Homonyms with forgeable identifiers

Carole Delporte-Gallet, Hugues Fauconnier, Hung Tran-The

Homonyms with forgeable identifiers

Carole Delporte-Gallet[1], Hugues Fauconnier[1], and Hung Tran-The[1]

LIAFA- Université Paris-Diderot
email:{cd,hf,Hung.Tran-The}@liafa.jussieu.fr

Abstract. We consider here the Byzantine Agreement problem (BA) in synchronous systems with *homonyms* in the case where some identifiers may be forgeable. More precisely, the n processes share a set of l ($1 \leq l \leq n$) identifiers. Assuming that at most t processes may be Byzantine and at most k ($t \leq k \leq l$) of these identifiers are forgeable in the sense that any Byzantine process can falsely use them, we prove that Byzantine Agreement problem is solvable if and only if $l > 2t + k$. Moreover we extend this result to systems with authentication by signatures in which at most k signatures are forgeable and we prove that Byzantine Agreement problem is solvable if and only if $l > t + k$.

Regular paper

1 Introduction

Most of distributed algorithms assume that each process has an unique identity. Yet assuming that every process has unique identity given by a checkable identifier might be too strong (and costly) an assumption in practice especially if some processes are malicious. For example, very simple systems giving MAC addresses as identifier are not reliable, because MAC addresses may be duplicated and more sophisticated mechanisms using for example digital signatures are costly and difficult to implement. Moreover, for privacy reason, the processes may prefer to not have an unique identifier. For example, agents may be registered as members of groups and may want to act as member of the groups not as individuals. Hence it could be useful and interesting to relax the unicity of identifiers assumption. However, lack of identifiers is very restrictive and in fully anonymous systems very few problems are solvable (e.g. [1, 3, 4, 11]).

In [6], the authors presented a general model with homonyms in which processes may share the same identifier and may then be homonyms. In this model n processes share a set of l identifiers ($1 \leq l \leq n$). More precisely, each process p has an unique identifier belonging to the set of l identifiers, but several processes may have the same identifier. The processes cannot distinguish between processes having the same identifier. A process may only send messages to *all the* processes with some identifier and when a process receives a message it knows only the identifier of the origin of the message without knowing which particular process it is. When $l = n$ each proces has its own identifier and at the other extreme the case $l = 1$ corresponds to the fully anonymous system.

An important point in the model described in [6] is that there is no "masquerading" and even a Byzantine process is not able to lie about its identifier: if any process p receives a message, then process p knows that the sender of this message has identifier id. Hence even a Byzantine

process cannot lie about its identifier. Yet it is rather natural that Byzantine processes may at least form a coalition and "exchange" their identifiers.

In this paper we present an extension of homonyms processes in which some identifiers are "forgeable" and a Byzantine process may freely use any such identifier. Moreover we restrict ourselves to the classical synchronized rounds model. More precisely, we keep on ensuring that each process has an unique identity, but some identifiers are forgeable in the sense that Byzantine processes may use such an identifier id to send messages. A process receiving this message falsely believes that the identifier of the sender is id. Of course the set of forgeable identifiers is not known by the processes, we only assume that we have l identifiers for the n processes, and among these identifiers at most k are forgeable. As Byzantine are able to form coalitions and exchange their identifies, we assume that the set of forgeable identifiers contains at least all identifiers of Byzantine processes. Hence if t is the maximum number of Byzantine processes, we have $t \leq k \leq l$.

To determine the power of the model of homonyms with forgeable identifiers, as in [6], we are going to consider the problem of Byzantine agreement [13]. As a Byzantine process is able to send in each round messages with all forgeable identifiers, intuitively, it means that it is the same as having at least one Byzantine process per forgeable identifier. Recall from [6] that Byzantine agreement is solvable in the homonyms model if and only if $l > 3t$, then considering forgeable identifiers as similar to groups of processes with Byzantine processes, we get directly a solution with l forgeable identifiers if $l > 3k$ and we could suppose that we have a solution if and only if $l > 3k$. But surprisingly, we prove a better bound, we prove that there is solution for the Byzantine agreement with k forgeable identifiers if and only if $l > 2t + k$. In fact, this result comes from the fact that if a Byzantine process forges the identifier a group of process containing correct processes, this group of processes has the same behaviour as a group of processes containing Byzantine process and correct processes together and it is proven in [7] that such groups of processes are weaker adversaries than groups containing only Byzantine processes.

