Verifying Randomized Byzantine Agreement[⋆]

Marta Kwiatkowska and Gethin Norman[⋆⋆]

School of Computer Science, University of Birmingham,
Birmingham B15 2TT, United Kingdom
{M.Z.Kwiatkowska,G.Norman}@cs.bham.ac.uk

Abstract. Distributed systems increasingly rely on fault-tolerant and secure authorization services. An essential primitive used to implement such services is the Byzantine agreement protocol for achieving agreement among n parties even if t parties ($t < n/3$) are corrupt and behave maliciously. We describe our experience verifying the randomized protocol ABBA (Asynchronous Binary Byzantine Agreement) of Cachin, Kursawe and Shoup [5], a practical protocol that incorporates modern threshold-cryptographic techniques and forms a core of powerful asynchronous broadcast protocols [4]. The protocol is efficient (runs in constant expected time), optimal (it tolerates the maximum number of corrupted parties) and provably secure (in the random oracle model). We model the protocol in Cadence SMV, replacing the coin tosses with nondeterministic choice, and provide a proof of the protocol correctness for all n under the assumption that the cryptographic primitives are correct. The proof is fully automated except for one high-level inductive argument involving probabilistic reasoning. We validate probabilistic reasoning through deriving abstractions for finite configurations (for n up to 20) and model checking those with the probabilistic model checker PRISM.

Keywords: Induction and compositional model checking, probabilistic model checking, randomized distributed algorithms.

1 Introduction

Motivation. Distributed systems, for example the Internet, rely on trusted services such as secure directory and authorization services for protection against malicious adversaries that can corrupt servers and control the network. Centralised trusted services are widely deployed, but they introduce a single point of failure which has performance implications. One possible solution, proposed by Cachin [3], is to *distribute* the trusted service among a number of servers and use replication algorithms for masking the faulty ones. The main difficulty with this approach is that existing replication methods can only tolerate benign faults, such as stopping and omission failures, whereas distributed trusted services must be able to handle arbitrary (Byzantine) failures. Recently, Cachin *et*

[⋆] Supported in part by EPSRC grants GR/M04617 and GR/N22960.
[⋆⋆] Contact author. Tel: +44 121 414-4789, fax: +44 121 414-4281.

al [4] proposed a class of reliable broadcast protocols suitable for such a setting. An essential primitive used to implement these protocols is Byzantine agreement, an algorithm for achieving agreement on a value among n parties, some of which may be corrupted or faulty, even when the corrupted parties behave maliciously. Since there is no deterministic algorithm for achieving agreement in the asynchronous setting even against benign failures [12], known Byzantine agreement algorithms either exploit additional primitives, for example authentication in the form of unforgeable signatures [7], or use randomization [25].

The prerequisite component of the broadcast protocols presented in [4] is the randomized protocol called ABBA (Asynchronous Binary Byzantine Agreement) of Cachin, Kursawe and Shoup [5] which combines the use of cryptography and randomization in the form of threshold signatures and coin-tossing schemes. The idea of a threshold scheme [29] is to allow parties to generate *shares* of the secret value. The value is known only after a certain threshold number of shares have been produced, and moreover it is computationally infeasible to obtain the secret given fewer shares. Thus, the cryptographic primitives prevent malicious (corrupted) parties influencing the remaining (honest) parties by sending forged messages or correctly predicting the values of the coins before they are tossed. The ABBA protocol is of practical relevance (in fact it has already been implemented in Java [6] as part of an architecture for asynchronous networks) due to the following factors: it guarantees safety and liveness in a general asynchronous model with no timing assumptions, making it very appropriate for e.g. the Internet; it has a theoretical complexity of constant expected time; and the cryptographic primitives are relatively efficient compared to the latencies of the Internet. Furthermore, it is provably secure in the random oracle model. Also, in contrast to the non-randomized approaches of [7,8], it guarantees both safety and liveness without additional timing assumptions.

Aims. Our aim is to automate the analysis of the ABBA protocol using the methodology introduced in our earlier paper [17] and based on [22]. In [17] we used Cadence SMV and the probabilistic model checker PRISM [16] to verify the simpler randomized agreement protocol of Aspnes and Herlihy [2] that tolerates only benign stopping failures. We achieved this through a combination of mechanical inductive proofs (for all n for non-probabilistic properties) and checks (for finite configurations in case of probabilistic properties) plus one high-level manual proof. The ABBA protocol, however, presented us with a number of difficulties not encountered earlier:

- To verify the protocol for any number, n, of parties, we have to model their ids with the abstract data type `ordset` [22]. Unfortunately, the limitations of this type (restriction to comparison with constants only) make it impossible to identify an *arbitrary* static subset of t parties ($t < n/3$) that are corrupt. Likewise, *counting* a proportion of elements of such a type is not feasible.
- The efficiency of the protocol depends on a probabilistic property which states that the higher the round number (which is unbounded) that an honest party has advanced to, the lower the probability of this occurring. Thus, the

model is infinite, and currently very little is known about methods for the abstraction and verification of infinite state probabilistic systems; for papers on these topics see e.g. [1,9,18].

– The ABBA protocol is based on the random oracle model, an idealisation of the collision-free hash functions used in real cryptographic settings. The random oracle model is necessary in order to prove security of the scheme, by placing certain statistical assumptions that prevent any adversary with realistic computing power 'guessing' the secret.

