ProVerifによるワンタイムパスワード認証方式に対する形式的な安全性検証

2014年3月19日

○荒井研一*，岩本智裕†，金子敏信*

*東京理科大学
†東京理科大学大学院理工学研究科

はじめに

- 本研究

  ワンタイムパスワード認証方式について
  Proverifを用いて形式的に記述し，安全性検証
  を行う

ワンタイムパスワード認証方式に対する
ProVerifの有用性を示す

本発表の検証対象：ROSI，SAS-X(2)

ワンタイムパスワード認証方式

- ワンタイムパスワード（OTP）認証方式

  セッション毎にパスワードを変更することにより，
  通信路が盗聴されたとしても安全性を保つこと
  のできる認証方式

- OTP認証方式の満たすべき安全性
  - サーバに保存されている情報が流出したとしてもなりすましができない
  - 通信データが改ざんされたとしてもサーバが改ざんデータを受理しない

ことが求められる

ProVerif

- 形式モデル(Dolev-Yaoモデル)での暗号プロトコルの自動検証ツール
- Blanchetらが開発(自身のサイトで公開)
- ホーン節によるプロトコル表現
- 秘匿，認証などの安全性特性を検証可能

さまざまな暗号プロトコルの安全性が検証されている

Dolev-Yaoモデル

完全な暗号(perfect encryption)を仮定

(例)共通鍵暗号方式:

