サービス連携における認可制御のための属性証明書を発行するサーバーレスアプリケーション

市原 創 [1,*]

概要：ゼロトラストアーキテクチャの浸透に伴い，ネットワークや組織の境界を越えてサービスを連携してアプリケーションを構成する事例が増加している．サービスの連携は主に API で接続されており，統合された認証基盤(IdP)が ID を管理し，標準化された認証・認可機構で制御される．代表的な認証・認可プロトコルである SAML, OpenID Connect は，認証・認可サーバーの発行するアクセストークンに認証・認可情報を乗せて連携するのが特徴だが，常時問い合わせ可能な認可サーバーが必要であり，障害に対する堅牢性，可用性，認可ポリシー管理の負担軽減など実用上の多様な要求に適さない場合がある．本稿ではサービスの外部連携の認可機構として，分散化した属性認証局が発行する属性証明書で認可制御を行うアプローチを提案する．属性認証局はサーバーレスアプリケーションとして運用することで，一定の信頼性と拡張性を維持し，属性証明書は発行と認証・認可のフェーズを非同期とすることで可用性の高い制御を可能とする．提案手法のシステムアーキテクチャと想定するユースケースを示し，その利点と今後の実証に向けた課題を検討する．

キーワード：サーバーレス，認可制御，属性証明書

Serverless Application to Issue Attribute Certificates for Authorization Control in Service Federation

Hajime Ichihara[1,*]

Abstract: With the spread of zero-trust architecture, there has been an increase in the number of cases where services are federated across network and organizational boundaries to form applications. The federation of services is mainly connected via APIs, with an integrated authentication infrastructure (IdP) managing identities and controlled by standardized authentication and authorization mechanisms. SAML and OpenID Connect, which are representative authentication and authorization protocols, are characterized by their ability to link authentication and authorization information with access tokens issued by authentication and authorization servers. However, they require an authorization server that can be queried at all times and may not be suitable for various practical requirements such as robustness against failures, availability and reduced burden of authorization policy management. This paper proposes an approach to authorization control using attribute certificates issued by a decentralized attribute certification authority as an authorization mechanism for external linkage of services. The attribute certification authority is operated as a serverless application to maintain a certain level of reliability and scalability, and the attribute certificate enables highly available control by making the issuance and authentication/authorization phases asynchronous. The advantages and challenges of the proposed system architecture and assumed use cases were discussed.

Keywords: Serverless, Authorization Control, Attribute Certificates

1. はじめに

クラウドコンピューティングやモバイルデバイス，IoT の普及，リモートによるワークスタイルの拡大により，ローカルネットワークとグローバルネットワーク，個人と組織の境界を越えたトラフィックが増加しており，従来のような内外ネットワークを分離して信頼・非信頼を判断する考え方ではセキュリティを確保することが困難になりつつある．このような背景から，ネットワーク境界を問わずにあらゆるユーザーやデバイスなどのエンティティ，アプリケーションやトラフィックデータを信頼せずに常に検証するゼロトラストアーキテクチャ(ZTA)のモデルが提唱されている[1] ．ZTA のネットワークではリソースへのアクセスを要求するエンティティを識別しアクセス制御を行う必要があるため，IDaaS のような ID 管理基盤(IdP)が重要な役割を担っている．

エンタープライズ分野では自組織の提供するサービスに他組織の構成員や個人，特定デバイスからの API アクセスを許可する事例や，他社サービスと API 連携してアプリケーションを構成する事例が増えている．このような外部エンティティとの連携の仕組みは従来からサービス指向アーキテクチャとして発展してきたが，近年はクラウドの普及と技術の標準化，実装が充実してきたことで幅広く導入され，自動化も進んでいることで連携対象となるエンティティは増加傾向にある．

1 キヤノン IT ソリューションズ株式会社，東京都港区港南 2-16-6 キヤノン S タワー，Canon IT Solutions Inc., 16-6, Konan 2-chome, Minato-ku, Tokyo 108-0075, Japan

* ichihara.hajime@canon-its.co.jp

サービス連携は HTTPS の SOAP や RESTful-API で構成するのが主流であり外部エンティティからのアクセス要求は API ゲートウェイが橋渡しをする．その際，IdP と連携する認証・認可サーバーがエンティティを認証し，ポリシーに従い認可を与える．認証には主に OpenID Connect や SAML プロトコルが使われている．エンティティがリソースへのアクセスを要求すると認証・認可サーバーはユーザーを認証した上でポリシーに従い, JWT や SAML アサーションなどの形式のアクセストークンを発行する．外部エンティティはアクセストークンを提示することで API ゲートウェイを通して自組織のサービスメッシュ内のリソースにアクセスすることが許可される．

このように JWT や SAML などの動的なアクセストークンを使った認証・認可は安全性，信頼性，サポート状況などから多くのサービス連携事例で普及している．これらはセッション単位のリソースアクセスに適しており，既存の認証・認可システムとの親和性も高いことから，ZTA における現実的な認証・認可手法と考えられる．

