チュートリアル
結合可能安全性の形式検証における最近の研究動向

吉田 真紀
独立行政法人 情報通信研究機構（NICT）
ネットワークセキュリティ研究所

概要

- 紹介内容
  - Dahl と Damgård による Eurocrypt 2014 の成果 [DD14]
    - Universally Composable Symbolic Analysis (UCSA) の拡張
- DD14の成果
  - 検証対象の拡張

	検証対象(2者間プロトコル)			
	機能	暗号技術	攻撃者	証明例
従来	鍵交換、 相互認証、 認証鍵交換	電子署名、 DH鍵交換、 公開鍵暗号	静的	教科書的 プロトコル
[DD14]	任意	認証付き通信路、 準同形暗号、 コミットメント、 ゼロ知識証明	静的だが 一方の結託 有り	紛失通信 プロトコル [DNO08]

概要

- 紹介内容
  - Dahl と Damgård による Eurocrypt 2014 の成果 [DD14]
    - Universally Composable Symbolic Analysis (UCSA) の拡張
- DD14の成果
  - 検証対象の拡張

	検証対象(2者間プロトコル)			
	機能		攻撃者	
従来	鍵交換、 相互認証、 認証鍵交換		静的	
[DD14]	任意	認証付き通信路 準同形暗号、 コミットメント、 ゼロ知識証明	静的だが 一方の結託 有り	紛失通信 プロトコル [DNO08]

高機能な
暗号技術

第一線の
プロトコル

[DD14]の成果の見方：実用的観点 ☹

- 実プロトコルの評価には不十分 or オーバスペック
  - ハッシュ関数を使ったプロトコルは対象外
  - ゼロ知識証明を使った実プロトコルは現時点で無し
  - 紛失通信プロトコルも使われる予定無し

⇒ 課題は数多くある

	検証対象（2者間プロトコル）			
	機能		攻撃者	
従来	鍵交換、相互認証、認証鍵交換		静的	
[DD14]	任意	認証付き通信路、準同形暗号、コミットメント、ゼロ知識証明	静的だが一方の結託有り	紛失通信プロトコル[DNO08]

