2005－CSEC－29（10）
2005／5／19

複数認証エージェントで利用可能な単一ワンタイムパスワード方式について

鵜尾 健司*　　大東 俊博*　　白石 善明†　　森井 昌克††

*徳島大学 工学部 知能情報工学科
〒 770–8506 徳島市南常三島町 2–1
{uo,ohigashi}@is.tokushima-u.ac.jp

† 近畿大学 理工学部 情報学科
〒 577–8502 東大阪市小若江 3–4–1
zenmei@info.kindai.ac.jp

†† 神戸大学 工学部 電気電子工学科
〒 657–8501 神戸市灘区六甲台 1–1
mmorii@kobe-u.ac.jp

あらまし 電子商取引の普及等に伴い，通信相手の認証は増々重要な技術となってきている．携帯電話や大量のアクセスを想定する場合，計算量を軽減できるワンタイムパスワード (OTP) が注目されている．OTP は 1 対 1 の認証を仮定しているが，通常，ユーザは複数のサービスを利用しており，それぞれのサーバは互いに独立であるため，ユーザはサーバごとに認証情報を登録している．本稿では認証サーバに代わる複数のエージェントが，ワンタイムパスワードを用いてユーザと認証を行う方式を提案する．エージェントは認証セッションごとにサーバから認証情報を受け取りユーザを認証する．提案方式はエージェントがユーザの認証情報を保持しない方式であり，かつサーバ，ユーザ，およびエージェントが OTP において登録時の秘密情報を保持しない実用性に優れた方式となっている．

A Single One-Time Password Method Usable by Multi-Authentication Agents

Kenji UO* ,　　Toshihiro OHIGASHI* ,　　Yoshiaki SHIRAISHI†, and Masakatu MORII††

*Dept. of Info. Sci. and Intell. Sys.,
The University of Tokushima,
Tokushima, 770–8506 Japan.
{uo,ohigashi}@is.tokushima-u.ac.jp

†Department of Informatics,
Kinki University,
Higashi-Osaka, 577–8502 Japan.
zenmei@info.kindai.ac.jp

††Department of Electrical and Electronics Engineering,
Kobe University,
Kobe, 657–8501 Japan.
mmorii@kobe-u.ac.jp

Abstract One-Time Password (OTP) method is a secure password-based authentication method by changing password in each session. A lot of OTP methods have been proposed, but most methods don't consider use of multi-authentication agents. The authorization model using multi-authentication agents achieves high usability and scalability. In this paper, we propose a single OTP method usable by multi-authentication agents. The proposed method can protect secret information of user from not only unknown opponents but also agents and other users. Additionally, the proposed method has high scalability because of independence of user from agent.

1　まえがき

電子商取引の普及や行政機能の電子化に伴い，通信相手を認証することは必要不可欠な技術として注目されている．認証方式には公開鍵暗号を用いる方式と共通鍵暗号や一方向性ハッシュ関数を用いる方式がある．公開鍵暗号を用いる方式は安全性が高いが多くの計算量が必要という問題点がある．共通鍵暗号や一方向性ハッシュ関数を用いる方式は公開鍵暗号方式と比べて計算量が格段に低いため，計算能力の低い端末でも実装できるという利点がある．共通鍵暗号や一方向性ハッシュ関数を用いた一般的な方式はパスワードを用いた認証方式である．各認証セッションごとにパスワードが変化する OTP 方式に関する研究が盛んに行われている [1, 2, 3, 4].

