

Reliable Communication over Partially Authenticated Networks*

Amos Beimel[†] Matthew Franklin[‡]

January 14, 1998

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, so we suggest a simpler solution: giving some pairs of processors (other than the pairs connected by channels) authentication keys, i.e., the means to identify messages from the

*An extended abstract of this paper appears in the proceedings of the WDAG '97 conference, volume 1320 of *Lecture Notes in Computer Science*, pages 245-259, Springer, 1997.

[†]Division of Engineering and Applied Sciences, Harvard University, 40 Oxford st., Cambridge, MA 02138. Email: beimel@deas.harvard.edu. This work was done while the author was a postdoctoral fellow at DIMACS.

[‡]AT&T Labs — Research, Florham Park, NJ. E-mail: franklin@research.att.com.

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. The proof of sufficiency is constructive, and we present an *efficient* protocol that achieves reliable communication whenever it is possible.

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. We give an exact characterization when a can transmit a message to b . However, the protocol we present for proving the sufficiency of the characterization is not efficient (it requires $2^{O(n \log n)}$ rounds, where n is the number of processors in the network). It remains an open problem to characterize when efficient communication from a to b is possible.

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. Our necessary and sufficient condition for single-pair fault-restricted reliable communication depends critically on a recursively defined graph which includes all the edges of the communication graph and some of the edges of the authentication graph. Somewhat surprisingly, it turns out that single-pair fault-restricted reliable communication from a to b does not imply single-pair fault-restricted reliable communication from b to a .

We also explore reliable and *private* communication, i.e., communication in which the faulty processors learn no information about the transmitted message. We show that all-pairs reliable and private communication is possible whenever all-pairs reliable communication is (provided that the communication channels are private). This also characterizes when all

secure multiparty computation are possible in partially authenticated networks, as Ben-Or et al. [1] and Chaum et al. [3] show that 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). We further prove that reliable and private communication from a to b is possible whenever we have reliable communication from a to b and from b to a .

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, 18], reliable message transmission [4, 5], and reliable and private message transmission [16, 5]. Simple impossibility results and 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.

Digital signatures have been used for Byzantine agreement, see, e.g. [15] and discussion and references in [13] (some of these reference use the term authentication for digital signatures). Signatures are stronger than authenticated messages, which are used in our work. They enable the signer to sign the message such that every other party can verify the signature, but cannot sign a different message. To the best of our knowledge, no previous papers addressed reliability in partially authenticated networks.

The tasks we consider in this paper bear some relation to practical group communication tasks [10]. Reliable multicast protocols are often designed to operate in distributed adversarial environments similar to the ones we consider. When these multicast protocols are built on top of a network of reliable single-receiver channels, then they could be implemented directly from our results. It is an interesting research question to find more direct and efficient implementations of reliable multicast using our techniques.

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, which is used in Section 5 to characterize the single-pair reliable communication problem. Finally, in Section 6, we discuss private communication.

2 Preliminaries

2.1 The Model

The system is synchronous and 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 were 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).

Our results remain true versus a stronger “rushing” adversary. Intuitively, a rushing adversary can delay the determination of parties to corrupt and messages to send until the “last possible moment” of every round.

Definition 2.1 [Reliable Protocol] *Let $a, b \in V$ be a transmitter and a receiver, and $t \leq n - 2$. We say that a message transmission protocol from a to b is (t, ϵ) -reliable if, when the adversary can control any set T of at most t parties such that $T \subseteq V \setminus \{a, b\}$, 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.*

At the end of a protocol there are three options: (1) b accepts the message transmitted by a , or (2) b accepts an incorrect message (different than the message transmitted by a), or (3) b detects that the transmission has failed. We say that the protocol is *perfectly detecting* if b never accepts an incorrect message, i.e., the second option never happens.

In the rest of the paper we distinguish between *sending* a message and *transmitting* it; a message is sent on an edge or a path, while a transmission of a message means executing a protocol in which the transmitter transmits a message to a receiver (there is only one transmitter and one receiver). Similarly, a message is *received* on an edge or a path, while a message is *accepted* by the receiver at the end of a transmission protocol.

2.2 Authentication Schemes

We briefly define authentication schemes; the reader is referred to [17] for more details. Let s be a positive integer and K be a finite set, called the set of keys. An authentication scheme for messages in $\{0, 1\}^s$ is a pair $\langle \mathbf{AUTH}, \mu \rangle$, where $\mathbf{AUTH} : \{0, 1\}^s \times K \rightarrow \{0, 1\}^*$ is a function, and μ 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 $\alpha = \mathbf{AUTH}(m, k)$ and sends to Bob the message m together with the authentication tag α . When Bob receives a pair m, α , he verifies that the authentication is correct by computing the value $\mathbf{AUTH}(m, k)$ and comparing it to α . If they are equal then Bob assumes that the message was transmitted by Alice. The scheme is secure if an eavesdropper Eve, knowing $\langle \mathbf{AUTH}, \mu \rangle$ but not k , cannot generate a pair which Bob accepts. Formally, the scheme $\langle \mathbf{AUTH}, \mu \rangle$ is called an (ℓ, ϵ) -authentication scheme if every probabilistic strategy that sees any ℓ pairs, $\langle m_1, \mathbf{AUTH}(m_1, k) \rangle, \langle m_2, \mathbf{AUTH}(m_2, k) \rangle, \dots, \langle m_{\ell}, \mathbf{AUTH}(m_{\ell}, k) \rangle$, cannot generate, with probability greater than ϵ , a pair m, α , such that $m \neq m_i$ for every i , and $\alpha = \mathbf{AUTH}(m, k)$, where the probability is over the distribution of the authentication keys, and the random input of the probabilistic strategy.

We note that one can use short keys to authenticate long messages while providing information theoretic security. For example, Krawczyk [11, 12] constructs efficient (ℓ, ϵ) -authentication schemes in which the length of the message is s , the length of the authentication tag is $O(\log(s/\epsilon))$ and the length of the key is $O(\ell \log(s/\epsilon))$.

