Levels of Authentication in Distributed Agreement

Malte Borcherding

Institute of Computer Design and Fault Tolerance
University of Karlsruhe
76128 Karlsruhe, Germany
malte.borcherding@informatik.uni-karlsruhe.de

Abstract. Reaching agreement in the presence of Byzantine (arbitrary) faults is a fundamental problem in distributed systems. It has been shown that message authentication is a useful tool in designing protocols with high fault tolerance, but it imposes the additional problem of key distribution.

In the past, agreement protocols using message authentication required complete agreement on all public keys. Because this pre-agreement has to rely on techniques outside the system (e.g., trusted servers which never fail), it is useful to consider lower levels of key distribution which need as few assumptions as possible.

In this paper, we identify several levels of key distribution and describe their properties with regard to the achievable fault tolerance in two agreement problems.

Keywords: Byzantine agreement, crusader agreement, authentication, distributed systems, fault tolerance

1 Introduction

The problem of distributed agreement arises when a set of nodes in a distributed system need to have a consistent view of a message sent by one of them, despite the presence of arbitrarily faulty nodes. Several kinds of agreement have been defined in the past. The most stringent kind of agreement is *Byzantine agreement* (BA) as defined in [LSP82]. It requires that the following three conditions be met:

(B1) All correct nodes agree on the same value.
(B2) If the sender is correct, all correct nodes agree on the value of the sender.
(B3) Each correct node eventually decides on a value.

One variant of this agreement is *crusader agreement* (CA), introduced in [Dol82]. Here, it is not necessary that all nodes agree on the same value if the sender is faulty. It is required, though, that those nodes which do not agree *know* that the sender is faulty:

In: *Proceedings of the 10th International Workshop on Distributed Algorithms (WDAG)*, Bologna, Italy, LNCS 1151, pp. 40–55, Springer-Verlag 1996.

(C1) All correct nodes that do not explicitly know that the sender is faulty agree on the same value.
(C2) If the sender is correct, all correct nodes agree on the value of the sender.
(C3) Each correct node eventually decides on a value or knows that the sender is faulty.

Protocols for distributed agreement are generally divided into two classes: authenticated protocols and non-authenticated protocols. In authenticated protocols, all messages are signed digitally in a way that the signatures cannot be forged and a signed message can be unambiguously assigned to its signer. This mechanism allows a node to prove to others that it has received a certain message from a certain node. Authenticated protocols can tolerate an arbitrary number of faulty nodes. In non-authenticated protocols, no messages are signed. For BA and CA, these protocols require more than two thirds of the participating nodes to be correct ([LSP82, Dol82]).

While authenticated protocols offer the best fault tolerance, it is not at all trivial to distribute the public keys of all participants of an agreement protocol consistently amongst each other. Typically, key distribution requires a single trusted entity, or a group of entities which is completely reliable as a whole ([Gon93]). Since these assumptions restrict the usual assumptions about the participant's behaviour, they should be kept as weak as possible. Hence, it is useful to have a look at possible scenarios with different kinds of key distribution and different common knowledge about these distributions.

2 Model of Computation

In this section we describe the model of computation used throughout this paper. Our world consists of a fully interconnected network with n nodes (processors), t of which may be faulty. In order to avoid special cases, we assume that $n \geq 3$ and $n > t + 1$. For $n < 3$ there are trivial solutions, and for $n \leq t + 1$ agreement always holds by definition.

The nodes operate at a known minimal speed, and messages are transmitted reliably in bounded time. Furthermore, a receiver of a message can identify its immediate sender. Communication takes place in successive *rounds.* In each round a node may send messages to other nodes and receives all messages sent to it in the current round. The actions a node takes in the next round depend solely on the messages it has received so far. We make no assumptions about the *type of failures* that occur. If a node is faulty, it may behave in an arbitrary manner. This type of behaviour is usually referred to as *Byzantine fault.*

In addition, we assume the existence of an unforgeable signature scheme. Examples for signature schemes which are unforgeable with a sufficiently high probability (given today's state of the art) are DSA and RSA [Nat92, RSA78]. In these schemes, a prospective signer has a pair of keys, namely a private key and a public key. The private key is used for signing, while the public key can be used to verify a signature made with the private key.

The assumption of a signature scheme alone is not very strong (see section 3.2). It is often necessary to make assumptions about the distribution of the public keys. The strongest assumption is that of *complete authentication.* It comprises the following four properties:

(A1) If a correct node assigns a signed message to a correct node P, then P has signed the message.
(A2) A message signed by a correct node P is assigned to P by all correct nodes.
(A3) If a message is assigned to a (possibly faulty) node P by a correct node, then all correct nodes assign it to P.
(A4) All correct nodes can sign messages.

