情報処理学会論文誌 Vol.57 No.3 1008–1021 (Mar. 2016)

Shibbolethによる認証フェデレーションのための認証方式グループ化機能の開発

河野 圭太[1,a)] 中村 素典[2]

受付日 2015年6月22日, 採録日 2015年12月7日

概要：利用する情報サービス（Service Provider：SP）に応じて統合認証基盤で利用する認証方式を使い分けることにより，安全かつ便利な認証連携を実現できる．しかしながら，大学等の学術機関において多数の導入実績がある Shibboleth による認証フェデレーションでこれを実現するためには，SP 管理者が連携先の全認証サーバ（Identity Provider：IdP）で提供されている認証方式に関する詳細な情報を事前に把握しておく必要があり，現実的でない．この問題を解決するため，本研究では，Shibboleth IdP における認証方式のグループ化機能を開発した．本機能により，IdP・SP 間での綿密な情報共有が不要となり，異なる組織の IdP・SP が連携する認証フェデレーションにおいて，実用的な認証連携を実現できる．

キーワード：Shibboleth，IdP，SAML，認証フェデレーション，認証連携

Development of Authentication Method Grouping Function for Authentication Federations with Shibboleth

Keita Kawano[1,a)] Motonori Nakamura[2]

Received: June 22, 2015, Accepted: December 7, 2015

Abstract: Using different authentication methods on the integrated authentication infrastructure for each Service Provider (SP) enables a secure and convenient federated authentication. It is, however, impractical to achieve this on the authentication federations with Shibboleth, widely deployed at academic institutions like universities. The administrators of each SP have to understand the detailed information about the authentication methods provided on all the cooperating authentication servers (Identity Providers: IdPs) beforehand. We have developed an authentication method grouping function on Shibboleth IdP to address this issue. With the developed function, IdPs and SPs are not required to share the detailed information. This achieves a practical federated authentication within the authentication federations consisting of IdPs and SPs in different institutions.

Keywords: Shibboleth, IdP, SAML, authentication federation, federated authentication

1. はじめに

近年，世界的な ICT の浸透にともない，利用者が日常的に用いる情報サービスの数が増加している [1]．そのような状況下で，利用者の利便性とシステムの安全性を確保するため，多くの組織では，システム間の認証連携を実現する統合認証基盤が導入されている [2], [3], [4]．統合認証基盤により，利用者は，信頼できるいつもの認証サーバによる一度の認証で，複数の情報サービスを利用（シングルサインオン）できる．

クラウドサービスの活用が進む中，このような認証連携の流れは組織内にとどまらず，組織外の情報システムとの認証連携も進んでいる [5], [6], [7], [8]．利用者は，自組織の認証サーバによる認証で，自組織が提供する情報サービスに加えて，他組織が提供する情報サービスを利用できる．

学術界では，認証フェデレーションと呼ばれる共同体の

[1] 岡山大学
Okayama University, Okayama 700–8530, Japan
[2] 国立情報学研究所
National Institute of Informatics, Chiyoda, Tokyo 101–8430, Japan
[a)] keita@cc.okayama-u.ac.jp

構築により，組織の枠を超えた情報サービスの相互・共同利用に関する取り組みも進んでいる [9]．認証フェデレーションでは，信頼フレームワークと呼ばれる枠組みの中で，各組織が定められたポリシを遵守し，互いを信頼し合うことにより，複数組織間の認証連携をスムーズに実現できる [10]．

日本では，国立情報学研究所が運営する「学認（GakuNin）」がその役割を担っている [11]．大学等の学術機関は，学認に自組織の認証サーバを参加させることにより，自組織の利用者に契約済みの電子ジャーナル等の情報サービスを自組織の認証サーバによる認証で利用させることができる．また，学認に自組織の情報サービスを参加させることにより，他組織の利用者に自組織の情報サービスをその利用者が所属する組織の認証サーバによる認証で利用させることもできる．

学認では，海外の主要な学術フェデレーションと同様，Security Assertion Markup Language（SAML）に基づく組織間の認証連携・属性交換が実現されており，技術的には Internet2 が開発した Shibboleth が使用されている [12], [13]．そのため，大学等の学術機関において，学認による認証連携との親和性を考慮し，Shibboleth を用いた統合認証基盤の構築事例が多数報告されている [14], [15], [16], [17]．また，近年では，インターフェデレーションと呼ばれる認証フェデレーション間の連携も進んでおり，ますます利用者の利便性が広がる傾向にある [18]．

このような組織内外にまたがる認証連携が進むにつれて，統合認証基盤が提供すべき安全性の重要度が増している．認証方式として ID・パスワード認証を提供する統合認証基盤では，1 組の ID とパスワードの漏洩が，その利用者が利用できる複数の情報サービスの不正利用につながる [19], [20]．ここで，本稿における認証方式とは，認証サーバにおいて利用者を認証するために用いられる方式のことであり，SAML 標準では認証コンテキストによって示される [21]．1 つの認証サーバでは認証コンテキストが定まれば認証方式も定まるため，本稿では特に必要な場合を除き，認証コンテキストも認証方式と記述する．安全性を高めるためには，スマートカード認証やバイオメトリクス認証等，強度の高い認証方式を採用し，なりすましの被害を軽減させることが求められるが，このような認証方式を全面的に採用することは，コストの増加に加えて，利用者の利便性低下につながる．

そこで，身元保証レベル（Level of Assurance：LoA）という概念の下，統合認証基盤の認証方式を，利用する情報サービスごとに変更する方法が確立されつつある [22], [23]．ここでは，利用する情報サービスが保有する情報資産の保護レベルに応じて，統合認証基盤が提供する複数の認証方式から，適切な LoA を有する認証方式を選択する．たとえば，重要度が低い情報資産しか保有しない情報サービスを利用する場合には ID・パスワード認証を許容する一方で，重要度が高い情報資産を保有する情報サービスを利用する場合にはスマートカード認証を要求する．これにより，利用者の利便性を著しく低下させることなく，必要な安全性を確保できる．

我々は，認証フェデレーションのような多様な利用者，組織が存在する環境では，このような認証方式の選択は，利用者，認証サーバ管理者，情報サービス管理者の意向を組み合わせて反映させることが重要と考えている．たとえば，セキュリティ意識が高い利用者は，認証サーバ管理者，情報サービス管理者の要求にかかわらず，自身の認証につねに強度が高い方式を要求する．また，認証サーバ管理者によっては，ある情報サービスを利用する自組織の利用者に対して，当該情報サービス管理者が許容する下限よりも強度が高い方式を要求する．利用者，認証サーバ管理者，情報サービス管理者の要求（要求自体が存在しないものを除く）に共通して存在する認証方式のいずれかを利用者や認証サーバ管理者が選択することにより，三者の多様な思惑を反映させた安全かつ便利な認証連携が実現できる．本稿では，このうち，主に情報サービス管理者の意向を反映させた認証方式の選択に焦点を当てる．

ここで，上述したように大学等の学術機関において採用実績が多い Shibboleth 認証サーバ（Identity Provider：IdP）では，利用する認証方式の選択に関する十分な実装がなされておらず，実用上の課題が残されている[*1]．

とりわけ，Shibboleth IdP では，後述するように，比較表現による要求認証方式の指定がサポートされていない．そのため，ある情報サービス（Service Provider：SP）の管理者が，自サービスの利用者に特定レベル以上の LoA を有する認証方式を要求する場合，その利用者が所属する組織の IdP で提供している該当レベル以上の LoA を有する全認証方式を，認証要求中で指定する必要がある．認証フェデレーションのような環境を考慮した場合，異なる組織の IdP・SP 間で綿密な情報共有を必要とするこの制限は許容できない．

そこで，本研究では，Shibboleth IdP における認証方式のグループ化機能を開発し，この問題の解決を図る．本機能により，SP 管理者は，LoA 等に対応するグループ形式で IdP が利用する認証方式を要求することにより，自身の意向を反映できる．そのため，認証フェデレーションのような環境で，実用的な運用レベルで認証方式の選択を実現できる．

以下，2 章では，関連技術として，Shibboleth の概要および Shibboleth IdP における認証方式選択の問題点を述べる．また，3 章では，2 章で述べる問題点を解決するために本研究で開発した Shibboleth IdP における認証方式

[*1] 2014 年 12 月に Shibboleth IdP V3 がリリースされたが，本稿では Shibboleth IdP V2 を前提とする [24], [25]．

グループ化機能について述べる．最後に，4 章，5 章で，考察とまとめ，今後の課題を述べる．

2. 関連技術

本章では，まず，2.1 節において，本研究の対象とする Shibboleth の概要とその動作原理を述べ，2.2 節において，Shibboleth IdP における認証処理の概要を述べる．また，2.3 節において，LoA の概要を述べる．さらに，2.4 節において，Shibboleth IdP における認証方式選択の問題点を述べ，2.5 節において，関連研究について述べる．

