SSH パスワードユーザ認証の脆弱性とその考察

齋藤孝道[†1] 鬼頭利之[†2]
萩谷昌己[†3] 溝口文雄[†4]

ネットワーク通信における安全性確保のためのセキュリティプロトコルとして，SSH（Secure SHell）がある．サーバとクライアント間でのバルク通信に先立ち行われる SSH Handshake の過程で，サーバ認証の後，ユーザ認証を行う．ユーザ認証には，パスワードを用いた方式と公開鍵を用いた方式があり，そのどちらかを行うことにより，相互認証を達成し，安全な通信を行うことができるとされている．しかしながら，SSH パスワードユーザ認証には，プロトコルの設計上の問題により，脆弱性があることが確認できた．本論文では，SSH パスワードユーザ認証における攻撃を具体的に示したうえで，攻撃モデルに基づき，その攻撃の特徴を明確にする．さらに，その原因を特定し，修正を提案する．

On Architectural Defects of SSH User-authentication Protocols

Takamichi Saito,[†1] Toshiyuki Kito,[†2] Masami Hagiya[†3]
and Fumio Mizoguchi[†4]

Although some flaws have been found out in SSH (Secure SHell), there is not so much discussion about its architecture or design security. Therefore, in this paper, reconsidering the SSH handshake, we analyse some critical flaws of the SSH password-based user authentication, and show the reason by three types of attack modeles.

1. はじめに

SSH（Secure Shell）[1)~6)] は，盗聴や改竄，成りすましなどの脅威に晒された安全でないネットワークを介して，遠隔の計算機での仮想端末，コマンドの実行とファイルの転送を，安全に実現するソフトウェアとして広く利用されている．このように，通信における安全性☆を確保するプロトコルを，一般に，セキュリティプロトコルと呼び，SSH のほかに，SSL（Secure Socket Layer）や IPsec などがある．一方，CHAP（Challenge Handshake Authentication Protocol）や PAP（Password Handshake Authentication Protocol）のように，認証を行うプロトコルやその部位を，鍵交換とあわせたものも含み，認証プロトコルと呼び，本論文では，SSH の認証プロトコルの安全性を議論する．

SSH には，主に，バージョン 1[2)]（以降，SSH1 と呼ぶ）とバージョン 2[3)~7)]（以降，SSH2 と呼ぶ）の互換性のない 2 つのプロトコルバージョンがある．SSH1 と SSH2 ともに，プロトコルバージョンの交換，セッション鍵の交換，ユーザ認証という 3 つの手続きをこの順序で行い，SSH クライアントと SSH サーバとの間で，通信路を確立する（以降，この確立過程を SSH Handshake と呼ぶ）．SSH Handshake において，サーバ認証の後に行うユーザ認証にはパスワードを用いた方式（以降，パスワードユーザ認証と呼ぶ）と公開鍵を用いた方式（以降，公開鍵ユーザ認証と呼ぶ）があり，そのどちらかの利用により，SSH サーバと SSH クライアント（ユーザ）の間で相互認証を実現する．これにより，通信における真正性，機密性と完全性を確保した通信を行うことができるようになる．

しかしながら，SSH Handshake において，サーバ認証とユーザ認証の組合せによる相互認証の設計に問題があり，攻撃者によりパスワードを奪取され，成りすまし攻撃が実現可能であることが確認できた．パ

†1 明治大学
Meiji University
†2 株式会社東芝研究開発センター
Corporate Research & Development Center, TOSHIBA Corporation
†3 東京大学
The University of Tokyo
†4 東京理科大学
Tokyo University of Science

☆ ここでいう安全性とは，真正性，機密性，完全性からなる．

スワードユーザ認証の利用の問題点については，一般にも指摘されており，SSH の運用においては，「パスワードユーザ認証は行うべきでない」ともいわれている．パスワードユーザ認証へのこのような批判は，8 文字のパスワードを認証鍵とする場合と 1,024 bit の公開鍵を認証鍵として利用する場合では，明らかに認証鍵としての安全性に差異が生じることも，その理由の 1 つとして考えられる．しかし，本論文では，そのような鍵長の議論ではなく，前述したとおり，認証プロトコルの設計における脆弱性についてを議論する．特に，既存の「dsniff[10] による攻撃」と「本論文で新たに示す攻撃」を区別するために，「認証プロトコルにおける攻撃モデル」を用いて，その違いを明確にすることを目指す．そのうえで，モデルに基づいた原因を特定し，修正プロトコルを示す．ただし，本論文の議論は，5.2 節で示すとおり，PKI（Public Key Infrastructre）が対象とする，いわゆる「公開鍵のすり替え」問題でないことに，読者は十分に注意されたい．「公開鍵のすり替え」問題自体に関しては，すでに対策は知られており[11)~13)]，ここではむしろ，攻撃モデルの 1 つとして比較対象とする．

以降，2 章では，本論文で全体を通して用いる表記と SSH の鍵を含めた表記についての説明を行い，3 章で，SSH プロトコルについて，4 章で，攻撃過程の詳細を示し，5 章で，攻撃モデルを用いて，その攻撃の特徴と対処法を明らかにする．6 章で，パスワードユーザ認証の修正を示し，最後に，本論文をまとめる．

2. 準　　備

ここでは，本論文を通して用いる用語や記法についてまとめる．

2.1 SSH で用いる鍵について

ここでは，SSH で用いる，以下の 4 種類の鍵についての説明をする．

ホスト鍵：SSH1，SSH2 サーバによって，生成され利用される公開鍵．

サーバ鍵：SSH1 のみで用いられる一時的な公開鍵である．これは，一定の間隔（標準では，1 時間[1)]）で，SSH1 サーバによって更新される．

ユーザ鍵：SSH サーバによる公開鍵ユーザ認証のための公開鍵である．

セッション鍵：セッションごとに作成される共通鍵．

2.2 記　　法

主体に関して：

SSH サーバプログラムを実行する SSH サーバは S で表し，SSH クライアントプログラムを実行する SSH クライアントは C で表す．さらに，攻撃者は $\mathcal{I}$ によって表す．