Remark 2.2 In our definitions we consider information-theoretic authentication schemes and our protocols are immune against an adversary with unlimited power. A different way to present our results is to assume that the authentication schemes are only computationally secure (i.e., any *efficient* algorithm cannot break the scheme with non-negligible probability), and assume that the adversary is an efficient algorithm. For our reliable protocols in Sections 3–5, any successful attack implies a successful break of the underlying authentication scheme; this is not true for the private and reliable protocol of Section 6.

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 [14] 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, ϵ) -reliable message transmission between any pair of parties which uses a $(1, \epsilon/(2nt))$ -authentication scheme. Furthermore, the protocol is perfectly detecting.*

We first describe in Fig. 1 a subprotocol for sending messages reliably on edges in G_A . In the next lemma we prove its correctness.

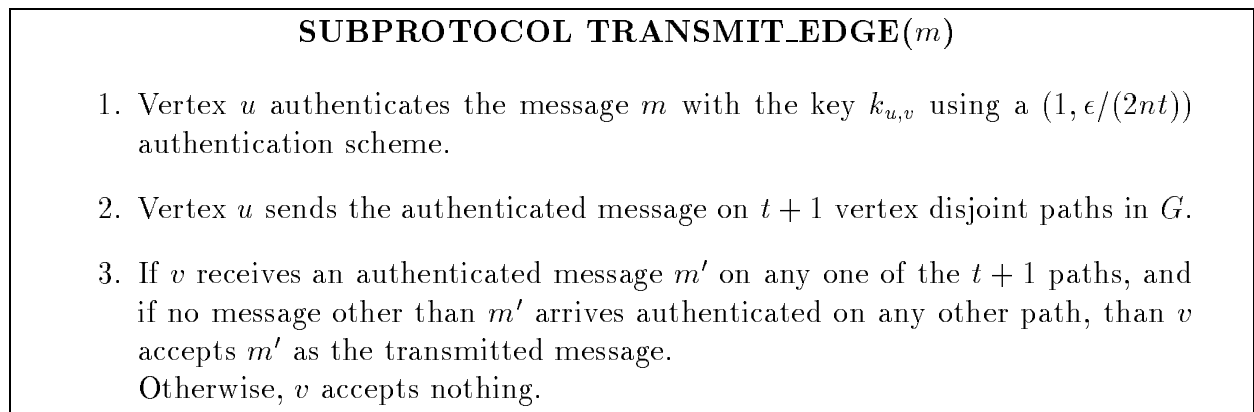


Figure 1: A subprotocol for transmitting a message on an edge from G_A .

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 $\epsilon > 0$, Subprotocol TRANSMIT_EDGE, described in Fig. 1, is an efficient $(t, \epsilon/(2n))$ -reliable protocol for transmitting a message from u to v which uses a $(1, \epsilon/(2nt))$ -authentication scheme. Furthermore, Subprotocol TRANSMIT_EDGE is perfectly detecting.*

Proof: Since there are at most t Byzantine vertices, v will receive the right authenticated message on at least one path. Thus v will accept 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 accept nothing, and thus the subprotocol is perfectly detecting (that is, v never accepts an incorrect message). \square

In Fig. 2 we describe Protocol SIMPLE_TRANSMIT for reliable message transmission. We complete the proof of Theorem 3.1 by proving that this protocol is reliable (Lemma 3.3) and perfectly detecting (Lemma 3.4).

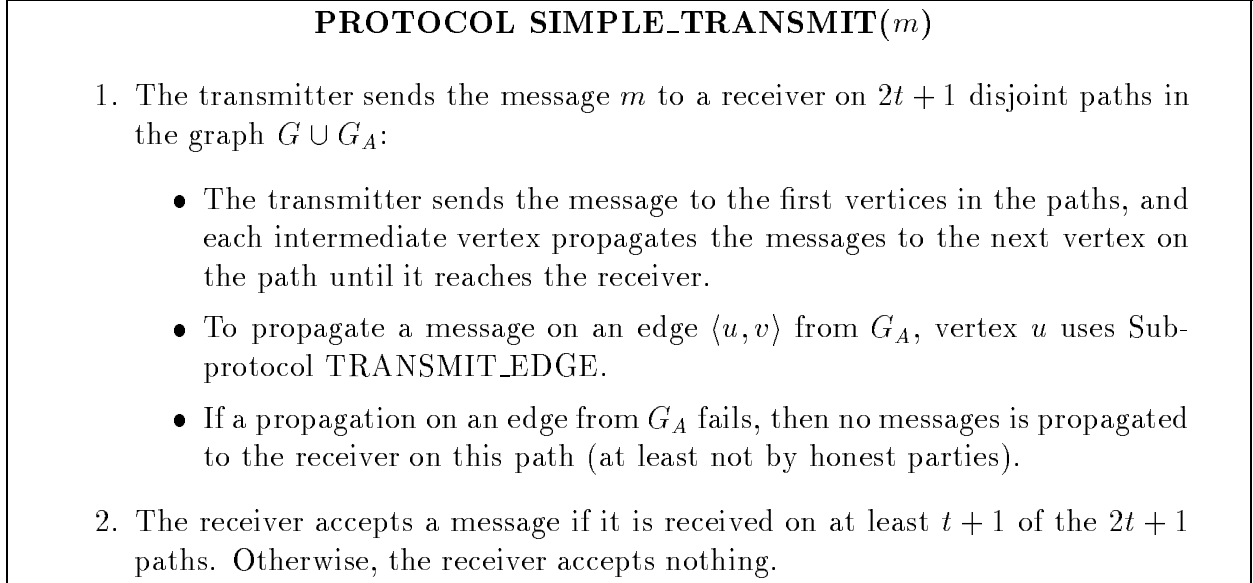


Figure 2: A protocol for transmitting a message when G is $t + 1$ connected and $G \cup G_A$ is $2t + 1$ connected.

