On the Number of Authenticated Rounds in Byzantine Agreement

Malte Borcherding

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

Abstract. Byzantine Agreement requires a set of nodes in a distributed system to agree on the message of a sender despite the presence of arbitrarily faulty nodes. Solutions for this problem are generally divided into two classes: authenticated protocols and non-authenticated protocols. In the former class, all messages are (digitally) signed and can be assigned to their respective signers, while in the latter no messages are signed. Authenticated protocols can tolerate an arbitrary number of faults, while non-authenticated protocols require more than two thirds of the nodes to be correct.
In this paper, we investigate the fault tolerance of protocols that require signatures in a certain number of communication rounds only. We show that a protocol that is to tolerate one half of the nodes as faulty needs only few authenticated rounds (logarithmic in the number of nodes), while tolerating more faults requires about two authenticated rounds per additional faulty node.

Keywords: Byzantine Agreement, fault tolerance, distributed systems, authentication

1 Introduction

The problem of Byzantine Agreement (introduced in [LSP82]) arises when a set of nodes in a distributed system need to have a consistent view of messages uttered by one of them, despite the presence of arbitrarily faulty nodes. The problem is defined as follows: One of the nodes is distinguished as *sender* who attempts to transmit a value to the rest of the nodes. A protocol solving Byzantine Agreement must fulfill the following conditions:

- Each correct node eventually decides for a value.
- All correct nodes decide for the same value.
- If the sender is correct, all nodes decide for the value of the sender.

Protocols solving Byzantine 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

In *Proceedings of the WDAG-9, Le Mont Saint-Michel, France, 1995*, pp. 230–241, LNCS 972, Springer-Verlag.

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. These protocols require more than two thirds of the participating nodes to be correct ([LSP82]).

While signatures allow for very fault-tolerant protocols, signing messages is a time-consuming action; typical durations for an RSA signature with a 512 bit key are 50 to 100 ms. Hence, it is useful to investigate the fault tolerance properties of authenticated protocols which require as few signatures as possible.

In this paper, we have a closer look at protocols where the nodes have to sign messages in certain communication rounds only. One implication of our results is that tolerating one half of the nodes as faulty requires a number of authenticated rounds logarithmic in the number of nodes, while tolerating more faults needs about two authenticated rounds per additional faulty node.

2 Preliminaries

2.1 System Model

Our world consists of n nodes connected by a complete network. We assume that t of the nodes may behave in an arbitrary manner, while $c = n - t$ behave correctly. The nodes operate at a known minimal speed, and messages are transmitted reliably in bounded time. The receiver of a message can identify its immediate sender, and we assume the existence of an authentic signature scheme such that a signature cannot be forged and each node knows whom a signature on a message belongs to.

During a protocol execution, the nodes communicate in successive *rounds*. In each round, a node may send messages to other nodes, receive the messages sent to it in the current round and perform some local computation. m of the rounds $(s_1, \ldots, s_m)$ are distinguished as *authenticated rounds*. In these rounds, all messages are to be signed.

2.2 The *EIG* Protocol

Our examinations are based on the Exponential Information Gathering *(EIG)* protocol which was introduced by Bar-Noy *et al.* [BNDDS87], 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 forwards all messages it received in the previous round to the other nodes[1].

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

[1] In [FL82, DS83] it is shown that $t+1$ rounds are necessary and sufficient to reach agreement with or without authentication.

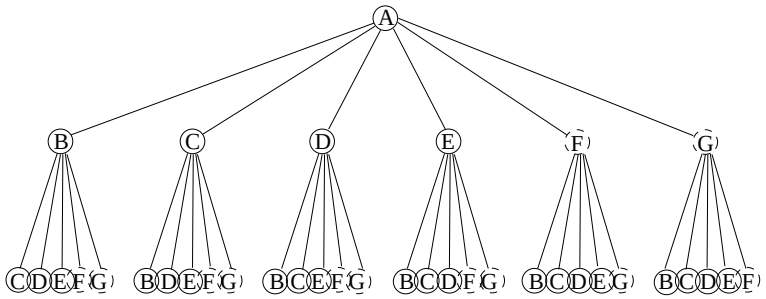

Fig. 1. An *EIG* tree

following levels, the vertices have one child less than those of the previous level. Hence, on level t, each vertex has $n-t$ children which constitute the leaves of the tree (we consider the root level as level 1).