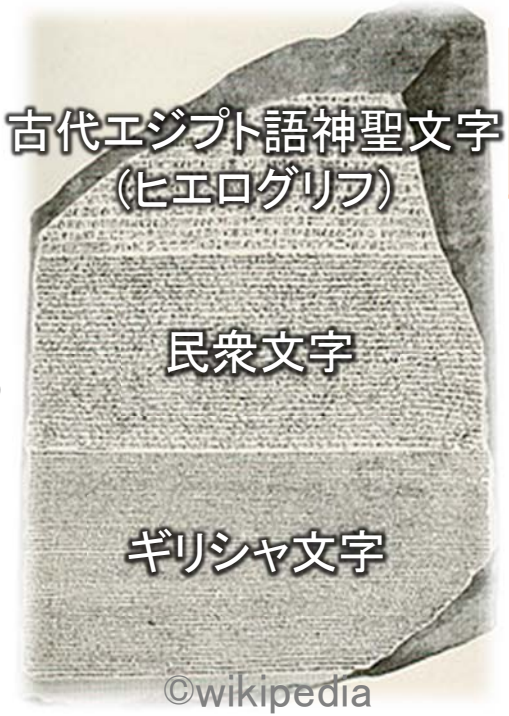
高機能な暗号技術

第一線のプロトコル

[DD14]の成果の見方：学術的観点 ☺

- ロゼッタ・ストーン？
  - 計算論的モデル（汎用結合可能安全性UCの枠組み）と記号論的モデル（π計算）をつなぐノウハウの固まり

プトレマイオス5世施政下の
宗教会議の布告



©wikipedia

細かい違いはあれど、
同一の文章が全部で三つの
書記法で著されている

古代エジプト語の文書が続々と翻訳

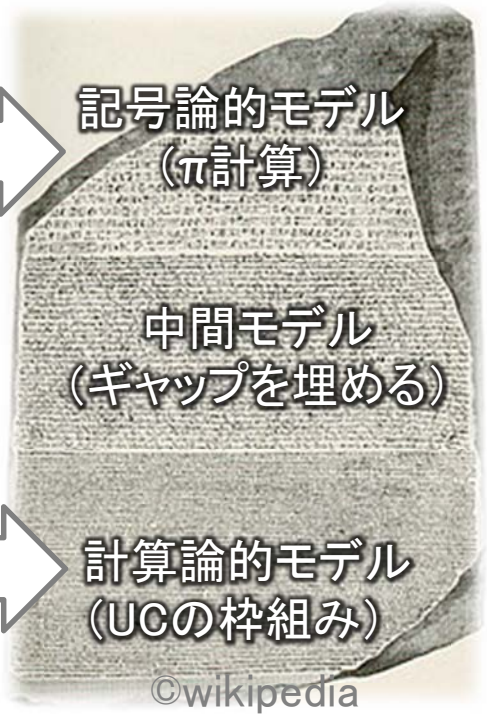
[DD14]の成果の見方：学術的観点 ☺

- ロゼッタ・ストーン？
  - 計算論的モデル（汎用結合可能安全性UCの枠組み）と記号論的モデル（π計算）をつなぐノウハウの固まり

記号論的安全性
実プロセスと
理想プロセスの
観測等価性

計算論的健全性

計算論的安全性
実プロトコルと
理想プロトコルの
識別不能性



©wikipedia

細かい違いはあれど、
同一システムが全部で三つの
モデルで著されている

- 検証の枠組みの整備
  - UCの定式化に近い形で検証対象を定義
- 検証ツールの有効活用
  - ProVerifの「観測等価性の検証」機能を新たに利用

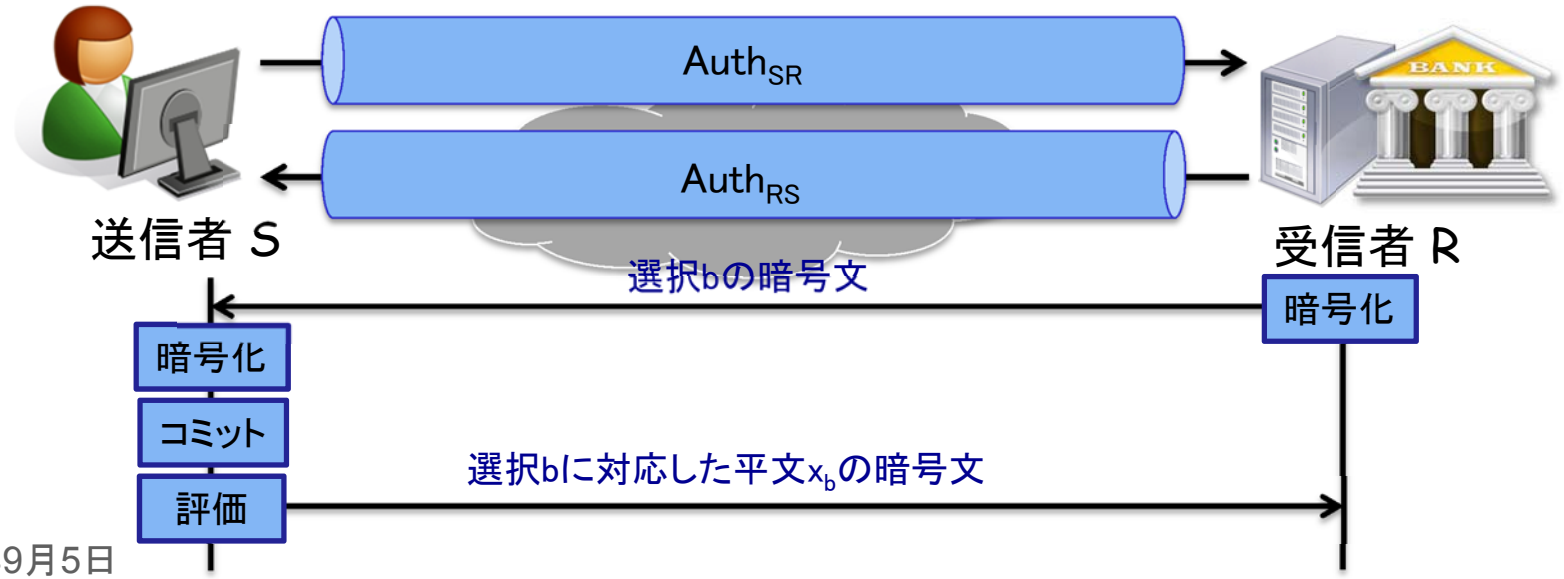
様々な拡張の可能性

以降の発表の流れ

- [DD14]のUCSAの紹介
  - 検証対象
  - 検証の枠組み
  - 変換
  - 計算量的健全性の保証
- 今後の展望

検証対象：2者間プロトコル（1/2）

- 送受信者のプログラムからなる
- 通信に認証付き通信路を使用
- 暗号的操作は2種類
  - コミットメントによるコミット・開示
  - 準同型暗号による暗号化・評価・復号



検証対象：2者間プロトコル（2/2）

- 「計」「記」モデルをつなげるテクニック
  - コミット値と暗号文にはゼロ知識証明をつけてパッケージ化
    - コミット値と暗号文をどのように計算したか知っていることの証明
    - コミットした値が暗号化された値と指定された依存関係をもつことの証明
  - パッケージ化されていなければゴミデータ扱い（停止）

⇒値の由来（既知の値から指定された形で計算）を保証

⇒記号論的モデル（項による表現）との対応付けに役立つ

$$
\begin{array}{rl}
\textit{commitment package:} & [\mathtt{comPack} : D, ck, \pi_U, crs] \\
\textit{encryption package:} & [\mathtt{encPack} : C, ek, \pi_T, crs] \\
\textit{evaluation package:} & [\mathtt{evalPack} : C, C_1, C_2, ek, D_1, D_2, ck, \pi_e, crs]
\end{array}
$$

