UNIVERSITY OF Southampton

Verifying System-level Security of a Smart Ballot Box

Dana Dghaym, Thai Son Hoang, Michael Butler, Runshan Hu, Leonardo Aniello and Vladimiro Sassone

ABZ 2021 - virtual, 9th June 2021

UNIVERSITY OF Southampton

Outline

- Motivation
- Case study: Smart Ballot Box
- Rigid Events and Parameters
  - Preserving availability property during refinement
- Event-B System Model of the Smart Ballot Box
- Conclusions and Future Work

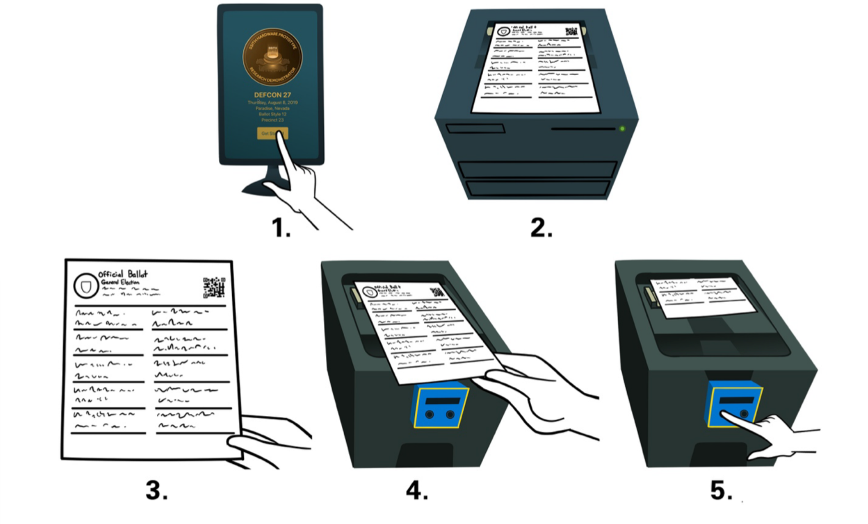
Motivation

- Application of refinement-based formal modelling in building a Correct-by-Construction secure system.
- Refinement of the availability property of secure systems.
- Overall Aim of case study: show how the Smart Ballot Box can be correctly implemented on capability hardware according to the system-level security specification.

Case Study: Smart Ballot Box

- Key Function of SBB:
  - Ensures only valid ballot papers are cast in ballot boxes for later tabulation.

- Security Properties:
  - Confidentiality, integrity and availability.



Galois and Free & Fair. The BESSPIN Voting System (2019).

Rigid Events and Parameters

- Event availability in Event-B
  - Determined by its enabledness condition .
  - Guard strengthening can affect event availability during refinement.
- Extend the notion of event enabledness to include parameters, given event *e* we define enabledness:

```
event e
any  p, q
where  G(p,q)
then … end
```

$$\boxed{\text{Enabled}_p(e) \overset{\text{def}}{=} \exists\, q \,.\, G(p, q)}$$

- We call events we are interested in their **availability** with respect to *p* rigid events & *p* are the rigid parameters.

Rigid Events and Parameters (2)

- Textual Representation: Event *e* must be enabled for any parameter *rp* satisfying $\boldsymbol{Enabled}_{rp}(e)$

```
event [e]
any  [rp] op  where  Ga(rp, op) then…end
```

- Syntactic Rules:

  1. Rigid events can only be refined by *rigid* events
  2. The abstract *rigid* parameters must be retained in the concrete events

- In general, more rigid parameters can be introduced in later refinements, but they will only be relevant to *proof* in further refinements.

Preserving Availability through Refinement

- Preserve availability property through refinement by:
  - Proving that the concrete event does not strengthen the enabledness of the abstract event, we propose *enabledness* PO: ***ENBL***

$$\boxed{I(v), J(v, w), Ga(rp, oap, v) \vdash \exists\ ocp\ .\ Gc(rp, ocp, v, w)}$$

```
event [ae]
any  [rp] oap
where  Ga(rp, oap, v)
then
// abstract actions
end
```

```
event [ce]
any  [rp] ocp
where  Gc(rp, ocp, v, w)
then
// concrete actions
end
```

Preserving Availability (2)

- In Event-B an abstract event can be refined by a group of concrete events ce_i $(i \in 1..n)$.
- ***ENBL*** PO can be generalized as follows where ocp_i and Gc_i are the concrete events and guards of ce_i

$$\boxed{\forall\, rp, oap \,.\, Ga(rp, oap) \Rightarrow \vee_i\, (\exists\, ocp_i \,.\, Gc_i(rp, ocp_i))}$$

SBB System Model: Refinement Strategy

0. Abstract level: Model an ideal voting system.

1. Model possible attackers' behavior by distinguishing between different types of ballot papers.

2. Introduce time and invalidate ballots with expired timestamps.

   - Time can be the subject of more attacks.