From a more practical point of view, it is easy to implement homonyms with help of digital signatures as with [9] in which at each identifier is associated a public key and processes with the same identifiers share corresponding private keys. In this way we get a (strictly) stronger authentication mechanism as defined in [14]. With this authentication mechanism a process cannot retransmit falsely messages. More precisely, if the identifier is unforgeable, then it is not possible for any process q to wrongly pretend that it received message m coming from a process with this identifier. It is well known that with this kind of authentication, the Byzantine agreement problem can be solved if and only if $n > 2t$ in the classical case in which all processes have unique and different unforgeable identifiers, giving a $n > 2k$ bound with k forgeable identifiers. With homonyms and at most k forgeable identifiers, we prove that Byzantine agreement is solvable if and only if $l > t + k$. Again the *a priori* expected result would be $l > 2k$.

Due to the lack of space some proofs are omitted and will appear in the full version of the paper.

The rest of the paper is organized as follows. Section 2 describes the model of homonyms with forgeable identifiers and gives the specification of Byzantine Agreement. Then in Section 3

we prove the impossibility results concerning Byzantine Agreement with homonyms and forgeable identifiers. In Section 4, we propose a specification of Authenticated Broadcast and give a corresponding algorithm. Section 5 contains the algorithm for Byzantine Agreement using Authenticated broadcast. Then, in Section 6 we study the authentication case. Finally in Section 7, we discuss some related work and perspectives.

2 Model and definitions

Identifiers and homonyms. We consider a distributed message-passing system of n processes. Each process gets an unique identifier from a set of identifiers $\mathcal{L} = \{1, 2, \ldots, l\}$. We assume that each identifier is assigned to at least one process but some processes may share the same identifier. Hence we have $l \leq n$. If p is a process then $Id(p)$ is the *identifier* of process p.[1] For an identifier id, the group of processes with identifier id, $G(id)$, is the set of all processes with identifier id.

For example, if $l = 1$ then the system is fully anonymous and if $n = l$ each process has an unique identifier.

Process failure. A *correct process* does not deviate from its algorithm specification. Some processes may be *Byzantine*, such a process can deviate arbitrarily from its algorithm specification. In particular, a Byzantine process may send different messages than its algorithm specifies or fails to send the messages it is supposed to. In the following t is an upper bound on the number of Byzantine processes.

From [13, 14], we know that Byzantine Agreement is impossible to solve if $n < 3t$, so we assume $n > 3t$.

Forgeable Identifiers. To each message m is associated the *origin group* of m that is an identifier in $\mathcal{L}$ denoted $from(m)$. When the sender of the message m is a correct process p, this identifier is the identifier of this process: $from(m) = Id(p)$. Remark that $from(m)$ enables only to know the origin group but does not enable to know which process in this group is the sender.

We assume that Byzantine processes have the power to forge some identifiers. The set of identifiers that can be forge by Byzantine processes is a subset of $\mathcal{L}$ and is denoted $\mathcal{F}$. In the following k designs an upper bound of the number of identifiers that can be forged: $|\mathcal{F}| = k$.

Let id_f be an identifier in $\mathcal{F}$, a Byzantine process with identifier id may send a message m to group id' with the forged identifier id_f. In this case, a process p with identifier id' receives the message m with $from(m) = id_f$. As a Byzantine process acts as an adversary it may divulge any information, then we assume here that if p is a Byzantine process then $Id(p)$ is also in $\mathcal{F}$.

Consequently, if a process p receives a message m with $from(m) = id$, p knows that this message has been either sent by a correct process with identifier id or sent by a Byzantine process which has forged the identifier id.

[1] For convenience, we sometimes refer to individual processes using names like p but these names cannot be used by processes in the algorithms.

We name (n, l, k, t)-homonym model such a model. In the following a *correct* group designs a group of processes with some identifier that contains only correct processes and whose its identifier is not forgeable.

