr(D') = 0$ iff there exists a solution to the system of $|E_2^{ind}|$ equations

$$parity_D(e, f) + x_{e,a(f)} + x_{e,b(f)} + x_{f,a(e)} + x_{f,b(e)} = 0 \mod 2 \qquad (e, f) \in E_2^{ind}\,. \tag{2}$$

Hence, G is planar iff the system of equations (2) has a solution over $\mathbb{F}_2$.

So, to summarize: the algorithm starts with an arbitrary drawing D of G, in which $parity_D(e, f)$ can be computed efficiently for all $(e, f) \in E_2^{ind}$. Then, it attempts to solve the system in Eq. (2).

Let us denote $n = |V|$. It is well known the graph is planar only if $|E| \leq 3n - 6$. Therefore, the number of unknowns in the system in Eq. (2) is $O(n^2)$. This implies that the complexity of the algorithm is $O(n^6)$.

3 Background: Cryptographic tools and protocols

In this section we provide a description of the cryptographic tools and protocols that we will use in our privacy-preserving planarity testing. The first ones are classical and general-purpose building tools: homomorphic encryption (Section 3.1), oblivious transfer (Section 3.2), Yao's protocol for solving the Millionaires' problem [44] (Section 3.3), Yao's garbled circuit protocol (Section 3.4), and the BGW (Ben-Or Goldwasser Wigderson) protocol [7] (Section 3.5). We then describe a protocol for performing Gaussian elimination of an encrypted matrix (Section 3.6), which it is based on [35]. Finally, we describe a protocol for computing the rank of an encrypted matrix (Section 3.7), which is based on [28].

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 allows performing arithmetic computations in the ciphertext domain. A probabilistic encryption function is essential when dealing with plaintexts that come from a small and publicly known domain (such as binary plaintexts, as is the case in our problem). The Paillier cipher [36] is both homomorphic and probabilistic; it is semantically secure under the decisional composite residuosity assumption [36]. Its plaintext domain is the group $\mathbb{Z}_\nu$ for a modulus ν which is a product of two large primes.

An example of a homomorphic cipher over $\mathbb{F}_2$ can be found in [20] (GM). We proceed to describe a generalization of that cipher that works over the extension field $\mathbb{F}_{2^k}$ that was proposed in [17]. Let $p(x)$ be an irreducible polynomial of degree k. Then $\mathbb{F}_{2^k}$ is the set of all polynomials of degree strictly less than k with coefficients in $\mathbb{F}_2$. *Addition* of two polynomial elements in $\mathbb{F}_{2^k}$ is defined as a regular polynomial addition, whereas *multiplication* is defined by multiplying the two polynomials and then taking the residue of the product modulo $p(x)$.

To encrypt an element $a \in \mathbb{F}_{2^k}$, one encrypts separately each coefficient of the polynomial that a describes, with an encryption scheme over $\mathbb{F}_2$ (such as GM). Hence, the *addition of encrypted values* is done coefficient-wise using the homomorphic property of the encryption over $\mathbb{F}_2$, which requires $O(k)$ homomorphic additions of encrypted elements. Performing *encrypted multiplication* of $\mathcal{F}(a \cdot b)$, where $a, b \in \mathbb{F}_{2^k}$, a is given in the clear and b is given in its encrypted form, $\mathcal{F}(b)$, is a bit more complicated. Let $A[x]$ and $B[x]$ be the polynomials with degree strictly less than k which a and b represent, respectively. Then, our goal is to compute $C[x] = A[x]B[x] \mod p(x) = \sum_{j=0}^{2k-2} c_j x^j \mod p(x)$, where $C[x]$ is encrypted. It is easy to see that the multiplication of the publicly known polynomial $A[x]$ with the coefficient-encrypted polynomial $B[x]$ can be carried out using the homomorphic properties of the GM encryption. The more tricky part is to compute the reduction of $C[x]$ modulo the publicly known polynomial $p(x)$.

Let us denote the encrypted coefficients of $C[x]$ by $\mathcal{F}(c_0), \ldots, \mathcal{F}(c_{2k-2})$ and $u_i[x] = x^{k+i} \mod p(x)$ for $0 \leq i \leq k-2$. Since $p(x)$ is publicly known, $u_i[x]$ are publicly computable polynomials of degree $k-1$ with coefficients in $\mathbb{F}_2$. Let $\alpha[x] = c_{k-1}x^{k-1} + \ldots + c_0 x^0$ and let $\alpha_i[x]$ be

defined as $u_i[x]$ if $c_{k+i} = 1$, and 0 otherwise, $0 \leq i \leq k-2$. Then, we need to compute

$$C[x] \mod p(x) = \alpha[x] + \sum_{i=0}^{k-2} \alpha_i[x]\,. \tag{3}$$

Given $u_i[x]$ and $\mathcal{F}(c_{k+i})$ for $0 \leq i \leq k-2$, it is possible to compute the encryption of $\alpha_i[x]$ for $0 \leq i \leq k-2$ using the homomorphic properties of the GM encryption. Finally, the sum on the right hand side of Eq. (3) can be computed via addition of homomorphic encrypted values. The overall computation can be done by $O(k^2)$ operations and it is worth mentioning that once $u_i[x]$ are computed, they can be reused for future homomorphic computations.

Boneh et al. [9] proposed a special encryption function that preserves homomorphism over any number of additions, but, in addition, preserves homomorphism over a single multiplication. Hence, their cipher can be used to compute any quadratic polynomial of the plaintexts in the ciphertext domain. In their method there are two groups, $\mathbb{G}$ and $\mathbb{G}_1$, and a bilinear map between them, $e : \mathbb{G} \times \mathbb{G} \mapsto \mathbb{G}_1$. Both groups are of order $\nu = q_1 q_2$. We can encrypt in either $\mathbb{G}$ or in $\mathbb{G}_1$.

$\mathcal{E}$ — Encryption in $\mathbb{G}$: g is a generator of $\mathbb{G}$ and h is an element of order q_1 in $\mathbb{G}$. Then, we can encrypt every integer in $\mathbb{F}_\nu$ by $m \mapsto g^m h^r$, for some random $r \in \mathbb{F}_{q_1}$.

$\mathcal{E}_1$ — Encryption in $\mathbb{G}_1$: Define $g_1 = e(g, g)$ and $h_1 = e(g, h)$. Then g_1 is a generator of $\mathbb{G}_1$ and h_1 is an element of order q_1 in $\mathbb{G}_1$. Then we can encrypt every integer in $\mathbb{F}_\nu$ by $m \mapsto g_1^m h_1^r$, for some random $r \in \mathbb{F}_{q_1}$.

In our algorithm we assume that P_1 and P_2 generate such a cipher and keep the private key secret from T. In addition, they create another homomorphic encryption function $\mathcal{F}$ over $\mathbb{F}_2$, such that $\mathcal{F}(x_1 \bigoplus x_2) = \mathcal{F}(x_1) \cdot \mathcal{F}(x_2)$.

3.2 Oblivious Transfer

A 1-out-of-2 Oblivious Transfer protocol (OT) involves two players. P_1 has two values v_0, v_1, while P_2 has a selection bit i. At the end of the protocol P_2 gets v_i but learns nothing about v_{1-i}, while P_1 learns nothing. There are many implementations of that functionality. We present here one such implementations in Protocol 1.

Protocol 1 begins with P_2 generating a key pair (k^{pub}, k^{pri}) in some public-key cipher and another public key, $k^{\perp}$. As we assume that the players are semi-honest, P_2 does not cheat and does not generate the private key of $k^{\perp}$. In addition, $k^{\perp}$ should be indistinguishable from k^{pub} in order to prevent P_1 from identifying the index of the correct public key. A known cryptosystem that has this property is El-Gamal, because the public keys in that cipher can be any member in some cyclic group. Next, P_2 sends the two public keys to P_1. P_1 proceeds to encrypt v_0 and v_1 with those public keys and sends the two encryptions to P_2. Finally, P_2 is able to receive the value v_i by decrypting the relevant encrypted value that it gets. P_2 is unable to receive the value v_{1-i}, because it does not have the private key corresponding to $k^{\perp}$.

The protocol entails two public-key encryptions for P_1 followed by one decryption for P_2. It has two communication rounds and $2k + 2c$ bits transfer, where k and c are the number of bits needed to represent encryption keys and ciphertexts in the chosen cipher.

Protocol 1 1-out-of-2 Oblivious Transfer

Input: P_1 holds two values: v_0, v_1; P_2 has one bit $i \in \{0, 1\}$.
Output: P_2 receives v_i without learning v_{1-i}; P_1 learns nothing.

1: P_2 generates a key pair (k^{pub}, k^{pri}) in some public key cipher, and another public key $k^{\perp}$ that P_2 does not have its corresponding private key.
2: P_2 sends (k_0^{pub}, k_1^{pub}) to P_1, where $k_i^{pub} \leftarrow k^{pub}$ and $k_{1-i}^{pub} \leftarrow k^{\perp}$.
3: P_1 computes $c_j = \mathcal{E}_{k_j^{pub}}(v_j)$, $j = 0, 1$, and sends c_0, c_1 to P_2.
4: P_2 decrypts $v_i = \mathcal{E}^{-1}_{k^{pri}}(c_i)$.

3.3 Yao's Millionaires' problem

Yao's millionaires' problem involves two parties P_1 and P_2. P_1 holds a private integer a while P_2 holds a private integer b. They wish to determine whether $a \leq b$ without disclosing anything more than the final result to each of the participants. Yao proposed a solution to this problem in [44], but herein we describe the more efficient algorithm of [32]. It relies on reduction from the latter problem to the set intersection problem as we proceed to explain.

Let $s = s_n s_{n-1} \dots s_1 \in \{0,1\}^n$ be a binary string of length n. We denote the 0-encoding of s as the following set S_s^0 of binary strings,

$$S_s^0 = \{s_n s_{n-1} \dots s_{i+1} 1 |\ s_i = 0; 1 \leq i \leq n\}\,.$$

The 1-encoding of s is defined similarly as follows:

$$S_s^1 = \{s_n s_{n-1} \dots s_i |\ s_i = 1; 1 \leq i \leq n\}\,.$$

It easy to see that each of the sets S_s^1, S_s^0 have at most n elements.

Theorem 3.1 states that $a > b$ iff the sets S_a^1 and S_b^0 have a common element (where here a and b are the the binary encodings of the corresponding integers). In order to get some intuition for this claim, we consider the following example. Let $a = 7 = 111_2$ and $b = 6 = 110_2$. Both integers are of length $n = 3$ bits. We have $S_a^1 = \{1, 11, 111\}$ and $S_b^0 = \{111\}$. Since $S_a^1 \cap S_b^0 \neq \emptyset$ we can conclude that $a > b$ as excepted. In the opposite case, where $a = 6 = 110_2$ and $b = 7 = 111_2$, we have $S_a^1 = \{1, 11\}$ and $S_b^0 = \emptyset$. Since $S_a^1 \cap S_b^0 = \emptyset$, the inequality $a > b$ does not hold (or, equivalently, $a \leq b$).

Theorem 3.1. *Let $a = a_n a_{n-1} \dots a_1 \in \{0,1\}^n, b = b_n b_{n-1} \dots b_1 \in \{0,1\}^n$. Then $a > b$ if and only if the sets S_a^1, S_b^0 have a common element.*

Proof. If $a > b$, there exists $1 \leq i \leq n$ such that $a_i = 1$ and $b_i = 0$, while $a_j = b_j$ for all $i < j \leq n$. Thus, $a_n a_{n-1} \dots a_i \in S_a^1$ while $b_n b_{n-1} \dots b_{i+1} 1 \in S_b^0$. Hence, both S_a^1 and S_b^0 include the common element $a_n a_{n-1} \dots a_i = b_n b_{n-1} \dots b_{i+1} 1$.

Assume next that S_a^1 and S_b^0 have a common element, $t = t_n t_{n-1} \dots t_i \in S_a^1 \cap S_b^0$, for some $i \in [n]$ and $t_i = 1$. Since $t \in S_a^1$, then $a_n a_{n-1} \dots a_i = t_n t_{n-1} \dots t_i$; at the same time, since $t \in S_b^0$, then $b_n b_{n-1} \dots b_{i+1} 1 = b_n b_{n-1} \dots b_{i+1} \overline{b_i} = t_n t_{n-1} \dots t_i$. Therefore, $a > b$. □

It remains to develop a protocol that relies on Theorem 3.1, by implementing private set intersection efficiently. To that end we rely on the fact that one should only compare strings of the same length in S_a^1 and S_b^0. This restriction of the number of needed comparisons reduces the overall number of comparisons from $O(n^2)$ to $O(n)$. Protocol 2 implements the private set intersection.

In Step 1, P_1 generates an additively homomorphic encryption function $\mathcal{F}$, such as Paillier (see Section 3.1)[2]. In Step 2, P_1 sends to P_2 a $2 \times n$ matrix $M[i,j]$, $i \in \{0,1\}$, $j \in [n]$, that encodes its input $a = a_n a_{n-1} \dots a_1$ in the following manner: $M[a_j, j] = \mathcal{F}(0)$, while $M[\overline{a_j}, j] = \mathcal{F}(r_j)$ where r_j is a random element from $\mathbb{Z}_\nu$, the plaintext domain of the chosen encryption function $\mathcal{F}$.

Assume now that P_2 wants to compare one of its own 0-encoding strings $t = t_n t_{n-1} \dots t_i \in S_b^0$, $i \in [n]$, with the corresponding string of the same length in S_a^1. To that end, P_2 computes in Step 4 the value

$$c_t = \Pi_{j=i}^n M[t_j, j]\,.$$

[2]It is possible also to implement the protocol with a multiplicatively homomorphic encryption function with some minor changes; see [32].

If $t \in S_a^1$ then, with certainty, $c_t = \mathcal{F}(0)$, owing to $\mathcal{F}$'s homomorphism and the definition of the matrix M. If, on the other hand, $t \notin S_a^1$, then c_t would equal the encryption of a random value in $\mathbb{Z}_\nu$ so that, with overwhelming probability, $c_t \neq \mathcal{F}(0)$.

Let us denote $y_t := \mathcal{F}^{-1}(c_t)$. The only information that matters for us is whether $y_t = 0$ (a case that indicates non-empty intersection) or $y_t \neq 0$. The exact value of y_t, in case it is not zero, is irrelevant and in fact can leak to P_1 some sensitive information on P_2's input b. Hence, to eliminate such information, P_2 updates the value of c_t in Step 5 to $c_t \leftarrow c_t^{r_t}$ where $r_t \in \mathbb{Z}_\nu^*$. By the homomorphism of $\mathcal{F}$, we get as a result that $c_t = \mathcal{F}(z_t)$ where $z_t = y_t r_t$. Clearly, the selection of r_t from $\mathbb{Z}_\nu^*$ (namely, r_t is not a multiple of either p or q, where $\nu = pq$) ensures that $z_t \neq 0$ iff $y_t \neq 0$. Moreover, z_t does not surrender any information on y_t, beyond the equality or inequality of y_t to zero, since any given $z_t \neq 0$ could have originated from any $y_t \neq 0$ by a suitable selection of r_t.

It remains now to check whether $z_t = 0$ for any $t \in S_b^0$ (in which case the intersection is non-empty) or not. To hide the size of S_b^0 (as it equals the number of zero bits in b), P_2 generates in Step 7 $\ell := n - |S_b^0|$ random elements z_j, $j \in [\ell]$. It then sends to P_1 a random permutation of the n elements $\{c_t : t \in S_b^0\} \cup \{z_j : j \in [\ell]\}$ (Step 8). Finally (Step 9), P_1 decrypts the received n values. It decides that the intersection is non-empty, with high probability, iff one of the n decrypted values is zero. In that case it outputs that $a > b$. Otherwise, $a \leq b$.

Protocol 2 A protocol for solving Yao's millionaires' problem

Input: P_1 holds an integer $a = a_n a_{n-1} \ldots a_1 \in \{0,1\}^n$; P_2 holds an integer $b = b_n b_{n-1} \ldots b_1 \in \{0,1\}^n$.
Output: Whether $a > b$.

