

# 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

- 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  $I, O, H$  ( $I \cup O \cup H = A$  : action)

■ (確率的) 遷移関数

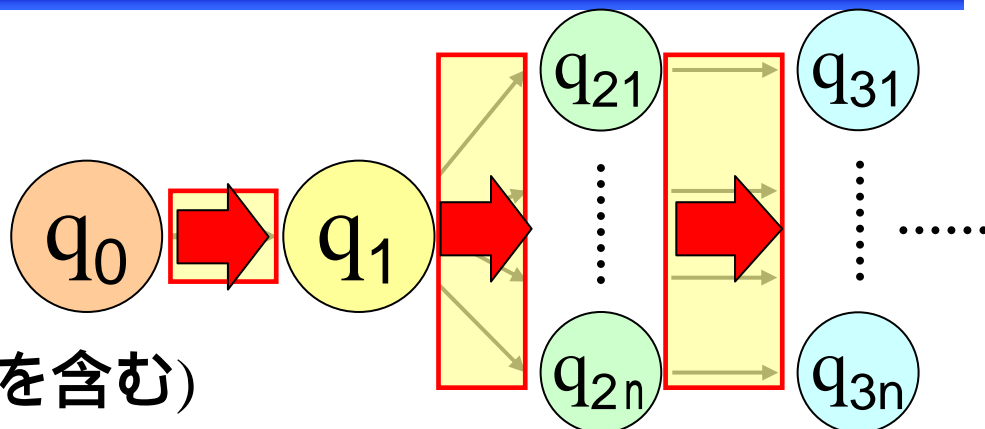
$R$  :  $O \cup H$  を“同種な”action に分割 task

■  $output('1'), output('2') \dots$  :  $O$  output(\*) : 値を出力する task

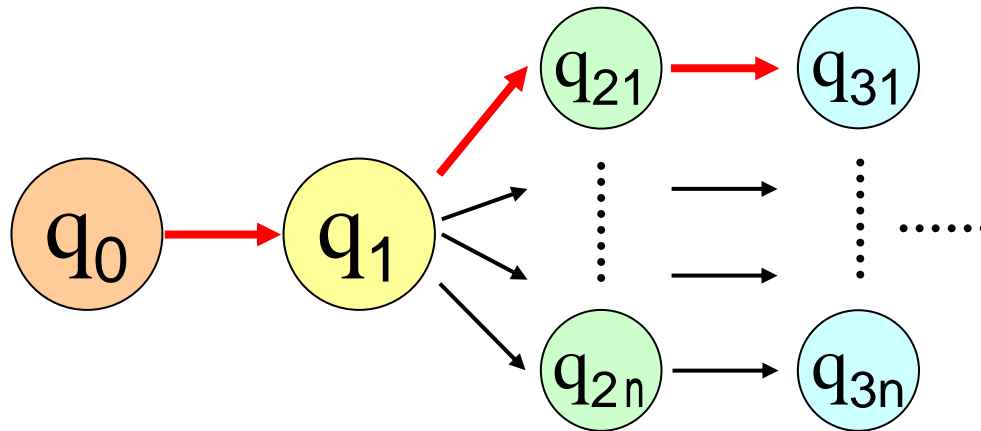
■ PIOA の一連の動作 = task の列 : task scheduler

$$= T_1 T_2 T_3 \dots \quad (T_i \text{ は task})$$

(input action はいいつでも実行可)



# task-PIOA の構造 2/2



- *trace* : 一連の動作中の, 入力, 出力action の集合

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

- *tdist*( ) : *trace* の確率分布

- *tdists*(*T*) : *tdist* の集合

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



# 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,  
 $\text{tdists}(T_1 \parallel Env) \subseteq \text{tdists}(T_2 \parallel Env)$   
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

$$T_1 \leq_0 T_2 \Rightarrow T_1 \parallel T_3 \leq_0 T_2 \parallel T_3$$

# 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  $T_1, T_2$  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  $\rho_1$  に対しても  
多項式時間  $\rho_2$  が存在し,

$$\left| P_{accept}(T_1 \parallel Env, \rho_1) - P_{accept}(T_2 \parallel Env, \rho_2) \right| \leq negl.$$

を満たす。

同様に推移性, 結合性を持つ。

# 安全性証明の目標

---

$$\underline{RS \leq_{neg,pt} IS \text{ を示すこと}}$$

- **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.

