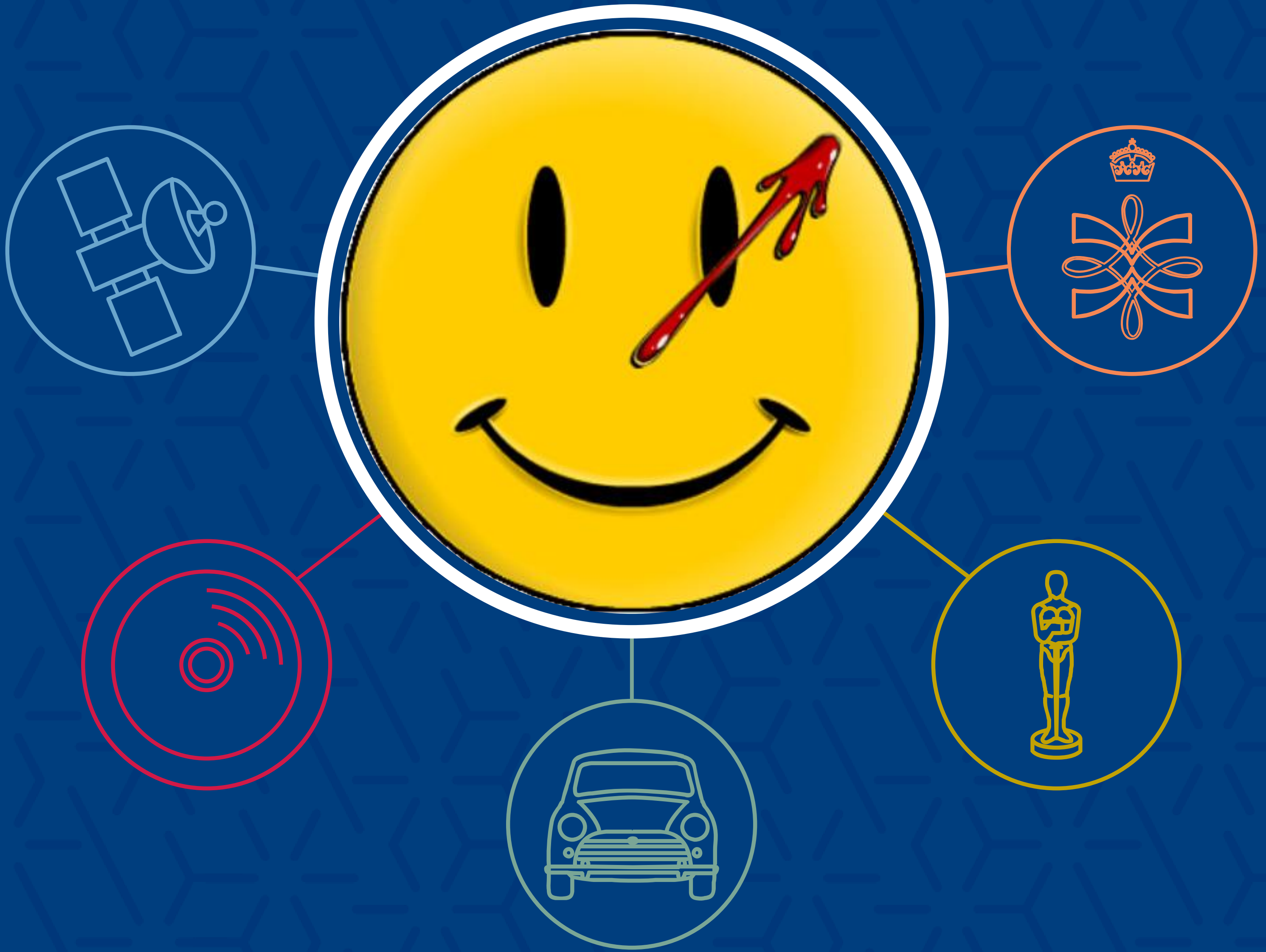
A (somewhat) gentle introduction to machine-checked cryptographic proofs

