Reducing the Communication Complexity of Agreement Protocols By Applying A New Signature Scheme called SIGSEAM

Omar Bousbiba
Dependability of Computing Systems
University of Duisburg-Essen
Essen, Germany
bousbiba@dc.uni-due.de

***Abstract*—Distributed computing systems need agreement protocols when global consistency must be achieved in a fault-tolerant way. However, solving the Byzantine agreement problem in an efficient way in terms of communication complexity is still a challenging task. In synchronous systems with stringent time requirements not only the fault tolerance, but also the limitation of the communication complexity are crucial for practical usability. Many agreement protocols use digital signatures. This paper presents a novel signature generation technique to merge several signatures into a single one. This advantage opens a design space for agreement protocols with significantly reduced message overhead. Moreover, the new signature technique can also be applied to existing agreement and/or consensus protocols (Turquois and ESSEN, for example) without affecting the fault tolerance properties of the protocol.**

Keywords— Malicious Byzantine Faults; Agreement protocols; Digital Signatures for Fault Tolerance.

I. INTRODUCTION

Distributed systems are becoming more and more important in our electronic society. In case of safety relevance, it is important to make these systems resilient against faults. Fault tolerance techniques can be applied to increase various dependability properties. Many real-time applications require fail-operational behaviour. Take a fail-safe brake-by-wire system as an example. It has to provide its functionality all the time. In the presence of a fault, the four-wheels braking is reduced to diagonal-wheel braking. Consequently, a decision has to be taken which pair of wheels has to be passivated (in a non-blocking way, of course).

The agreement problem is recognized as a fundamental element in fault-tolerant distributed computing (i.e., safe brake, collision avoidance, semiautomated vehicles, etc.). The problem has been known for decades as Byzantine agreement (BA) [1][2]. In order to solve it, two conditions have to be satisfied, known as interactive consistency (IC):

IC1: All fault-free nodes obtain exactly the same view

IC2: The information provided by a fault-free node is part of this view.

Due to its paramount importance, the problem has attracted a great deal of attention in the past. It has been investigated extensively and many solutions have been proposed. Many of the approaches [3][4][5] focused on reducing the communication complexity in terms of the number of messages, the number of nodes (related to the number of faults to be tolerated), and required storage.

Signature techniques contribute a lot to a reduction in communication complexity, because they protect the origin of the message against undetectable corruption when the message is forwarded from node to node [2].

Typical sequences of actions during the execution of an agreement protocol are the following ones:

1. Send a signed message to one/more neighboring node(s)
2. Forward a message from node to node(s), where each forwarding node cosigns the message
3. Collect incoming messages (which can be numerous) including signature checks
4. a) Take a local decision on the message to be sent in the next round, b) termination with some value or a consistency vector [3][4].

The steps 1 to 4 may be repeated several times, depending on the particular protocol.

Typically, the following situation occurs frequently: In some phases, a node X receives different messages $M_1,\ldots,M_k$ all of which it has to forward to a neighbor node Y. If all the messages $M_1,\ldots,M_k$ have been signed by different nodes $N_1,\ldots,N_k$ and all nodes contain identical payload contents A (see Figure 1), then node X cannot summarize the messages and send only one message with payload contents A to node Y, because the signatures would be lost then. Instead X has to forward the k messages separately (in some protocols it is sufficient to filter out a subset of the messages).

Consequently, a signature mechanism which allows messages to be merged has the potential to greatly reduce the communication overhead of an agreement protocol. Figure 1 illustrates an example of the idea behind signature merging.

 ISBN: 978-1-61208-429-9

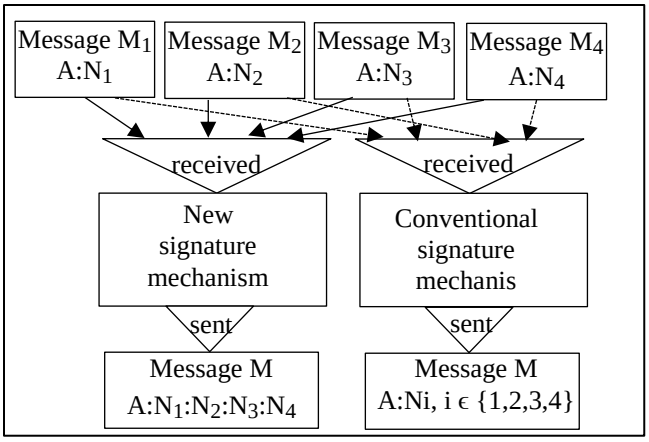


Figure 1. (left) new (right) conventional - signature mechanism.