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 labels of the vertices from the root to the vertex in question. Figure 1 depicts such a tree for $n = 7$. In this example, we assume that F and G are faulty; we have marked the respective vertices with dashed circles.

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 the 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 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. A level that corresponds to an authenticated round will be called *authenticated level.*

After a node has completed its tree (after round $t+1$), it applies the following *resolve function* to the vertices: The resolved value for a vertex its stored value iff it is a leaf, and the majority of the resolved values of its children otherwise. If there is no such majority, a default value is used. The value a node eventually uses as decision value is the resolved value of the root.

We require that a vertex on an authenticated level is to be resolved using values signed by the node corresponding to the label of the vertex. Messages without these signatures are not considered in the resolve function (they are not even assigned a default value). If all children of a vertex are removed due to missing correct signatures, it is treated as a leaf and resolved to its own value. A vertex is called *common*, if its resolved value is the same in the trees of all correct nodes for all possible executions of the protocol.

2.3 Compact *EIG* trees

For our purposes, we use *binary* trees as a more compact representation of the original *EIG* trees. These binary trees only contain information about *numbers* of correct and faulty children of a vertex, rather than actually listing all children.

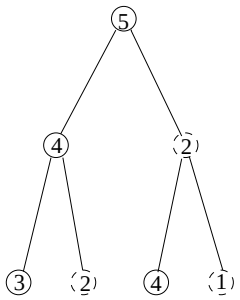

Fig. 2. Compact representation of an *EIG* tree

Figure 2 depicts the compact representation of the tree of Figure 1. It is interpreted as follows: We assume that five nodes, including the sender, are correct, and two nodes are faulty. Hence, the message of the sender is echoed by four correct nodes and two faulty nodes. The echoes of the correct nodes are echoed by three correct and two faulty nodes, those of the faulty nodes are echoed by four correct and one faulty node(s). So a vertex in the compact tree represents a class of vertices of the original *EIG* tree. Each class comprises vertices with the same sequence of faulty/correct vertices on the path to the root. We call a vertex in the compact tree *common*, if all the represented vertices of the *EIG* tree are common; a vertex in the compact tree is said to be *resolved to its stored value* if this is true for all represented vertices of the original *EIG* tree. A vertex is called *correct*, if it represents messages of correct nodes, and *faulty* otherwise. When referring to the levels of such a tree, we will start with level 1 for the root level. Levels corresponding to authenticated rounds will be called *authenticated levels*.

Each vertex in a tree is identified by a binary string. It represents the (unique) path from the root to the vertex, including both ends: If we pass a correct vertex, we append a "1" to the string, and a "0" otherwise. So, in our example, the vertex with the value 2 on the third level is identified by the string "110". We will write x^k for a succession of k equal characters x. As variables for strings, we will use σ and τ.

On these strings, we define functions $|\sigma|_0$, $|\sigma|_1$, and $|\sigma|$, giving the numbers of 0s and 1s in the string, and its length, respectively. Furthermore, we have a function $v(\sigma)$ which gives the label of the vertex. It is defined as follows:

$$v(\sigma 0) = \max(0, t - |\sigma|_0)$$
$$v(\sigma 1) = \max(0, c - |\sigma|_1).$$

Lemma 1. *The vertices of a binary EIG tree have the following properties:*

(a) $v(\sigma 0) = 0 \Leftrightarrow \sigma = 0^t$
(b) $v(\sigma 1^k 1) > v(\sigma 1^k 0) \Rightarrow \forall \tau, |\tau| \leq k : v(\sigma\tau 1) > v(\sigma\tau 0)$

Proof. Immediate from the definition of $v(\cdot)$. □

3 Correctness Requirements

If we express the conditions for Byzantine agreement in terms of resolving vertices of the *EIG* trees, we have the following requirements: The *EIG* protocol is correct iff

- In every execution of the protocol, the root is common and
- if the sender is correct, every node resolves the root of its tree to the value it received in the first round.

