

# 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<sup>\*</sup>，岩本智裕<sup>†</sup>，金子敏信<sup>\*</sup>

<sup>\*</sup>東京理科大学

<sup>†</sup>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

暗号プリミティブ(関数)  
・定義域, 値域, 性質  
etc

安全性特性  
・秘匿, 認証  
etc

検証対象の  
プロトコル

ホーン節

ProVerif (Resolutionアルゴリズム)

攻撃なし(true)

攻撃あり(false)

# Blanchetらのプロセス計算の 構文規則について

$M, N ::=$   
 $a, b, c, k, m, n, s$   
 $x, y, z$   
 $(M_1, \dots, M_k)$   
 $h(M_1, \dots, M_k)$   
 $M = N$   
 $M <> N$   
 $M \&\& M$   
 $M \parallel M$   
 $\text{not}(M)$

terms  
names  
variables  
tuple  
constructor/destructor application  
term equality  
term inequality  
conjunction  
disjunction  
negation

$P, Q ::=$   
 $0$   
 $P \mid Q$   
 $!P$   
 $\text{new } n : t; P$   
 $\text{in}(M, x : t); P$   
 $\text{out}(M, N); P$   
 $\text{if } M \text{ then } P \text{ else } Q$   
 $\text{let } x = M \text{ in } P \text{ else } Q$   
 $R(M_1, \dots, M_k)$

processes  
null process  
parallel composition  
replication  
name restriction  
message input  
message output  
conditional  
term evaluation  
macro usage

$T ::=$   
 $x : t$   
 $x$   
 $(T_1, \dots, T_n)$   
 $=M$

patterns  
typed variable  
variable without explicit type  
tuple  
equality test

拡張 $\pi$ 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  $\text{encrypt}(m,k)$

**Destructor**: 復号  $\text{decrypt}(c,k)$

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

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 8



# プロトコル表現 (ホーン節)

✓ 「メッセージ」 = terms

$M ::= x \mid f(M_1, \dots, M_n) \mid k[M_1, \dots, M_n]$   
 $\text{encrypt}(m, sk)$

✓ 「特性 (性質)」 = facts

$F ::= \text{attacker}(M)$

✓ 「プロトコル, 攻撃者」 = Horn clauses

$F_1 \wedge \dots \wedge F_n \Rightarrow F$

$\text{attacker}(m) \wedge \text{attacker}(sk) \Rightarrow$

$\text{attacker}(\text{encrypt}(m, sk))$

# 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  $\text{encrypt}(m, k)$



$\text{attacker}(m) \wedge \text{attacker}(k) \Rightarrow \text{attacker}(\text{encrypt}(m, k))$

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



$\text{attacker}(\text{encrypt}(m, k)) \wedge \text{attacker}(k) \Rightarrow \text{attacker}(m)$