Contribution. We overcome the above challenges as follows. We model the full protocol in Cadence SMV, having replaced the random outcomes with nondeterministic choices. The mentioned technical difficulties with the `ordset` data type were resolved largely by searching for a variant of the model which preserves the key property that the correctness argument relies on. Since this is a randomized agreement protocol, a correctness argument must demonstrate **Validity**, **Agreement** and **Probabilistic Termination**. The proof of **Probabilistic Termination** reduces to a straightforward, high-level inductive argument based on a number of lemmas and cryptographic assumptions. We assume the cryptographic properties and automate the proof of each lemma. Together with the proofs of Validity and Agreement, which are more straightforward and fully automated, we obtain a partially mechanised argument for the correctness of the ABBA protocol *for all* n and *for all* rounds.

It should be emphasised that we cannot automate the final inductive argument as it is probabilistic: Cadence SMV cannot handle probabilities, whereas PRISM currently only handles finite configurations and does not support data reduction. Instead we additionally validate the probabilistic analysis as follows. By observing that, for a fixed n, the problem can be reduced to model checking a finite state abstraction of the protocol, we manually construct an abstraction and model check it using PRISM, validating the probabilities for up to $n = 20$ parties. Additionally, we verify (for a finite configuration) the correctness of the abstraction using the process algebra CSP [26] and the FDR tool [11] based on the method in [19]; this depends on the ability to code probabilities in the action names, and therefore precludes the use of Cadence SMV.

This paper describes our experience with verifying the ABBA protocol, focusing on the Cadence SMV proof. Full details of all models we have constructed and their model checking can be found at the PRISM web page [24] and in [15].

2 The Protocol

Requirements for Agreement: *Agreement* problems arise in many distributed domains, for example, when it is necessary to agree whether to commit or abort a transaction in a distributed database. A *distributed agreement protocol* is an algorithm for ensuring that a collection of distributed parties, which start with some initial value (0 or 1) supplied by an environment, eventually terminate agreeing on the same value. A *randomized* protocol uses random assignment, for example electronic coin tossing, and its termination is therefore probabilistic. The requirements for a randomized agreement protocol are:

Validity: If all parties have the same initial value, then any party that decides must decide on this value.

Agreement: Any two parties that decide must decide on the same value.

Probabilistic Termination: Under the assumption that all messages between non-corrupted parties eventually get delivered, *with probability 1*, all initialized and non-corrupted parties eventually decide.

We consider the Byzantine agreement protocol of Cachin, Kursawe and Shoup [5] which is set in a completely asynchronous environment, allows the maximum number of corrupted parties and makes use of cryptography and randomization. There are n parties, an adversary which is allowed to corrupt at most t of them (where $t < n/3$), and a trusted dealer. The parties proceed through possibly unboundedly many rounds: in each round, they attempt to agree by casting votes based on the votes of other parties. In addition to **Validity** and **Agreement**, the protocol guarantees **Probabilistic Termination** in a constant expected number of rounds which is validated through the following property:

Fast Convergence The probability that an honest party advances by more than $2r + 1$ rounds is bounded above by $2^{-r} + \varepsilon$ where ε is a negligible function in the security parameter.

The Model and Cryptographic Primitives: The setting is a *static* corruption model: the adversary must decide whom to corrupt at the very beginning of the execution of the protocol. Once the adversary has decided on the corrupted parties these are then simply absorbed into the adversary. The *dynamic* corruption model, where the adversary can adaptively choose who to corrupt based on the information it has accumulated, is not considered, since it would not be possible to obtain practical (provably secure) implementations of the cryptographic primitives used in the protocol in this model.

The adversary also has complete control over the network: it can schedule and deliver the messages that it desires. The honest parties can therefore be considered as passive: they just respond to requests made by the adversary and do not change state in between such requests. Thus, the adversary can delay messages for an arbitrary length of time, except that it must eventually send each message.

One of the cryptographic primitives the protocol uses is a *threshold random-access coin-tossing scheme*. This models an unpredictable function F, of which each party holds a *share*, that maps the name of a coin for each round r to its value $F(r) \in \{0, 1\}$. Each party can generate a share of each coin, where $n - t$ shares are both necessary and sufficient to construct the value of a particular coin. The remaining cryptographic primitives of the protocol are non-interactive threshold signature schemes used to prevent the adversary from forging messages. The idea is that parties hold *shares* of the secret key of a signature scheme, and may generate shares of signatures on individual messages. Again, the meaning of the threshold is that a proportion, in this case $n - t$, of signature shares are necessary and sufficient to construct a signature.

It should be noted that threshold cryptography or some form of secret sharing is essential to ensure both fault-tolerant and secure systems, see e.g. [7,8]. Randomization can be avoided, but at the cost of stronger model assumptions.

In [5,29] it has been shown that the non-interactive threshold signature scheme and the random-access coin tossing scheme have efficient implementations and are proved secure in the random oracle model, in the sense of being unforgeable and robust (see Section 3) on the assumption that the RSA and DH problems are hard. The random oracle model is an idealised model, in which the hash functions have been replaced with a truly random function. The parties are computationally bounded by a polynomial in k (a security parameter, typically the length of string). This allows one to state the infeasibility of 'guessing' the value of a coin or a signature[1]. For this case study we consider a version of the protocol which assumes the correctness of the cryptographic elements. In particular, this means that we can remove the possibility of the adversary modifying or inserting fake messages.

