セキュリティプロトコル安全性検証の理想と現実

茨城大学
米山 一樹
kazuki.yoneyama.sec@vc.ibaraki.ac.jp

2016/6/27 CRYPTRECシンポジウム2016

本講演の概要

- TLS（Transport Layer Security）
  - ブラウザ等で利用される重要なセキュリティプロトコル
  - 現在IETFにてv1.3の策定中（頻繁な仕様変更）

- プロトコル設計者は安全性をどうやって保証・確認すればよいのか？
  - 暗号理論的安全性証明
  - 自動検証ツールによる安全性検証
  - 2つのアプローチを俯瞰的に解説

自己紹介

- 元NTTセキュアプラットフォーム研究所
  - 暗号プロトコルの設計・安全性評価に従事
    特に、認証鍵交換の研究
- 現在、茨城大学工学部情報工学科所属

- 注：TLSそのものや実装や運用に関する攻撃にはそれほど詳しくないので、暗号学的視点からのお話になります

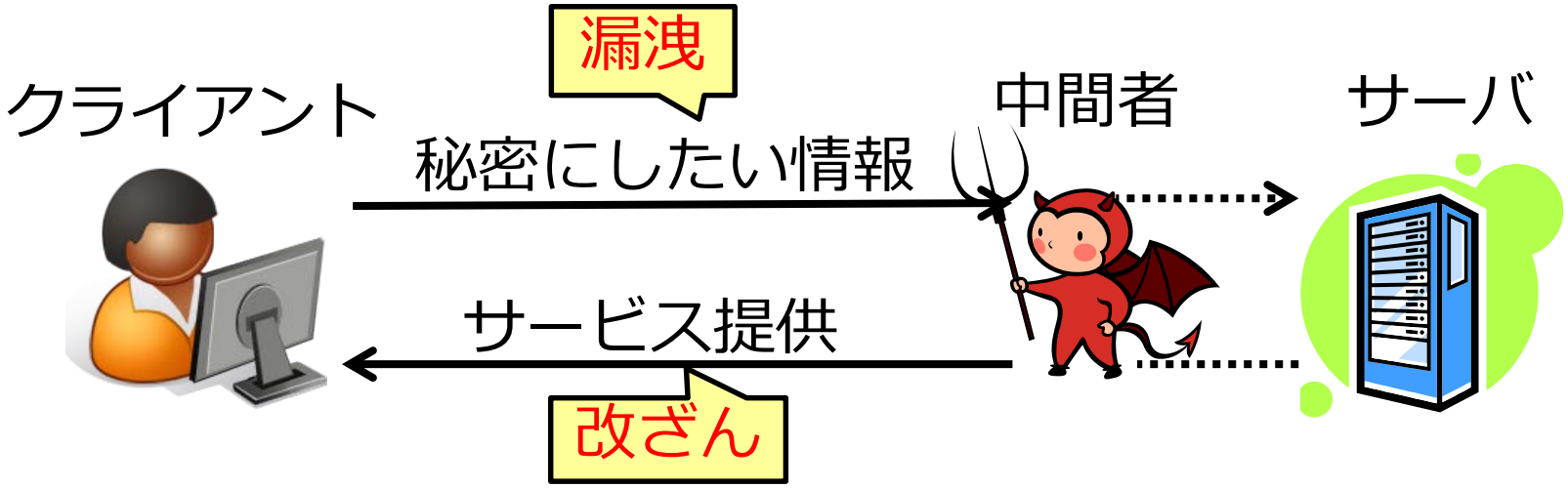
目次

- 背景
- 暗号理論に基づく安全性証明
- 形式手法に基づくツールによる自動検証
- 両アプローチの融合の試み
- 今後の展望

背景

中間者攻撃の脅威

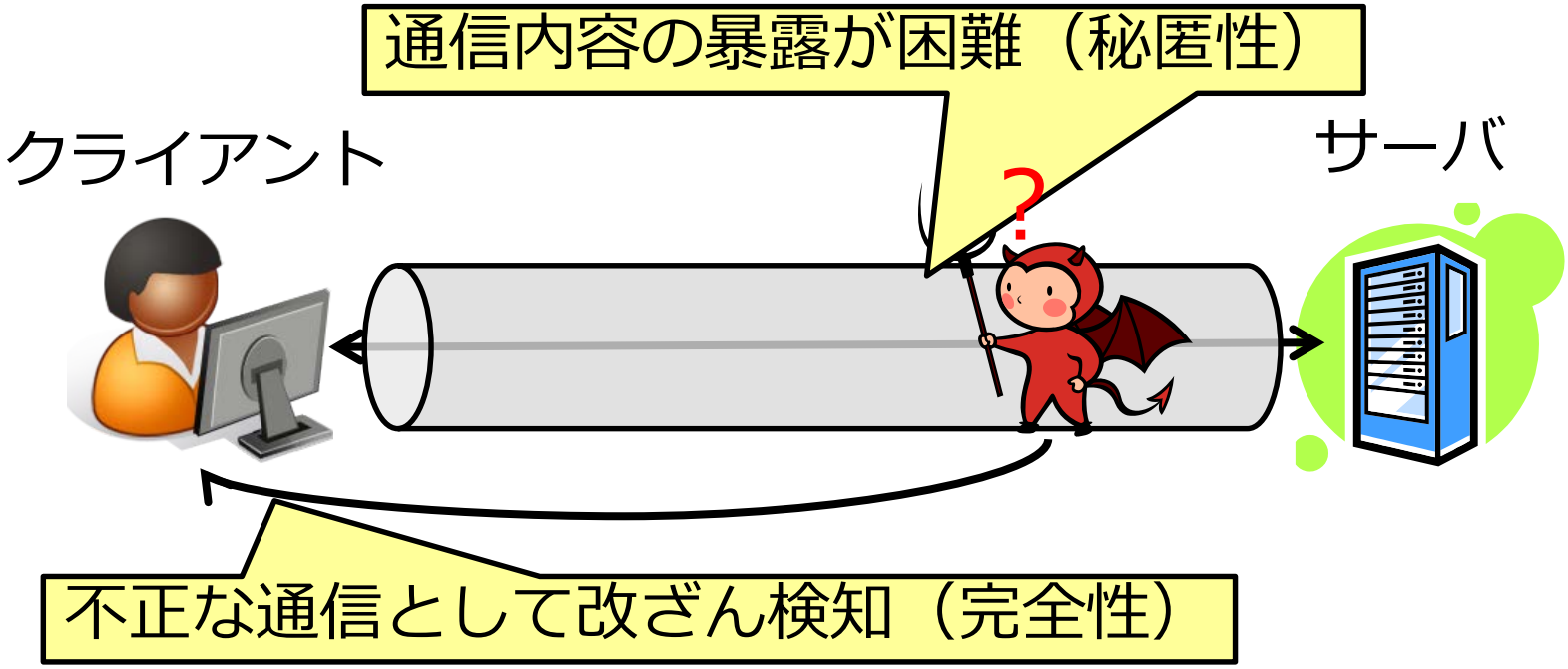
- 通信相手は本物のサーバだろうか？



インターネット自体には中間者を防ぐ仕組みはない

SSL/TLSとは？

- トランスポート層で秘匿性・完全性等を実現するためのセキュリティプロトコル



SSL/TLSの構造

- 2つのサブプロトコルからなる
  - ハンドシェイクプロトコル
    - クライアントとサーバ間で、公開鍵暗号技術を用いて認証を行い、セッション鍵を共有
  - レコードプロトコル
    - 共有したセッション鍵により、共通鍵暗号技術を用いて通信内容を暗号化・改ざん防止

- ハンドシェイク（HS）プロトコルに注目

ハンドシェイクプロトコルのイメージ

認証局

公開鍵 ek の証明書

証明書

証明書を検証

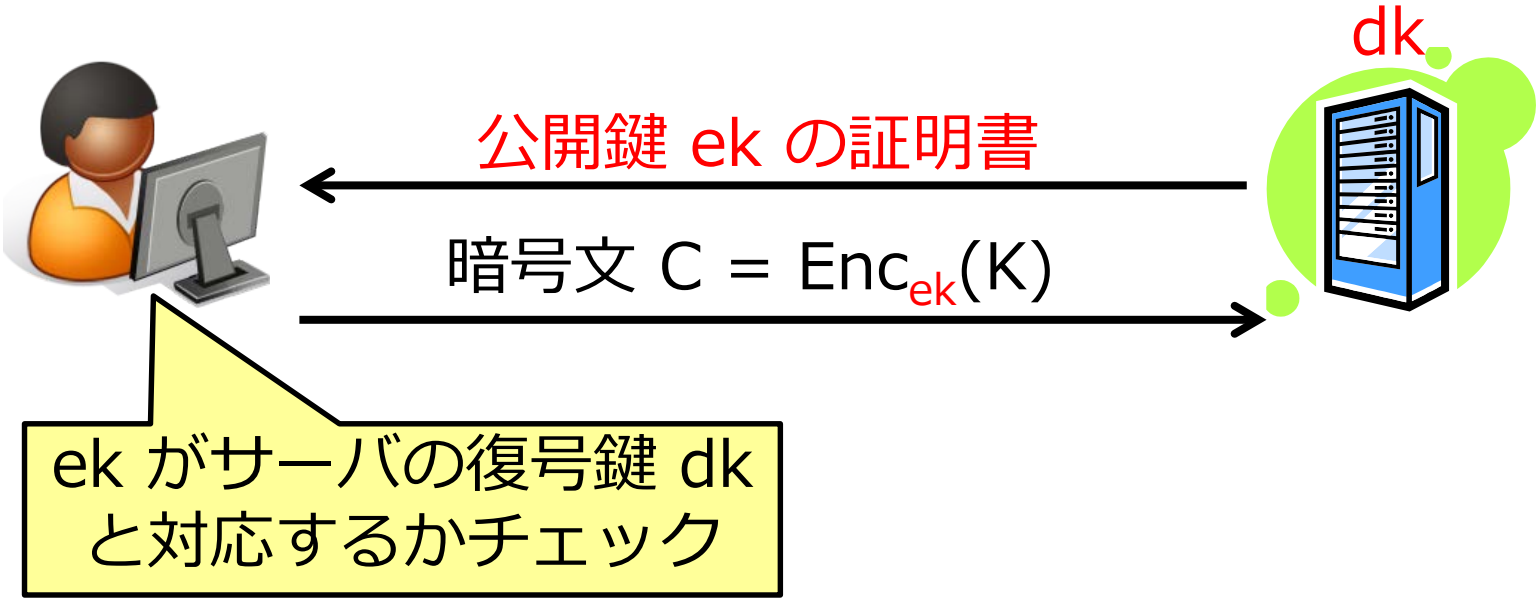
共有鍵 K をランダムに選んで ek で暗号化（$C = Enc_{ek}(K)$）