ただし，組織を越えたサービス連携では多様なシステムの特性や要求事項があり，前述のアクセストークン方式だけでは要求を満たせない場合がある．例えば IoT やサイバーフィジカルシステムのように自律的でリアルタイムな連携を必要とするケースでは次に挙げられるような課題の解決は優先度の高い要求事項となる可能性がある．

(1) 耐障害性，可用性，性能

アクセストークンによる認可シーケンスは，短期的なアクセストークンの発行を想定したプロトコルとなっており，認可サーバーが稼働して，外部エンティティが常時問い合わせできることが期待される．アクセス対象のリソースが分散している場合でも，認可サーバーはコントロールプレーンとして同期的に動作するためアクセスが集中しやすい．システム全体における単一障害点となるリスクもあり，トラフィックの多いシステムでは性能面でボトルネックとなる恐れもある．対策としてサーバーを冗長化する，エンティティに近接するエッジサーバーに配置する，信頼性の高い外部サービスに委託する等が考えられるが，いずれの場合も維持管理コストを負担しなければならない．

また，認可サーバーが稼働していても，認可対象のエンティティやリソース側の要因により一定期間認可サーバーに接続できない場合も考えられる．そのような状態でもエンティティ同士の認可制御を必要とするようなシステムもあり得る．

(2) 属性管理の負担増加

認可制御においてはアクセス対象であるリソース情報とアクセスを要求するエンティティの属性情報を事前に認可サーバーに登録し管理する必要がある．サービスとユーザー，ロールが多岐にわたり，紐付く権限の重複が増えるに伴い管理者の負担も著しく増加する．特にリソース自身の変化や属性情報の変化が頻繁である場合，管理の負担は爆発的に増加する恐れがある．

(3) ドメイン間の信頼関係

外部サービスとの連携では接続先との相互認証を前提とするため事前に相互の信頼関係を確立しておかなければならない．提供するサービスによっては短期的，中期的な信頼関係を結び一時的かつ限定的な認可を与えるケースも考えられるが，組織同士の固定的な信頼関係だけでなく，プロジェクト単位の契約に基づく関係，所属部門や所属構成員同士の関係，一時的な権限に基づく特定デバイスの個別アクセスなど，より細かい単位で信頼関係を結び必要最低限の認可制御をしたい場合が考えられる．しかし連携の単位が細かくなり関係が増加すれば，相互に信頼すべきエンティティが増加し管理の複雑化やガバナンスの問題を生じる懸念もある．

本稿では，サーバーレスアプリケーションとして実装した属性認証局が発行する属性証明書を利用して，サービス連携における認可制御を行う方式を提案する．提案方式は属性証明書の発行権限を分散し，アクセス権限は個々のサービス単体で検証できるため，属性管理の負担を軽減し，検証の耐障害性や可用性を高めることができる．

本論文の構成は次のとおりである．2 章で従来の認証・認可関連技術の特徴と課題について説明し，3 章では先行する研究について述べる．4 章で提案方式のシステムアーキテクチャの概要を説明し，5 章で適用され得るユースケースについて述べ，6 章では提案方式の利点と検討すべき課題を考察する．最後に 7 章で本論をまとめる．

2. 関連技術

2.1 SAML

SAML は OASIS で策定されたセキュリティに関する認証，属性，認可情報をやり取りするための XML ベースのマークアップ言語である[2] ．SAML プロトコルは要求と応答の手順を定義しており，SAML アサーションは主体に関する属性や権限等の情報の証明書であり，セキュリティトークンとして機能する．SAML アサーションは SAML オーソリティと呼ばれる権威機関が発行する．SAML を利用するには接続するユーザーやドメインのアイデンティティを管理する Identity Provider(IdP)と Service Provider(SP)がそれぞれの秘密鍵で署名したメタデータを交換し，信頼関係を結ぶ必要がある．SAML には認証・認可のための豊富な標準が定義されており，SAML アサーションによる動的で柔軟な認可制御ができることが特徴だが，連携が増えると

SAML オーソリティに負荷が集中し単一障害点となる懸念がある．

2.2 OpenID Connect と OAuth

OpenID Connect(OIDC)は，OpenID Foundation が策定した標準的な分散型認証・認可プロトコルである．認可制御の標準プロトコルである OAuth2.0 に認証機能が統合されており，個人 ID における Single Sign On(SSO)で広く利用されている．OIDC は JWT でセキュリティトークンのやりとりを行う．JWT は Claim と呼ばれる属性情報を JSON 形式で記述し，発行者の署名を付加したトークンである．

OAuth[3]は連携に先立ち認証情報を交換することで信頼関係の構築が可能だが，事前の信頼関係が無くともユーザー同意に基づき自動的に認証することも可能であり SSO における利便性が高い．リフレッシュトークンを使った非同期アクセスも可能だが，短期のアクセストークン更新には認可サーバーが必要であり，負荷集中や単一障害点となる懸念は残る．