2.1 Shibboleth

Shibboleth は，OASIS で策定された SAML 標準に基づき，組織間のシングルサインオン・属性交換を実現するためのオープンソースソフトウェアである [12], [26]．学認を含む多くの学術的な認証フェデレーションで採用されており，認証連携基盤の中心的な役割を担っている．

Shibboleth では，利用者は，ある SP を利用するために，自組織の IdP で認証を受け，その結果に基づき，SP での認可を受ける．ある SP のリソースにアクセスしようとした利用者が未認証である場合，当該 SP は利用者に IdP での認証を指示する．利用者は，自組織の IdP の発見を支援するための Discovery Service（DS）を用いて，自組織の IdP に到達し，認証を受ける．利用者は認証結果を SP に提示することにより，当該 SP のリソースを利用できるようになる．このような認証・認可の分離により，組織間での認証連携を安全かつスムーズに実現できる．

図 1 に，SAML V2.0 の Web Browser SSO Profile を前提とした場合の，Shibboleth の動作原理を示す [27], [28]．図 1 は，ある SP の利用者が，自組織の IdP で認証を受けた結果に基づき，当該 SP を利用する様子を示している．単純化のため，図 1 では利用者が DS の支援により自組織の IdP を発見する手順を省略している [29]．

まず，(1) 利用者が SP のリソースにアクセスしようとすると，SP で有効なセッションの有無がチェックされる．有効なセッションが存在しない場合，シングルサインオンプロセスが開始され，(2) SP で生成された IdP への認証要求は，利用者のブラウザを介して IdP へリダイレクトされる．IdP でも有効なセッションの有無がチェックされ，有効なセッションが存在しない場合には，(3) IdP で提供している認証方式で，利用者の認証が行われる．なお，有効なセッションが存在する場合には，手順 (3) は省略される．認証に成功すると，(4) IdP で生成された SP への認証応答は，再び利用者のブラウザを介して SP へリダイレクトされる．(5) SP での認可結果を受け，利用者が当該 SP のリソースにアクセスできるようになる．

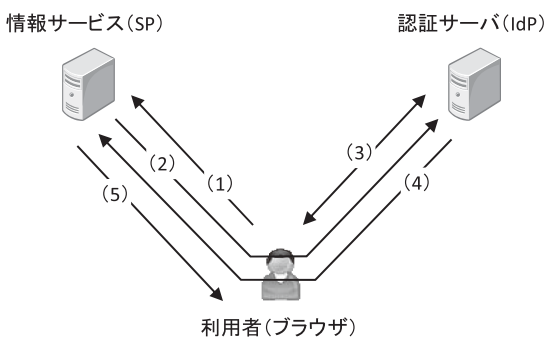

図 1 Shibboleth の動作原理
Fig. 1 Operating principal of Shibboleth.

ここで，Shibboleth が準拠する SAML 標準では，手順 (2) で生成される IdP への認証要求に，SP 管理者が許容できる認証方式のリストを含めることができる [30]．SP 管理者は，認証要求中の RequestedAuthnContext に AuthnContextClassRef または AuthnContextDeclRef を含めることにより，これを指定できる．たとえば，認証方式としてクライアント証明書認証を要求する場合，AuthnContextClassRef に TLSClient を指定する[*2]．また，RequestedAuthnContext に Comparison を含めることにより，比較表現による認証方式の指定もできる．

Shibboleth IdP では，RequestedAuthnContext 内の AuthnContextClassRef または AuthnContextDeclRef として指定された文字列と，AuthenticationMethod としてログインハンドラ（IdP で提供している認証方式に対するハンドラ）ごとに対応付けられた文字列を比較することにより，要求された認証方式を同定する．1 つのログインハンドラに対して複数の AuthenticationMethod を対応付けることもできるが，本研究で利用を前提としている Shibboleth IdP の UsernamePasswordLoginHandler，RemoteUserLoginHandler には複数の AuthenticationMethod を対応付けることが推奨されていないため，本稿ではログインハンドラと AuthenticationMethod は 1 対 1 で対応付けられていることを前提とする [31], [32]．そのため，利用するログインハンドラが定まれば，利用する認証方式（ログインハンドラやそれに対応するサーブレット，コンテナベースの認証機能の組合せで構成される）も定まる．

IdP で提供している認証方式のうち，当該 SP が許容できない認証方式は，IdP の AuthenticationEngine と呼ばれる機能の中で，ログインハンドラごとにフィルタリングされる．なお，Comparison に関しては，edu.internet2.middleware.shibboleth.idp.authn パッケージに含まれる Saml2LoginContext.java のコメントに「For the immediate future, we only support the "exact" comparator.」と記載されているとおり，「exact（厳密一致）」のみがサポートされている [33]．Shibboleth IdP では，ログに警告を記録するものの，RequestedAuthnContext に「exact」

[*2] 実際には，urn:oasis:names:tc:SAML:2.0:ac:classes:TLSClient のような形式で指定される [21]．

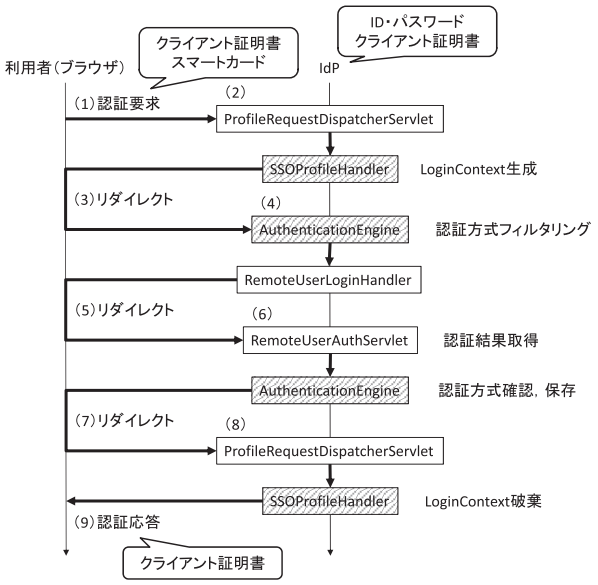

図 2 Shibboleth IdP の認証処理
Fig. 2 Authentication process of Shibboleth IdP.

以外の Comparison を含む認証要求を RequestedAuthnContext がまったく指定されていない（Comparison だけでなく，AuthnContextClassRef や AuthnContextDeclRef も指定されていない）認証要求として扱うため，注意が必要である．

2.2 Shibboleth IdP の認証処理

2.1 節で示したように，IdP では，自組織の利用者に対する認証が行われる．LoA を考慮した認証方式の選択を行う場合等，IdP で複数の認証方式を提供し，条件に応じて使い分けることもできる．IdP で複数の認証方式が提供される場合，認証要求中の RequestedAuthnContext の内容等に従い，実際に利用する認証方式が決定される．

図 2 に，SAML V2.0 の Web Browser SSO Profile を前提とした場合の，Shibboleth IdP における認証方式の選択を含む認証処理の概要を示す．図 2 の IdP では，ID・パスワード認証，クライアント証明書認証が提供されている．また，SP 管理者は当該 SP の利用者にクライアント証明書認証またはスマートカード認証を要求しており，SP で生成する IdP への認証要求中でそれを表現している．なお，図中の斜線部は，後述する LoginContext の読み書きが発生する処理部を示している．

まず，2.1 節における手順（2）に従い，（1）認証要求が IdP に到達する．IdP に到達した認証要求は，（2）ProfileRequestDispatcherServlet を経て，SSOProfileHandler で処理される．ここでは，IdP における認証処理の状態を保持するための LoginContext が生成される．このとき，後の処理で活用するため，認証要求中の RequestedAuthnContext の内容を，LoginContext に保存しておく．その後，認証方式の選択処理へと進むため，（3）要求は再度 IdP へとリダイレクトされる．

次に，（4）AuthenticationEngine において，認証方式の選択が行われる．ここでは，手順（2）で LoginContext に保存された RequestedAuthnContext の内容に基づき，IdP で提供している認証方式のうち，SP 管理者が許容しない認証方式がログインハンドラごとにフィルタリングされる．図 2 の例では，IdP が提供する認証方式，SP 管理者が要求する認証方式のどちらにも存在するクライアント証明書認証のみが今回利用可能な認証方式として残る．

AuthenticationEngine では，フィルタリング後に残った認証方式から実際に利用する認証方式を選択し，対応するログインハンドラ（図 2 の例では，RemoteUserLoginHandler）へ処理を引き渡す．なお，当該利用者が，フィルタリング後に残った認証方式のいずれかですでに認証を済ませ，有効なセッションが残っている場合，その認証結果を再利用（実際の認証処理を省略）することもできる [34]．