Privacy and Verifiability for online voting



François Dupressoir

joint work with V Cortier, CC Drăgan, B Warinschi, P-Y Strub, B Schmidt, B Grégoire, and others

UNIVERSITY OF SURREY

Online Voting: Some Context

» Electronic Voting is already widely used
  - Non-governmental elections and votes
  - Legally-binding decision-making (Switzerland, Estonia, …)

» Problem: attacks to *privacy* and *integrity* scale up
  - Traditional approaches (observe and audit) are insufficient

» Solution: Throw *crypto* in and mix
  - Prove strong privacy and integrity guarantees, under…
  - … standard cryptographic assumptions, and …
  - … simple trust assumptions.

What is Online Voting? (Syntax)

- The *election authority* sets up the election~~, generates the voter roll, checks eligibility, …~~
  - Modelled as a **Setup** algorithm
- The *voters* cast their votes, and later may want to check them
  - Modelled as a **Vote** algorithm
- The *ballot box* receives ballots
  - Modelled as a **Valid** algorithm (+ a chunk of state)
- The *bulletin board* holds a public view of the ballots received and other verifiability evidence
  - Modelled as a **Publish** algorithm (+ a chunk of state)
- The *trustees* compute the tally from ballots held within the ballot box
  - Modelled as a **Tally** algorithm
- The *general public* may want to check the good conduct of the election
  - Modelled as a **Verify** algorithm

A Typical (secure) Online Voting System

» **Setup** generates a keypair for the election
  - Usually shared between trustees so that a threshold of them need to collaborate to decrypt

» **Vote** encrypts voter choice under the election public key, may protect the integrity of the ballot

» **Valid** typically prevents direct replay of encrypted ballots, rejects ill-formed ballots
  - May prevent revotes, …

» **Publish** typically selects a subset of information to publish
  - May publish nothing at all (no verifiability)

» **Verify** checks that tallying was performed correctly

Two Ways of Tallying

» Homomorphically

- **Vote** uses (partially) homomorphic encryption
- **Tally** computes homomorphically over ciphertext to get encrypted result
- A threshold of trustees decrypt the result once they agree tallying is finished, and produce a NIZK proof of correct decryption

» Using mix-nets

- **Vote** uses re-randomizable encryption
- A network of mix-servers sequentially re-randomize the ciphertexts after shuffling them, producing NIZK proofs of correct shuffling
- A threshold of trustees decrypt individual ballots once they agree shuffling is finished
- **Tally** can then be performed publicly

