**DIMACS Technical Report 97-41**
**August 1997**

# Reliable Communication over Partially Authenticated Networks[1]

by

Amos Beimel[2] Matthew Franklin[3]

[1]An extended abstract of this paper will appear in the proceedings of the WDAG '97 conference, 1997.

[2]DIMACS, Piscataway, NJ. E-mail: beimel@dimacs.rutgers.edu.
[3]AT&T Labs — Research, Florham Park, NJ. E-mail: franklin@research.att.com.

---

DIMACS is a partnership of Rutgers University, Princeton University, AT&T Labs, Bellcore, and Bell Labs.

DIMACS is an NSF Science and Technology Center, funded under contract STC–91–19999; and also receives support from the New Jersey Commission on Science and Technology.

## ABSTRACT

Reliable communication between parties in a network is a basic requirement for executing any protocol. In this work, we consider the effect on reliable communication when some pairs of parties have common authentication keys. The pairs sharing keys define a natural "authentication graph", which may be quite different from the "communication graph" of the network. We characterize when reliable communication is possible in terms of these two graphs, focusing on the very strong setting of a Byzantine adversary with unlimited computational resources.

**Key Words:** Reliable Communication, Private Communication, Authentication Keys, Graph Connectivity, Byzantine Failures.

# 1 Introduction

Suppose that some processors are connected by a network of reliable channels. All of the processors cooperate to execute some protocol, but some of them are maliciously faulty. Dolev [4] and Dolev et al. [5] proved that if there are $t$ faulty processors, then every pair of processors can communicate reliably if and only if the network is $(2t + 1)$-connected. What happens if we want to tolerate more faulty processors? Adding more reliable channels is costly. We suggest a simpler solution. Let some pairs of processors (other than the pairs connected by channels) share authentication keys, i.e., the means to identify messages from the other. We show how to extend the authentication capabilities of a distributed environment through interaction to enable reliable communication.

Reliable communication involves a transmitter which sends a message to a receiver. There are two versions of this problem: "single-pair" and "all-pairs". For single-pair reliable communication, we specify the transmitter and receiver, and want to succeed against any coalition of at most $t$ faulty processors. For all-pairs reliable communication, we want to succeed for any transmitter, any receiver, and any coalition of at most $t$ faulty processors. The network of channels defines a natural "communication graph", with an edge between two vertices for every channel between two processors. The pairs of parties sharing authentication keys define a natural "authentication graph", with an edge between two vertices for every shared key.

Success for the all-pairs reliable communication problem depends in a natural way on the connectivity of these graphs. We show that all-pairs reliable communication is possible if and only if the communication graph is $(t+1)$-connected and the union of the two graphs is $(2t + 1)$-connected.

When there is a specific sender $a$ and a specific receiver $b$, the situation is not so straightforward. It might seem reasonable to conjecture that reliable communication is possible if and only if the communication graph is $(t + 1)$-connected between $a$ and $b$ and the union of the two graphs is $(2t + 1)$-connected between $a$ and $b$. This condition is necessary, but not sufficient. A natural way to strengthen the condition is to require that the communication graph is $(t + 1)$-connected everywhere, not just between $a$ and $b$. This strengthened condition is sufficient, but not necessary. For example, none of the graphs in Fig. 3 meet the strengthened condition, although all of them support reliable communication from $a$ to $b$ versus one malicious fault. As an exercise to develop intuition about our techniques, the reader is invited to find protocols for these graphs before reading further.

To develop a characterization for single-pair reliable communication, we find it useful to characterize yet a third version of the problem: "single-pair fault-restricted." In this version, we specify the sender $a$, the receiver $b$, and two fault sets $T_0, T_1$. We wish to transmit a message reliably from $a$ to $b$ under the assumption that either $T_0$ or $T_1$ contains *all* of the faulty processors. This version of the problem is not so straightforward either. For example, it might seem reasonable to conjecture that single-pair fault-restricted reliable communication from $a$ to $b$ is possible if and only if it is possible from $b$ to $a$. However, this is not true in general, as the counter-example in Fig. 4 illustrates. Our necessary and sufficient condition for single-pair fault-restricted reliable communication depends critically on a recursively defined graph which includes all of the communication graph and some of the authentication graph.

We also explore reliable and *private* communication, i.e., communication in which the faulty processors learn no information about the transmitted message. All-pairs reliable and private communication is possible whenever all-pairs reliable communication is. Reliable and private communication from $a$ to $b$ is possible whenever we have reliable communication from $a$ to $b$ and from $b$ to $a$. As shown by Ben-Or et al. [1] and Chaum et al. [3], all-pairs reliable and private communication implies the possibility of any secure multiparty computation (when less than a third of the parties are maliciously faulty).

**Historical notes:** The connectivity requirements for several distributed tasks in several models has been studied in many papers; for example Byzantine agreement [4], approximate Byzantine agreement [6], reliable message transmission [4, 5], and reliable and private message transmission [11, 5]. Simple impossibility results and more references can be found in [8]. We mention that in Byzantine agreement all honest parties should agree on the same message while in reliable communication only the transmitter and the receiver agree on the message. To the best of our knowledge, none of the previous papers considered how to strengthen reliability by adding authentication keys.

**Organization:** In Section 2, we describe our model, and supply background definitions. In Section 3, we characterize the all-pairs reliable communication problem. In Section 4, we characterize the single-pair fault-restricted reliable communication problem. In Section 5, we characterize the single-pair reliable communication problem. In Section 6, we consider private communication.

# 2 Preliminaries

## 2.1 The Model

The system is composed of $n$ parties connected by an incomplete network. We describe the network by an undirected graph $G = \langle V, E\rangle$, where $V$ is the set of parties in the network (i.e., $|V| = n$), and $E$ describes the communication channels. That is, there is an edge $\langle u, v\rangle$ in $E$ if and only if there is a communication channel between $u$ and $v$. We assume that these communication channels are reliable: an adversary that does not control $u$ or $v$ (but might control all other vertices in the network) cannot change or delete a message sent on the edge $\langle u, v\rangle$ or insert a message on the channel. We assume that some pairs of parties share authentication keys (to be discussed in Section 2.2). We describe which pairs of parties have a common authentication key by a graph $G_A = \langle V, E_A\rangle$. That is, $u$ and $v$ have a common key, denoted by $k_{u,v}$, if and only if $\langle u, v\rangle \in E_A$. These keys are chosen according to some known probability distribution, and every set of vertices has no information on the keys of disjoint edges (except for their a-priori probability distribution).

We consider protocols for message transmission, in which a transmitter $a \in V$ wants to transmit a message $m$ to a receiver $b \in V$. We assume that the system is synchronous. That is, a protocol proceeds in rounds; at the beginning of each round each party $v \in V$ sends messages to some of its neighbors in the graph $G$. These messages get to the neighbors before the beginning of the next round. The protocol specifies which messages each party sends to its neighbors in each round. The messages sent by a party $v \in V$ depend on a local random input held by $v$, the keys $v$ holds (specified in $G_A$), the messages $v$ got in previous rounds, and the number of the round. The messages that the transmitter sends can also depend on the message $m$. We assume that all parties in the system know the topology of the graphs $G$ and $G_A$. Furthermore, all the parties in the system know in which round party $a$ starts to transmit a message to party $b$.

During the execution there might be Byzantine attacks (also known as "active attacks"). An adversary, with an unlimited power, controls a subset $T$ of the parties. The adversary knows the protocol, the distribution under which the authentication keys where chosen, and the topology of the network (i.e., $G$ and $G_A$). During an execution of the protocol, the adversary can choose $T$ dynamically. The inclusion of a party can be done any time before, during, or after the execution of the protocol. For every party in $T$, the adversary knows all the messages received by that party, its random inputs, and its keys. From the moment a party is included into $T$, the adversary determines the messages this party sends thereafter (possibly deviating from the protocol specification in an arbitrary manner).