2.3 属性認証局と属性証明書

属性証明書は，ITU-T の X.509 規格に追加され IETF の RFC 3281(後に RFC 5755[4]に改定)でも標準化された証明書のインターネット標準である．X.509 PKC が所有者(Holder)の静的な ID を証明しているのに対し，属性証明書は PKC に関連付けられた動的な属性情報や権限情報を証明する. PKC と同様の ASN.1 形式で記述できるが公開鍵は持たず，属性や認可情報に発行者の署名が付加される．既存の PKI トラストチェーンとも親和性が高く，信頼性が高いことが特徴である．証明書運用や権威機関のモデルが示されてはいるが, 運用や検証プロセスが比較的複雑であり，セッション単位の動的な認証・認可が主流となった現在では，ユースケースが限定的であり広範には普及していない[5]．

3. 関連研究

3.1 属性情報による認証・認可制御

アクセス制御に属性情報を利用する方式は従来から研究されてきた．畠山ら[6]は ZTN におけるアクセス制御の連携で，アイデンティティ情報以外の特定の属性を含むコンテキスト情報を連携する手法を提案している．コンテキスト情報は IdP とは別に CAP(Context Attribute Provider)と呼ばれるエンティティが収集・共有し，ユーザーの同意を得てシステムで共有される．

属性情報連携の遅延の問題については，属性を静的属性と動的属性に分離し，動的属性を遅延の少ないサイトにローミングすることで，遅延時間を短縮する研究がある[7]．しかし，これらの提案手法はいずれも遅延以外の障害時やオフライン環境での運用を想定する技術として考案されたものではない．

3.2 属性証明書による認証・認可制御

所有者の厳密な本人性を証明する公開鍵証明書と異なり，属性証明書は属性情報のみを証明するため，特定の属性に関連した権限を制御するために使うことができる．

柿崎ら[8]は属性証明書を使ってプライバシー保護された権限行使を PKI 上に実現する方法を提案している．その後の研究[9]ではユースケースを想定した認可サービスの実装を行い, 属性証明書による認可制御の有効性を示した．また，属性情報の分散管理方式[10][11]も提案し，集中管理の安全性・可用性の問題について論じている．

Ruland ら[12]は IEC 61850 をモデルとする分散制御システムで属性証明書を使った Attribute-based access control(ABAC)を提案している．提案システムはサブジェクトとオブジェクトの属性を表現する属性証明書の保存と配布に LDAP サーバーを使用することで，属性証明書を効率的に管理する手法を採用しているが，証明書を発行する管理システムの可用性については研究対象としていない．

4. システムアーキテクチャの提案

4.1 属性証明書による認証・認可の分散・非同期化

提案方式は X.509 公開鍵証明書(PKC)の事前配布により信頼を形成し，属性やアクセス権限が記述された属性証明書を発行・提示することで外部エンティティのアクセス要求への認可を制御する．属性証明書に記述された属性やアクセス権限はサービス毎に分散・非同期的に検証可能であり，特定のサーバーへの問い合わせを必要としないため，負荷集中や単一障害点となる懸念を軽減することができる．IdP は外部エンティティの ID の管理を行うが，より細かい属性ベースのアクセス制御(ABAC)やアクセス権限は属性証明書の発行者(Issuer)の管理する属性認証局に委譲し，分権的に属性証明書を発行・管理する．

4.2 サーバーレスの活用

アプリケーション基盤としてクラウドが活用されるようになったことで，最適コストで拡張性の高い保守・運用を可能とするクラウドネイティブコンピューティングに注目が集まっている．特に要求発生時に計算資源を動的に割り当てて実行するサーバーレスアプリケーションの手法を用いることで，アプリケーションの拡張・変更など柔軟で自動的な運用を可能にし，コストを最適化することができるようになった．提案方式は属性認証局をサーバーレスアプリケーションとして実装することで，その運用上の利点を最大限に活用する．

