State Separation for Code-Based Game-Playing Proofs

Chris Brzuska, Antoine Délignat-Lavaud, Cédric Fournet,
Konrad Kohbrok, Markulf Kohlweiss

December 6, 2018

Aalto University
Microsoft Research Cambridge
University of Edinburgh

Motivation

. . . in the beginning there was miTLS.

Motivation

. . . in the beginning there was miTLS.

- implementation of TLS in $\mathrm{F}^{\star}$
- various nice guarantees:
    - constant-time code
    - memory safe
    - functionally correct

Motivation

. . . in the beginning there was miTLS.

- implementation of TLS in $\mathrm{F}^{\star}$
- various nice guarantees:
  - constant-time code
  - memory safe
  - functionally correct
- "cryptographically verified" - proof in code?

How are they doing that?

Our Contributions

Make it easier to do:

Possible applications:

Our Contributions

Make it easier to do:

- modular composed proofs

Possible applications:

- TLS

Our Contributions

Make it easier to do:

- modular composed proofs
- key composition

Possible applications:

- TLS
- Messaging

Our Contributions

Make it easier to do:

- modular composed proofs
- key composition
- hybrid arguments

Possible applications:

- TLS
- Messaging
- Multi-Instance

Our Contributions

Make it easier to do:

- modular composed proofs
- key composition
- hybrid arguments
- (partially) machine-checkable proofs

Possible applications:

- TLS
- Messaging
- Multi-Instance
- $\mathrm{F}^\star$, other proof assistants

Related Work

- Universal Composability ([C01])
- Abstract- and Constructive Crypto ([MR11],[M11])
- “The Joy of Cryptography” (Rosulek)
- EasyCrypt ([BGHB11])

Packages - Security Games

$\texttt{IND-CPA}_e^b$

GEN()

assert $k = \perp$

$k \leftarrow e.\mathsf{KGen}(1^n)$

return ()

ENC(m)

assert $k \neq \perp$

if $b = 0$ **then**

  $c \leftarrow e.\mathsf{Enc}(k, m)$

else

  $c \leftarrow e.\mathsf{Enc}(k, 0^{|m|})$

return c

$\texttt{IND-CPA}_e^b$

GEN()

assert $k = \bot$

$k \leftarrow e.\mathrm{KGen}(1^n)$

return ()

ENC(m)

assert $k \neq \bot$

if $b = 0$ **then**

  $c \leftarrow e.\mathrm{Enc}(k, m)$

else

  $c \leftarrow e.\mathrm{Enc}(k, 0^{|m|})$

return c

Packages - Security Games

$\texttt{IND-CPA}_e^b$

GEN()

assert $k = \perp$

$k \leftarrow e.\mathsf{KGen}(1^n)$

return $()$

ENC(m)

assert $k \neq \perp$

if $b = 0$ **then**

$\quad c \leftarrow e.\mathsf{Enc}(k, m)$

else

$\quad c \leftarrow e.\mathsf{Enc}(k, 0^{|m|})$

return c

- a package contains oracle descriptions and their state

Packages - Security Games

GEN

ENC

$\texttt{IND-CPA}_e^b$

GEN()

assert $k = \bot$

$k \leftarrow e.\mathsf{KGen}(1^n)$

return ()

ENC(m)

