(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2019-46060
(P2019-46060A)
(43)公開日　平成31年3月22日(2019.3.22)

(51)Int.Cl.		F I			テーマコード（参考）
G06F 21/41	(2013.01)	G06F	21/41		5B084
G06F 13/00	(2006.01)	G06F	13/00	510A	
G06F 21/33	(2013.01)	G06F	21/33	350	
G06F 21/62	(2013.01)	G06F	21/62	318	

審査請求 未請求　請求項の数 11　ＯＬ　（全 18 頁）

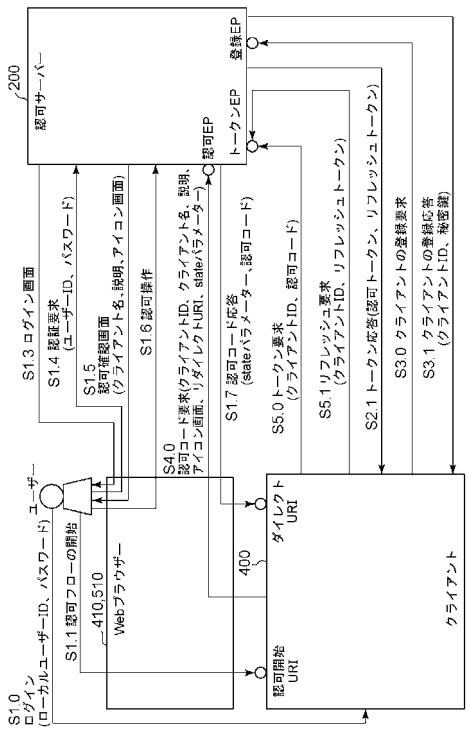
(21)出願番号　特願2017-167285 (P2017-167285)
(22)出願日　平成29年8月31日 (2017.8.31)

(71)出願人　000001007
キヤノン株式会社
東京都大田区下丸子３丁目３０番２号
(74)代理人　100126240
弁理士　阿部　琢磨
(74)代理人　100124442
弁理士　黒岩　創吾
(72)発明者　松ヶ下　勇人
東京都大田区下丸子３丁目３０番２号キヤノン株式会社内
(72)発明者　山中嶋　和成
東京都大田区下丸子３丁目３０番２号キヤノン株式会社内

Ｆターム(参考)　5B084 AA01 AA11 AA30 AB36 BB01 BB12 DB02 DC01 DC02 DC03

(54)【発明の名称】権限委譲システム、制御方法、およびプログラム

(57)【要約】
【課題】　ＯＡｕｔｈ　２．０の処理を実行する上での安全性を損なわずに、クライアントのＵＲＬが変更された場合の処理の煩雑さを解消することを目的とする。
【解決手段】　クライアントによるリソースサーバーへのアクセスを認可サーバーが認可するための認可コード要求を、前記クライアントが認可サーバーに送信する認可コード要求送信手段と、前記認可コード要求に対する応答である認可コード応答を受信する認可コード応答受信手段と、を有する権限委譲システムであって、前記認可コード要求送信手段によって送信される認可コード要求は、前記認可コード応答の宛先を指定するための宛先情報を含み、前記宛先情報は署名情報が付与されており、前記認可サーバーは前記認可コード要求送信手段によって送信された認可コード要求に含まれる宛先情報に基づいて、前記認可コード要求に対する認可コード応答を送信する。
【選択図】　図６



【特許請求の範囲】
【請求項１】
クライアントによるリソースサーバーへのアクセスを認可サーバーが認可するための認可コード要求を、前記クライアントがユーザーエージェントを介して認可サーバーに送信する認可コード要求送信手段と、
前記認可コード要求に対する応答である認可コード応答を受信する認可コード応答受信手段と、
を有する権限委譲システムであって、
前記認可コード要求送信手段によって送信される認可コード要求は、前記認可サーバーが前記認可コード応答を応答する応答先を指定するための応答先情報と前記応答先情報に付与された署名情報とを含み、
前記認可サーバーは前記認可コード要求送信手段によって送信された認可コード要求に含まれる応答先情報に基づいて、前記認可コード要求に対して認可コード応答を送信する権限委譲システム。
【請求項２】
前記署名情報は、
前記クライアントが保持する秘密鍵を用いて前記応答先情報に付与され、
前記認可サーバーは、
前記認可サーバーが保有する公開鍵を用いて、前記応答先情報に付与されている署名情報を検証する請求項１に記載の権限委譲システム。
【請求項３】
前記クライアントが前記認可サーバーに対して前記クライアントに関する情報を登録するためのクライアント登録要求を送信した後に、
前記認可サーバーは前記クライアントを識別するためのクライアント識別情報と、前記クライアントを認証するための前記公開鍵と前記秘密鍵とを生成し、
前記認可サーバーは前記クライアント識別情報と前記公開鍵とを関連付けて管理し、
前記クライアントは、
前記クライアント登録要求に対する応答として前記クライアント識別情報と前記秘密鍵とを受信する請求項２に記載の権限委譲システム。
【請求項４】
前記認可コード要求を受信した認可サーバーは、
前記認可コード要求とともに受信した前記クライアント識別情報に基づいて、前記署名情報を検証するための公開鍵を特定する請求項３に記載の権限委譲システム。
【請求項５】
前記ユーザーエージェントは、
前記クライアントによる前記リソースサーバーへのアクセスを認可することをユーザーが同意するための認可確認画面を提供する認可確認画面提供手段を有し、
前記認可コード要求は、
前記認可確認画面に提示する前記クライアントに関する情報と、
前記クライアントに関する情報を提示する際の言語を指定する言語情報と、
を含む請求項１乃至請求項４に記載の権限委譲システム。
【請求項６】
前記言語情報は、
前記クライアントに予め設定されている言語情報または前記ユーザーエージェントからのリクエストに含まれる言語情報のいずれか一つである請求項５に記載の権限委譲システム。
【請求項７】
前記応答先情報は、
前記認可コード応答を前記クライアントが前記認可サーバーから受信するために前記認可コード応答の応答先を指定するリダイレクトＵＲＩである請求項１乃至請求項６のいず

れか一つに記載の権限委譲システム。
【請求項８】
前記リダイレクトＵＲＩは、
前記アクセスを前記認可サーバーが認可するための認可処理を開始するためのリクエストが含むユーザーエージェントのＨｏｓｔヘッダーに基づいて決定され、
前記Ｈｏｓｔヘッダーに基づいて決定できない場合は、
前記クライアントが新たにＩＰアドレスを取得し、取得されたＩＰアドレスに基づいて前記リダイレクトＵＲＩが決定される請求項７に記載の権限委譲システム。
【請求項９】
前記クライアントに関する情報は、
クライアント名乃至前記クライアントに関する説明を含む請求項４乃至請求項８に記載の権限委譲システム。
【請求項１０】
権限委譲システムで実行される、
クライアントによるリソースサーバーへのアクセスを認可サーバーが認可するための認可コード要求を、前記クライアントが認可サーバーに送信する認可コード要求送信ステップと、
前記認可コード要求に対する応答である認可コード応答を受信する認可コード応答受信ステップと、
を有する制御方法であって、
前記認可コード要求送信ステップによって送信される認可コード要求は、前記認可サーバーが前記認可コード応答を応答する応答先を指定するための応答先情報と前記応答先情報に付与された署名情報とを含み、
前記認可サーバーは前記認可コード要求送信ステップによって送信された認可コード要求に含まれる応答先情報に基づいて、前記認可コード要求に対する認可コード応答を送信する権限委譲システムで実行される制御方法。
【請求項１１】
権限委譲システムで実行される、
クライアントによるリソースサーバーへのアクセスを認可サーバーが認可するための認可コード要求を、前記クライアントが認可サーバーに送信する認可コード要求送信ステップと、
前記認可コード要求に対する応答である認可コード応答を受信する認可コード応答受信ステップと、
を有する制御方法が実現されるためのプログラムであって、
前記認可コード要求送信ステップによって送信される認可コード要求は、前記認可サーバーが前記認可コード応答を応答する応答先を指定するための応答先情報と前記応答先情報に付与された署名情報とを含み、
前記認可サーバーは前記認可コード要求送信ステップによって送信された認可コード要求に含まれる応答先情報に基づいて、前記認可コード要求に対する認可コード応答を送信する権限委譲システムで実行される制御方法が実現されるためのプログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、Ｗｅｂサービスへのアクセス権限を検証する権限委譲システムに関する。
【背景技術】
【０００２】
個々のＷｅｂサーバーはＷｅｂサービスを提供するためにＡＰＩを公開しており、公開されているＡＰＩを介してＷｅｂサービス同士が連携する形態がある。その際セキュリティの観点から、Ｗｅｂサービスが管理するユーザーの認証情報を受け渡すことなく、他のＷｅｂサービスによってアクセスされることを認可する手段が必要となる。

