# MFA et 2FA dans l'IdP Shibboleth, le serveur CAS d'Apereo et les Fédérations

**Guillaume Rousse**
RENATER
23-25, rue Daviel
75013 Paris

**Ludovic Auxepaules**
RENATER
23-25, rue Daviel
75013 Paris

## Résumé

*Dans cette présentation, nous abordons l'authentification forte dans le contexte des deux principaux systèmes d'authentification utilisés dans la fédération Éducation-Recherche : l'Identity Provider (IdP) de Shibboleth et le serveur CAS d'Apereo. Nous y exposons tout d'abord les choix, les cas d'usage qui justifient la mise en place de l'authentification renforcée. Nous précisons également les contraintes et les difficultés liées au renforcement de l'authentification à l'échelle des fédérations d'identités. Ensuite, nous décrivons différentes mises en œuvre possibles de l'authentification renforcée dans l'IdP Shibboleth et dans le serveur CAS, ainsi que les profils publiés par REFEDS (the Research and Education FEDerations group). Enfin, nous concluons par les avancées introduites par WebAuthn (Web Authentication) du W3C en 2019.*

## Mots-clefs

*MFA, authentification, CAS, IdP, Shibboleth, fédération, sécurité*

# 1 Introduction

## 1.1 Authentification renforcée

L'utilisation d'un couple identifiant/mot de passe comme moyen d'authentification a l'avantage d'être simple et universelle. Elle a malheureusement le défaut d'être fragile : en effet ces mots de passe se devinent (attaques par force brute), se volent à l'utilisateur, directement (attaques par hameçonnage) ou indirectement par compromission d'un service externe pour lequel l'utilisateur aura utilisé le même mot de passe à différents endroits (vols de données). Cette seule technique n'est ainsi pas adaptée à un usage sensible, qui exige plus de garanties de sécurité.

Il est possible de remplacer l'utilisation d'un mot de passe par un autre moyen d'authentification plus robuste, par exemple un certificat numérique. Mais face aux difficultés logistiques d'une part, de conduite du changement d'autre part, il est généralement préférable de combiner plusieurs moyens d'authentification, plutôt que de simplement substituer un moyen par un autre. On parle alors de facteurs[1] d'authentification, qui peuvent être classés en plusieurs catégories :

- **un facteur mémoriel** correspond à ce que l'utilisateur connaît : un code PIN, un mot de passe, etc. ;

1 au sens mathématique du terme, ce sont les éléments constitutifs d'un tout.

- **un facteur matériel** correspond à ce que l'utilisateur possède : une clé U2F (*Universal Second Factor*), un certificat numérique sur carte à puce, etc. ;
- **un facteur corporel** correspond à ce qu'est l'utilisateur : une empreinte digitale ou rétinienne, par exemple ;
- **un facteur réactionnel** correspond à ce que l'utilisateur sait faire : sa signature, par exemple.

Ces facteurs peuvent faire intervenir différentes technologies, telles que les jetons de sécurité (ou *tokens*) qu'ils soient matériels ou logiciels, l'authentification mobile (SMS, appels téléphoniques, applications OTP…), les badges (cartes à puce ou magnétique), le GPS…

On parle en général d'authentification multi-facteurs (MFA, pour *Multi Factor Authentication*), ou authentification forte, dès que celle-ci fait intervenir au moins deux facteurs (2FA) de catégories différentes.

Cette authentification peut également être adaptée en fonction du contexte : en nécessitant ou pas le recours à un second facteur en fonction de critères, tels que le lieu, l'horaire, le périphérique utilisé...

## 1.2 Service d'authentification

Un service d'authentification permet d'externaliser un besoin fonctionnel, celui d'authentifier les utilisateurs, de façon à le mutualiser entre plusieurs applications. Deux briques logicielles sont couramment utilisées pour jouer ce rôle dans la communauté française de l'enseignement supérieur et de la recherche :

- le serveur d'authentification unique CAS d'Apereo ;
- le fournisseur d'identité (*Identity Provider*) Shibboleth.

S'ils tendent aujourd'hui à devenir plus ou moins équivalents fonctionnellement [1], ces deux logiciels correspondent historiquement à deux cas d'utilisation distincts de délégation de l'authentification d'une application à un service d'authentification :

- une application cliente délègue l'authentification à **un seul service d'authentification**, car les utilisateurs proviennent d'un établissement unique (généralement, le même que celui qui met en œuvre le service) ;
- une application cliente délègue l'authentification à **plusieurs services d'authentification**, car les utilisateurs proviennent d'un ensemble d'établissements ; cet ensemble évolue le plus souvent dynamiquement hors du contrôle de l'application.

Seul le deuxième cas d'usage justifie à nos yeux l'appellation de service fédéré, et s'il arrive que des applications correspondant au premier cas soient inscrites par facilité dans une fédération d'identité, c'est à la fois inutile et surtout dangereux du point de vue de la sécurité[2] : une simple relation bilatérale est amplement suffisante. Nous parlerons donc d'une application non fédérée pour le premier cas et d'une application fédérée (ou service fédéré) pour le deuxième.

Ces deux cas d'usage sont en fait différents du point de vue de la gouvernance, des possibilités techniques, et des conséquences de la mise en œuvre de l'authentification renforcée. Nous allons donc chercher à répondre aux questions suivantes, pour chacun de ces deux contextes :

- Qui est légitime pour décider de l'usage de l'authentification forte ?
- Qui est pertinent pour prendre cette décision ?
- Qui est capable techniquement d'imposer cette décision ?

2 Même s'il reste possible de mettre en place un contrôle d'accès pour limiter les fournisseurs d'identité autorisés au niveau applicatif, il est dommage de se priver de la frontière de sécurité établie par la limitation des entités autorisées : inscrire une application dans une fédération, c'est accepter de communiquer avec tous les fournisseurs d'identité qui y sont enregistrés.

# 2 Cas d'usage du MFA

L'authentification forte est une modalité d'authentification, parmi d'autres, et l'authentification est une mesure de sécurité, parmi d'autres. Or, une mesure de sécurité n'a qu'un seul intérêt[3], celui de diminuer un risque, alors qu'elle a aussi un coût, humain, financier, ou les deux à la fois, et ce coût pèse à la fois sur les administrateurs et les utilisateurs du service d'authentification. Il y a donc un rapport coût/bénéfice à évaluer, par le biais d'une analyse de risque, et une décision à prendre sur le besoin d'une telle mesure, avant même de s'intéresser à la faisabilité technique.

## 2.1 Application non fédérée

Le cas d'une application non fédérée rejoint celui de n'importe quelle application interne à un établissement, et fait intervenir les deux acteurs classiques, un prescripteur, généralement le RSSI, et un réalisateur, généralement la DSI.

Le RSSI réalise l'analyse sécurité, et décide ensuite si une authentification renforcée est nécessaire, sur quels périmètres et sur quels critères. Ces critères sont par exemple :

- une application, en fonction des données qu'elle manipule ;
- un ou plusieurs utilisateurs, en fonction des privilèges qu'ils possèdent ;
- un contexte physique ou de lieu, tel que l'utilisation depuis les locaux de l'établissement ou sur des périphériques autorisés, depuis « l'extérieur » ou l'étranger ;
- un contexte temporel, tel que l'utilisation en dehors des heures de travail (le soir, le week-end) ;
- une combinaison de ces différents critères.

