

電子情報通信学会 2015年ソサイエティ大会

依頼シンポジウム「セキュリティプロトコルのためのシステム数理とその周辺」

暗号プロトコル検証入門

磯部 祥尚

産業技術総合研究所
情報技術研究部門

2015年9月9日

この資料の一部またはすべての内容について、作成者の許可なき使用・複製・配布を固くお断りします

1. はじめに

- 暗号プロトコルの基礎

2. NSPKプロトコル*1のモデル化

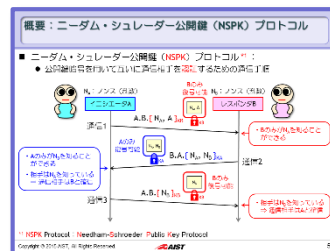
- 並行プロセスによるモデル化

3. NSPKプロトコルの検証

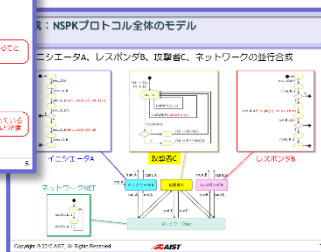
- モデル検査器による網羅的な検証

4. まとめ

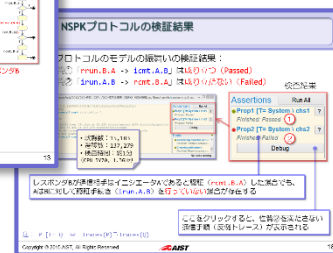
1. はじめに



2. モデル化



3. 検証



概要

暗号プロトコルの形式的な検証方法の紹介（参考：Loweの論文^[1]）

- 暗号プロトコルのモデル化と検証の基礎
- モデル検査器による並行プロセスの検証

[1] G. Lowe : Breaking and Fixing the Needham-Schroeder Public-Key Protocol using FDR, TACAS, LNCS 1055, pp.147-166, 1996.