本稿では，認証のために用いるパスワードが変化するだけでなく，保持しておく秘密情報も使い捨てで変化する方式を OTP 方式と定義している．この OTP 方式では，基準となる最初の登録情報を指紋や虹彩等，バイオメトリック情報から導かれる ID 情報とした場合にその管理，安全性に優れた方式となっている．

インターネットで提供される個人情報を利用したサービスはクライアント・サーバ型であり，ユーザの認証に用いる秘密情報はサーバが保持している．通常，複数のサービスを利用するためには，ユーザはサーバ毎に秘密情報を記憶する必要がある．ユーザは利用するサービスが増えるに従って多くの秘密情報を管理しなければならない．さらに，ユーザは利用するサービスを増やすごとに秘密情報を登録する必要があり，スケーラビリティが高いとは言えない．そこで，ユーザと認証サーバの間にサービス毎の認証を行うエージェントを配置し，ユーザはエージェントと認証する方式が考えられる．複数エージェントを配置した認証方式を図 1 に示す．この方式では，複数のエージェントとサーバ間でユーザを認証する情報が管理されるため，ユーザは認証サーバへ登録した秘密情報だけで複数のエージェントと認証することが可能となる．サービスが増えてもユーザの秘密情報を新たに登録する必要なくスケーラビリティが改善される．本稿では，複数のエージェントを用いる認証を実現する OTP 方式について議論する．OTP 方式の中で複数のエージェントによる認証に対応可能な方式として，辻らによって提案された SAIFU[4] がある．OTP 方式で複数のエージェントによる認証を実現するためには，認証セッション毎に変わるユーザの認証情報をユーザが認証したいエージェントに渡す必要がある．SAIFU では認証セッション終了後に認証サーバがすべてのエージェントに対してユーザの認証情報を更新するための情報を送信する．そのため，認証サーバとエージェント間の通信量がエージェント数に比例して多くなってしまうという問題点がある．また，SAIFU はエージェントと認証サーバ間の回線が安全な通信路である場合の使用を前提に設計された方式であるため，インターネット上などの安全でない通信路では適用することは困難である．認証サーバの通信量を減少させるためには，ユーザが認証したいエージェントにのみ認証サーバからユーザの認証情報を配布する方法が考えられる．しかし，この方法ではエージェントと認証サーバ間で相互認証をする機能を備えていない場合はエージェントになりすましてユーザの認証情報を奪われる危険性があるため注意が必要である．また，正規のエージェント間でのなりすましを防ぐために認証サーバで秘密に保持しているユーザの秘密情報はユーザが認証したいエージェントに対しても漏洩しないような方式にすることが望まれる．

本稿では，実用的な複数のエージェントで認証可能な OTP 方式を提案する．提案方式では，エージェントは認証サーバからセッション毎に変化するユーザの認証情報を安全に受け取りユーザと認証を行う．エージェントとサーバ間の通信は OTP を用いて相互認証をする機能を備えているため第三者にユーザの認証情報が漏洩する危険性が少ない．また，ユーザと認証サーバだけが知っているユーザの秘密情報を認証セッション中で用いることにより認証をしているエージェント自身の不正行為も防げるようにしている．提案方式はユーザ，エージェントが共に単一の秘密情報を記憶していればよいため，ユーザ数，エージェント数の増加による認証方式全体の秘密情報の増加を抑えることができる．さらに，エージェントはユーザの認証情報を保持しておく必要がなく，他のエージェントとの情報の交換の必要もない．したがって，提案方式はエージェントの追加や削除が容易に行え，スケーラビリティの高い方式であると言える．また，提案方式はユーザは認証サーバと通信を直接する必要がないため，サーバを内部ネットワークに配置して直接アクセスできないようにしている環境でも動作するという利点もある．

2 記号の定義

本稿で用いる記号を次のように定義する．

U	ユーザ
ID	ユーザ ID
A	エージェント
No	エージェントを識別する番号
H	サーバ
E	攻撃者
SU	ユーザが登録するときだけ利用する秘密情報
SA	エージェントが登録するときだけ利用する秘密情報
M_i	ユーザとエージェントの使い捨て鍵と乱数をマスクしたもの
T_i	ユーザがエージェントを認証したことを示すもの
P_i	i 番目のセッションの認証子
KU_i	P_i を生成するために A と H が共有する使い捨て鍵
KA_i	A と H が相互認証するために A と H が共有する使い捨て鍵
CU_i	認証の成功をサーバがユーザへ保証するもの
CA_i	認証の成功をサーバがエージェントへ保証するもの
RU_i	i 回目の認証セッションでユーザが生成する擬似乱数
RA_i	i 回目の認証セッションでエージェントが生成する擬似乱数
RH_i	ユーザやエージェントが登録されるときに H が生成する擬似乱数
h	一方向性ハッシュ関数．$h(m)$ は m を一回ハッシュしたもの
$\oplus$	ビット単位の排他的論理和
$\|$	ビット列の連結
$A \longrightarrow B : X$	A は B に X を公開通信路で送信
$A \Longrightarrow B : X$	A は B に X を秘密通信路で送信