assert $k \neq \bot$

if $b = 0$ **then**

$\quad c \leftarrow e.\mathsf{Enc}(k, m)$

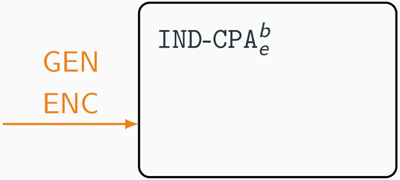
else

$\quad c \leftarrow e.\mathsf{Enc}(k, 0^{|m|})$

return c

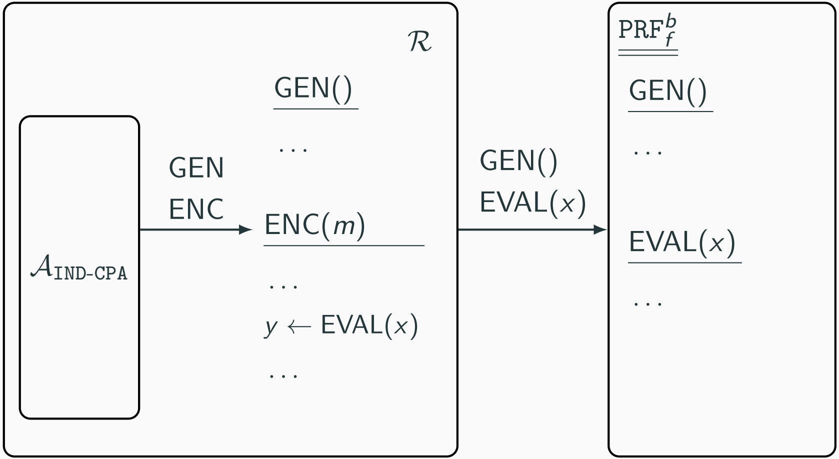
- a package contains oracle descriptions and their state
- it provides these oracles for other algorithms to use

Packages - Security Games

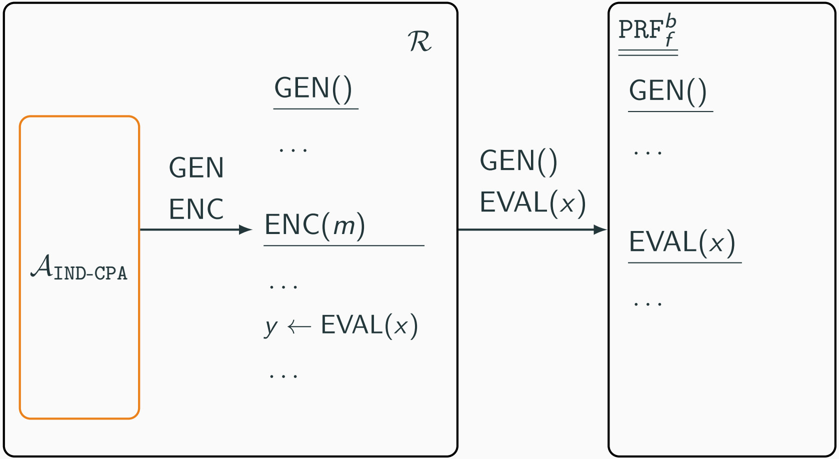


- a package contains oracle descriptions and their state
- it provides these oracles for other algorithms to use
- packages are composable.

