EFFICIENT DYNAMIC TRAITOR TRACING

OMER BERKMAN*, MICHAL PARNAS†, AND JIŘÍ SGALL‡

Abstract. The notion of traitor tracing was introduced by Chor, Fiat and Naor, in order to combat piracy scenarios. Recently, Fiat and Tassa proposed a dynamic traitor tracing scenario, in which the algorithm adapts dynamically according to the responses of the pirate. Let n be the number of users and p the number of traitors.

Our main result is an algorithm which locates p traitors, even if p is unknown, using a watermarking alphabet of size $p+1$ and an optimal number of $\Theta(p^2 + p\log n)$ rounds. This improves the exponential number of rounds achieved by Fiat and Tassa in this case. We also present two algorithms that use a larger alphabet: For an alphabet of size $p+c+1$, $c \geq 1$, an algorithm that uses $O(p^2/c + p\log n)$ rounds. For an alphabet of size $pc+1$, an algorithm that uses $O(p\log_c n)$ rounds.

Our final result is a lower bound of $\Omega(p^2/c + p\log_{c+1} n)$ rounds for any algorithm that uses an alphabet of size $p+c$, assuming that p is not known in advance.

Key words. Analysis of algorithms, asymptotic complexity, cryptography, traitor tracing.

AMS subject classifications. 68Q25, 94A60

1. Introduction. In the electronic world where information is easily copied and retransmitted, the issue of protecting intellectual property becomes a great concern. While owners of such information are interested in selling it, they need to protect themselves. There are two ways to protect such property. One option is to prevent the redistribution. Another option is to devise means to detect misconduct once redistribution has occurred. The second issue is the one addressed by traitor tracing (of course, the existence of such a scheme may also deter piracy from even occurring).

An excellent example is the world of Pay-TV systems. The owners of a Pay-TV system would like to broadcast only to their paying customers, and to be able to add and delete viewers as needed, while ensuring that this process happens in a timely manner. In order to protect the material that is being broadcast, encryption is utilized and keys are changed periodically, in addition to the existence of secure hardware. The goal of the pirate is to enable non-paying persons to view the broadcasts. The pirate, registered as a legitimate subscriber of the system, can achieve this goal by transferring the decryption keys or by rebroadcasting the entire content. In order to make tracing difficult, the pirate can control several subscriptions, and alternate between them. We refer to each such subscription as a *traitor*.

We assume from now on that the pirate operates by rebroadcasting the content. Our algorithms can be easily adapted to deal with the other mode of piracy, where keys are published.

Traitor tracing schemes should be capable of detecting the traitors, so that they can be disconnected from the system, without harming any legitimate user. Furthermore, such schemes should supply legal evidence of the pirate's identity.

Chor et al [3] introduced the notion of traitor tracing, and provided a solution to this problem. They use a static approach, which means that all security measures

*The Academic College of Tel-Aviv-Yaffo, 4 Antokolsky st., Tel-Aviv 64044, Israel. Tel. 972-3-5211866, fax. 972-3-5211871. E-mail omer@mta.ac.il.

†The Academic College of Tel-Aviv-Yaffo, 4 Antokolsky st., Tel-Aviv 64044, Israel. Tel. 972-3-5211864, fax. 972-3-5211871. E-mail michalp@mta.ac.il.

‡Mathematical Institute, AS CR, Žitná 25, CZ-11567 Praha 1, Czech Republic; and Dept. of Applied Mathematics, Faculty of Mathematics and Physics, Charles University, Praha. E-mail sgall@math.cas.cz, http://www.math.cas.cz/~sgall. Partially supported by grant A1019901 of GA AV ČR, and grant 201/97/P038 of GA ČR.

are applied once, at the onset of the protocol. As there is a single opportunity for applying these measures, they must suffice in order to locate the traitors once piracy is observed. If the number of traitors in reality is larger than that assumed by the static scheme, then the algorithm has no way of tracing the traitors, and the traitors may even frame an innocent user. Other related work can be found in [2, 5, 6, 9, 8, 10, 11]. For a comprehensive overview of static schemes and related work see [7].

A basic tool used by many traitor tracing schemes is cryptographic fingerprinting (e.g. [13, 1, 2, 7]), which allows to generate different versions of the same content and send them to subsets of users. This can be done for example by watermarking each version, with no noticeable degradation in quality. A basic assumption of both previous schemes and our schemes is that the pirate cannot remove the watermarks or combine different versions into a new one, see e.g. [4].

In a typical static scheme that uses fingerprinting, the content is divided into segments, and each segment is watermarked. The number of different versions that can be generated from a segment is referred to as the *watermarking alphabet size*. It is possible, of course, to use an alphabet whose size equals the number of subscribers and locate the traitors immediately. However, this requires an enormous bandwidth. Therefore, a smaller alphabet is used, so that the same version of a given segment is sent to many subscribers.

1.1. Dynamic Traitor Tracing Schemes. Recently, Fiat and Tassa [7], introduced *dynamic* traitor tracing schemes. Their schemes adapt themselves throughout the algorithm in order to force the pirate to reveal more and more information. Eventually, the algorithm locates all traitors (or stops piracy). An additional nice feature of the dynamic schemes of [7], is that there is no need for an a priori bound on the possible number of traitors, as the algorithms adapt themselves when a new traitor is discovered.

The static schemes described above are modified by [7] to fit the dynamic setting as follows: In each round, the algorithm divides the set of users into disjoint subsets, where the number of subsets is bounded by the watermarking alphabet size. Then, different versions of the content are transmitted by the system to these subsets of users, one version per subset. Whenever the pirate broadcasts one of these versions, it is evidence that the corresponding subset contains a traitor. When this happens, the algorithm changes the allocation of versions to the users, thus starting a new *round*. (Otherwise, the pirate could continue to broadcast the same version, with no further knowledge gained by the algorithm.) Eventually, if the pirate keeps rebroadcasting, the information gathered allows the algorithm to locate and disconnect all traitors.

A *segment* in the dynamic setting is defined as the part of the content transmitted to the users within the duration of a round. We note that sending a distinct version securely to each subset can be done by encryption. See Section 2.1 in [7] for a discussion of the bandwidth overhead required for sending the encrypted versions.

We are interested in the following two complexity measures of such a dynamic traitor tracing scheme:

- the *number* of different versions used (i.e. the size of the watermarking alphabet), which is proportional to the bandwidth requirements, and
- the *time* or *number of rounds* needed, which is proportional to the number of segments used.

We ignore the computation performed by the traitor tracing algorithm, but we note that all proposed algorithms, both previous and ours, are efficient.

Let n be the number of all subscribers and p the number of traitors. It is shown in [7] that:

(A) Any deterministic algorithm must use at least $p+1$ versions in order to locate even a single traitor, regardless of the number of rounds, and even if p is known in advance.

(B) Using $2p+1$ versions, it is possible to locate all p traitors in $O(p \log n)$ rounds.

(C) Using $p+1$ versions, it is possible to locate all p traitors in $O(3^p p \log n)$ rounds.

1.2. Our Results. We present a family of deterministic algorithms, which locate all p traitors in a polynomial number of rounds. The exact bounds depend on the size of the watermarking alphabet. Specifically, we show:

(1) Using $p+1$ versions, it is possible to locate all p traitors in an optimal number of $\Theta(p^2 + p \log n)$ rounds (see lower bound (4) below). This improves the exponential bound in (C), thus solving an open problem raised in [7]. Note that the algorithm in (B) runs in a polynomial number of rounds, but uses twice as many versions as our algorithm. By (A), the minimal number of versions needed is $p+1$. Thus, our algorithm achieves the optimal time for the minimal alphabet size.

(2) Using $p+c+1$ versions, for any $1 \leq c \leq p$, it is possible to locate p traitors in $O(p^2/c + p \log n)$ rounds. For example, if $c = \Omega(p)$, the time is $O(p \log n)$.

(3) Using $pc+1$ versions, for any $c \geq 2$, it is possible to locate p traitors in $O(p \log_c n)$ rounds. For example, if $c = n^\epsilon$ for a constant $\epsilon > 0$, the time is $O(p)$.

All our algorithms, similarly as algorithms (B) and (C) of [7], apply even if the number of traitors p is not known in advance. This means that, for example, if the algorithm is allowed to use $p+1$ versions, then the algorithm uses only $t+1$ versions when only $t \leq p$ traitors are known to exist. As noted before, the computation performed by the algorithms between consecutive rounds is efficient.

In addition, we provide the following lower bound for the time needed to locate p traitors:

(4) The time needed to locate all p traitors, when p is not known in advance and $p+c$ versions are used, is $\Omega(p^2/c + p \log_{c+1} n)$.

This bound is tight when c is a constant (result (2) above) and when $c \geq p^{1+\varepsilon}$, for a constant $\varepsilon > 0$ (result (3) above). Otherwise, the gap is very small: only a factor of $\log c$. It remains an open problem to close this gap.

Organization: The remainder of this paper is organized as follows. In Section 2 we describe the problem and the general strategy that will be used by our algorithms. In Section 3 we describe an algorithm that uses $p+1$ versions and $O(p^3 \log n)$ rounds. This algorithm is used as a building block by later algorithms. The optimal algorithm that uses $p+1$ versions is presented in Section 4, thus establishing (1). In Section 5 we present two algorithms that use more than the minimal number of versions required and prove bounds (2) and (3). In Section 6 we present another algorithm that uses $p+1$ versions and runs in time $O(p^3 \log n)$. While this is not optimal, it is interesting that the same bound as that given in Section 3 can be obtained using a different approach. Finally, in Section 7 we prove lower bound (4). We conclude with a few open problems in Section 8.

2. Preliminaries.

2.1. The Model. We now describe a combinatorial game which is an abstraction of the dynamic scenario described in the introduction and serves as a model for the rest of the paper.

Denote by U the set of users, and let $n = |U|$. Let $T \subseteq U$ be the set of traitors controlled by the pirate and let $p = |T|$.

In each *round*, the algorithm assigns to each user a version of the current segment from a set of versions C. If we give version $c \in C$ to a subset of users, then the subset is said to be *colored* by c. Then the pirate broadcasts one of the colors as an *answer*. If the pirate broadcasts color c, this implies that one of the users colored by c is a traitor. A traitor is *located* if only a single user is colored by c, and c is given as an answer. Since the number of traitors is not known in advance, this is the only way to locate a traitor. A traitor that has been located is *disconnected*, i.e., removed from U.

We assume that the pirate always gives an answer, as long as he has active traitors. The pirate loses the game when he does not answer (meaning that he goes out of business, because the algorithm located all traitors). We measure the maximal number of colors used in any single round and the number of rounds needed in the worst case. The goal of the algorithm is to minimize these numbers in the worst case; more precisely to achieve some given tradeoff between the two measures.

In fact the pirate does not have to decide in advance who the p traitors are (as we implicitly assumed above). Instead, the pirate can choose the traitors in an adaptive manner throughout the game. However, once he decides to corrupt a user, he is committed to this decision until the end of the game. (This is similar to the general situation in the area of on-line algorithms and other adversary arguments. Since the algorithms are deterministic, the adversary can simulate the algorithm in advance, and thus committing to the set T in advance does not pose any restriction.)