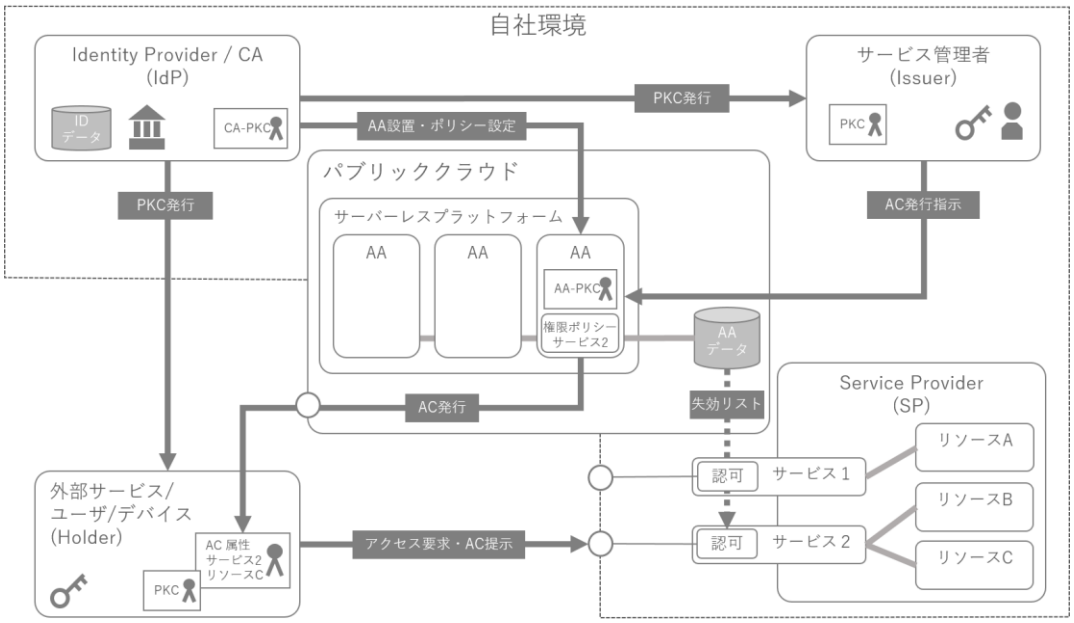

図 1　システムアーキテクチャ

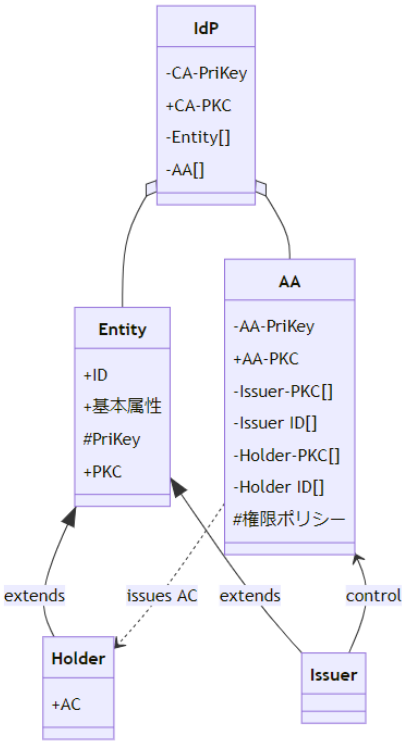

図 2　構成要素の論理的な保持情報

4.3 アーキテクチャの概要

例として自社サービスへのアクセスを外部エンティティ（サービス/ユーザー/デバイス）に許可するシステムを考える．図 1 に提案するシステムの構成，図 2 に構成要素の論理的な保持情報をクラス図として示す．

図 1 括弧内に示す各構成要素の役割は次のように定義した．外部エンティティやサービス管理者の ID を管理する Identity Provider と X.509 PKC を発行する認証局(CA)を IdP とする．提供するサービスは Service Provider で動作しており，これを SP とする．サービス管理者は SP のサービスを管理する立場のユーザーであり，サービスへのアクセス権限管理のために属性証明書(AC)を発行する権限を有するため，Issuer とする．サービスに接続しアクセスを要求する外部エンティティは発行された AC を保有しサービス利用時に提示するため，Holder とする．Issuer と Holder は認証用の鍵ペアを保有しているものとする．

サービス連携の認証・認可において属性情報を柔軟に活用するため，接続するエンティティの認証に必要な基本属性（所属組織，部署，役割，名前など）と，外部からのアクセス制御の認可に必要な個別属性（ローカルな役割，参加プロジェクト，権限など）を分離する．エンティティの基本属性は IdP が ID と共に管理し，Issuer と Holder の ID も管理する．各エンティティは IdP の CA が発行した PKC を保有し認証に使用する．個別属性は IdP から権限委譲された属性認証局(AA)が AC として発行する．AA は Issuer ID と権限ポリシーを持ち Issuer の指示のみにより AC を発行することができる．AA はクラウドのサーバーレスアプリケーションとして構築し，ステートレスな API を実行できる．

AA は Issuer と Holder の関係をデータベースで管理しており，AA の論理的単位(AA-PKC の数)の基準はサービス，リソース，部署，個人など柔軟に定義して良いが，AA の持つ権限ポリシーとの整合性を維持する必要がある．

4.4 Issuer と AA の権限

IdP は AA に AC 発行の権限を委譲する．AA に委譲した権限を行使するのは IdP 管理下で許可された Issuer のみである．AA をコントロールできる Issuer と発行可能な属性

の権限スコープは IdP の管理するポリシーによって限定される．IdP は Issuer が制御可能なリソースの範囲と所有権を管理しており，AA を設置する権限を持つ．AA は当該リソースの権限範囲に関する権限ポリシーと操作用の API を持つ．権限ポリシーは AA に発行された AA-PKC の AAControls エクステンションに記述しても良い．AA が制御可能なリソースは URI 形式で記述され AA の権限スコープを示す．