**Definition 2.1** *Let $a, b \in V$ be a transmitter and a receiver, and $\mathcal{B}$ be a collection of subsets of $V \setminus \{a, b\}$. We say that a message transmission protocol is* $(\mathcal{B}, \epsilon)$-reliable *if, when the adversary can control any set in $\mathcal{B}$, for every message $m$ the probability that $b$ accepts the message $m$, given that $a$ transmitted $m$, is at least $1 - \epsilon$, where the probability is over the random inputs of the parties, the distribution of the authentication keys, and the random input of the adversary.*

If $b$ does not accept the message transmitted by $a$ then $b$ might accept an incorrect message without knowing it. We say that the protocol is *perfectly detecting* if $b$ never accepts an incorrect message, i.e., $b$ either accepts the correct message or does not accept any message. A message transmission is $(t, \epsilon)$-reliable if it is $(\mathcal{ALL}_t, \epsilon)$-reliable, where $\mathcal{ALL}_t = \{T \subseteq V \setminus \{a, b\} : |T| \leq t\}$.

## 2.2 Authentication Schemes

We briefly define authentication schemes; the reader is referred to [12] for more details. Let $K$ be a finite set, called the set of keys. An authentication scheme is a pair $\langle \text{AUTH}, \mu\rangle$, where

AUTH : $\{0,1\}^* \times K \to \{0,1\}^*$ is a function, and $\mu$ is a probability distribution on the set of keys, $K$. Assume that a pair of parties, Alice and Bob, share a common key $k \in_\mu K$. If Alice wants to authenticate a message $m$, she computes the value $\alpha = \text{AUTH}(m,k)$ and sends to Bob the message $m$ together with $\alpha$. When Bob receives a pair $m, \alpha$, he verifies that the authentication is correct by computing the value $\text{AUTH}(m,k)$ and comparing it to $\alpha$. If they are equal then Bob assumes that the message was transmitted by Alice. The scheme is secure if Eve, knowing $\langle \text{AUTH}, \mu \rangle$ but not $k$, cannot generate a pair which Bob accepts. Formally, the scheme $\langle \text{AUTH}, \mu \rangle$ is called an $(\ell, \epsilon)$-authentication scheme if every probabilistic algorithm that sees any $\ell$ pairs, $\langle m_1, \text{AUTH}(m_1,k)\rangle, \langle m_2, \text{AUTH}(m_2,k)\rangle, \ldots, \langle m_\ell, \text{AUTH}(m_\ell,k)\rangle$, cannot generate, with probability greater than $\epsilon$, a pair $m, \alpha$, such that $m \neq m_i$ for every $i$, and $\alpha = \text{AUTH}(m,k)$, where the probability is over the distribution of the authentication keys, and the random input of the probabilistic algorithm.

### 2.3 Connectivity

The reliability of a network is closely related to its connectivity. In this work we consider *vertex* connectivity of *undirected* graphs. A path $P$ passes through a set $T$ if there is a vertex $u \in T$ in the path. Otherwise, we say that $P$ misses $T$. A set $T \subseteq V \setminus \{u,v\}$ is called a $(u,v)$-separating set if *every* path between $u$ and $v$ passes through $T$. A graph $G = \langle V, E\rangle$ is $(t+1,u,v)$-connected if there is no $(u,v)$-separating set of size at most $t$. A graph $G$ is $t$-connected if it is $(t,u,v)$-connected for every pair of vertices in the graph. Menger [10] proved that a graph $G$ is $(t,u,v)$-connected if and only if $\langle u,v\rangle \in E$ or there exists $t$ vertex disjoint paths between $u$ and $v$. Furthermore, there is an efficient algorithm that checks whether a graph is $(t,u,v)$-connected (for details see e.g. [7]). Notice that if there is an edge between $u$ and $v$ then the graph is $(t,u,v)$-connected for every $t$ and there is no $(u,v)$-separating set.

## 3 All-Pairs Reliable Communication

In this section we characterize when reliable transmission between every pair of parties is possible. We prove that, when some pairs of parties share authentication keys, reliable transmission is possible if and only if the communication graph is $(t+1)$-connected and the union of the communication and authentication graphs is $(2t+1)$-connected.

### 3.1 Sufficiency

**Theorem 3.1** *If the graph $G$ is $(t+1)$-connected and the graph $G \cup G_A$ is $(2t+1)$-connected, then for every $\epsilon > 0$ there is an efficient protocol for $(t,\epsilon)$-reliable message transmission between any pair of parties which uses a $(1, \epsilon/(2nt))$-authentication scheme. Furthermore, the protocol is perfectly detecting.*

We start with a subprotocol for sending messages reliably on edges in $G_A$.

**Lemma 3.2** *If the graph $G$ is $(t+1)$-connected then for every edge $\langle u, v\rangle \in G_A$ and for every $\epsilon > 0$ there exists an efficient subprotocol for $(t, \epsilon/(2n))$-reliably transmitting a message from $u$ to $v$ which uses a $(1, \epsilon/(2nt))$-authentication scheme. Furthermore, this subprotocol is perfectly detecting.*

**Proof:** The subprotocol is as follows. Vertex $u$ authenticates the message with the key $k_{u,v}$ using a $(1, \epsilon/(2nt))$ authentication scheme and sends it to $v$ on $t+1$ disjoint paths in $G$. If $v$ receives an authenticated message $m$ on any one of the $t+1$ paths, and if no message other than $m$ arrives authenticated on any other path, than $v$ accepts $m$ as the received message. Otherwise, $v$ accepts nothing.

Since there are at most $t$ Byzantine vertices, $v$ will receive the right authenticated message on at least one path. Thus $v$ will receive the correct message whenever $v$ does not receive a different authenticated message on another path. The probability that the adversary can cause $v$ to receive a different authenticated message on another path is bounded by $t$ times the probability of the adversary forging a single authenticated message, i.e., $t \cdot \epsilon/(2nt) = \epsilon/(2n)$. If $v$ receives a different authenticated message on another path, then $v$ will receive nothing, and thus the subprotocol is perfectly detecting. □

We can now state the protocol for reliable message transmission. When a transmitter wants to transmit a message to a receiver, it sends the message on $2t+1$ disjoint paths in the graph $G \cup G_A$. That is, the transmitter sends the message to the first vertices in the paths, and each intermediate vertex propagates the messages to the next vertex on the path until it reaches the receiver. To propagate a message on an edge from $G_A$ we use the subprotocol from Lemma 3.2. If a propagation on an edge from $G_A$ fails, then no messages is propagated to the receiver on this path (at least not by honest parties). The receiver accepts a message if it is received on at least $t+1$ of the $2t+1$ paths. Otherwise, the receiver accepts nothing. We complete the proof of Theorem 3.1 by proving that this protocol is reliable (Lemma 3.3) and perfectly detecting (Lemma 3.4).

**Lemma 3.3** *This protocol achieves $(t, \epsilon)$-reliable message transmission between any pair of parties.*

**Proof:** If the adversary has not disrupted one of the subprotocols for sending a message on an edge in $G_A$, then the correct message is received on at least $t+1$ paths. Each vertex, except for the transmitter and receiver, appears at most once on each of the $2t+1$ paths. There are at most $2n$ edges on the union of all the paths. Thus there are at most $2n$ executions of the subprotocol for sending a message on an edge in $G_A$. The probability that the adversary can disrupt at least one of these executions is bounded by $2n$ times the probability that the adversary can disrupt a single execution, i.e., $2n \cdot \epsilon/(2n) = \epsilon$. □

**Lemma 3.4** *This protocol is perfectly detecting.*

**Proof:** The subprotocol for sending a message on an edge in $G_A$ is perfectly detecting. Thus the receiver can only receive an incorrect message on a path in $G \cup G_A$ if there is a faulty vertex on that path, no matter how many edges in $G_A$ are on the path (and no matter how many false messages the adversary has managed to authenticate in the subprotocols). Since the paths are vertex disjoint, the receiver can receive an incorrect message on at most $t$ of the paths, and so will never accept an incorrect message. $\square$

## 3.2 Necessity

The next theorem states that if the sufficient condition does not hold then reliable message transmission is not possible, i.e. it is also a necessary condition. The result does not assume how the common keys are used. That is, even if the keys in $G_A$ are used for purposes other than authentication (e.g., for encryption) then reliable message transmission is not possible. Furthermore, the result does not assume any limit on the size of the keys.