続けて，実際の認証処理へと進むため，（5）要求は再度 IdP へとリダイレクトされる．図 2 の例では，クライアント証明書認証が実施され，（6）RemoteUserAuthServlet で認証結果を得る．その後，要求は再度 AuthenticationEngine で処理され，実際に利用された認証方式が手順（2）で LoginContext に保存した RequestedAuthnContext の内容を満たすかどうかが確認される．要求内容を満たす場合，後の認証応答作成処理で活用するため，実際に利用された認証方式の情報を，LoginContext に保存しておく．要求内容を満たさない場合，認証要求とは異なる方式で認証されたことになるため，認証エラーとする．また，ここでは，今後の再利用に備えて，使用された認証方式による認証成功の記録を，指定された期間保管しておく．

その後，認証応答の作成処理へと進むため，（7）要求は再度 IdP へとリダイレクトされ，（8）ProfileRequestDispatcherServlet を経て，再度 SSOProfileHandler で処理される．ここでは，後処理として LoginContext を破棄するとともに，LoginContext に保存された使用認証方式の情報（AuthnContext）を含む認証応答を生成する．最後に，（9）生成された認証応答は SP へとリダイレクトされ，SP での認可を受ける．

このように，Shibboleth IdP では，LoginContext に保存された内容に基づき，各種処理が逐次実行される．

2.3 Level of Assurance（LoA）

Shibboleth 等の活用により，SP は IdP を信用し，その認証結果を受けて，認可制御を実施する．このとき，SP の観点からは，IdP の認証結果がどの程度信頼できるかを把握できることが求められる．たとえば，重要度が高い情報資産を保有する SP 管理者は，IdP からの認証結果に高い信頼性を要求する．

このような情報サービスの保護レベルを考慮した認証

表 1 ITU-T X.1254 における LoA の分類
Table 1 Classification of LoAs in ITU-T X.1254.

Level	Description
1 - Low	Little or no confidence in the claimed or asserted identity
2 - Medium	Some confidence in the claimed or asserted identity
3 - High	High confidence in the claimed or asserted identity
4 - Very high	Very high confidence in the claimed or asserted identity

連携を実現するため，身元保証レベル（Level of Assurance：LoA）が用いられる [22], [23]．認証結果の信頼性には，ID の作成方法，配布方法，認証メカニズムの強度等，複数の要素が影響するが，これらに対する要求事項を複数のレベルに分け，IdP・SP 間で共有する[*3]．たとえば，ITU-T X.1254 においては，**表 1** のように 4 レベルの LoA が規定されている [22]．高レベルの LoA ほど，ID の作成方法，配布方法，認証メカニズムの強度等に，厳しい制約が課せられる．IdP では，SP 管理者が要求したレベル以上の LoA を有する認証方式を用いて，利用者の認証を行う．

なお，IdP・SP 間で，LoA 等のポリシを個別に規定することもできるが，認証フェデレーションにおいては，信頼フレームワーク内で一律のポリシを策定し，運用レベルを統一することが求められる [10]．

2.4 Shibboleth IdP における認証方式選択の問題点

2.1 節で示した RequestedAuthnContext の利用により，Shibboleth IdP においても SP 管理者の意向を反映させた認証方式の選択を実現できる．たとえば，特定の認証方式による認証のみを許容したい場合，当該認証方式を RequestedAuthnContext 内の AuthnContextClassRef として IdP への認証要求へ含めることで，これを実現できる．また，LoA を考慮した認証方式の選択が求められる場合，同様に，SP 管理者が要求するレベル以上の LoA を有する認証方式を IdP への認証要求へ含めることで，これを実現できる．

しかしながら，前述したように，Shibboleth IdP では，RequestedAuthnContext 内の Comparison に関しては「exact」しかサポートされていないため，LoA を考慮した認証方式の選択が求められる場合には，SP で当該利用者が所属する組織の IdP が提供する認証方式のうち，該当する全認証方式を AuthnContextClassRef に列挙する必要がある．このためには，連携する全 IdP で提供されている認証方式とその LoA すべてを SP 管理者が事前に把握しておく必要があり，異なる組織間での認証連携を実現する認証フェデレーションでは現実的でない．とりわけ，IdP で同種の認証方式を提供する場合等，文献 [21] の初期リストに含まれない認証方式による指定が必要な場合には，この問題が顕著になる．これは，たとえ同一組織内（学内フェデレーション）の IdP・SP 間であっても，運用組織が異なれば同様に発生する問題である．

各種ログインハンドラに対応付ける AuthenticationMethod と SP が認証要求に含める AuthnContextClassRef を単純に LoA に基づく記述に改める方法も考えられるが，同一の IdP 内で，同一レベルの LoA を有する複数の認証方式を提供できない，個別認証方式の形式による指定との併用ができない，連携先（個別 SP や認証フェデレーション）ごとに LoA の分類方法が異なる場合に対応できない等，運用上の問題が発生する．なお，同一の IdP 内で連携先ごとに LoA の分類方法が異なる例としては，連携先ごとの定義レベル数の違いのほか，同一レベルに対する定義の違い，定義の解釈の違いも考えられる．このような場合，同一の IdP における同一の認証方式を連携先ごとに異なるレベルを有する認証方式として扱うことが必要になる場合がある．

なお，Shibboleth IdP V3 では，「exact」以外の Comparison もサポートされるようになり，本問題の解決が期待されるが，この場合にも，実装および運用上の問題が発生する．たとえば，「LoA1 以上」を表現するために，AuthnContextClassRef を「LoA1」，Comparison を「minimum」とした場合も，Shibboleth IdP V3 では，実際に使用した認証方式の情報を AuthnContext とする認証応答を生成する[*4]．このため，特に実際に利用した認証方式が文献 [21] の初期リストに含まれない認証方式であった場合，実際にどのような LoA を有する認証方式が用いられたのかを SP 側で判断できず，LoA1 を有する認証方式が用いられた場合には読み込みのみ可能とする一方で，LoA2 を有する認証方式が用いられた場合には読み書き可能とする等，LoA に応じた機能制限に活用できない．これは SAML 標準利用上の問題ではあるが，SAML 標準自体にも改善の余地がある．前述したように，本稿では Shibboleth IdP V2 を前提としているため，この問題についての詳細な検討は，今後の課題とする．

2.5 関連研究

GUARD プラグイン（Gakunin mUlti Authentication mechanism with Risk-based Decision plug-in）は，本研究と同様に，Shibboleth IdP で利用者を認証する方式を選択するための機能である [36], [37]．GUARD プラグインでは，IdP で提供する複数の認証方式をレベルとして抽象化

[*3] 単純化のため，本節以外では，認証メカニズムの強度が LoA を支配すると仮定している．

[*4] Shibboleth IdP V3.1.1 で動作を確認した．なお，Shibboleth IdP V3.1.1 の標準では，「minimum」を「exact」として扱うように設定されている [35]．

し，利用する SP が取り扱う情報の重要度に応じて，認証レベルの選択を行う．

このとき，利用者の IP アドレスに応じて要求する認証レベルを変更でき，たとえば，学内からは ID・パスワード認証を許容する一方，学外からはクライアント証明書認証を必須とするような運用もできる．また，and 条件，or 条件による認証方式の指定ができるため，ID・パスワード認証に加えてクライアント証明書認証を要求するような運用や，クライアント証明書認証またはスマートカード認証のいずれかを利用者に選択させるような運用もできる．

GUARD プラグインは，本研究と比較して，IdP 管理者が規定した画一的な認証レベル群による運用に特化した実装となっており，連携先の SP や認証フェデレーションごとに認証レベルの分類方法が異なる場合や，認証レベル以外のグループ分けが必要な場合に対応できない．

Multi-Context Broker（MCB）も，本研究と同様に，Shibboleth IdP で利用者を認証する方式を選択するための機能である [38]．MCB では，各認証コンテキストごとに，対応する認証方式と上位レベルの認証コンテキストを指定でき，上位の認証コンテキストに対応する認証方式ですでに認証済みであれば，実際の認証が省略される．また，利用者ごとに利用可能な認証コンテキスト群を指定する機能を持つため，利用者の意向を反映させた認証方式の選択に応用できる．

MCB は，本研究とは異なり，IdP で提供する全認証方式を 1 つのログインハンドラで処理する実装を採用しており，従来のログインハンドラを用いる認証方式はサブモジュールとして実装し直す必要がある．現在，JAAS による認証方式，REMOTE_USER 方式による認証方式，Duo Security がサブモジュールとして提供されている．また，1 つのログインハンドラで全認証方式を処理する特性上，authenticationDuration によるセッション保持期間の指定等，従来機能によるログインハンドラごとの設定を利用できない [34]．

3. 認証方式グループ化機能の開発

2.4 節で明らかにした問題点を解決するため，本研究では，Shibboleth IdP における認証方式グループ化機能を開発した．まず，3.1 節で，開発した認証方式グループ化機能の概要を述べる．また，3.2 節で，本機能の処理中に発生する形式変換のタイミングについて言及する．さらに，3.3 節で，本機能の実装について述べ，3.4 節で，本機能の動作確認について述べる．

