Security Analyses on Key Exchange Protocols using Task-structured PIOA Framework

米山 一樹
國分 雄一
國廣 昇
太田 和夫

電気通信大学

本発表の概要

- Task-PIOAフレームワークを用いた鍵交換プロトコル（Diffie-Hellman）の安全性解析
  - DDH仮定の再定式化
  - DHプロトコルのtask-PIOAフレームワークにおける表現（Real system）
  - 鍵交換理想機能のtask-PIOAフレームワークにおける表現（Ideal system）
  - Real systemとIdeal systemがimplementation relationを満たすことの証明

目次

- **背景, 研究の目的**
- Task-structured PIOA (task-PIOA) **フレームワーク**
  (PIOA : **確率的**I/O**オートマトン**)
  - task-PIOA **の構造**
  - **識別不可能性** (Implementation Relation)
  - **安全性証明の流れ**
- Diffie-Hellman**の鍵交換プロトコルの解析**
  - DDH(Decisional Diffie-Hellman)**仮定**
  - Real System **と** Ideal System**の構成**
  - **安全性証明**
- **まとめ**

背景

- Task-PIOAフレームワーク [CCK+06]
  - 数理的モデルを用いた暗号プロトコルの安全性検証手法の一つ
  - 数理的モデルのメリット：**自動検証**が可能
  - Dolev-Yaoモデルや，それを基にしたフレームワークでは難しい
    計算量的困難性を扱うことが可能である（とされている）．

- しかし…実際の解析例は紛失通信（GMWプロトコル）のみ
  - 計算量的困難性：hard-core predicate for trap-door permutation
  - 攻撃者のモデル：passive&static

- **その他のプロトコル/プリミティブの記述能力は？**
- active or adaptiveな攻撃者の定式化は？

研究の目的

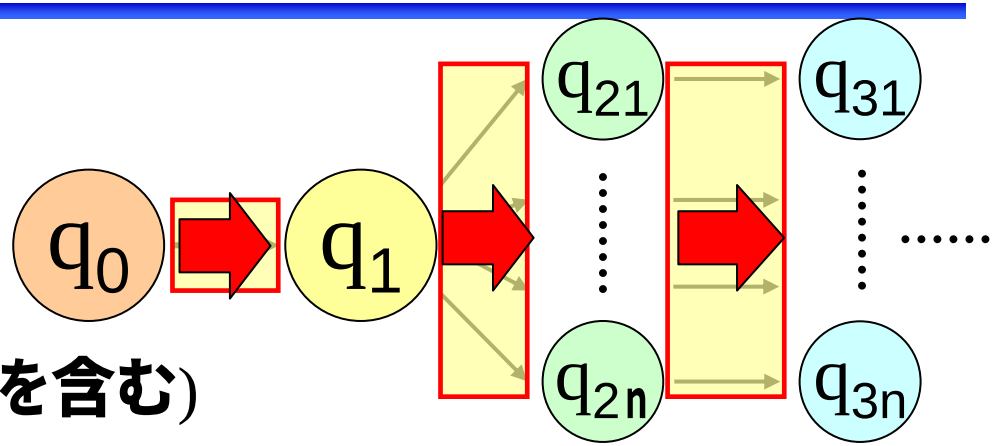
- **紛失通信以外のプロトコルの安全性解析**
  - **今回は鍵交換** (Diffie-Hellman) **プロトコルに適用する.**
  - **計算量的困難性**：Diffie-Hellman**判定** (DDH) **問題**
  - **攻撃者のモデル**：Passive&Static

- active or adptive**な攻撃者の定式化**
  - **中間者** (Man-in-the-Middle) **攻撃**
  - **理想機能との対話インターフェイス**

Task-PIOAフレームワーク

task-PIOA の構造 1/2

- task-PIOA $T = (P, R)$

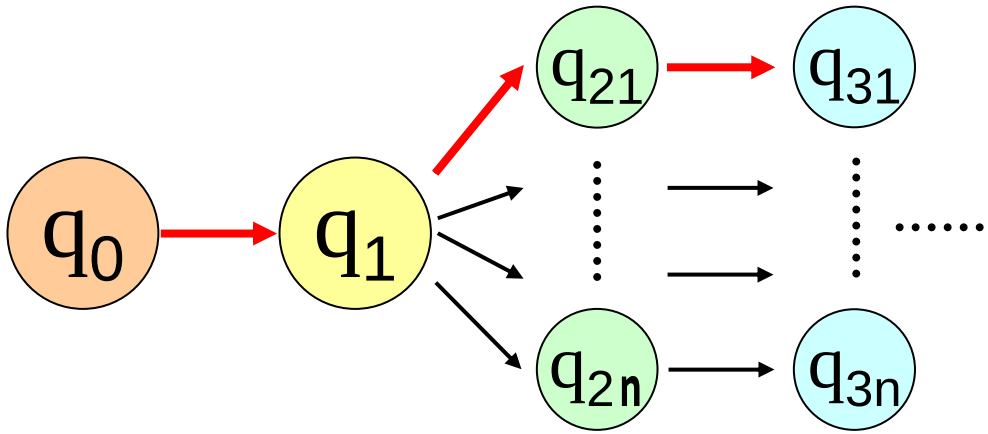


  - P : PIOA
    - **状態集合** Q (**初期状態を含む**)
    - **入力**, **出力**, **内部** action I, O, H (I∪O∪H = A : action)
    - (**確率的**) **遷移関数**
  - R : O∪H **を“同種な”**action **に分割 →** task
    - output(‘1’), output(‘2’) ... : O → output(∗) : **値を出力する** task
    - PIOA **の一連の動作** = task **の列** : task scheduler **ρ**