1: P_1 generates a probabilistic public key encryption function $\mathcal{F}$ which is additively homomorphic and keeps to itself the private key.
2: P_1 initializes a matrix M of size $2 \times n$ where $M[a_j, j] = \mathcal{F}(0)$ and $M[\overline{a_j}, j] = \mathcal{F}(r_j)$ for $j \in [n]$ and $r_j \in_R \mathbb{Z}_\nu$. Then it sends the matrix M to P_2.
3: **for** each $t = t_n t_{n-1} \ldots t_i \in S_b^0$ **do**
4: P_2 assigns $c_t \leftarrow \Pi_{j=i}^{j=n} M[t_j, j]$.
5: P_2 computes $c_t \leftarrow c_t^{r_t}$ for $r_t \in \mathbb{Z}_\nu^*$.
6: **end for**
7: P_2 generates $\ell \leftarrow n - |S_b^0|$ encrypted random values $z_j = \mathcal{F}(r_j)$, $j \in [\ell]$, where $r_j \in_R \mathbb{Z}_\nu$.
8: P_2 sends to P_1 a random permutation of the n elements $\{c_t : t \in S_b^0\} \cup \{z_j : j \in [\ell]\}$. Let $\{w_1, \ldots, w_n\}$ denote the sequence that P_2 sends to P_1.
9: P_1 decrypts $m_i = \mathcal{F}^{-1}(w_i)$ for $i \in [n]$ and outputs $a > b$ if $\exists m_i$ where $m_i = 0$; otherwise, it outputs that $a \leq b$.

Complexity. Step 2 entails $2n$ encryptions for P_1. Then, in Steps 4-5, since the $n - i - 1$ prefix bits between t_i and t_{i-1} are equal, P_2 performs at most $2n - 3$ multiplications and $O(n)$ modular exponentiations. Finally, in Step 9 P_1 performs at most n decryptions. The protocol entails two communication rounds in which $3n$ encrypted values are transferred (Steps 2 and 8).

Privacy. Privacy can be lost whenever the players exchange messages. In Step 2, P_2 gets the encrypted matrix M. Since all of M entries are encrypted, and owing to the fact that the encryption function is probabilistic, P_2 cannot distinguish between zero encryptions and random encryptions. Next, in Step 8, P_1 receives from P_2 n encrypted values $w_i, i \in [n]$. P_1 cannot reveal the number of elements in S_b^0 due to the bogus $n - |S_b^0|$ encrypted elements that P_2 added. When P_1 decrypts those values and reveals a value m_i that equals to 0, it cannot learn for which bit that value corresponds, due to the random permutation that P_2 applied in Step 8. Furthermore, to prevent from P_1 to infer information from the decrypted random values, P_2 performed the scaling in Step 5, as explained

earlier. Thus, the protocol does not reveal anything more than the desired output of whether $a > b$ or not.

3.4 Yao's garbled circuit Protocol

Yao's garbled circuit protocol [44] is a cryptographic protocol that enables two parties, P_1 and P_2, to evaluate a function over their inputs without a trusted third party and without revealing to each other their private inputs. Assume that P_i has an input a_i which we think of as a binary string, $i = 1, 2$. The two parties wish to compute $f(a_1, a_2)$, for some function f known to both of them.

Let C be a boolean circuit that realizes the function f. It is sufficient to note that there exists a mapping from any polynomial time function with fixed sized inputs to a boolean circuit that calculates the same output [19]. The two parties can generate C together.

Let g be a gate that receives two input wires x and y (both are single bits) and outputs a single bit-wire z. The gate can be implementing any boolean functionality (say `OR`, `AND`, their negations etc.). Such a gate can be represented by a truth table; Table 1 shows an example when the gate is an `AND` gate.

x	y	$z = \texttt{AND}(x, y)$
0	0	0
0	1	0
1	0	0
1	1	1

Table 1: Truth table of the `AND` gate

P_1 garbles, independently, each of the circuit's gates. Letting g be one of those gates, P_1 takes its truth table and generates a random and independent encryption key, in some symmetric cipher, for each possible value of each wire. Namely, it generates:

- two keys k_1^0 and k_1^1 for each of the two possible values of the first input x;
- two keys k_2^0 and k_2^1 for each of the two possible values of the second input y;
- two keys k_3^0 and k_3^1 for each of the two possible values of the output $z = g(x, y)$.

Furthermore, P_1 computes, for each of the four possible values of (x, y), the value $E_{k_1^x}\left(E_{k_2^y}(k_3^{z=g(x,y)}||0^n)\right)$, where n is some security parameter; see Table 2.

x	y	$z = \texttt{AND}(x, y)$	garbled value
k_1^0	k_2^0	k_3^0	$E_{k_1^0}\left(E_{k_2^0}(k_3^0\|\|0^n)\right)$
k_1^0	k_2^1	k_3^0	$E_{k_1^0}\left(E_{k_2^1}(k_3^0\|\|0^n)\right)$
k_1^1	k_2^0	k_3^0	$E_{k_1^1}\left(E_{k_2^0}(k_3^0\|\|0^n)\right)$
k_1^1	k_2^1	k_3^1	$E_{k_1^1}\left(E_{k_2^1}(k_3^1\|\|0^n)\right)$

Table 2: Garbled values for an `AND` gate

Next, P_1 sends to P_2 the encrypted garbled values for each of the gates in the circuit. Specifically, if g is such a gate, P_1 sends to P_2 the four values

$$E_{k_1^x}\left(E_{k_2^y}(k_3^{z=g(x,y)}||0^n)\right), \quad x = 0, 1,\ y = 0, 1 \tag{4}$$

in a random order. In addition, P_1 sends to P_2 the garbled values corresponding to each of its own input wires. For example, if g is some gate in the circuit that is fed by an input bit x which P_1 owns, then P_1 will send to P_2 the garbled value k_1^x corresponding to that gate — k_1^0 if $x = 0$ and k_1^1 if $x = 1$.

Now, P_2 starts computing the garbled circuit, gate by gate, as we proceed to describe. Let g be a gate of which the two input wires are inputs from P_1 and P_2; the first wire x corresponds to the input from P_1 and the second wire y corresponds to the input from P_2. Assume that $x = a$ and $y = b$. Then P_2 has at this stage the garbled value k_1^a, which P_1 has sent to it, but it does not have k_2^b. So P_2 engages in a 1-out-of-2 OT protocol (see Section 3.2) vis-a-vis P_1 to receive its own garbled value k_2^b. Namely, while P_1 holds the two values (k_2^0, k_2^1), P_2 receives from it the relevant garbled value, which is k_2^b, and only that value. Next, P_2 decrypts the four values associated with the gate g, as given in Eq. (4), by applying on them the decryption $E^{-1}_{k_1^a}$ followed by the decryption $E^{-1}_{k_2^b}$. With probability $1 - O(2^{-n})$, only one of the decrypted values will end with 0^n. That value equals $k_3^{c=g(a,b)}||0^n$; after stripping the suffix 0^n, P_2 gets the garbled value corresponding to the correct value of g's output wire.

P_2 can now proceed to traverse the entire circuit in a similar manner, until it gets the garbled values on all output wires. Those values can be converted into their bit values by having P_1 publish upfront conversion tables for those output wires.

We provide a summary of the protocol in Protocol 3.

Protocol 3 Yao's Garbled Circuit Protocol – Summary

Input: P_1 and P_2 hold integer inputs x_1, x_2 respectively, and an integer function $f(\cdot, \cdot)$.
Output: P_2 gets $f(x_1, x_2)$.

1: P_1 generates a boolean circuit C that receives as inputs x_1 and x_2 and outputs $f(x_1, x_2)$.
2: P_1 generates two random and independent encryption keys for each input wire.
3: P_1 computes for each of C's gates the encrypted garbled values for each of the four possible values of the gate's inputs (Eq. (4)).
4: P_1 sends those encrypted garbled values, in a random order, to P_2 (for each gate); in addition, P_1 sends to P_2 its own garbled input values corresponding to x_1.
5: For each of the input gates, P_2 performs a 1-out-of-2-OT protocol to receive from P_1 its garbled values for that gate.
6: P_2 uses the garbled values and calculates each gate output by decrypting the four possible values of the garbled circuit, as received in Step 4, and it identifies the correct output as the one that ending with 0^n.
7: P_2 uses the correct garbled value for the gate's output wire as the garbled input to the next gates.
8: When P_2 gets the garbled values of all output gates, it translates those values to the correct bit values according to the conversion tables.

Complexity. Let $|G|$ denote the number of gates in the circuit C. The protocol entails only 3 communication rounds (one for Step 4 and two for implementing the OT in Step 5, see Section 3.2). Messages are sent in Steps 4 and 5; it is easy to see that the overall communication complexity is $\mathcal{O}(|G|)$. The overall computational cost is bounded by $\mathcal{O}(|G| \cdot [E] + |G| \cdot [OT])$ where $[E]$ denotes the cost of a single encryption or decryption and $[OT]$ denotes the cost of an OT protocol.

Privacy. A full proof of the protocol's privacy is given in [33].

3.5 The BGW Protocol

In this section we describe the Ben-Or Goldwasser Wigderson (BGW) protocol [7], with the efficiency improvement of [18]. Consider n parties, P_i, $1 \leq i \leq n$, each holding a private integer x_i in some finite field $\mathbb{F}$. They wish to jointly compute a function over those inputs, $f(x_1, \ldots, x_n) = (y_1, \ldots, y_n)$, without disclosing to each other their private input. To that end, the parties $P_1, \ldots, P_n$ agree on some arithmetic circuit C that computes f over the finite field $\mathbb{F}$; the field's size must be greater than the number of participating parties as well as greater than the apriori bound on the input and output values. The circuit consists of two different types of gates: addition gates, and multiplication gates. Let $\alpha_1, \ldots, \alpha_n$ be distinct non-zero elements in $\mathbb{F}$; then α_i will be used as a public identifier of P_i. 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-n secret sharing scheme (see [41]), with $t < n/2$. 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_i shares its input x_i with all parties; i.e., it chooses a random polynomial g_i of degree t such that $g_i(0) = x_i$, and it then sends to each party P_j the value $g_i(\alpha_j)$.

The circuit emulation phase. In this phase, the parties emulate the computation of $C(x_1, \ldots, x_n)$, where in each gate, the parties compute shares of the value of the output wire using their shares of the input wire 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 $f_1(\alpha_i)$ and $f_2(\alpha_i)$ are the shares that P_i holds for the two input wires to an addition gate, then $f(\alpha_i) = f_1(\alpha_i) + f_2(\alpha_i)$ is a valid sharing of the output wire. Indeed, the polynomial $f(x) := f_1(x) + f_2(x)$ has the same degree as $f_1(x)$ and $f_2(x)$, and its constant term satisfies $f(0) = f_1(0) + f_2(0)$.

Multiplication gates. The case of multiplication gates is more involved as it requires interaction among the parties. In particular, given shares $f_1(\alpha_i)$ and $f_2(\alpha_i)$ for the two input wires of a multiplication gate, then $f(\alpha_i) := f_1(\alpha) \cdot f_2(\alpha_i)$ 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 is $2t$ and not 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 [18], which is based on the fact that if f is a polynomial of degree at most $n-1$ and $\alpha_1, \ldots, \alpha_n$ are n 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, $f_1(0) \cdot f_2(0) = f(0) = \sum_{i=1}^{n} \lambda_i \cdot f(\alpha_i)$, where $\lambda_i := \prod_{j \neq i} \alpha_j / (\alpha_i - \alpha_j)$ are the Lagrange coefficients.

The multiplication sub-protocol proceeds as follows. Given the shares $f_1(\alpha_i), f_2(\alpha_i)$ of the party P_i, the party P_i locally multiplies these two shares and gets the point $f(\alpha_i)$. Then, it chooses a polynomial $g_i(x)$ of degree t such that $g_i(0) = f(\alpha_i) = f_1(\alpha_i) \cdot f_2(\alpha_i)$. It then shares the polynomial g_i with all parties, so that each party P_j receives the share $g_i(\alpha_j)$. At the end of this stage, each party P_j holds the shares $g_1(\alpha_j), \ldots, g_n(\alpha_j)$. Next, let us define the polynomial $h(x) := \sum_{i=1}^{n} \lambda_i \cdot g_i(x)$, which has a degree at most t. Each party P_j locally computes the linear combination $\sum_{i=1}^{n} \lambda_i \cdot g_i(\alpha_j) = h(\alpha_j)$, which is its share in the implicitly defined polynomial $h(x)$. Note that $h(x)$ is a polynomial of degree-t, and $h(0) = \sum_{i=1}^{n} \lambda_i \cdot g_i(0) = \sum_{i=1}^{n} \lambda_i \cdot f(\alpha_i) = f_1(0) \cdot f_2(0)$.

Output reconstruction phase. In this phase, each party P_i receives all the shares of the output wire that hides its respective output y_i, reconstruct y_i and outputs it.

The security of the BGW protocol was proven in [4].

3.6 Oblivious testing of the solvability of an encrypted linear system of equations

In this section we describe a method for an oblivious testing of the solvability of a system of linear equations. That method is based on a protocol that performs oblivious Gaussian elimination for the purpose of testing whether a given matrix has a full rank or not. (All protocols that are presented here are based on [35].)

Throughout this section we assume a setting that involves two players – T and P. P holds the key pair in a homomorphic encryption function $\mathcal{F}$ over $\mathbb{F}_2$. T holds an encryption $\mathcal{F}(M)$ of some matrix over $\mathbb{F}_2$. The goal is for T to perform some computations on M, without learning anything about M.

The first computation will test whether a given square matrix has a full rank or not. The main ingredient in that computation will be Gaussian elimination. The input that T holds at the beginning of that computation is $\mathcal{F}(M)$, where M is a square matrix of dimensions $k \times k$. At the end of the computation T will have an encryption $\mathcal{F}(M')$ of another $k \times k$ matrix M' such that M' is upper triangular and M' has full rank if and only if M has a full rank. The protocol presented in Section 3.6.1 achieves that goal with high probability.

Next, we present in Section 3.6.2 a related protocol for a non-square matrix M. Assume that T holds $\mathcal{F}(M)$ where M is of dimensions $k_a \times k_b$, where $k_a \leq k_b$, and $\text{rank}(M) = r$. Then, with high probability, the protocol ends with T getting $\mathcal{F}(M')$ where M' is a $k_a \times k_b$ upper triangular matrix with $\text{rank}(M') \leq r$. In addition, with constant probability it holds that $\text{rank}(M') = r$.

Finally, we present in Section 3.6.3 a protocol for the oblivious testing of the solvability of a system of linear equations. Assuming that T holds, in addition to $\mathcal{F}(M)$, also $\mathcal{F}(\mathbf{b})$ where $\mathbf{b}$ is a row vector of dimension k_b, the protocol ends with T having $\mathcal{F}(flag)$ where $flag$ is a binary flag indicating whether the system $\mathbf{x}M = \mathbf{b}$ has a solution $\mathbf{x} \in \mathbb{F}_2^{k_a}$.

3.6.1 Oblivious Gaussian elimination of a square matrix

In preparation to describing the main protocol, we begin by describing basic computations that T can perform over values that are encrypted by the homomorphic cipher $\mathcal{F}$, even though T cannot decrypt $\mathcal{F}$ (a cipher whose decryption key is known only to the other player P).

Routine 1: Linear combinations. Assume that T holds the encryption of two k-dimensional vectors, $\mathcal{F}(\mathbf{v}_1)$, $\mathcal{F}(\mathbf{v}_2)$. Let $a_1, a_2 \in \mathbb{F}_2$ be two scalars. Then $\mathcal{F}(a_1\mathbf{v}_1 + a_2\mathbf{v}_2) = \mathcal{F}(\mathbf{v}_1)^{a_1} \cdot \mathcal{F}(\mathbf{v}_2)^{a_2}$.

Routine 2: Multiplication by a random matrix. Assume that T has a random matrix R of dimensions $\ell \times k$ and it wishes to compute $\mathcal{F}(RM)$. Since each row in RM is a linear combination of M's rows, such a computation can be carried out along the lines described above.