3.1 概要

2.4 節で述べたように，異なる組織間での認証連携を実現する認証フェデレーションにおいて，現状の Shibboleth IdP を用いて SP 管理者の意向を反映させた認証方式の選択を実現することは，運用上の問題から現実的でない．

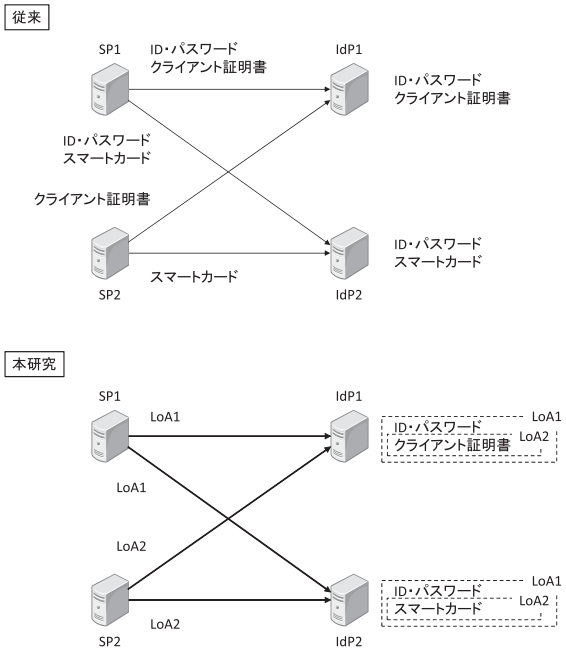

図 3 認証方式の要求方法
Fig. 3 Request method for authentication method.

本研究では，IdP で提供する認証方式をグループ化し，SP が IdP への認証要求に含める許容認証方式をグループ形式で指定できる機能を開発することにより，この問題の解決を図る．ここで，LoA に基づく認証方式の選択に注力する場合，グループにレベルの概念を持たせ，特別な指定をしなくても低レベルグループが高レベルグループを完全に包含するようにあらかじめグループの関係性を持たせておくことも考えられるが，本研究では，グループにはレベルの概念を持たせず，汎用的なグループとして扱う．このように，汎用的なグループ形式による指定を可能とすることにより，個別ポリシによる認証方式群の指定や，連携先ごとの LoA 分類方法の違い等に対応できる．たとえば，自組織内の SP と特定の認証フェデレーションの SP に対して認証サービスを提供する IdP において，所属する認証フェデレーションとは異なる分類方法の LoA を自組織内で定義し，自組織ではスマートカード認証を自組織内の LoA2 を有する認証方式として扱う一方で，所属する認証フェデレーションでは認証フェデレーション内の LoA1 を有する認証方式として扱う等の対応ができる．

図 3 に，従来の個別認証方式の形式による認証方式の要求方法と，本研究のグループ形式による認証方式の要求方法の違いを示す．図 3 では，認証フェデレーション内に 2 つの IdP（IdP1，IdP2）と 2 つの SP（SP1，SP2）が存在している．

IdP1 では，当該フェデレーション内の LoA1 を有する認証方式として ID・パスワード認証を，LoA2 を有する認証方式としてクライアント証明書認証を提供しており，IdP2

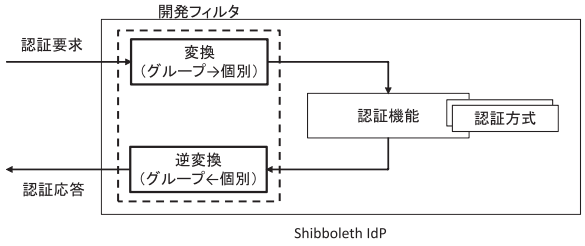

図 4 システム開発イメージ
Fig. 4 System development image.

では，LoA1 を有する認証方式として ID・パスワード認証を，LoA2 を有する認証方式としてスマートカード認証を提供している．また，SP1，SP2 は，自サービスの利用者に，それぞれ LoA1，LoA2 を有する認証方式による認証を要求している．なお，当該認証フェデレーション内では，信頼フレームワークのポリシにより，LoA の運用レベルが統一されていることを想定している．

図 3 に示すように，従来の方法では，SP 側で，それぞれの IdP で提供している全認証方式とその LoA を把握し，要求するレベル以上の LoA を有する全認証方式を認証要求に含める必要がある．一方，本研究では，事前に合意された LoA（グループ形式）に基づく認証方式の要求ができるため，SP 側で，それぞれの IdP で提供されている個別認証方式を把握する必要がない．このように，異なる組織の IdP・SP 間での事前の情報共有を最小限とすることにより，実用的なレベルでの運用が可能になる．

なお，本研究では，図 3 に示すように，SP 側は認証方式の要求内容を変更するのみでよく，SP 側に特別な実装を必要としない．また，従来の個別認証方式の形式による認証方式の指定も，引き続き利用できる．

図 4 に，本研究におけるシステム開発イメージを示す．まず，SP からリダイレクトされた認証要求に対して，LoA 等に対応するグループ形式で指定された SP が要求する認証方式を，当該 IdP で提供されている個別認証方式の形式へと変換する．この後，従来の認証機能による利用者の認証を実施し，認証応答を生成する．このとき，SP からの認証要求との整合性を取る必要性から，認証応答についても，個別認証方式の形式から，グループ形式への逆変換を実施する．これは，3.4 節で述べるような，SP 側で実施されるアクセス制御による影響を回避するために必要である．

このような開発の採用により，従来の機能によるシングルサインオンの実現等を保ったまま，認証方式のグループ化を実現できる．特に，従来の認証機能において個別認証方式の形式による処理を保つことにより，個別認証方式の形式による指定と併用する場合や，複数のグループに所属する認証方式が存在する場合にも，厳密なシングルサインオンを提供できる．たとえば，所属する認証フェデレーション内の LoA1 を有する認証方式としてスマートカード認証による利用者の認証を実施した後，自組織内の LoA2 を有する認証方式としてとしてスマートカード認証を要求する場合にも，すでに実施済みのスマートカード認証により，実際の認証を省略できる．

本研究では，既存の認証機能による処理が，2.2 節で述べたように，IdP に保存される LoginContext の内容に基づき逐次実行される点に着目し，LoginContext の内容を矛盾なく書き換えることによって，これらの処理を実現する．つまり，図 4 では認証要求，認証応答に対して直接的に変換，逆変換の処理をするように図示したが，実際には LoginContext の内容を書き換えることにより，これに相当する動作をさせている．より詳細には，認証要求に対する変換処理では，認証要求内の RequestedAuthnContext に基づき LoginContext に保存される requestAuthenticationMethods を書き換える．また，認証応答に対する逆変換処理では，使用認証方式の情報として LoginContext に保存され，認証応答の生成時に用いられる authenticationMethodInformation を書き換える．これにより，SAML 標準の Authentication Request Protocol を構成する 2.1 節における手順（2）や 2.2 節における手順（1）の認証要求および，2.1 節における手順（4）や 2.2 節における手順（9）の認証応答では，認証方式をグループ形式で取り扱うことができる [30]．

なお，本研究では，実運用開始後のメンテナンス性を考慮し，Tomcat のフィルタ機能を用いてこれらの機能を実装した [39]．Shibboleth IdP のソースとは独立させ，フィルタとして実装することにより，IdP のバージョンアップ作業時等に影響を及ぼしにくい構成とした．また，フィルタ機能の開発に際しては，Shibboleth IdP やその拡張機能のソースコードを参考にした [40], [41], [42]．

3.2 形式変換のタイミング

本研究における認証方式グループ化機能の開発においては，LoginContext に保存された SP が要求する認証方式の形式変換タイミングが重要である．本機能を用いた場合にも，IdP・SP 間で正しく認証連携を継続させるためには，適切なタイミングでフィルタ機能を実行し，形式を変換する必要がある．

まず，認証要求内の RequestedAuthnContext をグループ形式から個別認証方式の形式に変換する機能（より詳細には，RequestedAuthnContext に基づき LoginContext 内に保存される requestAuthenticationMethods を変換する機能）については，2.2 節で概要を示したように，LoginContext の生成および requestAuthenticationMethods の保存が 2.2 節における手順（2）の SSOProfileHandler で実施され，requestAuthenticationMethods を用いた認証方式のフィルタリングが 2.2 節における手順（4）の AuthenticationEngine で実施されることから，この間（2.2 節における手順（3）と（4）の間）に実施する．

```
public class AuthnMethodFilter implements Filter {
  public void init(...) ... { // 共通処理
    マッピングの作成
  }
  public void doFilter(...) ... {
    if (認証前) { preprocess(...); }
    else if (認証後) { postprocess(...); }
  }
  protected void preprocess(...) { // 変換処理
    要求認証方式の保存
    要求認証方式の変換
  }
  protected void postprocess(...) { // 逆変換処理
    保存した要求認証方式の取得
    使用認証方式の逆変換
  }
}
```

図 5 開発フィルタの機能概要
Fig. 5 An outline of functions of development filter.

