TOSHIBA
Leading Innovation

Bug fixes for CryptoVerif

Yoshikazu Hanatani †
Yousuke Kakuno ‡
Kazuo Ohta ‡

† Toshiba
‡ the University of Electro-Communications

Topics

I will introduce 2 problems of CryptoVerif ver 1.07 and countermeasures.

- **Problem 1**
  - Try to prove the security of DH key exchange in the various weakened attack model, CryptoVerif outputs abnormal end.

```
Internal error: Invalid_argument("List.for_all2")
Please report bug to Bruno.Blanchet@ens.fr, including input file and output
```

- **Problem 2**
  - Cryptoverif judges that the modified FDH signature scheme is EuF-CMA without the Onewayness.

```
RESULT Proved event bad ==> false with probability 1. / |E|
All queries proved.
```

Problem 1

[HOM08]
Treatment of CDH Assumption for Blanchet Framework (Y.Hanatani, K.Ohta, H.Muratani)

Formalize 4 kinds of rewriting rules representing the CDH assumption.

Formalize 4 kinds of attack models for DH key exchange.

Verify the security of each combination by using CryptoVerif

Verification results in [HOM08]

	Model1	Model2	Model3	Model4
Rule1	○	E	E	E
Rule2	○	E	E	E
Rule3	○	○	E	×
Rule4	○	○	○	○

Expected results for theoretically

	Model1	Model2	Model3	Model4
Rule1	○	×	×	×
Rule2	○	×	○	×
Rule3	○	○	×	×
Rule4	○	○	○	○

○: Can prove security
×: Cannot prove security

E: Abort verification by abnormal end

Outline (1)

< Goal >

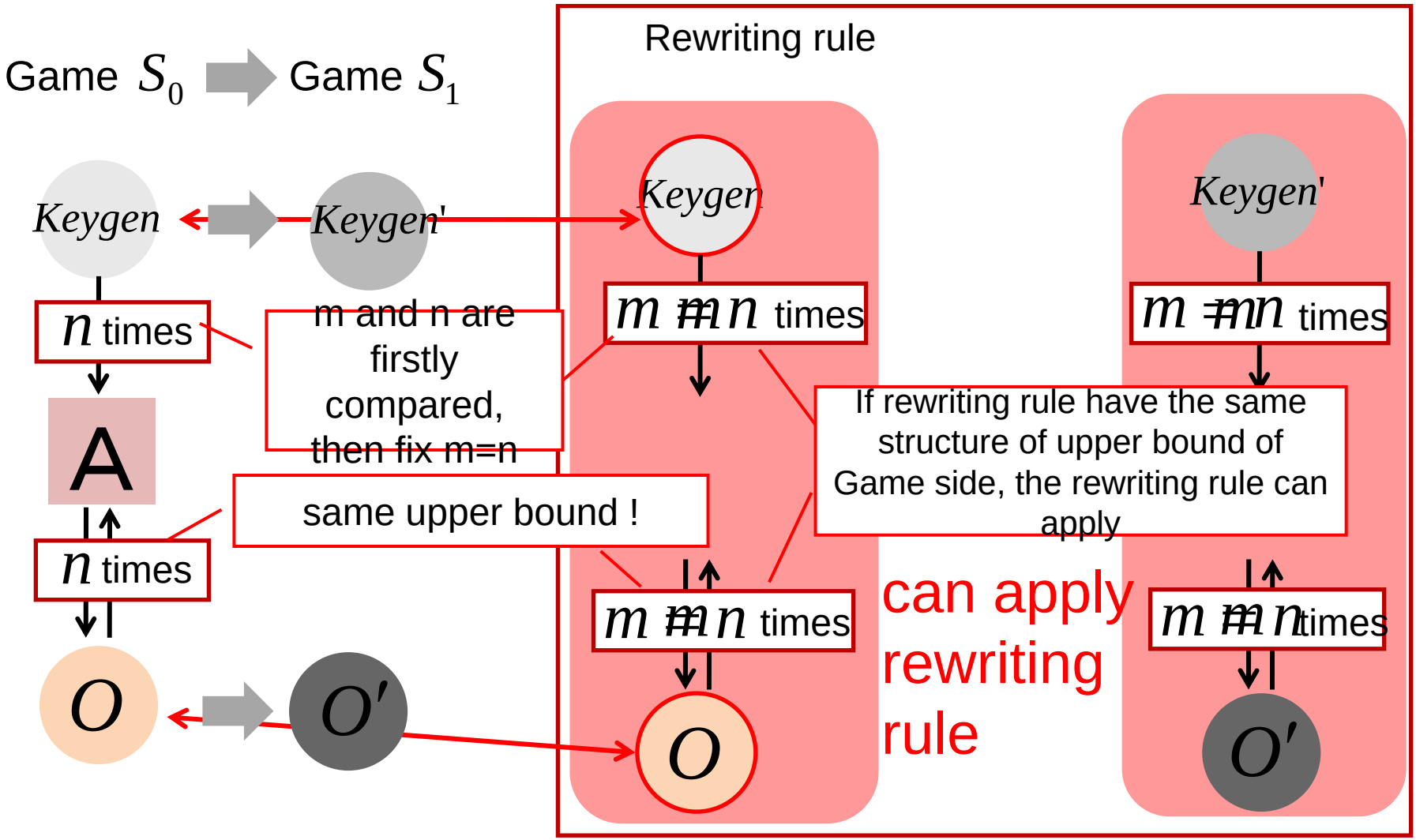
Find out the reason of the abnormal end.