La DSI met en œuvre ces exigences de sécurité, et propose éventuellement des mesures compensatoires dans le cas où la mise en œuvre est difficile à mener (en matière de conduite du changement, de support, de gestion des exceptions...).

Le RSSI peut éventuellement ensuite mettre en place un processus de contrôle, afin de vérifier que la mise en œuvre est conforme et que d'éventuels problèmes sont identifiés et palliés.

Un cas d'usage classique d'authentification multi-facteur consiste par exemple à renforcer la sécurité de l'accès au webmail institutionnel pour tout ou partie des utilisateurs, mais uniquement lorsqu'il est utilisé depuis le réseau externe de l'établissement.

Dans ce contexte, ni la légitimité, ni la pertinence de ces choix ne posent de difficultés ; les possibilités de mise en œuvre sont vastes. Au moins trois stratégies peuvent être envisagées :

- bloquer tous les accès directs au webmail depuis le réseau externe, et imposer l'usage d'un VPN, dont l'accès nécessite l'utilisation d'un moyen d'authentification différent de celui du webmail (un certificat électronique par exemple) ;
- mettre en place la logique nécessaire du côté de l'application, ce qui exige un support protocolaire pour communiquer une exigence variable au service d'authentification ;
- mettre en place la logique nécessaire du côté du service d'authentification.

La première stratégie est sans doute la plus simple à mettre en œuvre techniquement, dans la mesure où elle ne demande aucune configuration avancée, mais elle demande une évolution des pratiques des utilisateurs. Nous présentons des exemples des deux autres stratégies plus loin dans cet article.

Cette situation est simple, parce que tous les acteurs impliqués relèvent d'une autorité unique, en général le directeur de cet établissement. Ils peuvent se réunir pour se concerter, s'accorder sur la sémantique des

3 Si l'on exclut celui de pouvoir présenter des articles aux JRES...

exigences, identifier des problèmes, trouver des solutions, et en cas de désaccord, rechercher un arbitrage auprès de l'autorité hiérarchique.

## 2.2 Application fédérée

En revanche, dans le contexte d'une fédération d'identité, le modèle précédent est plus difficilement applicable. En effet, hormis le cas particulier d'une fédération d'identité locale à un établissement, il n'existe plus d'autorité unique, mais une par établissement concerné, qui sont toutes légitimes pour formuler des exigences de sécurité, sur le périmètre dont elles sont responsables, et réaliser ces exigences, à travers les briques techniques qu'elles mettent en œuvre. Il n'y a donc plus deux acteurs, mais autant d'acteurs que d'établissements participant à la fédération, dont chacun assume au moins un des deux rôles suivants :

- **opérateur d'authentification**, par le biais d'un ou plusieurs fournisseurs d'identité ;
- **opérateur de service**, par le biais d'un ou plusieurs fournisseurs de service.

Ces différents acteurs n'ont, le plus souvent, pas de relation organisationnelle ou contractuelle directe entre eux, ce qui limite fortement les interactions possibles. En revanche, ils ont tous une relation contractuelle avec un organisme particulier, celui qui gère la fédération et édicte les règles relatives à son usage. Un troisième rôle apparaît en plus des deux premiers : celui d'opérateur de la fédération.

Nous passons en revue chacun de ces rôles, afin d'illustrer ce qu'ils :

- exigent des autres acteurs d'une part, lorsqu'ils agissent donc comme **prescripteur**,
- doivent faire pour répondre aux exigences des autres d'autre part, lorsqu'ils agissent comme **réalisateur**.

### 2.2.1 L'opérateur de la fédération

L'opérateur de fédération peut être prescripteur en matière d'authentification. Il définit un socle technique minimal pour l'ensemble des participants, incluant notamment des exigences de sécurité. RENATER définit notamment une obligation pour tout fournisseur d'identité, au chapitre « recommandations et obligations » du cadre technique de la fédération de la Fédération Éducation Recherche :

> *La page d'authentification doit être sécurisée par l'utilisation de TLS, avec un certificat valide, délivré par une autorité de certification reconnue* [2].

Dans le contexte d'une fédération généraliste, il paraît difficile d'aller plus loin, et d'exiger, par exemple, la mise en œuvre d'un mécanisme d'authentification forte, comme préalable à toute participation à cette fédération, sans une visibilité sur les différents contextes d'utilisation possibles. Dans une fédération spécialisée, qui ne regrouperait que des participants fortement concernés par les questions de sécurité, comme une fédération de CERTs, cette exigence pourrait être plus pertinente. Ce n'est pas tant la légitimité que la pertinence de cette exigence par rapport aux objectifs de la fédération qui est en cause.

### 2.2.2 L'opérateur de service

L'opérateur d'un service est uniquement prescripteur en matière d'authentification. En définissant des exigences d'authentification préalables à l'accès au service, il est à la fois légitime, car il s'agit de son service, et pertinent, car il connaît une grande partie du contexte, à savoir la sensibilité des informations manipulées. Certains critères supplémentaires, permettant d'évaluer plus précisément le risque, ne lui sont cependant pas accessibles, ou pas complètement :

- les **privilèges possédés** par certains utilisateurs ne concernent que ceux spécifiques à l'application : pas directement les privilèges liés à leur établissement d'origine ;
- les notions de **réseau interne ou externe**, ou d'**heure ouvrée ou non**, sont relatives aux utilisateurs, et donc à leur établissement d'origine.

Il paraît donc possible de formuler une exigence d'authentification renforcée, **soit de manière globale** pour tous les utilisateurs du service, **soit de manière plus précise** pour certaines classes d'utilisateurs uniquement (typiquement les administrateurs du service).

La conséquence d'une telle exigence, néanmoins, sera de bloquer l'accès au service pour les utilisateurs incapables de satisfaire celle-ci, typiquement parce que leur établissement d'origine ne leur fournit pas les moyens d'authentification satisfaisant cette exigence. Si seule une population limitée est visée, l'impact de la mesure est également limité : même pour une application fédérée, les administrateurs proviennent souvent d'un établissement unique. Si en revanche, une part importante des utilisateurs est concernée, le changement est plus impactant, et réduit drastiquement l'audience potentielle du service.

### 2.2.3 L'opérateur d'authentification

Un opérateur d'authentification est principalement réalisateur, dans la mesure où il doit permettre à ses utilisateurs de satisfaire des exigences formulées par des services au sein de la fédération. Cet aspect se limite essentiellement à des choix d'implémentation présentés plus loin dans l'article.

Mais à partir du moment où les utilisateurs concernés relèvent de son autorité, un opérateur d'authentification est également légitime comme prescripteur, en leur imposant sa propre politique de sécurité. Par exemple, en utilisant les critères temporels ou géographiques qui ne sont pas utilisables par un service. Cela peut conduire à un scénario dans lequel un opérateur d'authentification exige de ses utilisateurs une authentification renforcée pour accéder à un service proposé par un tiers qui lui n'exige rien de tel à la base... Que ce scénario résulte d'un choix délibéré, par exemple pour limiter les risques d'usurpation d'identité quel que soit le contexte, ou plus vraisemblablement d'un simple effet de bord après avoir répondu à des exigences internes, la pertinence est plus difficile à établir.

