利用先 Web サービスを認証サーバに対して秘匿する ID 連携方式の提案と実装

磯 侑斗†　　齋藤 孝道‡

†明治大学大学院
214-8571　神奈川県川崎市多摩区東三田 1-1-1
ce36004@meiji.ac.jp

‡明治大学
214-8571　神奈川県川崎市多摩区東三田 1-1-1
saito@cs.meiji.ac.jp

あらまし　近年，OpenID や SAML などを利用した，外部のアイデンティティプロバイダによって認証を行うウェブサイトが増加している．しかしながら，このような認証方式が用いられる場合，アイデンティティプロバイダはユーザの利用先ウェブサイトを把握でき，例えば，ユーザの行動追跡や競合するサービスを利用させないといった囲い込みができてしまう．

本論文では外部のアイデンティティプロバイダを利用しながらも，ユーザが利用するウェブサイトをアイデンティティプロバイダが把握できない認証方式を提案し，その実装を示す．

A Proposal and Implementation of ID Federation That Conceals a Web Service From an Authentication Server

Yuto Iso †　　Takamichi Saito ‡

† Graduate School of Meiji University
1-1-1, Higashimita, Tama-ku Kawasaki-shi, Kanagawa, 214-8571, Japan
ce36004@meiji.ac.jp

‡ Meiji University
1-1-1, Higashimita, Tama-ku Kawasaki-shi, Kanagawa, 214-8571, Japan
saito@cs.meiji.ac.jp

Abstract Recently, it is getting poplar that a website authenticates a user with external Identity Provider using OpenID or SAML. However, such an authentication scheme exposes to the Identity Provider where its user is going. Consequently, for instance, the Identity Provider can track their users and refuse to use their competitor's service.

In this paper, we propose an authentication method that an Identity Provider cannot track their users while using the Identity Provider.

1　はじめに

Web を取り巻く技術の進歩により，電子メールや SNS など様々なサービスが Web アプリケーションとして提供されることが多くなった．

これら Web アプリケーションの利用にはパスワードによるユーザ認証が必要なものが多く，ユーザは Web アプリケーションごとにアカウントを作成，管理する必要がある．そのため複数の Web アプリケーションを利用するためには，ユーザは Web アプリケーションごとに認証を行う必要があり，管理すべきパスワードの数が増加するという問題が生じる．

ユーザにとって複数のパスワードの管理は困難であり，パスワードの使い回しといった不適切な管理がなされてしまう場合がある．パスワードが使い回された場合，ある Web アプリケーションからパスワードが漏洩すると，同一のパスワードを設定した他の Web アプリケーションにも被害が及ぶリスクが生じてしまう．

このようなユーザ認証の手間や問題を解決する方法として，SAML [1]や OpenID Authentication [2]などを利用した，外部の認証サーバを利用するという方法がある．ユーザは認証サーバのアカウントのみで複数の Web アプリケーションを利用することができ，アカウントの作成や管理といった問題から開放される．また，Web アプリケーション管理者にも，独自の認証機能の実装やパスワードの管理を行わずに済むといったメリットが存在する．

しかし，SAML や OpenID Authentication に代表される既存の方式では，認証に関する情報のやり取りのために Web アプリケーションが認証サーバと直接通信したり，Web ブラウザをリダイレクトさせたりするため，認証サーバは，いつ，誰が，どの Web アプリケーションを利用したかを把握できてしまう．認証サーバはこの情報を用いることにより，例えば，ユーザの行動追跡や競合するサービスを利用させないといった囲い込みができてしまう．

本稿では，外部の認証サーバを利用しながらも，ユーザがどの Web アプリケーション利用するのかを認証サーバが知ることのできない認証方式を提案し，その実装を示す．また，提案方式に対する攻撃について考察する．

2　関連技術

ここで，本稿に関連する技術を説明する．

2.1　SAML

SAML（Security Assertion Markup Language）は後述する IdP と SP 間でアサーションと呼ばれる，認証，認可，属性に関するデータを交換するための XML ベースのフレームワークである．SAML を利用すると，ユーザは IdP と認証するだけで，IdP と信頼関係のある SP も利用できるようになる．