The Protocol: The protocol is given in Figure 1. Each party's initial value is sent to it by a trusted dealer. The parties proceed in rounds, casting pre-votes and main-votes. A party constructs both the *signature share* and *justification* for each pre-vote and main-vote it casts using a threshold signature scheme. The justification for each vote is the *signature* obtained by combining the signature shares of the messages that the party used as the basis for this vote. For example, if a party casts a main-vote for 1 in round r, then the corresponding justification is the signature obtained through combining the signature shares present in the $n-t$ messages which contain pre-votes for 1 in round r that the party must have received. For further details on these justifications see [5].

Observe that the power of the adversary is limited by the requirement that all votes carry a signature share and a justification, and the assumption that the threshold signature scheme is secure (the adversary cannot forge either signature shares or signatures). The presence of the signature shares and this assumption implies that the adversary cannot forge any messages of the honest parties, that is, cannot send a message in which it pretends to be one of the honest parties. The adversary can make one honest party believe that the initial vote of a corrupted party is 0, while another honest party believe it is 1, since these messages do not require justification. However, since all the remaining votes need justification, the adversary cannot just make up the pre-votes and main-votes of the corrupted parties. For example, if in round r there are at least $n - t$ pre-votes for 0 and between 1 and $n - t - 1$ pre-votes for 1 (all of which carry proper justification), then there is justification in round r for both a main-vote for 0 and for `abstain`, but not for 1. Thus, the adversary can make one honest party believe a corrupted

[1] Schemes proved secure in the random oracle model retain their properties when instantiated with concrete hash functions, although a provably negligible probability of failure remains.

party has a main-vote for 0 in round r, while another honest party believe that the same corrupted party has a main-vote for `abstain`.

2.1 Modelling the ABBA protocol in Cadence SMV

The Cadence SMV model of the ABBA protocol closely follows that given in Figure 1, although some modifications were necessary to the original which we outline below. For the full Cadence SMV code see [24]. We use the data type `ordset` [22] to model the round numbers of the protocol, which are *unbounded*, and prove the correctness of the protocol for *any* number of parties, or, in other words, for *all* values of n and t such that $n > 3t$. Formally, we define the `ordset` data types $\mathit{ROUNDS}=\{0, 1, \dots\}$ for round numbers and $\mathit{PROC}=\{1, \dots, n - t\}$ for ids of honest parties.

Probabilistic choices: Since Cadence SMV cannot model probabilistic behaviour, we have replaced the outcome of the coin tossing protocol by a nondeterministic choice. The threshold coin scheme is modelled by requiring that the coin in round r is not revealed until at least $n - 2t$ honest parties have read the main-votes they require in round r.

Reading of votes: Since we use the `ordset` data type to model the set of honest parties, the only constants with which we can compare variables of this type are 1 and $n-t$. Thus, *counting* votes, i.e. expressing conditions "i has read the votes of $j_1, \dots, j_k$" for some $k \leq n - t$, becomes infeasible. To overcome this, instead of modelling a party reading individual votes, we store the *total* number of votes for each choice, and base future voting decisions on these totals. The protocol is not affected as the votes remain unchanged once they have been made. Since corrupted parties can send different votes to different parties, we store the totals of the honest parties and the corrupted parties separately; in fact, we store only boolean variables denoting what votes corrupted parties can cast and we have shown that it is sufficient to only count up to $n-2t = (n-t)-t$ honest party votes (by means of an `ordset` data type $\mathit{VOTES}=\{0, \dots, n-2t\}$), with the remaining t votes coming from corrupted parties.

A further necessary change concerns when the pre-votes are cast: a party now casts its pre-vote for round $r + 1$ when it has finished reading the main-votes of the *previous* round (step 3.) and not at the start of round $r + 1$ (step 1.). To see that this modification is sound observe that, since once these votes have been read the pre-vote is determined, and the adversary decides which votes a party reads, the adversary *already* knows what pre-vote will be cast at this earlier time. Therefore, casting pre-votes at this earlier point does not affect the power of the adversary. Note that, since the coin for round r might not be revealed when a pre-vote is now cast, we require parties to cast two pre-votes: one supposing the coin equals 0 and the other supposing the coin equals 1. The motivation behind this change is that the proof of **Fast Convergence** relies on knowing, even when the corresponding coin has not yet been revealed, what the

Protocol ABBA for party i with initial value v_i.

0. PRE-PROCESSING. Generate a signature share on the message

$$(\texttt{pre-process}, v_i)$$

and send all parties a message of the form

$$(\texttt{pre-process}, v_i, \textit{signature share}).$$

Collect $2t + 1$ pre-processing messages.

Repeat the following steps 1-4 for rounds $r = 1, 2, \ldots$

1. PRE-VOTE. If $r = 1$, let v be the majority of the received pre-processing votes. Else, select $n - t$ justified main-votes from round $r - 1$ and let:

$$v = \begin{cases} 0 & \text{if there is a main-vote for } 0 \\ 1 & \text{if there is a main-vote for } 1 \\ F(r-1) & \text{if all main-votes are } \texttt{abstain}. \end{cases}$$

Produce a signature share on the message $(\texttt{pre-vote}, r, v)$ and the corresponding justification, then send all parties a message of the form

$$(\texttt{pre-vote}, r, v, \textit{justification}, \textit{signature share}).$$

2. MAIN-VOTE. Collect $n - t$ properly justified round r pre-vote messages, and let

$$v = \begin{cases} 0 & \text{if there are } n-t \text{ pre-votes for } 0 \\ 1 & \text{if there are } n-t \text{ pre-votes for } 1 \\ \texttt{abstain} & \text{if there are pre-votes for 0 and 1.} \end{cases}$$

Produce a signature share on the message $(\texttt{main-vote}, r, v)$ and the corresponding justification, then send all parties a message of the form

$$(\texttt{main-vote}, r, v, \textit{justification}, \textit{signature share}).$$

3. CHECK FOR DECISION. Collect $n - t$ justified main-votes of round r. If all these are main-votes for $v \in \{0, 1\}$, then decide on v, and continue for one more round. Otherwise simply proceed.