2.2. A Simplified Model. The following lemma shows that locating a single traitor is as hard as locating all p traitors, in the worst case. Specifically, we show that this seemingly easier problem of locating only one traitor saves at most $p-1$ rounds. This may be somewhat counterintuitive at first, and so we try to give some intuition.

Let us call a pirate's strategy *reasonable* if it never answers by a color given to a single user, unless he has no choice (when each one of the traitors received a unique color given only to him). For reasonable strategies, the statement holds trivially, since the algorithm can locate a traitor only when all traitors have a unique color. Then it is easy to locate the remaining traitors in additional $p-1$ rounds, as claimed.

It remains to prove that reasonable strategies result in the maximal number of rounds, for a given fixed number of colors. If the goal of the algorithm is to locate one traitor, this is trivial, since it brings no advantage for the pirate to lose the game before he must. But, when the goal of the algorithm is to locate all p traitors, then it is not clear in advance that a reasonable strategy is the pirate's best resort. It may be better in this case for the pirate to reveal one traitor, and keep a bigger uncertainty about the location of the remaining traitors. We thus have to prove that the statement is true for any strategy, not necessarily reasonable strategies.

LEMMA 2.1. *There exists an algorithm that locates one of the p traitors in m rounds if and only if there exists an algorithm that locates all p traitors in $m+p-1$ rounds. The bound on the number of colors is the same for both algorithms.*

Proof. If we have an algorithm that locates all traitors, it must eventually get p answers that allow it to locate all traitors. If we stop after the first traitor is located, we have an algorithm that locates one traitor, takes at most m rounds instead of $m+p-1$, and does not use more colors.

Now suppose that we have an algorithm A that locates a single traitor in at most

m rounds. We devise an algorithm B that uses A as a black box in order to locate all traitors. To do this, algorithm B runs algorithm A and passes to A the pirate's answers with the following exception: whenever a color c that was given by A to a single user is given as an answer by the pirate, algorithm B disconnects that user (which is clearly a traitor), but **does not** pass the pirate's answer to A. Thus, as far as A is concerned, it still waits for an answer from the pirate, and the user that was colored by c is still a part of the user set U. However, since this user has been disconnected, the pirate cannot give the color c that was given only to this user as an answer in the next round.

As far as B is concerned a round has occurred, and we charge this round in which a traitor was located to B. All other rounds are charged to A, including the final round in which a traitor is located. Algorithm B stops when all traitors are located, that is, when no more answers are received from the pirate. Clearly, the number of colors used by B is the same as the number of colors used by A.

To finish the proof we show that B performs at most $m+p-1$ rounds, assuming that A performs at most m rounds. Indeed, we charged $p-1$ rounds to B (the rounds in which the first $p-1$ traitors were located). Also, the sequence of answers that are passed by B to A is valid, and thus A locates a single traitor (the last traitor located) in at most m rounds. Thus, the total number of rounds performed by B is at most $m+p-1$ as claimed. $\square$

We will assume from now on that any algorithm stops after locating one traitor. Thus, without mentioning it explicitly in the algorithm, we assume that the pirate never broadcasts a color given to a single user (otherwise, the algorithm stops immediately). By Lemma 2.1 the change in the number of rounds is at most $p-1$, which is insignificant compared to the bounds that we show. In fact, locating all p traitors requires $\Omega(p)$ rounds no matter how many colors are used.

In practice it may be desired to reduce the number of colors used after a traitor is located. It is easy to modify our algorithms to achieve this as well.

2.3. Graph Notation. The algorithms that we present partition the set of all users into disjoint subsets, and give all users in the same subset a common color. We represent the current state of the algorithm by an undirected graph $G=(V,E)$. Each vertex of G represents a subset of users, and each user belongs to exactly one vertex. An edge (X,Y) means that the subset $X\cup Y$ contains a traitor, i.e., in some previous round, the answer was the color of $X\cup Y$, or (less frequently) the color of some subset of $X\cup Y$. A special vertex I represents the subset of "innocent users" (i.e., the subset of users not known at the present stage to contain a traitor). A vertex is called a *singleton* if it contains exactly one user.

2.4. A Basic Algorithm. Two of the schemes presented in [7] are based on the following basic algorithm, which is formulated here using our graph notation. In this algorithm the graph consists of $2t+1$ vertices one of which is the special vertex I, and t disjoint edges none of which is adjacent to I. Recall that each edge implies that at least one of its endpoints contains a traitor. Since the graph has t disjoint edges, there exist at least t traitors. See Figure 2.1.

Basic Algorithm: Start with a graph $G=(V,E)$ with $I=U$, $V=\{I\}$, $E=\emptyset$, and $t=0$. Repeat forever:

(1) Find a vertex X that contains a traitor.

(2) If $X=I$: Split I into two new vertices of equal size (or differing in size by 1 if $|X|$ is odd), and connect them by an edge. Set $I=\emptyset$ and $t=t+1$.

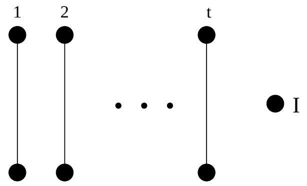

FIG. 2.1. *An example of the graph G used by the Basic algorithm.*

(3) Otherwise: Let Y be the vertex that is connected to X by an edge. Set $I = I \cup Y$, split X into two vertices of (almost) equal size, and connect them by an edge.

The first algorithm in [7] uses $2t+1$ colors, giving each vertex a distinct color. Thus, in each round the pirate has to broadcast the color of some vertex X, and all p traitors are located in $O(p \log n)$ rounds. The second algorithm in [7] uses only $t+1$ colors, but Step (1) requires an exponential time in t, and thus the algorithm is impractical.

2.5. Generalized Graph Notation. In order to decrease the number of colors needed, our algorithms use graphs that generalize the above structure. Instead of the t pairs of connected vertices, we have disjoint cliques, and possibly some additional edges. More formally, we define the following graphs:

DEFINITION 2.2. *Let $t \geq k \geq 0$ be two integers. A graph $G = (V, E)$ is a* (t, k)-graph *if:*

(1) *G contains $t + k + 1$ vertices, one of which is the special vertex I. The vertices are subsets of U, all of them except possibly I are non-empty, and every user belongs to exactly one vertex.*

(2) *For any edge $(X, Y) \in E$, the subset $X \cup Y$ contains a traitor.*

(3) *The vertices in $V \setminus \{I\}$ are partitioned into k disjoint cliques $\{Q_1, \ldots, Q_k\}$, where $|Q_i| = t_i \geq 2$ for each $1 \leq i \leq k$.*

Note that a (t, k)- graph may contain additional edges between the cliques. Further constraints on the structure of the graph will be described in each one of the algorithms. The following lemma is easy to verify, if we remember that two vertices are connected by an edge if at least one of them contains a traitor.

LEMMA 2.3. *Let G be a (t, k)-graph. A clique Q_i of t_i vertices contains at least $t_i - 1$ traitors, and the graph G contains at least t traitors. The number of vertices of a (t, k)-graph is at most $2t + 1$.*

3. The Clique Algorithm. In this section we describe an algorithm that uses $p+1$ colors. The algorithm runs in $O(p^3 \log n)$ rounds and is not optimal. In the next section this algorithm is used to derive an algorithm that uses $(p+1)$ colors, and runs in an optimal number of $\Theta(p^2 + p \log n)$ rounds.

We start with an overview of the Clique algorithm. By Lemma 2.3, a (t, k)-graph is known to contain at least t traitors. Therefore, we are allowed to use $t+1$ colors. Since a (t, k)-graph can contain up to $2t + 1$ vertices, we may not have enough colors to give each vertex a distinct color. Thus, our algorithm pairs the k cliques, and forces the pirate in each round to either transmit a color given to only one vertex, or to disclose an edge connecting a pair of cliques. If a color given to only one vertex is transmitted, then we split this vertex into two vertices (producing a new clique of size two), and thus advance towards locating at least one of the traitors. Disclosed edges, on the other hand, are added to the graph, in order to merge eventually each pair of cliques into one larger clique (containing all but one of the vertices of the two cliques). This process is repeated, until the graph contains exactly one clique of size $t+1$. Then, we can give each one of the vertices a distinct color and force the pirate to transmit a color given to only one vertex (allowing us again to split this vertex).

3.1. Data Structures and Invariants.

The graph. We maintain a (t,k)-graph $G = (V, E)$ with a special vertex I and cliques Q_i of size t_i.

Zones and blocks. The vertices of the graph G are partitioned into two zones Z_1, Z_2. The vertices in Zone Z_1 are partitioned into blocks. There are no edges connecting vertices from different blocks or zones. The zones and the blocks are defined as follows:

$\mathbf{Z_1}$ The vertices in this zone are partitioned into blocks. Each block is a graph induced by the vertices of two cliques Q_i and Q_j, of sizes t_i and t_j respectively. The block does not contain a clique of size $t_i + t_j - 1$. Equivalently, this means that there are four distinct vertices $X_1, X_2 \in Q_i$ and $Y_1, Y_2 \in Q_j$, such that $(X_1, Y_1), (X_2, Y_2) \notin E$.

$\mathbf{Z_2}$ This zone contains the special vertex I. In addition it may contain one clique Q_i of size t_i. The vertex I and the clique Q_i do not form a clique of size $t_i + 1$. Equivalently, this means that there exists a vertex $X \in Q_i$, such that $(X, I) \notin E$.

See Figure 3.1 for an example of the data structures used by the Clique algorithm.

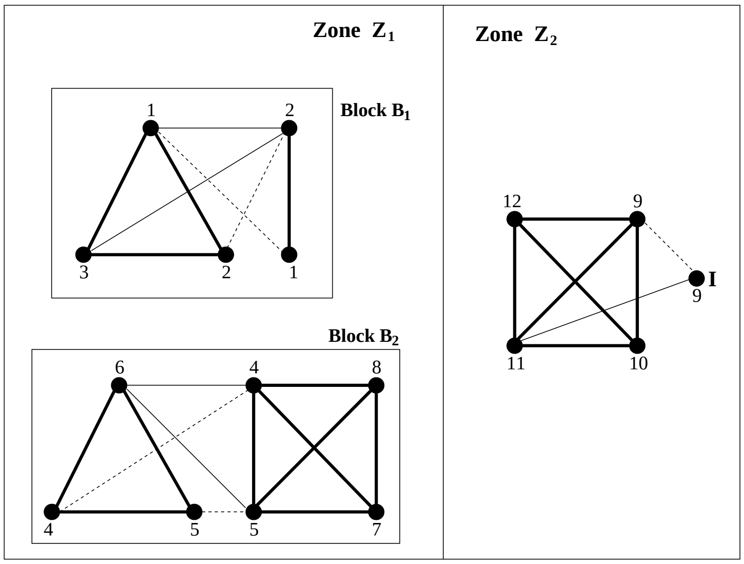

FIG. 3.1. *An example of the data structures used by the Clique algorithm. This is a (t,k)-graph, where $t = 11$ and $k = 5$. The bold lines represent clique edges, and the thin lines represent edges between cliques, or between a clique and I. This is also an example of the distribution of colors in Phase 1 of the Clique algorithm. The labels denote the colors given to the vertices. The dashed lines denote the pairs of non-adjacent vertices that received the same color.*

3.2. Description of the Algorithm. Start with a $(0,0)$-graph $G = (V, E)$ with $I = U$, $V = \{I\}$, $E = \emptyset$, and $t = 0$. Repeat the following two phases:

Phase 1: Distributing the colors to the vertices of G

Allocate $t_i - 1$ colors to each clique Q_i, for a total of t colors. The last color is allocated to I. We now describe how to distribute these allocated colors in each zone and block of G.