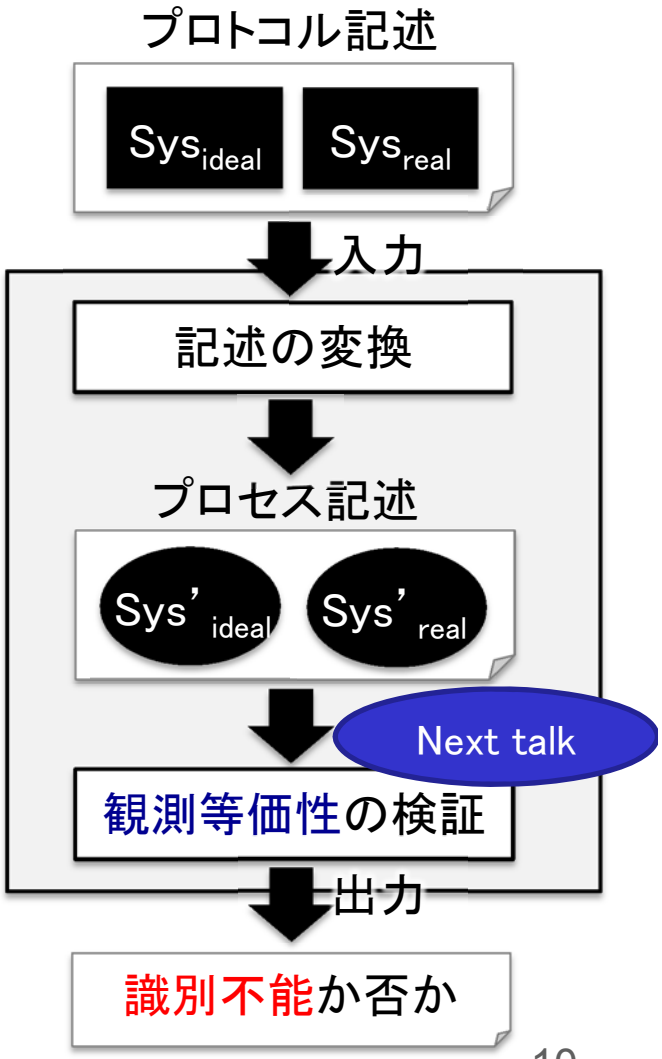
計算論的モデル
（p.8）

$$
\begin{array}{r}
\mathbf{comOf}(\mathbf{comPack}(x_d, x_{ck}, x_\pi, x_{crs})) \rightsquigarrow x_d \\
\mathbf{ckOf}(\mathbf{comPack}(x_d, x_{ck}, x_\pi, x_{crs})) \rightsquigarrow x_{ck} \\
\mathbf{proofOf}(\mathbf{comPack}(x_d, x_{ck}, x_\pi, x_{crs})) \rightsquigarrow x_\pi \\
\mathbf{crsOf}(\mathbf{comPack}(x_d, x_{ck}, x_\pi, x_{crs})) \rightsquigarrow x_{crs}
\end{array}
$$

記号論的モデル
（項書換規則 Fig.89）

検証の枠組み

- 2者間プロトコルの記述言語を定義
- 検証法の入力
  - 理想世界のプロトコル記述 Sys_{ideal}
  - 現実世界のプロトコル記述 Sys_{real}
- 検証法の処理
  - 入力を記号論的モデル(π計算)の記述(プロセス記述)に変換
  - 記号論的モデルでの等価性(観測等価性)検証
- 検証法の出力
  - 入力の計算論的モデル(UCの枠組み)での等価性(識別不能性)
- 計算論的健全性の保証
  - 観測等価ならば識別不能



記述の変換例

- [DNO08]の紛失通信プロトコルの送信者側

プロトコル記述 Sys_{real} (Fig.3)

$$
\begin{aligned}
P^S_{OT} \doteq\ & \mathrm{input}_{\emptyset}[receive_{RS} : c_b];\\
& \quad \text{if } \mathsf{verEncPack}_{bit,R,R}(c_b) \text{ then}\\
& \quad \mathrm{input}^S_{\emptyset}[in^S_{OT} : x_{01}];\\
& \quad \text{if } \mathsf{isPair}(x_{01}) \text{ then}\\
& \quad \text{let } x_0 \leftarrow \mathsf{first}(x_{01});\\
& \quad \text{let } x_1 \leftarrow \mathsf{second}(x_{01});\\
& \quad \text{if } \mathsf{isValue}(x_0) \text{ then}\\
& \quad \text{if } \mathsf{isValue}(x_1) \text{ then}\\
& \quad \text{let } c_x \leftarrow \mathsf{eval}_{sel,R,S,S}(c_x, x_0, r_0, x_1, r_1);\\
& \quad \mathrm{output}[send_{SR} : c_x];\\
& \quad \mathrm{stop}
\end{aligned}
$$

値の上での等価性確認を表す

値の計算を表す

プロセス記述 Sys'_{real} (Fig.97)

$$
\begin{aligned}
Q_S \doteq\ & \text{new } r, r_0, r_1;\\
& \quad \mathrm{in}[receive_{RS}, c_b];\\
& \quad \text{if } \mathbf{verEncPack}_{bit}(c_b) = \mathbf{true} \text{ then}\\
& \quad \text{if } \mathbf{ekOf}(c_b) = ek_R \text{ then}\\
& \quad \text{if } \mathbf{crsOf}(c_b) = crs_R \text{ then}\\
& \quad \mathrm{in}^S[in^S_{OT} : c_b];\\
& \quad \text{if } \mathbf{isPair}(x_{01}) = \mathbf{true} \text{ then}\\
& \quad \text{let } x_0 = \mathbf{first}(x_{01});\\
& \quad \text{let } x_1 = \mathbf{second}(x_{01});\\
& \quad \text{if } \mathbf{isValue}(x_0) = \mathbf{true} \text{ then}\\
& \quad \text{if } \mathbf{isValue}(x_1) = \mathbf{true} \text{ then}\\
& \quad \text{let } c'_b = \mathbf{encOf}(c_b);\\
& \quad \text{let } c'_x = \mathbf{eval}_{sel}(c'_b, x_0, x_1, r);\\
& \quad \text{let } d_0 = \mathbf{com}(x_0, r_0, ck_S);\\
& \quad \text{let } d_1 = \mathbf{com}(x_1, r_1, ck_S);\\
& \quad \text{let } p_x = \mathbf{proof}_{sel}(c'_x, c'_b, ek_R, d_0, d_1, ck_S, crs_S);\\
& \quad \text{let } c_x = \mathbf{evalPack}(c'_x, c'_b, \ldots, ck_S, p_x, crs_S);\\
& \quad \mathrm{out}[send_{SR}, c_x]
\end{aligned}
$$