Modify CryptoVerif to avoid the abnormal end.

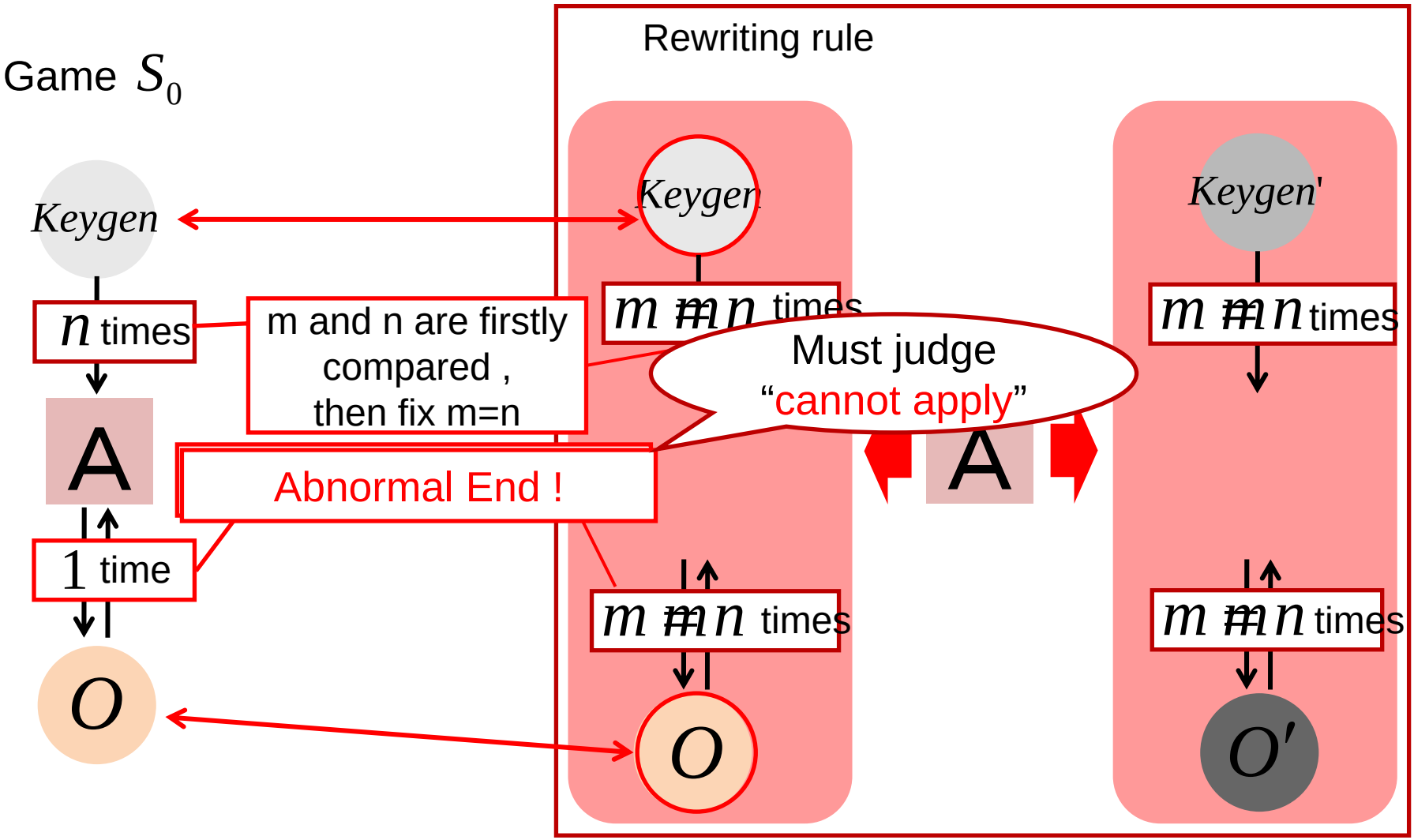
< Result >

Modified CryptoVerif return expected proof about our examples.