【０００３】
その手段として、Ｗｅｂサービス同士の連携を実現させるための標準プロトコル（ＯＡｕｔｈ　２．０）の採用が進んでいる。ＯＡｕｔｈ　２．０は、ユーザーの認証情報をそのユーザーの同意に基づいてＷｅｂサービス間で安全に受け渡す（すなわち委譲する）ための仕組みであり、詳細は後述する。
【０００４】
ＯＡｕｔｈ　２．０によれば、ユーザーが認可操作を行うとＷｅｂサービスＢは認可コードを受信する。認可コードとは、ＷｅｂサービスＡへのアクセスがユーザーによって認められたことを証明するためのコードである。受信した認可コードと、ＷｅｂサービスＢであることを証明するための情報を用いて、ＷｅｂサービスＢはＷｅｂサービスＡに対して認可トークンの発行要求を送信する。認可トークンとは、ＷｅｂサービスＡが公開するＡＰＩにＷｅｂサービスＢがアクセスすることを認可するためのトークンである。ＷｅｂサービスＢが認可トークンを受信することで、ＷｅｂサービスＢがＷｅｂサービスＡのＡＰＩにアクセスすることが認可される。ＷｅｂサービスＢである事を認証するための情報は、ＷｅｂサービスＢを一意に識別するためのＩＤおよび、秘匿情報であるシークレット、デジタル証明書によるデジタル署名が挙げられる。
【０００５】
ユーザーの認可操作によりＷｅｂサービスＢがＷｅｂサービスＡのＡＰＩにアクセスすることを認可することを権限委譲と称する。なお、ＯＡｕｔｈ　２．０では、ユーザーの認可操作により認可コードを発行し、認可コードから認可トークンを発行するサーバーを認可サーバーと称する。また、ＡＰＩを公開するサーバーをリソースサーバー、公開されたＡＰＩにアクセスする主体をクライアントと称する。上記の例でいえば、ＷｅｂサービスＡを提供するサーバーが認可サーバー兼リソースサーバーであり、ＷｅｂサービスＢを提供するサーバーがクライアントである。
【０００６】
図１を用いてＯＡｕｔｈ　２．０の処理フローであるＡｕｔｈｏｒｉｚａｔｉｏｎ　Ｃｏｄｅ　Ｇｒａｎｔを説明する。まず、ＯＡｕｔｈ　２．０を実行するための事前操作として、認可サーバーにクライアントをＯＡｕｔｈ　２．０のクライアントとして登録するための登録要求を行う（Ｓ０．０）。具体的には、クライアントの登録要求は認可サーバーの登録エンドポイント（図中ではエンドポイントを「ＥＰ」と記載）に対して送信され、クライアントの起動時か、後述のＳ１．１の認可フローの開始時にクライアントが未登録であった場合に開始される。登録要求の方法としては、クライアントが能動的に認可サーバーと通信する方法や、ユーザーがＷｅｂブラウザーを介して認可サーバーにアクセスしてクライアントを登録する方法が挙げられる。
【０００７】
Ｓ０．０の登録要求には、後述の認可確認画面に表示するためのクライアント名、説明、アイコン画像、および必須パラメーターであるリダイレクトＵＲＩが含まれる。リダイレクトＵＲＩとは、認可サーバーからの認可コード応答をクライアントが受信するために、認可サーバーが認可コード応答を送信する応答先を指定する応答先情報（アドレス）である。認可コード応答については後述する。クライアントの登録要求を受信した認可サーバーはクライアントを識別するクライアントＩＤと、クライアントを認証する秘匿情報であるクライアントシークレットとを発行し、クライアントの登録応答としてクライアントに返却する（Ｓ０．１）。認可サーバーは、Ｓ０．１でクライアントＩＤとクライアントシークレットと、Ｓ０．０で受信した各種情報およびリダイレクトＵＲＩとを紐づけて保持する。クライアントは、Ｓ０．１で受信したクライアントＩＤとクライアントシークレットを保持する。以上がＯＡｕｔｈ　２．０を実行する事前操作であるクライアントの登録フローである。
【０００８】
次に、認可サーバーにおいてユーザーを認証するためのフローを、図１を用いて説明する。ユーザーはクライアントにログインする（Ｓ１．０）。クライアントは、ログインし

たユーザーを特定するための情報であるログインコンテキストを生成し保持する。生成されたログインコンテキストからログインしたユーザーを特定する情報（ローカルユーザーＩＤ等）を取得する事ができる。ユーザーは、Ｗｅｂブラウザーを介して認可開始のＵＲＩにアクセスすることで、ＯＡｕｔｈ　２．０の認可フローを開始する（Ｓ１．１）。クライアントは認可フローを開始するための認可開始ＵＲＩにアクセスされると、認可サーバーの認可エンドポイントに対して認可コード要求を送信する（Ｓ１．２）。認可コード要求には、クライアントＩＤ、リダイレクトＵＲＩ、ｓｔａｔｅパラメーターを含む。

【０００９】

ｓｔａｔｅパラメーターは認可コード要求と認可コード応答とを一意に紐づけるための情報であり、ＣＳＲＦ（ｃｒｏｓｓ－ｓｉｔｅ　ｒｅｑｕｅｓｔ　ｆｏｒｇｅｒｙ）攻撃や、トークン置換（以下、認可コード置換）攻撃を防ぐために用いられる。そのため、ｓｔａｔｅパラメーターは推測不可能でかつ重複しない値である必要がある。なお、後述の認可コード応答でクライアントが受信するｓｔａｔｅパラメーターとＳ１．２の認可コード要求で送信したｓｔａｔｅパラメーターとの一致を検証する。さらに認可コード要求を実行したユーザーを識別するために、クライアントで発行されるｓｔａｔｅパラメーターはリダイレクトＵＲＩとログインコンテキストと紐付いてクライアントで管理される。

【００１０】

Ｓ１．２で認可コード要求を受信した認可サーバーは、ユーザーが認可サーバーにログインしていなかった場合、Ｗｅｂブラウザーにユーザーを認証するためのログイン画面を応答する（Ｓ１．３）。ユーザーは、Ｗｅｂブラウザーを介してユーザーＩＤ、パスワードを入力し、認可サーバーに対して認証要求を行う（Ｓ１．４）。認証要求を受信した認可サーバーは、Ｓ１．４で受信したユーザーＩＤとパスワードとの組み合わせが事前に登録されている組み合わせと一致するかを検証し、一致する場合は認証トークンを発行する。発行された認証トークンはＷｅｂブラウザーのＣｏｏｋｉｅに応答される。

【００１１】

認可サーバーはユーザーに対して、クライアントの認可を同意するための認可確認画面をＷｅｂブラウザーに応答する（Ｓ１．５）。ただし、Ｓ１．２で受信したクライアントＩＤとリダイレクトＵＲＩとの組み合わせが、認可サーバーに予め登録されたクライアントＩＤとリダイレクトＵＲＩとの組み合わせと一致しない場合は、Ｗｅｂブラウザーにエラー画面を応答する。これにより、不正なＵＲＩへのリダイレクト（転送）を防ぐことができる。また、ログインしているユーザーが既に同一のクライアントＩＤで認可操作済みであった場合はＳ１．５を省略する事もできる。認可済みのユーザーＩＤとクライアントＩＤの組み合わせを以下、同意情報と称する。

【００１２】

Ｓ１．６のユーザーによる認可操作の後、認可サーバーは認可コードを発行し、クライアントに対して認可コード応答として認可コードとｓｔａｔｅパラメーターを送信する（Ｓ１．７）。具体的には、認可コードとｓｔａｔｅパラメーターとをリダイレクトＵＲＩにクエリパラメーターとして付与し、リダイレクトＵＲＩで指定された応答先に認可コードとｓｔａｔｅパラメーターとをリダイレクトするようにＷｅｂブラウザーに送信する。Ｓ１．７で発行された認可コードは、認可サーバーにおいてクライアントＩＤ、ユーザーＩＤ、リダイレクトＵＲＬと紐づけて保存される。さらに認可サーバーは同意情報を保存する。

【００１３】

リダイレクトＵＲＩに対して認可コード応答を受信したクライアントは、認可コード応答に含まれるｓｔａｔｅパラメーターと、クライアントが管理するｓｔａｔｅパラメーターとが一致するかを検証する。検証の結果、ｓｔａｔｅパラメーターが一致した場合、クライアントは認可サーバーのトークンエンドポイントに対してトークン要求を送信する（Ｓ２．０）。トークン要求にはクライアントＩＤ、クライアントシークレット、Ｓ１．７で取得した認可コード、および、Ｓ１．２で受信したリダイレクトＵＲＩが含まれる。