4.5 AA へのアクセス

図 3 に AA の設置と AC の発行要求，AC 取得までの処理の流れを示す．IdP は AA 設置時に AA の署名用の鍵ペア生成を指示し，管理下にある CA から AA の公開鍵証明書（AA-PKC）を発行する．また AA はコントロール権を持つ Issuer の PKC を保有する．Issuer が AA の API を呼び出す際は，Issuer-PKC と AA-PKC を用いて相互認証を行う．また，Holder は AC の発行を受ける際に AA にアクセスする必要があるため，AA は Holder-PKC と AC 取得用の API を持つ．Holder が AA の API を呼び出すときは，Holder-PKC と AA-PKC を用いて相互認証を行う．

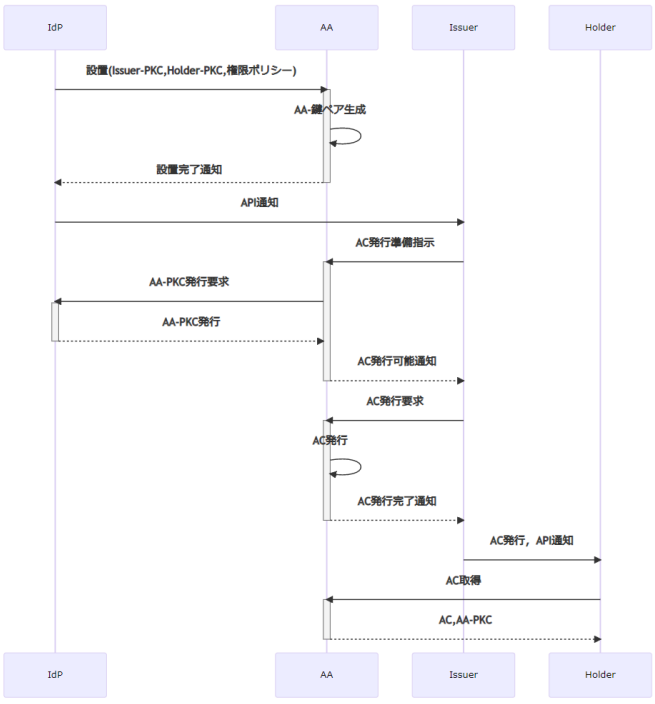

図 3　AA 設置，AC 発行フェーズ

4.6 属性証明書による ABAC

図 4 に Holder が AC を提示して SP のリソースにアクセスする処理の流れを示す．Holder が AC を SP に提示する手順は push 型と pull 型があるが，提案システムは push 型を前提とする．Holder はサービスの公開 API にアクセスを要求する際に自身の PKC と AC，AC 発行を受けた AA の AA-PKC を自身の署名を付加して提示する．SP は AA-PKC の妥当性，Holder-PKC の妥当性，AC の署名と Holder の署名を検証する．SP は CA-PKC を信頼する PKC として保有し検証に使用する．また，CA の発行した各 PKC の失効情報も定期的に取得し，保有しているものとする．

提案方式では AC を使ったアクセス制御の単位を特定しないが，AC の非同期性を活かすには検証処理と認可処理はサービス単位で実施することが望ましい．各サービスは定期的に AA から AC 失効情報を取得し，アクセス要求時には提示された AC が有効であるかどうかを判断する．AC の検証に問題がなければ，Holder の要求するリソースへのアクセス認可を，AC に記述された属性や権限情報に基づき判断する．

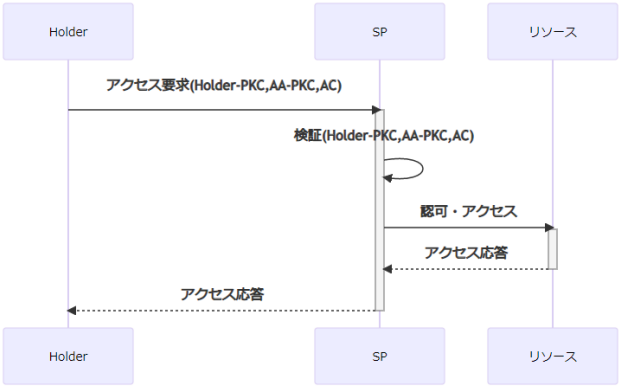

図 4　AC 認証・認可フェーズ

4.7 ライフサイクル管理

AA の設置・廃止

AA の初期設置・廃止は必ず IdP を通じて行わなければならないが，Issuer の設置要求，廃止要求をトリガとしても良い．AA の廃止は AA-PKC の失効を意味するため，IdP の CA は AA-PKC の失効処理を行い，失効情報に登録する．クラウド上の AA の秘密鍵をはじめとする AA データは即座に破棄する．

AA-PKC 失効・更新

AA-PKC が失効することは，管理下にある AC が全て失効することであるため，AA 廃止と同等の扱いとなる．したがって，AA-PKC の有効期限切れにより更新が必要な場合も一旦 AA を廃止し，新しく AA を設置し，AC を再発行する必要がある．