**Lemma 3.5** *If $G$ is not $(t+1)$-connected and $\epsilon < 1/2$, then there exists a transmitter and receiver for which $(t, \epsilon)$-reliable transmission is not possible.*

**Proof:** If $G$ is not $(t+1)$-connected, then there is a subset $T$ of $t$ vertices that separates some pair $a, b$. Suppose that $a$ wishes to transmit a message to the receiver $b$. If the vertices in $T$ crash-fail then $b$ would not get any information about the message transmitted by $a$. Assuming the message space has at least two possible messages, for at least one message a guess by $b$ succeeds with probability at most $1/2$. $\square$

**Lemma 3.6** *Suppose that $G$ is $(t+1)$-connected but $G \cup G_A$ is not $(2t+1)$-connected. Then there exists vertices $a, b$ and subsets of vertices $T_0, T_1$ of size $t$ such that for every pair of messages $m_0, m_1$ the receiver $b$ cannot distinguish whether the set $T_0$ is Byzantine and the message transmitted by $a$ is $m_0$ or the set $T_1$ is Byzantine and the message transmitted by $a$ is $m_1$.*

**Proof:** Since $G \cup G_A$ is not $(2t+1)$-connected there exists vertices $a, b$ and an $(a, b)$-separating set $T$ of size at most $2t$ in $G \cup G_A$. Define $B$ to be the set of vertices which have a path to $b$ in $G \cup G_A$ that misses $T$ (i.e., $v \in B$ if $T$ is not a $(v, b)$-separating set in $G \cup G_A$). Define $A$ to be $V \setminus (B \cup T)$. Finally, let $T_0, T_1$ be an arbitrary partition of $T$ into two disjoint sets of size at most $t$. Fig. 1 describes these sets. Notice that there are no edges between $A$ and $B$ since $T$ is a separating set in $G \cup G_A$.

Fix two messages $m_0 \neq m_1$ arbitrarily, and define two executions 0 and 1 as follows. In execution $i \in \{0, 1\}$, the Byzantine set is $T_i$, the message $m_i$ is transmitted by $a$, the random inputs of the vertices in $A \cup T_{\bar{i}}$ are $\{r_u^i : u \in A \cup T_{\bar{i}}\}$, and the authentication keys of pairs in $A \cup T$ are $\left\{k_{u,v}^i : u, v \in A \cup T, \langle u, v \rangle \in E_A\right\}$. The behavior of the Byzantine set $T_i$ in

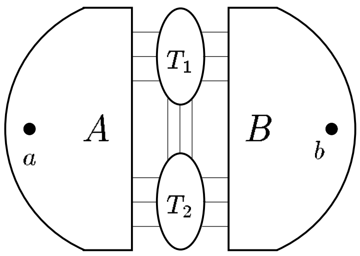


Figure 1: A description of the partition of the vertices.

execution $i$ is to send no messages whatsoever to $A \cup T_{\bar{i}}$, and to send to $B$ exactly the same messages that are sent to $B$ by the (honest) $T_i$ in execution $\bar{i}$.

In order for the Byzantine $T_i$ to behave as specified in execution $i$, the adversary needs to simulate the behavior of $A \cup T_i$ for execution $\bar{i}$. We show how this can be done. The adversary simulates, round by round, the behavior of the vertices in $A \cup T_i$ for execution $\bar{i}$, using $\left\{r_u^{\bar{i}} : u \in A \cup T_i\right\}$ as the random inputs for $A \cup T_i$, and $\left\{k_{u,v}^{\bar{i}} : u, v \in A \cup T, \langle u, v\rangle \in E_A\right\}$ as the authentication keys for pairs of parties in $A \cup T$. At the beginning of each round, each simulated party has a history of messages that it got in the simulation of the previous rounds, its simulated local random input, and its simulated keys. The simulated party sends during the simulation the same messages that the honest party would send in the original protocol in the same state. The simulated messages that $T_i$ sends to $B$ are really sent by the parties. All other messages are used only to update the history for the next round. The messages which are added to the history of each simulated vertex are the real messages that are sent by the parties in $B$ and the simulated messages that are sent by the vertices in $A \cup T_i$. No messages from $T_{\bar{i}}$ are added to the history. These two executions are described in Fig. 2.

The history of messages of each simulated vertex in execution $i$ is the same as the history of the vertex in execution $\bar{i}$. Therefore, the messages sent by $T_0$ and $T_1$ to members of $B$ in both executions are exactly the same. Thus the parties in $B$ cannot distinguish whether the set $T_0$ is Byzantine and the message transmitted by $a$ is $m_0$, or the set $T_1$ is Byzantine and the message transmitted by $a$ is $m_1$. Since $b \in B$, the receiver $b$ cannot distinguish these cases either. □

**Theorem 3.7** *If $(t, \epsilon)$-reliable transmission is possible between any pair of parties with $\epsilon < 1/2$, then the graph $G$ is $(t+1)$-connected and the graph $G \cup G_A$ is $(2t+1)$-connected.*

**Proof:** Consider all pairs of the above executions, where the random inputs $\{r_u^0 : u \in A \cup T_1\}$, $\{r_u^1 : u \in A \cup T_0\}$, and keys $\left\{k_{u,v}^0 : u, v \in A \cup T, \langle u, v\rangle \in E_A\right\}$, $\left\{k_{u,v}^1 : u, v \in A \cup T, \langle u, v\rangle \in E_A\right\}$ range over all their possible values. In each pair of executions, whenever $b$ is correct in one

| | Execution 0 | Execution 1 |
|---|---|---|
| Byzantine set | $T_0$ | $T_1$ |
| Real execution of $A$ | | |
| Real message | $m_0$ | $m_1$ |
| Random inputs | $\{r_u^0 : u \in A \cup T_1\}$ | $\{r_u^1 : u \in A \cup T_0\}$ |
| Keys of $A \cup T$ | $\left\{k_{u,v}^0 : u, v \in A \cup T, \langle u, v\rangle \in E_A\right\}$ | $\left\{k_{u,v}^1 : u, v \in A \cup T, \langle u, v\rangle \in E_A\right\}$ |
| Behavior | $T_0$ sends no messages to $A \cup T_1$ | $T_1$ sends no messages to $A \cup T_0$ |
| Simulation of $A$ by the Byzantine set | | |
| Simulated message | $m_1$ | $m_0$ |
| Random inputs | $\{r_u^1 : u \in A \cup T_0\}$ | $\{r_u^0 : u \in A \cup T_1\}$ |
| Keys of $A \cup T$ | $\left\{k_{u,v}^1 : u, v \in A \cup T, \langle u, v\rangle \in E_A\right\}$ | $\left\{k_{u,v}^0 : u, v \in A \cup T, \langle u, v\rangle \in E_A\right\}$ |
| Behavior | $T_1$ sends no messages to $A \cup T_0$ | $T_0$ sends no messages to $A \cup T_1$ |

Figure 2: The two executions that confuse $b$.

execution it errs in the second. Thus, in any strategy by $b$ for choosing whether to receive $m_0$ or $m_1$ there is some $i$ such that when $m_i$ is transmitted, the receiver accepts $m_i$ with probability at most 1/2.

Consider the following adversary. The adversary chooses $T_i$ as the Byzantine set, chooses the random local inputs for the parties in $A \cup T_0$, and chooses keys for pairs of parties in $A \cup T$ according to the same distribution that the real keys are distributed. (The adversary does not try to guess the actual keys and random inputs that parties in $A \cup T$ have, but only chooses them from the same distribution which they are chosen in the real execution.) The adversary simulates the parties in $A$ with $m_{\bar{i}}$, the keys and the local random inputs it chose. When $a$ transmits the message $m_i$, the receiver $b$ errs with probability at least half. Thus, if $G \cup G_A$ is not $(2t+1)$-connected then every scheme has reliability of at least 1/2. $\square$

# 4 Single-Pair Fault-Restricted Communication

We now prove a characterization of single-pair fault-restricted reliable communication. That is, the sender $a$ transmits a message to the receiver $b$ when one of two given sets $T_0, T_1$ is guaranteed to contain all of the faulty processors. The proof of sufficiency is constructive, i.e., we present a protocol that achieves reliable communication whenever it is possible.

## 4.1 Ideas of the Protocol

In this section we informally present the ideas of the protocol, and try to motivate the definition of a graph $G^*$ which is used in the characterization. We sketch a series of protocols, relying on increasingly weaker assumptions, and ending with our actual protocol. For simplicity, let us assume that the adversary can never authenticate false messages unless it holds the authentication key (ignoring for the moment the negligible probability that this assumption is violated).