SAML の構成主体

- Identity Provider（IdP）

End User のアイデンティティ情報を管理し，SP に対して End User の認証情報を提供する役割を持つ認証サーバである．予め End User のアカウントが登録されている．また，IdP と SP はアサーションなどの送信先となるエンドポイント URL や暗号化鍵などを予め交換し，互いに信頼関係がある．

- Service Provider（SP）

End User や他の主体にサービスを提供する Web アプリケーションサーバである．SP は信頼関係のある IdP からアサーションを受け入れユーザを認証する．

- End User

IdP によって認証を受け，SP を利用するユーザである．End User は，一般に Web ブラウザによって IdP や SP にアクセスする．

動作フロー

SAML ではバインディングと呼ばれる，アサーションを交換する方法を複数規定してい

るが, ここでは HTTP Redirect/POST バインディングについて説明する（図 1）.

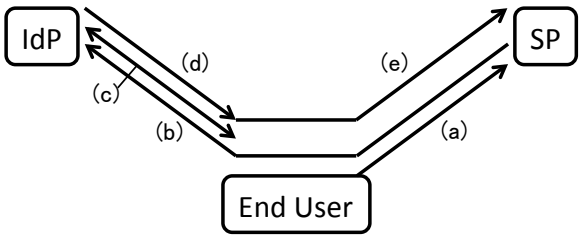

図 1 HTTP Redirect/POST バインディング

(a) End User が SP の提供する認証が必要なサービスをリクエストする.

(b) SP は End User を IdP へ HTTP リダイレクトさせる. リダイレクト先 URL のクエリ文字列に認証を要求するアサーションを含めることにより，IdP に認証要求を送信する.

(c) IdP は End User をパスワードを用いるなどして認証する.

(d) IdP は SP からの認証要求に対するアサーションを End User に送信する.

(e) End User は IdP から受け取ったアサーションを HTTP POST メソッドを用いて SP に送信する．SP はアサーションを検証することによって，End User が IdP によって認証済みであることを確認できる.

2.2 OpenID Authentication

OpenID Authentication は，ユーザが所有する URI または XRI 形式の識別子によって複数の Web アプリケーションの認証を受けることを可能にするフレームワークである．OpenID Authentication を利用すると，ユーザは後述する OP と認証するだけで， RP も利用できるようになる.

OpenID Authentication の構成主体

- OpenID Provider（OP）

End User が提示した識別子を End User が所有しているという証明を RP に提示する認証サーバである．OP には予め End User のアカウントが登録されている.

- Relying Party（RP）

OP に対してユーザの認証を要求する Web アプリケーションサーバである．ユーザの提示した識別子によって OP を特定し，鍵の交換などを行うため，事前に OP の鍵などを登録する必要はない.

- End User

OP によって認証を受け，RP を利用するユーザである．End User は，Web ブラウザによって OP や RP にアクセスする.

認証フロー

OpenID Authentication で End User が RP を利用するまでのフローを以下に示す（図 2）.

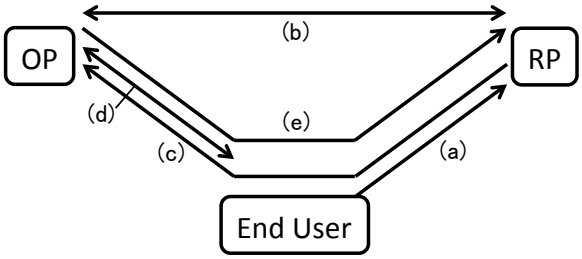

図 2 OpenID Authentication の認証フロー

(a) End User は RP へリソースの要求を行う．その後，End User は自身の識別子を提示する．RP は識別子から OP の URI を得る.

(b) RP と OP の間でメッセージの完全性と真正性を検証するための共有秘密鍵を交換する.

(c) RP は OP へ認証要求を送信する．認証要求は HTTP リダイレクトにより End User を介して送信される.

(d) OP は End User の認証を行い，認証応答を生成する．このとき，OP は認証応答を RP へ送信してよいかを End User に確認する.

(e) OP は認証応答を HTTP リダイレクトにより RP へ送信する．RP は OP から取得した認証応答を検証することにより，(a) で提示された識別子がユーザの所有するものであると確認できる.

