TOSHIBA
Leading Innovation

task-PIOAフレームワークとBlanchetのフレームワークの証明能力に関する一考察

○ 花谷 嘉一 †
米山 一樹 ‡
國分 雄一 ‡
太田 和夫 ‡

† (株) 東芝 研究開発センター
‡ 電気通信大学

目次

- 背景
- 安全性証明の手順
- task-PIOAフレームワーク
- Blanchetのフレームワーク
- 比較
- まとめ

背景

暗号プロトコルの安全性を主張する方法

- 根拠は示せないけど，安全だと説得する.
- 既存の攻撃法が適用できないことを示す.
- 世界中の人が攻撃を試みてるけど，破れない.
- 攻撃をモデル化し，攻撃モデルに含まれるあらゆる攻撃法に対する安全性を証明する.

→ 安全性証明.

複雑で手間がかかる.
誤りを見落とすことも有.

→ 手間の軽減や誤りを防止するために，
安全性証明を形式的検証で行う方法が
注目をされている.

形式的検証のフレームワーク

- Blanchet-Pointcheval
- task-PIOA
- 確率Hore論理
- 確率プロセス計算
- BPW
- Coq

  などなど・・・

なぜ task-PIOA と Blanchet のフレームワーク？

たまたま この二つのフレームワークに触れる機会があった.

- **フレームワークにおける証明の戦略や見た目が全然違う。**
  - 取り扱うことができる暗号プロトコルに差はないか？
  - フレームワークで保証できる安全性は同じなのか？

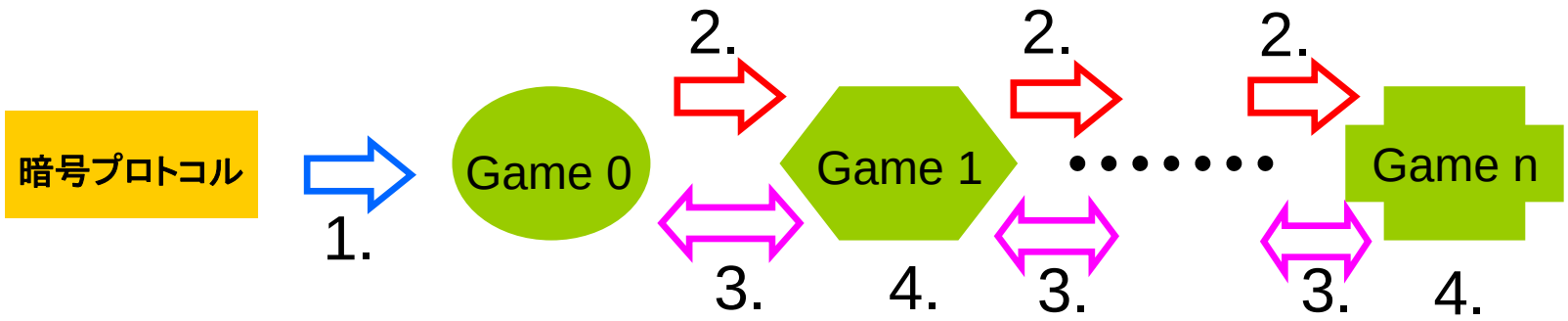
比較してみよう！

“Automated Security Proofs with Sequences of Games”
B. Blanchet　CRYPTO 2006

“Using Probabilistic I/O Automata to Analyze An Oblivious Transfer Protocol”
R. Canetti et. al.　e-Print