$IS$

$\leq_0$

$DH2$

$\leq_{neg,pt}$

$DH1$

$\leq_0$

*Int*

$\leq_0$

$RS$

# 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

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

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。

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

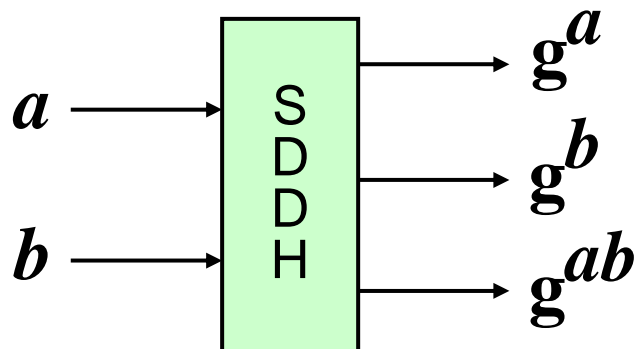
ただし,

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

$SDDHR$  : 「 $(g^a, g^b, R)$  を出力する task-PIOA」.

# SDDH と SDDHR

## □ SDDH

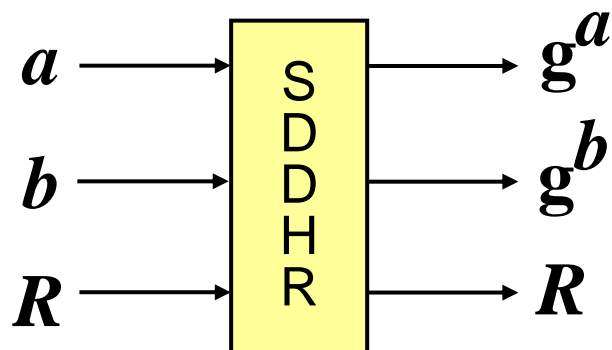


input : rand( $a$ ), rand( $b$ )

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

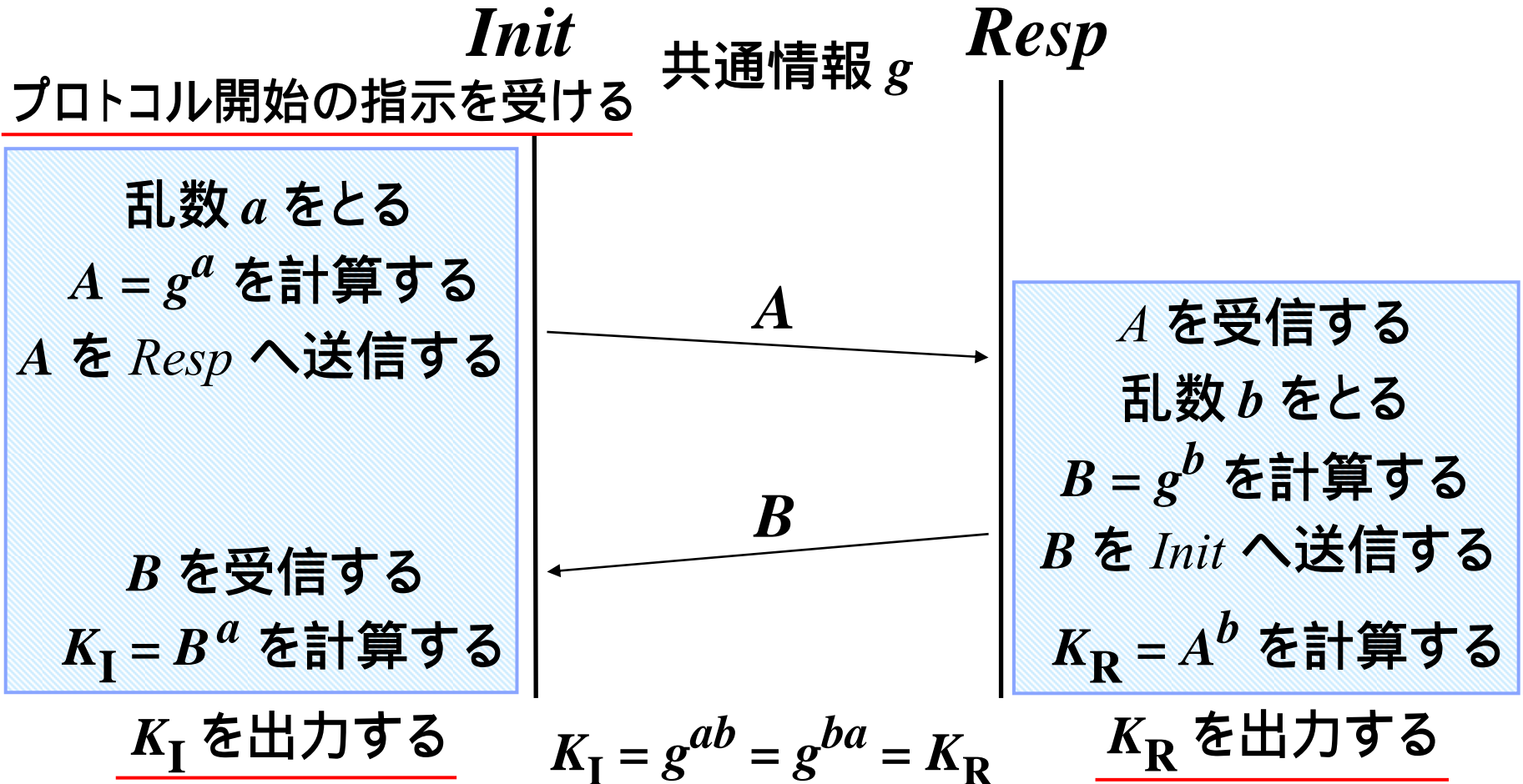
DDH仮定が成り立つことと,

$SDDH \leq_{neg,pt} SDDHR$  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  $received := \top$