In terms of private/public keys, properties (A1) and (A2) state that there is agreement on the public keys of the correct nodes, and the correct nodes keep their secret keys secret. Property (A3) extends the agreement to the public keys of the faulty nodes. Owing to the Byzantine failures, it cannot be assumed that the faulty nodes keep their secret keys secret. Fault models which assume that faulty nodes do not give their secrets away (or sign messages on behalf of others) are used in [GLR95, EM96].

3 Levels of Authentication

Solutions for agreement problems usually assume either no authentication or complete authentication. These assumptions are only two extreme points in a spectrum of possible scenarios. In this section, we will identify several different levels of authentication, give situations in which they arise, and present their properties with regard to the achievable fault tolerance.

3.1 No Authentication

In this situation, no means of key distribution and no signature scheme is available or wanted, e.g., due to lack of processor speed, lack of private local storage or insufficient trust into existing methods. Here applies the long-known requirement $n > 3t$ ([PSL80, LSP82]) which makes agreement between three nodes impossible if one may fail.

3.2 Local ("Byzantine") Authentication

This type of agreement can be reached when no means of agreement on the keys is provided and each node distributes its public key by itself, using a signature scheme. With a challenge-response key distribution protocol, properties (A1) and (A2) can be enforced if (A4) holds and a signature scheme exists ([Bor95]). That is, a faulty node can distribute different public keys to different nodes, but it can not claim a public key of a correct node for itself. This makes local authentication strictly stronger than no authentication: A faulty node can not forge messages

σ_1	$\overset{C}{=}$	σ_3	$\overset{B}{=}$	σ_2
$A:0$				$A:1$
$\Downarrow$				$\Downarrow$
$C:0$	$\Rightarrow$	$C:0 \neq B:1$	$\Leftarrow$	$B:1$

Fig. 1. Proof of Theorem 1

sent by correct nodes without invalidating the signature. Furthermore, a message signed by a correct node cannot be misattributed to a faulty node.

Although no complete agreement on the public keys of the *faulty* nodes can be guaranteed, this level of authentication has been shown to be useful for Failure Discovery, a sub-problem of Byzantine agreement ([HH93, Bor95]). Using local authentication, Byzantine agreement with few messages in the failure-free runs is possible.

Unfortunately, in this setting the impossibility of Byzantine agreement for $n \leq 3t$ cannot be overcome, as stated in the following theorem.

Theorem 1. *It is not possible to reach Byzantine agreement if one third of the nodes is faulty and only local authentication is assumed.*

Proof. This proof is a variation of the proof in [LSP82]: Let $n \leq 3t$. Then it is possible to partition the nodes into three nonempty sets A, B, and C, such that each set contains at most t elements. The members of these sets will be denoted a_i, b_i, and c_i, respectively. The sender will always be a_1.

We consider the most general protocol with an arbitrary number of rounds: If a correct node receives a message in round r, it signs this message and sends it to all nodes in round $r + 1$. Thus, the maximum possible information flow is achieved. We do not specify how the information is used to reach the final decision. We will only observe whether or not a node receives different messages in different runs of a protocol. If the messages are the same, the node has to draw the same conclusions in the respective runs.

We will consider three possible scenarios σ_1, σ_2, and σ_3. In σ_1, the nodes in B are faulty, and a_1 sends 0. In σ_2, the nodes in C are faulty, and a_1 sends 1. In σ_3, the nodes in A are faulty, and a_1 sends send 0 to C and 1 to B.

Hence, in σ_1, the nodes in C have to decide for 0, and in σ_2, the nodes in B have to decide for 1. We will show that in σ_3, the nodes in B receive the same messages as in σ_2, while the nodes in C receive the same messages as in σ_1. So, in σ_3, the nodes in B and C have to decide for different values, which contradicts (B1). An example with tree nodes will be given below.

We will use the following notation for the messages: 0_{a_1BCB} means that the value 0 was first signed by a_1, then by a node in B, followed by a node in C, and then again by a node in B. Small letters denote actual signatures, while capitals represent the signature of some member of the respective set. For nodes in A there will be two sets of signatures. They will be denoted A_B (a_{iB}) and A_C (a_{iC}), respectively.

The three scenarios will be constructed in a way that the signatures on all messages seen by correct nodes in B and C are of the following form:

- The first signature is a_{1B} if the value is 1, and a_{1C} if the value is 0.
- A_B is followed by B or A_B.
- A_C is followed by C or A_C.
- B and C are followed by A_B, A_C, B, or C.