3 提案方式

提案方式は認証サーバに代わる複数のエージェントが OTP を用いてユーザと認証を行う認証方式である．提案方式ではひとつの OTP で複数のエージェントと認証ができる．提案プロトコルは登録処理と認証処理に分けられる．登録処理はユーザ登録とエージェント登録があり，それぞれ独立に初めに一度だけ行われる．認証処理はユーザがログインする度に毎回行われ，認証後ユーザ，エージェント，サーバのすべてで秘密情報が更新される．

3.1 登録処理

登録処理はユーザ登録とエージェント登録があり，ユーザ登録はユーザとサーバ間，エージェント登録はエージェントとサーバ間で独立に行われる．ユーザ登録手順を図 2，エージェント登録手順を図 3 に示す．

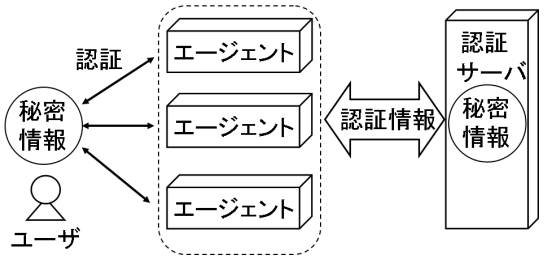

図 1: 複数エージェントを配置した認証方式

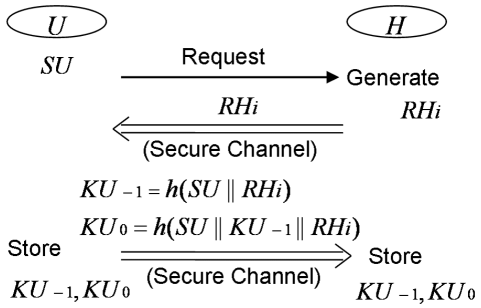

図 2: ユーザ登録手順

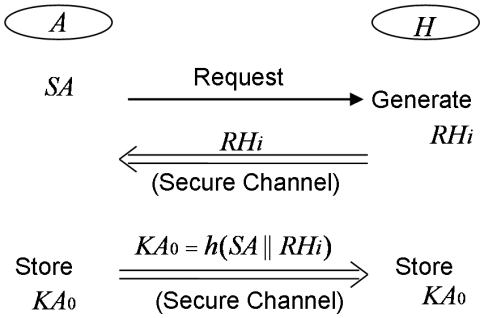

図 3: エージェント登録手順

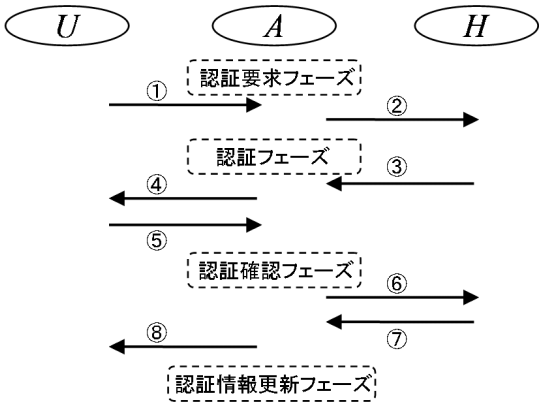

図 4: 認証処理

3.1.1 ユーザ登録

U は以下のようにユーザ登録を行う．

Step RU1 $U \longrightarrow H$:登録要求.

