2014 IEEE 27th Computer Security Foundations Symposium

A Peered Bulletin Board for Robust Use in Verifiable Voting Systems

Chris Culnane and Steve Schneider
Department of Computing, University of Surrey

***Abstract*—The Secure Web Bulletin Board (WBB) is a key component of verifiable election systems. However, there is very little in the literature on their specification, design and implementation, and there are no formally analysed designs. The WBB is used in the context of election verification to publish evidence of voting and tallying that voters and officials can check, and where challenges can be launched in the event of malfeasance. In practice, the election authority has responsibility for implementing the web bulletin board correctly and reliably, and will wish to ensure that it behaves correctly even in the presence of failures and attacks. To ensure robustness, an implementation will typically use a number of peers to be able to provide a correct service even when some peers go down or behave dishonestly. In this paper we propose a new protocol to implement such a Web Bulletin Board, motivated by the needs of the vVote verifiable voting system. Using a distributed algorithm increases the complexity of the protocol and requires careful reasoning in order to establish correctness. Here we use the Event-B modelling and refinement approach to establish correctness of the peered design against an idealised specification of the bulletin board behaviour. In particular we have shown that for n peers, a threshold of $t > 2n/3$ peers behaving correctly is sufficient to ensure correct behaviour of the bulletin board distributed design. The algorithm also behaves correctly even if honest or dishonest peers temporarily drop out of the protocol and then return. The verification approach also establishes that the protocols used within the bulletin board do not interfere with each other. This is the first time a peered secure web bulletin board suite of protocols has been formally verified.**

I. Introduction

Verifiable voting systems such as Prêt à Voter [CRS05], [RBH+09], Scantegrity [CCC+10], Helios[Adi08], Wombat [BNFL+12], STAR-Vote [BBB+13] and Civitas [CCM08] typically have a requirement to publish information concerning votes cast and how they have been processed, in order to provide verifiability. Voters and other external parties are able to check the published information and challenge the election if any cheating has occurred. Such systems are generally described using a "Bulletin Board" for publication: a repository of the information collected throughout the election, made publicly available for inspection.

There are certain (generally implicit) security assumptions on the bulletin board: that once items are on the bulletin board then they will not be removed, that the final information given at the end of the election is fixed and cannot be adjusted, and that it will provide the same view of that information to all parties. For example, Adida's characterisation [Adi06] states that "Cryptographic voting protocols revolve around a central, digital bulletin board. As its name implies, the bulletin board is public and visible to all, via, for example, phone and web interfaces. All messages posted to the bulletin board are authenticated, and it is assumed that any data written to the bulletin board cannot be erased or tampered with." Alternatively a bulletin board has been described as a "broadcast channel with memory" [Pet05], [CGS97], [KTV12], with a Web Bulletin Board treated as a public broadcast channel.

Achieving these properties in an implementation is not so straightforward. A current view is that "*we don't know how to build a secure bulletin board*" [Wag13], and to date there is no generally available implementation of a secure bulletin board. In practice bulletin boards are generally implemented by collecting election information as it progresses, and publishing the information via a website, as done for example by Helios, Wombat, and STAR-Vote, or making it available via a git repository as in the Norway 2013 e-voting trial[Nor13]. However, these are not tamper proof, and information can be changed on them unless there are additional safeguards such as the cryptographic mechanisms based on hash chains proposed by Heather and Lundin [HL08]. The design of STAR-VOTE uses multiple peers to tolerate faulty or malicious components, and has the election authority sign the bulletin board contents, thus changes can occur only with the collusion of the electoral authority.

DOI 10.1109/CSF.2014.20

The bulletin board presented in this paper arises from the need to implement a bulletin board as part of the vVote system being developed for the Victorian State election 2014 [BCH+12]. The Victorian State election runs over a two week period of "early voting" before election day itself, and the bulletin board is required to publish its information daily during the election. For robustness and trust the bulletin board will be comprised of a number of peers to receive items, provide receipts, and publish information. The rate at which votes may be received means that the peers cannot sustain the overhead of a consensus protocol every time an item is posted, so they each maintain a local copy of their view of the bulletin board, and agree on the bulletin board only when it is time to publish. A further challenge is that the bulletin board may need to reject some items, for example any vote on a ballot previously used or audited, so that incompatible posts are not published. We achieve this requirement provided a threshold of the peers are honest and operational the bulletin board will behave correctly, even in the presence of individual peers going down, external attacks and a minority of dishonest peers.

This paper presents a new bulletin board protocol designed to run with a network of peers and to operate correctly when a threshold of the peers are honest and operational. We provide a formal model and verification of the protocol, using the framework of Event-B [Abr10]. We verify the protocol in the context of a Dolev-Yao attacker [DY83], who has control over the network and a minority of peers.

II. A PEERED BULLETIN BOARD PROTOCOL

We present an implementation of a bulletin board that accepts items to be posted (if they do not clash with previous posts), issues receipts, and periodically publishes what it has received. The bulletin board published for any particular period must include all items that had receipts issued during that period. Robustness is achieved through the use of several peered servers which cooperate on accepting items, issuing receipts, and publishing the bulletin board. They make use of a threshold signature scheme which allows a subset of the peers above a particular threshold to jointly generate signatures on data. The peers collectively provide the bulletin board service as long as a threshold of them are honest, and as long as a threshold of them are involved in handling any item posted to the bulletin board. Thus the implementation is correct in the presence of communication failures, unavailability or failure of peers, and also dishonesty of peers. The threshold t required to achieve this must be greater than two-thirds of the total number n of peers: $t > 2n/3$. There is no single point of failure: the system can tolerate failure or non-participation of any component, as long as a threshold of peers remain operational at any stage. It also allows for different threshold sets of peers to be operational at different times. For example, a peer may be rebooted during the protocol, missing some item posts, and may then resume participation.

The key properties we require for this bulletin board are:

- (bb.1) only items that have been posted to the bulletin board may appear on it;
- (bb.2) any item that has a receipt issued must appear on the published bulletin board;
- (bb.3) two clashing items must not both appear on the bulletin board;
- (bb.4) items cannot be removed from the bulletin board once they are published.

It follows from bb.2 and bb.3 that if two items clash then receipts must not be issued for both of them.

The bulletin board provides a protocol for the posting of an item and its acknowledgement with a receipt, and provides another two related protocols for the publishing of the bulleting board: an optimistic one, and a fallback.

Notation: The protocols make use primarily of individual signatures and a threshold digital signature key SSK. We use sk_i to refer to Peer i's individual signing key, and ssk_i to refer to Peer i's share of the threshold signing key SSK. The item $sig_k(x)$ denotes x signed with signature key k.

Posting and acknowledgement: The protocol for posting an item x in period p, and issuing the acknowledgement, is as follows:

1. $User \to P_i \quad : \quad x \qquad\qquad (i \in I)$
   each P_i checks no clash with previous posts
2. $P_i \to P_j \quad : \quad sig_{sk_i}(p,x) \quad (i,j \in I, j \neq i)$
   each P_i waits for a threshold number of signatures
3. $P_i \to User \quad : \quad sig_{ssk_i}(p,x) \quad (i \in I)$

To post an item x, the User should first send x to each of the peers, as shown in Round 1. Each peer checks that x does not clash with any posts it has received previously (from the current period

or previous periods). The peers then sign (p,x) with their own individual signing key, and send the result to each of the other peers, as shown in Round 2. Peers store all of the received signatures into their local database. Finally, once a peer has obtained a threshold number of signatures on (p,x) (including its own), it sends its share of the threshold signature on (p,x) back to the User. Once the User has received a threshold number of such shares it is able to combine them to provide a signature on (p,x), and this serves as the receipt. This protocol is shown in Figure 1. It is repeated for each item to be posted in the period.

Publishing the Bulletin Board: the bulletin board is published at the end of the period. The aim is for the peers to agree on the contents of the bulletin board and to issue their signature share on it to a public hosting service that can combine the signature shares and make the resulting signature publicly available.