The condition that the nodes eventually decide for a value is fulfilled trivially by the limited height of the trees and the bounded time for the execution of a round.

3.1 General Vertices

In order to find out the exact requirements for a correct protocol execution, we first look at the requirements for a general vertex to be common. We have to distinguish the cases of a correct and a faulty vertex:

Lemma 2. *A correct vertex σ is common and resolved to its stored value iff at least one of the following conditions is true:*

(a) $v(\sigma) = 0$
(b) $|\sigma| = t + 1$
(c) σ *is on an authenticated level*
(d) $v(\sigma 1) > v(\sigma 0)$ *and* $\sigma 1$ *is common and resolved to its stored value*

Proof. "⇒": by inspection of a vertex σ for which (a)–(d) do not hold. "⇐": (a), (b) and (d) are trivial. (c): A correct node will only utter consistent messages with the correct signature. Hence, all nodes store the same signed value in their trees and resolve this vertex to the stored value. □

Recursive application of Lemma 2, making use of Lemma 1 (b), yields

Corollary 3. *Let σ be a correct vertex on a non-authenticated level $l < t+1$, and let s be the next authenticated level after l or, if that does not exist, $t + 1$. Then σ is common and resolved to its stored value iff $v(\sigma 1^{s-l-1} 1) > v(\sigma 1^{s-l-1} 0)$.*

We state the requirements for faulty vertices to be common only for the special case 0^l, because this will be the only case we need.

Lemma 4. *A faulty vertex 0^l is common iff at least one of the following conditions is true:*

(a) 0^l0 and 0^l1 are common, and if l is an authenticated level, then $v(0^l1) > 0$
(b) $v(0^l) = 0$

Proof. "$\Rightarrow$": Suppose 0^l0 or 0^l1 is not common, and $v(0^l) > 0$. Then the resolved values for 0^l can differ, since the resolved values for the common children can be chosen by the faulty nodes in a way that the non-common vertex is crucial for the majority. If l is an authenticated level and $v(0^l1) = 0$, then 0^l0 could be resolved to unsigned values. Hence, 0^l would be resolved to its stored value, which is not common. If $v(0^l1) > 0$ and a correct node has stored a signed value, all other nodes will consider this signed value when resolving 0^l. "$\Leftarrow$": simple. □

Taking into account Lemma 1 (a), we get the following

Corollary 5. *0^{t+1} is common.*

The next lemma shows that the requirements for a correct vertex to be common are the same as for that vertex to be common *and* to be resolved to its stored value.

Lemma 6. *A correct common vertex σ is resolved to its stored value.*

Proof. Suppose it is resolved to a different value. Then σ has to be on a non-authenticated level $l \neq t+1$ (Lemma 2). Let s be as in Corollary 3. From Corollary 3 follows that $v(\sigma 1^{s-l-1}1) \leq v(\sigma 1^{s-l-1}0)$ and $\sigma 1^{s-l-1}0$ is common. With an argument similar to the proof of Lemma 4, the latter condition would require that $v(\sigma 1^{s-l-1}0^{t+1-|\sigma|-(s-l-1)}) > 0$ which contradicts Lemma 1 (a). □

This Lemma allows us to use Corollary 3 on Lemma 4 (a). Taking into account Lemma 1 (b) and Lemma 2, we can transcribe Lemma 4 as follows:

Corollary 7. *Let 0^l be a faulty vertex on level $l < t+1$, and let s be the next authenticated level after l or, if that does not exist, $t+1$. Then 0^l is common iff one of the following conditions is true:*

(a) $s > l+1$ and $v(0^l1^{s-l-1}1) > v(0^l1^{l-s-1}0)$ and 0^s is common.
(b) $s = l+1$ and 0^s is common.