(1) Color each block in Zone Z_1 separately, using the colors allocated to the pair of cliques in it: Let B be a block in Z_1 and let Q_i and Q_j be the two cliques in B. By the definition of the blocks in Zone Z_1, there exist four distinct vertices $X_1, X_2 \in Q_i$ and $Y_1, Y_2 \in Q_j$, such that $(X_1, Y_1), (X_2, Y_2) \notin E$. Color $X_1 \cup Y_1$ with one color and $X_2 \cup Y_2$ with another color. Color the remaining $t_i + t_j - 4$ vertices of Q_i, Q_j using the remaining $t_i + t_j - 4$ colors allocated to these two cliques, one color per vertex.

(2) If Z_2 contains only the vertex I, then color I using the color allocated to it. Otherwise, Z_2 contains also a clique Q_i and there exists a vertex $X \in Q_i$, such that $(X, I) \notin E$. Use one color for $X \cup I$, and color the remaining $t_i - 1$ vertices of Q_i using $t_i - 1$ colors, one color per vertex. (If I is empty, all vertices of Q_i are colored with distinct colors.) Therefore, t_i colors are used, which is the total number of colors allocated to Q_i and I.

See Figure 3.1 for an example of the distribution of colors.

Phase 2: Reorganizing the graph following the pirate's response

(1) If the pirate broadcasts a color given to only one vertex X: Remove X and the edges incident with it from G. Split X into two new vertices X_1, X_2 of equal size (or differing in size by one, if $|X|$ is odd), and add the edge (X_1, X_2).

- **(1.1)** If $X = I$: Define a new clique Q consisting of the two vertices X_1, X_2. Set $I = \emptyset$, $t = t + 1$, and $k = k + 1$. Incorporate the clique Q into the zones (see Step (3)).
- **(1.2)** Otherwise: Let Q_i be the clique to which X belongs. Set $Q_i = Q_i \backslash \{X\}$.
  - **(1.2.1)** If $|Q_i| = 1$ then remove all edges incident with the vertex remaining in Q_i, remove this vertex from Q_i and add it to I. Place X_1 and X_2 into Q_i. The clique Q_i now consists of the two vertices X_1, X_2.
  - **(1.2.2)** Otherwise, Q_i is still a legal clique. In this case the two sets X_1, X_2 will form a new clique Q. Set $k = k + 1$ and incorporate the new clique Q into the zones (see Step (3)).

(2) If the pirate broadcasts a color given to a pair of vertices X, Y: Add the edge (X, Y) to G, and:

- **(2.1)** If this edge connects vertices belonging to a pair of cliques Q_i, Q_j in some block B, and after adding this edge, the block B contains a large clique Q of size $t_i + t_j - 1$: Let Z be the remaining vertex in B that does not belong to this large clique (i.e. $Z \notin Q$). Remove Z and all edges incident with it from its clique, and add Z to I. Furthermore, set $k = k - 1$, remove the cliques Q_i, Q_j and their block B. Incorporate the new clique Q into the zones (see Step (3)). See Figure 3.2 for an example.
- **(2.2)** If this edge connects I and a vertex of the clique Q_i in Zone Z_2, and after adding this edge, Q_i and I form a clique on $t_i + 1$ vertices: Add I as a vertex to Q_i and create a new special vertex $I = \emptyset$. Set $t = t + 1$ (since t_i was increased by 1).

(3) If a new clique Q was created during one of the above steps, reorganize the zones as follows to incorporate Q: If Zone Z_2 contains only I, place Q in Zone Z_2.

Otherwise, Zone Z_2 contains a clique R in addition to I. In this case, remove all edges incident with I, remove R from Z_2, pair it with the new clique Q, and create in Zone Z_1 a new block containing the cliques Q and R.

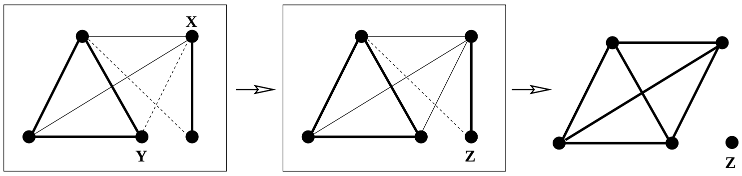

FIG. 3.2. *An example of merging a pair of cliques in Step (2.1) of Phase 2 of the Clique algorithm. On the left: A block of Zone Z_1, in which X and Y received the same color. In the middle: The block after the pirate answered with the joint color of X and Y, and the edge (X, Y) was added. Finally, on the right: The two cliques were merged into one big clique, and all edges incident with Z were removed.*

Remark: In Step (3) of Phase 2, we removed all edges incident with the vertex I. Removing these edges makes the structure of G simpler, but in practice we may want to leave them in the graph, since they provide more information. Since the number of these edges is $O(p)$, it turns out that the analysis of the algorithm (Theorem 3.2 below) is valid whether these edges are removed or not.

3.3. Correctness and Efficiency.

LEMMA 3.1. *The invariants of the algorithm are preserved at all times.*

Proof.

The graph. An edge is added only if we got an answer from the pirate corresponding to that edge, and thus one of its endpoints must contain a traitor. Whenever the structure of the cliques is changed in Step (1) or Step (2) of Phase 2, we verify that every clique has at least two vertices and that the values of t and k are updated properly, so that the new graph is a (t, k)-graph for the new values. Step (3) of Phase 2 only removes edges incident with I, which maintains the structure of a (t, k)-graph.

Zone constraints. If adding a new edge would violate the zone invariants, i.e., it would create a big clique, then we reorganize the cliques in Step (2) of Phase 2. In Step (1) and Step (3) of Phase 2, no edges are added between cliques. Reorganizing the cliques does not violate the zone constraints. If two cliques are placed in the same block, then there are no edges between them (since they were in different blocks or zones before). Similarly if a clique is placed in Zone Z_2, then there are no edges between this clique and I.

Edges between different zones and blocks. Whenever the cliques are reorganized or a new clique is created, all the edges that would connect vertices from different zones or blocks are removed. □

In order to bound the number of rounds used by the algorithm, we first need to bound the number of split operations performed throughout the algorithm. Define a *split* to be any round in which a vertex of G is split (i.e., a round in which Step (1) in Phase 2 is performed).

LEMMA 3.2. *The total number of splits is $O(p^2 \log n)$.*

Proof. Whenever the set I is split, t increases. During the algorithm t never decreases and its value is at most p. Thus, the set I can be split at most p times.

For every traitor i, define x_i as follows. If traitor i is in I, then $x_i = 0$. Otherwise, traitor i is in a vertex $X \neq I$ and $x_i = \lceil \log |X| \rceil$. Each x_i is an integer between 0 and $\lceil \log n \rceil$. Between any two consecutive splits of I, x_i does not increase. It may be set to zero (if the corresponding vertex is added to I), decrease by one or two (if the corresponding vertex is split), or remain the same (otherwise). At each split of a vertex $X \neq I$, we know that X contains a traitor. It follows that when a spilt occurs, then for some traitor i, x_i is decreased by at least one. Thus, there are at most $p\lceil \log n \rceil$ splits between any two consecutive splits of I, and the total number of splits is $O(p^2 \log n)$. ∎

THEOREM 3.3. *The Clique algorithm locates a traitor in $O(p^3 \log n)$ rounds, using $p+1$ colors. Hence, all p traitors can be located in $O(p^3 \log n)$ rounds, using $p+1$ colors.*

Proof. By Lemma 3.1, the algorithm maintains a (t,k)-graph, and thus Lemma 2.3 implies that $t \leq p$. It is immediate from the allocation of colors in Phase 1, that the algorithm uses $t+1 \leq p+1$ colors.

We now analyze the number of rounds needed. In each round some progress is made: either a set is split, or an edge is added to the graph (since we assumed that the pirate always gives an answer as long as he has active traitors). We show that the number of such events is at most $O(p^3 \log n)$, and thus the algorithm must terminate and its running time is $O(p^3 \log n)$. This also proves that the algorithm eventually finishes by locating a traitor. By Lemma 2.1, all p traitors can be located in the worst case in at most $O(p^3 \log n + (p-1)) = O(p^3 \log n)$ rounds.

By Lemma 3.2, the total number of splits is $O(p^2 \log n)$. It remains to bound the number of edges added during the algorithm. The final graph is a (t,k)-graph for some k, where $t \leq p$. Such a graph can contain at most $2t+1 \leq 2p+1$ vertices, and thus at most $O(p^2)$ edges. However, in some rounds edges may be removed. We show below that the number of edges removed during the algorithm is $O(p^3 \log n)$. It implies that at most $O(p^2 + p^3 \log n) = O(p^3 \log n)$ edges are added during the algorithm and the theorem follows.

Edges are removed when: (i) a set is split or (ii) two cliques are merged into a larger clique. Each time such an event happens, at most $O(p)$ edges are removed (including the edges that are possibly removed in Step (3) of Phase 2). Event (i) can happen at most $O(p^2 \log n)$ times by Lemma 3.2. The number of times two cliques are merged is bounded from above by the number of times the number of cliques increases, and this happens only when a vertex is split. Thus, event (ii) can happen at most $O(p^2 \log n)$ times. Therefore, at most $O(p^3 \log n)$ edges are removed during the algorithm. ∎

It may seem at first that a more careful analysis can show that the number of splits is $O(p \log n)$, which would improve the overall running time by a factor of p. However, it is possible to demonstrate a run of the algorithm where indeed $\Theta(p^2 \log n)$ splits occur, and $\Theta(p^3 \log n)$ rounds are needed.

4. An Optimal Algorithm. In this section we describe an algorithm that uses $p+1$ colors, and runs in an optimal number of $\Theta(p^2 + p \log n)$ rounds. We start with an overview of the algorithm.

The Clique algorithm loses efficiency, because of the high number of $O(p^2 \log n)$ splits, and due to the fact that whenever we split a vertex, we may remove $O(p)$ edges. We solve these problems as follows:

The users are partitioned into two areas, Area 1 and Area 2. In Area 1 the users are organized into cliques of singletons (recall that a vertex is called a singleton if it contains exactly one user). Here we allow cliques of any size and use the Clique algorithm. Since all vertices are singletons, there are no split operations. It follows that the total number of rounds in Area 1 is $O(p^2)$.

