ABC specification for Coq Proof Assistant

Karolina Grzeszkiewicz

January 2023

1 Assumptions and structure

Assumptions behind the model:

- We are modelling the protocol with respect to consensus and accountability properties, not communication complexity – hence we do not model the optimisations, but rather we focus on the main skeleton of the protocol. In particular, instead of having a preliminary round of light certificate exchange (which has lower communication complexity than a full certificate exchange and eliminates the need for full certificate exchange in case of no light certificate conflicts), and then a full certificate exchange, we model only the full certificate exchange.
- Public Key Cryptography is used to sign the value that a given node is submitting, and the signature can then be verified by any other node. We do not adopt any particular scheme, but rather take the secret keys of nodes and the sign and verify functions are parameters of the model, with the minimal requirement that a signature is valid if and only if it was produced with the sign function. We also assume that secret keys are unique, and through the protocol semantics we enforce that the secret keys of honest nodes are not shared.
- Certificates (full certificates) consist of a value, and pairs consisting of node address and signature which can be verified knowing the node's address (for the sake of simplicity, usually this is done with private keys)
- Our model describes the possible executions of a system with N processes, t Byzantine processes, and a function δ from processes to values accepted in the BFT, where N-t processes follow the Accountable Confirmer protocol, and t processes can exhibit arbitrary behaviour constrained by the network semantics and limitations imposed by the cryptography used
- The system is parametrised by a set of consensus parameters: (1) a set of all addresses of nodes in the network (2) a set of addresses that belong to byzantine processes (must be a subset of (1)) (3) a function δ from addresses to values (values accepted in the BFT)

2 Framework parameters and axioms

Parameters:

$$
\begin{aligned}
\mathsf{Addr} &\triangleq \mathbb{N} \\
\mathsf{NodeAddr} &\subseteq \mathsf{Addr} \\
\mathsf{ByzAddr} &\subseteq \mathsf{NodeAddr} \\
\mathsf{HonestAddr} &::= \mathsf{NodeAddr} \setminus \mathsf{ByzAddr} \\
\mathsf{N} &::= |\mathsf{NodeAddr}| \\
\mathsf{t_0} &::= \lceil \tfrac{N}{3} \rceil - 1 \\
\mathsf{t} &::= |\mathsf{ByzAddr}|
\end{aligned}
$$

(a) **Network parameters**

$$
\begin{aligned}
\mathsf{Value} &: \mathsf{eqType} \\
\mathsf{Key} &: \mathsf{eqType} \\
\mathsf{Signature} &: \mathsf{eqType} \\
\mathsf{value\_bft} &: \mathsf{Addr} \longrightarrow \mathsf{option\ Value} \\
\mathsf{verify} &: \mathsf{Value} \longrightarrow \mathsf{Signature} \longrightarrow \mathsf{NodeAddr} \longrightarrow \mathsf{Bool} \\
\mathsf{secret\_key} &: \mathsf{Addr} \longrightarrow \mathsf{Key} \\
\mathsf{sign} &: \mathsf{Value} \longrightarrow \mathsf{Key} \longrightarrow \mathsf{Signature}
\end{aligned}
$$

(b) **State parameters**

Figure 1: Framework parameters.

Note that *value_bft* represents the values decided via the BFT protocol. Then $value\_bft$ can be any function from node addresses to values provided that the following Byzantine consensus properties hold. Similarly, we define the necessary properties of the secret_key, sign, verify functions.