**First Protocol:** Suppose that $G$ contains one path from $a$ to $b$ that misses $T_0$ and another path that misses $T_1$. If $a$ and $b$ shared an authentication key, the protocol could send $m, \alpha$ on both paths, where $\alpha = \text{AUTH}(m, k_{a,b})$. Then $b$ succeeds by accepting any message that arrives with the proper authentication. The First Protocol succeeds whenever $\langle a, b\rangle \in G_A$ and neither $T_0$ nor $T_1$ separates $a, b$ in $G$.

**Failure of the First Protocol:** Consider Graph 1 in Fig. 3. The First Protocol fails for this graph, because $a$ and $b$ do not share an authentication key. However, we can succeed for this example as follows. Suppose $a$ authenticates $m$ with $k_{a,w}$ and sends it to $w$ through $T_1$. If $w$ receives a message with the proper authentication, then $w$ authenticates it with $k_{w,b}$ and sends it to $b$ through $T_1$; otherwise $w$ sends nothing. If $T_1$ changes or removes an authenticated message during the protocol, then we say that $T_1$ has "disabled" the corresponding authentication edge. Either $b$ will receive a properly authenticated message along this "path," and will accept it, or $b$ will know that $T_1$ has disabled at least one authentication edge. Whenever $b$ knows that $T_1$ is at fault, $b$ accepts whatever was received along the path through $T_0$. This is the intuition behind our Second Protocol.

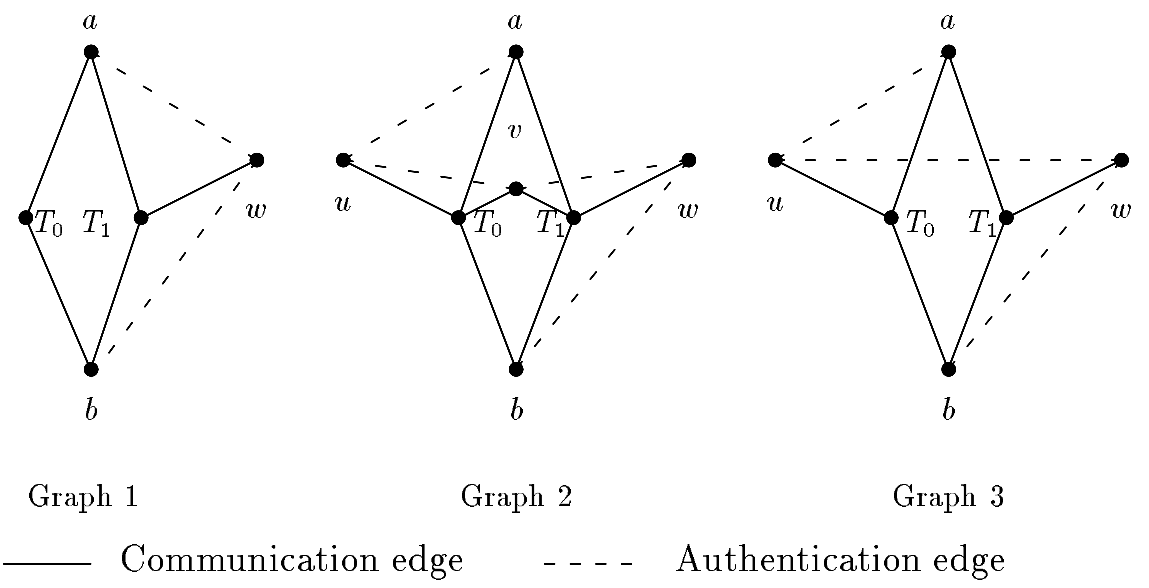


Figure 3: Examples of graphs in which $a$ can $(1, \epsilon)$-reliably transmit a message to $b$.

**Second Protocol:** Consider the graph $G'$ obtained from $G \cup G_A$ by removing all authentication edges $\langle u, v\rangle$ such that $T_0$ is a $(u, v)$-separating set in $G$. Suppose that $G$ contains a path that misses $T_1$, while $G'$ contains a path that misses both $T_0$ and $T_1$. The transmitter "sends" the message to the receiver on both paths. To "send" over an authentication edge means to authenticate with the corresponding key and send on a path in $G$ that misses $T_0$. The existence of such a path is guaranteed by the definition of $G'$. At the end of the execution of the protocol, if the receiver has gotten a message on the path in $G'$ that misses both $T_0$ and $T_1$, then this message is accepted. Otherwise, the receiver accepts the message sent along the path in $G$ that misses $T_1$.

To summarize, the idea of this protocol is to get information on the Byzantine set from the fact that a message was not received on a certain path. Either the Byzantine set does not disable an edge and the receiver learns the message, or the Byzantine set does disable an edge and the receiver learns which set is Byzantine.

**Failure of the Second Protocol:** Consider Graph 2 in Fig. 3. The Second Protocol fails for this graph, because there is no path in $G'$ that misses both $T_0$ and $T_1$. However, we can succeed for this example as follows. The message is sent along the two simple paths from $a$ to $b$ in $G$, one through $T_0$ and one through $T_1$. In addition, the message is sent along the "authentication path" through $u$, $v$, and $w$. This authentication path will either deliver the correct message, or it will expose the Byzantine set and allow $b$ to choose between the messages received along the simple paths. For the first hop of the authentication path, $a$ authenticates the message with $k_{a,u}$ and sends it to $u$ through $T_0$. Either $u$ receives a properly authenticated message, or $u$ learns that $T_0$ is Byzantine. (Note that $u$ will know when an expected message does not arrive, because we are assuming a synchronous system in which all parties know when the protocol has started.) Now the edge $\langle u, v\rangle$ is traversed, using the key $k_{u,v}$ to deliver some authenticated message from $u$ to $v$ through $T_0$. This message will either be the same message that $u$ received or the new message "$T_0$ is Byzantine" depending on what $u$ learned. The remaining hops of the authentication path proceed similarly. Eventually, $b$ will either receive a properly authenticated message under $k_{w,b}$ from $w$ through $T_1$ or $b$ will know that $T_1$ is Byzantine. If $b$ receives a properly authenticated message, then that message will either be the correct message from $a$ or it will expose the Byzantine set.

**Third Protocol:** Consider the graph $G''$ obtained from $G \cup G_A$ by removing all authentication edges $\langle u, v\rangle$ such that each of $T_0$ and $T_1$ is a $(u, v)$-separating set in $G$. Suppose that $G$ contains a path that misses $T_0$ and a path that misses $T_1$, while $G''$ contains a path that misses both $T_0$ and $T_1$. The transmitter "sends" the message to $b$ on all three paths. To "send" along an edge $\langle u, v\rangle \in G_A$, the vertex $u$ sends the authenticated message $\langle m, \textsc{auth}(m, k_{u,v})\rangle$ on a path in $G$ that either misses $T_0$ or misses $T_1$. Such a path exists by the definition of $G''$, and we can assume that all parties including $b$ know what this path is. If $v$ does not receive a properly authenticated message, then $v$ sends an alert message "$v$

has not received a message from $u$" to $b$. This alert message is propagated to $b$ along the path in $G''$ in the same manner as the regular message. Eventually, either $b$ will receive the correct message on the path in $G''$, or the Byzantine set will disable at least one edge and $b$ will receive an appropriate alert message. From this information, $b$ will know which message to accept from the simple paths in $G$.

**Failure of the Third Protocol:** Consider Graph 3 in Fig. 3. The Third Protocol fails for this graph, because the edge $\langle u, w\rangle$ does not get included in $G''$. That is, both $T_0$ and $T_1$ disconnect $u$ and $w$ in $G$. However, $T_1$ does not disconnect $u$ and $w$ if we are allowed to use $\langle b, w\rangle$. By the previous analysis, we ought to be allowed to use $\langle b, w\rangle$, since it cannot be disabled without implicating $T_1$.

**Final Protocol:** The additional idea in this protocol is that if every $T_i$ that can disable an edge is caught, then this edge can be considered as a reliable edge. That is, when checking if another authentication edge is reliable we use the edges of $G$ as well as those authentication edges which we already established as reliable. This motivates the recursive definition of $G^*$ in the next section.

