(To appear in Proceedings of IFIP DCCA-5, Illinois, U.S.A., September 1995)

Interactive Consistency Algorithms Based on Authentication and Error-correcting Codes

André Postma, Thijs Krol
University of Twente, Department of Computer Science
P.O. Box 217, NL7500 AE Enschede, the Netherlands
email: postma@cs.utwente.nl, krol@cs.utwente.nl

Abstract

In this paper, a class of interactive consistency algorithms is described, based on authentication and error-correcting codes. These algorithms require considerably less data communication than existing algorithms, whereas the required number of modules and communication rounds meet the minimum bounds. The algorithms based on authentication and error-correcting codes are defined and proved on basis of a class of algorithms called the Authenticated Dispersed Joined Communication Algorithms.

1. Introduction

Fault-tolerant systems will always be connected to basically unreliable input devices. An external faulty module producing broadcast errors (i.e. sending conflicting information to different modules of the fault-tolerant system) may cause a system breakdown in the fault-tolerant system, even if this system does not contain more faulty modules than it is designed to tolerate. This problem is called the Input Problem [1].

The Input Problem can be conquered by so-called Interactive Consistency Algorithms [2]. Such an algorithm runs on a system consisting of N modules of which one is the source, and in which at most T modules behave maliciously. Regardless of which modules are faulty and which data was sent by the source, the algorithm fulfils the so-called interactive consistency conditions [2,3]:

- *The well-functioning modules agree among each other on the data they think they have received from the source.*
- *If the source is well-functioning, the above-mentioned agreement should equal the data actually sent by the source.*

Interactive consistency algorithms can be characterized by the total number of modules, N, in the system, the maximal number of maliciously behaving modules, T, that can be tolerated, the number of rounds, K, of information exchange, the connectivity, C, of the graph representing the communication paths between the modules in the system, and the total amount of data that is exchanged between the modules. A further classification can be made into synchronous/asynchronous algorithms, and algorithms based on authenticated/non-authenticated messages.

In this paper, we will describe a class of synchronous, deterministic algorithms based on authenticated messages.

In *synchronous* algorithms, modules relay messages within a commonly known limited time span, which allows receivers to detect modules which refused to send data.

In this paper, we focus on algorithms based on *authenticated messages*, since they require considerably less data communication and modules than algorithms with non-authenticated messages [2].

In case of authenticated messages, the source module appends an unforgeable signature to the message, by means of which any receiving module can verify the authenticity of the source module. Furthermore, any mutilation of the contents of such a message can be detected. Unforgeable signatures can be obtained with cryptographic techniques [4]. We can use either symmetric or asymmetric cryptosystems for this purpose. For reasons of simplicity and efficiency we will use an asymmetric cryptosystem to create authenticated messages.

In an *asymmetric cryptosystem* (also called public-key cryptosystem [4,5]), for every module s in the system, two functions E_s and D_s are defined, which are each other's inverses. Module s keeps E_s secret, whereas D_s is made public to all modules.

A source module s may append an unforgeable signature to a message m by applying encryption function E_s (only known to s) on a *message digest* [6] of m (i.e. some check information with which can be checked whether or not message m has been mutilated). Any receiving module r can verify that the message was sent by s, and can detect any mutilations of m by decrypting the message digest of m with decryption function D_s (known to r).

Synchronous, deterministic algorithms for reaching interactive consistency based on authenticated messages can be constructed if and only if the total number of modules N

$\geq T + 1$, the number of communication rounds $K \geq T + 1$, and the graph connectivity $C \geq T + 1$. The problem is vacuous if there are less than T+2 modules [3].

A first solution to the problem was given in [2] and described as the SM algorithm in [3]. This algorithm, which we will refer to as the Lamport-algorithm, is inefficient with respect to the required amount of data communication. Therefore, in the literature a number of more efficient algorithms have been proposed. An extensive overview is given in [7]. In [8], a lower bound was given for the number of messages required, together with algorithms for 1-bit messages that meet this lower bound. For messages greater than 1 bit, a very efficient early-stopping algorithm is given in Theorem 6 in [9]. We will refer to this algorithm as the Dolev-algorithm.

In this paper, we will describe a class of synchronous deterministic algorithms based on authentication which solve the interactive consistency problem for messages of variable length for $N \geq T$+2 and $K \geq T$+1. The algorithms have similarities with the Lamport-algorithm. However, the required data communication is reduced considerably by reducing the number of directions in which data is sent, and, by replacing the broadcast functions in the Lamport-algorithm by encoder functions of error-correcting codes and replacing the voting functions in the decision-making process by the corresponding decoder functions. Our algorithms can be seen as a recursive application of the information dispersal method for transmission of data described in [10]. Although our algorithms are not early-stopping, in a number of relevant cases, the required amount of data communication is also considerably less than that needed in early-stopping algorithms, like the Dolev-algorithm [9].

This new class of interactive consistency algorithms based on authentication and error-correcting codes will be defined and proved on basis of a class of algorithms which we call the Authenticated Dispersed Joined Communication (ADJC) algorithms. These algorithms satisfy more liberal properties than those which are required for the interactive consistency algorithms. The ADJC-algorithms have strong similarities with the Dispersed Joined Communication (DJC) algorithms, which are designed for non-authenticated messages. The DJC-algorithms are described in [11].

2. The Authenticated Dispersed Joined Communication algorithms

2.1. Introduction

In this section, we will describe the Authenticated Dispersed Joined Communication (ADJC) algorithms. These algorithms consist of a broadcast process and a decision-making process. In the broadcast process, in a number of communication rounds, a message is transmitted from a source module to a set of destination modules in the presence of a number of maliciously behaving modules. In the decision-making process, the destination modules calculate a decision about what the source has sent on basis of the information they received during the communication rounds.

The ADJC-algorithms are defined such that they satisfy the following behavioural properties:

- *If the source and destination module are both well-functioning, then the decision calculated during the decision-making process in the destination module equals the original message in the source*
- *For an algorithm based on K rounds of information exchange which aims at communicating a message from a source module to a number of destinations, it holds that if the result calculated in two well-functioning destinations is different, then the number of maliciously behaving modules in the system is at least K.*

The interactive consistency conditions are implied by these properties, provided that the number of maliciously behaving modules in the system is less than the number of communication rounds.

In order to be able to tolerate a number of maliciously behaving modules, the communication between the source and the destinations is *dispersed*, i.e. the message which needs to be transmitted is encoded into symbols of an error-correcting code, and these symbols are sent via different paths from the source to the destinations. Furthermore, the communication for different destinations is *joined* as much as possible, i.e. all paths from the source to these destinations are shared as much as is compatible with the additional requirement that a message is never relayed to a module that it has already passed.

2.2. Definitions

The ADJC-algorithms are defined on a set **Ns** of fully interconnected modules. A particular ADJC-algorithm aims at sending a particular message from a particular source module a to all modules in the set **Ns**, by means of K communication rounds. In general, many ADJC-algorithms with the same properties will exist.