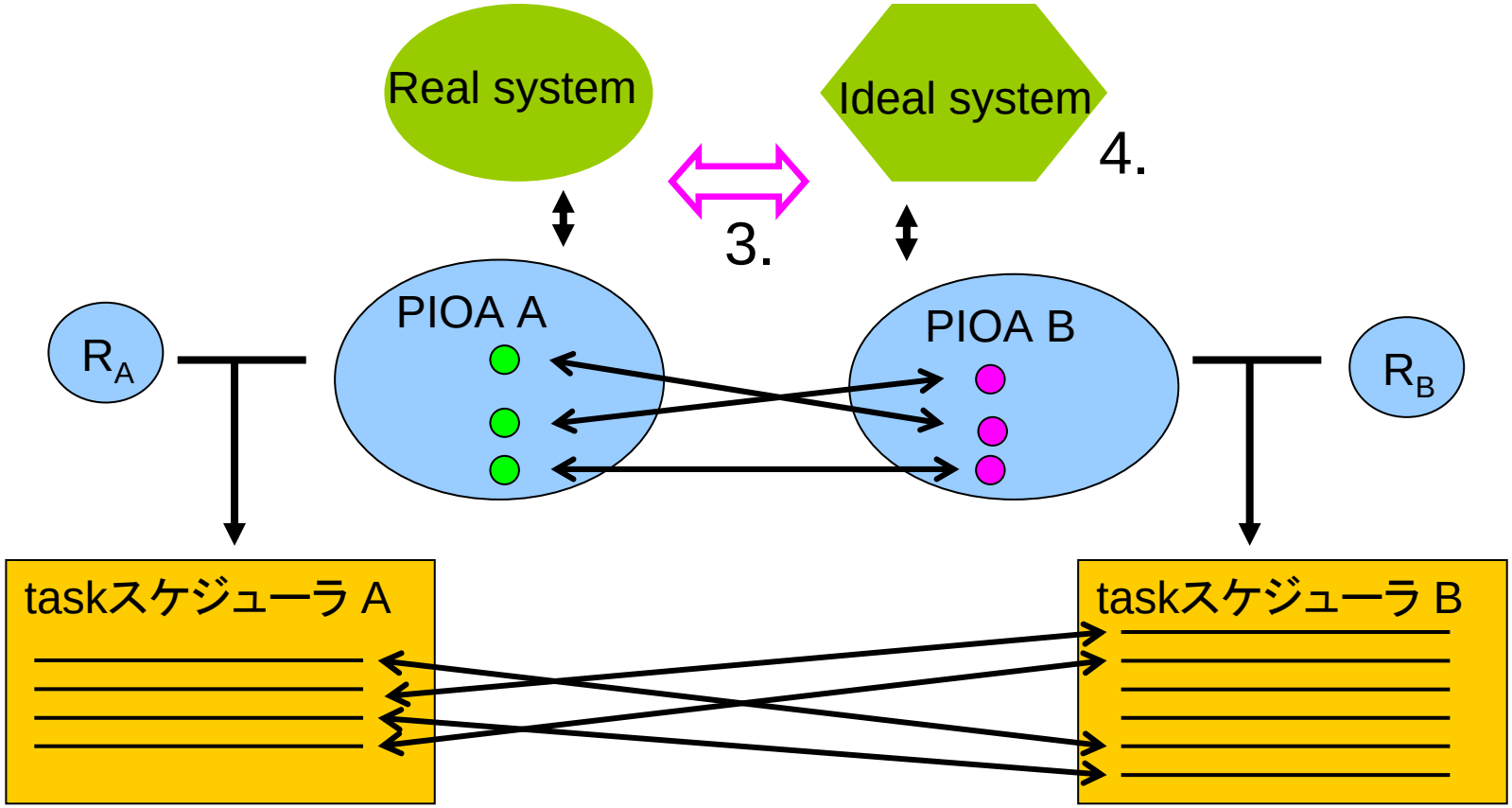
安全性証明の手順

- 帰着による証明
  - 実際の攻撃Gameと計算量的仮定の下では攻撃に成功しないGameとの差を評価.
- シミュレーションパラダイムによる証明
  - 実際の攻撃Gameと攻撃成功の余地が無い理想的なGameとの差を評価.



1. 攻撃ゲームをモデル化
2. 攻撃ゲームの一部を変形し，新たなゲームを作る.
3. ゲーム間の差を評価する.
4. Gameが安全性用件を満たすか調べる.
   ⇒ 満たしていたら証明終了.
5. 2～4を繰り返す.