Note that the condition $v(0^l1^{s-l-1}1) > v(0^l1^{l-s-1}0)$ in Corollary 7 (a) is equivalent to $c \geq t - 2l + s$ (from the definition of $v(\cdot)$). We will use this observation in the next section.

3.2 Requirements for a Common Root

Having stated the requirements for general vertices, we can easily deduce the requirements for a common root. We use Lemma 2 and Corollary 3 for a correct sender, and Corollary 5 and Corollary 7 for a faulty sender.

Figure 3 gives an overview of the requirements in the two cases of a correct and a faulty sender. We have sketched two binary *EIG* trees; the ">"-sign under two vertices means that the label of the left vertex must be greater than that of the right vertex. The horizontal lines denote authenticated levels.

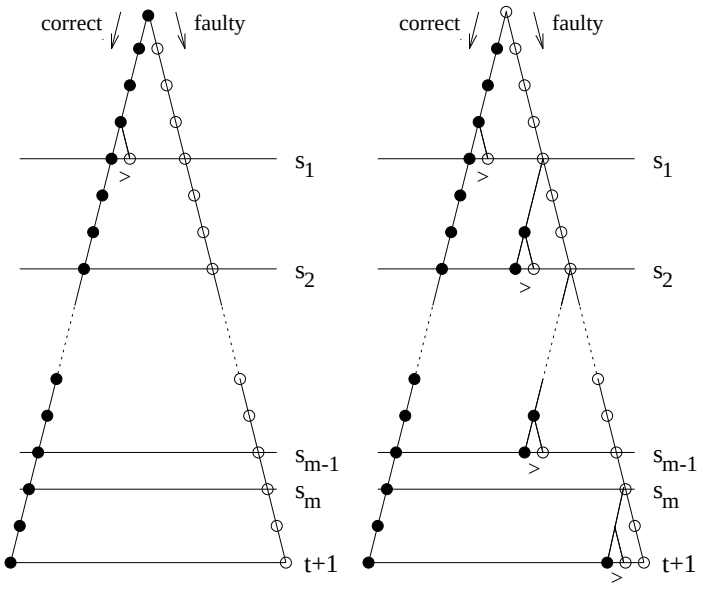

Fig. 3. Requirements for a correct protocol execution. Left: sender correct, right: sender faulty

Sender correct: Requirement RC distinguishes whether or not the root is on an authenticated level:

$$RC = \begin{cases} c \geq 1 & \text{if } s_1 = 1 \\ (c \geq t + s_1) & \text{else} \end{cases}$$

Sender faulty: Here, we present the requirements in a recursive manner. A requirement RF_i deals with the tree above and including level s_i. For the root to be common with a faulty sender, RF_0 must hold:

$$RF_0 = \begin{cases} RF_1 & \text{if } s_1 \leq 2 \\ (c \geq t + s_1 - 2) \wedge RF_1 & \text{else} \end{cases}$$

$$RF_i\ (i = 1, \ldots, m-1) = \begin{cases} RF_{i+1} & \text{if } s_{i+1} = s_i + 1 \\ (c \geq t - 2s_i + s_{i+1}) \wedge RF_{i+1} & \text{else} \end{cases}$$

$$RF_m = \begin{cases} c \geq 1 & \text{if } s_m = t \\ (c \geq 2t - 2s_m + 1) & \text{else} \end{cases}$$

Since the requirement $(c \geq t + s_1 - 2)$ from RF_0 is subsumed by $(c \geq t + s_1)$ from RC, it will not show up in the following considerations.

