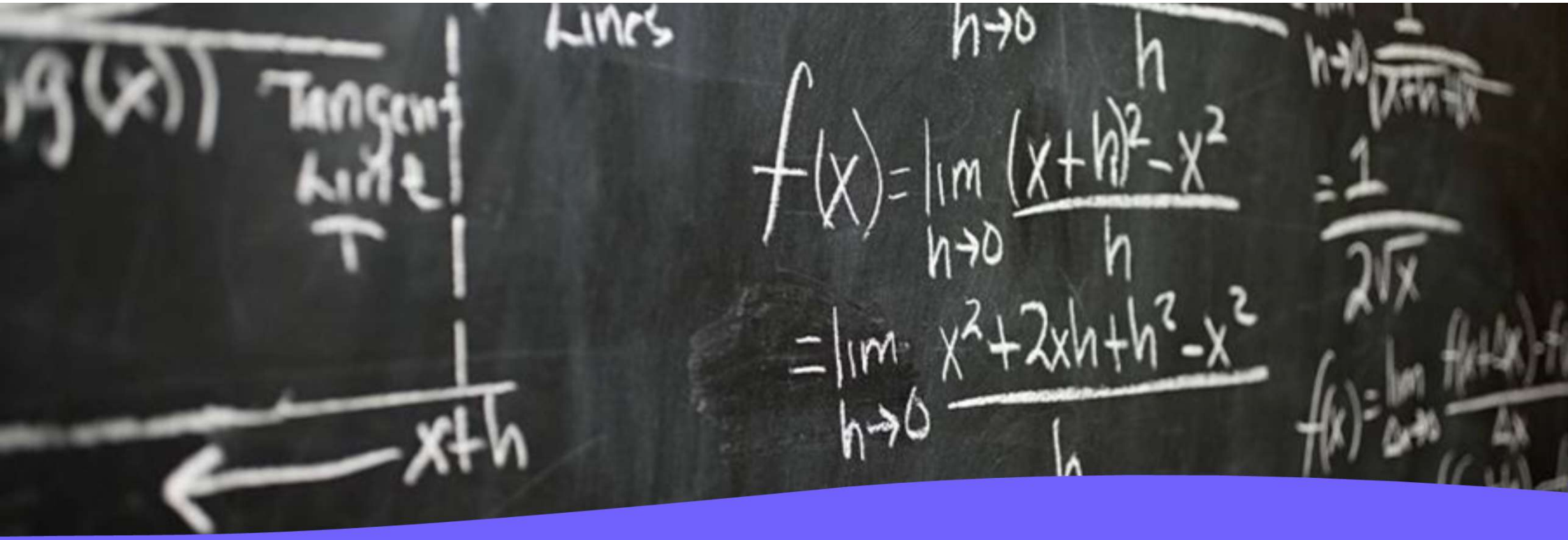


ProVerif と Tamarin-prover の観測等価性の違い

三重野 武彦　岡崎 裕之　布田 裕一　荒井 研一

Contents

1.発表の概要

自由度の高い、
形式的安全性検証を実現する

安全性検証に使用するツールの
優位性や使い分けなどを探っている

トレース特性や観測等価性

どのような違いがあるか

本講演では,DDHを使用して検証

Tamarin-prover の概要

A multiset rewriting rule is described as follows.

1 *rule Name:*

2 *[Pre(x)] – [Action(x)] –>[Post(x)]*

Lemma Annotations

lemma Name [Annotation1,Annotation2]:

e.g.)

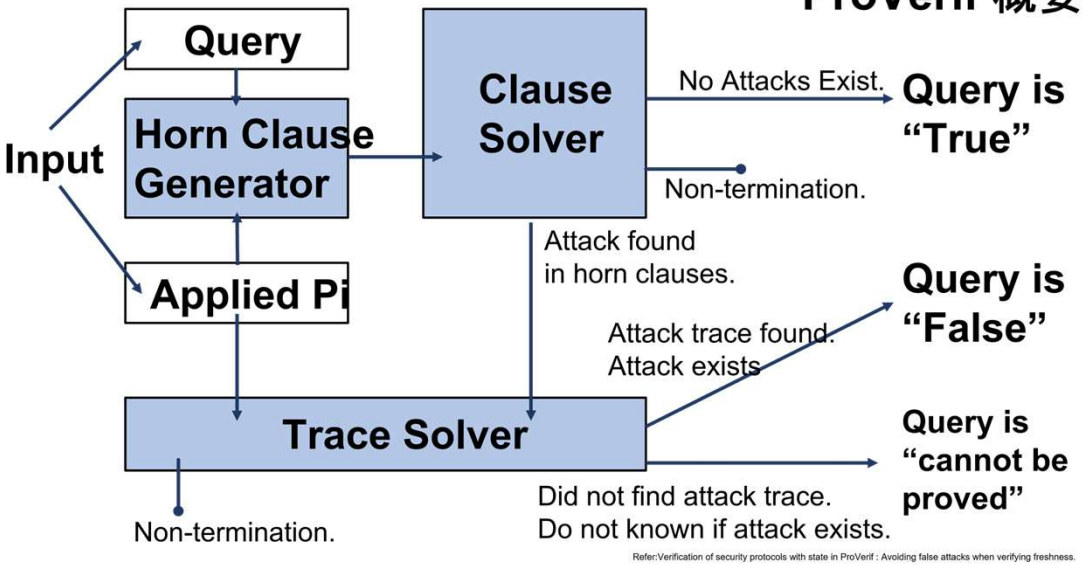
lemma nonce_secret:

"All m #i #j. Secret(m) @i & K(m) @j ==> F"

Tamarin-prover query handling

Secret(session1, key)@i&Key(key)@j.

ProVerif 概要



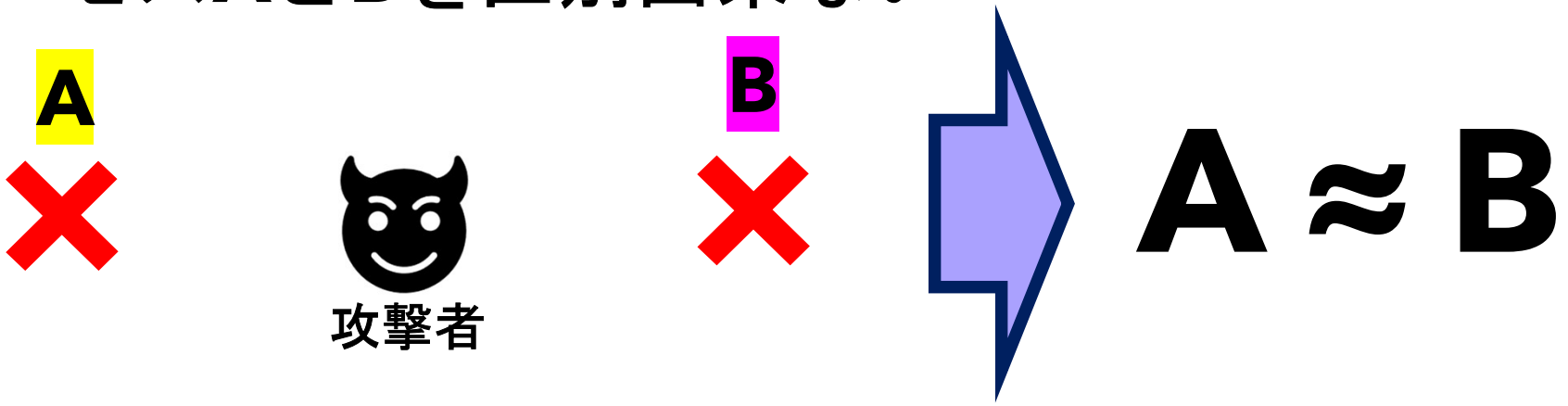
Refer:Verification of security protocols with state in ProVerif : Avoiding false attacks when verifying freshness.

