Simplifying Game-Based Definitions

Indistinguishability up to correctness and its application to stateful AE

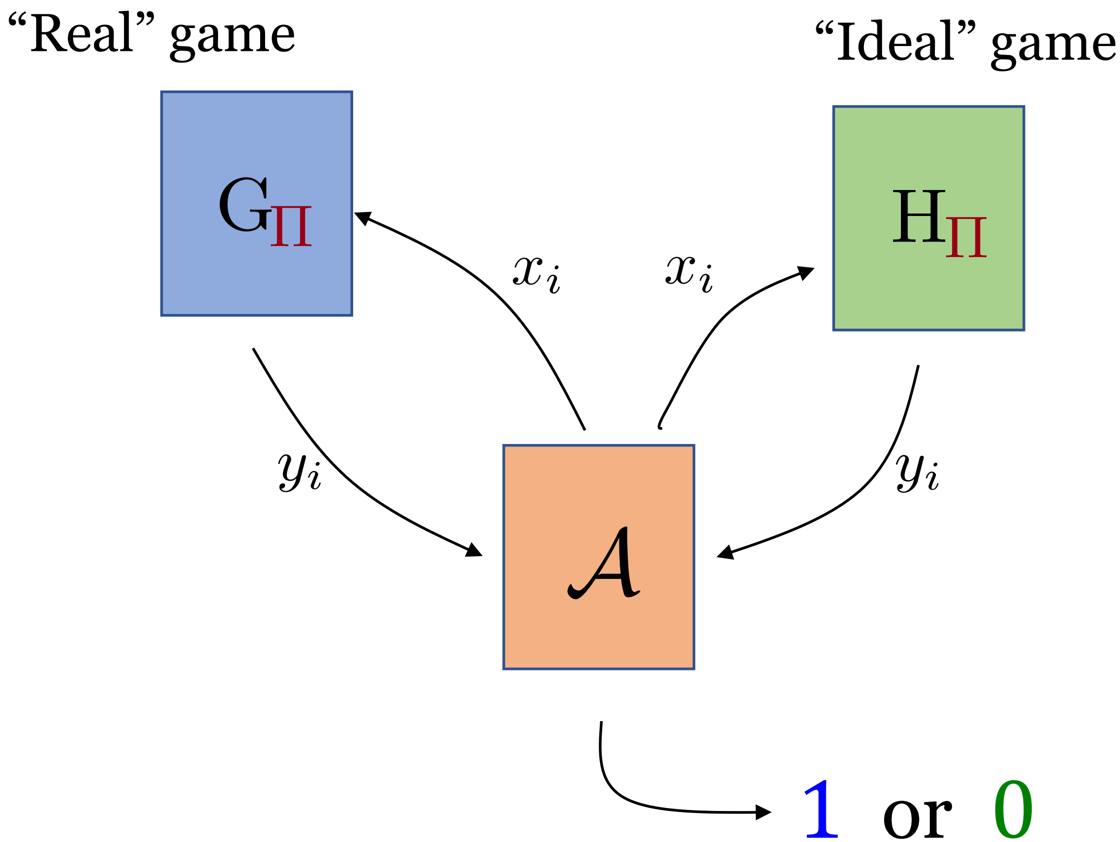
Phillip Rogaway **Yusi ("James") Zhang**