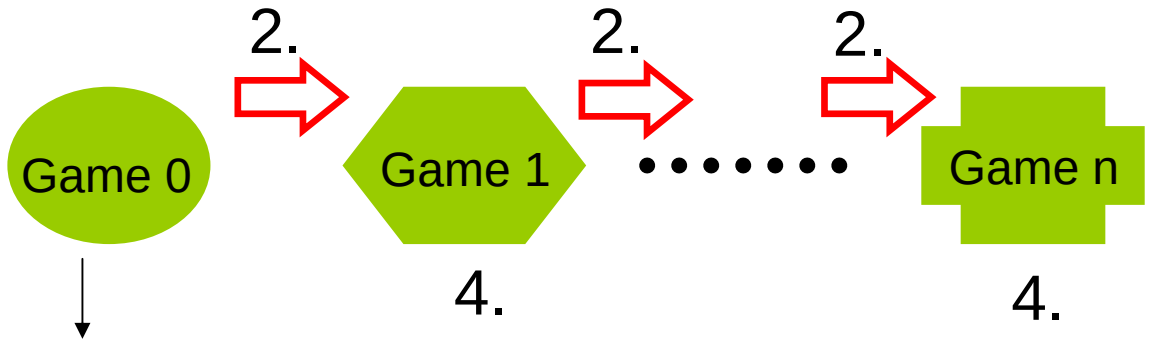
task-PIOAフレームワーク



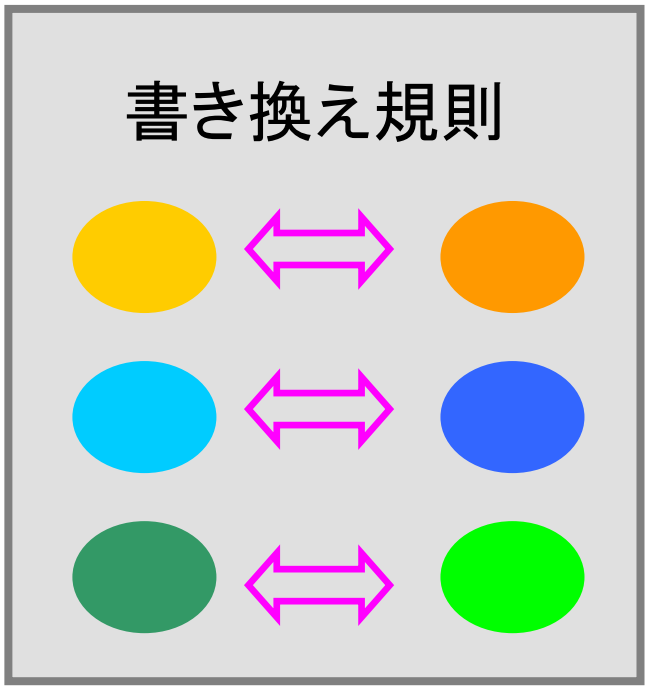
内部変数とtaskスケジューラに対応関係を定義し，対応するtaskスケジューラを実行したときに内部変数の対応関係が壊れないか検証する.

A の全てのtaskスケジューラに対して B に対応するtaskスケジューラが存在するとき，Real systemは安全であるという.

Blanchet のフレームワーク



Blanchetのプロセス計算で表現



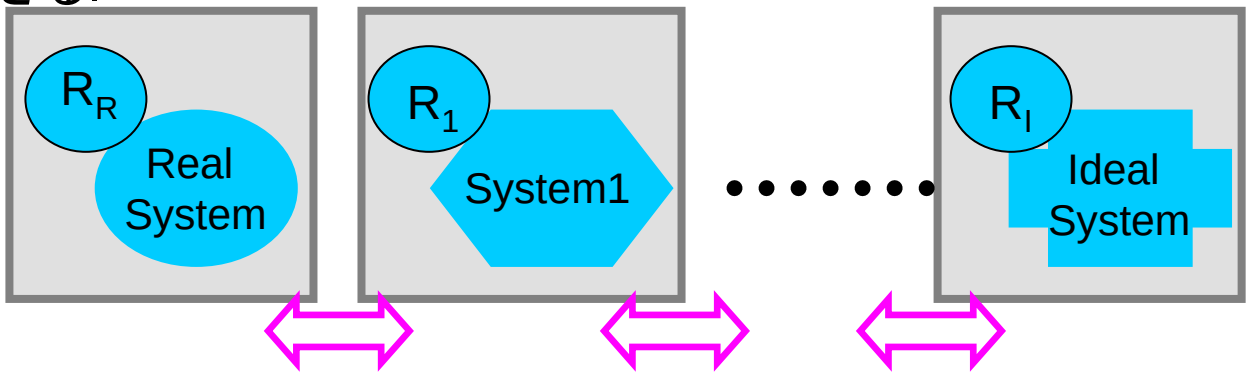
書き換え規則を適用し新たなゲームを生成.