Peer i's local record of the bulletin board $B_{i,p}$ is those items that it has received a threshold number of signatures on, which are those items it issues a signature share on towards the receipt.

The peers first of all run an optimistic protocol: this will succeed if at least a threshold of the local bulletin boards agree, which will be the case in practice if all peers are working properly. The optmistic protocol is given as follows:

1. $P_i \rightarrow P_j \quad : \quad sig_{sk_i}(p, B_{i,p}) \qquad (j \neq i)$
   P_i checks a threshold of boards agree
2. $P_i \rightarrow WBB \quad : \quad B_{i,p},\ sig_{ssk_i}(p, B_{i,p}) \quad (i \in I)$

The peers each sign their local copy of the bulletin board, and send them to each other. If a threshold agree then they can issue the bulletin board and a share of the threshold signature on the board. This is illustrated in Figure 2.

If the optimistic protocol does not run successfully, because the signed boards do not agree, that indicates that local bulletin boards are different. In this case the peers exchange information about their bulletin boards using the fallback protocol as follows:

1. $P_i \rightarrow P_j \quad : \quad D_{i,p} \qquad (j \neq i)$
   each P_i updates its database with new data

Each peer sends its database $D_{i,p}$ of signatures it has collected from the posting period to all the other peers, which update their databases with any signatures that are missing. They can then recalculate their local bulletin board. This is illustrated in Figure 3. Observe that the database $D_{i,p}$ consists of signed items, so any dishonest or compromised peer can only include information that has been properly signed by some peer. It can inject fake items by adding its own signature, but such items will not have been previously circulated so will have no other signatures in the peers' databases, thus the honest peers will not include them in their versions of the bulletin board $B_{i,p}$.

After the fallback protocol is completed, the peers return to the optimistic protocol and repeat. This is only required once under our liveness and threshold assumptions. We assume for liveness either (1) that all peers are online and able to communicate during the fallback protocol (with no assumptions about the posting phase or correct behaviour of users), or (2) that a threshold of honest peers are online and able to communicate during the fallback protocol, and at every stage of the posting phase a threshold set of peers were live and able to communicate and that the posting users behaved honestly. Under either of these two assumptions only one round of the fallback protocol is needed. If neither of the liveness assumptions hold then the protocol still behaves safely and ensures correctness of the bulletin board when it is published, but additional rounds of the fallback protocol may be required.

The difference with Byzantine Agreement protocols [Lyn96], which tend to require up to $(n-t)+1$ rounds to achieve agreement, is that the databases the peers start with have some consistency between them. If thresholds of peers received posts correctly in the posting phase, then the honest peers involved in the exchange of information in the fallback protocol will all obtain the full bulletin board after one round.

III. Modelling and Verification Framework

We use the action systems approach of Event-B [Abr10], [MAV05] as our formal framework to model the protocol and to verify it. In this approach systems are described in terms of the **states** that they can be in, and the **events** that transform the state.

A system is defined as a *machine*, which encapsulates its state, and its events. State information is described in terms of state variables and invariants on them. The machine describes how the state is initialised, and how it can be updated with events.

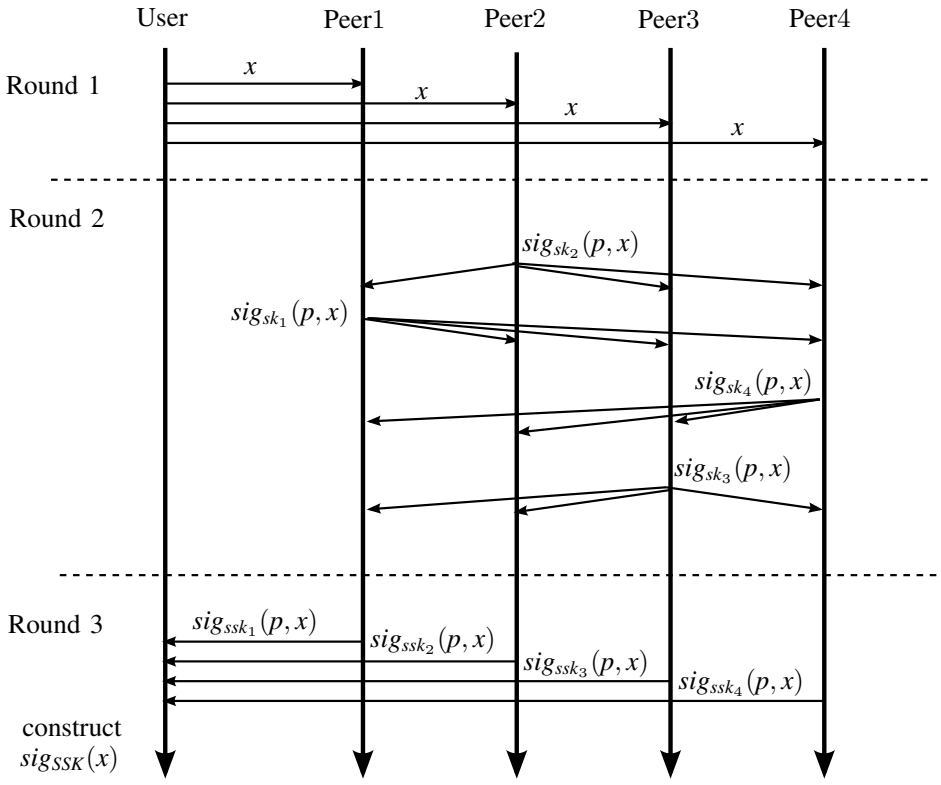

Fig. 1. Posting Protocol

machine M
variables v
invariant $I(v)$
events $init, ev, \ldots$
end

$ev \mathrel{\widehat{=}}$
when $G(v)$
then $v :| BA(v, v')$
end

Fig. 4. Template of an Event-B machine and an event.

The Event-B approach supports *refinement*, a relationship showing when one system implements another. This approach allows a specification to be captured as an ideal machine that expresses the required behaviour. An implementation satisfies the specification if it is a refinement.

Figure 4 illustrates how a machine is defined. Machine M is given with a list of state variables v, a state invariant $I(v)$, and a set of events $ev, \ldots$ to update the state. Initialisation is a special event *init* which sets the initial state of the machine, and its guard is *true*.

Each event has a *guard* $G(v)$ over the variables v, and a *body*, usually written as an assignment S on the variables. The assignment is associated with a *before-after predicate* $BA(v, v')$ describing changes of variables upon event execution, in terms of the relationship between the variable values before (v) and after (v'). In Event-B an event may also introduce local variables, which can be included in the guard (which constrains what values they can take), and in the body where they can be used to define the change of state. Such events are constructed as:

$evt \mathrel{\widehat{=}}$ **any** x
where $G(v, x)$
then $v :| BA(v, x, v')$
end

Some of the conditions on x may be included in the **any** clause rather than the **where** clause for readability (see e.g. *post* and *a_msg*1 of Figure 5). Nondeterministic assignment has its own syntax: $x :\in S$ assigns x some arbitrary element of S. This is an abbreviation for

any s **where** $s \in S$ **then** $x := s$ **end**.