- **rand( $a$ )<sub>Init</sub>**      乱数  $a$  をとる

Effect :

If  $aval =$       then  $aval := a$

- **fix-Aval**       $A$  を計算

Precondition :

$aval$       ,  $Aval =$

Effect :

$Aval := g^{aval}$

- **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<sub>Init</sub>**       $K_I$  を計算

Precondition :

$aval$  ,  $Bval$       ,  $Kval_{Init} =$

Effect :

$Kval_{Init} := Bval^{aval}$

- **out( $K$ )<sub>Init</sub>**       $K_I$  を出力

Precondition :

$K = Kval_{Init}$

Effect :

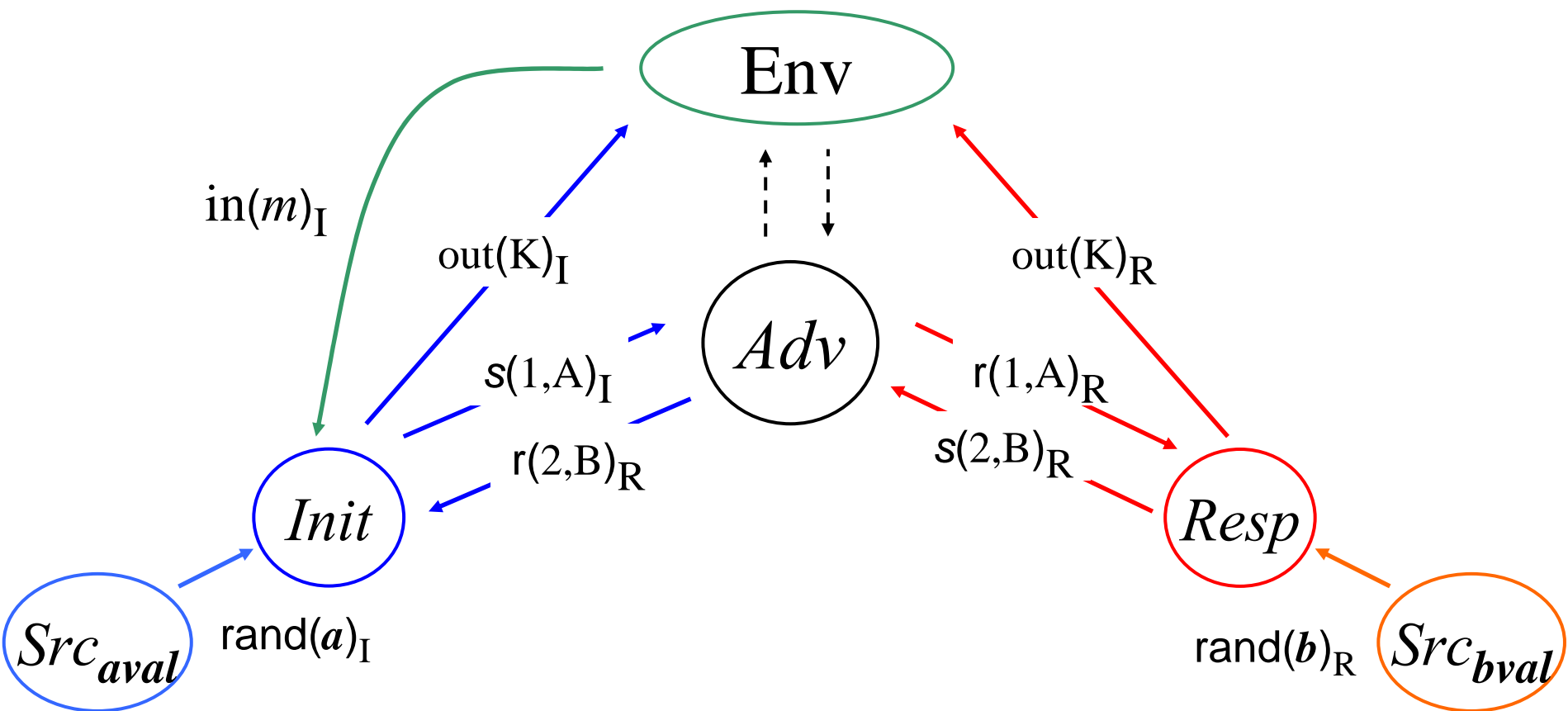
none

# 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 *Func<sub>t</sub>*

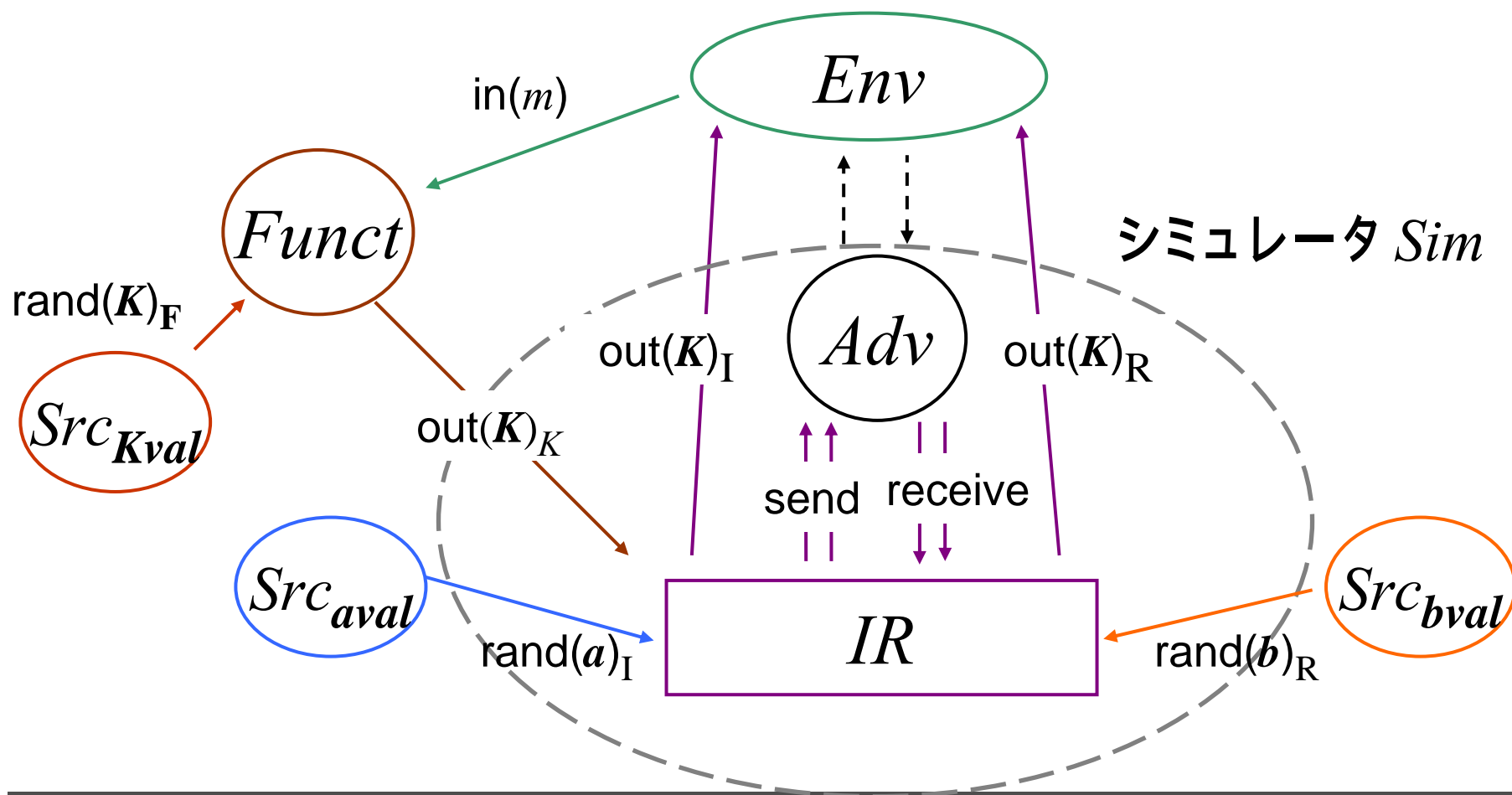
- i. 環境から指示を受け取る .  $\text{in}(m)$
- ii. 鍵空間からランダムに鍵をとる .  $\text{rand}(K)_{\text{Func}_t}$
- iii. とった値を出力する .  $\text{out}(K)_{\text{Func}_t}$

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 .

