

# FCS-ARSPA-WITS,CSF,FCC'08参加報告 自動検証と形式的証明について

Reynald Affeldt  
産業技術総合研究所 情報セキュリティ研究センター

川本 裕輔  
東京大学大学院情報理工学系研究科

JSIAM-FAIS  
平成20年9月18日

# FCS-ARSPA-WITS,CSF,FCC'08参加報告

## 自動検証と形式的証明について

### 講演予稿集の論文のまとめ

#### ● 自動検証

- より高度な安全性
  - *Type-checking Zero-knowledge*
  - 電子投票の安全性: coercion-resistanceなど.  
(*Automated Verification of Remote Electronic Voting Protocols in the Applied Pi-Calculus*)
  - 誤検知のないプロトコル検証に向けて (*A Note on First-order Logic*)
  - *Statically Detecting Message Confusions in a Multi-protocol Setting*
- 実際のシステム
  - プロトコルの実装 (*Refinement Types for Secure Implementations*)
  - 産業標準 (*Formal Analysis of PKCS#11*)

#### ● 形式的証明

- 計算論的モデルの形式化
  - ゲームベースの安全性証明  
(*Formal Certification of Code-based Cryptographic Proofs, A Formal language for Cryptographic Pseudocode*)
- 記号モデルに対するresult certification
  - 形式的証明の自動検証  
(*Towards Producing Formally Checkable Security Proofs, Automatically; Certificates for Tree Automata Completion*)

# 概要

## Result certificationについて

- *Towards Producing Formally Checkable Security Proofs, Automatically*  
Goubault-Larrecq

- Applied-pi計算の応用

- *Composition of Password-based Protocols*  
Delaune, Kremer, Ryan

- 具体的なプロトコルの解析

- *Formal Analysis of PKCS#11*  
Delaune, Kremer, Steel

# 一階論理を用いたプロトコル検証

- 定式化:
  - 暗号プロトコル = 節集合
    - $I([x]_y) \wedge I(y) \rightarrow I(x)$  (復号) のような論理式
  - 検証の目標 = 導出可能でない事実
    - 例:  $\neg I(\text{Secret})$
- 検証:
  - 導出可能な事実の全数探索
    - $\perp$  が導出可能であれば, 潜在的な攻撃が存在する
    - $\perp$  が導出可能でなければ, プロトコルが安全である
      - 集合  $\{I([x]_y) \wedge I(y) \rightarrow I(x), \dots, \neg I(\text{Secret})\}$  が整合的

# プロトコル検証の形式的証明

- 問題点:
  - 検証ツールはyes/noを出力する
  - どうやってバグがないことを保証するのか？
- 解決法:
  - 定理証明支援系を用いた形式的証明
    - Naïve な手法:
      - 検証ツールのバグがないことを機械的に証明する
      - ⇒ 現実的でない
    - Result certification:
      - 検査可能な安全性証明
      - ⇒ どのようなものを与えればよいのか？
      - ⇒ どのように既存のツールに基づいて形式的証明すればよいのか？

## 整合性とモデル

- 論理式集合が“モデル”を持つとき, 整合的であるという
    - モデルとは
      - 領域 (= 何らかの要素の集合, 無限集合でも良い)
      - 関数記号と述語記号の解釈
    - 例:
      - 論理式  $P(a) \wedge \neg P(b)$  がモデル  $\mathcal{M}$  を持つ
        - 記法:  $\mathcal{M} \models P(a) \wedge \neg P(b)$
        - モデル  $\mathcal{M}$  の例:  $\mathcal{M}$  の領域を集合  $\{0,1\}$  とし,  $a$  を  $0$ ,  $b$  を  $1$ ,  $P$  を  $\{0\}$  と解釈するモデル
  - 検証ツールが見つかるモデルは役立たないかも
    - $\text{lfp } T_{S_\infty} \models S$  が成り立つかどうかは決定不能
- ⇒ result certificationにとって適切でないモデルもある