Lemma 3.3 *Protocol SIMPLE_TRANSMIT, described in Fig. 2, is a (t, ϵ) -reliable message transmission protocol between any pair of parties.*

Proof: If the adversary has not disrupted one of the executions of Subprotocol TRANSMIT_EDGE then the correct message is received on at least $t + 1$ paths and is accepted by b . We next give an upper bound on the probability that the adversary disrupts one of the executions of the subprotocol. 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 Subprotocol TRANSMIT_EDGE. 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., $2n \cdot \epsilon / (2n) = \epsilon$. \square

Lemma 3.4 *Protocol SIMPLE_TRANSMIT, described in Fig. 2, is perfectly detecting.*

Proof: Subprotocol TRANSMIT_EDGE 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, ϵ) -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

In the next lemma we construct pairs of executions of the protocol in which the sender transmits different messages, the adversary controls different subsets of parties, and yet the views of the receiver are identical. These “ambiguous” pairs of executions are used in Theorem 3.7 to construct an adversary that disrupts the transmission of a message. We stress that Lemma 3.6 contains no probabilities. If the adversary follows the strategy outlined in Fig. 4, then *every* execution is *guaranteed* to be part of an ambiguous pair.

Lemma 3.6 *Let a and b be vertices, and $T_0, T_1 \subseteq V \setminus \{a, b\}$ be disjoint subsets, such that $T_0 \cup T_1$ is an (a, b) -separating set in $G \cup G_A$. Assume that there is some protocol for message transmission from a to b . For every pair of messages m_0, m_1 there are pairs of executions of the protocol such that in one execution T_0 is Byzantine and the message transmitted by a is m_0 , in the other execution T_1 is Byzantine and the message transmitted by a is m_1 , and the receiver b gets exactly the same messages in both executions.*

Proof: Let $T = T_0 \cup T_1$, define B as 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 $G \cup G_A$), and define A as $V \setminus (B \cup T)$. Fig. 3 describes these sets. Notice that there are no edges between A and B since T is a separating set in $G \cup G_A$.

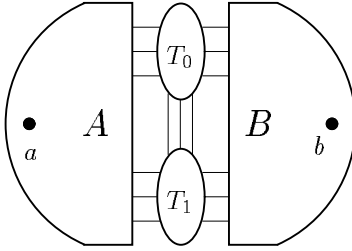


Figure 3: A description of the partition of the vertices.

Define two executions 0 and 1 as follows. In both executions the vertices in B hold the random inputs $\{r_u : u \in B\}$, and the keys $\{k_{u,v} : u \in B, v \in V, \langle u, v \rangle \in E_A\}$. 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 $\{k_{u,v}^i : u, v \in A \cup T, \langle u, v \rangle \in E_A\}$. The behavior of the Byzantine set T_i in 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 $A \cup T_i$ in execution \bar{i} . To achieve this task, the adversary simulates, round by round, the behavior of the vertices in $A \cup T_i$ for execution \bar{i} , using $\{r_u^{\bar{i}} : u \in A \cup T_i\}$ as the random inputs for the vertices in $A \cup T_i$, and $\{k_{u,v}^{\bar{i}} : u \in A \cup T_i, v \in A \cup T, \langle u, v \rangle \in E_A\}$ and $\{k_{u,v} : u \in T_i, v \in B, \langle u, v \rangle \in E_A\}$ as the keys held by parties in $A \cup T_i$. 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_i are added to the history. These two executions are described in Fig. 4.

	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 $A \cup T$	$\{k_{u,v}^0 : u, v \in A \cup T, \langle u, v \rangle \in E_A\}$	$\{k_{u,v}^1 : u, v \in A \cup T, \langle u, v \rangle \in E_A\}$
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 $A \cup T$	$\{k_{u,v}^1 : u, v \in A \cup T, \langle u, v \rangle \in E_A\}$	$\{k_{u,v}^0 : u, v \in A \cup T, \langle u, v \rangle \in E_A\}$
Behavior	T_1 sends no messages to $A \cup T_0$	T_0 sends no messages to $A \cup T_1$

Figure 4: The two executions that confuse b .

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, and the members of B , and in particular b , receive and send the same messages in both executions. Thus, 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 . \square

Theorem 3.7 *If (t, ϵ) -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: Assume that $G \cup G_A$ is not $(2t + 1)$ -connected, and assume that there is some protocol for message transmission from a to b in $G \cup G_A$. There exist vertices a, b and an (a, b) -separating set T in $G \cup G_A$ of at most $2t$ vertices. Let T_0, T_1 be an arbitrary partition of T into two disjoint sets of size at most t , and fix any $m_0 \neq m_1$. Consider all the pairs of executions from Lemma 3.6, where the random inputs $\{r_u^0 : u \in A \cup T_1\}$, $\{r_u^1 : u \in A \cup T_0\}$ and $\{r_u : u \in B\}$, and keys $\{k_{u,v}^0 : u, v \in A \cup T, \langle u, v \rangle \in E_A\}$, $\{k_{u,v}^1 : u, v \in A \cup T, \langle u, v \rangle \in E_A\}$ and $\{k_{u,v} : u \in B, v \in V, \langle u, v \rangle \in E_A\}$ range over all their possible values. In each pair of executions, whenever b accepts the correct message in one execution it errs in the second. Thus, for 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 random local inputs for the parties in $A \cup T_i$, and chooses keys for pairs of parties in $A \cup T$ according to the same distribution that the real keys are distributed. The adversary simulates the parties in $A \cup T_i$ with $m_{\bar{t}}$, the keys and the local random inputs it chose. 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. Furthermore, the adversary always tries to convince b that $m_{\bar{t}}$ is transmitted (even if $m_{\bar{t}}$ is really transmitted). 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 the reliability of every protocol is at least $1/2$. \square

4 Single-Pair Fault-Restricted Communication

In this section we consider single-pair fault-restricted communications. 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. This version of the problem is only a tool for characterizing when (t, ϵ) -reliable transmission between a given pair of parties a and b is possible (see Section 5). We present a characterization of single-pair fault-restricted reliable communication. The proof of sufficiency is constructive, i.e., we present a protocol that achieves reliable communication whenever it is possible. However, the protocol is not efficient (it requires $2^{O(n \log n)}$ rounds).