$E(M,K)$ ⇒ 鍵Kにおける平文Mの暗号化

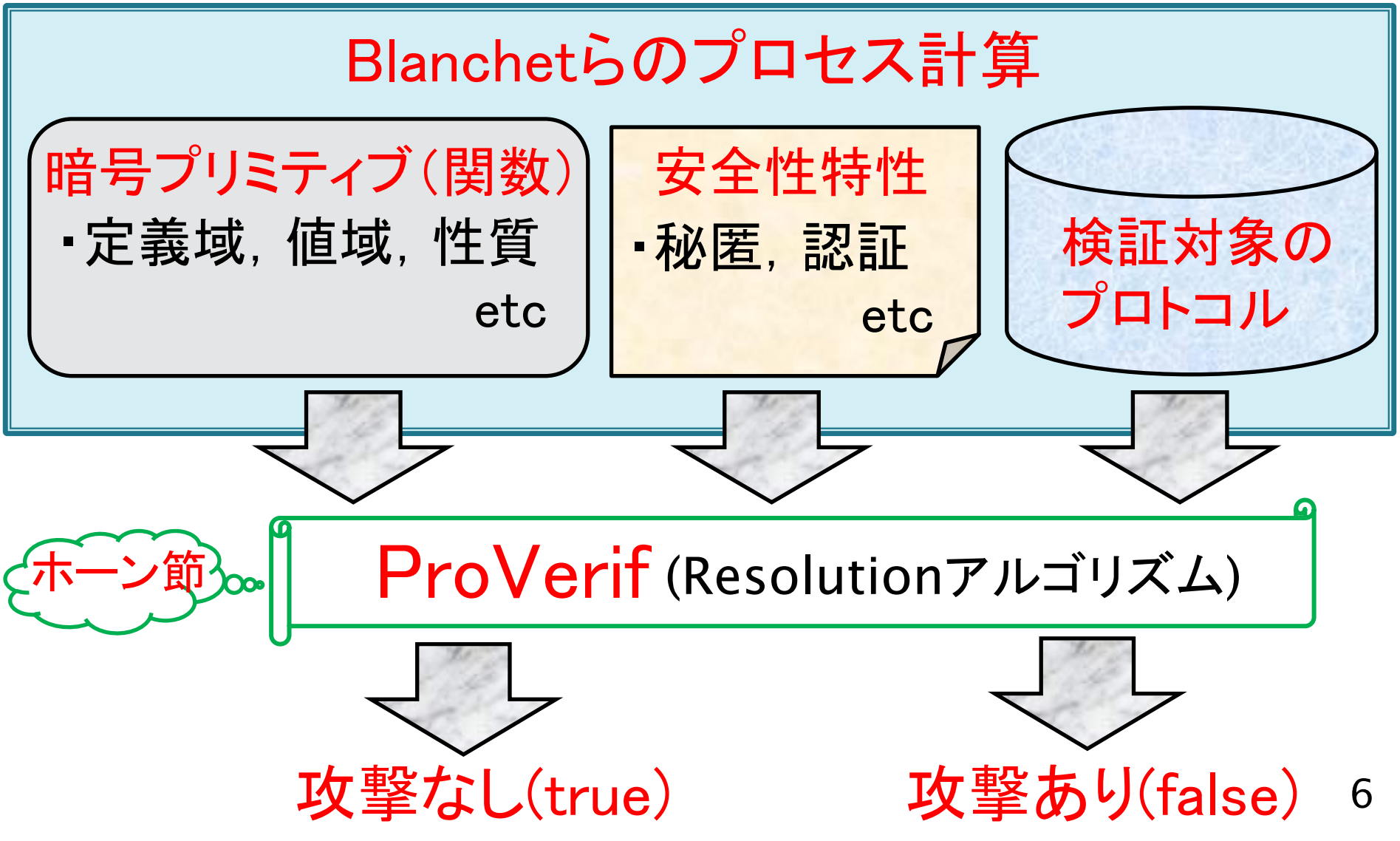
$D(C,K)$ ⇒ 鍵Kにおける暗号文Cの復号

$$M = D(E(M,K),K)$$

⇓

鍵Kを知らなければ，平文Mに関する
情報はまったく得られない

ProVerifによる検証



Blanchetらのプロセス計算の構文規則について

$M, N ::=$	terms
a, b, c, k, m, n, s	names
x, y, z	variables
$(M_1, \ldots, M_k)$	tuple
$h(M_1, \ldots, M_k)$	constructor/destructor application
$M = N$	term equality
$M <> N$	term inequality
M && M	conjunction
$M \mid\mid M$	disjunction
$\mathbf{not}(M)$	negation
$P, Q ::=$	processes
0	null process
$P \mid Q$	parallel composition
$!P$	replication
$\mathbf{new}\ n : t; P$	name restriction
$\mathbf{in}(M, x : t); P$	message input
$\mathbf{out}(M, N); P$	message output
$\mathbf{if}\ M\ \mathbf{then}\ P\ \mathbf{else}\ Q$	conditional
$\mathbf{let}\ x = M\ \mathbf{in}\ P\ \mathbf{else}\ Q$	term evaluation
$R(M_1, \ldots, M_k)$	macro usage
$T ::=$	patterns
$x : t$	typed variable
x	variable without explicit type
$(T_1, ..., T_n)$	tuple
$=M$	equality test

拡張π計算

暗号プロトコルの記述

Constructor及びDestructorの例

(例)共通鍵暗号方式:

Constructor: 暗号化 encrypt(m,k)

Destructor: 復号 decrypt(c,k)

decrypt(encrypt(m,k),k) → m

fun

reduc or equation

ProVerif のコード

```
fun encrypt(bitstring,key): bitstring.
reduc forall x: bitstring, y: key; decrypt(encrypt(x,y),y) = x.
```

公開鍵暗号, 署名, MAC, ハッシュ, Diffie-Hellman鍵交換, etc

プロトコル表現（ホーン節）

- ✓「メッセージ」 ＝ terms
  $M ::= x \mid f(M_1, \ldots, M_n) \mid k[M_1, \ldots, M_n]$
  encrypt(m, sk)
- ✓「特性（性質）」 ＝ facts
  $F ::=$ attacker(M)
- ✓「プロトコル，攻撃者」 ＝ Horn clauses
  $F_1 \wedge \cdots \wedge F_n \Rightarrow F$
  attacker(m) $\wedge$ attacker$(sk) \Rightarrow$
  attacker(encrypt(m,sk))

ProVerifにおけるホーン節について

- ProVerifにおけるホーン節の例

（例）共通鍵暗号：

暗号化 $\mathrm{encrypt}(m, k)$

$\mathrm{attacker}(m) \land \mathrm{attacker}(k) \Rightarrow \mathrm{attacker}(\mathrm{encrypt}(m, k))$

復号 $\mathrm{decrypt}\,(\mathrm{encrypt}(m, k), k) \rightarrow m$

$\mathrm{attacker}(\mathrm{encrypt}(m, k)) \land \mathrm{attacker}(k) \Rightarrow \mathrm{attacker}(m)$

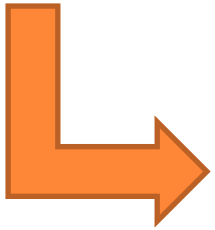
ProVerifにおけるホーン節について(2)