Therefore, we define classes

$$\mathcal{A}(T,K,a,\mathbf{Ns}) \qquad (1)$$

of ADJC-algorithms in which:

T is the maximum number of maliciously behaving modules which is tolerated

K is the number of communication rounds

a is the source module of the algorithm

Ns is the set of modules in the system.

Obviously, these classes $\mathcal{A}(T,K,a,\mathbf{Ns})$ of ADJC-algorithms are only defined if

$$K \geq 1 \text{ and } a \in \mathbf{Ns} \qquad (2a)$$

In order to exclude some pathological classes, we additionally require

$$|\mathbf{Ns}| \geq K + 1 \qquad (2b)$$

An algorithm in the class $\mathcal{A}(T,K,a,\mathbf{Ns})$ forwards the original message in the source module in K communication rounds to all modules in **Ns**. The original message in the source a is denoted by

$$m(\underline{\mathbf{s}},a) \text{ or by } m(a) \qquad (3)$$

The prefix $\underline{\mathbf{s}}$ to the source module identifier a is used only when we need to distinguish between different messages in the same module a. In that case, it denotes the path along which the message travelled to module a.

If a message $m(\underline{\mathbf{s}},a)$ (or $m(a)$) is sent to the modules in **Ns** by means of an algorithm from the class $\mathcal{A}(T,K,a,\mathbf{Ns})$, then the results calculated in the modules d (with $d \in \mathbf{Ns}$) are denoted by

$$dec_K((\underline{\mathbf{s}},a),d) \text{ (or by } dec_K((a),d)) \qquad (4)$$

We will define $\mathbf{B}(a)$ as the next-set of module a, i.e. the set of modules to which the message $m(a)$ from module a is sent in the first round of the algorithm $\mathcal{A}(T,K,a,\mathbf{Ns})$.

2.2.1. Definition of $\mathcal{K}_{(a)}$ and $\mathcal{K}_{(a)}^{(-1)}$. In the ADJC-algorithms, for authentication purposes, for every module $a \in \mathbf{Ns}$, two functions $\mathcal{K}_{(a)}$ and $\mathcal{K}_{(a)}^{(-1)}$ (which are each other's inverses) are defined in such a way, that it is computationally infeasible to calculate $\mathcal{K}_{(a)}$ from $\mathcal{K}_{(a)}^{(-1)}$. Module a reveals $\mathcal{K}_{(a)}^{(-1)}$, but keeps $\mathcal{K}_{(a)}$ secret. Notice that module a is now able to authenticate any message by appending a message digest to it, encrypted with $\mathcal{K}_{(a)}$.

2.2.2. The form of correct messages. Messages transmitted in the ADJC-algorithms consist of two parts: an information part and a message digest, which is a function of the information part. For authentication purposes, the message digest is encrypted with the sender's secret key.

A message $m(\underline{\mathbf{s}},a,b)$ which travelled along path $\underline{\mathbf{s}}$ to a ($a \in \mathbf{Ns}$), and is sent from a to b ($b \in \mathbf{B}(a)$) has the form

$$< i_{(\underline{\mathbf{s}},a,b)},\ \mathcal{K}_{(a)}(md(i_{(\underline{\mathbf{s}},a,b)})) >,$$

in which $i_{(\underline{\mathbf{s}},a,b)}$ is the information part of the message sent from a to b and $\mathcal{K}_{(a)}(md(i_{(\underline{\mathbf{s}},a,b)}))$ is the message digest $md(i_{(\underline{\mathbf{s}},a,b)})$, which is a function of $i_{(\underline{\mathbf{s}},a,b)}$ and which is encrypted with a's secret key $\mathcal{K}_{(a)}$.

For an empty string $\underline{\mathbf{s}}$, we define $m(\underline{\mathbf{s}},a)$ as the original message in a.

2.2.3. Definition of $\mathcal{Z}_{(a)}$ and $\mathcal{Z}_{(a)}^{(-1)}$. Message $m(\underline{\mathbf{s}},a)$ which is received in module a along path $\underline{\mathbf{s}}$ is related to the messages $m(\underline{\mathbf{s}},a,b)$ which a sends to all $b \in \mathbf{B}(a)$ in such a way, that $m(\underline{\mathbf{s}},a)$ is encoded into $|\mathbf{B}(a)|$ symbols of an erasure-correcting code $\mathcal{Z}_{(a)}$ such that:

$$i_{(\underline{\mathbf{s}},a,b)} = \mathcal{Z}_{(a)}(b)(m(\underline{\mathbf{s}},a)) \text{ for all } b \in \mathbf{B}(a)$$

Here, $\mathcal{Z}_{(a)}(b)$ is the partial encoder function of $\mathcal{Z}_{(a)}$ which delivers the symbol that has to be sent to b.

Let $\mathcal{Z}_{(a)}$ be the encoder function of a T-erasure-correcting code (and $|\mathbf{B}(a)| > T$). Then, for a set $S_{(\underline{\mathbf{s}},a)}$ of $|\mathbf{B}(a)|$ information parts $i_{(\underline{\mathbf{s}},a,b)}$ (for $b \in \mathbf{B}(a)$) of which for at most T information parts holds $i_{(\underline{\mathbf{s}},a,b)}$ = NIL, and of which all other information parts satisfy $i_{(\underline{\mathbf{s}},a,b)} = \mathcal{Z}_{(a)}(b)(m(\underline{\mathbf{s}},a))$, the decoder function $\mathcal{Z}_{(a)}^{(-1)}$ is defined by:

$$\mathcal{Z}_{(a)}^{(-1)}(S_{(\underline{\mathbf{s}},a)}) = m(\underline{\mathbf{s}},a)$$

For other sets S, we define $\mathcal{Z}_{(a)}^{(-1)}(S)$ = NIL.

2.2.4. Definition of the functions $\mathcal{Y}_{(a)}$ and $\mathcal{Y}_{(a)}^{(-1)}$. The authentication function $\mathcal{Y}_{(a)}$ appends an encrypted message digest to an information part. Thus, for all information parts i:

$$\mathcal{Y}_{(a)}(i) = <i, \mathcal{K}_{(a)}(md(i))>$$

i.e Applying $\mathcal{Y}_{(a)}$ on i results in applying function md on i, by which message digest $md(i)$ is obtained, which is encrypted with a's secret key $\mathcal{K}_{(a)}$ and appended to i.

We will also denote an encrypted message digest $\mathcal{K}_{(a)}(md(i))$ as $c_{(a)}(i)$.

Function $\mathcal{Y}_{(a)}^{(-1)}$ is defined by:

$$\mathcal{Y}_{(a)}^{(-1)}(<i,c_{(a)}(i)>) = i, \text{ iff } \mathcal{K}_{(a)}(md(i)) = c_{(a)}(i)$$

$$\mathcal{Y}_{(a)}^{(-1)}(<i,c_{(a)}(i)>) = \text{NIL otherwise}$$

i.e. $\mathcal{Y}_{(a)}^{(-1)}$) verifies if $<i,c_{(a)}(i)>$ is mutilated by checking if the encrypted message digest $c_{(a)}(i)$ is compatible with i. If the result of applying functions md and $\mathcal{K}_{(a)}$ to i is equal to $c_{(a)}(i)$ then the message is not mutilated and i is returned, else $\mathcal{Y}_{(a)}^{(-1)}$ returns NIL.

