Computationally Sound Symbolic Analysis for EUC Security of Key Exchange

鍵交換のEUC安全性の計算論的に健全な記号的解析

鈴木 斎輝、吉田 真紀、藤原 融
大阪大学 情報科学研究科

発表の流れ

- 背景と従来研究
- 本発表の結果
- まとめと今後の課題

背景

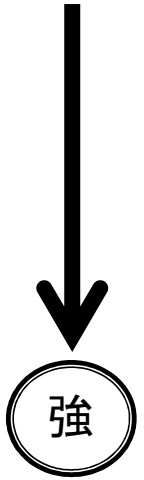
- ネットワークにおけるプロトコルの実行環境は動的
  - 例：Webサーバとの認証で利用されるTLS
  - プロセスは並列、逐次的に結合される
  - 必要に応じて、PKIの公開鍵など、情報を共有する
- 多くのプロトコルで動的な結合、情報共有は当然
- 動的な実行環境に応じた安全性を保証することが必須

人手による証明は困難なため、記号的解析が望まれる

動的な実行環境に応じた安全性

- 定式化されている安全性
  - 動的な結合のみ [Canetti01]
    - Universal Composability (Basic UC)
  - 動的な結合＋静的な情報共有 [Canetti-Rabin03]
    - UC with Joint state (JUC)：想定したプロセスとのみ情報共有
  - 動的な結合＋動的な情報共有 [Canetti et al. 07]
    - Generalized UC (GUC)：任意のプロセスと情報共有
    - Externalized UC (EUC)：GUCの等価な単純化
- 現時点で記号的に解析可能な安全性
  - Basic UC のみ
    - 受動的corruption[Canetti-Herzog04],[Patil05],[村谷-花谷06],[村谷-花谷07]から適応的 corruption [Canetti-Gajek10]へ強化

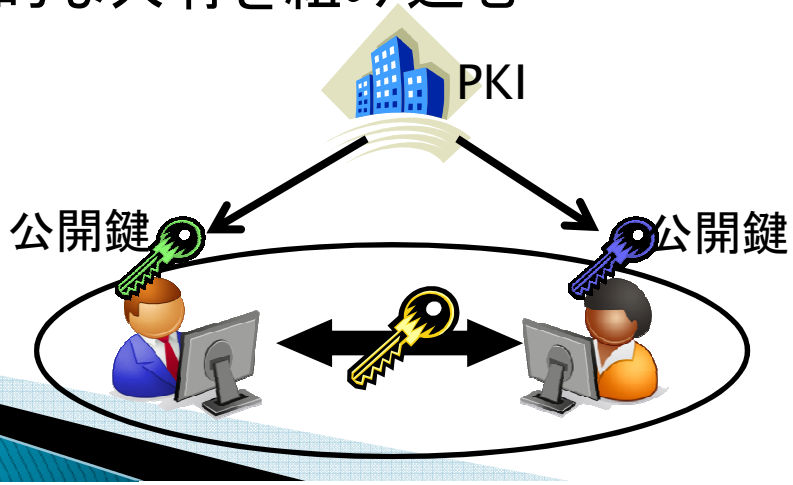
解析すべき安全性



Basic UC から EUC へ強化したい

本発表の目的と内容

- 目的
  - 適応的 corruptionに対するEUC安全性の記号的解析
- 内容
  - 検証対象プロトコルは基本的なものとする
    - 公開鍵暗号を使用する2者間鍵交換プロトコル
  - 動的に共有される情報はPKIの公開鍵
  - 従来の記号的解析[Canetti-Gajek10]を拡張し、公開鍵の動的な共有を組み込む



考えるべき課題

- 公開鍵の動的な共有を組み込む
  - 動的な共有を行うことで起こる実行状況を取りこぼさないようにしなければならない
- 公開鍵の動的な共有の定義では
  - 鍵生成関数はプロセスごとに異なる可能性がある
  - 関数は任意
- 従来の記号的な表現の方針
  - 関数を記号で表現し、関数の性質を推論規則で表現