2. 観測等価性とは

プロセス計算に確率的な側面を導入し,
プロセス計算の中で,計算論的な識別不可能性を
観測等価性として表現した.

形式モデル:
識別不可能性の概念: 一般的に観測等価性と表現　[参考文献.A]

プロセスAとBを区別出来ない



[参考文献.A]
JOHN C. MITCHELL, AJITH RAMANATHAN, ANDRE SCEDROV, AND VANESSA TEAGUE.
“A PROBABILISTIC POLYNOMIAL-TIME PROCESS CALCULUS FOR THE ANALYSIS OF CRYPTOGRAPHIC PROTOCOLS”,
Theoretical Computer Science, Volume 353, Issues 1–3, 14 March 2006, Pages 118-164.

関連研究

ProVerif

1. Proving More Observational Equivalences with ProVerif

Vincent Cheval and Bruno Blanchet,
In Proceedings of the 2nd International Conference on Principles of Security and Trust (POST'13).

Tamarin-prover

2. Automated Symbolic Proofs of Observational Equivalence

D.A. Basin, J. Dreier, and R.Sasse,
In 22nd Conference on Computer and Communications Security (CCS'15),
pages 1144–1155. ACM Press, 2015.

比較には触れていない

Other

ProVerifを使用した観測等価性

・投票プロトコル自動検証（Kremer, Ryan ’05） , etc

3. DDH(Diffie-Hellman) 仮定

生成器gを持つ素数qの巡回群Gが与えられたとき,
(g^a, g^b, g^{ab})は(g^a, g^b, g^c)と計算上区別がつかない.
ここで a, b, c は Z_q^*からのランダム要素である.

DDH問題(判定DH)

a, b, c∈Z をランダムに選択

(g, g^a, g^b, g^{ab}) と (g, g^a, g^b, g^c)

を識別する問題

(p, g, g^a, g^b, g^c, Z)

DDH問題を解く
アルゴリズム

0 or 1

検証結果

観測等価性は既に証明されている

Tamarin-prover

ProVerif

何故この結果？

Biprocess 0 (the initial process)

{1} new a: exponent;
{2} new b: exponent;
{3} new c: exponent;
{4} out(d, (exp(g,a),exp(g,b),choice[exp(exp(g,a),b),exp(g,c)]))

4. Bi-systems の概要

[参考文献.B]

観測等価性の証明を自動化 : *Bi-system*

Bi processes

diff[-, -] : 多項式書き換えシステム

2つの実行を「同時に実行する]プロセスを入力し、攻撃者に観測させる
Ex. P(A,B)とP(B,A) → P(diff[A, B], diff[B, A])

S : bi-system

Replace:A

diff[A, B] → L(S)

Replace:B

diff[A, B] → R(S)

L(S) and R(S)
観測等価

[参考文献.B]
Blanchet, B., Abadi, M., Fournet, C.: Automated verification of selected equivalences for security protocols. Journal of Logic and Algebraic Programming 75(1), 3–51 (2008)

5. 検証概要

1. ProVerif
2. Tamarin-prover

項のみが異なるプロセス間の観測等価性: 検証する operator

#1. ProVerif

choice[<term>,<term>] ※1

#2. Tamarin-prover

diff (x, y) ※2

[補足事項]

※1：プロセスに choice を含む場合,「query」,「noninterf」,「weaksecret」は使用できない

※2：ユーザが指定できる「lemma」が無い

DDH-Code

ProVerif

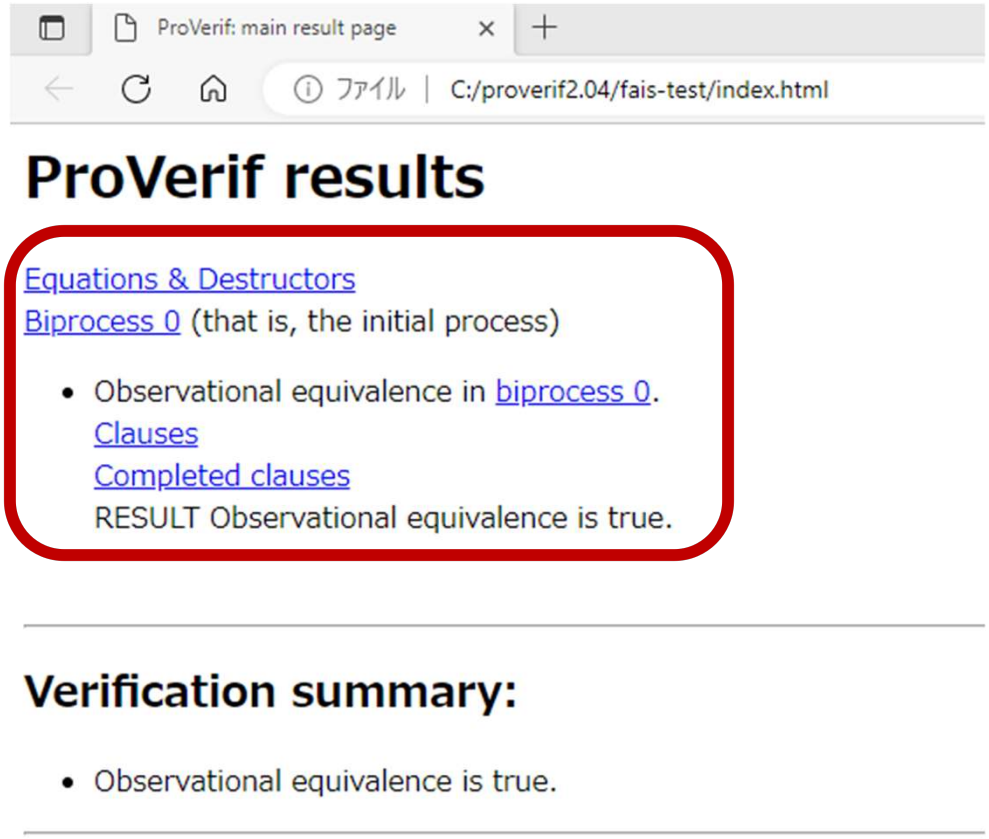
```
(*code: decision_diffie_hellman ProVerif*)
type G.
type exponent.
const g: G [data].
free d: channel.

fun exp(G, exponent): G.

equation forall x: exponent, y: exponent;
exp(exp(g,x),y) = exp(exp(g,y),x).

process
new a: exponent; new b: exponent; new c: exponent;
out(d, (exp(g,a), exp(g,b), choice[exp(exp(g,a),b), exp(g,c)]))
```