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, α on both paths, where $\alpha = \mathbf{AUTH}(m, k_{a,b})$. 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. 5. 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.

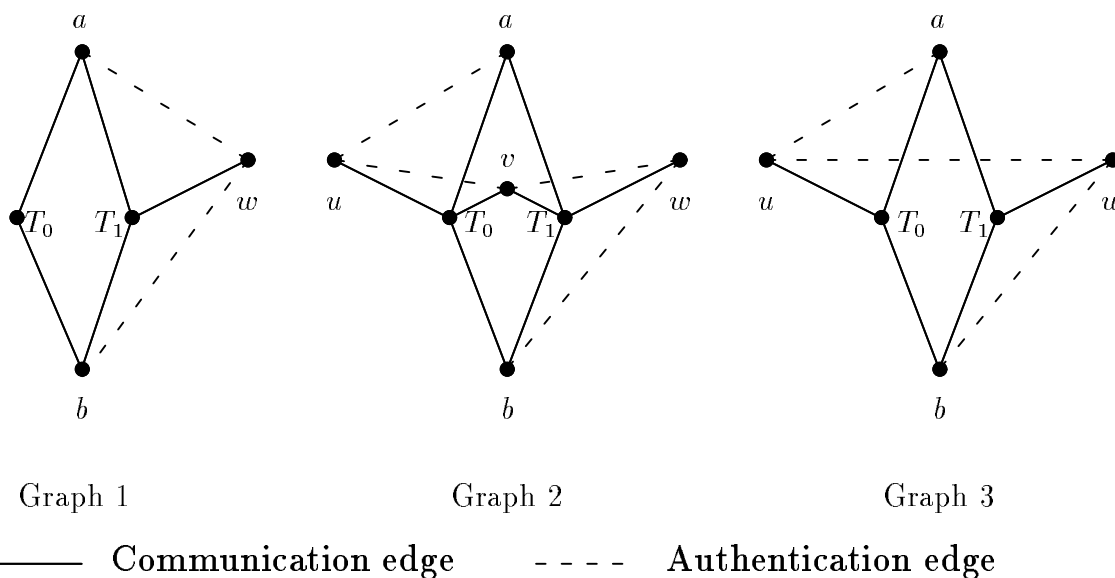


Figure 5: Examples of graphs in which a can $(1, \epsilon)$ -reliably transmit a message to b .

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. 5. 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 both T_0 and T_1 are 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 $\langle m, \text{AUTH}(m, k_{u,v}) \rangle$ 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. 5. 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 separate u and w in G . However, T_1 does not separate 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 inductive definition of G^* in the next section and a recursive protocol in Section 4.3.

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 [The graph G^*] *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. We inductively define a sequence of subsets of authentication edges $E_0 \subseteq E_1 \subseteq E_2 \subseteq \dots$, where $E_0 = \emptyset$ and $\langle u, v \rangle \in E_i$ whenever the following properties hold:*

1. $\langle u, v \rangle \in E_A$.
2. $u, v \notin T_0 \cup T_1$.
3. At most one of T_0 or T_1 is a (u, v) -separating set in $G_{i-1} = \langle V, E \cup E_{i-1} \rangle$.
4. Either v or u is in the connected component of b in the graph $\langle V \setminus (T_0 \cup T_1), E \cup E_{i-1} \rangle$.

If $E_{k+i} = E_k$ for every integer $i \geq 1$ then $G^* = \langle V, E^* \rangle = \langle V, E \cup E_k \rangle$ (we will prove that such k exists).

The graph G^* depends upon a, b, T_0 and T_1 . We use the notation $G^*(a, b, T_0, T_1)$ when we want to emphasize this dependency. Informally, Property (3) ensures that v learns the Byzantine set if the edge is disabled, and Property (4) ensures that it can tell b about it. As an example for this definition, consider Graph 3 described in Fig. 5. 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 [Confusing Pair] *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 $G^*(a, b, T_0, T_1)$

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 an (a, b) -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 an (a, b) -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 4.3 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.

4.3 Protocol for Proving Sufficiency

First we present Protocol SEND, described in Fig. 6, which sends a message along a path from vertex u to vertex v in G^* . In this protocol either b accepts the message sent by u or the receiver b learns which set is Byzantine (if the adversary breaks the authentication scheme then v might accept an incorrect message).

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 $\langle u, v \rangle \in E^* \setminus E$, define $\mathbf{PATH}(u, v)$ as a fixed path with minimum level among all the simple paths between u and v in G^* that either miss T_0 or miss T_1 . Furthermore, for every v that is adjacent to an authentication edge $\langle u, v \rangle$ in G^* , define $\mathbf{PATH_TO_}b(v)$ as a fixed path with minimum level among all the paths from v to b in G^* which miss $T_0 \cup T_1$. (It is possible that $\mathbf{PATH_TO_}b(v) \neq \mathbf{PATH}(v, b)$.) The following proposition is an immediate consequence of Definition 4.1:

Proposition 4.5 *If $\langle u, v \rangle \in E_j$ then*

- *The level of $\mathbf{PATH}(u, v)$ is at most $j - 1$.*
- *The levels of $\mathbf{PATH_TO_}b(v)$ and $\mathbf{PATH_TO_}b(u)$ are at most j .*
- *Furthermore, the level of either $\mathbf{PATH_TO_}b(v)$ or $\mathbf{PATH_TO_}b(u)$ is at most $j - 1$.*

If $\langle u, v \rangle \in E_j$ and the level $\mathbf{PATH_TO_}b(v)$ is j (thus, u is in the connected component of b in the graph $\langle V \setminus (T_0 \cup T_1), E \cup E_{j-1} \rangle$ while v is not) then $\mathbf{PATH_TO_}b(v) \stackrel{\text{def}}{=} v, \mathbf{PATH_TO_}b(u)$. All the parties in V know $\mathbf{PATH}(u, v)$ and $\mathbf{PATH_TO_}b(v)$, which are part of the specification of the protocol.