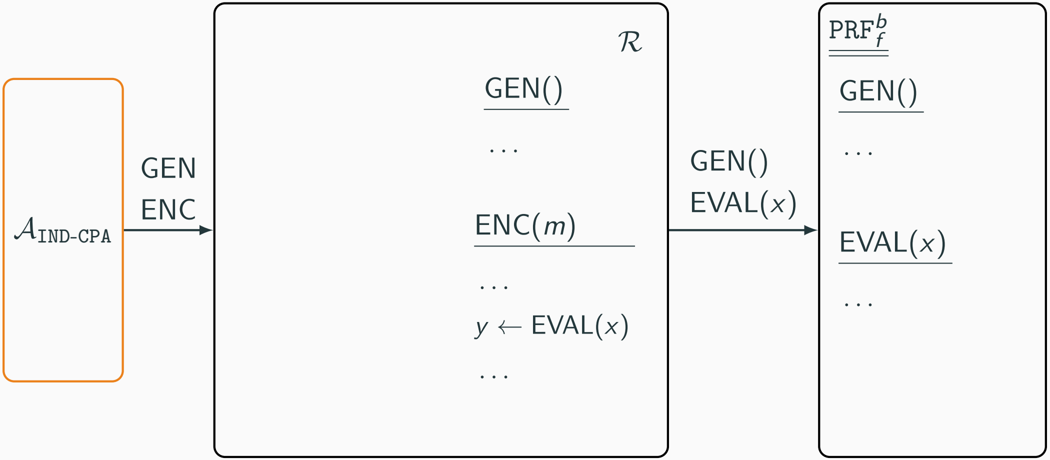
Packages - Reductions



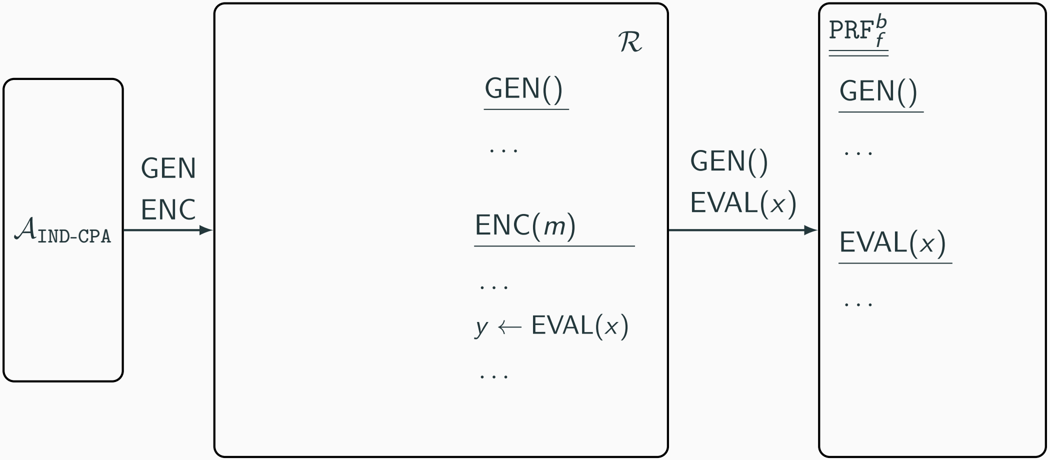
Packages - Reductions



Packages - Reductions

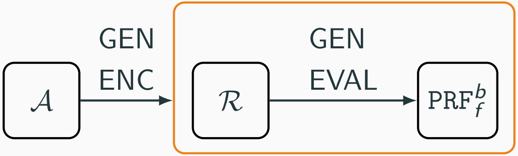


Packages - Reductions

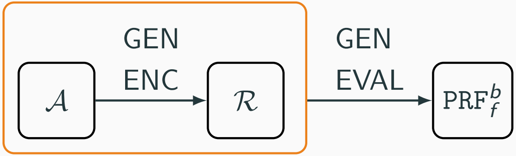


Packages - Reductions

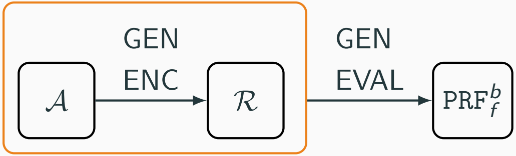
$$\mathcal{A} \xrightarrow{\substack{\text{GEN} \\ \text{ENC}}} \mathcal{R} \xrightarrow{\substack{\text{GEN} \\ \text{EVAL}}} \mathrm{PRF}_f^b$$


$\mathcal{A}$
GEN
ENC
$\mathcal{R}$
GEN
EVAL
PRF_f^b

Packages - Reductions



Packages - Reductions



Packages . . .

- contain oracle descriptions and state,
- can provide oracles to other packages,
- and can call oracles provided by other packages.

Example I

Reducing $\texttt{IND-CPA}_e$ to $\texttt{PRF}_f$

Overview

Proof Goal:

$$\boxed{\texttt{IND-CPA}_e^0} \overset{\epsilon_1(\mathcal{A}_{\texttt{IND-CPA}})}{\approx} \boxed{\texttt{IND-CPA}_e^1}$$

Proof Goal:

$$\boxed{\mathtt{IND\text{-}CPA}_e^0} \overset{\epsilon_1(\mathcal{A}_{\mathtt{IND\text{-}CPA}})}{\approx} \boxed{\mathtt{IND\text{-}CPA}_e^1}$$

Assumption:

$$\boxed{\mathtt{PRF}_f^0} \overset{\epsilon_2(\mathcal{A}_{\mathtt{PRF}})}{\approx} \boxed{\mathtt{PRF}_f^1}$$

Overview

Proof Goal:

$$\boxed{\texttt{IND-CPA}_e^0} \overset{\epsilon_1(\mathcal{A}_{\texttt{IND-CPA}})}{\approx} \boxed{\texttt{IND-CPA}_e^1}$$

Assumption:

$$\boxed{\texttt{PRF}_f^0} \overset{\epsilon_2(\mathcal{A}_{\texttt{PRF}})}{\approx} \boxed{\texttt{PRF}_f^1}$$

Concrete Security

Relate $\epsilon_1(\cdot)$ to $\epsilon_2(\cdot)$ in two steps:

1. Simulation correctness
2. Applying assumptions

Step 1: Simulation Correctness

$\boxed{\mathcal{R}} \rightarrow \boxed{\mathrm{PRF}_f^0}$

Step 1: Simulation Correctness

To prove: Perfect Indistinguishability

$\boxed{\texttt{IND-CPA}_e^0} \equiv \boxed{\mathcal{R}} \to \boxed{\texttt{PRF}_f^0}$

and

$\boxed{\texttt{IND-CPA}_e^1} \equiv \boxed{\mathcal{R}} \to \boxed{\texttt{PRF}_f^1}$

Step 2: Applying Assumptions

$$\boxed{\texttt{IND-CPA}_e^0} \equiv \boxed{\mathcal{R}} \to \boxed{\texttt{PRF}_f^0}$$

and

$$\boxed{\texttt{IND-CPA}_e^1} \equiv \boxed{\mathcal{R}} \to \boxed{\texttt{PRF}_f^1}$$

$$\boxed{\texttt{PRF}_f^0} \overset{\epsilon_2(\mathcal{A}_{\text{PRF}})}{\approx} \boxed{\texttt{PRF}_f^1}$$

Step 2: Applying Assumptions

$$\boxed{\texttt{IND-CPA}_e^0} \equiv \boxed{\mathcal{R}} \to \boxed{\texttt{PRF}_f^0}$$

$$\wr\!\wr\ \epsilon_2\Big(\underbrace{\boxed{\mathcal{A}_{\texttt{IND-CPA}}} \to \boxed{\mathcal{R}}}_{\mathcal{A}_{\text{PRF}}}\Big)$$

$$\boxed{\texttt{IND-CPA}_e^1} \equiv \boxed{\mathcal{R}} \to \boxed{\texttt{PRF}_f^1}$$

$$\boxed{\texttt{PRF}_f^0} \overset{\epsilon_2(\mathcal{A}_{\text{PRF}})}{\approx} \boxed{\texttt{PRF}_f^1}$$

General Proof Pattern

$$
\begin{aligned}
&\boxed{\texttt{IND-CPA}^0_e} \equiv \boxed{\mathcal{R}} \to \boxed{\texttt{PRF}^0_f} \\
&\qquad\qquad\qquad\quad \wr\ \epsilon_2\Big(\underbrace{\boxed{\mathcal{A}_{\texttt{IND-CPA}}} \to \boxed{\mathcal{R}}}_{\mathcal{A}_{\mathrm{PRF}}}\Big) \\
&\boxed{\texttt{IND-CPA}^1_e} \equiv \boxed{\mathcal{R}} \to \boxed{\texttt{PRF}^1_f}
\end{aligned}
$$

Some notes:

- graphs have precise meaning
- an inline notation exists

Example II

Key Composition

Overview

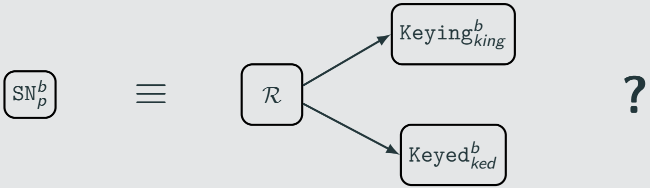
Proof Goal:

$$\boxed{\mathtt{SN}_p^0} \overset{\epsilon_1(\mathcal{A}_{\mathtt{SN}})}{\approx} \boxed{\mathtt{SN}_p^1},$$ where SN (security notion) could be PKE-CCA

Proof Goal:

$$\boxed{\mathtt{SN}^0_p} \overset{\epsilon_1(\mathcal{A}_{\mathtt{SN}})}{\approx} \boxed{\mathtt{SN}^1_p},$$ where SN (security notion) could be PKE-CCA

Common Pattern



where Keying could be KEM-CCA and Keyed could be DEM-CCA.