# ProVerifにおけるホーン節について(2)

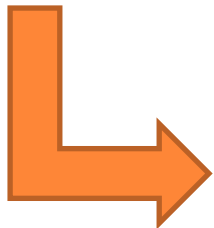
## ▶ ProVerifにおけるホーン節の例

### ▪ (例) プロトコル表現

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



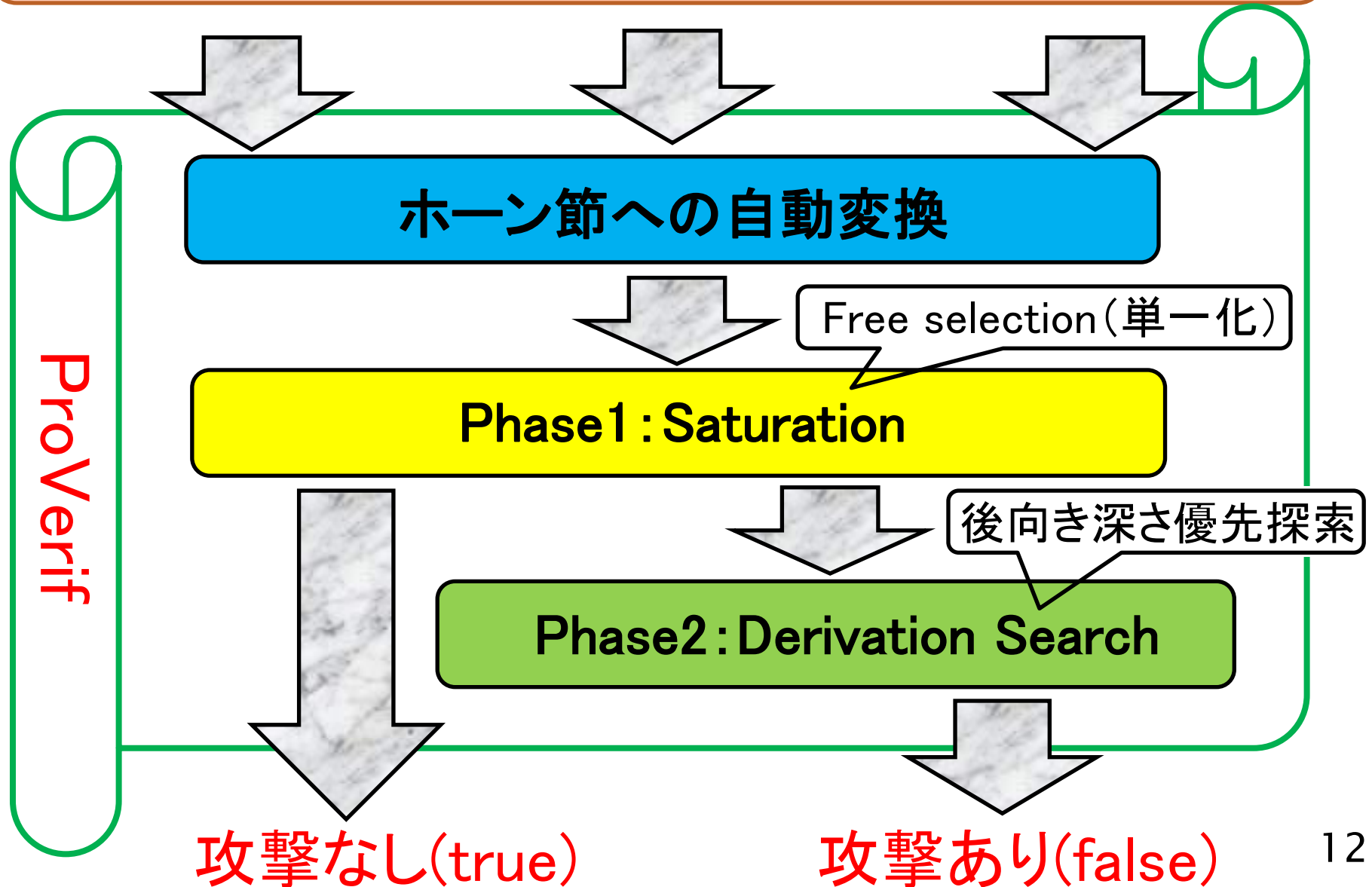
$\text{attacker}(\text{pencrypt}(\text{sign}(y, SK_A), PK_B)) \Rightarrow \text{attacker}(\text{encrypt}(s, y))$



攻撃者はBに対して  $\text{pencrypt}(\text{sign}(y, SK_A), PK_B)$  を送信し、Bの返信である  $\text{encrypt}(s, y)$  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