# 3 Aspects protocolaires de SAML et MFA

Les fédérations d'identité utilisées dans notre communauté, et notamment la Fédération Éducation-Recherche, sont basées sur le protocole SAML [3], et plus précisément sa version 2.0 (la version 1.0 n'étant utilisée qu'à des fins de compatibilité). Dans ce protocole, le fournisseur de service (SP pour *Service Provider*) transmet à l'utilisateur une requête d'authentification, et le renvoie vers son fournisseur d'identité (IdP pour *Identity Provider*). Le fournisseur d'identité traite alors cette requête, transmet une réponse à l'utilisateur et le renvoie vers le service demandeur.

## 3.1 Contexte d'authentification SAML

Le fournisseur de service peut formuler des exigences sur les modalités d'authentification souhaitées, en définissant dans sa requête un contexte d'authentification explicite (élément *RequestedAuthnContext*). Le contenu exact de cet élément [4], permet de formuler ce contexte de deux manières différentes.

La forme complexe, basée sur une **déclaration de contexte** (élément *AuthnContextDeclRef*) permet de décrire de façon exhaustive l'intégralité du processus d'authentification, tel que la nature de l'élément confidentiel (clé, mot de passe, code PIN…), les contraintes qui pèsent sur lui (longueur, complexité…), la façon dont il a été transmis initialement à l'utilisateur (par voie électronique, ou lors d'une rencontre en face à face), le moyen de stockage physique (système de fichiers, HSM pour *Hardware Security Module*…), la forme de stockage (en clair, chiffré...), le canal de communication vers le serveur d'authentification (en clair, via un canal sécurisé par TLS…), etc. Mais dans le contexte d'une fédération d'identité, il est peu adapté de formuler des exigences aussi précises, que personne ne saura satisfaire, non pas par incapacité à le faire en réalité, mais par incapacité à le décrire aussi finement dans son fournisseur d'identité. Comme le résume fort simplement l'un des auteurs principaux de cette spécification :

> *Nobody uses those and nobody should use DeclRef, unless they really want to point to a per-transaction document that isn't general at all and pertains to that specific assertion. Nobody does this and so nobody should be using the element* [5].

La forme simple, en revanche, fait appel à des **classes de contexte** (élément *AuthnContextClassRef*), qui sont des constantes normalisées et identifiées par des URIs. Elles correspondent aux cas les plus couramment utilisés. Le standard en définit plusieurs, avec la sémantique associée, dont par exemple :

- *urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport* : authentification par mot de passe, transmis sur un canal sécurisé ;
- *urn:oasis:names:tc:SAML:2.0:ac:classes:TLSClient* : authentification TLS par certificat client ;
- *urn:oasis:names:tc:SAML:2.0:ac:classes:TimeSyncToken* : authentification par OTP temporel (TOTP).

Il n'existe pas de classe unique correspondant à l'authentification multi-facteurs, mais plusieurs correspondent à des combinaisons pré-définies, telles que :

- *urn:oasis:names:tc:SAML:2.0:ac:classes:InternetProtocolPassword* : authentification par l'usage d'une adresse IP et d'un mot de passe ;
- *urn:oasis:names:tc:SAML:2.0:ac:classes:AuthenticatedTelephony* : authentification par l'usage d'un numéro de ligne téléphonique et d'un mot de passe ;
- *urn:oasis:names:tc:SAML:2.0:ac:classes:MobileTwoFactorContract* : authentification par le biais d'un dispositif de signature électronique disposant d'un dispositif sécurisé de stockage des clés (une carte SIM, par exemple), qui demande une preuve explicite de l'identité de l'utilisateur (code PIN, empreinte digitale…).

Cette réduction de l'éventail des valeurs possibles rend l'utilisation de ces classes beaucoup plus simples que la solution précédente. En contrepartie, ces classes ne couvrent pas forcément toutes les méthodes existantes, et toutes les méthodes existantes ne sont pas forcément simples à classifier. Par exemple, il est difficile d'identifier quelle classe correspond le mieux à l'utilisation d'un OTP transmis par SMS comme second facteur d'authentification, alors qu'il s'agit d'une méthode couramment utilisée. La spécification prévoit explicitement la nécessité pour les participants d'une fédération de s'entendre au préalable entre eux sur la liste des classes utilisables, quitte à en ajouter d'autres si nécessaire, et sur les critères d'utilisation de ces classes. La spécification seule n'est pas suffisante pour être pleinement opérationnelle.

## 3.2 Profils SFA et MFA de REFEDS

REFEDS[4], dans le contexte du projet *REFEDS Assurance Suite* [6], a défini une nouvelle notion, celle de **profil d'authentification**, afin de décrire l'association de méthodes d'authentification, d'une part, et de contraintes sur ces méthodes, d'autre part, comme, par exemple, une longueur minimale de mot de passe fixée à 8 caractères pour un alphabet de 72 caractères. Ces profils sont identifiés par des classes d'authentification (au sens de la spécification précédente) de façon à être utilisables dans les échanges SAML :

- *https://refeds.org/profile/sfa* : **une authentification à facteur unique** conforme aux exigences du profil [7] ;
- *https://refeds.org/profile/mfa* : **une authentification multi-facteurs** conforme aux exigences du profil [8].

## 3.3 Exemples concrets

Ces éléments de langage définis, nous présentons l'exemple d'une requête d'authentification exigeant l'utilisation d'une authentification par mot de passe via un canal sécurisé :

```
<AuthnRequest>
  <Issuer>http://sp.example.com</Issuer>
```

4 Le groupe REFEDS (*the Research and Education FEDerations group*) articule les discussions et recueille les besoins de mutualisation relatifs aux fédérations et à la gestion d'identité dans le contexte mondial de l'enseignement et de la recherche.

```
  <RequestedAuthnContext>
    <AuthnContextClassRef>
        urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
    </AuthnContextClassRef>
  </RequestedAuthnContext>
</AuthnRequest>
```

Il est également possible d'être plus libéral, et de spécifier une liste de méthodes possibles, en énumérant les classes par ordre de préférence. Ci-dessous l'exemple d'une requête demandant de préférence une authentification TLS par certificat client, mais acceptant également une authentification par mot de passe via un canal sécurisé :

```
<AuthnRequest>
  <Issuer>http://sp.example.com</Issuer>
  <RequestedAuthnContext>
    <AuthnContextClassRef>
        urn:oasis:names:tc:SAML:2.0:ac:classes:TLSClient
        urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
    </AuthnContextClassRef>
  </RequestedAuthnContext>
</AuthnRequest>
```

Il est même possible d'être encore plus libéral en laissant le fournisseur d'identité décider quelle méthode utiliser pour répondre, non plus dans une liste fermée, mais par comparaison entre différentes méthodes disponibles. Voici l'exemple d'une requête se satisfaisant de n'importe quelle méthode d'authentification plus sécurisée que l'authentification par mot de passe, sur un canal sécurisé :

```
<AuthnRequest>
  <Issuer>http://sp.example.com</Issuer>
  <RequestedAuthnContext Comparison="better">
    <AuthnContextClassRef>
        urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
    </AuthnContextClassRef>
  </RequestedAuthnContext>
</AuthnRequest>
```

Néanmoins, la spécification ne définit aucun ordre entre les classes disponibles, et laisse aux participants l'appréciation de la notion de « plus sécurisée ».