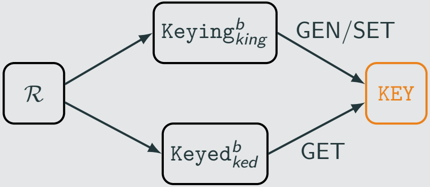
Overview

Proof Goal:

$$\boxed{\mathtt{SN}_p^0} \overset{\epsilon_1(\mathcal{A}_{\mathtt{SN}})}{\approx} \boxed{\mathtt{SN}_p^1},$$ where $\mathtt{SN}$ (security notion) could be $\mathtt{PKE\text{-}CCA}$

Common Pattern

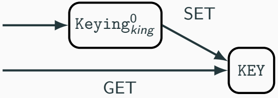
$$\boxed{\mathtt{SN}_p^b} \quad \equiv$$



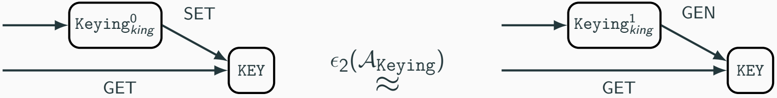
where $\mathtt{Keying}$ could be $\mathtt{KEM\text{-}CCA}$ and $\mathtt{Keyed}$ could be $\mathtt{DEM\text{-}CCA}$.

Assumptions

`Keying` Assumption

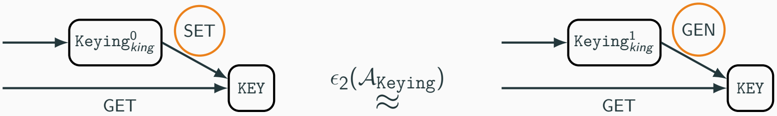


`Keying` **Assumption**

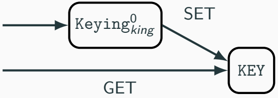


Assumptions

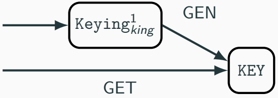
`Keying` Assumption



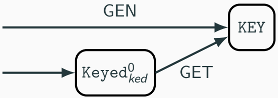
`Keying` Assumption



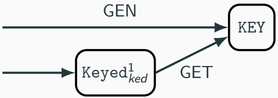
$$\epsilon_2(\mathcal{A}_{\texttt{Keying}})$$
$$\approx$$