- ProVerifにおけるホーン節の例

・(例)プロトコル表現

参加者Bはpencrypt(sign(y, SK_A), PK_B)を受信したときに,
encrypt(s, y)を返信する

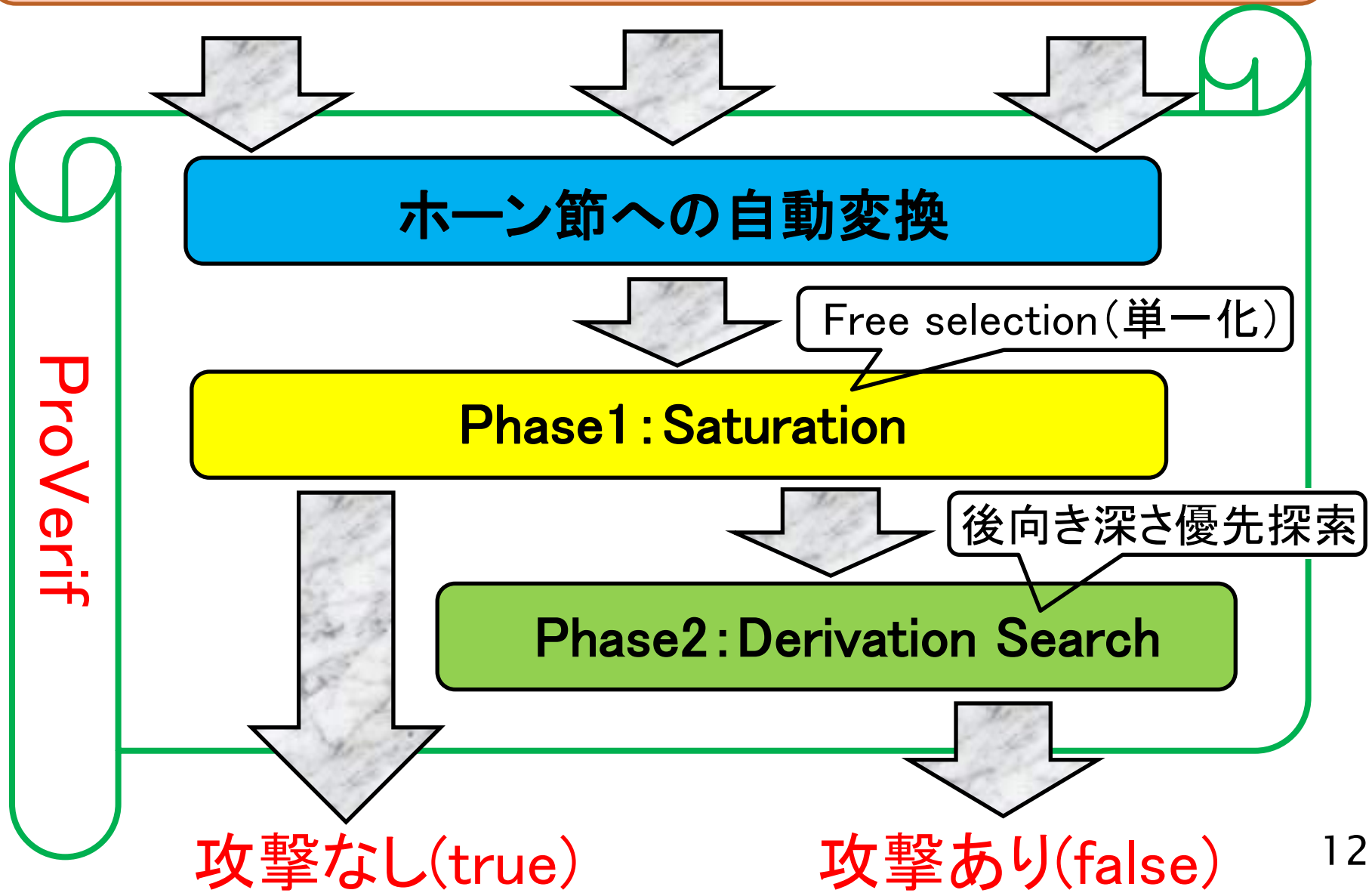
attacker(pencrypt(sign(y, SK_A), PK_B)) ⇒ attacker(encrypt(s, y))