項の上での等価性確認を表す
（項書換規則の利用）

項の構成を表す

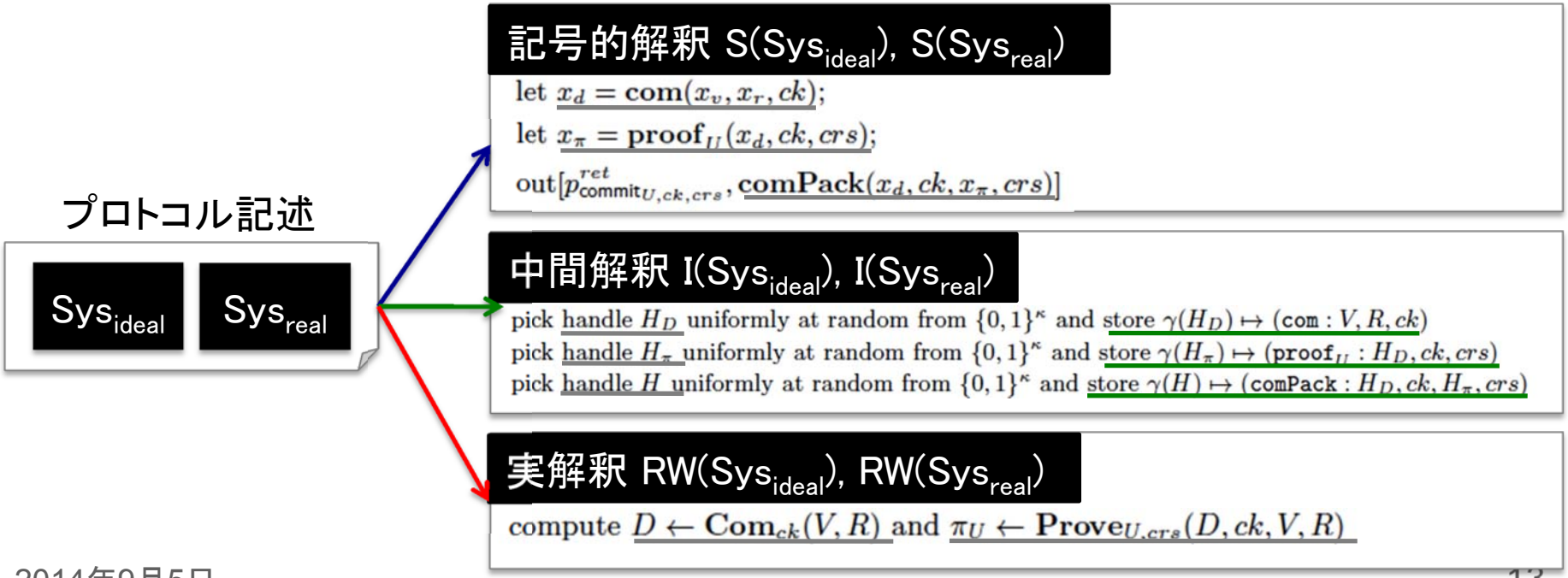
計算論的モデルと
記号論的モデルで
解釈が大きく違う

計算論的健全性の保証(1/3)

- プロトコル記述 Sys に対して中間解釈を定義
  - 記号的解釈 S (Sys): π 計算での解釈、項を扱う
  - 中間解釈 I (Sys): UCの枠組みでの解釈だが、値を直接扱わない
  - 実解釈 RW(Sys): UCの枠組みでの解釈、値を直接扱う