*1 NSPKプロトコル：公開鍵暗号を用いた暗号プロトコル

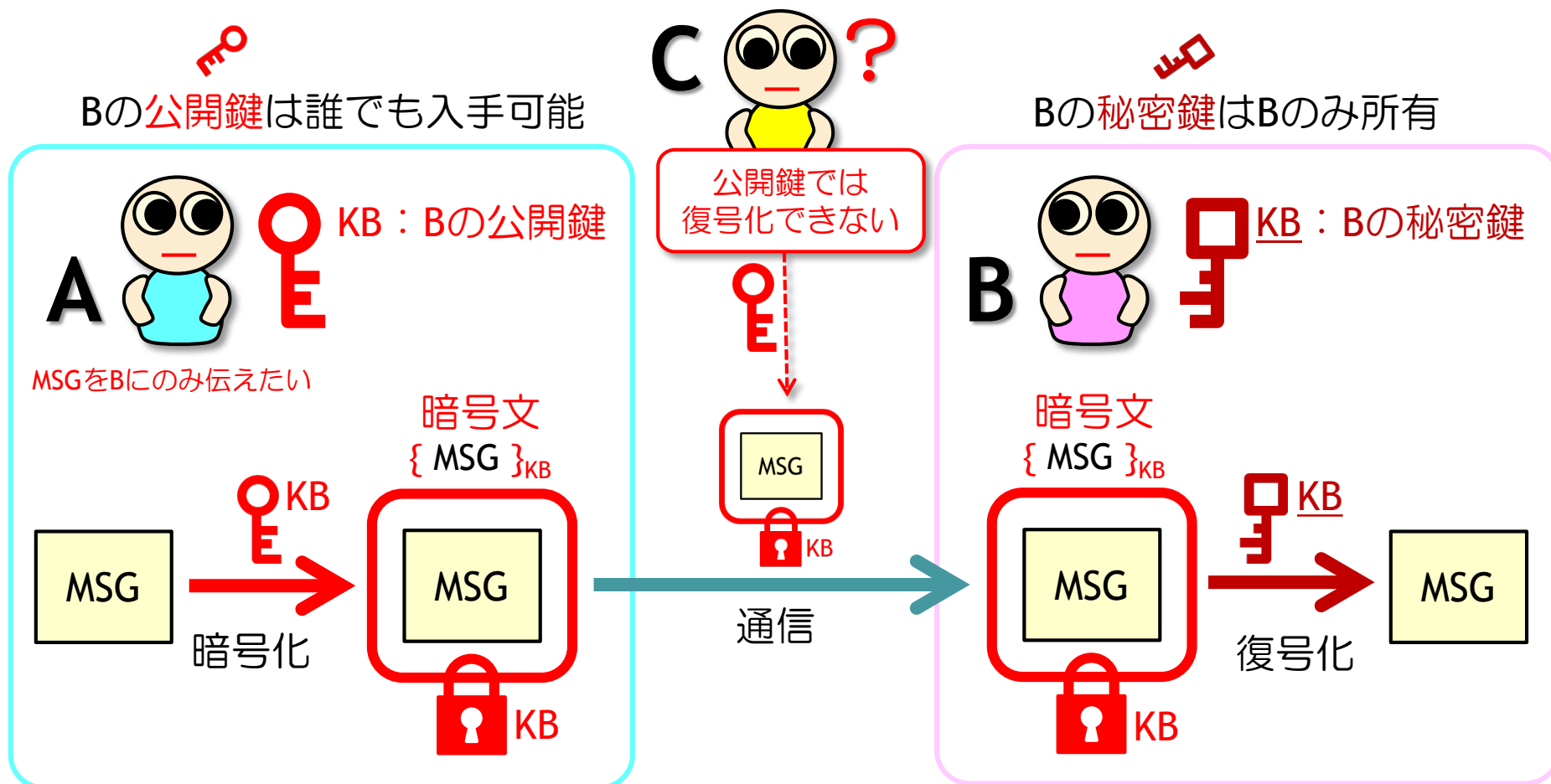
はじめに

- NSPKプロトコルの概要
- NSPKプロトコルの問題

準備：公開鍵暗号方式

■ 公開鍵暗号方式：

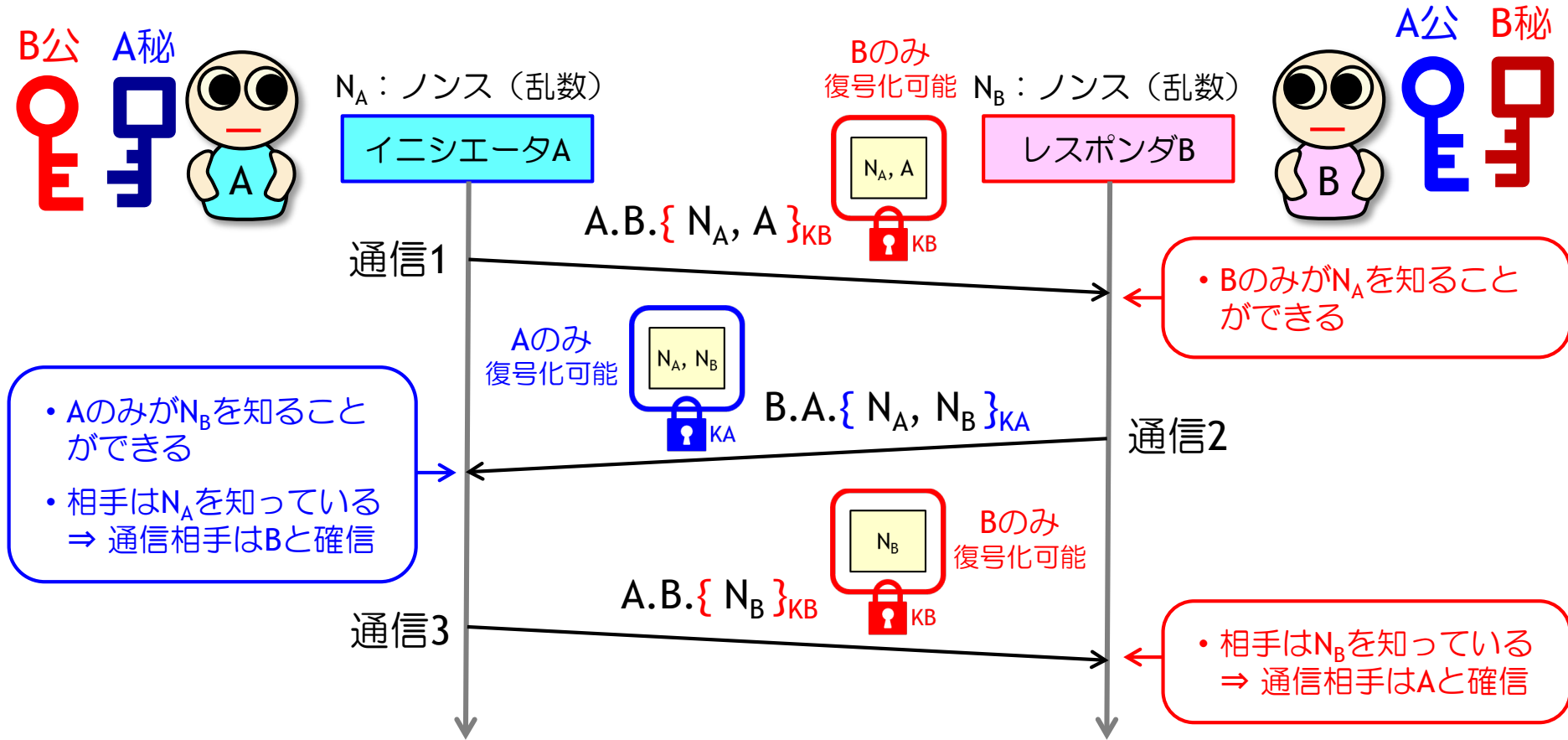
- メッセージを公開鍵で暗号化し、暗号文を秘密鍵で復号化する方式



概要：ニードム・シュレーダー公開鍵（NSPK）プロトコル

■ ニードム・シュレーダー公開鍵（NSPK）プロトコル *1：

- 公開鍵暗号を用いて互いに通信相手を認証するための通信手順



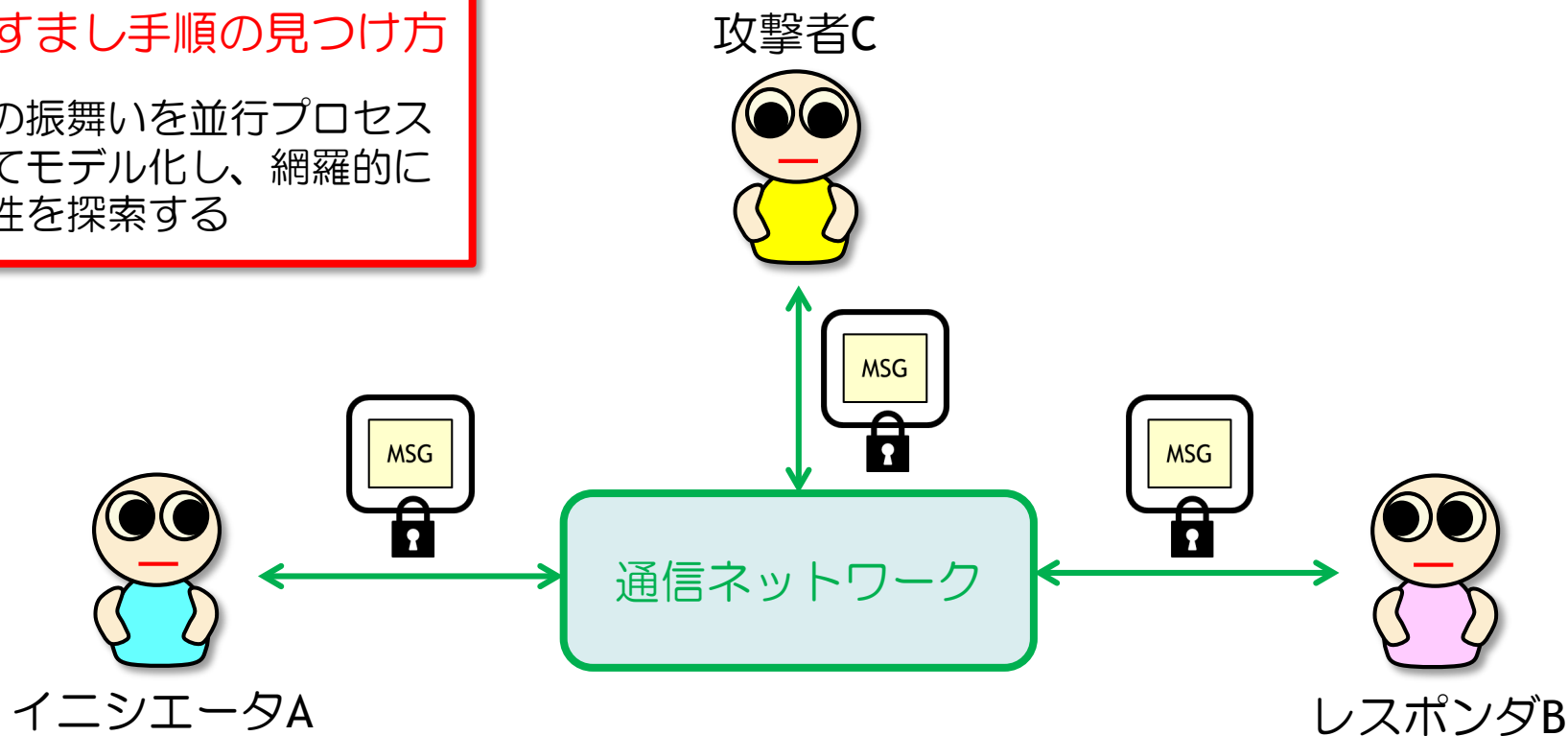
*1 NSPK Protocol : Needham-Schroeder Public Key Protocol

問題：NSPKプロトコルの脆弱性

- NSPKプロトコルでは攻撃者が通信相手に**なりすます**脆弱性が知られている
 - 仮定：攻撃者は送信元の詐称、メッセージの横取りが可能（**暗号解読は不要**）
⇒ 攻撃者CがイニシエータAに**なりすます**ことが可能