## 4.2 Characterization of Single-Pair Fault-Restricted Communication

The intuition behind the definition of the graph $G^*$ is that it contains all the edges that cannot be disabled without $b$ learning which set disabled them. We formally define $G^*$, and from $G^*$ the notion of a "confusing pair":

**Definition 4.1** *Let $a, b \in V$ be the transmitter and receiver, and $T_0, T_1 \subseteq V \setminus \{a, b\}$ be a pair of sets. For any $v \in V$, let $[v]$ denote the connected component in $G \setminus (T_0 \cup T_1)$ that contains $v$. Let $E_1 \subseteq E_2 \subseteq \ldots$ be a sequence of subsets of authentication edges, such that $\langle u, v\rangle \in E_i$ whenever the following is true:*

1. *$\langle u, v\rangle \in E_A$.*
2. *$u, v \notin T_0 \cup T_1$.*
3. *The vertex $v$ is in the connected component of $b$ in the graph $\langle V \setminus (T_0 \cup T_1), E \cup E_{i-1}\rangle$. That is, $v \in [b] \cup \{[u'] : \langle u', v'\rangle \in E_{i-1}\}$*
4. *At most one of $T_0$ or $T_1$ is a $(u, v)$-separating set in $G \cup E_{i-1}$.*

*If $E_{k+1} = E_k$ then $G^* = \langle V, E^*\rangle = \langle V, E \cup E_k\rangle$.*

The graph $G^*$ depends upon $a$, $b$, $T_0$ and $T_1$. We use the notation $G^*_{a,b}$ when $a$ and $b$ are not clear ($T_0$ and $T_1$ will be clear throughout the paper). For example, consider Graph 3 described in Fig. 3. We first add the edge $\langle w, b\rangle$ since $T_0$ is not a separating set in $G$. Now, the set $T_1$ is not a $(u, w)$-separating set in the current graph and therefore $\langle u, w\rangle$ is added, and finally $\langle a, u\rangle$ is added. Thus, the Final protocol succeeds in this graph.

**Definition 4.2** *Let $T_0, T_1 \subseteq V \setminus \{a, b\}$. The sets $T_0$ and $T_1$ are an $a, b$-*confusing pair *if*

1. *Either $T_0$ or $T_1$ is an $(a, b)$-separating set in $G$, or*

2. *The set $T_0 \cup T_1$ is an $(a, b)$-separating set in $G^*$*

Given these definitions, our characterization can now be stated formally.

**Theorem 4.3** *Let $T_0, T_1 \subseteq V \setminus \{a, b\}$ such that $\langle T_0, T_1\rangle$ is not a confusing pair. Then, for every $\epsilon > 0$, there is a protocol for $(\{T_0, T_1\}, \epsilon)$-reliable message transmission from $a$ to $b$.*

**Theorem 4.4** *If $\langle T_0, T_1\rangle$ is a confusing pair then $(\{T_0, T_1\}, \epsilon)$-reliable message transmission between $a$ and $b$ is not possible with $\epsilon < 1/2$.*

The protocol for proving Theorem 4.4 is described in Section 4.3, and the protocol is analyzed in Section 4.4. The proof of Theorem 4.4 is given in Section 4.5. We first discuss the properties of $G^*$. For every $G$ and $G_A$ it can be shown that $E_{k+1} = E_k$ for every $k \geq n$.

**Lemma 4.5** *$E_k = E_n$ for every $k \geq n$ and for every $G$ and $G_A$.*

**Proof:** Denote $B_i$ as the set of vertices in the connected component of $b$ in $\langle V \setminus (T_0 \cup T_1), E \cup E_i\rangle$. If $B_i = B_{i-1}$ then $E_k = E_i$ for every $k \geq i$ (since $\langle u, v\rangle \in E_A$ is added to $E_k$ if and only if there exists a path in $G$, which misses some $T_i$, from $u$ to some $w \in B_{k-1}$). Since $B_i \subseteq V$, the size of $B_i$ can increase at most $n - 1$ times and the lemma follows. □

One might think that if $a$ can transmit a message to $b$ then $b$ can transmit a message to $a$. However, the definition of $G^*$ is not symmetric with respect to $a$ and $b$, and in Fig. 4 we show a graph in which $a$ can transmit a message to $b$ but $b$ cannot transmit a message to $a$.

## 4.3 Protocol for Proving Sufficiency

First we present Protocol SEND, described in Fig. 5, which sends a message along a path in $G^*$. In this protocol either $v$ accepts the message sent by $u$ or the receiver $b$ learns which set is Byzantine. For this protocol we need the following notation. The *level* of an edge $\langle u, v\rangle \in G^*$ is the minimum $j$ such that $\langle u, v\rangle \in E_j$. If $\langle u, v\rangle \in E$ then its level is 0. The level of a path is the maximum level of an edge on the path. For every $\langle u, v\rangle \in E_j \setminus E$ we fix a simple path between $u$ and $v$ in $G^*$, denoted by $\text{PATH}(u, v)$, such that the level of the path

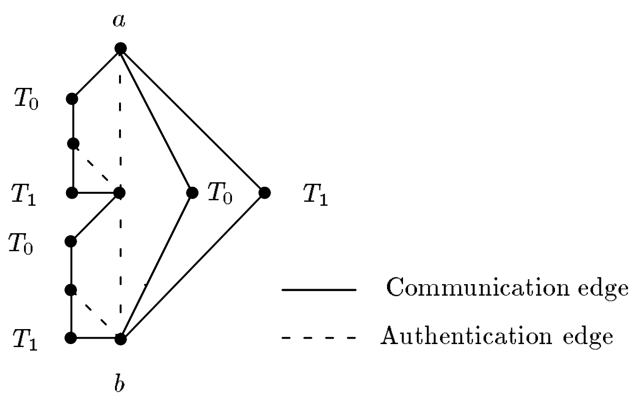


Figure 4: A graph in which $a$ can transmit a message to $b$, but $b$ cannot transmit a message to $a$.

is less than the level of the edge and the path either misses $T_0$ or misses $T_1$. By Definition 4.1 such a path exists. Furthermore, for every $v$ that is adjacent to an authentication edge $\langle u, v\rangle$ in $G^*$, define $\text{PATH}(v, b)$ as a fixed path with minimum level among all the paths from $v$ to $b$ in $G^*$ which miss $T_0 \cup T_1$. Again by Definition 4.1 such path exists, and its level is at most the level of $\langle u, v\rangle$. If the level of every path from $v$ to $b$ in $G^*$ is at least the level of $\langle u, v\rangle$ (thus, $u$ was added to the connected component of $b$ in the graph $\langle V \setminus (T_0 \cup T_1), E \cup E_{i-1}\rangle$ before $v$) then $\text{PATH}(v, b) \stackrel{\text{def}}{=} v \circ \text{PATH}(u, b)$ (where $\circ$ is concatenation of paths). All the parties in $V$ know $\text{PATH}(u, v)$ and this is part of the specification of the protocol. In Protocol SEND, described in Fig. 5, there are two recursive calls to SEND. The call at line (1) is part of the propagation of the message on $P$ and is called an internal call. The call at line (3) informs the receiver $b$ whether $v_{i+1}$ has accepted a message, and is called an alert call. The alert call can be executed in parallel to the rest of the protocol.

In Fig. 6 we describe Protocol TRANSMIT in which $a$ transmits a message to $b$.

## 4.4 Proof of Sufficiency

We first establish that the execution of $\text{SEND}(m, \text{PATH}(u, v), \text{Destb})$ always terminates, although it might take as many as $n^n$ rounds.

**Lemma 4.6** *Let $j$ be the level of $\text{PATH}(u, v)$. The execution of $\text{SEND}(m, \text{PATH}(u, v), \text{Destb})$ terminates after at most $O(n^{j+1}) = 2^{O(n \log n)}$ rounds.*

**Proof:** The proof is by induction on the level of $\text{PATH}(u, v)$. For the base case $j = 0$, $O(n)$ rounds suffice for sending a message along $\text{PATH}(u, v)$ which is a simple path in $G$. For the induction step, observe that the alert recursive calls are executed in parallel to the rest of the execution. Thus, the number of rounds in which the protocol terminates is the sum of the number of rounds of each internal recursive call and the maximum number of rounds required for an alert recursive call. The level of every path in a recursive internal call is at most $j-1$, where $j$ is the level of $\text{PATH}(u,v)$. Thus, by the induction hypothesis, each recursive internal call takes $n^j$ rounds, and the execution of $\text{PATH}(u,v)$ terminates after at most $n \cdot n^j = n^{j+1}$ rounds since the length of $\text{PATH}(u,v)$ in $G^*$ is at most $n$.