通報に関して：

Msg は任意の通報を示す．ただし，返信確認の際の通報は，Ack とする．また，C が Msg を S に通報する場合，以下のように表す：

$$C \rightarrow S \ : \ Msg$$

さらに，C と S が Msg をお互いに通報し合うことを以下のように表す：

$$C \leftrightarrow S \ : \ Msg$$

次に，他の主体に成りすます攻撃者についての表記を示す．たとえば，主体 S に成りすます攻撃者 $\mathcal{I}$ が，主体 C に対して通報 Msg を送信する場合は，以下のように表す：

$$\mathcal{I}(S) \rightarrow C \ : \ Msg$$

暗号化と鍵に関して：

通報 Msg を鍵 K で暗号化したものを $\{Msg\}_K$ とし，公開鍵 P に対する秘密鍵 P^{-1} で署名されている通報 Msg を $\{Msg\}_{P^{-1}}$ と示す．さらに，通報 Msg が鍵 P と P' で暗号化されている場合には，$\{\{Msg\}_P\}_{P'}$ と書く．また，公開鍵 P_S は S のホスト鍵を示し，公開鍵 H_S は S のサーバ鍵を示す．また，鍵 P_C は，C を操作するユーザの公開鍵を示す．さらに，共通鍵 KS_{CS} もしくは KS_{SC} は，本論文では本質ではないので同一と見なし，S と C のセッション鍵を示す．同様に，当該セッションを識別するセッション識別子も S_{CS}，S_{SC} と表記し，この 2 つを区別しない．

SSH2 での鍵交換に関して：

鍵交換に関して，以下のとおりに定める[4)]．p は大きく安全な素数であり，g は，$GF(p)$ の部分群に対する生成元であり，q は部分群の位相である．また，V_S は S のバージョンであり，V_C は，C のバージョンを表す．I_C は，C のセッション鍵の交換の通報を表し，また，I_S は S のセッション鍵の交換の通報を表す．S と C によってそれぞれ生成される乱数を y, x $(1 < x, y < q)$ とそれぞれ表す．K_S は，$g^y \bmod p$ であり，K_C は，$g^x \bmod p$ である．ただし，K_S と K_C の選び方については，ここでは本質ではないので，議論から外す．さらに，セッション鍵を生成するための秘密情報を $g^{xy} \bmod p$ で表す．本論文では，これもセッション鍵 KS_{CS} と見なす．

ユーザ認証に関して：

$modulus$ は任意の公開鍵の一部もしくは全部を示す．$challenge$ は，S によって生成される 256 ビッ

トの乱数を表す．ただし，乱数の選び方についても，ここでは本質ではないので，議論から外す．また，*uname* と *password* はユーザのログイン名とパスワードをそれぞれ示す．特に，$modulus_C$（$uname_C$ や $password_C$）と書くとき，それらは C に関するのものである．

その他：

SA は，暗号化アルゴリズム，圧縮アルゴリズム，認証方法などの組である暗号スイートを示す．また，RN は乱数を表す．ただし，この乱数の選び方についても，ここでは本質ではないので，議論から外す．S_{CS} もしくは S_{SC} により S と C の当該セッションを識別する．$H(Msg_1)$ は，Msg_1 から生成されるハッシュ値を表す．同様に，$H(Msg_1, \ldots, Msg_n)$（ただし，$n > 1$）は，Msg_1〜Msg_n を結合したハッシュ値を示す．

3. SSH について

3.1 SSH Handshake の概要

最初に，「プロトコルバージョンの交換」において，S と C は，たとえば，`"SSH-1.5-1.2.22"` といった SSH バージョンに関する情報を通報し合い，同じバージョンをサポートしているかどうかを互いに確認する．

「セッション鍵の交換」において，S は，当該セッションの SA と P_S を C へ送信する．特に，SSH1 では，これらにあわせて，H_S も C へ送信する．送信されたホスト鍵 P_S が，C に保存されているホスト鍵と一致していないとき，ユーザはそのホスト鍵を受け入れるか否かを選択しなければならない．ユーザがそれを受け入れない場合，当該セッションは終了となる．

「ユーザ認証」では，S が，C を操作するユーザを認証する．本論文では，ユーザ認証方式として，SSH が提供するユーザ認証の中で，パスワードユーザ認証と公開鍵ユーザ認証を扱う．また，S と C は，当該セッションにおいて，同じユーザ認証方式の利用を，事前に合意する．

3.2 セッション鍵の交換

プロトコルバージョンの交換については，上記のとおりなので，割愛して，この節では，セッション鍵の交換について説明する．この手続きにおいて，セッション鍵の交換とともに，SSH クライアントは，SSH サーバを認証する．

3.2.1 SSH1 におけるセッション鍵の交換

$$
\begin{aligned}
&\text{(P1)} \quad S \to C : P_S, H_S, SA_S, RN \\
&\text{(P2)} \quad C \to S : SA_C, RN, \{\{KS_{CS}\}_{P_S}\}_{H_S} \\
&\text{(P3)} \quad S \to C : \{ack\}_{KS_{CS}}
\end{aligned}
\qquad \text{(P.3.1)}
$$

3.2.2 SSH2 におけるセッション鍵の交換

$$
\begin{aligned}
&\text{(P1)} \quad S \leftrightarrow C : SA_S, SA_C \\
&\text{(P2)} \quad C \to S : K_C \\
&\text{(P3)} \quad S \to C : K_S, P_S, \{H(V_C, V_S, I_C, I_S, P_S, K_C, K_S, KS_{CS})\}_{P_S^{-1}} \\
&\text{(P4)} \quad C \leftrightarrow S : \{KS_{CS}\}_{KS_{CS}}
\end{aligned}
\qquad \text{(P.3.2)}
$$

3.3 ユーザ認証

ユーザ認証において，セッション鍵 KS_{CS} を共有しているので，これにより，通報を暗号化する．

3.3.1 SSH1 におけるパスワードユーザ認証

$$
\begin{aligned}
&\text{(P1)} \quad C \to S : \{uname\}_{KS_{CS}} \\
&\text{(P2)} \quad S \to C : \{ack_1\}_{KS_{CS}} \\
&\text{(P3)} \quad C \to S : \{password\}_{KS_{CS}} \\
&\text{(P4)} \quad S \to C : \{ack_2\}_{KS_{CS}}
\end{aligned}
\qquad \text{(P.3.3)}
$$