Normal Operation of CryptoVerif

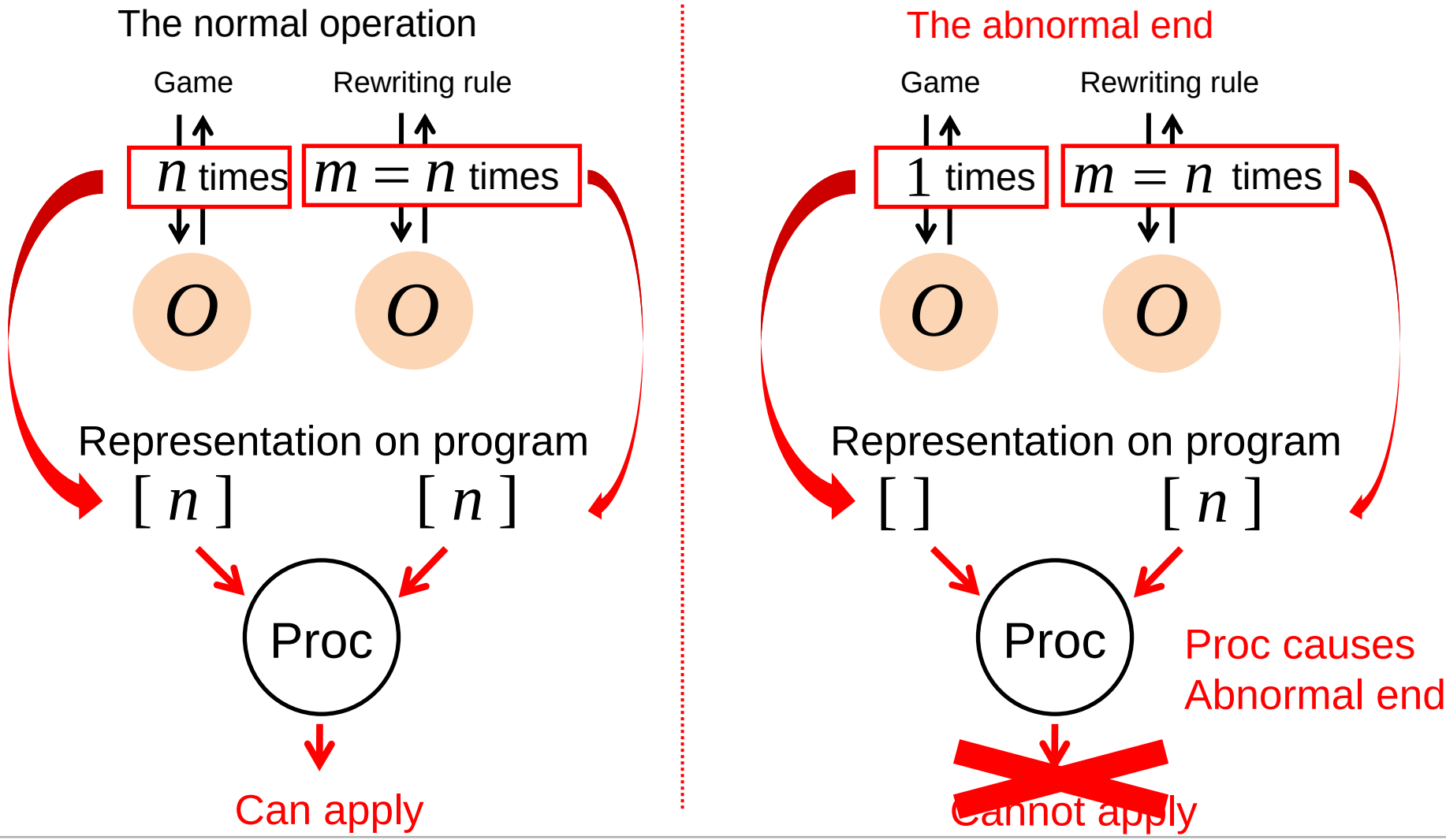


Abnormal operation of CryptoVerif



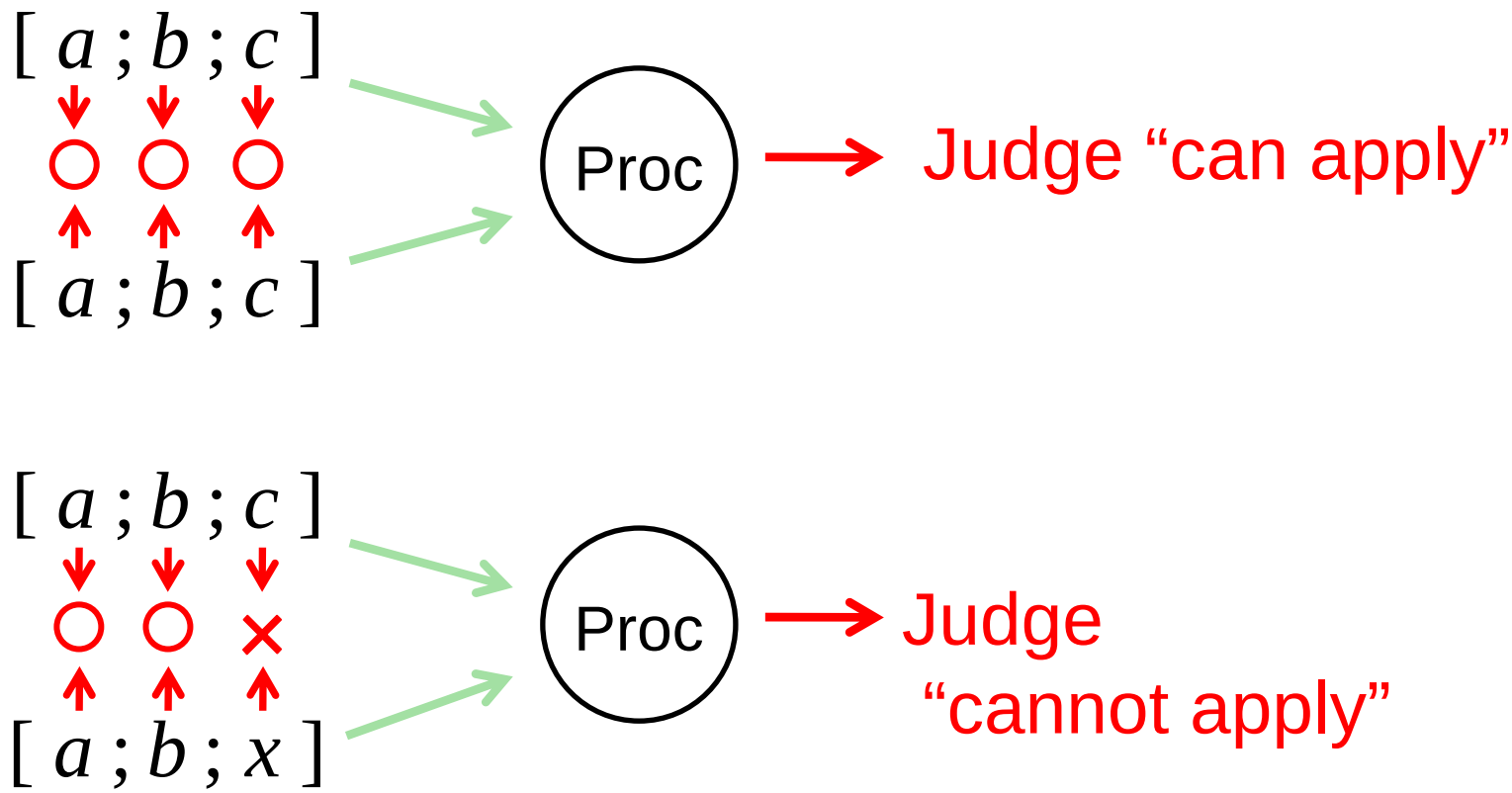
The reason of the abnormal end

In the verification, CryptoVerif compares the structure of upper bounds of oracle queries in game and rewriting rule by using the procedure Proc.