» We want tally-agnostic definitions for privacy and verifiability

Defining and Proving Privacy for Online Voting

Take 1

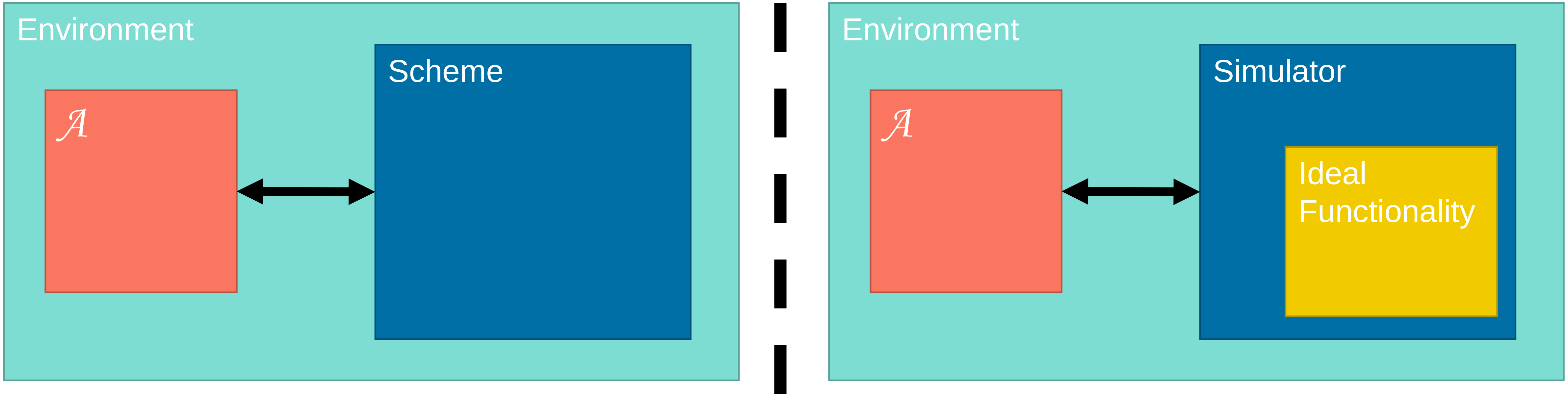
Defining Privacy I

» Long history of bad game-based definitions

» Ideally, want guarantees as strong as those given by a true Trusted Third Party (Simulation-Based Security)

- There exists a simulator such that no adversary can distinguish the scheme from the simulator

» Simulations are really hard to deal with

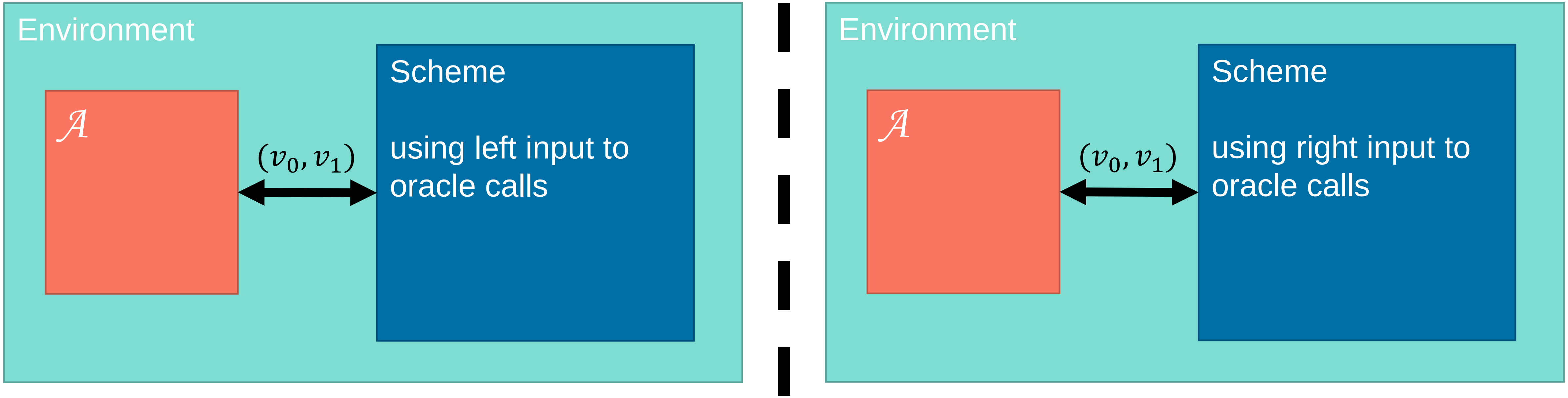


Defining Privacy II

» Bernhard, Cortier, Galindo, Pereira and Warinschi (S&P 15) define BPRIV
  - Prove that, with two simpler conditions, it implies simulation-based privacy

» BPRIV is “mostly” game-based
  - Easy to manipulate and instantiate

» BPRIV is a “Left or Right” game



Defining Privacy III (BPRIV)

- **Vote** oracle is the only one that is made Left or Right
  - Voter choice is the only thing whose privacy we care to protect
- The adversary is additionally given the ability to form and cast ballots without using **Vote**
  - Models voters who may be under adversary control
- In the Left game, everything works as expected:
  - Simply run the scheme using the left input to **Vote** oracle queries
- In the Right game, things get ~~complicated~~ fun:
  - Maintain Ballot Boxes corresponding to both sets of inputs
  - **Publish** gives the public bulleting board produced from the Right box
  - **Tally** computes the result using the Left box,
and *simulates* a proof that the result was computed correctly from the Right box

