Automated Verification of Accountability in Security Protocols

Robert Künnemann, Ilkan Esiyok and Michael Backes
Part I: What we talk about when we talk about accountability
Accountability for $\varphi$
Why Accountability

\[ \neg \text{honest}(A, B, C) \]

\[ \text{honest}(A) \Rightarrow \text{accountability for } \varphi \]

\[ \text{honest}(A, B, C) \Rightarrow \varphi \]
Why is it so hard?

soundness: \( \text{verdict}(t) \subseteq \text{corrupted}(t) \)

completeness: \( \text{verdict}(t) \supseteq \text{corrupted}(t) \)

(can imitate protocol)

\( \text{verdict}(t) = \{A \mid t|_A \text{ observably different from spec} \} \)

(e.g., PeerReview)

\( \text{verdict}(t) = \{A \mid A \text{ performed action outside spec causing } \neg \varphi \} \)

no complete view in the internet :(

Out-of-spec action causing \( \neg \varphi \) does not mean the out-of-spec process is a cause.

(Counterexample: A is buggy CA. Emits slightly malformed certificate, which is used in attack, but malformedness is irrelevant. Had A followed the spec, same attack would have happened.)
Causation

- Event(s) $A$ caused $\neg \varphi$ iff
  - $A$ and $\neg \varphi$, in fact, happened.
  - In any counterfactual where $A$ happens, $\neg \varphi$ happens.
  - $A$ is subset-minimal.

- "Umbrella" caused "not wet", as
  - I had an umbrella and did not get wet.
  - As long as I have my umbrella, I cannot get wet.
  - Without the umbrella, I could get wet.
Causation

- Event(s) A caused \( \neg \phi \) iff
  - A and \( \neg \phi \), in fact, happened.
  - In any counterfactual where A happens, \( \neg \phi \) happens.
  - A is subset-minimal.

- Output all sets of parties S, s.t.
  - \( t \not\models \neg \phi \) and corrupted(t) \( \supseteq S \)
  - there is related t' s.t. t' \( \not\models \neg \phi \)
    and corrupt(t')=S,
  - S is subset-minimal.
Part II: Accountability in terms of trace properties
**Case 1: weakest possible relation**

- Consider t' is related to t iff corrupt(t') ⊆ corrupt(t)
- Idea: verdict function defined as

\[
\text{verdict}(t) = \begin{cases} 
V_1 & \text{if } \omega_1(t) \\
\vdots \\
V_n & \text{if } \omega_n(t) 
\end{cases}
\]

- cases are **exhaustive** and **exclusive**, and for each i:
  - **sufficiency**: Agents in \(V_i\) can produce violating trace
  - **verifiability**: \(V_i=\emptyset \iff \varphi\)
  - **minimality**: can't do with less than \(S \in V_i\)
  - **uniqueness**: whenever \(\omega_i\) is observed, parties in \(V_i\) are corrupted
  - **completeness**: (omitted)
Case 2: arbitrary relation

- "But that's not what happened" -> relation $r$ between $t$ and $t'$
- idea for translation: cases are liftings $R$ of relation $r$
- combination of 11 different conditions, including lifting condition:

$$verdict(t) = \begin{cases} 
V_1 & \text{if } \omega_1(t) \\
V_2 & \text{if } \omega_2(t) \\
V_3 & \text{if } \omega_3(t)
\end{cases}$$
Part III: Implementation
Part III Implementation

SAPIC calculus + verdict function + accountability lemmas

SAPIC

multiset rewrite rules + lemmas

tamarin-prover

attack / verification / timeout

☑ weakest possible relation
☑ arbitrary relation (lifting lemma offset to user)
☑ control-flow relation:
  ▪ two-trace lemma: for all $t, t'$, if $t$ in related $\omega_i$ and $\omega_j$, control-flow is the same
  ▪ translate process so it can run "twice", producing two traces in sequence
Part IV case studies

| protocol                          | type          | # lemmas generated | # helping lemmas | time  |
|----------------------------------|---------------|--------------------|------------------|-------|
| Certificate Transp.              |               |                    |                  |       |
| model by Bruni et al             | ✓,\(r_w\)     | 31                 | 0                | 41s   |
| extended model                   | ✓,\(r_w\)     | 21                 | 0                | 50s   |
| OCSP Stapling                    |               |                    |                  |       |
| trusted resp.                    | ✓,\(r_w\)     | 7                  | 3                | 945s  |
| untrusted resp.                  | \(X, r_w\)    | 7                  | 3                | 12s   |
| Centralized monitor              |               |                    |                  |       |
| faulty                           | \(X, r_c\)    | 17                 | 0                | 5s    |
| fixed                            | ✓,\(r_c\)     | 17                 | 0                | 3s    |
| replication                      | ✓,\(r_c\)     | 17                 | 0                | 7s    |
| Accountable alg.                 |               |                    |                  |       |
| modified-1                       | ✓,\(r_c\)     | 27                 | 1                | 5792s |
| modified-2                       | ✓,\(r_c\)     | 27                 | 1                | 2047s |

\((✓)\): verification  \((X)\): attack  \((r_w)\): weak relation  \((r_c)\): control-flow r.
Conclusion

- Accountability is identifying misbehaving parties
- "misbehaving party" = "party whose deviation caused $\neg \varphi$"
- This definition is practical and can be verified automatically

**Ongoing work:**
- Integrate SAPIC calculus and translation in tamarin-prover
  - See development branch
- Support arbitrary number of parties
- Accountability in the decentralised setting
  - Central adversary is not w.l.o.g.!
- Accountability in the cryptographic setting
  - Trace properties: 👍 Indistinguishability: 🤔
Thank you!
Why is it so hard?

soundness: \( \text{verdict}(t) \subseteq \text{dishonest}(t) \)

\[
\text{verdict}(t) = \{ P \mid \text{t} \mid P \text{ observably different from spec} \} 
\]

(e.g., PeerReview)

completeness: \( \text{verdict}(t) \supseteq \text{dishonest}(t) \)

(can imitate protocol)

\[
\text{verdict}(t) = \{ P \mid \text{action by P and outside spec caused } \neg \varphi \} 
\]

no complete view in the internet :

If P followed spec, she might still cause \( \neg \varphi \)!

provocation

This work: \{ P \mid \text{Had P followed spec, then } \varphi \}
Case 1: weakest possible relation

- Consider t' is related to t iff corrupt(t') ⊆ corrupt(t)
- Idea: verdict function defined as

\[
\text{verdict}(t) = \begin{cases} 
V_1 & \text{if } \omega_1(t) \\
\vdots & \\
V_n & \text{if } \omega_n(t)
\end{cases}
\]

- cases are **exhaustive** and **exclusive**
- **sufficiency**: \( S \in V_i \Rightarrow \exists t. \text{corrupted}(t) = S \text{ and } \neg \varphi(t) \)
- **verifiability**: \( V_i = \emptyset \iff \varphi \)
- **minimality**: can't do with less than \( S \in V_i \)
- **uniqueness**: whenever \( \omega_i \) is observed, parties in \( V_i \) are corrupted
- **completeness**: (.. left out ..)
Conclusion

- Accountability via causation works and can be verified automatically

- Ongoing work:
  - integrate SAPIC calculus and translation in tamarin-prover
  - support arbitrary number of parties

- Accountability in the decentralised setting (unpublished work)
  - original definition in decentralised setting, parties deviate individually
  - provocation problem $\rightarrow$ centralised setting is not w.l.o.g.!
  - optimality requirement: deviating parties exchange no more information than necessary. conjectured to be equal to centralised setting.