University of California, Davis, USA

CRYPTO 2018

1. **Introduction**
2. **IND|C**
3. **Examples**

IND definition for formalizing cryptographic goals



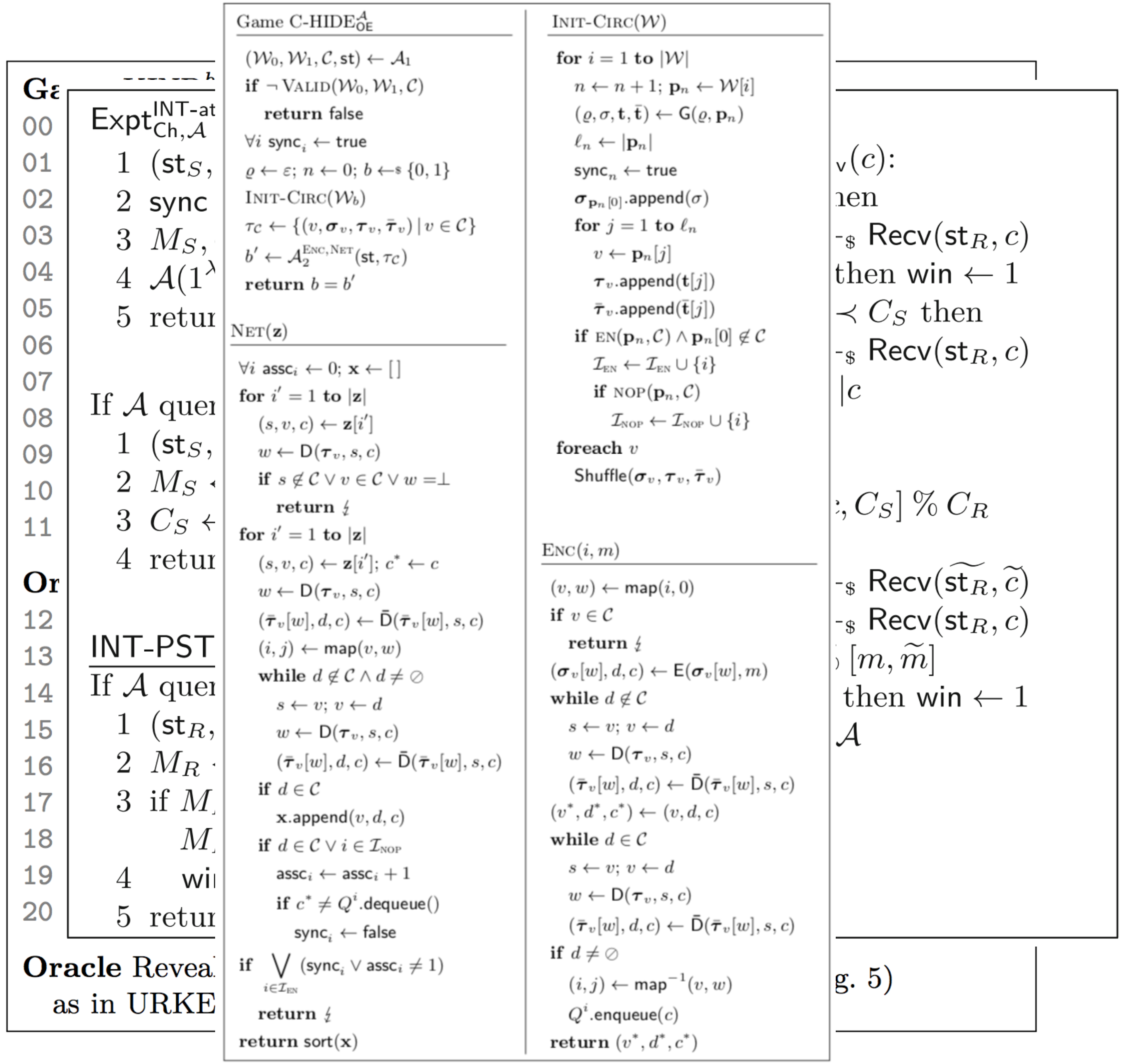
$$\mathbf{Adv}_{\Pi}^{\mathsf{xxx}}(\mathcal{A}) =$$

$$\mathbf{Adv}_{G_\Pi, H_\Pi}^{\mathsf{ind}}(\mathcal{A}) = \Pr[G_\Pi^{\mathcal{A}} \to 1] - \Pr[H_\Pi^{\mathcal{A}} \to 1]$$

[PR18: *Towards Bidirectional Ratcheted Key Exchange*]

[FGMP15: *Data is a Stream: Security of Stream-based Channels*]

[DS18: *Untagging Tor: A Formal Treatment of Onion Encryption*]



Game C-HIDE$^{\mathcal{A}}_{\mathsf{OE}}$

$(\mathcal{W}_0, \mathcal{W}_1, \mathcal{C}, \mathsf{st}) \leftarrow \mathcal{A}_1$
if $\neg$ VALID$(\mathcal{W}_0, \mathcal{W}_1, \mathcal{C})$
  return false
$\forall i$ sync$_i \leftarrow$ true
$\varrho \leftarrow \varepsilon$; $n \leftarrow 0$; $b \leftarrow_\$ \{0,1\}$
INIT-CIRC$(\mathcal{W}_b)$
$\tau_{\mathcal{C}} \leftarrow \{(v, \boldsymbol{\sigma}_v, \boldsymbol{\tau}_v, \bar{\boldsymbol{\tau}}_v) \mid v \in \mathcal{C}\}$
$b' \leftarrow \mathcal{A}_2^{\mathrm{Enc},\mathrm{Net}}(\mathsf{st}, \tau_{\mathcal{C}})$
return $b = b'$