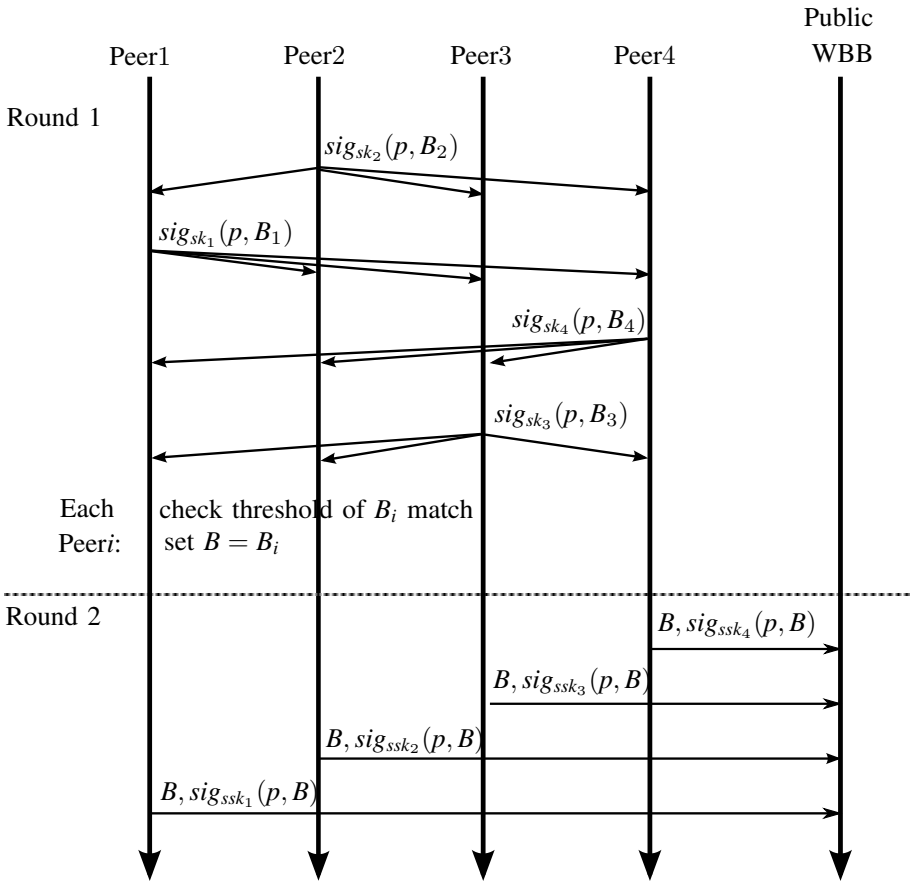

Fig. 2. Optimistic Protocol

We also note a technical Event-B condition: that in these models all events have some feasible final state: whenever $G(v,x)$ is true then there is some v' such that $BA(v,x,v')$ holds. This means that we need not be concerned with proof conditions for establishing feasibility.

The Event-B approach to semantics, provided in [Abr10], [MAV05], is to associate proof obligations with machines. The key proof obligation on an event is that it preserves the invariant: when an event is called within its guard, then the state resulting from executing the body should meet the invariant. This is true for the machines presented in this paper.

A. Event-B refinement

In Event-B, the intended refinement relationship between machines is directly written into the refinement machine definitions. As a consequence of writing a refining machine, a number of proof obligations arise. Here, a machine and its refinement take the following form:

machine M_0	**machine** M_1
variables v	**refines** M_0
invariant $I(v)$	**variables** w
events $init_0, ev_0, ev'_0, \ldots$	**invariant** $J(v,w)$
end	**events** $init_1, ev_1, ev'_1, \ldots$
	end

The machine M_0 is refined by machine M_1, written $M_0 \preccurlyeq M_1$, if the given *linking invariant* $J(v,w)$ on the variables of the two machines is established by their initialisations, and preserved by all events. Any transition performed by a concrete event of M_1 can be matched by a step of the corresponding abstract event of M_0, or matched by *skip* for newly introduced events, in order to maintain J. This is similar to the approach of downwards simulation data refinement

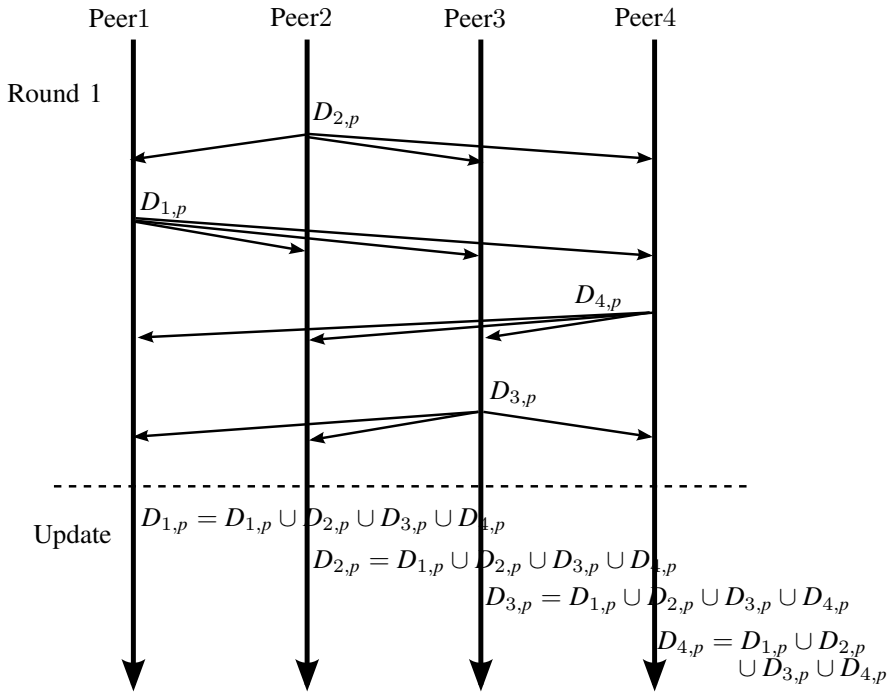

Fig. 3. Fallback Protocol

[DB01], where the *simulation relation* plays the role of the linking invariant. Formally, the refinement relation $M_0 \preccurlyeq M_1$ between abstract machine M_0 and concrete machine M_1 holds if the following proof obligations given below hold for all events:

GRD_REF: Guard Strengthening If a concrete event matches an abstract one, then this rule requires that when the concrete event is enabled, then so is the matching abstract one.

INV_REF: Simulation There are two parts to this rule. INV_REF_1 ensures that the occurrence of events (including initialisation) in the concrete machine can be matched in the abstract one. INV_REF_2 is concerned with new events, which are treated as refinements of *skip*. In this case the abstract state does not change.

Refinement with respect to A: It may be that an environment interacts with a machine M_0 only on some subset A of its events. In that case we can consider a refinement M_1 of M_0 with respect to A. This requires that M_1 also has all the events A, and that GRD_REF and INV_REF_1 must hold for all the events in A. However, other events of M_1 can be matched either by *skip*, or by some matching event (not in A) in M_0, in which case the guard must also match. We will use this notion of refinement to express our requirements on the bulletin board protocol, with the set $A = \{post, ack, publish\}$.

B. Framework for Bulletin Board Modelling and Verification

We are concerned with developing a peered bulletin board that can operate correctly in an unreliable environment, and with some potentially misbehaving peers. In particular, communications between the bulletin board and its users may be under the control of an adversary, who may intercept, divert, block, duplicate and spoof messages. The bulletin board is designed for use in in such an environment.

The specification of the bulletin board will encapsulate the required behaviour. This will be described as an Event-B model *BBSpec* with a description in terms of the architecture shown in Figure 5, of an ideal bulletin board in the context of a reliable communication medium. Users may use the events *post*, *ack* and *publish* to interact with the bulletin board, but communication occurs via the medium.