Our aim is to determine authenticated rounds s_i $(i = 1, \ldots, m)$ such that RC and RF_0 are fulfilled for a minimal c. We will call these s_i an *optimal distribution of authenticated rounds*. It will be convenient to consider only distributions which have no successive authenticated rounds except at the beginning of the protocol. The next theorem shows that there is an optimal distribution with this property.

Theorem 8. *There is an optimal distribution of authenticated rounds for which the following holds: If there is a succession of more than one authenticated rounds, it starts at round 1.*

Proof. We show that a distribution with a succession of authenticated rounds not starting at 1 can be optimized. From the assumption, there is a k such that $s_k > 1$, $s_{k+1} = s_k + 1$, and $s_{k-1} < s_k - 1$ (for convenience, we define s_0 to be 0). From RF_{k-1} (or RC, if $k = 1$), we require $c \geq t - 2s_{k-1} + s_k$. Now consider another distribution with the same authenticated rounds s'_i, except that $s'_k = s_k - 1$. Now RF_{k-1} becomes $c \geq t - 2s'_{k-1} + s'_k = t - 2s_{k-1} + s_k - 1$. Furthermore, RF_k changes from no requirement to $c \geq t - 2s'_k + s'_{k+1} = t - s_k + 3$. Both requirements allow for a smaller c than the original RF_{k-1} (note that $s_{k-1} \leq s_k - 2$). □

So, for a distribution with a block of b successive rounds at the beginning, we have the following inequations (note that we defined s_0 to be 0):

$$\begin{aligned} c \geq t - 2s_{i-1} + s_i &\Leftrightarrow 2s_{i-1} - s_i + c \geq t \ (i = b+1, \ldots, m) \\ c \geq 2t - 2s_m + 1 &\Leftrightarrow 2s_m + c \geq 2t + 1 \end{aligned} \tag{1}$$

Since we are interested in a minimal c, we replace the "$\geq$"s by "="s and solve the resulting system of equations, including the information that the first b rounds are authenticated (hence $s_i = i$ for $i = 1, \ldots, b$). These $m + 1$ linear equations can be written as an $(m + 1) \times (m + 2)$-matrix. Columns 1 to m represent the coefficients of s_1 to s_m, the $m + 1$st column represents the coefficient of c, and the last column represents the constants on the right hand side of the equations:

$$\begin{pmatrix}
1 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & 1 \\
0 & 1 & 0 & \cdots & \cdots & \cdots & \cdots & 0 & 2 \\
\vdots & \ddots & \ddots & \ddots & & & & \vdots & \vdots \\
0 & \cdots & 0 & 1 & 0 & \cdots & \cdots & 0 & b \\
0 & \cdots & 0 & 2 & -1 & 0 & \cdots & 0\,1 & t \\
\vdots & & & \ddots & \ddots & \ddots & \ddots & \vdots\,\vdots & \vdots \\
0 & \cdots & \cdots & \cdots & 0 & 2 & -1 & 0\,1 & t \\
0 & \cdots & \cdots & \cdots & \cdots & 0 & 2 & -1\,1 & t \\
0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & 2\,1\,1 & 2t+1
\end{pmatrix} \tag{2}$$

Eliminating the elements below the diagonal yields matrix (3):

$$\begin{pmatrix}
1 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & 1 \\
0 & 1 & 0 & \cdots & \cdots & \cdots & \cdots & 0 & 2 \\
\vdots & \ddots & \ddots & \ddots & & & & \vdots & \vdots \\
0 & \cdots & 0 & 1 & 0 & \cdots & \cdots & 0 & b \\
0 & \cdots & \cdots & 0 & -1 & 0 & \cdots & 0\;\;1 & t-2b \\
0 & \cdots & \cdots & 0 & -1 & 0 & \cdots & 0\;\;3 & 3t-4b \\
\vdots & & & & \ddots & \ddots & \ddots & \vdots\;\;\vdots & \vdots \\
0 & \cdots & \cdots & \cdots & 0 & -1 & 0 & 2^{m-b-1}-1 & (2^{m-b-1}-1)t-2^{m-b-1}b \\
0 & \cdots & \cdots & \cdots & \cdots & 0 & -1 & 2^{m-b}-1 & (2^{m-b}-1)t-2^{m-b}b \\
0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & 2^{m-b+1}-1 & 2^{m-b+1}(t-b)+1
\end{pmatrix} \tag{3}$$