NET$(\mathbf{z})$

$\forall i$ assc$_i \leftarrow 0$; $\mathbf{x} \leftarrow [\,]$
for $i' = 1$ **to** $|\mathbf{z}|$
  $(s, v, c) \leftarrow \mathbf{z}[i']$
  $w \leftarrow \mathsf{D}(\boldsymbol{\tau}_v, s, c)$
  if $s \notin \mathcal{C} \vee v \in \mathcal{C} \vee w = \perp$
    return $\lightning$
for $i' = 1$ **to** $|\mathbf{z}|$
  $(s, v, c) \leftarrow \mathbf{z}[i']$; $c^* \leftarrow c$
  $w \leftarrow \mathsf{D}(\boldsymbol{\tau}_v, s, c)$
  $(\bar{\boldsymbol{\tau}}_v[w], d, c) \leftarrow \bar{\mathsf{D}}(\bar{\boldsymbol{\tau}}_v[w], s, c)$
  $(i, j) \leftarrow \mathsf{map}(v, w)$
  while $d \notin \mathcal{C} \wedge d \neq \oslash$
    $s \leftarrow v$; $v \leftarrow d$
    $w \leftarrow \mathsf{D}(\boldsymbol{\tau}_v, s, c)$
    $(\bar{\boldsymbol{\tau}}_v[w], d, c) \leftarrow \bar{\mathsf{D}}(\bar{\boldsymbol{\tau}}_v[w], s, c)$
  if $d \in \mathcal{C}$
    $\mathbf{x}$.append(v, d, c)
  if $d \in \mathcal{C} \vee i \in \mathcal{I}_{\mathrm{NOP}}$
    assc$_i \leftarrow$ assc$_i + 1$
    if $c^* \neq Q^i$.dequeue()
      sync$_i \leftarrow$ false
if $\bigvee_{i \in \mathcal{I}_{\mathrm{EN}}}$ (sync$_i \vee$ assc$_i \neq 1$)
  return $\lightning$
return sort$(\mathbf{x})$

INIT-CIRC$(\mathcal{W})$

for $i = 1$ **to** $|\mathcal{W}|$
  $n \leftarrow n + 1$; $\mathbf{p}_n \leftarrow \mathcal{W}[i]$
  $(\varrho, \sigma, \mathbf{t}, \bar{\mathbf{t}}) \leftarrow \mathsf{G}(\varrho, \mathbf{p}_n)$
  $\ell_n \leftarrow |\mathbf{p}_n|$
  sync$_n \leftarrow$ true
  $\boldsymbol{\sigma}_{\mathbf{p}_n[0]}$.append$(\sigma)$
  for $j = 1$ **to** ℓ_n
    $v \leftarrow \mathbf{p}_n[j]$
    $\boldsymbol{\tau}_v$.append$(\mathbf{t}[j])$
    $\bar{\boldsymbol{\tau}}_v$.append$(\bar{\mathbf{t}}[j])$
  if EN$(\mathbf{p}_n, \mathcal{C}) \wedge \mathbf{p}_n[0] \notin \mathcal{C}$
    $\mathcal{I}_{\mathrm{EN}} \leftarrow \mathcal{I}_{\mathrm{EN}} \cup \{i\}$
    if NOP$(\mathbf{p}_n, \mathcal{C})$
      $\mathcal{I}_{\mathrm{NOP}} \leftarrow \mathcal{I}_{\mathrm{NOP}} \cup \{i\}$
foreach v
  Shuffle$(\boldsymbol{\sigma}_v, \boldsymbol{\tau}_v, \bar{\boldsymbol{\tau}}_v)$

ENC(i, m)

$(v, w) \leftarrow \mathsf{map}(i, 0)$
if $v \in \mathcal{C}$
  return $\lightning$
$(\boldsymbol{\sigma}_v[w], d, c) \leftarrow \mathsf{E}(\boldsymbol{\sigma}_v[w], m)$
while $d \notin \mathcal{C}$
  $s \leftarrow v$; $v \leftarrow d$
  $w \leftarrow \mathsf{D}(\boldsymbol{\tau}_v, s, c)$
  $(\bar{\boldsymbol{\tau}}_v[w], d, c) \leftarrow \bar{\mathsf{D}}(\bar{\boldsymbol{\tau}}_v[w], s, c)$
$(v^*, d^*, c^*) \leftarrow (v, d, c)$
while $d \in \mathcal{C}$
  $s \leftarrow v$; $v \leftarrow d$
  $w \leftarrow \mathsf{D}(\boldsymbol{\tau}_v, s, c)$
  $(\bar{\boldsymbol{\tau}}_v[w], d, c) \leftarrow \bar{\mathsf{D}}(\bar{\boldsymbol{\tau}}_v[w], s, c)$
if $d \neq \oslash$
  $(i, j) \leftarrow \mathsf{map}^{-1}(v, w)$
  Q^i.enqueue(c)
return (v^*, d^*, c^*)

Changes made in one year by the **authors** themselves.

$\textsc{Auth}_i$ from [Boyd-Hale-Miolsnes-Stebila 2016]

$\mathsf{Exp}^{\mathsf{auth}_i}_{\Pi,\mathcal{A}}()$:

$k \xleftarrow{\$} \mathrm{Kgn}()$
$st_{\mathrm{E}} \leftarrow \perp, st_{\mathrm{D}} \leftarrow \perp$
$u \leftarrow 0,\ v \leftarrow 0$
$\mathbf{r} \leftarrow 0$
$\mathcal{A}^{\mathsf{Send}(\cdot),\mathsf{Recv}(\cdot)}()$
return $\mathbf{r}$

Oracle $\mathsf{Send}(m)$:

$u \leftarrow u + 1$
$(sent_u, st_{\mathrm{E}}) \leftarrow \mathrm{Snd}(k, m, st_{\mathrm{E}})$
return $sent_u$ to $\mathcal{A}$

Oracle $\mathrm{Recv}(c)$:

There should be a "return r" here.