Signatures A_B and A_C cannot be recognized by correct nodes in C and B, respectively. The behaviour of the nodes in the three scenarios is as follows:

Scenario σ_1 (B faulty): All nodes distribute consistent keys in the key distribution protocol. The nodes in A sign all messages with A_C. In the first round of the agreement protocol, a_1 signs the value 0 and sends it to all nodes. In the following rounds, the nodes in A and C sign all messages correctly and send them to all nodes.
The nodes in B treat all messages from B and C correctly. When a node b_i receives a message from A, it replaces those signatures A_C, which are consecutively at the end, with A_B. The signatures A_B are chosen arbitrarily by the nodes in B and cannot be recognized by nodes in A or C. If all signatures on the message are from nodes in A, the value is set to 1 before substituting the signatures. Finally, b_i signs the message correctly and sends it to all nodes. If, for example, b_i receives $0_{a_{1C}c_2a_{4C}a_{2C}}$, it will echo the message as $0_{a_{1C}c_2a_{4B}a_{2B}b_i}$, while a message $0_{a_{1C}}$ becomes $1_{a_{1B}b_i}$.

Scenario σ_2 (C faulty): All nodes distribute consistent keys in the key distribution protocol. The nodes in A sign all messages with A_B. In the first round of the agreement protocol, a_1 signs the value 1 and sends it to all nodes. In the following rounds, the nodes in A and B sign all messages correctly and send them to all nodes.
The nodes in C treat all messages from B and C correctly. When a node c_i receives a message from A, it replaces those signatures A_B, which are consecutively at the end, with A_C. The signatures A_C are chosen arbitrarily by the nodes in C and cannot be recognized by nodes in A or B. If all signatures on the message are from nodes in A, the value is set to 0 before substituting the signatures. Finally, c_i signs the message correctly and sends it to all nodes. If, for example, c_i receives $1_{a_{1B}b_2a_{4B}a_{2B}}$, it will echo the message as $1_{a_{1B}b_2a_{4C}a_{2C}c_i}$, while a message $1_{a_{1B}}$ becomes $0_{a_{1C}c_i}$.

Scenario σ_3 (A faulty): In the key distribution protocol, the nodes in B and C distribute consistent keys. The nodes in A send different keys A_B and A_C to the nodes in B and C.
In the first round of the agreement protocol, a_1 sends $1_{a_{1B}}$ to B and $0_{a_{1C}}$ to C. In the following rounds, the nodes in B and C sign all messages correctly and send them to all nodes. The nodes in A also sign all messages and send them to all nodes. Before sending a message to B, though, they replace those signatures A_C which are consecutively at the end, with A_B. If there are only signatures by nodes in A, the value is set to 1. Likewise, in

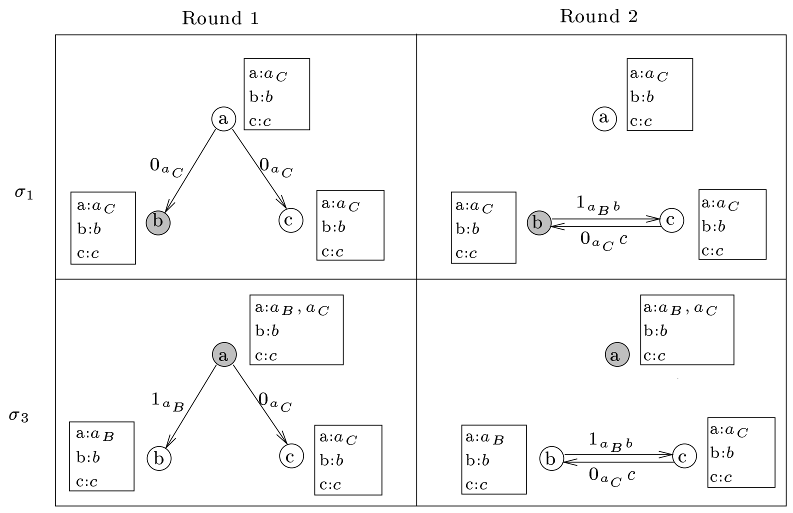

Fig. 2. Impossibility of Byzantine agreement with local authentication

messages to C, signatures A_B are replaced by A_C, and the value is set to 0. This is possible since the nodes in A may cooperate. If, for example, a_i receives $1_{a_{1B}b_2a_{4B}a_{2B}}$, it will forward this message to C as $1_{a_{1B}b_2a_{4C}a_{2C}a_{iC}}$.

It can easily be seen that for each message received by a node in B in scenario σ_2, there is an indistinguishable message in scenario σ_3, and vice versa. The same holds for the nodes of C in scenarios σ_1 and σ_3. Since a node b_i has to decide for 1 in σ_2, and a node c_i has to decide for 0 in σ_1, they decide for different values in σ_3. This violates (B1). This reasoning is depicted in Fig. 1. $\square$

Example 1. The idea of the proof will be demonstrated at the most simple example: There are three nodes a, b, and c. One of them may be faulty, and the sender is a. Scenarios σ_1 and σ_3 are depicted in Fig. 2. Messages to and from a in the second round are omitted, as well as self-addressed messages. The boxes at each node show the respective views of the public keys, and faulty nodes are shadowed.