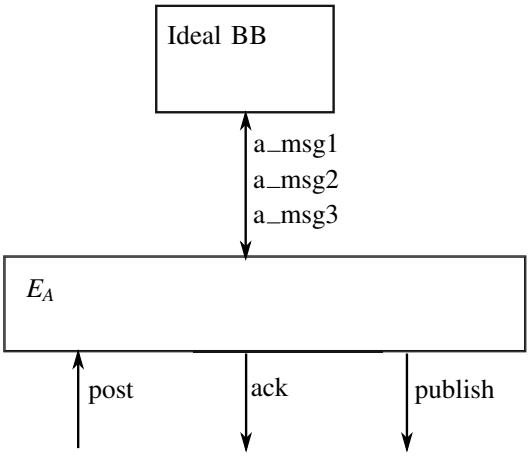

Fig. 5. *BBSpec*: ideal bulletin board and communication medium

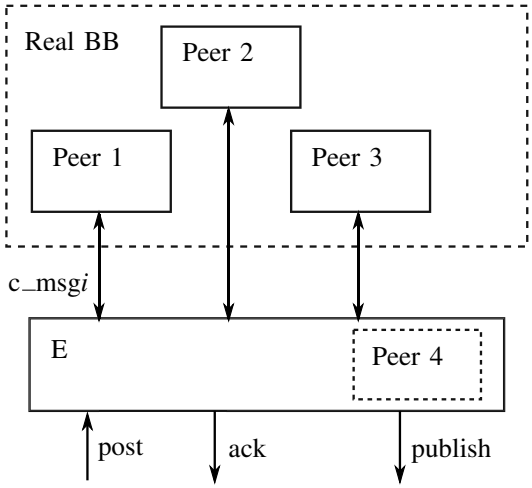

Fig. 6. *BBProt*: Protocol model for analysis, with $t = 3$ and $n = 4$

The bulletin board has its own corresponding interactions with the medium, labelled a_msg1, a_msg2 and a_msg3. These events are also within the model *BBSpec*, but they are not accessible directly to users. Hence it is the behaviour of *BBSpec* on the set of events $\{post, ack, publish\}$ that must be matched by any implementation.

The bulletin board implementation uses a number of peers, for robustness and in order to distribute trust. There are a total of n peers, and we use a threshold signature scheme in which we require t shares in order to produce a signature. Our model of the protocol will be an Event-B model *BBProt*, in which we consider the adversary to control the communication medium to and from the peers and the *WBB*, and between them. Hence any communication can be blocked. We also consider that the adversary can control up to $n - t$ peers. This means that such peers can sign and create any messages for sending, whether or not such messages are in accordance with the protocol, provided they have the appropriate keys.

We consider that (at least) a threshold t of the n peers are honest: that they follow the protocol. Without loss of generality we will consider peers 1 to t to be honest, and $t + 1$ to n may behave arbitrarily (which includes honest behaviour). This labelling of the peers captures the general case where some arbitrary $n - t$ peers may be dishonest, since the protocol is symmetric with respect to the labelling of the peers.

The model *BBProt* includes the Dolev-Yao adversary, and peers $t+1$ to n considered to be under the control of the adversary. The setup is illustrated in Figure 6.

BBProt offers the same three external events as *BBSpec*, namely *post*, *ack* and *publish*. However it contains the peers explicitly, including peers controlled by the adversary, and so the communication patterns with and between the peers will be quite different to those in the specification. Those communications are modelled by events *c_msgi*. The requirement for correctness will be that *BBSpec* $\preccurlyeq$ *BBProt* with respect to $\{post, ack, publish\}$.

The model *BBProt* includes the events that make up the various bulletin board protocols. Since these events can be performed whenever their guards are true, this means that interleavings of different protocols are naturally considered within this framework. Thus our approach to verification automatically allows for possible interference between the protocols, and a proof of correctness establishes that the protocols cannot interfere in an adverse way.

IV. Bulletin Board Modelling

A. Specification

The ideal behaviour of the bulletin board is given according to the specification *BBSpec* of Figure 7. Receipts will be issued with the period the item was posted in, and a bulletin board will be published for

machine *BBSpec*
variables $E_A, R_p, C_p \quad (p \in \mathbb{N})$
invariant $E_A \subseteq ITEM \cup RECEIPT \cup PUBLISH$
$R_p \subseteq ITEM \quad (p \in \mathbb{N})$
$C_p \subseteq ITEM \quad (p \in \mathbb{N})$
events
init $\widehat{=}$ $E_A := \{\} \parallel \Big\|_{p \in \mathbb{N}} (R_p := \{\} \parallel C_p := \{\})$
post(x) $\widehat{=}$ **when** $x \in ITEM$
then $E_A := E_A \cup \{x\}$ **end**;
$r \longleftarrow$ ack $\widehat{=}$ $r :\in (E_A \cap RECEIPT)$;
$P \longleftarrow$ publish $\widehat{=}$ $P :\in (E_A \cap PUBLISH)$;
a_msg1 $\widehat{=}$
any x, p
where $x \in E_A \cap ITEM$
$\wedge\ p \in \mathbb{N} \wedge clashset(x) \cap (\bigcup_p R_p) = \{\}$
then $R_p := R_p \cup \{x\}$
end;
a_msg2 $\widehat{=}$
any x, p
where $x \in R_p \wedge (sig_{SSK}(p,B) \in E_A \Rightarrow x \in B)$
then $E_A := E_A \cup \{sig_{SSK}(p,x)\}$
$\parallel C_p := C_p \cup \{x\}$
end;
a_msg3 $\widehat{=}$
any Y, p
where $C_p \subseteq Y \subseteq R_p \wedge E_A \cap PUBLISH_p = \{\}$
then $E_A := E_A \cup \{sig_{SSK}(p,Y)\}$
end
end

Fig. 7. *BBSpec*: specification of ideal bulletin board behaviour

this period. We define

$$\begin{aligned} RECEIPT &= \{sig_{SSK}(p,x) \mid x \in ITEM \wedge p \in \mathbb{N}\} \\ PUBLISH &= \{sig_{SSK}(p,B) \mid B \subseteq ITEM \wedge p \in \mathbb{N}\} \\ PUBLISH_p &= \{sig_{SSK}(p,B) \mid B \subseteq ITEM\} \end{aligned}$$

The model uses the set E_A to capture the contents of the communication medium between the user and the bulletin board as shown in Figure 5. Messages are sent by adding messages to E_A (as in the descriptions of *post*, *a_msg*2 and *a_msg*3), and received by obtaining them from E_A (as in *ack*, *publish* and *a_msg*1).

The external events of *BBSpec* capture the interactions available to the user: *post*(x) allows a value x to be sent to the bulletin board; $r \longleftarrow ack$ occurs when a receipt $r = sig_{SSK}(p,x)$ is output as acknowledgement of posting of an item x; and $P \longleftarrow publish$ is the output of a signed bulletin board $P = sig_{SSK}(p,B)$.

The internal events describe the behaviour for the bulletin board. The event *a_msg*1 models the posting of an item x in period p. The bulletin board may refuse posts if they are inconsistent with previously accepted posts. We express this by introducing an irreflexive symmetric binary relation *clash* such that $clash(x,x')$ captures when two items x and x' should not both appear on the bulletin board. For convenience we define $clashset(x) = \{x' \mid clash(x,x')\}$ to be the set of all events that clash with x. The guard of the event expresses that x will be accepted if there is no x' already received in any period clashing with x: in other words, that $clashset(x) \cap (\bigcup_p R_p) = \{\}$.

Allowing for untrusted peers requires us to include some nondeterminism within the specification of the bulletin board. In particular, dishonest peers and the untrusted medium can prevent or delay receipts from being issued for some received posts, so the specification must allow for this possibility. The model specifying the bulletin board thus uses two databases: R consisting of received posts, and C consisting of confirmed posts—those which have been acknowledged with receipts. *a_msg*2 models the issuing of a receipt, and in accordance with the inherent nondeterminism allows for a receipt to be issued even after the board is published, provided the item is on the board as required by (bb.2).

Finally, *a_msg*3 provides the published board: any item published must be in R in accordance with (bb.1); and all confirmed posts C must be published in accordance with (bb.2). Thus we require $C \subseteq Y \subseteq R$ for a published board Y. In other words, items that have been submitted to the bulletin board but not confirmed might or might not appear in Y. We retain a level of uncertainty over what is published, because this level of uncertainty is present in the implementation when some of the bulletin board peers are untrusted. *a_msg*3 allows no more than one bulletin board to be published for any period, meeting requirement (bb.4): once published, the bulletin board is fixed.

Observe that if every posting has a receipt, then the bulletin board will contain all posted items: ($C = B = R$).

The resulting Event-B model is given in Figure 7.