3.3.2 SSH1 における公開鍵ユーザ認証

$$
\begin{aligned}
&\text{(P1)} \quad C \to S : \{uname\}_{KS_{CS}} \\
&\text{(P2)} \quad S \to C : \{ack_1\}_{KS_{CS}} \\
&\text{(P3)} \quad C \to S : \{modulus\}_{KS_{CS}} \\
&\text{(P4)} \quad S \to C : \{\{challenge\}_{P_C}\}_{KS_{CS}} \\
&\text{(P5)} \quad C \to S : \{H(challenge, S_{CS})\}_{KS_{CS}} \\
&\text{(P6)} \quad S \to C : \{ack_2\}_{KS_{CS}}
\end{aligned}
\qquad \text{(P.3.4)}
$$

3.3.3 SSH2 におけるパスワードユーザ認証

$$
\begin{aligned}
&\text{(P1)} \quad C \to S : \{uname, password\}_{KS_{CS}} \\
&\text{(P2)} \quad S \to C : \{ack\}_{KS_{CS}}
\end{aligned}
\qquad \text{(P.3.5)}
$$

3.3.4 SSH2 における公開鍵ユーザ認証

$$
\begin{aligned}
&\text{(P1)} \quad C \to S : \{uname, P_C, \{uname, P_C, S_{CS}\}_{P_C^{-1}}\}_{KS_{CS}} \\
&\text{(P2)} \quad S \to C : \{ack\}_{KS_{CS}}
\end{aligned}
\qquad \text{(P.3.6)}
$$

4. 攻撃過程

ここでは，SSH1 と SSH2 に対する攻撃手順を示す．この攻撃は，セッション鍵の交換とユーザ認証 2 段階を経て達成される．

4.1 SSH1 に対する攻撃

4.1.1 step 1：セッション鍵の交換

攻撃者 $\mathcal{I}$ は，SSH クライアント C と SSH サーバ S の間で，セッション鍵の交換をそれぞれ行う：

$$\begin{array}{lll}
\text{(P1)} & S \to \mathcal{I}(C) & : SA, RN, P_S, H_S \\
\text{(P1}') & \mathcal{I} \to C & : SA', RN', P_{\mathcal{I}}, H_{\mathcal{I}} \\
\text{(P2)} & C \to \mathcal{I} & : SA', RN', \{\{KS_{C\mathcal{I}}\}_{P_{\mathcal{I}}}\}_{H_{\mathcal{I}}} \\
\text{(P2}') & \mathcal{I}(C) \to S & : SA, RN, \{\{KS_{\mathcal{I}S}\}_{P_S}\}_{H_S} \\
\text{(P3)} & S \to \mathcal{I}(C) & : \{ack\}_{KS_{\mathcal{I}S}} \\
\text{(P3}') & \mathcal{I} \to C & : \{ack\}_{KS_{C\mathcal{I}}}
\end{array} \quad \text{(P.4.1)}$$

SSH クライアント C におけるユーザが，正規のサーバのふりをした攻撃者 $\mathcal{I}$ に接続をし，$\mathcal{I}$ が S に接続した後，S は $\mathcal{I}$ に通報を送信する (P1)．$\mathcal{I}$ はそれらを受け取ると，自身の公開鍵 $P_{\mathcal{I}}$ を含めた独自の通報を C に送信する (P1′)．その後，C は，C と $\mathcal{I}$ との通信路のセッション鍵 $KS_{C\mathcal{I}}$ を生成し，それを暗号化し，$\mathcal{I}$ に送信する (P2)．それを受け取った $\mathcal{I}$ は，$\mathcal{I}$ と S とのセッション鍵 $KS_{\mathcal{I}S}$ を同様に生成し，それを暗号化し，S に送信する (P2′)．その後，S は $KS_{\mathcal{I}S}$ を共有したことを示す通報を送信する (P3)．また，$\mathcal{I}$ も，同様に，$KS_{C\mathcal{I}}$ を共有したことを示す通報を送信する (P3′)．

ここまでで，$KS_{C\mathcal{I}}$ は，C と $\mathcal{I}$ との間で共有され，同様に，$KS_{\mathcal{I}S}$ は，$\mathcal{I}$ と S との間で共有される．

4.1.2 step 2：ユーザ認証

前 4.1.1 項に続き，以下のいずれかのユーザ認証を行う．

4.1.2.1 パスワードユーザ認証の場合

このとき，以下のように，$\mathcal{I}$ が C に成りすます攻撃が成立する：

$$\begin{array}{lll}
\text{(P1)} & C \to \mathcal{I} & : \{uname_C\}_{KS_{C\mathcal{I}}} \\
\text{(P1}') & \mathcal{I}(C) \to S & : \{uname_C\}_{KS_{\mathcal{I}S}} \\
\text{(P2)} & S \to \mathcal{I}(C) & : \{ack_1\}_{KS_{\mathcal{I}S}} \\
\text{(P2}') & \mathcal{I} \to C & : \{ack_1\}_{KS_{C\mathcal{I}}} \\
\text{(P3)} & C \to \mathcal{I} & : \{password_C\}_{KS_{C\mathcal{I}}} \\
\text{(P3}') & \mathcal{I}(C) \to S & : \{password_C\}_{KS_{\mathcal{I}S}} \\
\text{(P4)} & S \to \mathcal{I}(C) & : \{ack_2\}_{KS_{\mathcal{I}S}} \\
\text{(P4}') & \mathcal{I} \to C & : \{ack_2\}_{KS_{C\mathcal{I}}}
\end{array} \quad \text{(P.4.2)}$$

C が，$\mathcal{I}$ に，セッション鍵 $KS_{C\mathcal{I}}$ で暗号化されたログイン名 $uname_C$ を送信する (P1)．C に成りすますため，$\mathcal{I}$ は $uname_C$ を $KS_{\mathcal{I}S}$ で暗号化して送信する (P1′)．それを受信した S は，$uname_C$ が存在するかどうかを確認する．そのうえで，S は，クライアントの次の通報を促すために ack_1 を送信する (P2)．$\mathcal{I}$ は，それをセッション鍵 $KS_{\mathcal{I}S}$ を用いて復号し，$KS_{C\mathcal{I}}$ で暗号化した ack_1 を送信する (P2′)．その後，C は $KS_{C\mathcal{I}}$ で暗号化されたパスワードを $\mathcal{I}$ に送信する (P3)．この時点で，$\mathcal{I}$ はユーザのパスワードを得る．$\mathcal{I}$ は続けて，$KS_{\mathcal{I}S}$ で暗号化されたパスワードを S へ送信する (P3′)．この時点で，S は，$\mathcal{I}$ を，C であると判断をする．最後に，S は ack_2 を $\mathcal{I}$ に送信し (P4)，$\mathcal{I}$ は同様にそれを C に転送する (P4′)．$\mathcal{I}$ は，C に感知されないようにするためにセッションを張りつづける．