【００１４】

Ｓ２．０でトークン要求を受信した認可サーバーは、クライアントＩＤとクライアントシークレットの組み合わせが予め登録された組み合わせと一致するかを検証する。検証の結果、一致が確認されるとクライアントが認証される。さらに認可サーバーは、Ｓ２．０で受信した認可コードを保持しているか、保持している場合にはその認可コードは有効期限内か、認可トークンと紐づいたクライアントＩＤとリダイレクトＵＲＩがＳ２．０のトークン要求で受信したものと一致するか、を検証する。この検証により、Ｓ１．２の認可コード要求を送信したクライアントとＳ２．０のトークン要求を送信したクライアントとが一致するかを認可サーバーで検証する事ができる。

【００１５】

検証が成功した場合、認可サーバーはクライアントに対して認可トークンを発行し、トークン応答としてクライアントに応答する（Ｓ２．１）。その際、認可トークンを再取得するためのリフレッシュトークンもクライアントに対して発行し、トークン応答として応答する事もできる。クライアントは、Ｓ２．１で受信した認可トークンを用いてリソースサーバーが公開するＡＰＩにアクセスすることが可能となる。また、認可トークンを発行した後に、認可サーバーで管理する認可コードを破棄することでリプレイ攻撃を防ぐことが可能となる。

【００１６】

Ｓ２．１のトークン応答にリフレッシュトークンが含まれる場合、クライアントにおいてログインコンテキストとリフレッシュトークンとが紐付けて管理される。それにより、次回以降のＡＰＩへのアクセス時に認可操作（Ｓ１．２～Ｓ１．７）を実施せずに認可トークンを再度取得する事ができる。具体的には、Ｓ１．１の認可開始を受けた際に、クライアントにおいてユーザーのログインコンテキストとリフレッシュトークンとが紐付いているかを確認する。紐づいていない場合は前述のＯＡｕｔｈ　２．０のフロー（Ｓ１．２以降の処理）を実施する。紐づいている場合は、認可サーバーのトークンエンドポイントに対してリフレッシュ要求を行う（Ｓ２．２）。このリフレッシュ要求はクライアントＩＤ、クライアントシークレット、およびリフレッシュトークンを含む。

【００１７】

リフレッシュ要求を受信した認可サーバーは、クライアントＩＤ、クライアントシークレットの組み合わせがＳ０．１で事前登録されたものと一致するかを検証する。一致が確認されクライアントが認証されると、受信したリフレッシュトークンが認可サーバーで保持されているか、保持されている場合はそのリフレッシュトークンは有効期限内か、さらにリフレッシュトークンに紐づいたクライアントＩＤがリフレッシュ要求のものと一致するかを検証する。これらの検証が全て成功した場合、認可サーバーは認可トークンを発行し、クライアントにトークン応答として認可トークンを送信する。その際、認可トークンを再取得するために新たなリフレッシュトークンを再発行し、トークン応答と同時にクライアントに対して送信する形態も可能である。また、認可サーバーにおいて新たなリフレッシュトークンを発行した後に、認可サーバーでそれまで管理されていたリフレッシュトークンを破棄する事でリプレイ攻撃を防ぐことができる。以上がＯＡｕｔｈ　２．０におけるＡｕｔｈｏｒｉｚａｔｉｏｎ　Ｃｏｄｅ　Ｇｒａｎｔの処理フローである。ＯＡｕｔｈ　２．０による処理フローにより、認可サーバーが管理するユーザーの認証情報をクライアントに送信する代わりに、認可サーバーが認可トークンを発行し、クライアントは発行された認可トークンを用いてリソースサーバーが公開するＡＰＩにアクセスすることができる。特許文献１では、ＯＡｕｔｈ　２．０による処理フローを用いて、複数の外部サービスシステムと連携する情報処理システムが開示されている。

【先行技術文献】

【特許文献】

【００１８】

【特許文献１】特開２０１６－６６２４

【発明の概要】

【発明が解決しようとする課題】

【００１９】
しかし図１で示した処理フローでは、クライアントのＵＲＩが変更された場合には認可サーバーに登録されているリダイレクトＵＲＩも変更しなければならず、クライアントのＵＲＩが変更される度に、認可サーバーに対してリダイレクトＵＲＩの変更依頼を行う必要があり、手間がかかる。クライアントのＵＲＩが変更される例としては、クライアントがプリンターやＭＦＰ等のデバイスであった場合、設置場所の移動によりＵＲＩのホストが変更されることがある。また、その他の例としては、デバイスのＩＰアドレスがＤＨＣＰなどのプロトコルにより自動で決定される場合はデバイスの電源のＯＮ／ＯＦＦによってＩＰアドレスが変わり、その結果、ＵＲＩのホストが変わることがある。本願発明は、ＯＡｕｔｈ　２．０の処理を実行する上での安全性を損なわずに、クライアントのＵＲＩが変更された場合の処理の煩雑さを解消することを目的とする。
【課題を解決するための手段】
【００２０】
クライアントによるリソースサーバーへのアクセスがユーザーによって許可されたことに応じて認可サーバーが認可コードを発行するための認可コード要求を、前記クライアントが前記認可サーバーに送信する認可コード要求送信手段と、
前記認可コード要求に対する応答である認可コード応答を受信する認可コード応答受信手段と、
を有する権限委譲システムであって、
前記認可コード要求送信手段によって送信される認可コード要求は、前記認可サーバーが前記認可コード応答を応答する応答先を指定するための応答先情報と署名情報とを含み、
前記署名情報が検証された後に、前記認可サーバーは前記認可コード要求送信手段によって送信された認可コード要求に含まれる応答先情報に基づいて、前記認可コード要求に対する認可コード応答を送信する。
【発明の効果】
【００２１】
本願発明により、ＯＡｕｔｈ　２．０の処理を実行する上での安全性を損なわずに、クライアントのＵＲＩが変更された場合の処理の煩雑さを解消することができる。
【図面の簡単な説明】
【００２２】
【図１】ＯＡｕｔｈ　２．０のＡｕｔｈｏｒｉｚａｔｉｏｎ　Ｃｏｄｅ　Ｇｒａｎｔの処理フロー。
【図２】本実施形態に係る権限委譲システムの構成図。
【図３】権限委譲システムを構成する各種デバイスのハードウェア構成図。
【図４】権限委譲システムを構成する各種デバイスのソフトウェアモジュール構成図。
【図５】Ｗｅｂブラウザーが表示するユーザー認証画面と、クライアント４００の認可確認画面の一例。
【図６】本実施形態に係るＯＡｕｔｈ　２．０のＡｕｔｈｏｒｉｚａｔｉｏｎ　Ｃｏｄｅ　Ｇｒａｎｔの処理フロー。
【図７】本実施形態に係る認可コード要求が含むＪＷＴの一例。
【図８】本実施形態に係る認可トークンが含むＪＷＴの一例。
【図９】本実施形態に係る、クライアント４００においてリダイレクトＵＲＩを決定する処理フロー。
【発明を実施するための形態】
【００２３】
以下、本発明を実施するための最良の形態について図面を用いて説明する。
【００２４】
まず図２を用いて、本発明の実施形態に係る権限委譲システムについて説明する。ＷＡＮ（Ｗｉｄｅ　Ａｒｅａ　Ｎｅｔｗｏｒｋ）１００はＷＷＷ（Ｗｏｒｌｄ　Ｗｉｄｅ　Ｗ

ｅｂ）システムによって構築されている。ＷＡＮ１００と各種デバイス２００～５００はＬＡＮ（Ｌｏｃａｌ　Ａｒｅａ　Ｎｅｔｗｏｒｋ）１０１を介して接続されている。

【００２５】

認可サーバー２００はＯＡｕｔｈ　２．０を実現するためのサーバーであり、認証要求の受信や認可コードの発行、管理等の処理を行う。リソースサーバー３００はＷｅｂサービスを提供するためのＡＰＩを公開している。図２では、認可サーバー２００とリソースサーバー３００はＬＡＮ１０１で接続されている形態を示しているが、ＷＡＮ１００を介して接続される構成も可能である。また、認可サーバー２００はＬＡＮ１０１を介して不図示のデータベースサーバーと接続されており、認可サーバー２００が自身の機能を実現する際に用いるデータをそのデータベースサーバーに格納するように構成してもよい。図２では認可サーバー２００とリソースサーバー３００を別のサーバーであるものとして説明しているが、同一のサーバー上に両サーバーの機能が構成されている形態でもよい。