攻撃者はBに対してpencrypt(sign(y, SK_A), PK_B)を送信し,
Bの返信であるencrypt(s, y)をインターセプトする

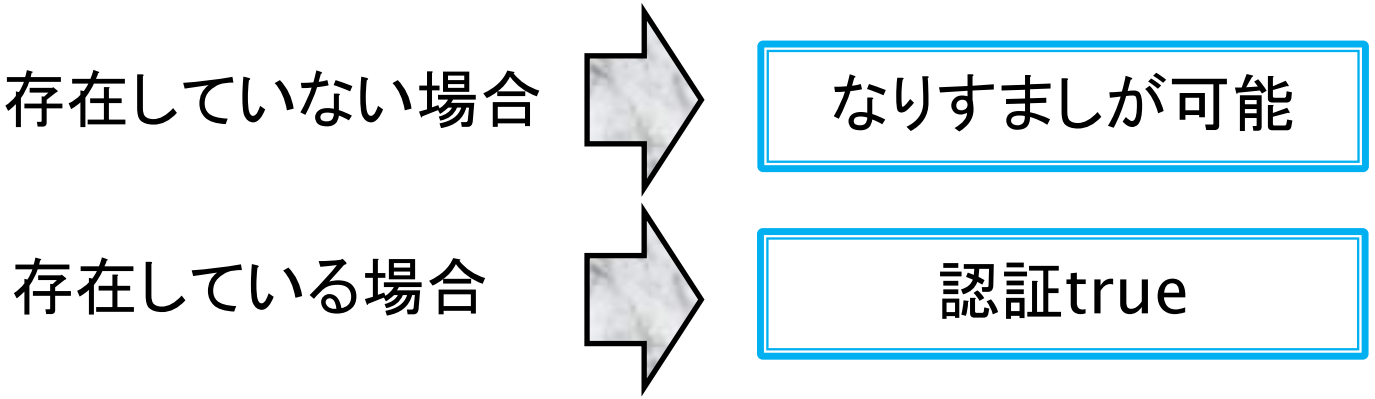
攻撃者視点

ProVerifによる検証



ProVerifでの認証について

- 対応表明
- 通信のプロセス中で情報が生成された時を「事前 event」,確認した時を「事後event」
- 事後eventが出現した際に、それに対応する事前eventが必ず存在しているか否かを検証



OTP認証方式に求められる安全性

- ある時点でサーバに保存されている情報が流出したとしても、攻撃者はユーザになりすませない

Stolen-Verifier（SV）attackに対する耐性

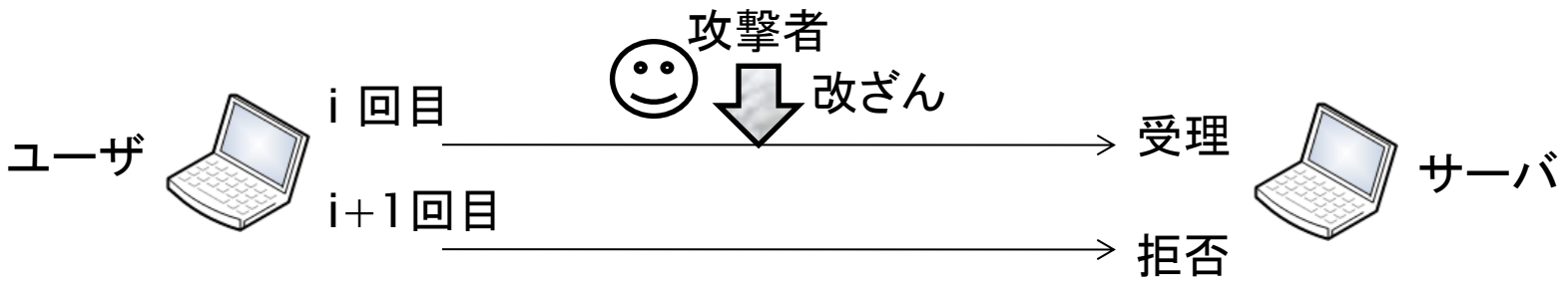
SV attack:
攻撃者が
・ある時点でサーバが保存している情報
・過去のセッションの通信データ
を入手しユーザになりすます攻撃

OTP認証方式に求められる安全性(2)

- 通信データが改ざんされたとしてもサーバが改ざんデータを受理しない

Denial-of-Service (DoS) attack に対する耐性

DoS attack:
攻撃者が
・過去のセッションの通信データ
を入手し、ある時点の通信データを改ざんして誤った情報をサーバに受理させ、次のセッションにおいてユーザからの接続要求を拒否させる攻撃



ProVerifによるOTP認証方式の形式化

- ProVerifは，i回目認証時にサーバ(ユーザ)が保持した値をi+1回目認証時に利用するという概念が扱えない

```
let processS(UID:id,x:sskey,h2SN1:bitstring)=
    in(c,(UID2':id,c12':bitstring,c22':bitstring));
    let h1xUID2=hash((x,UID)) in
    let c12''=hash(xor(h1xUID2,h2SN1)) in
    let h2SN2''=xor(c12',c12'') in
    let h3SN2'=hash(h2SN2'') in
    let h1SN1'=xor(c22',h3SN2') in
    if hash(h1SN1')=h2SN1 then
    if h2SN2'' <> h2SN2 then
    out(c,xor(h2SN1,h3SN2'));
```