2.3. Construction of ADJC-algorithms

Assume a system consists of a set **Ns** of fully interconnected modules. At most $Z_{(T,\mathbf{Ns})}$ of these modules are allowed to behave maliciously. For any $T \geq 0$, and any non-empty set of modules **S**, $Z_{(T,\mathbf{S})}$ is defined by $Z_{(T,\mathbf{S})} = T\mathbf{min}(|\mathbf{S}| - 1)$.

The algorithms in the class $\mathcal{A}(Z_{(T,\mathbf{Ns})},K,a,\mathbf{Ns})$ will be defined recursively with respect to K. Here, K is the number of communication rounds, and a denotes the source module of the algorithm. The basis of the recursion is the case $K = 1$.

2.3.1. The construction of the algorithms in the class $\mathcal{A}(Z_{(T,\mathbf{Ns})},1,a,\mathbf{Ns})$. An algorithm in the class $\mathcal{A}(Z_{(T,\mathbf{Ns})},1,a,\mathbf{Ns})$ is based on only one round of information exchange. Let the

original message in a be $m(\underline{\mathbf{s}},a)$.

During round 0, module a sends the original message directly and unchanged to all modules in $(\mathbf{Ns} \setminus \{a\})$. Module a also keeps a copy of $m(\underline{\mathbf{s}},a)$ itself in order to use it in the decision-making process.

During round 1, the decision-making process is executed in which in each module $d \in (\mathbf{Ns} \setminus \{a\})$, the message $m(\underline{\mathbf{s}},a,d))$ (received from a) is taken as decision $dec_1((\underline{\mathbf{s}},a),d)$. The decision $dec_1((\underline{\mathbf{s}},a),a)$ in module a is equal to the stored message $m(\underline{\mathbf{s}},a)$.

The behavioural aspects of the algorithms in the class $\mathcal{A}(\ Z_{(T,\mathbf{Ns})},1,a,\mathbf{Ns})$ are defined by:

$$d \in (\mathbf{Ns} \setminus \{a\}) \Rightarrow m(\underline{\mathbf{s}},a,d) = m(\underline{\mathbf{s}},a)$$
$$d \in (\mathbf{Ns} \setminus \{a\}) \Rightarrow dec_1((\underline{\mathbf{s}},a),d) = m(\underline{\mathbf{s}},a,d)$$
$$dec_1((\underline{\mathbf{s}},a),a) = m(\underline{\mathbf{s}},a)$$

2.3.2. The recursive construction of the algorithms in the class $\mathcal{A}(\ Z_{(T,\mathbf{Ns})},K,a,\mathbf{Ns})$ with $K > 1$ in terms of algorithms from the set of classes $\mathcal{A}(\ Z_{(T,\mathbf{Ns}\setminus\{a\})},K\text{-}1,b,\mathbf{Ns}\setminus\{a\})$ with $b \in \mathbf{Ns}\setminus\{a\}$. Let the original message in a be $m(\underline{\mathbf{s}},a)$. The construction of the algorithms in the class $\mathcal{A}(\ Z_{(T,\mathbf{Ns})},K,a,\mathbf{Ns})$ with $K > 1$ is based on applying the partial encoder functions $\mathcal{Z}_{(a)}(b)$ (for every $b \in \mathbf{B}(a)$ where $\mathbf{B}(a) \subset (\mathbf{Ns} \setminus \{a\})$) of some $Z_{(T,\mathbf{Ns}\setminus\{a\})}$-erasure-correcting code $\mathcal{Z}_{(a)}$ on message $m(\underline{\mathbf{s}},a)$ (here, $Z_{(T,\mathbf{Ns}\setminus\{a\})} = T\mathbf{min}(|\mathbf{Ns} \setminus \{a\}| - 1)$). This results in the encoding of message $m(\underline{\mathbf{s}},a)$ into symbols of a $Z_{(T,\mathbf{Ns}\setminus\{a\})}$-erasure-correcting code. These symbols are denoted by $i_{(\underline{\mathbf{s}},a,b)}$. By applying authentication function $\mathcal{Y}_{(a)}$ to each symbol $i_{(\underline{\mathbf{s}},a,b)}$, the messages $m(\underline{\mathbf{s}},a,b)$ are obtained which are transmitted to the modules $b \in \mathbf{B}(a)$. Every module b forwards its message $m(\underline{\mathbf{s}},a,b)$ to the destinations by means of an algorithm from the class $\mathcal{A}(\ Z_{(T,\mathbf{Ns}\setminus\{a\})},K\text{-}1,b,\mathbf{Ns}\setminus\{a\})$.

The ADJC-algorithms in the class $\mathcal{A}(\ Z_{(T,\mathbf{Ns})},K,a,\mathbf{Ns})$ with $K > 1$ are constructed as follows:

(1) During round 0

(a) Let the original message in a be $m(\underline{\mathbf{s}},a)$. This message $m(\underline{\mathbf{s}},a)$ is kept stored in a in order to be used later on during round K in the decision-making process.

(b) Furthermore, in the source module a, for every $b \in \mathbf{B}(a)$ a partially encoded and authenticated version $m(\underline{\mathbf{s}},a,b)$ of the original message $m(\underline{\mathbf{s}},a)$ is calculated such that $m(\underline{\mathbf{s}},a,b) = \mathcal{Y}_{(a)}(\mathcal{Z}_{(a)}(b)(m(\underline{\mathbf{s}},a)))$.

(c) Each module $b \in \mathbf{B}(a)$ receives from a a message $m(\underline{\mathbf{s}},a,b)$. The next-set $\mathbf{B}(a)$ must contain at least $Z_{(T,\mathbf{Ns}\setminus\{a\})} + 1$ modules.

(2) During the rounds 1, … K - 1, each module b in the next-set $\mathbf{B}(a)$ forwards the received message $m(\underline{\mathbf{s}},a,b)$ to the destinations indicated by $(\mathbf{Ns} \setminus \{a\})$ by means of an algorithm from the class $\mathcal{A}(\ Z_{(T,\mathbf{Ns}\setminus\{a\})},K - 1,b,\mathbf{Ns} \setminus \{a\})$. The results of these algorithms in the destinations $d \in (\mathbf{Ns} \setminus \{a\})$ are denoted by $dec_{K-1}((\underline{\mathbf{s}},a,b),d)$. These results are calculated during the first part of the decision-making process which is executed in round K.

(3) During the second part of the decision-making process, executed during round K, the decision $dec_K((\underline{\mathbf{s}},a),d)$ in the modules d with $d \in (\mathbf{Ns} \setminus \{a\})$ is obtained as follows. First, we apply function $\mathcal{Y}_{(a)}^{(-1)}$ on the results $dec_{K-1}((\underline{\mathbf{s}},a,b),d)$ with $b \in \mathbf{B}(a)$. The resulting information parts $i_{(\underline{\mathbf{s}},a,b)}$ form the symbols of a $Z_{(T,\mathbf{Ns}\setminus\{a\})}$-erasure-correcting code $\mathcal{Z}_{(a)}$. The result of applying the corresponding decoder function $\mathcal{Z}_{(a)}^{(-1)}$ on this set of information parts is taken as decision $dec_K((\underline{\mathbf{s}},a),d)$. The decision $dec_K((\underline{\mathbf{s}},a),a)$ is obtained by taking the message value $m(\underline{\mathbf{s}},a)$ which had been kept stored in module a.