変換後のGameが安全性要件を満たすか調べる.

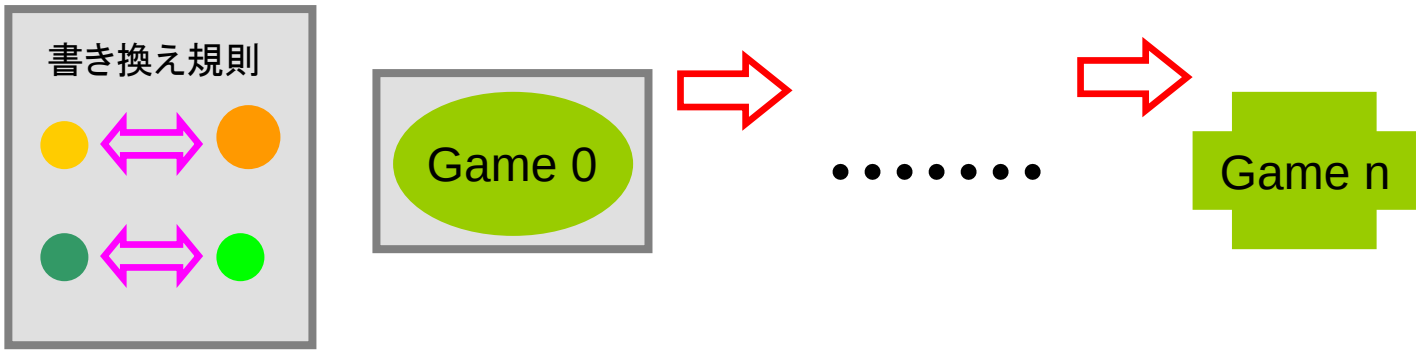
変換できたら検証を終了する.

安全性証明ができたとしても・・・

- **task-PIOA**
  - systemを正しくPIOAで表現して，Rが適切に設定されていれば安全性を保証できる.



- **Blanchet**
  - Game0 を正しくプロセス計算で表現して，書き換え規則が適切に設定されていれば安全性を保証できる.



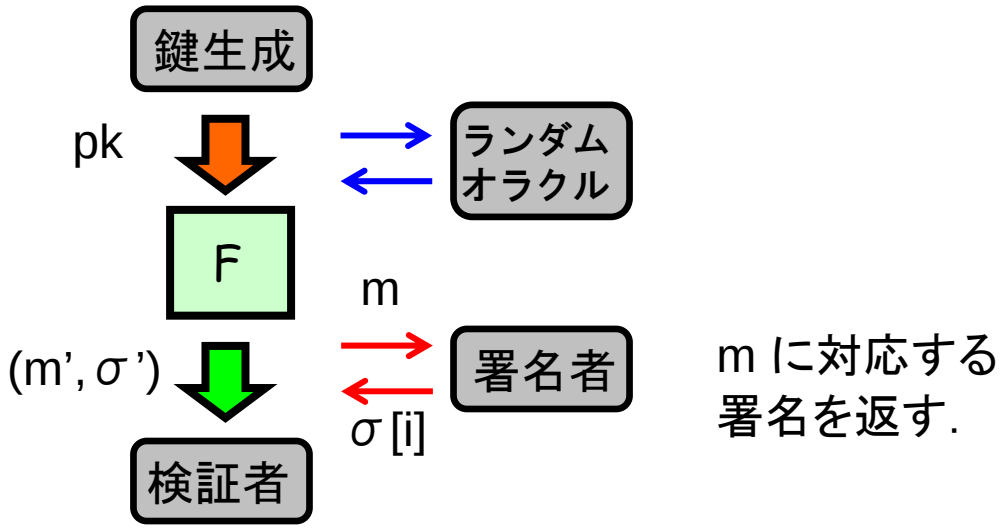
比較

EF-CMA安全の攻撃モデルを例にとって,
次の二点を比較してみる.