従来の方針では、鍵生成関数の性質を表現できない

解決方針と本発表の意義

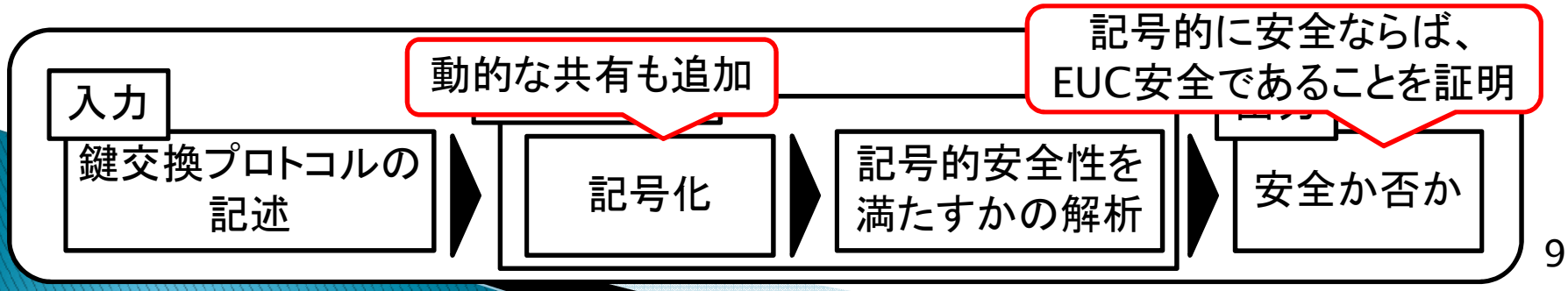
- 解決方針
  - 鍵生成関数の性質を表す推論規則を定めない
  - そのままでは、実行状況を取りこぼす
    - 取りこぼす実行状況は攻撃者が鍵生成関数の性質を使って、値を得ている実行状況
  - そこで、関数の引数を攻撃者に与える
    - この引数に依存した値であれば、鍵生成関数の性質を使うことなく得ることができる
- 本発表の意義
  - EUC 安全性の解析を可能にした
  - 関数が任意である場合の証明方針を立てた