## 3.4 Réponse

Le fournisseur d'identité recevant une requête comportant une exigence sur l'authentification doit produire une réponse conforme, incluant un élément *RequestedAuthn* précisant la classe de contexte effectivement utilisée. Par exemple, la réponse suivante est conforme à une requête demandant l'utilisation du profil REFEDS MFA :

```
<Response>
  <Issuer>http://idp.example.com/metadata.php</Issuer>
  <AuthnStatement>
    <AuthnContext>
      <AuthnContextClassRef>https://refeds.org/profile/mfa</AuthnContextClassRef>
    </AuthnContext>
  </AuthnStatement>
</Response>
```

En cas d'impossibilité de satisfaire la demande, car, par exemple, le fournisseur d'identité n'est pas capable de produire une telle authentification (ou s'il est capable de le faire, mais que sa configuration ne lui permet pas de le savoir, ce qui revient au même), il doit répondre par une erreur d'authentification.

La question immédiate qui se pose du point de vue de la sécurité est de savoir ce qui empêche le fournisseur d'identité de mentir sur la méthode réellement utilisée, puisqu'en agissant ainsi il pourrait favoriser les intérêts de ses propres utilisateurs par rapport à un tiers externe, le fournisseur de service. Techniquement, rien ne l'en empêche, comme rien ne l'empêche non plus de mentir sur le résultat de l'authentification, ou sur les attributs associés à l'utilisateur. Si ceci peut surprendre, il convient de rappeler à cet égard qu'une fédération d'identité est avant tout basée sur une relation de confiance entre participants, et que, si cette confiance n'existe pas, il y a peu d'intérêt à y participer.

# 4 Mise en œuvre dans l'IdP Shibboleth

Nous présentons ici trois exemples d'utilisation de l'authentification renforcée avec l'IdP Shibboleth, via un second facteur fourni par un tiers externe. La solution technique utilisée pour fournir ce second facteur, Duo, ne l'a pas été pour ses qualités propres, qui n'ont pas été évaluées, mais parce que son intégration avec l'IdP Shibboleth est aujourd'hui assurée par le consortium Shibboleth [10], ce qui constitue une garantie de pérennité. Ces différents exemples montrent l'articulation entre les différents éléments du système, et la souplesse de mise en œuvre.

L'IdP Shibboleth gère l'authentification par le biais de flux (*Authentication flows*), qui correspondent à des processus différents (mot de passe, certificat, second facteur, etc.). Chacun de ces flux est étiqueté par une ou plusieurs des classes d'authentifications présentées précédemment. Grâce à ce vocabulaire normalisé, le fournisseur d'identité est capable d'identifier parmi les flux disponibles ceux capables de correspondre aux exigences d'une requête d'authentification émanant d'un fournisseur de services. Si plusieurs flux sont utilisables, le premier est utilisé [11].

## 4.1 Authentification renforcée inconditionnelle

Ce scénario correspond à une application non fédérée, configurée pour n'avoir confiance que dans un IdP unique, exigeant une authentification renforcée quel que soit l'utilisateur et quel que soit le contexte.

Il est inutile de configurer le SP associé à l'application pour formuler des exigences particulières dans ses requêtes d'authentification, puisque l'IdP utilise systématiquement l'authentification renforcée.

L'IdP est configuré pour ne proposer que le flux d'authentification MFA, et la configuration des transitions d'état pour ce flux indique juste d'enchaîner les sous-flux *Password* et *Duo* :

```
<util:map id="shibboleth.authn.MFA.TransitionMap">
    <entry key="">
        <bean parent="shibboleth.authn.MFA.Transition" p:nextFlow="authn/Password" />
    </entry>
    <entry key="authn/password">
        <bean parent="shibboleth.authn.MFA.Transition" p:nextFlow="authn/Duo" />
    </entry>
</util:map>
```

## 4.2 Authentification renforcée conditionnelle

### 4.2.1 Évaluation de la condition coté application

Ce scénario correspond à une application fédérée, exigeant une authentification renforcée pour ses administrateurs, qui proviennent tous de son propre établissement, mais se contentant d'une authentification par mot de passe pour les autres utilisateurs provenant de l'ensemble de la communauté.

L'exemple ci-dessous montre comment un SP Shibboleth associé à cette application peut être configuré pour exiger par défaut une authentification par mot de passe :

```
<SSO discoveryProtocol="SAMLDS" discoveryURL="https://discovery.domain.tld"
    authnContextClassRef=
"urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport"
    authnContextComparison="minimum">SAML2</SSO>
```

Une fois l'utilisateur authentifié, l'application attribue un rôle à l'utilisateur, en fonction de sa propre configuration. Si ce rôle est un rôle d'administrateur, elle examine alors la méthode d'authentification utilisée pour cette première authentification, via la variable *Shib-AuthnContext-Class* transmise implicitement à l'application en sus des attributs demandés [12]. Si le contenu de cette variable ne correspond pas à la valeur requise, alors l'utilisateur est redirigé vers l'IdP qu'il vient d'utiliser pour une seconde authentification, avec cette fois-ci une exigence plus forte, telle que *https://refeds.org/profile/mfa*.

L'IdP n'est pas configuré pour ne proposer que le flux d'authentification MFA, mais celui-ci est configuré pour avoir un comportement variable, en fonction des exigences présentes dans la requête d'authentification, de façon à court-circuiter l'utilisation du second facteur s'il n'est pas demandé, grâce à la stratégie suivante :

```
<util:map id="shibboleth.authn.MFA.TransitionMap">
     <entry key="">
        <bean parent="shibboleth.authn.MFA.Transition"
            p:nextFlow="authn/Password" />
    </entry>
    <entry key="authn/Password">
        <bean parent="shibboleth.authn.MFA.Transition"
            p:nextFlowStrategy-ref="checkSecondFactor" />
    </entry>
</util:map>

<bean id="checkSecondFactor"
    parent="shibboleth.ContextFunctions.Scripted"
    factory-method="inlineScript">
    <constructor-arg>
      <value>
          <![CDATA[
            authCtx = input.getSubcontext(
               "net.shibboleth.idp.authn.context.AuthenticationContext");
            mfaCtx = authCtx.getSubcontext(
              "net.shibboleth.idp.authn.context.MultiFactorAuthenticationContext");
            if (mfaCtx.isAcceptable()) {
              nextFlow = null;
            } else {
              nextFlow = "authn/Duo";
            }
            nextFlow;
        ]]>
      </value>
    </constructor-arg>
</bean>
```

De cette façon, lors de la première requête d'authentification, l'utilisateur se contente de saisir son mot de passe. Lors de la deuxième requête d'authentification, le mécanisme de SSO en place lui évite de le faire une seconde fois afin que seul le second facteur soit à satisfaire.