本講演で解説：
なりすまし手順の見つけ方

個々の振舞いを並行プロセスとしてモデル化し、網羅的に脆弱性を探索する

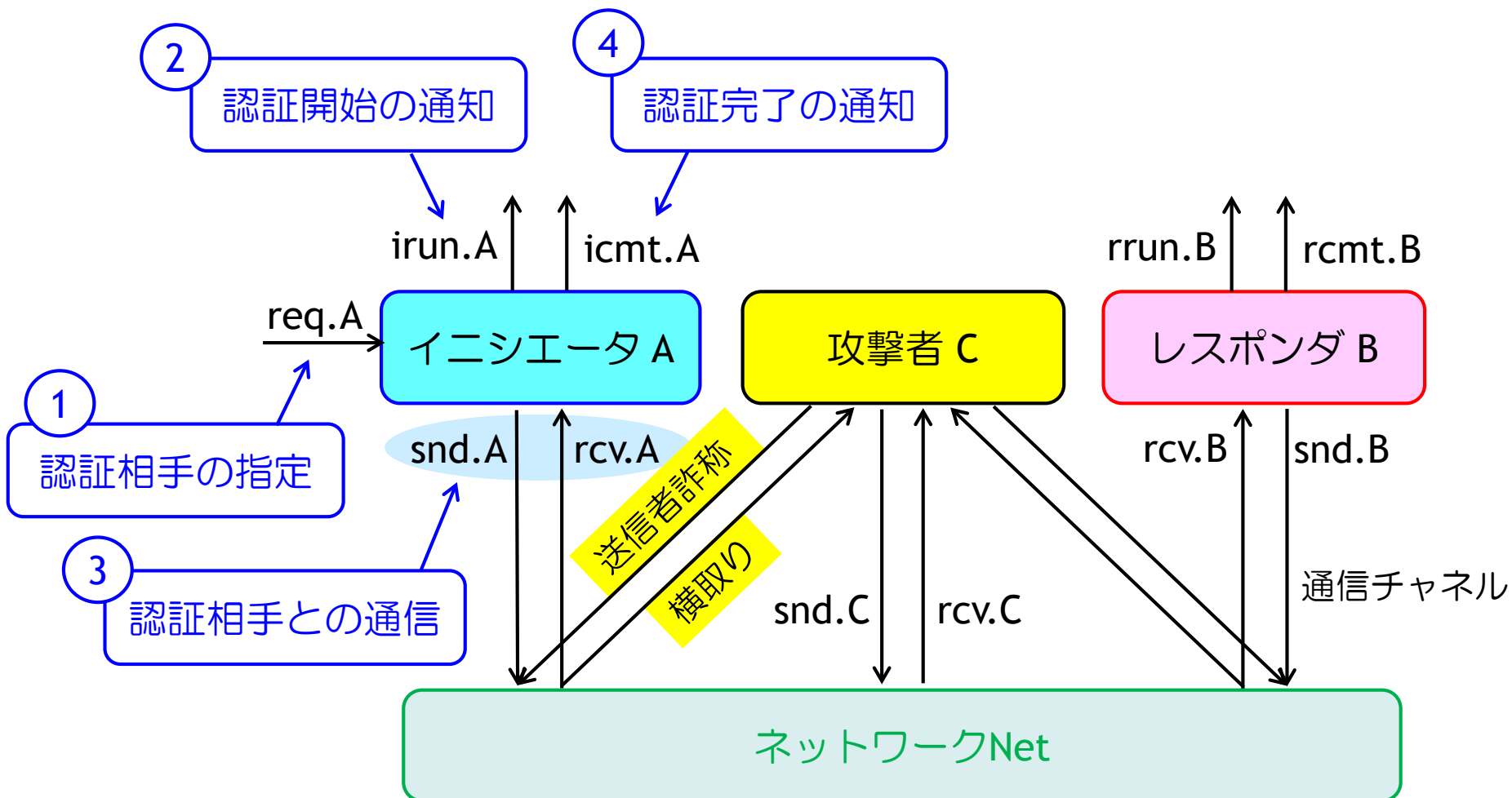


NSPKプロトコルのモデル化

- NSPKプロトコルの構造モデル
- 各要素の振舞い
- 全ての要素の並行合成

構造：NSPKプロトコルの構造モデル

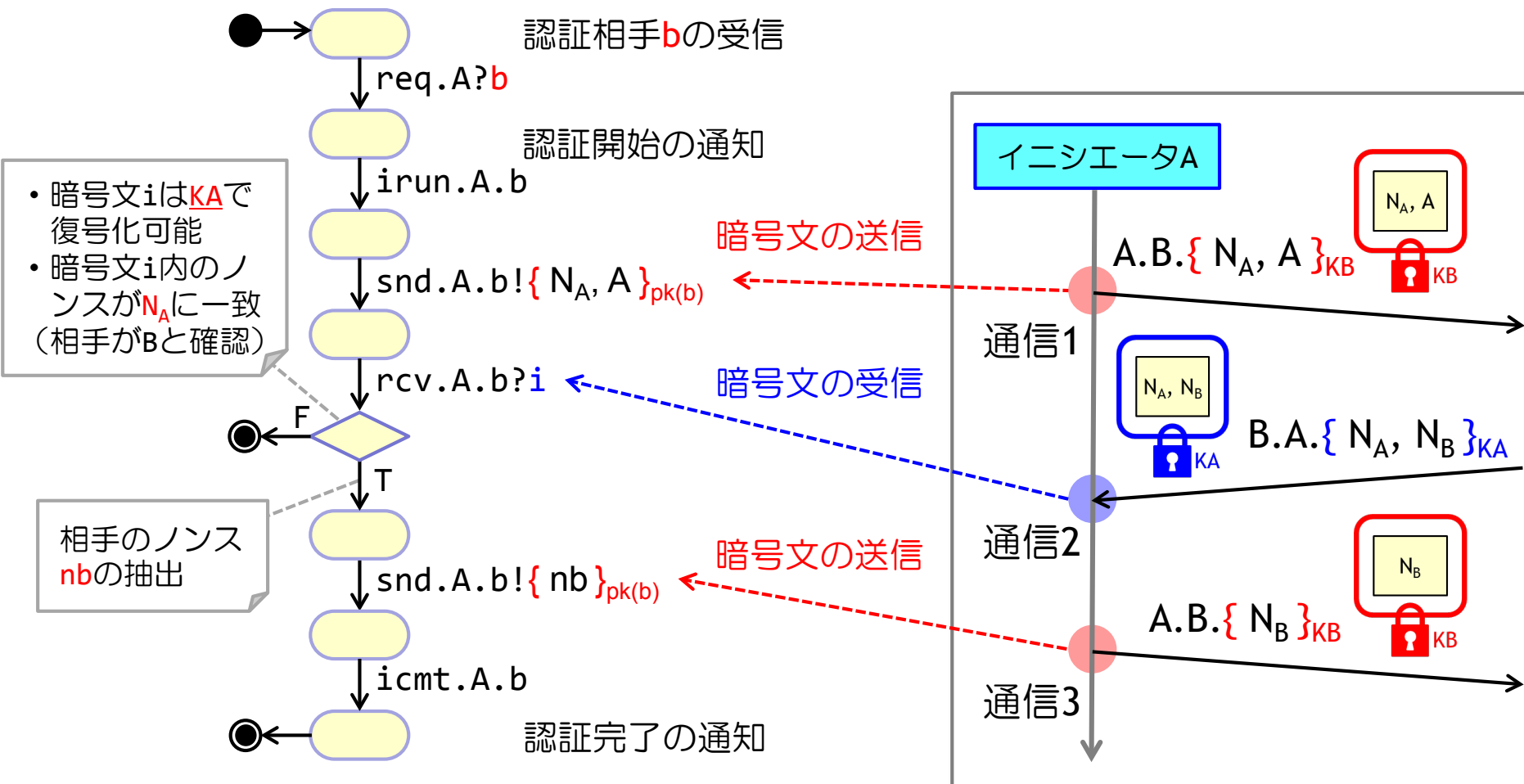
■ 攻撃者を含むNSPKプロトコルの検証用モデルの構造



動作：イニシエータAの振舞いモデル

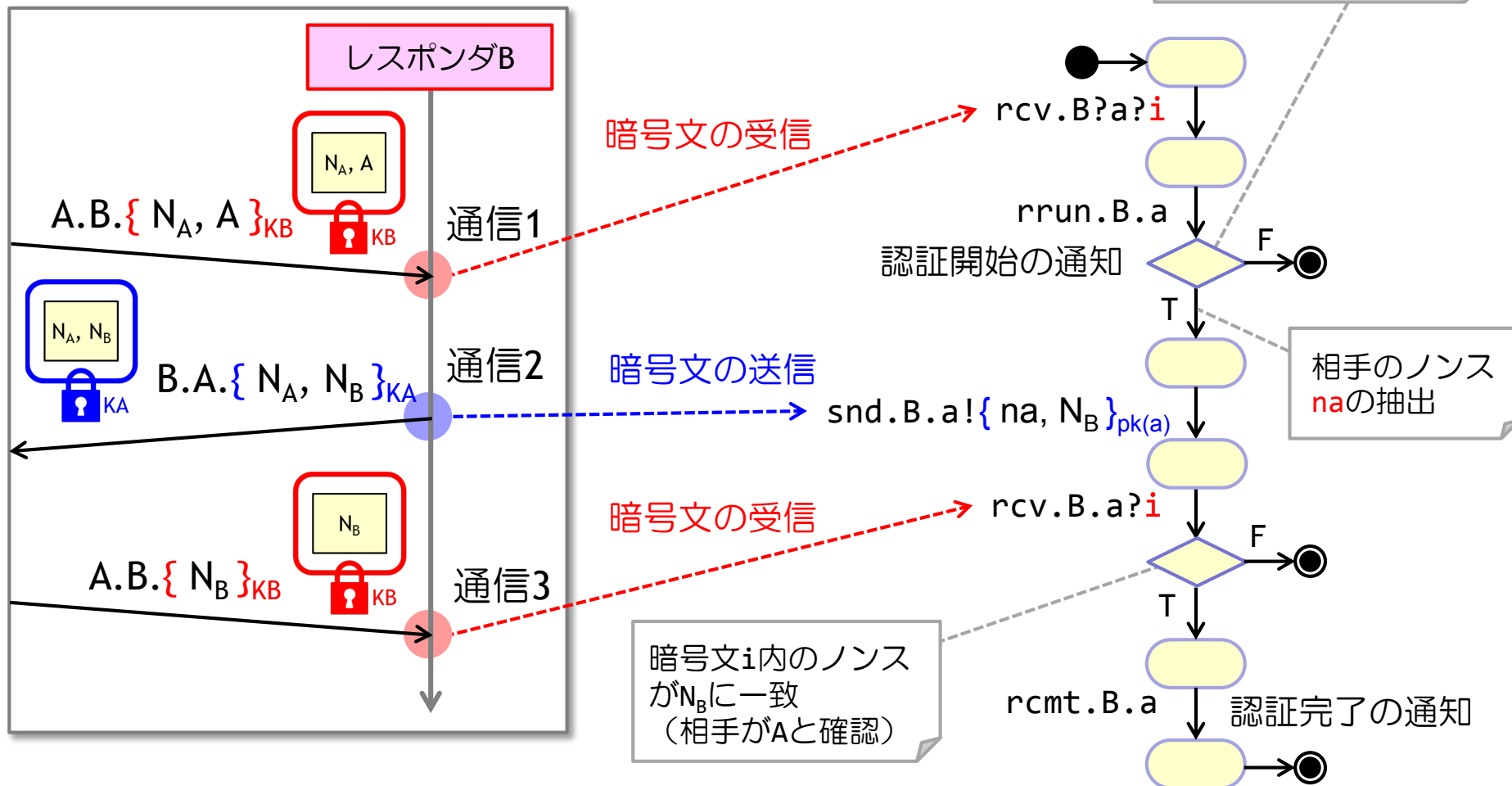
■ イニシエータAの通信手順（送信と受信の順番）

$pk(a)$: ユーザaの公開鍵



動作：レスポンドBの振舞いモデル

■ レスポンドBの通信手順（送信と受信の順番）



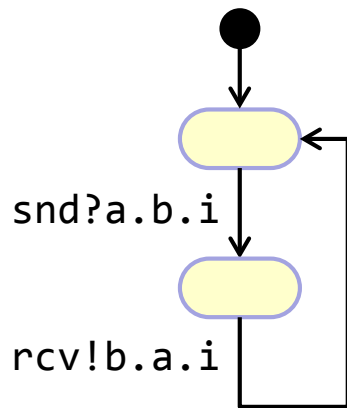
- ・ 暗号文 i は KB で復号化可能
- ・ 送信元と暗号文 i 内の送信者名が一致 (送信元詐称対策)