The behavioural aspects of the algorithms in the class $\mathcal{A}(\ Z_{(T,\mathbf{Ns})},K,a,\mathbf{Ns})$ with $K > 1$are defined by:

$$m(\underline{\mathbf{s}},a,b) = \mathcal{Y}_{(a)}(\mathcal{Z}_{(a)}(b)(m(\underline{\mathbf{s}},a))) \text{ for all } b \in \mathbf{B}(a)$$

$dec_{K-1}((\underline{\mathbf{s}},a,b),d)$ follows from $m(\underline{\mathbf{s}},a,b)$ based on an algorithm from the class $\mathcal{A}(\ Z_{(T,\mathbf{Ns}\setminus\{a\})},K - 1,b,\mathbf{Ns} \setminus \{a\})$

$d \neq a \Rightarrow dec_K((\underline{\mathbf{s}},a),d) = \mathcal{Z}_{(a)}^{(-1)}$ applied on the values $\mathcal{Y}_{(a)}^{(-1)}(dec_{K-1}((\underline{\mathbf{s}},a,b),d))$ with $b \in \mathbf{B}(a)$

$$d = a \Rightarrow dec_K((\underline{\mathbf{s}},a),d) = m(\underline{\mathbf{s}},a)$$

2.4.The existence of Authenticated Dispersed Joined Communication Algorithms in the classes $\mathcal{A}(\ Z_{(T,\mathbf{Ns})},K,a,\mathbf{Ns})$

The construction of ADJC-algorithms is possible if and only if

- A next-set $\mathbf{B}(a)$ can be found which is a non-empty subset of $(\mathbf{Ns} \setminus \{a\})$ and which contains at least $Z_{(T,\mathbf{Ns}\setminus\{a\})} + 1$ modules.
- A $Z_{(T,\mathbf{Ns}\setminus\{a\})}$-erasure-correcting code exists of which the code words consist of $|\mathbf{B}(a)|$ symbols.
- The classes $\mathcal{A}(\ Z_{(T,\mathbf{Ns}\setminus\{a\})},K - 1,b,\mathbf{Ns} \setminus \{a\})$ of ADJC-algorithms with $b \in \mathbf{B}(a)$ are all non-empty.

(5)

We are able to prove that:

THEOREM 1

- For all T with $T \geq 0$, and
- for all K with $K \geq 1$, and
- for all fully interconnected systems consisting of $|\ \mathbf{Ns}\ |$ modules, and
- for all source modules $a \in \mathbf{Ns}$,

the class of algorithms $\mathcal{A}(Z_{(T,\mathbf{Ns})},K,a,\mathbf{Ns})$ is non-empty if and only if $|\ \mathbf{Ns}\ | \geq K + 1$ ❑

The proof of this theorem is beyond the scope of this paper.

2.5.Some behavioural properties of the Authenticated Dispersed Joined Communication algorithms in the presence of at most $Z_{(T,\mathbf{Ns})}$ maliciously behaving modules

The ADJC-algorithms satisfy the following behavioural properties:

THEOREM 2

Let **Ns** be a set of fully interconnected modules. At most $Z_{(T,\mathbf{Ns})}$ of these modules behave maliciously. Suppose that by means of any ADJC-algorithm from the class $\mathcal{A}(Z_{(T,\mathbf{Ns})},K,a,\mathbf{Ns})$ a message $m(a)$ is transmitted from the source module a to all modules in the set **Ns**. Let the decisions calculated in the destinations d with $d \in \mathbf{Ns}$ be denoted by $dec_K((a),d)$, then

- 1. If the source module a and a destination d are both well-functioning, then the result $dec_K((a),d)$ of the algorithm calculated in d equals the original message $m(a)$ in a.
- 2. For any two well-functioning destinations d and e, it holds that if the results $dec_K((a),d)$ and $dec_K((a),e)$ are unequal, then the number of maliciously behaving modules in the system is at least K ❑

Proof:

We first prove property 1.

Assume that a and d are two well-functioning modules in the system ($a,d \in \mathbf{Ns}$) in which a is the source module of the algorithm, and d is one of the destinations.

If $a = d$, then, according to the construction of the algorithms of class $\mathcal{A}(Z_{(T,\mathbf{Ns})},K,a,\mathbf{Ns})$ described in Section 2.3, it holds that $dec_K((a),d) = m(a)$.

So we only need to consider the case $d \neq a$, i.e. $d \in (\mathbf{Ns} \setminus \{a\})$. For these cases we prove $dec_K((a),d) = m(a)$ by induction with respect to K.

Basis: $K = 1$

Algorithms in the class $\mathcal{A}(Z_{(T,\mathbf{Ns})},1,a,\mathbf{Ns})$ send the message $m(a)$ directly and unchanged to the destination d, and the decision calculated in d equals the message received from a. So, because we assume that a and d are functioning correctly, it holds that $dec_1((a),d) = m(a)$.

Induction step: $K > 1$.

The algorithms in the class $\mathcal{A}(Z_{(T,\mathbf{Ns})},K,a,\mathbf{Ns})$ have been constructed from the algorithms in the class $\mathcal{A}(Z_{(T,\mathbf{Ns}\setminus\{a\})},K-1,b,\mathbf{Ns} \setminus \{a\})$ with $b \in \mathbf{B}(a)$ and $\mathbf{B}(a) \subset (\mathbf{Ns} \setminus \{a\})$.

From the latter algorithms, we know from the induction hypothesis that if a message $m(a,b)$ is communicated by a well-functioning source module b to a well-functioning destination d in $(\mathbf{Ns} \setminus \{a\})$, then

$$dec_{K-1}((a,b),d) = m(a,b) \qquad (6)$$

From the construction we know the following:

In module a, for all $b \in \mathbf{B}(a)$, by means of the partial encoder functions $\mathcal{Z}_{(a)}(b)$, the original message is encoded into $|\mathbf{B}(a)|$ symbols. By applying $\mathcal{Y}_{(a)}$ to such a symbol, a message digest is appended to it, resulting in a message $m(a,b)$. During round 0, every module $b \in \mathbf{B}(a)$ receives a message $m(a,b)$ where:

$$m(a,b) = \mathcal{Y}_{(a)}(\mathcal{Z}_{(a)}(b)(m(a))) \qquad (7)$$

Each of these modules b forwards the message $m(a,b)$ to the destinations in $(\mathbf{Ns} \setminus \{a\})$ by means of an algorithm from the class $\mathcal{A}(Z_{(T,\mathbf{Ns}\setminus\{a\})},K-1,b,\mathbf{Ns} \setminus \{a\})$. As the result of these algorithms, in each destination d of the set $(\mathbf{Ns} \setminus \{a\})$, $|\mathbf{B}(a)|$ decisions $dec_{K-1}((a,b),d)$ will become available. If module b is functioning correctly, then, according to (6) and (7), it holds that

$$dec_{K-1}((a,b),d) = \mathcal{Y}_{(a)}(\mathcal{Z}_{(a)}(b)(m(a))) \qquad (8)$$

In each module d, function $\mathcal{Y}_{(a)}^{(-1)}$ is applied to these decisions $dec_{K-1}((a,b),d)$ with $b \in \mathbf{B}(a)$.