Axioms:

$$
\begin{aligned}
&\textit{bft\_termination} &&: \mathsf{t} \leq \mathsf{t_0} \implies \forall \mathsf{n} \in \mathsf{NodeAddr} \setminus \mathsf{ByzAddr}, \exists\, \mathsf{v} : \mathsf{Value}, \mathsf{value\_bft(n)} = \mathsf{Some\ v} \\
&\textit{bft\_agreement} &&: \mathsf{t} \leq \mathsf{t_0} \implies \forall \mathsf{n_1}, \mathsf{n_2} \in \mathsf{NodeAddr} \setminus \mathsf{ByzAddr}, \mathsf{value\_bft(n_1)} = \mathsf{value\_bft(n_2)} \\
&\textit{two\_correct\_nodes} &&: |\mathsf{HonestAddr}| \geq 2 \\
&\textit{authentication} &&: \forall\, v : \mathsf{Value}, sig : \mathsf{Signature},\ n \in \mathsf{NodeAddr}, \mathsf{verify}(v, sig, \mathsf{public\_key}(n)) \\
& && \quad \iff sig = \mathsf{sign}(v, \mathsf{secret\_key}(n)) \\
&\textit{uniqueness of secret keys} &&: \forall\, n_1,\ n_2 \in \mathsf{NodeAddr},\ n_1 \neq n_2 \implies \mathsf{secret\_key}(n_1) \neq \mathsf{secret\_key}(n_2) \\
&\textit{uniqueness of signatures} &&: \forall v_1,\ v_2\ : \mathsf{Value},\ n_1,\ n_2 \in \mathsf{NodeAddr},\ v_1 \neq v_2 \lor n_1 \neq n_2 \\
& && \quad \implies \mathsf{sign}(v_1, \mathsf{secret\_key}(n_1)) \neq \mathsf{sign}(v_2, \mathsf{secret\_key}(n_2))
\end{aligned}
$$

Figure 2: Axioms of the framework parameters.

BFT validity is not needed here since it asserts that the value decided by correct nodes must have been proposed by a correct node. This property is not relevant to the Accountable Confirmer algorithm.

3 Protocol semantics

System configurations:

$$\triangle \in \mathsf{GlobState} \triangleq \mathsf{Addr} \longrightarrow \delta$$

$$\mathsf{P} \in \mathsf{PacketSoup} \triangleq \mathcal{P}(\mathsf{Packet})$$

$$\sigma \in \mathsf{Config} \triangleq \mathsf{GlobState} \times \mathsf{PacketSoup}$$

Local states:

$$\delta \in \mathsf{LocalState} \triangleq \mathsf{Addr} \times \mathsf{Bool} \times \mathsf{Certificate} \times \mathcal{P}(\mathsf{Certificate})$$
$$c \in \mathsf{Certificate} \triangleq \mathsf{Value} \times \mathcal{P}(\mathsf{NodeAddr} \times \mathsf{Signature})$$

Messages and Packets:

$$\begin{aligned}
p \in \mathsf{Packet} &\triangleq \mathsf{Addr} \times \mathsf{Addr} \times \mathsf{Msg} \times \mathsf{Bool} \\
P_{rcv} &:= \{\langle \mathsf{src}\ p, \mathsf{dest}\ p, \mathsf{msg}\ p\rangle \mid p \in P \ \wedge\ \mathsf{received}\ p = \mathit{true}\} \\
P_{sent} &:= \{\langle \mathsf{src}\ p, \mathsf{dest}\ p, \mathsf{msg}\ p\rangle \mid p \in P\} \\
m \in \mathsf{Msg} &::= \mathsf{SubmitMsg}(\langle \mathsf{v} : \mathsf{Value},\ \mathsf{sig} : \mathsf{Signature}\rangle) \\
&\ \mid\ \ \mathsf{ConfirmMsg}(\mathsf{c} : \mathsf{Certificate})
\end{aligned}$$