# 有限モデルを用いたResult Certification

- そもそも
  - ほとんどの安全なプロトコルは有限モデルを持つ
- そこで,
  - プロトコルモデル  $S$  が与えられたときに,
  - 有限モデル  $M$  を生成し,
    - Paradox や h1 のような既存のツールを用いる
  - 定理証明支援系で  $M \models S$  を検査する方法を与える

# Coqで $\mathcal{M} \models S$ を検査する方法は?

- 小さな有限モデル (要素数が5以下) に対して
  - 領域をCoqにおける帰納的型とする
  - 関数記号と述語記号の解釈をCoqにおける関数とする
  - $S$  の各節  $C$  に対して, 変数に対する要素の割り当てをすべて検査する
- より大きな有限モデル (h1 で見つけられるようなもの) に対して
  - h1 が *alternating tree automaton*  $S_{prod}$  を返す
  - $\text{Det}(S_{prod}) \models S$  が成り立つかどうか確かめる
    - $\text{Det}(S_{prod})$  が決定性ツリーオートマトンである
      - 有限モデルの表現方法のひとつ
    - $\text{Det}(S_{prod}) \models S$  が  $\text{Ifp } T_{S_{prod}} \models S$  と同値である
  - $S$  の各節  $C$  に対して,  $\vdash C$  が成り立つことを確かめる
    - $\vdash$  とは, (Coq で実装された) モデル検査器
    - $S$  の任意の節  $C$  に対して  $\vdash C$  であるとき,  $\text{Det}(S_{prod}) \models S$  (Coqで証明)



# 概要

- Result certificationについて
  - *Towards Producing Formally Checkable Security Proofs, Automatically*  
Goubault-Larrecq
- ➔ Applied- $\pi$ 計算の応用
  - *Composition of Password-based Protocols*  
Delaune, Kremer, Ryan
- 具体的なプロトコルの解析
  - *Formal Analysis of PKCS#11*  
Delaune, Kremer, Steel

# 合成プロトコルの安全性

- 一般に
  - プロトコル  $P_1$  と  $P_2$  が秘密  $w$  を共有するとき,  $vW.P_1$  と  $vW.P_2$  が安全ならば,  $vW.(P_1|P_2)$  は安全か?
- この論文では
  - 共有される秘密 = ユーザが選んだパスワード
  - 安全性 = guessing attacks への耐性

## Guessing Attackの例: Handshakeプロトコル

- $A$  と  $B$  が 鍵  $w$  を共有し,  $A$  が  $B$  を認証しようとしている

$$A \text{ — senc}(n, w) \rightarrow B$$

$$A \leftarrow \text{senc}(f(n), w) \text{ — } B$$

- Guessing attack:

攻撃者は次のメッセージを観測

$$x_1 = \text{senc}(n, w) \qquad x_2 = \text{senc}(f(n), w)$$

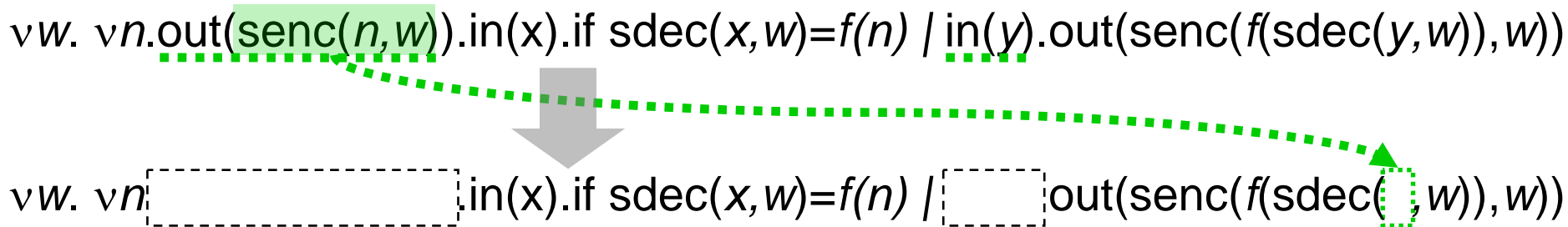
受動的に  $x$  を推測しようとする

$$f(\text{sdec}(x_1, x)) =? \text{sdec}(x_2, x)$$



# Applied pi計算におけるフレーム

- フレーム = 攻撃者によって観測されるメッセージ  
– プロトコル実行に現れるメッセージ
- 例:



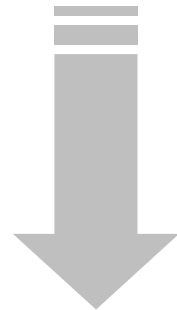
# Applied pi計算におけるフレーム

- フレーム = 攻撃者によって観測されるメッセージ  
– プロトコル実行に現れるメッセージ
- 例:

$\nu w. \nu n. \text{out}(\text{senc}(n, w)). \text{in}(x). \text{if } \text{sdec}(x, w) = f(n) \mid \text{in}(y). \text{out}(\text{senc}(f(\text{sdec}(y, w))), w)$



$\nu w. \nu n. \text{in}(x). \text{if } \text{sdec}(x, w) = f(n) \mid \text{out}(\text{senc}(f(\text{sdec}(\text{senc}(n, w), w))), w)$



$\mid \{\text{senc}(n, w) \mid x_1\}$

Activeな代入

フレーム

$\nu w. \nu n. \{\text{senc}(n, w) \mid x_1\} \mid \{\text{senc}(f(n), w) \mid x_2\}$

# Guessing Attacks への耐性

- 次のフレームを考える ( $w'$  は fresh な名前)

$$\triangleright \phi = \nu w. \nu n. \{ \text{senc}(n, w) / x_1 \} \mid \text{senc}(f(n), w) / x_2 \mid \{ w / x \}$$

$$\triangleright \phi' = \nu w'. \nu w. \nu n. \{ \text{senc}(n, w) / x_1 \} \mid \text{senc}(f(n), w) / x_2 \mid \{ w' / x \}$$

$f(\text{sdec}(x_1, x)) =? \text{sdec}(x_2, x)$  が成り立つ,  $\phi$  と  $\phi'$  が識別できてしまう

$\Rightarrow \phi$  と  $\phi'$  は 静的同値でない

$\Rightarrow$  Handshake プロトコルは guessing attacks に対して安全でない

- 一般的な定義:

$\phi = \nu w. \phi'$  が  $w$  への guessing attack に対して安全であるとは

$$\nu w. (\phi' \mid \{ w / x \}) \approx \nu w'. \nu w. (\phi' \mid \{ w' / x \})$$

$w'$  は fresh な名前,  $x$  は "fresh" な変数

# 合成について

- well-tagged プロセス
  - プロセスが  $w$  に関して  $\alpha$ -well-tagged であるとは、 $w$  の全ての出現が  $\text{hash}(\alpha, w)$  の形をしていることとする。
  - $v w.P$  から  $v w.P\{\text{hash}(\alpha, w)/w\}$  への変換は、 $w$  への guessing attack に対する安全性を保存する。
- 定理:
  - $P_1$  と  $P_2$  が  $w$  に関して  $\alpha$ -well-tagged かつ  $\beta$ -well-tagged ( $\alpha \neq \beta$ ) とする。  
 $v w.P_1$  と  $v w.P_2$  が  $w$  への guessing attack に対して安全であれば、 $v w.(P_1 | P_2)$  も  $w$  への guessing attack に対して安全である。



# 概要

- Result certificationについて
  - *Towards Producing Formally Checkable Security Proofs, Automatically*  
Goubault-Larrecq
- Applied pi計算の応用
  - *Composition of Password-based Protocols*  
Delaune, Kremer, Ryan
- ➡ 具体的なプロトコルの解析
  - *Formal Analysis of PKCS#11*  
Delaune, Kremer, Steel

# 動機

- PKCS#11
  - アプリケーションと暗号デバイス(smartcardsなど)の間のインターフェイス
- 問題:
  - 何度も安全性が破られてきた
  - よく知られていない次善策
- この論文では:
  - 鍵管理に対する形式的モデルを定義
  - 到達可能性の決定可能性を証明
  - 効果的な自動化を実現