また，認証応答内の AuthnContext を個別認証方式の形式からグループ形式に逆変換する機能（より詳細には，使用認証方式の情報として LoginContext 内に保存される authenticationMethodInformation を逆変換する機能）については，同様に 2.2 節で概要を示したように，2.2 節における手順（6）の AuthenticationEngine において認証成功の記録が authenticationMethodInformation と紐づけて保管され，2.2 節における手順（8）の SSOProfileHandler において authenticationMethodInformation を用いて認証応答が生成されることから，この間（2.2 節における手順（7）と（8）の間）に実施する．これにより，IdP では，従来どおり逆変換前の個別認証方式ごとに認証の成功を記録することで適切なシングルサインオンを実現する一方で，認証応答に対しては認証要求に整合させて逆変換後のグループ形式での認証ステートメントを発行できる．

3.3 実装

本研究では，前節までに述べた内容に基づき，Shibboleth IdP における認証方式のグループ化を実現するフィルタを開発した．本機能は，図 4 に示したように，大きく，既存の認証機能実行前の変換処理と，既存の認証機能実行後の逆変換処理に分けられるが，グループ情報（グループ形式と個別認証方式の形式のマッピング）の管理機能等が重複することから，本研究では，これらを単一のフィルタとして実装し，フィルタの doFilter 関数内で，認証状態の有無によって実行する関数を分けることにした．**図 5** に，開発フィルタ（AuthnMethodFilter）の機能概要を示す．なお，引数等の省略を「...」で表現している．また，図 5 では省略したが，文献 [42] の実装と同様に，2.2 節における手順（1）と（2）の間等，LoginContext が存在しない場合や，LoginContext に認証の失敗が記録されている場合に，処理を中止し，後続のフィルタに処理を引き渡すようにした．

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE MappingRules [
  <!ELEMENT MappingRules (MappingRule+)>
  <!ELEMENT MappingRule  (Source, Target)>
  <!ELEMENT Source       (AuthnMethod)>
  <!ELEMENT Target       (AuthnMethod+)>
  <!ELEMENT AuthnMethod  (#PCDATA)>
]>
<MappingRules>
  <MappingRule>
    <Source>
      <AuthnMethod>LoA2</AuthnMethod>
    </Source>
    <Target>
      <AuthnMethod>urn:oasis:names:tc:SAML:2.0:ac:
classes:TLSClient</AuthnMethod>
    </Target>
  </MappingRule>
  <MappingRule>
    <Source>
      <AuthnMethod>LoA1</AuthnMethod>
    </Source>
    <Target>
      <AuthnMethod>urn:oasis:names:tc:SAML:2.0:ac:
classes:TLSClient</AuthnMethod>
      <AuthnMethod>urn:oasis:names:tc:SAML:2.0:ac:
classes:PasswordProtectedTransport</AuthnMethod>
    </Target>
  </MappingRule>
</MappingRules>
```

図 6 グループ情報の定義例
Fig. 6 An example of definition of group information.

まず，共通処理として，フィルタの init 関数内で，XML で定義されたグループ情報を取り込む機能を実装した．グループ情報の定義も，文献 [42] の実装と同様に，web.xml の init-param で指定したファイルから取り込めるようにした．ここでは，java.util.HashMap を用いて，変換処理で用いるグループ形式から個別認証方式の形式（複数可）へのマッピング（mappingRules）と，逆変換処理で用いる個別認証方式の形式からグループ形式（複数可）へのマッピング（reverseMappingRules）を生成するようにした．mappingRules，reverseMappingRules ともに，HashMap の key を String 型，value を String 型の List 型とした．

図 6 は，LoA1 を有する認証方式として ID・パスワード認証（PasswordProtectedTransport）を，LoA2 を有する認証方式としてクライアント証明書認証（TLSClient）を提供している IdP における，グループ情報の定義例を示

表 2 mappingRules の例
Table 2 An example of mappingRules.

key	value
LoA2	TLSClient
LoA1	TLSClient, PasswordProtectedTransport

表 3 reverseMappingRules の例
Table 3 An example of reverseMappingRules.

key	value
TLSClient	LoA2, LoA1
PasswordProtectedTransport	LoA1

している．図 6 の例では，SP 側で LoA1 を認証要求中の AuthnContextClassRef として指定するだけでクライアント証明書認証を LoA1 を有する認証方式としても利用できるよう，TLSClient を LoA1 グループにも所属させている．このように，現在の実装では，包含関係に基づくグループ化が必要な場合にも，IdP 管理者自身がその包含関係を満たすようにグループ情報を定義する必要がある．なお，ここでは LoA1，LoA2 という文字列をグループ名称として用いたが，ここには任意の文字列を指定できる．実際，自組織内等，特定のフェデレーション内で独自に定義した分類方法の LoA を使う際には，http://okayama-u.ac.jp/loa1 等，他と競合しない文字列を設定する必要がある．

表 2，**表 3** に，図 6 をグループ情報の定義として用いた場合の mappingRules，reverseMappingRules の内容を示す．なお，表 2，表 3 では，単純化のため，List 型を表現しておらず，本文中と同様，個別認証方式の形式の表記も簡略化している．

次に，変換処理では，後の逆変換処理で利用するため，変換前の要求認証方式の情報（LoginContext 内の requestAuthenticationMethods）を LoginContext 内の別領域に保存するとともに，mappingRules を利用して，LoginContext 内の requestAuthenticationMethods を変換するようにした．LoginContext には，propsMap という汎用的に利用できる java.util.Map が用意されていたため，変換前の requestAuthenticationMethods はここに保存することにした．

また，逆変換処理では，reverseMappingRules を利用して，LoginContext 内の使用認証方式の情報（authenticationMethodInformation）と，先の変換処理において propsMap に保存した変換前の requestAuthenticationMethods との整合をとる処理を実装した．すなわち，変換前の requestAuthenticationMethods が存在し，かつ，authenticationMethodInformation（より詳細には，authenticationMethodInformation 内の authenticationMethod）が変換前の requestAuthenticationMethods に含まれない場合に，個別認証方式の形式からグループ形式への逆変換を試行し，

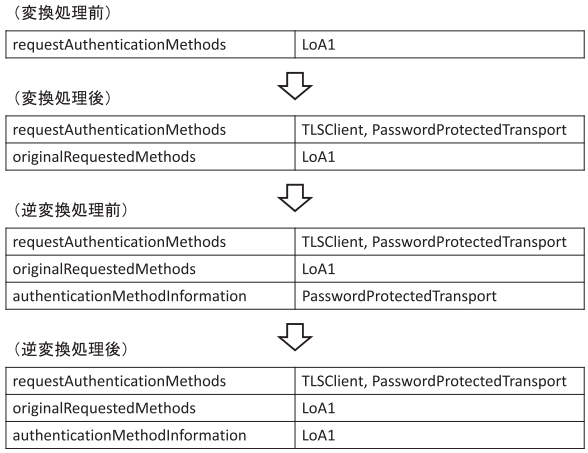

図 7 LoginContext の変化
Fig. 7 Transformations of LoginContext.

reverseMappingRules に基づく逆変換の候補が変換前の requestAuthenticationMethods に含まれる場合に，逆変換を実施するようにした．たとえば，図 6 の例で LoA1 を有する認証方式としてクライアント証明書による認証が行われた場合，authenticationMethodInformation は TLSClient となる．これは変換前の requestAuthenticationMethods に含まれていないため，reverseMappingRules を利用した逆変換が試行される．reverseMappingRules に基づく逆変換の候補としては，LoA1，LoA2 があげられるが，この場合は変換前の requestAuthenticationMethods と一致する LoA1 への逆変換が行われる．なお，reverseMappingRules に基づく逆変換の候補も変換前の requestAuthenticationMethods に含まれない場合には，認証要求とは異なる方式で認証されたことになるため，認証エラーとするようにした[*5]．これは，要求を満たさない方式により認証された結果に基づく認証応答の生成を防ぐための機能であり，2.2 節で触れたように，従来の Shibboleth IdP でも，同様の機能が実装されている．また，複数の逆変換候補が変換前の requestAuthenticationMethods に含まれる場合，現在の実装では，グループ定義において先に定義された候補に逆変換している．

図 7 に，先ほどと同様，図 6 の例で LoA1 を有する認証方式としてクライアント証明書による認証が行われた場合の LoginContext の内容の変化を示す．図 7 では，単純化のため，変換が行われる箇所のみを図示している．また，List 型等の表現をしておらず，本文中と同様，個別認証方式の形式の表記も簡略化している．なお，originalRequestedMethods は変換前の requestAuthenticationMethods を保存するために本実装で指定した propsMap の key である．

図 7 に示すように，認証要求から生成されたグループ形式の requestAuthenticationMethods は，変換処理におい

[*5] すでに 2.2 節における手順（6）の AuthenticationEngine で確認されているが，他のフィルタによる書き換え等を考慮し，安全のため再確認している．

表 4 IdP と SP の仕様
Table 4 Specification of IdP and SPs.

役割	OS とソフトウェアのバージョン
IdP	CentOS release 6.5
	Apache httpd 2.2.15
	Apache Tomcat 6.0.24
	OpenJDK 1.7.0_79
	Shibboleth IdP V2.4.4
	uApproveJP 2.5.0
SP	CentOS release 6.5
	Apache httpd 2.2.15
	Shibboleth SP V2.5.4

て，originalRequestedMethods として保存された後に，個別認証方式の形式に変換される．この結果，2.2 節における手順（1）の認証要求内で，個別認証方式の形式で認証方式を指定していた場合と同様の動作をする．また，逆変換処理前である 2.2 節における手順（6）の段階では，実際に利用された認証方式を保持する authenticationMethodInformation，要求された RequestedAuthnContext の内容を保持する requestAuthenticationMethods のどちらの値も個別認証方式の形式であるため，異なる方式で認証されたことに起因する認証エラーは発生しない．一方，逆変換処理後である 2.2 節における手順（8）の段階では，authenticationMethodInformation がグループ形式に逆変換されているため，グループ形式で認証応答が生成される．

3.4 動作確認

3.3 節で述べた機能を有するフィルタを IdP に組み込み，動作確認を行った．動作確認に用いた IdP と SP の仕様を**表 4** に示す．なお，SP については標準の設定ファイルで変更できる設定を除き，特別なカスタマイズは加えていない．また，大学等の学術機関での実運用を考慮し，そのような国内機関で広く採用されている uApproveJP を導入した状態で，動作確認を行った [43]．以降では uApproveJP に関する記述を省略したが，開発フィルタが uApproveJP の動作に影響を与えないことを確認した．さらに，IdP の動作に関するエラー処理として，2.2 節における手順（1）の認証要求が IdP に到着した時点ですでに当該利用者の LoginContext が存在している場合に，その LoginContext を削除するフィルタを導入した状態で，動作確認を行った．IdP，SP は，ともに，Citrix XenServer 6.2.0 上の仮想サーバとして動作させた．

ここでは，3.2 節で述べたタイミングでフィルタが動作するよう，Tomcat の web.xml で定義する url-pattern を/AuthnEngine（2.2 節における手順（3）と（4）の間に対応）および/profile/SAML2/Redirect/SSO（2.2 節における手順（7）と（8）の間に対応），dispatcher を REQUEST とした．なお，url-pattern を/profile/SAML2/Redirect/SSO とすることにより，2.2 節における手順（1）と（2）の間でもフィルタが動作するが，この段階では LoginContext が存在していないため，前述したように，当該フィルタの処理は中止され，後続のフィルタに処理が引き継がれる．また，IdP では，RemoteUserLoginHandler によりクライアント証明書認証（TLSClient），UsernamePasswordLoginHandler により ID・パスワード認証（PasswordProtectedTransport）を提供し，図 6 に示したグループ情報の定義を適用した [44]．さらに，relying-party.xml で定義する DefaultRelyingParty の defaultAuthenticationMethod を PasswordProtectedTransport とすることにより，ID・パスワード認証をデフォルトの認証方式とした [45]．