Routine 3: Multiplying scalars. Assume that T has $\mathcal{F}(a)$ and $\mathcal{F}(b)$ and it wishes to get $\mathcal{F}(ab)$. Towards that end, it generates two randoms r_a and r_b, computes $\mathcal{F}(a + r_a) = \mathcal{F}(a) \cdot \mathcal{F}(r_a)$ and $\mathcal{F}(b + r_b)$, and sends those two encrypted values to P. P decrypts them and gets $a + r_a$ and $b + r_b$. Owing to the randomness of r_a and r_b, P learns no information from the two decrypted values. P then sends to T the encrypted value $\mathcal{F}((a + r_a) \cdot (b + r_b))$. Finally, T computes

$$\mathcal{F}(ab) = \mathcal{F}((a + r_a) \cdot (b + r_b)) \cdot [\mathcal{F}(a)^{r_b} \cdot \mathcal{F}(b)^{r_a} \cdot \mathcal{F}(r_a r_b)]^{-1} \ .$$

Routine 4: Computing `OR` operation. Assume that T has $\mathcal{F}(a)$ and $\mathcal{F}(b)$ and it wishes to get $\mathcal{F}(a \vee b)$. Towards that end, T and P applying Routine 3 to get $\mathcal{F}(ab)$ in the hands of T. Finally, T

uses Routine 1 to compute

$$\mathcal{F}(a \vee b) = \mathcal{F}(a) \cdot \mathcal{F}(b) \cdot \mathcal{F}(ab)\,.$$

Routine 5: Basic column elimination. T has $\mathcal{F}(M)$ and a row index j, $1 \leq j \leq k$. The goal is to let T have $\mathcal{F}(M')$ where M' is the matrix whose rows are given by[3]

$$M'_i = M_i \;\; \forall i \in [1, j] \quad \text{and} \quad M'_i = M_i + (M[i,1]M[j,1]) \cdot M_j \;\; \forall i \in [j+1, k]\,. \tag{5}$$

In case where the leading term in M_j is 1, then such a procedure results in a matrix M' where all leading terms in the first column below the jth row are zeroed. If, on the other hand, the leading in M_j is 0, then $M' = M$. Protocol 4 performs that computation.

Protocol 4 Basic Column Elimination

Input: T has $\mathcal{F}(M)$ and $j \in [1, k]$.
Output: T gets $\mathcal{F}(M')$ where M' is defined in Eq. (5).

1: **for** $j < i \leq k$ **do**
2: T computes with P the value $\mathcal{F}(M[i,1] \cdot M[j,1])$ (by Routine 3 - Multiplying scalars).
3: T computes with P the vector: $\mathcal{F}((M[i,1]M[j,1]) \cdot M_j)$ (by applying Routine 3 on each entry).
4: T computes $\mathcal{F}(M'_i) \leftarrow \mathcal{F}(M_i + (M[i,1]M[j,1]) \cdot M_j)$ (by Routine 1 - Linear combinations).
5: **end for**

Routine 6: Oblivious column elimination. Protocol 4 is effective only if T selects a row M_j that has a leading term that equals 1. But as T does not have the clear matrix M, Protocol 4 cannot be applied as is. Towards resolving this obstacle, we make the following observation.

Claim 3.2. *Assume that T knows that at least one of the first m rows of M, for some $m < k$, has a leading term that equals 1. Then by applying Protocol 4 m times, each time with a different index $j = 1, \ldots, m$, T will end up with $\mathcal{F}(M')$ where the left column of M' includes exactly one entry that equals 1, among the first m rows, while all the other entries equal 0.*

Proof. Let j_0 be the first index such that $M[j_0, 1] = 1$. Hence, applying Protocol 4 for each $j = 1, \ldots, j_0 - 1$ will leave the matrix unchanged (since the leading term in those rows is zero). Then, applying Protocol 4 with $j = j_0$ will zero all the leading terms in all rows. Finally, applying Protocol 4 with $j = j_0 + 1, \ldots, m$ will also leave the matrix unchanged since, again, those rows have a leading term that equals zero. □

Hence, Claim 3.2 enables us to perform Gaussian elimination on the first column, but it relies on the assumption that the matrix M has a nonzero leading entry in at least one row among the first m rows, for some m. To enable such an assumption, we state the following theorem.

Theorem 3.3. *Let M be a $k \times k$ matrix over $\mathbb{F}_2$ and R be a random matrix of same dimensions over $\mathbb{F}_2$. Let $m = \omega(\log(k))$ such that $m < k/2$. Then:*

(a) With probability $1 - neg(k)$, R is of full rank.

(b) If R is of full rank and the leftmost column of M is non-zero, then with probability $1 - neg(k)$, at least one of the top m rows in RM will have a non-zero leading entry.

[3] Hereinafter, if M is a matrix then M_i is its ith row and $M[i,j]$ is the jth entry in that row.

Protocol 5 Oblivious Column Elimination

Input: T has $\mathcal{F}(M)$.

Output: (WHP) T gets $\mathcal{F}(M')$ where M' is also a square $k \times k$ matrix, has the same rank as M, and its left most column is $\mathbf{e}_1^t$[4].

1: T and P pick a random non-singular matrix $R \in (\mathbb{F}_2)^{k \times k}$.
2: T computes $\mathcal{F}(M')$ where $M' = RM$ (Routine 2).
3: **for** every $i \in [m(k)]$ **do**
4: T and P compute $\mathcal{F}(M') \leftarrow BasicColumnElimination(\mathcal{F}(M'), i)$.
5: **end for**
6: T updates the first row in M' by $M'_1 \leftarrow \sum_{i=1}^{m(k)} M'_i$ (Routine 1).
7: T and P compute $\mathcal{F}(M') \leftarrow BasicColumnElimination(\mathcal{F}(M'), 1)$.

Proof. (a) The process of selecting R consists of k consecutive selections of row vectors in $\mathbb{F}_2^k$. It is easy to see that if the first ℓ row vectors are independent, then the probability to select the next row vector which would be independent of its predecessors is $1 - 2^{\ell-k}$, $\ell = 0, \ldots, k-1$. Hence, the probability of selecting k independent vectors is $1 - O(2^{m-k})$. Since $m < k/2$, that probability is $1 - \text{neg}(k)$.

(b) Denote the leftmost column in M by $\mathbf{c}$, and the top m rows in R by $\mathbf{r}_1, \ldots, \mathbf{r}_m$. Then $(RM)[i,1] = \mathbf{r}_i\mathbf{c}$, for $1 \leq i \leq m$. Since $\mathbf{r}_1, \ldots, \mathbf{r}_m$ are random and independent vectors in $\mathbb{F}_2^k$, then $(RM)[i,1] = 0$ with probability $1/2$. Hence, the probability that at least one of those values is non-zero is at least $1 - 2^{-m} = 1 - \text{neg}(k)$. □

Therefore, if T selects a random square matrix R and computes $\mathcal{F}(RM)$ (using Routine 2), then with high probability it can proceed to apply the procedure described in Claim 3.2 towards getting $\mathcal{F}(M')$ where M' has a left most column with all leading terms 0, except one that equals 1. Note that due to the multiplication by R, M' is not row equivalent to M (namely, it is not the result of applying only Gaussian elimination on M), but it has the same rank as M.

We are now ready to present Protocol 5 for oblivious column elimination.

The protocol begins by a selection of a random non-singular matrix R (Step 1). Then, T computes $\mathcal{F}(M')$ where $M' = RM$, using the previously described Routine 2. Afterwards, T and P perform the basic column elimination procedure with each of the first $m(k)$ rows. (We note that $m(k)$ can be set to any value which makes the probability of failure $2^{-m(k)}$ sufficiently small.) At this stage, the matrix M' that T holds encrypted, has a first column that equals $\mathbf{e}_{j_0}^t$ for some $j_0 \in [1, m(k)]$. In Steps 6-7, we "move" the 1 entry in the first column to be in the first row. To do that, T replaces the first row with the sum of the first $m(k)$ rows by invoking Routine 1. At this point, the first column has a 1 entry both in the first and j_0th rows. In order to zero the 1 entry in the j_0th row, we invoke the basic column elimination procedure with the first row. Hence, we end up with T holding $\mathcal{F}(M')$ where M' is as required. The rank of M' equals the rank of RM since it is obtained from RM only by means of elementary row operations, and the rank of RM equals the rank of M since R is of full rank.

Routine 7: Oblivious Gaussian elimination of a square matrix. Protocol 6 implements Protocol 5 recursively on the input matrix M. It starts by applying Protocol 5 on the first column of the matrix and then it continues by recursion on the minor $M'_{-1,-1}$ — the $(k-1) \times (k-1)$ matrix that is obtained from M' (the matrix that T hold encrypted after the first step of applying Protocol 5 on

[4] $\mathbf{e}_j$ denotes the vector in which the jth entry is 1 and all other entries are 0.

Protocol 6 Oblivious Gaussian Elimination

Input: T has $\mathcal{F}(M)$.

Output: (WHP) T gets $\mathcal{F}(M')$ where M' is an upper triangular $k \times k$ matrix which has full rank iff M has a full rank.

1: T and P compute $\mathcal{F}(M') \leftarrow ObliviousColumnElimination(\mathcal{F}(M))$.

2: T and P compute $\mathcal{F}(M'_{-1,-1}) \leftarrow ObliviousGaussianElimination(\mathcal{F}(M'_{-1,-1}))$.

the first column) by removing the first row and first column. At the end of this recursive process, T holds an encryption of an upper triangular matrix M' that has a full rank iff the original matrix M has a full rank.

	Computational	**# rounds**	**# messages**	**# bits**
Routine 1	$k[Mul]$	0	0	0
Routine 2	$lk(k-1)[Mul]$	0	0	0
Routine 3	$T: 3[\mathcal{F}] + 5[Mul] + [Inv]$; $P: 2[\mathcal{F}^{-1}] + [\mathcal{F}]$	2	2	3λ
Routine 4	$T: 2[Mul]$; $[Routine\ 3]$	2	2	3λ
Routine 5	$(k-j)\cdot((k+1)\cdot[Routine\ 3]+[Routine\ 1])$	4	4	$O(\lambda \cdot k)^{\dagger}$
Routine 6	$k^2(k-1)[Mul] + m \cdot ([Routine\ 5] + [Routine\ 1])$	m	m	$\tilde{O}(\lambda \cdot k)^{\dagger}$
Routine 7	$\sum_{i=1}^{n}[Routine\ 6(M_{i\times i})] = \tilde{O}(k^4 \cdot ([\mathcal{F}]+[Mul]+[Inv]))$	$\tilde{O}(k^{0.275})^{\dagger}$	$\tilde{O}(k^{0.275})^{\dagger}$	$\tilde{O}(\lambda \cdot k^2)^{\dagger}$

Table 3: Computational and communication costs of Routines 1-7. The computational column refers only to T computations, except for routines that depend on Routine 3. Herein, if f is any operation or Routine, we let $[f]$ denote its cost. Assume the security parameter of $\mathcal{F}$ is λ and denote encryption, decryption, multiplication, inverse in the cipher domain as: $[\mathcal{F}]$, $[\mathcal{F}^{-1}]$, $[Mul]$, $[Inv]$ respectively. The lower bound of m is $\omega(\log(k))$ and the upper bound is $k/2$. Cells that are denoted by †, are updated according to the improvement technique that is desribed in [35].

3.6.2 Oblivious Gaussian elimination of a non-square matrix

It is possible to handle non-square matrix M, of dimensions $k_a \times k_b$, where $k_a \leq k_b$, by multiplying the matrix from left and right by square full-rank random matrices, R_a and R_b, of dimensions $k_a \times k_a$ and $k_b \times k_b$ respectively: $M^* = R_a M R_b$. The two parties perform the Oblivious Gaussian elimination protocol on M^* towards triangulating the left $k_a \times k_a$ block of the matrix (the right $k_b - k_a$ columns are updated but are not eliminated). We now state a simple claim from [10].

Claim 3.4. *Under the above assumptions, if rank$(M) \geq r$ then with constant probability rank$(M^*) = r$.*

Claim 3.4 (see [10, Theorem 3] for a proof) implies that after applying the Oblivious Gaussian elimination protocol on M^*, T will get $\mathcal{F}(N^*)$ where N^* is upper triangular; moreover, with constant probability, N^* has exactly r nonzero terms on its diagonal.

Protocol 7 Solvability of a Linear System

Input: T has $\mathcal{F}(M)$ and $\mathcal{F}(\mathbf{b})$ where $M \in \mathbb{F}_2^{k_a \times k_b}$ and $\mathbf{b} \in \mathbb{F}_2^{1 \times k_b}$.

Output: (WHP) T gets $\mathcal{F}(flag)$ where $flag$ is a binary flag indicating whether the system $\mathbf{x}M = \mathbf{b}$ has a solution $\mathbf{x} \in \mathbb{F}_2^{k_a}$.

1: T picks randomly two non-singular matrices R_a and R_b of dimensions $k_a \times k_a$ and $k_b \times k_b$ respectively.

2: T computes $\mathcal{F}(M^*)$ and $\mathcal{F}(\mathbf{b}^*)$ for $M^* = R_a M R_b$ and $\mathbf{b}^* = \mathbf{b}R_a$ (by Routine 1 - Linear combinations).

3: T and P perform Oblivious Gaussian Elimination on the top left $k_a \times k_a$ block of $\begin{pmatrix} M^* \\ \mathbf{b}^* \end{pmatrix}$.

4: Denote by M'' the resulting $(k_a + 1) \times k_b$ matrix for which T holds its $\mathcal{F}$ encryption.

5: T and P compute the product $\prod_{i=1}^{k_b}(1 - M''[k_a + 1, i])$ (by Routine 3 - Multiplying scalars).

3.6.3 Oblivious testing of the solvability of a system of linear equations

Assume that T holds, in addition to $\mathcal{F}(M)$, also $\mathcal{F}(\mathbf{b})$ where $\mathbf{b}$ is a row vector of dimension k_b. We present here a protocol that ends with T having $\mathcal{F}(flag)$ where $flag$ is a binary flag indicating whether the system $\mathbf{x}M = \mathbf{b}$ has a solution $\mathbf{x} \in \mathbb{F}_2^{k_a}$.

Let R_a and R_b be two random matrices of full rank of dimensions $k_a \times k_a$ and $k_b \times k_b$ respectively. Denote $M^* = R_a M R_b$ and $\mathbf{b}^* = \mathbf{b}R_a$.

Claim 3.5. *Under the above conditions,* $\mathbf{x}M = \mathbf{b}$ *is solvable iff* $\mathbf{y}M^* = \mathbf{b}^*$ *is solvable.*

Proof. Assume there exist a vector $\mathbf{c} \in \mathbb{F}_2^{k_a}$ where $\mathbf{c}M = \mathbf{b}$, then the rows of M spans $\mathbf{b}$. Multiplying M by a full-rank matrix R_a does not change the row space of M. Thus, there exist $\mathbf{c}^*$ such that $\mathbf{c}^* R_a M = \mathbf{b}$. Multiplying both sides with matrix R_b results in $\mathbf{c}^* R_a M R_b = \mathbf{b}R_b$ as expected. Similar approach may prove the other direction. □

By Claim 3.4, if $\text{rank}(M) = r$ then with constant probability the rank of the top left $k_a \times k_a$ block of M^* is also r. Hence, with constant probability, the original system $\mathbf{x}M = \mathbf{b}$ is solvable iff the row vector $\mathbf{b}^*$ is spanned by the rows of the matrix M^*. In order to check whether $\mathbf{b}^*$ belongs to the row space of M^*, we apply the Oblivious Gaussian elimination on the top left $k_a \times k_a$ block of $\begin{pmatrix} M^* \\ \mathbf{b}^* \end{pmatrix}$. In doing so, we apply on the last row of the matrix all operations of column elimination. Now, $\mathbf{b}^*$ belongs to the row space of M^* iff at the end of that process, all elements in the last row are zeroed. That condition holds iff $\prod_{i=1}^{k_b}(1 - M''[k_a + 1, i]) = 1$, where M'' is the resulting $(k_a + 1) \times k_b$ matrix for which T holds its $\mathcal{F}$ encryption. By applying Routine 3 for multiplying scalars, T gets $\mathcal{F}(flag)$ where $flag$ is a binary flag indicating the solvability of the original system.