Tamarin-prover

```
(*code: decision_diffie_hellman Tamarin-prover*)
theory decision_diffie_hellman
begin
builtins: diffie-hellman
functions: g/0

rule DDH:
[ Fr(~a), Fr(~b), Fr(~c) ] --[ ]->
[ Out(<g^(~a), g^(~b), diff(g^(~a)^(~b),g^(~c))>) ]
end
```

[参考]ProVerif
P60. Chapter4 ： ProVerif 2.04: Automatic Cryptographic Protocol Verifier, User Manual and Tutorial

ProVerif : html出力コマンドを使用し、実施している内容を可視化



CUIからのLog出力以外も実施

Tamarin-prover : interactive mode を使用

Proof scripts

```
theory decision_diffie_hellman begin

Diff Rules
LHS: Message theory
RHS: Message theory
LHS: Message theory [Diff]
RHS: Message theory [Diff]
LHS: Multiset rewriting rules (3)
RHS: Multiset rewriting rules (3)
LHS: Multiset rewriting rules [Diff] (3)
RHS: Multiset rewriting rules [Diff] (3)
LHS: Raw sources (4 cases, deconstructions complete)
RHS: Raw sources (4 cases, deconstructions complete)
LHS: Raw sources [Diff] (4 cases, deconstructions
complete)
RHS: Raw sources [Diff] (4 cases, deconstructions
complete)
LHS: Refined sources (4 cases, deconstructions complete)
RHS: Refined sources (4 cases, deconstructions complete)
LHS: Refined sources [Diff] (4 cases, deconstructions
complete)
RHS: Refined sources [Diff] (4 cases, deconstructions
complete)
LHS: Lemmas

RHS: Lemmas

Diff-Lemmas

lemma Observational_equivalence:
by sorry

end
```

Visualization display

Applicable Proof Methods: Goals sorted according to the 'smart' heuristic (for diff proofs)

1. **rule-equivalence** // Prove equivalence using rule equivalence

a. **autoprove** (A. **for all solutions**)
b. **autoprove** (B. **for all solutions**) with proof-depth bound 5

Constraint system

proof type: none

current rule: none

system: none

mirror system: none

protocol rules:

construction rules:

destruction rules:

0 sub-case(s)

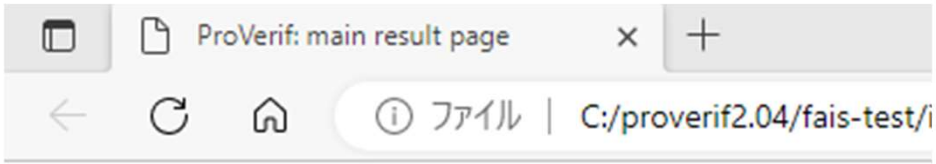
観測等価性の証明をする場合には、”sorry ”を押下する

CUIからのLog出力以外も実施

6. 検証の詳細結果

1. ProVerif
2. Tamarin-prover

ProVerif - verification details



ProVerif results

Equations & Destructors
Biprocess 0 (that is, the initial process)

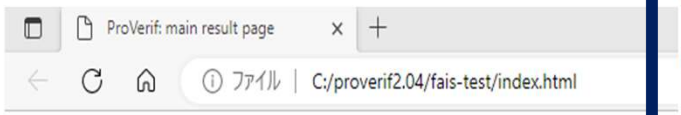
- Observational equivalence in biprocess 0.
  Clauses
  Completed clauses
  RESULT Observational equivalence is true.

Verification summary:

- Observational equivalence is true.

```
(*log: decision_diffie_hellman ProVerif*)
Linear part:
exp(exp(g,x),y) = exp(exp(g,y),x)
Completing equations...
Completed equations:
exp(exp(g,x),y) = exp(exp(g,y),x)
Convergent part: No equation.
Biprocess 0 (that is, the initial process):
{1}new a: exponent;
{2}new b: exponent;
{3}new c: exponent;
{4}out(d, (exp(g,a),exp(g,b),choice[exp(exp(g,a),b),exp(g,c)]))
-- Observational equivalence in biprocess 0.
Translating the process into Horn clauses...
Termination warning: v ≠ v_1 && attacker2(v_2,v) && attacker2(v_2,v_1) -> bad
Selecting 0
Termination warning: v ≠ v_1 && attacker2(v,v_2) && attacker2(v_1,v_2) -> bad
Selecting 0
Completing...
Termination warning: v ≠ v_1 && attacker2(v_2,v) && attacker2(v_2,v_1) -> bad
Selecting 0
Termination warning: v ≠ v_1 && attacker2(v,v_2) && attacker2(v_1,v_2) -> bad
Selecting 0
RESULT Observational equivalence is true.

Verification summary:
Observational equivalence is true.
```

ProVerif - verification details



ProVerif results

Equations & Destructors
Biprocess 0 (that is, the initia process)

- Observational equivalence in biprocess 0.
  Clauses
  Completed clauses
  RESULT Observational equivalence is true.

Verification summary:

- Observational equivalence is true.

Equations

Linear part

Initial equations:

- exp(exp(g,x),y) = exp(exp(g,y),x)

Completed equations:

- exp(exp(g,x),y) = exp(exp(g,y),x)

Convergent part

No equation.

Completed destructors

(fail-any || @mayfail_v) => fail-any
(v || @mayfail_v) => @mayfail_v if v ≠ true
(true || @mayfail_v) => true

not(fail-any) => fail-any
not(v) => true if v ≠ true
not(true) => false

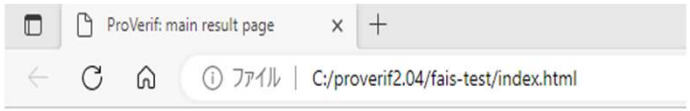
is_nat(fail-any) => fail-any
is_nat(v) => false if is_not_nat(v)
is_nat(v) => true if is_nat(v)

(v ≥ fail-any) => fail-any
(fail-any ≥ @mayfail_v) => fail-any
(v ≥ v_1) => fail-any if is_not_nat(v_1)
(v ≥ v_1) => fail-any if is_not_nat(v)
(v ≥ v_1) => false if v_1 ≥ v + 1
(v ≥ v_1) => true if v ≥ v_1

Biprocess 0 (the initial process)

{1} new a: exponent;
{2} new b: exponent;
{3} new c: exponent;
{4} out(d, (exp(g,a),exp(g,b),choice[exp(exp(g,a),b),exp(g,c)]))

ProVerif - verification details



ProVerif results

Equations & Destructors
Biprocess 0 (that is, the initial process)

- Observational equivalence in biprocess 0.
  Clauses
  Completed clauses
  RESULT Observational equivalence is true.

Verification summary:

- Observational equivalence is true.