Initially, all the users are placed in Area 2. The only case when users are moved from Area 2 to Area 1 is when we find a set of two users containing a traitor. Then we add them to Area 1 as a clique of two singletons. Area 2 is divided into blocks, each with a constant number of known traitors. This allows us to achieve an amortized cost of $O(1)$ rounds per split and an amortized cost of $O(\log n)$ rounds until we pin down a traitor to a set of two users. Thus, at an amortized cost of $O(\log n)$ we increase the number of traitors in Area 1 (by placing there the new clique of two singletons that we found in Area 2). The number of known traitors can increase at most p times, and thus the resulting number of rounds is $O(p^2 + p \log n)$, exactly matching our lower bound.

This grand plan has some difficulties. The main obstacle is that the Clique algorithm cannot be used on the blocks of Area 2, since we are not allowed to create large cliques on non-singleton vertices (as this may result in many splits). An additional problem is that in each block of Area 2 we cannot use more colors than the number of traitors known to exist in the block (otherwise, we would violate the bound of $p + 1$ colors). We now sketch briefly the solutions used in Area 2 to overcome these difficulties:

Most of Area 2 is organized into blocks containing at least 3 known traitors each (Zone Z_4 below). In each such block, our goal is to either (i) find a set of two users containing a traitor, or (ii) split the block into two parts, one part with 3 known traitors and one part with one known traitor. In case (i), we can place these two users as a clique of two singletons in Area 1. In case (ii), the number of known traitors in Area 2 has increased, while maintaining the invariant of blocks with a constant number of known traitors in this area. Since our means are very limited, it may happen that we can show that there are 4 traitors in the block, but we are not able to partition them into two parts as required in (ii). In such a case we continue with a block known to contain more traitors. Eventually, when the number of traitors known to exist in the block raises to 7, we are able to achieve the required partition, using the Basic algorithm from Section 2.4 with 7 colors.

Note that even if we know that there are 7 traitors in the block, we do not necessarily know their exact location among the two parts into which the block was partitioned. Hence, the total number of traitors known to exist in some particular block may decrease (this never happens in the Clique algorithm).

Another major problem is that we cannot treat the two areas, Area 1 and Area 2, as two completely independent processes. If we did, we would need in each of the two areas one more color than the number of traitors known to exist in that area. This would add up to $p + 2$ colors instead of the desired $p + 1$ colors. To solve this, we distinguish between the total number of traitors known to exist and the number of traitors known to exist in each one of the areas. Using a single additional color for both areas we are able to prove the existence of an additional traitor without knowing to which area it belongs. (Technically this is done by using the concept of *marking* of vertices in the algorithm.) Once we know that an additional traitor exists, we can safely use one extra color and continue the algorithm with one additional color in each area.

4.1. Data Structures and Invariants.

The graph. We maintain an undirected graph $G = (V, E)$. The vertices are subsets of the user set U, and form a partition of U. If there is an edge $(X, Y) \in E$, then $X \cup Y$ contains a traitor.

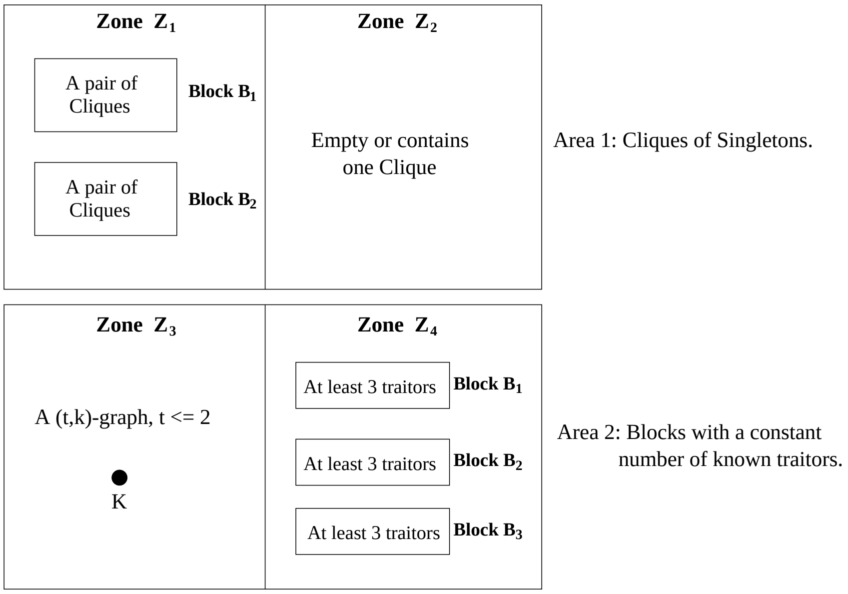

FIG. 4.1. *An example of the data structures and the graph used by the optimal algorithm.*

Zones and blocks. The vertices of the graph G are partitioned into four zones Z_1, Z_2, Z_3, Z_4. Furthermore, the vertices in Zones Z_1 and Z_4 are partitioned into blocks B_i. There are no edges connecting vertices from different zones or from different blocks. Let b_i be the number of traitors known to exist in block B_i, and z_i the number of traitors known to exist in Zone Z_i. The value of z_1 and z_4 is always the sum of all b_i's over the blocks of the corresponding zone. The zones and the blocks are defined as follows:

Area 1 (Cliques of singletons):

- $\mathbf{Z_1}$ This zone is partitioned into blocks B_i, where the number of vertices in B_i is $b_i + 2$, and all vertices are singletons. The vertices of block B_i can be partitioned into two cliques, each containing at least two vertices (therefore, the number of traitors known to exist in B_i is b_i). However, B_i does not contain a clique of size $b_i + 1$.
- $\mathbf{Z_2}$ This zone is either empty or contains a clique of size $z_2 + 1 \geq 2$ whose vertices are all singletons.

Area 2 (Blocks with a constant number of known traitors):

- $\mathbf{Z_3}$ This zone is a (t, k)-graph where $t \leq 2$. The number of traitors known to exist in Z_3 is $z_3 = t \leq 2$. The special vertex I that contains "innocent users" is denoted here by K. Users from other zones may be added to the set K during the algorithm.
- $\mathbf{Z_4}$ This zone is partitioned into blocks B_i, where the number of traitors known to exist in B_i is $3 \leq b_i \leq 7$.

Number of traitors. Let $z = z_1 + z_2 + z_3 + z_4$. The value of z is updated automatically, according to the values of b_i and z_i in the individual blocks and zones. The value z is a lower bound on the number of traitors known to exist, however it may increase or decrease. We can always use safely $z + 1$ colors.

We define another value T. Similar to z, the value T is a lower bound on the number of traitors known to exist. However, T never decreases and always $T \geq z$. Whenever z increases so that we would have $T < z$, we set $T = z$. (T may be larger than z, if we know that there exist additional traitors but we do not know to which zone they belong. The practical implication is that the algorithm may use in this case $z + 2$ colors.)

Marks. The vertices in Zones Z_1 and Z_2 are all singletons, and they may be *marked*. The invariant is that either the marked user is a traitor, or that the total number of traitors is at least $T + 1$ (but we do not know which is the case). Additionally, in no clique in Zones Z_1 and Z_2 are all vertices marked. As a rule, whenever T increases, all marks are removed.

4.2. Description of the Algorithm. We describe the algorithm with respect to each block and zone separately. When the pirate broadcasts a color given to the vertices in one of the blocks or zones, we advance the corresponding algorithm and do not change the coloring in the remaining zones.

We now describe in details the algorithm with respect to each one of the zones. The initial graph is $G = (V, E)$, where $K = U$, $V = \{K\}$ and $E = \emptyset$ (hence, at the beginning all zones are empty except Zone Z_3). Set $T = z = 0$.

Zone Z_1: We run the Clique algorithm. We need only z_1 colors, since the number of cliques in Z_1 is even (as they are paired into blocks and there is no vertex I in Zone Z_1). Since all vertices are singletons, every step creates a new edge (unless we locate a traitor).

Eventually, two cliques in some block B_i are merged into one clique Q of size $b_i + 1$. Let Z be the remaining vertex in B_i that does not belong to this large clique Q. Remove all edges incident with Z, and add Z to K in Zone Z_3. Reorganize the zones as follows: If Zone Z_2 is empty, the clique Q is placed in Zone Z_2. Otherwise, we pair the clique in Z_2 with Q into a new block in Z_1, and Z_2 becomes empty.

Zones Z_2 and Z_3: We use $z_2 + z_3 + 1$ colors for these two zones if $z = T$, and $z_2 + z_3 + 2$ colors if $z < T$.

(1) In Z_3 we do the following:

- **(1.1)** We run the Clique algorithm on Zone Z_3, until we know that it contains $z_3 = 3$ traitors. To do this in only $O(\log n)$ rounds, let c be a constant such that if there were only 2 traitors among the users, then the Clique algorithm would run for at most $c \log n$ steps. If the Clique algorithm runs for more than $c \log n$ steps without reaching $z_3 = 3$, we conclude that there are 3 traitors in the zone and we set $z_3 = 3$.
- **(1.2)** When $z_3 = 3$, all vertices in Z_3 are placed as a block in Zone Z_4. Zone Z_3 is now empty, and we set $z_3 = 0$ and $K = \emptyset$.

(2) If Zone Z_2 is not empty and $z < T$: Use a separate color for each vertex of Zone Z_2.

(3) If Zone Z_2 is not empty and $z = T$: If the algorithm for Z_3 uses at the current round only z_3 colors (out of the $z_3 + 1$ colors allocated to it), use a separate color for each vertex of Zone Z_2.

Otherwise, modify the algorithm for Z_3 as follows. Suppose that the algorithm for Z_3 would use at this round $q = z_3 + 1$ colors, for coloring the subsets $X_1, \ldots, X_q$ (note that $q \leq 3$). Replace this round by the following q rounds. Fix an unmarked vertex S in Z_2. In round k, $k = 1, \ldots, q$, color the subsets $X_1, \ldots, X_q$ in Z_3 as before, color S with the color of X_k, and color the remaining vertices of Z_2 with distinct colors. (The number of colors used in this case is $q + z_2 = z_2 + z_3 + 1$ as required). Then:

(3.1) If the pirate broadcasts the color of X_j, for some $j \neq k$: Advance the algorithm in Step (1) above of Z_3.

(3.2) Otherwise, the pirate eventually broadcasts the q colors of all pairs $X_k \cup S$. In this case mark S. If all the vertices in Z_2 are marked, set $T = T + 1$ (and remove all marks).

Remark: Recall that the clique in Zone Z_2 may be moved to Zone Z_1, and in a later stage may be moved back to Zone Z_2, as part of a larger clique (after two cliques are merged). Therefore, there may be marks on vertices of both Zone Z_1 and Z_2.

The algorithm used for the blocks in Zone Z_4 is fairly complex. The main building block used here is the following Algorithm (X). We now provide the specification of Algorithm (X), and then describe the algorithm for Zone Z_4. The description and analysis of Algorithm (X) is postponed to Section 4.4.

Specification of Algorithm (X): *The input is a block containing at least 3 traitors. In each step the algorithm uses q colors, where q is the number of traitors currently known to exist in the block. At all times $3 \leq q \leq 7$. The algorithm is allowed to discard some users; in the context of the whole algorithm these users are added to the set K in Zone Z_3.*

After $O(\log n)$ rounds the algorithm either (i) finds a subset of two users (in the block) that contains a traitor, or (ii) finds two disjoint subsets of the input block, one containing at least 3 traitors and one containing at least one traitor.