暗号文 C

dk を用いて C を復号し共有鍵 K を得る（$K = Dec_{dk}(C)$）

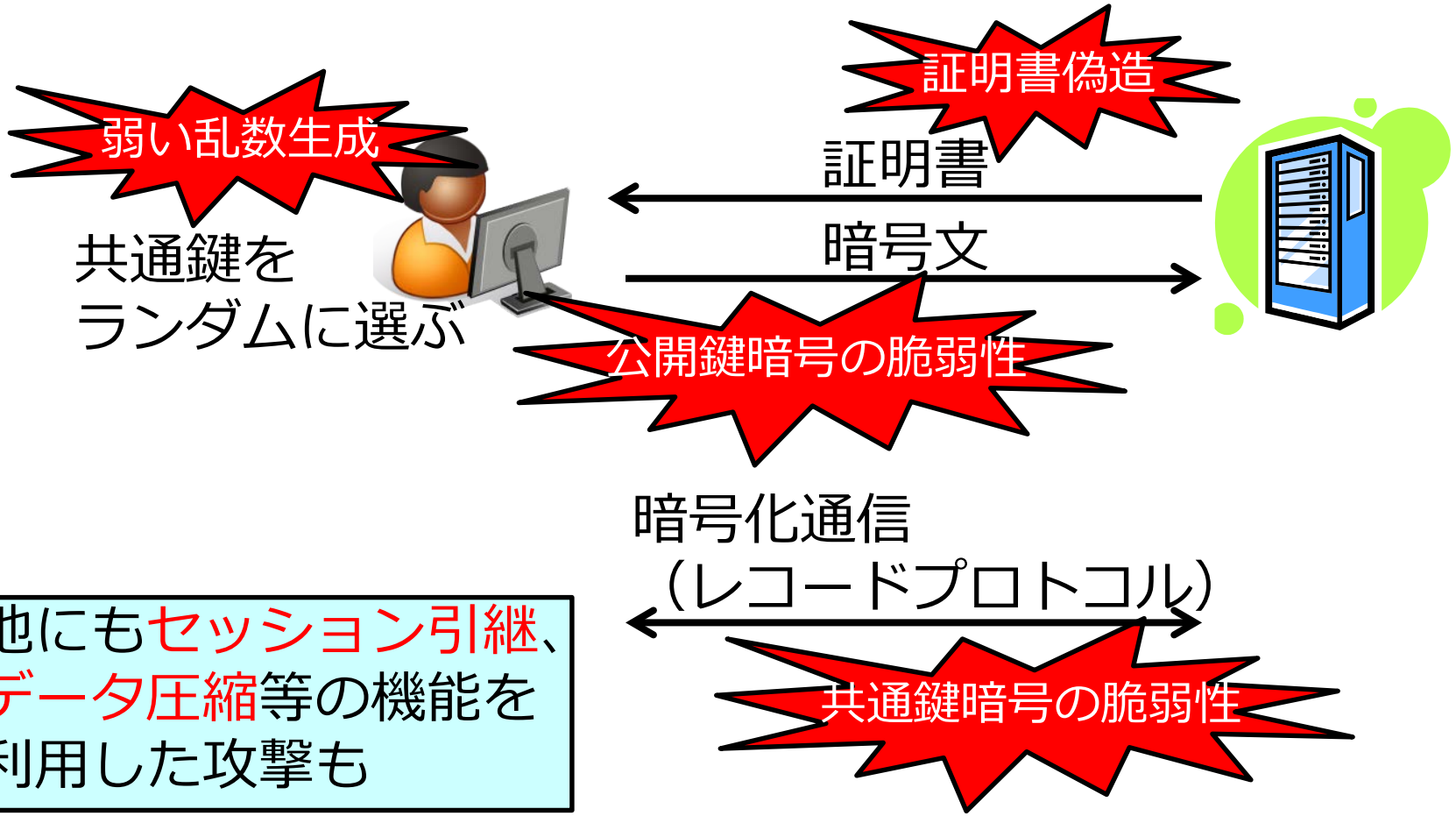
サーバ認証の直感的理解

「正しい復号鍵 dk を持つサーバだけが
共有鍵 K を用いて暗号文を復号できる」
ことを保証



SSL/TLSに対する攻撃

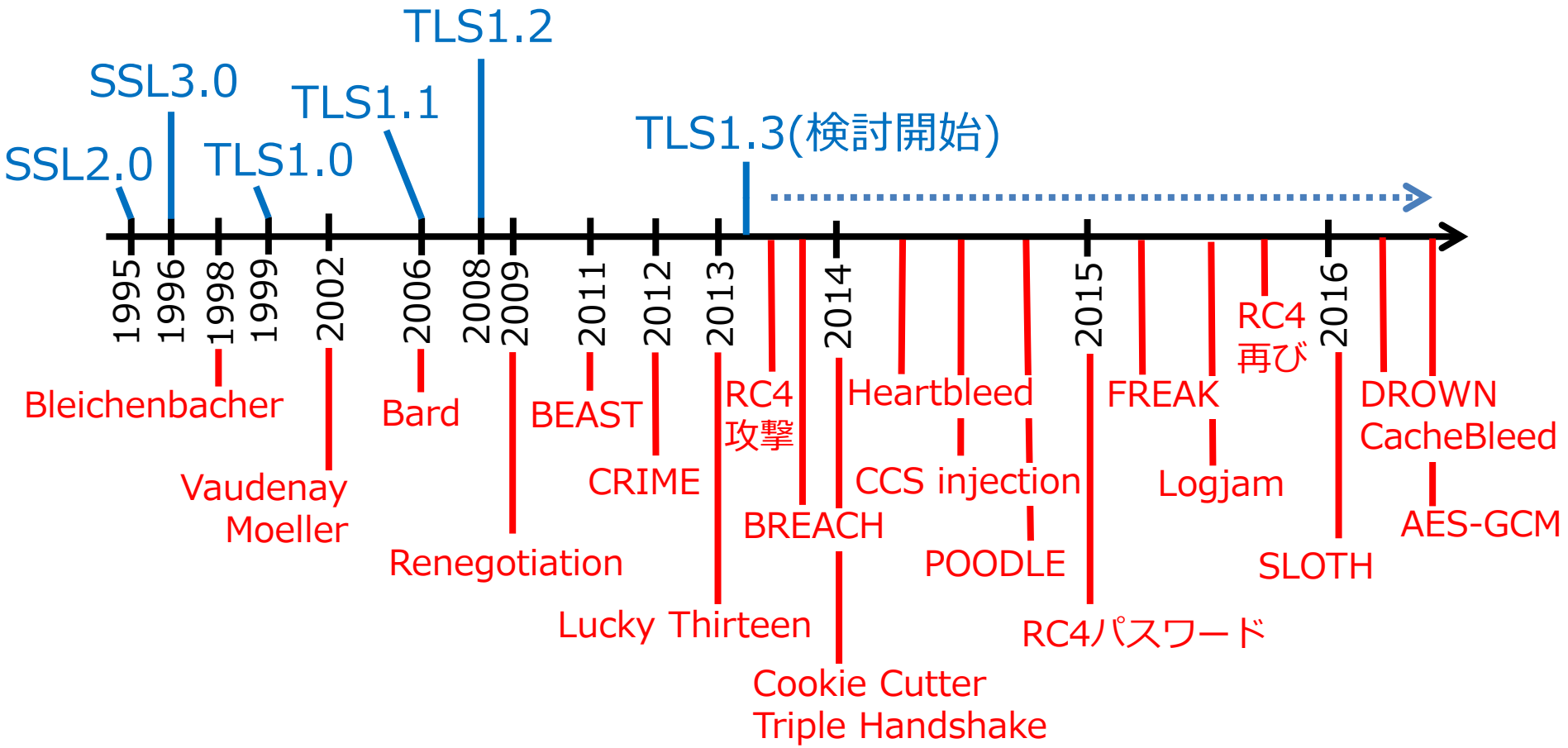
- 一見すると安全に見えるが…



他にもセッション引継、
データ圧縮等の機能を
利用した攻撃も

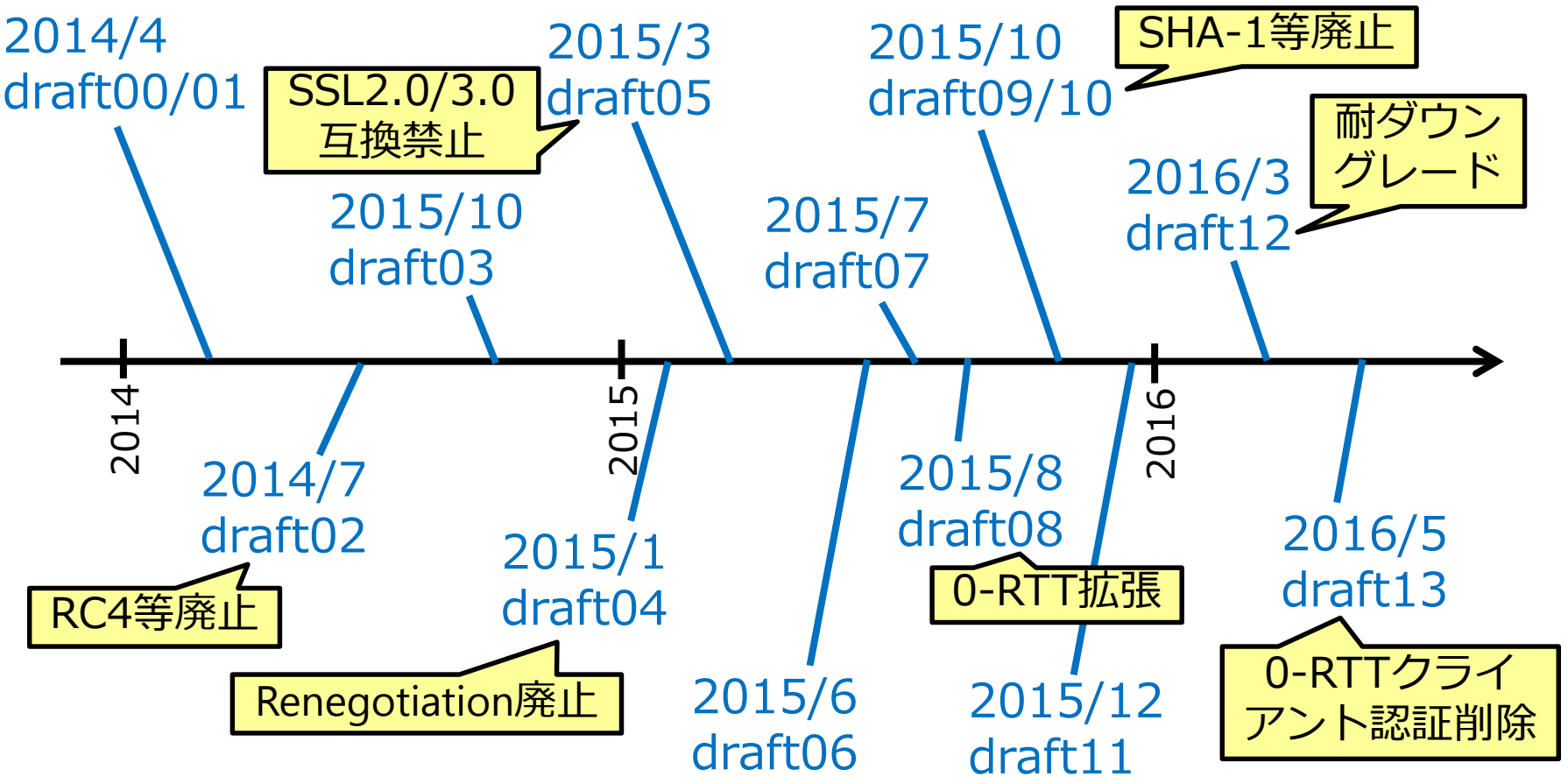
SSL/TLSの攻防の歴史

最近は攻撃発見のペースが加速



TLS1.3の仕様策定の歩み

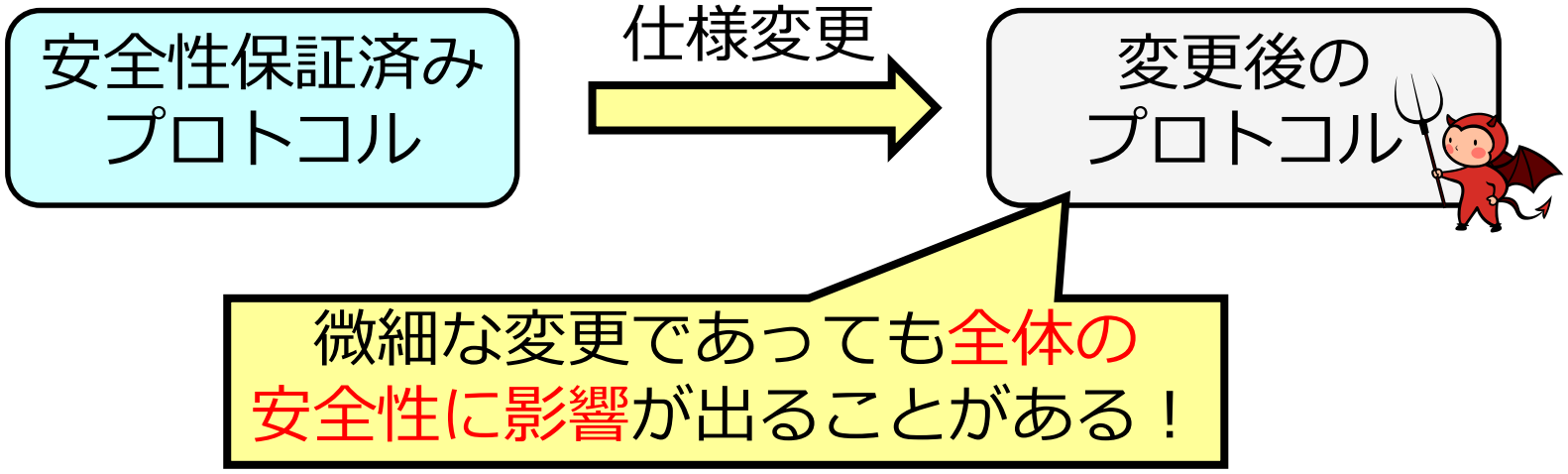
最近は2-3ヶ月ごとに仕様変更