以降の発表の流れ

- 従来の記号的解析とその拡張
- 対象とするプロトコルのEUC安全性
- 記号化
- 健全性の証明と完全性の考察
- まとめと今後の課題

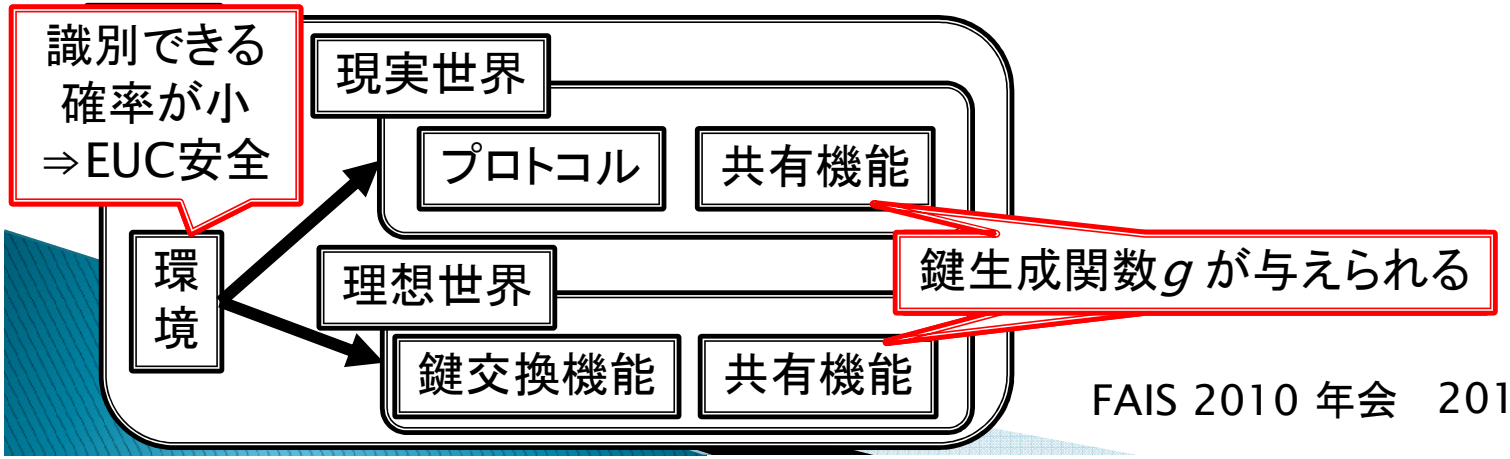
従来の記号的解析とその拡張

- 従来の記号的解析
  - 対象プロトコルの記号化と記号的な実行状況を定義
  - 記号的な実行状況に対して鍵交換の記号的安全性を定義
  - 解析の計算論的健全性を証明
    - 記号的安全性を満たすとき、UC安全であること
  - 記号的に表現したプロトコルが定義した安全性を満たすかを解析
- 本発表での拡張
  - 公開鍵の動的な共有の記号化も追加
  - 記号的な実行状況を動的な共有も含むものへ拡張
  - 記号的安全性を満たすとき、EUC安全であることを証明



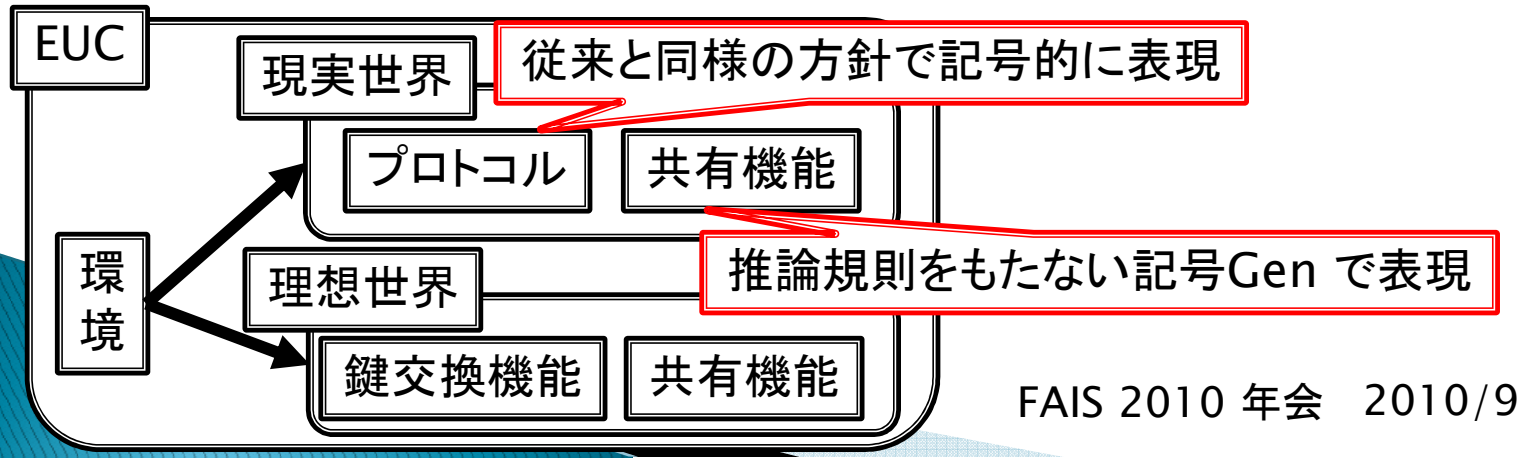
対象とするプロトコルのEUC安全性

- 現実世界：攻撃者の存在下で対象プロトコルが動作
- 理想世界：攻撃者のシミュレータの存在下で鍵交換の理想的な機能が動作
- 両世界に公開鍵を共有するための機能が存在
  - プロトコルを動作させるパーティがアクセスし、公開鍵を登録し、入手
- EUC安全性
  - ∀攻撃者, ∃シミュレータ, ∀環境, 現実世界と理想世界の実行状況を環境が識別できる確率が十分に小さい
  - 実行の際に、公開鍵共有の機能へ登録時に用いる鍵生成関数gを環境が与える