# 安全性が破られた例: Key Separation Attack

- 基本的な定義:
  - 鍵は「ハンドル」を介してアクセスされる
    - 記法:  $h(n,k)$  は鍵  $k$  に対するノンス  $n$  によるハンドル
    - ハンドル  $h(n,k)$  から鍵  $k$  を直接取り出せない
    - 鍵  $k$  から新しいハンドル  $h(n,k)$  を直接作れない
  - 鍵は「属性」を与えられている (*wrap*, *decrypt*, *sensitive* など)
    - 正確に言うと、ハンドルにおけるノンスが属性を与えられている

- 攻撃者はハンドル  $h(n_1, k_1)$  と  $h(n_2, k_2)$  を知っている
- $n_1$  は属性 *sensitive* と *extract* を持ち,  
 $n_2$  は属性 *wrap* と *decrypt* を持つ.
- 攻撃:
  1. Wrap:  $h(n_2, k_2), h(n_1, k_1) \rightarrow \text{senc}(k_1, k_2)$
  2. SDdecrypt:  $h(n_2, k_2), \text{senc}(k_1, k_2) \rightarrow k_1$

$\Rightarrow$  *sensitive*なのに、 $k_1$  が取り出せてしまう

# 形式的モデル

- 状態 (S, V)
  - S は, 攻撃者の知識を表す基底項の集合
  - V は, 属性を割り当てる付値
  - 例:
    - $S_0 = \{ h(n_1, k_1), h(n_2, k_2) \}$
    - $V_0$  s.t.  $V_0(\text{wrap}, n_2), V_0(\text{decrypt}, n_2), V_0(\text{sensitive}, n_1), V_0(\text{extract}, n_1)$  hold
- API 規則  $T;L \text{ — new } n \rightarrow T';L'$ 
  - T, T' は項の集合, L, L' は命題の集合, n は名前の集合
  - 例:
    - Wrap(Sym/Sym):  $h(x_1, y_1), h(x_2, y_2); \text{wrap}(x_1), \text{extract}(x_2) \rightarrow \text{senc}(y_2, y_1)$
    - SDecrypt:  $h(x_1, y_1), \text{senc}(y_2, y_1); \text{decrypt}(x_1) \rightarrow y_2$
- クエリ (T, L)
  - (T, L) が充足される: ある  $\theta$  と (S, V) が存在し,  $T\theta \subseteq S$  かつ  $V(L\theta)$  を満たす
  - 例:  $(\{ h(x, y), y \}, \text{sensitive}(x))$  は compromised な sensitive な鍵に対するクエリ

# 決定可能性について

- Mode
  - 型と似ている
    - 例:  $h : \text{Nonce} \times \text{Key} \rightarrow \text{Handle}$ ,  $\text{senc} : \text{key} \times \text{key} \rightarrow \text{Cipher}$
  - ill-modedな項
    - 例: 攻撃者が項  $\text{senc}(\text{senc}(k_1, k_2), k_1)$  を導出できる
- 定理:
  - well-modedな規則によって生成されるシステムにおいて, well-modedなクエリの充足可能性を調べるときは, well-modedな項だけを考慮すればよい
- 系:
  - 次のとき, クエリの充足可能性決定問題が決定可能である
    - 名前の集合が有限
    - well-modedな規則がメッセージ長の制約を意味する

# 実験

- NuSMVモデル検査器へのencoding:
  - それぞれの項と属性は, 命題変数としてencodeされる
    - 有界個の鍵とハンドル ⇒  
攻撃者の知識における可能な項の有限集合
  - 各 API 規則は, group instanceでencodeされる
- 方法論:
  - 対称鍵から始める
  - 攻撃が見つかったら, 攻撃を防ぐように API を制限する
  - 非対称鍵を加えて, 以上の操作を繰り返す
- 結果:
  - 既知の攻撃を再発見できた
  - PKCS#11 の古いバージョンに新たな攻撃を見つけた
  - 新しいバージョンの PKCS#11 には攻撃が見つからなかった

*blank slide*