```
<Location /secure>
  AuthType shibboleth
  ShibCompatWith24 On
  ShibRequestSetting requireSession 1
  ShibRequestSetting authnContextClassRef LoA1
  Require authnContextClassRef LoA1
</Location>
```

図 8 shib.conf の設定例
Fig. 8 An example of configuration of shib.conf.

まず，3 つの SP（SP1，SP2，SP3）を用意し，それぞれ，LoA1，LoA2，PasswordProtectedTransport を認証要求中の AuthnContextClassRef とするように設定した．**図 8** に，この場合の SP1 の shib.conf の一部を示す．まず，ShibRequestSetting authnContextClassRef LoA1 の記載により，利用者のブラウザを介して IdP へリダイレクトされる認証要求中の AuthnContextClassRef を LoA1 とするようにした [46]．ただし，一般的に用いられている署名なしの認証要求を許容する設定では，利用者自身が認証要求を生成した場合等，必ずしも SP 管理者が指定した AuthnContextClassRef を含む認証要求に対する認証応答のみが到着するとは限らない*6．そこで，Require authnContextClassRef LoA1 の記載により，指定外の方式により認証を受けた利用者のリソースへのアクセスを拒否するようにした [48]．認証要求中の AuthnContextClassRef を指定する場合には，つねに同様の設定を行った．なお，認証応答中の AuthnContextClassRef によるアクセス制御は，shibboleth2.xml の AccessControl を用いて実施することや，Web サーバの環境変数を用いてアプリケーション側で実施することもできるが，今回は最も簡単な Require による方法を用いた [49]．また，shibboleth2.xml の設定等を含め，その他については一般的な SP の設定と同じにした．この状態で，SP1，SP2，SP3 の順にそれぞれのリソースにアクセスした．

*6 動作確認に用いた Shibboleth SP では，今のところ InResponseTo を用いた確認は行われない [47]．

まず，SP1 にアクセスすると，LoA1 を有する認証方式として ID・パスワード認証が選択され，認証画面（ID・パスワード入力画面）が表示された．これは，Shibboleth IdP では，すでに認証済みで有効なセッションが残っている認証方式を選択できない場合，デフォルトの認証方式が選択されるためである [44], [50]．ID・パスワードを入力し，SP1 のリソースにアクセスできることを確認後，SP2 にアクセスすると，LoA2 を有する認証方式としてクライアント証明書認証が選択され，認証画面（クライアント証明書選択画面）が表示された．これは，LoA2 を有する認証方式に，すでに認証済みの ID・パスワード認証が含まれず，クライアント証明書認証のみが含まれているためである．クライアント証明書を選択し，SP2 のリソースにアクセスできることを確認後，SP3 にアクセスすると，すでに ID・パスワード認証済みであるため，実際の認証が省略され，SP3 のリソースにアクセスできた．

引き続き，SP1，SP2，SP3 を，それぞれ，TLSClient，LoA2，LoA1 を認証要求中の AuthnContextClassRef とするように設定し，保存されたクッキーを削除するためブラウザを再起動後，先ほどと同様に，SP1，SP2，SP3 の順にそれぞれのリソースにアクセスした．

まず，SP1 にアクセスすると，クライアント証明書認証が選択され，認証画面（クライアント証明書選択画面）が表示された．これは，TLSClient を直接指定したためである．クライアント証明書を選択し，SP1 のリソースにアクセスできることを確認後，SP2 にアクセスすると，すでにクライアント証明書認証済みであるため，実際の認証が省略され，SP2 のリソースにアクセスできた．その後，SP3 にアクセスすると，同様に，すでにクライアント証明書認証済みであるため，実際の認証が省略され，SP3 のリソースにアクセスできた．

このように，個別認証方式の形式による指定とグループ形式による指定を併用した場合にも，個別認証方式単位での厳密なシングルサインオンを実現できることを確認した．

また，詳細な手順は省略するが，認証要求中の AuthnContextClassRef を指定しなかった場合，存在しない認証方式を要求した場合のいずれも，開発フィルタを用いない通常の Shibboleth IdP と同様に動作することを確認した．これは，当該条件では開発フィルタにおいて，何の変換，逆変換も行われないためである．

次に，利用者の不正な操作による認証方式の変更に対する耐性を確認した．

まず，SP1 を LoA2 を認証要求中の AuthnContextClassRef とするように設定し，保存されたクッキーを削除するためブラウザを再起動後，SP1 のリソースにアクセスした．すると，LoA2 を有する認証方式としてクライアント証明書認証が選択され，認証画面（クライアント証明書選択画面）が表示された．この状態で，いったん認証をキャンセルし，直接の URL 指定により ID・パスワード認証の認証画面（ID・パスワード入力画面）を表示させた．ID・パスワードを入力して認証を済ませると，LoA2 の指定によりクライアント証明書認証が要求されているにもかかわらず ID・パスワード認証が実施されたため，Shibboleth IdP の標準機能が活用され，認証エラーとなった．

また，同様の手順でクライアント証明書認証をキャンセルした後，LoA1 を AuthnContextClassRef とする認証要求を手動で生成し直し，ブラウザの URL 入力欄を用いて直接 IdP へ認証要求を送り直したところ，最初の手順による確認時と同様に，LoA1 を有する認証方式として ID・パスワード認証が選択され，認証画面（ID・パスワード入力画面）が表示された．ID・パスワードを入力して認証を済ませると，SP 側で 403 Forbidden が表示された．これは，SP 側で Require authnContextClassRef LoA2 の設定が有効に働いたためである．

最後に，開発フィルタに関する web.xml の設定をコメントアウトし，IdP のアプリケーションを再起動した場合の動作を確認した．SP1，SP2 を，それぞれ，TLSClient，LoA2 を認証要求中の AuthnContextClassRef とするように設定し，保存されたクッキーを削除するためブラウザを再起動後，SP1，SP2 の順にそれぞれのリソースにアクセスした．まず，SP1 にアクセスすると，クライアント証明書認証が選択され，認証画面（クライアント証明書選択画面）が表示された．クライアント証明書を選択し，SP1 のリソースにアクセスできることを確認後，SP2 にアクセスすると，LoA2 を有する認証方式が存在しないため，認証エラーとなった．これは Shibboleth IdP の標準動作であり，web.xml のコメントアウトのみで開発フィルタを無効化できることを示している．

4. 考察と今後の課題

最後に，本研究に対する考察と今後の課題を述べる．

まず，本研究では，同一の認証フェデレーション内では IdP・SP 間でグループそのものの定義が共通化されていることを前提としている．そのため，IdP・SP 間で解釈の違いを発生させないよう，明確なルール作りが必要となり，初期コストは高くなる．しかしながら，特定 IdP の事情により新たに文献 [21] の初期リストに含まれない認証方式を導入した結果をフェデレーション内で共有する必要がなくなり，異なる組織間の連携を疎に保てるという利点がある．

次に，本研究では，グループ情報定義の汎用性を優先し，グループにレベルの概念を持たせなかった結果，LoA に基づくグループを作成する場合等，完全な包含関係を持つグループを作成する場合に，同一の認証方式を複数のグループに所属させる必要が生じている．IdP 管理者の作業の手間や作業ミスの発生を考慮すると，あるグループを別のグループのメンバとして所属させる等，完全な包含関係を定

義できるような拡張を検討する必要がある．

また，3.3 節で述べたように，現在の実装では，複数の逆変換候補が変換前の requestAuthenticationMethods に含まれる場合，グループ定義において先に定義された候補に逆変換しているが，この点についても検討が必要である．SAML 標準では，認証要求中の AuthnContextClassRef が複数存在する場合，その記載順に認証を試行することが規定されているため，これに準じて，認証要求中の AuthnContextClassRef に最初に現れる候補に逆変換する方法が考えられる [30]．一方で，図 6 の例のように，1 つの認証方式が複数のグループに所属する場合，レベルが上位のグループ等，IdP 管理者がより適切と考える候補に逆変換することも考えられる．

さらに，本研究では，LoginContext の内容を書き換えることによってグループ形式から個別認証方式の形式への変換，個別認証方式の形式からグループ形式への逆変換を行っているが，他のフィルタでも同様に LoginContext の内容を書き換える場合，フィルタの適用順序に注意する必要がある．たとえば，岡山大学では，特定サブネットから特定 SP への認証要求が発生した場合に使用する認証方式を変更するためのフィルタを運用している [51]．既存のフィルタは個別認証方式の形式に対して本研究と同様の変換，逆変換処理を行うため，この場合，本研究の開発フィルタによる変換処理，既存のフィルタによる変換処理，既存のフィルタによる逆変換処理，本研究の開発フィルタによる逆変換処理の順に動作するように，web.xml を設定する必要がある．

最後に，2.1 節で述べたように，Shibboleth IdP には RequestedAuthnContext に「exact」以外の Comparison を含む認証要求を RequestedAuthnContext がまったく指定されていない認証要求として扱う問題があるが，本研究の開発フィルタでもこの問題は解決されていない．開発フィルタの有用性を高めるためにはこの問題に対する対処も必要である．

5. おわりに