If c is correct, it will receive 0_{a_C} in the first round of σ_1 and σ_3. In the second round, it will receive $0_{a_C a_C}$, $0_{a_C c}$, and $1_{a_B b}$, where a_B is not recognizable to c. From these messages, c can deduce that one of the following cases must hold:

1. a is correct and has sent 0. The unrecognizable signature has been produced by b. In this case, c has to decide for 0.
2. b is correct, but a has sent different values to b and c. Furthermore, a has sent different public keys to b and c. Then, c has to decide for the same value as b.

In order to fulfill (B2), c has to decide for 0. If a is faulty, b sees messages

Protocol for Crusader Agreement

1. The sender signs its value and sends it to all others.
2. If a node recognizes the signature, it transmits the received value and signature to all others. Otherwise, it decides for a faulty sender.
3. If a node has not decided in the second step, it looks at the values signed by the sender it has seen so far. If there is exactly one distinct value, it decides for that value. Otherwise, it decides for a faulty sender.

Fig. 3. Crusader agreement for crusader authentication

consistent with σ_2 and decides for 1, which violates (B1). If b and c always decide for some default value, one of them violates (B1) in σ_1 or σ_2.

The proof for the achievable fault tolerance for crusader agreement is similar to the proof given above. Hence, to reach crusader agreement under local authentication, more than two thirds of correct nodes are necessary.

3.3 Sound, but Incomplete Authentication

For this kind of authentication we assume that no two correct nodes have different public keys of a third node. However, a node may not know the public keys of all other nodes. In this setting, two cases can be distinguished. In the first case, there is no agreement on who knows which keys, while in the second case, this agreement is assumed.

Crusader Authentication Here, we assume that only the public keys of the faulty nodes are distributed incompletely, but it is not known who actually knows whose public key.

This requires a more benign behaviour of the faulty nodes during the key distribution process. A faulty node may still choose not to send its public key to some other nodes, but it does not distribute different keys. Apart from that, it may behave arbitrarily. Instead of (A3), (A3)$'$ holds in this context:

(A3)$'$ All correct nodes which assign a certain message to a node assign it to the same node.

We will refer to this level of authentication as *crusader authentication*, since the public keys of the nodes are agreed upon like values sent by crusader agreement: If a correct node A has a key of B, it is the correct one. Otherwise, A knows that B is faulty.

In such a setting, *crusader agreement* is possible for any number of faulty nodes. It can be reached in only two rounds (see Fig. 3).

Theorem 2. *The protocol in Fig. 3 reaches crusader agreement for any* $n \geq t+2$ *if crusader authentication holds.*

Proof. (C1) is trivially fulfilled for those nodes which decide that the sender is faulty. Now assume that two correct nodes decide for different values. Then both must have received their values with the sender's signature in the first round. Furthermore, they must not have received a different signed value in the second round. But that is impossible, since both must have sent their values to all others in the second round.

(C2) is fulfilled, because a correct sender signs its value in the first round and sends it to all nodes. Since the sender is correct, all nodes assign the message correctly and send it to all others in the second round. There exists exactly one value signed by the sender, so all correct nodes will decide for that value. (C3) is fulfilled by the limited number of rounds. □

Byzantine agreement can be reached with $n \geq 2t + 1$. The protocol in Fig. 4 has this fault tolerance. It is a variation of the Exponential Information Gathering *(EIG)* protocol which was introduced by Bar-Noy *et al.* [BDDS87], based on the protocol in [LSP82]. In this protocol, the sender starts by sending its value to all other nodes. In the following t rounds, each node signs and forwards messages received in the previous round to the other nodes.

During protocol execution, each node maintains an *EIG* tree which contains the received information in a structured manner. Such a tree has $t+1$ levels, one level per communication round. The root has $n-1$ children, and in each of the following levels, the vertices have one child less than those of the previous level. Hence, on level r, each vertex has $n-r$ children (we consider the root level as level 1). A part of such a tree for $n = 7$ is shown in Fig. 5.

The vertices have labels which are assigned in the following manner: The root is labeled with the sender's name. In the following levels, the children of a vertex are labeled with the names of the nodes not yet on the path from the root. We identify a vertex in the tree by the the sequence of labels from the root to the respective vertex. Note that in no such sequence a node's name appears twice. A vertex labeled with the name of a correct (faulty) node will be called a correct (faulty) vertex. From the construction, a vertex on level r has at most t faulty children and at least $n-r-t$ correct children.

In the first round of the protocol, each node stores the value received from the sender in the root of its *EIG* tree. In the following rounds, each correct node broadcasts the contents of the level of its tree most recently filled in, and fills the next level with messages it receives. If a node X receives a message from Y claiming that it has stored v in vertex $ABCDE$, X stores v in vertex $ABCDEY$ of its *EIG* tree. Hence, a value v in vertex $ABCDEY$ is interpreted as "Y said E said ... B said A said v". If a node failed to send a value, a default value is stored.