$$\rho = T_1\, T_2\, T_3 \cdots \quad (T_i \text{ は task})$$

(input action **はいつでも実行可**)

task-PIOA の構造 2/2



- ***trace*** : **一連の動作中の，入力，出力**action **の集合**

$$trace(\alpha) = \{a_i \in \alpha \mid a_i \in I \cup O\}$$

- ***tdist*(ρ)　：　*trace* の確率分布**
- ***tdists*(*T*) :　*tdist* の集合**

$$tdists(T) = \{tdist(\rho) \mid T \text{ は } \rho \text{ を受け付ける}\}$$

識別不可能性の表現

- Implementation Relation：$T_1 \leq_0 T_2$

定義　**どのようなEnvに対しても，**

$$\mathrm{tdists}(T_1 \,\|\, Env) \subseteq \mathrm{tdists}(T_2 \,\|\, Env)$$

が成り立つ．

・**推移性**

$T_1 \leq_0 T_2, T_2 \leq_0 T_3 \Rightarrow T_1 \leq_0 T_3$

・**結合性**

$T_1 \leq_0 T_2 \Rightarrow T_1 \| T_3 \leq_0 T_2 \| T_3$

識別不可能性の表現

- **計算量的な**Implementation Relation：$T_1 \leq_{neg,pt} T_2$

  2**つの** task-PIOA T1, T2 **が多項式時間内で区別できない.**

定義　**どのような多項式時間** Env **と** ρ_1 **に対しても**
多項式時間 ρ_2 **が存在し,**

$$\left|Paccept(T_1 \| Env, \rho_1) - Paccept(T_2 \| Env, \rho_2)\right| \leq negl.$$

を満たす.

同様に推移性, 結合性を持つ.

安全性証明の目標

$RS \leq_{neg,pt} IS$ **を示すこと**

- RS : *Real System* = **現実の証明対象プロトコル**
  - **各プロトコル参加者**
  - Adversary
  - Environment
- IS : *Ideal System* = **理想化した証明対象プロトコル**
  - **シミュレータ (プロトコル参加者の抽象化＋**Adversary)
  - Environment (***RS*と同じ**)
  - **理想機能**

証明の流れ

- はじめから $RS \leq_{neg,pt} IS$ の証明を行うことは難しい．
  → $\leq$ の推移性を用い，証明を分割．
  分割の方法は以下の2種類．
- 計算量的仮定を基にした分割
  - 計算量的仮定を $\leq_{neg,pt}$ の式で定義しておく．
  - 仮定の形式に合うように，RS と IS を変形する．
- 中間となるシステム ***Int*** : ***Intermediate System*** を用いた分割
  - ***RS***，計算量的仮定，IS の間を埋めるために用いる．
  - ***Int*** の前後で，$\leq_0$ が成り立つよう構成する．

Diffie-Hellmanの鍵交換プロトコルの解析

DDH仮定

計算量理論的定式化

$$\left|\Pr[A(g, g^a, g^b, g^{ab}) = 1] - \Pr[A(g, g^a, g^b, R) = 1]\right| \geq non\text{-}negl.$$

を満たす PPT **アルゴリズム** A **は存在しない.**

task-PIOA **フレームワークで再定式化する.**

$$SDDH \leq_{neg,pt} SDDHR$$

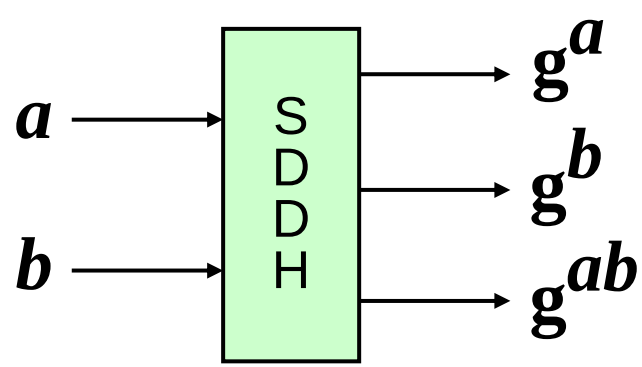
ただし,

$SDDH$ ： 「$(\mathbf{g}^a, \mathbf{g}^b, \mathbf{g}^{ab})$ **を出力する** task-PIOA」,

$SDDHR$ ： 「$(\mathbf{g}^a, \mathbf{g}^b, \boldsymbol{R})$ **を出力する** task-PIOA」.