3 既存方式の考察

SAML や OpenID Authentication などを利用

した場合， End User がどの Web アプリケーションを利用するのかを, IdP や OP は把握できてしまう．

例えば，SAML ではアサーションに発行者に関する情報が記述されているので，IdP は SP が認証要求を行う際（図 1 の (b)）に End User が利用する SP の識別子を獲得できる．OpenID Authentication では，共有秘密鍵の交換のために RP と OP が直接通信（図 2 の (b)）を行う．さらに，End User を RP から OP へ HTTP リダイレクトさせる際（図 2 の (c)）に認証応答の送信先（図 2 の (e)）を URL のクエリ文字列に含めるので OP は End User がどの RP を利用するのか把握できる．

また，これら既存方式ではユーザを経由した Web アプリケーションと認証サーバとの間で情報のやりとりを HTTP リダイレクトや HTTP POST によって実現しているが, この方法では HTTP ヘッダのリファラを参照することにより送信元が伝わってしまう．

このように，既存の方式では，アサーションなどの検証や情報の送受信を行うために Web アプリケーションの識別子を利用する．

しかし，それ以外にも例えば，ユーザの行動追跡や競合サービスを利用させないといった囲い込みに利用することも可能である．これを防ぐためには Web アプリケーションの識別子を認証サーバに渡さなければよいが, SAML や OpenID Authentication などの既存の方式を利用した場合，ユーザが利用する Web アプリケーションの識別子情報が認証サーバに渡ることが避けられない．ユーザが利用先 Web アプリケーションを認証サーバに知られたくない場合，SAML や OpenID Authentication といった既存の認証方式ではそれが適わないということが分かる．

4 提案方式

4.1 概要

前述の課題を解決するため，外部の認証サーバを利用しながらも，ユーザがどの Web アプリケーションを利用するのかを認証サーバが知ることのできない認証方式を提案する．

本方式では，後述する Token とユーザ識別子などを認証サーバがディジタル署名し，その結果を Web アプリケーションサーバが後述の方法により検証することにより認証を行う．また，Web アプリケーションサーバは事前に認証サーバの公開鍵を取得し，認証サーバとのやり取りはユーザを経由して行う．

認証サーバと Web アプリケーションサーバとの間の情報のやりとりにおいて，前述のとおり，HTTP リダイレクトなどを使うとリファラによって，Web アプリケーションサーバの識別子情報が認証サーバに伝わってしまう．そこで本提案方式では後述する Google Chrome の拡張機能 [3]を用いることによりリファラ情報の伝播を防止すること期待した．

以上の 2 つのアプローチにより，通信内容に Web アプリケーションを特定できる情報が含まれず，また，Web アプリケーションサーバの識別子が伝わることなく認証サーバと通信が行えるので，ユーザの利用する Web アプリケーションを秘匿して認証を行うことが可能となる ．

なお，本提案方式の通信は後述する盗聴などの攻撃を防ぐため，SSL/TLS 通信を前提とする．

4.2 提案方式で利用する用語の定義

- Identity Provider（IdP）

パスワード認証などを用いて End User を認証し, End User から受け取った Token（後述）に対し IdP の公開鍵ペアの秘密鍵によりディジタル署名し，End User に返す．IdP は秘密鍵が外部に漏洩しないよう管理する．

- Relying Party（RP）

IdP を利用してユーザ認証を行う Web アプリケーションサーバである．RP には予め IdP の公開鍵が登録されている．また, RP の Web ページには，アクセスの都度生成される Endpoint，Nonce，Timestamp，及び Token が埋め込まれている．

RP は End User を End User が登録された IdP

の URL とユーザ識別子の組で識別する．これは異なる IdP 間でユーザ識別子が重複する可能性があるためである．

- End User

予めアカウントを登録してある IdP によって，パスワード認証などによりユーザ認証がされ，RP を利用するユーザである．End User は Web ブラウザを利用して IdP や RP にアクセスする．