$v \leftarrow v + 1$
$rcvd_v \leftarrow c$
$(m, \alpha, st_{\mathrm{D}}) \leftarrow \mathrm{Rcv}(k, c, st_{\mathrm{D}})$ $= 0)]$ **then**
if $(\alpha = 1) \wedge \mathsf{cond}_i$ **then**
  $\mathbf{r} \leftarrow 1$
  return $\mathbf{r}$ from experiment
return $\perp$ to $\mathcal{A}$

Basic authentication, no replays:
$\mathsf{cond}_2 = (\nexists w : c = sent_w) \vee (\exists w < v : c = rcvd_w)$

Basic authentication, no replays, strictly increasing, no drops:
$\mathsf{cond}_4 = (u < v) \vee (c \neq sent_v)$

Problems with the IND paradigm

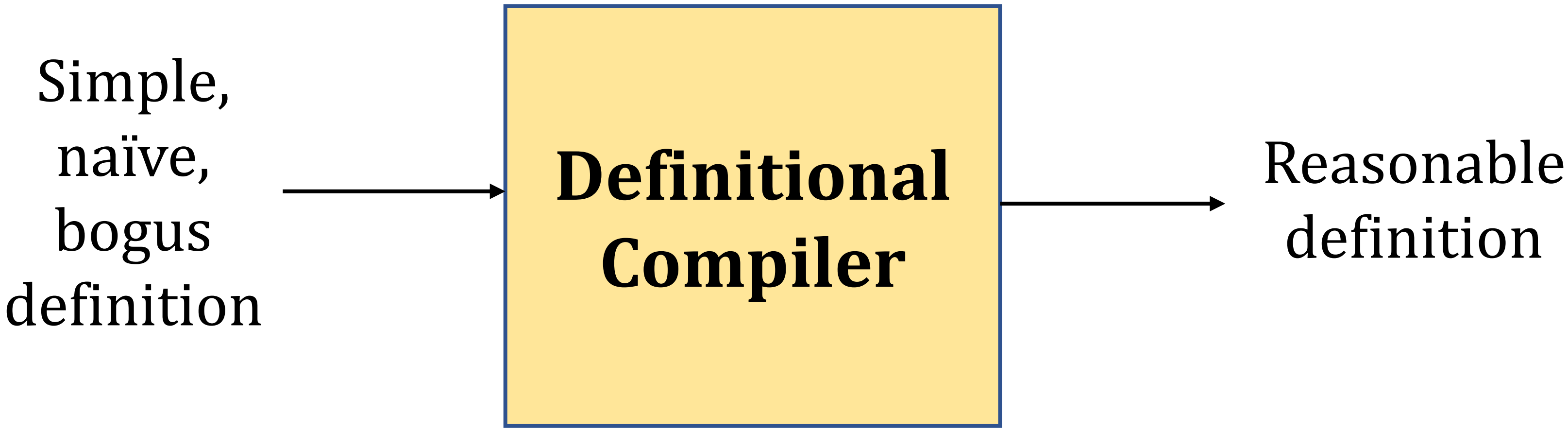
1. Defs can get **so complicated/subtle** they're hard to debug/believe.