Step RU2 $H \Longrightarrow U$:RH_i .

Step RU3 U は以下のように KU_{-1} と KU_0 を計算し，保存．$KU_{-1} = h(\ SU \parallel RH_i)$，$KU_0 = h(\ SU \parallel K_{-1} \parallel\ RH_i)$.

Step RU4 $U \Longrightarrow H$:KU_{-1}，KU_0.

Step RU5 H は KU_{-1}，KU_0 を保存.

3.1.2 エージェント登録

A は以下のようにユーザ登録を行う．

Step RA1 $A \longrightarrow H$:登録要求.

Step RA2 $H \Longrightarrow U$:RH_i .

Step RA3 A は以下のように KA_0 を計算し，保存．$KA_0 = h(\ SA \parallel\ RH_i)$.

Step RA4 $A \Longrightarrow H$:KA_0.

Step RA5 H は KA_0 を保存.

3.2 認証処理

U の i 回目のログインに対して図 4 のように認証処理をしている．認証処理は 4 つのフェーズから構成されている．認証要求フェーズは，エージェントにユーザが認証するための情報を配布することを認証サーバに要求するフェーズである．認証サーバへの要求では，ユーザとエージェントの両方が正しい場合でなければ作ることができないデータを渡すことで第三者や他のエージェントやユーザによるなりすましを防いでいる．認証フェーズでは，ユーザと認証するための一時的な認証情報を認証サーバからエージェントへ安全に渡し，共有された認証情報を使ってエージェントとユーザが相互認証をする．認証サーバは，ユーザと認証するための一時的な認証情報が正しいエージェントのみによって復元できるような M_i をエージェントへ送信している．認証確認フェーズでは，ユーザとエージェントが正しく認証できたことを認証サーバが検証するフェーズである．ユーザがエージェントを正しく認証したことを認証サーバが検証するためのデータ T_i とエージェントがユーザを認証できたことを認証サーバが検証するためのデータ P_i を認証サーバへ送信する．P_i の中には T_i の情報も含まれており，ユーザの認証確認用の情報を中継するエージェントがユーザの確認情報を改ざんできなくしている．認証情報更新フェーズではユーザ，エージェント，認証サーバの使い捨て鍵を更新するフェーズである．各更新用の式は，通信路上を流れていない情報となっている．

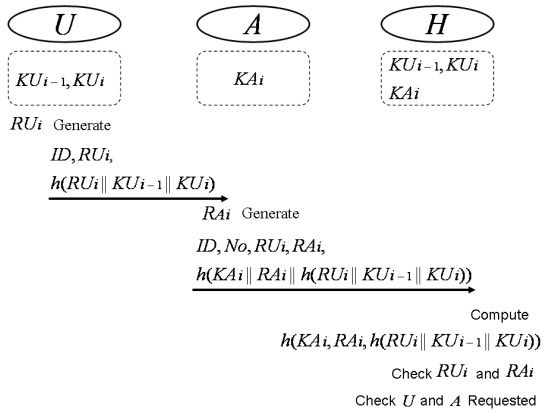

図 5: 認証要求フェーズ

3.2.1 認証要求フェーズ

認証要求フェーズは認証情報を要求してきたエージェントをサーバが認証するフェーズである．図 5 に示す認証要求フェーズの流れを以下に示す．

Step AR1 乱数 RU_i を生成．通信路での RU_i の改ざんを防ぐために RU_i を含んだ $h(RU_i \parallel KU_{i-1} \parallel KU_i)$ を計算．

Step AR2 $U \longrightarrow A$:ID, RU_i, $h(RU_i \parallel KU_{i-1} \parallel KU_i)$.

Step AR3 乱数 RU_i を生成．通信路での RA_i の改ざんの検知のために RA_i を含ませた $h(KA_i \parallel RA_i \parallel h(RU_i \parallel KU_{i-1} \parallel KU_i))$ を計算．

Step AR4 $A \longrightarrow H$:ID, No, RU_i, RA_i, $h(KA_i \parallel RA_i \parallel h(RU_i \parallel KU_{i-1} \parallel KU_i))$.