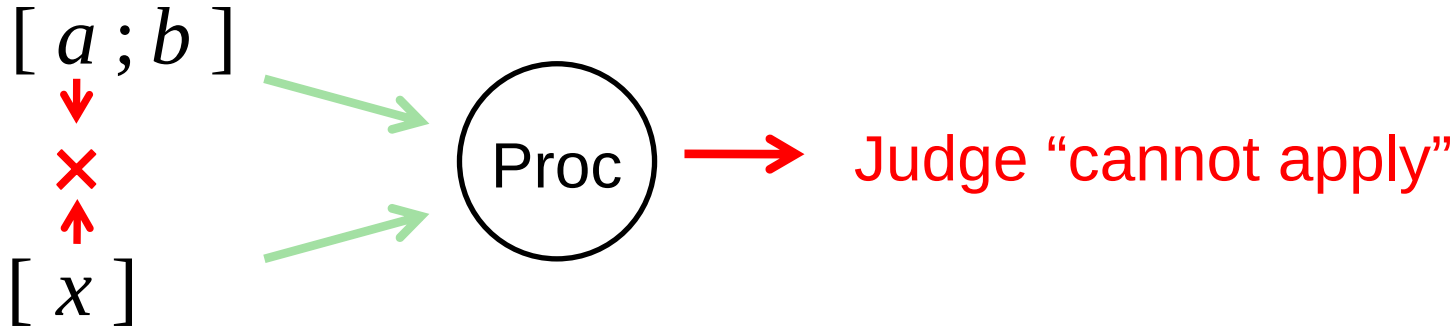
Comparison procedure : Proc (1/2)

（ｉ）Inputted lists have same number of element

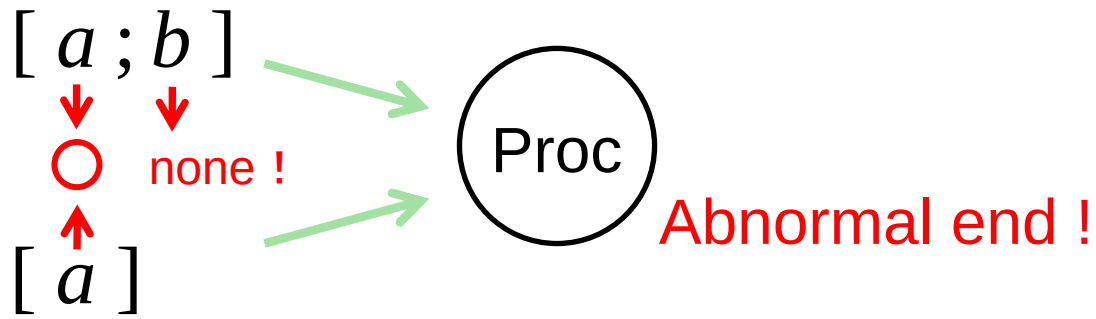


Comparison procedure: Proc (2/2)