This is the specification that we will show is met by our design. The model includes a bulletin board and its environment. As well as the state of the bulletin board, we include the state E_A of the environment, containing the communications that it is managing, because we will want to consider the bulletin board protocol design in a model including the Dolev-Yao adversary, which provides an asynchronous communication medium.

We see that *BBSpec* exhibits the original requirements (bb.1)–(bb.4). (bb.1) holds because any published bulletin board is a subset of the items in R_p, which is a subset of the items in E, and any item x in E must be there as a result of $post(x)$; hence only posted items can appear on the bulletin board. (bb.2) holds because C_p contains the items for which a receipt has been provided, and a_msg3 states that a bulletin board for period p must contain at least the events in C_p. (bb.3) holds because two clashing items can never both be present in the set of all received posts $\bigcup_p R_p$, because the guard in a_msg1 prevents acceptance of items that clash with previous ones. (bb.4) follows because a_msg3 ensures that at most one bulletin board will be published for any period p: once a signed bulletin board is output then it cannot be overwritten or superceded.

The fact that *BBSpec* satisfies the requirements (bb.1)–(bb4) means that any refinement of *BBSpec* will also meet these requirements.

B. Event-B Modelling and analysis

The threat model built into the model of the protocol incorporates our robustness considerations, in particular that the correctness of the bulletin board is not dependent on the correct behaviour of any individual component, as long as a threshold behave correctly. It allows for the case where peers behave honestly but occasionally are down (either through connection loss, or through temporary server loss): this is modelled simply by the absence of messages between the communication medium and the peer, and allows for peers to miss some posts. The model also includes the case where peers $t+1$ to n can lose or otherwise alter their databases of received posts. However, the honest peers 1 to t do not lose their databases: for correctness we require that a threshold of peers do not lose their data.

The set *MESSAGE* of all possible messages m in the model is given as follows:

$$m \quad ::= \quad k \mid i \mid sig_k(m) \mid \{m_1, \ldots, m_n\} \mid (p, m)$$

$$\begin{array}{rrcl} & & \vdash & \{\} \\ & \{k, m\} & \vdash & sig_k(m) \\ \#S \geqslant t \Rightarrow & \{sig_{ssk_k}(m) \mid k \in S\} & \vdash & sig_{SSK}(m) \\ & \{sig_k(m)\} & \vdash & m \\ & \{m, B\} & \vdash & B \cup \{m\} \\ m \in B \Rightarrow & \{B\} & \vdash & m \\ & \{p, m\} & \vdash & (p, m) \\ & \{(p, m)\} & \vdash & p, m \end{array}$$

Fig. 8. Dolev-Yao adversary derivations

where $k \in KEY$, $i \in ITEM$, $p \in \mathbb{N}$, $\{\ldots\}$ denotes the set of messages listed, and (p, m) denotes a pair. Observe that a message can itself consist of a set of messages, and thus *MESSAGE* covers all rounds of the protocol. In particular $RECEIPT \subseteq MESSAGE$ and $PUBLISH \subseteq MESSAGE$.

The Dolev-Yao environment: The Dolev-Yao adversary is modelled through the use of the set E to retain all messages that are sent and received by protocol parties. The adversary is also able to generate new messages to introduce into protocol executions. In particular, he can sign any message with any key that he possesses; he can combine shares of a signature into a threshold signature; he can extract the message from a signature; and he can add and remove messages from a set of messages. These capabilities are captured in the derivation rules of Figure 8, which show how a new message can be generated from a set of messages. We will model adversary behaviour by including an event for each rule, allowing the adversary to introduce new events to the set E.

C. Simulation

We aim to establish that the concrete system *BBProt* refines the abstract system *BBSpec* with respect to the external events $\{post, ack, publish\}$.

To establish refinement we show that any concrete move can be matched by an abstract move, or (for events other that *post*, *ack* and *publish*) matched by *skip*. To do this we need to identify the *linking invariant*, the relationship between the abstract and concrete states, and show that any concrete move from a concrete state is matched for any corresponding abstract state by some abstract move or *skip*.

Linking invariant

We thus have to identify when, in the concrete system, abstract events are considered to have occurred.

- abstract *a_msg*1 occurs when the bulletin board receives x. In the concrete model this corresponds to t peers having received x and signed it. Since there can be up to $n-t$ dishonest peers, this means $t-(n-t) = 2t-n$ honest peers having signed x.
- abstract *a_msg*2 occurs when the bulletin board issues a signature on (p,x). This corresponds to the combining of t returns of signature shares $sig_{ssk_j}(p,x)$.
- abstract *a_msg*3 occurs when a signed database $sig_{SSK}(p,t(D))$ is produced. This corresponds to the combining of t returns of signature shares $sig_{ssk_j}(p,t(D))$.

The abstract state s_A is the pair of databases R and C, and medium E_A.

The concrete state s_C is the set of databases I_j, D_j and pub_j for the peers, E for the Dolev-Yao environment.

For the linking invariant to relate the abstract state to the concrete state:

- R_p is the set of items for which the adversary (and possibly other peers) can provide a threshold of $sig_{sk_j}(p,x)$, and so can include x on the published bulletin board. If at least $2t-n$ honest peers have signed (p,x), then it is within the adversary's control to produce a further $n-t$ signatures, giving a threshold of signatures on (p,x).
- C_p is the set of items for which the adversary has a receipt—evidence that sufficiently many peers have a threshold of $sig_{sk_j}(p,x)$ to ensure that it will appear on the published bulletin board.

These are expressed in Clauses (9) and (10) respectively of *BBProt* given in Figure 9.

D. Implementation

In the implementation, each peer maintains a counter p_j which it uses to track the period it is currently accepting posts for. The counter will be incremented when it has finished accepting posts for one period and begins accepting posts for the next. It also maintains a separate state space for each period. For example, *BBProt* will use $R_{j,p}$ for j's record of what it received in period p, and so will have a separate set for each period.

The resulting model *BBProt* is given in the various clauses below. The model is shown in the events within the description. The key to the refinement proof is that the interleaving of the events across the different periods do not interfere, even though peers can progress their periods independently and can be involved in publication of one bulletin board while receiving items for another.

Declaration and Invariant: Two further definitions will be useful when expressing the model:

$$\begin{aligned} SIG1_p &= \{sig_{SSK}(p,x) \mid x \in ITEM\} \\ t(D) &= \{x \mid \#\{k \mid sig_{sk_k}(x) \in D\} \geqslant t\} \end{aligned}$$

Given a set D of signed items, the set $t(D)$ is those items for which D contains a threshold number of different signatures. If D is used to track the signed items received by a peer, then $t(D)$ is those items for which it has received a threshold number.

The invariant properties are the key to the proof of refinement, and hence of correctness. These are given in Figure 9. Clauses (1)–(8) are the key invariant properties of the refinement machine *BBProt* itself, and (9)–(11) capture the linking invariant which relates the refinement to the abstract machine *BBSpec* and hence establishes the refinement relationship.

The invariant clauses give relationships between various parts of the state of *BBProt*. (1)–(8) capture behaviour exhibited by the honest peers j where $1 \leqslant j \leqslant t$. (1) states that for any honest peer j, if j has provided a signature share on x for period p and also on board B for the same period, then the post x must be on the board B. This captures one aspect of honest behaviour. (2) states that if j's signature share on (p,x) has been communicated then j has received at least the threshold t number of signatures on (p,x). (3) states that if a threshold signature on (p,x) has been constructed then at least thre threshold number of signature shares on (p,x) have been communicated. (4) states that if j has communicated a signature share on a board B then j must have received at least t peer signatures for each item on B. (5) states that if a threshold signature on (p,B) has been constructed then a threshold of signature shares on (p,B) have been communicated. (6) states that all the peer signatures received by an honest peer have been communicated. (7) states that if peer j has produced a signature share on (p,B) then its current period must be greater than p. (8) states

that j will provide a signature share on at most one bulletin board for any period p. All of these clauses are true after initialisation, and they are established to be preserved by each of the events of *BBProt*, including attacker events. Hence they are true in any state that *BBProt* can be brought to.