4. COIN. Generate a *coin share* of the coin for round r and send all parties a message of the form

$$(r, \textit{coin share}).$$

Collect $n - t$ shares of the coin for round r and combine to get the value of $F(r) \in \{0, 1\}$.

Fig. 1. Asynchronous binary Byzantine agreement protocol ABBA

pre-votes of the honest parties that have read the main-votes for the previous round will be.

Therefore in the Cadence SMV code we define the following variables:

$\mathit{pre\_votes}$: `array` ROUNDS `of array` $0..1$ `of array` $0..1$ `of` VOTES;
$\mathit{main\_votes}$: `array` ROUNDS `of array` $\{0, 1, \mathit{abstain}\}$ `of` VOTES;
$\mathit{corrupted\_pre}$: `array` ROUNDS `of array` $0..1$ `of array` $0..1$ `of boolean`;
$\mathit{corrupted\_main}$: `array` ROUNDS `of array` $\{0, 1, \mathit{abstain}\}$ `of boolean`;

where $\mathit{pre\_votes}[r][c][v]$ equals the current number of honest pre-votes for v in round r supposing the coin for round $r-1$ equals c when revealed; $\mathit{main\_votes}[r][v]$ equals the number of honest main-votes for v in round r; $\mathit{corrupted\_pre}[r][c][v]$ is true if supposing the coin for round $r-1$ equals c, a corrupted party can cast a pre-vote for v in round r; and $\mathit{corrupted\_main}[r][v]$ is true if a corrupted party can cast a main-vote for v in round r.

3 Verification using Cadence SMV

Cadence SMV [21] is a proof assistant which allows the verification of large, complex, systems by reducing the verification problem to small subproblems that can be solved automatically by model checking. The techniques supported are *induction*, *circular compositional reasoning*, *temporal case splitting* and *data type reduction.*

Recall that, to verify the ABBA protocol correct, we need to establish the properties of **Validity**, **Agreement** and **Fast Convergence**. A number of assumptions were needed in order to perform the verification. These include the correctness of the cryptographic primitives; for example, we assume the following properties of the threshold coin-tossing scheme:

Robustness For any round r it is computationally infeasible for an adversary to produce $n - t$ valid shares of the coin for round r such that the output of the share combining algorithm is not $F(r)$.

Unpredictability An adversary's advantage in the following game is negligible. The adversary interacts with the honest parties and receives less than $n-2t$ shares of the coin for round r from honest parties, then at the end of the interaction outputs a bit $v \in \{0, 1\}$. The adversaries advantage is defined as the distance from $1/2$ of the probability that $F(r) = v$.

These assumptions are implicit in the Cadence SMV model, in that they restrict the power of the adversary. For example, the adversary cannot forge messages or make up any of the votes of the corrupted parties which require justification.

The remaining assumptions concern fairness statements which correspond to the fact that the adversary must eventually send all messages (which ensure that parties eventually cast votes), and technicalities which, due to the limitations of the `ordset` data type, we were unable to prove in Cadence SMV. In particular, since variables of different `ordset` data types cannot be compared, we cannot show that "if at least k honest parties cast their votes in a round then there are at least k votes", as distinct data types PROC and VOTES represent the set of honest parties and number of votes respectively. We give two illustrative examples of the assumptions made. For example, we assume[2] that in any round there cannot be $n-2t$ honest pre-votes for both 0 and 1:

```
forall (r in ROUNDS) for(c = 0; c ≤ 1; c = c + 1)
assumption1[r][c] : assert G ¬(pre_votes[r][c][0]=n − 2t ∧ pre_votes[r][c][1]=n − 2t);
```

We also assume the following fairness statements when proving **Fast Convergence** (see [24] for the manual proof).

Proposition 1. *For any party that enters round $r+1$:*

(a) *if the party does not decide by round r, then the coin for round r is tossed;*
(b) *if the party does not decide by round $r+1$, then the coin in round $r+1$ is tossed.*

3.1 Validity and Agreement

Both these arguments are independent of the actual probability values, and hence can be verified by conventional model checking methods. The proofs we obtain are fully automatic. The only assumptions used in the proofs are either similar to *assumption1* or concern fairness, see [24] for further details. Below we give a brief outline of the arguments based on two lemmas. The Cadence SMV proof of Lemma 1 is included for illustration purposes. Complete Cadence SMV proofs can be found at [24].

Lemma 1. *If in round r there are main-votes for v, then there are none for $\neg v$.*

Lemma 2. *If a party i decides on v in round r, then there are less than $n-2t$ main-votes for* `abstain` *in round r from honest parties.*

Validity: We prove that if all honest parties have the same initial preference, then all honest parties decide on this value in the initial round. Suppose all honest parties have the same initial value v, then in round 1 the pre-votes of all parties will be v, since all will see a majority of pre-processing votes for v (a majority of pre-processing votes requires at least $t+1$ votes, that is, at least one vote from an honest party). It then follows that all parties will have a main-vote for v in round 1, and hence all decide on v in the first round.