Finally, we multiply rows $b+1$ to $m+1$ by -1 and eliminate the elements above the diagonal:

$$\begin{pmatrix}
1 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & 1 \\
0 & 1 & 0 & \cdots & \cdots & \cdots & \cdots & 0 & 2 \\
\vdots & \ddots & \ddots & \ddots & & & & \vdots & \vdots \\
\vdots & & 0 & 1 & 0 & \cdots & \cdots & 0 & b \\
\vdots & & & 0 & 1 & 0 & \cdots & 0 & 1\cdot\frac{t+1-2^{m-b+1}b}{2^{m-b+1}-1}+2b \\
\vdots & & & & 0 & 1 & 0 & \cdots\;0 & 3\cdot\frac{t+1-2^{m-b+1}b}{2^{m-b+1}-1}+4b \\
\vdots & & & & & \ddots & \ddots & \ddots\;\vdots & \vdots \\
0 & \cdots & \cdots & \cdots & \cdots & 0 & 1 & 0 & (2^{m-b}-1)\cdot\frac{t+1-2^{m-b+1}b}{2^{m-b+1}-1}+2^{m-b}b \\
0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & 1 & \frac{2^{m-b+1}(t-b)+1}{2^{m-b+1}-1}
\end{pmatrix} \tag{4}$$

From matrix (4), we can deduce the following (real-valued) solutions for the s_i and c:

$$s_i = \begin{cases} i & , \; 1 \le i \le b \text{ (only for } b > 0) \\ 2^{i-b}b + (2^{i-b}-1)\cdot\frac{t+1-2^{m-b+1}b}{2^{m-b+1}-1} & , \; b < i \le m \end{cases}$$

$$c \geq \frac{2^{m-b+1}(t-b)+1}{2^{m-b+1}-1} \tag{5}$$

Replacing c by $n-t$ and solving (5) for m yields (note that (5) can only be fulfilled for $n+b-2t>0$):

$$m \geq \log_2(n+1-t) - log_2(n+b-2t) + b - 1. \tag{6}$$

Since we are interested in as few authenticated rounds as possible, we try to minimize m. To find a b such that m is minimal, we solve $m'(b)=0$ which leads to:

$$1 - \frac{1}{(n+b-2t)\cdot \ln(2)} = 0.$$

Since $m''(b) = \frac{1}{(n+b-2t)^2 \cdot \ln(2)} > 0$ and $b \geq 0$, we have the minimal m at:

$$b = \max(0, 2t - n + \underbrace{\frac{1}{ln(2)}}_{\approx 1.44}).$$

Together with (6), this yields:

$$m \geq \begin{cases} \log_2(\frac{n+1-t}{n-2t}) - 1 & \text{if } 2t \leq n - \frac{1}{ln(2)} \\ \log_2(n+1-t) + 2t - n + \underbrace{\frac{1}{\ln(2)} + \log_2(\ln(2)) - 1}_{\approx -0.08} & \text{else} \end{cases}$$

To summarize these results and to show how they are cast into integer values, we state the following main theorem:

Theorem 9. *Let n be a number of nodes with at most t of them being faulty. Then there is a protocol solving Byzantine Agreement with m authenticated rounds, where*

$$m = \begin{cases} \lceil \log_2(\frac{n+1-t}{n-2t}) - 1 \rceil & \text{if } 2t \leq n-2 \\ \lceil \log_2(n+1-t) + 2t - n \rceil & \text{else} \end{cases}$$