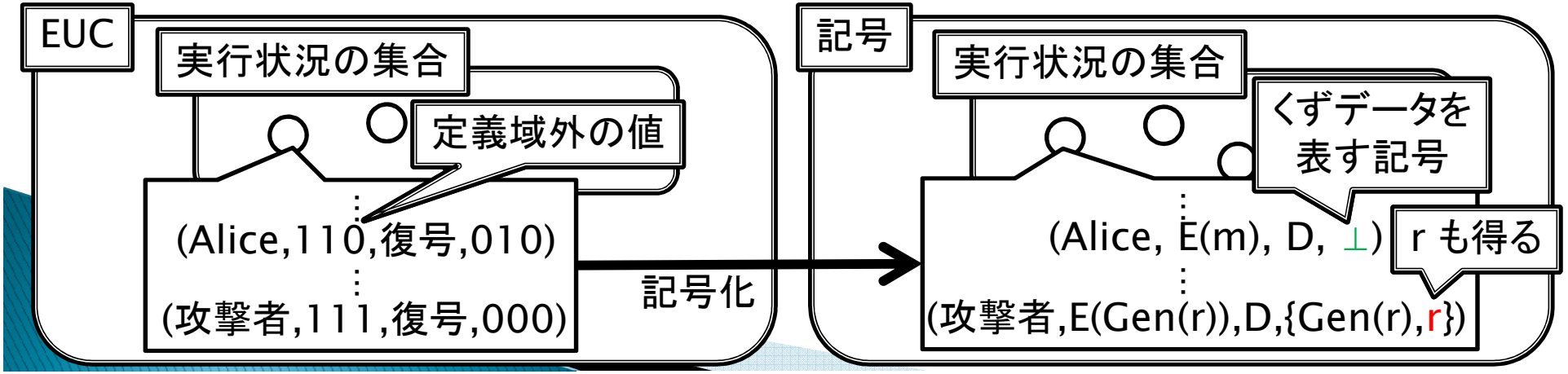
プロトコルと共有機能の記号化

- プロトコルは従来と同様の方針で記号的に表現
  - 関数を記号で、関数の性質を推論規則で表現
- 共有機能の記号的な表現
  - 鍵生成関数gを記号化した記号 Gen を導入
    - Gen には推論規則を定めない