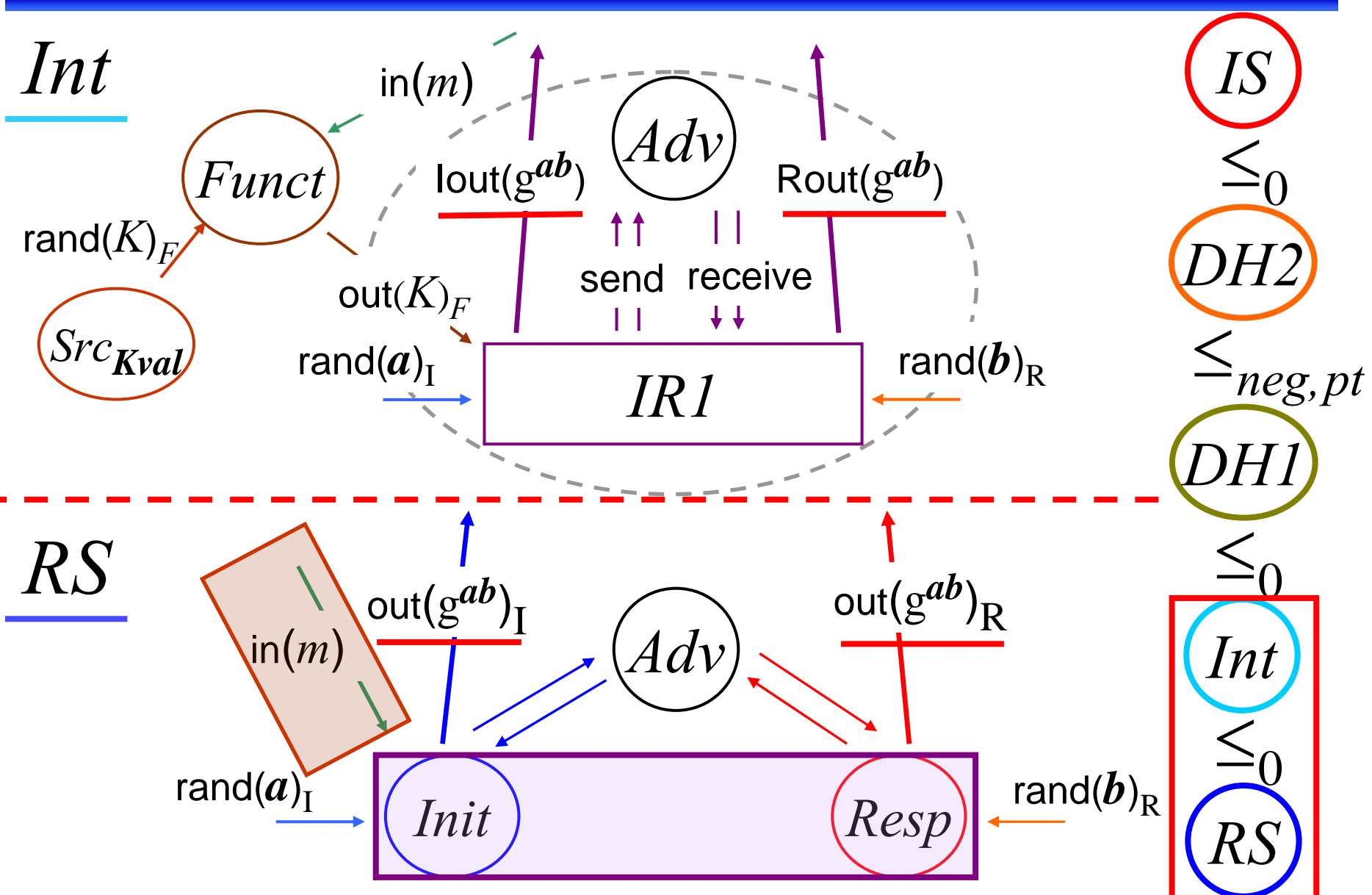
# IS の構成

- IS は以下の task-PIOA の結合で表される。

$$IS = \text{Funct} \parallel \text{Sim} \parallel \text{Env} \parallel \text{Src}_{\text{aval}} \parallel \text{Src}_{\text{bval}} \parallel \text{Src}_{\text{Kval}}$$