SDDH と SDDHR

- SDDH



input : rand(a), rand(b)

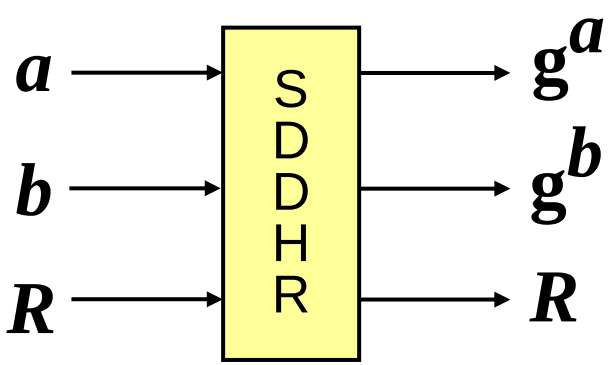
output : out(A), $A = g^a$

out(B), $B = g^b$

out(C), $C = g^{ab}$

internal : fix-Aval, fix-Bval, fix-Cval

- SDDHR



input : rand(a), rand(b), rand(R)

output : out(A), $A = g^a$

out(B), $B = g^b$

out(C), $C = R$

internal : fix-Aval, fix-Bval

DDH仮定の等価性

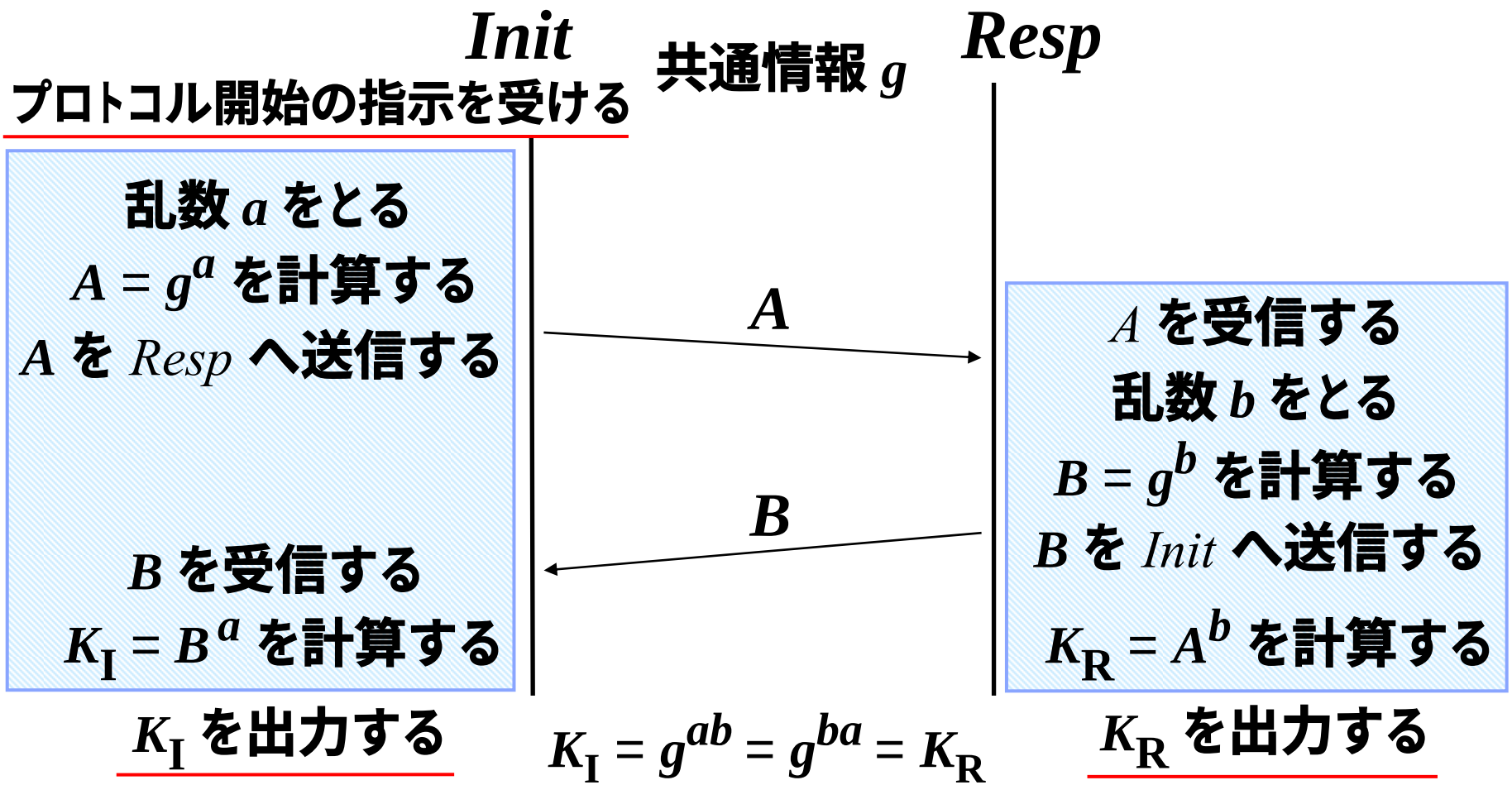
定理

DDH仮定が成り立つことと，

$SDDH \leq_{neg,pt} SDDHR$ が成り立つことは等価である．

Diffie-Hellman の鍵交換プロトコル

- Initiator *Init* と Responder *Resp* は以下の手順で鍵を共有する.