相手のノンズ na の抽出

暗号文 i 内のノンズが N_B に一致 (相手がAと確認)

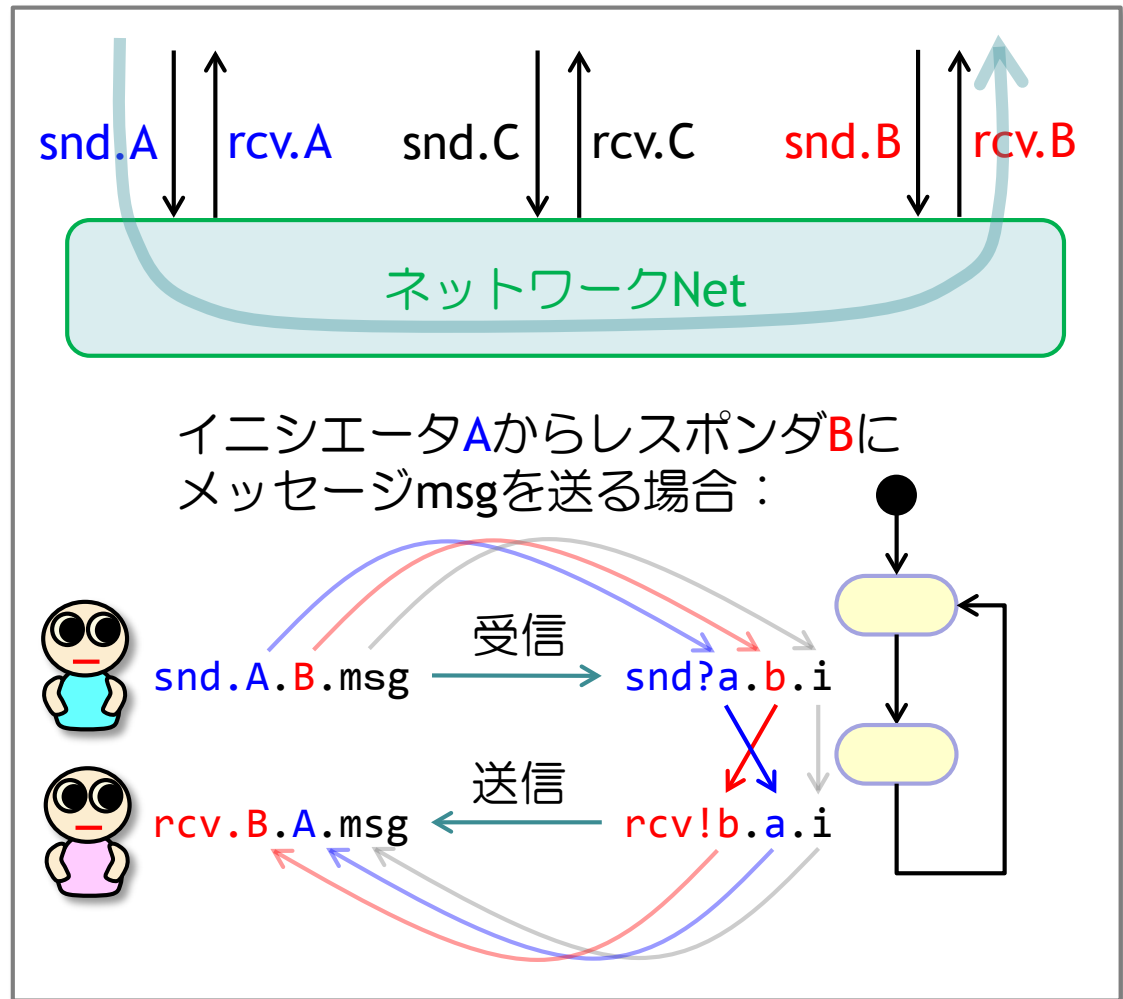
動作：通信ネットワークNETのモデル

■ 通信ネットワークの振舞い（データ送受信）



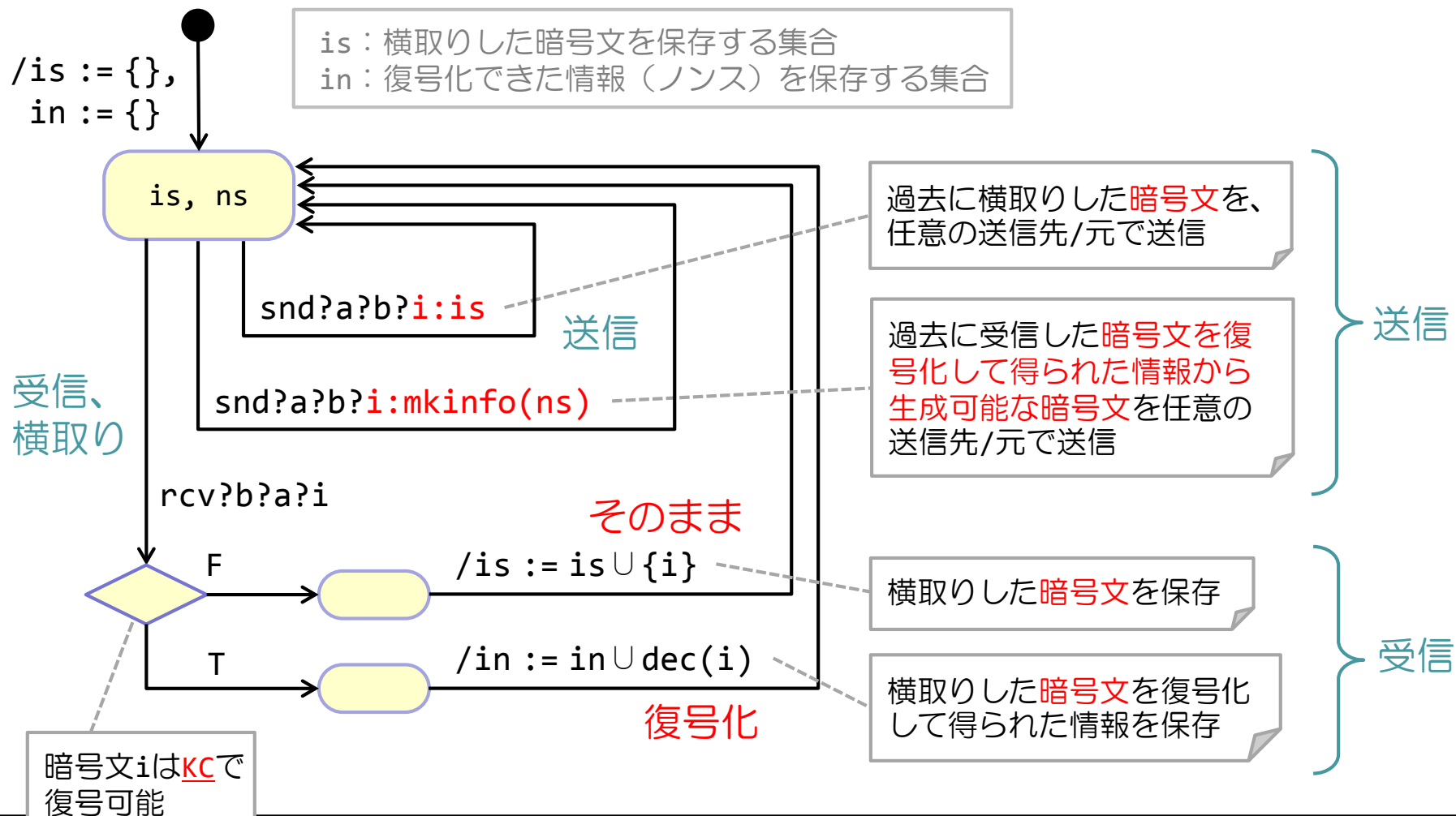
ネットワークに送信されたデータを受信し、送信先の受信用チャンネルに転送する

a : 送信元
b : 送信先
i : データ



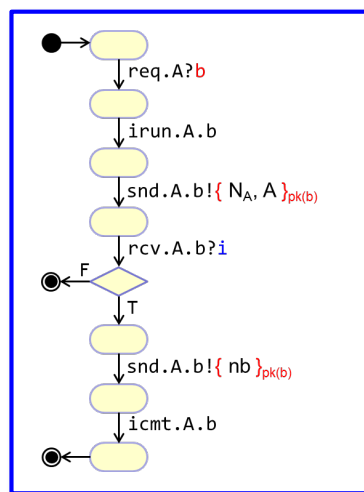
動作：攻撃者Cの振舞いモデル

■ 攻撃者Cの振舞い（受信/横取りした情報の保存、送信元を詐称した送信）

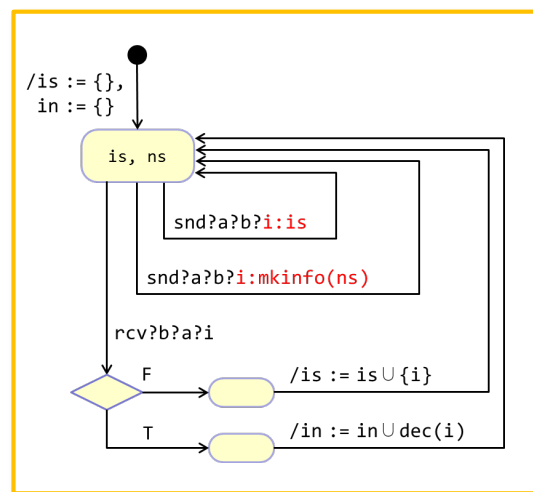


合成：NSPKプロトコル全体のモデル

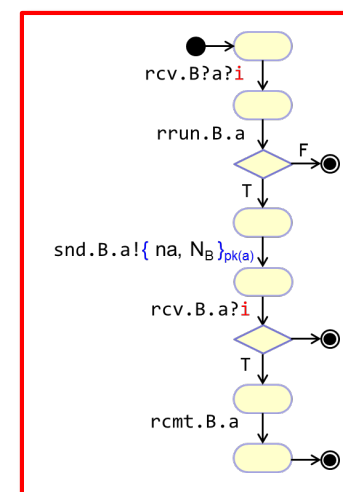
■ イニシエータA、レスポндаB、攻撃者C、ネットワークの並行合成



イニシエータA

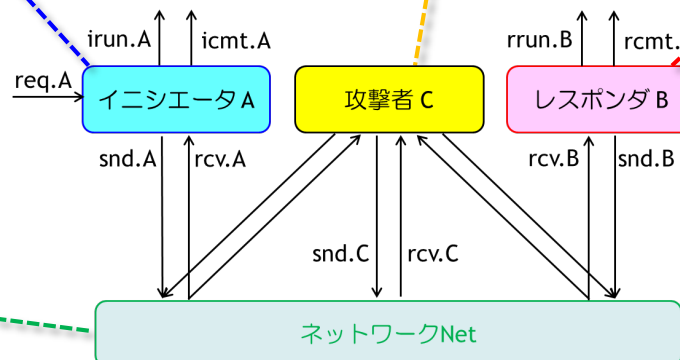
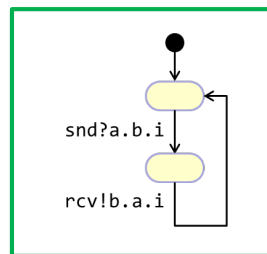


攻撃者C



レスポндаB

ネットワークNET



このモデルを用いて
なりすましを探索

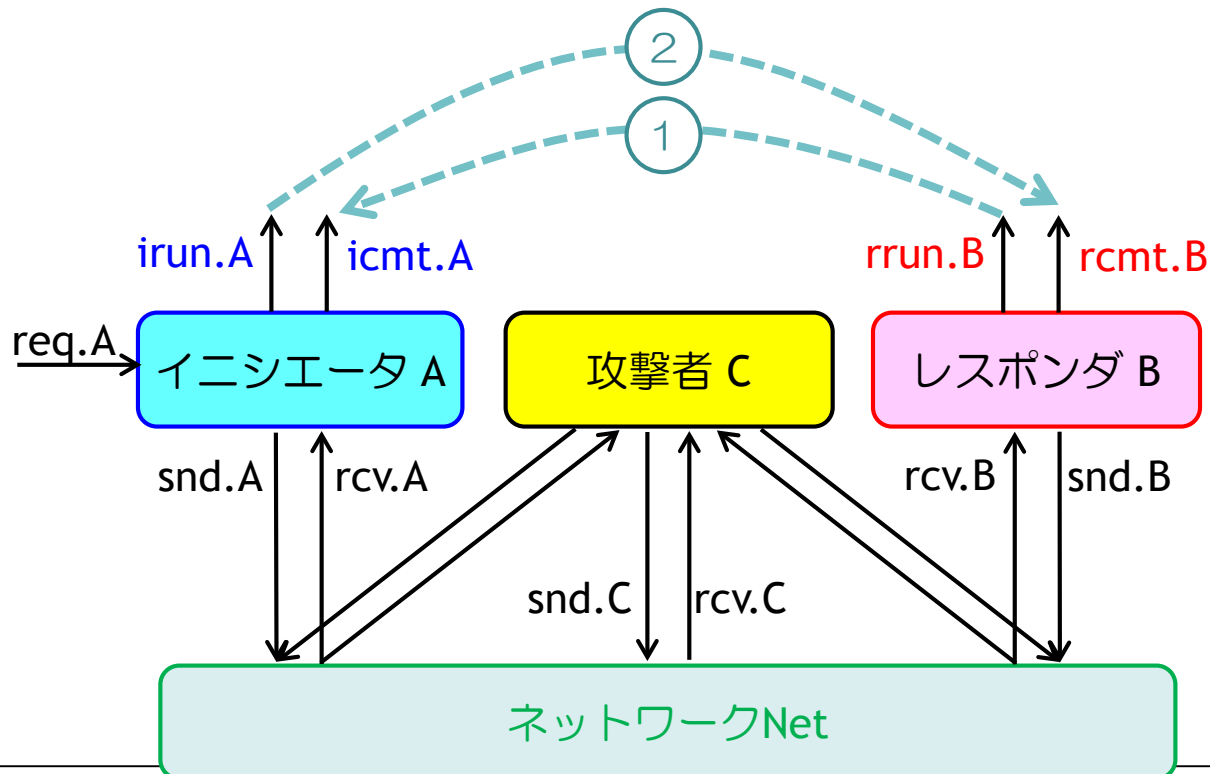
NSPKプロトコルの検証

- NSPKプロトコルの性質
- 検証手順と結果
- 反例トレースの分析（デバッグ）
- NSPKプロトコルの修正と検証

性質：NSPKプロトコルの性質

■ NSPKプロトコルが満たすべき性質：

- 性質①： $\text{rrun.B.A} \rightarrow \text{icmt.A.B}$
(AがBを通信相手と認証完了する前に、BがAの認証手続きを開始している)
- 性質②： $\text{irun.A.B} \rightarrow \text{rcmt.B.A}$
(BがAを通信相手と認証完了する前に、AがBの認証手続きを開始している)