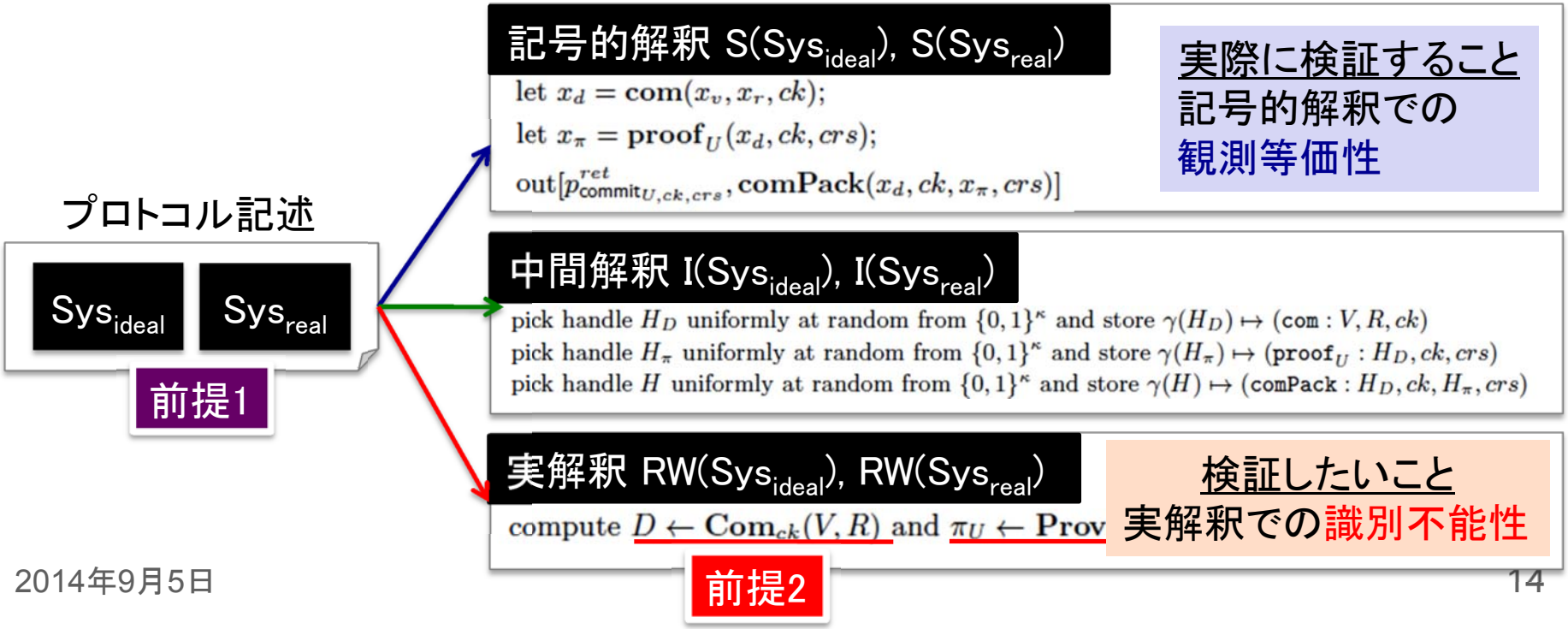
計算論的健全性の保証(2/3)

- 「計」「記」モデルをつなげるテクニック
  - 中間解釈では、値の代わりに「ハンドル名」を扱う
    - ハンドル名はビット列から一様ランダムに選択
  - 値は計算されることなく、グローバルメモリに保管