Init の action 一覧 (Code)

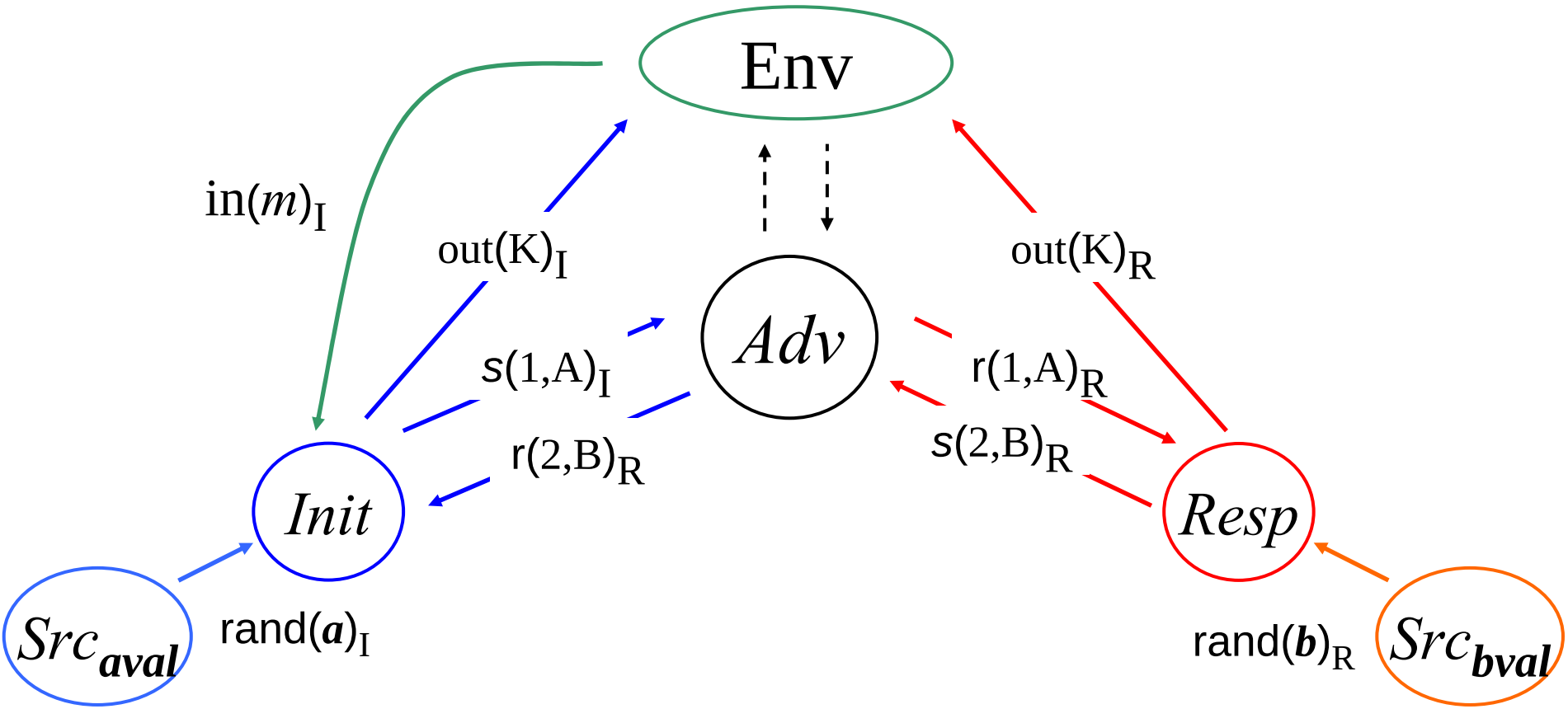
- **in(m)** プロトコル開始の指示
  Effect :
  If ***received*** = ⊥ then ***received*** := ⊤
- **rand(a)$_{\mathrm{Init}}$** 乱数 a をとる
  Effect :
  If ***aval*** = ⊥ then ***aval*** := ***a***
- **fix-Aval** A を計算
  Precondition :
  aval ≠ ⊥, ***Aval*** = ⊥
  Effect :
  Aval := $g^{\boldsymbol{aval}}$
- **send(1, A)** A を送信
  Precondition :
  A = ***Aval*** ≠ ⊥, *received* ≠ ⊥
  Effect :
  none
- **receive(2, B)** B を受信
  Effect :
  If ***Bval*** ≠ ⊥ then ***Bval*** := ***B***
- **fix-Kval$_{\mathrm{Init}}$** K_{I} を計算
  Precondition :
  aval , ***Bval*** ≠ ⊥, ***Kval***$_{\mathrm{Init}}$ = ⊥
  Effect :
  Kval$_{\mathrm{Init}}$:= ***Bval*** $^{\boldsymbol{aval}}$
- **out(K)$_{\mathrm{Init}}$** K_{I} を出力
  Precondition :
  K = ***Kval***$_{\mathrm{Init}}$ ≠ ⊥
  Effect :
  none

RS の構成

- *RS* は，以下のように task-PIOA の結合で表される．

$$RS = Init \parallel Resp \parallel Adv \parallel Env \parallel Src_{aval} \parallel Src_{bval}$$

(*Src* は，乱数を選ぶ task を持つオートマトン．)



IS を構成するために

- 必要なものは次の3つ
  - 理想機能
  - シミュレータ *Sim*
  - *Env* (*RS* と同じ)

- 理想機能とは，プロトコルが提供する機能を理想化したもの．

- シミュレータは，*IR*　　*Init* + *Resp* と *Adv* (*RS* と同じ)からなる．
  - 理想機能からの入力を用いて，処理を行う点が異なる．