From the construction, we know that the next-set $\mathbf{B}(a)$ for algorithms in the class $\mathcal{A}(Z_{(T,\mathbf{Ns})},K,a,\mathbf{Ns})$ contains at least $(T + 1)\mathbf{min}(|\mathbf{Ns} \setminus \{a\}|)$ modules.

Now, we can distinguish two cases:

1. $|\ \mathbf{Ns} \setminus \{a\}\ | \geq T + 1$
2. $|\ \mathbf{Ns} \setminus \{a\}\ | < T + 1$

Case 1: $|\ \mathbf{Ns} \setminus \{a\}\ | \geq T + 1$

Since $|\ \mathbf{Ns} \setminus \{a\}\ | \geq T + 1$, the next-set $\mathbf{B}(a)$ contains at least $T + 1$ modules. So, $\mathbf{B}(a)$ contains at least one well-functioning module, and thus, module d contains at least one decision which satisfies (8).

Case 2: $|\ \mathbf{Ns} \setminus \{a\}\ | < T + 1$

In this case, the next-set $\mathbf{B}(a)$ contains $|\ \mathbf{Ns} \setminus \{a\}\ |$ modules. Therefore, since $\mathbf{B}(a) \subset (\mathbf{Ns} \setminus \{a\})$, we must conclude that $\mathbf{B}(a) = (\mathbf{Ns} \setminus \{a\})$. Because $d \in (\mathbf{Ns} \setminus \{a\})$, and we assumed that d is a well-functioning module, $\mathbf{B}(a)$ contains

at least one well-functioning module (viz. module d), and thus, module d contains at least one decision which satisfies (8).

So, in both cases, module d contains at least one decision which satisfies (8). From the foregoing, we see that at most $Z_{(T,\mathbf{Ns}\setminus\{a\})}$ modules b behave maliciously (Recall that $Z_{(T,\mathbf{Ns}\setminus\{a\})} = T\mathbf{min}(|\mathbf{Ns}\setminus\{a\}|\text{-}1)$). So at most $Z_{(T,\mathbf{Ns}\setminus\{a\})}$ decisions do not satisfy (8). The code $\mathcal{Z}_{(a)}$ is $Z_{(T,\mathbf{Ns}\setminus\{a\})}$-erasure-correcting, and thus, the result of applying decoder function $\mathcal{Z}_{(a)}^{(-1)}$ on the values $\mathcal{Y}_{(a)}^{(-1)}(dec_{K-1}((a,b),d))$ (which is taken as final decision $dec_K((a),d)$) must be equal to $m(a)$. This completes the proof of property 1.

Now we prove property 2.

Assume that two destinations d and e, with $d,e \in \mathbf{Ns}$ behave correctly and that $dec_K((a),d) \neq dec_K((a),e)$.

If the source module a functions correctly then, by property 1, it holds that $dec_K((a),d) = m(a) = dec_K((a),e)$ conflicting the assumption $dec_K((a),d) \neq dec_K((a),e)$. Thus, module a is behaving maliciously and thus (since we assumed that d and e function correctly), $d \neq a$ and $e \neq a$. Again, we use induction with respect to K.

Basis: $K = 1$

We already concluded that module a is behaving maliciously. Hence, the system contains at least one maliciously behaving module.

Induction step: $K > 1$

Recall that during round 0 the symbols are sent by module a to the modules b with $b \in \mathbf{B}(a)$. During the rounds 1, …K, each of these symbols $m(a,b)$ is forwarded to the destinations in the set ($\mathbf{Ns} \setminus \{a\}$) by means of an algorithm from the class $\mathcal{A}(Z_{(T,\mathbf{Ns}\setminus\{a\})},K\text{-}1,b,\mathbf{Ns}\setminus\{a\})$. The results of these algorithms calculated in the modules d with $d \in (\mathbf{Ns} \setminus\{a\})$ are denoted by $dec_{K-1}((a,b),d)$.

Let $dec_{K-1}((a,b),d)$ and $dec_{K-1}((a,b),e)$ with $b \in \mathbf{B}(a)$ be decisions calculated in modules d and e. We already concluded that module a is behaving maliciously and that $d \neq a$ and $e \neq a$. Hence, $d,e \in (\mathbf{Ns} \setminus\{a\})$. Since $d \neq a$ and $e \neq a$, the decision $dec_K((a),d)$ is based on applying the functions $\mathcal{Y}_{(a)}^{(-1)}$ and $\mathcal{Z}_{(a)}^{(-1)}$ on the decisions $dec_{K-1}((a,b),d)$ with $b \in \mathbf{B}(a)$, whereas the decision $dec_K((a),e)$ is based on applying the same functions $\mathcal{Y}_{(a)}^{(-1)}$ and $\mathcal{Z}_{(a)}^{(-1)}$ on the decisions $dec_{K-1}((a,b),e)$ with $b \in \mathbf{B}(a)$. It follows that

$$\forall\, b : b \in \mathbf{B}(a) \Rightarrow dec_{K-1}((a,b),d) = dec_{K-1}((a,b),e) \tag{9}$$

would imply $dec_K((a),d) = dec_K((a),e)$. The latter, however, conflicts with the assumption $dec_K((a),d) \neq dec_K((a),e)$, so we conclude

$$\exists\, b : b \in \mathbf{B}(a) \wedge dec_{K-1}((a,b),d) \neq dec_{K-1}((a,b),e) \tag{10}$$

Recall that our assumption implies $d,e \in (\mathbf{Ns} \setminus \{a\})$. So from the definition of the construction of the algorithms in the class $\mathcal{A}(Z_{(T,\mathbf{Ns})},K,a,\mathbf{Ns})$ we know that the decisions $dec_{K-1}((a,b),d)$ resp. $dec_{K-1}((a,b),e)$ are the result of algorithms from the classes $\mathcal{A}(Z_{(T,\mathbf{Ns}\setminus\{a\})},K\text{-}1,b,\mathbf{Ns} \setminus \{a\})$ with $b \in \mathbf{B}(a)$.

According to the induction hypothesis, it holds for the latter classes that if the modules d and e are both functioning correctly and $dec_{K-1}((a,b),d) \neq dec_{K-1}((a,b),e)$, then the number of maliciously behaving modules in the set ($\mathbf{Ns} \setminus \{a\}$) must be at least K-1. And thus, with (10), we conclude that the set ($\mathbf{Ns} \setminus \{a\}$) must contain at least K-1 maliciously behaving modules. We already concluded that module a behaves maliciously. Hence, the set $\mathbf{Ns}$ contains at least K maliciously behaving modules.