本研究では，Shibboleth を用いた認証フェデレーションにおいて，実用的な運用レベルで LoA 等を考慮した認証連携を行うため，Shibboleth IdP における認証方式のグループ化機能を開発した．

まず，Shibboleth の動作原理と Shibboleth IdP における認証処理の概要を述べ，既存の Shibboleth IdP において LoA 等を考慮した認証連携を実現するためには，SP 管理者が，連携先の全 IdP で提供されている認証方式に関する詳細な情報を事前に把握しておく必要がある等，運用上の問題があることを明らかにした．

この問題を解決するため，本研究では，既存の認証機能による個別認証方式の形式による認証処理を維持したまま，グループ形式による認証方式の要求による認証連携を実現するためのフィルタを開発した．本研究では，既存の認証機能による処理が，LoginContext の内容に基づき逐次実行される点に着目し，LoginContext の内容を矛盾なく書き換えることによって，これを実現した．また，開発フィルタが期待どおりに動作することを確認した．

今後は，本稿で焦点を当てた SP 管理者の意向だけでなく，IdP 管理者，利用者の意向を組み合わせて反映させた認証方式の選択を実現する方式への拡張を進める予定である．また，2014 年 12 月にリリースされた Shibboleth IdP V3 についてもより詳細な調査を進め，本問題への対応について検討を進める予定である．

謝辞 本研究の一部は JSPS 科研費 26330158 の助成を受けたものである．

参考文献

[1] 総務省：関係情報：情報通信関連：情報通信白書平成 26 年版 (online)，入手先 ⟨http://www.soumu.go.jp/johotsusintokei/whitepaper/h26.html⟩（参照 2015-06-02）．

[2] 秋山豊和，寺西裕一，岡村真吾，坂根栄作，長谷川剛，馬場健一，中野博隆，下條真司，長岡 亨：大阪大学における全学 IT 認証基盤の構築，情報処理学会論文誌，Vol.49, No.3, pp.1249–1264 (2008).

[3] 内藤久資，梶田将司，小尻智子，平野 靖，間瀬健二：大学における統一認証基盤としての CAS とその拡張，情報処理学会論文誌，Vol.47, No.4, pp.1127–1135 (2006).

[4] 河野圭太，藤原崇起，大隅淑弘，岡山聖彦，山井成良，稗田隆：岡山大学における生涯 ID を実現する統合認証システムの構築，学術情報処理研究，No.15, pp.171–175 (2011).

[5] 西村浩二：広島大学におけるクラウドサービス利用と認証連携，第 8 回統合認証シンポジウム，pp.25–36 (2015).

[6] 上田 浩，古村隆明，石井良和，外村孝一郎，植木徹：Office365 への移行と認証連携事例の評価，大学 ICT 推進協議会 2013 年度年次大会，W3E-6 (2013).

[7] 伊藤智博，立花和宏，奥山澄雄，高野勝美，田島靖久，吉田浩司：ADFS による学術認証フェデレーション対応 SharePoint サービスの構築，学術情報処理研究，No.16, pp.33–40 (2012).

[8] Norris, W.: Achieving Single Sign-on with Google Apps and Shibboleth 2.0 (online), available from ⟨https://shibboleth.usc.edu/docs/google-apps/⟩ (accessed 2015-06-02).

[9] REFEDS: Federations | REFEDS (online), available from ⟨https://refeds.org/federations⟩ (accessed 2015-06-02).

[10] Maler, E., Nadalin, A., Reed, D., Rundle, M. and Thibeau, D.: The Open Identity Trust Framework (OITF) Model (online), available from ⟨http://openidentityexchange.org/wp-content/uploads/the-open-identity-trust-framework-model-2010-03.pdf⟩ (accessed 2015-06-02).

[11] 国立情報学研究所：学術認証フェデレーション 学認 GakuNin (online)，入手先 ⟨https://www.gakunin.jp/⟩（参照 2015-06-02）．

[12] OASIS: OASIS Security Services (SAML) TC | OASIS (online), available from ⟨https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security⟩ (accessed 2015-06-02).

[13] Shibboleth Consortium: Shibboleth (online), available from ⟨https://shibboleth.net/⟩ (accessed 2015-06-02).
[14] 只木進一，江藤博文，大谷 誠，渡辺健次：認証基盤の効率化と「学認」への対応，情報処理学会研究報告，Vol.2012-IOT-17, No.10 (2012).
[15] 松平拓也，笠原禎也，高田良宏，東 昭孝，二木 恵，森 祥寛：大学における Shibboleth を利用した統合認証基盤の構築，情報処理学会論文誌，Vol.52, No.2, pp.703–713 (2011).
[16] 足立紘亮，新村正明：複数の IdP へのシングルサインオンを可能にする認証システムの提案，情報処理学会研究報告，Vol.2011-IOT-13, No.17 (2011).
[17] 中國真教：Shibboleth と OpenAM を組み合わせたハイブリッド型シングルサインオン認証基盤の構築，第 6 回統合認証シンポジウム，pp.77–96 (2012).
[18] 山地一禎：学認の利用価値を高めるサービス連携最新動向，第 8 回統合認証シンポジウム，pp.69–85 (2015).
[19] 秋山豊和，西村 健，山地一禎，中村素典：Web ブラウザ拡張機能を用いた認証連携基盤の機能拡張フレームワークの提案，電子情報通信学会技術研究報告，IA2012-3, pp.13–18 (2012).
[20] 長谷川孝博，松村宣顕，高橋秀年，井上春樹：大学情報基盤におけるパスワード定期更新の運用と利用者動向，学術情報処理研究，No.17, pp.107–114 (2013).
[21] Kemp, J., Cantor, S., Mishra, P., Philpott, R. and Maler, E.: Authentication Context for the OASIS Security Assertion Markup Language (SAML) V2.0 (online), available from ⟨http://docs.oasis-open.org/security/saml/v2.0/saml-authn-context-2.0-os.pdf⟩ (accessed 2015-09-08).
[22] ITU-T: Entity authentication assurance framework, Recommendation ITU-T X.1254 (2012).
[23] Chehab, M.I. and Abdallah, A.E.: Assurance in Identity Management Systems, *Proc. IAS 2010*, pp.216–221 (2010).
[24] Shibboleth Consortium: Shibboleth Identity Provider V3.0.0 Released (online), available from ⟨http://shibboleth.net/pipermail/announce/2014-December/000092.html⟩ (accessed 2015-06-02).
[25] Shibboleth Consortium: Shibboleth IdP V2 End-of-Llife dates (online), available from ⟨http://shibboleth.net/pipermail/announce/2015-May/000112.html⟩ (accessed 2015-06-02).
[26] Shibboleth Consortium: Shibboleth Consortium – What's Shibboleth (online), available from ⟨https://shibboleth.net/about/⟩ (accessed 2015-06-02).
[27] Shibboleth Consortium: Shibboleth Consortium – How Shibboleth Works (online), available from ⟨https://shibboleth.net/about/basic.html⟩ (accessed 2015-06-02).
[28] Hughes, J., Cantor, S., Hodges, J., Hirsch, F., Mishra, P., Philpott, R. and Maler, E.: Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0 (online), available from ⟨http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf⟩ (accessed 2015-09-08).
[29] 河野圭太，中村素典：Shibboleth IdP における LoA を考慮した認証方式グループ化機能の開発，情報処理学会研究報告，Vol.2014-IOT-26, No.2 (2014).
[30] Cantor, S., Kemp, J., Philpott, R. and Maler, E.: Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0 (online), available from ⟨http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf⟩ (accessed 2015-09-08).