Now examine the time required for each alert call for an authentication edge $\langle v_i, v_{i+1}\rangle$ in $\text{PATH}(u,v)$. If the level of $\text{PATH}(v_{i+1},b)$ is at most $j-1$ then the message is propagated to $b$ after at most $n^j$ rounds. Otherwise, $\text{PATH}(v_{i+1},b) = v_{i+1}\circ\text{PATH}(v_i,b)$, thus, $v_i$ will get the message (on $\text{PATH}(v_{i+1},v_i)$) after at most $n^j$ rounds, and the message is propagated to $b$ after at $n^j$ additional rounds. We conclude that all messages sent in the execution of $\text{PATH}(u,v)$ (caused by internal and alert recursive calls) are performed at most $O(n^{j+1})$ rounds after the execution has started. By Lemma 4.6, this quantity is bounded by $2^{O(n\log n)}$. □

We next prove that in the execution of Protocol SEND either the correct message is accepted, or $b$ can identify the Byzantine set. Intuitively, if $b$ has not received the message on $\text{PATH}(v,b)$ then there was at least one authentication edge $\langle u,v\rangle$ that was disabled. Consider the first such edge. Then $b$ should receive a message saying that "v has not accepted a message from u". From all previous edges $b$ should receive a message saying that the message was received. The receiver $b$ know that $\text{PATH}(u,v)$ misses $T_i$, and thus $T_{\bar{i}}$ must be Byzantine.

**Lemma 4.7** *Assume that $v$ executes* $\text{SEND}(m, \text{PATH}(v,b), \text{TRUE})$*, where $m$ is either an alert message or a regular message. If the adversary has not authenticated any message without having the authentication key, then either $b$ accepts the message sent by $v$ or $b$ can identify which set is Byzantine.*

**Proof:** If $b$ receives a message on $\text{PATH}(v,b)$ then it is the sent by $v$ (since $\text{PATH}(v,b)$ misses $T_) \cup T_1$). We prove by induction on the level of $\text{PATH}(v,b)$, that if $b$ does not receive a message on the path then $b$ can identify the Byzantine set. If the level of $\text{PATH}(v,b)$ is zero then this is a path in $G$ that misses all possible Byzantine parties in $T_0 \cup T_1$, thus $b$ receives the message $m$ on this path.

Now, let $\text{PATH}(v,b)$ be a path of level $j > 0$. Since alert messages are only sent during recursive calls (when Destb = FALSE), if an alert message is sent on $\text{PATH}(v_{i+1},b)$ then the level of $\text{PATH}(v_{i+1},b)$ is at most $j-1$. If $b$ does not receive an alert message then, by the induction hypothesis, $b$ can identify the Byzantine set. Thus, we assume that $b$ receives all alert messages.

Consider all the alert messages that the receiver $b$ received during the execution of $\text{SEND}(m, \text{PATH}(v,b), \text{TRUE})$ (excluding the alert messages resulting from alert recursive calls). By the induction hypothesis, if $b$ receives some alert message from a vertex $v_{i+1}$ then this is the message sent by $v_{i+1}$. By Definition 4.1, for every authentication edge $\langle v_i, v_{i+1}\rangle \in G^*$, we have that $v_i, v_{i+1} \notin T_0 \cup T_1$. Thus, an alert message claims that "$v_{i+1}$ has accepted a message from $v_i$" if and only if $v_{i+1}$ has really accepted a message from $v_i$.

Assume that all alert messages that $b$ received are of the type "$v$ has accepted a message from $u$". Then no authentication edge was disabled during the recursive calls and the only authentication edges that were disabled are on $\text{PATH}(v,b)$. In this case the message that $b$ receives on $\text{PATH}(v,b)$ is "$v_{i+1}$ has not accepted a message from $v_i$" for some $v_i, v_{i+1}$ on $\text{PATH}(v,b)$, and this message can be trusted. There is only one set that hits $\text{PATH}(v_i, v_{i+1})$ and $b$ learns that this set must be Byzantine. Otherwise, let $\langle v_i, v_{i+1}\rangle$ be the first edge for which $b$ receives an alert message claiming that "$v_{i+1}$ has not accepted a message from $v_i$". No edge used in the recursive call during the execution of $\text{SEND}(m, \text{PATH}(v_i, v_{i+1}), \text{FALSE})$ was disabled, and the edge $\langle v_i, v_{i+1}\rangle$ was disabled by a vertex on $\text{PATH}(v_i, v_{i+1})$ which did not propagate the message it got. Again, there is only one set that hits $\text{PATH}(v_i, v_{i+1})$ and $b$ learns that this set must be Byzantine. □

We are ready to prove the sufficient condition:

**Proof:** (Theorem 4.3) The protocol described in Fig. 6 suffices. First notice that by Definition 4.2 the paths $P_0, P_1$ and $\text{PATH}(a,b)$ exist. By Lemma 4.7, if the adversary has not authenticated a false message, $b$ receives the message transmitted by $a$. Let $\ell$ be the maximum number of times that any single authentication key is used, and let $k$ be the total number of times that all of the authentication keys are used; by Lemma 4.6 both of these values are well-defined. Then, using an $(\ell, \epsilon/k)$-authentication scheme will ensure that the probability that the adversary has authenticated a false message is at most $\epsilon$. □

## 4.5 Proof of Necessity

We next prove the necessary condition, that is, if $\langle T_0, T_1\rangle$ is a confusing pair in $G^*$ then reliable transmission is not possible.

**Proof:** (Theorem 4.4) If $T_0$ or $T_1$ is an $(a,b)$-separating set in $G$ then when they crash-fail no message transmission is possible. Otherwise, the proof is a generalization of the proof of Theorem 3.7. We only describe the modifications to that proof. Unlike the proof of Theorem 3.7, $T_0 \cup T_1$ is not a separating set in $G \cup G_A$ but rather in $G^*$. That is, we have to deal with the edges that we deleted from $G^*$. The sets $A$ and $B$ are defined with respect to $G^*$. Formally, define $B$ as the set of vertices that have a path in $G^*$ to $b$ that does not use any vertex from $T_0 \cup T_1$. That is, $B = \{v \in V \setminus T_0 \cup T_1 \ : \ T_0 \cup T_1 \text{ is not a } (v,b)\text{-separating set in } G^*\}$. Furthermore, let $A \stackrel{\text{def}}{=} V \setminus (B \cup T_0 \cup T_1)$. Since $\langle T_0, T_1\rangle$ is a confusing pair, $a \in A$.

The strategy for the adversary is defined as in the proof of Theorem 3.7. There are two problems. The first problem is that $T_0$ and $T_1$ can intersect. Therefore, during the simulation and the protocol the parties in $T_0 \cap T_1$ do not send any messages. The second problem is that while simulating the parties in $A$, the adversary does not know the keys of edges $\langle u,v\rangle \in E_A$, where $u \in A$ and $v \in B$. This seems to be a problem, since the parties in $B$ know these keys and can distinguish if different keys are used during the simulation. However, $T_0 \cup T_1$ is a $(a,b)$-separating set in $G^*$, and any edge $\langle u,v\rangle \in E_A$ between $A$ and $B$ is an authentication edge such that for every $w \in B$ both $T_0$ and $T_1$ are a $(u,w)$-separating set in $G$ (since $\langle u, v\rangle \in E_A \setminus E^*$). Therefore, both sets will disable this edge, and the vertices in $B$ might know that the correct key $k_{u,v}$ is not used, but will not be able to know which set to blame.