### 4.2.2 Évaluation de la condition coté authentification

Ce scénario correspond à une application non fédérée, configurée pour n'avoir confiance que dans un IdP unique, exigeant une authentification renforcée depuis le réseau externe à l'établissement.

Il est inutile de configurer le SP pour formuler des exigences particulières, puisque c'est l'IdP qui adapte cette fois-ci son comportement en fonction du contexte.

L'IdP n'est pas configuré pour ne proposer que le flux d'authentification MFA, mais celui-ci est configuré pour avoir un comportement variable, en fonction de l'adresse source de la requête d'authentification, de façon à court-circuiter l'utilisation du second facteur s'il n'est pas nécessaire, grâce à la stratégie suivante :

```
<util:map id="shibboleth.authn.MFA.TransitionMap">
     <entry key="">
        <bean parent="shibboleth.authn.MFA.Transition"
            p:nextFlow="authn/Password" />
    </entry>
    <entry key="authn/Password">
        <bean parent="shibboleth.authn.MFA.Transition"
            p:nextFlowStrategy-ref="checkSecondFactor" />
    </entry>
</util:map>

<bean id="InternalNetwork" class="org.opensaml.profile.logic.IPRangePredicate"
    p:httpServletRequest-ref="shibboleth.HttpServletRequest"
    p:ranges="#{ '192.168.10.0/24' }" />

<bean id="checkSecondFactor"
    parent="shibboleth.ContextFunctions.Scripted"
    factory-method="inlineScript"
    p:customObject-ref="InternalNetwork">
    <constructor-arg>
      <value>
        <![CDATA[
          if (custom.apply(input)) {
            nextFlow = null;
          } else {
            nextFlow = "authn/Duo";
          }
          nextFlow;
        ]]>
      </value>
    </constructor-arg>
</bean>
```

# 5 Mise en œuvre du MFA dans le serveur CAS d'Apereo

Historiquement, le serveur d'authentification CAS est utilisé comme SSO dans de nombreux établissements de l'enseignement supérieur et de la recherche. Il convient donc de s'intéresser aux possibilités offertes par le serveur CAS dans le cadre de l'authentification renforcée.

## 5.1 Possibilités offertes par le MFA dans CAS

Depuis les versions 5, le serveur CAS est configurable de base avec une très grande variété de fournisseurs externes d'authentification multi-facteurs et d'options permettant d'activer ou non une phase d'authentification renforcée. Le serveur CAS peut ainsi enchaîner un second facteur d'authentification (ou plusieurs) après l'étape d'authentification principale si « une condition ou un déclencheur » l'exige. Treize déclencheurs (*triggers*)[5] sont prévus et sont testés selon l'ordre d'exécution suivant [13] :

1. *Adaptive* : règles en fonction du pays, du navigateur, de l'heure, des jours, des adresses IP… ;
2. *Global* : pour toutes les applications et tous les utilisateurs sans distinction ;
3. *Opt-In Request Parameter/Header* : par rapport à un paramètre ou un entête http de la page de login du serveur CAS (paramètre *authn_method* par défaut) ;
4. *REST Endpoint* : en appelant un service REST pour vérifier si un second facteur est nécessaire ;
5. *Groovy Script* : par rapport au résultat d'un script Groovy[6] ;
6. *Principal Attribute Per Application* : en fonction de valeurs d'attribut (exactes ou regex) pour un service donné (approche hybride) ;
7. *Global Principal Attribute Predicate* : en fonction des valeurs d'attributs via un script Groovy ;
8. *Global Principal Attribute :* tous les utilisateurs ayant une valeur d'attribut (exacte ou regex) ;
9. *Global Authentication Attribute* : tous les utilisateurs dont la valeur d'un attribut d'authentification correspond exactement à un fournisseur d'authentification MFA ;
10. *Applications* : pour un ou plusieurs services spécifiques (cette option est la plus courante) ;
11. *Grouper* : en fonction de certains groupes définis dans le gestionnaire de groupes *Grouper* ;
12. *Entity ID Request Parameter :* en fonction de l'*entity-id* du SP (exact ou regex) lors de l'authentification déléguée par un IdP externe au serveur CAS ;
13. *Other* : défini et programmé pour d'autres besoins spécifiques.

La configuration de l'authentification multi-facteurs dans CAS offre également la possibilité de :

- **gérer des défaillances** (*Failure Mode*) des fournisseurs d'authentification MFA ;
- **évaluer quel fournisseur MFA est le plus adapté** si plusieurs fournisseurs sont applicables (*Ranking Providers*) ;
- **contourner l'authentification multi-facteur** (*Bypass rules*) en fonction de certains critères (plages IP, horaires, groupes…) ;
- **laisser la possibilité à l'utilisateur d'autoriser des périphériques ou des navigateurs de confiance** (*Trusted Devices / Browsers*) et ainsi de se passer du second facteur d'authentification dans certains contextes.

La liste des fournisseurs d'authentification MFA pouvant être supportés par CAS comme second facteur d'authentification est assez large : *Duo Security (mfa-duo), Authy Authenticator (mfa-authy), YubiKey*

5 « Chaque déclencheur va essayer d'ignorer la requête d'authentification et passer la requête au déclencheur suivant ».
6 Groovy est un langage de programmation orienté objet qui utilise une syntaxe très proche de Java et est directement compilé, soit à la volée dynamiquement, soit classiquement avec un compilateur en *bytecode*.

*(mfa-yubikey), RSA/RADIUS (mfa-radius), Google Authenticator (mfa-gauth),* des dispositifs physiques compatibles *FIDO U2F (mfa-u2f),* des *OTP (One-Time Password) « simples » (mfa-simple)*... De plus, il est possible de définir son propre fournisseur MFA ainsi que sa propre intégration du MFA dans CAS par différents moyens ; un retour d'expérience [14] a notamment été présenté aux JRES 2017 par l'Université Paris 1 Panthéon Sorbonne, portant sur leur solution Esup-OTP qui laisse la possibilité à l'utilisateur de gérer lui-même les seconds facteurs d'authentification. Esup-OTP peut-être intégrée au serveur CAS tel un fournisseur d'authentification MFA supplémentaire et est notée *mfa-esupotp*.

## 5.2 Configuration globale du MFA

Cet exemple de configuration du MFA est le plus simple à mettre en place dans le serveur CAS, car l'authentification renforcée y est activée avec un second facteur donné pour l'ensemble des authentifications gérées par CAS. Cette configuration est équivalente fonctionnellement à l'authentification renforcée inconditionnelle présentée dans la partie 4.1 pour l'IdP Shibboleth. Pour cela, il est nécessaire de renseigner dans le fichier de configuration principal de CAS (*cas.properties*) la ligne suivante qui précise quel fournisseur d'authentification MFA il est nécessaire d'utiliser globalement :

```
# Activate MFA globally for all, regardless of other settings
cas.authn.mfa.globalProviderId=mfa-simple

# Describe the global failure mode in case provider cannot be reached
# cas.authn.mfa.globalFailureMode=CLOSED

# Design the attribute chosen to communicate the authentication context
# cas.authn.mfa.authenticationContextAttribute=authnContextClass

# Identify the request content type for non-browser MFA requests
# cas.authn.mfa.contentType=application/cas
```

Les lignes de configuration commentées ci-dessus sont des valeurs par défaut qui décrivent globalement :

- le comportement du serveur CAS lorsque le fournisseur d'authentification MFA externe est défaillant : ici l'authentification est interrompue (*CLOSED*) ;
- l'attribut par défaut contenant le contexte de l'authentification dans la réponse de CAS à l'application (*authnContextClass*) ;
- le type de contenu de la « requête MFA » en dehors d'un navigateur.