鍵交換の理想機能

- **汎用結合可能性** (Universal Composability) **における理想機能** (functionality) **を参考にして構成することが可能.**

鍵交換の理想機能を表すtask-PIOA Funct

i. **環境から指示を受け取る.** $\mathrm{in}(\boldsymbol{m})$

ii. **鍵空間からランダムに鍵をとる.** $\mathrm{rand}(K)_{\mathit{Funct}}$

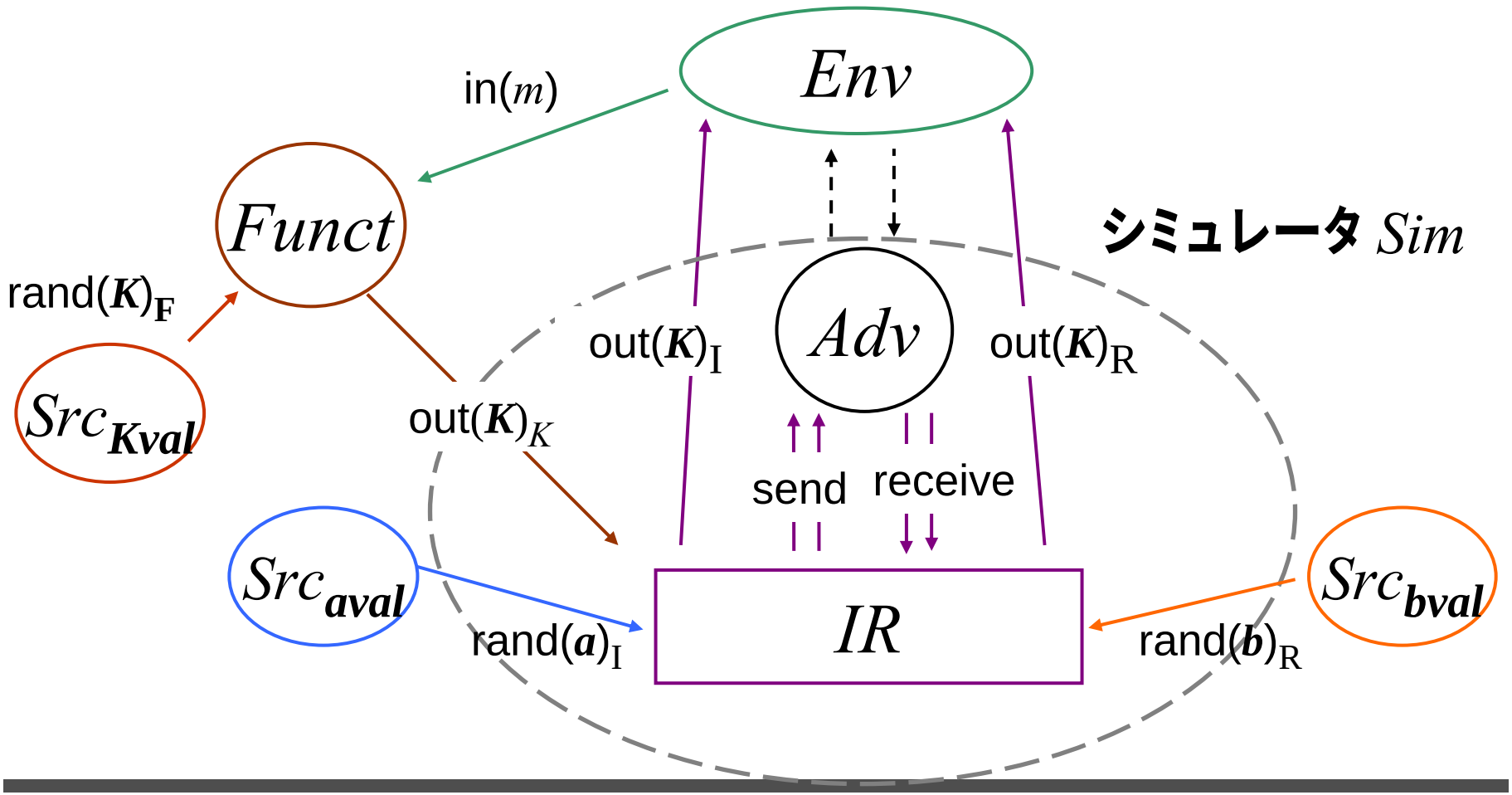
iii. **とった値を出力する.** $\mathrm{out}(K)_{\mathit{Funct}}$

注) 今回は passive&static Adversary **を考えているため,**
便宜的に必要なインターフェイスだけを簡略化して記述した.