したがって，SSH1 パスワードユーザ認証においては，攻撃が成り立ち，パスワードが奪取されてしまう．

4.1.2.2 公開鍵ユーザ認証の場合

ここでは，SSH1 公開鍵ユーザ認証に対して同じ攻撃を考えてみる：

$$\begin{array}{lll}
\text{(P1)} & C \to \mathcal{I} & : \{uname_C\}_{KS_{C\mathcal{I}}} \\
\text{(P1}') & \mathcal{I}(C) \to S & : \{uname_C\}_{KS_{\mathcal{I}S}} \\
\text{(P2)} & S \to \mathcal{I}(C) & : \{ack_1\}_{KS_{\mathcal{I}S}} \\
\text{(P2}') & \mathcal{I} \to C & : \{ack_1\}_{KS_{C\mathcal{I}}} \\
\text{(P3)} & C \to \mathcal{I} & : \{modulus_C\}_{KS_{C\mathcal{I}}} \\
\text{(P3}') & \mathcal{I}(C) \to S & : \{modulus_C\}_{KS_{\mathcal{I}S}} \\
\text{(P4)} & S \to \mathcal{I}(C) & : \{\{challenge\}_{P_C}\}_{KS_{\mathcal{I}S}} \\
\text{(P4}') & \mathcal{I} \to C & : \{\{challenge\}_{P_C}\}_{KS_{C\mathcal{I}}} \\
\text{(P5)} & C \to \mathcal{I} & : \{H(challenge, S_{C\mathcal{I}})\}_{KS_{C\mathcal{I}}}
\end{array} \quad \text{(P.4.3)}$$

(P1)〜(P2′) は，パスワード認証方式と同じなので説明は省略する．C は $modulus_C$ を送信する (P3)．$\mathcal{I}$ は，それを復号し，$modulus_C$ を送信する (P3′)．それを受信した S は $challenge$ を生成し，P_C でそれを暗号化し，$\mathcal{I}$ に返信する (P4)．$challenge$ は，P_C で暗号化されているので，$\mathcal{I}$ は $challenge$ を復号して獲得することはできない．そのため，$\mathcal{I}$ は，たとえば，$\{challenge\}_{P_C}$ を，そのまま，C に送信する (P4′)．その後，C は暗号化された通報を P_C^{-1} で復号して，$challenge$ と $S_{C\mathcal{I}}$ を結合したハッシュ値を計算し，それを $\mathcal{I}$ に送信する (P5)．$\mathcal{I}$ と S は，セッション識別子 $S_{\mathcal{I}S}$ を共有しているので，このとき，$\mathcal{I}$ は，$\{H(challenge, S_{\mathcal{I}S})\}_{KS_{\mathcal{I}S}}$ という通報を作成しなければならない．しかしながら，これを作成することができない．よって，$\mathcal{I}$ が接続を続ける場合，S は攻撃者の存在を検知することができる．

したがって，SSH1 公開鍵ユーザ認証は，この攻撃

を防ぐことができる．

4.2 SSH2 に対する攻撃

SSH2 に対する攻撃手順を示す．

4.2.1 step 1：セッション鍵の交換の手続き

4.1.1 項と同様に，攻撃者 $\mathcal{I}$ は，SSH クライアント C と SSH サーバ S との間で，セッション鍵の交換をそれぞれ行う：

$$
\begin{array}{ll}
\text{(P1)} & S \leftrightarrow \mathcal{I}(C) \quad : SA \\
\text{(P1}') & \mathcal{I} \leftrightarrow C \quad : SA' \\
\text{(P2)} & C \rightarrow \mathcal{I} \quad : K_C \\
\text{(P2}') & \mathcal{I}(C) \rightarrow S \quad : K_{\mathcal{I}} \\
\text{(P3)} & S \rightarrow \mathcal{I}(C) \quad : K_S, P_S, \{H(V_{\mathcal{I}}, V_S, \\
 & \quad I_{\mathcal{I}}, I_S, P_S, K_{\mathcal{I}}, K_S, KS_{\mathcal{I}S})\}_{P_S^{-1}} \\
\text{(P3}') & \mathcal{I} \rightarrow C \quad : K_{\mathcal{I}}, P_{\mathcal{I}}, \{H(V_C, V_{\mathcal{I}}, \\
 & \quad I_C, I_{\mathcal{I}}, P_{\mathcal{I}}, K_C, K_{\mathcal{I}}, KS_{C\mathcal{I}})\}_{P_{\mathcal{I}}^{-1}} \\
\text{(P4)} & C \rightarrow \mathcal{I} \quad : \{KS_{C\mathcal{I}}\}_{KS_{C\mathcal{I}}} \\
\text{(P4}') & \mathcal{I}(C) \rightarrow S \quad : \{KS_{\mathcal{I}S}\}_{KS_{\mathcal{I}S}}
\end{array}
\qquad \text{(P.4.4)}
$$

「C と $\mathcal{I}$」と「C に成りすました攻撃者 $\mathcal{I}$ と S」との間で，通報をそれぞれ交換する (P1，P1′)．次に，C は K_C を求め，$\mathcal{I}$ にそれを送信する (P2)，同様に，$\mathcal{I}$ も $K_{\mathcal{I}}$ を求めて，S にそれを送信する (P2′)．それを受信した S は，セッション鍵 $KS_{\mathcal{I}S}$ を求める．そのうえで，$V_{\mathcal{I}}$，V_S，$I_{\mathcal{I}}$，I_S，P_S，$K_{\mathcal{I}}$，K_S，$KS_{\mathcal{I}S}$ を結合し，そのハッシュ値からセッション識別子 $S_{\mathcal{I}S}$ を求める．さらに，S は，K_S，P_S とともに，秘密鍵 P_S^{-1} で電子署名したハッシュ値を送信する (P3)．同様に，$\mathcal{I}$ は，セッション鍵 $KS_{C\mathcal{I}}$ などを求め，$K_{\mathcal{I}}$，$P_{\mathcal{I}}$ とともに，$P_{\mathcal{I}}^{-1}$ で電子署名したハッシュ値を送信する (P3′)．それらを受信した C は，保存されている公開鍵と，公開鍵 $P_{\mathcal{I}}$ が，同じであるかどうかを確認する．それが等しいとき，C はセッション鍵 $KS_{C\mathcal{I}}$ を求め，$P_{\mathcal{I}}$ を用いて電子署名を検証する．最後に，C と $\mathcal{I}$，(C のふりをした) $\mathcal{I}$ と S は，セッション鍵が共有されていることを確認する（手続き 4，4′）．