Synchronous rounds. We consider a *synchronous model* of rounds. The computation proceeds in rounds. In each round, each process first sends a set of messages, which depends on its current state, to some identifiers. Then, each process receives all the messages sent to its identifier in the same round and finally changes its state according to the set of received messages.

As several processes may share the same identifier, in a round a process may receive several identical messages coming from processes with the same identifier (or Byzantine process that has forged this identifiers). But we assume here that when a process receives a message m with $from(m) = id$, it does not know how many correct processes with identifier id (or Byzantine process that has forged id) have sent this message.[2]

A Byzantine process can deviate arbitrarily from its algorithm specification and Byzantine process may send any set of messages (possibly an empty set) with any identifiers in $\mathcal{F}$. Moreover, contrary to correct processes, we assume that Byzantine processes are able to send different messages to different processes in the same group.

Byzantine Agreement. In the following we are interested in the Byzantine agreement problem [13, 14]. Recall that solutions to this problem are the basis of most of fault tolerant algorithms (e.g. [15]). *Byzantine Agreement* is an irrevocable decision problem that has to satisfy the following properties:

1. *Validity:* If all correct processes propose the same value v, then no value different from v can be decided by any correct process.
2. *Agreement:* No two correct processes decide different values.
3. *Termination:* Eventually every correct process decides some value.

3 Impossibility result

Following the spirit of the impossibility of Byzantine Agreement in [8], we prove our impossibility results in (n, l, k, t)-homonym model.

Proposition 1. *Byzantine Agreement is unsolvable in (n, l, k, t)-homonym model if $l \leq 2t + k$.*

Proof. It suffices to prove there is no synchronous algorithm for Byzantine Agreement when $l = 2t + k$. To derive a contradiction, suppose there is an algorithm $\mathcal{A}$ for Byzantine agreement with $l = 2t + k$. Let $\mathcal{A}_i(v)$ be the algorithm executed by a process with identifier i when it has input value v.

We divide the set of processes into 4 subsets: $A = \cup_{0<i\leq t} G(i)$, $B = \cup_{t<i\leq 2t} G(i)$, $C = \cup_{2t<i\leq 3t} G(i)$ and $F = \cup_{3t<i\leq k+2t} G(i)$.

[2] Our results can be extended to the model of *numerate* processes as defined in [6] for which each process receives in a round a multiset of messages and is able to count the number of copies of identical messages they receive in the round.

We consider a system of 8 sets: two sets A_0 and A_1 (resp B_0 and B_1, C_0 and C_1, F_0 and F_1) with the same number of processes and the same repartition in group as A (resp. B, C, F). Each process of A_0 with identifier i executes the code $\mathcal{A}_i(0)$ and the analog for others sets and other input values.

Imagine setting up a system S as shown in Figure 1. Every process correctly executes the algorithm assigned to it. Communication between groups are indicated by continuous line. A dash arrow from group X to group Y indicates that processes of X send its message also to Y. (We do not pretend that we have a Byzantine Agreement algorithm for this system.)

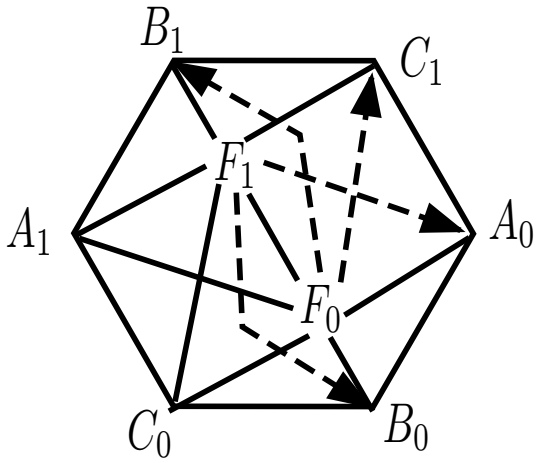

Fig. 1. communication from a process registered on serverl_2 to processes on server l_3.

We now define three executions of the algorithm $\mathcal{A}$.