Zone Z_4: We run Algorithm (X) on each of the blocks in this zone, thus using z_4 colors. When Algorithm (X) finishes in one of the blocks B_i, we have one of the following two cases.

(1) Block B_i is split into two subsets, one with at least 1 traitor and the second with at least 3 traitors: The first subset is added to Zone Z_3 (see Step (3)). The second subset remains in this zone as a block of itself.

(2) We have a subset of two users in B_i that contains a traitor: We create a new clique Q of size 2, with two singleton vertices containing these two users, and reorganize the zones as follows. If Zone Z_2 is empty, we place the clique Q in Zone Z_2. Otherwise, we remove the single clique in Z_2, pair it with the new clique Q, and create in Zone Z_1 a new block containing these two cliques; Z_2 is now empty. The rest of block B_i is added to Zone Z_3 (see Step (3)).

(3) In each of the above cases, some subset X of block B_i, known to contain at least one traitor, is added to Zone Z_3. This is done as follows: Remove the edges incident with X, divide X into two (almost) equal vertices connected by an edge, and add them as a clique of size 2 to Zone Z_3. Restructure the cliques of Zone Z_3 as necessary, in a similar way to Step 3 in Phase 2 of the Clique algorithm. Set $z_3 = z_3 + 1$. If $z_3 = 3$, go to Step (1.2) in Z_3.

4.3. Correctness and Efficiency.

LEMMA 4.1. *The invariants of the algorithm are preserved, assuming that the same holds for Algorithm (X).*

Proof.

The graph, zone constraints, edges between different zones and blocks. In Area 1 and Zone Z_3 these invariants are maintained because of the properties of the Clique algorithm. In Zone Z_4 they are maintained because of the properties of Algorithm (X). One additional invariant is that $z_3 \leq 2$. If this invariant is violated, then the zones are restructured appropriately in Step (1.2) of Zone Z_3. At all times the vertices in Area 1 are singletons: this is true when new vertices are created in Step (2) of Zone Z_4, and maintained by the Clique algorithm.

We need to check that the invariants are not violated when adding users to Zone Z_3 from other zones. If some users are added to K in Zone Z_3, then edges incident with K are still valid since the corresponding set of users can only become bigger. If a 2-clique is added to Z_3 by Step (3) of Z_4, then the structure of Z_3 is changed consistently with the Clique algorithm. If $z_3 > 2$ then the zones are restructured.

The marks. The vertices are only marked in Step (3.2) of Zones Z_2 and Z_3. If a vertex is marked and is not a traitor, it follows that all the $q = z_3 + 1$ sets $X_1, ..., X_q$ described there contain a traitor. This would imply that there are at least $z + 1$ traitors overall as required. Since this step is invoked only when $z = T$, the vertex is marked consistently with the invariant. The invariant of the marked vertices is maintained as long as T is unchanged, and as soon as T increases, all marks are removed.

It cannot happen that all vertices in some clique are marked. If this happens upon marking a new vertex in Zone Z_2, then the value of T is increased and all marks are removed. A new clique is created when two cliques in Zone Z_1 are merged. In this case the new clique contains all the vertices of one of the merged cliques, and thus they cannot be all marked.

The number of traitors. We need to check that there are always at least T traitors. The value T is changed in two cases. First, when z increases so that $z > T$. In this case the existence of T traitors is guaranteed by the invariants of the individual blocks and zones.

The second case is in Step (3.2) of Zones Z_2 and Z_3 when all vertices in Z_2 are marked. If some vertex in Z_2 is not a traitor then by the marking invariant there are at least $T + 1$ traitors, and the increase of T is justified. Otherwise, all the vertices in Zone Z_2 are traitors, for a total of $z_2 + 1$ traitors. Thus, the total number of traitors is at least $z + 1$, and since $T = z$ in this step, the increase of T is justified as well. $\square$

THEOREM 4.2. *The algorithm locates a traitor in $O(p^2 + p \log n)$ rounds, using $p + 1$ colors. Hence, all p traitors can be located in $O(p^2 + p \log n)$ rounds, using $p + 1$ colors.*

Proof. The number of colors used is at most $z_1 + z_2 + z_3 + z_4 + 1 = z + 1$ if $z = T$, and $z_1 + z_2 + z_3 + z_4 + 2 = z + 2$ if $z < T$. In both cases, at most $T + 1 \leq p + 1$ colors are used.

We now analyze the number of rounds needed by the algorithm. After at most $q = 3$ rounds in Step (3) of Zones Z_2 and Z_3, we either mark a vertex or advance one of the other algorithms (most often the Clique algorithm used in Step (1) of Z_3, but it may happen that one of the other algorithms changes z, in which case we cannot continue the series of rounds in Step (3) of Zones Z_2 and Z_3). In the first case we charge these rounds to marking a vertex. The second case adds at most $q - 1$ steps per one step of the other algorithms, so the total number of rounds may increase by a factor of 3.

Let $s = z_1 + z_2$ be the number of traitors known to exist in Zones Z_1 and Z_2 (which contain only singleton vertices). Let $r = z_1 + z_2 + z_3 + 3b$, where b is the number of blocks in Zone Z_4. Note that r is a lower bound on the number of traitors in all zones, since the blocks of Zone Z_4 contribute only 3 traitors each instead of b_i (which may be as large as 7).

The values of T, s, and r are between 0 and p at all times. The values of s and T never decrease, and thus each of them increases at most p times. The value of r decreases only when a subset of two users containing a traitor is found in a block in Z_4. In such a case s increases by 1 and r decreases by 2 (since b decreases by 1 and z_3 increases by 1). It follows that r may increase at most $3p$ times.

The value of r increases after each (amortized) $O(\log n)$ rounds in a block in Zones Z_4 and Z_3, not counting the rounds charged to marking of vertices in Zone Z_2. Thus, the number of these rounds is $O(p \log n)$.

Each of the remaining rounds is either charged to marking a vertex, or creates an edge between two cliques of singletons. We unmark vertices or remove edges when T increases or when two cliques are merged. We increase T at most p times, and each time $O(p)$ vertices are unmarked. This accounts for $O(p^2)$ rounds. A clique of singletons is never split. So we get a new clique only when s increases, which happens at most p times. Thus, two cliques are merged at most p times. Each time we remove at most p edges and one marked vertex. This accounts for $O(p^2)$ rounds. At the end we have $O(p)$ marked vertices and $O(p^2)$ edges. This accounts for $O(p^2)$ rounds as well. Thus, the total number of rounds needed to locate one traitor is $O(p^2 + p \log n)$. Again, by Lemma 2.1, all p traitors can be located in the worst case in at most $O(p^2 + p \log n + (p-1)) = O(p^2 + p \log n)$ rounds. □

4.4. Algorithms for Blocks in Zone Z_4. Algorithm (X) described in this section receives as an input a block known to contain at least 3 traitors. Its goal is to either (i) find a subset of two users containing a traitor, or (ii) partition the block into two parts, one with 3 traitors and one with a single traitor. This is achieved in $O(\log n)$ rounds.

The plan is to first partition the block into a constant number of $q \geq 3$ parts, where each part contains at least one traitor. Then we either halve one of these q parts, and thus come closer to achieving goal (i); or we prove that the block contains at least $q+1$ traitors, and thus advance towards goal (ii).

Since we cannot use more colors than the number of traitors known to exist in this block, the pirate can confuse our algorithm if there are in fact more traitors. We found the following conditional invariant extremely useful in achieving our aims:

DEFINITION 4.3. *A block is q-good if it is partitioned into $q+1$ sets of users A_1, ..., A_q, and J with the property that either each of the sets A_i contains one traitor or the block contains more than q traitors.*

In other words, a q-good block either contains exactly q traitors located in q disjoint sets, or there are more than q traitors, but we have no knowledge about their location. (Such a block is good in the sense that the algorithm is making progress in the desired direction.)

Thus, instead of finding an uncoditional partitioning of the block into 3 parts, each containing a traitor, we try to find a partition which demonstrates that the block is 3-good. However, this may not succeed, and instead we might just learn that there are 4 traitors. Then we try to find a 4-good block, etc. Eventually, when we are able to prove that there are at least 7 traitors in the block, we can partition the block unconditionally as required in (ii), using the Basic algorithm of Section 2.4 with 7 colors.

The main part of the algorithm is how to halve of one of the sets A_i in a q-good block (and still remain with a q-good block), or to deduce that there are at least $q+1$ traitors in the block. Our algorithm is motivated by the exponential algorithm of Fiat and Tassa [7]. Their method is as follows: Split each one of the sets $A_1, ..., A_q$ into two (almost) equal parts. Choose one part from each one of the sets A_i, and color these q chosen parts with q unique colors. Use an additional color for all the remaining users (consisting of the parts of the sets A_i that did not receive a unique color and of the set J). Try all 2^q combinations of choosing q parts from each one of the sets A_i. If we get an answer in one of the parts that were colored by a unique color, then we have achieved the required split of the corresponding set A_i. Otherwise, we can conclude that there are more than q traitors. However, this method uses $q+1$ colors, and we can only use q colors. Thus, our algorithm is modified to use one common color for two of the q chosen parts of the sets A_i. After trying all of the $\binom{q}{2}2^q$ combinations and analyzing the answers, we are able to halve a set or prove that there are at least $q+1$ traitors. Since q is a constant, the number of combinations and rounds needed is also a constant.

4.4.1. Description of the Algorithms. Algorithms (I) through (IV) described below are used as building blocks for Algorithm (X), which is then used by the algorithm for Zone Z_4, in Section 4.2.

Algorithms (I) through (IV) and Algorithm (X) stop whenever they find *a subset of two users that contains a traitor*. In fact this event always stops not only the current algorithm, but also stops Algorithm (X) in the current block. We do not mention this in the descriptions of the algorithms explicitly. This also means that when we have an edge in this part of the graph, at least one of its endpoints is not a singleton. Similarly, recall that the algorithm stops upon locating a traitor. Therefore, if we get a vertex as an answer, it is not a singleton.

The algorithms are allowed to *discard* some users. In the context of the Optimal algorithm described in Section 4.2, these users are added to the set K in Zone Z_3.

The correctness of the individual algorithms follows immediately from the remarks in their description. After the description of each algorithm we argue briefly to bound its running time.

Algorithm (I): *The input is a constant $q \geq 3$ and a q-good block. Using q colors and $O(\log n)$ rounds, the algorithm proves that there are more than q traitors in this block. (The algorithm also stops upon finding a set of two users containing a traitor. It never discards any users.)*

Throughout the algorithm, we maintain a q-good block. Additionally, each set A_i is split into two disjoint sets $A_{i,0}$ and $A_{i,1}$ of equal size (more exactly, their cardinalities may differ by one and if A_i is a singleton, then one of the sets may be empty).