AC 失効

Issuer の指示により AC を失効する場合は AA で管理するデータベースに AC の失効リスト(ACRL)の情報を記録する．サービスは運用要件に従い定期的にデータベースにアクセスし，ACRL を取得することで AC の検証に利用する．ただし，1 日など十分に短い有効期間の AC を扱う場合は失効リストによる失効管理を省略する設計も考えられる．OCSP によるリアルタイムな失効確認を行っても良いが非

同期性や負荷分散のメリットを損なう可能性がある．

Issuer / Holder の PKC 失効・更新

Issuer と AA との関係は IdP によって変更可能とし，AA を操作できる権限を持つ Issuer の権限を無効化し，別のエンティティに付与することも可能とする．Issuer の PKC が失効する場合, IdP は AA のデータベースにおける Issuer の PKC を新しい PKC または別のエンティティの PKC に書き換えることで，AA の運用を継続することができる．

Holder の PKC を失効する場合は，IdP により AA データベースの Holder-PKC を削除し，対象 AC の失効処理を行う．Holder-PKC を再発行する場合は，対象 AC の失効処理を行い AA データベースの Holder-PKC を更新し，AC を再発行する．

CA-PKC 失効・更新

CA-PKC の失効は IdP が信頼の基点として機能しなくなるため，速やかに IdP の認証やサービスの停止を行う．有効期限切れによる更新の場合は，移行期間を設けるなどして新しい CA-PKC の事前発行を行い, 移行期間中に各 PKC, AC の更新処理を進めていくなどの運用が考えられる．

5. ユースケース

5.1 業務委託契約におけるアクセス権限の共有

ユースケースの例として業務委託契約において，委託元が機密保持契約を結ぶ委託先に対して自社内で管理する情報リソースやコラボレーションツールなどのアプリケーションへの短期的，限定的なアクセスに認可を与えるケースを考える．業務委託契約においては，会社同士の比較的長期の信頼関係の元で，構成員が中期的なプロジェクトに参画し，一時的な作業を行うケースがある．アプリケーションやリソースの管理者はプロジェクトにおける組織的な権限に基づき，Issuer として AC を発行し，委託先事業者との柔軟な連携と認可制御を実現できる．

委託先は外部エンティティとして委託元サービスに接続するため，予め認証のための ID の発行を委託元に依頼する．委託元は IdP を通して委託先 ID と PKC の発行，AA 設置を行い，委託元サービスのアクセスに限定した AC 発行権限を AA とサービス管理者である Issuer に委譲する．Issuer は契約に基づき，アクセスを許可するリソースを特定し，AA にアクセスして AC を発行する．

5.2 IoT / CPS デバイスの一時的な認可制御

近年は IoT に加えて，物理空間のデバイス・センサ・制御システムとサイバー空間を高度に組み合わせた自律的な系であるサイバーフィジカルシステム(CPS)に注目が集まり研究や実用化が進んでいる．CPS の構成要素として産業機器や医療機器，車載システムなど多種多様なデバイスが挙げられるが，これらはスマートフォンのように常時インターネットに接続可能であるとは限らない上に安全性の面などから高い可用性を求められる場合がある．オフラインのデバイス同士が近距離無線通信で接続し一時的にアプリケーションへのアクセス認可を与えたい場合，予め発行された PKC，AC であれば，特定のサーバーへの接続が不可能な状態でも認証・認可制御が可能である．

6. 考察

6.1 サーバーレスアーキテクチャの利点

属性証明書のライフサイクルオペレーションの頻度は，接続するエンティティと有効期間に依存しており，変動する要素が大きい．AA をサーバーレスアプリケーションとして実装することで運用コストの最適化を図ることができる．発行証明書が少なくオペレーションの頻度が少ない場合は，AA の計算資源は解放し，逆にエンティティと発行証明書が増え，オペレーション頻度が急増した場合は，直ちに必要な計算資源を確保することができる．

サーバーレスアプリケーションは動的に計算資源を確保するため一般的に応答性の低さが指摘される．これは認証・認可サーバーのようにサービスへのアクセスとの同期性を求められるアクセス制御には問題になるが，AA のオペレーションはサービスへのアクセスとは同期しないため, サービスの応答性能には影響を及ぼさない．

また，コード変更やパッチ適用，障害対応など運用に関する手間を最小化する効果も期待できる．

6.2 AC によるアクセス制御

AC は PKC と比較して短期の属性を扱うことに向いており，あるプロジェクトにおいて短期間付与される役割のような属性情報も表現することができる．また，IdP の認証・認可サーバーに依存せずにサービスへのアクセス制御が可能なことは負荷分散や堅牢性，業務内容に応じた柔軟な権限制御などの面で優位な特性となる. AC は既存の X.509 証明書を解析できる環境があれば検証処理が容易に実装できるため，既存のルートストアやソフトウェア資産との親和性が高いことも利点となる．