With $b = \max(0, 2t-n+2)$, the authenticated rounds s_i $(i = 1, \ldots, m)$ are:

$$s_i = \begin{cases} i & \text{for } 1 \leq i \leq b \text{ (only if } b > 0) \\ \lceil 2^{i-b}b + (2^{i-b}-1) \cdot \frac{t+1-2^{m-b+1}b}{2^{m-b+1}-1} \rceil & \text{for } b < i \leq m \end{cases}$$

Proof. We have to show that the constraints remain satisfied if we use rounded m and s_i. To round up m is valid, since that lowers the right hand side of (5) and hence allows for even lower c. That rounding up the s_i is valid follows from c and t being integer values: The constraints in (1) are of the form $c \geq t - 2s_{i-1} + s_i$. It follows that $c \geq t - 2\lceil s_{i-1} \rceil + s_i$. Since all summands except s_i are integers, we can safely deduce that $c \geq t - 2\lceil s_{i-1} \rceil + \lceil s_i \rceil$.

Finally, we show that the selected b is an optimal integer: Since there is only one optimal real-valued b, the optimal integer-valued b must be one of $\max(0, 2t-n+1.5 \pm 0.5)$. Since both bs yield the same values for m, we can choose the higher one. □

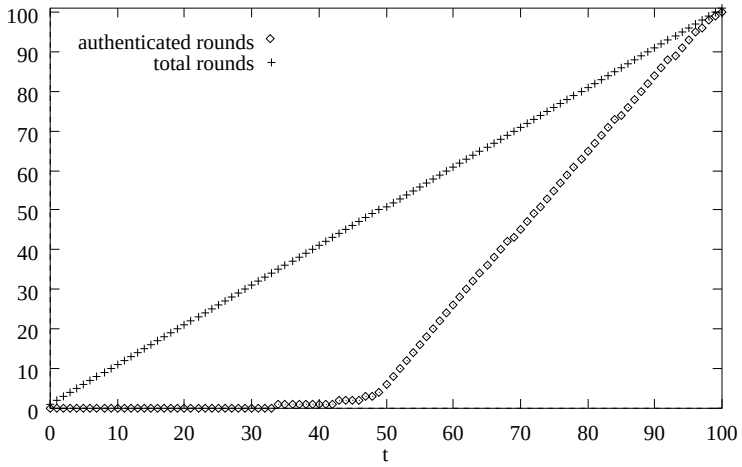

Fig. 4. Minimal authenticated rounds for a given t with $n = 100$.

Example. Figure 4 depicts the required number of authenticated rounds with respect to the number of tolerated faults for $n = 100$. As expected, a maximum of $(n-1)/3 = 33$ faulty nodes can be tolerated without authentication. To tolerate $n/2 = 50$ faulty nodes, only $\lceil \log_2(n/2+1) \rceil = \lceil \log_2(51) \rceil = 6$ authenticated rounds are necessary, namely rounds $1, 2, 4, 7, 14$, and 26.

4 Conclusion

We have investigated the fault-tolerance properties of Byzantine Agreement protocols when messages are signed in certain rounds only. We have shown how to determine the authenticated rounds, depending on the number of faulty nodes to be tolerated. One implication of the results is that $n/2$ faulty nodes can be tolerated with as few as $\lceil \log_2(n/2+1) \rceil$ authenticated rounds, while each additional faulty node requires about two more authenticated rounds.

Further work in this area is necessary. We have not yet shown that the results in this paper are optimal, since other, maybe better-suited, resolve functions in the *EIG* protocol are possible. Furthermore, the *EIG* protocol is far from efficient with respect to the number of messages, and does not employ early-stopping mechanisms in case less than t of the nodes actually behave faulty.

References

[BNDDS87] 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, 1987.

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

[FL82] Michael J. Fischer and Nancy A. Lynch. A lower bound for the time to assure interactive consistency. *Information Processing Letters*, 14(4):183–186, 1982.

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

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