存在していない場合



なりすましが可能

存在している場合



認証true

# 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



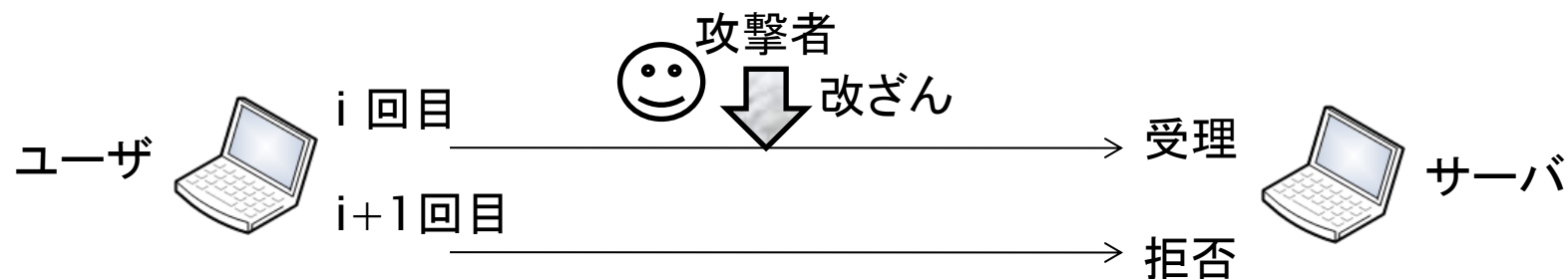
Denial-of-Service (DoS) attack に対する耐性

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'')).
```

# 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<sup>[1]</sup>
- ▶ 辻らによりSV attackに対するぜい弱性の発見<sup>[2]</sup>
- ▶ 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 against  
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
- ▶  $\oplus$  : 排他的論理和
- ▶  $VR_i^j$ :  $h^j ( P || N_i )$ ,
  - 例.  $VR_3^2 = h^2 ( P || N_3 ) = h( h( P || N_3 ) )$

# ROSIプロトコル：登録フェーズ



ユーザ

Inputs  $ID, P, N_1$



サーバ

$ID, P, N_1$

$VR^2_1 = h( h( P || N_1 ) )$  を計算

( $ID, x, VR^2_1$  を保持)

$h( x || ID ) \oplus P$  ,  
 $VR^1_1 = h(P || N_1)$  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 :  $h(x || ID) \oplus P, VR^1_i$   
 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

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

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

( $\alpha_i, \beta_i$  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

$VR^2_i \oplus VR^3_{i+1}$  : 認証処理

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

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<sup>2</sup><sub>i</sub>

←  
サーバからID, x, VR<sup>2</sup><sub>i</sub> を盗む

$$\begin{aligned} & \text{ID} \\ \alpha_i &= h( h( x \parallel \text{ID} ) \oplus \text{VR}^2_i ) \oplus \text{VR}^2_{i+1} \\ \beta_i &= \text{VR}^3_{i+1} \oplus \text{VR}^1_i \end{aligned}$$

→  
ID,  $\alpha_i$ ,  $\beta_i$  をインターセプト

$h( h( x \parallel \text{ID} ) \oplus \text{VR}^2_i )$  を計算

$$\begin{aligned} \text{VR}^2_{i+1} &\leftarrow \alpha_i \oplus h( h( x \parallel \text{ID} ) \oplus \text{VR}^2_i ) \\ \text{VR}^1_i &\leftarrow \beta_i \oplus h(\text{VR}^2_{i+1} ) \end{aligned}$$



$\alpha_i, \beta_i$  から  $\text{VR}^2_{i+1}, \text{VR}^1_i$  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,  $VR^2_i$

$$\begin{aligned} & \text{ID} \\ & \alpha'_i = h(h(x || \text{ID}) \oplus VR^2_i) \oplus (VR^2_{i+1})' \\ & \beta'_i = (VR^3_{i+1})' \oplus VR^1_i \end{aligned}$$

$VR^2_i = h(VR^1_i)$  : 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