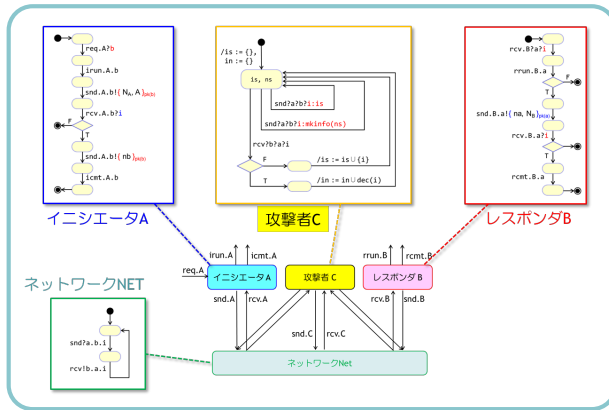


方法：NSPKプロトコルの検証手順

■ NSPKプロトコル全体のモデルの振舞いの検証手順：

1. 並行プロセスの仕様記述言語CSP *1 でNSPKプロトコルを記述する
2. CSP記述の検査ツールFDR *2（モデル検査器）で網羅的に振舞いを検査する

NSPKプロトコル全体のモデル System



CSP記述（一部）

```
-- Initiator
UsrA = A1
A1 = req.A?b -> irun.A.b -> A2(b)
A2(b) = snd.A.b!Enc.pk(b).D1.(N.A).A -> A3(b)
A3(b) = rcv.A.b?i:Info2 -> A4(b,dec(A,i))
A4(b,d) = if (nnc1(d) == (N.A))
then A5(b,nnc2(d)) else STOP
A5(b,nb) = snd.A.b!Enc.pk(b).D3.nb -> A6(b)
A6(b) = icmt.A.b -> SKIP

-- Responder
UsrB = B1
B1 = rcv.B.a?i:Info1 -> rrun.B.a
-> B2(a,dec(B,i))
B2(a,d) = if (usr(d) == a) then B3(a,nnc1(d))
else STOP
B3(a,na) = snd.B.a!Enc.pk(a).D2.na.(N.B) -> B4(a)
B4(a) = rcv.B.a?i:Info3 -> B5(a,dec(B,i))
B5(a,d) = if (nnc1(d) == (N.B)) then B6(a)
else STOP
B6(a) = rcmt.B.a -> SKIP

-- Intruder
UsrC = C1({},{})
C1(is, ns)
= rcv.B.a?i:UAB?i -> C2(is, ns, i)
[] snd.A.b?b:UAB?i:is -> C1(is, ns)
[] snd.A.b?b:UAB?i:mkinfos(ns) -> C1(is, ns)
```

