評価ツールProVerifによる評価（NTTグループ）

日本電信電話株式会社
NTTコミュニケーション科学基礎研究所
櫻田英樹

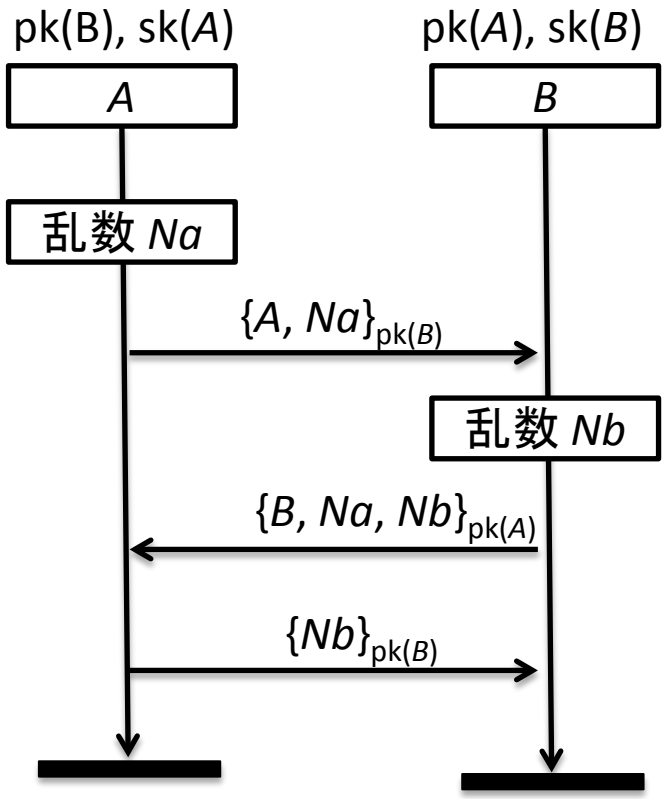
ProVerifツールの紹介

- 開発元：フランス国立情報学自動制御研究所(INRIA)
- 入力：プロトコル仕様、セキュリティプロパティをProVerifの記法で記述したもの
- 出力：「安全」/「安全でない」+攻撃例 /「評価不能」
- ツールの特徴
  - 様々なプロトコルの仕様を記述できる
    - 記述力の高いプロトコル仕様記述言語
    - 様々な暗号技術をユーザが定義できる（コミットメント、ブラインド署名なども）
  - 様々なセキュリティプロパティを評価できる
    - 秘匿性（例：鍵が漏れない、匿名性）
    - 認証（例：なりすましができない、リプレイができない）
    - オフライン辞書攻撃など
  - 攻撃者の能力・セッション数を制限せず評価できる（PAL4）

Proverif入力例（プロトコル記述部分、参加者*A*を抜粋）

入力例：

```
let processA =
                （一部省略）
    (* Message 1 *)
    new Na: nonce;
    out(c, encrypt((A, Na), pkB));
    (* Message 2 *)
    in(c, m2: bitstring);
    let (=B, =Na, Nb: nonce) = decrypt(m2, skA) in
    event beginA(A, B, Na, Nb);
    (* Message 3 *)
    out(c, encrypt(nonce_to_bitstring(Nb), pkB));
    (* Security *)
    get honest(=B) in
    event endA(A, B, Na, Nb);
    out(c, sencrypt(secretANa, Na));
    out(c, sencrypt(secretANb, Nb)).
```