The linking invariant is given by clauses (9)–(11). Clauses (9) and (10) were described on the previous page; clause (11) states that the abstract set of communications is the projection of the concrete set to the abstract messages. The linking invariant underpins Lemma IV.2, which establishes the refinement relation between *BBSpec* and *BBProt*: given a pair of linkes states of *BBSpec* and *BBProt*, each event in *BBProt* can be matched either by no progress (i.e. *skip*) within *BBSpec*, or by the equivalent event within *BBSpec*, whereby the linking invariant continues to hold in the resulting states.

External events: External events are modelled as in Figure 10.

Posting and acknowledgement protocol: Posting and acknowledgement captures the protocol of Figure 1 as input and output events between the individual peers and the medium. $Peer_j$ may only accept and acknowledge items, and issue its share of the receipt, for items in its current period. The associated events are given in Figure 11.

Commit and publish protocol: The commit protocol for the bulletin board of period p_j is started by incrementing p_j. Thus no further posts will be accepted for that bulletin board, and the events in the commit and publish protocols are then enabled. These are given in Figure 12.

Dolev-Yao environment: As well as handling all communication and thus having access to all messages passed between the agents, the adversary has actions corresponding to the derivations of Figure 8. These are given in Figure 13.

This gives rise to some additional clauses in the invariant as shown in Figure 14, to capture the limits of what the adversary can introduce. These are necessary for the refinement proof, in particular to establish that the concrete adversary can only generate protocol messages that the abstract adversary can generate, and that the additional message fragments (for example shares of threshold signatures) do not give the adversary any additional power.

machine *BBProt*

refines *BBSpec*

variables $E,\ I_{j,p},\ D_{j,p},\ S_{j,p},\ c_j,\ p_j\ \ (1 \leqslant j \leqslant t)$

invariant

/* Types */

$E \subseteq MESSAGE$

$I_{j,p} \subseteq ITEM$

$D_{j,p} \subseteq \{sig_{sk_k}(p,x) \mid x \in ITEM\}$

$S_{j,p} \subseteq \{sig_{sk_k}(p,B) \mid B \subseteq ITEM\}$

$c_j \in \mathbb{N}$

$p_j \in \mathbb{N}$

/* Key invariant properties */

$1 \leqslant j \leqslant t \Rightarrow$

(1) $j \in c[p,x] \wedge j \in s[p,B] \Rightarrow x \in B$

(2) $sig_{ssk_j}(p,x) \in E \Rightarrow \#d_j[p,x] \geqslant t$

(3) $sig_{SSK}(p,x) \in E \Rightarrow \#c[p,x] \geqslant t$

(4) $sig_{ssk_j}(p,B) \in E \Rightarrow B \subseteq t(D_{j,p})$

(5) $sig_{SSK}(p,B) \in E \Rightarrow \#s[p,B] \geqslant t$

(6) $D_{j,p} \subseteq E$

(7) $j \in s[p,B] \Rightarrow c_j > p$

(8) $j \in s[p,B_1]$
$\wedge\ B_1 \neq B_2 \Rightarrow j \notin s[p,B_2]$

/* linking invariant */

(9) $R_p =$
$\{x \in ITEM \mid \#\{j \mid 1 \leqslant j \leqslant t$
$\wedge\ sig_{sk_j}(p,x) \in E\} \geqslant 2t - n\}$

(10) $C_p = \{x \in ITEM \mid sig_{SSK}(p,x) \in E\}$

(11) $E_A = E \cap (ITEM \cup RECEIPT$
$\cup\ PUBLISH)$

where:

$d_j[p,x] = \{1 \leqslant k \leqslant n \mid sig_{sk_k}(p,x) \in D_j\}$

$c[p,x] = \{1 \leqslant k \leqslant n \mid sig_{ssk_k}(p,x) \in E\}$

$s[p,B] = \{1 \leqslant k \leqslant n \mid sig_{ssk_k}(p,B) \in E\}$

Fig. 9. *BBProt*: declaration of variables and invariant

E. Simulation

The following counting lemma is useful in the refinement proofs:

Lemma IV.1. *If $A \subseteq \{1,\ldots,n\}$, $B \subseteq \{1,\ldots,n\}$, $\#A \geqslant t$, $\#B \geqslant t$, and $t > 2n/3$, then there is some $j \leqslant t$ such that $j \in A$ and $j \in B$.*

Thus we obtain the key result: that *BBProt* provides

events

init $\widehat{=} E := \{sk_k \mid k > t\} \cup \{ssk_k \mid k > t\} \parallel$
$\quad \Big\|_{j,n}(I_{j,n} := \{\} \parallel D_{j,n} := \{\}$
$\quad \parallel S_{j,n} := \{\} \parallel p_j := 0 \parallel c_j := 0)$;

post$(x) \widehat{=}$ **when** $x \in ITEM$
then $E := E \cup \{x\}$ **end**;

$r \longleftarrow$ ack $\widehat{=} r :\in (E \cap RECEIPT)$;

$P \longleftarrow$ publish $\widehat{=} P :\in E \cap PUBLISH$;

Fig. 10. *BBProt*: external events

c_msg1$_j(x) \widehat{=}$ /* receive item x */
when $x \in E \cap ITEM$
then $I_{j,p_j} := I_{j,p_j} \cup \{x\}$
end;

c_msg2a$_j \widehat{=}$ /* send signature share on x */
any x
where
$x \in I_{j,p_j} \wedge$
$clashset(x) \cap \{y \mid \exists p'.\ sk_j(p', y) \in \bigcup_p D_{j,p}\} = \{\}$
then $E := E \cup \{sig_{sk_j}(p_j, x)\}$
$\quad \parallel D_{j,p_j} := D_{j,p_j} \cup \{sig_{sk_j}(p_j, x)\}$
end;

c_msg2b$_j \widehat{=}$ /* receive signature share on x */
any x, k
where $sig_{sk_k}(p_j, x) \in E$
then $D_{j,p_j} := D_{j,p_j} \cup \{sig_{sk_k}(p_j, x)\}$
end;

c_msg3$_j \widehat{=}$ /* send signature share */
any x
where $x \in t(D_{j,p_j})$
then $E := E \cup \{sig_{ssk_j}(p_j, x)\}$
end;

Fig. 11. *BBProt*: Posting and acknowledgement

an implementation of *BBSpec*:

Lemma IV.2. *BBSpec* $\preccurlyeq$ *BBProt with respect to* $\{post, ack, publish\}$

Proof (sketch)

We need to prove that if $J(s_A, s_C)$, and $s_C \xrightarrow{m_C} s'_C$ then either $J(s_A, s'_C)$ (m_c is matched by *skip*), or $\exists\, m_A, s'_A$ such that $s_A \xrightarrow{m_A} s'_A$ and $J(s'_A, s'_C)$ (m_c is matched by m_A).

c_msg4$_j \widehat{=}$ /* start commit protocol */
begin
$p_j := p_j + 1$
end;

c_msg5a$_j \widehat{=}$ /* send database */
any $p < p_j$
then $E := E \cup \{D_{j,p}\}$
end;

c_msg5b$_j \widehat{=}$ /* receive k's database */
any D, p
where $D \in E \wedge D \subseteq SIG1_p \wedge p < p_j$
then $D_{j,p} := D_{j,p} \cup D$
end;

c_msg6$_j \widehat{=}$ /* publish signature share */
when $c_j < p_j \wedge$
$\quad \#\{k \mid sig_{sk_k}(c_j, t(D_{j,c_j})) \in S_{j,c_j}\} \geqslant t$
then $E := E \cup \{sig_{ssk_j}(c_j, t(D_{j,c_j}))\}$
$\quad \parallel c_j := c_j + 1$
end;

c_msg7a$_j \widehat{=}$ /* send signed board */
any $p < p_j$
then $E := E \cup \{sig_{sk_j}(p, t(D_{j,p}))\}$
end;

c_msg7b$_j \widehat{=}$ /* receive signed board */
any B, p
where $sig_{sk_j}(p, B) \in E \wedge p < p_j$
then $S_{j,p} := S_{j,p} \cup \{sig_{sk_j}(p, B)\}$
end;

Fig. 12. *BBProt*: commitment protocols (fallback and optimistic)

The proof establishes each case in turn. We show two key example cases: *c_msg*2*a* and *c_dy*2

Case *c_msg*2*a*. *Peer* $j \rightarrow DY : sig_{sk_j}(p_j, x)$. If $\#\{k \mid 1 \leqslant k \leqslant t \wedge sig_{sk_k}(p_j, x) \in E\} = 2t - n - 1$ and $\#\{k \mid 1 \leqslant k \leqslant t \wedge sig_{sk_k}(p_j, x) \in E'\} = 2t - n$ then matched by $m_A = a\_msg1$ for x, p_j. Otherwise matched by *skip*.