ここまでで，4.1.1 項と同様に，$S_{C\mathcal{I}}$ と $KS_{C\mathcal{I}}$ は，C と $\mathcal{I}$ との間で共有され，同様に，$S_{\mathcal{I}S}$ と $KS_{\mathcal{I}S}$ は，$\mathcal{I}$ と S との間で共有される．

4.2.2 step 2：ユーザ認証の手続き

前 4.2.1 項に続き，以下のいずれかのユーザ認証を行う．

4.2.2.1 パスワードユーザ認証の場合

このとき，以下のように，$\mathcal{I}$ が C に成りすます攻撃が成立する：

$$
\begin{array}{ll}
\text{(P1)} & C \rightarrow \mathcal{I} : \\
 & \{uname_C, password_C\}_{KS_{C\mathcal{I}}} \\
\text{(P1}') & \mathcal{I}(C) \rightarrow S : \\
 & \{uname_C, password_C\}_{KS_{\mathcal{I}S}} \\
\text{(P2)} & S \rightarrow \mathcal{I}(C) \quad : \{ack\}_{KS_{\mathcal{I}S}} \\
\text{(P2}') & \mathcal{I} \rightarrow C \quad : \{ack\}_{KS_{C\mathcal{I}}}
\end{array}
\qquad \text{(P.4.5)}
$$

ユーザ認証の手続きでは，SSH クライアント C は，セッション鍵 $KS_{C\mathcal{I}}$ で暗号化された $uname_C$ と $password_C$ を，正規のサーバに成りすました攻撃者 $\mathcal{I}$ に送信する (P1)．この通報を受信した $\mathcal{I}$ はユーザのパスワードを手に入れる．さらに，正規の SSH サーバに成りすますために，$\mathcal{I}$ は接続を続けて，S に $uname_C$ と $password_C$ を送信する (P1′)．それを受信した S は，保存された情報を用いて，$uname_C$ と $password_C$ を確認する．最後に，S は認証が成功したか失敗したかを示す ack を送信する（手続き 2)．同様に，$\mathcal{I}$ はその通報を C に送信する（手続き 2′)．

したがって，SSH2 パスワードユーザ認証においては，攻撃が成り立ち，パスワードが奪取されてしまう．

4.2.2.2 公開鍵ユーザ認証の場合

ここでは，SSH2 公開鍵ユーザ認証に対して同じ攻撃を考えてみる：

$$
\begin{array}{ll}
\text{(P1)} & C \rightarrow \mathcal{I} \quad : \{uname_C, P_C, \\
 & \{uname_C, P_C, S_{C\mathcal{I}}\}_{P_C^{-1}}\}_{KS_{C\mathcal{I}}}
\end{array}
\qquad \text{(P.4.6)}
$$

C は，$uname_C$，P_C にあわせて，C の秘密鍵 P_C^{-1} で電子署名された $\{uname_C, P_C, S_{C\mathcal{I}}\}$ を，正規のサーバに成りすました $\mathcal{I}$ に送信する (P1)．次の通報において，攻撃者 $\mathcal{I}$ は C に成りすますために，S に対して，$\{uname_C, P_C, S_{\mathcal{I}S}\}_{P_C^{-1}}$ を送信しなければならない．しかしながら，$\mathcal{I}$ は C の秘密鍵 P_C^{-1} を持っていないので，その通報を作成できない．

したがって，SSH2 公開鍵ユーザ認証は，この攻撃を防ぐことができる．

5. 攻撃モデルによる分類

5.1 攻撃モデルについて

ここでは，認証プロトコルにおいて想定される攻撃モデルを用いて，4.1，4.2 節で示した攻撃とその他の攻撃とを分類し，違いを明確にする．

ここでの攻撃モデルには，認証プロトコルにおけるイニシエータとレスポンダという 2 つの正規の主体があり，さらに，それらに攻撃を行う攻撃者がいる．ここでは，イニシエータは，認証プロトコルの最初に，

要求を出した通信主体とする．

認証の際，イニシエータとレスポンダの一方もしくは両者が，攻撃者によって，成りすまされている可能性がある．すわわち，本来，任意のあるイニシエータ A は，任意のあるレスポンダ B と接続していると想定しており，レスポンダ B は，イニシエータ A と接続していると想定している．しかしながら，認証プロトコルの設計上の不備があるとき，その意図を違えることが可能である場合があり，それが可能である場合，その認証プロトコルを脆弱であると見なすことができる．

実際の TCP/IP アプリケーション（SSH 通信など）における接続と，イニシエータとレスポンダの意図のずれの組合せを考慮すると，以下の 3 つの組合せがあり，それにより攻撃を分類する：

(1) MITM（Man In The Middle）攻撃モデル
(2) レスポンダを騙す攻撃モデル
(3) イニシエータを騙す攻撃モデル

このモデルに基づいた攻撃シナリオを適宜作成したうえで，認証プロトコルの安全性を検査するツールが，齋藤によって提案されている[8]．

以降では，この 3 つについて，その概念と具体例をそれぞれ説明する．

5.2 MITM 攻撃モデル

5.2.1 MITM 攻撃

いわゆる MITM 攻撃では，イニシエータ A とレスポンダ B は，互いに他方へと接続する意思があり，そのように試みるが，実際には，接続を攻撃者 $\mathcal{I}$ に奪われ，攻撃者 $\mathcal{I}$ 経由でのデータ通信を余儀なくされる（**図 1**）．一般的には，適切に，相互認証を行い，暗号鍵によるセッションの保護をすることで，この攻撃を防ぐことができる．

そのような認証プロトコルの設計上の問題とは別に，「公開鍵のすり替え」があるときも，5.2.2 項で示すように，この攻撃が成り立つ．

5.2.2 MITM 攻撃の例

SSH1 のセッションのハイジャックをして，パスワードを奪取を実現する dsniff というソフトウエアがある[10]．このソフトウエアは `dnsspoof` と `sshmitm` というプログラムからなる．前者を用いて，攻撃者 $\mathcal{I}$ は，DNS（Domain Name System）サーバが返す情報を書き換え，後者を用いて，SSH クライアント C から SSH サーバ S への接続を奪い取り，自身の SSH サーバ（`sshmitm`）に接続させ，自身の公開鍵 $P_{\mathcal{I}}$ を送り込むことができる．このように，この攻撃は「公開鍵のすり替え」により，互いに，実際の接続相手と意図とを違わせているため，MITM 攻撃に分類できる．

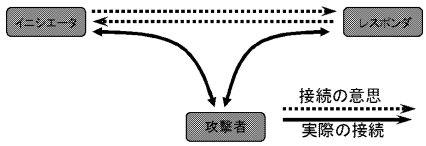

図 1 一般的な MITM 攻撃
Fig. 1 Standard MITM attack.