モデル検査器FDR



満たすべき性質 **Prop1, Prop2**

性質①： **rrun.B.A** -> **icmt.A.B**
 性質②： **irun.A.B** -> **rcmt.B.A**

〔トレースの
包含関係〕 $\cap$ ← 自動判定

```
Prop1 = rrun.B.A -> icmt.A.B -> STOP
chsl = diff(Events, {rrun.B.A, icmt.A.B})
assert Prop1 [T= System W chsl]

Prop2 = irun.A.B -> rcmt.B.A -> STOP
chsl2 = diff(Events, {irun.A.B, rcmt.B.A})
assert Prop2 [T= System W chsl2]
```

NSPKプロトコル全体は
性質①②を満たすか？

*1 **CSP**（Communicating Sequential Processes）： C. A. R.Hoare, オクスフォード大学, 1978

*2 **FDR**（Failures-Divergences Refinement）： オクスフォード大学, 学術/研究目的でのみ無料で利用可能

表明：NSPKプロトコルの検証式

- NSPKプロトコルが性質①「 $\text{rrun.B.A} \rightarrow \text{icmt.A.B}$ 」満たすとは？

表明： $\text{traces}(\text{Prop1}) \supseteq \text{traces}(\text{System} \setminus \text{chs1})$

性質①のCSP記述：

$\text{Prop1} = \text{rrun.B.A} \rightarrow \text{icmt.A.B} \rightarrow \text{STOP}$

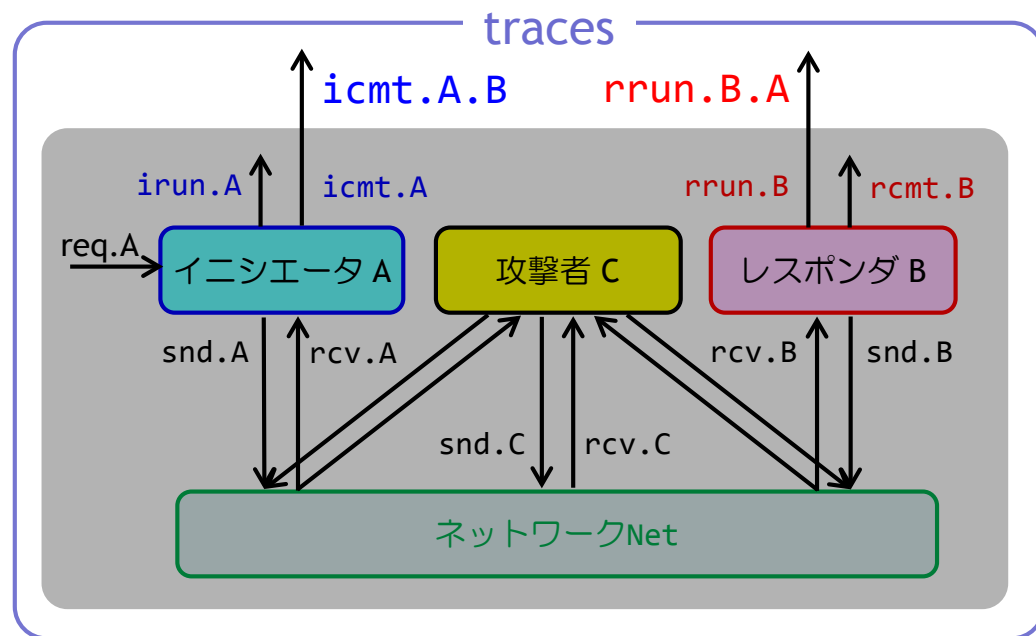
$\text{rrun.B.A}, \text{icmt.A.B}$ 以外の全ての事象（通信、通知、入力）を隠す

Prop1のトレースの集合

traces

$\langle \rangle$,
 $\langle \text{rrun.B.A} \rangle$,
 $\langle \text{rrun.B.A}, \text{icmt.A.B} \rangle$

?
 $\supseteq$



性質②の検証も同様

結果：NSPKプロトコルの検証結果

■ NSPKプロトコルのモデルの振舞いの検証結果：

- 性質①「**r**run.B.A -> **i**cmt.A.B」は成り立つ (Passed)
- 性質②「**i**run.A.B -> **r**cmt.B.A」は成り立たない (Failed)

検査結果

C:/Users/isobe/home/aist/2015/030-研究 (CSP) /020-信学会講演 (NSPK) /050-原稿/csp/Needham-Schroeder-prop2.csp

File Help

Welcome to FDR 3.3.1 (8d0597a2383fb17dd80b21c5aa03e8273cfbd7eb)
Type :help for help
Needham-Schroeder-prop2.csp> :statistics 0
Status: Passed
Compiled in 64.14s
Checked in 0.24s
States: Visited 15,183 over 19 plys at 62,225 states/second (268 minutes/gigastate)
Transitions: Visited 33,496 at 137,279 transitions/second
Total Storage: 134 MB cache
Requires 28 bytes per state
Needham-Schroeder-prop2.csp

• 状態数：15,183
• 遷移数：137,279
• 検査時間：約1分
(CPU 5Y70, 1.3GHz)

Assertions

Prop1 [T= System \ chs1] ?
Finished: Passed

Prop2 [T= System \ chs2] ?
Finished: Failed

Debug

Tasks

Generating machine constraints
Finished: Visited 1 processes

Checking refinement
Finished: Reconstructing counterexamples

No tasks running

Assertions Run All

Prop1 [T= System \ chs1] ?
Finished: Passed ①

Prop2 [T= System \ chs2] ?
Finished: Failed ②

Debug

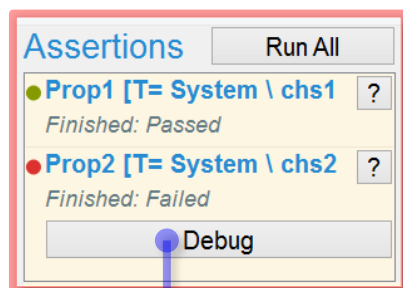
レスポンドBが通信相手はイニシエータAであると認証 (**r**cmt.B.A) した場合でも、
AはBに対して認証手続き (**i**run.A.B) を行っていない場合が存在する

ここをクリックすると、性質②を満たさない
通信手順（反例トレース）が表示される

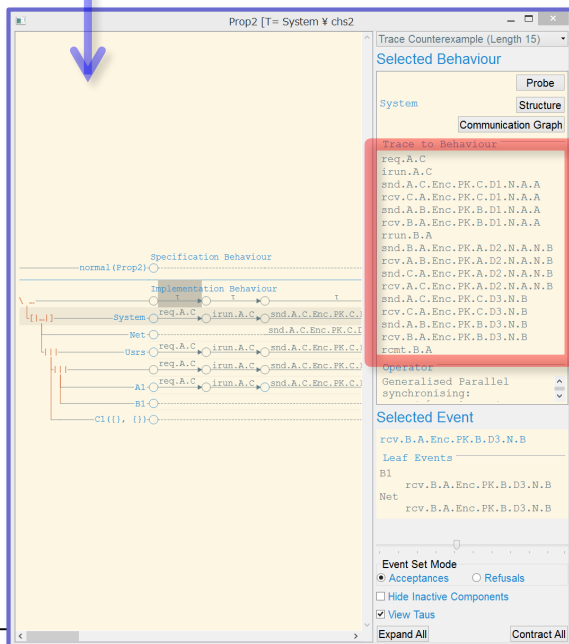
注： $P [T= Q] \Leftrightarrow \text{traces}(P) \supseteq \text{traces}(Q)$