【００２６】

クライアント４００はＯＡｕｔｈ２．０におけるクライアントに相当し、例えばプリンターやＭＦＰ、もしくはＰＣやスマートフォン等が挙げられる。端末５００はＯＡｕｔｈ２．０におけるユーザーエージェントに相当し、ユーザーは端末５００を介して、認可サーバー２００に対するユーザーの認証要求やクライアント４００に対するログイン操作等、各種デバイスの機能を利用することができる。端末５００として具体的にはＰＣやスマートフォン等が挙げられる。

【００２７】

クライアント４００および端末５００はそれぞれＷｅｂブラウザー４１０とＷｅｂブラウザー５１０を備える。ユーザーはＷｅｂブラウザー４１０またはＷｅｂブラウザー５１０を操作して後述の認可操作を実行する。クライアント４００と端末５００はＬＡＮ１０１を介して接続されている。以降、Ｗｅｂブラウザー４１０とＷｅｂブラウザー５１０のどちらで実行するかを問わない場合は、符番を振らずに「Ｗｅｂブラウザー」と称する。

【００２８】

次に図３を用いて、認可サーバー２００、リソースサーバー３００およびクライアント４００、端末５００のハードウェア構成を説明する。なお、図３は一般的な情報処理装置のブロック図であり、本実施形態の各種デバイスには一般的な情報処理装置のハードウェア構成やＩａａＳ（Ｉｎｆｒａｓｔｒｕｃｔｕｒｅ　ａｓ　ａ　Ｓｅｒｖｉｃｅ）として提供される情報処理装置の仮想的なハードウェア構成を適用できる。図３では、クライアント４００を例に説明するが、リソースサーバー３００や認可サーバー２００、端末５００も同様のハードウェア構成を有するものとする。

【００２９】

ＣＰＵ２００１は、ＲＡＭ２００２、ＲＯＭ２００３、外部メモリ２０１１などからプログラムを取り出してプログラムの命令を実行し、クライアント４００の制御を行うユニットである。後述のシーケンスはこのプログラムの命令が実行されることにより実現される。また、ＣＰＵ２００１はシステムバス２００４に接続される各ブロックを制御する。

【００３０】

ＲＡＭ２００２は、ＣＰＵ２００１が命令を実行する際に使用するワークメモリである。ＲＯＭ２００３や外部メモリ２０１１に保存されたＯＳやアプリケーション等のプログラムがＲＡＭ２００２へとロードされ、そのプログラムの命令をＣＰＵ２００１が順次読みだすことで命令を実行する。ＲＯＭ２００３は、アプリケーションプログラムおよびＯＳを含む組込済みプログラム、およびデータ等が記録されている記憶装置である。

【００３１】

ＫＢＣ（キーボードコントローラ）２００５は、ＫＢ（キーボード）２００９や不図示のポインティングデバイスからの入力を制御するユニットである。ＣＲＴＣ（Ｃａｔｈｏｄｅ　Ｒａｙ　Ｔｕｂｅ　Ｃｏｎｔｒｏｌｌｅｒ）２００６は、ＣＲＴディスプレイ２０１０の表示を制御するユニットである。ＤＫＣ（Ｄｉｓｋ　Ｃｏｎｔｒｏｌｌｅｒ）２００７は外部メモリ２０１１に対するデータアクセスを制御するユニットである。ＮＣ（ネ

ットワークコントローラ）２００８はＷＡＮ１００やＬＡＮ１０１を介して接続された他のデバイスとの通信制御処理を実行する。なお、ＩａａＳとして提供される仮想的な情報処理装置の場合は、ＫＢＣ２００５やＣＲＴＣ２００６等を備えず、ＮＣ２００８を介して接続される端末が備えるキーボードやＣＲＴディスプレイから操作されるよう構成される。

【００３２】

尚、後述の説明においては、特に断りのない限り、各種デバイスの機能が実行される際のハードウェアの主体はＣＰＵ２００１であり、ソフトウェアの主体はＲＡＭ２００２、ＲＯＭ２００３、外部メモリ２０１１等にインストールされたプログラムであるものとする。

【００３３】

次に図４を用いて、認可サーバー２００、リソースサーバー３００、クライアント４００、端末５００が有する機能について説明する。認可サーバー２００は認可サーバー部２１０、ＨＴＴＰサーバー部２２０を有する。ＨＴＴＰサーバー部２２０はＷＡＮ１００を介してクライアント４００と端末５００に接続されており、Ｗｅｂブラウザーや後述のクライアントアプリケーション４２０とＨＴＴＰ通信を行う機能である。また、ＨＴＴＰサーバー部２２０はＳＳＬ／ＴＬＳによる通信が可能であり、不図示の証明書ストアを有する。

【００３４】

認可サーバー部２１０はＨＴＴＰサーバー部２２０を介してＷｅｂブラウザー５１０からの要求を受信し、受信した要求に対する結果を応答する機能である。具体的には、ユーザー認証の要求をＨＴＴＰサーバー部２２０がＷｅｂブラウザー５１０から受信し、認証が成功したユーザーのユーザー情報が紐づいた認証トークンを生成し、Ｗｅｂブラウザー５１０に認証トークンを通知する。認証トークンとは、ユーザーが認可サーバー２００にログインしている事を示すためのトークン、またはユーザーが認可サーバー２００において認証されているかを検証するためのトークンである。認証トークンを用いることで認可サーバー２００はユーザーを識別することが可能となる。一方の認可コードは、認証されたユーザーの認可操作により権限委譲されたクライアント４００がユーザーに成り代わってリソースサーバー３００のＡＰＩにアクセスすることを許可された事を示すトークンである。また、認可サーバー部２１０は、認可トークンに署名情報を付与するための秘密鍵を保持するように構成する事もできる。その場合は、この秘密鍵を用いて認可トークンに署名情報を付与し、署名情報付きの認可トークンをクライアント４００に対して発行する。

【００３５】

リソースサーバー３００はリソースサーバー部３１０を有する。リソースサーバー部３１０は、Ｗｅｂサービスを提供するためのＡＰＩを公開する機能である。なお、認可サーバー２００と同様に、ＨＴＴＰサーバー部を備え、ＨＴＴＰサーバー部を介して外部との受送信を実行する形態でもよい。

【００３６】

クライアント４００はＷｅｂブラウザー４１０とクライアントアプリケーション４２０と認証部４３０を有する。Ｗｅｂブラウザー４１０はＷＷＷを利用するためのユーザーエージェントによって実現される機能であり、端末５００が備えるＷｅｂブラウザー５１０も同様の機能である。Ｗｅｂブラウザー４１０はユーザーの操作により、認可サーバー２００およびクライアントアプリケーション４２０と通信を行う。クライアントアプリケーション４２０は、リソースサーバー３００が公開するＡＰＩを実行することで自身が提供する機能と合わせたＷｅｂサービスをユーザーに提供する。本実施例では、クライアントアプリケーション４２０がＯＡｕｔｈ　２．０におけるクライアントに相当する。

【００３７】

認証部４３０はユーザーを認証するための機能である。ユーザーはクライアント４００の機能を利用するために、クライアント４００における不図示の入力画面においてローカ

ルユーザーＩＤとローカルユーザーパスワードを入力する。入力を受けたクライアント４００は、認証部４３０において予め登録されている情報（ローカルユーザーＩＤとローカルユーザーパスワード）と入力された情報とを照合することでユーザーの認証処理を行い、ログインコンテキストを生成する。なお認証処理の形態はこれに留まらず、例えば、ＩＣカードを使った認証や指紋等の生体認証でもよい。

【００３８】

ログインコンテキストは、クライアント４００でローカルユーザーを識別するための情報であり、例えば、ローカルユーザーＩＤから構成される。このログインコンテキストはクライアントアプリケーション４２０と認証部４３０で共有される。また、本実施例ではクライアント４００へのログイン処理についてユーザーが直接クライアント４００を操作してログインする形態で説明するが、Ｗｅｂブラウザー５１０を介してリモートでログインする形態でもよい。その場合、認証部４３０は不図示のログイン画面をＷｅｂブラウザー５１０に応答する。ユーザーはそのログイン画面にローカルユーザーＩＤとローカルユーザーパスワードを入力することでユーザーは認証される。その際、認証部４３０においてログインコンテキストが生成され、クライアントアプリケーション４２０と認証部４３０で共有される。

【実施例１】

【００３９】

本実施例では、ＯＡｕｔｈ　２．０の処理を実行する上での安全性を損なわずに、クライアントのＵＲＩが変更された場合の処理の煩雑さを解消する処理について説明する。図１で説明した処理については同じ符番を振り、詳細な説明は省略する。

【００４０】

まず、図５を用いて、認可サーバー２００がユーザーを認証するためのログイン画面と、ユーザーに対してクライアント４００の認可の同意を問うための認可確認画面について説明する。