[31] Shibboleth Consortium: IdPAuthUserPass – Shibboleth 2 – Confluence (online), available from ⟨https://wiki.shibboleth.net/confluence/display/SHIB2/IdPAuthUserPass⟩ (accessed 2015-09-15).
[32] Shibboleth Consortium: IdPAuthRemoteUser – Shibboleth 2 – Confluence (online), available from ⟨https://wiki.shibboleth.net/confluence/display/SHIB2/IdPAuthRemoteUser⟩ (accessed 2015-09-15).
[33] Shibboleth Consortium: [java-shib-idp2] Contents of /tags/2.4.4/src/main/java/edu/internet2/middleware/shibboleth/idp/authn/Saml2LoginContext.java (online), available from ⟨http://svn.shibboleth.net/view/java-shib-idp2/tags/2.4.4/src/main/java/edu/internet2/middleware/shibboleth/idp/authn/Saml2LoginContext.java?view=markup⟩ (accessed 2015-09-04).
[34] Shibboleth Consortium: IdPAuthnSession – Shibboleth 2 – Confluence (online), available from ⟨https://wiki.shibboleth.net/confluence/display/SHIB2/IdPAuthnSession⟩ (accessed 2015-06-02).
[35] Shibboleth Consortium: [java-identity-provider] Contents of /tags/3.1.1/idp-conf/src/main/resources/conf/authn/authn-comparison.xml (online), available from ⟨http://svn.shibboleth.net/view/java-identity-provider/tags/3.1.1/idp-conf/src/main/resources/conf/authn/authn-comparison.xml?view=markup⟩ (accessed 2015-09-16).
[36] 松平拓也：金沢大学統合認証基盤（KU-SSO）更新に向けた取り組み—クライアント証明書の活用，第 8 回統合認証シンポジウム，pp.37–53 (2015).
[37] 松平拓也：Shibboleth 用多要素認証導入のための技術ガイド (online)，入手先 ⟨https://www.gakunin.jp/?action=pages_view_main&active_action=repository_view_main_item_detail&item_id=227&item_no=1&page_id=85&block_id=227⟩ (accessed 2015-06-02).
[38] Internet2 Wiki: Multi-Context Broker – InCAssurance – Internet2 Wiki (online), available from ⟨https://spaces.internet2.edu/display/InCAssurance/Multi-Context+Broker⟩ (accessed 2015-06-16).
[39] The Apache Software Foundation: Apache Tomcat – Welcome! (online), available from ⟨http://tomcat.apache.org/⟩ (accessed 2015-06-02).
[40] Shibboleth Consortium: SourceAccess - Shibboleth 2 – Confluence (online), available from ⟨https://wiki.shibboleth.net/confluence/display/SHIB2/SourceAccess⟩ (accessed 2015-06-02).
[41] SWITCH: /src/main/java/ch/SWITCH/aai/uApprove – リポジトリ – uApprove – SWITCH Forge (online), available from ⟨https://forge.switch.ch/redmine/projects/uapprove/repository/show/src/main/java/ch/SWITCH/aai/uApprove⟩ (accessed 2015-06-02).
[42] 国立情報学研究所：FPSP（Filter Per SP，ユーザに対する特定 SP へのアクセス制限）プラグイン – GakuNinShibInstall – meatwiki (online)，入手先 ⟨https://meatwiki.nii.ac.jp/confluence/pages/viewpage.action?pageId=12158554⟩（参照 2015-06-02）.
[43] 国立情報学研究所：ユーザ同意取得システム：uApproveJP (Jet Pack) – GakuNinShibInstall – meatwiki (online)，入手先 ⟨https://meatwiki.nii.ac.jp/confluence/pages/viewpage.action?pageId=13501031⟩（参照 2015-06-02）.
[44] Shibboleth Consortium: IdPUserAuthn – Shibboleth 2 – Confluence (online), available from ⟨https://wiki.shibboleth.net/confluence/display/SHIB2/IdPUserAuthn⟩ (accessed 2015-06-02).
[45] Shibboleth Consortium: IdPRelyingParty – Shibboleth 2 – Confluence (online), available from ⟨https://wiki.shibboleth.net/confluence/display/SHIB2/

IdPRelyingParty⟩ (accessed 2015-09-08).
[46] Shibboleth Consortium: NativeSPContentSettings – Shibboleth 2 – Confluence (online), available from ⟨https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPContentSettings⟩ (accessed 2015-09-07).
[47] Shibboleth Consortium: [SSPCPP-604] Web SSO Profile: InResponseTo attribute is not validated – Shibboleth JIRA (online), available from ⟨https://issues.shibboleth.net/jira/browse/SSPCPP-604⟩ (accessed 2015-09-17).
[48] Shibboleth Consortium: NativeSPhtaccess – Shibboleth 2 – Confluence (online), available from ⟨https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPhtaccess⟩ (accessed 2015-09-07).
[49] Shibboleth Consortium: NativeSPXMLAccessControl – Shibboleth 2 – Confluence (online), available from ⟨https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPXMLAccessControl⟩ (accessed 2015-09-24).
[50] Shibboleth Consortium: [java-shib-idp2] Contents of /tags/2.4.4/src/main/java/edu/internet2/middleware/shibboleth/idp/authn/AuthenticationEngine.java (online), available from ⟨http://svn.shibboleth.net/view/java-shib-idp2/tags/2.4.4/src/main/java/edu/internet2/middleware/shibboleth/idp/authn/AuthenticationEngine.java?view=markup⟩ (accessed 2015-09-04).
[51] 河野圭太，藤原崇起，稗田 隆：岡山大学事務情報システムにおける Shibboleth との連携を考慮した多要素認証の導入，情報処理学会研究報告，Vol.2014-IOT-27, No.5 (2014).

河野 圭太 （正会員）

平成 12 年大阪大学工学部電子情報エネルギー工学科卒業．平成 14 年同大学大学院工学研究科博士前期課程修了．平成 16 年同大学院情報科学研究科博士後期課程を修了し，同年岡山大学総合情報基盤センター助手．平成 19 年同センター助教，平成 22 年同大学情報統括センター助教を経て，平成 23 年同センター准教授．博士（情報科学）．モバイルネットワーク，分散システムの研究に従事．IEEE，電子情報通信学会各会員．

中村 素典 （正会員）

1994 年京都大学大学院工学研究科博士後期課程単位取得退学．立命館大学理工学部助手，京都大学経済学部助教授，京都大学学術情報メディアセンター助教授等を経て，2007 年より国立情報学研究所特任教授，現在に至る．博士（工学）．IEEE，日本ソフトウェア科学会，電子情報通信学会各会員．コンピュータネットワーク，ネットワークコミュニケーション，認証連携等の研究に従事．