More formally, define a set $C \subseteq A$, where $u \in C$ if for every $v \in B$ both $T_0$ and $T_1$ are a $(u, v)$-separating set in $G$. Consider an execution of a protocol, where the adversary sends messages according to the strategy defined in the proof of Theorem 3.7. We claim that the messages that are sent by parties in $C$ do not influence $B$. If $u \in C$ can influence a party $v \in B$, there must be a path $u, w_1, \ldots, w_{j-1}, w_j = v$ in $G$ such that $u$ sends a message to $w_1$, then $w_1$ sends a message to $w_2$ that depends on the message that it got from $u$ and so on until a message gets to $v$. This path must pass through $T_i$ (the Byzantine set). Let $w_j$ be the first vertex from $T_i$ on the path (i.e., $u, w_1, \ldots, w_{j-1} \in A \cup T_{\bar{i}}$). According to the strategy of the adversary, $w_j$ ignores the messages that it gets from parties in $A \cup T_{\bar{i}}$, and thus the messages that it sends do not depend on the message sent by $u$. Similarly, the messages sent by $u \in C$ when $T_i$ simulates the parties in $A$ do not influence the messages it sends to $B$, since the path from $u$ to a vertex in $B$ passes through at least one vertex in $T_{\bar{i}}$. No messages are sent by parties in $T_{\bar{i}}$ during the simulation, and so the simulated messages received by parties in $T_{\bar{i}}$ influence neither the parties in $A \cup T_i$ nor the parties in $B$, and can be ignored. Therefore, the same arguments as in the proof of Theorem 3.7 complete the proof of Theorem 4.3. $\square$

# 5 Single-Pair Reliable Communication

In this section we consider $(t, \epsilon)$-reliable transmission between a given pair of parties $a$ and $b$. We first show that if there is a fault restricted protocol for every pair of sets of size $t$ then there is a protocol that is reliable against every set of size at most $t$.

**Lemma 5.1** *If for every pair of sets $\langle T_0, T_1\rangle$, where $|T_0|, |T_1| = t$ and $T_0, T_1 \subseteq V \setminus \{a, b\}$, the transmitter $a$ can $(\{T_0, T_1\}, \epsilon)$-reliably transmit a message to $b$, then there is a $(t, \epsilon \cdot \binom{n}{t})$-reliable protocol in which $a$ transmits a message to $b$.*

**Proof:** For every pair of sets $\langle T_0, T_1\rangle$, of size $t$, the transmitter executes the $(\{T_0, T_1\}, \epsilon)$ reliable protocol. Since the Byzantine set is not necessarily contained in $T_0$ or in $T_1$, the receiver $b$ might not accept any message in the execution. The receiver accepts a message $m'$ such that there is a set $T$ of size $t$, for which $b$ accepts the message $m$ in the $(\{T, T'\}, \epsilon)$ reliable protocol for every $T'$ of size $t$.

Let $T_0$ be a set of size $t$ that contains all Byzantine parties in the execution. The probability that one of the $(\{T_0, T_1\}, \epsilon)$ reliable protocols has failed is at most $\epsilon \cdot \binom{n}{t}$. If none of these executions has failed then the combined protocol succeeds. For every $T_1$, the receiver $b$ receives the message $m$ in the protocol for $\langle T_0, T_1\rangle$. Assume that there is a set $T'_0$ such that the receiver $b$ receives the same message $m'$ in the protocol for $\langle T'_0, T_1\rangle$ for every $T_1$. In particular $b$ receives the message $m'$ in the protocol for $\langle T_0, T'_0\rangle$, and $m = m'$. $\square$

Using the fault-restricted characterization from the preceding section, we prove the following:

**Theorem 5.2** *There is a $(t,\epsilon)$-reliable protocol in which vertex $a$ can transmit a message to vertex $b$ if and only if for every $T_0, T_1 \subseteq V \setminus \{a,b\}$ such that $|T_0|, |T_1| = t$, the set $T_0 \cup T_1$ is not an $(a,b)$-separating set in $G^*$.*

**Proof:** If there are sets $T_0, T_1 \subset V \setminus \{a,b\}$ such that $|T_0|, |T_1| = t$ and the set $T_0 \cup T_1$ is an $(a,b)$-separating set in $G^*$, then by Theorem 4.4 the receiver $a$ cannot transmit a message to $b$ with reliability less than 1/2. On the other hand, if the conditions of the theorem hold then $G$ is $(t+1,a,b)$-connected. By Lemma 5.1 and Theorem 4.4 $a$ can transmit a message to $b$. □

# 6 Privacy from Reliability

In this section we consider private and reliable communication, i.e., transmission in which the adversary should not learn any information about the transmitted message. For this, we have to assume that the reliable channels are private: the adversary has no information on the messages sent on the channel $\langle u,v \rangle$ unless it controls $u$ or $v$. We present a general protocol for private and reliable transmission that succeeds if $G$ is $(t+1,a,b)$-connected and reliable communication is possible. This protocol is also presented in [9]. We use this protocol to show that for both all-pairs and single-pair, the sufficiency condition for reliable communication from $a$ to $b$ and from $b$ to $a$ is a sufficient condition for reliable communication with privacy.

We formally define private transmission. In the following definition $\mathcal{C}_T$ denotes the communication received by a set $T$ during an execution of the protocol. This communication is a function of the transmitted message, the random inputs of the vertices, and the authentication keys. If $T$ is Byzantine then $\mathcal{C}_T$ also depends on the strategy of the adversary. Informally, a protocol is perfectly private if the probably distribution of $\mathcal{C}_T$ is independent of the message. The protocol is $\delta$-private if for every two messages $m_0, m_1$, the distance between the probability distribution of $\mathcal{C}_T(m_0)$ and $\mathcal{C}_T(m_1)$ is at most $\delta$. Formally,

**Definition 6.1** *A message transmission protocol is $\delta$-private against a set $T$ if for every strategy of the Byzantine set $T$ and every two messages $m_0, m_1$ the set $T$ cannot distinguish which message was sent as follows: for every set of keys $\{k_{u,v} : \langle u,v \rangle \in E_A;\ u \in T\}$, every two set of random inputs $\{r_u : u \in T\}$ it holds that*

$$\sum_c \left| \Pr[\mathcal{C}_T(m_0, \{r_u\}_{u\in V}, \{k_{u,v}\}_{\langle u,v\rangle \in E_A}) = c] - \Pr[\mathcal{C}_T(m_1, \{r_u\}_{u\in V}, \{k_{u,v}\}_{\langle u,v\rangle \in E_A}) = c] \right| \leq 2\delta \ . \tag{1}$$

*The probabilities are taken over the random keys of the parties in $V \setminus T$ and the keys of pairs of parties in $V \setminus T$, and the sum is over all possible values of the communication. A message* *transmission protocol is* $(t, \delta)$-private *if it is* $\delta$*-private against every set of size at most* $t$. *A protocol is* perfectly $t$-private *if it is* $(t, 0)$*-private.*

We do not consider the setting of privacy without reliability, since that condition could be met by a protocol without any messages. We next present the outline of the protocol for private message transmission. The exact protocol is described in Fig. 6. The protocol uses the fact that there are $t+1$ disjoint paths from $a$ to $b$, and that $a$ and $b$ can communicate reliably without privacy. In the first step of the protocol The transmitter $a$ sends a different random one-time pad on each of the $t+1$ paths to $b$ . Using reliable non-private communication, and a randomized authentication procedure, $a$ and $b$ determine which pads have been received correctly by $b$ (Steps (3)-(5) of the protocol). The transmitter $a$ then encrypts the message using the sum of the pads that pass the test, and sends this encryption to $b$ reliably and non-privately. We use universal hashing [2, 13] for the randomized authentication procedure, and perform all arithmetic in a large finite field $\mathbf{F}$.

We first argue that this protocol is reliable with high probability.

**Lemma 6.2** *Assume that* $|\mathbf{F}| \geq 3n/\epsilon$ *and each reliable non-private transmission is* $(t, \epsilon/3)$*-reliable. Then, Protocol* PRIVATE-TRANSMISSION *is* $(t, \epsilon)$*-reliable.*

**Proof:** If both of the reliable non-private transmissions are successful, then $r_j^a, s_j^a = r_j^b, s_j^b$ for all $1 \leq j \leq t+1$, and $G^a, z^a = G^b, z^b$. The probability that $c_j^a \neq c_j^b$ for any given $j \in G^a$ is at most $1/|\mathbf{F}|$, since it can only happen when $r_j^a = (d_j^b - d_j^a)/(c_j^a - c_j^b)$. But $m^a = m^b$ unless $c_j^a \neq c_j^b$ for some $j \in G^a$. If each reliable non-private transmission fails with probability $\epsilon/3$, then this protocol fails to be reliable with probability at most $\frac{2\epsilon}{3} + \frac{t+1}{|\mathbf{F}|} \leq \epsilon$. □

We argue that the protocol is private with high probability.