これにより，$\mathcal{I}$ はパスワードの奪取を達成する．ここで，$\mathcal{I}$ は，適宜，SSH サーバや SSH クライアントに成りすましていることに注意されたい．

$$
\begin{array}{lll}
\text{(P1)} & S \to \mathcal{I}(C) & : P_S, H_S, SA, RN_S \\
\text{(P1}') & \mathcal{I}(S) \to C & : P_{\mathcal{I}}, H_{\mathcal{I}}, SA, RN_{\mathcal{I}} \\
\text{(P2)} & C \to \mathcal{I}(S) & : \\
 & & \quad SA, RN_C, \{\{KS_{C\mathcal{I}}\}_{P_{\mathcal{I}}}\}_{H_{\mathcal{I}}} \\
\text{(P2}') & \mathcal{I}(C) \to S & : \\
 & & \quad SA, RN_{\mathcal{I}}, \{\{KS_{\mathcal{I}S}\}_{P_S}\}_{H_S} \\
\text{(P3)} & S \to \mathcal{I}(C) & : \{ack\}_{KS_{\mathcal{I}S}} \\
\text{(P3}') & \mathcal{I}(S) \to C & : \{ack\}_{KS_{C\mathcal{I}}}
\end{array}
\qquad \text{(P.5.1)}
$$

最初に，C は S に接続を試みるが，$\mathcal{I}$ は `dnsspoof` を用いて，C を $\mathcal{I}$ 上の `sshmitm` に接続させることにより，セッションを奪取する．次に，S は，$\mathcal{I}$ に，H_S，P_S，SA，RN_S を送信する (P1)．その後，$\mathcal{I}$ は，C に $\mathcal{I}$ のホスト鍵 $H_{\mathcal{I}}$，サーバ鍵 $P_{\mathcal{I}}$，SA，$RN_{\mathcal{I}}$ を送信する (P1′)．この段階において，C は S に接続しようとしているので，C が受信するホスト鍵は P_S でなければならない．しかしながら，当然，C が受け取ったホスト鍵 $P_{\mathcal{I}}$ と C に登録された公開鍵とが異なっているため，SSH クライアントプログラムはその違いをユーザに警告をする．ここで，注意深いユーザであるならば，なにかしらの危険があることを気づくかもしれないが，そのままその鍵を受け入れて，接続を続けてしまうユーザも当然いるだろう．また，攻撃が行われているのではなく，単に，サーバ側でホスト鍵の変更があっただけかもしれない．これは，SSH サーバの管理者からのアナウンスがなければ判別はできない．この段階で，攻撃者の存在を検知して，セッションを切らなければ，パスワードを奪取されてしまう．

しかし，前述のとおり，これは，認証プロトコルの設計上の問題ではなく，認証における公開鍵の運用方式の問題である．つまり，SSL のように PKI を導入し，公開鍵のすり替えが起こらないようにすれば，この攻撃は防ぐことができる．一方，以降で示す 2 つの攻撃は，認証プロトコルの設計上の不備が原因である

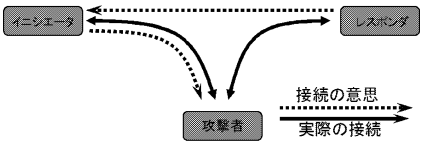

図 2 レスポンダを騙す攻撃
Fig. 2 Fake-initiator attack.

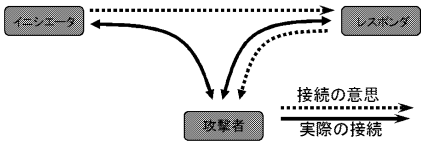

図 3 イニシエータを騙す攻撃
Fig. 3 Fake-responder attack.

ため，PKI 導入といった対策を行っても防ぐことはできない．

5.3 レスポンダを騙す攻撃モデル

5.3.1 レスポンダを騙す攻撃

イニシエータ A は，何かしらの理由で，攻撃者を，正規のレスポンダとして判断し，攻撃者 $\mathcal{I}$ へ自発的に接続を行う．ただし，このとき，イニシエータは，接続相手が攻撃者だとは思っていない．一方で，レスポンダ B は，イニシエータ A からの接続と判断している．しかしながら，実際には，攻撃者 $\mathcal{I}$ が，レスポンダ B を騙して，イニシエータ A に成りすまして，セッション鍵を奪うことになる（**図 2**）．この攻撃のポイントは，攻撃者 $\mathcal{I}$ が，正規のイニシエータ A からの通報を悪用して，レスポンダ B に対して，イニシエータ A に成りすましていることである．

この攻撃モデルに基づく攻撃を**レスポンダを騙す攻撃**と呼ぶことにする．

5.3.2 レスポンダを騙す攻撃の例

4.1，4.2 節で示したパスワードユーザ認証における攻撃は，このモデルの攻撃に分類できる．イニシエータ A としてのクライアント（SSH クライアント上のユーザ）B は，攻撃者 $\mathcal{I}$ を，正規のレスポンダ（SSH サーバ）として考えているので，攻撃者 $\mathcal{I}$ へ自発的に接続を行っている．そのうえで，攻撃者は，イニシエータ A からの通報を悪用して，レスポンダ B に対して，正規のイニシエータ A に成りすましている．

また，下記の NS（Needham-Schroeder）公開鍵認証プロトコルへの著名な攻撃も，このモデルの攻撃に分類できる：

$$\begin{array}{lll} \text{(P1)} & A \to B & : \{A, N_A\}_{P_B} \\ \text{(P2)} & B \to A & : \{N_A, N_B\}_{P_A} \\ \text{(P3)} & A \to B & : \{N_B\}_{P_B} \end{array} \qquad \text{(P.5.2)}$$