4 Local Node Semantics

Receive step transitions: $\delta \xrightarrow{p}_{\rho} (\delta', ps)$

RCVSUBMIT

$$\frac{\begin{array}{c} \mathit{nsigs}' = (\mathsf{if}\ (v = v'\ \mathsf{and}\ \mathsf{verify}(v, \mathit{sig}, \mathsf{from})\ \mathsf{and\ not}\ \mathit{conf})\ \mathsf{then}\ \{\langle \mathit{from}, \mathit{sig}\rangle\} \cup \mathit{nsigs}\ \mathsf{else}\ \mathit{nsigs}) \\ \mathit{conf}' = |\mathit{nsigs}'| \geq \mathsf{N} - \mathsf{t}_0 \\ \mathit{ps} = (\mathsf{if}\ \mathit{conf}'\ \mathsf{then}\ \{\langle \mathit{this}, n, \mathsf{ConfirmMsg}\ \langle v, \mathit{nsigs}'\rangle\rangle \mid n \in \mathsf{NodeAddr}\}\ \mathsf{else}\ \emptyset \end{array}}{\langle \mathit{this}, \mathit{conf}, \langle v, \mathit{nsigs}\rangle, \mathit{certs}\rangle \xrightarrow{\langle \mathit{from}, \mathit{this}, \mathsf{SubmitMsg}\ \langle v', \mathit{sig}\rangle\rangle}_{\rho} (\langle \mathit{this}, \mathit{conf}', \langle v, \mathit{nsigs}'\rangle, \mathit{certs}\rangle, \mathit{ps})}$$

RCVCONFIRM

$$\frac{\forall \langle n, sig\rangle \in nsigs,\ \mathsf{verify}(v, sig, n)}{\langle this, conf, cert, certs\rangle \xrightarrow{\langle from,\ this,\ \mathsf{ConfirmMsg}\ \langle v,\ nsigs\rangle\rangle}_{\rho} (\langle this, conf, cert, \{\langle v,\ nsigs\rangle\} \cup certs\rangle, \emptyset)}$$

Internal step transitions: $\delta \longrightarrow_{\iota} (\delta', ps)$

INTSUBMIT

$$\frac{\begin{array}{c} sig = \mathsf{sign}(\mathsf{value\_bft}(this), \mathsf{secret\_key}(this)) \\ ps = \{\langle this, n, \mathsf{SubmitMsg}\ \langle \mathsf{value\_bft}(this), sig\rangle\rangle \mid n \in \mathsf{NodeAddr}\}\end{array}}{\langle this, conf, cert, certs\rangle \ \longrightarrow_{\iota}\ (\langle this, conf, cert, certs\rangle,\ ps)}$$

5 Network transitions

Network transitions: $\langle \Delta, P\rangle \overset{s}{\Longrightarrow} \langle \Delta', P'\rangle$

NETIDENTITY

$$\langle \triangle, P\rangle \overset{\mathsf{id}}{\Longrightarrow} \langle \triangle, P\rangle$$

NETDELIVER (p, δ)

$$\frac{\begin{array}{c} p \in P \qquad \mathsf{dest}\ p = a \\ \mathsf{received}\ p = false \qquad a \in \mathsf{HonestAddr} \qquad \Delta(a) = \delta \qquad \delta \xrightarrow{p}_{\rho} (\delta', ps)\end{array}}{\langle \Delta, P\rangle \overset{\mathsf{rcv}\ a}{\Longrightarrow} \langle \Delta[a \mapsto \delta'], P \setminus \{p\} \cup ps \cup \{\langle \mathsf{src}\ p, \mathsf{dest}\ p, \mathsf{msg}\ p, true\rangle\}\rangle}$$

NETDELIVER corresponds to a global state transition of delivering a randomly picked unread message p from packet soup P to a destination a with state δ, where a can be a must be a non-Byzantine address.

NETINTERNAL (a, δ)

$$\frac{\Delta(a) = \delta \qquad a \in \mathsf{HonestAddr} \qquad \delta \longrightarrow_{\iota} (\delta', ps)}{\langle \Delta, P\rangle \overset{\mathsf{int}}{\Longrightarrow} \langle \Delta[a \mapsto \delta'], P \cup ps\rangle}$$

This corresponds to a node taking an internal step from state δ to state δ', emitting packets, used for transitions not triggered by receiving any message, i.e. broadcasting submit messages.