【００４１】

図５（ａ）は、ユーザーが認可サーバー２００にログインするためのログイン画面の一例であり、Ｗｅｂブラウザーに表示される。ユーザーがＷｅｂブラウザーを介して認可サーバー２００の認可エンドポイントに認可コード要求を送信し、ユーザーが認可サーバー２００にログインしていない場合にＷｅｂブラウザーに表示される。ログイン画面５００は、ユーザーＩＤ入力欄５００１、パスワード入力欄５００２、ログイン操作を実行するためのログインボタン５００３を備える。ログインボタン５００３が押下された後の処理については後述する。

【００４２】

図５（ｂ）は、ユーザーが認証された結果、認可サーバー２００がＷｅｂブラウザーに応答する認可確認画面の一例である。認可確認画面５１００は、ユーザーに対して同意を求める内容として、認可先のクライアント４００のクライアント名５１０１、クライアント４００に関する説明５１０２、および、アイコン画像５１０３を有する。さらに認可確認画面５１００は、ユーザーがクライアント４００を認可するための許可ボタン５１０４、認可を拒否するための拒否ボタン５１０５を備える。許可ボタン５１０４、拒否ボタン５１０５が押下された後の処理については後述する。

【００４３】

次に、図６を用いて本願発明の特徴を備えたＯＡｕｔｈ　２．０のＡｕｔｈｏｒｉｚａｔｉｏｎ　Ｃｏｄｅ　Ｇｒａｎｔの処理フローを説明する。なお、図１と処理が同じものについては同じ符号を付与しており、詳細な説明は省略する。また、図１のＳ０．０を後述のＳ３．０、Ｓ０．１を後述のＳ３．１、Ｓ１．２を後述のＳ４．０、Ｓ２．０を後述のＳ５．０、Ｓ２．２を後述のＳ５．１に置き換えることで本実施例におけるＯＡｕｔｈ　２．０の処理フローを実行することができる。

【００４４】

まず、ＯＡｕｔｈ　２．０を実行する事前操作としてクライアント４００の登録フロー

について図６を用いて説明する。本実施例ではクライアント４００が能動的に認可サーバー２００と通信してクライアント４００の登録要求を実行する形態で説明するが、ユーザーがＷｅｂブラウザーを介して認可サーバー２００にアクセスし、クライアント４００の登録要求を実行する形態でもよい。クライアント４００の登録フローはクライアント４００が起動したときか、もしくはＳ１．１の認可フローの開始時にクライアント４００が未登録であった場合に開始するものとする。

【００４５】

クライアント４００は認可サーバー２００にクライアント４００の登録要求を送信する（Ｓ３．０）。登録要求を受信した認可サーバー２００はクライアント４００を識別するためのクライアントＩＤと、クライアント４００を認証するための暗号鍵と復号鍵（公開鍵と秘密鍵）の鍵ペアを生成する。以降の実施例では、秘密鍵と暗号鍵を一例に説明する。認可サーバー２００は、生成されたクライアントＩＤと秘密鍵とを登録応答としてクライアント４００に返却する（Ｓ３．１）。クライアント４００ではクライアントＩＤと秘密鍵とが紐付いて保存され、認可サーバー２００ではクライアントＩＤと公開鍵とが紐づいて保存される。本実施例ではクライアントＩＤが「ｃｌｉｅｎｔ＿０１」であるものとして説明する。クライアント４００が有する紐付け情報の一例を表１に、認可サーバー２００が有する紐付け情報の一例を表２に示す。

【００４６】

【表１】

表１

クライアントD	秘密鍵
ｃｌｉｅｎｔ＿０１	秘密鍵A

【００４７】

【表２】

表２

クライアントD	公開鍵
ｃｌｉｅｎｔ＿０１	公開鍵A

【００４８】

登録応答時にクライアント４００に送信される情報は、上記の形態に留まらず、秘密鍵の主体情報としてクライアントＩＤを埋め込み、登録応答は秘密鍵のみをクライアント４００に送信する形態も可能である。または、認可サーバー２００で予め秘密鍵を生成し、クライアント４００の生産時にその秘密鍵を予めインストールしておくことでクライアント４００の登録フロー（Ｓ３．０～Ｓ３．１）を実行しない形態も可能である。以上がクライアント４００の登録フローである。

【００４９】

従来は、クライアント４００の事前登録の際（Ｓ０．０）にリダイレクトＵＲＩとクライアント４００に関する情報とを認可サーバー２００に送信し、送信された情報を認可サーバー２００で管理した。それに対し、本願発明での事前登録（Ｓ３．０）は、それらの情報を送信したり、送信された情報を認可サーバー２００で管理したりはしない。

【００５０】

次に、図６を用いて、ユーザーがクライアント４００にログインしてから、クライアント４００が認可サーバー２００に対して認可コード要求を送信するまでのフローを説明する。ユーザーはクライアント４００にログインする（Ｓ１．０）。その際のローカルユーザーＩＤは「ｌｏｃａｌ＿ｕｓｅｒ＿０１」であるものとする。クライアント４００はロ

ーカルユーザーＩＤを特定可能なログインコンテキストを生成し保持する。なおログインコンテキストはユーザーのログアウト操作で消滅するように構成することもできるし、ログインコンテキストの有効期限を設定しその時間の経過によって消滅するよう構成することもできる。

【００５１】

次に、ユーザーはＷｅｂブラウザーを介してクライアント４００の認可開始のＵＲＩにアクセスする（Ｓ１．１）。ユーザーエージェントがＷｅｂブラウザー４１０である場合、Ｗｅｂブラウザー４１０を起動してアクセスするか、もしくはＷｅｂブラウザー４１０のブックマークを用いてアクセスする事ができる。もしくは、クライアントアプリケーション４２０の不図示のＵＩを操作する事により、Ｗｅｂブラウザー４１０が起動し開始される構成にする事もできる。ユーザーエージェントがＷｅｂブラウザー５１０である場合は、Ｗｅｂブラウザー４１０がＷｅｂブラウザー５１０によるリモートアクセスを受信し、Ｗｅｂブラウザー５１０において認可開始のＵＲＩを入力してアクセスするか、ブックマークを用いてアクセスする事ができる。もしくはＷｅｂブラウザー５１０によるリモートアクセスによりクライアントアプリケーション４２０が不図示の画面を応答し、その画面に貼られている認可開始のＵＲＩへのリンクを押下する事でアクセスするという構成も考えられる。

【００５２】

Ｓ１．１でクライアント４００が認可開始ＵＲＩへのアクセスを受信すると、クライアント４００は認可サーバー２００の認可エンドポイントに対して認可コード要求を送信する（Ｓ４．０）。具体的には、認可サーバー２００の認可エンドポイントへのリダイレクト指示をＷｅｂブラウザーに送信する。Ｓ４．０の認可コード要求には、認可コード応答のレスポンスタイプに認可コードを指定するための情報と、認可コード要求と認可コード応答とを一意に紐づけるためのｓｔａｔｅパラメーターとを含む。

【００５３】

さらにＳ４．０の認可コード要求はＪＷＴ（ＪＳＯＮ Ｗｅｂ Ｔｏｋｅｎ）を含む。具体的にはＯＡｕｔｈ ２．０ ＪＷＴ Ｐｒｏｆｉｌｅにおいてｃｌｉｅｎｔ＿ａｓｓｅｒｔｉｏｎ＿ｔｙｐｅ：ｊｗｔ－ｂｅａｒｅｒ を宣言し、そのｃｌｉｅｎｔ＿ａｓｓｅｒｔｉｏｎとしてＪＷＴをパラメーターとして設定する。図７にＪＷＴをパラメーターに設定した際の認可コード要求の一例を示す。ＪＷＴはヘッダー部（「Ｈｅａｄｅｒ」から始まる部分）、ペイロード部（「Ｐａｙｌｏａｄ」から始まる部分）、デジタル署名部（「Ｅｎｃｏｄｅｄ」から始まる部分）から構成されており、それぞれがＢａｓｅ６４というエンコード方式によってエンコードされている。

【００５４】

ペイロード部には、「ｉｓｓ」（発行者を示す）および「ｓｕｂ」（主体を示す）としてクライアントＩＤ「ｃｌｉｅｎｔ＿０１」が設定される。「ａｕｄ」（利用者を示す）には、認可サーバー２００の認可エンドポイントのＵＲＩを設定し、「ｅｘｐ」（有効期限）、「ｉａｔ」（発行日時）も設定する。「ｃｌｉｅｎｔ＿ｎａｍｅ」にはクライアント名を設定し、「ｄｅｓｃｒｉｐｔｉｏｎ」にはクライアント４００の説明を設定する。今回は、クライアント４００のクライアント名として「デバイスＸＸ」が設定され、クライアント４００に関する説明として「ＹＹに設置されている￥ｒ￥ｎデバイスＸＸです。」が設定されている。「ｒｅｄｉｒｅｃｔ＿ｕｒｉ」にはリダイレクトＵＲＩが設定され、今回は「ｈｔｔｐｓ：／／１９２．１６８．１．１／ｒｅｄｉｒｅｃｔ」が設定されている。必要に応じて「ｉｃｏｎ＿ｉｍａｇｅ」にアイコン画像に関する情報をアイコン画像の画像形式とともに設定する。また、アイコン画像に関する情報として、アイコン画像が存在するＵＲＩを設定する事も可能であり、認可サーバー２００に保持している画像があれば、その画像を識別するための情報を設定する事も可能である。