反例：性質②の反例トレースの表示

- 性質②の検査結果が“Failed”になる原因を分析する



デバッガの起動



irun.A.B なしに rcmt.B.A が発生
するトレースの例（性質②の反例）

Trace to Behaviour

req.A.C

irun.A.C

注：A.C

snd.A.C.Enc.PK.C.D1.N.A.A

rcv.C.A.Enc.PK.C.D1.N.A.A

snd.A.B.Enc.PK.B.D1.N.A.A

rcv.B.A.Enc.PK.B.D1.N.A.A

rrun.B.A

snd.B.A.Enc.PK.A.D2.N.A.N.B

rcv.A.B.Enc.PK.A.D2.N.A.N.B

snd.C.A.Enc.PK.A.D2.N.A.N.B

rcv.A.C.Enc.PK.A.D2.N.A.N.B

and.A.C.Enc.PK.C.D3.N.B

rcv.C.A.Enc.PK.C.D3.N.B

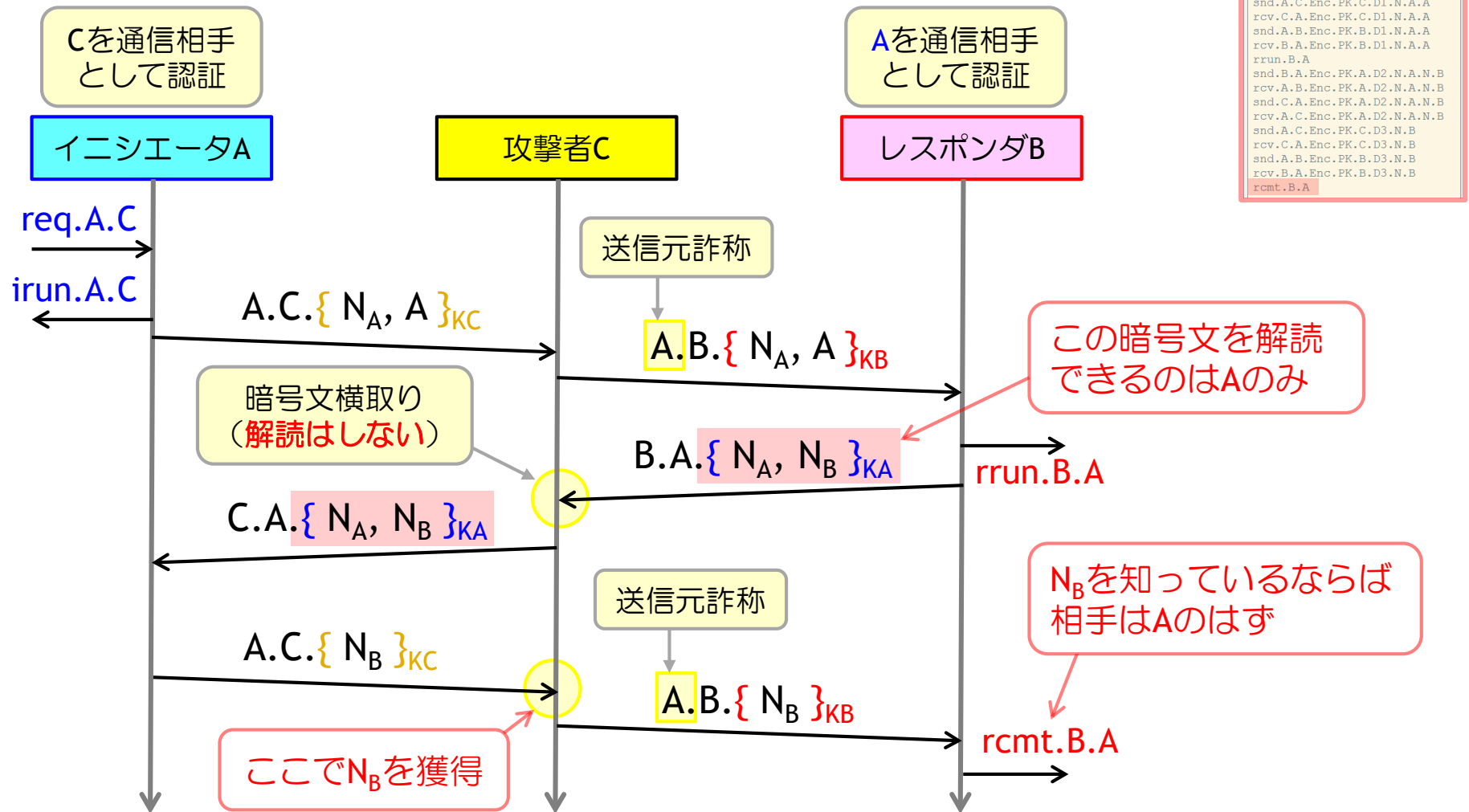
snd.A.B.Enc.PK.B.D3.N.B

rcv.B.A.Enc.PK.B.D3.N.B

rcmt.B.A

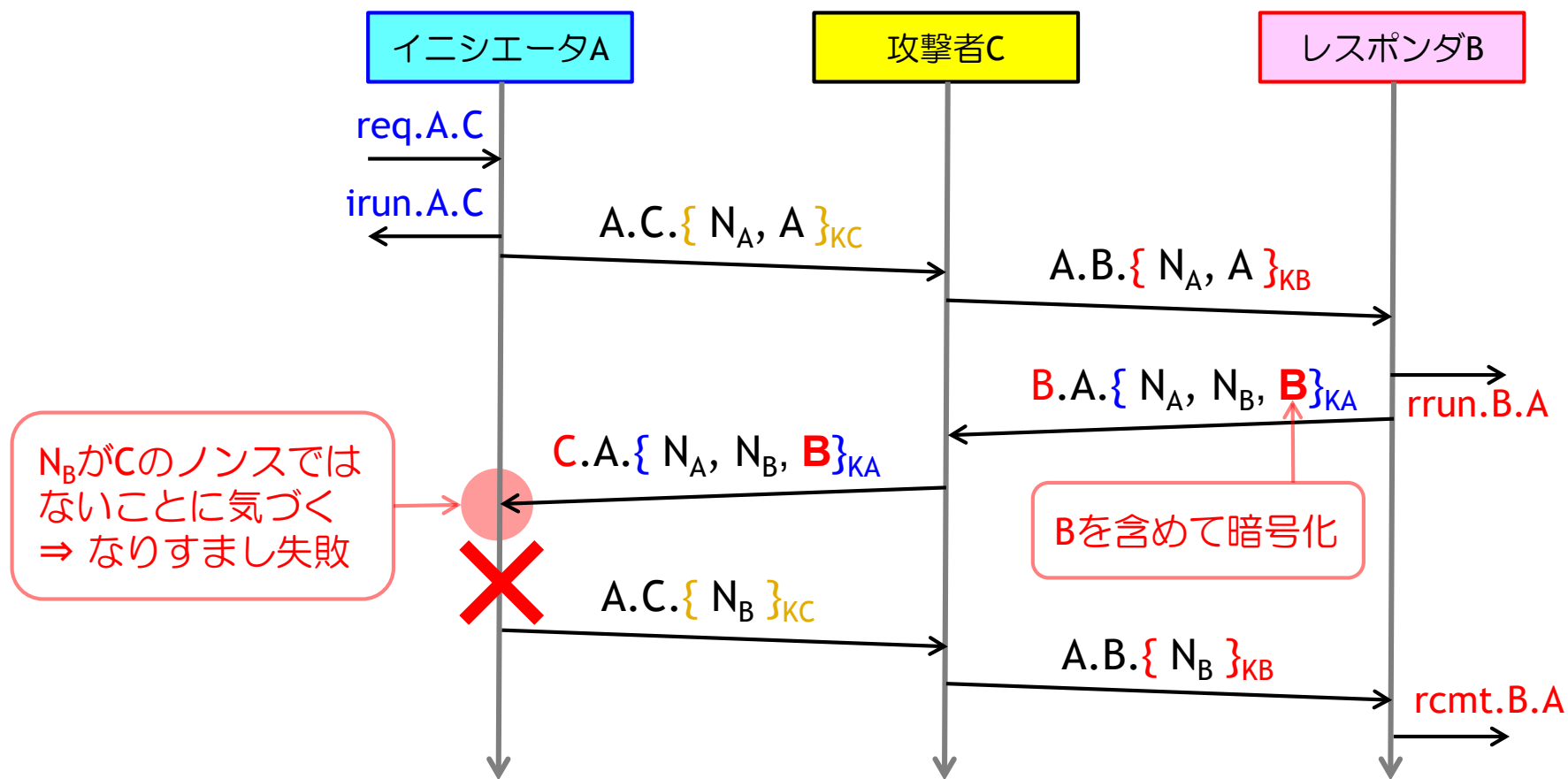
分析：性質②の反例トレースの分析

- 反例トレースを分析し、なりすまし手順の例を得る



修正：NSPKプロトコルの修正

- NSPKプロトコルの通信2で、レスポンドの名前を暗号文に含める
⇒ なりすましの回避



確認：修正版NSPKプロトコルの検証結果

■ 修正版NSPKプロトコルのモデルの振舞いの検証結果：

- 性質①「**r**run.B.A -> **i**cmt.A.B」は成り立つ (Passed)
- 性質②「**i**run.A.B -> **r**cmt.B.A」は成り立つ (Passed)