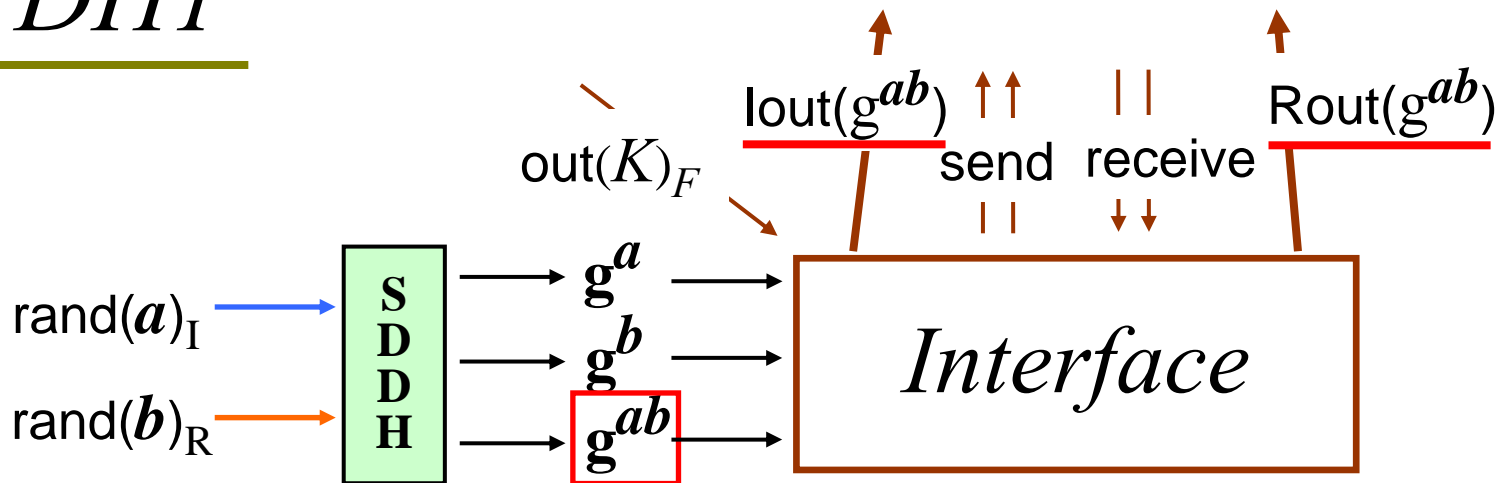


# 第1ステップ $RS \leq_0 Int$



# 第2ステップ $SubInt \leq_0 DH1$

DH1



$IS$

$\leq_0$

$DH2$

$\leq_{neg,pt}$

$DH1$

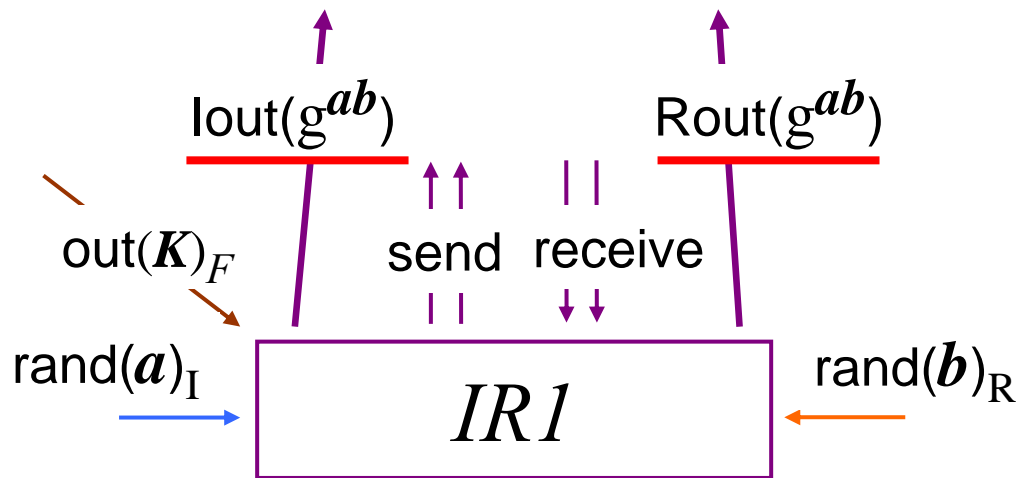
$\leq_0$

$Int$

$\leq_0$

$RS$

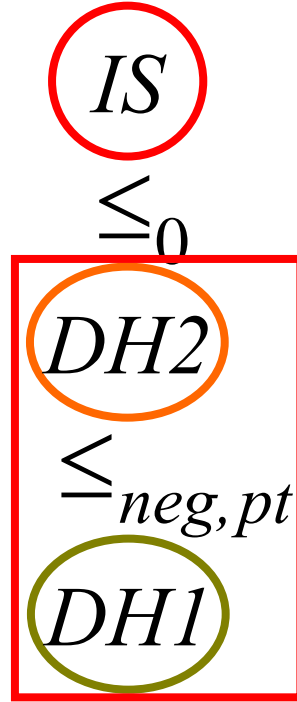
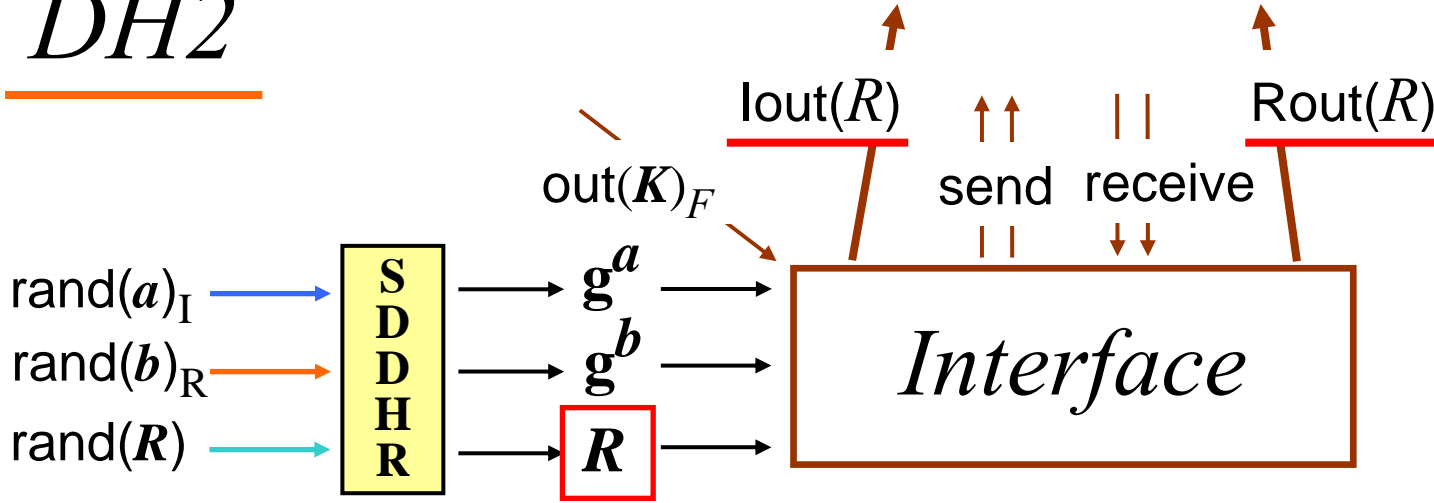
SubInt