NETBYZSUBMIT p

$$\frac{p : \text{Packet} \qquad \text{src}\, p \;\in\; \text{ByzAddr} \qquad \text{msg\_type}\,(\text{msg}\, p) \;=\; \text{MSubmit}}{\langle \triangle, P \rangle \xRightarrow{\text{byz}\, p} \langle \triangle, P \cup \{p\} \rangle}$$

Byzantine nodes can send submit messages with arbitrary values and signatures – even if the signature is invalid. By RCVSUBMIT honest nodes should ignore messages with invalid signatures.

NETBYZCONFIRM p

$$\frac{\begin{array}{c} p : \text{Packet} \qquad \text{src}\, p \;\in\; \text{ByzAddr} \\ \exists\, v : \text{Value},\; \mathit{nsigs} \in \mathcal{P}(\text{NodeAddr} \times \text{Signature}),\; \text{msg}\, p \;=\; \text{MConfirm}\, \langle v, \mathit{nsigs} \rangle \\ \forall\, \langle n, \mathit{sig} \rangle \;\in\; \mathit{nsigs},\; n \in \text{HonestAddr} \;\wedge\; \mathit{sig} = \text{sign}(v, \text{secret\_key}(n)) \\ \implies \exists\, n' \in \text{NodeAddr}, b : \text{Bool},\; \langle n, n',\; \text{MSubmit}\, \langle v, \mathit{sig} \rangle,\; b \rangle \in P \end{array}}{\langle \triangle, P \rangle \xRightarrow{\text{byz}\, p} \langle \triangle, P \cup \{p\} \rangle}$$

The confirm messages emitted by byzantine nodes are subject to some constraints due to the fact that they can't forge the signatures of other nodes without knowing their secret keys. Now we assume that the honest nodes follow the given Public Key Encryption protocol by not sharing their private keys, but we can't expect the byzantine nodes to keep their secret keys secret. For instance, the byzantine nodes might be collaborating and intentionally sharing their secret keys with each other, or simply be dismissive of the rules of secure communication for any other reason. Hence, we only require that for all signatures of honest nodes in the certificate prepared by a byzantine node, if the signature for a given value is valid, then there is a corresponding submit message in the packet soup with that signature and value (either sent to the preparing byzantine node, or to any other node as the byzantine node might have been eavesdropping on the communication channel and storing signatures of other nodes). Yet there are no requirements on signatures of byzantine nodes (as they might be forged or correspond to messages in the packet soup).'

6 Properties of the Accountable Confirmer

Terminating Convergence:

If the number of faulty processes does not exceed t_0 and all correct processes submit the same value, then the value is eventually confirmed by every correct process.

This is a liveness property so we do not prove it.

Agreement:

If the number of faulty processes does not exceed t_0, then no two correct processes confirm different values.

The proof of this property is trivial since if $t \leq t_0$ then all honest processes submit the same value, and an honest process can only confirm the same value it has submitted.

Validity:

Value confirmed by a correct process was submitted by a correct process.

Also holds trivially for the same reason as above.

Accountability:

If two correct processes confirm different values, then every correct process eventually detects at least $t_0 + 1$ faulty processes and obtains a proof of culpability of all detected processes.

$$\begin{aligned}
&\exists\, a_1, a_2 \in \mathsf{HonestAddr}, \mathsf{confirmed}\ \triangle(a_1)\ \wedge\ \mathsf{confirmed}\ \triangle(a_2)\ \wedge\ \mathsf{value\_bft}\ a_1 \neq \mathsf{value\_bft}\ a_2 \\
&\implies \forall\, a \in \mathsf{HonestAddr}, \\
&|certs\ \triangle(a)| \geq \mathsf{N} - \mathsf{t}_0 \wedge proof\ \triangle(a) \neq \emptyset \implies |proof(\triangle(a))| \geq \mathsf{t}_0 + 1
\end{aligned}$$

It is a liveness property and the formulation above does not assert anything interesting so we do not prove it.