A. Contribution and Outline

The goal of this work is the provision of a novel signature mechanism, which opens an extended design space for agreement protocols with lower communication complexity in terms of message transmissions. By signature merging, the number of messages, and thus the overall transmitted information can be reduced. The new method does not use cryptographically strong signatures. Instead, the signatures are designed to withstand faults, even with Byzantine behaviour, but not intelligent attacks of humans. Besides the (very short) computation time for signature merging, the protocol does not need extra time for reaching agreement.

The rest of the paper is organized as follows: Section II characterizes the agreement protocols relevant for this paper. The new signature method is presented in Section III and its application to agreement protocols in Section IV. The improvement is shown in Section V by providing a quantification of the overhead. A summary and an outline of ongoing work are given in Section VI.

II. CONSIDERED AGREEMENT PROTOCOLS

A. Protocols

Since the time when the agreement problem was introduced by Lamport et. al. [1], many solutions have been proposed. Most of the work is focused on reducing the number of messages, the required number of nodes per tolerated fault and the storage consumption.

It has turned out that signatured protocols need significantly less messages. However, without signature merging there is a limitation to further reduction. In this paper, the merging approach is applied to two protocols: Turquois [3] and ESSEN [5]. For each of these protocols a variant is derived that takes benefit of signature merging.

Turquois is a protocol which solves the consensus problem in asynchronous systems composed of **n** ad hoc nodes where a subset **f** (with $f < \frac{n}{3}$) of them can fail in an arbitrary manner. It is the first work which addresses the problem of reaching consensus in the presence of omission faults. However, Turquois solves the problem at the expense of a relatively high communication and storage overhead. The high number of message transmissions is caused by the message validation process. In the worst case, a node has to transmit more than $\frac{n+f}{2}$ messages received from previous round(s). A signature technique has a great impact on the message and storage overhead as will be shown later in this paper.

ESSEN is a protocol that solves the Byzantine agreement problem even in the presence of "malicious cooperation" faults. This means two faulty nodes may "secretly" exchange their information, such as keys and signed messages. In ESSEN, the communication complexity is very low for up to four arbitrary faults. The protocol requires a fully synchronous system (clock synchronization is presupposed). The message storage consumption is the space of only three messages. The required number of nodes grows quadratically with the number of tolerated faults. As with Turquois, the protocol uses signatures without merging functionality. The benefit of adding a signature scheme with merging capability will be shown later in this paper.

B. Signatures

For the purpose of fault tolerance, cryptographically strong signatures are not needed, because the signatures serve as countermeasures against "stupid faults" rather than "intelligent attacks". Consequently, signatures with relatively low computation time can be used (as reported in [7]). Signature techniques greatly improve the communication complexity of agreement/consensus protocols. However, when using an existing signature technique [7][8] a receiver has only two options to deal with after a message has been received. Either each received signature is stored separately, as is done in Turquois, or some kind of filter mechanism is applied (e.g., only the message with the highest number of signatures is stored). In both cases the overhead for both message storage and communication may become high.

By the proposed signature merging technique the receiver(s) get the opportunity to combine the messages into a single one without affecting the information about the signature source. This means, the new message will still contain the information of all signature sources (as shown in Figures 1 and Section III).

III. NEW SIGNATURE SCHEME SIGSEAM

The proposed signature scheme is intended to withstand arbitrary technical faults rather than intelligent attacks. For the purpose of fault tolerance simple signature generation methods are sufficient [6][7][8]). The signature technique presented in this paper is based on a multiplication scheme similar to [7]. It achieves almost the same effectiveness as

 ISBN: 978-1-61208-429-9

[7]. A thorough investigation on the new signature merging mechanism is still work in progress. Next, the algorithms for signature generation and checking are presented in detail.