Case *c_dy*2. If $x \in ITEM$ and $sig_{SSK}(p, x) \notin E$ and $sig_{SSK}(p, x) \in E'$, then this is matched by *a_msg*2. It remains to show that (1) $x \in R_p$ and (2) $sig_{SSK}(p, B) \in E_A \Rightarrow x \in B$.

1) $x \in R_p$: We have that $\#c[p, x] \geqslant t$. Hence there is some $k \leqslant t$ with $k \in c[p, x]$, so by invariant (2) it follows that $\#d_k[p, x] \geqslant t$. By invariant (6) it follows that $\#\{k \mid sig_{sk_k}(p, x) \in E\} \geqslant t$, and hence that $\#(\{k \mid sig_{sk_k}(p, x) \in$

c_dy1 $\widehat{=}$ /* signature share on m */
any m, s
where $m \in E \wedge s \in E$
then $E := E \cup \{sig_s(m)\}$
end;

c_dy2 $\widehat{=}$ /* threshold signature on m */
any S, m
where $\#S \geqslant t \wedge \{sig_{ssk_k}(m) \mid k \in S\} \subseteq E$
then $E := E \cup \{sig_{SSK}(m)\}$
end;

c_dy3 $\widehat{=}$ /* extracting m from signature */
any m, s
where $sig_s(m) \in E$
then $E := E \cup \{m\}$
end;

c_dy4 $\widehat{=}$ /* adding m to B */
any m, B
where $m \in E \wedge B \in E$
then $E := E \cup \{B \cup \{m\}\}$
end;

c_dy5 $\widehat{=}$ /* extracting m from B */
any m, B
where $B \in E \wedge m \in B$
then $E := E \cup \{m\}$
end;

c_dy6 $\widehat{=}$ /* pairing */
any m, p
where $m \in E \wedge p \in \mathbb{N}$
then $E := E \cup \{(p, m)\}$
end;

c_dy7 $\widehat{=}$ /* splitting */
any m, p
where $(p, m) \in E$
then $E := E \cup \{p, m\}$
end

Fig. 13. *BBProt*: adversary actions

$E\} - \{t+1 \ldots n\}) \geqslant t - (n-t) = 2t - n$. Hence by (9) $x \in R_p$, as required.

2) $sig_{SSK}(p, B) \in E_A \Rightarrow x \in B$: Assume $sig_{SSK}(p, B) \in E_A$. Then $\#s[p, B] \geqslant t$ by (5). Also we have $\#c[p, x] \geqslant t$, so by Lemma IV.1 there is some $k \leqslant t$ with $k \in c[p, x]$ and $k \in s[p, B]$. Hence from (1) it follows that $x \in B$ as required.

If $B_C \subseteq ITEM$ and $sig_{SSK}(p, B_C) \notin E$ and $sig_{SSK}(p, B_C) \in E'$, then this is matched by *a_msg*3, with $B = B_C$. We must show that (1) $E_A \cap PUBLISH = \{\}$, (2) $C_p \subseteq B_C$ and (3) $B_C \subseteq R_p$.

1) $E_A \cap PUBLISH = \{\}$: we establish this by contradiction. If $sig_{SSK}(p, B) \in E_A$ for some $B \neq B_C$, then $\#s[p, B] \geqslant t$ by (5). Also we have $s[p, B_C] \geqslant t$ by the guard of *c_dy*2. Hence from Lemma IV.1 there is some $k \leqslant t$ with $k \in s[p, B]$ and $k \in s[p, B_C]$, contradicting (8).
2) $C \subseteq B_C$: consider some $x \in C$. Then $sig_{SSK}(p, x) \in E$, so $\#c[p, x] \geqslant t$ by invariant (3). Further, $\#s[B_C] \geqslant t$ by invariant (5). Hence from Lemma IV.1 there is some $k \leqslant t$ with $k \in c[p, x]$ and $k \in s[p, B_C]$. Hence by invariant (1), $x \in B_C$, as required.
3) $B_C \subseteq R$: We have from invariant (3) that $sig_{ssk_j}(p, B_C) \in E$ for some $j \leqslant t$. Now consider $x \in B_C$. Then $x \in t(D_j)$ by invariant (4). Hence $x \in t(E)$ by invariant (6), and so $\#\{k \mid 1 \leqslant k \leqslant n \wedge sk_k(x) \in E\} \geqslant t$ from the definition of $t(E)$, and hence $\#\{k \mid 1 \leqslant k \leqslant t \wedge sk_k(x) \in E\} \geqslant 2t - n$. Thus $x \in R$ as required.

Otherwise *c_dy*2 is matched by *skip*.

The other cases follow a similar pattern. Full proofs are provided in an extended version of this paper posted at [CS14].

This concludes the proof that $BBSpec \preccurlyeq BBProt$ with respect to $\{post, ack, publish\}$, establishing the correctness of the bulletin board, and in particular that it meets properties (bb.1)–(bb.4).

/* adversary bound invariant */
$E \cap (\{sk_k \mid k \leqslant t\} \cup \{ssk_k \mid k \leqslant t\}) = \emptyset$

$$\begin{array}{rcl} \bigcup_{e \in E} items(e) & \subseteq & E \\ \bigcup_{e \in E} sigs(e) & \subseteq & E \end{array}$$

where

$$\begin{array}{rcl} items(k) & = & \{\} \\ items(i) & = & \{i\} \\ items(sig_k(x)) & = & items(x) \\ items(\{m_1, \ldots, m_n\}) & = & \bigcup_i items(m_i) \\ items((p, m)) & = & items(m) \\ \\ sigs(k) & = & \{\} \\ sigs(i) & = & \{\} \\ sigs(sig_k(m)) & = & \{sig_k(m)\} \cup sigs(m) \\ sigs(\{m_1, \ldots, m_n\}) & = & \bigcup_i sigs(m_i) \\ sigs((p, m)) & = & sigs(m) \end{array}$$

Fig. 14. *BBProt*: invariant for adversary actions

V. Discussion

A. Summary

In this paper we have presented a distributed protocol for running a bulletin board using a number of peers, which can tolerate a number of them failing, in the context of a threat model which has the communication between the peers controlled by a Dolev-Yao adversary, who also is able to control some of the peers. This provides robustness and distributed trust: the bulletin board can tolerate some peers failing, and we require that a threshold of peers should be honest but no individual peer is required to be trusted. Provided a threshold of the peers behave according to the protocol, the key properties demanded of the bulletin board hold. In particular, only items posted to the bulletin board will be posted on it, any item whose receipt is acknowledged by the bulletin board must be posted on it, and the bulletin board will not accept conflicting items.

