異なる連携プロトコルを仲介するプロキシ型属性情報管理システム

畠山　誠†

日本電気株式会社　共通基盤ソフトウェア研究所†

1．はじめに

プロバイダ同士が連携して属性情報を送受信するために様々なプロトコルが策定されている[1, 2, 3]。これらの技術を適用することにより、プロバイダはユーザからだけでなく他のプロバイダからもユーザの属性情報を取得できる。しかし、既存のプロトコル間には相互接続性がない。異なるプロトコルを利用するプロバイダは属性情報の管理方式が異なり、他のプロバイダがどの属性情報を管理しているか把握できない。そのため、プロバイダは属性情報を取得できるか否か判断できず、属性情報を交換することができない。そこで、連携プロキシを介した属性情報管理システムを提案する。連携プロキシが属性情報交換を集中的に制御することにより、プロバイダの属性情報管理方式に依存せず、属性情報を送受信することができる。

2．プロバイダ間での属性情報の交換

プロバイダ同士が属性情報を送受信すると、ユーザは属性情報を何度も入力する必要が無くなり、サービス利用の利便性が増す。プロバイダ同士が属性情報を交換する例を図1に示す。

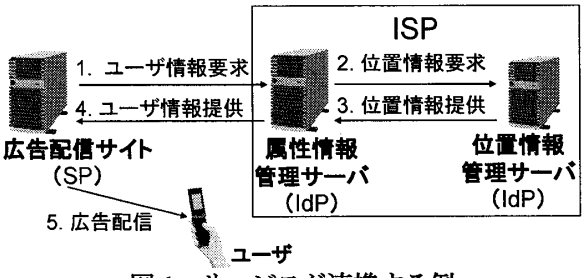

図1　サービスが連携する例

この例では、サービスプロバイダ（SP）である広告配信サイトが、広告を配信するために必要となる情報（例えば、特定のエリアにいる、興味を持っている人に広告を配信するための情報など）を取得する。これらの情報をユーザが全て広告配信サイトに入力することは煩わしいため、ユーザ情報を保有する他の事業者から取得する。例えば、ユーザ情報を保有するアイデンティティプロバイダ（IdP）であるISPの属性情報管理サーバより情報を取得する。属性情報サーバが必要な情報を全て保有していない場合、ISP内の別のサーバ（例えば、IdPである位置情報管理サーバ）よりユーザの情報を取得する。このように広告配信サイトが必要とする情報をユーザ以外から取得することによりユーザの処理を減らすことができる。

プロバイダが複数のプロバイダと連携して属性情報を送受信する場合には、複数の通信方式に対応する必要がある。プロバイダごとに通信方式やプロトコルが異なる場合、通信相手に合わせて通信方式やプロトコルを変更しなければならない。また、属性情報を送受信するプロバイダは通信方式やプロトコルに依存しない属性情報発見方式が求められる。属性情報を要求するプロバイダが属性情報要求電文を送付するプロバイダを決定するためには、属性情報を保有するプロバイダを把握する必要がある。さらに、プロバイダはユーザの個人情報を取り扱うため、ユーザのプライバシを保護することが求められる。

3．連携プロキシを介した属性情報交換

複数のプロバイダが連携して属性情報を交換するために、連携プロキシを介した属性情報交換システムを提案する。連携プロキシは図2に示すとおり属性情報を保有するIdPの間で属性情報の送受信を仲介する。連携プロキシが持つ機能は、属性情報を保有するIdPにアクセスするための条件（アクセス情報）を管理する機能、属性情報を保有するIdPを発見する機能、通信相手に応じてプロトコルを変換する連携解決機能、プライバシ保護のためユーザ同意に基づき属性情報へのアクセスを制御する機能である。連携解決とアクセス制御の詳細については[4]を参照されたい。なお、連携プロキシは属性情報の通信を集中管理するため、全てのプロバイダより信頼されていることを前提とする。

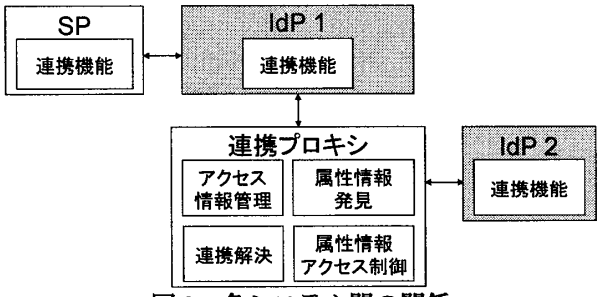

図2　各システム間の関係

"Attribute Exchange using a federation proxy connecting multiple federation protocols"

† Makoto Hatakeyama, Common Platform Software Research Laboratories, NEC Corporation.