- フレームワークで保証している安全性の違い.
- フレームワークで取り扱うことのできる攻撃モデルの違い

EF-CMA安全のモデル

公開鍵と秘密鍵を生成し,
公開鍵を偽造者に与える.



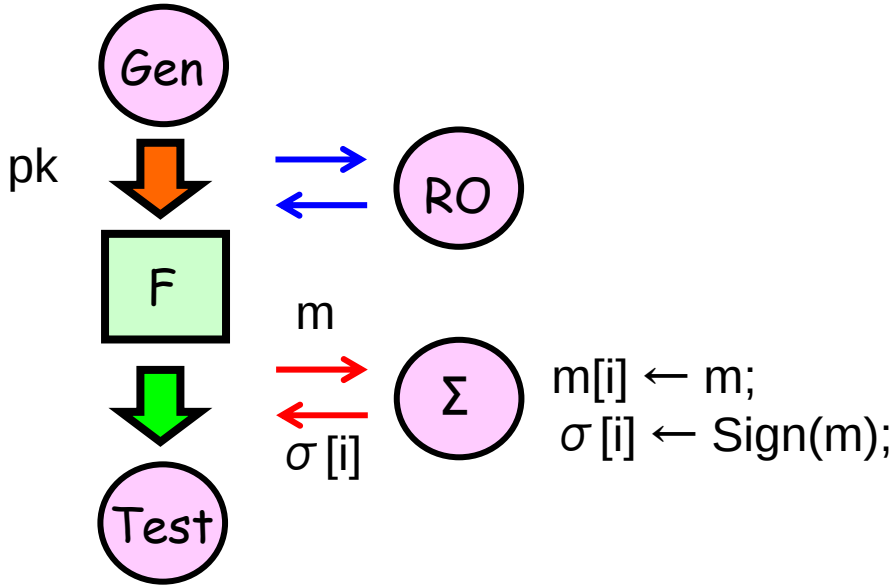
m に対応する
署名を返す.

偽造署名の条件を満たしているか調べる.

1. 検査式に合格する.
2. 文書m' に対応する署名を署名者に問い合わせていない.

Blanchetのフレームワークで証明していること

例：デジタル署名のEF-ACMA安全性

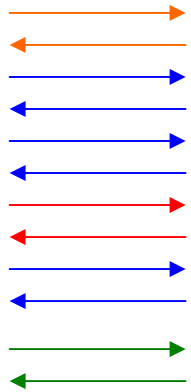


if defined(m[i], σ [i]) then 0;

・・・

ih≦qh RO | (Gen; is ≦ qs Σ | Test)