様々なプロトコルを記述できるが記述量は多め（120行程度）

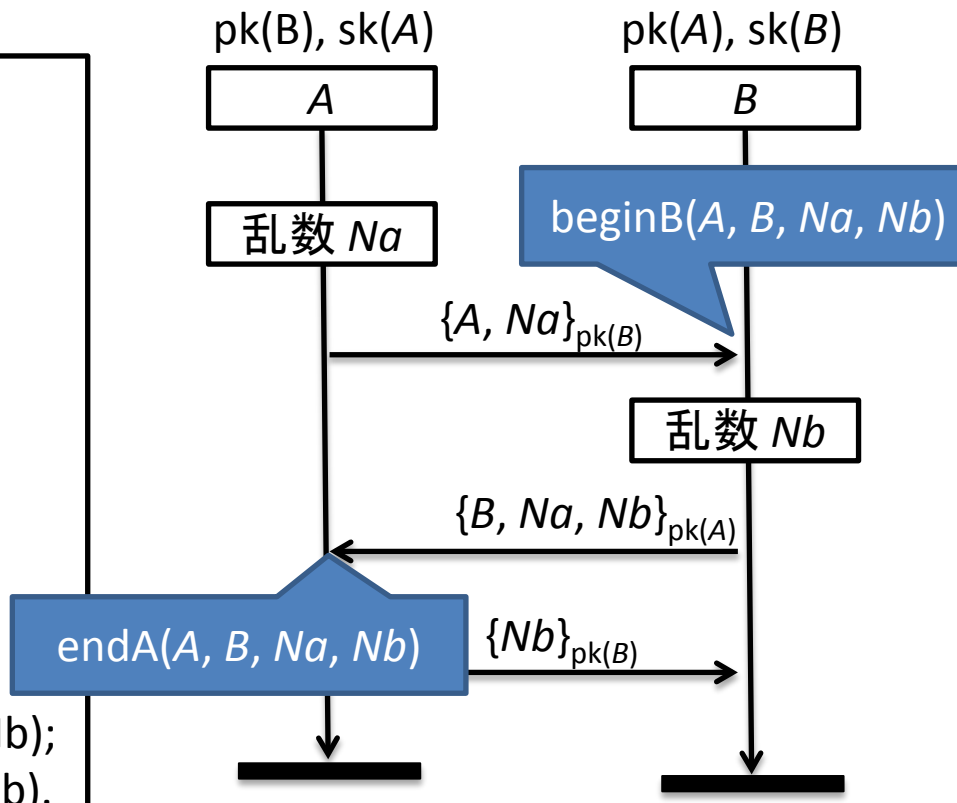
Proverif入力例（セキュリティプロパティ記述部分）

入力例：

```
(* AによるBの認証 *)
query x: host, y: host, nx: nonce, ny: nonce;
    inj-event(endA(x, y, nx, ny)) ==>
    inj-event(beginB(x, y, nx, ny)).

(* BによるAの認証 *)
query x: host, y: host, nx: nonce, ny: nonce;
    inj-event(endB(x, y, nx, ny)) ==>
    inj-event(beginA(x, y, nx, ny)).

(* 使用した乱数の秘匿性 *)
query   attacker(secretANa); attacker(secretANb);
        attacker(secretBNa); attacker(secretBNb).
```



「交換した乱数（鍵）が両者で一致する」という性質も評価可能

出力例（安全でない場合）

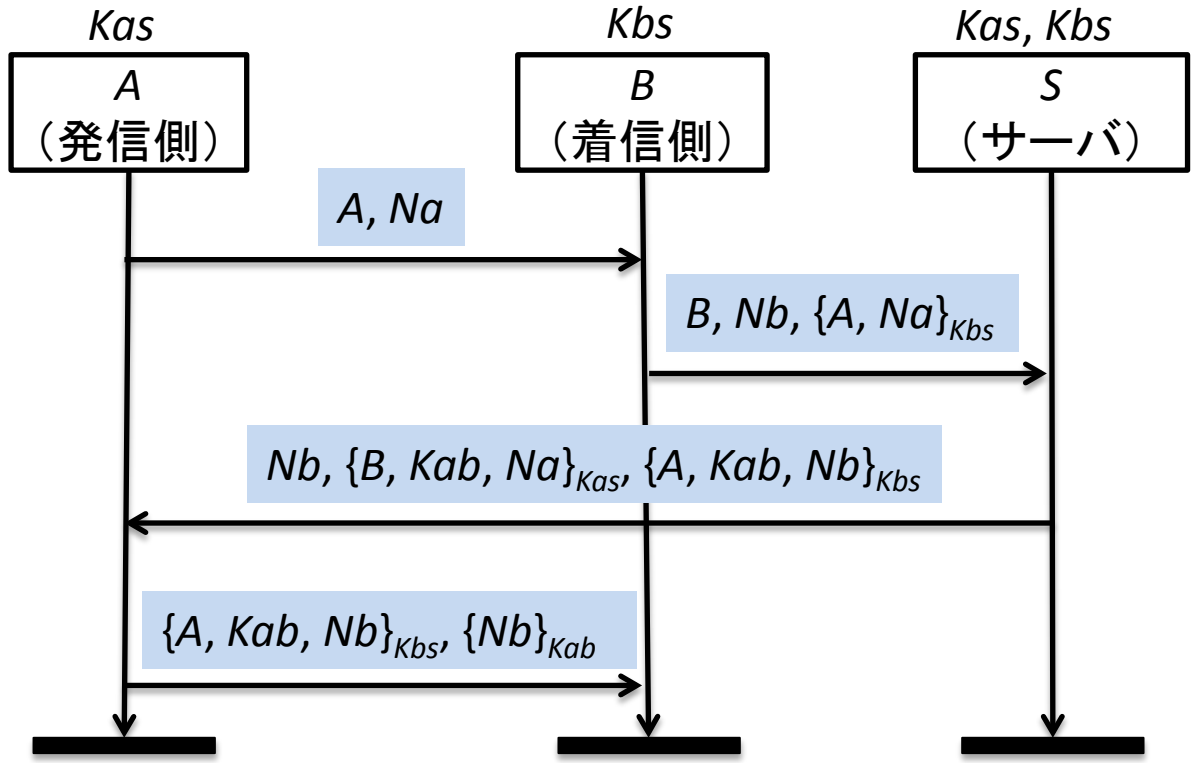
- 安全性の評価結果を出力
- 安全でない場合は、具体的な攻撃の例も
- 評価時間は数秒～１分程度であることが多いが、
  １日以上待っても結果が得られないこともある