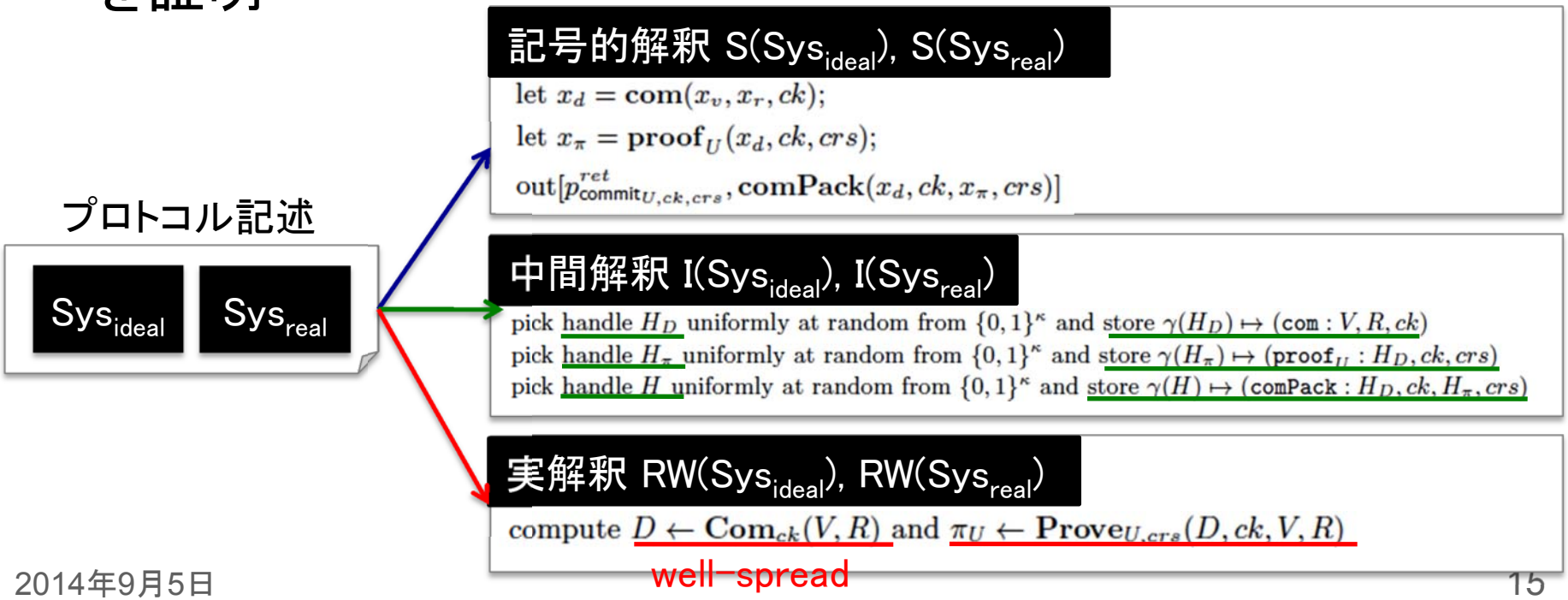
計算論的健全性の保証(3/3)

- Theorem 6.3: $S(Sys_{ideal}) \approx S(Sys_{real}) \Rightarrow I(Sys_{ideal}) \approx I(Sys_{real})$
  - 前提1:メッセージの消失がない(プロトコルとして送受信の対応有り)
- Corollary 5.7: $I(Sys_{ideal}) \approx I(Sys_{real}) \Rightarrow RW(Sys_{ideal}) \approx RW(Sys_{real})$
  - 前提2:使用している暗号技術が安全



計算論的健全性の証明

- 解釈間の各違いについて、それが原因で、
  実/理想プロトコルが識別可能とならないことを証明
- 具体的には、
  「解釈の違いから識別可能⇒暗号技術への攻撃有り」
  を証明



今後の展望

- UCSAそのもの関する様々な研究開発
  - [DD14]のUCSAの適用例を増やし、その有効性を確認
    - 例：[DNO08]のプロトコルを安全でないように改変して適用し、
      安全でないと判定されるか確認
  - 従来の検証対象を[DD14]の枠組みで焼き直す
    - 例：鍵交換と相互認証の理想機能とシミュレータを記述や、
      使用する暗号技術をダウングレード（準同形暗号→公開鍵暗号）
  - [DD14]のUCSAを拡張し、実プロトコルへの適用を目指す
    - 例：ハッシュ関数を使ったプロトコルを検証可能とする
- UCSAの基盤となる検証ツールの研究開発
  - UCSAに特化した数理的手法
- 現状で手作業の部分を自動化
  - 参考：次のチュートリアル「ProVerifによる結合可能安全性の形式検証」

…等々多数