All calculations are done modulo **m**, where **m** is set to a power of two (**m** = 2^x with x ϵ $\mathbb{N}$). Typical values may be m = 2^{16} or m = 2^{32}. The generation of private and public signature keys is done as follows: Each node chooses two arbitrary natural numbers **a, b** from $\mathbf{m_{odd}}$ = [m/16, m/2] $\cap$ $\mathbb{N}_{odd}$. The product **c** = **a** ▪ **b** is calculated. The value of parameter **a** is used as private key. The pair (**b**, **c**) is taken as public key, which is publically distributed to all nodes to be used for signature checking.

Signature generation: The signature of the original sender of a message is calculated over the payload data **d** and the sequence number **n** (e.g., the sequence number is changed from round to round) only. The following signature function σ_0 and a usual CRC function are used by the first signing node (e.g., source node, indexed with zero):

$$\sigma_0(\mathrm{n},\mathrm{d}) \coloneqq \mathrm{CRC}(\mathrm{n},\mathrm{d}) \cdot \mathrm{a_i}. \tag{1}$$

Cosignature generation of a forwarding node is done as follows: The cosignature value is calculated over the signature value σ_j with j $\geq$ 0 by applying the following cosignature function σ_i. The index represents the number of signing and/or cosigning nodes:

$$\sigma_i \coloneqq \begin{cases} \sigma_j + \mathrm{CRC}(\mathrm{n},\mathrm{d}) \cdot (\mathrm{a_i}+1), & \text{if j is even} \\ \sigma_j + \mathrm{CRC}(\mathrm{n},\mathrm{d}) \cdot (\mathrm{a_i}-1), & \text{otherwise} \end{cases}, \tag{2}$$

where j < i. Depending on the number of signatures in σ_j the secret key a_i of the cosigning node is used as either (a_i + 1) or (a_i – 1) depending or whether or not the number of already added (co-) signatures is even.

Compared to usual (co-) signature schemes there is an important point: In the proposed merging signature scheme the new cosignature σ_i replaces the existing (co-) signature in a message to be forwarded. However, the indices of all signing nodes are kept in the message. Thus, there is a list "Who has signed?" in each message.

The signature value signed by j nodes can be expressed by the following sum function:

$$s = \mathrm{CRC}(\mathrm{n},\mathrm{d}) \cdot \left((\mathrm{j}+1) \bmod 2 + \textstyle\sum_{\mathrm{i}=0}^{\mathrm{j}} \mathrm{a_i}\right). \tag{3}$$

A receiver uses the following signature check function τ(n, d, s) after reception of a message with number n, payload data d and signature s. The check is passed if the following equation is correct:

$$s \cdot \textstyle\prod_{i=0}^{j} b_i = \mathrm{CRC}(\mathrm{n},\mathrm{d}) \cdot \left(\mathrm{e} \cdot \prod_{i=0}^{j} b_i + \sum_{i}^{j} \left[c_i \cdot \prod_{k=0,k\neq i}^{j} b_k \right]\right). \tag{4}$$

If **s** has been signed by an odd number of nodes, then parameter e is set to zero. Otherwise parameter e is set to one. In case of an odd number of nodes having (co-) signed the message we obtain:

$$s \cdot \textstyle\prod_{i=0}^{j} b_i = \mathrm{CRC}(\mathrm{n},\mathrm{d}) \cdot \sum_{i=0}^{j}\left(\mathrm{c}_i \cdot \prod_{k=0,k\neq i}^{j} b_k\right) \tag{5}$$

$$s \cdot \prod_{i=0}^{j} b_i = \mathrm{CRC}(\mathrm{n},\mathrm{d}) \cdot \left((\mathrm{j}+1)\bmod 2 + \sum_{\mathrm{i}=0}^{\mathrm{j}} \mathrm{a_i}\right) \cdot \prod_{i=0}^{j} b_i \Leftrightarrow$$