出力例（抜粋）:

```
in(c, encrypt((hostB,a_49198),pk(skA_49203))
new Nb_17 creating Nb_49205 at {35} in copy
event(beginB(hostB,hostA,a_49198,Nb_49205)) at {36} i
out(c, encrypt((a_49198,Nb_49205),pk(skB_49204))) at  copy a_49197
in(c, (a_49198,pk(a_49200),a_49199)) at {45} in copy a_49202
insert pke_keys(a_49198,pk(a_49200),a_49199) at {48} in copy a_49202
get pke_keys(a_49198,pk(a_49200),a_49199) at {31} in copy a_49201
get pke_keys(hostB,pk(skB_49204),skB_49204) at {32} in copy a_49201
in(c, encrypt((a_49198,Nb_49205),pk(skB_49204))) at {33} in copy a_49201
new Nb_17 creating Nb_49206 at {35} in copy a_49201
event(beginB(a_49198,hostB,Nb_49205,Nb_49206)) at {36} in copy a_49201
out(c, encrypt((Nb_49205,Nb_49206),pk(a_49200))) at {37} in copy a_49201
in(c, encrypt((hostA,Nb_49205),pk(skA_49203))) at {38} in copy a_49197
get honest(hostB) at {40} in copy a_49197
event(endB(hostB,hostA,a_49198,Nb_49205)) at {41} in copy a_49197
The event endB(hostB,hostA,a_49198,Nb_49205) is executed in session a_49197.
A trace has been found.
RESULT inj-event(endB(x_32995,y_32996,nx_32997,ny_32998)) ==>
        inj-event(beginA(x_32995,y_32996,nx_32997,ny_32998)) is false.
```

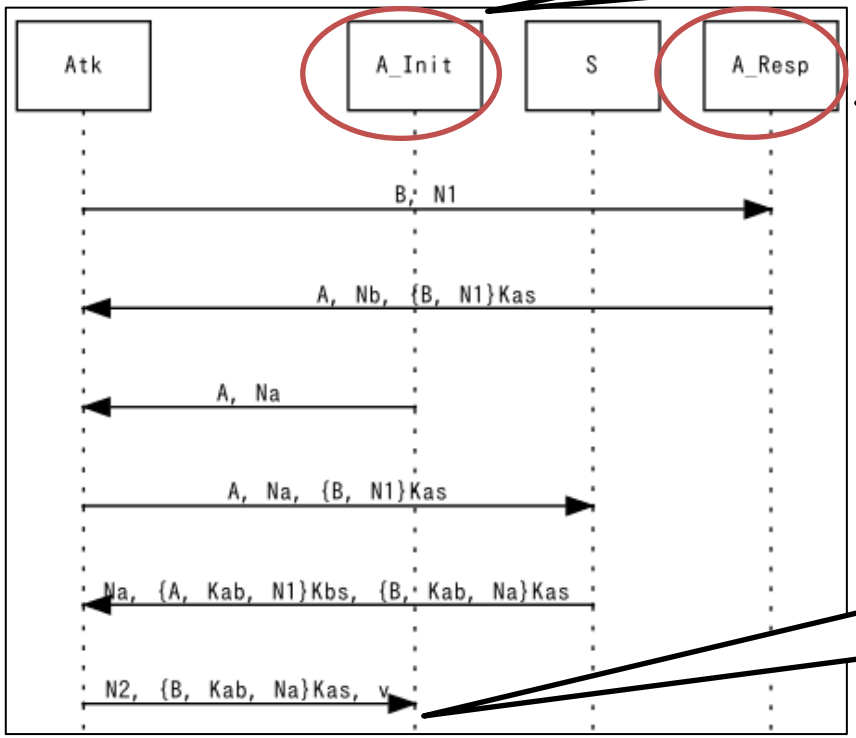
評価実験：BAN-Yahalomプロトコル

- 秘密鍵暗号利用、サーバを介した相互認証プロトコル
- ProVerifの入力ファイル：全部で120行
- 評価に要した時間：0.8秒（Core i7 2GHz, メモリ2GB）