La configuration du module « MFA simple » (*cas-server-support-simple-mfa* utilisable depuis CAS 6.x seulement) permet d'envoyer un OTP par mail ou par sms. Sa configuration est réalisée dans le fichier *cas.properties*. Dans l'exemple ci-dessous, les différents éléments configurés sont la durée de vie du jeton d'authentification transmis à l'utilisateur (120 secondes), le moyen utilisé pour transférer le jeton (un mail envoyé à l'utilisateur : *simple.mail.* dans la clé des propriétés de configuration), l'émetteur, le contenu du message envoyé et enfin l'attribut de personne dont la valeur est utilisée comme destinataire du mail :

```
# Simple Multifactor Authentication (>CAS 6.x)
cas.authn.mfa.simple.timeToKillInSeconds=120

# Email submissions
cas.authn.mfa.simple.mail.from=noreply@univ.fr
cas.authn.mfa.simple.mail.text=Le jeton \u00e0 renseigner dans la mire
d'authentification est : %s
cas.authn.mfa.simple.mail.subject=Votre Jeton d'authentification renforc\u00e9e
cas.authn.mfa.simple.mail.attributeName=email
```

## 5.3 Configuration adaptative du MFA

Le recours ou non à l'authentification renforcée dans le serveur CAS (ou **le rejet de certaines requêtes d'authentification**) en fonction de contextes géographiques, temporels ou physiques est gérable de différentes manières et adaptable aux besoins. La première possibilité consiste à activer un ou plusieurs éléments de configuration du module *Adaptive Authentification* [15] (correspondant au premier déclencheur d'authentification MFA dans le serveur CAS décrit dans la partie 5.1). L'exemple ci-dessous rend obligatoire l'authentification renforcée (avec *mfa-simple*) lors de chaque demande d'authentification réalisée le week-end ou en dehors des heures ouvrées de travail (ici entre 20h et 7h) :

```
cas.authn.adaptive.requireTimedMultifactor[0].providerId=mfa-simple
cas.authn.adaptive.requireTimedMultifactor[0].onOrAfterHour=20
cas.authn.adaptive.requireTimedMultifactor[0].onOrBeforeHour=7
cas.authn.adaptive.requireTimedMultifactor[0].onDays=Saturday,Sunday
```

Le serveur CAS peut aller plus loin et examiner précisément les requêtes d'authentification précédentes afin de détecter celles présentant un **comportement suspect** (c'est-à-dire qui divergent du comportement habituel de l'utilisateur sur une période donnée). Pour cela, le module *Risk-based Authentication* (qui étend le module *Adaptive Authentication*) supporte cette fonctionnalité mais est plus complexe à appréhender : il nécessite de configurer différentes dépendances, par exemple *cas-server-support-geolocation-maxmind* pour la géolocalisation avec les bases de *MaxMind*, *cas-server-support-events-memory* pour le stockage des évènements d'authentification en mémoire et *cas-server-support-electrofence* (pour le module *Risk-Based* lui-même). CAS mémorise sur une période donnée tous les évènements d'authentification afin de pouvoir les parcourir, les analyser et calculer pour chaque nouvelle authentification un score/degré de risque qui la qualifie et permet soit de la laisser passer (si le score est inférieur à un seuil prédéfini), soit de prendre des **mesures pour atténuer le risque** (si le score est supérieur à un seuil prédéfini) en rejetant la demande d'authentification ou en proposant une seconde étape d'authentification à l'utilisateur. Un ou plusieurs paramètres sont configurables pour le calcul du risque : l'adresses IP, les navigateurs utilisés (*user-agent*), la géolocalisation, la date et l'heure de connexion pendant une fenêtre de temps donnée. Le paramétrage du seuil est plutôt « obscur » dans ce contexte et nécessite des tests afin d'être ajusté aux besoins. Optionnellement, l'utilisateur final peut également être notifié par mail à chaque tentative d'authentification suspecte.

Un autre module, cas-*server-support-trusted-mfa*, peut permettre d'adapter l'utilisation ou non de l'authentification MFA par rapport aux périphériques utilisés par l'utilisateur : en effet, sur les périphériques de confiance enregistrés par un utilisateur, il n'y aura pas d'authentification MFA. Pour distinguer **les périphériques utilisés** les uns par rapport aux autres, une **empreinte numérique unique** (*fingerprint*) doit être calculée par le serveur CAS. Ce calcul peut reposer sur une combinaison de plusieurs éléments : l'adresse IP du client, un cookie généré aléatoirement à partir de l'adresse IP du client, la géolocalisation, le navigateur utilisé.

Il est important de noter que les modules liés à l'authentification adaptative du serveur CAS restent encore assez expérimentaux et nécessitent des tests avant d'être réellement utilisables en production ; il y est nécessaire, enfin, d'améliorer les interfaces graphiques et les messages affichés aux utilisateurs finaux.

## 5.4 Configuration du MFA pour une application donnée

Les configurations liées à un service donné sont très courantes dans le serveur CAS et sont facilitées par la déclaration sous forme de fichiers *json* dans le *service registry* de *CAS*. Chaque fichier de configuration est appliqué à un ensemble d'applications répondant une expression régulière (regex) sur l'url utilisée ou l'*entity-id*. L'exemple suivant permet d'activer un second facteur d'authentification avec l'usage d'un

dispositif physique compatible avec les spécifications de FIDO U2F (*Universal Second Factor*) [16][7]. L'authentification MFA est activée pour tous les services dont l'URL répond à l'expression régulière décrite dans *serviceId*. La politique liée à l'authentification MFA est exprimée dans *multifactorPolicy* :

```
{
  "@class" : "org.apereo.cas.services.RegexRegisteredService",
  "serviceId" : "^https://app.univ.fr/secured(\\z|/.*)",
  "name" : "Application secured by CAS and U2f",
  "id" : 100,
  "description" : "Application secured with username/password and U2f MFA protection",
  "attributeReleasePolicy" : {
    "@class" : "org.apereo.cas.services.ReturnAllAttributeReleasePolicy"
  },
  "multifactorPolicy" : {
    "@class" : "org.apereo.cas.services.DefaultRegisteredServiceMultifactorPolicy",
    "multifactorAuthenticationProviders" : [ "java.util.LinkedHashSet", [ "mfa-u2f" ] ],
    "failureMode" : "CLOSED"
  },
  "evaluationOrder" : 100
}
```