$$s \cdot \prod_{i=0}^{j} b_i = \mathrm{CRC}(\mathrm{n},\mathrm{d}) \cdot \sum_{\mathrm{i}=0}^{\mathrm{j}} \mathrm{a_i} \cdot \prod_{i}^{j} b_i \Leftrightarrow$$

$$s \cdot \prod_{i=0}^{j} b_i = \mathrm{CRC}(\mathrm{n},\mathrm{d}) \cdot \sum_{i=0}^{j} \mathrm{a}_i \cdot \mathrm{b}_i \cdot \prod_{k=0,k\neq i}^{j} b_k \Leftrightarrow$$

$$s \cdot \prod_{i=0}^{j} b_i = \mathrm{CRC}(\mathrm{n},\mathrm{d}) \cdot \sum_{i=0}^{j}\left(\mathrm{c}_i \cdot \prod_{k=0,k\neq i}^{j} b_k\right) \quad \text{Q. E. D.}$$

The implications from right to left are obvious. The implication from left to right based on the same conclusion, as shown in [8]. The proof is done by contradiction: Due to the fact that all calculations are done modulo m (**m** = 2^x with x ϵ $\mathbb{N}$) for b ϵ m_{odd} the product $s \cdot \prod_{i=0}^{j} b_i$ returns a unique value in modulo m. However, parameter b is odd and the only prime factor of m is 2. Consequently, 2 must be a prime factor of value b ϵ m_{odd} (contradiction) Q.E.D.

IV. MODIFIED AGREEMENT PROTOCOL

A detailed explanation of the two protocols ESSEN and Turquois can be found here [3][5]. In the following, only the parts of the algorithm which have been modified are discussed in detail. The modified protocol variants are called SEAM and Turquois*, respectively.

A) SEAM

Storing of received messages: A data message is stored in the secondary buffer, when (in addition to the four conditions given in [5]) also the following two conditions are satisfied:

1. The node is member of group ExtG (see [5])
2. The node has not transmitted a message yet.

Otherwise, if all six conditions are not satisfied, the data message is rejected (for more details see [5]).

Merging of signatures: The messages in the primary and the secondary buffer are merged, iff both messages contain at least 2f – 2 (parameter f indicates the number of tolerated faults) different signature sources. The content of the

secondary buffer is deleted after transmission (regardless of whether or not the message has been merged). Moreover, in contrast to [5], only the primary and default buffer are used for the final decision. All other parts of the algorithm remain unchanged.

B) Turquois*

Merging of signatures: In each round, all messages with identical content are merged (instead of stored separately). This means not more than two different messages are stored within a round. All other parts of Turquois remain unchanged. Both variants SEAM and Turquois* have been simulated for up to 30 nodes and up to 10^9 rounds. The number of simultaneously faulty nodes has been limited to 6 in case of SEAM and 9 in case of Turquois (for more details see [3][5]). In all simulation runs the interactive consistency was fully preserved.

V. Complexity of the investigated protocols by Using SigSeam

In this section, the communication complexity in terms of redundant nodes as well as message transmission overhead is quantified by simulation. The modified protocols SEAM and Turquois* are compared with their original versions ESSEN and Turquois, respectively. The outcomes are shown in Figure 2. Summarizing the results it can be said that the new signature technique greatly improves the communication complexity of both protocols. In case of ESSEN the number of redundant nodes as well as the number of required message transmissions has been reduced from $1 + \frac{f^2+f}{2} + \left\lfloor \frac{(f-1)}{2} \right\rfloor$ down to $\frac{3f+2\left\lceil \frac{f-1}{2} \right\rceil + \left\lceil \frac{f^2}{2} \right\rceil}{2}$. In case of Turquois, the high number of $(3f+1)\frac{(n+f+2)}{2}$ message transmission (worst case) has been reduced to a constant of 3 messages per node, whereas the number of required nodes remains unchanged. This means 9f + 3 messages in all.

VI. Conclusion and Future work

The simulation results have clearly shown that the proposed signature technique with merging functionality significantly improves the efficiency of agreement protocols and does not affect the time taken to reach agreement.