Defining Privacy IV (BPRIV, a formal view)

$\mathsf{Exp}^{\mathsf{bpriv},\beta}_{\mathcal{A},\mathcal{V},\mathsf{Sim}}(\lambda, m)$

$$\mathsf{BB}_0, \mathsf{BB}_1 \leftarrow [\,]$$
$$(\mathsf{pk}, \mathsf{sk}, \mathsf{uL}) \leftarrow \mathsf{Setup}(1^\lambda, m)$$
$$\beta' \leftarrow \mathcal{A}^{\mathcal{O}}(1^\lambda, \mathsf{pk}, \mathsf{uL})$$
$$\textbf{return } \beta'$$

Oracle $Otally()$ for $\beta = 0$

$$(r, \Pi) \leftarrow \mathsf{Tally}(\mathsf{BB}_0, \mathsf{sk})$$
$$\textbf{return } (r, \Pi)$$

Oracle $Otally()$ for $\beta = 1$

$$(r, \Pi) \leftarrow \mathsf{Tally}(\mathsf{BB}_0, \mathsf{sk})$$
$$\Pi' \leftarrow \mathsf{Sim}(\mathsf{pk}, \mathsf{Publish}(\mathsf{BB}_1), r)$$
$$\textbf{return } (r, \Pi')$$

Oracle $Ocast(b)$

$$\textbf{if } \big(\mathsf{Valid}(\mathsf{BB}_\beta, \mathsf{uL}, b, \mathsf{pk})\big) \textbf{ then}$$
$$\quad \mathsf{BB}_0 \leftarrow \mathsf{BB}_0 + [b];\ \mathsf{BB}_1 \leftarrow \mathsf{BB}_1 + [b]$$

Oracle $Oboard()$

$$\textbf{return } \mathsf{Publish}(\mathsf{BB}_\beta)$$

Oracle $Ovote(id, v_0, v_1)$

$$\ell \leftarrow \mathsf{uL}[id]$$
$$\textbf{if } (\ell \neq \perp) \textbf{ then}$$
$$\quad b_0 \leftarrow \mathsf{Vote}(id, v_0, \ell, \mathsf{pk});\ b_1 \leftarrow \mathsf{Vote}(id, v_1, \ell, \mathsf{pk})$$
$$\quad \textbf{if } \big(\mathsf{Valid}(\mathsf{BB}_\beta, \mathsf{uL}, b_\beta, \mathsf{pk})\big) \textbf{ then}$$
$$\quad\quad \mathsf{BB}_0 \leftarrow \mathsf{BB}_0 + [b_0];\ \mathsf{BB}_1 \leftarrow \mathsf{BB}_1 + [b_1]$$

Proving Privacy?

- 2 *published* attempts at proving BPRIV for Helios-like protocols
  - Both had minor issues and a significant gap
- Zero-Knowledge proofs evidence a mathematical relation between a (secret) witness and a (public) statement
- The language of valid statements should be in NP
- In our case, the statement talks about the random oracle
  - If we make it stateless, not in NP
  - If we make it stateful, need new theory
- This was never highlighted as an issue…

A New Problem

» Nobody understands cryptographic proofs
  - Hard to write, but *even harder to read*

» Formalize the proof in EasyCrypt
  - Introduces an asymmetry between proof writer and proof reader
  - Removes focus from the proof itself, and
  - Allows evaluator to focus on definitions and claims

» Key insight:
  - All crypto security notions (even simulation-based) are post-conditional equivalences between open probabilistic programs
  - Relational reasoning is well suited

» EasyCrypt allows us to dive below program logics and into semantics

You Know What we Did Last Summer (I)

» We formalize BPRIV and its associated properties for Labelled MiniVoting (Berhnard et al.)