検査結果

```
File Help
Welcome to FDR 3.3.1 (8d0597a2383fb17dd80b21c5aa03e8273cfbd7eb)
Type :help for help
Needham-Schroeder-prop2-correct.csp> :statistics 0
Status: Passed
Compiled in 577.80s
Checked in 0.07s
States: Visited 7,152 over 16 plys at 106,746 states/second (156 minutes/gigastate)
Transitions: Visited 14,489 at 216,254 transitions/second
Total Storage: 134 MB cache, 0 B compressed, 0 B raw (0.00), 36 blocks
Requires 28 bytes per state (28.00 GB per gigastate)
Needham-Schroeder-prop2-correct.csp>
```

Assertions Run All

- Prop1 [T= System \ chs1] ?
Finished: Passed
- Prop2 [T= System \ chs2] ?
Finished: Passed

Tasks

- Children (4/4 finished)
- Generating machine constraints
Finished: Visited 1 processes
- Generating machine constraints
Finished: Visited 1 processes
- Checking refinement
Finished: Reconstructing counterexamples

No tasks running

- 状態数： 7,152
- 遷移数：14,489
- 検査時間：約10分
(CPU 5Y70, 1.3GHz)

Assertions Run All

- Prop1 [T= System \ chs1] ?
Finished: Passed
- Prop2 [T= System \ chs2] ?
Finished: Passed

FDRのファイル： <https://staff.aist.go.jp/y-isobe/CSP-NSPK/>

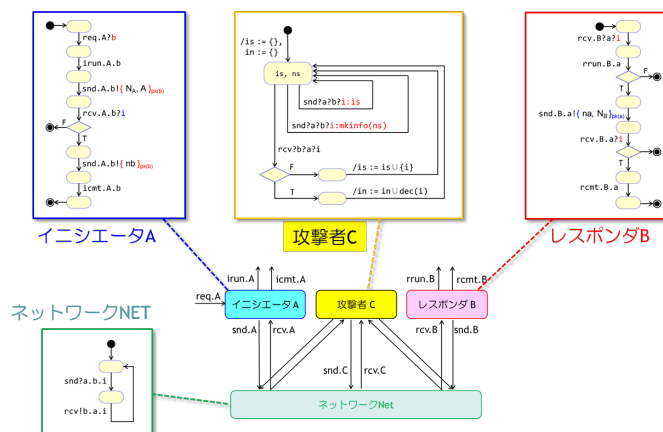
おわりに

- 暗号プロトコルの検証

まとめ：暗号プロトコルの検証

- NSPKプロトコルを例に、暗号プロトコルの基本的な検証方法を紹介した
 - NSPKプロトコル（攻撃者含む）の並行プロセスによるモデル化
 - 並行プロセスの仕様記述言語CSPによる形式化
 - CSP記述のモデル検査器FDRによる検証

並行プロセスモデル



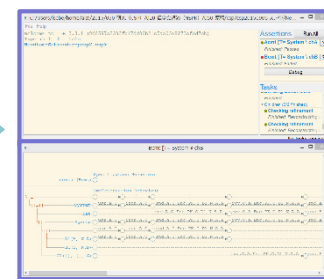
CSP記述

```
-- Initiator
UsrA = A1
A1 = req.A?b -> irun.A.b -> A2(b)
A2(b) = snd.A.b!Enc.pk(b).D1.(N.A).A -> A3(b)
A3(b) = rcv.A.b?i:Info2 -> A4(b,dec(A,i))
A4(b,d) = if (nnc1(d) == (N.A))
then A5(b,nnc2(d)) else STOP
A5(b,nb) = snd.A.b!Enc.pk(b).D3.nb -> A6(b)
A6(b) = icmt.A.b -> SKIP

-- Responder
UsrB = B1
B1 = rcv.B?a?i:Info1 -> rrun.B.a
-> B2(a,dec(B,i))
B2(a,d) = if (usr(d) == a) then B3(a,nnc1(d))
else STOP
B3(a,na) = snd.B.a!Enc.pk(a).D2.na.(N.B) -> B4(a)
B4(a) = rcv.B.a?i:Info3 -> B5(a,dec(B,i))
B5(a,d) = if (nnc1(d) == (N.B)) then B6(a)
else STOP
B6(a) = rcmt.B.a -> SKIP

-- Intruder
UsrC = C1({},{}),
C1(is, ns)
= rcv?b?a:UA?b?i -> C2(is, ns, i)
[] snd?b?b:UA?b?i:is -> C1(is, ns)
[] snd?b?b:UA?b?i:mkinfos(ns) -> C1(is, ns)
```

モデル検査器FDR



- モデル検査器：プロセスの振舞いの性質を網羅的に検査するツール

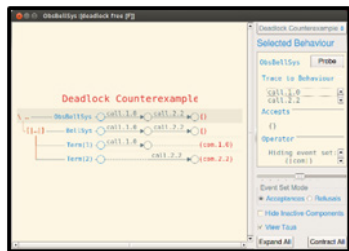
- 発生確率の低い不具合を発見（並行プロセスの不具合はテストでは発見困難）
- 不具合の原因特定に有効な情報（反例）を表示

付録：モデル検査器の例

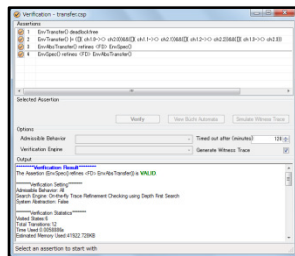
■ CSPに基づくモデル検査器の例：

- **FDR**：CSPの代表的なモデル検査器（オクスフォード大学、教育・研究目的ならば無料）
<https://www.cs.ox.ac.uk/projects/fdr/>
FDRの商用ライセンスはFormal Systems (Europe) Limited から購入できる
<http://www.fsel.com/index.html>
- **PAT**：CSPの高機能なモデル検査器（シンガポール大学、非営利研究目的ならば無料）
<http://www.comp.nus.edu.sg/~pat/>
PATの商用ライセンスはキャッツ株式会社から購入できる
http://www.zipc.com/product/pat_pro/
- **ProB**：FDR互換性のモデル検査器（デュッセルドルフ大学とサウサンプトン大学、EPL v1.0）
<http://www.stups.uni-duesseldorf.de/ProB/>
ProBの有償サポートはformalmind社から受けることができる
<http://www.formalmind.com/>
- **SyncStitch**：GUIが使いやすいモデル検査器（株式会社 PRINCIPIA、評価版あり）
<http://www.principia-m.com/jp/syncstitch.html>

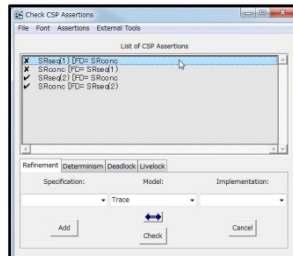
FDR



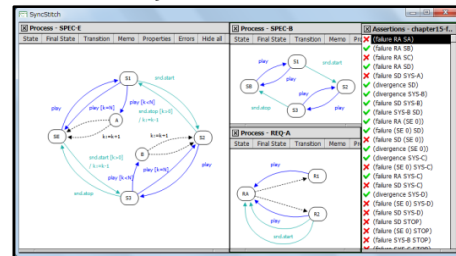
PAT



ProB



SyncStitch



■ CSPの教科書：

- C.A.R. Hoare, Communicating Sequential Processes, Prentice Hall, 1985.

(<http://www.usingcsp.com/cspbook.pdf> から無料ダウンロード可能)

CSPの提案者Hoare先生のCSPの最初の教科書です。この本の日本語訳がありますが、おそらく絶版です。
C.A.R. ホーア（著）, 吉田信博（訳） ホーアCSPモデルの理論, 丸善, 2002.

- A. W. Roscoe, The theory and practice of concurrency, Prentice Hall, 1998.

(<http://www.comlab.ox.ac.uk/people/bill.roscoe/publications/68b.pdf> からダウンロード可能)

Roscoe先生のCSPの教科書です。FDRの説明もあります。この本の改訂版が2010年に出版されています。
A. W. Roscoe, Understanding Concurrent Systems, Springer, 2010.

- S. Schneider, Concurrent and Real-time Systems: the CSP Approach, John Wiley & Sons, 1999.

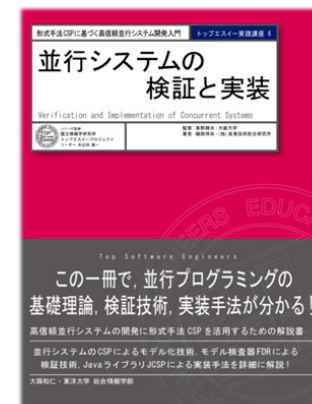
(<http://www.computing.surrey.ac.uk/personal/st/S.Schneider/books/CRTS.pdf> からダウンロード可能)

時間付CSP（Timed-CSP）の教科書です。時間制約をCSPのイベントに付けて検証することができます。

- 磯部祥尚, 並行システムの検証と実装

－ 形式手法CSPに基づく高信頼並行システム開発入門, 近代科学者, 2012.

CSP, FDR, JCSPを活用するための入門書です



本スライド：<https://staff.aist.go.jp/y-isobe/CSP-NSPK/>