BAN-Yahalomの評価：*A*による*B*の認証

評価結果：安全でない



AはBを相手としてプロトコル（発信側）を実行

Aは受信側としてもプロトコルを実行

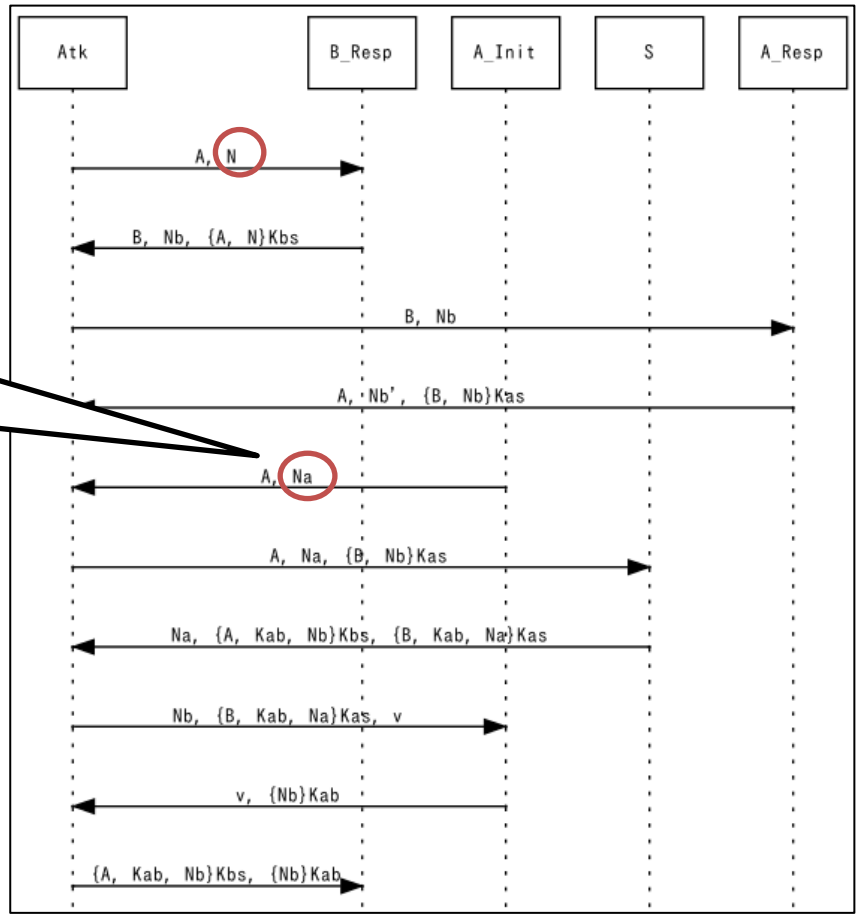
Bが存在しないにもかかわらず、実行を完了

※ProVerifが出力した攻撃を図示

BAN-Yahalomの評価：*B*による*A*の認証

評価結果：安全でない

Bが実行を完了するとき
Aが確かに存在するが、
パラメータが一致しない



※ProVerifが出力した攻撃を図示

ProVerifはかなり複雑な攻撃も
正確に発見可能

ProVerifに関する知見

- プロトコル仕様
  - 他のツール（Scytherなど）に比べて記述能力が高い一方、記述の量が多いため記述ミスを生じやすく発見しづらい。ミスの発見のための工夫が必要。
  - セキュリティプロパティを詳細に記述できる一方、評価者によってバラつきが生じやすい。
- モデル記述
  - 様々な暗号プリミティブの性質を記述して使えるが、性質によっては評価が停止しなくなる。例えばDiffie-Hellman鍵共有で用いられる等式

$$(w^x)^y=(w^y)^x$$

などをそのまま使うと停止しないため、工夫が必要。

ProVerifに関する知見（抜粋）

- 評価の実行および結果の解釈
  - 「安全である」という結果が出力された場合でも、入力したプロトコル記述にミスがあれば、結果は信用できない。「動かないプロトコルは安全」という場合があるので注意が必要。
  - いくら待っても評価が停止しない場合の対策にノウハウがある。ProVerifの実行オプションの指定や、プロトコルの記述の仕方など。

結論

- ProVerifはプロトコル評価のために非常に有用
  - 様々なプロトコルとその安全性を記述・評価可能
  - 使いこなすために多少工夫が必要な場面も
- ProVerifに限らず、入力の記述の正しさは課題
  - 評価結果の検証や複数の評価を併用・比較など、評価手順のマニュアル化により改善
- 評価実験を通じて、ツールの使い方に関する知見を整備することができ、さらにツールによらない一般的な課題を発見し、解決の方向性を見出すことに成功した