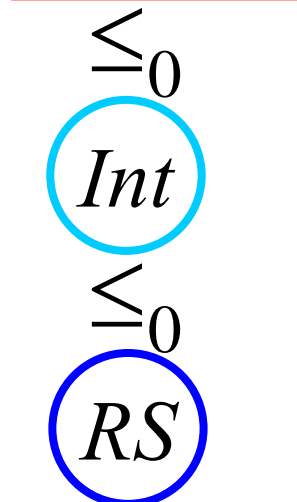
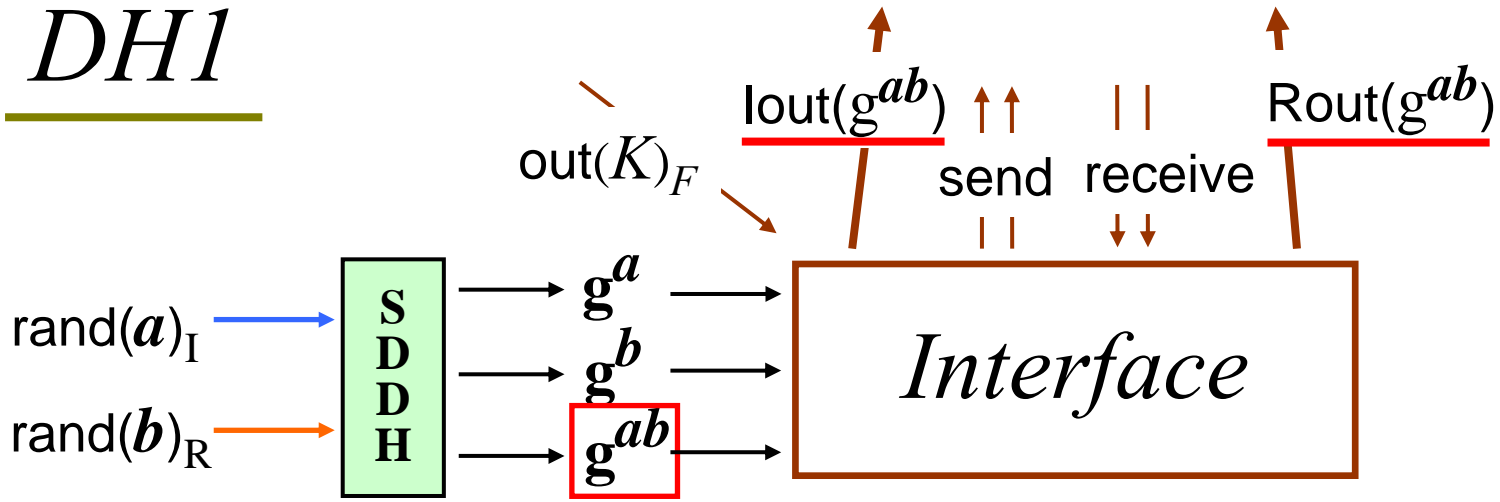


