On Writing and Reading
MPC Security Proofs:
An Explorational Study of State Separation

Sabine Oechsner
Aarhus University

Joined work with Chris Brzuska
(Aalto University)
What is this "state separation"?
Code-based game-playing

• Formalizing game-playing [BR06]
• Specify games and primitives in pseudocode
• Show indistinguishability of game hops by transforming pseudocode and reductions
• Well-established tool in some crypto communities, lesser so in MPC
• Proof assistant: EasyCrypt
Issues

• Composing notions is tedious (typically: define new game and reprove security)
• Gets challenging for complex protocols and security notions like in the MPC world
  – for those who remember my talk: formal verification of MPC in EasyCrypt [HKOSS18]
  – using code-based game-playing for MPC security proof
  – had to resort to ad-hoc security definitions
Is code-based game-playing even suitable for complex protocols?
State-separating proofs

• introduced recently by [BDFKK18] for analysis of TLS 1.3
• addition to code-based game-playing
• goal: facilitate modular reasoning
  – package code
  – state separation: packages have local state
  – state sharing between packages is explicit
  – algebraic approach (terms or graphs)
Concepts

Oracle $O$
- stateful randomized algorithm
- described in (pseudo)code

Package P
- provides output interface = set of oracles
- has input interface = set of oracles
- oracles of a package share local state
- composition (sequential and parallel)
Concepts

Cryptographic game $G$
- package with empty input interface
- multi-instance game = parallel instances of same game

Game with adversary
- standard notation: $\mathcal{A}^{O_1(\cdot),O_2(\cdot)}$
- here: $\mathcal{A}^G$ or rather $\mathcal{A}$
Concepts

Indistinguishability of games

- Two games $G^0$ and $G^1$ with same output interface are computationally indistinguishable if for all PPT adversaries, the distinguishing advantage is negligible.
Concepts

Reduction R

- "adapter" for games
- consider games $H^b$ that will be reduced to games $G^b$:

- identify $G^b$
- amounts to finding cuts in graphs
- $R$ and $G^b$ do not share state
Benefits

• make state sharing and separation explicit
• modular modelling and reasoning
  – break down complexity
• reductions as cuts in graphs
• composition of game-based notions
Question:

If state-separating proofs help understand key exchange protocols, can they also help with MPC?
Answer:

Yes!

I will show how to...

– model sim-based security as games (SFE)
– prove security in a modular way (Yao's garbling scheme)
– compose notions (SFE from garbling scheme and OT)
YAO'S GARBLING SCHEME
Yao and state-separating proofs

What I want to show:
State-separating proofs can help restructure existing proofs for better understanding!

[LP09], [BHR12]

Benefits:
• modular reasoning
• "automatic" reductions
Garbling scheme

- introduced by [BHR12]
- Algorithms:

  - projective: $x \in \{0,1\}^\lambda$, bitwise $En(\cdot,\cdot)$
Yao's garbling scheme

Focus on circuit garbling algorithm:

Circuit garbling $Gb$:

• assign two bitstrings ("keys") to each wire

• for each gate, compute 4 ciphertexts:
  – encrypt output wire keys under each combination of corresponding input wire keys

• to decode result, need map from output wire keys to bits
Yao and state-separating proof

Real garbling of a circuit layer:
Yao and state-separating proof

Garbling of whole circuit:
– combine layers
– shared KEY packages

Observations:
– This is a game!
– explicit state sharing with layer above/below
What can we do now?