（ii）Inputted lists have different number of elements. type 1



（iii）Inputted lists have different number of elements. type 2



Policy of modification

（ i ）The lists have same number of elements

⟶ Can apply / Cannot apply

（ ii ）The lists have different number of elements : type 1

⟶ Cannot apply

（iii）The lists have different number of elements : type 2

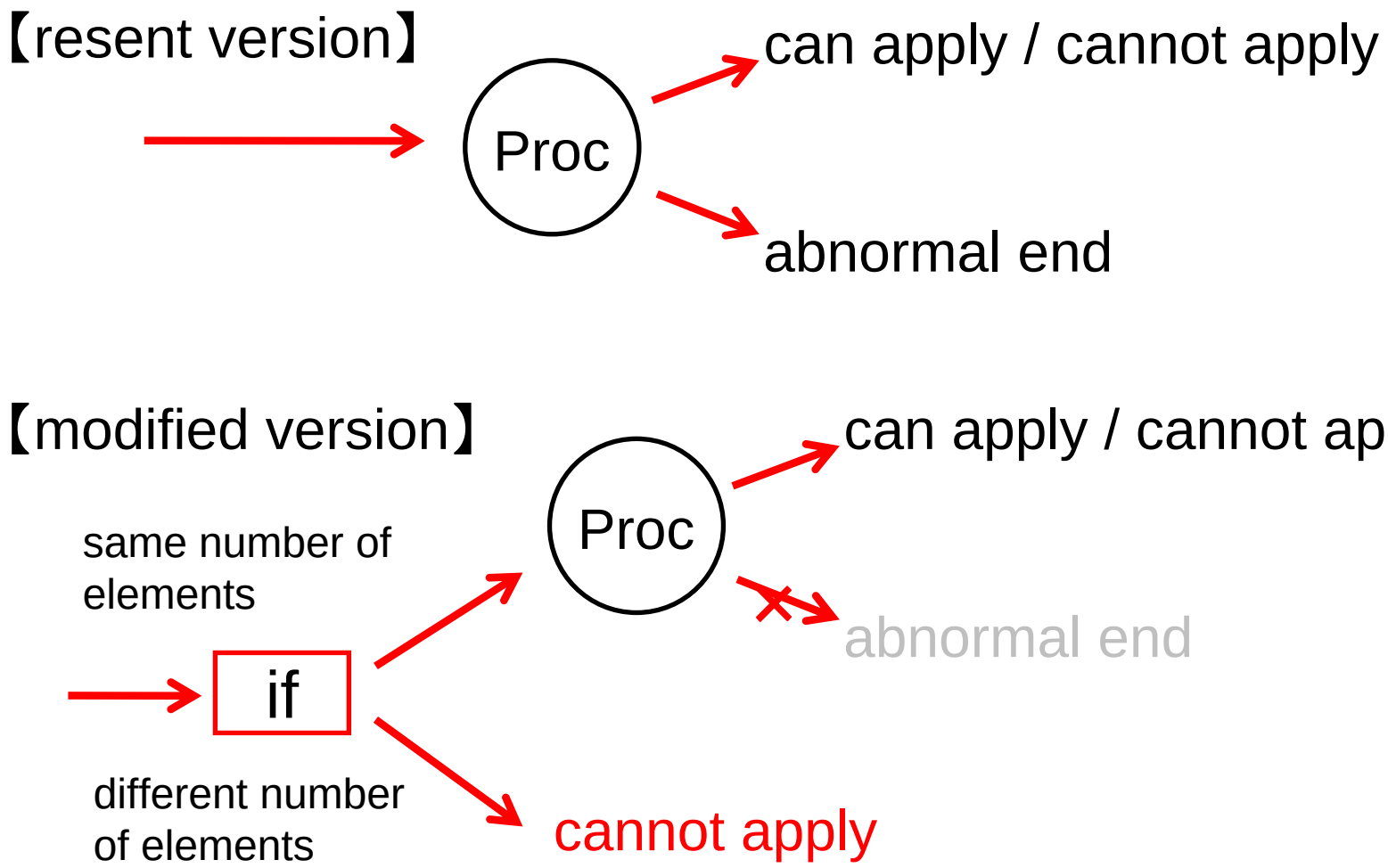
⟶ Abnormal end

（iii）The lists have different number of elements : type 2

⟶ Cannot apply

Modification plan

When the lists have different number of elements, Proc might cause the abnormal end.



Result of the countermeasure

Original CryptoVerif

	Model1	Model2	Model3	Model4
Rule1	○	E	E	E
Rule2	○	E	E	E
Rule3	○	○	E	×
Rule4	○	○	○	○

Modified version

	Model1	Model2	Model3	Model4
Rule1	○	×	×	×
Rule2	○	×	○	×
Rule3	○	○	×	×
Rule4	○	○	○	○

○ : Can prove security
× : Cannot prove security

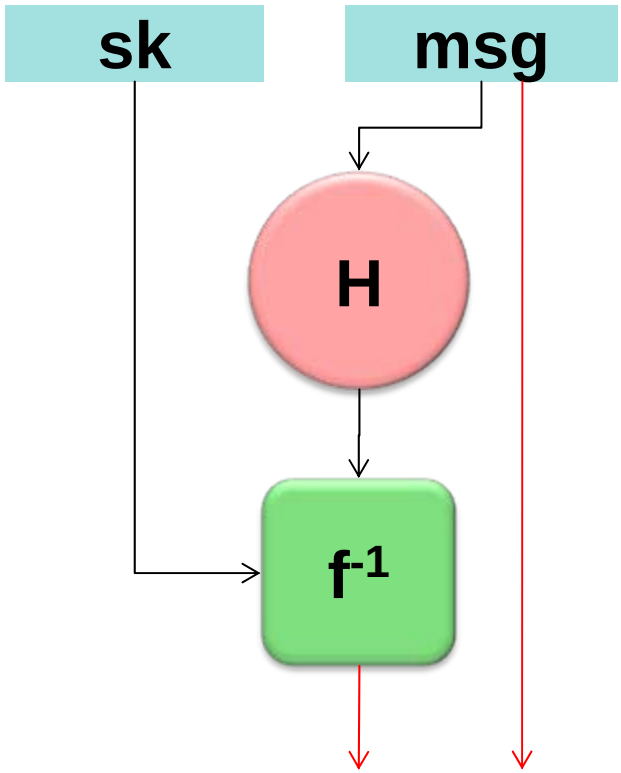
E : Abort verification by abnormal end

Succeed to improve CryptoVerif.

Problem 2

Formalize modified FDH signature for CryptoVerif.

Verified the security of the modified FDH signature by using CryptoVerif.



Original Scheme

H : {0,1}* -> D

f : D -> D

f^{-1}: D-> D

f is a permutation with OW.