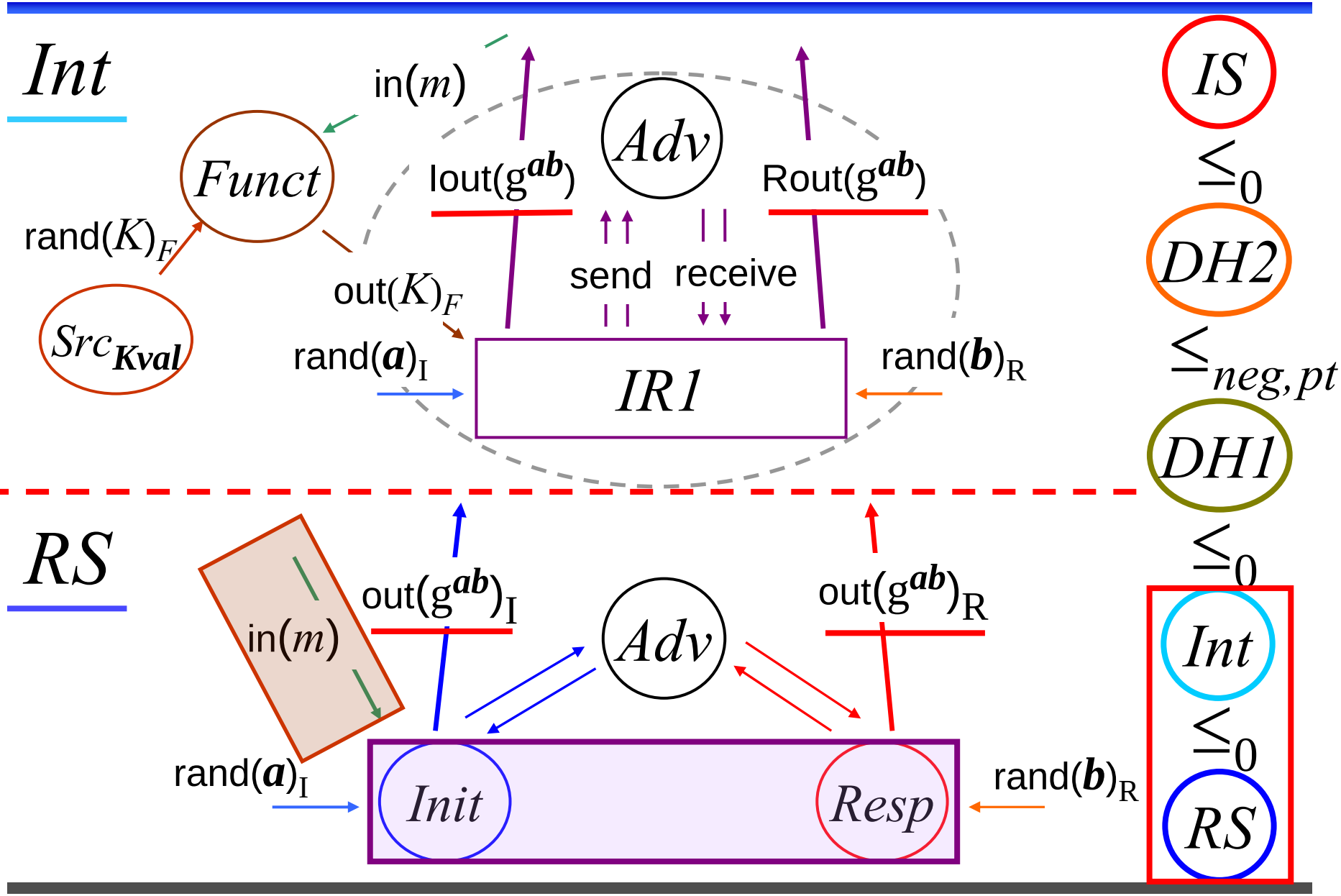
IS の構成

- *IS* は以下の task-PIOA の結合で表される．

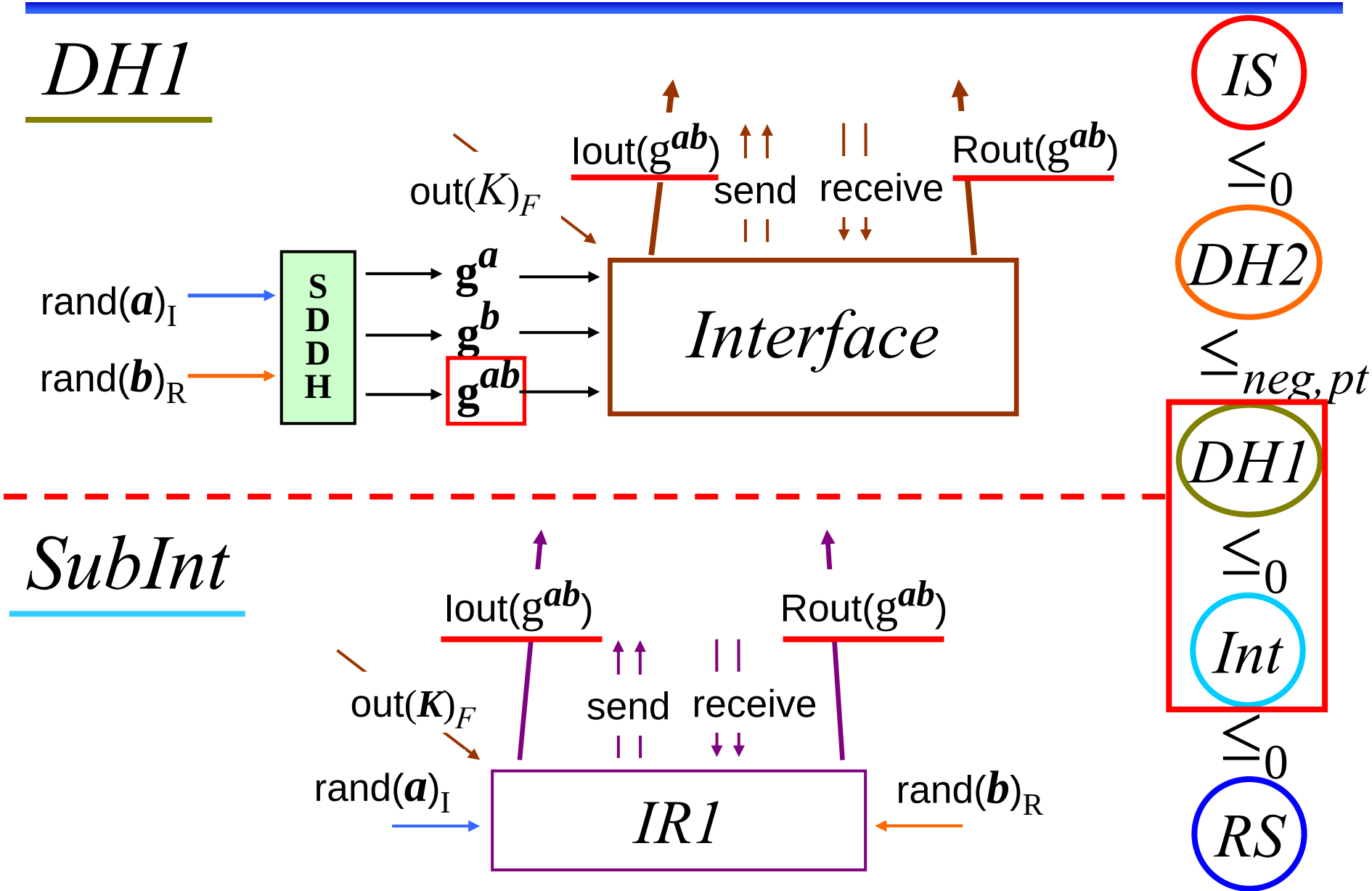
$$IS = Funct \parallel Sim \parallel Env \parallel Src_{aval} \parallel Src_{bval} \parallel Src_{Kval}$$