The protocol has a one-sided error. If the original system is not solvable then T will always get at the end $\mathcal{F}(0)$. If, on the other hand, the system is solvable then if in the first step the rank of the top-left $k_a \times k_a$ sub-matrix of M^* is the same as that of M, T will get eventually $\mathcal{F}(1)$, as needed. However, as the latter condition holds only with constant probability, T and P can execute the protocol several times and take the `OR` of the results in order to make the error probability negligible. Assume we denote the error constant as $c < 1$, so the probability that the protocol outputs the correct value becomes $1 - (c^t) = 1 - neg(t)$ after T performs Routine 4 t times over the outputs of Protocol 7.

In [35, Theorem 7], the authors continue to improve the complexity of *Solvability of a Linear System* and get the following results:

Complexity. Consider the notations from Table 3. Then, the computational complexity costs are $O(k_a k_b^2 \cdot [Mul])$ in Step 2, $\tilde{O}(k_a^4 \cdot ([Mul + [\mathcal{F}] + [Inv]))$ in Step 3, and $k_b \cdot ([Routine\ 3] + [Mul])$ in Step 5.

By executing Protocol 7 t times, and performing Routine 5 between the results, we get a computation complexity of $\tilde{O}(t \cdot (k_a k_b^2 \cdot [Mul] + k_a^4 \cdot ([Mul + [\mathcal{F}] + [Inv])))$, communication rounds and messages equals to $\tilde{O}(t \cdot k_a^{0.275})$, and $\tilde{O}(t \cdot \lambda k_a k_b)$ bits transferred.

Privacy. Since T does not hold the private key to the encryption scheme, all the operations that it performs locally via the homomorphic property of the encryption do not reveal any information about the plain values of the matrix. Only Routine 3 involves player P in the computation and it easy to see that it is not leak any information to P and therefore not to T due the random values that masks the values that transmitted. In the end, the only value that get reveal is the output, i.e $\mathcal{F}(flag)$ that indicates (W.H.P) whether the linear system $\mathbf{x}M = \mathbf{b}$ has a solution $\mathbf{x} \in \mathbb{F}_2^{k_a}$, as excepted.

3.7 Computing the rank of an encrypted matrix

In this section we describe a method for obliviously computing the rank of a matrix M that is encrypted under an additively homomorphic encryption. This method is based on [28].

Assume a setting that involves two players – T and P. T holds an encrypted matrix $\mathcal{F}(M)$ where $M \in \mathbb{F}^{k_a \times k_b}$, and $\mathcal{F}$ is an additively-homomorphic encryption over $\mathbb{F} = \mathbb{F}_{2^k}$. The parameter k that defines the size of the field is assumed to be *sufficiently large*. Without loss of generality we assume that $k_a \leq k_b$. The goal of those two players is to compute the rank of the matrix M, so that at the completion of the protocol T will hold $\mathcal{F}(rank(M))$.

The computation is separated into two major stages. In the first stage, the purpose is to transform the encrypted matrix $\mathcal{F}(M)$ into $\mathcal{F}(N)$ where N is a precondition square matrix such that $rank(M) = deg(p_N) - 1$, where p_N denotes the minimal polynomial of the matrix N. The goal of the second stage is to compute the minimal polynomial of the encrypted matrix N by using related linearly recurrent sequences.

This section is organized as follows. In Section 3.7.1 we provide the necessary background regarding linearly recurrent sequences. Next, in Section 3.7.2 we describe how to compute the minimal polynomial of an encrypted matrix using a recurrent sequence. Finally, in Section 3.7.3 we describe the necessary preparations that are needed to compute the matrix's rank via the minimal polynomial computations.

3.7.1 Preliminaries

Let $\mathbb{F}$ be a field and $V \neq \{0\}$ be a vector space over $\mathbb{F}$. Then $V^{\mathbb{N}}$ is the (infinite-dimension) vector space of infinite sequences $(m_i)_{i\in\mathbb{N}}$, where $m_i \in V$.

Definition 3.6. *A sequence $(m_i)_{i\in\mathbb{N}}$ is* **linearly recurrent (over $\mathbb{F}$)** *if there exist $n \in \mathbb{N}$ and $f_0, \cdots, f_n \in \mathbb{F}$ with $f_n \neq 0$ such that*

$$\sum_{0\leq j\leq n} f_j \cdot m_{i+j} = f_0 m_i + f_1 m_{i+1} + \cdots + f_n m_{i+n} = 0 \quad \forall i \in \mathbb{N}\,.$$

The polynomial $f = \sum_{0\leq j\leq n} f_j \cdot x^j \in F[x]$ of degree n is called the generating polynomial for the sequence $\mathbf{m} := (m_i)_{i\in\mathbb{N}}$. (Other common names for f are the characteristic or the annihilating polynomial.) The set of all generating polynomials for $\mathbf{m}$, together with the zero polynomial, forms an ideal in $\mathbb{F}[x]$. It is worth mentioning that $V^{\mathbb{N}}$ is a $\mathbb{F}[x]$-module over $\mathbb{F}$ and also a principal ideal domain. Thus, there exists a unique monic polynomial that generates that ideal. We denote that polynomial by $p_{\mathbf{m}}$ and call it *the minimal polynomial* of the sequence $\mathbf{m} = (m_i)_{i\in\mathbb{N}}$. The degree of $p_{\mathbf{m}}$ is called the *recursion order* of the sequence.

Given a square matrix $M \in \mathbb{F}^{k_b \times k_b}$, we are interested in the following three sequences:

- $\mathbf{M} := (M^i)_{i\in\mathbb{N}}$ where the sequence elements are from $V = \mathbb{F}^{k_b \times k_b}$.
- $\mathbf{m} := (M^i\mathbf{v})_{i\in\mathbb{N}}$ where the sequence elements are from $V = \mathbb{F}^{k_b}$.
- $\mathbf{m'} := (\mathbf{u}^T M^i \mathbf{v})_{i\in\mathbb{N}}$, where $\mathbf{u}, \mathbf{v} \in_R \mathbb{F}^{k_b}$ and the sequence elements are from $V = \mathbb{F}$.

The following lemma describes some basics properties of those sequences.

Lemma 3.7. *Let $p_M^{char} = det(\lambda I - M)$ be the characteristic polynomial of the matrix $M \in \mathbb{F}^{k_b \times k_b}$, and let $p_{\mathbf{m}'}$, $p_{\mathbf{m}}$, and $p_{\mathbf{M}}$ be the minimal polynomials of the sequences $\mathbf{m}', \mathbf{m}, \mathbf{M}$, respectively. Then $p_{\mathbf{m}'} \mid p_{\mathbf{m}} \mid p_{\mathbf{M}} \mid p_M^{char}$, where $p|q$ means here that the polynomial p divides the polynomial q.*

The proof of Lemma 3.7 is based on the Cayley-Hamilton theorem; details can be found in [28].

Corollary 3.8. *The sequences* $\mathbf{m}', \mathbf{m}, \mathbf{M}$ *are linearly recurrent of order at most* k_b.

We also define the minimal polynomial of a *matrix* M, as follows.

Definition 3.9. *The minimal polynomial* p_M^{min} *of the matrix* $M \in \mathbb{F}^{k_b \times k_b}$ *is defined as* $p_M^{min} = p_{\mathbf{M}}$, *i.e. as the minimal polynomial of the sequence* $\mathbf{M} = (M^i)_{i \in \mathbb{N}}$.

3.7.2 Oblivious computation of the minimal polynomial

Assume that the player T holds the encrypted matrix $\mathcal{F}(M)$, where $M \in \mathbb{F}^{k_b \times k_b}$ and $\mathcal{F}$ is a public-key additively-homomorphic encryption scheme over $\mathbb{F} = \mathbb{F}_{2^k}$. The other player, P, holds the key pair for $\mathcal{F}$. T wishes to compute an encryption of the minimal polynomial of M, $\mathcal{F}(p_M^{min})$ (namely, the encryption of each one of the polynomial's coefficients).

In order to compute the minimal polynomial of the latter matrix, we use Lemma 3.10 which claims that, with high probability, it is sufficient to compute the minimal polynomial of the linearly recurrent sequence $\mathbf{m}' = (\mathbf{u}^\intercal M^i \mathbf{v})_{i \in \mathbb{N}}$ where $\mathbf{u}, \mathbf{v} \in_R \mathbb{F}^{k_b}$.

Lemma 3.10. *Let* p_M^{min} *be the minimal polynomial of a matrix* $M \in \mathbb{F}^{k_b \times k_b}$, *and define the related sequence* $\mathbf{m}' = (\mathbf{u}^\intercal M^i \mathbf{v})_{i \in \mathbb{N}}$, *where* $\mathbf{u}, \mathbf{v} \in_R \mathbb{F}^{k_b}$ *are two randomly selected vectors. Then* $p_{\mathbf{M}} = p_{\mathbf{m}'}$ *with probability at least* $1 - 2deg(p_{\mathbf{M}})/|\mathbb{F}|$.

(The proof of Lemma 3.10 is omitted here; it can be found in [42, Exercise 12.15].)

Since, by Corollary 3.8, the recursion order of the sequence $\mathbf{m}'$ is at most k_b, it is sufficient to compute the minimal polynomial using only the first $2k_b$ elements of the sequence. To that end, we shall invoke the Berlekamp–Massey algorithm, which we describe later on.[5]

In view of the above, we proceed to show how T can obtain the encryption of the first $2k_b$ elements in the sequence $\mathbf{m}' = (\mathbf{u}^T M^i \mathbf{v})_{i \in [2k_b]}$ where $\mathbf{u}, \mathbf{v} \in_R \mathbb{F}^{k_b}$, from the input $\mathcal{F}(M)$ that it holds. First, T and P compute $\mathcal{F}(M^{2^j})$ for $0 \leq j \leq \log(k_b)$. This is done by performing $\log(k_b)$ sequential matrix multiplications by using the square-and-multiply method. The multiplication of encrypted matrices is done by invoking Routine 3 (see Section 3.6.1) for multiplying encrypted scalars.

Next, T computes $\mathcal{F}(M\mathbf{v})$ using Routine 2 (Section 3.6.1). Then, T uses $\mathcal{F}(M^2)$ in order to compute

$$\mathcal{F}(M^3\mathbf{v}|M^2\mathbf{v}) = \mathcal{F}(M^2) \cdot \mathcal{F}(M\mathbf{v}|\mathbf{v})$$

where hereinafter, if X and Y are matrices with the same number of rows then $X|Y$ denote their horizontal concatenation. T proceeds to use the same technique in order to get

$$\mathcal{F}(M^7\mathbf{v}|M^6\mathbf{v}|M^5\mathbf{v}|M^4\mathbf{v}) = \mathcal{F}(M^4) \cdot \mathcal{F}(M^3\mathbf{v}|M^2\mathbf{v}|M\mathbf{v}|\mathbf{v})\,.$$

T proceeds in this manner until it gets

$$\mathcal{F}(M^{2k_b-1}\mathbf{v}|M^{2k_b-2}\mathbf{v}|\ldots|M^{k_b}\mathbf{v}) = \mathcal{F}(M^{k_b}) \cdot \mathcal{F}(M^{k_b-1}\mathbf{v}|\ldots|M\mathbf{v}|\mathbf{v})\,.$$

Finally, by holding the vectors $\mathcal{F}(M^i\mathbf{v})_{i \in [2k_b]}$ and $\mathbf{u}^\intercal$, T computes $\mathcal{F}(\mathbf{u}^\intercal M^i \mathbf{v})$ for $0 \leq i \leq 2k_b - 1$ using Routine 2 from Section 3.6.1.

[5] We note that it is also possible to compute the minimal polynomial via the extended Euclidean algorithm, see [14].

By holding the latter $2k_b$ encrypted values and because the recursion order of $\mathbf{m}'$ is at most k_b, it is possible to compute the minimal polynomial of the sequence $\mathbf{m}'$ via the Berlekamp–Massey algorithm. As that algorithm is well-known, we omit herein further details; the interested reader may refer to [8].

The Berlekamp-Massey Algorithm implies that it is possible to construct a boolean circuit of size $O(k_b^2)$ that computes the minimal polynomial. However, in our setting, it is necessary to compute the boolean circuit over encrypted values. Hence, we proceed to describe a general method that applies Yao's garbled circuit protocol (Section 3.4) in such settings.

Computing Yao's garbled circuit protocol over encrypted values. Suppose that T holds an encrypted value $\mathcal{F}(a)$ where $a \in \mathbb{F}$ and P holds $\mathcal{F}$'s private key. They wish to compute $g(a)$ for some function g, where g can be described as a boolean circuit, C_g. To that end, we describe a modified version of Protocol 3 from Section 3.4.

First, T generates a random value $r \in \mathbb{F}$ and computes $\mathcal{F}(a+r)$ (using Routine 1 in Section 3.6.1) and sends it to P, who proceeds to decrypt it and recover $a+r$. Then, both players create a boolean circuit C'_g that uses r as the input of T and $a+r$ as P's input. They also create "adapter" gates in the beginning of the circuit C'_g that subtract the values r from $a+r$, and then feeds the result s as an input to the garbled circuit C_g. Finally, they continue to follow Yao's GC protocol, until they get the output value $g(a)$. It is easy to see that the privacy and the complexity remain the same as in the original protocol. Furthermore, the extension of that idea in order to compute the circuit with $n \in \mathbb{N}$ encrypted values as inputs, $\mathcal{F}(x_i)_{i\in[n]}$, is straightforward.

We are now ready to describe Protocol 8 for computing an encryption of the minimal polynomial of an encrypted matrix. In Steps 1-3, T computes $\mathcal{F}(\mathbf{u}^\intercal M^i \mathbf{v})$ for $0 \leq i \leq 2k_b - 1$ and $\mathbf{u}, \mathbf{v}$ are randomly selected vectors as descibed above. Next, it invokes the algorithm of Berlekamp–Massey on the latter sequence in order to compute its minimal polynomial. To that end, as we explained above, T should construct a modified Yao's garbled circuit that enables the two players to compute the circuit over encrypted values (Step 4). Finally, in Step 5, T outputs the result.

Protocol 8 Computing an Encryption of the Minimal Polynomial Of an Encrypted Matrix (MinPoly)

Input: T has $\mathcal{F}(M)$ where $M \in \mathbb{F}^{k_b \times k_b}$, $\mathbb{F} = \mathbb{F}_{2^k}$.
Output: T gets $\mathcal{F}(p_M^{min})$ where p_M^{min} is the minimal polynomial of the matrix M.

1: T computes $\mathcal{F}(M^{2^i})\ \forall i \in [\log(k_b)]$.
2: T chooses two random vectors $\mathbf{u}, \mathbf{v} \in_R \mathbb{F}^{k_b}$.
3: T computes $\mathcal{F}(m'_i) = \mathcal{F}(\mathbf{u}^\intercal M^i \mathbf{v})\ \forall i \in [2k_b]$ using the method that is described in Section 3.7.2.
4: T computes $\mathcal{F}(p_{\mathbf{m}'})$ from $\{\mathcal{F}(m'_i) : i \in [2k_b]\}$, using Yao's-garbled circuit for the Berlekamp-Massey algorithm.
5: Return $\mathcal{F}(p_{\mathbf{m}'})$ as the encryption of the minimal polynomial of the matrix M.