In Procedure SEND, described in Fig. 6, 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.

Procedure SEND has a Boolean parameter Destb, which indicates if the final destination of the message is b . The purpose of this mysterious parameter is to ensure that if the final destination of a message is b and the message is sent on a path of level j , then all alert calls use paths whose level is less than j . This enables to analyze the properties of Procedure SEND in Lemma 4.11. We stress that Destb = TRUE is not equivalent to the condition that b is the last vertex in P . For example, if $\langle w, b \rangle$ is an authentication edge in P then there would be an internal recursive call to SEND with **PATH**(w, b). In this case the final destination of the message is the last vertex in P and therefore Destb = FALSE.

In Fig. 7 we describe Protocol TRANSMIT in which a transmits a message to b . In this protocol the transmitter sends the message on three paths and the receiver accepts one of the messages received on these paths.

4.4 Proof of Sufficiency

We first establish that the execution of SEND(m, P, Destb) always terminates, although it might take as many as $2^{O(n \log n)}$ rounds. To prove this statement we consider the graph G^* , and prove that it is properly defined. That is, the sequence of subsets of authentication edges of Definition 4.1 converges to some E_k for some $k \leq n$.

Lemma 4.6 *Let G and G_A be any graphs, and $E_0 \subseteq E_1 \subseteq E_2 \subseteq \dots$ be the sequence of subsets of authentication edges of Definition 4.1. Then, $E_{n+i} = E_n$ for every integer $i \geq 1$.*

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 $v \in B_{k-1}$ and 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. \square

Lemma 4.7 *Let j be the level of P . The execution of SEND(m, P, 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 P . If the level of P is 0 then $O(n)$ rounds suffice for sending a message along P 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 P . Thus, by the induction hypothesis, each recursive internal call takes n^j rounds, and the execution of P terminates after at most $n \cdot n^j = n^{j+1}$ rounds since the length of P in G^* is at most n .

Now examine the time required for each alert call for an authentication edge $\langle u, v \rangle$ in P . If the level of **PATH_TO**. $b(v)$ is at most $j - 1$ then the message is propagated to b after at most n^j rounds. Otherwise, **PATH_TO**. $b(v) = v, \text{PATH_TO}$. $b(u)$, thus, u will get the message

PROCEDURE SEND(m, P, Destb)

PARAMETERS:

m – a message,

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

Destb – Boolean variable, TRUE if the final 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 and $m_{i+1} \leftarrow m_i$,

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

1. v_i executes $\text{SEND}(\langle m_i, \mathbf{AUTH}(m_i, k_{v_i, v_{i+1}}) \rangle, \mathbf{PATH}(v_i, v_{i+1}), \text{FALSE})$.

(* This is an internal call. *)

2. IF v_{i+1} received $\langle \hat{m}, \hat{\alpha} \rangle$ such that $\hat{\alpha} \neq \mathbf{AUTH}(\hat{m}, k_{v_i, v_{i+1}})$

THEN $m_{i+1} \leftarrow$ “ v_{i+1} has not accepted a message from v_i ”,

OTHERWISE, IF $\text{Destb} = \text{TRUE}$

THEN $m_{i+1} \leftarrow$ “ v_{i+1} accepted a message from v_i ” $\circ \hat{m}$,

OTHERWISE, $m_{i+1} \leftarrow \hat{m}$.

3. IF ($\text{Destb} = \text{FALSE}$) THEN v_{i+1} executes (in parallel)

$\text{SEND}(\text{“alert”} \circ m_{i+1}, \mathbf{PATH_TO_}b(v_{i+1}), \text{TRUE})$.

(* This is an alert call. *)

Figure 6: 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($m, \text{PATH_TO_}b(a), \text{TRUE}$)

IF the receiver b has received a message on $\text{PATH_TO_}b(a)$ 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 7: A procedure to transmit a message from a to b .

(on $\text{PATH}(v, u)$ of level $j - 1$) after at most n^j rounds, and the message is propagated to b after at most n^j additional rounds. We conclude that all messages sent in the execution of P (caused by internal and alert recursive calls) are performed at most $O(n^{j+1})$ rounds after the execution has started. By Lemma 4.7, this quantity is bounded by $2^{O(n \log n)}$. \square

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 $\text{PATH_TO_}b(v)$ 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 “alert, 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 knows that $\text{PATH}(u, v)$ misses T_i , and thus T_i must be Byzantine.

Consider an execution by v of $\text{SEND}(m, \text{PATH_TO_}b(v), \text{TRUE})$, where m is either an alert message or a regular message. Assume that the adversary has not authenticated any message without having the authentication key. If there were no Byzantine parties then the message b receives on $\text{PATH_TO_}b(v)$ is of the form:

$$\text{“}v_{i_t+1} \text{ accepted a message from } v_{i_t}\text{”} \circ \dots \circ \text{“}v_{i_1+1} \text{ accepted a message from } v_{i_1}\text{”} \circ m', \quad (1)$$

where $m' = m$ and $\langle v_{i_1}, v_{i_1+1} \rangle, \dots, \langle v_{i_t}, v_{i_t+1} \rangle$ are the authentication edges on $\text{PATH_TO_}b(v)$. We say that the received message is syntactically correct if the prefix of the message is as it is in (1) (with all authentication edges from the path). The next proposition follows from the fact that $\text{PATH_TO_}b(v)$ misses all possible Byzantine parties in $T_0 \cup T_1$ and only parties on the path can change the message received on the path.