安全性保証・確認の問題

- 重要なセキュリティプロトコルは安全性が保証されていることが望ましい

- 設計者はどうやって安全性を保証・確認？



安全性保証のアプローチと仕様変更サイクル

- 暗号理論に基づく安全性証明
  -  （モデル内の）あらゆる攻撃者に対する安全性を保証
  -  証明が複雑で頻繁な仕様変更に対応困難
- 形式手法に基づくツールによる自動検証
  -  特定の攻撃が可能かどうかだけ検査（≠ 証明）
  -  既存の検証コードを元にすれば、仕様変更後の検証が容易

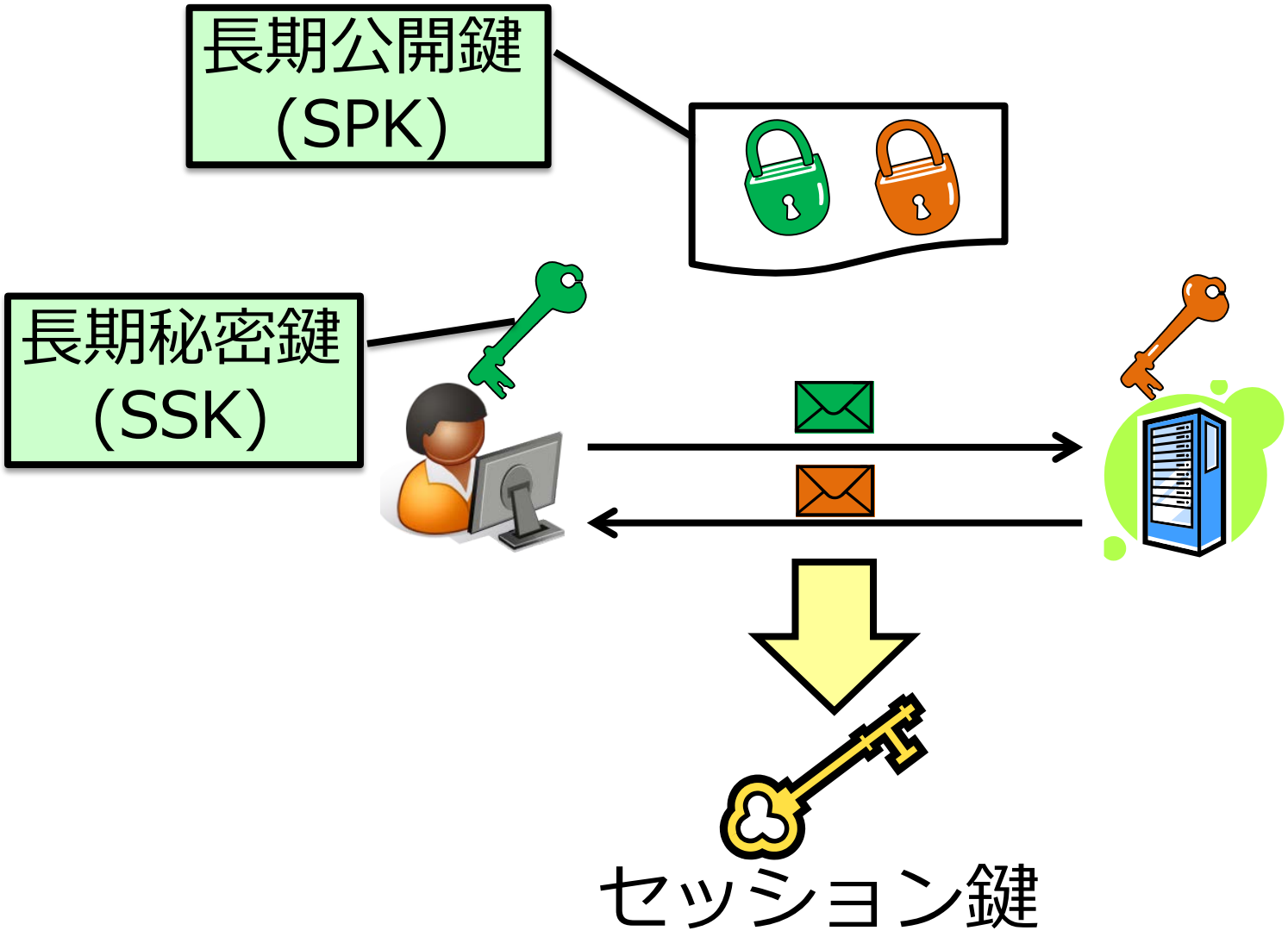
本講演の目的

- SSL/TLSにおける2つの安全性保証・確認手法を簡単に解説
  - 安全性証明：ACCEモデルにおける証明
  - 自動検証：モデル検査手法を用いた検証
  - 融合アプローチ：ゲーム変換の機械補助検証