- Endpoint

IdP の発行した認証応答の送信先となる RP の URL である．

- Nonce

Token を毎回異なる値にするためのランダムな文字列である．72 ビット程度の乱数列を用いることを想定する．

- Timestamp

Token の発行日時を表す文字列である．Token が有効期間内であるかを確認するために使用する．

- Token

Endpoint，Nonce 及び Timestamp を連結した文字列をハッシュ関数によってハッシュ化した文字列であり，以下の式の通り定める．ただし H はハッシュ関数，|| は文字列の連結を表す．

Token = H(*Endpoint* || *Nonce* || *Timestamp*)

4.3 動作フロー

提案方式の動作を図 3 のシーケンス図に示す．

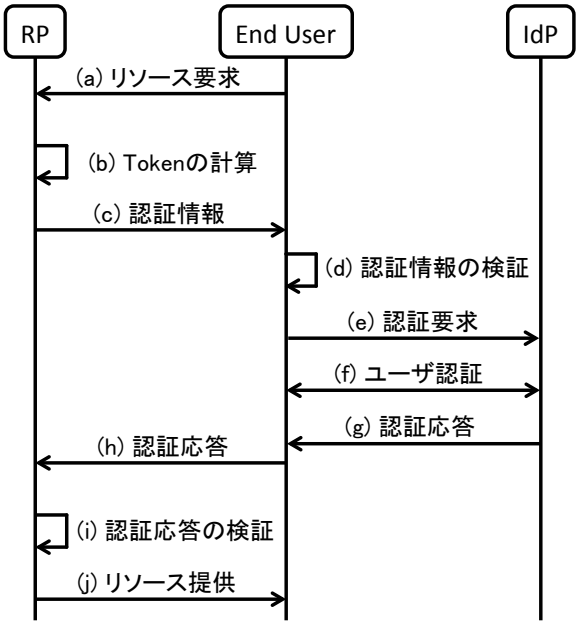

図 3 認証シーケンス

前述のとおり，前提として IdP には End User のアカウントが既に登録されているものとする．また，RP は予め IdP の真正な公開鍵を IdP の URL とともに保存している．さらに，公開鍵の秘密鍵は漏洩していないことを想定する．公開鍵を IdP の URL とともに保存するのは，複数の IdP が存在する場合に，使用する公開鍵を特定するためである．

(a) End User は認証が必要なリソースを要求する．

(b) RP は，Endpoint, Nonce, 及び Timestamp を定め，Token を計算する．

(c) RP は Token，Endpoint，Nonce 及び Timestamp を End User に送信する．

(d) End User は Endpoint，Nonce，及び Timestamp を連結した文字列をハッシュ化した値と Token が一致するかを検証する（Token の検証と呼ぶ）．また，RP のドメインと Endpoint のドメインが一致するかを確認する（Endpoint の検証と呼ぶ）．これらが一致しない場合はエラーとする．

(e) End User は認証要求として，Token と Timestamp を IdP に送信する．

(f) IdP はパスワード等を用いてユーザを認証する．

(g) IdP は認証応答として，Token，

Timestamp，IdP の URL，及びユーザ識別子に加え，これらを連結した文字列をディジタル署名した値を End User に送信する．

(h) End User は (g) で受け取った認証応答を Endpoint の示す Web サーバへ送信する．

(i) RP は End User から受け取った認証応答について，Token が (d) で発行したものであるか，すでに使用済みでないか，ディジタル署名が真正か，有効期間内であるかを検証する．署名の真正性を検証する際の公開鍵は予め取得した公開鍵の中から，認証応答に含まれる IdP の URL によって特定される．

(j) この時点で，RP は End User が IdP によって認証されたと確認できるので，リソースを提供する．

5 実装

5.1 環境

提案方式を以下の環境で実装した．

- IdP 及び RP

OS: CentOS 6.4 kernel 2.6.32-358.11.1.el6.x86_64
HTTP サーバ: Apache HTTP Server 2.2.15
サーバーサイドスクリプト: PHP 5.3.3
データベースサーバ: MySQL 5.1.69

- End User

OS: Microsoft Windows7 SP1 x64
ブラウザ: Google Chrome 29.0.1547.55 beta
暗号処理ライブラリ: CryptoJS 3.1.2 [5]