Due to the structure of the tree, not all received messages are stored. Those messages in which a node reports about a message which was once sent by itself are ignored. In Fig. 5, labels and stored values are separated by a colon. The faulty vertices are shadowed.

When a node has completed its tree (after round $t+1$), it uses the collected data to decide for the outcome of the protocol. This is done by *resolving* each

Protocol for Byzantine Agreement

1. The nodes fill their EIG trees for $t+1$ rounds. They only consider messages which carry the recognizable signature of the immediate sender.
2. A leaf is resolved to its value with the last signature removed.
3. For a non-leaf on level $r < t+1$ with label X, two cases are distinguished:
   - The signature of X is known: Consider the set of resolved children which are signed correctly. If it has at least $t-r+1$ members, take the (relative) majority value. The vertex is then resolved to that value without the last signature. If the set has fewer members, the vertex is resolved to a default value.
   - Signature of the X is unknown: Take a set of maximum size of resolved children which carry the same signature. If it contains at least $t-r+1$ elements, take the relative majority. The vertex is then resolved to that value, with the last signature removed. If no such set exists, the vertex is resolved to a default value.
4. The result of the protocol is the resolved value of the root.

Fig. 4. Byzantine agreement for crusader authentication

vertex to a certain value, depending on the resolved values of its children. The exact rules are given in steps 2 and 3 of the protocol. A vertex which is resolved to the same value by all correct nodes will be called *common*. The following example will demonstrate the rules for resolving.

Example 2. Let $n = 7$ and $t = 3$ (see Fig. 5). Let us further assume that the five children of a vertex at level $r = 2$ are resolved to the following values: default, default, $1_{ab}, 2_{ab'}, 2_{ab'}$. A reducing node who knows the signature of b notices that only one of these values has been signed correctly. Since $1 < t-r+1 = 2$, it will choose the default value.

A node which does not know the signature of b uses values $2_{ab'}$ and $2_{ab'}$ for its decision, since they constitute the largest set of values with the same signature, and the set has two elements[1]. The majority value is $2_{ab'}$, and the vertex will be resolved to 2_a. Since two correct nodes can reduce this vertex to different values, it is not common.

The following two lemmas will be used to prove the correctness of the protocol:

Lemma 3. *Assuming crusader authentication and $n \geq 2t+1$, the following holds: A correct vertex is resolved to its stored value, with the last signature removed.*

[1] We assume implicitly that it is possible to decide whether two signatures were made with the same private key. This can be achieved when each signature is required to come with the respective public key.

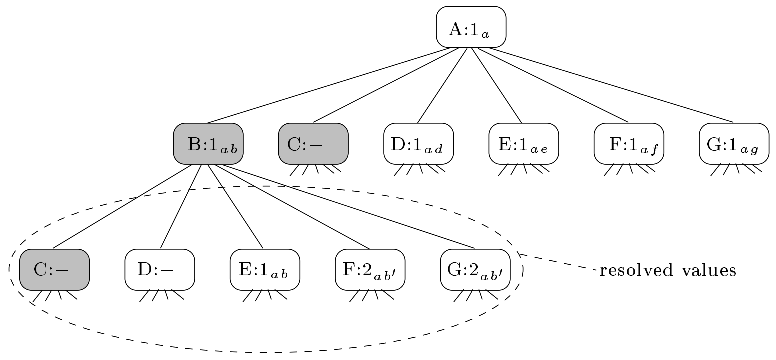

Fig. 5. Part of an EIG tree

Proof. The proof is by induction on the levels of the tree, from the leaves to the root. For a leaf, the lemma is trivially correct. Now suppose that for all vertices above level r the lemma is true. Then it is also true for a correct vertex on level r, because it has at least $t - r + 1$ correct children (due to $n \geq 2t + 1$).

From the induction hypothesis, all of them will be resolved to their stored values. Since the vertex on level r is correct, its stored value is the same as the resolved values of the children. Furthermore, the signatures on the values are correct. Hence, the vertex will be resolved as stated in the lemma. □

Lemma 4. *Assuming crusader authentication and $n \geq 2t + 1$, the following holds: A faulty vertex which has only faulty ascendants*[2] *is common.*

Proof. The proof is again by induction on the levels from the leaves to the root. On the last level $(t + 1)$, the lemma is true, because there is no such vertex. Now suppose that for all vertices above level r the lemma is true. We will now show that the lemma is true for the vertices on level r.

Consider a faulty vertex on level r which has only faulty ascendants. With the induction hypothesis and Lemma 3, all children of this vertex are common. Then all correct nodes resolve that vertex to the same value, because they will base their decision on the same set of values. This can be shown as follows: A set of $t - r + 1$ elements can not contain only values from faulty vertices, since a faulty vertex with faulty ascendants in level r has at most $t - r$ faulty children. But if there is one value from a correct vertex in the set, then all values carry the correct signature, since a correct node will only forward values with correct signatures. □

Theorem 5. *Assuming crusader authentication and $n \geq 2t + 1$, the following holds: The algorithm in Fig. 4 reaches Byzantine agreement.*

Proof. If the sender is correct, the root of the tree will be resolved to the value received in the first round by all correct nodes, due to Lemma 3. Hence, all correct nodes agree on the correct value.