(1) In successive $\binom{q}{2}2^q = O(1)$ rounds, use all $\binom{q}{2}2^q$ colorings of the following form. For any $a_1, \dots, a_q \in \{0,1\}$, and $1 \leq j < j' \leq q$: Use one color for the set $A_{j,a_j} \cup A_{j',a_{j'}}$, and $q-2$ colors for the sets A_{i,a_i}, $i \notin \{j,j'\}$, one color per set. The last color is given to the union of the remaining $q+1$ sets (including J).

(2) If the pirate ever broadcasts a color given to only one (non-singleton) set $A_{i,a}$: Add $A_{i,1-a}$ to J, set $A_i = A_{i,a}$, split A_i into $A_{i,0}$ and $A_{i,1}$, and go to Step (1).

(3) If in the above $\binom{q}{2}2^q$ rounds the pirate broadcasts the colors of all possible 2^q sets containing J, each at least once, then the block contains more than q traitors and we are done.

(4) Otherwise, we may suppose after renumbering, that the pirate never broadcasts the color of $J \cup A_{1,1} \cup \dots \cup A_{q,1}$. This means that the pirate broadcasts the color of $A_{j,0} \cup A_{j',0}$ for all $j \neq j'$. If two of the sets A_j and $A_{j'}$ are singletons, then $A_{j,0} \cup A_{j',0} \subseteq A_j \cup A_{j'}$ is a subset of cardinality at most two that contains a traitor, and we are done.

Otherwise, there is at most one singleton set. Renumber the sets so that A_1 and A_2 are not singletons (using $q \geq 3$). The sets $A_{1,0}, \ldots, A_{q,0}$ form a clique of size q, and therefore contain at least $q-1$ traitors. Below we find a set $A_{j,1}$ that does not contain a traitor, assuming that there are only q traitors. Given such a set $A_{j,1}$, we add it to J, set $A_j = A_{j,0}$ and go to Step (1).
In order to find such a set $A_{j,1}$, we distinguish between two subcases. Let us examine the answer given by the pirate in the round when we used one color for the set $J \cup A_{1,0} \cup A_{2,1} \cup \cdots \cup A_{q,1}$ and one color for the set $A_{1,1} \cup A_{2,0}$.

(4.1) If the answer was $A_{1,1} \cup A_{2,0}$: Assuming that there are only q traitors, $A_{2,1}$ does not contain a traitor. Otherwise, $A_{2,0}$ has no traitor, and thus $A_{1,1}$ contains a traitor; it follows that there are more than q traitors (together with the $q-1$ traitors in $A_{1,0} \cup \cdots \cup A_{q,0}$).

(4.2) If the answer was $J \cup A_{1,0} \cup A_{2,1} \cup \cdots \cup A_{q,1}$: Assuming that there are only q traitors, $A_{1,1}$ does not contain a traitor. Otherwise, $A_{1,0}$ does not contain a traitor, and thus either J or some $A_{j,1}$, $j \neq 1$, contains a traitor; it follows that there are more than q traitors.

After $O(1)$ rounds, if the algorithm is not done, one of the non-singleton sets is split and half of it is added to J. Thus, the total number of rounds is $O(q \log n) = O(\log n)$ (since $q = O(1)$).

Algorithm (II): *The input is a constant $q \geq 3$ and a block known to contain at least q traitors. Using q colors and $O(\log n)$ rounds, the algorithm either* (i) *proves that there are more than q traitors in the block or* (ii) *finds a subset of the input block which consists of k cliques with a total of $k+q$ vertices (i.e., q of the vertices have to contain a traitor), for some odd number k. (The algorithm also stops upon finding a set of two users containing a traitor.)*

We run the Clique algorithm on this block, with its own special set I. Initially, I contains all the users. Let c be a constant such that if there were only q traitors among the users, then the Clique algorithm would run for at most $c \log n$ rounds. If the Clique algorithm does not finish in $c \log n$ steps, we conclude that there exist more than q traitors and we are done by (i).

If the Clique algorithm needs $q+1$ colors, then there are k cliques with a total of $k+q$ vertices, for some k. If k is odd, we are done. Otherwise, we discard the set I and continue running the Clique algorithm with only q colors, since we do not need the color for I. Whenever a vertex is supposed to be added to I, it is discarded (this happens when a vertex in a clique of size two is split, or when two cliques are merged). Eventually, either two cliques are merged, or a vertex contained in a clique of size at least 3 is split. In either case we get an odd number of cliques and we are done by (ii).

The number of rounds of this algorithm is $O(\log n)$ as claimed because we explicitly restricted it.

Algorithm (III): *The input is a block consisting of a clique Q on 5 vertices. Using 4 colors and $O(\log n)$ rounds, the algorithm either* (i) *finds a 4-good block which is a subset of the input block or* (ii) *finds two disjoints subsets of the input block, one containing at least 3 traitors and one containing at least one traitor. (The algorithm also stops upon finding a set of two users containing a traitor.)*

(1) Let A be a non-singleton vertex of Q. Remove the edges incident with A, and split A into two vertices A_1 and A_2 of (almost) equal size. The block now contains the clique $Q \setminus \{A\}$, and the vertices A_1 and A_2.

(2) Find two disjoint non-edges, one containing A_1 and one containing A_2. Use two colors for these non-edges and two colors for the remaining two vertices. (A *non-edge* is a pair of vertices in this block that are not connected by an edge in the graph G.)

(3) If the pirate broadcasts a color given to one vertex: We are done by (ii), since this vertex is a subset with one traitor and the remaining 4 vertices contain 3 traitors.

(4) If the pirate broadcasts a color given to two vertices: Add that edge and repeat Step (2). If the two disjoint non-edges in Step (2) do not exist, one of the following cases occurs:

(4.1) We have a 5-clique containing one of A_1 and A_2: Discard the other set A_i and go back to Step (1).

(4.2) We have a graph on the 6 vertices containing all edges except for those in the triangle A_1, A_2, B, for some vertex B: This is a 4-good block with $J = \emptyset$, one of the sets being $A_1 \cup A_2 \cup B$, and the other 3 sets being the remaining vertices. Thus, we are done by (i).

After a constant number of rounds, if the algorithm does not end, it goes back to Step 1 with a 5-clique in which one of the vertices is halved. Thus, after $O(\log n)$ rounds we obtain a clique with two singletons and the algorithm stops.

Algorithm (IV): *The input is a block consisting of a clique Q on 4 vertices. Using 3 colors and $O(\log n)$ rounds, the algorithm finds a 3-good block which is a subset of the input block. (The algorithm also stops upon finding a set of two users containing a traitor.)*

(1) Let A be a non-singleton vertex of Q. Remove the edges incident with A, and split A into two vertices A_1 and A_2 of (almost) equal size.

(2) Find two disjoint non-edges, one containing A_1 and one containing A_2. Use two colors for these non-edges and one color for the remaining vertex.

(3) If the pirate broadcasts a color given to only one (non-singleton) vertex B, one of the following cases occurs:

(3.1) If the edge (A_1, A_2) does not exist: Unite back $A = A_1 \cup A_2$ and split B into (almost) equal parts B_1 and B_2, connected by an edge. The remaining three vertices form a clique, and there are no other edges. Set $A_1 = B_1$ and $A_2 = B_2$ and go to Step (2).

(3.2) Otherwise: We have a 3-good block with $J = \emptyset$, and the three sets are A, B and the union of the remaining two vertices. Thus, we are done.

(4) If the pirate broadcasts a color given to two vertices: Add that edge and repeat Step (2). If the two disjoint non-edges in Step (2) do not exist, one of the following cases occurs:

(4.1) We have a 4-clique containing one of A_1 and A_2. Discard the other set A_i and go back to Step (1).

(4.2) We have a graph on the 5 vertices containing all edges except for those in the triangle A_1, A_2, B, for some vertex B. This is a 3-good block with $J = \emptyset$, one of the sets is $A_1 \cup A_2 \cup B$, and the other two sets are the remaining vertices. Thus, we are done.

After a constant number of steps, if the algorithm does not end, it goes back to Step 1 with a 4-clique in which one of the vertices is halved (note that Step (3.1) may occur only once). Thus, after $O(\log n)$ rounds we obtain a clique with two singletons and the algorithm stops.

Algorithm (X): *The input is a block containing at least 3 traitors. In each step the* *algorithm uses q colors, where q is the number of traitors currently known to exist in the block. After $O(\log n)$ rounds the algorithm finds two disjoint subsets of the input block, one containing at least 3 traitors and one containing at least one traitor. (The algorithm also stops upon finding a set of two users containing a traitor.)*

(1) Prove that the block contains at least 4 traitors (or finish): Run Algorithm (II) with $q = 3$. If it proves that there are 4 traitors, go to Step (2). Otherwise, we have k cliques with a total of $k + 3$ vertices, where $k \in \{1, 3\}$. If $k = 3$, then the block is 3-good (with $J = \emptyset$ and the 3 sets each being the union of one clique). If $k = 1$, we have a 4-clique: Run Algorithm (IV) to find a 3-good block. In either case we get a 3-good block. Run Algorithm (I) to prove that the block contains 4 traitors, and go to Step (2).

(2) Prove that the block contains at least 5 traitors (or finish): Now we have a block with 4 traitors. Run Algorithm (II) with $q = 4$. If it proves that there are 5 traitors, go to Step (3). Otherwise, we have k cliques with a total of $k + 4$ vertices, where $k \in \{1, 3\}$.

- **(2.1)** If $k = 3$, then the number of traitors in the 3 cliques is 1, 1, and 2. We split the block into a clique with one traitor and a subset containing the remaining two cliques with 3 traitors, and we are done.
- **(2.2)** If $k = 1$, then we have a 5-clique, and we run Algorithm (III). If we get two subsets with 1 and 3 traitors, we are done. Otherwise, we have a 4-good block. We run Algorithm (I) with $q = 4$ to prove that the block contains 5 traitors, and go to Step (3).

(3) Now we have a block with 5 traitors. Run Algorithm (II) with $q = 5$.

- **(3.1)** If we get cliques, we split the block into two subsets—one with at least one traitor and one with at least 3 traitors—and we are done. This is done as follows: If there are at least 3 cliques, one subset is the smallest clique and the other one is the rest. If there is a single clique, one subset is the union of two of the vertices of the clique, and the other subset contains the remaining vertices.
- **(3.2)** If we get a block with 6 traitors: Run Algorithm (II) with $q = 6$. If we get cliques, we split them as in Step (3.1). Otherwise, we have a block with 7 traitors. We run the Basic algorithm described in Section 2.4 (for locating p traitors with $2p + 1$ colors) until it needs more than 7 colors. This happens when it finds 4 disjoint subsets each containing a traitor (the pairs in the Basic algorithm). Split the block into one of these subsets and the remaining subsets.

Algorithm (X) runs Algorithms (I) through (IV) $O(1)$ times, and each of them takes $O(\log n)$ rounds. Step (3.2) also needs $O(\log n)$ rounds. Thus, the total number of rounds is $O(\log n)$.

5. Using More Colors. In this section we extend our results, and present two algorithms which use more than $p + 1$ colors. The number of rounds needed to locate all traitors is reduced as the number of colors grows.