# 第3ステップ $DH1 \leq_{neg,pt} DH2$

## DH2

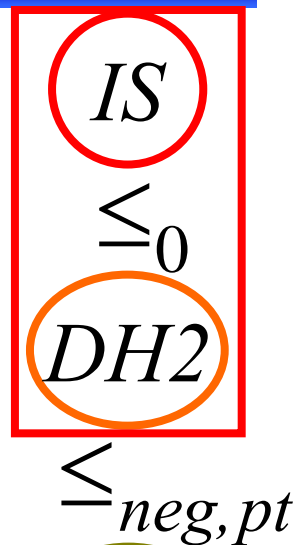
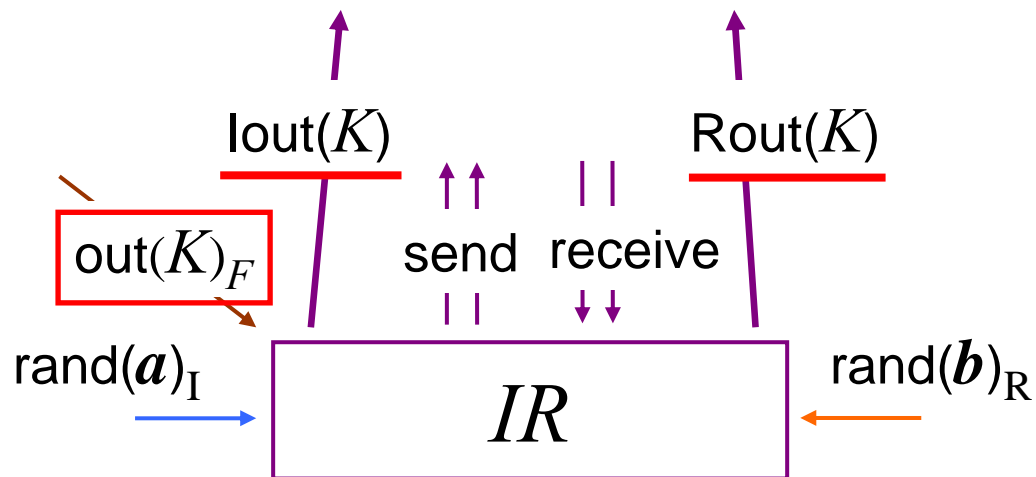


## DH1

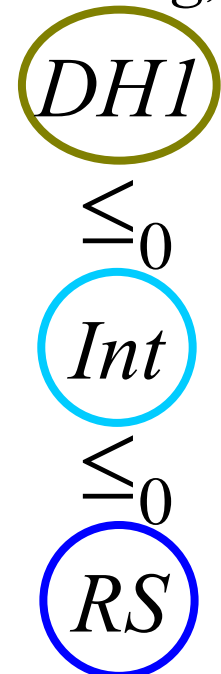
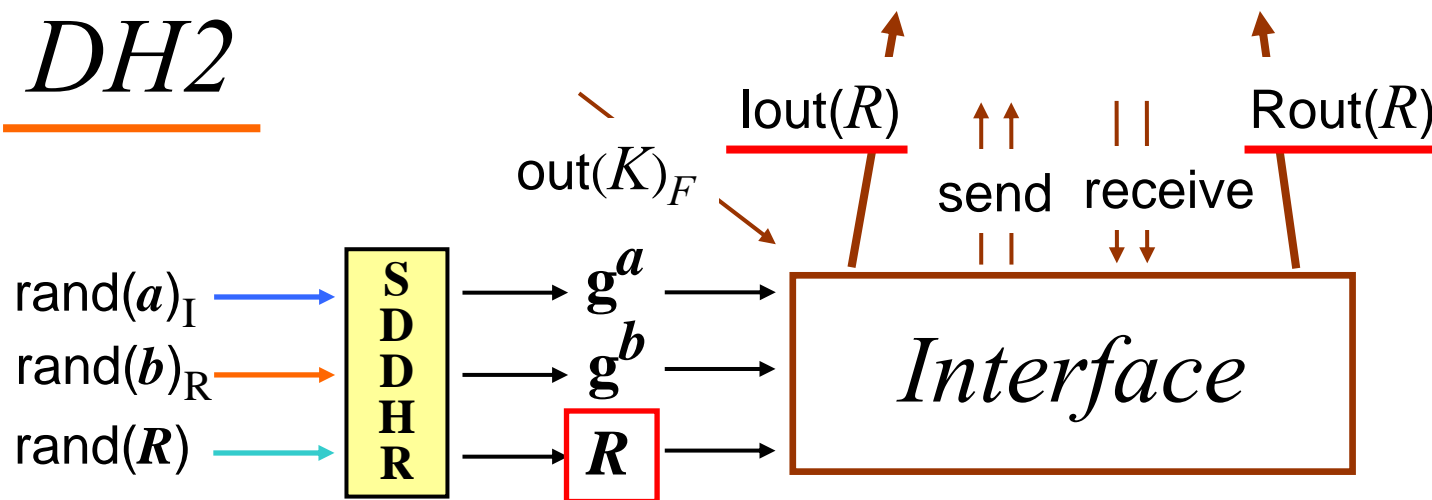


# 第4ステップ $DH2 \leq_0 SubIS$

## SubIS



## DH2



# 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。