In execution α, processes of A, B and F are correct and have input 1, all the processes in C are Byzantine and the identifiers in F may be forged. By validity, all the correct processes must decide 1. A process c in C sends (1) the same messages to processes in A and F as the corresponding process in C_0 in system S, (2) the same messages to processes in B and F as the corresponding process in C_1 in system S, Moreover one process in C sends the same messages as processes in F_0 to processes in A, B and F.

In execution β, processes of B, C and F are correct and have input 0, all the processes in A are Byzantine and the identifiers in F may be forged. By validity, all the correct processes must decide 0. A process a in A sends (1) the same messages to processes in C and F as the corresponding process in A_1 in system S, (2) the same messages to processes in B and F as the corresponding process in A_0 in system S, Moreover one process in A sends the same messages as processes in F_1 to processes in A, B and F.

In execution γ, processes of A, C and F are correct. Processes of A and F have input 1 and processes of C have input 0. All the processes in B are Byzantine and the identifiers in F may be forged. Processes in B send (1) the same messages to processes in A and F as the

corresponding processes in B_1 in system S (and the same that in α) (2) the same messages to processes in C as the corresponding processes in B_0 in system S (and the same that in γ)

Moreover one process in B sends the same messages as processes in F_0 to processes in A, C and F. Note that processes in F and this Byzantine process sends the same message to processes in A as in run α and to processes in C as in run γ.

So, for the correct processes in A executions α and γ are undistinguishable and they decide 1. For the correct processes in C executions β and γ are undistinguishable and they decide 0. This decision in execution γ contradicts the agreement property of the Byzantine agreement.

4 Authenticated Broadcast

Our algorithms for Byzantine Agreement use an adaptation of Authenticated Broadcast as introduced by Srikanth and Toueg [16] in the classical case where each process has a different identifier ($n = l$).

First, recall some principles for Authenticated Broadcast. Authenticated Broadcast in [16] is defined by two primitives: $Broadcast(p, m, r)$ and $Accept(p, m, r)$. Computation is broken up in *superrounds*. Roughly speaking a process p broadcasts a message m in superround r by $Broadcast(p, m, r)$. If the process p is correct all processes receive the message and $Accept(p, m, r)$. If the process p is a Byzantine process, the Authenticated Broadcast guarantees that if some process $Accept(p, m, r)$ at some superround r' then all correct processes $Accept(p, m, r)$ no later than in superround $r'+1$. Furthermore no correct process can $Accept(p, m, r)$ from a correct process p if p has not broadcast it.

Considering the (n, l, k, t)-homonym model, we decompose the synchronous computation in superrounds too. All the processes of a group have to invoke broadcast of a message m in the superround r in order to ensure that this message will be accepted in the following superround.

More precisely, our Authenticated Broadcast is defined by two primitives: $Broadcast(i, m, r)$ and $Accept(i, m, r)$ where i is the identifier of some group. We assume that a process broadcasts at most one message in a superround. The Authenticated Broadcast primitive is specified as follows:

1. *Correctness:* If all the processes in a correct group i perform $Broadcast(i, m, r)$ in superround r then every correct process performs $Accept(i, m, r)$ during superround r.
2. *Relay:* If a correct process performs $Accept(i, m, r)$ during superround $r' \geq r$ then every correct process performs $Accept(i, m, r)$ by superround $r' + 1$.
3. *Unforgeability:* If some correct process performs $Accepts(i, m, r)$ in superround $r' \geq r$ then all correct processes in group i must $Broadcast(i, m, r)$ in superround r.

The algorithm is described in Figure 2. A superround r is composed of the two rounds $2r$ and $2r + 1$.