This completes the proof of Theorem 2. ❑

3.A class of algorithms for reaching interactive consistency based on the ADJC-algorithms

The class of algorithms for reaching interactive consistency based on authentication and error-correcting codes is a subclass of the ADJC-algorithms defined by:

$$\mathcal{A}(T,K,a,\mathbf{Ns}) \text{ with } T \geq 0, \text{ and } K = T + 1 \tag{11}$$

From Theorem 1, we find that the class of algorithms $\mathcal{A}(T,T+1,a,\mathbf{Ns})$ is non-empty if and only if $|\mathbf{Ns}| \geq T + 2$.

THEOREM 3

Any Interactive Consistency Algorithm from the class $\mathcal{A}(T,T+1,a,\mathbf{Ns})$, with $|\mathbf{Ns}| \geq T + 2$ satisfies the interactive consistency conditions, i.e.:

$$\forall\, d : a,d \in \overline{\mathbf{F}} \Rightarrow dec_{T+1}((a),d) = m(a)$$

$$\text{and } \forall\, d,e : d,e \in \overline{\mathbf{F}} \Rightarrow dec_{T+1}((a),d) = dec_{T+1}((a),e)$$

in which $\overline{\mathbf{F}}$ is any set of well-functioning modules such that $|\overline{\mathbf{F}}| \geq |\mathbf{Ns}| - T$, and $\overline{\mathbf{F}} \subset \mathbf{Ns}$ and $dec_{T+1}((a),d)$ (resp. $dec_{T+1}((a),e)$) with $d,e \in \mathbf{Ns}$ denotes the decision in a module d (resp. e) about what module a tried to send. ❑

Proof:

Consider Interactive Consistency Algorithms which are defined by the non-empty class of ADJC-algorithms $\mathcal{A}(T,T+1,a,\mathbf{Ns})$ with $T \geq 0$, and $|\mathbf{Ns}| \geq T + 2$.

Let $\overline{\mathbf{F}}$ be any set of well-functioning modules such that $|\overline{\mathbf{F}}| \geq |\mathbf{Ns}| - T$, and $\overline{\mathbf{F}} \subset \mathbf{Ns}$.

From the first part of Theorem 2, we know that if the source module a and a destination d are both well-functioning, then the result $dec_K((a),d)$ of the algorithm calculated in d equals the original message $m(a)$ in a. Thus:

$$\forall\ d : a,d \in \overline{\mathbf{F}} \Rightarrow dec_K((a),d) = m(a)$$

This proves the first part of the interactive consistency properties.

From the second part of Theorem 2, we know that for any two well-functioning destinations d and e it holds that if the results $dec_K((a),d)$ and $dec_K((a),e)$ are unequal, then the number of maliciously behaving modules in the system is at least K

However, this conflicts with the constraint $K = T + 1$ and the assumption that at most T modules behave maliciously. Thus, if both modules d and e are behaving correctly, the decisions $dec_K((a),d)$ and $dec_K((a),e)$ must be identical. So

$$\forall\ d,e : d,e \in \overline{\mathbf{F}} \Rightarrow dec_K((a),d) = dec_K((a),e).$$

This completes the proof of Theorem 3. ❑

4.Some remarks on the construction of Interactive Consistency Algorithms which are based on authentication and error-correcting codes

Some freedom is left in the definition of the Interactive Consistency Algorithms based on authentication and error-correcting codes, starting from the ADJC-algorithms. A reduction in the number of messages that needs to be transmitted between the modules can be obtained in two ways, viz.:

- by minimizing the number of directions in which the messages are broadcast each round. This is done in the so-called Minimum Direction algorithms, described in 4.1.
- by maximizing the length of the applied erasure-correcting code. This is done in the so-called Maximum Coding algorithms, described in 4.2.

From the construction we immediately see that maximizing the length of the error-correcting code causes an increase in the number of modules to which messages are sent. However, the size of the messages that is transmitted decreases with an increasing code length. We will show that this decrease is more efficient than reducing the number of directions.

4.1. The Minimum Direction algorithms

In this subclass of ADJC-algorithms, only repetition codes are applied. Thus, during each communication round a received message $m(\underline{\mathbf{s}})$ is broadcast unchanged to a number of modules given by the set $\mathbf{B}(\underline{\mathbf{s}})$. During each round, except the last one, each message is relayed to exactly $Z_{(T,\mathbf{Ns} \setminus \mathrm{set}(\underline{\mathbf{s}}))}$ modules. (For a string $\underline{\mathbf{s}}$, set($\underline{\mathbf{s}}$) is defined as the set of modules contained in string $\underline{\mathbf{s}}$). During the last round a message $m(\underline{\mathbf{s}})$ is sent to the modules in $\mathbf{Ns} \setminus \mathrm{set}(\underline{\mathbf{s}})$.

4.2. The Maximum Coding algorithms

When a $Z_{(T,\mathbf{Ns} \setminus \mathrm{set}(\underline{\mathbf{s}}))}$-erasure-correcting code $\mathcal{Z}_{(\underline{\mathbf{s}})}$ is applied, consisting of code words of $n_{(\underline{\mathbf{s}})}$ symbols of size $b_{(\underline{\mathbf{s}})}$ and data words consisting of $k_{(\underline{\mathbf{s}})}$ symbols of the same size, then the total amount of information broadcast by a module during round t ($1 \le t \le T$-1) is a factor $n_{(\underline{\mathbf{s}})}/k_{(\underline{\mathbf{s}})}$ more than the amount of information received by that module during round t - 1. (Notice that in the last communication round, no encoding takes place). From [12], we know that a $Z_{(T,\mathbf{Ns} \setminus \mathrm{set}(\underline{\mathbf{s}}))}$-erasure-correcting code $\mathcal{Z}_{(\underline{\mathbf{s}})}$ can be constructed if and only if $n_{(\underline{\mathbf{s}})} \ge k_{(\underline{\mathbf{s}})}+Z_{(T,\mathbf{Ns} \setminus \mathrm{set}(\underline{\mathbf{s}}))}$. In the Maximum Coding algorithms, the fraction $n_{(\underline{\mathbf{s}})}/k_{(\underline{\mathbf{s}})}$ is kept as low as possible. This is achieved by maximizing $k_{(\underline{\mathbf{s}})}$, while keeping $n_{(\underline{\mathbf{s}})}$ as low as possible. We choose $n_{(\underline{\mathbf{s}})} = k_{(\underline{\mathbf{s}})}+Z_{(T,\mathbf{Ns} \setminus \mathrm{set}(\underline{\mathbf{s}}))}$, and maximize $k_{(\underline{\mathbf{s}})}$, i.e. the partial encoding is spread across as many modules as is allowed. Codes with these parameters always exist if the symbol size $b_{(\underline{\mathbf{s}})}$ satisfies $b_{(\underline{\mathbf{s}})} \ge {}^2\log(n_{(\underline{\mathbf{s}})}$-1) [1]. They are the so-called Maximum Distance Separable (MDS) codes. We choose the set $\mathbf{B}(\underline{\mathbf{s}})$ to be as large as possible, thus

$$|\mathbf{B}(\underline{\mathbf{s}})| = |\mathbf{Ns} \setminus \mathrm{set}(\underline{\mathbf{s}})| \text{ for } 1 \le |\underline{\mathbf{s}}| \le T$$