Agreement: We prove that if the first honest party to decide decides on v in round r, then all honest parties decide on v either in round r or round $r+1$. Therefore, suppose party i is the first honest party to decide and it decides on v in round r. Then i must have received an honest main-vote for v, and hence, by Lemma 1, there are no main-votes for $\neg v$ in round r. Therefore, any party that decides in round r must decide on v. Now, by Lemma 2, there are less than $n-2t$ honest main-votes for `abstain`, and since a party reads at least $n-2t$ honest main-votes, a party must receive an honest main-vote for something other than `abstain` in round r and Lemma 1 implies this must be for v.

[2] If there are $n-2t$ honest pre-votes for 0 and for 1, then since $n>3t$ we have at least $2(n-2t)=n-t+(n-3t)>n-t$ honest parties which is a contradiction.

Putting this together, all honest parties receive a main-vote for v and none for $\neg v$ in round r, thus all have a pre-vote for v in round $r+1$. It follows that all will have a main-vote for v in round $r+1$, and hence all will decide on v in round $r+1$.

Proof of Lemma 1: The statement of Lemma 1 in Cadence SMV is:

```
forall (r in ROUNDS)
  lemma1[r] : assert G ((main_votes[r][0]>0 ∨ corrupted_main[r][0]) ⇒
                        (main_votes[r][1]=0 ∧ ¬corrupted_main[r][1]));
```

The proof of *lemma1* requires a number of sub-lemmas. First, we show that for any round r and value c of the coin in round r, the total number of honest pre-votes for any value does not decrease:

```
forall (r in ROUNDS) for(c = 0; c ≤ 1; c = c + 1)
for(v = 0; v ≤ 1; v = v + 1) forall (n in VOTES)
  prop1[r][c][v][n] : assert G (pre_votes[r][c][v]≥n ⇒ G (pre_votes[r][c][v]≥n));
```

The value of *pre_votes*$[r][c][v]$ is dependent on *all* honest parties. We remove this dependency by first proving that *pre_votes*$[r][c][v]$ does not decrease in one step by *case splitting* on the party that affects the value of *pre_votes*$[r][c][v]$ in this single step, i.e. the party currently scheduled (given by the value of the variable *act*). We also change the abstractions, for example including $r-1$ in the abstraction of *ROUNDS* as a party updates *pre_votes*$[r][c][v]$ in round $r-1$.

```
forall (r in ROUNDS) for(c = 0; c ≤ 1; c = c + 1)
for(v = 0; v ≤ 1; v = v + 1) forall (n in VOTES) {
  prop2[r][c][v][n] : assert G (pre_votes[r][c][v]≥n ⇒ X (pre_votes[r][c][v]≥n));
  forall (i in PROC) { /* case split on the party that is scheduled */
    subcase prop2[r][c][v][n][i] of prop2[r][c][v][n] for act=i;
      using ROUNDS⇒{r − 1, r}, /* change abstractions */ . . .
        prove prop2[r][c][v][n][i]; } }
```

We now prove *prop1*, using *prop2* and assuming that *prop1* holds at time $t-1$:

```
forall (r in ROUNDS) for(c = 0; c ≤ 1; c = c + 1)
for(v = 0; v ≤ 1; v = v + 1) forall (n in VOTES) {
  using (prop1[r][c][v][n]), /* assume prop1 holds at time t − 1 */
    prop2[r][c][v][n] /* use prop2 */, . . .
    prove prop1[r][c][v][n]; }
```

Similarly, we show that once the coin has been tossed it does not change value:

```
forall (r in ROUNDS) for(c = 0; c ≤ 1; c = c + 1)
  prop3[r][c] : assert G (coin[r]=c ⇒ G (coin[r]=c));
```

The proof of *lemma1* is via the following argument. Suppose in a state s there is a main-vote for 0 in round r and coin in round r equals $c \in \{0,1\}$. If there is an honest main-vote for 0, then there exists an honest party i which cast this vote, and therefore if, at the time i casts this vote, we have $coin[r] = c'$ then $pre\_votes[r][c'][0]{=}2n-t$. Using *prop1* and *prop3*, it follows that $c' = c$ and $pre\_votes[r][c][0]{=}2n-t$ in state s. On the other hand, if there is a corrupted main-vote for 0, then $pre\_votes[r][c][0]{=}2n-t$ in state s. Now suppose that there is also a main-vote for 1 in round r, then either there is an honest party j which cast this vote or it is a corrupted vote, and similarly we have $\mathit{pre\_votes}[r][c][1]{=}2n-t$ in state s, which is a contradiction (see *assumption1*).

To adapt this proof to Cadence SMV we need *witnesses*: the parties i and j described above. This is accomplished by the following *history variables* which record the first honest party to cast a main-vote for each value in each round:

```
hist : array ROUNDS of array 0..1 of PROC;
forall (r in ROUNDS) for(v = 0; v ≤ 1; v = v + 1)
  next(hist[r][v]) := next(main_votes[r][v])>0 ∧ main_votes[r][v]=0 ? act : hist[r][v];
```

We now prove *lemma1*, case-splitting on $\mathit{hist}[r][0]$ and $\mathit{hist}[r][1]$, using *prop1*, *prop3* and *assumption1* and changing the abstraction of *VOTES*.

```
forall (r in ROUNDS) forall (i in PROC) forall (j in PROC) {
  subcase lemma1[r][i][j] of lemma1[r] for i=hist[r][0] ∧ j=hist[r][1];
    using prop1[r], prop3[r], assumption1[r], VOTES⇒{0, 2n − t}, . . .
    prove lemma1[r][i][j]; }
```

3.2 Fast Convergence

The proof of this property involves a manual argument based on properties (**P1** – **P6**) that are proved automatically in Cadence SMV. Here we state them in English; for the corresponding formal statements and Cadence SMV proofs see [24]. First we proved that, for any party that enters round $r+1$ and does not decide in round r:

P1 If before the coin in round r is tossed there is a concrete pre-vote (i.e. a vote *not* based on the value of the coin) for v in round $r+1$ and after the coin in round r is tossed it equals v, then the party decides in round $r+1$.