サーバの処理を逐次的に記述

i回目認証時に保持したh2SN2''

||

i+1回目認証時のh2SN1

h2SN2''の値をi+1回目に渡せない

(例)ROSIプロトコルのサーバ処理

ProVerifによるOTP認証方式の形式化(2)

対応方法：

- サーバ(ユーザ)の処理を複数回分まとめて記述
  (複数回分を1単位とする)

```
let processS(UID:id,x:sskey,h2SN1:bitstring)=
    in(c,(UID2':id,c12':bitstring,c22':bitstring));
    let h1xUID2=hash((x,UID)) in
    let c12''=hash(xor(h1xUID2,h2SN1)) in
    let h2SN2''=xor(c12',c12'') in
    let h3SN2'=hash(h2SN2'') in
    let h1SN1'=xor(c22',h3SN2') in
    if hash(h1SN1')=h2SN1 then
    if h2SN2'' <> h2SN2 then
    out(c,xor(h2SN1,h3SN2'));
```

```
    in(c,(UID3':id,c13':bitstring,c23':bitstring));
    let h1xUID3=hash((x,UID)) in
    let c13''=hash(xor(h1xUID3,h2SN2'')) in
    let h2SN3''=xor(c13',c13'') in
    let h3SN3'=hash(h2SN3'') in
    let h1SN2'=xor(c23',h3SN3') in
    if hash(h1SN2')=h2SN2'' then
    out(c,xor(h2SN2'', h2SN3'')).
```

(例)ROSIプロトコルのサーバ処理(2回分をまとめて記述ver.)

ProVerifによるOTP認証方式の形式化(3)

- SV attackの検証
  - i回目認証時にSV attackを試みる場合，その直前(i−1回目認証終了時)にサーバに保存されている情報を故意に通信路に流出
    → この条件下で，攻撃者がサーバに対して正規ユーザになりすますことができるかを検証
    → i回目認証及びi+1回目認証を続けてパスできるかを検証
- DoS attackの検証
  - ProVerifにおける(通常の)認証の検証法を利用
    →i回目認証時にDos attackを試みる場合，攻撃者がi回目認証時の通信データを改ざんして誤った情報をサーバに受理させることができるかを検証
    →i回目認証をパスできる ⇒ Dos attack成功
    (i回目認証はパスできるがi+1回目認証はパスできない)

ROSIプロトコル

- Chienらによって提案されたOTP認証プロトコル[1]
- 辻らによりSV attackに対するぜい弱性の発見[2]
- DoS attackに対して耐性を持つ
- 登録フェーズ，認証フェーズが存在
- 相互認証

[1] H.Y.Chien and J.K.Jan, “Robust and Simple Authentication Protocol,” Computer Journal, vol.46, no.2, pp.193-201, 2003.
[2] T. Tsuji, and A. Shimizu, “One-Time Password Authentication Protocol aganist Theft Attacks,” IEICE Trans. Commun., vol.E87-B, no.3, pp.523-529,2004.

ROSIプロトコル：準備

- ID　：ユーザID
- P　：ユーザパスワード
- x　：サーバ秘密鍵
- h　：ハッシュ関数
- i　：セッション番号
- N_i　：i回目セッションの乱数
- ||　：結合
- ⊕　：排他的論理和
- VR^j_i: $h^j (P || N_i)$,
  - 例. $VR^2_3 = h^2 (P || N_3) = h(h(P || N_3))$

ROSIプロトコル：登録フェーズ

ユーザ

サーバ

Inputs ID,P,N_1

ID,P,N_1

$VR^2{}_1 = h(\, h(\, P||N_1)\,)$ を計算

(ID, x, $VR^2{}_1$ を保持)