- 今後、プロトコル検証において両手法をどう利用していくべきか展望

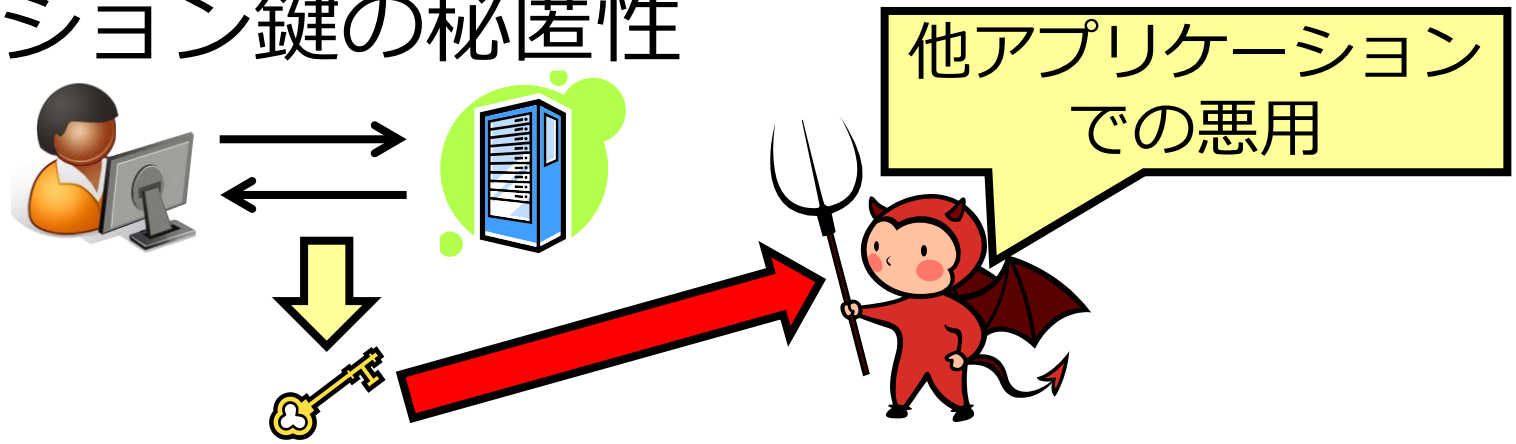
暗号理論に基づく安全性証明

セッション鍵共有（認証鍵交換）のモデル

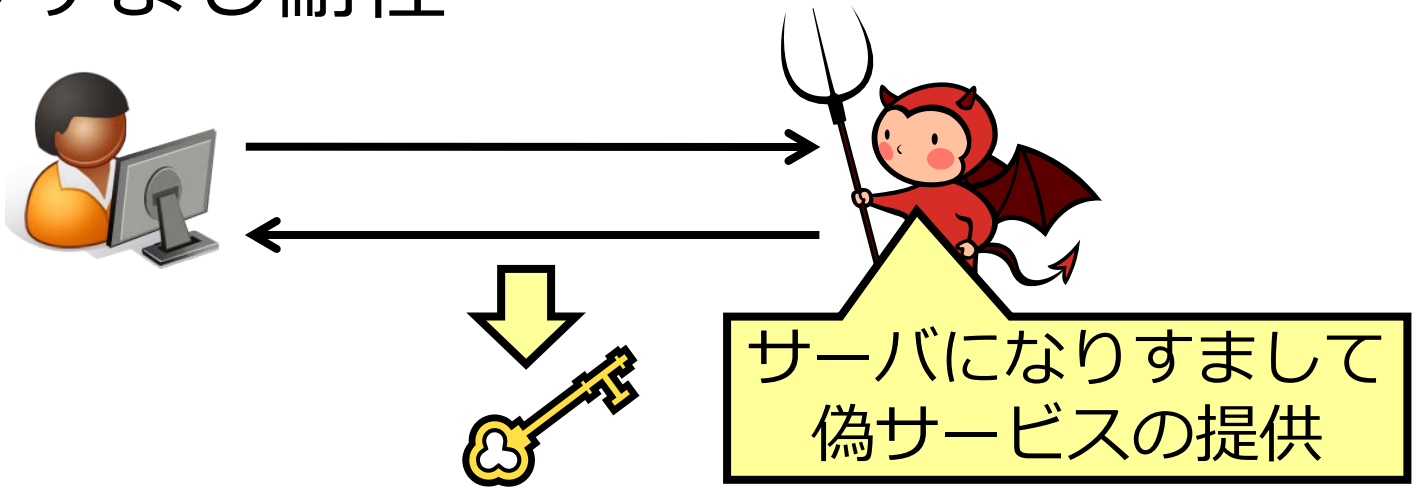


いろんな安全性が要求される

- セッション鍵の秘匿性

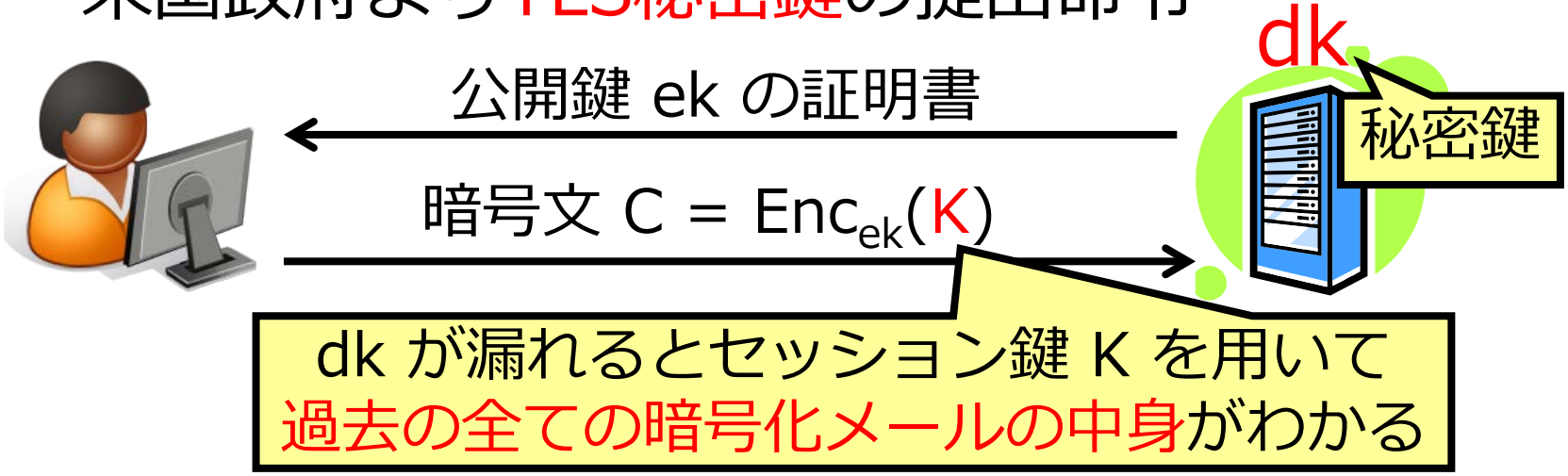


- なりすまし耐性



フォワード安全性

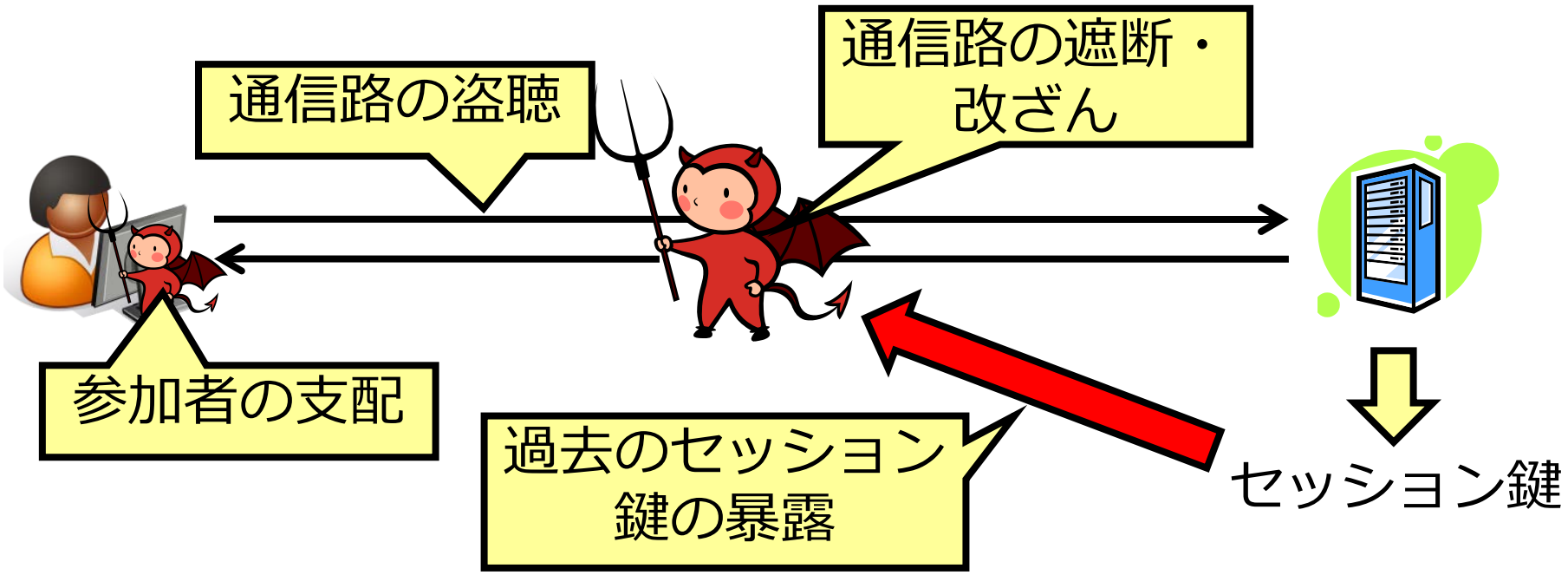
- Lavabit事件
  - スノーデンが使用していたメールサービス
  - 米国政府よりTLS秘密鍵の提出命令