Complexity. Consider the notations from Table 3. Then, Step 1 entails $\log(k_b)$ encrypted matrix multiplications of size $k_b \times k_b$. The cost of such operations is $\tilde{O}(k_b^3 \cdot ([\mathcal{F}] + [Mul] + [Inv]))$; they entail 2 rounds, 2 messages and $O(\lambda \cdot k_b^2)$ bits transferred. Then, Step 3 is more involved and consists of computing $\mathcal{F}(M \cdot \mathbf{v})$, $\mathcal{F}(M^i \cdot \mathbf{v})$ and then $\mathcal{F}(\mathbf{u}^\intercal M^i \cdot \mathbf{v})$ for all $i \in [2k_b]$ in complexity of $\tilde{O}(k_b^3 \cdot [Routine\ 3])$ with $\log(k_b)$ rounds, $\log(k_b)$ messages and $O(\lambda \cdot k_b)$ bits transferred. Finally in Step 4, by using the results from Section 3.4, the computation costs $O(k_b^2([\mathcal{F}] + [OT]))$ with 3 rounds, 2 messages, and $O(\lambda \cdot k_b^2)$ bits transferred.

Remark 3.11. *The cost of* $[Routine\ 3]$ *over* $\mathbb{F}_{2^k}$ *equals* $2 \cdot O(k) \cdot [Mul] + 3[Mul] + 2 \cdot O(k^2) + [Inv]$ *for* T*; and* $[\mathcal{F}^{-1}]$ *and one multiplication in the field* $\mathbb{F}_{2^k}$ *for* P*. Due to the technique of multiplying a plain value with an encrypted value over* $\mathbb{F}_{2^k}$*, which is described in Section 3.1.*

3.7.3 Computing the rank of an encrypted matrix

The computation of the encrypted matrix's rank is done by computing its minimal polynomial and using the following lemma [26].

Lemma 3.12. *Let* $B \in \mathbb{F}^{k_b \times k_b}$ *be a matrix of (unknown) rank* r *for which the first* r *leading principals,* $B_1, \ldots, B_r$ *are invertible. Let* X *be a randomly chosen diagonal matrix in* $\mathbb{F}^{k_b \times k_b}$*. Then* B*'s rank* r *equals* $deg(p_{XB}) - 1$ *with probability greater than* $1 - \frac{{k_b}^2}{|\mathbb{F}|}$*.*

In our case, the matrix $M \in \mathbb{F}^{k_a \times k_b}$ for which the rank is sought is not square and does not have the property of leading principals as states in Lemma 3.12. Hence, before applying the lemma, the two players T and P has to perform some preparations. First, in order to transform the matrix M into a square matrix with the same rank, T pads it with $k_b - k_a$ zero columns so that it becomes a square $k_b \times k_b$ matrix of the same rank. Then, in order to transform the matrix to one that has leading invertible principals up to the r'th one, T multiplies the square matrix that it holds by upper and lower Toeplitz matrices. Lemma 3.13 [26] ensures that the resulting matrix has the required property with high probability.

Lemma 3.13. *Let* M *be a matrix in* $\mathbb{F}^{k_b \times k_b}$ *of (unknown) rank* r*. Let* U *and* L *be two random upper and lower unitriangular Toeplitz matrices in* $\mathbb{F}^{k_b \times k_b}$*. (Triangular matrix is called unitriangular, if the entries on the main diagonal are all 1.) Let* $B = UML$ *and denote the* $i \times i$ *leading principal of* B *by* B_i*. Then, the probability that* $det(B_i) \neq 0$ *for all* $1 \leq i \leq r$ *is greater than* $1 - \frac{{k_b}^2}{|\mathbb{F}|}$*.*

Finally, in order to have a matrix that fulfills all requirements of Lemma 3.12, T multiplies the matrix with a random diagonal matrix $X \in \mathbb{F}^{k_b \times k_b}$ (by using Routine 2 from Section 3.6.1). By combining the probabilities of Lemmas 3.13 and 3.12, we infer that the probability of the latter matrix having the same rank r as the original matrix M is greater than $1 - 2\frac{{k_b}^2}{|\mathbb{F}|}$. Since our underlying field $\mathbb{F} = \mathbb{F}_{2^k}$ can be made as large as desired by picking k large enough, we can choose k so that the latter probability is smaller than ε for any given threshold ε.

It is worth mentioning that Lemmas 3.12 and 3.13 follow from the well-known Schwartz-Zippel lemma [40, 46], which (in its weaker version) states the following. (A proof can be found in [34].)

Lemma 3.14 (Schwartz-Zippel)**.** *Let* f *be an* m*-variate polynomial of degree exactly* $d \geq 1$ *over a field* $\mathbb{F}$*. Then the number of zeros of* f *is at most* $d|\mathbb{F}|^{m-1}$*, or alternatively,*

$$Pr_{x \in \mathbb{F}^m}[f(x) = 0] \leq \frac{d}{|\mathbb{F}|}\,.$$

We may now describe Protocol 9 for rank computations. In Steps 1-3, T performs the needed transformations on the matrix M to a matrix N that has the same rank (with high probability) and complies with the conditions of Lemma 3.12 that allow computing the rank by the degree of the minimal polynomial. Then (Step 4), T and P invoke Protocol 8 to obtain the encrypted degree of the minimal polynomial which is, with high probability, the rank of the original matrix.

Protocol 9 Computing the Rank of an Encrypted Matrix (ComputeRank)

Input: T has $\mathcal{F}(M)$ where $M \in \mathbb{F}^{k_a \times k_b}$, $\mathbb{F} = \mathbb{F}_{2^k}$, and $k_a \leq k_b$; P has the private decryption key of additively-homomorphic encryption function $\mathcal{F}$.

Output: T gets $\mathcal{F}(rank(M))$.

1: T pads $k_b - k_a$ columns of $\mathcal{F}(M)$ with $\mathcal{F}(0)$ and gets $\mathcal{F}(M')$ of size $k_b \times k_b$.
2: T picks a random upper unitriangular Toeplitz matrix U, a random lower unitriangular Toeplitz matrix L, and a random diagonal matrix X, all from $\mathbb{F}^{k_b \times k_b}$.
3: T computes $\mathcal{F}(N)$ where $N = XUM'L$, using Routine 2 from Section 3.6.
4: T and P invoke Protocol 8 ($MinPoly$) on N except that in the last step, they use a circuit that only outputs to T the encrypted degree of the minimal polynomial $deg(p_N^{min})$ instead of the encryption of the actual polynomial p_N^{min}.

Complexity. Consider the notations from Table 3 and Remark 3.11. Then, Step 1 costs are $[\mathcal{F}] \cdot k(k_b - k_a)$. Next in Step 3, the multiplication of two unitriangular Toeplitz matrices and one diagonal matrix costs $O(k_b^3([Mul] + k^2))$. Finally, Step 4 costs are: $\tilde{O}(k_b^3 \cdot [Routine\ 3])$ with $\log(k_b)$ rounds, $\log(k_b)$ messages and $O(\lambda \cdot k_b^2)$ bits transferred, according to Protocol 8.

Privacy. Since all computations are done on $\mathcal{F}$-encrypted values, the protocol's privacy derives from the security of $\mathcal{F}$ and the security of Yao's garbled circuit protocol and Routines 1,2 from Section 3.6.1.

4 Privacy preserving planarity testing

We begin this section with a formal definition of our problem (Section 4.1). We then provide a birds-eye view of the way in which we intend to tackle it (Section 4.2). In the subsequent Sections 4.3, 4.4, 4.5, 4.6, we provide the details of our proposed solutions. We analyze the privacy of the proposed protocols in Section 4.7, and their computational and communication costs in Section 4.8.

4.1 Problem definition

There are two players, P_1 and P_2, each one holding a private planar graph on the same set of vertices, $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$. They wish to determine whether the union graph $G = (V, E)$, where $E = E_1 \cup E_2$ is planar as well, without revealing any additional information to the other player on one's private graph. The algorithm that we proceed to describe assumes a third party which is non-trusted. The third party T will help in executing the computations but it is not allowed to learn information on the private graphs. We assume that all parties are semi-honest. Namely, they follow the protocols' specifications, but at the same time they try to extract from their own view during the protocols' run information on the input private graph of the other players.

Comment. We assume that there are only two players merely for the sake of simplicity. The extension of our protocols to any number of players is straightforward.

4.2 Overview of the proposed solutions

4.2.1 First stage – testing the number of edges in the unified graph

Denote $V = \{v_1, \ldots, v_n\}$. The unified graph $G = (V, E)$ is planar only if

$$|E| = |E_1 \cup E_2| \leq 3n - 6\,. \tag{6}$$

Hence, in the first stage, the three players, P_1, P_2 and T, engage in a secure protocol for checking whether inequality (6) 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 protocol which we outline in Section 4.2.2.

The protocol for verifying inequality (6) is given in Section 4.3. The only information that the three players learn after running this protocol is whether the inequality holds or not; they do not learn any further information on the size of E_1, E_2, $E_1 \cap E_2$, or $E_1 \cup E_2$.

4.2.2 Second stage – a privacy-preserving implementation of the Hanani-Tutte planarity test

The idea in this stage is that the three players construct the Hanani-Tutte (HT hereinafter) system of linear equations, Eq. (2), for the unified graph G. Towards that end, they begin by constructing the HT 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). That stage can be constructed publicly with no privacy risks, since the vertex set V is known to all and K_V is simply the complete graph on V. The main effort is in extracting from that large system of equations the subset of HT equations corresponding to the unified graph G; in other words, the goal of that part of the protocol is to let the mediator T have the subset of equations that relate to two independent edges in K_V which both exist in G. To protect the unified graph data from T, it will get only an encrypted version of the subset of linear equations corresponding to G. The last part of the computation is dedicated to determining whether

that system has a solution or not (where no party actually sees the system as it is held only by T and it is encrypted using a cipher that T cannot decrypt).

To allow this approach, we must start with a basic drawing of K_V (which induces also a basic drawing for the sub-graph G). Let us consider the following embedding of V in $\mathbb{R}^2$. If $V = \{v_1, \ldots, v_n\}$, then v_j is mapped into the point

$$v_j \mapsto \chi(v_j) := (\cos(2\pi j/n), \sin(2\pi j/n)), \quad 1 \leq j \leq n. \tag{7}$$

Then K_V's edges are

$$e_{i,j} = \{v_i, v_j\}, \quad 1 \leq i < j \leq n. \tag{8}$$

Let D denote the drawing of K_V in which the vertices are embedded as in Eq. (7) and an edge $e_{i,j}$ is represented by the straight line segment between $\chi(v_i)$ and $\chi(v_j)$. Consider now the two edges $e_{i,j}$ and $e_{k,\ell}$ and assume, WLOG, 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 system of linear equations (2) can be constructed publicly, by each of P_1, P_2 and T, for this drawing D of the complete graph K_V. Let us denote the set of all pairs of independent edges in K_V by K_2^{ind}. Since K_V has $\binom{n}{2}$ edges, we infer that

$$N_{K_V} := |K_2^{ind}| = \frac{1}{2} \cdot \binom{n}{2} \cdot \binom{n-2}{2}. \tag{9}$$

The system (2) will include an equation for each such pair of independent edges. 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.

The graph $G = (V, E)$ is planar, iff that partial system of $|E_2^{ind}|$ equations has a solution. In Sections 4.4 and 4.5 we present our protocols for carrying out the computation that we outlined herein.

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

Here we present Protocol 10 that allows the two parties, P_1 and P_2 (with the help of T) to compare the size of the unified edge set $E = E_1 \cup E_2$ against the planarity upper bound $3n - 6$, Eq. (6), in order to decide whether they need to proceed with the planarity testing.

Let $E_{K_V} := \{e_{i,j} : 1 \leq i < j \leq n\}$ denote the edge set in the complete graph K_V, where $e_{i,j}$ were defined in Eq. (8). For each $e \in E_{K_V}$ and $h \in \{1, 2\}$, let α_e^h be a bit that indicates whether E_h contains the edge e. Then $e \in E$ iff

$$\alpha_e^1 \vee \alpha_e^2 = \alpha_e^1 + \alpha_e^2 - \alpha_e^1 \cdot \alpha_e^2 = 1. \tag{10}$$

Hence,

$$|E| = \sum_{1 \leq i < j \leq n} (\alpha_{e_{i,j}}^1 + \alpha_{e_{i,j}}^2 - \alpha_{e_{i,j}}^1 \cdot \alpha_{e_{i,j}}^2). \tag{11}$$

Therefore, P_1 and P_2 may invoke the BGW protocol to compute additive shares in $|E|$ in a secure manner. Specifically, by fixing a prime $p > \binom{n}{2}$, the two players may run the BGW protocol for computing shares s_1 and s_2 in $|E|$ over the field $\mathbb{F}_p$ so that each of the two shares distributes uniformly in $\mathbb{F}_p$ (and, hence, conveys no information on $|E|$ to the party that holds it) and $s_1 + s_2 = |E| \mod p$. This first stage in the computation is described in Steps 1-5 of Protocol 10.

Next, the two players continue to check whether $|E|$, which is the sum modulo p of the two random shares that they hold, is greater than $3n-6$. That part of the computation is carried out in Steps 6-10 of Protocol 10, which we proceed to explain.

In Step 6, P_1 generates a random number $r \in [0, p - \binom{n}{2})$. Since $s_1 + s_2 \mod p = |E|$ and $|E| \leq \binom{n}{2}$, then the value y which T recovers in Step 9 equals $|E| + r$, where the latter sum is in the standard sense of integers. (The selection of p to be significantly larger than the theoretical upper bound on $|E|$ is made so that T can extract almost no information on $|E|$ from y, as we discuss later on. For the correctness of the protocol, any $p > \binom{n}{2}$ would do.) Finally, T (that has $y = |E| + r$) and P_1 (that has $r + (3n-6)$) compare the two values in order to check the necessary condition for planarity. They do so by running a protocol for Yao's Millionaires' problem [37] (which we described in Section 3.3).

Protocol 10 Testing the size of the unified edge set

1: **for** $h = 1, 2$ **do**
2: P_h sets values to α_e^h for all $e \in E_{K_V}$; specifically, $\alpha_e^h = 1$ if $e \in E_h$ and $\alpha_e^h = 0$ otherwise.
3: **end for**
4: P_1 and P_2 choose a finite field $\mathbb{F}_p$ where $p \gg \binom{n}{2}$.
5: P_1 and P_2 invoke BGW protocol in order to compute additive shares, s_1 and s_2, in $|E| = \sum_{1 \leq i < j \leq n} (\alpha_{e_{i,j}}^1 + \alpha_{e_{i,j}}^2 - \alpha_{e_{i,j}}^1 \cdot \alpha_{e_{i,j}}^2)$ over $\mathbb{F}_p$.
6: P_1 will generate a random $r \in [0, p - \binom{n}{2})]$.
7: P_1 sends to T the value $s_1 + r \mod p$.
8: P_2 sends to T the value s_2.
9: T computes $y := s_1 + s_2 + r \mod p$.
10: T and P_1 engage in a Yao's millionaires' protocol to check whether $y \leq r + (3n-6)$.
11: If the last inequality verification fails, then the unified graph is not planar.

Privacy analysis. The first part of the protocol (Steps 1-5) involves only P_1 and P_2. They engage in a BGW protocol for computing shares in $|E|$. The latter protocol is information-theoretic secure [7] and, hence, neither of the two players gets any wiser during that part of the protocol. In the second part P_2 is not involved and P_1 and T engages in a Yao's millionaires' protocol, which is also perfectly secure. Finally, we consider T who recovers in Step 9 the value $y = |E| + r$. That value may reveal some information on $|E|$ in probability that is $O(1/p)$, as implied by Lemma 4.1 below. Hence, by choosing p to be sufficiently large, P_1 and P_2 can guarantee that the probability of T inferring an upper or a lower bound on $|E|$ to be negligible.

Lemma 4.1. *Let: (a) w be an integer random variable taking values in $[B] = \{0, 1, 2, \ldots, B\}$; (b) r be uniformly distributed over $[R]$ for $R \geq B$; and (c) $x = w + r$. Then in probability $1 - \frac{B}{R+1}$, x reveals no information on w, while otherwise x reveals either an upper or a lower bound on w, but nothing beyond that.*

Lemma 4.1 (see [21, Lemma 4] for a proof) implies that the probability of T inferring anything on $|E|$ is no larger than $\frac{\binom{n}{2}}{p - \binom{n}{2}} = O(1/p)$.