Modified Scheme

H : {0,1}* -> E

f : D -> E

f^{-1}: E-> D

f is a bijective function with OW.

The orders of D and E are enough large.

CryptoVerif judges
"The modified FDH signature is EuF-CMA **without** Onewayness".

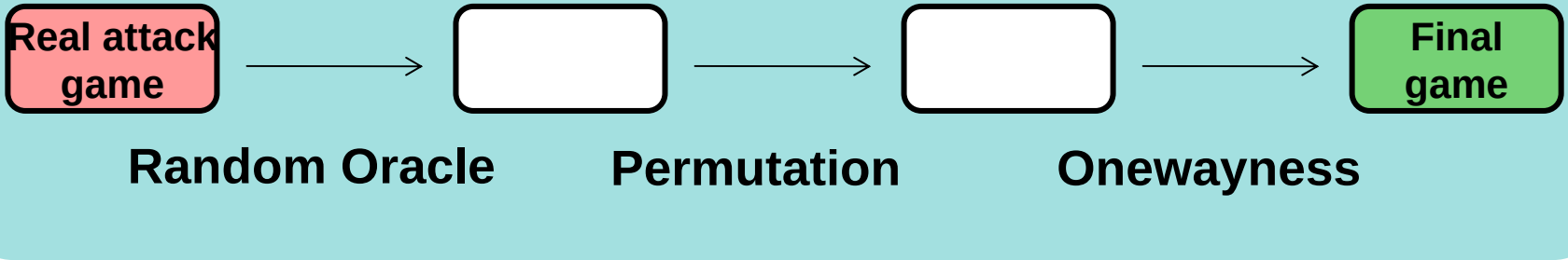
Proof flows of FDH sig. and mFDH sig.

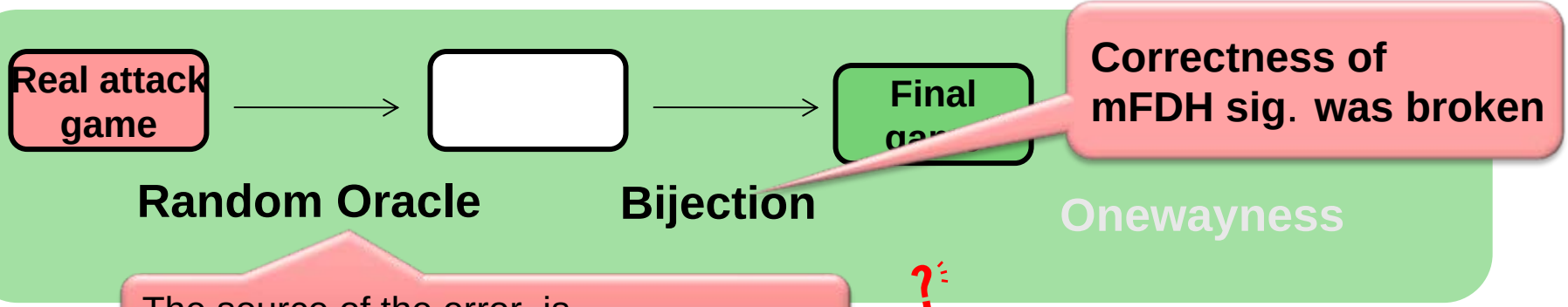
Rewriting rule for FDH sig.

Random Oracle : 0
Feature of permutation : 0
Onewayness: nk*nf*Succ(~)

Rewriting rule for mFDH sig.

Random Oracle : 0
Feature of bijection : 0
Onewayness: nk*nf*Succ(~)





The source of the error is
the rewriting rule of the random oracle

Outline (2)

< Goal >

Find out the reason of the invalid proof.

Improve CryptoVerif to avoid the invalid proof.

< Result >

Modified CryptoVerif avoids the invalid proof about our examples.

Succeed to improve CryptoVerif. I believe.

The analysis of the miss judge (1/2)

Rewriting with respect to Random Oracle to verify mFDH by CryptoVerif.

Game0

Random oracle

```
! !_13 <= qH
 in(c4[!_13], x: bitstring);
 out(c5[!_13], hash(x))
```

⇩ Bijection

```
! !_13 <= qH
 in(c4[!_13], x: bitstring);
 let x_23: bitstring = cst_bitstring in
 find @i_29 <= qS suchthat defined(y_34[@i_29], m[@i_29], x_21[@i_29], r_20[@i_29]) && otheruses(r_20[@i_29]) && (x = m[@i_29]) then
   out(c5[!_13], f(pkgen(r), y_34[@i_29]))
 orfind @i_28 <= qH suchthat defined(y_37[@i_28], x[@i_28], x_23[@i_28], r_22[@i_28]) && otheruses(r_22[@i_28]) && (x = x[@i_28]) then
   out(c5[!_13], f(pkgen(r), y_37[@i_28]))
 else
   new y_37: D;
   let r_22: E = cst_E in
   out(c5[!_13], f(pkgen(r), y_37))
```

↓ if x was queried then return above returned r.