2. People **mess up** /are **vague** even with basic defns.
   [BHK 09/15: *Subtleties in the Definition of IND-CCA?*]

3. Hard to **justify** your games capture what you want?

4. There's **no theory** on how to use IND to create defns.

Simplifying IND-based definitions



IND|C

Oracle editing

Class **Real** **Ideal**

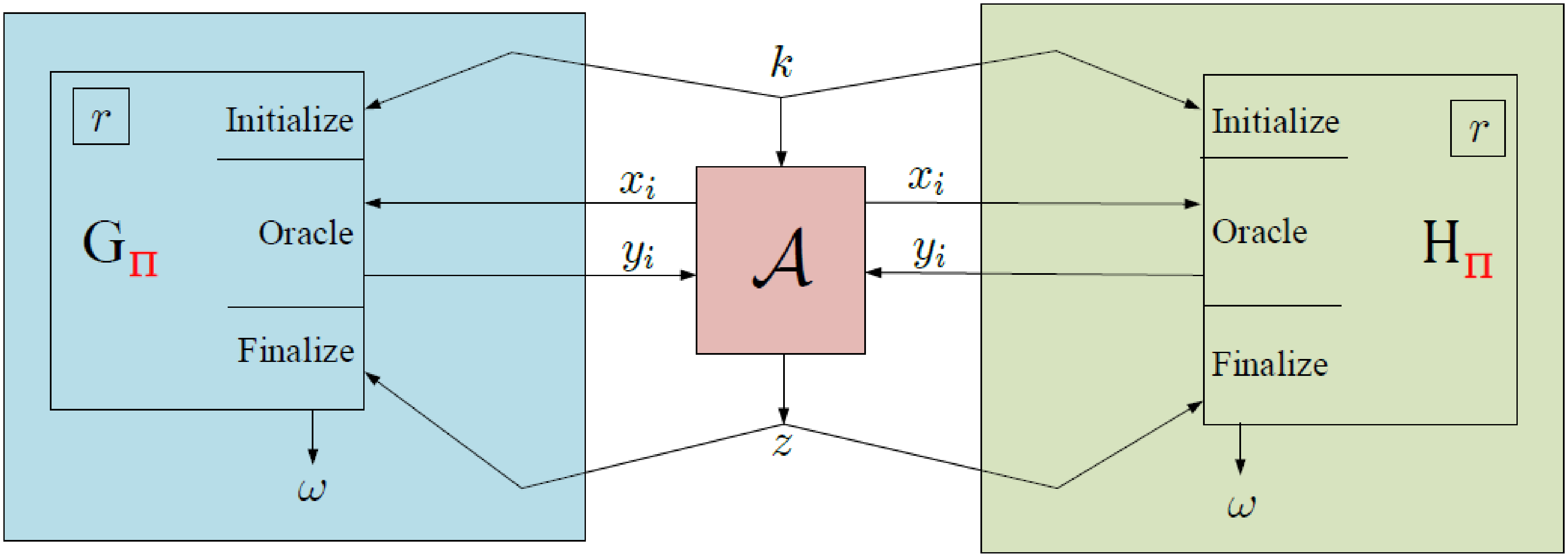
C G_{Π} utopian games $\Pi \in \mathrm{C}$ H_{Π}