Step AR5 ID, No から U と A を識別し，自身が所有している KA_i を使って $h(KA_i \parallel RA_i \parallel h(RU_i \parallel KU_{i-1} \parallel KU_i))$ を計算し，通信相手が A であることを検証し，RU_i，RA_i を検証．

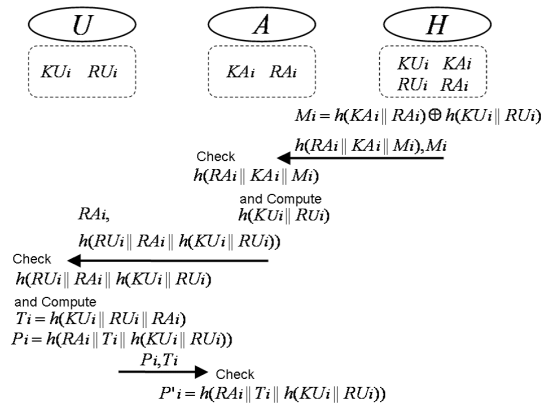

図 6: 認証フェーズ

3.2.2 認証フェーズ

認証フェーズでは H が認証した A に U の認証情報を送信し，A と U の間で相互認証を行う．図 6 に示す認証フェーズの流れを以下に示す．

Step AA1 U が生成できる $h(KU_i \| RU_i)$ を含んでいて A だけが取り出せる．M_i を作成．$M_i = h(KA_i \| RA_i\ ) \oplus h(KU_i \| RU_i)$．$M_i$ の改ざんを検知できるように M_i を含ませた $h(RA_i \parallel KA_i \parallel M_i)$，$M_i$ を作成．

Step AA2 $H \longrightarrow A$: $h(RA_i \parallel KA_i \parallel M_i)$，$M_i$).

Step AA3 A は受信した M_i と自身が所有している RA_i，KA_i を使って $h(RA_i \parallel KA_i \parallel M_i)$ と比較し H から送信されたのか検証．M_i の改ざんも検証．M_i と $h(KA_i \| RA_i\ )$ から $h(KU_i \| RU_i)$ を取り出す．

Step AA4 $A \longrightarrow U$:RA_i，$h(RU_i \parallel RA_i \parallel h(KU_i \parallel RU_i))$.

Step AA5 U は受信した RA_i と自身が所有している RU_i，KU_i を使って $h(RU_i \parallel RA_i \parallel h(KU_i \parallel RU_i))$ を計算．A から送信されたのか確認すると同時に RA_i の改竄を検証．A を認証したことを示すために $T_i = h(KU_i \parallel RU_i \parallel RA_i)$ を生成．$P_i = h(RA_i \parallel T_i \parallel h(KU_i \parallel RA_i))$ を計算．

Step AA6 $U \longrightarrow A$:T_i，P_i.

Step AA7 A は受信した T_i と自身が所有している RA_i，$h(KU_i \| RU_i)$ を使って P_i'= $h(RA_i \parallel T_i \parallel h(KU_i \parallel RA_i)$ を計算し，P_i'=P_i なら認証成功 P_i は T_i を含んでいるため T_i の改ざんも検証．