【００５５】

各種情報を設定した後に、ヘッダー部およびペイロード部の文字列をＢａｓｅ６４でエンコードし、その文字列に対してクライアント４００が保持している秘密鍵を使ってデジ

タル署名を付与する。Ｓ４．０においてＪＷＴを取得した認可サーバー２００は、クライアントＩＤから公開鍵を特定し、その公開鍵を使ってＪＷＴに含まれるデジタル署名を検証する事でクライアント４００を認証し、ＪＷＴの各文字列が改竄されていない事を検証する。その結果、Ｓ４．０の認可コード要求のＪＷＴに含まれるリダイレクトＵＲＩはクライアント４００が設定したものであり、改竄されていない事が検証できる。

【００５６】

以上が、ユーザーがクライアント４００にログインしてから、クライアント４００が認可サーバー２００に対して認可コード要求を送信するまでのフローである。ＪＷＴを用いることで認可コード要求に含まれるリダイレクトＵＲＩを信用できるため、認可サーバー２００は事前登録されたリダイレクトＵＲＩと比較する必要もなく、さらには認可サーバー２００にリダイレクトＵＲＩを事前登録する必要もない。したがって、クライアント４００のＵＲＩが変更され、リダイレクトＵＲＩが変更された場合でも、変更後のクライアント４００のＵＲＩを用いて認可サーバー２００に認可コード要求として送信することができる。

【００５７】

また、クライアント名、説明、アイコン画像等、認可確認画面に表示されるクライアント４００に関する情報を表記する際の使用言語は、クライアント４００に格納されているローカルユーザーの使用言語に関する言語情報や、ＷｅｂブラウザーのＡｃｃｅｐｔ－Ｌａｎｇｕａｇｅヘッダーに設定されている言語情報（Ｗｅｂブラウザーからのリクエストに含まれる言語情報）に基づいて決定してもよい。つまり、Ｓ４．０の認可コード要求に含まれるクライアント４００に関する情報をこれらの言語情報に基づいて記載することもできる。これにより、認可サーバー２００はクライアント４００に関する情報を受信するだけで、クライアント４００にログインしているローカルユーザーの使用言語に即した認可確認画面をユーザーに提供することができる。

【００５８】

次に、Ｗｅｂブラウザーを介してユーザーにログイン画面を提示してからクライアント４００に認可コードを発行する処理について、図６を用いて説明する。認可エンドポイントに対して認可コード要求を受信した認可サーバー２００は、ユーザーが認可サーバー２００にログインしていない場合はログイン画面を提示する（Ｓ１．３）。ログイン画面の一例は図５ａに示した通りである。ユーザーはログイン画面５０００にユーザーＩＤとパスワードを入力し、ログインボタン５００３を押下することで認可サーバー２００に認証要求を送信する（Ｓ１．４）。認証要求を受信した認可サーバー２００は、ユーザーＩＤとパスワードの組み合わせが認可サーバー２００に事前登録されている情報と照合し、一致する場合は認証トークンを発行する。発行された認証トークンはＷｅｂブラウザーのＣｏｏｋｉｅに応答される。ここで、認証トークンはランダムな推測不可能な文字列で構成されてもよいし、ログインしたユーザーの識別情報とログイン日時を含んだ暗号化された文字列として構成する事もできる。前者の場合、認証トークンはログインしたユーザーの識別情報（本実施例ではユーザーＩＤ）と組み合わせて認可サーバー２００で保持される。本実施例のユーザーＩＤは「ｕｓｅｒ＿０１」であるものとする。

【００５９】

認可サーバー２００は認可確認画面をＷｅｂブラウザーに応答する（Ｓ１．５）。認可確認画面の一例は図５（ｂ）に示した通りである。ただし、Ｓ４．０の認可コード要求で受信したＪＷＴのデジタル署名を公開鍵で検証して不正だった場合、不図示のエラー画面を応答し処理を終了する。デジタル署名の検証の処理により、不正なＵＲＩへのリダイレクトを防ぐことができる。以降は、ＪＷＴのデジタル署名が正当であったものとして説明する。

【００６０】

Ｓ４．０の認可コード要求で受信したＪＷＴに含まれる値（クライアント名５１０１、説明５１０２、および、アイコン画像５１０３）に基づいて認可確認画面５１００がＷｅｂブラウザーに表示される。ここでユーザーが拒否ボタン５１０５を押下し、さらにクラ

イアントＩＤとリダイレクトＵＲＩの組み合わせが事前登録のものと一致した場合は、ユーザーがクライアント４００の認可を拒否した旨を示す情報を認可サーバー２００がリダイレクトＵＲＩのクエリパラメーターに付与する。そして、そのリダイレクトＵＲＩで指定された応答先に情報をリダイレクトするようにＷｅｂブラウザーに応答する。

【００６１】

このように、ＪＷＴを用いることで不正な認可コード要求を拒絶し、認可コード要求が拒絶されたことを示す表示画面をＷｅｂブラウザーに提供することができる。また、Ｓ４．０の要求が拒絶されなかったとしても、認可確認画面においてユーザーが認可を拒否することで、拒否されたことを示す情報をＷｅｂブラウザーに送信できる。

【００６２】

一方、ユーザーが許可ボタン５１０４を押下した場合は認可操作が実行され（Ｓ１．６）、認可サーバー２００は認可コードを発行する。Ｓ１．６で発行された認可コードおよび、Ｓ４．０の認可コード要求で受信したｓｔａｔｅパラメーターをクエリパラメーターとしてリダイレクトＵＲＩに付与し、そのリダイレクトＵＲＩで指定された応答先にＷｅｂブラウザーが認可コード及びｓｔａｔｅパラメーターをリダイレクトするよう応答する（Ｓ１．７）。発行された認可コードはクライアントＩＤ、ユーザーＩＤ、リダイレクトＵＲＩと紐づけて認可サーバー２００において保存される。認可サーバー２００において保存された認可コードは、後述のトークン要求を受信した際のクライアント４００の検証で用いられる。今回は、クライアントＩＤ「ｃｌｉｅｎｔ＿０１」、ユーザーＩＤ「ｕｓｅｒ＿０１」、リダイレクトＵＲＩ「ｈｔｔｐｓ：／／１９２．１６８．１．１／ｒｅｄｉｒｅｃｔ」と紐づけて認可コードが保存されるものとする。認可コードは推測不可能なランダムな文字列である必要があり、かつ有効期限を持つことが好ましい。また、認可サーバー２００は、ユーザーから認可の同意をうけたとして同意情報（ユーザーＩＤとクライアントＩＤ）をログインしているユーザーの情報として登録する。

【００６３】

Ｓ１．７で認可コード応答を受信したクライアント４００は、認可サーバー２００のトークンエンドポイントに対してトークン要求を行う（Ｓ５．０）。このトークン要求には認可フローがＡｕｔｈｏｒｉｚａｔｉｏｎ Ｃｏｄｅ Ｇｒａｎｔである事を示す定義「ｇｒａｎｔ＿ｔｙｐｅ＝ａｕｔｈｏｒｉｚａｔｉｏｎ＿ｃｏｄｅ」と、取得した認可コードおよびクライアント認証情報とを含むＪＷＴ（ＪＳＯＮ Ｗｅｂ Ｔｏｋｅｎ）である。ここで設定するＪＷＴは、より具体的にはＯＡｕｔｈ ２．０ ＪＷＴ Ｐｒｏｆｉｌｅにおけるｃｌｉｅｎｔ＿ａｓｓｅｒｔｉｏｎ＿ｔｙｐｅ：ｊｗｔ－ｂｅａｒｅｒ を宣言し、そのｃｌｉｅｎｔ＿ａｓｓｅｒｔｉｏｎとしてＪＷＴをパラメーターに設定する。図８にＪＷＴで示されたトークン要求の一例を示す。図７の認可コード要求と重複する部分については、詳細な説明は省略する。