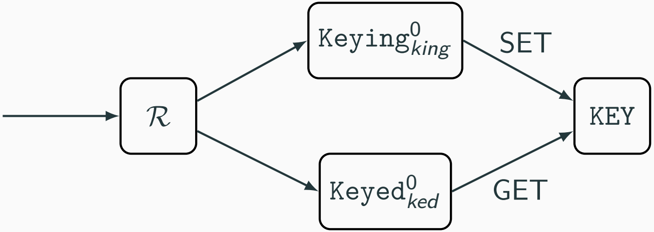
`Keyed` Assumption



$$\epsilon_3(\mathcal{A}_{\texttt{Keyed}})$$
$$\approx$$

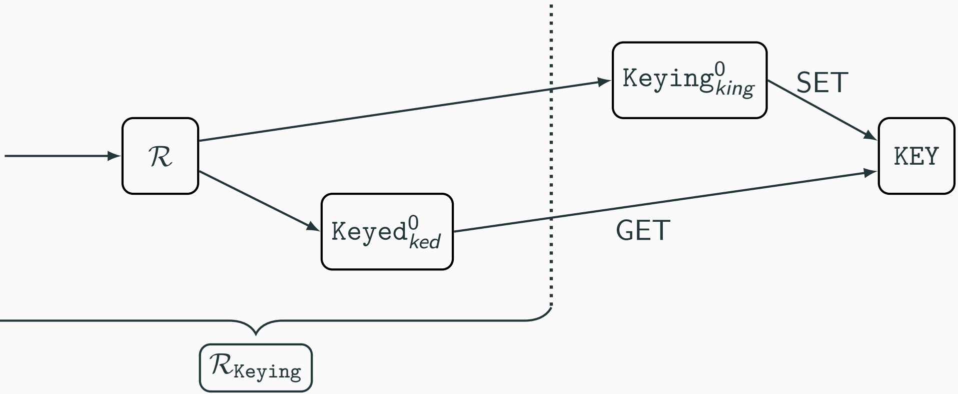


Applying the Assumptions



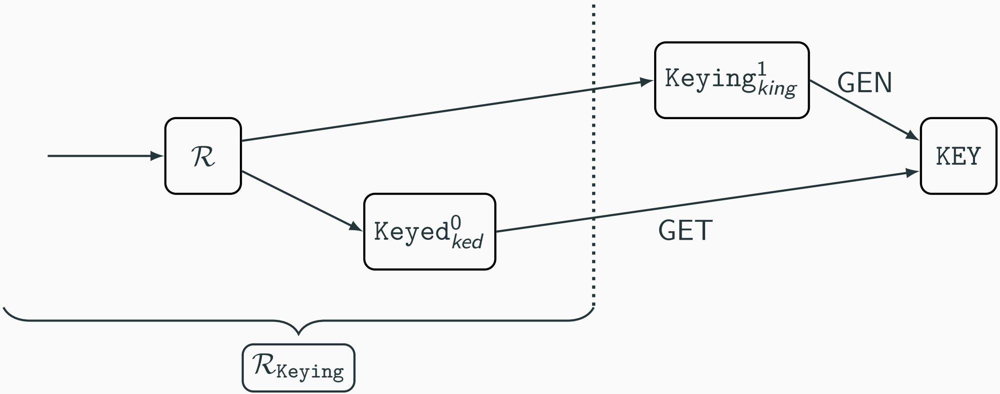
$$\epsilon_1(\mathcal{A}_{\texttt{SN}}) =$$

Applying the Assumptions



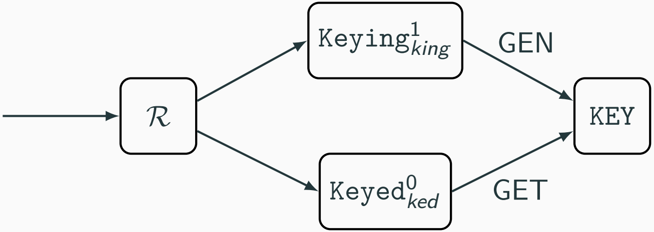
$\epsilon_1(\mathcal{A}_{\texttt{SN}}) =$

Applying the Assumptions



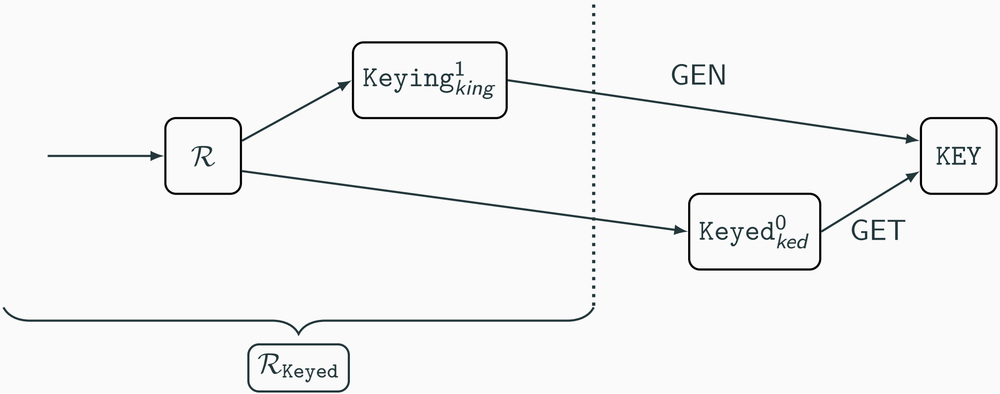
$$\epsilon_1(\mathcal{A}_{\mathtt{SN}}) = \epsilon_2(\boxed{\mathcal{A}_{\mathtt{SN}}} \rightarrow \boxed{\mathcal{R}_{\mathtt{Keying}}})$$