Proposition 4.8 *If b receives a syntactically correct message and the adversary has not authenticated any message without having the authentication key, then m' (the suffix of the message) was transmitted by v .*

We next assume that the received message is not syntactically correct. In this case b will detect a set T_i that contains all Byzantine vertices. There are three cases to consider:

1. b receives and accepts all alert messages sent during the execution, and all of them have the form “alert, v_{i+1} accepted a message from v_i ” $\circ\hat{m}$.
2. b receives and accepts all alert messages sent during the execution, and at least one of them has the form “alert, v_{i+1} has not accepted a message from v_i ”.
3. b does not accept some alert messages sent during the execution.

In the next two claims we show how b detects the Byzantine set if it received all alert messages.

Claim 4.9 *Assume that b does not receive a syntactically correct message and it receives and accepts all alert messages sent during the execution, and all of them have the form “alert, v_{i+1} accepted a message from v_i ” $\circ\hat{m}$. Then b can detect the Byzantine set.*

Proof: We note that only authentication edges not adjacent to possible Byzantine parties in $T_0 \cup T_1$ are added to G^* . Thus, when b accepts an alert message claiming that a v_{i+1} has accepted a message from v_i then v_{i+1} really accepted this message.

If the conditions of the claim hold then no authentication edge was disabled during the recursive calls and the only authentication edges that were disabled are on $\text{PATH_TO}_b(v)$. In this case the message that b receives on $\text{PATH_TO}_b(v)$ is “ v_{i+1} has not accepted a message from v_i ” for some $\langle v_i, v_{i+1} \rangle$ on $\text{PATH}(v, b)$, and this message can be trusted since $\text{PATH_TO}_b(v)$ contains no Byzantine parties. There is only one set that hits $\text{PATH}(v_i, v_{i+1})$ and b learns that this set must be Byzantine. \square

Claim 4.10 *Assume that b does not receive a syntactically correct message and it receives and accepts all alert messages sent during the execution, and at least one of them has the form “alert, v_{i+1} has not accepted a message from v_i ”. Then b can detect the Byzantine set.*

Proof: Let $\langle x, y \rangle$ be the first authentication edge for which b receives an alert message claiming that “ y has not accepted a message from x ”. That is, x got the message transmitted by v and no edge used in the recursive call during the execution of $\text{SEND}(m, \text{PATH}(x, y), \text{FALSE})$ was disabled, and the edge $\langle x, y \rangle$ was disabled by a vertex on $\text{PATH}(x, y)$ which did not propagate the message it got. There is only one set that hits $\text{PATH}(x, y)$ and b learns that this set must be Byzantine. \square

Lemma 4.11 *Assume that v executes $\text{SEND}(m, \text{PATH_TO}_b(v), \text{TRUE})$, 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 syntactically correct message on the path then b accepts the suffix of the message. By Proposition 4.8 this is a correct decision. Otherwise, we prove the lemma by induction on the level of $\text{PATH_TO}_b(v)$. If the level is zero, this is a path in G that misses all possible Byzantine parties in $T_0 \cup T_1$, thus b receives a syntactically correct message on this path.

Now, let $\text{PATH_TO}_b(v)$ be a path of level $j > 0$. If all alert messages are received and accepted by b then by Claim 4.9 and Claim 4.10, the receiver b detects which set is Byzantine. Otherwise, we use a “missing” alert message to detect the Byzantine set.

Alert messages are only sent during recursive calls (when $\text{Dest}_b = \text{FALSE}$). Thus, by Proposition 4.5, every alert message is sent on a path $\text{PATH_TO}_b(w)$ for some vertex w such that the level of $\text{PATH_TO}_b(w)$ is at most $j - 1$. Therefore, the alert messages which b does not accept are sent on paths of level at most $j - 1$, and by the induction hypothesis b detects which set is Byzantine. \square

We are ready to prove the sufficient condition:

Proof [Theorem 4.3]: Protocol TRANSMIT, described in Fig. 7, satisfies the requirements of the Theorem. First notice that by Definition 4.2 the paths P_0, P_1 and $\text{PATH_TO}_b(a)$ exist. By Lemma 4.11, if the adversary has not authenticated a false message, b receives the message transmitted by a . Let ℓ 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 4.7 both of these values are well-defined. Using an $(\ell, \epsilon/k)$ -authentication scheme ensures that the probability that the adversary has authenticated a false message is at most ϵ . \square

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. Recall that the idea of the proof of Theorem 3.7 is to construct pairs of “ambiguous” executions in which different messages are transmitted while the receiver gets exactly the same messages. This is done in Lemma 3.6, and is used in Theorem 3.7 to construct an adversary that disrupts the transmission of one message. Unlike Lemma 3.6, the set $T_0 \cup T_1$ is not a separating set in $G \cup G_A$ but rather in G^* , that is, we have to deal with the authentication edges deleted from G^* . The proof of Lemma 3.6 should be generalized to this case. Once we have done this, Theorem 4.4 follows from the same arguments as in the proof of Theorem 3.7.

In Lemma 3.6 we consider a pair of executions, in one execution T_0 is Byzantine, and a transmit m_0 and in the second T_1 is Byzantine, and a transmit m_1 . The Byzantine set T_i in one execution sends the same messages as the honest T_i sends in the other executions.

We next mention the modifications to the proof of Lemma 3.6. We define the sets A and B 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, $v \in B$ if $v \in V \setminus T_0 \cup T_1$ and $T_0 \cup T_1$ is not a (v, b) -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 an (a, b) -confusing pair, $a \in A$.