$h(\, x \,||\, ID\,) \oplus P$,
$VR^1{}_1 = h(P||N1)$ を計算

$h(\, x \,||\, ID\,) \oplus P$, $VR^1{}_1$

Stores $h(\, x \,||\, ID\,) \oplus P$, $VR^1{}_1$

Stores ID, x, $VR^2{}_1$

＊登録フェーズはセキュアな通信路を用いて最初に一回だけ行われる

ROSIプロトコル：i回目認証フェーズ

ユーザ

Stored :h(x || ID) ⊕ P , VR^1_i

Inputs ID,P

N_{i+1}を生成

VR^2_i ,VR^2_{i+1} ,VR^3_{i+1} を計算

サーバ

Stored : ID, x , VR^2_i

ID

$\alpha_i = h(h(x || ID) \oplus VR^2_i) \oplus VR^2_{i+1}$

$\beta_i = VR^3_{i+1} \oplus VR^1_i$

h(x || ID) を計算

$VR^2_{i+1} \leftarrow \alpha_i \oplus h(h(x || ID) \oplus VR^2_i)$

$VR^1_i \leftarrow \beta_i \oplus h(VR^2_{i+1})$

(α_i, β_i から VR^2_{i+1},VR^1_i を取り出す)

$VR^2_i \stackrel{?}{=} h(VR^1_i)$:認証処理

$VR^2_i \oplus VR^3_{i+1}$

$VR^2_i \oplus VR^3_{i+1}$:認証処理

Stored: $VR^2_i \rightarrow VR^2_{i+1}$

Stored: $VR^1_i \rightarrow VR^1_{i+1}$

ROSIプロトコルに対する辻らの攻撃法

ユーザ

攻撃者

サーバ

Stored :ID, x, VR^2_i

サーバからID, x, VR^2_i を盗む

ID
$\alpha_i = h(h(x \,||\, ID) \oplus VR^2_i) \oplus VR^2_{i+1}$
$\beta_i = VR^3_{i+1} \oplus VR^1_i$

ID, α_i , β_i をインターセプト

$h(h(x \,||\, ID) \oplus VR^2_i)$ を計算

$VR^2_{i+1} \leftarrow \alpha_i \oplus h(h(x \,||\, ID) \oplus VR^2_i)$
$VR^1_i \leftarrow \beta_i \oplus h(VR^2_{i+1})$

α_i, β_i から VR^2_{i+1}, VR^1_i を取り出す

ROSIプロトコルに対する辻らの攻撃法(2)