$\mathsf{Setup}(1^\lambda, m)$

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda)$
$\textbf{for } i \textbf{ in } 1..m \textbf{ do}$
$\quad id \leftarrow_{\$} \mathsf{ID}$
$\quad \mathsf{uL}[id] \leftarrow \mathsf{Flabel}(id)$
$\textbf{return } (\mathsf{pk}, \mathsf{sk}, \mathsf{uL})$

$\mathsf{Vote}(id, \ell, v, \mathsf{pk})$

$c \leftarrow \mathsf{Enc}(\mathsf{pk}, \ell, v)$
$\textbf{return } (id, \ell, c)$

$\mathsf{Valid}(\mathsf{BB}, \mathsf{uL}, b, \mathsf{pk})$

$(id, \ell, c) \leftarrow b$
$e_1 \leftarrow \forall id'.\ (id', \ell, c) \notin \mathsf{BB}$
$e_2 \leftarrow (\ell = \mathsf{uL}[id])$
$e_3 \leftarrow \mathsf{ValidInd}(b, \mathsf{pk})$
$\textbf{return } (e_1 \wedge e_2 \wedge e_3)$

$\mathsf{Tally}(\mathsf{BB}, \mathsf{sk})$

$dbb = [\,]$
$\textbf{for } i \textbf{ in } 1..|\mathsf{BB}| \textbf{ do}$
$\quad (id, \ell, c) = \mathsf{BB}[i]$
$\quad dbb[i] \leftarrow (id, \mathsf{Dec}(\mathsf{sk}, \ell, c))$
$r \leftarrow \rho(dbb)$
$pbb \leftarrow \mathsf{Publish}(\mathsf{BB})$
$\Pi \leftarrow \mathsf{P}((\mathsf{pk}, pbb, r), (\mathsf{sk}, \mathsf{BB}))$
$\textbf{return } (r, \Pi)$

» **Tally** is trusted, which we really do not want in practice

» We generalize over previous definitions by parameterizing the scheme

- **Flabel**, **ValidInd**, ρ, $\mathcal{R}$

You Know What we Did Last Summer (II)

Electric Boogaloo

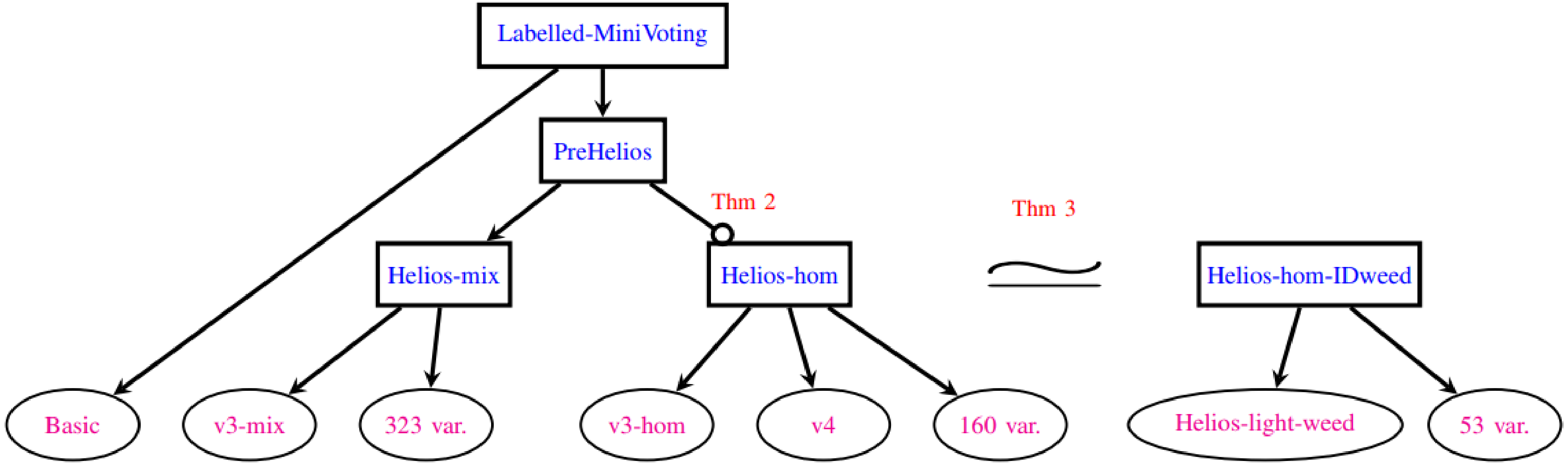
- By verifiably secure refinement, we transfer the security of “Labelled MiniVoting” to:
  - Mixnet-based Helios (Helios v3-mix)
  - Homomorphic Helios (Helios v3-hom, Helios v4)
- By observational equivalence, we further transfer the privacy result to a previously unproved optimized version of Helios v4
- By verifiable instantiation, we obtain machine-checked privacy proofs for *over 500 variants of Helios*