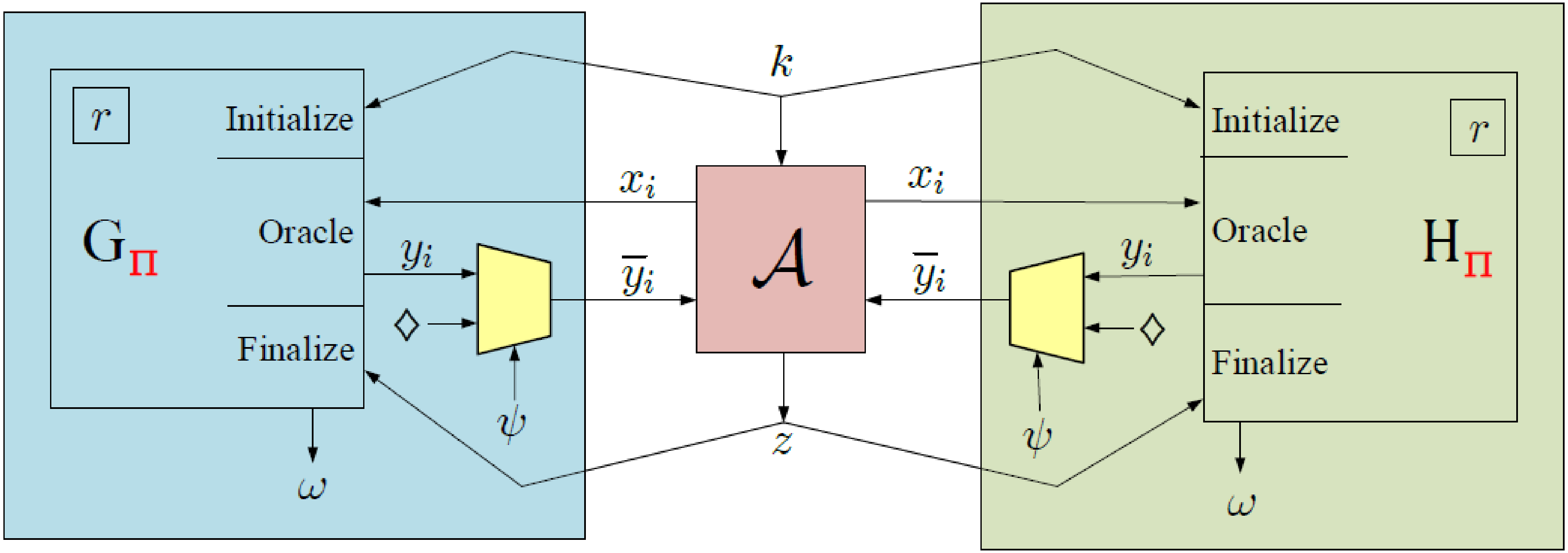
Oracle Editing

$\widetilde{\mathrm{G}}_{\Pi}$ edited games $\widetilde{\mathrm{H}}_{\Pi}$

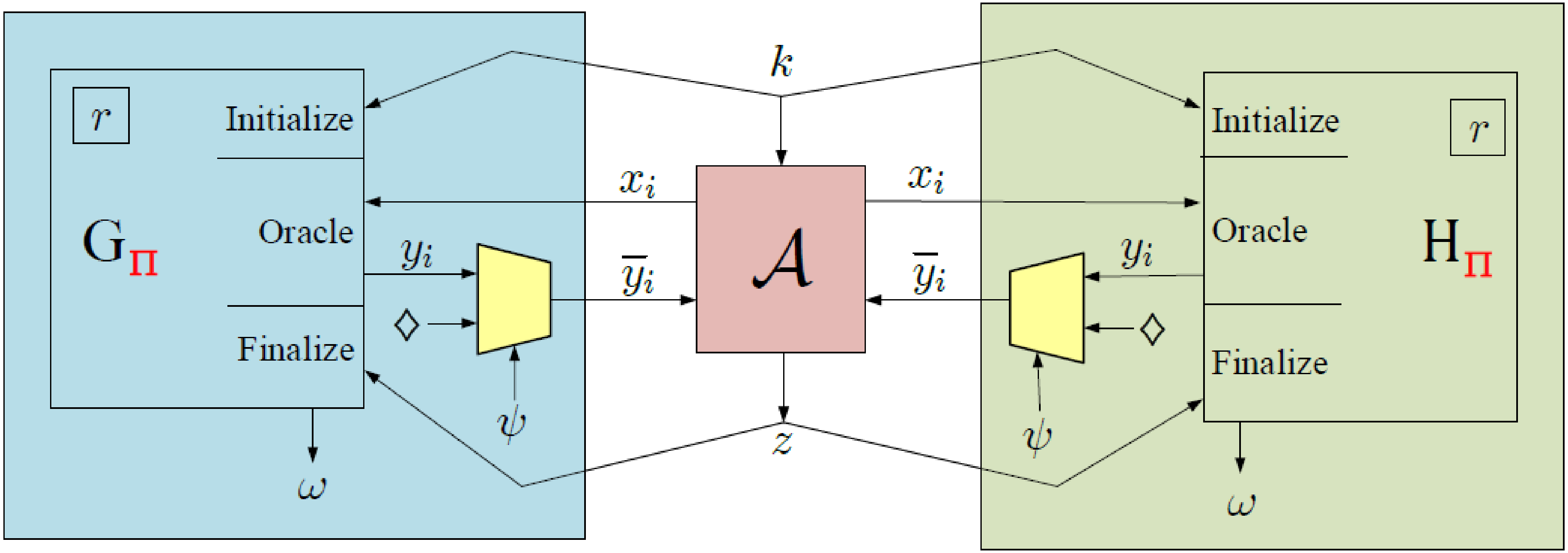
$$\mathbf{Adv}^{\mathsf{xxx}}_{\Pi}(\mathcal{A}) = \mathbf{Adv}^{\mathsf{indc}}_{\mathrm{G}_{\Pi},\mathrm{H}_{\Pi},\mathrm{C}}(\mathcal{A}) = \mathbf{Adv}^{\mathsf{ind}}_{\widetilde{\mathrm{G}}_{\Pi},\widetilde{\mathrm{H}}_{\Pi}}(\mathcal{A})$$

Oracle editing by silencing





Silencing function $\psi = \mathrm{Silence}_{\mathrm{C,G}}(\mathbf{t})$ operates on a query-terminated transcript $\mathbf{t} = (x_1, y_1, x_2, y_2, \ldots, x_i)$



Silencing function $\psi = \text{Silence}_{C,G}(\mathbf{t})$ operates on a query-terminated transcript $\mathbf{t} = (x_1, y_1, x_2, y_2, \ldots, x_i)$

Silence if given ***t*,** the answer is **fixed** across all $\Pi \in C$.

$$\text{Valid}_{C,G}(x_1, y_1, \ldots, x_j, y_j) = (\exists \Pi \in C)(\exists k \in \{0,1\}^*)(\exists r \in \{0,1\}^\infty)(\forall i \in [1..j]) \; [G_{\Pi}(k, x_1, \ldots, x_i, r) = y_i]$$

$$\text{Fixed}_{C,G}(x_1, y_1, \ldots, x_j, y_j, x) = (\exists!\, y)\text{Valid}_{C,G}(x_1, y_1, \ldots, x_j, y_j, x, y)$$

$$\psi = \text{Silence}_{C,G}(x_1, y_1, \ldots, x_j) = \bigvee_{1 \le i \le j} \text{Fixed}_{C,G}(x_1, y_1, \ldots, x_i)$$

An important caveat

$\boxed{\text{Silencing function } \psi \text{ must be efficiently computable!}}$

... at least on the domain that matters: transcripts that **can** arise in G_Π or H_Π (for $\Pi \in C$) interactions with an adversary.