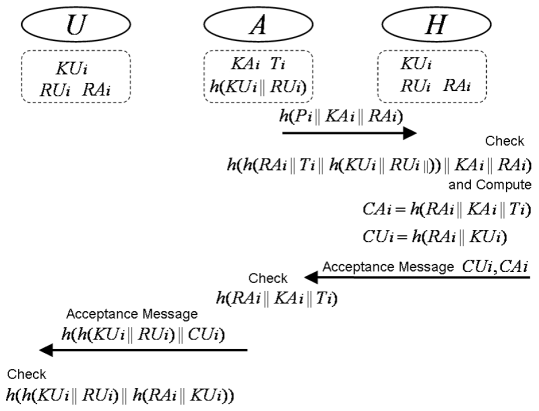

図 7: 認証確認フェーズ

3.2.3 認証確認フェーズ

認証確認フェーズでは認証が成功したことを H が確認し，A と U に伝える処理を行う．図 7 に示す認証確認フェーズの流れを以下に示す．

Step AV1 $A \longrightarrow H$: $h(P_i \parallel KA_i \parallel RA_i)$

Step AV2 Hは自身が所有しているKU_i, RU_i, RA_iを使って$T_i = h(KU_i \parallel RU_i \parallel RA_i)$, $P_i = h(RA_i \parallel T_i \parallel h(KU_i \parallel RU_i))$を計算．さらに$h(P_i \parallel KA_i \parallel RA_i)$を計算し，$U$と$A$の間で認証が成功したことを確認する．$H$が$U$と$A$の間の認証を確認したことを示すために$CA_i$=$h(RA_i \parallel KA_i \parallel T_i)$, CU_i=$h(RA_i \parallel KU_i)$を計算.

Step AV3 $H \longrightarrow A$:認証受理，CU_i，CA_i.

Step AV4 Aは自身が所有しているRA_i，KA_i，T_iを使って$h(RA_i \parallel KA_i \parallel T_i)$を計算し，$H$が認証を確認したことを検証する．$CA_i$を$A$が受け取って$U$へ送信したことを示すために$h(\ h(KU_i \parallel RU_i) \parallel CU_i)$を作成.

Step AV5 $A \longrightarrow U$:認証受理，$h(\ h(KU_i \parallel RU_i) \parallel CU_i)$

Step AV6 Uは自身が所有しているKU_i, RU_i, RA_iを使って$h(\ h(KU_i \parallel RU_i) \parallel h(RA_i \parallel KU_i))$を検証し，$A$から認証され，$H$が認証を確認したことを確認する．

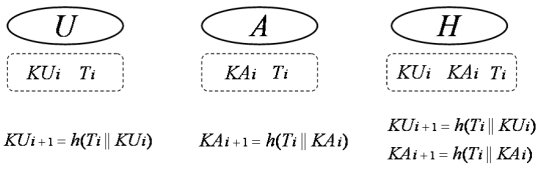

図 8: 認証情報更新フェーズ

3.2.4 認証情報更新フェーズ

認証情報更新フェーズではU，A，Hが次の認証セッションのために認証情報を更新する．図 8 に示す認証情報更新フェーズの流れを以下に示す.

Step AU1 認証情報の更新

U: $KU_{i+1} = h(T_i \parallel KU_i)$.

A: $KA_{i+1} = h(T_i \parallel KA_i)$.

H: $KU_{i+1} = h(T_i \parallel KU_i)$.
$KA_{i+1} = h(T_i \parallel KA_i)$

4 安全性の評価

代表的な攻撃手法である Man in the Middle attack , Replay attack，Denial of Service attack，A の不正行為についての耐性を考察することで提案方式の安全性を評価する.

4.1 Man in the Middle attack

Man in the Middle (MIM) attack とは，攻撃者 E が通信する 2 者間の間に割り込み，通信データの盗聴や改ざんを行う攻撃である．提案方式に対して MIM attack を適用することを考える．提案方式には U と A の間の通信と A と H の間の通信があるので，E が U と A の間，または A と H の間に割り込んで通信データを改ざんする危険性がある.

4.1.1 E が U と A の間に割り込む場合

U と A の間の通信で E に改ざんされると認証ができなくなる危険性のある情報は，A が送信する RA_i と U の送信する T_i である．提案方式はこれらの値を毎回検証するため，E の改ざんを検知できる.

4.1.2 E が A と H の間に割り込む場合

A と H の間の通信で E に改ざんされると認証ができなくなる危険性のある情報は，A が送信する RU_i, RA_i と U の送信する M_i である．提案方式はこれらの値を毎回検証するため，E の改ざんを検知できる.

4.2 Replay attack

Replay attack とは過去の認証セッションの通信データを E が盗聴し，再利用することでなりすましを行う攻撃方法である.