ユーザ

攻撃者

サーバ

y (Atacker's key) を生成

N'_{i+1} を生成

$(VR^2_{i+1})' = h(h(y \,||\, N'_{i+1}))$

$(VR^3_{i+1})' = h((VR^2_{i+1})')$

Stored : ID, x , VR^2_i

ID

$\alpha_i' = h(h(x \,||\, ID) \oplus VR^2_i) \oplus (VR^2_{i+1})'$

$\beta_i' = (VR^3_{i+1})' \oplus VR^1_i$

$VR^2_i = h(VR^1_i)$:認証処理

$VR^2_i \oplus VR^3_{i+1}$:認証処理

$VR^2_i \oplus VR^3_{i+1}$

$VR^2_i \oplus (VR^3_{i+1})'$

$VR^3_{i+1} \leftarrow (VR^3_{i+1})'$

Stored: $VR^1_i \rightarrow VR^1_{i+1}$

Stored: $VR^2_i \rightarrow (VR^2_{i+1})'$

なりすましが可能で、以降 サーバは攻撃者を正規ユーザとして認証し続ける

ProVerifによって導出された攻撃法について

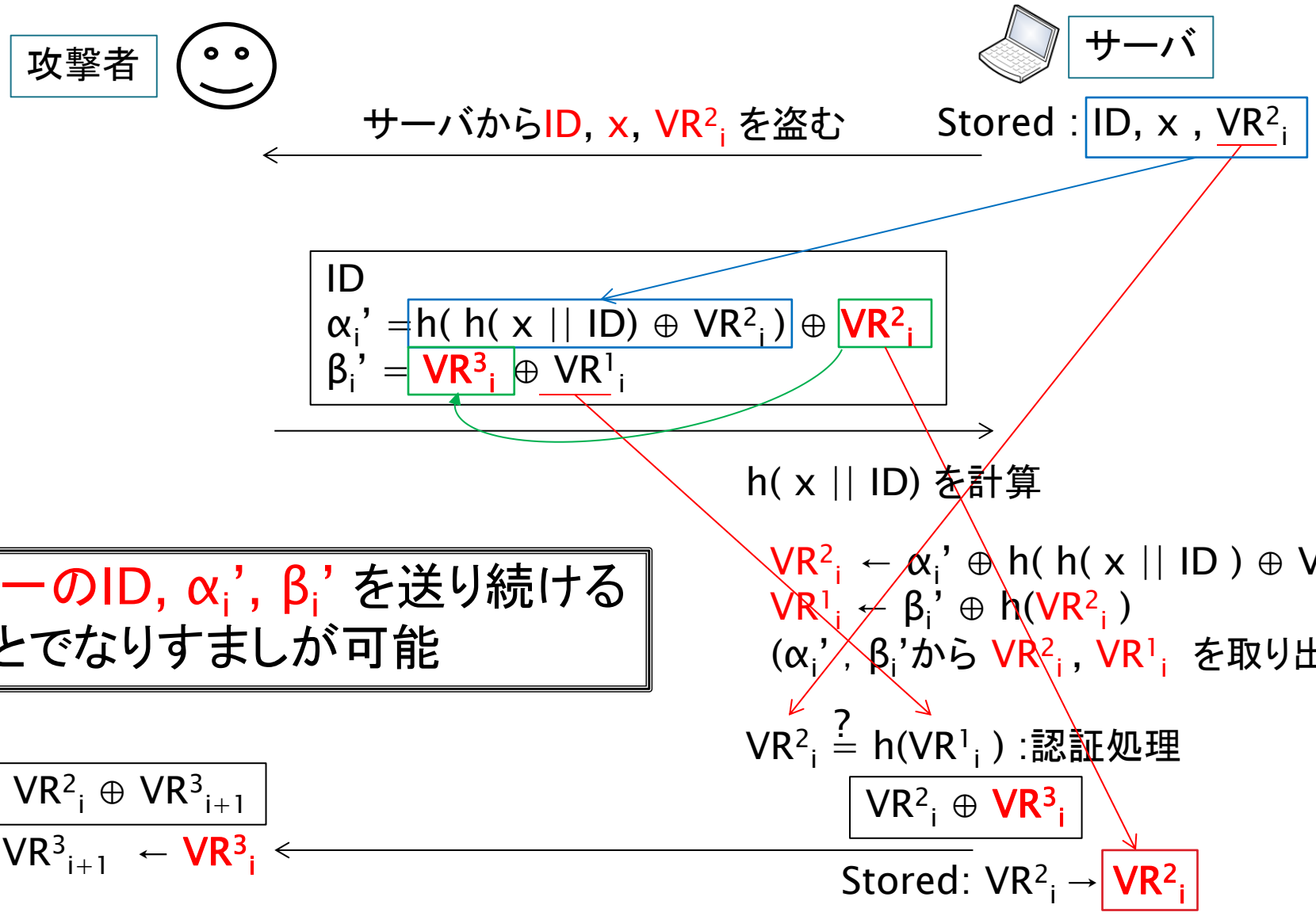
ProVerifによる検証

既存の攻撃法とは異なる攻撃法

辻らによる攻撃法

サーバの処理に条件文を加える

ProVerifによって導出された攻撃法



同一のID, α_i', β_i' を送り続けることでなりすましが可能

ProVerifによるROSIの形式的な検証

- 既存攻撃法とは異なるSV attackを検出
  - 同一の認証用情報を送り続けることでなりすましが可能
- 既存攻撃法も検出可能
- DoS Attackは不可能
- ProVerifによる検証結果と既知の研究結果が一致

SAS-X(2)プロトコル

- 辻らによって提案されたOTP認証プロトコル[3]
- 鵜尾らによりDoS attackに対するぜい弱性の発見[4]
- SV attackに対しては耐性を持つ
- 登録フェーズ，認証フェーズが存在
- 一方向認証

[3] T. Tsuji, T. Nakahara, and A. Shimizu, “A one time password authentication method,” IEICE Technical Report, OIS2005-83, vol.105, no.529, pp.23-28, 2006.
[4] 鵜尾健司，白石善明，森井昌克，“Stolen Verifier attack に耐性のあるワンタイムパスワード方式の評価と提案，” IPSJ Symposium Series，vol.2006，no.11，pp.543-548，2006.

SAS-X(2)プロトコル：準備

- ID　：ユーザID
- P　：ユーザパスワード
- h　：ハッシュ関数
- i　：セッション番号
- N_i　：i回目セッションの乱数
- ||　：結合
- ＋　：算術加算
- ⊕　：排他的論理和
- $E_{F_i}(M)$: 共通鍵暗号方式の暗号化関数, F_iを鍵としてMを暗号化
- $D_{F_i}(C)$: 共通鍵暗号方式の復号関数, F_iを鍵としてCを復号