First, recall the principles of the algorithm of [16] with T Byzantine processes. To propose a value v in superround r, process p sends message $(init, p, v, r)$ to all processes (including itself). A process receiving such a message becomes a "witness" for (p, v, r) and sends a message of type *echo* to all processes. Any process that has $T + 1$ witnesses for (p, v, r) becomes

Code for process p with identifier $i \in \{1, ..., l\}$

Variable:

1 $\mathcal{M} = \emptyset$;

Main code:

2 IN ROUND R

3 **if** $R = 2r$ **then** if $Broadcast(i, m, r)$ **to perform**

4 **then send** $(\mathcal{M} \cup (init, i, m, r), R)$ to all

5 **else send** $(\mathcal{M} \cup (noinit, i, \perp, r), R)$ to all

6 **else send** $(\mathcal{M}, R)$ to all;

on receipt of the messages of the round

7 For all $h \in \{1, ..., l\}$

8 Let $\mathcal{M}[h, R]$ be the set of messages with form $(init, h, *, r)$ received from processes in group h

9 **if** $(R = 2r)$ and $M[h, R] = \{(init, h, m, r)\}$

10 **then** $\mathcal{M} = \mathcal{M} \cup (echo, h, m, r)$

11 For all $r \in \{1, ..., R/2\}$

12 For all $m \in$ possible messages

13 **if** $(echo, h, m, r)$ received from at least $l - 2t$ distinct groups

14 **then** $\mathcal{M} = \mathcal{M} \cup (echo, h, m, r)$

15 **if** $(echo, h, m, r)$ received from at least $l - t$ distinct groups

16 **then** $Accept(h, m, r)$

Fig. 2. Authenticated Broadcast algorithm in the (n, l, k, t)-homonym model.

itself witness (because at least one correct process has sent this message). When a process receives more than $(2T + 1)$ witnesses, it accepts (p, v, r) ($T + 1$ correct processes have sent this message then all correct processes will find $T + 1$ witnesses of this message).

The algorithm follows the same principles, to propose a value v in supperround r process p with identifier i sends message $(init, i, v, r)$ to all processes (including itself) (line 4). A process receiving such a message from some processes with identifier i becomes "witness" for (i, v, r) and sends a message of type *echo* to all processes (line 10). Any process having $l - 2t$ witnesses for (i, v, r) becomes itself witness (if $l - 2t > k$ at least one process in a correct group has sent this message) (line 13). When a process receives more than $(l - t)$ witnesses, it accepts (i, v, r) (at least $t + 1$ processes from correct groups have sent this message) (lines 15 to 16). In this way we ensure *correctness* and *relay* properties

To ensure the unforgeability property, a correct process that has no message to broadcast in a superround broadcast a $noinit$ message in the corresponding even round. In this way, if some correct process with identifier i has no message to broadcast in superround r, every correct process gets $M[i, 2r]$ (line 9) different from one message $init$ and wil not become witness of any message $(i, *, r)$.

The formal proof is standard and will be omitted due to lack of space. We have:

Proposition 2. *If $l > 2t + k$, the algorithm Figure 2 implements Authenticated Broadcast in (n, l, k, t)-homonym model.*

5 Byzantine Agreement Algorithms

Code for process p with identifier i

Variable:

1 $input = \{v\}$; /* $v \in \{0, 1\}$ is the value proposed value */
2 $value = 0$;
3 $state = false$;

Main code:

4 SUPERROUND 1
5 **if** $input = 1$ **then** $Broadcast(i, 1, 1)$
6 Let $A_1 = \{h : p \text{ has } Accepted(h, 1, 1)\}$
7 **if** $|A_1| \geq t + 1$ **then**
8 $state = true$

9 SUPERROUND r **from** 2 **to** $2k + 2$
10 **if** $state = true$ **then**
11 $Broadcast(i, 1, r)$
12 $state = false$;

13 Let $A_r = \{h : p \text{ has } Accepted(h, 1, 1)\}$
14 **if** $|A_r| \geq t + 1$
and has $Accepted(i_u, 1, r_u)$ from $\frac{r+1}{2}$ distinct identifiers i_u with $r_u \geq 2$
15 **then** $value = 1$

16 **if** $|A_r| \geq t + 1$
and has $Accepted(i_u, 1, r_u)$ from $\frac{r}{2}$ distinct processes i_u not including i with $r_u \geq 2$
17 **then** $state = true$;

18 AT THE END OF SUPERROUND $2k + 2$
19 **if** $value = 1$ **then**
20 DECIDE 1
21 **else**
22 DECIDE 0;

Fig. 3. Synchronous Byzantine Agreement algorithm with at most k forgeable identifiers and at most t Byzantine processes

Our algorithm follows the line of the algorithm in [16] defined in the classical case where each process has a different identifier ($n = l$). One of the main difference here is the fact that the behaviour of groups of processes is different from the behaviour of processes. In particular, correct processes in groups with forgeable identifiers or in groups containing Byzantine processes have to decide and have to participate to the decision.