• define layer games to reason precisely on the level of a single gate or layer
  – example: reason about messages in ciphertexts in a layer

• use layer assumptions in bigger proof
  – rest of garbling will automatically become reduction
What about "proper" simulation-based notions?
SECURE FUNCTION EVALUATION
Start simple

Setting:

– 2PC for deterministic function
– passive security
– stand-alone model
– consider only privacy (no correctness)
– identify a protocol with views of parties $P_i$
Security notion ($P_1$ only):

An SFE protocol for $C$ is secure if there exists a PPT algorithm $S_1$ s.t.

$$\{\text{view}_1(C, x_1, x_2)\}_{x_1,x_2} \equiv \{S_1(C, x_1, C(x_1, x_2))\}_{x_1,x_2}$$

Wait... Aren't those games?
SFE security

Real game $\text{SFE}^0$:
1. Adv chooses $x_1$ and $x_2$ (and implicitly C)
2. Game outputs $\text{view}_1(C, x_1, x_2)$
SFE security

Ideal game SFE\(^1\):

1. Adv chooses \(x_1\) and \(x_2\) (and implicitly C)
2. Simulator
   - gets \(x_1, C(x_1, x_2)\) and \(C\)
   - outputs \(\text{view}_1(C, x_1, x_2)\)
SFE security

An SFE protocol is secure if there exists a simulator package $\text{SIM}^{sfe}$ s.t. $\text{SFE}^0$ and $\text{SFE}^1$ are comp. indistinguishable.
We already saw garbling schemes...
Can we build an SFE protocol?
Yao's garbled circuits

Evaluator $P_1$

$\tilde{y} \leftarrow Ev(\tilde{C}, \tilde{x}[1..n])$

$y \leftarrow De(Z,\tilde{y})$

Garbler $P_2$

$(\tilde{C}, e, Z) \leftarrow Gb(C)$

$\forall j = m+1..n : \tilde{x}[j] \leftarrow En(e[j], j, x[j])$
Yao's garbled circuits

Building block: oblivious transfer

OT^0

A

\text{SETBIT}[0,j]

\text{GETBIT}[0,j]

\text{GETMAP}[0,j]

\text{KEY}[0,j]

\text{SETMAP}[0,j]

\text{VIEW}_{\text{rec}}^{\text{ot}}

OT^{1}

A

\text{SETBIT}[0,j]

\text{GETBIT}[0,j]

\text{GETMAP}[0,j]

\text{KEY}[0,j]

\text{SETMAP}[0,j]

\text{GARBLE}[0,j]

\text{EN}[0,j]

\text{SIM}_{\text{rec}}^{\text{ot}}[j]

\text{VIEW}_{\text{rec}}^{\text{ot}}
Yao's garbled circuits

Building block: garbling scheme

GS^0
Yao's garbled circuits

Building block: garbling scheme
Yao's garbled circuits

Building block: garbling scheme
Yao's garbled circuits: Security

What you will see:
- game-hopping from real to ideal game
- in graphs (with precise meaning!)
- steps: identify suitable cut in graph and use assumptions (rest of graph becomes reduction)

Assumptions:
- garbling scheme is secure
- OT protocol has passive security
Step 0: Real game $SFE^0$
Step 1: Implement SFE
Step 1: Implement SFE
Step 1: Implement SFE
Step 2: Identify real OT game
Step 3: Idealize OT
Step 3: Idealize OT
Step 4: Identify real garbling scheme game
Step 4: Identify real garbling scheme game
Step 5: Idealize garbling scheme
Step 5: Idealize garbling scheme
Step 6: Identify ideal SFE game
Step 6: Identify ideal SFE game
CONCLUSION
Conclusion

1. Simulation-based security notions are also games
   – techniques for game-based proofs might carry over
   – compose "traditional" games for construction
   – define ad-hoc games for intermediate steps

2. Intermediate steps in simulation-based proofs are already treated as game hops
   – possible to make more precise!
   – tool for finding proofs and writing them down
Conclusion

3. There is no need for writing explicit security reductions.

4. More structure for proofs has more benefits.
   – didactic purposes
   – formal verification
Conclusion

Open questions:

– Does this work for more parties? Active security? Universally composable security notions?

What I want to get feedback on:

– Do you think this approach could be helpful for finding or writing other proofs?
– Does our modelling help with modelling other similar protocols?

To appear on eprint soon!