各サービス単位で検証・認可を実行するためには検証・認可処理を実装できるライブラリが継続的に利用できなければならない．提案の AA 操作は API ベースであり，より簡便な実装でアクセス制御が実現できる外部エンティティ用・内部サーバー用の双方のライブラリの整備が必要である．また提示された検証済 AC の URI ベースの権限に基づきアクセス認可を判断するリクエスト受け付けモジュールも必要となる．

6.3 AA の永続データへのアクセス制御

提案手法では Issuer-PKC，Holder-PKC による認証で AA へのアクセスを制御するが，エンティティの数が増えると管理すべき PKC と関連情報, 更新頻度が著しく増加することが予想される．AA の永続データはクラウドストレージに保存されるため，近年研究が進んでいる属性ベース暗号 (ABE) を適用することで，各エンティティの属する属性に基づいたアクセス制御が実現できる可能性がある．ABE は秘密鍵に関連付けられる属性に依存した暗号化を可能とし，失効のようなライフサイクルにも対応する研究が進んでいる [13][14][15][16] ．

6.4 AC の適正な有効期間について

AC によるアクセス制御は鍵や証明書の不正な利用に注意が必要である. 外部エンティティの PKC については秘密鍵とデバイスとの関係を紐づける，発行する AC の有効期間を短めに設定するなどの対策が考えられる．一般的に AC を利用するメリットは AC の有効期間が PKC の有効期間よりも十分短い場合に発揮される．

6.5 AA の論理的単位に関する考察

AA(AA-PKC)の論理的な設置単位については自明ではなく，いくつかのバリエーションの中で要求に応じた設計を選択可能である．まず，Issuer の単位より小さいレベル (Holder 単位など)で設置することは IdP の管理負担が増大するため，最小設置単位をサービスの管理権限を持つ Issuer 以上の単位とすれば目的に合致する．ただし，実際のサービス管理において単一の Issuer に権限を依存してしまうのはリスクがあるため，複数の Issuer により共同管理することがより現実的である．提案手法では Issuer と AA の関係はデータベース上で管理するため，共同管理による権限委譲を表現することは比較的容易である．サービス単位を越えた共通の AA を設置する考え方もある．この場合は AC 発行権限を特定の Issuer に集約することになるため，単位が大きくなると失効・更新などの際に発生する管理作業が増大し，提案手法の管理負担を軽減する効果が失われる可能性が高い.

6.6 ライフサイクルにおける課題

AC の有効期間が長く，同一 AA の発行する AC の数が多くなると，AC 失効時に展開する失効リストの肥大化を招く恐れがある．そのため AC の有効期間は必要最小限であることが望ましい．ただし，極端に短い有効期間を必要とするケースはセッション単位で発行するアクセストークンと実質的に変わらず, 本方式の優位性は失われる. AC の数は AA の論理的単位にも依存するため，ユースケースの要件において提案手法が適しているかどうかは検討を必要とする.

発行した AC が有効期間内に失効した場合は，失効リストなどを通じてサービスに通知しなければならない．サービスは AC の有効期間を意識した間隔で事前に失効リストの問い合わせを行える運用が望ましい．また，緊急性の高い失効の場合は強制的に失効リストの更新を行う仕組みも検討が必要である.

AC が有効期間内に失効した場合，Holder 側のアクセス要求は拒否される．Holder が再度アクセスを要求するには AC の再発行を Issuer に依頼しなければならない．サービス側事情による強制失効時にユーザビリティの低下を回避するには，Holder のアクセス失敗をトリガとして AC 発行依頼を自動的に処理できる仕組みを整備するなどの対策が考えられる.

6.7 発行の透明性と監査

WebPKI では Certificate Transparency(CT)により CA の PKC 発行ログを記録し，監視・監査の透明性を確保する仕組みが普及している．提案したアーキテクチャでは AA を分権的に運用するため，AC 発行履歴に関する透明性を確保し，不正な AC 発行がないことを検証できることが望ましい．AA は AC を発行する際に AC の発行記録をログサーバーに送信し，AC に関するシリアル No，発行者，発行日時, 有効期限等を改ざん困難なハッシュ木ログに記録し，IdP が常時監査可能な状態で運用するなどの方式が考えられる.

6.8 AA アプリケーションの信頼性と保守運用

サーバーレスアプリケーションの AA で発行される証明書の信頼性はデプロイされたアプリケーションコードの信頼性にも依拠している．したがってデプロイに対する信頼性はリモートアテステーションにより担保することが望ましい．サーバーレスアーキテクチャにおけるリモートアテステーションについては Alder ら[17]の先行研究がある. また，IETF[18]では標準化が検討されている.

実装の保守・運用については正しいプロセスを遵守する開発組織が必要となる．AA のコアとなるコードはシンプル・少量にし，オープンソースにすることで広く検証されることが望ましい．また，コードに重大な欠陥や脆弱性，攻撃が存在しないかを監視する第三者が必要である.