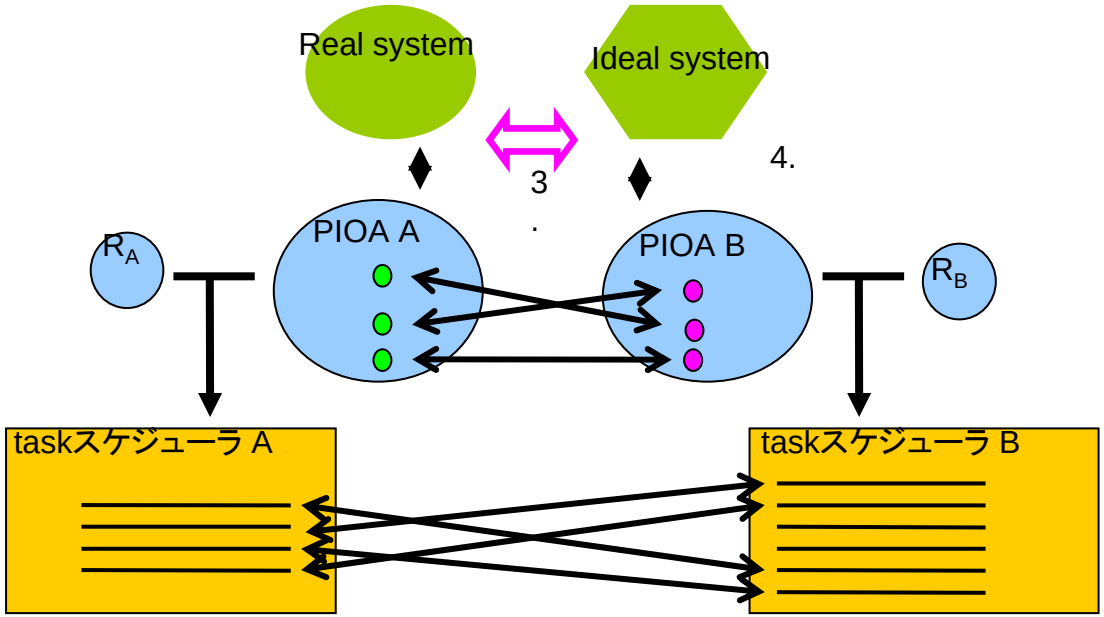
オラクルで攻撃をモデル化
→ あるオラクルの処理中に
別のオラクルが呼ばれる
動作は考慮していない.

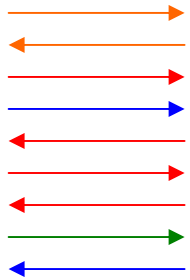


task-PIOAフレームワークで証明していること

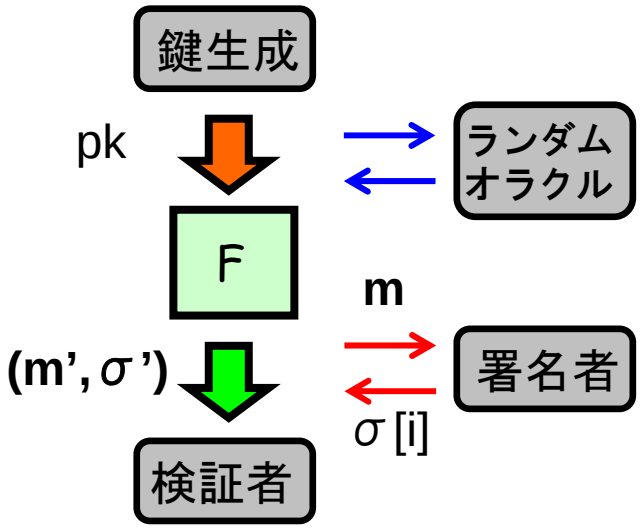
task-PIOA を用いて攻撃をモデル化 →

入力，内部，出力の3種類のActionを考慮.
実行される可能性のある全てのActionの組み合わせに対して対応がとれるか検証.





比較



m に対応する
署名を返す.

mを署名者に問い合わせて,
対応するσを受け取る前に,
(m, σ)を検証者に送る場合.

攻撃としては成功と見るべき.

偽造署名の条件を満たしているか調べる.

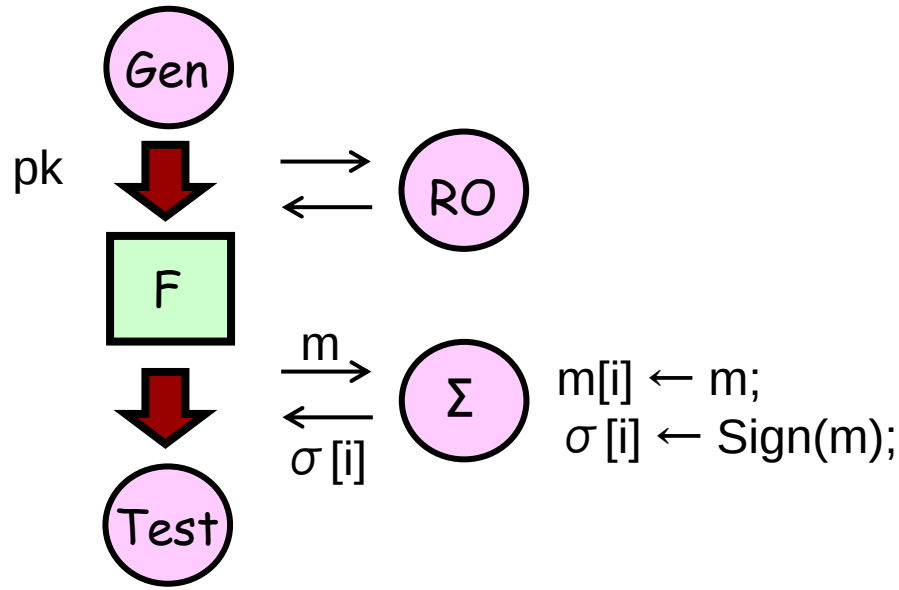
1. 検査式に合格する.
2. 文書m' に対応する署名を署名者に問い合わせていない.