**Lemma 6.3** *Assume that each reliable non-private transmission is* $(t, \epsilon/3)$*-reliable. Then, Protocol* PRIVATE-TRANSMISSION *is* $(t, 2\epsilon/3)$*-private. Furthermore, if each reliable non-private transmission is perfectly detecting then the protocol is perfectly* $t$*-private.*

**Proof:** The adversary's only source of information about $m^a$ is $z^a$. It suffices to show that the adversary has no information about $c_{j*}^a$ for some $j^* \in G^a$. There exists a path $j^*$ with no faults on it, i.e., $c_{j*}^a, d_{j*}^a = c_{j*}^b, d_{j*}^b$. If the reliable non-private transmission from $b$ to $a$ succeeds, then $j^* \in G^a$. The adversary's only source of information about $c_{j*}^a$ is $r_j^a, s_j^a$, from which all possible values of $c_{j*}^a$ are equally likely. Notice that privacy may be threatened when the reliable non-private transmission from $b$ to $a$ fails. A partial disruption of this transmission (while otherwise following the protocol honestly) could cause $G^a$ to contain only paths on which there is a fault. This could allow the adversary to learn $c_j^a$ for every $j \in G^a$, and hence determine $m^a$ from $z^a$. However, if the reliable transmission protocol is perfectly detecting then Protocol PRIVATE-TRANSMISSION is perfectly private. □

Thus, Theorem 3.1 implies the following:

**Theorem 6.4** *Let $G$ be a graph describing a network of reliable and private channels. If the graph $G$ is $(t+1)$-connected and the graph $G \cup G_A$ is $(2t+1)$-connected, then for every $\epsilon > 0$ there is an efficient protocol for $(t, \epsilon)$-reliable and perfectly $t$-private message transmission between any pair of parties which uses a $(2, \epsilon/(6nt))$-authentication scheme.*

Theorem 4.3 implies a similar result for single-pair transmission. Notice, that Protocol PRIVATE-TRANSMISSION requires reliable transmission in both directions. This cannot be avoided; if reliable and private is possible from $a$ to $b$ then reliable transmission is possible from $b$ to $a$ ($a$ transmits to $b$ a secret key, $b$ uses this key to authenticate the message and sends the authenticated message on $t+1$ disjoint paths).

**Theorem 6.5** *Let $G$ be a graph describing a network of reliable and private channels. If for every $T_0, T_1 \subseteq V \setminus \{a, b\}$, such that $|T_0|, |T_1| = t$, the set $T_0 \cup T_1$ is not an $(a, b)$-separating set in the graphs $G^*_{a,b}$ and $G^*_{b,a}$, then for every $\epsilon > 0$ there is a protocol for $(t, \epsilon)$-reliable and $(t, 2\epsilon/3)$-private message transmission between $a$ and $b$.*

# References


[1] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for noncryptographic fault-tolerant distributed computations. In *Proc. of the 20th Annu. ACM Symp. on the Theory of Computing*, pages 1–10, 1988.

[2] J. Carter and M. Wegman. Universal classes of hash functions. *J. of Computer and System Sciences*, 18:143–154, 1979.

[3] D. Chaum, C. Crepau, and I. Damgard. Multiparty unconditionally secure protocols. In *Proc. of the 20th Annu. ACM Symp. on the Theory of Computing*, pages 11–19, 1988.

[4] D. Dolev. The Byzantine generals strike again. *J. of Algorithms*, 3:14–30, 1982.

[5] D. Dolev, C. Dwork, O. Waarts, and M. Yung. Perfectly secure message transmission. *J. of the ACM*, 40(1):17–47, 1993.

[6] C. Dwork, D. Peleg, N. Pippenger, and E. Upfal. Fault tolerance in networks of bounded degree. *SIAM J. on Computing*, 17(5):975–988, 1988.

[7] S. Even. *Graph Algorithms*. Computer Science press, 1979.

[8] M. J. Fischer, N. A. Lynch, and M. Merritt. Easy impossibility proofs for distributed consensus problems. *Distributed Computing*, 1(1):26–39, 1986.

[9] M. K. Franklin and R. N. Wright. Reliable and private communication over echo lines. Manuscript, 1997.

[10] K. Menger. Allgemeinen kurventheorie. *Fund. Math.*, 10:96–115, 1927.

[11] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *Proc. of the 21st Annu. ACM Symp. on the Theory of Computing*, pages 73–85, 1989.

[12] G. J. Simmons. A survey of information authentication. In G. J. Simmons, editor, *Contemporary Cryptology, The Science of Information Integrity*, pages 441–497. IEEE Press, 1992.

[13] M. Wegman and J. Carter. New hash functions and their use in authentication and set equality. *J. of Computer and System Sciences*, 22:265–279, 1981.

**PROCEDURE SEND**$(m, P, \mathrm{Destb})$

PARAMETERS:

$m$ – a message

$P = v_0, v_1, \ldots, v_\ell$ – a path in $G^*$

Destb – Boolean variable, TRUE if destination of the message is $b$

$m_0 \leftarrow m$

FOR $i = 0$ TO $\ell - 1$ DO (* $v_i$ propagates the message to $v_{i+1}$ *)

IF $\langle v_i, v_{i+1} \rangle \in E$ THEN $v_i$ sends $m_i$ to $v_{i+1}$ on this edge.

OTHERWISE, $\langle v_i, v_{i+1} \rangle \in E_A$:

1. $v_i$ executes SEND$(\langle m_i, \textsc{auth}(m_i, k_{v_i,v_{i+1}})\rangle, \textsc{path}(v_i, v_{i+1}), \mathrm{FALSE})$
   (* This is an internal call *)
2. IF $v_{i+1}$ received $\langle \hat{m}, \hat{\alpha} \rangle$ such that $\hat{\alpha} \neq \textsc{auth}(\hat{m}, k_{v_i,v_{i+1}})$
   THEN $m_{i+1} \leftarrow$ "$v_{i+1}$ has not accepted a message from $v_i$".
   OTHERWISE, $m_{i+1} \leftarrow$ "$v_{i+1}$ has accepted $\hat{m}$ from $v_i$".
3. IF (Destb = FALSE ) THEN $v_{i+1}$ executes (in parallel)
   SEND("alert, $m_{i+1}$", $\textsc{path}(v_{i+1}, b)$, TRUE). (* This is an alert call *)

Figure 5: A procedure to send a message on a path in $G^*$.

**PROTOCOL TRANSMIT**$(m)$

The transmitter $a$ executes SEND to $b$ three times:
SEND$(m, P_0$,TRUE) – $P_0$ is a path from $a$ to $b$ in $G$ which misses $T_0$.
SEND$(m, P_1$,TRUE) – $P_1$ is a path from $a$ to $b$ in $G$ which misses $T_1$.
SEND$(m$, PATH$(a, b)$,TRUE)
IF the receiver $b$ has received a message on PATH$(a, b)$ THEN $b$ accepts it.
OTHERWISE, $b$ learns that $T_i$ is Byzantine for some $i \in \{0, 1\}$.
$b$ accepts the message received on $P_i$.

Figure 6: A procedure to transmit a message from $a$ to $b$.

**PROTOCOL PRIVATE-TRANSMISSION**

1. Along each disjoint path $j$, $1 \leq j \leq t+1$, the transmitter $a$ sends to $b$ the values $c_j^a, d_j^a \in_R \mathbf{F}$.
2. Let $c_j^b, d_j^b$ be what $b$ receives on path $j$, $1 \leq j \leq t+1$.
3. Using the reliable non-private communication protocol, $b$ transmits to $a$ the values $r_j^b, s_j^b$, $1 \leq j \leq t+1$, where $r_j^b \in_R \mathbf{F}$ and $s_j^b = r_j^b c_j^b + d_j^b$.
4. Let $r_j^a, s_j^a$, $1 \leq j \leq t+1$, be what $a$ receives.
5. Using the reliable non-private communication protocol, $a$ transmits to $b$ the values $G^a, z^a$, where $G^a = \{j : s_j^a = r_j^a c_j^a + d_j^a\}$ and $z^a = m^a + \sum_{j \in G^a} c_j^a$.
6. Let $G^b, z^b$ be what $b$ receives.
7. $b$ computes $m^b = z^b - \sum_{j \in G^b} c_j^b$.

Figure 7: A protocol for private transmission of a message