例: clause12
攻撃者が関数expを適用

Completed clauses

- attacker2(x,x_1) && attacker2(y,y_1) -> attacker2(exp(exp(g,y),x),exp(exp(g,y_1),x_1))

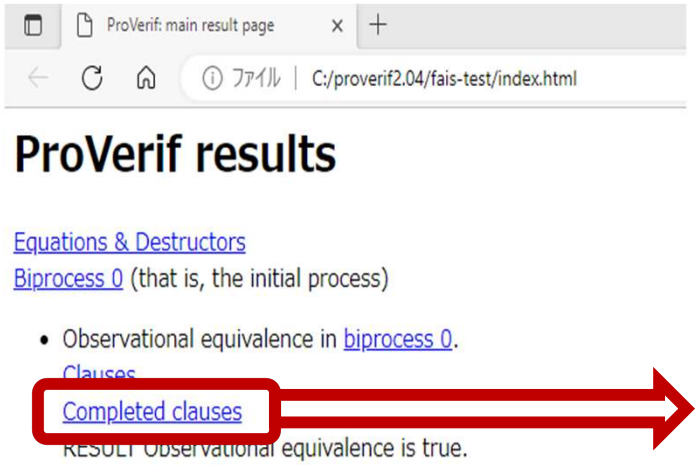
Initial clauses

- Clause 0: v ≠ true && attacker2(v,true) && attacker2(@mayfail_v,@mayfail_v_1) -> attacker2(@mayfail_v,true)
  (The attacker applies function ||.)
- Clause 1: v ≠ true && attacker2(true,v) && attacker2(@mayfail_v,@mayfail_v_1) -> attacker2(true,@mayfail_v_1)
  (The attacker applies function ||.)
- Clause 2: attacker2(true,true)
  (The attacker applies function true.)
- Clause 3: v ≠ true && attacker2(v,true) -> attacker2(true,false)
  (The attacker applies function not.)
- Clause 4: v ≠ true && attacker2(true,v) -> attacker2(false,true)
  (The attacker applies function not.)
- Clause 5: is_nat(v) && is_not_nat(v_1) && attacker2(v_1,v) -> attacker2(false,true)
  (The attacker applies function is_nat.)
- Clause 6: is_nat(v) && is_not_nat(v_1) && attacker2(v,v_1) -> attacker2(true,false)
  (The attacker applies function is_nat.)
- Clause 7: attacker2(g,g)
  (The attacker applies function g.)
- Clause 8: attacker2(false,false)
  (The attacker applies function false.)
- Clause 9: attacker2(v,v_1) && attacker2(v_2,v_3) -> attacker2(exp(v,v_2),exp(v_1,v_3))
  (The attacker applies function exp.)
- Clause 10: attacker2(v,exp(g,x)) && attacker2(v_1,y) -> attacker2(exp(v,v_1),exp(exp(g,y),x))
  (The attacker applies function exp.)
- Clause 11: attacker2(exp(g,x),v) && attacker2(y,v_1) -> attacker2(exp(exp(g,y),x),exp(v,v_1))
  (The attacker applies function exp.)
- Clause 12: attacker2(exp(g,x),exp(g,x_1)) && attacker2(y,y_1) -> attacker2(exp(exp(g,y),x),exp(exp(g,y_1),x_1))
  (The attacker applies function exp.)
- Clause 13: is_not_nat(v) && v_1 ≥ v_2 + 1 && attacker2(v_3,v_2) && attacker2(v,v_1) -> attacker2(fail-any,false)
  (The attacker applies function >=.)

cker2(v_3,v_1) && attacker2(v,v_2) -> attacker2(fail-any,true)

attacker2(v,v_2) && attacker2(v_3,v_1) -> attacker2(fail-any,false)
(The attacker applies function >=.)

ProVerif – verification details



Completed clauses

- attacker2(v,y) -> attacker2(exp(exp(exp(g,a[]),b[]),v),exp(exp(g,y),c[]))
- attacker2(v,y) -> attacker2(exp(exp(exp(g,b[]),a[]),v),exp(exp(g,y),c[]))
- attacker2(v,y) -> attacker2(exp(exp(g,b[]),v),exp(exp(g,y),b[]))
- attacker2(y,v) -> attacker2(exp(exp(g,y),b[]),exp(exp(g,b[]),v))
- attacker2(y,y_1) -> attacker2(exp(exp(g,y),b[]),exp(exp(g,y_1),b[]))
- attacker2(v,y) -> attacker2(exp(exp(g,a[]),v),exp(exp(g,y),a[]))
- attacker2(y,v) -> attacker2(exp(exp(g,y),a[]),exp(exp(g,a[]),v))
- attacker2(y,y_1) -> attacker2(exp(exp(g,y),a[]),exp(exp(g,y_1),a[]))
- attacker2(x,x_1) && attacker2(y,y_1) -> attacker2(exp(exp(g,y),x),exp(exp(g,y_1),x_1))
- attacker2(v,x) && attacker2(v_1,y) -> attacker2(exp(exp(g,v),v_1),exp(exp(g,y),x))
- attacker2(exp(exp(g,a[]),b[]),exp(g,c[]))
- attacker2(exp(exp(g,b[]),a[]),exp(g,c[]))
- attacker2(exp(g,b[]),exp(g,b[]))
- attacker2(exp(g,a[]),exp(g,a[]))
- attacker2(new-name[!att = v],new-name[!att = v])
- attacker2(d[],d[])
- attacker2(fail-any,fail-any)
- attacker2(v,v_1) -> output2(v,v_1)
- attacker2(v,v_1) -> input2(v,v_1)
- equal(v,v)
- attacker2(v,v_1) && attacker2(v_2,v_3) -> mess2(v,v_2,v_1,v_3)
- attacker2(v,v_1) && attacker2(v_2,v_3) && attacker2(v_4,v_5) -> attacker2((v,v_2,v_4),(v_1,v_3,v_5))
- attacker2(v,v_1) -> attacker2(v + 1,v_1 + 1)
- attacker2(0,0)
- attacker2(v,v_1) && attacker2(v_2,v_3) -> attacker2(exp(v,v_2),exp(v_1,v_3))
- attacker2(false,false)
- attacker2(g,g)
- attacker2(true,true)

ProVerif - verification details

CUI実行Log

```
(*log: decision_diffie_hellman ProVerif*)
Linear part:
exp(exp(g,x),y) = exp(exp(g,y),x)
Completing equations...
Completed equations:
exp(exp(g,x),y) = exp(exp(g,y),x)
Convergent part: No equation.
Biprocess 0 (that is, the initial process):
{1}new a: exponent;
{2}new b: exponent;
{3}new c: exponent;
{4}out(d, (exp(g,a),exp(g,b),choice[exp(exp(g,a),b),exp(g,c)]))
-- Observational equivalence in biprocess 0.
Translating the process into Horn clauses...
Termination warning: v ≠ v_1 && attacker2(v_2,v) && attacker2(v_2,v_1) -> bad
Selecting 0
Termination warning: v ≠ v_1 && attacker2(v,v_2) && attacker2(v_1,v_2) -> bad
Selecting 0
Completing...
Termination warning: v ≠ v_1 && attacker2(v_2,v) && attacker2(v_2,v_1) -> bad
Selecting 0
Termination warning: v ≠ v_1 && attacker2(v,v_2) && attacker2(v_1,v_2) -> bad
Selecting 0
RESULT Observational equivalence is true.
--------------------------------------------------------------
Verification summary:
Observational equivalence is true.
--------------------------------------------------------------
```