5.1. Using $p + c + 1$ Colors. We modify the Optimal algorithm from Section 4 to use $p + c + 1$ colors, for any $1 \leq c \leq p$. The additional colors are used for cliques of singletons with more than p/c vertices. This ensures that all cliques have at most $O(p/c)$ vertices, and therefore the number of edges removed in each round is smaller. The additional colors allow us not to use marks on the vertices in the corresponding part of the Optimal algorithm. (Recall that the marks were used to decrease the number of colors from $p + 2$ to $p + 1$.)

5.1.1. Data Structures and Invariants. The basic structure is similar to that of the optimal algorithm with the following modifications.

Zones and blocks.

$\mathbf{Z_1}, \mathbf{Z_2}$ As in the Optimal algorithm. Additionally, any clique in these zones is required to have at most T/c vertices.

$\mathbf{Z_3}, \mathbf{Z_4}$ As in the Optimal algorithm.

$\mathbf{Z_5}$ This new zone is partitioned into blocks B_i, where each B_i contains one clique of singletons of size $b_i + 1$, such that $T/c < b_i + 1 \leq 2T/c$.

Marks. There are no marks on the vertices.

5.1.2. Description of the Algorithm.

Zone Z_1**:** As in the Optimal algorithm (i.e., run the Clique algorithm using z_1 colors). If a new clique with more than T/c vertices is produced by merging two cliques, add it as a new block in Z_5 (instead of incorporating it into Z_1, Z_2).

Zones Z_2 **and** Z_5**:** Each vertex gets a unique color. If the size of a clique in Z_5 becomes $\leq T/c$ (because T increased), incorporate this clique into Z_1 and Z_2 (similarly as a new clique in the Clique algorithm).

Zone Z_3**:** Run the Clique algorithm as in the Optimal algorithm, using $z_3 + 1$ colors, with no modifications due to marks in Z_2.

Zone Z_4**:** As in the Optimal algorithm, using z_4 colors. If $T/c < 2$ and a new clique of two singletons is created, add it as a new block in Z_5 (instead of incorporating it into Z_1, Z_2).

5.1.3. Correctness and Efficiency.

LEMMA 5.1. *The invariants of the algorithm are preserved at all times.*

Proof. Each clique added to Zone Z_5 is produced by merging two cliques, each of size at most T/c, so its size is at most $2T/c$ as required. The bounds on the size of the cliques are maintained; to verify this we also use the fact that T never decreases. All other invariants are maintained similarly as in the optimal algorithm. ∎

THEOREM 5.2. *The algorithm locates a traitor in $O(p^2/c + p \log n)$ rounds, using $p + c + 1$ colors, for any $1 \leq c \leq p$. Hence, all p traitors can be located in $O(p^2/c + p \log n)$ rounds, using $p + c + 1$ colors.*

Proof. Compared to the optimal algorithm, we need one extra color for each clique in Zones Z_2 and Z_5. Let m be the number of these cliques, where $m - 1$ of these cliques belong to Z_5. Thus, $T \geq z_2 + z_5 \geq 1 + (m-1)T/c$. It follows that $m \leq c$ and thus the number of colors used is at most $p + c + 1$.

The analysis of the number of rounds is the same as in Theorem 4.2, with the following differences in the area of singletons. First, we have no marks, so we do not charge any work to marking and unmarking vertices. Second, the degree of any vertex in Zones Z_1 and Z_2 is $O(p/c)$. Thus, the number of edges removed during the algorithm and the final number of edges in G is bounded by $O(p^2/c)$, instead of $O(p^2)$. Hence, the total number of rounds needed to locate a traitor is $O(p^2/c + p \log n)$, and the theorem follows by using Lemma 2.1. ∎

5.2. Using $pc + 1$ Colors. Now we allow to use $tc + 1$ colors when t traitors are known to exist, for any integer $c \geq 2$. The algorithm is an immediate extension of the Basic algorithm described in Section 2.4.

In each stage of the algorithm we keep t disjoint sets S_i, each known to contain a traitor. Furthermore, each set is partitioned into c subsets $S_{i,j}$ of almost equal sizes.

In addition we have the set I of innocent users, which at the beginning contains all users.

(1) Distribute the colors as follows: For each set S_i, color the c subsets $S_{i,j}$ using c colors, one color per subset. Color I with an additional color. The total number of colors used is thus $tc+1$.

(2) If the pirate broadcasts the color of I: Create a new set $S_{t+1} = I$ and partition it into c (almost) equal subsets. Set $I = \emptyset$ and $t = t+1$.

(3) If the pirate broadcasts the color of a subset $S_{i,j}$: Add all subsets $S_{i,j'}$, $j' \neq j$, to I. Set $S_i = S_{i,j}$ and partition S_i into c new (almost) equal subsets.

THEOREM 5.3. *The algorithm locates a traitor in $O(p \log_c n)$ rounds, using $pc+1$ colors, for $c \geq 2$. Hence, all p traitors can be located in $O(p \log_c n)$ rounds, using $pc+1$ colors.*

Proof. A subset is split at each round of the algorithm. Since a split divides a subset into c equal parts, each subset other than I can be split at most $O(\log_c n)$ times. The subset I may be split an additional p times. Thus, the algorithm finishes in $O(p \log_c n)$ rounds. □

6. The Degree Algorithm. In this section we present an algorithm based on a different idea. This algorithm uses $p+1$ colors and runs in time $O(p^3 \log n)$. Thus, the bounds are the same as for the Clique algorithm.

The knowledge of the algorithm is represented by a similar graph as before. That is, vertices represent disjoint subsets of users from U, and an edge between vertices X and Y represents the fact that the subset $X \cup Y$ contains a traitor.

The main idea here is to eliminate vertices with a large degree. To do that, note that if the degree of a vertex is larger than p, then this vertex must contain a traitor (otherwise, each of its neighbors would contain a traitor, resulting in more than p traitors). Therefore, if we knew p in advance, we would know that such a vertex contains a traitor and the algorithm could have split it.

Since p is unknown, we maintain a value T (as in the previous algorithms), which is the number of traitors known by the algorithm to exist. Whenever a vertex of a large degree appears, we *suspect* that it contains a traitor. (The notion of a large degree will be defined later). Note that our knowledge about these vertices is only conditional: we know that either the vertex contains a traitor, or that the actual number of traitors is greater than T (however, we do not know which is the case). Consequently, when T is increased, we have no information about any traitor in these suspected vertices and we have to treat the users as innocent.

Among the remaining vertices, we distribute the colors so that each color is used by one vertex or by two vertices not connected by an edge. An existence of such a coloring is implied by the fact that the degree of these vertices is small. In each round progress is made by adding a new edge, splitting a vertex, or finding a vertex of a large degree.

6.1. Data Structures and Invariants.

The graph. We maintain an undirected graph $G = (V, E)$. The vertices are subsets of the user set U, and form a partition of U. If there is an edge $(X, Y) \in E$, then $X \cup Y$ contains a traitor.

The Zones. The vertices of the graph G are partitioned into two Zones Z_1, Z_2. There are no edges adjacent to vertices in Zone Z_1. We also maintain a value T, which is the number of traitors known to exist. The zones are defined as follows:

$\mathbf{Z_1}$ This zone contains z_1 vertices. The invariant is that if the actual number of traitors is exactly T, then each vertex in Z_1 contains a traitor. As stated above, the degree of all vertices in this zone is 0.

$\mathbf{Z_2}$ This zone is a (z_2, z_2)-graph, i.e., a graph on $2z_2+1$ vertices containing z_2 disjoint cliques of size 2 each, and the special vertex I (there may be edges between the cliques). The invariant is that all vertices in this zone have degree at most $T - z_1$.

Number of traitors. Let $z = z_1 + z_2$. Thus, z is the number of traitors known to exist in distinct vertices of G, assuming that the actual number of traitors is T. The invariant is that the total number of traitors is at least T and that $T \geq z$. This implies that the algorithm may use $T + 1$ colors. As a rule, T never decreases. Note that the total number of vertices in the graph is $|V| = z_1 + 2z_2 + 1 = z + z_2 + 1$.

6.2. Description of the Algorithm. Before describing the algorithm we give a lemma about the existence of a large matching, and show how it is applied to find a proper coloring in the algorithm. A *matching* in a graph is a set of disjoint edges. The *size* of the matching is the number of edges in the matching. The proof of the following lemma is standard and can be found in Appendix A.

LEMMA 6.1. *Let H be a graph with at least $2d$ vertices with degree at least d each. Then there exists a matching in H of size at least d.*

Let G' be the complement graph of G restricted to Z_2. I.e., the vertices of G' are the vertices in Z_2, and two distinct vertices are connected by an edge in G' if and only if they are not connected by an edge in G. The Zone Z_2 contains $2z_2 + 1$ vertices, each with degree at most $T - z_1$. Thus, the minimum degree of the graph G' is $2z_2 - (T - z_1) = z_2 + z - T$. By taking $d = z_2 + z - T$ in Lemma 6.1, there exists a matching of size at least $z_2 + z - T$ in G'.

We now describe the algorithm. The initial graph is $G = (V, E)$ with $I = U$, $V = \{I\}$ and $E = \emptyset$. The Zone Z_1 is empty and Z_2 contains the single vertex I. Set $T = z = 0$. Repeat the following two phases:

Phase 1: Distributing the colors to the vertices of G

Find a matching of size at least $z_2 + z - T$ in G' (which exists by Lemma 6.1 and the above discussion). Use one color for each edge in the matching (i.e., for the two vertices of the edge). Use a distinct color for every remaining vertex, including the vertices of Z_1. The number of colors used is at most $|V| - (z_2 + z - T) = (z + z_2 + 1) - (z_2 + z - T) = T + 1$.

Phase 2: Reorganizing the graph following the pirate's response

(1) If the pirate transmitted a color given to a single vertex X: Remove X and all edges incident with X. Split X into two (almost) equal subsets X_1 and X_2, and add the edge (X_1, X_2). Place X_1 and X_2 as a new clique in Zone Z_2.
- **(1.1)** If X was in Z_1: Set $z_1 = z_1 - 1$ and $z_2 = z_2 + 1$.
- **(1.2)** If X was in Z_2 in a clique with Y: Remove Y and all edges incident with Y, and set $I = I \cup Y$.
- **(1.3)** If X was in Z_2 and $X = I$: Create a new special vertex $I = \emptyset$ and set $z_2 = z_2 + 1$.

(2) If the pirate transmitted a color given to a pair of vertices (X, Y) (necessarily in Z_2): Add the edge (X, Y) to G, and:
- **(2.1)** For any vertex $Z \in Z_2$ of degree larger than $T - z_1$ do the following: Move Z to Zone Z_1, remove all edges incident with Z, and set $z_1 = z_1 + 1$. If $Z = I$, create a new special vertex $I = \emptyset$.

(2.2) Repeat step (2.1) until all vertices have degree at most $T - z_1$. (Note that after step (2.1) the value of $T - z_1$ decreases, which may cause some more vertices to have a large degree.)