3. Data refine the voter information by encrypting ballots.

4. Ensure the legitimacy of ballots through the Message Authentication Code (MAC).

SBB Model: Abstract Level

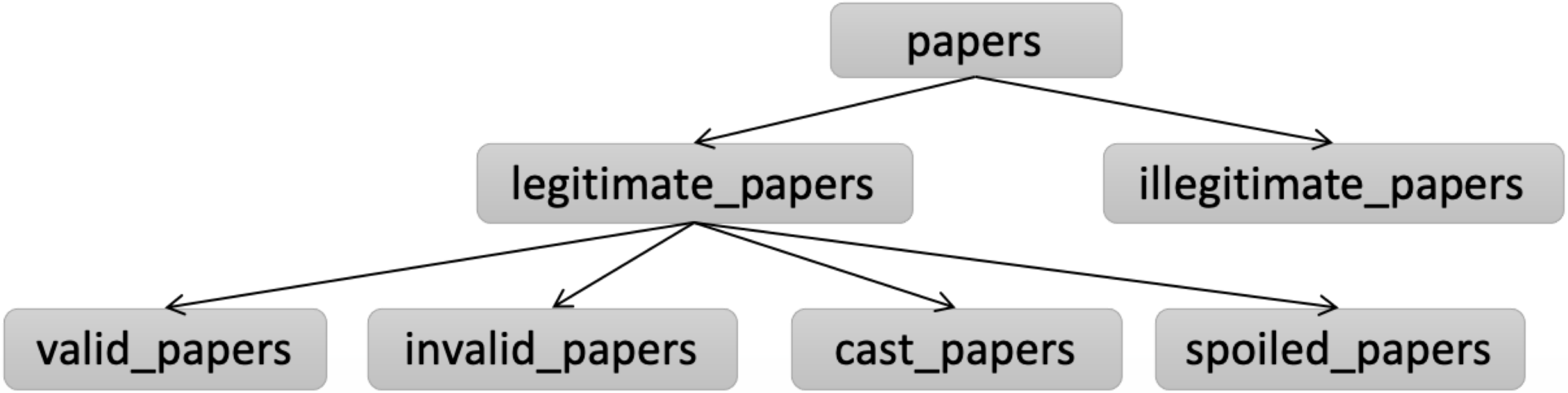
- Events: create_ballot, cast ballot, invalidate_ballot
- Model an ideal voting system
  - Each voter can have at most one legitimate ballot

$$ballots \in VOTER \rightarrowtail\!\!\!\!\!\!\!\!\!+\;\; VOTE$$

  - The cast ballots must be legitimate

$$cast \subseteq ballots$$

First Refinement: Ballot types



- Possible attacks
  - – Attacker create ballot/duplicate valid ballot ..
- Model the main security properties of SBB

1. Accept all valid ballots
2. Reject invalid ballots

First Refinement: Availability Property

- Availability property: Ensure valid ballots are not blocked from being cast.

- Availability property is captured by the guard of the relevant events.

  – Specify *cast_paper* as rigid event with *paper* as rigid parameter .

```
event [cast_paper] refines cast_ballot
any [paper] where
  @valid-paper: paper ∈ valid_papers
then
// actions for casting a ballot
end
```

Second Refinement: Time & Availability

```
event cast_paper refines cast_paper
any paper where
@typeof-paper: paper ∈ papers
// paper not already expired
// copy not already cast
// copy not already spoiled
// paper is not illegitimate
then
// cast the paper actions
end
```

Encoding *ENBL* PO as a theorem

```
theorem @accept-valid-paper:
∀ paper · paper ∈ valid_papers ⇒
// paper not already expired
 paper_time(paper) ≥ current_time −
expiry_duration
// copy not already cast
∧ paper_voter(paper) ∉
paper_voter[cast_papers]
// copy not already spoiled
∧ (∀ sp · sp ∈ spoiled_papers ⇒
paper_voter(paper) ≠ paper_voter(sp)
∨ paper_vote(paper) ≠ paper_vote(sp)
∨ paper_time(paper) ≠ paper_time(sp)
)
// paper is not illegitimate
∧ paper ∉ illegitimate_papers
```

Third Refinement: Ballot Encryption

- Introduce encryption to prevent SBB from accessing the voter's information.
    - Apply data refinement to replace *paper_voter* and *paper_vote* with encrypted ballot
- Prove ***ENBL*** PO due to *cast_paper* guards update as a result of refinement.

```
theorem @accept-valid-paper:
∀ paper · paper ∈ valid_papers ⇒
 paper_time(paper) ≥ current_time −
expiry_duration
// copy not already cast
∧ paper_encrypted_ballot(paper) ∉
paper_encrypted_ballot[cast_papers]
// copy not already spoiled
∧ (∀sp · sp ∈ spoiled_papers ⇒
paper_encrypted_ballot(paper) ≠
paper_encrypted_ballot(sp) ∨
paper_time(paper) ≠ paper_time(sp)
)
∧ paper ∉ illegitimate_papers
```

Fourth Refinement: Ballot Authentication

- Introduce MAC to check the legitimacy of the source issuing the ballot.
  - We assume the attacker does not know the secret key; therefore, it is crucial to ensure the secrecy of this key.

```
 @mac-legitimate_papers: ∀paper · paper ∈ legitimate_papers ⇒
paper_mac(paper) = MACAlgorithm(
paper_time(paper) ↦ paper_encrypted_ballot(paper) ↦ MACKey
)
```

- The guards of *cast_paper* and The ***ENBL*** PO will be updated accordingly.

Conclusions and Future Work

- Availability property of an event can be ensured through refinement by preserving the enabledness of its corresponding refined events.
  - A general PO (***ENBL***) that can be applied to any event with rigid parameters is provided.
- **Future Work**
  - Semantics model to justify the soundness of the rigid property of events
  - Tool support for the ENBL PO in Rodin: CamilleX

UNIVERSITY OF Southampton

Thank you

Questions?

This work is supported by the HD-Sec project, which was funded by the Digital Security by Design (DSbD) Programme delivered by UKRI to support the DSbD ecosystem.

Visit https://hd-sec.github.io for more information on the HD-Sec project.