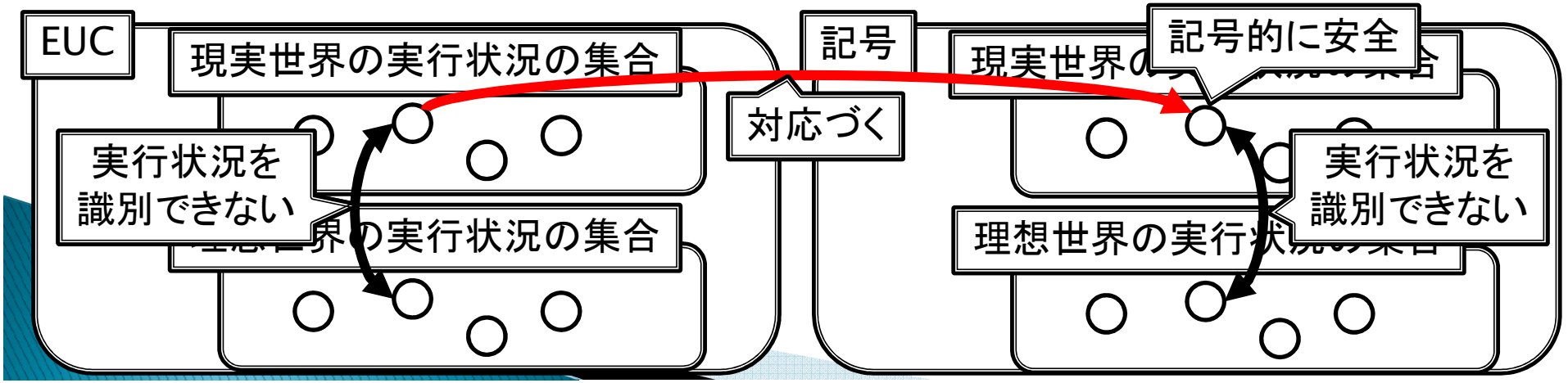
実行状況の定義

- 実行状況の定義
  - 誰がどの値にどの関数を使いどのような値を得たかを表す系列
- 記号的な実行状況の定義
  - 従来と同様の方針で定義
    - 値を記号化し、関数を対応する記号に置き換える
    - 関数の引数が定義域に含まれない場合、得られる値はくずデータを表す記号に置き換える
  - 実行状況を取りこぼさないように、g の引数を記号的攻撃者に与える
    - 記号的攻撃者が Gen(r) を得るとき r も得る



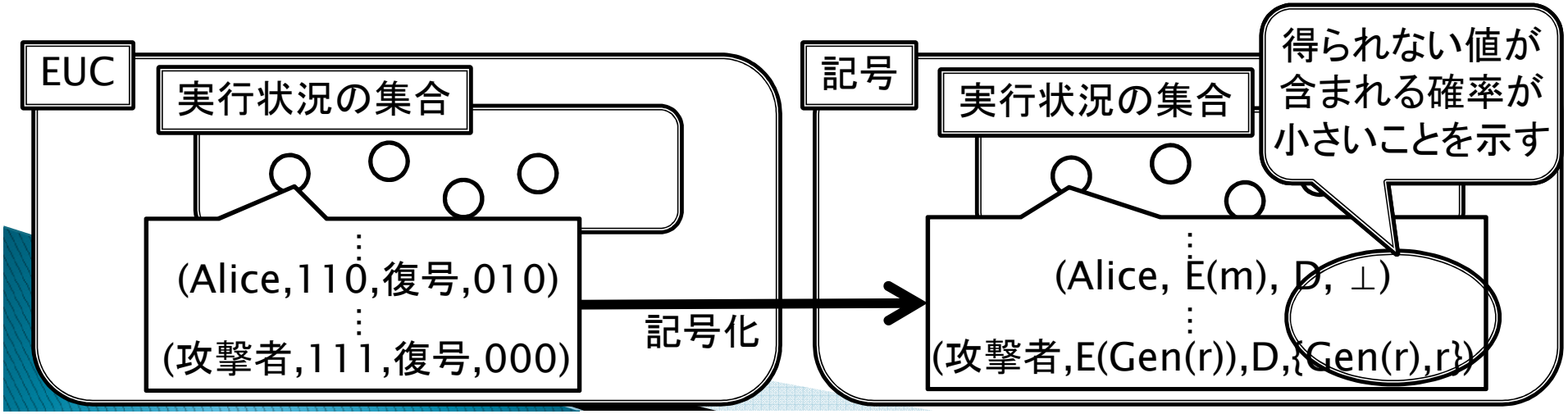
健全性証明の概要

- 健全性
  - 記号的に安全であるとき、対象プロトコルはEUC安全
- 証明の概要
  - 現実世界の実行状況を記号化したとき、記号的な実行状況に十分高い確率で対応づくことが示せる
  - 記号的に安全であるとき、記号的な現実世界と理想世界の実行状況が識別できないことが示せる
  - 実行状況が十分高い確率で対応づくため、EUCでの現実世界と理想世界の実行状況が識別できないことが示せる