[2] An ascendant is a vertex on the direct path to the root.

If the sender is faulty, the root is common due to Lemma 4. Hence, all correct nodes agree on the same value. □

The next theorem shows that the fault tolerance of $n \geq 2t + 1$ cannot be improved.

Theorem 6. *Byzantine agreement cannot be reached for $n \leq 2t$, if only crusader authentication holds.*

Proof. Suppose that $n \leq 2t$ holds. Then the nodes can be partitioned in four nonempty sets A, B, C, and D, such that no set contains more than $t/2$ members. Now consider four scenarios, where in each scenario the members of two sets behave faulty. The sender is always a_1.

In all four scenarios, the nodes in C will play the role of those who do not know the public keys of nodes in A. If A is correct, they will only pretend not to know the keys. If A if faulty, they will actually not know the keys.

Scenario σ_1 (C and D faulty): All public keys are known to all nodes. In the first round, a_1 sends 0_{a_1} to all, and the nodes in A and B forward all messages correctly.
The nodes in C behave correctly, except that they pretend not to know the public keys of the nodes in A. The nodes in D behave towards the nodes in C as follows: They pretend to receive from nodes in A messages of the form $m_{A' \ldots A'}$, where A' is recognizable to D. Furthermore, they forward messages $1_{A \ldots A}$ to C as $0_{A' \ldots A' D}$. Apart from this, they behave correctly.

Scenario σ_2 (B and C faulty): All public keys are known to all nodes. In the first round, a_1 sends 1_{a_1} to all, and the nodes in A and D forward all messages correctly.
The nodes in C behave correctly, except that they pretend not to know the public keys of the nodes in A. The nodes in B behave towards the nodes in C as follows: They pretend to receive from nodes in A messages of the form $m_{A' \ldots A'}$, where A' is recognizable to B. Furthermore, they forward messages $0_{A \ldots A}$ to C as $1_{A' \ldots A' B}$. Apart from this, they behave correctly.

Scenario σ_3 (A and B faulty): All public keys except those from nodes in A are known to all nodes. The keys of A are known to all except the nodes in C.
During the agreement protocol, the nodes behave exactly as in σ_2.

Scenario σ_4 (A and D faulty): All public keys except those from nodes in A are known to all nodes. The keys of A are known to all except the nodes in C.
During the agreement protocol, the nodes behave exactly as in σ_1.

The nodes in the respective sets have the following views (see Fig. 6):

- The nodes in B cannot distinguish σ_1 from σ_4. Hence, they decide in both scenarios for 0.
- The nodes in C cannot distinguish σ_3 and σ_4. Since the nodes in B decide for 0 in σ_4, the nodes in C have to decide for 0 in σ_3 and σ_4.

σ_1	$\overset{B}{=}$	σ_4	$\overset{C}{=}$	σ_3	$\overset{D}{=}$	σ_2
$A:0$						$A:1$
$\Downarrow$						$\Downarrow$
$B:0$	$\Rightarrow$	$B:0$		$D:1$	$\Leftarrow$	$D:1$
		$\Downarrow$		$\neq$		
		$C:0$	$\Rightarrow$	$C:0$		

Fig. 6. Proof of Theorem 6

- The nodes in D cannot distinguish σ_2 and σ_3. Hence, they decide in both scenarios for 1.

With this reasoning, the nodes in C and D decide for different values in σ_3, which contradicts (B1). $\square$

Partial Authentication In this setting it is common knowledge whose public keys are distributed completely. For simplicity, we will only regard the case in which a node's public key is either known to all nodes or to none. This situation arises when the keys are distributed by a globally trusted server which does not know all keys, or when some of the nodes are not able to sign messages. There is no relationship between the state of a node (faulty/correct) and the distribution of its public key. Here, (A3) holds, but (A4) becomes

(A4)$''$ Only a known set of s nodes can sign messages.

We will call this level of authentication *partial authentication*. A node whose public key is known is called *signer*. We can distinguish whether or not the sender is a signer.

Sender is no signer If $n > s \geq 2t + 1$, the sender sends its value to the signers in the first round. Then the signers distribute this value via an authenticated protocol with all other non-signing nodes as bystanders. The final result is the majority of the results of these protocols.

Such a protocol, where some node do not send any messages (and hence do not sign), is described in [DS83] (Theorem 6). In this protocol, the non-signing nodes have to draw their conclusions from the messages they receive from the signers. The protocol runs for $t+1$ rounds. For our purposes, the protocols could also be run for only t rounds.