SAS-X(2)プロトコル：登録フェーズ

ユーザ

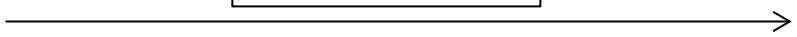
サーバ

Inputs ID,P

N_1, N_2, K を生成

$A_1 = h(ID \,||\, P \,||\, N_1)$
$F_1 = h(ID \,||\, A_1)$
$A_2 = h(ID \,||\, P \,||\, N_2)$
$F_2 = h(ID \,||\, A_2)$
$\gamma_1 = E_{F_2}(A_1)$ を計算

ID, K, F_1, γ_1



Stores A_1, F_1, A_2, F_2, K, N_2

Stores ID, K, F_1, γ_1

＊登録フェーズはセキュアな通信路を用いて最初に一回だけ行われる

SAS-X(2)プロトコル：i回目認証フェーズ

ユーザ

サーバ

Stored: A_i, F_i, A_{i+1}, F_{i+1}, K, N_{i+1}

Input : ID , P

N_{i+2} を生成

$A_{i+2} = h(ID \,||\, P \,||\, N_{i+2})$
$F_{i+2} = h(ID \,||\, A_{i+2})$
$\alpha_i = F_{i+1} \oplus F_i$
$\beta_i = (F_{i+1} + K) \oplus A_i$
$\gamma_{i+1} = E_{F_{i+2}}(A_{i+1})$ を計算

ID , α_i , β_i , γ_{i+1}

Stored: ID , K , F_i , γ_i

$F_{i+1} \leftarrow \alpha_i \oplus F_i$
$A_i \leftarrow \beta_i \oplus (F_{i+1} + K)$ を計算

$F_i \stackrel{?}{=} h(ID \,||\, A_i)$
$A_i \stackrel{?}{=} D_{F_{i+1}}(\gamma_i)$ により認証処理

Store:
$A_i \rightarrow A_{i+1}$, $F_i \rightarrow F_{i+1}$, $N_{i+1} \rightarrow N_{i+2}$,
$A_{i+1} \rightarrow A_{i+2}$, $F_{i+1} \rightarrow F_{i+2}$

Store: $F_i \rightarrow F_{i+1}$, $\gamma_i \rightarrow \gamma_{i+1}$

SAS-X(2)プロトコルに対する鵜尾らの攻撃法

ユーザ

攻撃者

サーバ

Stored: A_i, F_i, A_{i+1}, F_{i+1}, K, N_{i+1}

Input : ID , P

N_{i+2}を生成

$A_{i+2} = h(ID \,||\, P \,||\, N_{i+2})$

$F_{i+2} = h(ID \,||\, A_{i+2})$

$\alpha_i = F_{i+1} \oplus F_i$

$\beta_i = (F_{i+1} + K) \oplus A_i$

$\gamma_{i+1} = E_{F_{i+2}}(A_{i+1})$ を計算

送信されたγ_{i+1}を乱数γ'_{i+1}に改ざん

ID , α_i , β_i , γ'_{i+1}

Stored: ID, K, F_i, γ_i

$F_{i+1} \leftarrow \alpha_i \oplus F_i$

$A_i \leftarrow \beta_i \oplus (F_{i+1} + K)$ を計算

$F_i \stackrel{?}{=} h(ID \,||\, A_i)$

$A_i \stackrel{?}{=} D_{F_{i+1}}(\gamma_i)$ により認証処理

Store:

$A_i \rightarrow A_{i+1}$, $F_i \rightarrow F_{i+1}$, $N_{i+1} \rightarrow N_{i+2}$,

$A_{i+1} \rightarrow A_{i+2}$, $F_{i+1} \rightarrow F_{i+2}$

Store: $F_i \rightarrow F_{i+1}$, $\gamma_i \rightarrow \gamma'_{i+1}$

ProVerifによるSAS-X(2)の形式的な検証

- ProVerifによる検証
  - 鵜尾らによる攻撃法と同一のDoS attackが検出
- SV Attackは不可能
- ProVerifによる検証結果と既知の研究結果が一致

まとめと今後の課題

- ProVerifによるSAS-X(2)とROSIの安全性検証
  - 検証結果と既知の耐性評価が一致

プロトコル	SV attack 耐性	DoS attack耐性
SAS-X(2)	○	×
ROSI	×	○

表1 各プロトコルの既知の攻撃耐性評価

プロトコル	SV attack 耐性	DoS attack耐性
SAS-X(2)	○	×
ROSI	×	○

表2 ProVerifにより導出された攻撃耐性評価

ProVerifはDoS attack、SV attackを検出可能

OTP認証方式に対して有用

- 今後の課題：さまざまなOTP認証プロトコルの検証