このプロトコルに対する攻撃が Lowe によって示されている[9]．参考のために，攻撃過程を抜粋するが，説明は割愛する：

$$\begin{array}{lll} \text{(P1)} & A \to \mathcal{I} & : \{A, N_A\}_{P_{\mathcal{I}}} \\ \text{(P1}') & \mathcal{I}(A) \to B & : \{A, N_A\}_{P_B} \\ \text{(P2)} & B \to \mathcal{I}(A) & : \{N_A, N_B\}_{P_A} \\ \text{(P2}') & \mathcal{I} \to A & : \{N_A, N_B\}_{P_A} \\ \text{(P3)} & A \to \mathcal{I} & : \{N_B\}_{P_{\mathcal{I}}} \\ \text{(P3}') & \mathcal{I}(A) \to B & : \{N_B\}_{P_B} \end{array} \qquad \text{(P.5.3)}$$

この攻撃の原因は，レスポンダ B とセッション鍵の対応がないことにあるので，プロトコル上での変更としては，Lowe の示した修正以外に，セッション鍵とレスポンダの識別子 B との対応関係を与えるための以下の修正を施せばよい．これは，HS（Hagiya-Saito）プロトコルと呼ばれる[14]：

$$\begin{array}{lll} \text{(P1)} & A \to B & : \{A, N_A\}_{P_B} \\ \text{(P2)} & B \to A & : \{N_A, N_B\}_{P_A} \\ \text{(P3)} & A \to B & : \{B\}_{N_B} \end{array} \qquad \text{(P.5.4)}$$

5.4 イニシエータを騙す攻撃モデル

5.4.1 イニシエータを騙す攻撃

イニシエータを騙す攻撃は，5.3 節とは逆に，イニシエータ A には，自身がレスポンダ B と接続していると誤認させている．一方，レスポンダ B は，攻撃者 $\mathcal{I}$ を正規のイニシエータとして判断して，接続している．しかしながら，実際には，攻撃者 $\mathcal{I}$ がイニシエータ A を騙して，レスポンダ B に成りすまして，セッション鍵を奪うことになる（**図 3**）．この攻撃のポイントは，レスポンダを騙す攻撃とは逆に，攻撃者 $\mathcal{I}$ が，レスポンダ B からの通報を悪用して，イニシエータ A に対して，レスポンダ B に成りすましていることである．

この攻撃モデルに基づく攻撃を**イニシエータを騙す攻撃**と呼ぶことにする．

5.4.2 イニシエータを騙す攻撃の例

ここでは，イニシエータを騙す攻撃の具体例を示す．たとえば，今，以下のような認証プロトコルを考える．

$$\begin{array}{lll}(\mathrm{P1}) & A \to B & : A \\ (\mathrm{P2}) & B \to A & : \{A, B, N_B\}_{P_A} \\ (\mathrm{P3}) & A \to B & : \{N_A, N_B\}_{P_B} \\ (\mathrm{P4}) & B \to A & : \{H(B, N_A)\}_{N_B}\end{array} \quad (\mathrm{P.5.5})$$

これは，イニシエータ A とレスポンダ B の公開鍵（P_A，P_B）を用いて，相互認証を行い，セッション鍵 N_B を交換している．しかしながら，実際には，以下のとおり，イニシエータを騙す攻撃が成り立つ．

$$\begin{array}{lll}(\mathrm{P1}) & A \to \mathcal{I}(B) & : A \\ (\mathrm{P1}') & \mathcal{I} \to B & : \mathcal{I} \\ (\mathrm{P2}) & B \to \mathcal{I} & : \{\mathcal{I}, B, N_B\}_{P_\mathcal{I}} \\ (\mathrm{P2}') & \mathcal{I}(B) \to A & : \{A, B, N_B\}_{P_A} \\ (\mathrm{P3}) & A \to \mathcal{I}(B) & : \{N_A, N_B\}_{P_B} \\ (\mathrm{P3}') & \mathcal{I} \to B & : \{N_A, N_B\}_{P_B} \\ (\mathrm{P4}) & B \to \mathcal{I} & : \{H(B, N_A)\}_{N_B} \\ (\mathrm{P4}') & \mathcal{I}(B) \to A & : \{H(B, N_A)\}_{N_B}\end{array} \quad (\mathrm{P.5.6})$$

この攻撃の原因は，一般的に，イニシエータ A とセッション鍵の対応がないこと原因であるので，プロトコル上での変更としては，セッション鍵とイニシエータの識別子 A との対応関係を与えられる修正を施せばよい：

$$\begin{array}{lll}(\mathrm{P1}) & A \to B & : A \\ (\mathrm{P2}) & B \to A & : \{A, B, N_B\}_{P_A} \\ (\mathrm{P3}) & A \to B & : \{N_A, N_B\}_{P_B} \\ (\mathrm{P4}) & B \to A & : \{H(A, B, N_A)\}_{N_B}\end{array} \quad (\mathrm{P.5.7})$$

6. パスワードユーザ認証の修正

SSH Handshake は，セッション鍵交換でのサーバ認証を行った後，ユーザ認証を行っているという 2 段階の認証によって，相互認証を行っていると見なすことができる．すると，前述のようにサーバ認証でセッション鍵が奪取されたとき，プロトコル P.3.3，P.3.5 のユーザ認証における暗号化自体の意味がなくなり，安全でない通信路上で，（平文でパスワードを送信する）PAP によるユーザ認証を行っているのと等しい．したがって，セッション鍵の交換の後，パスワードユーザ認証においては，通信主体の識別子とセッション鍵を対応付けたうえで，ここでの共有秘密であるパスワードを暗号化鍵として暗号化された通報にする必要がある．これは，パスワードを認証用の暗号化鍵とした CHAP そのものとも解釈できる．よって，5.3 節での議論を基に，たとえば，P4 で，KS_{CS} と（乱数による）ノンス N_C により，SSH1 の場合，プロトコル P.3.3 の一部を，SSH2 の場合，プロトコル P.3.5 を以下のように変更すればよい☆：

SSH1 の場合：

$$\begin{array}{ll}(\mathrm{P3}) & C \to S : \\ & \{\{N_C, C, S, KS_{CS}\}_{password}\}_{KS_{CS}} \\ (\mathrm{P4}) & S \to C : \\ & \{H(N_C, S, C, KS_{CS}), ack_2\}_{KS_{CS}}\end{array} \quad (\mathrm{P.6.1})$$