The work on signature merging is still in progress. The coverage of special fault cases affecting the signatures themselves must be evaluated in detail. Besides bursts, bit flips, wrong data, also signature-related faults like copy-and-paste of signatures between messages, etc., have to be assessed with respect to the achieved coverage. Moreover, these results will be compared with 16-bit or 32-bit "light" versions of existing cryptographic signature techniques.

Acknowledgment

The author gratefully acknowledges the helpful discussions with Prof. Klaus Echtle who participated in the development of the concept of signature merging.

References

[1] M. Pease, R. Shostak, and L. Lamport, "Reaching agreement in the presence of faults", JACM, vol. 27, Apr. 1980, pp. 228–234, doi: 10.1145/322186.322188.

[2] L. Lamport, R. Shostak, and M. Pease, "The byzantine generalsproblem", ACM and TOPLAS, vol. 4, July 1982, pp. 382–401, doi:10.1145/357172.357176.

[3] H. Moniz, N. F. Neves, and M. Correia, "Turquois: Byzantine consensus in wireless ad hoc networks", IEEE, DSN, Chicago, IL, pp. 537-546, June 2010, pp. 537–546, doi: 10.1109/DSN.2010.5544268.

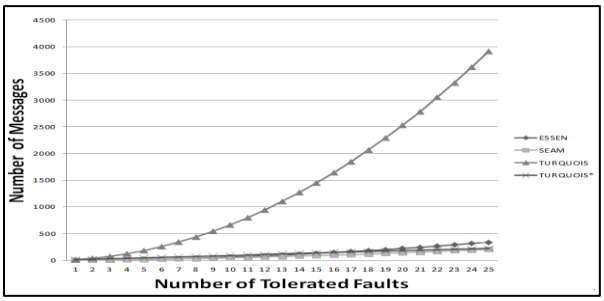
[4] M. Jochim and T. M. Forest, "An Efficient Implementation of the SM Agreement Protocol for a Time Triggered Communication Systems", SAE International Journal of Passenger Cars - Electronic and Electrical Systems, vol. 3, 2010, pp. 106-116, doi:10.4271/2010-01-2320.

[5] O. Bousbiba, "ESSEN - An Efficient Single Round Signature Protected Message Exchange Agreement Protocol for Wireless Distributed Networks", ACRS 28th, Workshop Proceedings, March 24 - 27, 2015, Porto, Portugal, ISEP, Berlin: VDE Verl., ISBN: 978-3-8007-3657-7.

[6] K. Echtle and T. Kimmeskamp, "Fault-Tolerant and Fail-Safe Control Systems Using Remote Redundancy", ARCS 22th, Workshop Proceedings, March 11, 2009, Delft, The Netherlands, Berlin: VDE Verl., ISBN: 978-3-8007-3133-6.

[7] L. Martin, "Relative signatures for fault tolerance and their implementation", Dependable Computing – EDCC-1, Lecture Notes in Computer Science, vol. 852, Oct. 1994, pp. 561–580, doi:10.1007/3-540-58426-9_158.

[8] K. Echtle, "Avoiding Malicious Byzantine Faults by a New Signature Generation Technique", Depenable Computing – EDCC3, Lecture Notes in Computer Science, vol. 1667, Sept. 1999, pp. 106-123, doi:10.1007/3-540-48254-7_9.



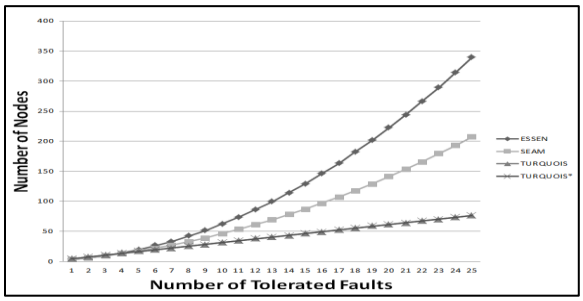


Figure 2. The idea of signature merging. Communication complexity: (a) message transmission overhead (b) required number of redundant nodes.