The algorithm proceeds in synchronous superrounds (set of successive rounds). In this algorithm, 1 is the only value that may be broadcast and value 0 is decided upon by default if 1 is not decided. Hence, all processes in superround 1 broadcast 1 if their input is 1. A process p sets variable *value* to 1 (and then will decides 1) in superround $r < 2k$, if it has (1) accepted $(i_u, 1, 1)$ messages from $t + 1$ distinct identifiers i_u, and (2) accepted $(j_u, 1, r_u)$ from $(r + 1)/2$

with $r_u \geq 2$. Condition (1) ensures that at least one correct group has broadcast and condition (2) corresponds to the one of [16]. Condition (1) ensures the *validity* property of consensus and the condition (2) ensures the *agreement* property. The *correctness* and *relay* properties of Authenticated Broadcast ensure that by the next superround (superround $r+1$), all messages accepted by process p are also accepted by all correct processes. Hence, they all set the variable *state* to *true*. By superround $r+2$, they broadcast and by *correctness* and *relay* properties, all correct processes decide 1 by setting *value* to 1. If p decides in superround $2k+2$, then it can be proved that at least one group of correct processes set its variable *value* to 1 by superround $2k$, and all correct processes decide by superround $2k+2$.

We now prove that the algorithm of Figure 3 satisfies the specification of Byzantine agreement. First we give without proofs the easy following lemmata:

Lemma 1. *If, at superround r, a correct process p has $|A_r| \geq t+1$ then at each superround r' with $r+1 \leq r' \leq 2k+2$ each correct process has $|A_{r'}| \geq t+1$.*

Lemma 2. *If every process of a correct group with identifier i_0 broadcasts some message in some superround $r > 1$, then every correct process sets $value$ to 1 by superround r.*

Proposition 3. *(Validity) If all correct processes propose the same initial value v then no value different from v can be decided by any correct process.*

Proof. Assume all correct processes propose 1.

In superround 1, every process of correct group with identifier i broadcasts $(i,1,1)$ message. The correctness of the authenticated broadcast ensures that every correct process will accept $(i,1,1)$ message. As there are at least $l-k > t$, it is easy to verify that $|A_1|$ of every correct process is greater than t. Thus every correct process sets $state$ to $true$ in superround 1 Line 8. In superround 2, every correct process p broadcasts $(Id(p),1,2)$. As $l > t+k$ there is at least one correct group i in which every process broadcast $(i,1,2)$. By Lemma 2, every correct process sets value to 1 in Line 15 and will decide 1.

Assume now that all the correct process propose 0. We prove that no correct process sets $value$ to 1. For contradiction, assume that some correct process p sets $value$ to 1 in some superround r. Thus, $|A_r|$ of p must be greater than t. Since there are at most t Byzantine processes, there is at least one correct group, say i, in A_r. By *unforgeability* property, all processes of this group broadcast $(i,1,1)$ in superround 1 and then the input value of every process in this group must be 1, a contradiction. Thus, every correct process keeps $value$ to 0 and decides 0 Line 22.

Proposition 4. *(Termination) Eventually every correct process decides some value.*

Proof. A correct process decides at superround $2k+2$ Line 20 or Line 22.

Assume that, in the execution, some correct process sets $value$ to 1 at Line 15, and decides 1. Let r_1 be the first superround in which some correct process sets $value$ to 1. Let p_1 be such a correct process that set $value$ to 1.