SSH2 の場合：

$$\begin{array}{ll}(\mathrm{P1}) & C \to S : \\ & \{\{uname, N_C, C, S, KS_{CS}\}_{password}\}_{KS_{CS}} \\ (\mathrm{P2}) & S \to C : \{H(N_C, S, C, KS_{CS}), ack\}_{KS_{CS}}\end{array} \quad (\mathrm{P.6.2})$$

しかしながら，このような変更をするために，SSH サーバは，ユーザのパスワードを平文で利用できるように，保持していなければならない．ただし，ここでの「平文で利用できるように」とは，平文で保存するという意味ではないことに注意されたい．

改めて指摘するまでもなくリモート端末の歴史的経緯や管理上の問題で，一般的な UNIX システムにおいて，パスワードは，その（不可逆変換である）ハッシュ値で保存されていることが多い．よって，このように，SSH サーバがユーザのパスワードを暗号化して保存するためには，SSH サーバに新たにユーザのパスワード登録するなどし，さらに，それを厳重に管理をしなければならない．

7. ま と め

本論文では，SSH Handshake におけるパスワードユーザ認証において，パスワードが奪われる攻撃に対して，SSH は脆弱であるということを具体的に示した．また，「認証プロトコルにおける攻撃モデル」を用いて，既存の攻撃と比較し，その特徴を明確にした．さらに，攻撃モデルにより，その脆弱性の原因を特定し，それに基づき修正を示した．

謝辞 有益なご意見・ご指摘をいただきました編集委員，査読者の方々に感謝いたします．

参 考 文 献

1) Barrett, D.J. and Silverman, R.E.: *SSH, The Secure Shell*, O'REILLY, ISBN: 0-596-00011-1

☆ ここでは，表記上，パスワードを暗号化鍵としているが，実際には，PKCS#5 などを用いて暗号化鍵へ変換したものを想定する．

(Feb. 2001).
2) Ylonen, T.: The SSH (Secure Shell) Remote Login Protocol, Internet Draft, Network Working Group (Nov. 1995).
3) Ylonen, T., Kivinen, T., Saarinen, M., Rinne, T. and Lehtinen, S.: SSH Protocol Architecture, draft-ietf-secsh-architecture-11.txt, Internet Draft, Network Working Group (Nov. 2001).
4) Ylonen, T., Kivinen, T., Saarinen, M., Rinne, T. and Lehtinen, S.: SSH Transport Layer Protocol, draft-ietf-secsh-transport-11.txt, Internet Draft, Network Working Group (Nov. 2001).
5) Ylonen, T., Kivinen, T., Saarinen, M., Rinne, T. and Lehtinen, S.: SSH Connection Protocol, draft-ietf-secsh-connect-14.txt, Internet Draft, Network Working Group (Nov. 2001).
6) Ylonen, T., Kivinen, T., Saarinen, M., Rinne, T. and Lehtinen, S.: SSH Authetication Protocol, dragt-ietf-secsh-userauth-13.txt, Internet Draft, Network Working Group (Nov. 2001).
7) Friedl, M., Probos, N. and Simpsom, W.A.: Diffie-Hellman Group Exchange for the SSH Transport Layer Protocol, dragt-ietf-secsh-dh-group-exchange-00.txt, IInternet Draft, Network Working Group (Apr. 2001).
8) 齋藤孝道：攻撃シナリオを用いた認証プロトコルの安全性検証器の実装，SCIS 予稿集，pp.1495–1500 (2005).
9) Lowe, G.: Breaking and Fixing the Needham-Schroeder Public-Key Protocol using FDR, *Tools and Algorithms for the Construction and analysis of Systems*, Margaria, T. and Steflen, B. (Eds), *2nd International Workshop*, *TACAS* '96, Margaria, T. and Steffen, B. (Eds.), LNCS, Vol.1055, pp.147–166 (1996).
10) http://www.monkey.org/~dugsong/dsniff/
11) PKI 関連技術解説 V 1.05，情報処理振興事業協会 セキュリティセンター (2002). http://www.ipa.go.jp/security/pki/index.html
12) Viega, J., Messier, M. and Chandra, P.（著），齋藤孝道（監訳）：OpenSSL —暗号・PKI・SSL/TLS ライブラリの詳細，オーム社 (2004).
13) Rescorla, E.（著），齋藤孝道，鬼頭利之，古森貞（監訳）：マスタリング TCP/IP（SSL/TLS 編），オーム社 (2003).
14) 萩谷昌己，竹村 亮，高橋孝一，齋藤孝道：束縛関係に基づく認証プロトコルの検証，コンピュータソフトウエア（日本ソフトウエア科学会 論文誌），Vol.20, No.3, pp.17–29 (2003).

(平成 17 年 7 月 11 日受付)
(平成 18 年 2 月 1 日採録)

齋藤 孝道（正会員）
明治大学理工学部情報科学科助教授．博士（工学）．

鬼頭 利之（正会員）
2003 年東京理科大学大学院理工学研究科情報科学専攻修士課程修了．同年（株）東芝入社．研究開発センターコンピュータ・ネットワークラボラトリー所属．ネットワークセキュリティ，システムセキュリティの研究開発に従事．

萩谷 昌己（正会員）
昭和 57 年東京大学大学院理学系研究科情報科学専攻修士課程修了．京都大学数理解析研究所を経て，現在，東京大学大学院情報理工学系研究科教授（コンピュータ科学専攻）．計算システムをモデル化し，特に演繹的な方法を用いて，その性質を計算機上で検証することに興味を持っている．最近では，電子計算機からなる計算システム以外にも，生物系や分子系も研究の対象としている．特に，分子コンピューティングの研究を行っている．

溝口 文雄（正会員）
1941 年生．1968 年東京理科大学大学院修士課程修了．同年同大学理工学部経営工学科助手．1987 年教授となり，現在，大学院研究科委員および情報メディアセンター長．工学博士．1994 年 Stanford 大学の Center for the Study of Language and Information（CSLI）の上級研究員となる．人工知能，認知科学およびソフトウエア工学に興味を持ち，Java を用いた情報家電や自律型ロボットの研究を中心に進めている．最近では，GRID 環境におけるライフサイエンスプロジェクトを推進しており，2002 年 12 月に IBM の Shared University Research（SUR）Award を受賞したほか，2003 年 3 月には情報メディアセンターが日本初の SUN の COE（Center Of Excellence）として選定された．現在，Artificial Intelligence Journal の論文編集委員．日本ソフトウエア科学会，日本認知科学会，米国人工知能学会各会員．