Blanchetでは考慮していない攻撃者に対しても,
task-PIOAは評価している.

FDH署名のEF-ACMA安全性

Blanchetのフレームワーク

ih≦qh RO | (Gen; is ≦ qs Σ | Test)



証明可能なことが知られている. [BP06]

FDH署名の安全性証明

task-PIOA フレームワーク

攻撃モデルを表現するためには・・・

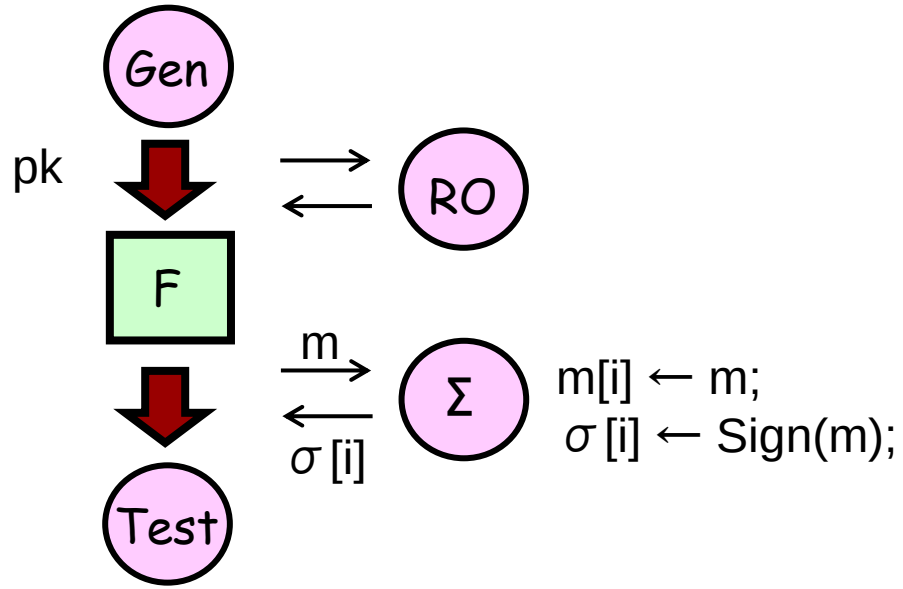
⇩

署名オラクルに問い合わせ済みの文書と署名の組をそれぞれ独立に扱う必要がある.

⇩

PIOAに文書と署名の数だけタスクと内部変数を用意し，独立に実行・記憶する.

⇩



タスクの数が有限に納まらない場合，全てのタスクスケジューラの検査は不可能.
よって，事前に問い合わせ回数の上限を数値で置かないとPIOAで記述できない.

まとめ

- **task-PIOAフレームワークとBlanchetフレームワークの証明能力の差を示した.**
  - task-PIOA: 証明の分解能はいいが，記述可能なプロトコルが限られる.
  - Blanchet : task-PIOAで記述できないものも取り扱えるが，
    証明の分解能はいまいち.

	モデル化	変形	評価	検査
task-PIOA	×	×	○	―
Blanchet	×	○	×	○

形式的に検証する部分

- **証明時の識別不可能性の評価の誤りを防止したい**
  - task-PIOA
- **証明のGameを作る手間を軽くしたい**
  - Blanchet

目的に応じて使い分けを


TOSHIBA
Leading Innovation