With the help of Lemma 2, we prove:

Lemma 3. *If $r_1 \leq 2k$, then every correct process sets $value$ to 1 by superround r_1+2.*

When r_1 is greater than $2k$, p_1 has accepted $(j_u, 1, r_u)$ from $\frac{2k+2}{2}$ distinct processes j_u, then from at least $k+1$ distinct processes. Then there is at least one correct group, say i_0, in this set. Then we get:

Lemma 4. *If $r_1 > 2k$, then every correct process sets $value$ to 1 by superround r_1.*

Proposition 5. *(Agreement) No two correct processes decide different values.*

Proof. Assume that some correct process sets $value$ to 1 at Line 15 and decides 1. Let r_1 be the first superround where some correct process sets $value$ to 1.

Lemma 4 shows that if $2k + 2 \geq r_1 > 2k$ then every correct process sets its $value$ variable to 1 by superround r_1. Lemma 3 shows that if $r_1 \leq 2k$ then every correct process sets its $value$ variable to 1 by superround $r_1 + 2$. Thus, all correct processes decide 1.

Otherwise, if no correct process sets $value$ to 1 then $value$ is 0 for all correct processes and they decide 0.

We have:

Theorem 1. *Assuming if $l > t + k$, algorithm of Figure 3 implements Byzantine Agreement using Authenticated Broadcast in (n, l, k, t) homonym model.*

Combining this with algorithms for Authenticated Broadcast we get:

Corollary 1. *If $l > 2t + k$, the algorithms of Figure 3 and 2 implement Byzantine Agreement in (n, l, k, t) homonym model.*

6 Authentication

Authentication [14] ensures that Byzantine process may "lie" about its own values but may not relay altered values without betraying itself as faulty. Currently this property may be ensured with a system of signatures using public keys cryptography.

The implementation of homonyms with authentication is rather natural: the members of a group share a secret key. The origin group of a message may be verified by a signature scheme of this group. Messages m are authenticated by the identifier of the sender. Each process can verify if a message carries the signature of a given identifier. As before we assume that the signatures of at most k identifiers can be forged.

With homonyms and this scheme of authentication, if id is not forgeable (any process with this identifier is correct) then it is not possible for any process to wrongly pretend that it has received some messages coming from identifier id, hence we get the authentication property of [14].

For this model with authentication, we improve our bound: $l > t + k$ is necessary and sufficient to achieve Byzantine Agreement. (Recall that in the classical model in which each process has its own identifier and without forgeable identifiers the bound is $n > 2t$.)

The proofs of the lower bound is essentially the same as for the case without authentication in Section 3.

The Byzantine Agreement algorithm of Figure 3 directly works with $l > t + k$ if we have an Authenticated Broadcast. It remains to get an Authenticated Broadcast with $l > k + t$.

In [16], in the classical case in which each process has its own identity, Authenticated Broadcast can be obtained simply with authentication: it suffices to verify the signatures: A process that receives a message (p, m, r) accepts it if it can verify p's signature (and then forwards this message). But if we apply this simple mechanism in our model, we do not get the unforgeability property. In a forgeable group with identifier i containing some correct processes, it is possible that some correct accepts (i, m, r). Indeed, this message has been sent by a Byzantine process that has forged the identifier i.

To implement Authenticated Broadcast, we use the signatures and the mechanism of witnesses as in our previous algorithm. The implementation is based on the one presented in Section 4 ans some easy changes.

At some point, in algorithm 2 when a process receives some messages in round R, it will send $echo$ in the next rounds. The process will forward all messages that produced this $echo$. In this way it gives a proof that it has the right to send $echo$. We get an Authenticated Broadcast algorithm from algorithm Figure2 by: (1) removing from the received message all messages with a bad signature, (2) forwarding the proof of each new $echo$ message, (3) replacing the line 13 of algorithm 2 by:

if received $(echo, h, m, r)$ and the proof of $(echo, h, m, r)$

and, (4) replacing the line 15 of algorithm 2 by

if received $(echo, h, m, r)$ and the proof of $(echo, h, m, r)$ from at least $l - t$ distinct groups

We have:

Theorem 2. *With authentication Byzantine Agreement is solvable in (n, l, k, t)-homonym model if and only if $l > t + k$,*

7 Related works and perspectives

When processes share identifiers and some of theses processes may be Byzantine, it is rather natural that some of these identifiers may be forged. Hence this work is a natural extension of [6] to forgeable identifiers.