$\mathsf{Valid}(\mathsf{BB}, \mathsf{uL}, b, \mathsf{pk})$

$$(id, \ell, c) \leftarrow b$$
$$e_1 \leftarrow \forall id'.\ (id', \ell, c) \notin \mathsf{BB}$$
$$e_2 \leftarrow (\ell = \mathsf{uL}[id])$$
$$e_3 \leftarrow \mathsf{ValidInd}(b, \mathsf{pk})$$
$$\mathbf{return}\ (e_1 \wedge e_2 \wedge e_3)$$

$\mathsf{ValidLight}(\mathsf{BB}, \mathsf{uL}, b, \mathsf{pk})$

$$(id, \ell, c) \leftarrow b$$
$$e_1 \leftarrow (b \notin \mathsf{BB})$$
$$e_2 \leftarrow (\ell = \mathsf{uL}[id])$$
$$e_3 \leftarrow \mathsf{ValidInd}(b, \mathsf{pk})$$
$$\mathbf{return}\ (e_1 \wedge e_2 \wedge e_3)$$



Verification Effort

- » About 1 person-year from start of project to “final” qed
  - Includes false starts, time for RA to learn both the crypto and formal tools
  - Roughly 1,500 lines of definitions (includes named variants)
  - Roughly 14,000 lines of proof (includes named variants)
  - Unnamed variants are automatically generated – ~150LoD, ~500LoP each

- » Initial proof for Labelled MiniVoting obtained about 75% of the effort in
  - But later iterations needed to extend treatment of proofs over relations that include random oracles

- » Identified a missing assumption in published proofs of Labelled MiniVoting
  - Does not affect practical security, since it is discharged on concrete instantiations

Defining and Proving Verifiability for Online Voting

Take 1

Verifiability in Online Voting

- We want verifiability
  - If something goes wrong, anywhere, we want to know it
- We want it with minimal trust assumptions
  - Force at least two parties to collude to subvert the election without detection

- End-to-End verifiability relies on:
  - Individual Verifiability: individual voters should be able to verify that their vote was both *cast as intended*, and *recorded as cast*
  - Universal Verifiability: anyone should be able to verify that all votes were *counted as recorded*

- But end-to-end verifiability does not prevent ballot stuffing
  - A malicious ballot box can just stick ballots in for voters who haven't voted

Belenios (Cortier et al. 2014)

- To prevent ballot stuffing, voters need cryptographic credentials
- A registrar manages a mapping from eligible voters to their public keys
  - The registrar does not maintain the voter roll – that *must* be trusted
- Signed ballots are signed by voters before being cast
  - A malicious ballot box cannot stuff ballots, as it doesn't have the voters' signing keys
- For privacy, signatures must be stripped before tallying
- Cortier et al. provide proofs of privacy and verifiability

Machine-Checked Privacy for Belenios

» Expectation:

1. Add registration and signing in Helios definitions and proofs
2. Run EasyCrypt
3. Minimally fiddle with proof
4. Profit

» Reality:

1. Add registration and signing in Helios definitions and proofs
2. Run EasyCrypt
3. ??
4. Wat?

Privacy for Belenios: No Dynamic Corruption

» Dynamic corruption allows adversary to replay an honest ballot to learn the vote it contains