Complexity. The bulk of the computational and communication costs of Protocol 10 is due to the BGW sub-protocol and Yao's millionaires' protocol on integers from $\mathbb{F}_p$. In the BGW sub-protocol, the two players need to compute $\binom{n}{2}$ multiplication gates and $3\binom{n}{2} - 1$ addition gates.

4.4 A first algorithm for private HT testing

Let

$$E_{K_V} := \{e_{i,j} : 1 \leq i < j \leq n\} \tag{12}$$

denote the edge set in the full graph K_V, where $e_{i,j}$ were defined in Eq. (8). For each edge $e \in E_{K_V}$ 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 $\alpha_e^1 \vee \alpha_e^2$. Consequently, the equation that corresponds to the pair of potential edges e and f, where $(e,f) \in K_2^{ind}$, is relevant iff

$$\chi_{e,f} := (\alpha_e^1 \vee \alpha_e^2) \wedge (\alpha_f^1 \vee \alpha_f^2) = 1\,. \tag{13}$$

Each of P_1, P_2 and T can construct the following system of linear equations for the drawing D of K_V which we described in Section 4.2.2:

$$x_{e,a(f)} + x_{e,b(f)} + x_{f,a(e)} + x_{f,b(e)} = parity_D(e,f) \mod 2 \qquad (e,f) \in K_2^{ind}\,. \tag{14}$$

That system has N_{K_V} equations (see Eq. (9)) – one equation for each pair of independent edges in K_V. The system of equations which determines the planarity of $G = (V,E)$ is a subset of the system in (14), see Eq. (2). That subset of equations includes only the equations relating to pairs of edges (e,f) where both e and f are in E. Hence, the problem now is to enable T to identify (in an oblivious manner) the subset of $|E_2^{ind}|$ equations, out of the system of N_{K_V} equations in (14), that correspond to pairs of independent edges in E_2^{ind} (namely, pairs of edges (e,f) where both e and f are in E and they are independent). Once that subset of equations is constructed, we proceed to test its solvability. This is done in Protocol 11 which we proceed to describe.

The protocol begins with P_1 and P_2 generating, privately, homomorphic encryption functions, as we described in Section 3.1 (Steps 1-2). They keep all decryption keys secret from T but notify it of the corresponding public encryption keys.

In Steps 3-5, each player sends to mediator T the encrypted bit values α_e^h for all every edge in the full grave K_V. Each such bit indicates whether that edge exists in P_h's private edge set E_h.

Next, in Steps 6-11, T computes for all pairs of independent edges in the full graph (K_2^{ind}) an $\mathcal{E}_1$-encryption of $\xi_{e,f} = (\alpha_e^1 + \alpha_e^2) \cdot (\alpha_f^1 + \alpha_f^2)$. Specifically, it first computes $\mathcal{E}(\alpha_e^1 + \alpha_e^2)$ by multiplying $\mathcal{E}(\alpha_e^1)$ and $\mathcal{E}(\alpha_e^2)$ (relying on the additive homomorphism of $\mathcal{E}$); then it computes, in a similar manner $\mathcal{E}(\alpha_f^1 + \alpha_f^2)$; finally, from the latter two values it constructs the $\mathcal{E}_1$-encryption of $\xi_{e,f}$ by the multiplication scheme that was devised at [9] (see Section 3.1).

Table 4 lists all possible values of $\xi_{e,f}$ as a function of the private bits α_e^h, α_f^h, $h = 1,2$. The goal is now to identify the equations in the linear system for which $\xi_{e,f} = 0$ since those equations correspond to pairs of edges that neither of which exists in E; those equations should be omitted. All equations corresponding to pairs of edges for which $\xi_{e,f} \in \{1,2,4\}$ should be retained.

In order to identify those equations it is necessary to decrypt $\mathcal{E}_1(\xi_{e,f})$. T cannot do that but P_1 and P_2 can. However, T cannot just hand over those encrypted values to P_1 (or P_2) since they may reveal sensitive information on the private graph of the other player. Indeed, as can be seen from Table 4, the value of $\xi_{e,f}$ reveals to P_1 (who knows the column index) information on P_2's input (which is the row index).

Hence, T performs two actions before handing over to P_1 all of the $\mathcal{E}_1$-encrypted values to decrypt. In Steps 8-9, T replaces $\mathcal{E}_1(\xi_{e,f})$ with $\mathcal{E}_1(\xi_{e,f})^{\rho_{e,f}}$, for some nonzero random multiplier $\rho_{e,f}$.

$\xi_{e,f}$	0,0	0,1	1,0	1,1
0,0	0	0	0	1
0,1	0	0	1	2
1,0	0	1	0	2
1,1	1	2	2	4

Table 4: The values of $\xi_{e,f}$ as a function of the values of (α_e^1, α_f^1) (columns) and (α_e^2, α_f^2) (rows)

Owing to the homomorphism with respect to addition, then the latter value equals $\mathcal{E}_1(\xi_{e,f} \cdot \rho_{e,f})$. It is easy to see that $\xi_{e,f} \cdot \rho_{e,f} = 0$ iff $\xi_{e,f} = 0$. However, if $\xi_{e,f} \in \{1, 2, 4\}$, then the value of $\xi_{e,f} \cdot \rho_{e,f}$ reveals no information on the original value of $\xi_{e,f}$ (except for the fact that it was nonzero). The second action that T takes is in Step 12. Here, T sends to P_1 the values of $w(e, f)$ under some secret permutation. Therefore, while P_1 can decrypt the entries of $w(e, f)$, it will only be able to know how many pairs of edges in the full graph had $\xi_{e,f} \neq 0$, but not the identity of those edges. We defer the full discussion of privacy to Section 4.7.1.

In addition to the above, T computes for each pair of independent edges, (e, f), the bit $\sigma_{e,f}$ which indicates whether they intersect or not (Step 10). Here we rely on the specific drawing of the graph as given in Eqs. (7)+(8).

Then, in steps 13-21, P_1 decrypts the entries of the $\hat{\mathbf{w}}$ that it received from T. If the decryption of a given entry, w', is zero then $\chi_{e,f} = 0$ (see Eq. (13)); otherwise $\chi_{e,f} = 1$. P_1 sends back to T the vector $\hat{\mathbf{w}}$ in which every entry that had $w' = 0$ is replaced with $\mathcal{F}(0)$ and all others are replaced by $\mathcal{F}(1)$. Then, in Step 22, T applies the inverse permutation on the entries of $\hat{\mathbf{w}}$ and gets for each $(e, f) \in K_2^{ind}$ a value $\mathcal{F}(\chi_{e,f})$ where $\chi_{e,f}$ is a binary flag indicating whether $(e, f) \in E_2^{ind}$.

Next, in the loop in Steps 23-35, T constructs the full system of linear equations corresponding to E_2^{ind}, under the encryption $\mathcal{F}$. Recall that T has the full set of N_{K_V} equations in the clear. The goal of this stage in the computation is to allow T to compute an $\mathcal{F}$-encryption of the subset of $|E_2^{ind}|$ relevant equations. Specifically, if a given row in the full matrix is relevant, we wish for T to get an entry-wise $\mathcal{F}$-encryption of it. If, on the other hand, a given row is not relevant, we wish for T to get an entry-wise $\mathcal{F}$-encryption of the row which consists of zeroes. It is needed to do so without T learning which rows were zeroed and which rows remained.

This is how it is done: Let $\mathbf{r}_{e,f}$ be the row in the augmented matrix that corresponds to $(e, f) \in K_2^{ind}$. $\mathbf{r}_{e,f}$ is a binary vector of dimension $N + 1$, 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 ($\sigma_{e,f} = parity_D(e, f)$). In the linear equation corresponding to (e, f) the coefficients of all variables are zero, except for 4 of those variables (see Eq. (2)). Hence, in the inner loop in Steps 26-33, T goes over the first N entries of $\mathbf{r}_{e,f}$; in each of the 4 entries that should be 1, T places the value $\mathcal{F}(\chi_{e,f})$, while in all the remaining ones it places the value $\mathcal{F}(0)$ (recall that $\mathcal{F}$ is a public-key cipher, so T can compute encryptions). 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 that $\sigma_{e,f} = 1$; otherwise T places the value $\mathcal{F}(0)$, in order to prevent from the parity's value to affect a row that is not relevant (Steps 34-38). As a result, if $\chi_{e,f} = 0$, T constructs (in a manner oblivious to it) 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. (2). In summary, T gets a system of N_{K_V} encrypted equations: $|E_2^{ind}|$ of those equations are the $\mathcal{F}$-encryption of the system (2), while the remaining ones are $\mathcal{F}$-encryptions of the trivial equation with all coefficients

and right hand side being zero.

Finally, in Steps 40-45, T and P_1 execute the the algorithm SOLVABLE in order to find out whether the above-constructed encrypted system of equations has a solution. The graph is planar iff the system is solvable.

Protocol 11 Privacy preserving HT planarity testing

P_1 and P_2 generate public key encryption functions $\mathcal{E}$, $\mathcal{E}_1$ as described in [9] and send to T the public encryption key.
P_1 and P_2 generate a probabilistic public key encryption function $\mathcal{F}$ which is additively homomorphic over $\mathbb{F}_{2^k}$ and send to T the public encryption key.
for $h = 1, 2$ **do**
    P_h sends the encrypted values $\{\mathcal{E}(\alpha_e^h) : e \in E_{K_V}\}$ to the mediator T.
end for
for all $(e, f) \in K_2^{ind}$ **do**
    T computes $\mathcal{E}_1(\xi_{e,f})$ where $\xi_{e,f} := (\alpha_e^1 + \alpha_e^2) \cdot (\alpha_f^1 + \alpha_f^2)$.
    T generates a random integer multiplier $1 \leq \rho_{e,f} < \nu = |\mathbb{G}_1|$.
    T computes $w(e, f) := \mathcal{E}_1(\xi_{e,f})^{\rho_{e,f}} = \mathcal{E}_1(\xi_{e,f} \cdot \rho_{e,f})$.
    If $e = e_{i,j}$ and $f = e_{k,\ell}$ where $i < j$, $k < \ell$ and $i < k$, T sets $\sigma_{e,f} \leftarrow \begin{cases} 1 & k < j < \ell \\ 0 & \text{otherwise} \end{cases}$
end for
T sends to P_1 the vector $\hat{\mathbf{w}} \leftarrow (w(e, f) : (e, f) \in K_2^{ind})$, where the entries of $\hat{\mathbf{w}}$ are secretly and randomly permuted by T.
for $1 \leq i \leq N_{K_V} = |K_2^{ind}|$ **do**
    P_1 computes $w' := \mathcal{E}_1^{-1}(\hat{\mathbf{w}}(i))$
    if $w' = 0$ **then**
        $\hat{\mathbf{w}}(i) \leftarrow \mathcal{F}(0)$
    else
        $\hat{\mathbf{w}}(i) \leftarrow \mathcal{F}(1)$
    end if
end for
P_1 sends to T the vector $\hat{\mathbf{w}}$.
T applies the inverse permutation on the entries of $\hat{\mathbf{w}}$ and gets for each $(e, f) \in K_2^{ind}$ a value $\mathcal{F}(\chi_{e,f})$ where $\chi_{e,f}$ is a binary flag indicating whether $(e, f) \in E_2^{ind}$.
for all $(e, f) \in K_2^{ind}$ **do**
    T allocates a vector $\mathbf{r}_{e,f}$ of dimension $N + 1$ where $N := \binom{n}{2} \cdot (n - 2)$.
    T creates a bijection $\Phi : [N] \to \{(g, v) : g \in E_{K_V}, v \in V \setminus \{a(g), b(g)\}\}$.
    for $i \in [N]$ **do**
        $(g, v) \leftarrow \Phi(i)$.
        if ($g = e$ and $v \in \{a(f), b(f)\}$) or ($g = f$ and $v \in \{a(e), b(e)\}$) **then**
            $\mathbf{r}_{e,f}(i) \leftarrow \mathcal{F}(\chi_{e,f})$
        else
            $\mathbf{r}_{e,f}(i) \leftarrow \mathcal{F}(0)$
        end if
    end for
    if $\sigma_{e,f}$=1 **then**
        $\mathbf{r}_{e,f}(N + 1) \leftarrow \mathcal{F}(\chi_{e,f})$
    else
        $\mathbf{r}_{e,f}(N + 1) \leftarrow \mathcal{F}(0)$
    end if
end for
Execute SOLVABLE($\mathbf{r}_{e,f} : (e, f) \in K_2^{ind}$).
if SOLVABLE returns **true then**
    Output "The union graph is planar".
else
    Output "The union graph is non-planar".
end if

4.5 A second algorithm for private HT testing

Our second algorithm for testing graph planarity using HT theorem is given in Protocol 12. It has a significantly reduced computational complexity compared to algorithm in Protocol 11, as we proceed to explain. The idea behind this variant of the algorithm is to reduce the HT linear system of equations only to those equations corresponding to pairs of edges in E_2^{ind} (see Eq. (2)) before verifying its solvability. Specifically, this is done by sending the whole full encrypted matrix (that contain an equation for every pair of edges in K_2^{ind}) to T, along with the information needed to determine the subset of relevant edge pairs from E_2^{ind}, and to do that in an oblivious manner.

The protocol begins in Step 1 by generating an homomorphic encryption function, in similarity to Step 2 in Protocol 11. Next, in Step 2, P_1 and P_2 agree on a permutation π over K_2^{ind}; that permutation will be used in order to hide from T the exact order of the rows that it will receive and prevent from it from inferring which specific rows it will eliminate later (in Step 23). In Step 3, P_1 and P_2 generate a bijection Φ, in similarity to the one defined in Step 25 in Protocol 11; that bijection determines the order of columns (variable) in the linear system of equations.

In the next loop (Steps 4-18), P_1 and P_2 construct an entry-wise $\mathcal{F}$-encryption of the linear equation corresponding to each edge-pair in K_2^{ind}; in addition they generate for each such pair random additive shares in a flag that determines whether that edge-pair is in E_2^{ind} or not. Specifically, they generate in Step 5 a row vector for each potential equation (i.e., an edge-pair in K_2^{ind}) and then fill its entries with the entry-wise $\mathcal{F}$-encrypted values: the coefficients are treated in Steps 6-13, while the right hand side is treated in Steps 14-15. At this stage, P_1 and P_2 has an $\mathcal{F}$-encryption of the full linear system over K_2^{ind}. In order to enable the identification of the relevant subset of equations corresponding to edge-pairs in E_2^{ind}, P_1 and P_2 proceed to jointly compute random additive shares in the expression $\xi_{e,f} = (\alpha_e^1 + \alpha_e^2) \cdot (\alpha_f^1 + \alpha_f^2)$ (Step 16). The computation is done by invoking the BGW protocol (Section 3.5). Because the value $\xi_{e,f}$ can be any value in $\{0, 1, 2, 4\}$, it is sufficient to do the computation over a field of size at least 5. The value $\xi_{e,f}$ contains "too much" information; we only care whether $\xi_{e,f} = 0$ or $\xi_{e,f} \neq 0$. Revealing the exact value of a non-zero $\xi_{e,f}$ may reveal to T excessive information on the overlapping between the two private graphs. Hence, in order to eliminate that excessive information, P_1 and P_2 multiply their shares with some nonzero random multiplier (Step 17), in similarity to Step 9 in Protocol 11.

Then, in Steps 19-20, T gets from P_1 and P_2 the full encrypted matrix of $|K_2^{ind}|$ equations and the shared values of the randomly multiplied $\xi_{e,f}$ for all $\{e, f\} \in K_2^{ind}$.

In Steps 21-24, T combines the shares that it received from P_1 and P_2 in order to see which rows correspond to relevant edge pairs in E_2^{ind}. Subsequently, T holds an entry-wise $\mathcal{F}$-encryption of the relevant $|E_2^{ind}|$ equations. Finally, Steps 26-31 are the equivalent of Steps 40-45 in Protocol 11: in those steps the players test whether the linear system has a solution and consequently find out whether the union graph is planar.