At least for the authentication case, groups signature as introduced first in [5] are close to our model. Groups signature enables to sign messages on behalf of a group and clearly can be used to implement the model of homonyms. Note that group signatures generally ensures other properties than the one we consider here. Groups signatures may be a valuable way to implement models with homonyms.

In other works [2, 10, 12], a mixed adversary model is considered in the classical ($l = n$): the adversary can corrupt processes actively (corresponding to Byzantine process) and can forge the signature of some processes.

In some way, we combine here the idea of group signatures and forgeable signatures but contrary to group signatures the goal is not to develop protocol ensuring strong properties

like anonymity or unforgeability but try to develop algorithms (like agreement) in presence of groups of processes with Byzantine processes and forgeable identifiers.

Here we proved that with forgeable identifiers and homonyms Byzantine Agreement can be solved in a "reasonable" way (and without any assumption about cryptographic system). Interestingly, the solvability of Byzantine Agreement depends only on the number of identifiers and the number of forgeable identifiers. Hence adding correct processes does not help to solve Byzantine Agreement.

A natural extension of this work could be to consider partially synchronous models.

As Byzantine Agreement is the basis for replication systems in the classical models in which each process has its own identity, a natural question is to know if it is still the case and envisage to develop algorithms for more difficult problems.

References

1. Hagit Attiya, Alla Gorbach, and Shlomo Moran. Computing in totally anonymous asynchronous shared memory systems. *Information and Computation*, 173(2):162–183, 2002.
2. Piyush Bansal, Prasant Gopal, Anuj Gupta, Kannan Srinathan, and Pranav K. Vasishta. Byzantine agreement using partial authentication. In *DISC*, volume 6950 of *LNCS*, pages 389–403, 2011.
3. Paolo Boldi and Sebastiano Vigna. An effective characterization of computability in anonymous networks. In *DISC*, volume 2180 of *LNCS*, pages 33–47. Springer, 2001.
4. Harry Buhrman, Alessandro Panconesi, Riccardo Silvestri, and Paul M. B. Vitányi. On the importance of having an identity or, is consensus really universal? *Distributed Computing*, 18(3):167–176, February 2006.
5. David Chaum and Eugène van Heyst. Group signatures. In *EUROCRYPT*, volume 547 of *LNCS*, pages 257–265, 1991.
6. Carole Delporte-Gallet, Hugues Fauconnier, Rachid Guerraoui, Anne-Marie Kermarrec, Eric Ruppert, and Hung Tran-The. Byzantine agreement with homonyms. In *PODC*, pages 21–30. ACM, 2011.
7. Carole Delporte-Gallet, Hugues Fauconnier, and Hung Tran-The. Byzantine agreement with homonyms in synchronous systems. In *ICDCN*, volume 7129 of *LNCS*, pages 76–90, 2012.
8. Michael J. Fischer, Nancy A. Lynch, and Michael Merritt. Easy impossibility proofs for distributed consensus problems. *Distributed Computing*, 1(1):26–39, January 1986.
9. Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.
10. S. Dov Gordon, Jonathan Katz, Ranjit Kumaresan, and Arkady Yerukhimovich. Authenticated broadcast with a partially compromised public-key infrastructure. In *SSS*, volume 6366 of *LNCS*, pages 144–158, 2010.
11. Rachid Guerraoui and Eric Ruppert. Anonymous and fault-tolerant shared-memory computing. *Distributed Computing*, 20(3):165–177, October 2007.
12. Anuj Gupta, Prasant Gopal, Piyush Bansal, and Kannan Srinathan. Authenticated byzantine generals in dual failure model. In *ICDCN*, volume 5935 of *LNCS*, pages 79–91, 2010.
13. Leslie Lamport, Robert Shostak, and Marshall Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, July 1982.
14. Marshall Pease, Robert Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *Journal of the ACM*, 27(2):228–234, April 1980.
15. Fred B. Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Computing Surveys*, 22(4):299–319, December 1990.
16. T. K. Srikanth and Sam Toueg. Simulating authenticated broadcasts to derive simple fault-tolerant algorithms. *Distributed Computing*, 2(2):80–94, 1987.