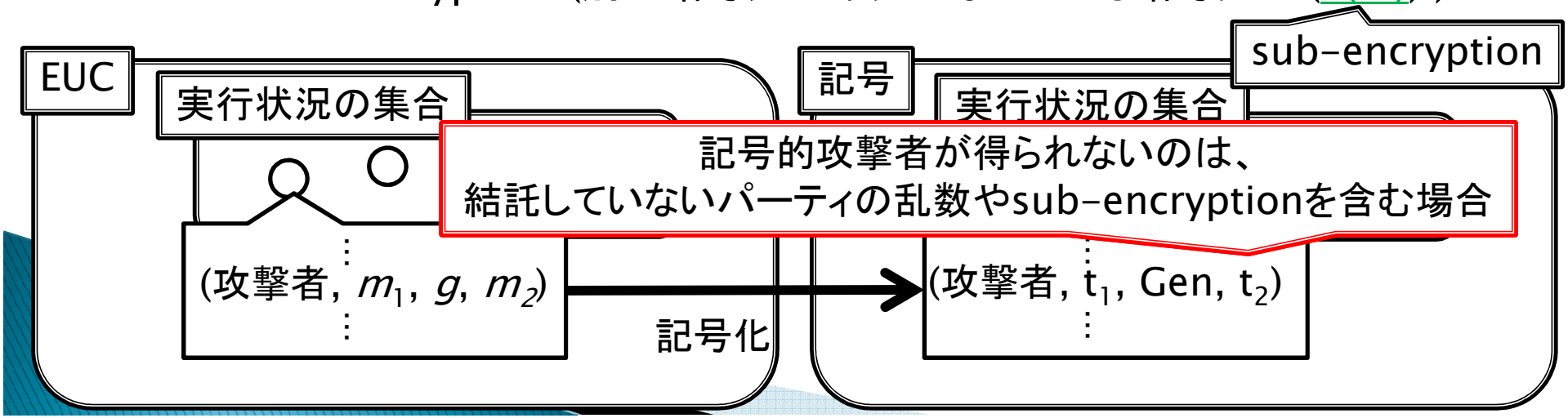
Mapping 補題の証明(1/3)

- Mapping 補題
  - 現実世界の実行状況を記号化したとき、記号的な実行状況に十分高い確率で対応づく
- 証明の本質
  - EUCの攻撃者が得た値の中に、記号的攻撃者が得られない値が含まれている確率が十分に小さいことを示す
- g の性質を使って得た値について考える



Mapping 補題の証明(2/3)

- g の性質を使って得た値が引数に依存する値のとき、記号的攻撃者もその値を得られる
  - g の引数を記号的攻撃者に与えるため
- g の性質を使って得た値が引数に依存しない値のとき、記号的攻撃者がその値を得られない場合がある
  - 以下の値を含む場合
    - 攻撃者と結託していないパーティの乱数
    - sub-encryption (別の暗号文の平文になっている暗号文: E(E(m)))



Mapping 補題の証明(3/3)

- どちらの値を含むものでもEUCの攻撃者が得る確率は十分に小さい
  - 乱数:一様分布から選ばれるため
  - sub-encryption:公開鍵暗号の安全性に矛盾するため
    - sub-encryption を得るためには、元の暗号文を復号するか適当な値を暗号化したものを一致させることになる
    - どちらもできる確率が十分小さくなければ、暗号文を識別可能になり、公開鍵暗号の安全性に矛盾する

完全性の考察

- 完全性
  - 記号的に安全でないとき、対象プロトコルはEUC安全でない
    ⇔対象プロトコルに攻撃がないのに、攻撃が出力される
- 拡張した記号的解析は完全性をもたない
  - 記号的攻撃者を鍵生成関数の引数を得られるように強化したため
- 攻撃者を強化したが、EUCであり得る攻撃者
  - この強化は鍵生成関数が一方向性をもたないことを意味する
  - 鍵生成関数が任意なので、一方向性をもたないこともあり得る
- 出力された攻撃がEUCの攻撃かの判定法を考える

まとめと今後の課題

- 適応的 corruptionに対するEUC安全性の記号的解析を提案
  - 対象プロトコルは公開鍵暗号を使用する鍵交換プロトコル
  - 公開鍵の動的な情報共有を記号化
  - 解析の計算論的健全性を証明

- 今後の課題
  - 記号的解析を拡張し、解析可能なプロトコルクラスを広げる
  - 公開鍵以外の情報の動的な共有を記号的解析に組み込む
  - 既存ツールを用いて、実プロトコルの解析を行う
  - 完全性をもつように記号化を改良