Applying the Assumptions



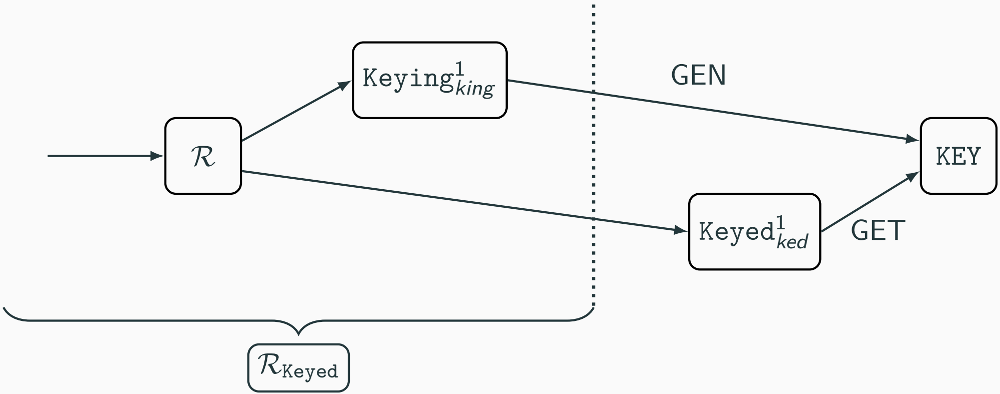
$$\epsilon_1(\mathcal{A}_{\texttt{SN}}) = \epsilon_2(\boxed{\mathcal{A}_{\texttt{SN}}} \rightarrow \boxed{\mathcal{R}_{\texttt{Keying}}})$$

Applying the Assumptions



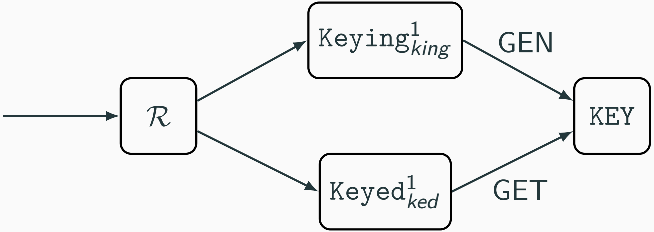
$$\epsilon_1(\mathcal{A}_{\texttt{SN}}) = \epsilon_2(\boxed{\mathcal{A}_{\texttt{SN}}} \to \boxed{\mathcal{R}_{\texttt{Keying}}})$$

Applying the Assumptions



$$\epsilon_1(\mathcal{A}_{\mathtt{SN}}) = \epsilon_2(\boxed{\mathcal{A}_{\mathtt{SN}}} \rightarrow \boxed{\mathcal{R}_{\mathtt{Keying}}}) + \epsilon_3(\boxed{\mathcal{A}_{\mathtt{SN}}} \rightarrow \boxed{\mathcal{R}_{\mathtt{Keyed}}})$$

Applying the Assumptions



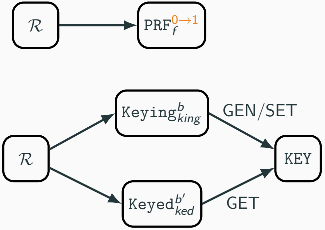
$$\epsilon_1(\mathcal{A}_{\texttt{SN}}) = \epsilon_2(\boxed{\mathcal{A}_{\texttt{SN}}} \to \boxed{\mathcal{R}_{\texttt{Keying}}}) + \epsilon_3(\boxed{\mathcal{A}_{\texttt{SN}}} \to \boxed{\mathcal{R}_{\texttt{Keyed}}})$$

Observations & Conclusions

- extension of BR-style "game-hopping"
    - packaging of code and state.

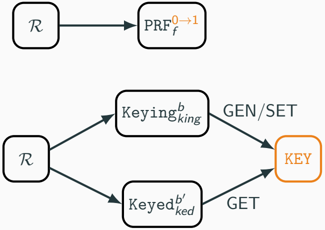
Observations & Conclusions

- extension of BR-style “game-hopping”
    - packaging of code and state.
- useful for composed protocols (TLS)



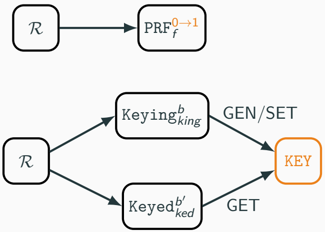
Observations & Conclusions

- extension of BR-style "game-hopping"
    - packaging of code and state.
- useful for composed protocols (TLS)
- enables key composition (Messaging)



Observations & Conclusions

- extension of BR-style "game-hopping"
  - packaging of code and state.
- useful for composed protocols (TLS)
- enables key composition (Messaging)
- less useful for . . .
  - implications ($\texttt{AE} \implies \texttt{IND-CCA}$)
  - smaller proofs



Work in Progress

- TLS 1.3 Key Schedule (for miTLS)
- Multi-Party Computation (Yao's Garbled Circuits)
- Protocol Design (Secret Handshake 2)