4.6 Testing the solvability of an encrypted system of linear equations

Here we describe two possible implementations of the sub-protocol SOLVABLE, which is invoked by Protocols 11 and 12. That sub-protocol receives a system of linear equations over $N = \binom{n}{2} \cdot (n-2)$ unknowns (which are $\{x_{g,v} : g \in E_{K_V}, v \in V \setminus \{a(g), b(g)\}\}$). The number of equations, which we denote below by N_e, is $N_e = N_{K_V}$, in the case of Protocol 11, and $N_e = |E_2^{ind}|$ in the case of Protocol 12. We let M denote the $N_e \times N$ matrix of coefficients, $\mathbf{b}$ denote the right hand side vector, and $M' = (M, \mathbf{b})$ be the corresponding augmented matrix. There are two players in the sub-protocol SOLVABLE: T and P_1. T holds an encryption $\mathcal{F}(M')$, where

Protocol 12 Privacy preserving HT planarity testing

P_1 and P_2 generate a probabilistic public key encryption function $\mathcal{F}$ which is additively homomorphic over $\mathbb{F}_{2^k}$ and send to T the public encryption key.
P_1 and P_2 secretly agree on a random permutation π over K_2^{ind}.
Letting $N := \binom{n}{2} \cdot (n-2)$, P_1 and P_2 create a bijection $\Phi : [N] \to \{(g,v) : g \in E_{K_V}, v \in V \setminus \{a(g), b(g)\}\}$.
for all $(e,f) \in K_2^{ind}$ **do**
    P_1 and P_2 allocate a vector $\mathbf{r}_{e,f}$ of dimension $N+1$.
    for $i \in [N]$ **do**
        $(g,v) \leftarrow \Phi(i)$.
        if ($g = e$ and $v \in \{a(f), b(f)\}$) or ($g = f$ and $v \in \{a(e), b(e)\}$) **then**
            $\mathbf{r}_{e,f}(i) \leftarrow \mathcal{F}(1)$
        else
            $\mathbf{r}_{e,f}(i) \leftarrow \mathcal{F}(0)$
        end if
    end for
    if $e = e_{i,j}$, $f = e_{k,\ell}$, $i < j$, $k < \ell$ and $i < k$, **then** $\sigma_{e,f} \leftarrow \begin{cases} 1 & k < j < \ell \\ 0 & \text{otherwise} \end{cases}$
    $\mathbf{r}_{e,f}(N+1) \leftarrow \mathcal{F}(\sigma_{e,f})$
    P_1 and P_2 invoke the BGW protocol in order to compute additive shares $s^1_{e,f}$, $s^2_{e,f}$ in the expression $\xi_{e,f} = (\alpha^1_e + \alpha^2_e) \cdot (\alpha^1_f + \alpha^2_f)$ over some field $\mathbb{F}_q$ where q is a prime ≥ 5.
    P_1 and P_2 generate a secret random $\rho_{e,f} \in \mathbb{F}^*_q$ and then update $s^h_{e,f} \leftarrow s^h_{e,f} \cdot \rho_{e,f}$, $h = 1, 2$.
end for
P_1 sends to T the matrix $(\mathbf{r}_{e,f} : (e,f) \in K_2^{ind})$.
P_h, $h = 1, 2$, sends to T the vector $(s^h_{e,f} : (e,f) \in K_2^{ind})$.
for all $(e,f) \in K_2^{ind}$ **do**
    if $s^1_{e,f} + s^2_{e,f} = 0$ **then**
        T discards the row vector $\mathbf{r}_{e,f}$ from the matrix $(\mathbf{r}_{e,f} : (e,f) \in K_2^{ind})$.
    end if
end for
Execute SOLVABLE($\mathbf{r}_{e,f} : (e,f) \in E_2^{ind}$).
if SOLVABLE returns **true then**
    Output "The union graph is planar".
else
    Output "The union graph is non-planar".
end if

$\mathcal{F}$ is a probabilistic additively homomorphic encryption over some binary field $\mathbb{F}_{2^k}$. In the first implementation, SOLVABLE$_1$ (described in Sub-Protocol 13), the field is $\mathbb{F}_{2^k}$ for some sufficiently large k. In the second implementation, SOLVABLE$_2$ (Sub-Protocol 14), the field is $\mathbb{F}_2$. The private decryption key of $\mathcal{F}$ is held by P_1. The goal is to determine whether the system $M' = (M, \mathbf{b})$ has a solution.

4.6.1 Solvable : version 1

Sub-Protocol 13 SOLVABLE$_1$

Input: T has $\mathcal{F}(M')$; P_1 has the private decryption key of $\mathcal{F}$.
Output: A bit indicating whether the linear system of equations M' is solvable.
1: T, with the help of P_1, computes $\mathcal{F}(rank_1) \leftarrow ComputeRank(M')$.
2: T, with the help of P_1, computes $\mathcal{F}(rank_2) \leftarrow ComputeRank(M)$.
3: T computes $\mathcal{F}(\delta) := \mathcal{F}(rank_1) \cdot \mathcal{F}(rank_2)^{-1} = \mathcal{F}(rank_1 - rank_2)$ and sends it to P_1.
4: P_1 decrypts and recovers δ.
5: **if** $\delta = 0$ **then**
6: return **true**
7: **else**
8: return **false**
9: **end if**

In Steps 1-2 of Sub-Protocol 13, T, the mediator, computes the $\mathcal{F}$-encrypted rank of the augmented matrix M' and the matrix of coefficients M, using Protocol 9, which we described in Section 3.7.

Next, in Step 3, using the homomorphic property of the encryption $\mathcal{F}$, T computes the encrypted value of the difference between the two ranks and sends it to P_1, who decrypts it in Step 4. By Rouché-Capelli theorem, the system represented in M' (namely, $M\mathbf{x} = \mathbf{b}$) is solvable if and only if the rank of M equals that of M', or equivalently, if and only if $\delta = 0$. Hence, P_1 outputs **true** or **false** accordingly (Steps 5-9).

4.6.2 Solvable : version 2

Sub-Protocol 14 SOLVABLE$_2$

Input: T has $\mathcal{F}(M')$; P_1 has the private decryption key of $\mathcal{F}$.
Output: A bit indicating whether the linear system of equations M' is solvable.
1: T invokes Protocol 7 with inputs $\mathcal{F}(M)$, $\mathcal{F}(\mathbf{b})$ and gets $\mathcal{F}(flag)$.
2: T sends to P_1 the value $\mathcal{F}(flag)$.
3: P_1 decrypts and recovers $flag$.
4: **if** $flag = 1$ **then**
5: return **true**
6: **else**
7: return **false**
8: **end if**

In Sub-Protocol 14 we rely upon the procedure that was given in [35] (and we described here in Section 3.6) that enables performing oblivious Gaussian elimination over a system of linear equations that is encrypted by an additively homomorphic encryption over $\mathbb{F}_2$.

Sub-Protocol 14 starts with T invoking Protocol 7 on the encrypted coefficient matrix M, concatenate with the encrypted parity vector b (Step 1). The output that it received is the encryption of the variable $flag$. This variable determines (with high probability) whether the system of equations is solvable. Because T does not has the decryption key, in Step 2 it sends the value to P_1, who decrypts it (Step 3). Then, in Steps 4-8, P_1 returns whether the augmented matrix is solvable according to the value of $flag$.

4.7 Privacy analysis

4.7.1 Protocol 11

The potential leakages of information to any player is due to messages that it receives from other players. We proceed to analyze below what each player may deduce from the messages that it receives during the algorithm.

In Steps 3-5, T receives from each of P_1 and P_2, their indicator variables under the encryption $\mathcal{E}$. Assuming that the *subgroup decision problem* [9] is hard, then breaking $\mathcal{E}$ is hard. Therefore, under that assumption, T cannot decrypt the messages that it receives in this stage, nor learn any information on the underlying plaintexts. Furthermore, as the encryption function $\mathcal{E}$ is probabilistic, given $\mathcal{E}(x)$ and $\mathcal{E}(y)$, it is hard to decide whether $x = y$ or $x \neq y$. Hence, as T expects to receive from each player a vector of length $\binom{n}{2}$, T does not learn from this step any information on the private graphs of P_1 and P_2.

In Step 12, P_1 gets the vector $\mathbf{w}'$. Owing to the random and secret permutation applied by T, P_1 does not know the pair of edges (e, f) that correspond to each entry. It only knows for each entry in that vector whether $\chi_{e,f} = 0$ or $\chi_{e,f} = 1$. Due to the multiplication by the random multiplier $\rho_{e,f} \in \mathbb{F}_\nu^*$, P_1 does not learn any information on $\xi_{e,f}$ beyond what is implied by $\chi_{e,f}$. Namely, if $\chi_{e,f} = 1$ then P_1 may infer that $\xi_{e,f} \in \{1, 2, 4\}$ but nothing further.

We see that P_1 may infer from $\mathbf{w}'$ the size of E_2^{ind} (which is the number of nonzero entries in $\mathbf{w}'$). Note that the size of E_2^{ind} does not determine the size of E, but it may be used to infer a lower bound on $|E|$ and hence a lower bound on $|E_2|$. If such information leakage is considered sensitive, we may mitigate it by having T adding to the vector $\mathbf{w}$, at random positions, entries of the form $\mathcal{E}_1(0)$ or $\mathcal{E}_1(1)$ at random (where each encryption is computed from fresh). While T can remove those entries from the vector $\hat{\mathbf{w}}$ that it receives back in Step 6, such random noise obfuscates the value of $|E_2^{ind}|$ from P_1. Another possibility is to split this computation between P_1 and P_2. Or of course to combine the two suggestions.

In Step 21, T receives a vector which is encrypted under $\mathcal{F}$. Assuming that $\mathcal{F}$ is a secure cipher, T does not learn any information here either.

In Step 40, we rely upon the security of the algorithms in [35] and [28].

4.7.2 Protocol 12

We analyze the privacy leakage during executing, by examining the messages that each player receives during the algorithm.

The first time that P_1 and P_2 share values happens in Steps 16-17, where the players involve in invoking the BGW protocol to compute shares of the equation in $\xi_{e,f} = (\alpha_e^1 + \alpha_e^2) \cdot (\alpha_f^1 + \alpha_f^2)$. Since the BGW protocol is secure, the players does not learn about each other edges.

In Step 19, T receives from P_1, the encrypted matrix $(\mathbf{r}_{e,f} : (e, f) \in K_2^{ind})$. Relying on the security of the encryption scheme of $\mathcal{F}$ and because T does not holds the private key of this scheme, it will not learn anything from the encrypted elements. In addition, since $|V|$ is publicly known, T might not deduce any new information from the size of the matrix it receives (i.e K_2^{ind}).

Next, in Step 20, the information that T receives when it combines the shares $(s_{e,f}^h : (e, f) \in K_2^{ind}), h \in \{1, 2\}$, is only whether $(e, f) \in E_2^{ind}$, hence it may deduce the number of independent edges in E. That is without knowing them explicitly, nor the structure of the union graph owning the secret permutation π of the rows applied by P_1. If the latter is consider undesired leakage, P_1 can sends a random number of bogus zeros rows that their shares equals that 1. This way, those rows

may blur the exact number of independents edges while keeping the solvability of the new matrix the same as $(\mathbf{r}_{e,f} : (e,f) \in E_2^{ind})$. Note that due the multiplication with $\rho_{e,f}^h$, $h \in \{1,2\}$, T will not be able to distinguish in the case whether just one player holds the independent edges or both.

Finally, the privacy of executing $SOLVABLE$ routine relies upon the security of the algorithms in [35] and [28].

4.8 Computational and communication costs

We analyze below the computational and communication costs of the suggested algorithms. Herein, if f is any operation, we let $[f]$ denote its cost. We use it for the cryptographic operations, $\mathcal{E}, \mathcal{E}^{-1}, \mathcal{E}_1, \mathcal{E}_1^{-1}, \mathcal{F}, \mathcal{F}^{-1}$, for multiplications in $\mathbb{G}$ and $\mathbb{G}_1$, denoted $[Mul]$, and for pairing computations, denoted $[e]$. As for the communication costs, we will denote them (in steps where there are such) by $[n_R, n_M, n_B]$ where n_R is the number of communication rounds, n_M is the number of messages sent, and n_B is the overall size of those messages in bits.

4.8.1 Protocol 11

1. Step 1-2: Negligible costs.
2. Steps 3-5: $\binom{n}{2}[\mathcal{E}]$ for each of P_1 and P_2. Communication costs: $[1, 2, n(n-1)\log\nu]$.
3. Step 7: For each $\{e,f\} \in K_2^{ind}$, T has to perform two multiplications in G (for the two addition operations) and one pairing computation (for the single multiplication). Hence, the total computational cost for T is $N_{K_V} \cdot (2[Mul] + [e])$. (No need to multiply by the random exponent of h, it is not essential.)
4. Steps 8-10: T performs here N_{K_V} exponentiations in G_1 and N_{K_V} operations for computing the parity vector, the overall cost is $N_{K_V}(\log\nu[Mul] + 1)$.
5. Step 12: Communication costs: $[1, 1, N_{K_V}\log\nu]$.
6. Steps 13-20: $N_{K_V}[\mathcal{E}_1^{-1}]$ plus $N_{K_V}[\mathcal{F}]$ for P_1.
7. Step 21: Communication costs: $[1, 1, N_{K_V}\log\mu]$, where μ is the modulus of the $\mathcal{F}$ encryption.
8. Step 22: Negligible costs.
9. Steps 23-39: $[N-3] \cdot N_{K_V} \cdot [\mathcal{F}]$ for T.
10. Step 40: We focus here on the case where the two players invoke $Solvable_2$. The two players invoke that sub-protocol on a matrix of size $k_a = \binom{n}{2} \cdot (n-2) = O(n^3)$ over $k_b = N_{K_V} = O(n^4)$, which costs: $\tilde{O}(n^{12} \cdot ([Mul] + [\mathcal{F}] + [Inv]))$. The communication costs: $[\tilde{O}(n^{0.825}), \tilde{O}(n^{0.825}), \tilde{O}(n^7 \cdot \log(\mu))]$.
11. Steps 41-45: Negligible costs.

4.8.2 Protocol 12

1. Step 1-3: Negligible costs.
2. Steps 6-13: P_1 performs $(N \cdot |K_2^{ind}|) \cdot [\mathcal{F}] = O(n^7 \cdot [\mathcal{F}])$ operations.
3. Step 14-15: P_1 performs $|K_2^{ind}| \cdot [\mathcal{F}] = O(n^4 \cdot [\mathcal{F}])$ operations.
4. Steps 16-17: P_1 and P_2 perform $|K_2^{ind}| = O(n^4)$ operations.
5. Step 19: Communication costs : $[1, 1, O(n^7)]$.
6. Step 20: Communication costs : $[1, 2, O(n^4)]$.
7. Step 21-25: T performs $|K_2^{ind}| = O(n^4)$ operations.
8. Step 26: T and P_1 invoke $Solvable_2$ sub-protocol on a matrix of size $k_b = \binom{n}{2} \cdot (n - 2) = O(n^3)$ over $k_a = |E_2^{ind}| = O(n^2)$, which costs $\tilde{O}(n^8 \cdot ([Mul] + [\mathcal{F}] + [Inv]))$. The communication costs: $[\tilde{O}(n^{0.55}), \tilde{O}(n^{0.55}), \tilde{O}(n^5 \cdot \log(\mu))]$.
9. Steps 27-31: Negligible costs.

5 Further results - Graph coloring and outer-planarity testing

Herein we present further results that benefit from the protocols that were proposed in the previous section. In Section 5.1 we describe an algorithm that tests the 3-colorability of a graph that is shared by two players and that was already found to be planar (using the algorithms of Section 4). Afterwards, in Section 5.2 we show how to test in a privacy preserving manner whether a distributed graph is outer-planar.

5.1 An algorithm for testing 3-colorability of planar graphs

Algorithm 15 is based on Grötzsch's theorem [45] which states that every planar triangle–free graph is 3-colorable[6]. Note that the triangle-free property alone, without the planarity property, does not assure that the graph is 3-colorable as is demonstrated by the so-called Grötzsch's Graph, see Figure 3.