A sketch of the proof is as follows: If the sender is faulty, there are at most $t - 1$ faulty signers left, and all correct nodes will agree on the values of the s sub-protocols. Hence, all correct nodes will eventually agree on the same value.

If the sender is correct, all correct signers will send the same values in their respective sub-protocols. These protocols have the property that the nodes reach agreement on the values of the correct nodes, even if the number of rounds is not greater than the number of faulty nodes. Hence, a majority of the results of the sub-protocols will be the correct value.

For $s \leq 2t$, no better fault tolerance than in the unsigned case can be achieved. This is shown in the following theorem:

Theorem 7. *If the sender of a protocol does not sign its messages and $s \leq 2t$ holds, then Byzantine agreement cannot be reached with $n \leq 3t$, assuming partial authentication.*

Proof. Suppose $n \leq 3t$ and $s = 2t$. Then it is possible to partition the nodes into three nonempty sets A, B, and C, such that A contains exactly the non-signers and no set contains more than t elements. The sender is a_1. Now consider three scenarios σ_1, σ_2, and σ_3:

Scenario σ_1 (B faulty): a_1 sends 0 in the first round. When a node b_i receives a message from A without any signatures (i.e., the message passed only nodes in A), it forwards 1_{b_i} to all nodes. All other messages are correctly signed and forwarded.

Scenario σ_2 (C faulty): a_1 sends 1 in the first round. When a node c_i receives a message from A without any signatures (i.e., the message passed only nodes in A), it forwards 0_{c_i} to all nodes. All other messages are correctly signed and forwarded.

Scenario σ_3 (A faulty): In the first round, a_1 sends 0 to C and 1 to B. In the remainder of the protocol, a node a_i forwards a signed message unchanged to all nodes. Unsigned messages are passed to B as 1 and to C as 0.

In σ_1, the nodes in C have to decide for 0, since the sender is correct. For the same reason, the nodes in B have to decide for 1 in σ_2. Scenario σ_3 has been constructed in a way that it is indistinguishable from σ_1 by C and from σ_2 by B. Hence, the nodes in B and C decide for different values in σ_3, contradicting (B1). □

With a similar argument, it can be shown that the same restrictions apply for crusader agreement.

Sender is signer For $n \geq s \geq 2t + 1$, the protocol from [DS83] mentioned above could be used. With the (less efficient) EIG-protocol in Fig. 7, one signer can be omitted, such that Byzantine agreement is possible for $n > s \geq 2t$.

Theorem 8. *Assuming partial authentication, the following holds: If the sender signs and $n > s \geq 2t$ holds, the protocol in Fig. 7 reaches Byzantine agreement.*

Proof. We distinguish whether the sender is correct or faulty:

Sender correct: There is exactly one value signed by the sender. Since there are at least $t + 1$ correct nodes, at least t of which are signers, at least one leaf is reduced to the sender's value signed by t correct nodes. This value will be considered recursively in the process of resolving. Hence, there is exactly one value for the result of the protocol, which is the value signed by the sender.

Protocol for Byzantine Agreement

1. The nodes fill their EIG trees for $t + 1$ rounds.
2. A leaf is resolved to its value. If the last sender was a signer, the signature is removed.
3. For a vertex on level $r < t+1$ with label X, two cases are distinguished:
   - X is signer: Take the majority of the vertice's resolved children which carry X's signature.
   - X is no signer: Take the majority of the vertice's resolved children.
4. The result of the protocol is the resolved value of the root.

Fig. 7. Byzantine agreement with partial authentication and a signing sender

Sender faulty: Here, we show that all vertices in the second level are common. The vertices which correspond to correct signers are common with the same argumentation as in the case of a correct sender.

The vertices which correspond to correct non-signers are common, because they have at least t correct and signed children (as opposed to at most $t-1$ faulty children). Since these vertices are common (same argumentation as above), there is always a majority of correctly resolved children.

Finally, there are the faulty vertices (signed or not). Their correct children are common, as can be shown as above. For their faulty children to be common, it is (by recursion) necessary that all faulty vertices at level $t+1$ with only faulty ascendants be common. This is the case, since there are no such vertices.

Hence, all correct nodes take the same set of values as a basis for the final decision. This leads to a common value. □

For $n = s = 2t$, any protocol for complete authentication can be used. Hence, Byzantine agreement is possible for any $n \geq s \geq 2t$. For $0 < s \leq 2t-1$, $n \geq 3t-1$ (or $n = s$) is necessary:

Theorem 9. *Under the assumption of partial authentication the following holds: If the sender signs its messages and* $0 < s \leq 2t - 1$ *holds, then Byzantine agreement is only possible if* $n \geq 3t - 1$ *or* $n = s$.