3．1．アクセス情報管理

連携プロキシは属性情報を発見するために、IdP が発行したアクセス情報を取得し、管理する。アクセス情報には、IdP の識別子、IdP が管理する属性情報の種類、連携用のユーザ ID が記載される。連携プロキシは複数の IdP より取得したアクセス情報をもとに、属性情報を要求する IdP を決定する。

連携プロキシがアクセス情報を保有していることを他の IdP を通知するために、連携プロキシは連携情報を送付する。連携情報には、連携プロキシが把握している属性情報の種類と連携用のユーザ ID が記載される。この連携情報をもとに IdP は連携プロキシに属性情報を要求するか否かを決定する。

3．2．属性情報の発見と属性情報の送受信

アクセス情報の登録と属性情報の交換処理を図 3 に示す。まず、IdP はアクセス情報を作成し連携プロキシに通知する。連携プロキシは把握する属性情報が変わったので、連携情報を更新して、連携プロキシと連携する全ての IdP に通知する。

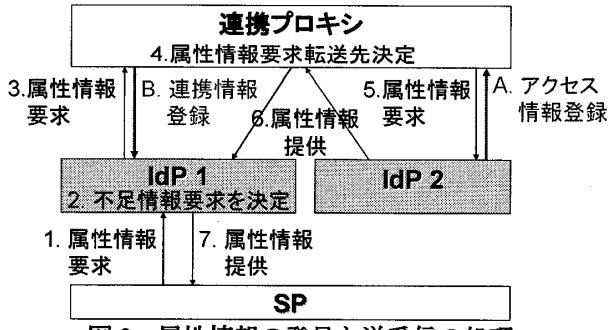

図 3　属性情報の発見と送受信の処理

属性情報の送受信は、SP が属性情報を IdP に要求する処理 (処理 1)により開始される。IdP1 が SP より属性情報の要求を受信すると、連携情報を参照して IdP1 自身が保有していない属性情報を連携プロキシに要求する (処理 2, 3)。IdP1 より属性情報を要求された連携プロキシは、アクセス情報をもとに属性情報を保有する IdP2 を発見し、属性情報を要求する (処理 4, 5)。要求された IdP2 は連携プロキシを介して IdP1 に属性情報を送付する (処理 6)。IdP1 は連携プロキシを介して取得した属性情報と IdP1 自身が保有する属性情報を SP に送付する (処理 7)。

連携プロキシや IdP はユーザのプライバシを保護するために通信相手に応じて ID を変換する。図 4 の処理 3、5、6 に示すように連携プロキシや IdP は通信先に応じて異なる ID を利用する。そのため、同じユーザであっても、通信するプロバイダや連携プロキシに応じて複数の ID が利用される。

4．考察

連携プロキシや IdP はユーザのプライバシを守るために名寄せを防ぐ。名寄せができるとプロバイダはユーザの全ての情報を追跡できるため、情報漏えいの危険性がある。そこで、各プロバイダは同じユーザであっても IdP ごとに異なる連携用 ID を利用してユーザを識別する。例えば、連携プロキシはアクセス情報や連携情報に記載する ID を IdP ごとに変更する。そのため、プロバイダは連携用 ID を利用してユーザを追跡することができない。連携プロキシや IdP が ID を変更することより名寄せを防ぎ、プライバシの漏えいを防ぐことができる。

1. SP が IdP1 に id というユーザの属性情報 *req_attr* を要求する電文 Req_to_idp1 を作成し、送付。
   Req_to_idp1 = {id, req_attr}
2. IdP1 が連携プロキシに要求する属性情報 req_attr_proxy を決定。ただし IdP1 自身が保有する属性情報 *attr_idp1* は連携プロキシに要求しない。
   $req\_attr\_proxy = req\_attr \cap \overline{attr\_idp1}$
3. IdP1 が連携プロキシに属性情報を要求する電文 Req_to_proxy を作成し、送付。このとき、電文中の ID を連携プロキシと通信するための ID (id_idp1_proxy) に設定する。
   Req_to_proxy = {id_idp1_proxy, *req_attr_proxy*}
4. 連携プロキシがアクセス情報を参照し、どの IdP に属性情報を要求するか決定。
5. 連携プロキシが IdP2 に属性情報要求を転送。このとき、電文中の ID を IdP2 と通信するための ID（id_idp2_proxy）に変換する。
   Req_to_idp2 = {id_idp2_proxy, *req_attr_proxy*}
6. IdP2 が属性情報 *res_attr_idp2* を IdP1 に提供。このとき要求電文の送受信と同様に ID を変換する。
   Res_to_proxy = {id_idp2_proxy, *res_attr_idp2*}
   Res_to_idp1 = {id_ idp1_proxy, *res_attr_idp2*}
7. IdP1 は属性情報 res_attr を SP に提供。
   $res\_attr = res\_attr\_idp2 \cup (req\_attr \cap attr\_idp1)$
   Res_to_sp = {id, *res_attr*}

図 4　属性情報の送受信処理

5．おわりに

本稿では、異なるプロトコル間で属性情報を交換する連携プロキシを提案した。連携プロキシを利用することにより、プロバイダはプライバシを守りながら属性情報を送受信できるようになる。

参考文献

[1] Liberty Alliance Project, “Liberty ID-WSF Web Services Framework Overview”, Version 1.1, November 2003. http://www.projectliberty.org/
[2] specs@openid.net, “OpenID Authentication 2.0 – Final,” 2007. Available online at: http://openid.net/developers/specs/
[3] P. Madesn and H. Itoh. “Challenges to supporting federated assurance”. IEEE Computer, Vol. 42, Issue. 5, pp. 42-49, 2009
[4] Makoto Hatakeyama, “Federation Proxy for Cross Domain Identity Federation,” In Proceedings of ACM CCS 2009 Workshop on Digital Identity Management, pp. 53-62, 2009.