There are two problems to generalize the proof. The minor problem is that T_0 and T_1 can intersect. To solve this problem, the parties in $T_0 \cap T_1$ (which are Byzantine in both executions) do not send any messages. The major 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 an (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 a pair of executions as defined in Lemma 3.6, where in execution $i \in \{0, 1\}$ the adversary uses keys $\{k_{u,v}^i : u \in C, v \in B, \langle u, v \rangle \in E_A\}$ for the simulation (as well as the other keys and random inputs considered in Lemma 3.6). 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 $u, w_1, \dots, w_{\ell-1}, w_\ell = v$ 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., $u, w_1, \dots, w_{j-1} \in A \cup T_{\bar{i}}$). The Byzantine w_j ignores the real 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 $T_{\bar{i}}$ and no messages are sent by parties in $T_{\bar{i}}$ during the simulation. Thus, the simulated messages of u have no influence on the messages received by the parties in B , and can be ignored. \square

4.6 Fault-Restricted Communication is Not Symmetric

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 the previous intuition is false.

Lemma 4.12 *In the graphs G and G_A described in Fig. 8, (T_0, T_1, ϵ) -reliable transmission is possible from a to b , but not from b to a .*

Proof: Consider the graph $G^*(a, b, T_0, T_1)$. All the edges of G_A are added to this graph, thus a, v_7, b is a path in $G^*(a, b, T_0, T_1)$ that misses both T_0 and T_1 . Furthermore, neither T_0 nor

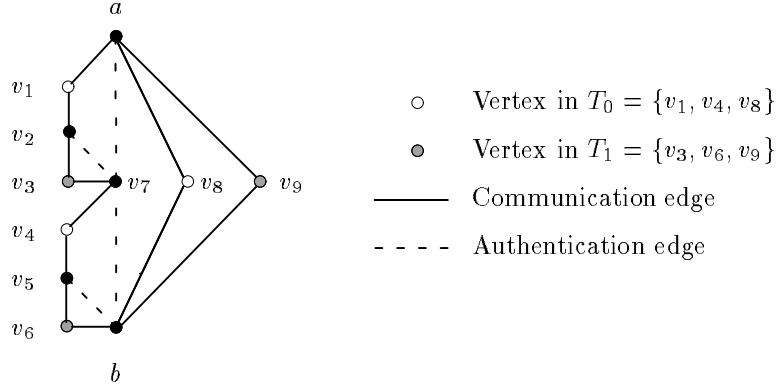


Figure 8: A graph in which a can transmit to b , but b cannot transmit to a .

T_1 is an (a, b) -separating set in G . Thus, by Theorem 4.3 vertex a can transmit a message to b . However, $G^*(b, a, T_0, T_1)$ does not contain any authentication edges, and $T_0 \cup T_1$ is an (a, b) -separating set in $G^*(b, a, T_0, T_1)$. Thus, by Theorem 4.4 vertex b cannot transmit a message to a . \square

5 Single-Pair Reliable Communication

In this section we consider (t, ϵ) -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 . Similar ideas appear in [6].

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. The $(\{T_0, T_1\}, \epsilon)$ -protocol is executed although its precondition (that either T_0 or T_1 contain all faulty processors) might be false. If the precondition does not hold, there is no guarantee on the message that b accepts. Furthermore, at the end of the execution, the protocol might not specify which message b should accept, in this case b accepts some default “error” message. Thus, at the end of the protocol b has many messages that it accepted in the different executions; most of them can be garbage but in the executions in which the precondition holds b accepted the message transmitted by a . The message that b accepts as

the message transmitted by a is some message m' such that there is a set T'_0 of size t , for which b accepts the message m' in the $(\{T'_0, T_1\}, \epsilon)$ -reliable protocol for every T_1 of size t .

We claim that with high probability m' was really transmitted by a . Let T_0 be a set of size t that contains all Byzantine parties in the execution, and m be the message transmitted by a . The probability that one of the $(\{T_0, T_1\}, \epsilon)$ -reliable protocols has failed for some T_1 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 accepts the message m in the $(\{T_0, T_1\}, \epsilon)$ -protocol. Assume that there is a set T'_0 such that b accepts the same message m' in the $(\{T'_0, T_1\}, \epsilon)$ -protocol for every T_1 . In particular b receives the message m' in the $(\{T_0, T'_0\}, \epsilon)$ -protocol, and $m = m'$. \square

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

Theorem 5.2 *There is a (t, ϵ) -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 $G^*(a, b, T_0, T_1)$.*

Proof: If there are sets $T_0, T_1 \subseteq V \setminus \{a, b\}$ such that $|T_0|, |T_1| = t$ and the set $T_0 \cup T_1$ is an (a, b) -separating set in $G^*(a, b, T_0, T_1)$, 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 (if T is an (a, b) -separating set in G , then T is an (a, b) -separating set in $G^*(a, b, T, T)$), and by Lemma 5.1 and Theorem 4.4 a can transmit a message to b . \square

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. A very similar protocol appeared previously 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. The communication \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 δ -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 δ . Formally,

Definition 6.1 [Private Protocol] *A message transmission protocol is δ -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: for every set of keys $\{k_{u,v} : \langle u, v \rangle \in E_A; u \in T\}$ and every two sets of random inputs $\{r_u : u \in T\}$ it holds that*

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

The probabilities are taken over the random inputs of the parties in $V \setminus T$ and the keys of pairs of parties in $V \setminus T$. The sum is over Γ the set of all possible communications.

A message transmission protocol from a to b is (t, δ) -private if it is δ -private against every set $T \subseteq V \setminus \{a, b\}$ of size at most t . A protocol is perfectly t -private if it is $(t, 0)$ -private.

We do not consider 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. 9. 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 a sends a different random one-time pad on each of the $t + 1$ paths to b . Using reliable non-private communication and randomized hashing, a and b determine which pads have been received correctly by b (Steps (3)-(5) of the protocol). Then a 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, 19] for the randomized hashing, 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, ϵ) -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 b randomly chooses the unique r_j such that $r_j^a = (d_j^b - d_j^a)/(c_j^a - c_j^b)$. But $m^a = m^b$ unless $c_j^a \neq c_j^b$ for some $j \in G^a$.