- フォワード安全性
  - 秘密鍵が漏れても、それ以前のセッション鍵は漏れない

Bellare-Rogaway (BR) モデル [BR93]

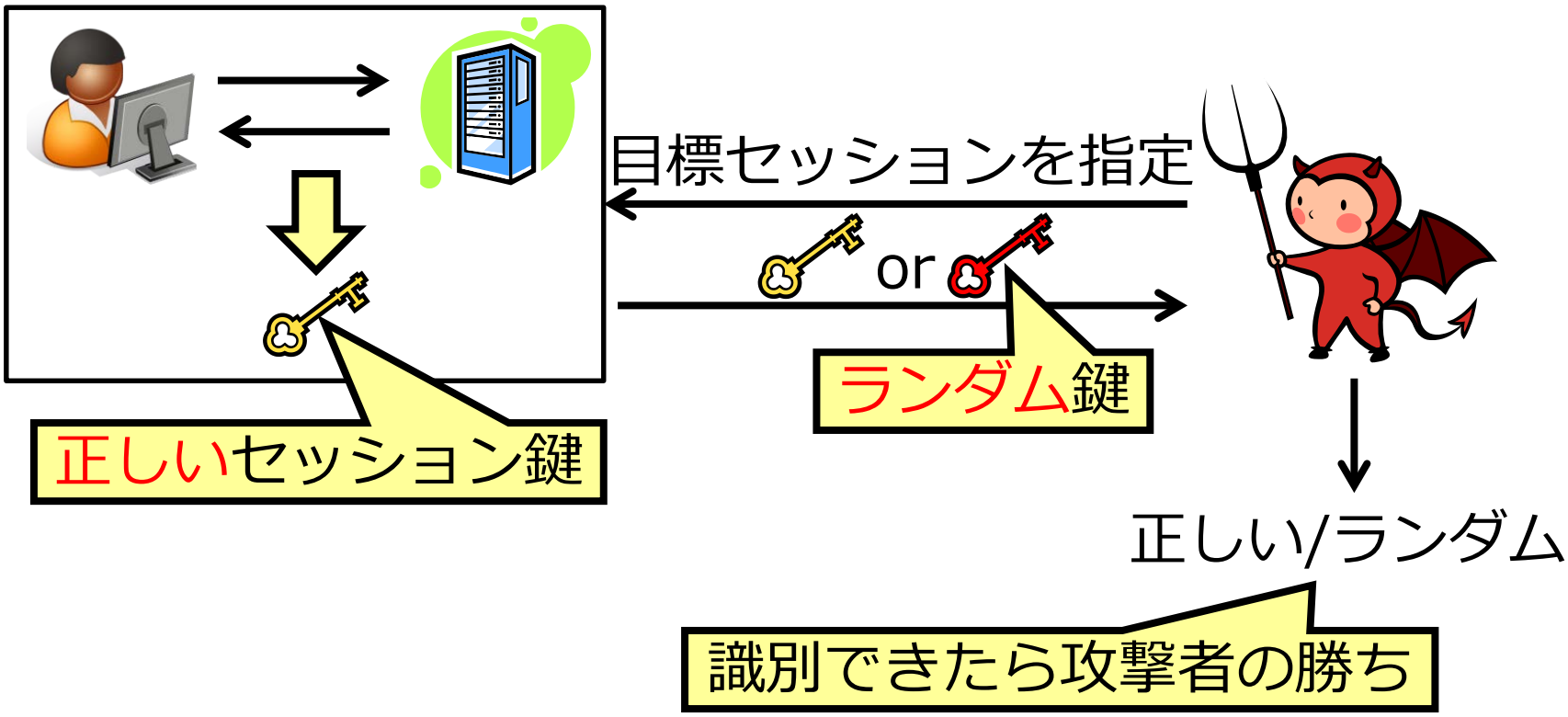
- 初めての認証鍵交換の安全性モデル
  - 任意の中間者攻撃の状況を表現
  - なりすまし耐性、フォワード安全性などを包含



[BR93]:Mihir Bellare, Phillip Rogaway, “Entity Authentication and Key Distribution.” CRYPTO 1993: 232-249

セッション鍵の安全性の定式化

- 正しいセッション鍵とランダムな鍵を識別
  - プロトコルが安全 ⇔ ¬∃ 識別できる攻撃者



TLSのハンドシェイク（HS）プロトコル

1.暗号スイートの合意

使用する暗号方式の組み合わせ

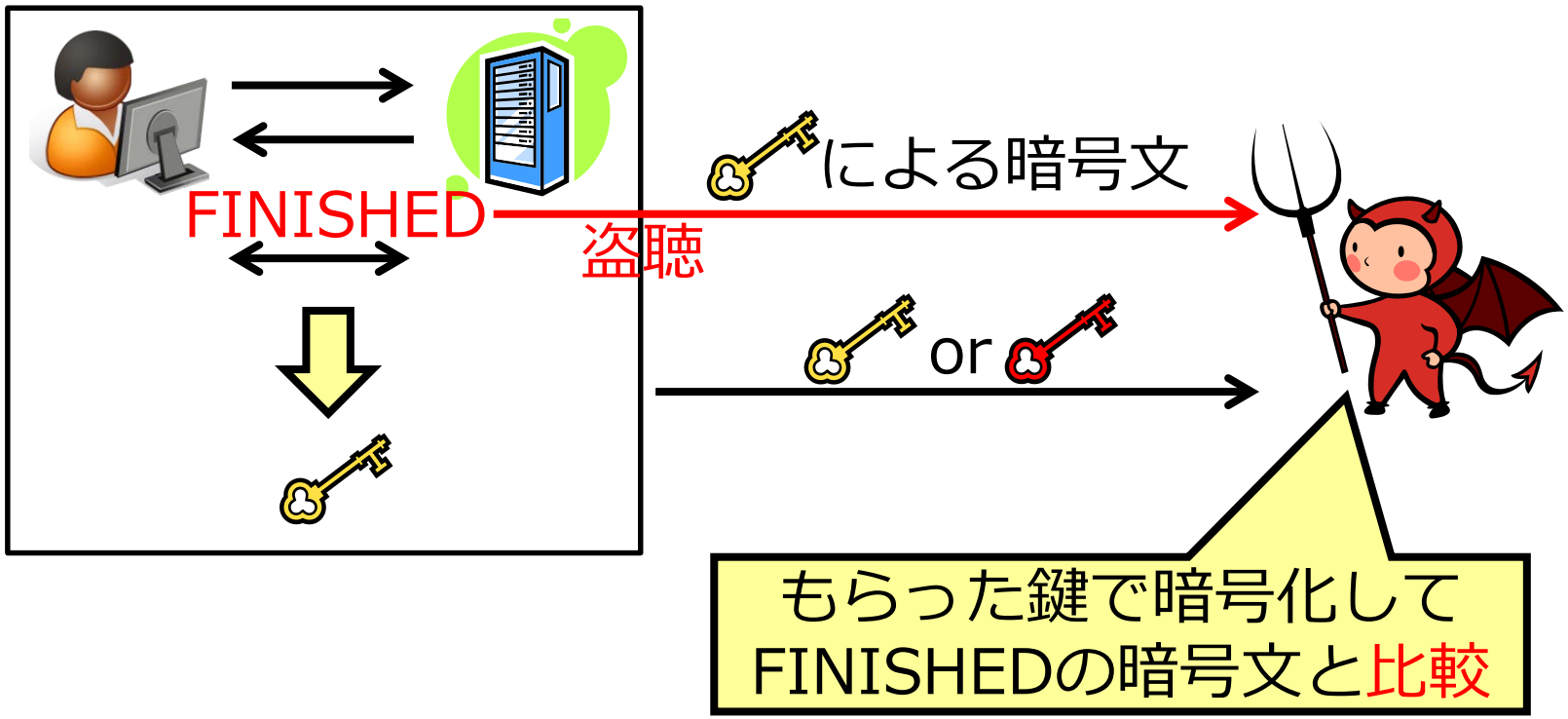
2.セッション鍵共有

合意した暗号スイートを使用

3.FINISHEDメッセージ

セッション鍵を用いて公開情報を暗号化して送り合い、自分のセッション鍵での暗号文と等しいかチェック

HSプロトコルはBRモデルで安全ではない



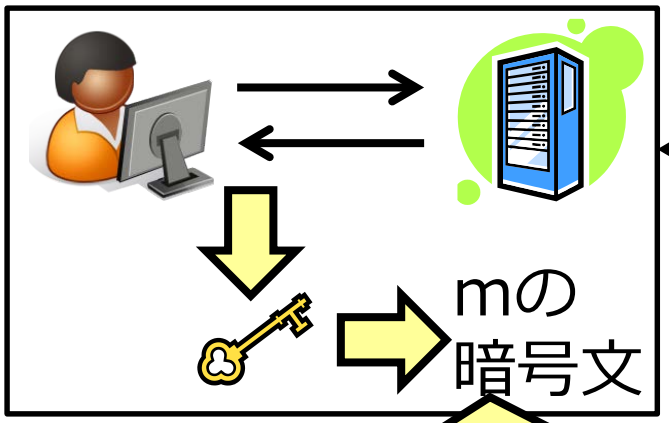
の時は一致し、 の時は
不一致なので、簡単に識別が可能

TLSに要求される安全性

- 一般の認証鍵交換の場合
  - セッション鍵がどんなアプリケーションで使われても安全でなくてはならない
  - TLSにはover-kill
- TLSの場合
  - HSプロトコル + レコードプロトコル全体で安全ならOK