提案方式は，認証セッション終了時に U と A それぞれの使い捨て鍵 KU_i と KA_i を更新する．通信路へ送信する通信データは必ず RU_i もしくは RA_i を含んでいる．提案方式は認証セッションごとに変化する秘密情報と乱数を使って認証しているため一度通信路へ送信された通信情報を再利用して U になりすますことはできない．また，通信路へ送信する通信データからは U，A の秘密情報がわからないため，E が次の認証セッションの秘密情報を得ることはできない．以上より，提案方式は Replay attack に対して安全であると言える.

4.3 Denial of Service attack

Denial of Service(DoS) attack とは，E が通信する 2 者の正規の認証を不可能にする攻撃である．DoS attack を実現方法として，E が U，A，もしくは H に偽の認証情報を受理させることで誤った認証情報を登録させ，以降の認証セッションを不可能にする方法がある.

提案方式に対して DoS attack を適用するには，E が U，A もしくは H に偽の認証情報を受理させる必要がある．しかし，提案方式では U と A 間，A と H 間の通信はすべてセッションごとの使い捨て鍵である KU_i もしくは KA_i を使って検証される．KU_i もしくは KA_i を知らない E は偽の認証情報を受理させることができない．以上より提案方式は DoS attack に対して安全であると言える.

4.4 A の不正行為

提案方式は A が KAi を使って不正行為を行う危険性がある．H の行う不正行為は H や他の A 向けて行う行為や，認証要求を送信してくる U に対して行う行為が考えられる．H や他の A への不正行為として U からの認証要求がないのに不正に H から U の認証情報を取り出す行為や U のふりをして A へ認証を試みることが考えられる．認証要求を送信してくる U に対しては他の A のふりをして U を騙して認証させる行為が考えられる.

4.4.1 H や他の A への不正行為

U からの認証要求がないのに不正に H から U の認証情報を取り出したり，U のふりをして A へ認証を試みるためには，U が生成する $h(RU_i \parallel KU_{i-1} \parallel KU_i)$ を生成する必要がある．しかし，A は U の秘密情報である KU_{i-1}，KU_i を保持していないため $h(RU_i \parallel KU_{i-1} \parallel KU_i)$ を生成できない．また U の KU_i は認証セッションごとに更新されていくため再利用もできない．

4.4.2 U に対しての不正行為

提案方式では，登録されている A が攻撃者 E となる場合が考えられる．攻撃者は登録要求フェーズの **Step AR3** から自身の No と KA_i を用いてセッションを乗っ取ることができる．このとき，Uはセッションが乗っ取られたことに気づかず認証が完了してしまう．しかしながら，この不正行為によって Uはこのセッションだけサービスが拒否されるが，不正行為を行ったエージェントにユーザの秘密情報が洩れることはない．さらに，H と U の秘密情報の同期がずれることもない．したがって，それ以降のセッションでは通常どおりのサービスが利用できる．Eはセッションを乗っ取るときに自身の No と KA_i を用いているので，Hは不正行為を行った A がわかる．そのため U はサービス拒否が判明した時にサーバに問い合わせることで不正行為を行った A を知ることができる．サービス拒否の判明後，不正行為を行った A を特定されてしまうので不正行為の抑止力となる．このような A の不正行為は実装段階の工夫で防ぐことができる．しかしながら，ユーザに公開値である A の No を持たせることで防ぐことも可能である．一例として，認証要求フェーズの **Step AR1，AR2** で認証したい A の No を含むハッシュ値を計算するようにすれば No の改竄チェックになり登録している A の違法行為を未然に防ぐことができる．

5 既存方式との比較

OTP 方式の中で複数エージェントによる認証に対応可能な方式として，辻らによって提案された SAIFU がある．本節では通信量，保持しておく秘密情報の数，想定している通信路について提案方式と SAIFU を比較し，提案方式の方が実用性に優れた方式であることを明らかにする．また，エージェント導入時のスケーラビリティの高さについて示す．