別の検証方法

ProVerif : Choice を使用しない – equivalence -

ProVerif equivalence

```
(*code: decision_diffie_hellman ProVerif for equivalence*)
type G.
type exponent.
const g: G [data].
free d: channel.

fun exp(G, exponent): G.

equation forall x: exponent, y: exponent;
 exp(exp(g,x),y) = exp(exp(g,y),x).

equivalence
 new a: exponent; new b: exponent; new c: exponent; out(d, (exp(g,a), exp(g,b), exp(exp(g,a),b)))
 new a: exponent; new b: exponent; new c: exponent; out(d, (exp(g,a), exp(g,b), exp(g,c)))
```

Choice を使用しない - equivalence -

ProVerif - verification details

equivalence Log ProVerif

```
Linear part:
exp(exp(g,x),y) = exp(exp(g,y),x)
Completing equations...
Completed equations:
exp(exp(g,x),y) = exp(exp(g,y),x)
Convergent part: No equation.
Process 1 (that is, the first initial process):
{1}new a: exponent;
{2}new b: exponent;
{3}new c: exponent;
{4}out(d, (exp(g,a),exp(g,b),exp(exp(g,a),b)))

Process 2 (that is, the second initial process):
{5}new a_1: exponent;
{6}new b_1: exponent;
{7}new c_1: exponent;
{8}out(d, (exp(g,a_1),exp(g,b_1),exp(g,c_1)))

-- Observational equivalence between two processes
-- Observational equivalence in biprocess 3 (that is, the merge of process 1 and process 2, simplified)
{1}new a_2: exponent;
{2}new b_2: exponent;
{3}new a_3: exponent;
{4}new b_3: exponent;
{5}new c_2: exponent;
{6}out(d, choice[(exp(g,a_2),exp(g,b_2),exp(exp(g,a_2),b_2)),(exp(g,a_3),exp(g,b_3),exp(g,c_2))])
Translating the process into Horn clauses...
Termination warning: v ≠ v_1 && attacker2(v_2,v) && attacker2(v_2,v_1) -> bad
Selecting 0
Termination warning: v ≠ v_1 && attacker2(v,v_2) && attacker2(v_1,v_2) -> bad
Selecting 0
Completing...
Termination warning: v ≠ v_1 && attacker2(v_2,v) && attacker2(v_2,v_1) -> bad
Selecting 0
Termination warning: v ≠ v_1 && attacker2(v,v_2) && attacker2(v_1,v_2) -> bad
Selecting 0
RESULT Observational equivalence is true.

Verification summary:

Equivalence between process 1 and process 2 is true.
```

Result

ProVerif - verification details

Choiceを使用

```
(*log: decision_diffie_hellman ProVerif*)
Linear part:
exp(exp(g,x),y) = exp(exp(g,y),x)
Completing equations...
Completed equations:
exp(exp(g,x),y) = exp(exp(g,y),x)
Convergent part: No equation.
Biprocess 0 (that is, the initial process):
{1}new a: exponent;
{2}new b: exponent;
{3}new c: exponent;
{4}out(d, (exp(g,a),exp(g,b),choice[exp(exp(g,a),b),exp(g,c)]))
-- Observational equivalence in biprocess 0.
Translating the process into Horn clauses...
Termination warning: v ≠ v_1 && attacker2(v_2,v) && attacker2(v_2,v_1)
Selecting 0
Termination warning: v ≠ v_1 && attacker2(v,v_2) && attacker2(v_1,v_2)
Selecting 0
Completing...
Termination warning: v ≠ v_1 && attacker2(v_2,v) && attacker2(v_2,v_1)
Selecting 0
Termination warning: v ≠ v_1 && attacker2(v,v_2) && attacker2(v_1,v_2)
Selecting 0
RESULT Observational equivalence is true.
--------------------------------------------------------------
Verification summary:
Observational equivalence is true.
--------------------------------------------------------------
```

equivalenceを使用

equivalence Log ProVerif

```
Linear part:
exp(exp(g,x),y) = exp(exp(g,y),x)
Completing equations...
Completed equations:
exp(exp(g,x),y) = exp(exp(g,y),x)
Convergent part: No equation.
Process 1 (that is, the first initial process):
{1}new a: exponent;
{2}new b: exponent;
{3}new c: exponent;
{4}out(d, (exp(g,a),exp(g,b),exp(exp(g,a),b)))

Process 2 (that is, the second initial process):
{5}new a_1: exponent;
{6}new b_1: exponent;
{7}new c_1: exponent;
{8}out(d, (exp(g,a_1),exp(g,b_1),exp(g,c_1)))
-- Observational equivalence between two processes
-- Observational equivalence in biprocess 3 (that is, the merge of process 1 and process 2, simplified)
{1}new a_2: exponent;
{2}new b_2: exponent;
{3}new a_3: exponent;
{4}new b_3: exponent;
{5}new c_2: exponent;
{6}out(d, choice[(exp(g,a_2),exp(g,b_2),exp(exp(g,a_2),b_2)),(exp(g,a_3),exp(g,b_3),exp(g,c_2))])

Translating the process into Horn clauses...
Termination warning: v ≠ v_1 && attacker2(v_2,v) && attacker2(v_2,v_1) -> bad
Selecting 0
Termination warning: v ≠ v_1 && attacker2(v,v_2) && attacker2(v_1,v_2) -> bad
Selecting 0
Completing...
Termination warning: v ≠ v_1 && attacker2(v_2,v) && attacker2(v_2,v_1) -> bad
Selecting 0
Termination warning: v ≠ v_1 && attacker2(v,v_2) && attacker2(v_1,v_2) -> bad
Selecting 0
RESULT Observational equivalence is true.

--------------------------------------------------------------
Verification summary:

Equivalence between process 1 and process 2 is true.

--------------------------------------------------------------
```

Tamarin-prover – verification details

CUI実行Log

```
(*log: decision_diffie_hellman Tamarin-prover*)
        :
    (omitted)
        :
rule (modulo E) DDH:
   [ Fr( ~a ), Fr( ~b ), Fr( ~c ) ] -->
   [ Out( <g^~a, g^~b, diff(g^~a^~b, g^~c)> ) ]
/* All well-formedness checks were successful. */
diffLemma Observational_equivalence:
rule-equivalence
  case Rule_DDH
  backward-search
    case LHS
    step( simplify )
    MIRRORED
  next
    case RHS
    step( simplify )
    MIRRORED
  qed
next
  case Rule_Destrd_0_fst
  backward-search
    case LHS
    step( simplify )
      next
        case pair
        MIRRORED
      qed
    qed
  qed
next
        :
    (omitted)
        :
next
  case Rule_Send
  backward-search
    case LHS
    step( simplify )
    MIRRORED
  next
    case RHS
    step( simplify )
    MIRRORED
  qed
qed
end
```

```
summary of summaries:
analyzed: fais-tamarin-ddh-20230113.spthy
  DiffLemma:   Observational_equivalence : verified (2522 steps)
```