Accountability – Soundness:

If a correct process obtains a proof of culpability of another processes, then the detected process is byzantine.

$$\begin{aligned}
\mathit{AccSoundness}(\langle \triangle, P \rangle) := \forall\, n \in \mathsf{NodeAddr},\ h \in \mathsf{HonestAddr}, \\
n \in proof(\triangle(h)) \implies n \in \mathsf{ByzAddr}
\end{aligned}$$

This is the property that we want to prove.

Accountability – Soundness':

If a correct process obtains a proof of culpability of another processes, then the detected process has behaved byzantine.

$$\mathit{AccSoundness}(\langle \triangle, P\rangle) := \forall\, n \in \mathsf{NodeAddr},\ h \in \mathsf{HonestAddr},$$
$$n \in \mathit{proof}(\triangle(h)) \implies \mathsf{behavedByz}(n, P)$$

where:

$\mathsf{behavedByz}(n, P) :=$

$\exists\, v_1,\ v_2 : \mathsf{Value}, \mathit{sig}_1, \mathit{sig}_2, a_1, a_2 \in \mathsf{NodeAddr},$

$\wedge\, v_1 \neq v_2$

$\wedge\, \langle n,\ a_1, \mathsf{SubmitMsg}\langle v_1, \mathit{sig}_1\rangle, \mathsf{true}\rangle \ \in P$

$\wedge\, \langle n,\ a_2, \mathsf{SubmitMsg}\langle v_2, \mathit{sig}_2\rangle, \mathsf{true}\rangle \ \in P$

$\wedge\, \mathit{sig}_1 = \mathsf{sign}(v_1, \mathsf{secret\_key}(n))$

$\wedge\, \mathit{sig}_2 = \mathsf{sign}(v_2, \mathsf{secret\_key}(n))$

This is a stronger notion of accountability, which does not necessarily hold. It is violated by the "pirate lied about another pirate" scenario in which one of the conflicting confirm messages was fabricated by a byzantine node *b*, and the signature of the byzantine node *n* in the certificate was also forged by *b* (given that *b* might have had access to *n*'s secret key) rather than being taken from a corresponding submit message in the Packet Soup.