In each round t (with $|\underline{\mathbf{s}}| = t$+1 and $0 \le t \le T$-1) of a Maximum Coding algorithm, a $Z_{(T,\mathbf{Ns}\setminus\mathrm{set}(\underline{\mathbf{s}}))}$-erasure-correcting code is applied with parameters:

- *number of code word symbols:*
$$n(\underline{\mathbf{s}}) = N - t - 1$$
- *number of data word symbols:*
$$k(\underline{\mathbf{s}}) = N - t - 1 - Z_{(T,\mathrm{Ns}\setminus\mathrm{set}(\underline{\mathbf{s}}))}$$
- *symbol size in number of bits:*
$$b(\underline{\mathbf{s}}) \ge {}^2\log(N\text{-}t\text{-}2) \quad \text{if } N - t - 1 > T + 1$$
$$b(\underline{\mathbf{s}}) \ge 1 \quad \text{if } N - t - 1 \le T + 1$$

5. A comparison with existing algorithms

We will now compare our IAC-algorithms with some well-known synchronous deterministic IAC-algorithms with authentication, described in the literature. They are the SM algorithm described in [3], referred to as the Lamport-algorithm, and the algorithm described in Theorem 6 in [9], which we will call the Dolev-algorithm.

5.1. The criteria

The criteria by means of which the algorithms will be compared are:

- The relative number of messages, *mess*, that needs to be transmitted between the modules in terms of the original message broadcast by the source
- The minimum size, *msize*, of the original message.

5.2. The Lamport-algorithm

In the Lamport-algorithm, the required number of messages *mess* is:

$$mess = \sum_{i=1}^{T+1} i! \cdot \begin{bmatrix} N-1 \\ i \end{bmatrix}$$

We choose the amount of information of the original message to be broadcast by the source be the unit. So the message size *msize* of all messages in the Lamport-algorithm is 1.

5.3. The Dolev-algorithm

For the Dolev-algorithm, we calculate the maximum number of messages (*mess*) that is required. We distinguish between the case $N > 2T+1$ and the case $N \leq 2T+1$.

5.3.1. The case $N > 2T+1$. In case $N > 2T+1$, the maximally required number of messages mess is given by the following formulas. If $T=0$, only 1 round of communication is performed and thus the number of messages $mess = N-1$. If $T=1$, two communication rounds are necessary and the required number of messages $mess = (N-1) + (2T+1)\cdot(N-2) = 4N-7$. If $T \geq 2$, then:

$$mess = (N-1) + (2T+1) \cdot (N-2) + (2T+1) \cdot (N-3)$$

5.3.2. The case $T+2 \leq N \leq 2T+1$. In case $T+2 \leq N \leq 2T+1$, the maximally required number of messages *mess* is given by the following formulas. If $T = 1$, two communication rounds are necessary and the required number of messages $mess = (N-1) + (N-1)\cdot(N-2) = 4$ (since $N = 3$). If $T \geq 2$, then:

$$mess = (N-1) + (N-1) \cdot (N-2) + (N-1) \cdot (N-3)$$

The message size *msize* of all messages in the Dolev-algorithm is 1.

5.4 The Minimum Direction algorithm

If $N > T+2$, there is some design freedom in the algorithms that reach Byzantine agreement.

In the Minimum Direction algorithms, therefore, in each communication round, a message is relayed to a set of $T+1$ receivers that did not receive the message before. If $N < 2T+1$, then after $N-T-1$ rounds, such a set can not be found anymore. In this case, the message is relayed to all T processors that did not receive the message before. In the next round, these processors, in turn, relay the message to T-1 others etc. In the last communication round, the message is sent to all $(N-T-1)$ processors that did not receive the message before.

Since we can only use repetition codes here, the message size *msize* of all messages in the algorithm is 1. In the following we will calculate the number of messages *mess* that is required for the case $N \geq 2T+1$ resp. $N < 2T+1$.

5.4.1. The case $N \geq 2T+1$. Here, the required number of messages *mess* is:

$$mess = \left(\sum_{i=1}^{T} (T+1)^i \right) + (T+1)^T \cdot (N-T-1)$$

The first term expresses the number of messages in the first T communication rounds, whereas the second term expresses the number of messages sent in round $T+1$.

5.4.2. The case $T+2 \leq N < 2T+1$. The required number of messages *mess* is:

$$\left(\sum_{i=1}^{N-T-1} (T+1)^i \right) + \left(\sum_{i=N-T}^{T+1} (T+1)^{N-T-1} \cdot (i-N+T+1)! \cdot \begin{bmatrix} T \\ i-N+T+1 \end{bmatrix} \right)$$

The first term expresses the number of messages in the first $N-T-1$ communication rounds, whereas the second term expresses the number of messages sent in the remaining rounds.

5.5. The Maximum Coding algorithm

The number of messages that is exchanged between the modules during the broadcast process can be calculated as follows. We will restrict ourselves to algorithms in which the choice of the code only depends on the round in which the encoding of the messages is performed. Let a code used during round t (where $0 \leq t \leq T-1$) consist of code words of $n_c(t)$ symbols, obtained from data words of $k_c(t)$ symbols, let the symbol size be $b_c(t)$ bits. Then, the total amount of messages that is transmitted during the broadcast process in terms of the number of unit messages, is [11]:

$$mess = (N - T - 1) \cdot \prod_{t=0}^{T-1} \left(\frac{n_c(t)}{k_c(t)} \right) + \sum_{t=0}^{T-1} \prod_{i=0}^{t} \left(\frac{n_c(i)}{k_c(i)} \right)$$

Now, we will derive an expression for the minimal size *msize* of the message in the source. In every round t (where $1 \le t \le T\text{-}1$), the product of the message symbol size and the number of symbols must be equal to the size of the data word in round t-1. Thus,

$$\forall\, t : 1 \le t \le T\text{-}1 : b_c(t\text{-}1) = k_c(t).b_c(t) \qquad (13)$$

For the original message in the source, we require:

$$msize = k_c(0).b_c(0) \qquad (14)$$

Moreover, the symbol size of each code must be sufficiently large, so in the case of MDS codes, we need to satisfy

$$\forall\, t : 0 \le t \le T - 1 : ((k_c(t) = 1 \Rightarrow b_c(t) \ge 1) \wedge (k_c(t) \ge 2 \Rightarrow b_c(t) \ge {}^2\log(n_c(t) - 1))) \qquad (15)$$

From relations (13), (14), and (15), *msize* can be calculated.

5.6. Results

In the tables on the previous two pages we compare the amount of data communication needed by the various algorithms. Here, the Minimum Direction and Maximum Coding algorithms are indicated by MinDir and MaxCod respectively. From these results, we conclude that for a small number of modules, the Dolev-algorithm is to be preferred, but for larger number of modules the Maximum Coding algorithm is favorable. When fault-tolerant services are implemented using the state machine approach [13], at least $2T$+1 modules are needed. For $N \ge 2T + 1$, the Maximum Coding algorithm is more efficient than the Dolev-algorithm. E.g., for T=2, the minimally required relative number of messages *mess* for the Maximum Coding algorithm is 14.1 (for $N = 8$), whereas for the Dolev-algorithm this number is 24 (for $N = 5$). However, the minimally required size *msize* of the initial message in the Maximum Coding algorithm grows with an increasing number of modules N. For large N this initial message size may become impractical.