P2 If before the coin in round r is tossed there are no concrete pre-votes in round $r+1$, then either the party decides in round $r+1$, or if after the coins in round r and round $r+1$ are tossed they are equal, then the party decides in round $r+2$.

In addition, we proved that the following properties hold.

P3 If the coin in round r has not been tossed, then neither has the coin in round $r+1$.

P4 In any round r there cannot be concrete pre-votes for 0 and 1.

P5 In any round r, if there is a concrete pre-vote for $v \in \{0,1\}$, then in all future states there is a concrete pre-vote for v.

P6 Each coin is only tossed once.

We complete the proof of **Fast Convergence** with a simple manual proof based on the following classification of protocol states:

- let $\mathit{Undec}(r)$ be the set of states in which the coin in round $r-1$ is not tossed and there are no concrete pre-votes in round r;

– for $v \in \{0,1\}$, let *Pre-vote*(r,v) be the set of states where the coin in round $r-1$ is not tossed and there is a concrete pre-vote for v in round r.

If follows from **P4** that these sets are pairwise disjoint and *any* state where the coin in round $r-1$ is not tossed is a member of one of these sets. The following proposition is crucial to establishing the efficiency of the protocol.

Proposition 2. *In an idealised system, where the values of the coins in rounds* $1, 2, \ldots, 2r-1$ *are truly random, the probability of a party advancing by more than* $2r+1$ *rounds is bounded above by* 2^{-r}*.*

Proof. We prove the proposition by induction on $r \in \mathbb{N}$. The case when $r = 0$ is trivial since the probability bound is 1. Now suppose that the proposition holds for some $r \in \mathbb{N}$ and suppose a party enters round $2r+1$. If a party decides in round $2r$, then by **Agreement** all parties will decide by round $2r+1$, and hence the probability that a party enters round $2r+3$ given a party enters round $2r+1$ is bounded above by 0. On the other hand, if no party decides in round $2r$, then by Proposition 1(a) the coin for round $2r$ is tossed. For any state s reached just before the coin is tossed we have two cases to consider:

– $s \in$ *Pre-vote*$(2r+1, v)$ for some $v \in \{0,1\}$: by **P1**, if the coin in round $2r$ equals v, any party which enters round $2r+1$ decides in round $2r+1$, and hence using **P6** it follows that the probability of a party advancing more than $2r+3$ rounds given that a party advances more than $2r+1$ rounds is bounded above by $1/2$.
– $s \in$ *Undec*$(2r+1)$: using **P5**, there are no concrete pre-votes in round $2r+1$ before the coin in round $2r+1$ is tossed, and hence by **P2** any party either decides in round $2r+1$, or, if the coins in round $2r$ and $2r+1$ are equal, it decides in round $2r+2$. Now, since in s the coin for round $2r$ has not been tossed, by **P3** neither has the coin for round $2r+1$. Therefore, using Proposition 1 and **P6** it follows that the probability of a party advancing more than $2r+3$ rounds given that a party advances more than $2r+1$ rounds is bounded above by $1/2$.

Putting this together and since $\mathbf{P}(A \cap B) = \mathbf{P}(A|B) \cdot \mathbf{P}(B)$, we have

$$\begin{aligned}
&\mathbf{P}(\text{a party advances} > 2r+3 \text{ rounds}) \\
&= \mathbf{P}(\text{a party advances} > 2r+3 \text{ rounds and a party advances} > 2r+1 \text{ rounds}) \\
&= \mathbf{P}(\text{a party advances} > 2r+3 \text{ rounds} \mid \text{a party advances} > 2r+1 \text{ rounds}) \cdot \\
&\quad \mathbf{P}(\text{a party advances} > 2r+1 \text{ rounds}) \\
&\leq 1/2 \cdot 2^{-r} = 2^{-(r+1)}
\end{aligned}$$

as required. □

It can be argued that in a real system the probability of a party advancing by more than $2r+1$ rounds is bounded above by $2^{-r} + \varepsilon$, where ε is negligible. This follows from the **Unpredictability** property of the coin tossing scheme (see Section 3) and **P6**; for more details see [4].

In addition to **Fast Convergence** we can directly prove that the protocol guarantees **Probabilistic Termination** in a constant expected number of rounds by applying the techniques developed in [27,23] which use probabilistic complexity statements. A probabilistic complexity statement has the form:

$$U \xrightarrow{\phi \leq c}_{p} U'$$

where U and U' are sets of states, ϕ is a complexity measure, c is a nonnegative real number and $p \in [0,1]$. Informally the above probabilistic complexity statement means that starting from any state in U, the probability of reaching a state in U' within complexity c is at least p. As in [23], the complexity measure of interest corresponds to the increase in the maximum round number among all the parties. We now sketch the argument for proving that the protocol guarantees **Probabilistic Termination** in a constant expected number of rounds using the probabilistic complexity statements. First, let $\phi_{MaxRound}$ be the complexity statement that corresponds to the increase in the maximum round number among all the parties, and define the following sets of states:

- $\mathcal{R}$, the set of reachable states of the protocol;
- $\mathcal{D}$, the set of reachable states of the protocol in which all parties have decided;
- *Undec*, the set of states in which the coin in round $r-1$ is not tossed and there are no concrete pre-votes in round r, where r is the current maximum round number among all parties;
- *Pre-vote*(v), the set of states where the coin in round $r-1$ is not tossed and there is a concrete pre-vote for v in round r, where r is the current maximum round number among all parties.