【００６４】

Ｓ５．０でトークン要求を受けた認可サーバー２００は、クライアントＩＤから特定した公開鍵を利用してＪＷＴの署名を検証する。検証が成功しクライアント４００が認証されると認可トークンを発行し、クライアント４００にトークン応答を行う（Ｓ２．１）。認可サーバーのトークンエンドポイントに対してリフレッシュ要求を行う（Ｓ５．１）。リフレッシュ要求におけるクライアント４００の認証方式が、Ｓ２．２ではクライアントＩＤとシークレットの組み合わせを用いて照合してクライアント４００を認証する。それに対し、のに対し、Ｓ５．１ではクライアントＩＤに付与されたデジタル署名を秘密鍵で検証し、クライアント４００を認証する。以上が、Ｗｅｂブラウザーにログイン画面を提示してからクライアント４００に認可コードを発行する処理である。

【００６５】

次に、認可コード要求において設定するリダイレクトＵＲＩを決定するための処理について、図９を用いて説明する。図９の処理はクライアント４００においてリダイレクトＵＲＩの決定する処理フローである。本処理は、クライアント４００が認可フローの開始要求（Ｓ１．１）を受信したことにより開始される（Ｓ９．１）。クライアント４００は認

可フローの開始要求のＨｏｓｔヘッダーを取得する（Ｓ９．２）。クライアント４００は取得したＨｏｓｔヘッダーのドメイン部が「ｌｏｃａｌｈｏｓｔ」であるかを判定する（Ｓ９．３）。ｌｏｃａｌｈｏｓｔは、プログラムが実行されているデバイス自身を指すホスト名であり、今回であれば、認可フローの開始要求を送信したＷｅｂブラウザーを指す。Ｓ９．３の判定結果により、認可フローの開始要求を送信したＷｅｂブラウザーを特定することができる。今回は、Ｗｅｂブラウザー４１０が認可フローの開始要求をクライアント４００に送信したものとする。Ｈｏｓｔヘッダーが「ｌｏｃａｌｈｏｓｔ」であった場合は、リダイレクトＵＲＩのドメイン部を「ｌｏｃａｌｈｏｓｔ」に決定する（Ｓ９．８）。例えば、「ｈｔｔｐｓ：／／ｌｏｃａｌｈｏｓｔ／ｒｅｄｉｒｅｃｔ」がリダイレクトＵＲＩとなる。

【００６６】

Ｓ９．３においてＨｏｓｔヘッダーのドメイン部が「ｌｏｃａｌｈｏｓｔ」でなかった場合、クライアント４００はＨｏｓｔヘッダーのドメイン部がＩＰアドレスであるかを判断する（Ｓ９．４）。ＩＰアドレスでない場合はＳ９．２で取得したＨｏｓｔヘッダーを用いて、不図示のＤＮＳサーバーに問い合わせてＩＰアドレスを取得する（Ｓ９．５）。例えばＨｏｓｔヘッダーの例として「ｗｗｗ．ｃａｎｏｎ．ｊｐ：４４３」である場合、ドメイン（Ｆｕｌｌｙ　Ｑｕａｌｉｆｉｅｄ　Ｄｏｍａｉｎ　Ｎａｍｅ：　ＦＱＤＮ）である「ｗｗｗ．ｃａｎｏｎ．ｊｐ」に対してポート番号「４４３」が付与されている。その場合、Ｈｏｓｔヘッダーの一部としてドメイン部分のみを切り出して、ＤＮＳサーバーに問い合わせる。ＩＰアドレスを取得した後は、後述のＳ９．６の処理を実行する。

【００６７】

Ｓ９．４においてＨｏｓｔヘッダーのドメイン部がＩＰアドレスであった場合、クライアント４００に予め設定されているＩＰアドレスと取得したＩＰアドレスをクライアント４００で比較する（Ｓ９．６）。ＩＰアドレスが一致するか否かを判定し（Ｓ９．７）クライアント４００、ＩＰアドレスが一致しなかった場合は、Ｓ９．１で受信したアクセスが不正であるとしてエラー終了する（Ｓ９．９）。ＩＰアドレスが一致した場合はＳ９．１で受信したアクセスが正当であると判断し、Ｓ９．２で取得したＨｏｓｔヘッダーのドメイン部を利用したＵＲＩを作成し、作成したＵＲＩをリダイレクトＵＲＩとして決定する（Ｓ９．８）。以上が、クライアント４００におけるリダイレクトＵＲＩの決定方法である。これにより、ＩＰアドレスやホスト名が変更されても、ＯＡｕｔｈ　２．０の処理フローにおける認可フローの開始要求をきっかけにしてリダイレクトＵＲＩを決定する事ができる。

【００６８】

本実施例により、ＯＡｕｔｈ　２．０のＡｕｔｈｏｒｉｚａｔｉｏｎ　Ｃｏｄｅ　Ｇｒａｎｔの処理フローにおいて、安全性を損なうことなく、リダイレクトＵＲＩや認可確認画面に提示するためのクライアントの情報について事前に登録、管理する必要がなくなり、かつ動的な変更に対して簡易に対応する事が可能となる。

【００６９】

［その他の実施例］

認可コード要求の際に、認可の範囲を示すｓｃｏｐｅパラメーターを指定することもできる。例えば、認可コード要求時に指定されたｓｃｏｐｅパラメーターを認可コード、認可トークン、リフレッシュトークンと紐付けて管理してもよい。また、認可確認画面５１００を表示する際に、ｓｃｏｐｅパラメーターが示す認可の範囲を表示するように構成してもよい。

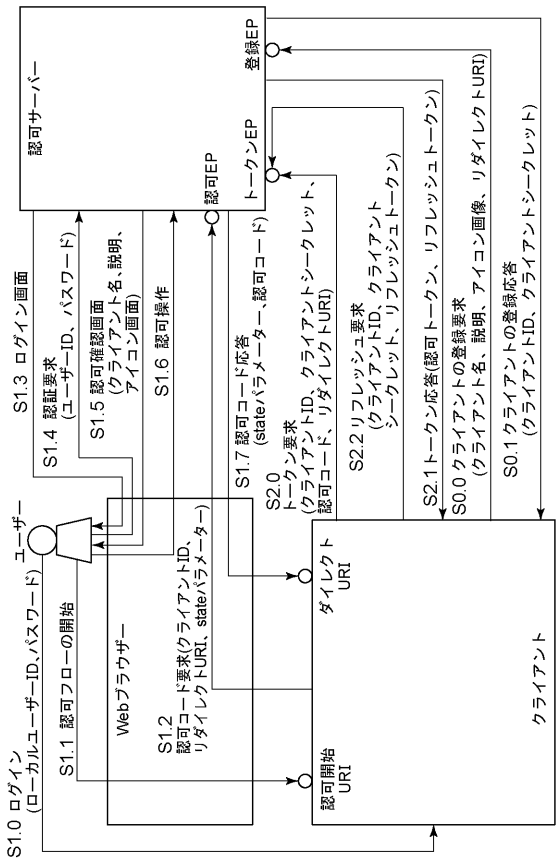
【００７０】

本発明は、以下の処理を実行することによっても実現される。即ち、上述した実施例の機能を実現するソフトウェア（プログラム）を、ネットワーク又は各種記憶媒体を介してシステム或いは装置に供給し、そのシステム或いは装置のコンピュータ（またはＣＰＵやＭＰＵ等）がプログラムを読み出して実行する処理である。