Authenticated Confidential Channel Establishment (ACCE) モデル [JKSS12]

- 指定した平文の暗号文とランダムな平文の暗号文を識別



平文 m

mの暗号文 or m'の暗号文

ランダム平文

mの
暗号文

レコードプロトコルに対応

m/ランダム or 偽造暗号文

識別するか  の暗号文を偽造できたら攻撃者の勝ち

[JKSS12]:Tibor Jager, Florian Kohlar, Sven Schäge, Jörg Schwenk:
On the Security of TLS-DHE in the Standard Model. CRYPTO 2012: 273-293

TLS1.2の安全性証明

- 暗号スイートごとにACCE安全性を証明
  - [JKSS12] DHE
  - [KSS13] DH, RSA
  - [KPW13] CCA
  - [LSY+14] PSK
  - [DS15] 暗号スイートの合意込
- HSプロトコルの安全性証明
  - [BFK+14] DH, RSA

TLS1.2の安全性についてはだいぶ整理されてきた

TLS1.3の安全性証明

- BRモデルの拡張による安全性証明
  - [DFGS15] draft-05
  - [DFGS16] draft-10
  - [LXZ+16] draft-10（multiple HS）
- ダウングレード耐性の証明
  - [BBF+16] draft-10
- 鍵確認安全性の証明
  - [FGSW16] draft-10

draft-11以降の安全性証明は未発表

形式手法に基づくツールによる自動検証

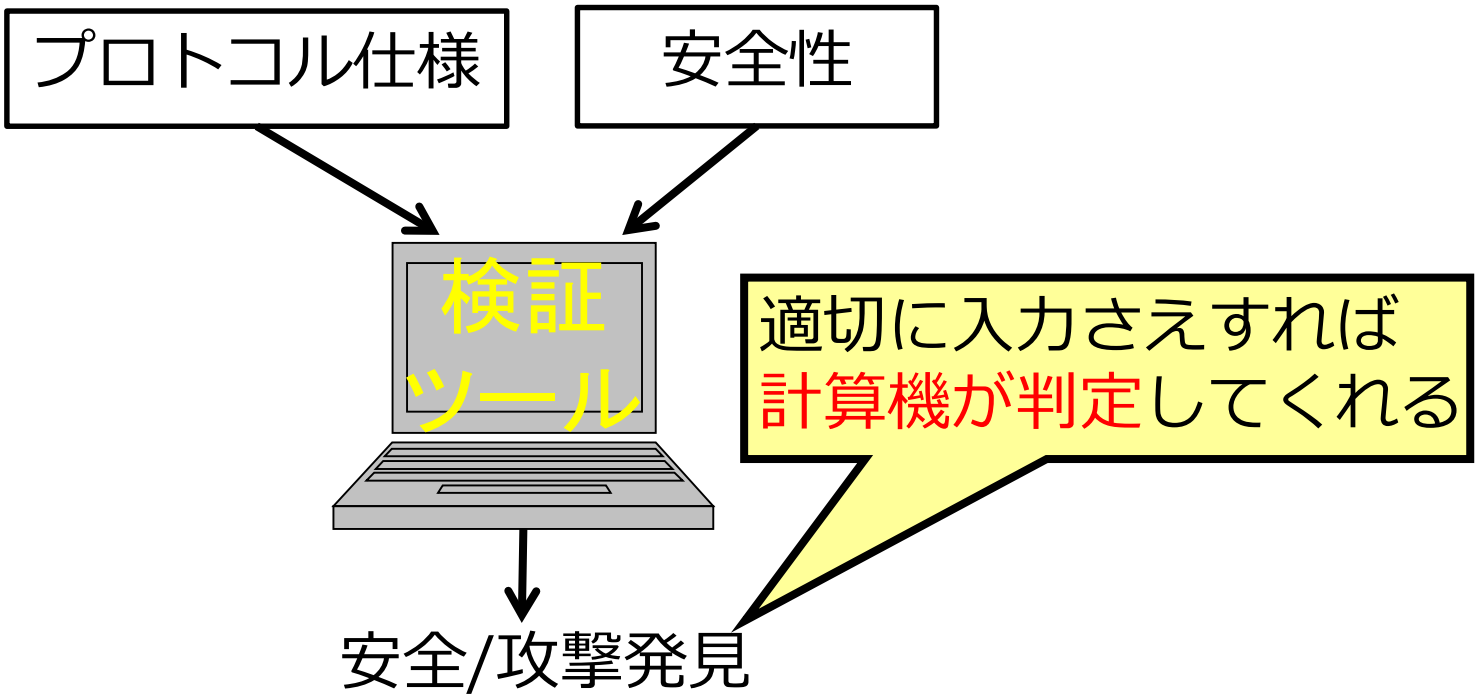
暗号理論的安全性証明の問題点

- プロトコルの進化に伴う複雑化
  - 適切な安全性定義が難しい
  - 安全性証明する方も確認する方も大変

仕様変更サイクルに追いつかない！

形式手法(Formal method)を用いた安全性検証

- プロトコルの仕様、満たしたい安全性を形式化
- 検証ツールにより自動検証



形式手法とは？

- コンピュータシステムの要求仕様を数学的にモデル化し、その設計や実装が仕様に対して正しいことを検証する技法の総称

- 期待される役割
  - 品質向上：信頼性の高いソフトウェア開発
  - 効率向上：プログラムのテストの（半）自動化
  - 対象の理解向上：問題の構造、概念の明確化

セキュリティプロトコルへの適用

- 要求仕様＝満たすべき安全性要件
  - 秘匿性、認証性、匿名性、etc.
- 設計や実装＝プロトコルの実行記述
- 「プロトコルの安全性要件を数学的にモデル化し、実行記述が安全性要件を満たしていることを検証する」

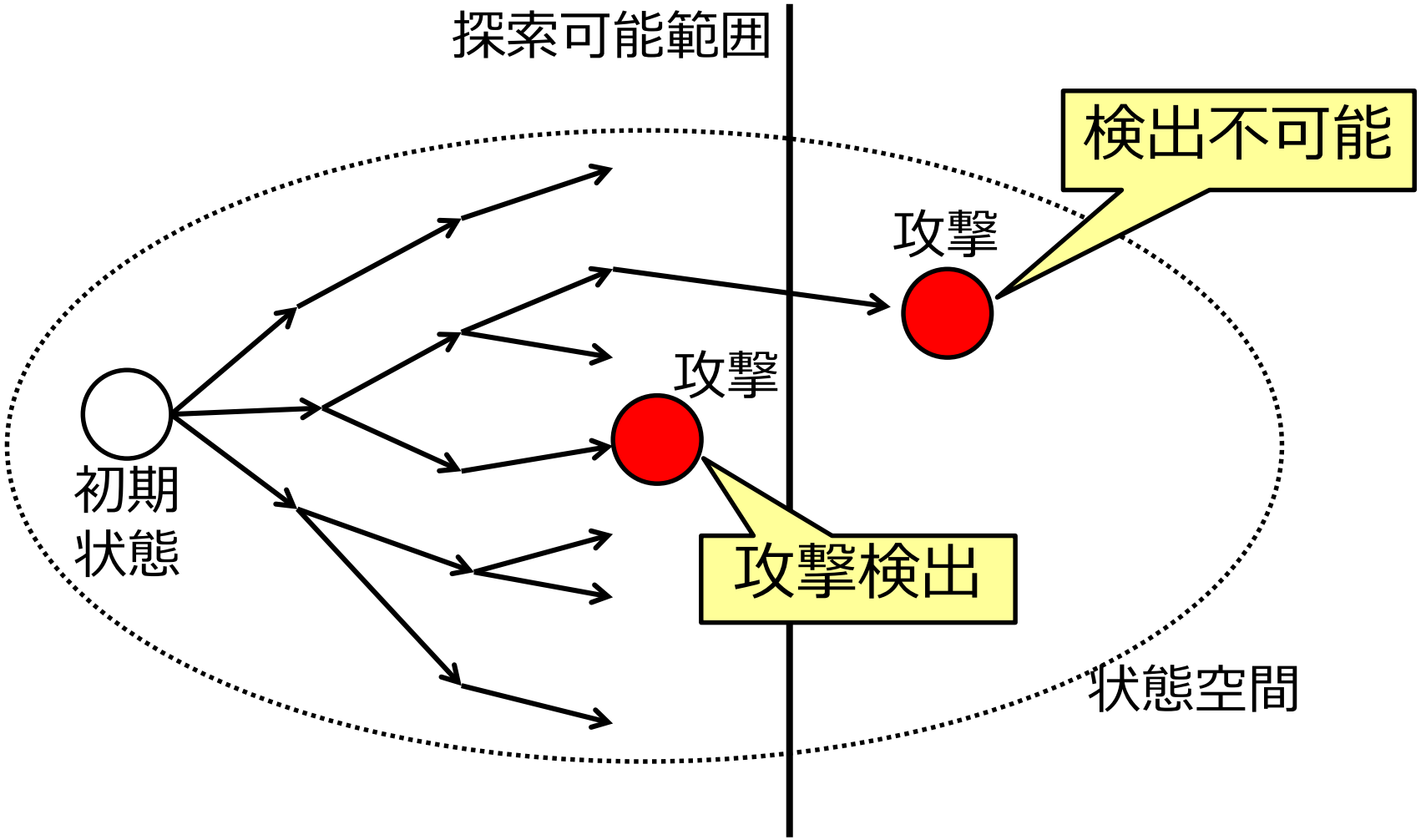
CRYPTRECにおける自動検証の活用例

- エンティティ認証
  - 電子政府推奨暗号としてISO/IEC 9798で定義されているプロトコルを採用
- 自動検証ツールScytherによる攻撃の発見
  - CRYPTREC技術報告書 “Evaluation of ISO/IEC 9798 Protocols”, 2011.
  - ISO/IECにフィードバックされ、方式の修正が行われた