The IND|C paradigm

1. Formalize **syntax** for a scheme Π.
   Formalize the **correctness condition** C.

2. Design **utopian** games G, H (don't exclude "trivial" wins).
   Along with C, this determines the IND|C security notion.

3. Verify that the silencing function $\mathrm{Silence}_{\mathrm{C,G}}$
   is **efficiently computable** on (C,G,H).

A PKE scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is a tuple of 3 algorithms.

$$(pk, sk) \twoheadleftarrow \mathcal{K}(k) \qquad m \leftarrow \mathcal{D}(sk, c)$$

$$c \twoheadleftarrow \mathcal{E}(pk, m)$$

Correctness:

$$(\forall k)(\forall m)[(pk, sk) \twoheadleftarrow \mathcal{K}(k); c \twoheadleftarrow \mathcal{E}(pk, m) \Rightarrow \mathcal{D}(sk, c) = m]$$

Example 1

Conventional IND-CCA-secure PKE

$\boxed{\mathrm{G1}_{\Pi}}$

$\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$

$\boxed{\mathrm{H1}_{\Pi}}$

Initialize(k)
$(pk, sk) \twoheadleftarrow \mathcal{K}(k)$
return

Oracle.Key()
return pk

Oracle.Enc(m)
return c

Oracle.Dec(c)
$m \leftarrow \mathcal{D}(sk, c)$
return m

Finalize(b)
return b

$c \twoheadleftarrow \mathcal{E}(pk, m)$

$c \twoheadleftarrow \mathcal{E}(pk, 0^{|m|})$

Must invalidate trivial wins:

- Exclusion-style
- Penalty-style

If a Dec(c) follows
$c \leftarrow$ Enc(m)
return 0

If a Dec(c) follows
$c \leftarrow$ Enc(m)
return 0

Example 1

IND|C-style CCA-secure PKE

Defining IND|C-CCA security for a PKE scheme Π=(K,E,D)

$$\boxed{G1} \qquad \boxed{H1}$$

Initialize(k)
$(pk, sk) \twoheadleftarrow \mathcal{K}(k)$
return

Oracle.Key()
return pk

Oracle.Enc(m)
$c \twoheadleftarrow \mathcal{E}(pk, m)$ (G1) $\qquad c \twoheadleftarrow \mathcal{E}(pk, 0^{|m|})$ (H1)
return c

