SSO（Single Sign On）

●エージェント型とリバプロ型は、同一ドメイン内の SSO を実現するだけであるが、アイデンティティ連携型（フェデレーションョン型）は、異なるドメイン間の SSO を実現させる。

エージェント型（Cookie）

●各 Web サーバにエージェントソフトをインストール。

●クライアントが最初にアクセスした Web サーバのエージェントは、クライアントから送られてきた認証情報をもとに、代行して認証サーバにアクセスして認証を行う。

●そして、識別情報が入った Cookie 等のチケットをクライアントに返す。

●以降、クライアントは、チケットと共に他の Web サーバにアクセスすると、当該 Web サーバのエージェントはチケットをもとに代行して認証サーバで認証を行う。

●結局、バックグラウンド（BG）では毎回認証が行われ、クライアントが意識する必要がないだけ。

リバプロ型

●認証も中継も全てリバプロが行う方式であり、リバプロの負荷が大きい。

アイデンティティ連携型

●アイデンティティ＝利用者の属性情報

①識別子（アイデンティファイア）

②認証情報（クレデンシャル：ユーザ ID、PW、メアド、生体情報等）

③所属や役職の属性値（アトリビュート）等

④クレデンシャルの定義から、PW リスト攻撃はクレデンシャルスタッフィング攻撃とも呼ばれる。

●SAML**（認証トークンから Cookie）**

①OASIS（標準化団体）により策定された SOAP（規約）をベースとした技術で、アイデンティティ連携型（フェデレーション型）の SSO を実現する。

②認証情報と認可情報等を異なるドメインの Web サービス間で交換するためのフレームワーク。

③クライアントが最初にアクセスした SP（Service Provider）は、**リダイレクト処理（クライアント経由の SAML Request）**により、クライアントと IdP（Identity Provider）との間で認証させ、認証されれば、IdP は**認証トークン（認証アサーション）**を発行し、**リダイレクト処理（クライアント経由の署名を付した SAML Response）**により認証トークンを SP に送る。

④SP が**署名**と**認証トークン**を確認したら、SP は **Cookie 等**を発行して、クライアントにサービスを提供する。

⑤以降、クライアントは、当該**Cookie 等**を使用することで、他の SP に対して、認証なし（BG における認証もない）でアクセスできる。
⑥エージェント型では、BG での認証があったが、SAML では、それはない。なぜなら、上記③の**Cookie 等**を他の SP に信用してもらうために、予め、IdP と各 SP は「信頼の輪」を形成しておくためである。

OAuth2.0（Open Authorization ver2.0）（認可コードからアクセストークン）

●サードパーティアプリケーション（サードアプリ）によるリソースサーバへの限定的なアクセスを可能にする認可フレームワーク。具体的な例として、ユーザは、リソースサーバである Google Photo のアカウントを持つが、クライアント端末の画像編集アプリが Google 製でない場合、この画像編集アプリはサードアプリとなり、リソースサーバにアクセスできないため、OAuth を使う。
●ユーザ（Web ブラウザ）はクライアント（サードアプリ）に対し、リソースサーバとのデータの受け渡しを要求するが、クライアントはその権限を持っていないため、クライアントは、Web ブラウザ経由のリダイレクト処理により、ユーザと認可サーバとの間の認可処理を行わせる。
●そして、認可サーバは、Web ブラウザ経由のリダイレクト処理により、クライアントに**認可コード**を送る。クライアントは、**アクセストークン**要求のために、その**認可コード**を認可サーバに送ると、認可サーバから**アクセストークン**が送られてくるので、その**アクセストークン**をリソースサーバに送ることで、ようやくデータの**受け渡し**が開始される。
●クライアントとリソースサーバとのデータの**受け渡し**には API が使用されているため、リソースサーバは API を介してアクセストークンの検証をする。
●上記はリソースサーバと認可サーバが別々であるが、一体となっている場合もある。その他、定型なパターンはないので、OAuth も含め、SAML や FIDO 等は問題をよく読まないと間違える。