適用の分類

- アプローチによる分類
  - モデル検査系
    - 自動検証可能、（基本的に）有限状態モデル
    - 攻撃の発見に向く
  - 定理証明系
    - 人手による指示が必要、状態空間制限無し
    - 安全性の証明に向く
- 抽象化レベルの分類
  - 記号論的
    - プロトコル中の暗号プリミティブを完全なものと仮定
  - 計算論的
    - 暗号プリミティブそのものの脆弱さも考慮

モデル検査系

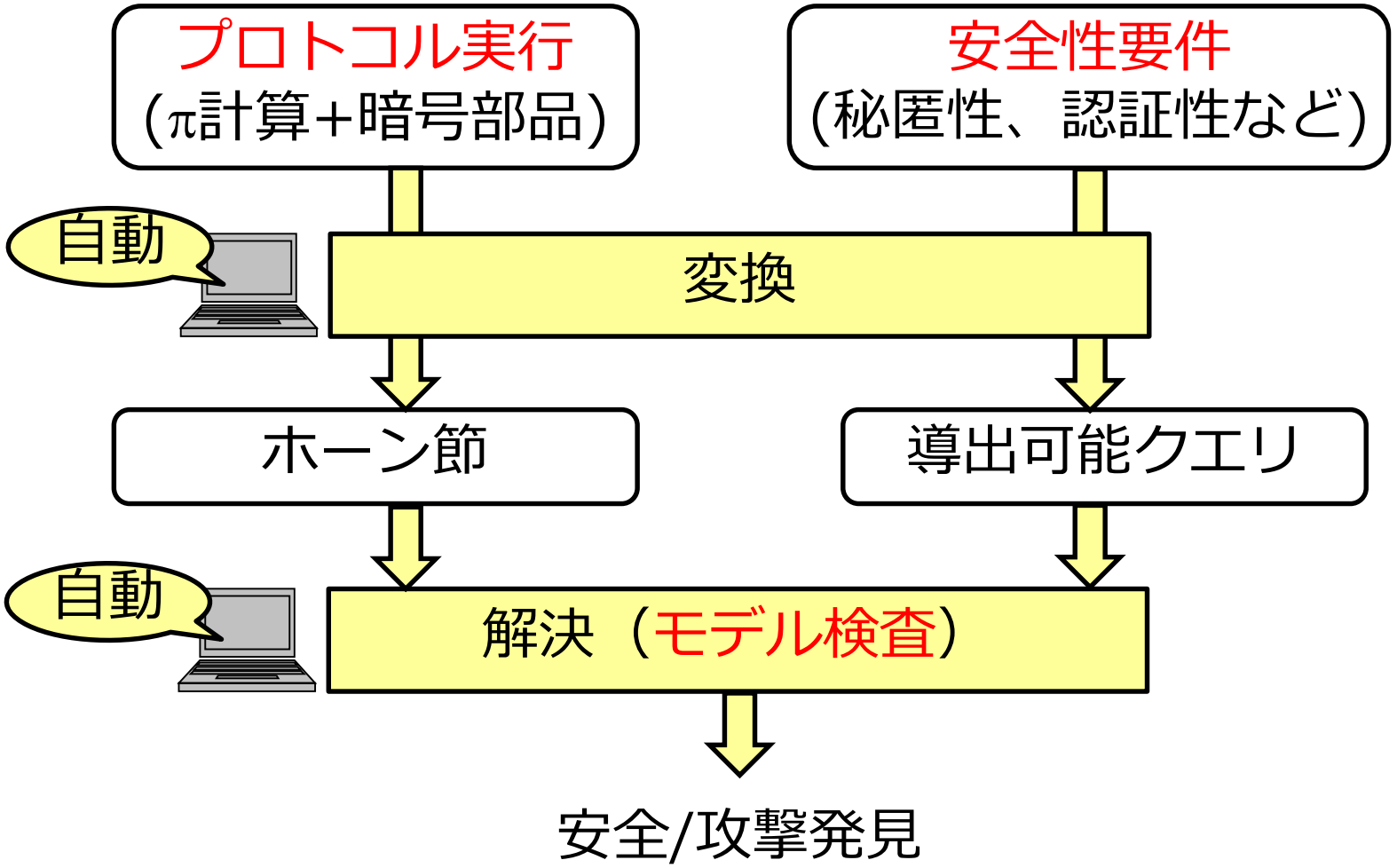


代表的なモデル検査系検証ツール

- ProVerif
  - 記号論的解析を行うモデル検査系ツール
  - フランス国立情報学自動制御研究所(INRIA)のBruno Blanchetが中心に開発
  - 様々なプロトコルや安全性を扱うことができ、汎用性が高い
  - 無限状態を検証可能
    （停止するかどうかは別問題）

http://prosecco.gforge.inria.fr/personal/bblanche/proverif/

ProVerifの検証手順



できること

- 扱える安全性が幅広い
  - 秘匿性：秘密情報が漏れない
  - 認証性：なりすましができない
  - 識別不可能性：2つの実行を見分けられない
- 安全でない場合は具体的な攻撃を出力
  - 探索が停止して攻撃が見つからなければ安全
  - たまに誤った攻撃を検出してしまう（擬陽性）

できないこと

- 保証できる安全性の限界
  - 攻撃が検出されない ≠ 安全性証明
  - 計算論的安全性は検証できない

- 代数的な性質は基本的には扱えない

$(g)^a = g^a, (g)^b = g^b$

べき乗関数を定義

自動では推論
してくれない

$(g^a)^b = (g^b)^a$

等式の関係を
定義する必要あり

TLSの仕様に対する自動検証

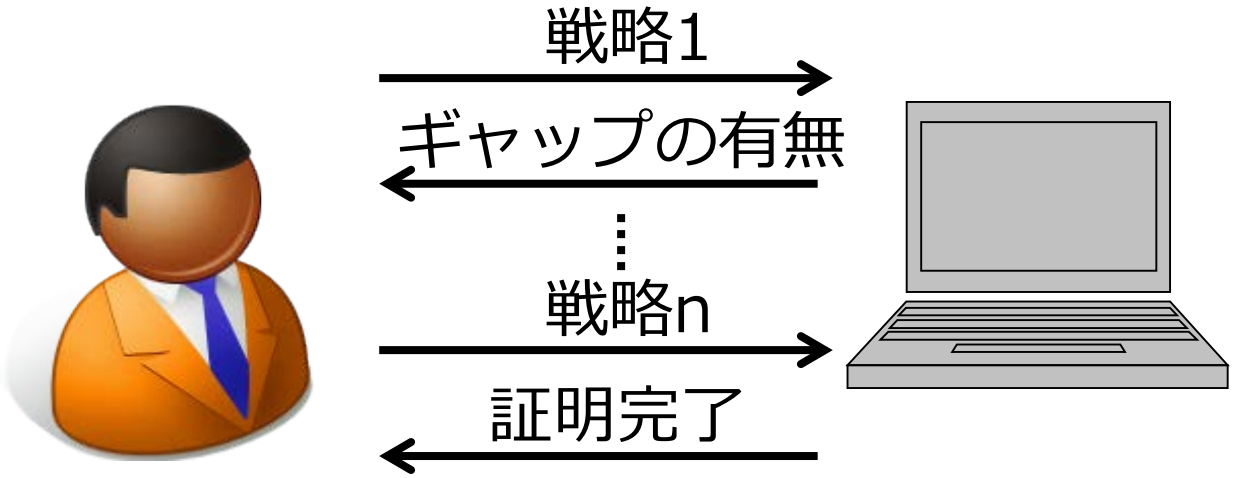
文献	検証対象	ツール
[ABB+05]	TLS1.0	AVISPA
[TV08]	TLS1.1	ProVerif
[Cre08]	TLS1.2	Scyther
[CHSM16]	TLS1.3 draft10	Tamarin prover
[荒井ら15,16]	TLS1.3 draft06-11	ProVerif