Ci-joint un exemple de réponse donnée par le serveur CAS à l'application suite à une authentification renforcée réussie pour un utilisateur **uiduser** ayant utilisé comme premier facteur d'authentification un mot de passe chiffré en BCRYPT et comme second facteur d'authentification une clé compatible U2F. La réponse est exprimée dans le protocole CAS 3.0 et certains éléments sont compatibles avec SAML1.1. Les différents éléments décrivant l'authentification sont exprimés sous forme d'attributs :

```
<cas:serviceResponse xmlns:cas='http://www.yale.edu/tp/cas'>
 <cas:authenticationSuccess>
    <cas:user>uiduser</cas:user>
    <cas:attributes>
     <cas:isFromNewLogin>true</cas:isFromNewLogin>
     <cas:bypassMultifactorAuthentication>false</cas:bypassMultifactorAuthentication>
     <cas:authenticationDate>
          2019-09-26T16:51:52.123127+02:00[Europe/Paris]
     </cas:authenticationDate>
     <cas:authnContextClass>mfa-u2f</cas:authnContextClass>
     <cas:successfulAuthenticationHandlers>
          BCRYPT
     </cas:successfulAuthenticationHandlers>
     <cas:successfulAuthenticationHandlers>
          U2FAuthenticationHandler
     </cas:successfulAuthenticationHandlers>
     <cas:credentialType>UsernamePasswordCredential</cas:credentialType>
     <cas:credentialType>U2FTokenCredential</cas:credentialType>
     <cas:samlAuthenticationStatementAuthMethod>
         urn:oasis:names:tc:SAML:1.0:am:password
     </cas:samlAuthenticationStatementAuthMethod>
     <cas:samlAuthenticationStatementAuthMethod>
         urn:oasis:names:tc:SAML:1.0:am:unspecified
     </cas:samlAuthenticationStatementAuthMethod>
     <cas:authenticationMethod>BCRYPT</cas:authenticationMethod>
     <cas:authenticationMethod>U2FAuthenticationHandler</cas:authenticationMethod>
     <cas:longTermAuthenticationRequestTokenUsed>
       False </cas:longTermAuthenticationRequestTokenUsed>
     <cas:email>uiduser@univ.fr</cas:email>
   </cas:attributes>
  </cas:authenticationSuccess>
</cas:serviceResponse>
```

7 Les périphériques U2F se présentent le plus souvent sous la forme de clés USB embarquant un élément cryptographique (la clé privée ne peut être rendue publique). Pour utiliser une clé U2F, il est nécessaire en premier lieu de l'enregistrer (pour initialiser une paire de clés spécifiques au serveur) et ensuite à chaque demande d'authentification de la présenter (un challenge est généré entre le serveur et la clé U2F en passant par le navigateur).

## 5.5 Configuration du MFA par délégation d'un IdP à un serveur CAS

Lorsque le serveur CAS est utilisé au sein d'un établissement pour gérer les authentifications des applications internes, un IdP Shibboleth externe, intégré dans une ou plusieurs fédérations d'identité, est souvent configuré dans ce contexte pour déléguer son authentification au serveur CAS. Le module ***Shib-CAS*** d'Unicon [17] permet notamment de coupler l'IdP et le serveur CAS de manière à ce que l'IdP soit configuré en tant que client CAS et qu'il puisse déléguer l'authentification à CAS. A l'issue de l'authentification, l'IdP Shibboleth récupère l'identifiant de la personne connectée.

La dernière version de Shib-CAS introduit le support du profil REFEDS MFA. Ainsi, une requête d'authentification renforcée reçue par l'IdP Shibboleth peut être traitée par Shib-CAS et notamment être convertie en une stratégie d'authentification compréhensible et supportée par CAS (de type *mfa-xxxx*). Cette stratégie est fournie par l'IdP au serveur CAS à l'aide de la valeur du paramètre http *authn_method* (correspondant au *trigger* MFA n°2 présenté dans la partie 5.1). Un flux d'authentification spécifique est configuré dans l'IdP Shibboleth pour utiliser Shib-CAS et propager la requête d'authentification MFA. Dans le fichier *general-authn.xml* de l'IdP Shibboleth, la configuration[8] est la suivante :

```
<bean id="authn/External" parent="shibboleth.AuthenticationFlow"
  p:passiveAuthenticationSupported="true"
  p:forcedAuthenticationSupported="true"
  p:nonBrowserSupported="false">
    <property name="supportedPrincipals">
      <list>
        <bean parent="shibboleth.SAML2AuthnContextClassRef"
            c:classRef="https://refeds.org/profile/mfa" />
        <bean parent="shibboleth.SAML2AuthnContextClassRef" c:classRef=
            "urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport" />
      </list>
    </property>
</bean>
```

Seul le fournisseur d'authentification MFA *Security Duo*[9] est configuré complètement à titre d'exemple dans Shib-CAS pour illustrer la délégation de l'authentification MFA entre l'IdP Shibboleth et le serveur CAS. Dans le fichier *idp.properties*, deux éléments de configuration sont nécessaires pour notifier CAS de l'authentification Duo à appliquer et pour que Shib-CAS interprète la réponse retournée par CAS (comme celle présentée dans la partie 5.4) en une réponse d'authentification compatible avec SAML 2.0 :

```
shibcas.casToShibTranslators =
net.unicon.idp.externalauth.CasDuoSecurityRefedsAuthnMethodTranslator
shibcas.parameterBuilders =
net.unicon.idp.authn.provider.extra.CasMultifactorRefedsToDuoSecurityAuthnMethodParamete
rBuilder
```

La classe *CasMultifactorRefedsToGoogleAuthenticatorAuthnMethodParameterBuilder* permet de construire le paramètre http transféré à CAS de cette manière *authn_method=mfa-duo* lorsque la requête d'authentification SAML2 exige le profil MFA de REFEDS dans l'IdP Shibboleth. La classe *CasDuoSecurityRefedsAuthnMethodTranslator* permet d'interpréter la réponse de CAS, d'y rechercher notamment que l'authentification MFA avec Duo a bien été réalisée et ensuite de construire une réponse similaire à celle décrite dans la partie 3.4 (en renseignant notamment l'élément *AuthnContext*) :

```
<cas:authnContextClass>mfa-duo</cas:authnContextClass>
```

8 La configuration reportée dans l'exemple est applicable à un IdP Shibboleth 3.4.6 avec le module Shib-CAS 3.3.0.
9 Le MFA avec *Duo* est configurable par l'ajoutant du module *cas-server-support-duo* et en suivant le guide [18].

Il est à noter que l'intégration du module de sécurité Pac4J, gérant toutes les délégations d'authentification dans CAS, offre la possibilité au serveur de CAS de supporter nativement le protocole SAML 2.0 depuis les versions 4.x et ainsi de jouer directement le rôle d'IdP [1]. Dans ce contexte, il est ainsi possible de déclencher une phase d'authentification MFA à l'aide des différents déclencheurs exposés précédemment. Néanmoins, le cas d'usage consistant à traiter des requêtes du MFA exprimées sous forme d'exigences par le SP (*i.e. authnContextClassRef*) n'est pas documenté dans CAS ; nous estimons donc que cette dernière possibilité n'est pas supportée à l'heure actuelle par le serveur CAS.

# 6 Conclusions et perspectives

En dehors du cas simple d'une application destinée uniquement à un public interne, nous avons vu que les cas d'usage de l'authentification forte à la fois légitimes, pertinents, et techniquement possibles au sein d'une fédération d'identité sont assez limités, à cause de la multiplicité des acteurs d'une part, et du peu d'interaction qui les lient. Si l'on y rajoute la complexité technique du sujet, ainsi que les coûts d'intégration et de maintien en condition, il est difficile d'envisager une généralisation dans un avenir proche.