Tamarin-prover – verification details

Proof scripts

```
theory decision_diffie_hellman begin

Diff Rules

LHS: Message theory

RHS: Message theory

LHS: Message theory [Diff]

RHS: Message theory [Diff]

LHS: Multiset rewriting rules (3)

RHS: Multiset rewriting rules (3)

LHS: Multiset rewriting rules [Diff] (3)

RHS: Multiset rewriting rules [Diff] (3)

LHS: Raw sources (4 cases, deconstructions complete)

RHS: Raw sources (4 cases, deconstructions complete)

LHS: Raw sources [Diff] (4 cases, deconstructions complete)

RHS: Raw sources [Diff] (4 cases, deconstructions complete)

LHS: Refined sources (4 cases, deconstructions complete)

RHS: Refined sources (4 cases, deconstructions complete)

LHS: Refined sources [Diff] (4 cases, deconstructions
complete)

RHS: Refined sources [Diff] (4 cases, deconstructions
complete)

LHS: Lemmas

RHS: Lemmas

Diff-Lemmas

lemma Observational_equivalence:
by sorry

end
```

Visualization display

Applicable Proof Methods: Goals sorted according to the 'smart' heuristic (for diff proofs)

1. rule-equivalence // Prove equivalence using rule equivalence

a. autoprove (A. for all solutions)
b. autoprove (B. for all solutions) with proof-depth bound 5

Constraint system

proof type: none

current rule: none

system: none

mirror system: none

protocol rules:

construction rules:

destruction rules:

0 sub-case(s)

証明を続ける為に、
”rule-equivalence ”
押下する

<u>２つのサブルーチンを生成する
それぞれの側で、全ての可能な
実行を構築する</u>

```
Diff-Lemmas

lemma Observational_equivalence:
rule-equivalence
  case Rule_DDH
  backward-search
    case LHS
    step( simplify )
    MIRRORED
  next
    case RHS
    step( simplify )
    MIRRORED
  qed
next
```

Tamarin-prover – verification details

Proof scripts

```
RHS: Multiset rewriting rules [Diff] (3)

LHS: Raw sources (4 cases, deconstructions complete)

RHS: Raw sources (4 cases, deconstructions complete)

LHS: Raw sources [Diff] (4 cases, deconstructions complete)

RHS: Raw sources [Diff] (4 cases, deconstructions complete)

LHS: Refined sources (4 cases, deconstructions complete)

RHS: Refined sources (4 cases, deconstructions complete)

LHS: Refined sources [Diff] (4 cases, deconstructions complete)

RHS: Refined sources [Diff] (4 cases, deconstructions complete)

LHS: Lemmas

RHS: Lemmas

Diff-Lemmas

lemma Observational_equivalence:
rule-equivalence
  case Rule_DDH
  backward-search
    case LHS
    step( simplify )
    MIRRORED
  next
    case RHS
    step( simplify )
    MIRRORED
  qed
next
  case Rule_Destrd_0_fst
  by sorry
next
  case Rule_Destrd_0_snd
  by sorry
next
  case Rule_Destrd_exp
  by sorry
next
  case Rule_Destrd_inv
  by sorry
next
  case Rule_Equality
  by sorry
next
  case Rule_Send
  by sorry
qed

end
```

Visualization display

Applicable Proof Methods: Goals sorted according to the 'smart' he

1. **backward-search** // Do backward search from rule

a. **autoprove** (A. **for all solutions**)
b. **autoprove** (B. **for all solutions**) with proof-depth bound 5

Constraint system

proof type: RuleEquivalence

current rule: IntrDestrd_0_fst

system: none

mirror system: none

protocol rules:
rule (**modulo** E) DDH:
[Fr(~a), Fr(~b), Fr(~c)]
-->
[Out(<g^~a, g^~b, diff(g^~a^~b, g^~c)>)]

construction rules:
rule (**modulo** AC) c_exp:
[!KU(x), !KU(x.1)] --[!KU(x^x.1)]-> [!KU(x^x.1)]

rule (**modulo** AC) c_fst:
[!KU(x)] --[!KU(fst(x))]-> [!KU(fst(x))]

rule (**modulo** AC) c_g:
[ ] --[!KU(g)]-> [!KU(g)]

rule (**modulo** AC) c_inv:
[!KU(x)] --[!KU(inv(x))]-> [!KU(inv(x))]

rule (**modulo** AC) c_mult:
[!KU(x), !KU(x.1)]
--[!KU((x*x.1))]->
[!KU((x*x.1))]

rule (**modulo** AC) c_one:
[ ] --[!KU(one)]-> [!KU(one)]

rule (**modulo** AC) c_pair:
[!KU(x), !KU(x.1)]
--[!KU(<x, x.1>)]->
[!KU(<x, x.1>)]

rule (**modulo** AC) c_snd:
[!KU(x)] --[!KU(snd(x))]-> [!KU(snd(x))]

construction rules:

rule (**modulo** AC) c_exp:
[!KU(x), !KU(x.1)] --[!KU(x^x.1)]-> [!KU(x^x.1)]

rule (**modulo** AC) c_fst:
[!KU(x)] --[!KU(fst(x))]-> [!KU(fst(x))]

rule (**modulo** AC) c_g:
[ ] --[!KU(g)]-> [!KU(g)]

rule (**modulo** AC) c_inv:
[!KU(x)] --[!KU(inv(x))]-> [!KU(inv(x))]

rule (**modulo** AC) c_mult:
[!KU(x), !KU(x.1)]
--[!KU((x*x.1))]->
[!KU((x*x.1))]

rule (**modulo** AC) c_one:
[ ] --[!KU(one)]-> [!KU(one)]

rule (**modulo** AC) c_pair:
[!KU(x), !KU(x.1)]
--[!KU(<x, x.1>)]->
[!KU(<x, x.1>)]

rule (**modulo** AC) c_snd:
[!KU(x)] --[!KU(snd(x))]-> [!KU(snd(x))]

Tamarin-prover - verification details

Isend
攻撃者が知っている値xを取り、それをプロトコルの入力In(x)に渡している

Irecv
プロトコルの出力Out(x)を受け取り，
攻撃者!KD(x)に渡している．

攻撃者が (g^a, g^b, g^ab) は
(g^a, g^b, g^c) と，計算上区別がつかない

Proof scripts

```
theory decision_diffie_hellman begin
Diff Rules
LHS: Message theory
RHS: Message theory
LHS: Message theory [Diff]
RHS: Message theory [Diff]
LHS: Multiset rewriting rules (3)
```

Multiset rewriting rules and restrictions [LHS]

Multiset Rewriting Rules

```
rule (modulo AC) isend:
   [ !KU( x ) ] --[ K( x ) ]-> [ In( x ) ]

rule (modulo AC) irecv:
   [ Out( x ) ] --> [ !KD( x ) ]

rule (modulo AC) DDH:
   [ Fr( ~a ), Fr( ~b ), Fr( ~c ) ]
  -->
   [ Out( <g^~a, g^~b, g^(~a*~b)> ) ]
```

Multiset rewriting rules and restrictions [RHS]