```
graph TD; A[ProVerifによる検証] --> B[既存の攻撃法とは異なる攻撃法]; A --> C[辻らによる攻撃法]; C --> D[サーバの処理に条件文を加える];
```

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

攻撃者



サーバ

サーバからID, x,  $VR^2_i$  を盗む

Stored: ID, x,  $VR^2_i$

$$\begin{aligned}
 & \text{ID} \\
 & \alpha'_i = h(h(x \parallel \text{ID}) \oplus VR^2_i) \oplus VR^2_i \\
 & \beta'_i = VR^3_i \oplus VR^1_i
 \end{aligned}$$

$h(x \parallel \text{ID})$  を計算

$$\begin{aligned}
 VR^2_i & \leftarrow \alpha'_i \oplus h(h(x \parallel \text{ID}) \oplus VR^2_i) \\
 VR^1_i & \leftarrow \beta'_i \oplus h(VR^2_i) \\
 & (\alpha'_i, \beta'_i \text{ から } VR^2_i, VR^1_i \text{ を取り出す})
 \end{aligned}$$

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

$$VR^2_i \oplus VR^3_i$$

Stored:  $VR^2_i \rightarrow VR^2_i$

同一のID,  $\alpha'_i$ ,  $\beta'_i$  を送り続けることでなりすましが可能

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

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

# 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<sup>[3]</sup>
- ▶ 鵜尾らによりDoS attackに対するぜい弱性の発見<sup>[4]</sup>
- ▶ 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
- ▶  $\oplus$  : 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

$$A_1 = h(\text{ID} \parallel P \parallel N_1)$$

$$F_1 = h(\text{ID} \parallel A_1)$$

$$A_2 = h(\text{ID} \parallel P \parallel N_2)$$

$$F_2 = h(\text{ID} \parallel A_2)$$

$$Y_1 = E_{F_2}(A_1) \text{ を計算}$$

ID, K,  $F_1, Y_1$

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

Stores ID, K,  $F_1, Y_1$

\* 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

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

Input : ID , P

$N_{i+2}$  を生成

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

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

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

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

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

→  $ID, \alpha_i, \beta_i, Y_{i+1}$



サーバ

Stored: ID , K ,  $F_i, Y_i$

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

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

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

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

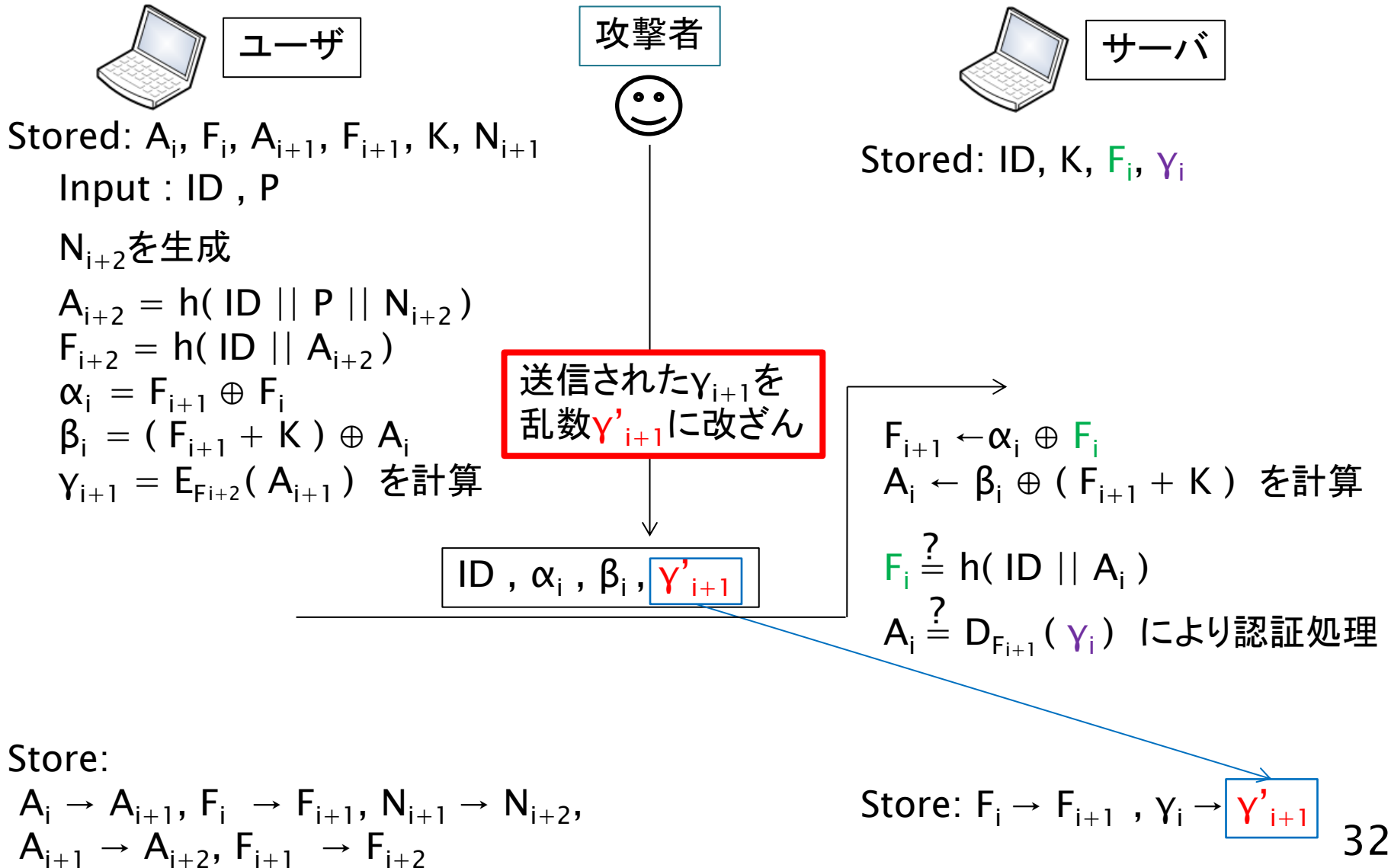
Store:

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

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

Store:  $F_i \rightarrow F_{i+1}, Y_i \rightarrow Y_{i+1}$

# SAS-X(2)プロトコルに対する鵜尾らの攻撃法





# 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