Cette difficulté est néanmoins fortement liée à la nature décentralisée que nous associons à la notion de fédération d'identité, parce que c'est le modèle de notre fédération nationale, la Fédération Éducation-Recherche. Dans ce modèle, dit ***Full Mesh***, chaque participant assure la mise en œuvre de sa propre infrastructure, et en assume lui-même les coûts. Il y a donc peu de place pour la mutualisation et les économies d'échelles. Dans un modèle alternatif, dit ***Hub&Spoke***, en revanche, la centralisation d'une partie de l'infrastructure réduit largement les coûts techniques, et en conséquence les freins à la mise en œuvre. De même, lorsque l'opérateur de la fédération opère lui-même un service d'authentification renforcée, et offre ainsi à sa communauté un service similaire à celui habituellement proposé par un prestataire commercial (tel que Duo, InWebo ou Yubico pour ne citer que des noms connus), il va également pouvoir réduire les coûts et faciliter la mise en œuvre. Le cas de la fédération nationale des Pays-Bas, SURFnet, opérée par le NREN[10] du même nom [19], est intéressant, parce qu'il montre qu'il est possible de réunir ces deux conditions favorables, avec à la fois un **service d'authentification renforcée**, nommé *SURFsecureID* [20], et un **hub d'authentification centralisé**. Nous ignorons cependant si faciliter ainsi la mise en œuvre du dispositif a suffi pour généraliser son usage.

Afin de rendre l'authentification plus forte, nous avons parlé jusque-là d'un second facteur d'authentification proposé à l'utilisateur final après une phase principale d'authentification (le plus souvent par identifiant/mot de passe). Cette année, une nouvelle spécification, *Web Authentication* (abrégée *WebAuthn)* [21], rédigée par le W3C et FIDO, et intégrée dans le *framework* FIDO2, vise à faire évoluer les authentifications web, car elle supporte des méthodes plus sécurisées et amorce la disparition de facteurs faibles (tel l'identifiant/mot de passe) en les remplaçant par des « clés publiques cryptographiques ». FIDO2 reprend les principes et intègre les protocoles liés à *WebAuthn*, U2F (*Universal Second Factor*) [16], UAF (*Universal Authentication Framework,* permettant de gérer les authentifications sans mot de passe sur des périphériques mobiles) [16]. *WebAuthn* est une API qui décrit l'utilisation des paires de clés asymétriques (leur création et leur gestion) et normalise le fonctionnement de tout type d'*Authenticators* (générant et protégeant les clés privées, et gérant les challenges d'authentification). Les principaux navigateurs supportent *WebAuthn* et les fournisseurs d'authentification tels que Duo, Yubico… commencent à proposer des solutions compatibles avec ce nouveau protocole.

10 Un Réseau national de la recherche et de l'enseignement (parfois abrégé en NREN pour *National Research and Education Network*) est une organisation chargée de fournir et de gérer une infrastructure réseau pour les centres de recherche, les écoles et les universités. RENATER est le NREN en France.

# Bibliographie


[1] L. Auxepaules et A. Chabli, *IdP de Shibboleth vs CAS d'Apereo « Le meilleur des deux mondes »*, JRES 2019, Dijon, décembre 2019

[2] *Utilisation des certificats SAML dans la Fédération Éducation-Recherche : principe et recommandations*, Services RENATER, https://services.renater.fr/federation/documentation/generale/certificats-saml

[3] S. Cantor et al., *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0,* OASIS SSTC, Document ID saml-core-2.0-os, http://www.oasis-open.org/committees/security/, mars 2005

[4] J. Kemp et al., *Authentication Context for the OASIS Security Assertion Markup Language(SAML) V2.0*, OASIS Standard, Document ID saml-authn-context-2.0-os, https://docs.oasis-open.org/security/saml/v2.0/saml-authn-context-2.0-os.pdf, 15 mars 2005

[5] S. Cantor, *AuthnContextRef vs. AuthnContextDeclRef?*, Shibboleth - Users List, https://shibboleth.1660669.n2.nabble.com/AuthnContextRef-vs-AuthnContextDeclRef-td7630169.html, 8 décembre 2016

[6] M. Linden, *REFEDS Assurance Framework*, 13th FIM4R Workshop: Federated Identity Management for Research Collaborations, https://indico.cern.ch/event/775478/contributions/3276430/attachments/1792530/2923104/RAF_FIM4R_11Feb2019.pdf, Wien, Austria, 11 février 2019

[7] *REFEDS Multi-Factor Authentication Profile (MFA)*, https://refeds.org/profile/mfa, 7 juin 2017

[8] *REFEDS Single-Factor Authentication Profile (SFA)*, https://refeds.org/profile/sfa, 28 août 2018

[9] *IdP 3 Multi-Factor Authentification*, Shibboleth Wiki, https://wiki.shibboleth.net/confluence/display/IDP30/MultiFactorAuthnConfiguration

[10] *Duo Authentication configuration*, IdP Shibboleth, Shibboleth Wiki, https://wiki.shibboleth.net/confluence/display/IDP30/DuoAuthnConfiguration

[11] *IdP 3 Authentication Flow Selection*, Shibboleth Wiki, https://wiki.shibboleth.net/confluence/display/IDP30/AuthenticationFlowSelection

[12] *SP 3 Attribute Access*, Shibboleth Wiki, https://wiki.shibboleth.net/confluence/display/SP3/AttributeAccess

[13] *Multifactor Authentication Triggers*, Apereo CAS Enterprise Single Sign-On, https://apereo.github.io/cas/6.0.x/mfa/Configuring-Multifactor-Authentication-Triggers.html

[14] A. Anli, *Authentification multifacteur avec CAS et ESUP OTP*, JRES 2017, https://2017.jres.org/fr/presentation?id=185, Nantes, novembre 2017

[15] *Adaptive Authentication*, Apereo CAS Enterprise Single Sign-On, https://apereo.github.io/cas/6.0.x/mfa/Configuring-Adaptive-Authentication.html

[16] *Fido UAF and U2F specifications overview*, Fido Alliance, https://fidoalliance.org/specifications/

[17] *REFEDS MFA Profile with shib-CAS-authn3*, Apereo Blog, https://apereo.github.io/2018/02/26/shibcasauthn-duomfa-refeds/, février 2018

[18] *Duo Security Authentication*, Apereo CAS Enterprise Single Sign-On, https://apereo.github.io/cas/6.0.x/mfa/DuoSecurity-Authentication.html

[19] M. Oostdijk, et al., *On the design of a multi-factor-authentication-as-a-service service*, TNC 2013, https://tnc2013.terena.org/core/presentation/48, Maastricht, Netherlands, 3-6 juin 2013

[20] *SURFsecureID*, SURFNet NREN Wiki, https://wiki.surfnet.nl/display/SsID/SURFsecureID

[21] *Web Authentication: An API for accessing Public Key Credentials Level 1*, W3C Recommendation, https://www.w3.org/TR/webauthn-1/, 4 mars 2019