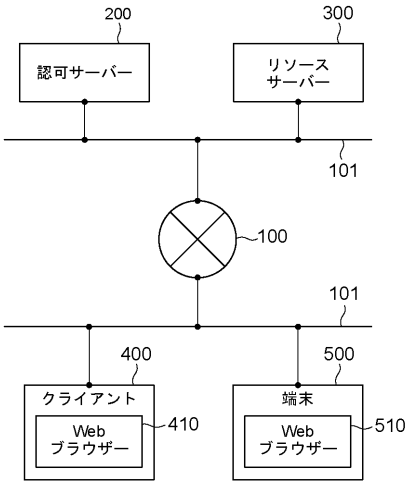
【符号の説明】

【００７１】
２００　認可サーバー
３００　リソースサーバー
４００　クライアント
５００　端末
４１０　Ｗｅｂブラウザー
５１０　Ｗｅｂブラウザー

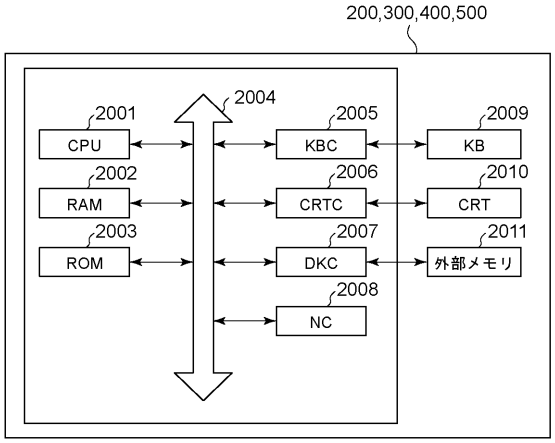
【図１】



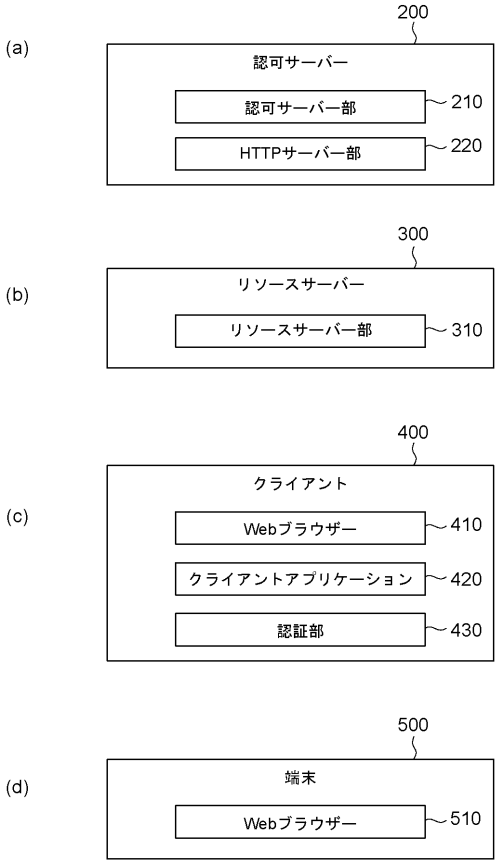
【図２】



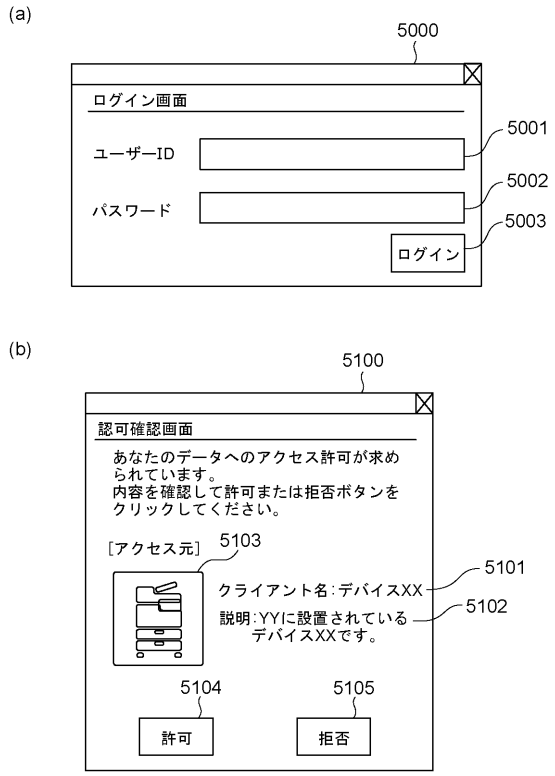
【図３】


200,300,400,500
2004
2001
CPU
2002
RAM
2003
ROM
2005
KBC
2006
CRTC
2007
DKC
2008
NC
2009
KB
2010
CRT
2011
外部メモリ

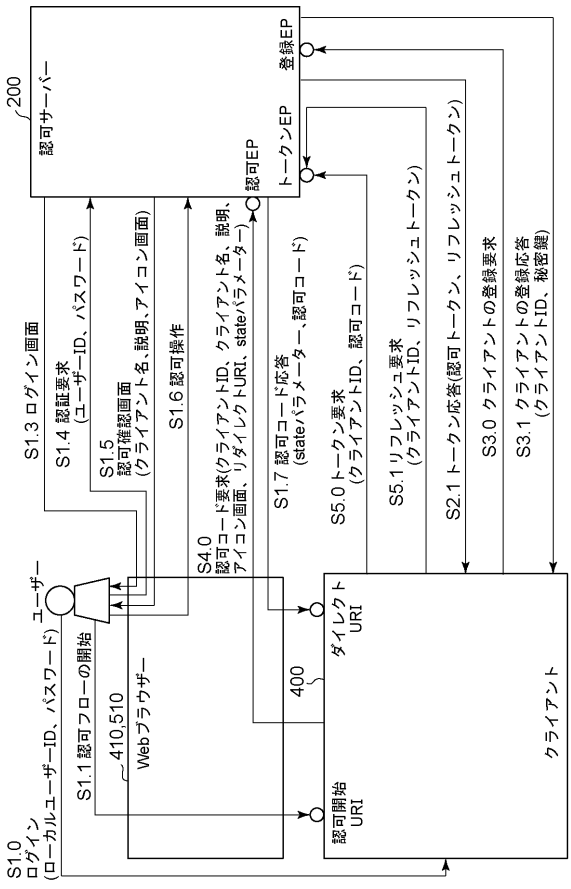
【図４】


(a)
200
認可サーバー
認可サーバー部
210
HTTPサーバー部
220
(b)
300
リソースサーバー
リソースサーバー部
310
(c)
400
クライアント
Webブラウザー
410
クライアントアプリケーション
420
認証部
430
(d)
500
端末
Webブラウザー
510

【図５】


(a)
5000
ログイン画面
ユーザーID
5001
パスワード
5002
ログイン
5003
(b)
5100
認可確認画面
あなたのデータへのアクセス許可が求められています。
内容を確認して許可または拒否ボタンをクリックしてください。
［アクセス元］
5103
クライアント名：デバイスXX
5101
説明：YYに設置されているデバイスXXです。
5102
5104
許可
5105
拒否

【図６】


200
認可サーバー
登録EP
トークンEP
認可EP
ユーザー
400
ダイレクトURI
クライアント
認可開始URI
Webブラウザー
410,510
S1.0 ログイン（ローカルユーザーID、パスワード）
S1.1 認可フローの開始
S1.3 ログイン画面
S1.4 認証要求（ユーザーID、パスワード）
S1.5 認可確認画面（クライアント名、説明、アイコン画面）
S1.6 認可操作
S1.7 認可コード応答（stateパラメーター、認可コード）
S2.1 トークン応答（認可トークン、リフレッシュトークン）
S3.0 クライアントの登録要求
S3.1 クライアントの登録応答（クライアントID、秘密鍵）
S4.0 認可コード要求（クライアントID、クライアント名、説明、アイコン画面、リダイレクトURI、stateパラメーター）
S5.0 トークン要求（クライアントID、認可コード）
S5.1 リフレッシュ要求（クライアントID、リフレッシュトークン）

【図７】

Authorization Request の JWT(JSON Web Token) 例

Header
{
“alg”: “RS256”,
“typ”: “JWT”
}

Payload
{
“iss”: “client_01”,
“sub”: “client_01”,
“aud”: “https://xxx.com/authrization”,
“exp”: 1472710413,
“iat”: 1472700413,
“client_name”: “デバイスXX”,
“description”: “YYに設置されている¥r¥nデバイスXXです。”,
“icon_image”: “
EAAAAAXNSR0IArs4c6QAAAARnQU1BAACxjwv8YQUAAAAJcEhZcwA
ADsMAAA7DAcdvqGQAAAPqSURBVGhD7ZrbS+NAFIfznwv2oY+VQDEg
uIKgiA+BlkL･･･(base64 encoded data)”,
“redirect_uri”: “https://192.168.1.1/redirect”
}

Encoded
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9･･･.(base64 encoded
header).eyJpc3MiOiJjbGllbnQwMDEiLCJzdWIiOiJodHRwczovL3h4eC5jb
20vIiwiYXVkIjoiaHR0cHM6Ly94eHguY29tL2F1dGhyaXphdGlvbiIsImV4cCI
6MTQ3MjcxMDQxMywiaWF0IjoxNDcyNzAwNDEzfQ･･･(base64 encoded
payload).k_SbabNV･･･(Signature)

【図８】

Token Request の JWT(JSON Web Token) 例

Header
{
“alg”: “RS256”,
“typ”: “JWT”
}

Payload
{
“iss”: “client_01”,
“sub”: “client_01”,
“aud”: “https://xxx.com/token”,
“exp”: 1472710413,
“iat”: 1472700413
}

Encoded
eyJhbGciOiJSUzI1NiIsInR5c･･･(base64 encoded
header).eyJpc3MiOiJjbGllbnQwMDEiLCJzdWIiOiJjbGllbnQwMDEiLCJhd
WQiOiJodHRwczovL3h4eC5jb20vdG9rZW4iLCJleHAiOjE0NzI3MTA0MT
MsImlhdCI6MTQ3Mjc･･･.(base64 encode
payload).IBFA00uTUAchvRyXWMlCqv5P･･･(Signature)

【図９】