To summarize, the transmission fails only if one of the reliable non-private transmissions fail, or $c_j^a \neq c_j^b$ for a $j \in G^a$. Thus, the probability that Protocol PRIVATE_TRANSMISSION fails to be reliable is at most $\frac{2\epsilon}{3} + \frac{t+1}{|\mathbf{F}|} \leq \epsilon$. \square

We argue that the protocol is private with high probability. The idea of the proof is to notice that the adversary does not hear the communication on at least one path j^* , thus has no information on the value of $c_{j^*}^a$. This implies that the adversary has no information on $\sum_{j \in G^a} c_j^a$ and therefore on the transmitted message which is masked with this sum.

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. If a detects that the reliable non-private transmission has failed, it terminates the protocol. Otherwise, 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. Vertex b accepts the message $m^b = z^b - \sum_{j \in G^b} c_j^b$.

Figure 9: A protocol for private transmission of a message.

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: There are at most t Byzantine parties, so there exists a path j^* with no faults on it. We assume that the adversary hears the messages sent on the t remaining paths (this assumption might only help the adversary to get more information about the transmitted message).

First consider a communication γ heard by T in an execution in which the reliable non-private transmission of Step (3) of the protocol has succeeded. We argue that the probability of γ is the same whatever the transmitted message m is. First of all, since the protocol is independent of m in Steps (1)–(4), then for every strategy of the adversary, the set G^a is chosen independently of m . The adversary's only source of information about the transmitted message is z^a . For every m there is exactly one value of $\sum_{j \in G^a} c_j^a$ that is consistent with m and z^a . Furthermore, the adversary knows the value $\sum_{j \in G^a \setminus \{j^*\}} c_j^a$ and the values $r_{j^*}^a$ and $s_{j^*}^a = r_{j^*}^a c_{j^*}^a + d_{j^*}^a$. I.e., given the information of the adversary for every m there is exactly one value for pair $c_{j^*}^a, d_{j^*}^a$ that is consistent with the communication γ , that is $c_{j^*}^a = m - z^a - \sum_{j \in G^a \setminus \{j^*\}} c_j^a$ and $d_{j^*}^a = s_{j^*}^a - r_{j^*}^a c_{j^*}^a$. Since $c_{j^*}^a$ and $d_{j^*}^a$ are chosen uniformly and independently, the probability that $C_T(m) = \gamma$ is independent of m .

If the reliable non-private transmission protocol is perfectly detecting then Steps (5)–(7) are executed only when the transmission of Step (5) has succeeded and Protocol PRIVATE_TRANSMISSION is perfectly private.

However, if the reliable non-private transmission from b to a fails without being detected then privacy may be threatened. 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 . Let Γ_{fail} be all the communications of Protocol PRIVATE_TRANSMISSION in which the transmission in Step (3) failed without a detecting it. Thus,

$$\begin{aligned} \sum_{\gamma \in \Gamma} |\Pr[\mathcal{C}_T(m_0) = \gamma] - \Pr[\mathcal{C}_T(m_1) = \gamma]| &= \sum_{\gamma \in \Gamma_{\text{fail}}} |\Pr[\mathcal{C}_T(m_0) = \gamma] - \Pr[\mathcal{C}_T(m_1) = \gamma]| \\ &\leq \Pr[\text{Step (3) fails while transmitting } m_0] + \Pr[\text{Step (3) fails while transmitting } m_1] \\ &\leq \frac{4\epsilon}{3}. \end{aligned}$$

Thus, by Definition 6.1, Protocol PRIVATE_TRANSMISSION is $(t, 2\epsilon/3)$ -private. \square

Lemma 6.3 and Theorem 3.1 imply 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, ϵ) -reliable and perfectly t -private message transmission between any pair of parties which uses a $(2, \epsilon/(6nt))$ -authentication scheme.*

Theorem 5.2 implies a similar result for single-pair transmission. Since Protocol TRANSMIT is not perfectly detecting, then the protocol is not perfectly private and with small probability the adversary can learn which message was transmitted by a . Notice, that Protocol PRIVATE_TRANSMISSION requires reliable transmission in both directions. This cannot be avoided; if reliable and private transmission 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 which must exist).

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 $G^*(a, b, T_0, T_1)$ and $G^*(b, a, T_0, T_1)$, then for every $\epsilon > 0$ there is a protocol for (t, ϵ) -reliable and $(t, 2\epsilon/3)$ -private message transmission between a and b .*

Acknowledgments: We would like to thank the WDAG '97 program committee and the anonymous referees for many valuable comments.

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] D. Powell (guest editor). Special section on group communication. *CACM*, 39(4), 1996.
- [11] H. Krawczyk. LFSR-based hashing and authentication. In Y. G. Desmedt, editor, *Advances in Cryptology—CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 129–139. Springer-Verlag, 1994.
- [12] H. Krawczyk. New hash functions for message authentication. In L. C. Guillou and J.-J. Quisquater, editors, *Advances in cryptology: EUROCRYPT '95*, volume 921 of *Lecture Notes in Computer Science*, pages 301–310. Springer, 1995.
- [13] N. A. Lynch. *Distributed Algorithms*. Morgan Kaufman Publishers, 1997.
- [14] K. Menger. Allgemeinen kurventheorie. *Fund. Math.*, 10:96–115, 1927.
- [15] M. Pease, R. Shostak, and L. Lamport. Reaching agreements in the presence of faults. *J. of the ACM*, 27(2):228–234, 1980.

- [16] 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.
- [17] 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.
- [18] E. Upfal. Tolerating a linear number of faults in networks of bounded degree. *Information and Computation*, 115(2):312–320, 1994.
- [19] 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.