Proof (Sketch). Byzantine agreement is always possible with $n = s$ ([LSP82]). Now suppose $n > s = 2t - 1$ and $n \leq 3t - 2$ holds. From Theorem 7 follows that it is generally impossible to reach Byzantine agreement on the messages sent by the non-signers. Hence, the non-signers may as well send no messages at all. Now let the faulty signers, except the sender, be in A, the correct signers in B, and the non-signers in C.

If the sender is faulty, A can behave towards C as if the sender sends “1”, while B receives (and sends) messages consistent with the sender saying “2”. The nodes in C cannot distinguish whether A or C is faulty, although they have to agree with the correct nodes. □

Crusader agreement is possible for any number of faulty nodes. It can easily be verified that the protocol of Fig. 3 reaches crusader agreement under the assumption of partial authentication and a signing sender.

3.4 Complete Authentication

In this setting it is assumed that all nodes agree on the public keys of all nodes. If $n \leq 3t$, this agreement may be reached using a trusted entity at the time of set-up (e.g., a trusted person traveling from site to site) or dynamically by using a trusted server. It is worth noting that if a trusted server would exist during execution of the protocols, it could also help solving Byzantine agreement very easily. With complete authentication, $n > t$ is possible. This case is dealt with to a great extent in the original papers ([PSL80, LSP82]), and in [DS83].

If $n \geq 3t+1$, one should install complete authentication by agreement on the public keys at the very beginning. Once this level of authentication is reached, an arbitrary number of nodes may become faulty without disturbing agreement. In addition, with complete authentication, very simple and message-efficient protocols become possible.

4 Summary

In this paper, we have focused on protocols for Byzantine agreement and crusader agreement in the presence of incomplete and incorrect authentication. We have identified several possible scenarios which yield different degrees of maximum fault tolerance.

	BA	**CA**
No auth.	$3t+1$ [LSP82]	$3t+1$ [Dol82]
Byz. auth.	$3t+1$	$3t+1$
Crus. auth.	$2t+1$	t
Partial auth., Sender doesn't sign	$2t+2$ if $s \geq 2t+1$; $3t+1$ else	$2t+2$ if $s \geq 2t+1$; $3t+1$ else
Partial auth., Sender signs	$2t$ if $s \geq 2t$; $3t-1$ if $s < 2t \wedge s < n$	t
Complete auth.	t [LSP82]	t

Table 1. Minimal n for different levels of authentication and types of agreement

Table 1 gives an overview of the results. It shows minimum values for n (number of nodes) with regard to t (maximum number of faulty nodes). In the results for partial authentication, s denotes the number of signing nodes.

In the past, only the two extreme levels "no authentication" and "complete authentication" have been investigated. As can be seen, there is a trade-off between the level of authentication and the possible fault tolerance. The stronger the authentication, the higher the possible fault tolerance. Furthermore, we have shown that there are environments where the two agreement problems under consideration do not have the same fault tolerance.

References

[BDDS87] Amotz Bar-Noy, Danny Dolev, Cynthia Dwork, and H. Raymond Strong. Shifting gears: Changing algorithms on the fly to expedite Byzantine agreement. In *Proceedings of the 6th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 42–51, Vancouver, Canada, 1987.

[Bor95] Malte Borcherding. Efficient failure discovery with limited authentication. In *Proceedings of the 15th International Conference on Distributed Computing Systems (ICDCS)*, pages 78–82, Vancouver, Canada, 1995. IEEE Computer Society Press.

[Dol82] Danny Dolev. The Byzantine generals strike again. *Journal of Algorithms*, 3(1):14–30, 1982.

[DS83] Danny Dolev and Raymond Strong. Authenticated algorithms for Byzantine agreement. *SIAM Journal of Computing*, 12(5):656–666, November 1983.

[EM96] Klaus Echtle and Asif Masum. A mutiple bus braodcast protocol resilient to non-cooperative Byzantine faults. In *Proceedings of the 26th International Symposium on Fault-Tolerant Computing (FTCS)*. IEEE Computer Society Press, 1996.

[GLR95] Li Gong, Patrick Lincoln, and John Rushby. Byzantine agreement with authentication: Observations and applications in tolerating hybrid and link faults. In *Proceedings of the Fifth Dependable Computing for Critical Applications (DCCA-5)*, 1995.

[Gon93] Li Gong. Increasing availability and security of an authentication service. *IEEE Journal on Selected Areas in Comunications*, 11(5):657–662, June 1993.

[HH93] Vassos Hadzilacos and Joseph Y. Halpern. The Failure Discovery problem. *Math. Systems Theory*, 26:103–129, 1993.

[LSP82] Leslie Lamport, Robert Shostak, and Marshall Pease. The Byzantine Generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, 1982.

[Nat92] National Institute of Standards and Technology. The Digital Signature Standard. *Communications of the ACM*, 35(7):36–40, July 1992.

[PSL80] M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *Journal of the ACM*, 27(2):228–234, April 1980.

[RSA78] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

This article was processed using the LaTeX macro package with LLNCS style