拡張機能

前述のとおり，RP と IdP 間で情報のやりとりを行うために Google Chrome の拡張機能を作成した．拡張機能は Google Chrome の機能の変更や拡張ができる，HTML，JavaScript，CSS などで記述された小さなプログラム（及びその実行環境）である．

今回作成した拡張機能は図 3 の (d) から (h) の処理を行う．

拡張機能が呼び出されると，表示中の Web ページに含まれる Endpoint, Nonce, Timestamp, 及び Token を読み込み，Token の検証と Endpoint の検証を行う．その後，Token と Timestamp を IdP に，JavaScript API の XMLHttpRequest [4]（XHR）による HTTP POST で送信する．

拡張機能の JavaScript から呼び出された XHR による HTTP リクエストには，その時に表示中の Web ページに関する情報がリファラなどに含まれない．そのため RP が発行した Token を，発行元 RP を知られずに IdP に送信することが可能となる．

図 4 RP の Web ページと拡張機能

5.2 動作

End User が RP の Web ページにアクセスすると IdP によって認証が必要である旨が表示される．前述のとおり，このページには Endpoint，Nonce，Timestamp，及び Token が埋め込まれている（図 3 の (c)）．End User が拡張機能のアイコン（図 4 の (a)）をクリックすると，拡張機能のポップアップが表示される（図 4 の (b)）．このとき，拡張機能は表示中の Web ページから認証情報を読み取り，Token の検証と Endpoint の検証を行う．検証の結果，エラーとなった場合は End User に警告を出す．

End User が所望の IdP の URL を選択し，送信ボタンをクリックすると Token と Timestamp を XHR による HTTP POST メソッドで IdP へ送信する．

その後，IdP の Web ページにおいて，End

User はパスワード等を用いて IdP と認証を行う．ただし，拡張機能の XHR が発行する HTTP リクエストには Cookie を付与することができるので，それを用いて，End User の認証済みセッションが有効である限り何度も認証する必要がないように実装した．

認証が完了すると IdP から認証応答が返される（図 3 の (g)）．これを Endpoint に HTTP POST メソッドで送信（図 3 の (h)）することにより，認証フローが完了する．

6 セキュリティに関する考察

本提案方式に対する攻撃とその対策について考察する．

6.1 盗聴

本提案方式では，SSL/TLS による暗号化通信を前提としている．暗号化を行わなかった場合，攻撃者が IdP から発行される認証応答を盗聴し，RP に提示することにより別のユーザになりすまして RP を利用することができてしまう．

この攻撃は RP が認証応答を検証する際（図 3 の (i)）に，Token が使用済みでないかを確認することである程度防げるが，攻撃者が End User より先に認証応答を RP に送信できる場合は，なりすましが可能となる．

6.2 DoS 攻撃

RP は End User からのリソース要求があると Token を生成し（図 3 の (b)）認証応答があるまで保持する必要が有る．そのため，攻撃者から大量のリクエストを受信すると RP は Token を保持しきれなくなってしまう可能性がある．

この攻撃の対策として，RP は IP アドレスによるアクセス制限などによってこの攻撃に対抗することができる．

6.3 悪意ある RP による中間者攻撃

SSL/TLS を用いた場合でも，図 5 のように攻撃者が悪意ある RP を設置し End User に利用させることにより，認証応答を受け取り，その認証応答を正規の RP に提示することにより End User になりすますという中間者攻撃が考えられる．

この攻撃にはいくつかのケースが存在するのでそれぞれのケースについて考察する．

メッセージの書き換えを行わない場合

攻撃者は正規の RP が発行する認証情報の内容を変更せずに（図 5 の (e) を行わずに）End User へ転送する．

この場合，Endpoint（正規 RP の Endpoint）と End User が利用しようとしている RP（悪意ある RP）が異なるので，図 5 の (g) で行われる Endpoint の検証によって処理が中止される．

Endpoint を書き換える場合

攻撃者は悪意ある RP に認証応答を送信させるために認証応答の送信先である Endpoint の書き換え（図 5 の (e)）を行う．

前述のとおり，拡張機能において Token の検証（図 5 の (g)）を行うため，Endpoint, Nonce，及び Timestamp のいずれかひとつでも書き換えられた場合にそれ検知できる．したがって，攻撃者によって Endpoint が書き換えられていた場合，Token の検証時に IdP が求める値と受け取った値が異なるため，認証フローを中断してこの攻撃を防ぐことができる．