Next we require the following property (which is straightforward to prove in Cadence SMV): from any state (under any fair scheduling of the non-determinism) the maximum round increases by at most one before we reach a state where either all parties have decided or the coin in the maximum round has not been tossed, which can be expressed as the following probabilistic complexity statement:

$$\mathcal{R} \xrightarrow{\phi_{MaxRound} \leq 1}_{1} \mathcal{D} \cup \mathit{Undec} \cup \mathit{Pre\text{-}vote}(0) \cup \mathit{Pre\text{-}vote}(1)\,.$$

Applying similar arguments to those given in Proposition 2 we can show that the following probabilistic complexity statements hold:

$$\mathit{Undec} \xrightarrow{\phi_{MaxRound} \leq 2}_{\frac{1}{2}} \mathcal{D} \quad \text{and} \quad \mathit{Pre\text{-}vote}(v) \xrightarrow{\phi_{MaxRound} \leq 2}_{\frac{1}{2}} \mathcal{D} \quad \text{for } v \in \{0,1\}.$$

Then, using the results presented in [27], the above statements can be combined to give:

$$\mathcal{R} \xrightarrow{\phi_{MaxRound} \leq 2+1}_{1 \cdot \frac{1}{2}} \mathcal{D}\,,$$

that is, from any state of the protocol the probability of reaching a state where all parties have decided while the maximum round increases by at most 3 is at least 1/2. Finally, again using the results presented in [27], it follows that from any state of the protocol all parties decide within at most $O(1)$ rounds.

4 Conclusions and Lessons Learnt

The ABBA protocol considered here is a deliverable of the MAFTIA project [20], and this work was prompted by earlier unsuccessful attempts to prove its correctness using data independence and FDR [11]. With the help of Cadence SMV proof assistant, we have succeeded in verifying the protocol correct for an unbounded number of distributed parties and protocol rounds, relative to the correctness of the cryptographic primitives. All proofs are automatic except for high-level arguments involving probabilistic reasoning. The proof consists of approximately 50 lemmas, requiring at most 20 MB of memory. The version of Cadence SMV we used is `08-20-01p1`.

As additional sanity checks we formulated finite state abstractions of the protocol for finite configurations, which we validated with FDR by coding the probabilistic choices in action names and using trace refinement, and used the probabilistic model checker PRISM to calculate the exact (minimum) probabilities (up to $n = 20$, requiring 2.63×10^{14} states and 45 minutes).

The main challenge was to formulate the model of the protocol while using only the restricted operations allowed for the `ordset` data type. In achieving this, which then enabled us to prove correctness for all n, certain changes to the protocol were required; for example, the counting of votes was replaced by a record of the total votes cast for each choice. We have argued that such changes do not reduce the power of the adversary. Due to the current limitations of the `ordset` data type, we could not automatically verify the correctness of these changes, but a modification to the Cadence SMV proof assistant that removes this restriction is possible. Also, we have assumed the correctness of the cryptographic elements of the protocol (the threshold coin tossing and threshold signature schemes). Producing automated proofs of correctness for these schemes does not appear to be an easy task, especially as the manual proofs depend upon the assumption that the RSA and DH problems are hard.

Given our existing experience with Cadence SMV [17] the actual construction of the proofs was relatively straightforward. Although there are over 50 lemmas, many are very simple and similar, and furthermore several were introduced only to simplify later proofs: by assuming simple properties in complex proofs more variables can be declared *free* without invalidating correctness, which then reduces the complexity of the corresponding model checking problems. Overall we found that using Cadence SMV did not require a steep learning curve, and that the tool offers a number of simple but powerful proof techniques.

There are a number of similarities between the strategies and proof techniques used in this case study and the one in [17], and from the success of these we believe that these techniques can be applied to the verification of many other randomized distributed algorithms. In both case studies the majority of properties, for example **Validity** and **Agreement**, require no probabilistic reasoning, and hence in these cases the proof can be fully automated by replacing (abstracting) the probabilistic behaviour with non-determinism. On the other hand, for properties which require probabilistic reasoning, for example expected time properties, by applying the techniques of [27] and isolating the probabilistic arguments, the proof can be split into a number of simpler properties which can be derived automatically either with Cadence SMV or PRISM (for finite configurations after hand-translating the model into PRISM's input language), with only the final high-level arguments verified manually.

Currently the PRISM tool does not support data reduction, which restricts its applicability to complete, finite state, models. Cadence SMV, on the other hand, does not support probabilistic reasoning. A fully automated proof of correctness could feasibly be derived using a theorem prover e.g. [14], but initial discussions indicate that a much greater human effort would be needed. An alternative goal is to develop proof techniques for probabilistic systems in the style of Cadence SMV, incorporating both those used in Cadence SMV and the proof rules for probabilistic complexity statements following [27]. Given an implementation of such rules as a layer on top of, for example, the PRISM model checking tool, one may be able to fully automate the proof of correctness of complex randomized distributed algorithms like the one discussed here. Additional proof rules could also be developed based on *Coin Lemmas* [28,13] and the compositionality results presented in [10].