$\beta = 0$

$$\mathsf{BB}_0 = [(id, b_0), (id, b_0)]$$

$$(v_0, \pi) \leftarrow \mathsf{Tally}(\mathsf{BB}_0)$$

Adversary sees board BB_0, result v_0, and real proof π

$\beta = 1$

$$\mathsf{BB}_1 = [(id, b_1), (id, b_1)]$$

$$\mathsf{BB}_0 = [(id, b_0), (id, b_1)]$$

$$(v_1, \pi) \leftarrow \mathsf{Tally}(\mathsf{BB}_0)$$

Adversary sees board BB_1, result v_1, and sim. proof π'

» Helios accidentally avoids this issue by preventing replay of ballots

- Which was put in place to stop an actual attack on privacy

Privacy for Belenios: Trust and the Registrar

- » A dishonest registrar can give invalid credentials to all voters but one
  - Tally reveals that voter’s preferences
- » Is this an attack?
  - Yes
- » Should we care about it?
  - Yes
- » Why? Both attacks are artificial: the adversary truly learns nothing in practice…

Privacy for Belenios – The Big Problem

- Current definitions and natural extensions are not robust
  - Consideration of elements usually left out of scope is needed

- In a follow-up, Cortier and Lallemand prove that *all* current definitions of privacy imply individual verifiability

- We can only get proofs in much weaker models than those we want

- There may be actual attacks we are missing because of “silly” definitional issues

- We’re *not* solving this here: we just accept a weaker definition and move on

Strong Verifiability for Belenios

» If the adversary:
  - Controls either the registrar or the ballot box,
  - Knows the election private key, and
  - Can corrupt individual voters statically.

» The final tally corresponds to the tally computed over:
  - The votes of all honest voters who perform individual verifiability checks,
  - A subset of the votes cast by honest voters who did not check,
  - At most as many corrupted votes as there were corrupted voters.

» In practice, this is very strong as it gets
  - Gives precise bounds on the distance between final result and actual result
  - The adversary does not know who will check; statistical arguments can give tighter bounds

Machine-Checked Verifiability for Belenios

- Really nothing to report; some extensions and clarifications to Cortier et al.'s result:
  - Give the private election key to the adversary
  - Refine what it means for a voter to have checked their ballot (in case of revotes)
  - When registrar is dishonest, even honestly generated ballots may be invalid and cannot be counted

- The proofs are "straightforward formalizations"

- Could further refine to allow checking on intermediate bulleting boards hen they are published

Belenios	LoC	Ver. Time (s)	Code Sim. (%)	Dev. Effort (PW)
General Concepts	5936	348	55% Helios	4
Privacy	2700	238	75% Helios	2
Verifiability	14590	1523	-	20
Variants	47030	3965	95% Belenios	1

Next Steps in Machine-Checked Cryptographic Proofs

Take 1

Go Down from the Bottom

» Our proofs do not cover the primitives:
  - Mix-nets are assumed to be perfect obliviously permuting decryption oracles
  - Zero-Knowledge proofs are taken as assumptions
    - Including those whose statements talk about random oracles
  - Encrypt+PoK is taken as non-malleable encryption

» These proofs are fun
  - Fun number theory
  - Interesting proof techniques, where simulators can rerun adversaries with fixed randomness
  - Zero-Knowledge is still not very well understood in terms of proofs, composition
  - Even though these things are conceptually simple, they involve *interactive systems*

Go Up from the Top

- » Interactive systems are also increasingly used by the crypto community for compositional security
  - Constructive Cryptography
  - Universal Composability

- » Having proof tools that support them will be crucial in scaling machine-checked crypto up to larger constructions

- » Ideas from distributed system verification could be looked into

Summary

- » Machine-checked crypto is costly
- » But worth it for select applications where trust in the system is paramount
  - Standards, voting, e-government, privacy, …

- » Definitions of privacy for electronic voting are brittle and inadequate

- » PL and PV can still contribute to machine-checked crypto
  - Weird rewinding semantics
  - (Relational) Semantics for interactive open probabilistic programs

UNIVERSITY OF SURREY