Multiset Rewriting Rules

```
rule (modulo AC) isend:
   [ !KU( x ) ] --[ K( x ) ]-> [ In( x ) ]

rule (modulo AC) irecv:
   [ Out( x ) ] --> [ !KD( x ) ]

rule (modulo AC) DDH:
   [ Fr( ~a ), Fr( ~b ), Fr( ~c ) ]
  -->
   [ Out( <g^~a, g^~b, g^~c> ) ]
```

Tamarin-prover – verification details

Proof scripts

```
RHS: Multiset rewriting rules [Diff] (3)

LHS: Raw sources (4 cases, deconstructions complete)

RHS: Raw sources (4 cases, deconstructions complete)

LHS: Raw sources [Diff] (4 cases, deconstructions
complete)

RHS: Raw sources [Diff] (4 cases, deconstructions
complete)

LHS: Refined sources (4 cases, deconstructions
complete)

RHS: Refined sources (4 cases, deconstructions
complete)

LHS: Refined sources [Diff] (4 cases, deconstructions
complete)

RHS: Refined sources [Diff] (4 cases, deconstructions
complete)

LHS: Lemmas

RHS: Lemmas

Diff-Lemmas

lemma Observational_equivalence:
rule-equivalence
  case Rule_DDH
  backward-search
    case LHS
    step( simplify )
    MIRRORED
  next
    case RHS
    step( simplify )
    MIRRORED
  qed
```

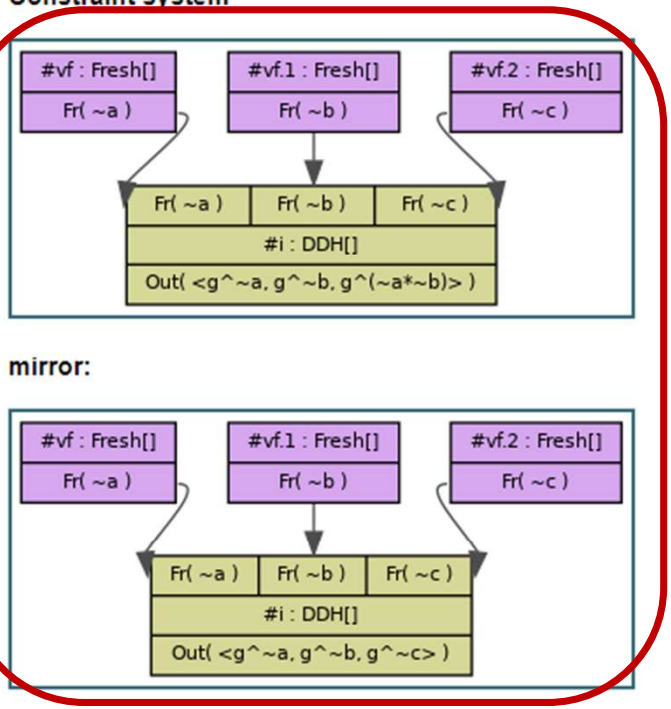
Method: MIRRORED ...

Applicable Proof Methods: Goals sorted according to the 'smart' heuristic (for diff proofs)

1. **MIRRORED** // Backward search completed

a. **autoprove** (A. **for all solutions**)
b. **autoprove** (B. **for all solutions**) with proof-depth bound 5

Constraint system



proof type: RuleEquivalence

current rule: ProtoDDH

system:
last: none

formulas:

equations:
subst:
conj:

lemmas:

allowed cases: refined

solved formulas: ∃ #i. (DiffProtoDDH( ) @ #i)

unsolved goals:

solved goals:
DiffProtoDDH( ) @ #i // nr: 0 (from rule DDH)" (useful2)"

mirror system:
nodes:
#i: **instance** (**modulo** AC) DDH:
[Fr(~a), Fr(~b), Fr(~c)]
-->
[Out(<g^~a, g^~b, g^~c>)]
#vf: **instance** (**modulo** AC) Fresh:
[ ] --> [Fr(~a)]
#vf.1: **instance** (**modulo** AC) Fresh:
[ ] --> [Fr(~b)]
#vf.2: **instance** (**modulo** AC) Fresh:
[ ] --> [Fr(~c)]
actions:
edges:
(#vf, 0) >--> (#i, 0), (#vf.1, 0) >--> (#i, 1),
(#vf.2, 0) >--> (#i, 2)
less:
unsolved goals:
last: none

formulas:

Tamarin-prover – verification details

Method: MIRRORED ...

rule (modulo AC) d_exp:
[!KD(x.4^(x.3*x.5)), !KU(inv((x.2*x.3)))]
-->
[!KD(x.4^(x.5*inv(x.2)))]

rule (modulo AC) d_exp:
[!KD(x.4^(x.3*x.5)), !KU((x.2*inv(x.3)))]
-->
[!KD(x.4^(x.2*x.5))]

rule (modulo AC) d_exp:
[!KD(x.4^(x.3*x.5*inv(x.2))), !KU((x.2*inv(x.3)))]
-->
[!KD(x.4^x.5)]

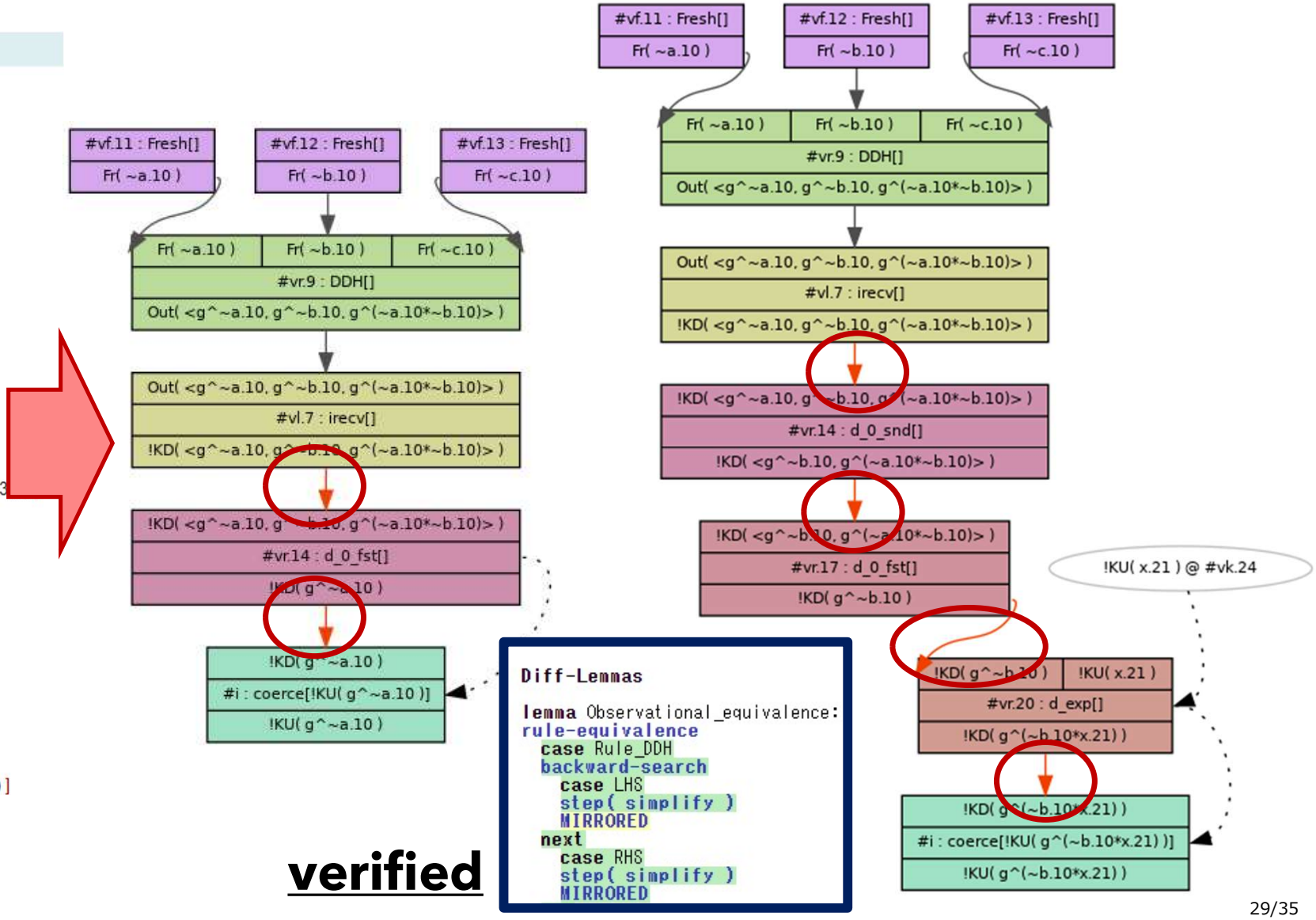
rule (modulo AC) d_exp:
[!KD(x.4^(x.3*x.5*inv(x.6))), !KU(inv((x.2*x.3)))]
-->
[!KD(x.4^(x.5*inv((x.2*x.6))))]