Acknowledgements

We are grateful to Ken McMillan for suggesting appropriate proof methods and thank William Simmonds for helpful discussions on the Byzantine agreement protocol.

References

1. P. Abdulla, C. Baier, P. Iyer, and B. Jonsson. Reasoning about probabilistic lossy channel systems. In *Proc. CONCUR'00*, volume 1877 of *LNCS*, pages 320–333. Springer, 2000.
2. J. Aspnes and M. Herlihy. Fast randomized consensus using shared memory. *Journal of Algorithms*, 11(3):441–460, 1990.
3. C. Cachin. Distributing trust on the Internet. In *Proc. DSN'01*, pages 183–192. IEEE Computer Society Press, 2001.
4. C. Cachin, K. Kursawe, F. Petzold, and V. Shoup. Secure and efficient asynchronous broadcast protocols (extended abstract). In *Proc. CRYPTO 2001*, volume 2139 of *LNCS*, pages 524–541. Springer, 2001.
5. C. Cachin, K. Kursawe, and V. Shoup. Random oracles in Constantinople: practical asynchronous Byzantine agreement using cryptography (extended abstract). In *Proc. PODC'00*, pages 123–132. ACM Press, 2000.
6. C. Cachin and J. Poritz. Secure intrusion-tolerant replication on the Internet. In *Proc. DSN-2002*, pages 167–176. IEEE Computer Society Press, 2002.
7. M. Castro and B. Liskov. Practical Byzantine fault tolerance. In *Proc. ACM Symp. on Operating Systems Design and Implementation (OSDI)*, pages 173–186, 1999.
8. Coca project: Cornell On-line Certification Authority, Cornell University. http://www.cs.cornell.edu/home/ldzhou/coca.htm.
9. P. D'Argenio, B. Jeannet, H. Jensen, and K. Larsen. Reachability analysis of probabilistic systems by successive refinements. In *Proc. PAPM-PROBMIV 2001*, volume 2165 of *LNCS*, pages 39–56. Springer, 2001.

10. L. de Alfaro, T. Henzinger, and R. Jhala. Compositional methods for probabilistic systems. In *Proc. CONCUR'01*, volume 2154 of *LNCS*, pages 351–365. Springer, 2001.
11. Failures divergence refinement (FDR2). Formal Systems (Europe) Limited, http://www.formal.demon.co.uk/FDR2.html.
12. M. Fischer, N. Lynch, and M. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(5):374–382, 1985.
13. K. Folegati and R. Segala. Coin lemmas with random variables. In *Proc. PAPM-PROBMIV'01*, volume 2165 of *LNCS*, pages 71–86. Springer, 2001.
14. J. Hurd. Verification of the Miller-Rabin probabilistic primality test. In *Proc. TPHOLs'01*, number EDI-INF-RR-0046 in Informatics Report Series, pages 223–238. Division of Informatics, University of Edinburgh, 2001.
15. M. Kwiatkowska and G. Norman. Automated verification of a randomized Byzantine agreement protocol. Technical Report CSR-01-11, University of Birmingham, School of Computer Science, 2001.
16. M. Kwiatkowska, G. Norman, and D. Parker. Probabilistic symbolic model checking with PRISM: A hybrid approach. In *Proc. TACAS'02*, volume 2280 of *LNCS*, pages 52–67. Springer, 2002.
17. M. Kwiatkowska, G. Norman, and R. Segala. Automated verification of a randomized distributed consensus algorithm using Cadence SMV and PRISM. In *Proc. CAV'01*, volume 2102 of *LNCS*, pages 194–206. Springer, 2001.
18. M. Kwiatkowska, G. Norman, and J. Sproston. Symbolic computation of maximal probabilistic reachability. In *Proc. CONCUR'01*, volume 2154 of *LNCS*, pages 169–183. Springer, 2001.
19. M. Kwiatkowska, G. Norman, and J. Sproston. Probabilistic model checking of deadline properties in the IEEE1394 Firewire root contention. *Special Issue of Formal Aspects of Computing*, 2002. To appear.
20. Malicious- and Accidental-Fault Tolerance for Internet Applications (MAFTIA) project. http://www.newcastle.research.ec.org/maftia/.
21. K. McMillan. A methodology for hardware verification using compositional model checking. *Science of Computer Programming*, 37(1–3):279–309, 2000.
22. K. McMillan, S. Qadeer, and J. Saxe. Induction and compositional model checking. In *Proc. CAV'00*, volume 1855 of *LNCS*, pages 312–327, 2000.
23. A. Pogosyants, R. Segala, and N. Lynch. Verification of the randomized consensus algorithm of Aspnes and Herlihy: a case study. *Distributed Computing*, 13(3):155–186, 2000.
24. PRISM web page. http://www.cs.bham.ac.uk/~dxp/prism/.
25. M. Rabin. Randomized Byzantine generals. In *Proc. FOCS'83*, pages 403–409. IEEE Computer Society Press, 1983.
26. A. Roscoe. *The Theory and Practice of Concurrency.* Prentice-Hall, 1997.
27. R. Segala. *Modelling and Verification of Randomized Distributed Real Time Systems.* PhD thesis, Massachusetts Institute of Technology, 1995.
28. R. Segala. The essence of coin lemmas. In *Proc. PROBMIV'98*, volume 22 of *ENTCS*, 1998.
29. V. Shoup. Practical threshold signatures. In *Proc. Eurocrypt 2000*, volume 1807 of *LNCS*, pages 207–220. Springer, 2000.