7 Invariant

$$
\begin{aligned}
&\textsc{Inv}(\langle \triangle, P\rangle) := \\
&\qquad \wedge\ \forall\ \mathit{certs} \in P(\text{Certificate}), n \in \text{proof}(\mathit{certs}), \\
&\qquad\quad \exists v_1, v_2 : \text{Value}, \mathit{sig}_1, \mathit{sig}_2 \in \text{Signature}, \mathit{nsigs}_1, \mathit{nsigs}_2 \in P(\text{NodeAddr} \times \text{Signature}), \\
&\qquad\quad\ \wedge \langle v_1, \mathit{nsigs}_1\rangle \in \mathit{certs} \\
&\qquad\quad\ \wedge \langle v_2, \mathit{nsigs}_2\rangle \in \mathit{certs} \\
&\qquad\quad\ \wedge v_1 \neq v_2 \\
&\qquad\quad\ \wedge \langle n, \mathit{sig}_1\rangle \in \mathit{nsigs}_1 \\
&\qquad\quad\ \wedge \langle n, \mathit{sig}_2\rangle \in \mathit{nsigs}_2 \\
&\qquad \wedge\ \forall\ h\ \in \text{Honest}, v : \text{Value}, \mathit{nsigs} \in P(\text{NodeAddr} \times \text{Signature}), \\
&\qquad\quad \langle v, \mathit{nsigs}\rangle \in \mathit{certs}(\triangle(h)) \\
&\qquad\quad\ \implies \exists n' \in \text{NodeAddr}, \langle n', h, \text{ConfirmMsg}\ \langle v, \mathit{nsigs}\rangle\rangle \in P_{rcv} \\
&\qquad\qquad\quad \wedge \forall \langle n, \mathit{sig}\rangle \in \mathit{nsigs},\ \mathit{sig} = \text{sign}(v, \text{secret\_key}(n)) \\
&\qquad \wedge\ \forall\ h\ \in \text{Honest}, n \in \text{NodeAddr}, v : \text{Value}, \mathit{nsigs} \in P(\text{NodeAddr} \times \text{Signature}), \\
&\qquad\quad \langle h, n, \text{ConfirmMsg}\ \langle v, \mathit{nsigs}\rangle\rangle \in P_{sent} \implies \mathit{cert}(\triangle(h)) = \langle v, \mathit{nsigs}\rangle \\
&\qquad \wedge\ \forall\ h\ \in \text{Honest}, v : \text{Value}, \mathit{nsigs} \in P(\text{NodeAddr} \times \text{Signature}), \\
&\qquad\quad \mathit{cert}(\triangle(h)) = \langle v, \mathit{nsigs}\rangle \\
&\qquad\quad\ \implies \forall \langle n, \mathit{sig}\rangle \in \mathit{nsigs}, \langle n, h, \text{SubmitMsg}\ \langle v, \mathit{sig}\rangle\rangle \in P_{rcv}\ \wedge\ \mathit{sig} = \text{sign}(v, \text{secret\_key}(n)) \\
&\qquad \wedge\ \forall b \in \text{ByzAddr}, h \in \text{NodeAddr}, v : \text{Value}, \mathit{nsigs} \in P(\text{NodeAddr} \times \text{Signature}), \\
&\qquad\quad \langle b, h, \text{ConfirmMsg}\ \langle v, \mathit{nsigs}\rangle\rangle \in P_{sent} \\
&\qquad\quad\ \implies \forall \langle n, \mathit{sig}\rangle \in \mathit{nsigs}, \\
&\qquad\qquad\quad n \in \text{HonestAddr}\ \wedge\ \mathit{sig} = \text{sign}(v, \text{secret\_key}(n)) \\
&\qquad\qquad\quad\ \implies \exists n' \in \text{NodeAddr},\ \langle n, n', \text{SubmitMsg}\ \langle v, \mathit{sig}\rangle\rangle \in P_{sent} \\
&\qquad \wedge\ \forall\ h\ \in \text{Honest}, n_1, n_2 \in \text{NodeAddr}, \mathit{sig}_1, \mathit{sig}_2 \in \text{Signature}, v_1, v_2 : \text{Value}, \\
&\qquad\quad (\langle h, n_1, \text{SubmitMsg}\ \langle v_1, \mathit{sig}_1\rangle\rangle \in P_{sent}\ \wedge\ \mathit{sig}_1 = \text{sign}(v_1, \text{secret\_key}(h)) \\
&\qquad\quad\ \wedge\ \langle h, n_2, \text{SubmitMsg}\ \langle v_2, \mathit{sig}_2\rangle\rangle \in P_{sent}\ \wedge\ \mathit{sig}_2 = \text{sign}(v_2, \text{secret\_key}(h))) \\
&\qquad\quad\ \implies\ v_1 = v_2 \\
&\qquad \wedge \forall p : \text{Packet},\ p \in P_{rcv} \implies p \in P_{sent} \\
&\qquad \wedge \forall n \in \text{NodeAddr},\ n \notin \text{HonestAddr} \implies n \in \text{ByzAddr}
\end{aligned}
$$

8 Other potentially useful properties

$$
\begin{aligned}
&\mathsf{CertIntersect}(\langle \triangle, P\rangle) := \\
&\qquad \forall\, a_1, a_2 \in \mathsf{NodeAddr} \setminus \mathsf{ByzAddr}, \\
&\qquad \mathsf{confirmed}(a_1) \land \mathsf{confirmed}(a_2) \\
&\qquad\; \implies |\mathsf{nodes}(\mathsf{certificate}(\mathsf{a}_1)) \cap \mathsf{nodes}(\mathsf{certificate}(\mathsf{a}_2))| \geq \mathsf{t}_0 + 1
\end{aligned}
$$