← if x was not queried then randomly choose r and return r.

Game6

simplify ⇩

Game 9

```
! !_13 <= qH
 in(c4[!_13], x: bitstring);
 new y_37: D;
 out(c5[!_13], f(pkgen(r), y_37))
```

The analysis of the miss judge (2/2)

Game 9

```
! !_13 <= qH
 in(c4[!_13], x: bitstring);
 new y_37: D;
 out(c5[!_13], f(pkgen(r), y_37))
```

In Game 9, the Random Oracle returns random value for each input.

⇒ Different hash values are used in even the same input in Signing, Verification and Forgery.

In this situation ...

All forged signatures cannot be accepted with overwhelming provability.

CryptoVerif judges mFDH scheme is secure!

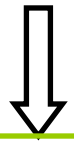
But, all correct signature cannot be accepted, either.

The reason of broken correctness

Rewriting rule of Random oracle

```
(x:bitstring) nH -> find u <= nH suchthat
                      defined(x[u],r[u])
                       ∧ otheruses(r[u])
                       ∧ x= x[u]
                      then r[u]
                    else
                      new r:D; r.
```

If r[u] dose not use out of find branch then otheruses(r[u]) = false.



Rewriting with bijection.

```
(x:bitstring) nH -> find u <= nH suchthat
                      defined(x[u],r[u],s[u])
                       ∧ otheruses(r[u])
                       ∧ x= x[u]
                      then f(g,s[u])
                    else
                      new s:E; f(g, s).
```

By rewriting game, r dose not use completely.

Remove otheruses from find branch

	mFDH Sig.	[BR93] PKE	Other examples
Original + include otheruses	×	○	○
Original + remove otheruses	○	×	○

○ : Can prove security
× : Cannot prove security

× : Invalid Proof

- **Succeed to prove the security of mFDH sig.**
- **But, [BR93]PKE cannot be proven.**

otheruses need for some verifications ...

Modification Plan

< Goal >

By adequately using otheruses, correctly prove the security of [BR93]PKE and the modified FDH signature.

< Apploch >

- Analyze operations of “defined” and “otheruses”.
- Modification plan
  - Modified usage of otheruses. [HKYO09]
    - Always can use the rewriting rules includes otheruses.

< Result >

- Modified CryptoVerif suceed to prove the examples.
- Notice: The effect of the modification was only confirmed by the experiment. Need to confirm theoretically.

The reason and one countermeasur

new r:D;
() → r

new s:E;
() → f(g,s)

```
in(x:bitstring)
    find u <= nH suchthat
          defined(x[u],r[u]) ∧ otheruses(r[u])
          ∧ x= x[u]
    then out (r[u])
    else
          new r:D;  out( r ).
```

```
in(x:bitstring)
    find u <= nH suchthat
          defined(x[u],r[u],s[u])∧otheruses(r[u])
          ∧ x= x[u]
    then out (f(g,s[u]))
    else
          new s:E;
          let r:D = cst_G in
          out( f(g, s) ).
```

⇒ otheruses(s[u])

Result of the countermeasur

	mFDH Sig.	[BR93] PKE	Other examples
original + include otheruses	×	○	○
original + remove otheruses	○	×	○
Modified + include otheruses	○	○	○

○ : Can prove security
× : Cannot prove security

× : Invalid Proof

Succeed to avoid the invalid proof of modified FDH sig.

Papers related to this talk

- [BP06] Blanchet, Pointcheval, “Automated Security Proofs with Sequences of Games”, CRYPTO 2006
- [HOM08] Hanatani, Ohta, Muratani, “Treatment of CDH Assumption for Blanchet Framework ”, (In Japanese) SCIS 2008
- [HYKO08] Hanatani, Yoneyama, Kakuno, Ohta, “Consideration on proof capabilities of CryptoVerif ”, (In Japanese) CSS 2008
- [KHYO09] Kakuno, Hanatani, Yoneyama, Ohta, “Remedying Abends of CryptoVerif”, (In Japanese), SCIS 2009
- [HKYO09] Hanatani, Kakuno, Yoneyama, Ohta, “Remendying invalid proofs of CryptoVerif” (In Japanese), FAIS 2009 spring.