rule (modulo AC) d_exp:
[!KD(x.4^(x.3*x.5*inv((x.2*x.6)))), !KU((x.2*inv(x.3
-->
[!KD(x.4^(x.5*inv(x.6)))]

rule (modulo AC) d_exp:
[!KD(x.4^(x.3*inv(x.2))), !KU((x.2*inv(x.3)))]
-->
[!KD(x.4)]

rule (modulo AC) d_exp:
[!KD(x.4^(x.3*inv(x.5))), !KU(inv((x.2*x.3)))]
-->
[!KD(x.4^inv((x.2*x.5)))]

rule (modulo AC) d_exp:
[!KD(x.4^(x.3*inv((x.2*x.5)))), !KU((x.2*inv(x.3)))]
-->
[!KD(x.4^inv(x.5))]



verified

```
Diff-Lemmas

lemma Observational_equivalence:
rule-equivalence
  case Rule_DDH
  backward-search
    case LHS
    step( simplify )
    MIRRORED
  next
    case RHS
    step( simplify )
    MIRRORED
```

別の検証方法

"restriction" も使用

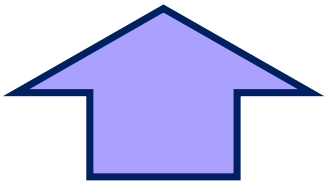
```
(*code: decision_diffie_hellman Tamarin-prover*)
theory decision_diffie_hellman_restriction
begin
builtins: diffie-hellman
functions: g/0

restriction Equality:
"All x y #i. Eq(x,y)@#i ==> x=y"

rule DDH_restriction:
  [ Fr(~a), Fr(~b), Fr(~c) ] --[Eq((g^(~a)^(~b)),g^(~c)) ]->
                  [ Out(<g^(~a), g^(~b), diff(g^(~a)^(~b),g^(~c))>) ]
end
```

DDHプロトコルの場合は,
g^abとg^cが=になる場合を除く

観測等価性はユーザが指定できるlemmaが無い
"restriction"を使って状態空間を取り除く

Result

```
summary of summaries:
analyzed: ddh-tamarin_diff_20230220_restriction.spthy
  DiffLemma:  Observational_equivalence  verified (124 steps)
```



```
summary of summaries:
analyzed: fais-tamarin-ddh-20230113.spthy
  DiffLemma:  Observational_equivalence : verified (2522 steps)
```

DDHプロトコルの場合は,
diff オペレータをそのまま使用するのではなく,
restriction を追加で使用した方が良い

Tamarin-prover – verification details

```
(*log: decision_diffie_hellman restriction Tamarin-prover*)
        :
    (omitted)
builtins: diffie-hellman
functions: fst/1, g/0, pair/2, snd/1
equations: fst(<x.1, x.2>) = x.1, snd(<x.1, x.2>) = x.2

restriction Equality [right]:∀
  " x y #i. (Eq( x, y ) @ #i) ⇒ (x = y)"
  // safety formula

                                    ty [left]:∀
                                    , y ) @ #i) ⇒ (x = y)"
                                    a

                                    DH_test:
                                      ~b ), Fr( ~c ) ]
                                      g~~c ) ]–>
                                    ~~b, diff(g~~a~~b, g~~c)> ) ]

                                    lness checks were successful. */

diffLemma Observational_equivalence:
rule-equivalence
  case Rule_DDH_test
  backward-search
    case LHS
    by step( simplify )
  next
    case RHS
    by step( simplify )
  qed

                                    1_0_fst

                                     )
                                    ( !KD( <x, x.1> ) ▲_0 #i ) )

                                     )
    by step( solve( !KD( <x, x.1> ) ▲_0 #i ) )
  qed
next
  case Rule_Destrd_0_snd
  backward-search
    case LHS
    step( simplify )
    by step( solve( !KD( <x, x.1> ) ▲_0 #i ) )
  next
    case RHS
    step( simplify )
    by step( solve( !KD( <x, x.1> ) ▲_0 #i ) )
  qed
      :
  (omitted)
      :
next
  case Rule_Send
  backward-search
    case LHS
    step( simplify )
    MIRRORED
next
    case RHS
    step( simplify )
    MIRRORED
  qed
qed
end

summary of summaries:
analyzed: ddh-tamarin_diff_20230220_restriction.spthy
  DiffLemma:  Observational_equivalence : verified (124 steps)
```

7. 結果の考察

1. ProVerif
2. Tamarin-prover

結果の考察

検証時間とLogサイズ

Tool Name	Operator	Time	Size of Log
ProVerif	choice	早い	少ない
Tamarin-prover	diff	時間がかかる	多い

Tool Name	Operator	Time	Size of Log
ProVerif	equivalence	遅くなる↓	増える↑
Tamarin-prover	restriction	早い↑	少ない↑

8. まとめと今後の課題

- Tamarin-prover： diff は, 厳密な観測等価性を検証するには向いているが,プロトコルによっては restrictionで制限をかける方が効率的(DDHプロトコルの場合)
- ProVerif: 用途に応じて使うクエリの場合分けが必要

モデル検査ツール ProVerifによる形式化技法と比較, 記述能力や検証結果の違いを更に明確にする

モジュールへの安全性検証に使用できる,ツールの優位性や使い分けなどを探る

ご清聴ありがとうございました.