SAIFU の通信では認証サーバからエージェントへの通信量が問題点として挙げられる．SAIFU は認証セッション終了後に全てのエージェントに対してユーザの認証情報を更新するための情報を送信する．もし，認証システム全体で n 個のエージェントを配置したとき，SAIFU を用いた場合に認証サーバからエージェントへの通信量は $O(n)$ になる．しかしながら，提案方式ではエージェント数に関わらず認証サーバはユーザが認証したいエージェントとのみ通信をすれば良いため通信量は $O(1)$ に抑えられる．同様にユーザ，エージェントからの通信量も $O(1)$ に抑えられるため，エージェントの多い大規模な認証システムでの使用を想定した場合，提案方式の方が実用的と言える．SAIFU では全てのエージェントはユーザの認証情報を保持しておかなければならない．もし，認証システム全体で m 人のユーザがいるとすれば，エージェントは $O(m)$ の秘密情報を保持する必要がある．更にエージェントの総数が n 台の場合，認証システム全体の秘密情報の総数は $O(m \cdot n)$ となる．しかしながら，我々の方式ではユーザとエージェントは自身の秘密情報だけを保持しておけば良いため秘密情報の数は $O(1)$ になる．認証サーバは全てのユーザとエージェントの秘密情報を保持する必要があるため $O(\max(m,n))$ の秘密情報を保持すれば良い．認証システム全体の秘密情報の総数を考えるとき，提案方式ではユーザとエージェントはそれぞれ $O(1)$ ずつ秘密情報を保持すればよいため，認証システム全体の秘密情報の総数は $O(\max(m,n))$ に抑えられる．これはユーザ，エージェントの数が多くなるような大規模な認証システムでの使用を想定した場合，提案方式は秘密情報の管理が比較的容易になるために実用的に優れていることを示している．SAIFU はエージェントと認証サーバ間の回線が安全な通信路である場合の使用を前提として設計されている．したがって，インターネットなどの安全でない通信路での使用を考えた場合，エージェントと認証サーバ間の安全性を保障する処理を新たに加えなければならず，直接利用することはできない．提案方式は初回登録以外の通信はすべて安全でない通信路を使うことを想定しているため，利用できる環境に制限が少ないという利点がある．以上のように提案方式は実用性に優れた方式である．

提案方式では新規にエージェントを設置する際はにユーザの認証情報を保持しておく必要がなく，他のエージェントとの認証情報の交換の必要もない．さらに，エージェントの追加は認証サーバで登録されているユーザの秘密情報に何の影響も与えない．したがって，提案方式でエージェントの追加や削除をする際には認証システムを停止する必要もないため非常に拡張性が高い方式と言える．

6 むすび

本稿では，実用的な複数のエージェントで認証可能な OTP 方式を提案した．提案方式では，エージェントは認証サーバからセッション毎に変化するユーザの認証情報を安全に受け取りユーザと認証を行う．エージェントとサーバ間の通信は OTP を用いて相互認証をする機能を備えているため第三者にユーザの認証情報が漏洩する危険性が少ない．さらに，エージェントはユーザの認証情報を保持しておく必要がなく，他のエージェントと情報の交換の必要もないのでスケーラビリティの高い方式である．

参考文献

[1] L. Lamport, “Password authentication with insecure communication,” Commun. ACM, vol.24, no.11, pp.770–772, Nov. 1981.

[2] N. Haller, “The S/Key(TM) one-time password system,” Proc. Internet Society Symposium on Network and Distributed System Security, pp.151–158, Feb. 1994.

[3] T. Tsuji, T. Kamioka, and A. Shimizu, “Simple And Secure password authentication protocol,ver.2(SAS-2),” IEICE Technical Report, OIS2002-30, vol.102, no.314, pp.7–11, Sep. 2002.

[4] T. Tsuji and A. Shimizu, “Secure Agreement Identification for Flexible Users(SAIFU)“ IEICE Technical Report, OIS2004-16, vol.104, no.238, pp.13–17,2004.