6.9 鍵管理

AA の鍵管理は提案手法の信頼性を左右するため，クラウドの HSM を利用するなど物理的セキュリティを考慮した鍵管理も検討すべきである．AC を発行する Holder 数が多い場合, AA で管理する PKC と AC が膨らむ恐れがある．現実的には，Holder を個人やデバイス単位で発行するのではなく, 特定組織の単位で発行し, PKCS#12 の形式で共有・展開するなどして共通化する運用も考えられる.

7. おわりに

本稿では，組織を越えたサービス連携における属性情報管理の課題に着目し，拡張性を備えたサーバーレスアプリケーションとして実装した属性認証局が属性証明書を発行し，認可制御を行うシステムアーキテクチャを提案した．提案システムは属性証明書により非同期的に権限を検証できるため，比較的長期的な信頼関係の元でオフライン環境やサービス単位で認可制御を行う必要があるユースケースに有用であることを示した．また，実装や運用においていくつかの検討課題が残されていることを考察した．今後は本システムアーキテクチャの有効性と課題，性能を評価するための実装と実証実験を行う．

参考文献

[1] NIST. “Zero Trust Architecture”. SP800-207, 2020-08-11, https://csrc.nist.gov/publications/detail/sp/800-207/final

[2] OASIS. “Security Assertion Markup Language (SAML) V2.0 Technical Overview”. 2008-03-25, http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0.html

[3] Internet Engineering Task Force (IETF). “The OAuth 2.0 Authorization Framework: Bearer Token Usage”. 2012, https://datatracker.ietf.org/doc/html/rfc6750

[4] Internet Engineering Task Force (IETF). “An Internet Attribute Certificate Profile for Authorization”. 2010, https://datatracker.ietf.org/doc/html/rfc5755

[5] 電子商取引推進協議会，日本情報処理開発協会 電子商取引推進センター. “属性認証ハンドブック”. 平成１６年度ＥＣ技術基盤の相互運用性に関する調査研究. 2005 年 2 月, https://www.jipdec.or.jp/archives/publications/J0004226.pdf

[6] 畠山昂大，小谷大祐，岡部寿男. “ゼロトラスト認証認可連携におけるユーザ同意付きコンテキスト共有”. マルチメディア，分散協調とモバイルシンポジウム 2114 論文集 2020 (2020-06-17): 640‐47.

[7] 下道高志，佐々木良一. “クラウドコンピューティング環境での認証連携における動的属性利用技術の提案と評価”. 情報処理学会論文誌 55, no. 3 (2014-03-15): 1186‐1200.

[8] 柿崎淑郎，辻秀一. “属性証明書を用いた認証方式の提案”. 情報処理学会コンピュータセキュリティ研究会報告, 2004-12-20. 2004-CSEC-27(2)

[9] 今川俊明，柿崎淑郎，辻秀一. “属性証明書を用いた認可サービスシステムの構築”. 情報処理学会第 69 回全国大会, 2007-03-06, 5W-1

[10] 柿崎淑郎，辻秀一. “一意にアクセス可能な属性情報の分散管理方式”. 情報処理学会研究報告情報システムと社会環境（IS） 2008, no. 81(2008-IS-105) (2008-08-21): 61‐68.

[11] 柿崎淑郎，吉田慶章，辻秀一. “一意なアクセスと属性間関係性の検証可能な属性情報分散管理方式”. 情報処理学会論文誌 51, no. 2 (2010-02-15): 604‐12.

[12] C. Ruland, J. Sassmannshausen, J. P. Satpathy. “An Attribute Certificate Management System for Attribute-Based Access Control”. 2018 International Conference on Computational Science and Computational Intelligence (CSCI), 2018-12-01. https://doi.org/10.1109/csci46756.2018.00015.

[13] A. Sahai, B. Waters, “Fuzzy identity-based encryption,” in Proc. Eurocrypt, 2005, pp. 457–473.

[14] V. Goyal, O. Pandey, A. Sahai, B. Waters, “Attribute-based encryption for fine-grained access control of encrypted data,” in Proc. ACM Conf. Comput. Commun. Security, 2006, pp. 89–98.

[15] J. Bethencourt, A. Sahai, B. Waters, “Ciphertext-policy attributebased encryption”, in Proc. IEEE Symp. Security

[16] K. Nomura, M. Mohri, Y. Shiraishi, M. Morii, “Attribute Revocable Multi-Authority Attribute-Based Encryption with Forward Secrecy for Cloud Storage” in Proc. IEICE TRANS.INF.&SYST., VOL.E100-D,N.10 2017.

[17] F. Alder, N. Asokan, A. Kurnikov, A. Paverd, M. Steiner. “S-FaaS: Trustworthy and Accountable Function-as-a-Service using Intel SGX”. CCSW@CCS, 2019-11-11, 185–99.

[18] Internet Engineering Task Force(IETF). “Remote ATtestation ProcedureS (rats)”. https://datatracker.ietf.org/group/rats/documents/