Assume the same setting as was considered in Section 4, where two players P_1 and P_2 share a graph, and let us assume that the two players had already found that the union graph is planar. They wish to proceed and check the 3-colorability of that graph, while still preserving privacy. To that end they invoke the BGW protocol which was described in Section 3.5 and rely on Corollary 5.2 that derives from Lemma 5.1.

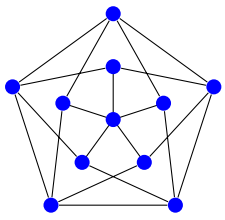
Lemma 5.1. *Let $G = (V, E)$ be a graph and let A be its adjacency matrix. Then for any $k \geq 1$ and vertices $u, v \in V$, the entry $A^k(u, v)$ equals the number of paths from u to v.*

Proof. The proof is by induction on k. The case of $k = 1$ is trivial. Let us assume that the lemma holds for k and we will prove it for $k+1$. Consider any path of length $k+1$ from u to v. Then there must be a vertex w that is adjacent to v. By induction, the number of paths from u to w of length k is given in $A^k(u, w)$. Now, the value of each entry $A(w, v)$ indicates the existence of the edge (w, v). Thus, $A^k(u, w)A(w, v)$ gives the number of paths of length $k + 1$ from u to v, in which the one before the last vertex is w. By summing over all possible w, we get that $A^{k+1}(u, v)$ is the overall number of paths of length $k + 1$ from u to v. As that term is the (u, v) entry in $A^k \cdot A$, the claim follows. □

Corollary 5.2. *A graph is triangle-free iff all the diagonal entries in A^3 are zero.*

Proof. 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, the corollary follows from Lemma 5.1. □

Figure 3: Grötzsch's graph. An example of triangle-free graph which is not planar and has chromatic number: 4.



[6]Triangle-free graph is a graph that does not contain any cycle of length 3.

Let $G_h = (V, E_h)$ be the private graph of player P_h, where E_h can be described by an adjacency matrix $A_h : V^2 \to \{0, 1\}$, $h \in \{1, 2\}$. The entries in A_h that corresponds to a pair of vertices (u, v) is 1 if and only if E_h has an edge between u and v. Those private graphs G_h induce the unified matrix $G = (V, E)$ for $E = E_1 \cup E_2$.

Assume z is any integer that the players P_h share between themselves, then $[z]$ denotes the set of all h shares that sum to z and $[z]_h$ denotes P_h's share, for $h \in \{0, 1\}$.

Note that because the graph is undirected, the adjacency matrix is symmetric i.e, $A_h = A_h^T$. Thus, we can reduce the number of computation only for the elements that are above the main diagonal, i.e. for the entries' indexes $1 \leq i \leq j \leq n$.

The algorithm begins when the players choosing a field $\mathbb{F}_p$ to be larger than the maximum value that appear in $[A^3]$ which is n^2, and greater than the sum of the diagonal entries which is equal to the maximal value of triangles in a a planar graphs, that according to [23] is $\leq 3n - 8$. This restriction is necessary in order to enable us to use BGW for the computation that we explain subsequently. In Step 2 the players create and share with each other their own adjacency matrix entries. Then, in Step 3 they compute the shared value of matrix entry $[a_{i,j}]_h$ where $a_{i,j} = (\alpha^1_{e_{i,j}} + \alpha^2_{e_{i,j}} - \alpha^1_{e_{i,j}} \cdot \alpha^2_{e_{i,j}})$ and $1 \leq i \leq j \leq n$ using BGW over the field $\mathbb{F}_p$ (see Eq. 10).

Next, via the BGW protocol the players compute their shared values of the entries of $[A^3]_h$, by performing the computation of $[b_{i,j}]_h = \sum_{k=1}^{k=n} [a_{ik}]_h [a_{kj}]_h$ for all $1 \leq i \leq j \leq n$, then they continue and compute $[c_{i,i}]_h = \sum_{k=1}^{k=n} [b_{ik}]_h [a_{kj}]_h$ for all $1 \leq i \leq n$, where the values are over the field $\mathbb{F}_p$ (Steps 4-5).

Then, in Steps 6-7 they compute the sum of those shared values in the main diagonal, $[s]_h = \sum_{i=1}^{i=n} [c_{i,i}]_h$, multiply it with a random non-zero term $q \in \mathbb{F}_p^*$ to prevent from T to reveal any information beside whether the trace is zero or not. Then, they send the the value, $q \cdot [s]_h$, to T. Finally, in Steps 8-9 T combines the shared values and conclude that the graph is triangle free iff the latter sum equals to zero.

Complexity. The bulk of the computational and communication costs of Protocol 15 is due to the BGW sub-protocol that happens in Steps 2-5. In that sub-protocol, the two players need to compute at most $\binom{n}{2} + n^2(n + 1)$ multiplication gates and $(n - 1)[n(n + 2) + 1]$ addition gates.

Privacy. Derived from the information–theoretic security of the BGW protocol there is no a privacy concern. The only value that is revealed during the protocol is the value in the final step as excepted.

Protocol 15 Privacy-preserving testing of the triangle-freeness of a distributed graph

Input: Each P_h holds its adjacency matrix A_h of its own graph $G_h = (V, E_h)$, $h \in \{1, 2\}$.
Output: A bit that indicates whether $G = (V, E)$, $E = E_1 \cup E_2$, is triangle-free.

1: P_1 and P_2 choose a finite field $\mathbb{F}_p$ where $p > n^2$.
2: Each P_h creates two shares in each entry in its own adjacency matrix A_h and sends to P_{3-h} its shares, $h = 1, 2$.
3: The two players invoke the BGW protocol to compute shares in each of the entries in the adjacency matrix $A = A_1 \bigvee A_2$.
4: The two players invoke the BGW protocol to compute shares in each of the entries in A^2.
5: The two players invoke the BGW protocol to compute shares in each of the entries on the diagonal of A^3.
6: P_1 and P_2 agree on a random nonzero term $q \in \mathbb{F}_p^*$.
7: P_h computes the sum $[s]_h$ of its own shares in the diagonal entries of A^3 and sends to T the value $q \cdot s_h$, $h = 1, 2$.
8: T recovers from $q[s]_1$ and $q[s]_2$ the value qs, where s is the trace of A^3.
9: T outputs "**The graph is triangle-free**" if $qs \neq 0$ and "**The graph has triangles**" otherwise.

To summarize, by applying first the protocols of Section 4 and then, if the unified graph G was found to be planar, proceeding to apply Protocol 15, it is possible to test, in a privacy-preserving manner and in polynomial time, whether G is 3-colorable or not. We mention that testing 3-colorability of general graphs is an NP-complete problem [13, 6].

5.2 Algorithm for testing whether a graph is outer-planar

An *outer-planar graph* is a graph that has a planar drawing for which all the vertices belong to the outer face of the drawing. Leveraging the protocols that we presented in Sections 4.4, 4.5, we will show how to build a privacy preserving algorithm that can test whether a graph is an outer-planar. In [16] it states that a graph G is an outer-planar iff the graph formed from G by adding a new vertex and edges connecting it to all the other vertices is a planar graph. Thus, we can test the outer-planarity property via a reduction to the problem of *planarity testing* that we already solved, as we proceed to describe.

Protocol 16 begins with P_1 and P_2 who agree on an extra vertex $v_{n+1} \notin V$ (Step 1). Then, in Step 2 P_1 adds an edge between all the vertices in V and v_{n+1}. Next, as we already explained above, in Steps 3-8 the players execute Protocol 11[7] to test if the modified graph is planar and output the result corresponding to that answer.

It is easy to see that the computational complexity and the correctness of this algorithm is similar to Protocol 11.

Protocol 16 Privacy preserving outer-planarity testing

1: P_1 and P_2 agree on an extra vertex $v_{n+1} \notin V$, and map it to a point outside of the unit circle.
2: P_1 computes $E_1 \leftarrow E_1 \cup \{(v_1, v_{n+1}), \ldots, (v_n, v_{n+1})\}$
3: T, P_1 and P_2 invoke Protocol 11 on their inputs.
4: **if** Protocol 11 returns **"The union graph is planar" then**
5: Output "The union graph is outer-planar".
6: **else**
7: Output "The union graph is not outer-planar".
8: **end if**

[7] Protocol 12 is also appropriate, we use Protocol 12 for the sake of convenience.

6 Conclusion

In this thesis we introduced the problem of privacy-preserving planarity testing of distributed graph. Our main contribution in this thesis are two algorithms for solving that problem. The two algorithms follow two different approaches that are based on the Hanani–Tutte Theorem. In addition, we showed how our algorithms can be leveraged in order to reduce the complexity of computing various privacy preserving algorithms that are based on the fact that the graph is planar, such as testing 3-colorability of planar graphs, or testing whether a given graph is outer-planar. The proposed algorithms are privacy-preserving in the sense that they protect the private edge sets of each of the players.

The work is based on theoretical cryptographic primitives such as homomorphic encryption, oblivious transfer, Yao's garbled circuits and Yao's millionaires' protocol, along with more complicated protocols like oblivious Gaussian elimination and rank computation of an encrypted matrix.

This study raises the following two problems for future research:

- To devise an improvement in terms of computation complexity of our planarity testing algorithms in order to render them efficient for larger graphs. This can be done by improving our approach using Hanani–Tutte Theorem with more efficient ways to test the solvability of the HT system equations, or to design a privacy preserving version of a planarity testing algorithm that is based on another approach.

- To devise privacy-preserving algorithms for solving graph problems, which are more efficient to planar graphs than they are for general graph. Examples of such problems are: the subgraph isomorphism problem or the maximal clique problem.

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. *Journal of 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] Richard Beigel and David Eppstein. 3-coloring in time $o(1.3289^n)$. *Journal of Algorithms*, 54(2):168–204, 2005.

[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] Richard E. Blahut. *Algebraic codes for data transmission*. Cambridge University Press, 2003.

[9] 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.

[10] Allan Borodin, Joachim von zur Gathen, and John E. Hopcroft. Fast parallel matrix and GCD computations. In *23rd Annual Symposium on Foundations of Computer Science*, pages 65–71, 1982.

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

[12] 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.

[13] David P. Dailey. Uniqueness of colorability and colorability of planar 4-regular graphs are np-complete. *Discrete Mathematics*, 30(3):289–293, 1980.

[14] Jean-Louis Dornstetter. On the equivalence between berlekamp's and euclid's algorithms. *IEEE Transactions on Information Theory*, 33(3):428–431, 1987.

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

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

[17] Matthew K. Franklin and Payman Mohassel. Efficient and secure evaluation of multivariate polynomials and applications. In *Applied Cryptography and Network Security, 8th International Conference (ACNS)*, pages 236–254, 2010.

[18] 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.

[19] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing*, pages 218–229, 1987.

[20] 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.

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

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

[23] Seifollah L. Hakimi and Edward F. Schmeichel. On the number of cycles of length k in a maximal planar graph. *Journal of Graph Theory*, 3(1):69–86, 1979.

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

[25] Tamás Horváth, Jan Ramon, and Stefan Wrobel. Frequent subgraph mining in outerplanar graphs. *Data Mining and Knowledge Discovery*, 21(3):472–508, 2010.

[26] Erich Kaltofen and David Saunders. On wiedemann's method of solving sparse linear systems. In *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes, 9th International Symposium*, pages 29–38, 1991.

[27] 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.

[28] Eike Kiltz, Payman Mohassel, Enav Weinreb, and Matthew K. Franklin. Secure linear algebra using linearly recurrent sequences. In *Theory of Cryptography, 4th Theory of Cryptography Conference (TCC)*, pages 291–310, 2007.

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

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

[31] Josef Leydold and Peter F. Stadler. Minimal cycle bases of outerplanar graphs. *The Electronic Journal of Combinatorics*, 5(16):1–14, 1998.

[32] 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.

[33] Yehuda Lindell and Benny Pinkas. A proof of security of yao's protocol for two-party computation. *Journal of Cryptology*, 22(2):161–188, 2009.

[34] Dana Moshkovitz. An alternative proof of the schwartz-zippel lemma. *Electronic Colloquium on Computational Complexity (ECCC)*, 17:96, 2010.

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

[36] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology - EUROCRYPT, International Conference on the Theory and Application of Cryptographic Techniques*, pages 223–238, 1999.

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

[38] Maurizio Patrignani. Handbook on graph drawing and visualization. chapter Planarity Testing and Embedding, pages 1–42. CRC Press, 2013.

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

[40] Jacob T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM (JACM)*, 27(4):701–717, 1980.

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

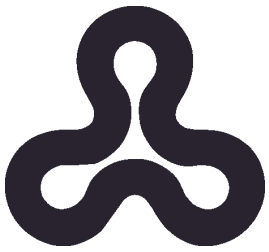
[42] Joachim von zur Gathen and Jürgen Gerhard. *Modern computer algebra (2. edition)*. Cambridge University Press, 2003.

[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 (TALG)*, 7(4):41:1–41:14, 2011.

[46] Richard Zippel. Probabilistic algorithms for sparse polynomials. In *Symbolic and Algebraic Computation, EUROSAM, An International Symposiumon Symbolic and Algebraic Computation*, pages 216–226, 1979.



האוניברסיטה הפתוחה
המחלקה למתמטיקה ולמדעי המחשב

בדיקה משמרת־פרטיות של מישוריות של גרפים מבוזרים

עבודת תזה זו הוגשה כחלק מהדרישות לקבלת תואר
"מוסמך למדעים" M.Sc. במדעי המחשב
באוניברסיטה הפתוחה
החטיבה למדעי המחשב

על־ידי
גיא ברשפ

העבודה הוכנה בהדרכתו של פרופ' תמיר טסה
אוקטובר 2017

תודות.

הכנת עבודת התיזה הייתה עבורי מטלה ממושכת ומאתגרת, והתאפשרה בזכות תמיכתם של מספר אנשים שברצוני להודות להם. בראש ובראשונה, אני רוצה להביע הערכה כנה למנחה שלי, פרופ' תמיר טסה. תמיר לימד אותי בשקדנות כיצד לכתוב את המאמר האקדמי הראשון שלי. השקעתו וזמינותו המרובה, סייעו לי להשלים את תואר המאסטר במדעי המחשב. אני מודה על הגישה והיכולות המחקריות שספגתי ממנו, הרעיונות ששאבתי ממאמריו הקודמים והכוונתו הסבלנית והבלתי מתפשרת להעמדת תוצר מוגמר באיכות גבוהה, שאני גאה בו מאוד. בנוסף, ברצוני להודות גם למשפחתי אשר האמינו בי לאורך כל הדרך ונתנו לי את המרחב שהייתי נדרש לו במהלך הכנת הפרוייקט התובעני הזה. במיוחד לאבי אשר עודד אותי לעסוק במחקר, אחי הקטן אלון, ואמי על התמיכה וההשקעה הבלתי נגמרת. תודה אחרונה וחביבה מוקדשת לטליה שוורץ, בת זוגתי, אשר ליוותה אותי מתחילת המסע הארוך בכתיבת התיזה, וגילתה הבנה רבה תוך הפגנת רגשות חיבה ואהבה.

תקציר

בעבודה זאת נציג אלגוריתמים משמרי פרטיות לבדיקה של מישוריות של גרפים המבוזרים בין מספר שחקנים. המידע הפרטי של כל שחקן אשר אנו רוצים לשמר במהלך החישוב הוא הקשתות השייכות לגרף שלו, בעוד שצמתי הגרף ידועים לכולם. האלגוריתמים משמרי הפרטיות שיוצגו בעבודה זו מבוססים על משפט Hannani-Tutte המראה שגרף נתון הוא מישורי אם ורק אם מערכת מסוימת של משוואות לינאריות מעל F_2 היא פתירה. האלגוריתמים משתמשים בטכניקות לבדיקת קיום פתרון למערכות משוואות לינאריות מוצפנות מעל F_2. בפרט, אנו עושים שימוש באלגוריתם לחישוב דירוג גאוס של מטריצות מוצפנות ובאלגוריתם לחישוב הדרגה של מטריצה מוצפנת. זו הפעם הראשונה שבעיית הבדיקה משמרת הפרטיות של מישוריות של גרפים מבוזרים נחקרת.

תוכן עניינים