The development of the modelling and verification approach for this kind of protocol is also one of the contributions of this paper. Correctness has been established formally using the Event-B framework, using simulation to show that the protocol is a refinement of an idealised bulletin board which has the desired properties. The model included a Dolev-Yao attacker and the description of the protocol steps followed by the peers. Carrying out the proof identified some nondeterminism inherent in the protocol and enabled us to include it in the idealisation to document the possible behaviour of the implementation. In particular, an adversary can create a situation where he controls whether or not an unreceipted item appears on the bulletin board, and so this is reflected as nondeterminism at the abstract level. In the context of the vVote system this will not be an issue in practice, since the voting ceremony requires that any unreceipted items should be cancelled. Hence the nondeterminism will not affect the tallying of the election: either the cancellation appears alongside a vote, or it appears without the vote, and in both cases the vote will not be counted.

B. Related work

Other proposals for bulletin board implementations using a set of peers are given by Krummenacher [Kru10], by Peters [Pet05], and in the STAR-Vote system [BBB+13]. These proposals all provide for some robustness, but differ in terms of threat models and in the properties that they provide, none of them have been formally analysed, and none of them are suitable for the context of the system for the Victorian State Election. Krummenacher's proposal does not scale well with the number of votes to be posted, since any post requires a lock on the peers. Peters' approach uses a dynamic group membership protocol to manage the changing availability of peers, and the interaction of this protocol with the vote casting protocol is not fully understood or formally analysed. STAR-Vote is designed for use in a single polling station, and the emphasis is on using replication to detect incorrect behaviour rather than its automatic correction, and to resolve discrepancies forensically.

Byzantine Agreement Protocols: The general problem of achieving generalised agreement across components where some might fail in adverse ways is known as the *Byzantine Agreement* problem [LSP82], and there is an extensive literature on approaches to the problem [Lyn96]. Such protocols require correct behaviour in strictly more than $2/3$ of the peers, the same requirement as we have on our bulletin board peers. However, Byzantine Agreement protocols are not really suitable for items being posted. These protocols are typically synchronous and proceed in rounds, which would be too inefficient for receiving large quantities of votes: too many rounds and too much synchronisation overhead would be required to process each vote if we wish the peers to agree on the receipt of every vote. Furthermore, not all peers would necessarily be aware when a protocol run is starting, since they may not receive the initial item. Instead we have provided a protocol for the peers simply to send messages to each other and to respond to messages received in an asynchronous fashion. This means that the peers do not all need to agree on each vote. Our threat model is also different to the typical threat model for Byzantine agreement protocols: ours allows honest peers to be excluded from the acceptance of some items to the bulletin board, without being considered as dishonest, whereas Byzantine agreement protocols consider any non-participating peer as failing.

We are closer to the problem of Byzantine agreement in the commit and publish phase, since this is where the peers seek consensus to agree on a bulletin board to publish. Our optimistic and fallback protocols are indeed close to the Floodset protocol [Lyn96], a basic agreement protocol. Even in this case we do not require the full power of Byzantine agreement

protocols: the use of signatures minimises the ability of dishonest peers to introduce additional confusing information to disrupt the protocol run, and we can limit the adversarial behaviour simply to peers ceasing to communicate.

Acknowledgements

We are grateful to James Heather, Peter Ryan, Vanessa Teague and Douglas Wikström for useful discussions on Bulletin Board design and approaches to verification; to Gavin Lowe and Joshua Guttman for detailed discussions on the formal modelling and verification approach; and to Thierry Lecomte, Helen Treharne, John Derrick and Graeme Smith for discussion and advice on the application of B modelling and refinement. Thanks also to Olivier Pereira and Dan Wallach for clarifying aspects of STAR-Vote. This work was supported by the EPSRC Trustworthy Voting Systems project EP/G025797/1.

REFERENCES

[Abr10] Jean-Raymond Abrial. *Modeling in Event-B - System and Software Engineering*. Cambridge University Press, 2010.

[Adi06] Ben Adida. *Advances in Cryptographic Voting Systems*. PhD thesis, MIT Cambridge, July 2006.

[Adi08] Ben Adida. Helios: Web-based open-audit voting. In *USENIX Security Symposium*, pages 335–348, 2008.

[BBB+13] Susan Bell, Josh Benaloh, Michael D. Byrne, Dana DeBeauvoir, Bryce Eakin, Gail Fisher, Philip Kortum, Neal McBurnett, Julian Montoya, Michelle Parker, Olivier Pereira, Philip B. Stark, Dan S. Wallach, and Michael Winn. STAR-vote: A secure, transparent, auditable, and reliable voting star-vote: A secure, transparent, auditable, and reliable voting system. *USENIX Journal of Election Technology and Systems (JETS)*, 1(1), August 2013.

[BCH+12] Craig Burton, Chris Culnane, James Heather, Thea Peacock, Peter Y. A. Ryan, Steve Schneider, Sriramkrishnan Srinivasan, Vanessa Teague, Roland Wen, and Zhe Xia. A supervised verifiable voting protocol for the Victorian Electoral Commission. In *Proc. 5th International Conference on Electronic Voting*, 2012.

[BNFL+12] Jonathan Ben-Nun, Niko Fahri, Morgan Llewellyn, Ben Riva, Alon Rosen, Amnon Ta-Shma, and Douglas Wikström. A new implementation of a dual (paper and cryptographic) voting system. In *5th International Conference on Electronic Voting (EVOTE)*, 2012.

[CCC+10] Richard Carback, David Chaum, Jeremy Clark, John Conway, Aleksander Essex, Paul S. Herrnson, Travis Mayberry, Stefan Popoveniuc, Ronald L. Rivest, Emily Shen, Alan T. Sherman, and Poorvi L. Vora. Scantegrity II municipal election at Takoma Park: The first e2e binding governmental election with ballot privacy. In *Proc. USENIX Security*, 2010.

[CCM08] Michael R. Clarkson, Stephen Chong, and Andrew C. Myers. Civitas: Toward a secure voting system. In *IEEE Symposium on Security and Privacy*, pages 354–368, 2008.

[CGS97] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *Advances in Cryptology*, number 1233 in Lecture Notes in Computer Science, pages 103–118. Springer-Verlag, 1997.

[CRS05] D. Chaum, P.Y.A. Ryan, and S. Schneider. A practical, voter-verifiable election scheme. In *European Symposium on Research in Computer Security*, number 3679 in Lecture Notes in Computer Science. Springer-Verlag, 2005.

[CS14] Chris Culnane and Steve Schneider. A peered bulletin board for robust use in verifiable voting systems. arXiv:1401.4151, January 2014.

[DB01] John Derrick and Eerke Boiten. *Refinement in Z and Object-Z*. Springer, 2001.

[DY83] Danny Dolev and Andrew Chi-Chih Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–207, 1983.

[HL08] James Heather and David Lundin. The append-only web bulletin board. In *Formal Aspects in Security and Trust*, pages 242–256, 2008.

[Kru10] Roland Krummenacher. Implementation of a web bulletin board for e-voting applications. In *MSE Seminar on E-Voting*. Institute for Internet Technologies and Applications, 2010.

[KTV12] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Clash attacks on the verifiability of e-voting systems. In *IEEE Symposium on Security and Privacy*, pages 395–409, 2012.

[LSP82] Leslie Lamport, Robert E. Shostak, and Marshall C. Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.

[Lyn96] Nancy A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.

[MAV05] Christophe Métayer, Jean-Raymond Abrial, and Laurent Voisin. Event-B language, 2005. RODIN Project Deliverable 3.2, http://rodin.cs.ncl.ac.uk/deliverables/D7.pdf, accessed 7 May 2014.

[Nor13] Norwegian Ministry of Local Government and Regional Development. VALG: The e-vote trial, 2013.

[Pet05] R.A. Peters. A Secure Bulletin Board. Master's thesis, Technische Universiteit Eindhoven, 2005.

[RBH+09] Peter Y. A. Ryan, David Bismark, James Heather, Steve Schneider, and Zhe Xia. Prêt à Voter: a voter-verifiable voting system. *IEEE Transactions on Information Forensics and Security*, 4(4):662–673, 2009.

[Wag13] David Wagner. Remote voting: What can we do? Keynote Address, Electronic Voting Technology Workshop/Workshop on Trustworthy Elections, 2013.