Oracle.Dec(c)
$m \leftarrow \mathcal{D}(sk, c)$
return m

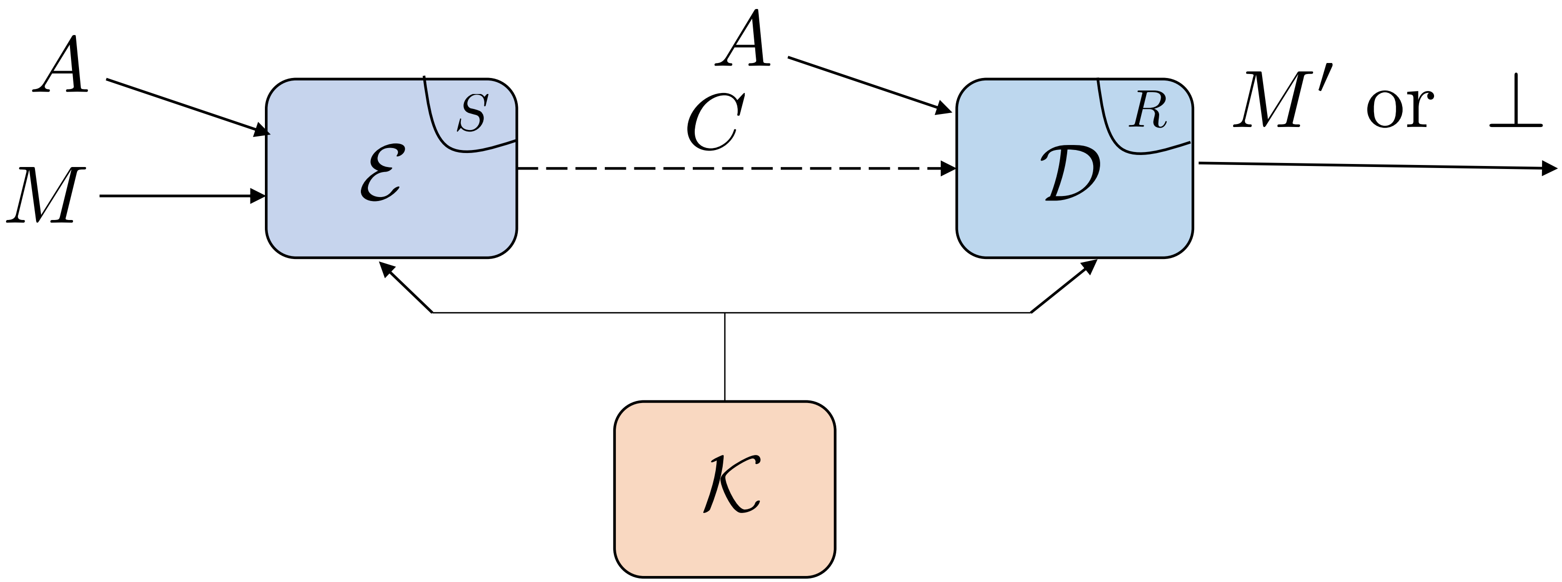
Finalize(b)
return b

$$\begin{aligned} \mathrm{C1} := \{\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})\colon &(\forall k)(\forall m) \\ &[(pk, sk) \twoheadleftarrow \mathcal{K}(k); \\ &\quad c \twoheadleftarrow \mathcal{E}(pk, m)\colon \\ &\quad \mathcal{D}(sk, c) = m]\} \end{aligned}$$

Theorem: IND|C-style CCA security is equivalent to conventional CCA security.

Example 2

Bellare, Kohno, Namprempre (2002/2004)
Kohno, Palacio, and Black (2003)
Boyd, Hale, Mjølsnes, and Stebila (2016)



$$\mathcal{E}\colon \mathcal{K} \times \mathcal{A} \times \mathcal{M} \times \mathcal{S} \to (\mathcal{C} \cup \{\perp\}) \times \mathcal{S}$$
$$\mathcal{D}\colon \mathcal{K} \times \mathcal{A} \times \mathcal{C} \times \mathcal{S} \to (\mathcal{M} \cup \{\perp\}) \times \mathcal{S}$$

Defining correctness

How picky should the receiver be?

Encrypting party sends messages 1, 2, 3, ...
A **level set** $L \subseteq \mathbb{N}^*$ defines the set of **permissible orderings** for the receiver to have received at some point in time.
$\mathbf{n} \in L$ means getting messages $\mathbf{n}$, in order, is acceptable.

C2(L) is the set of all sAE schemes $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ that satisfy:

C2[L]

$$(\forall k \in \mathcal{K})\,(\forall (a_1, m_1), (a_2, m_2), \ldots \in \mathcal{A} \times \mathcal{M})\,(\forall (n_1, \ldots, n_\beta) \in L)$$
$$\big[\; s_0 \leftarrow \varepsilon;\; r_0 \leftarrow \varepsilon;\; \alpha \leftarrow \max(n_1, \ldots n_\beta);$$
$$\textbf{for } i \leftarrow 1 \textbf{ to } \alpha \textbf{ do } (c_i, s_i) \leftarrow \mathcal{E}(k, a_i, m_i, s_{i-1});$$
$$\textbf{for } i \leftarrow 1 \textbf{ to } \beta \textbf{ do } (m'_i, r_i) \leftarrow \mathcal{D}(k, a_{n_i}, c_{n_i}, r_{i-1}):$$
$$((\forall i \in [1..\alpha])\;(c_i \neq \bot)) \Rightarrow ((\forall i \in [1..\beta])\;(m'_i = m_{n_i}))\big]$$

Defining sAE

$\mathrm{G2}_\Pi$

procedure Initialize
$k \twoheadleftarrow \mathcal{K}$

procedure Enc(a, m)
$(c, s) \twoheadleftarrow \mathcal{E}(k, a, m, s)$
return c

procedure Dec(a, c, σ)
$(m, r) \leftarrow \mathcal{D}(k, a, c, r)$
if σ **then return** $b \twoheadleftarrow \{0, 1\}$
return m

$\mathrm{H2}_\Pi$

procedure Initialize
$k \twoheadleftarrow \mathcal{K}$

procedure Enc(a, m)
$(c, s) \twoheadleftarrow \mathcal{E}(k, a, 0^{|m|}, s)$
return c

procedure Dec(a, c, σ)
if σ **then return** $b \twoheadleftarrow \{0, 1\}$
return $\perp$

We have an sAE construction that satisfies our IND|C CCA security notion.

IND|C variants

All of these as expressive as initial version
(with efficient computability side conditions)

1. **Silence-then-forgive**: instead of silence-then-shut-down

2. **Ideal-side editing**: Don't silence G; instead, replace H responses with G responses if those are fixed

3. **Penalty-style editing**: Don't silence: adjust Finalize so that the game outputs 0 if silencing would have happened

4. **Symmetric silencing**: For left-or-right games. Silence a query response if it is (a) fixed for a left-hand oracle, (b) fixed for a right-hand oracle, and (c) these fixed values are distinct

Final comments

Definitions coming out of IND|C are **abstract**
(but can be concretely re-characterized)**.**

A **speculative** proposal
(but we expect broadly applicable).

Might cover some of what **UC** does.
(ideal game $\cong$ ideal functionality)