Endpoint と Token を書き換える場合

攻撃者は End User に Endpoint の書き換えを検知されないように，偽の Endpoint によるオリジナルとは別の Token（偽の Token と呼ぶ）を計算し，正規の RP が発行したオリジナルの Token と置き換える（図 5 の (e)）．攻撃者は偽の Token を悪意ある RP の Endpoint と正規の RP から受け取った Nonce と Timestamp から 4.3 に示した方法によって求める．

偽の Token によって End User で Token の検証は突破できるが，RP の認証応答の検証（図 5 の (m)）において，認証応答に含まれる Token が図 5 の (d) で発行したものかを確認する．そのため，攻撃者によって Token が書

換えられていた場合，未発行の Token として検知できるのでこの攻撃を防ぐことができる．

Token を一時的に書き換える場合

RP の検証（図 5 の (m)）によって未発行の Token として検知されないために，攻撃者は Endpoint と Token を書き換えることによって得た認証応答を正規の RP に送信する際（図 5 の (l)）に，認証応答の Token を図 5 の (d) で得た正規の Token に書き換える．

認証応答のディジタル署名は IdP に送信した偽の Token に対して行われたものであるので，オリジナルの Token に書き換えた場合，図 5 の (m) で行う署名の検証に失敗するためで書き換えを検知でき，攻撃を防ぐことができる．

署名の検証を突破するためには IdP の秘密鍵で認証応答を署名し直す必要が有るため，IdP から秘密鍵が漏洩しない限り，なりすましに対して安全であると言える．

6.4 悪意ある IdP による End User のなりすまし

攻撃者が，被害者である End User が登録された IdP を装って認証応答を作成し，それを RP に送信することによって End User のふりをするケースが考えられる．

しかし，攻撃者が正規の IdP を装い認証応答を作成するためには，その IdP の秘密鍵が必要となる．したがって，IdP から秘密鍵が漏洩しない限り，この攻撃に対して安全といえる．

まとめ

本稿では，SAML や OpenID Authentication などを利用した場合に，ユーザの利用先 Web サービスを特定するが認証サーバに渡ってしまう課題を解決するために，新たな認証方式を提案した．また，提案した認証方式を実装したシステムを構築するとともに提案方式に対する攻撃について考察した．

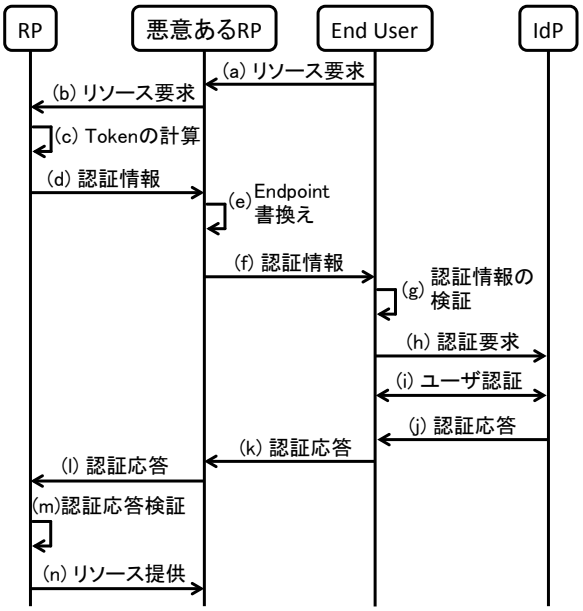

図 5 中間者攻撃

参考文献

[1] Security Assertion Markup Language (SAML) v2.0, https://www.oasis-open.org/standards#samlv2.0

[2] OpenID Authentication 2.0, http://openid.net/specs/openid-authentication-2_0.html

[3] What are extensions? - Google Chrome, http://developer.chrome.com/extensions/

[4] XMLHttpRequest W3C Working Draft 6 December 2012, http://www.w3.org/TR/2012/WD-XMLHttpRequest-20121206/

[5] crypto-js - JavaScript implementations of standard and secure cryptographic algorithms, https://code.google.com/p/crypto-js/