6. Conclusion

This paper describes and proves a new class of interactive consistency algorithms based on authentication and error-correcting codes on basis of a class of algorithms called the Authenticated Dispersed Joined Communication algorithms. We showed that for practical values of N and T these algorithms require considerably less data communication than existing algorithms based on authenticated messages.

7. References

[1] Krol, Th., **A Generalization of Fault-Tolerance Based on Masking**, Ph.D. thesis, Eindhoven University of Technology, 1991.

[2] Pease, M., Shostak, R., and Lamport, L., Reaching agreement in the presence of faults, in: **Journal of the ACM**, Vol.27, No.2, April 1980, pp.228-234.

[3] Lamport, L., Shostak, R., and Pease, M., The Byzantine Generals Problem, in: **ACM Transactions on Programming Languages and Systems**, Vol.4, No.3, July 1982, pp.382-401.

[4] Rivest, R.L., Shamir, A., and Adleman, L., A method for obtaining digital signatures and public-key cryptosystems, in **Comm.ACM**, 21 (1978), pp.120-126.

[5] Beckett, B., **Introduction to cryptology**, Blackwell Scientific Publications,1988.

[6] Mullender, S.J., **Distributed systems**, Addison-Wesley, New-York, 1993, p.533.

[7] Barborak, M., Malek, M., and Dahbura, A., The Consensus Problem in Fault-Tolerant Computing, in: **ACM Computing Surveys**, Vol.25, No.2, June 1993.

[8] Dolev, D., and Reischuk, R., Bounds on Information Exchange for Byzantine Agreement, in: **Journal of the ACM**, Vol.32, No.1, January 1985, pp.191-204.

[9] Dolev, D., and Strong, H.R., Authenticated algorithms for Byzantine agreement, in: **Siam J. Comput**., Vol.12, No.4, 1983, pp.656-666.

[10] Rabin, M.O., Efficient Dispersal of Information for Security, Load Balancing, and Fault Tolerance, in: **Journal of the ACM**, Vol.36, No.2, April 1989, pp.335-348.

[11] Krol, Th., Interactive Consistency Algorithms Based on Voting and Error-Correcting Codes, in: **The Twenty-Fifth International Symposium on Fault-Tolerant Computing, Digest of Papers, FTCS-25 Silver Jubilee**, IEEE Computer Society Press, Los Alamitos, California, June 1995, pp.89-98.

[12] Vanstone, S.A., and van Oorschot,P.C., **An Introduction to Error Correcting Codes with Applications**, Kluwer Academic Publishers, Boston, 1989.

[13] Schneider, F.B., Implementing Fault-Tolerant Services Using the State Machine Approach: A Tutorial', in: **ACM Computing Surveys**, Vol.22, No.4, December 1990, pp.299-319.

.

Table 1: *T* = 1

Algorithm	*N*	*msize*	*mess*	applied codes [*n*,*k*,*b*]
Lamport	3	1	4	
Dolev	3	1	4	
MinDir	3	1	4	[2,1,1]
MaxCod	3	1	4	[2,1,1]
Lamport	4	1	9	
Dolev	4	1	9	
MinDir	4	1	6	[2,1,1]
MaxCod	4	2	4.5	[3,2,1]
Lamport	5	1	16	
Dolev	5	1	13	
MinDir	5	1	8	[2,1,1]
MaxCod	5	6	5.3	[4,3,2]
Lamport	16	1	256	
Dolev	16	1	57	
MinDir	16	1	30	[2,1,1]
MaxCod	16	56	16.1	[15,14,4]

Table 2: *T* = 2

Algorithm	*N*	*msize*	*mess*	applied codes [*n*,*k*,*b*]
Lamport	4	1	15	
Dolev	4	1	12	
MinDir	4	1	15	[3,1,1]-[2,1,1]
MaxCod	4	1	15	[3,1,1]-[2,1,1]
Lamport	5	1	40	
Dolev	5	1	24	
MinDir	5	1	30	[3,1,1]-[3,1,1]
MaxCod	5	4	20	[4,2,2]-[3,1,1]
Lamport	6	1	85	
Dolev	6	1	40	
MinDir	6	1	39	[3,1,1]-[3,1,1]
MaxCod	6	12	15	[5,3,4]-[4,2,2]
Lamport	8	1	259	
Dolev	8	1	62	
MinDir	8	1	57	[3,1,1]-[3,1,1]
MaxCod	8	60	14.1	[7,5,12]-[6,4,3]
Lamport	16	1	2955	
Dolev	16	1	150	
MinDir	16	1	129	[3,1,1]-[3,1,1]
MaxCod	16	624	20	[15,13,48]-[14,12,4]

Table 3: *T* = 3

Algorithm	*N*	*msize*	*mess*	applied codes [*n*,*k*,*b*]
Lamport	5	1	64	
Dolev	5	1	24	
MinDir	5	1	64	[4,1,1]-[3,1,1]-[2,1,1]
MaxCod	5	1	64	[4,1,1]-[3,1,1]-[2,1,1]
Lamport	7	1	516	
Dolev	7	1	60	
MinDir	7	1	276	[4,1,1]-[4,1,1]-[4,1,1]
MaxCod	7	12	94	[6,3,4]-[5,2,2]-[4,1,1]
Lamport	9	1	2080	
Dolev	9	1	99	
MinDir	9	1	404	[4,1,1]-[4,1,1]-[4,1,1]
MaxCod	9	180	39.1	[8,5,36]-[7,4,9]-[6,3,3]
Lamport	14	1	19045	
Dolev	14	1	174	
MinDir	14	1	724	[4,1,1]-[4,1,1]-[4,1,1]
MaxCod	14	2880	29.4	[13,10,288]-[12,9,32]-[11,8,4]
Lamport	16	1	35715	
Dolev	16	1	204	
MinDir	16	1	852	[4,1,1]-[4,1,1]-[4,1,1]
MaxCod	16	5280	29.8	[15,12,440]-[14,11,40]-[13,10,4]

Table 4: *T* = 4

Algorithm	*N*	*msize*	*mess*	applied codes [*n*,*k*,*b*]
Dolev	6	1	40	
MaxCod	6	1	272	[5,1,1]-[4,1,1]-[3,1,1]-[2,1,1]
Dolev	9	1	112	
MaxCod	9	72	370.7	[8,4,18]-[7,3,6]-[6,2,3]-[5,1,1]
Dolev	21	1	353	
MaxCod	21	174720	50.1	[20,16,10920]-[19,15,728]-[18,14,52]-[17,13,4]
Dolev	22	1	372	
MaxCod	22	285600	50	[21,17,16800]-[20,16,1050]-[19,15,70]-[18,14,5]
Dolev	23	1	391	
MaxCod	23	367200	50.03	[22,18,20400]-[21,17,1200]-[20,16,75]-[19,15,5]