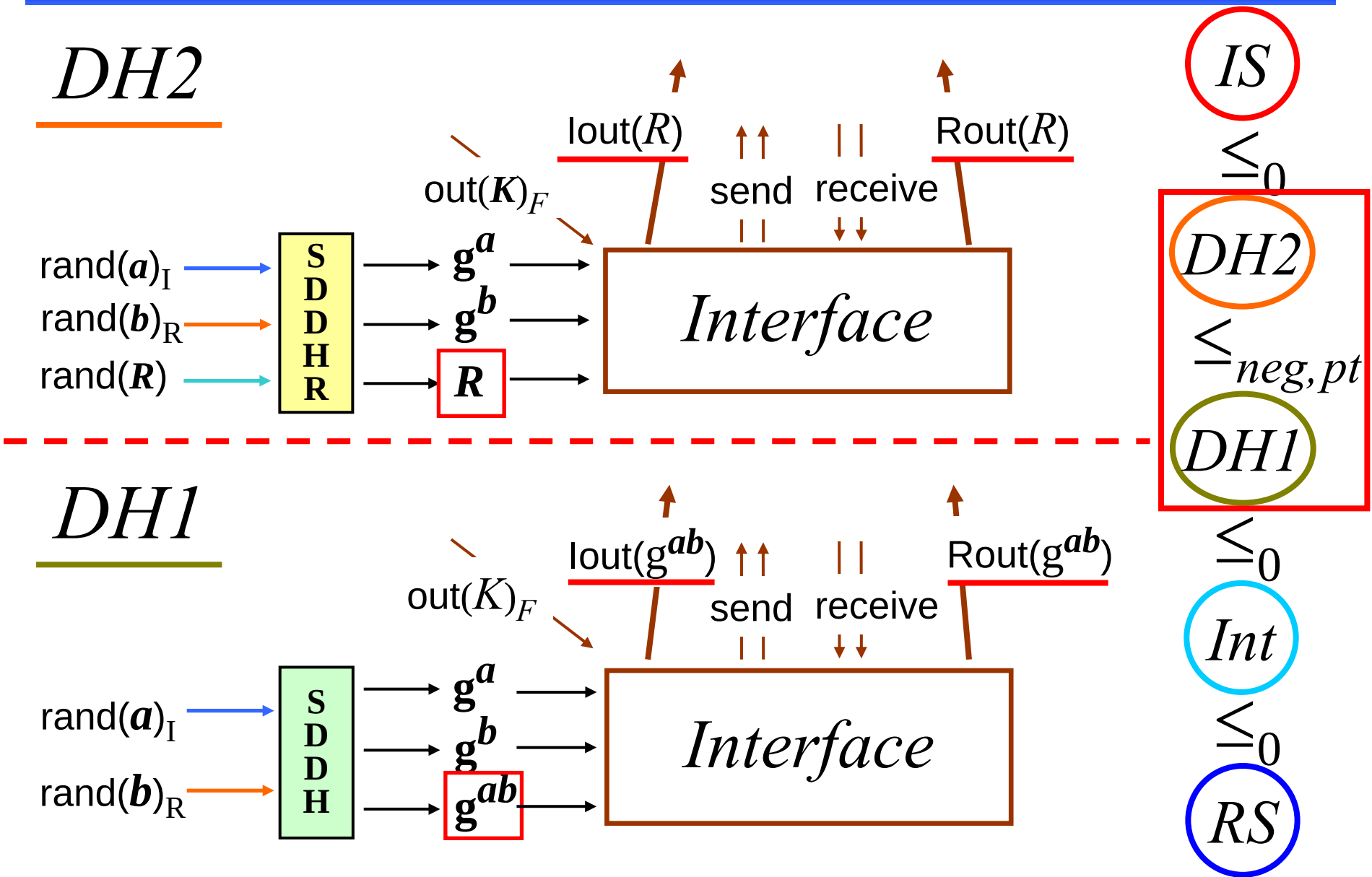
第1ステップ　$RS \leq_0 Int$



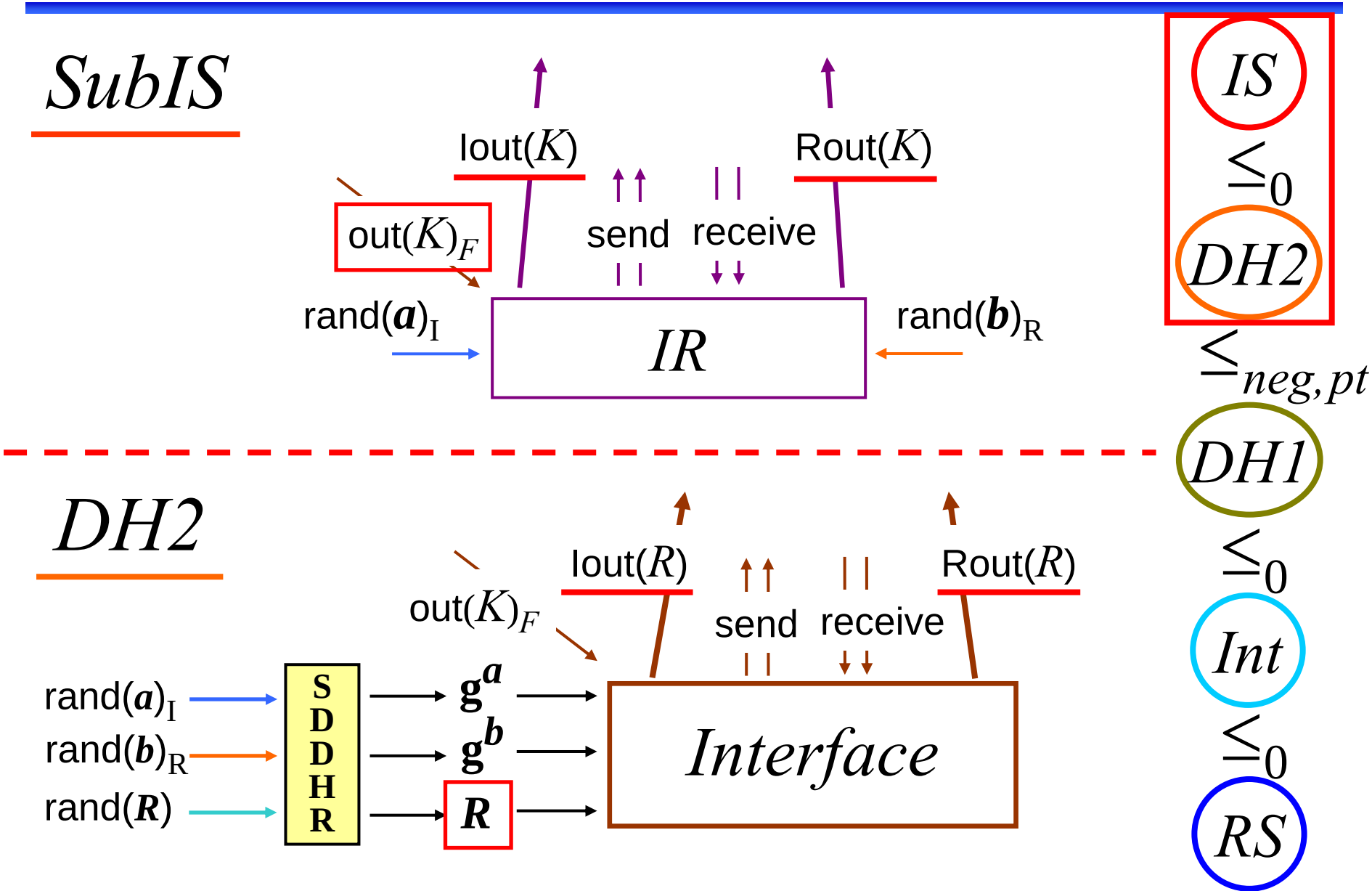
第2ステップ　$SubInt \leq_0 DH1$



第3ステップ　$DH1 \leq_{neg,pt} DH2$



第4ステップ　$DH2 \leq_0 SubIS$



まとめ

- task-PIOA フレームワークの記述能力の確認として，Diffie-Hellman の鍵交換プロトコルの安全性証明を行った．
- task-PIOA フレームワークにおけるDDH仮定の再定式化を行い，それが従来のDDH仮定と等価であることを証明した．
- 鍵交換の理想機能を，task-PIOA フレームワークの形式に変換した．