その他、TLSの実装や他プロトコルとの組み合わせ（EAP-TLS等）に対する自動検証も盛ん

両アプローチの融合の試み

定理証明系

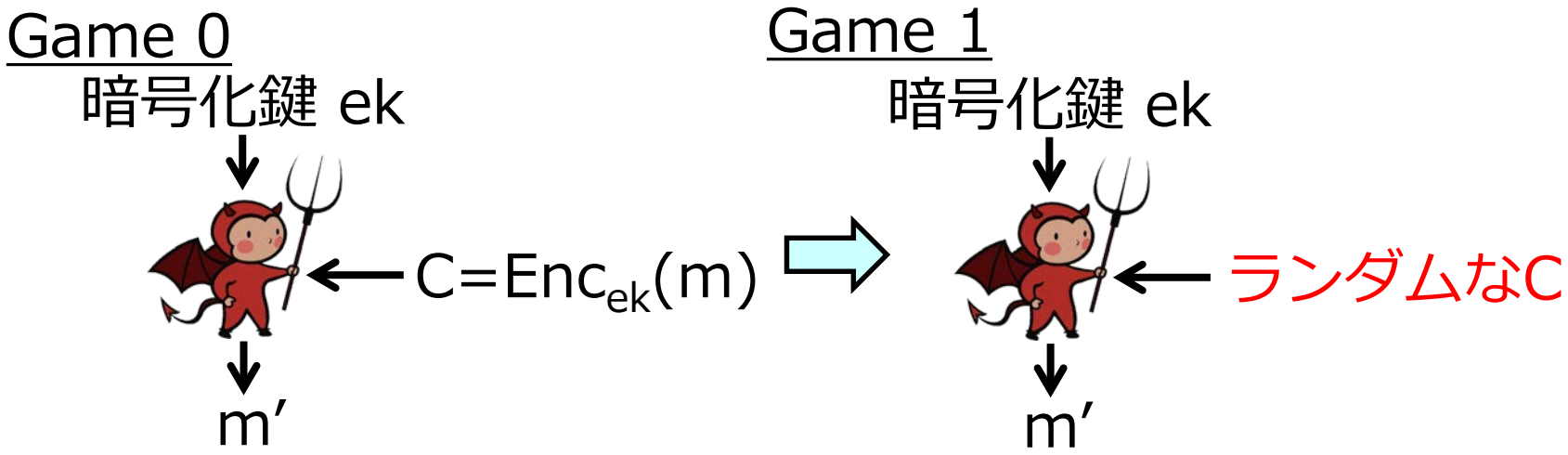
- 計算機と対話しながら証明の戦略を人間が指示することによって証明を生成する



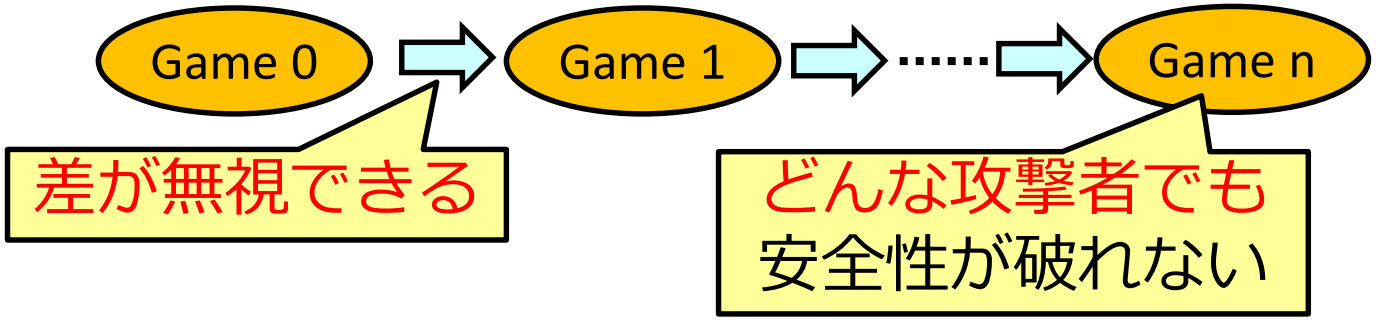
戦略に基づき、論理ギャップの有無を
計算機が判定（状態空間の制限は無い）

ゲーム変換による暗号理論的安全性証明

- 安全性ゲームの変換



- 安全性が破れないゲームまで変換



定理証明系によるゲーム変換

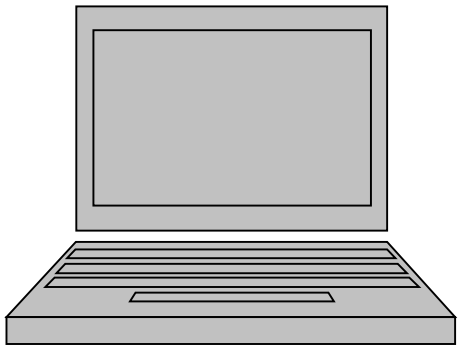
ゲームの変換法は戦略として人間が考え、
計算機は変換可能かどうか検証してくれる

Game 0 → Game 1?

変換OK (ギャップ小)

Game 1 → Game 2?

変換NG (ギャップ大)



Game 1 → Game 2′?

変換OK

代表的な定理証明系検証ツール

- EasyCrypt
  - 確率関係ホーア論理に基づき計算論的安全性を直接証明可能な定理証明系ツール
  - Madrid Institute for Advanced Studies (IMDEA)のGilles Barthelらが中心に開発
  - 適切な推論規則の組み合わせを与えることで変換による差が無視できることを証明
  - 公開鍵暗号、電子署名、認証鍵交換、差分プライバシ、など様々な方式の証明に成功

https://www.easycrypt.info/

TLSの安全性証明における活用例

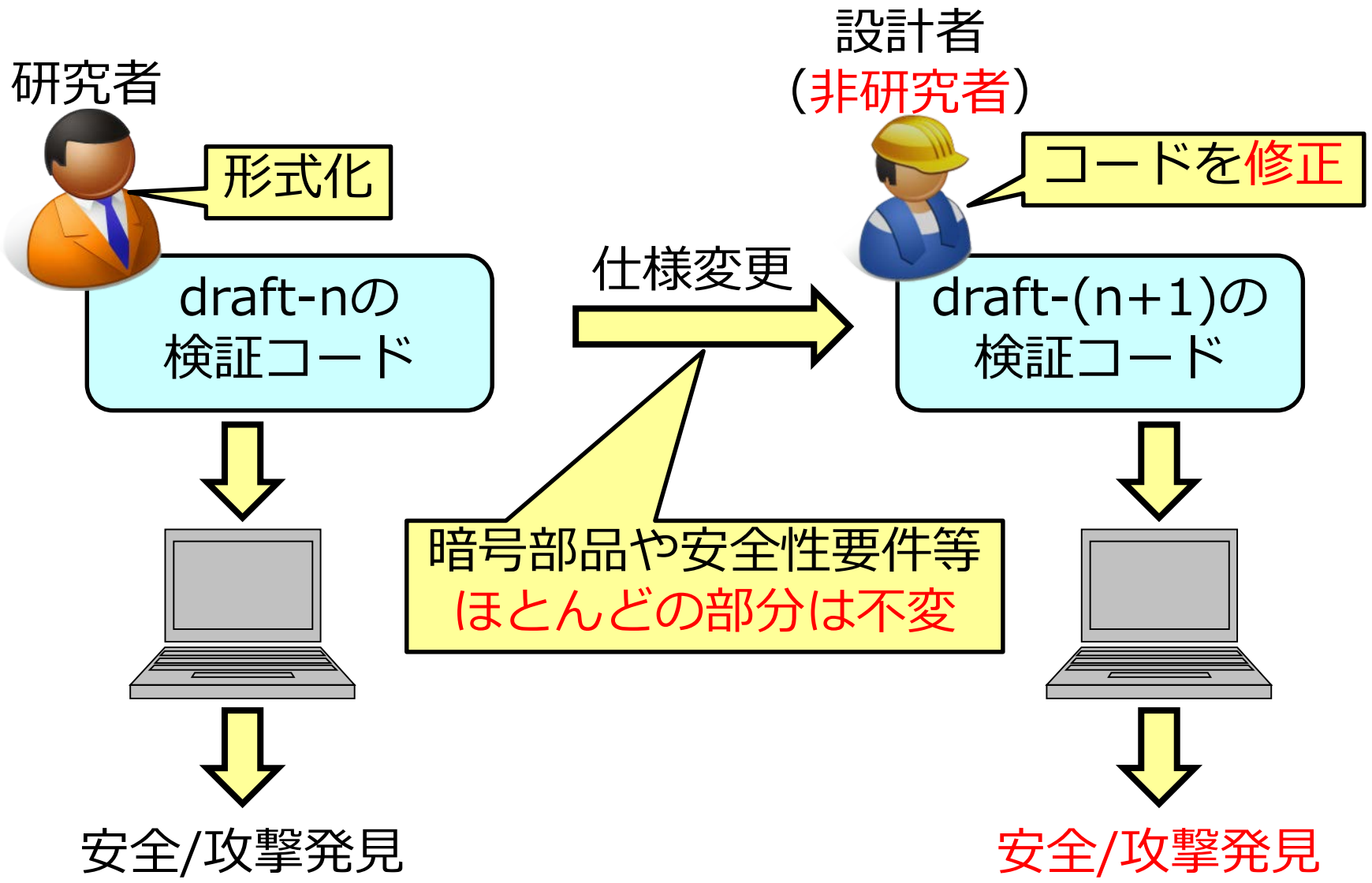
- [BFK+14]
  - TLS1.2のHSプロトコルの安全性証明
    - 暗号スイートを強い安全性を持つ特殊な鍵カプセル化メカニズム(KEM)として定式化
  - ある条件のもとでKEMが上記の安全性を満たすことをEasyCryptを用いて証明
    - 4回のゲーム変換
    - 約3,000行の検証コード

今後の展望

安全性証明の理想と現実

理想		現実
• TLS1.3の適切な安全性モデルが確立	⟺	• ACCEモデルのような統一的なモデルが無い – 仕様が固まらないと困難？
• 仕様変更ごとに設計者（or 誰か）が暗号理論的安全性証明を与える	⟺	• 安全性証明には経験と時間が必要 – 一部の変更でもどこに影響が出るか分からないため – 証明が正しいかどうかの確認にも時間がかかる

自動検証による安全性確認

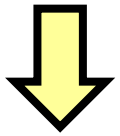


自動検証の活用に向けて

- 設計者
  - 曖昧性のないプロトコル仕様を記述
  - 実行環境の前提、攻撃者の能力、満たしたい安全性等を明確化
- 研究者
  - 再利用性の高い検証コード
  - 各プロトコルに適した評価手法、ツールの選定

自動検証の課題

基本的に記号論的安全性しか検証できない
（部品の脆弱性に起因する攻撃は検出不可能）



1. EasyCryptなどの定理証明系ツールの利用
   - 現状では非専門家には扱いづらい
2. 計算論的安全性の自動検証ツールの利用
   - CryptVerifなど（表現力がまだ不十分）
3. 設計と部品を切り分けたモジュール的解析
   - 結合時の安全性を保証する枠組みが必要