(2.3) Add to the special vertex I all the vertices which remain in Z_2 and do not belong to a clique of size 2 (i.e., the vertex I will contain all the vertices who had a neighbor Z (in their clique) that was moved to Z_1. The vertex I will also contain the old I, if it was not moved to Z_1). Update z_2 to reflect the current number of cliques of size 2 in Z_2.

(3) If after one of the previous steps $z > T$: Set $T = T + 1$ and add to I all the vertices that belong to Z_1. The Zone Z_2 remains unchanged.

6.3. Correctness and Efficiency.

LEMMA 6.2. *The invariants of the algorithm are preserved at all times.*

Proof.

The graph. An edge is added only if we got an answer from the pirate corresponding to that edge.

Zone Z_1. First consider the moment when we decide to place a vertex X in Zone Z_1. At that point X is in Z_2 and its degree is larger than $T - z_1$. Assume that the invariant is violated, i.e., X does not contain a traitor and the actual number of traitors is exactly T. It follows that there are more than $T - z_1$ traitors in the neighbor vertices of X in Zone Z_2. Also, Zone Z_1 contains z_1 traitors, by the invariant of the zone. Thus, the total number of traitors is strictly larger than T, a contradiction. We conclude that the invariant is maintained when a vertex is placed in Zone Z_1.

As long as a vertex remains in Z_1, the value of T does not change, by Step (3) of the algorithm. Thus, the invariant is maintained at all times.

Zone Z_2. Whenever one vertex of a clique is removed in Step (2.1), the other vertex is also removed in Step (2.3), and the value of z_2 is updated accordingly. Thus, the (z_2, z_2)-graph is maintained. No vertex in Z_2 has degree larger than $T - z_1$ after a round, because we remove in Step (2.2) all such vertices from Zone Z_2.

The number of traitors. T is increased only in Step (3). Assume to the contrary that there are only T traitors when T is increased. Then there are z_1 traitors in Zone Z_1 by its invariant and z_2 traitors in Zone Z_2. But then the total number of traitors is at least $z_1 + z_2 = z > T$, a contradiction. We conclude that there are at least $T + 1$ traitors and the increase of T is justified. Thus, there are at least T traitors at all times.

(Note that we cannot conclude that there are z traitors, since a vertex in Z_1 is known to contain a traitor only if the actual number of traitors is exactly T. Since this was found to be untrue as T has increased, we can not assume any more that the vertices in Z_1 contain traitors.) ∎

THEOREM 6.3. *The Degree algorithm locates a traitor in $O(p^3 \log n)$ rounds, using at most $p + 1$ colors. Hence, all p traitors can be located in in $O(p^3 \log n)$ rounds, using at most $p + 1$ colors.*

Proof. The algorithm always uses at most $T + 1 \leq p + 1$ colors, by the calculation in Phase 1 of the algorithm.

We now estimate the number of rounds. The value of T is increased at most p times and never decreases. So it is sufficient to prove that for each value of T, the number of rounds is at most $O(T^2 \log n)$. Since $T \leq p$, the total number of rounds until a traitor is located is $O(p^3 \log n)$.

When T does not increase, the algorithm takes one of the following actions: either (1) a vertex is split or moved to Zone Z_1 and at most $2T$ edges are removed or (2) an edge is added to G. Note that (1) may happen more than once during a single round. The Zone Z_1 has at most T vertices at the end, so the number of times a vertex is moved to Z_1 is at most T plus the number of splits. Similarly as in the previous algorithms, there are at most $O(T \log n)$ splits before T increases. Consequently, the number of steps of type (2) is bounded by the number of edges of G at the end, plus the number of edges removed during the algorithm. This gives the bound of $O(T^2 \log n)$ rounds before T increases. ∎

7. Lower Bounds. We now prove a lower bound on the number of rounds needed by any algorithm that uses $p + c$ colors, assuming that p is not known in advance.

LEMMA 7.1. *Using $p + c$ colors, at least $\Omega(p^2/c)$ rounds are needed to locate p traitors, when p is not known in advance.*

Proof. If $c \geq p$, the bound is true, since at least p rounds are needed to locate p traitors. Let $c < p$ and let $n = p + c + 1$ be the number of users, i.e., there are $c + 1$ innocent users. In any coloring with $p + c$ colors there are at least two users with the same color.

We use an adversary argument. In each round, the pirate transmits a color given to (at least) two users. We prove that the pirate can continue for at least $\Omega(p^2/c)$ rounds, consistently with the fact that there are p traitors. We maintain a graph G with users as vertices, initially with no edges. In each round we add an edge between the two users whose color was given as an answer (if the algorithm colors more than two users by the same color, we choose any two of them; this can only decrease the number of rounds). The pirate can continue as long as there is an independent set of size $c + 1$ in G (Recall that an independent set is a set of vertices with no edges between them.) In this case the pirate's answers are consistent with the p traitors being the vertices not in the independent set.

Let H be the graph on n vertices consisting of c disjoint cliques, each with $\lfloor n/c \rfloor$ or $\lceil n/c \rceil$ vertices, and no other vertices and edges. By Turan's theorem (see [12]) used for the complement graph of G, if G does not contain an independent set of size $c+1$, then G has at least as many edges as H. The number of edges in H is $\Theta(p^2/c)$ (the cliques are non-trivial, since $c < p$). Thus, the pirate can always continue for at least $\Omega(p^2/c)$ rounds. ∎

THEOREM 7.2. *Using $p+c$ colors, at least $\Omega(p^2/c + p \log_{c+1} n)$ rounds are needed to locate p traitors, if p is not known in advance and $n \geq (1 + \varepsilon)c$ for some constant $\varepsilon > 0$.*

Proof. We use an adversary argument, which proceeds in phases. In phase t, $t = 1, \ldots, p$, the pirate has already chosen the location of the first $t - 1$ traitors and is revealing the location of the next traitor. The algorithm knows about the existence of only t traitors, and thus can use $t+c$ colors. The pirate keeps a set A of candidates for the next traitor. At the beginning of each phase, the set A contains all users except for the first $t - 1$ traitors. If any of the first $t - 1$ traitors does not get a unique color, then the pirate broadcasts that color. Otherwise, A is colored by at most $c+1$ colors. Then the pirate broadcasts the color given to most users in A and removes all the other users from A. When the size of A is at most $c + 1$, the pirate fixes one user in A as the next traitor, and a new phase begins. No traitor is located until the end of phase p.

Assume, w.l.o.g., that $\varepsilon < 1$. The number of rounds in phase t is at least $\lceil \log_{c+1}(n + 1 - t) \rceil - 1$. For $t < \varepsilon p/2 \leq \varepsilon n/2$, we have $n + 1 - t > c + 1$ and $n + 1 - t > n/2$. Therefore, the number of rounds in phase t is at least $\Omega(\log_{c+1} n)$

and the total number of rounds is at least $\varepsilon p/2 \cdot \Omega(\log_{c+1} n) = \Omega(p \log_{c+1} n)$. Together with Lemma 7.1 this proves the theorem. □

8. Open Problems. We conclude with a few open problems. First it would be nice to close the small gap of $O(\log c)$ between the Lower and Upper bound in the case that $p+c$ colors are used, and p is not known in advance.

We achieve an asymptotically optimal number of rounds in the Optimal algorithm described in Section 4. However, it is very complex and the hidden constants are large. From a more practical viewpoint, it would be desired to find a simpler optimal algorithm for the $p+1$ case, with possibly better constants.

It may be interesting to investigate randomized algorithms. The first question is what is the proper model in this case. In the deterministic case, the pirate is able to simulate the algorithm. Thus, for example the pirate can deduce completely the distribution of the colors. In the randomized case, it is not realistic to assume this, and the game needs to be specified precisely.

Appendix A. Proof of Lemma 6.1.

LEMMA 6.1 *Let H be a graph with at least $2d$ vertices with degree at least d each. Then there exists a matching in H of size at least d.*

Proof. The proof is by induction. Given a matching of size $c < d$, we show how to find a matching of size $c+1$. Starting at $c = 0$ and repeating this process, we find a matching of size d.

Let v and w be two unmatched vertices of degree at least d each. If some neighbor of v or w is unmatched, then we extend the matching trivially. Otherwise, let $x_1, \ldots, x_d$ be d distinct neighbors of v in H, and $y_1, \ldots, y_d$ their respective neighbors in the matching of cardinality c. Some vertices y_j may be equal to some x_i, but in any case all the vertices y_j are distinct. Out of the $2c$ vertices of the matching of size c there are $2c-d$ vertices that are distinct from all the vertices y_j. Since $c < d$, we have $2c-d < d$. As all the neighbors of w are matched and its degree is at least d, it follows that w is connected to y_j for some j. Replace the edge (x_j, y_j) in the matching by the edges (v, x_j) and (y_j, w). □

Acknowledgments. We are in debt to Amos Fiat, Aria Lev, Tal Rabin, and Tamir Tassa for useful discussions and comments. We would especially like to thank the anonymous referees for the many suggestions and comments they made, which helped to improve the presentation of this paper.

REFERENCES

[1] G. R. Blakley, C. Meadows, G. B. Purdy, *Fingerprinting Long Forgiving Messages,* CRYPTO'85, Springer LNCS 218, 1985, pp. 180–189.

[2] D. Boneh and J. Shaw, *Collusion-Secure Fingerprinting for Digital Data*, IEEE Trans. on Information Theory, Vol. 44, 5, 1998, pp. 1897–1905. (See also CRYPTO'95, Springer LNCS 963, 1995, pp. 452-465).

[3] B. Chor, A. Fiat, and M. Naor, *Tracing Traitors*, CRYPTO'94, Springer LNCS 839, 1994, pp. 257–270.

[4] I.J. Cox, J. Kilian, T. Leighton and T. Shamoon, *A Secure, Robust Watermark for Multimedia*, Information Hiding, Springer LNCS 174, 1996, pp. 185–206.

[5] C. Dwork, J. Lotspiech and M. Naor, *Digital signets: Self-Enforcing Protection of Digital Information*, 28th STOC, 1996, pp. 489-498.

[6] A. Fiat and M. Naor, *Broadcast Encryption*, CRYPTO'93, Springer LNCS 773, 1993, pp. 480–491.

[7] A. Fiat and T. Tassa. *Dynamic Traitor Tracing,* CRYPTO'99, Springer LNCS 1666, 1999, pp. 354–371.

[8] M. Naor and B. Pinkas, *Threshold Traitor Tracing*, CRYPTO'98, Springer LNCS 1462, 1998, pp. 502–517.

[9] B. Pfitzmann, *Trials of Traced Traitors*, Information Hiding, Springer LNCS 1174, 1996, pp. 49–64.

[10] D. R. Stinson and R. Wei, *Combinatorial Properties and Constructions of Traceability Schemes and Frameproof Codes*, SIAM J. on Disc. Math., Vol. 11, 1, 1998, pp. 41–53.

[11] J. Schwenk, J. Ueberberg, *Tracing Traitors using Finite Geometries*, manuscript.

[12] P. Turán, *On an Extremal Problem in Graph Theory*, Math. Fiz. Lapok 48 (1941), pp. 436–452.

[13] N. R. Wagner, *Fingerprinting,* 1983 IEEE Symposium on Security and Privacy, pp. 18–22.