Mapping the Future of OAuth: Insights from the IETF WG

Michel Bonfim[1], **Marcelo Santos**[2], **Julião Braga**[3]

[1]Universidade Federal do Ceará (UFC) – Campus Quixadá
Quixadá – CE – Brasil

[2]Instituto Federal do Sertão Pernambucano – Campus Salgueiro
Salgueiro – PE – Brasil

[3]Universidade Federal do ABC (UFABC)
São Paulo – SP – Brasil

michelsb@ufc.br, marcelo.santos@ifsertao-pe.edu.br, j@braga.net.br

Abstract. *This paper provides a systematic overview and in-depth analysis of the key active Internet-Drafts within the Internet Engineering Task Force (IETF) OAuth Working Group (WG). By focusing on proposals that have achieved consensus (WG Call for Adoption), we identify the core trends shaping the future of the OAuth 2.0 protocol, with a strong emphasis on security enhancements, user privacy, and granular authorization.*

1. Introduction

OAuth 2.0, formalized in the specifications RFC 6749 (The Framework) and RFC 6750 (Bearer Token Usage) [Hardt 2012, Jones and Hardt 2012], has consolidated its position as the industry-standard protocol for delegated authorization. Its modular and extensible architecture has driven widespread adoption in scenarios ranging from large social and financial platforms to microservices architectures and Internet of Things (IoT) (Internet of Things) device ecosystems. However, the open nature of the internet and the rapid pace of innovation and security threats demand a continuous evolution of its security mechanisms and functionalities.

The OAuth WG[1] of the IETF conducts this process of refinement and standardization. The WG serves as the central forum responsible for identifying vulnerabilities—such as the historical redirect security issue that led to the development of Proof Key for Code Exchange (PKCE) (RFC 7636) [Sakimura et al. 2015]—and for creating new extensions, such as Demonstration of Proof-of-Possession (DPoP) (RFC 9449) [Fett et al. 2023]. This work ensures the protocol remains secure against emerging threats and supports new use cases and operating environments.

The OAuth WG has a rich history of standardization, with the turning point being the publication of OAuth 2.0 in 2012 (RFC 6749 and RFC 6750) [Hardt 2012, Jones and Hardt 2012]. Since then, the WG has actively worked on creating Best Current Practices (BCPs) and launching the OAuth 2.1 initiative [Hardt et al. 2025] to unify and simplify the core specification.

Despite this evolutionary trajectory, the inherent complexity of the base protocol and its vast ecosystem of 33 published RFCs and 14 active Internet-Drafts (as of

[1]https://datatracker.ietf.org/wg/oauth/about/

November 2025) presents a significant challenge for researchers and implementers. Furthermore, the protocol's development requires continuous engagement with other groups, such as the Authentication and Authorization for Constrained Environments Working Group (ACE WG)[2] and the Drone Remote Identification Protocol Working Group (DRIP WG)[3], underscoring the challenge of satisfying new interoperability requirements, particularly regarding OpenID Connect (OIDC).

In the face of this documentation dispersion, this work provides a systematic, prospective survey focused exclusively on the active Internet-Drafts within the WG, serving as a strategic guide to the evolution of OAuth 2.0, providing a structured overview of its ongoing development. The analysis identifies the future trajectory of OAuth 2.0 based on three core contributions:

1. Technological Trends: Mapping the evolution towards Security by Default (OAuth 2.1, PKCE) and Privacy by Design (Selective Disclosure for JSON Web Tokens (SD-JWT), Verifiable Credentials (VCs)).
2. Security Challenges: Analyzing new mitigation strategies, including Software Attestation, mandatory Proof-of-Possession (DPoP), and resolutions for critical IETF 124 risks (e.g., Server-Side Request Forgery (SSRF) and Code Invalidation).
3. New Application Areas: Delimiting the protocol's expansion into complex contexts, such as Cross-Domain Chaining [Schwenkschuster et al. 2025] and constrained IoT/Machine-to-Machine (M2M) environments (ACE WG, Transaction Tokens (TTs)), while addressing Multi-instance Scalability challenges.

The rest of this paper is organized as follows. Section 2 establishes the methodology, centering on the 14 adopted Drafts. Section 3 then structures the organization of drafts by category. Section 4 delivers a detailed analysis within each thematic group. Section 5 describes key concepts such as SD-JWT and TTs. Section 6 presents a summary of the identified trends and gaps. Finally, Section 7 concludes with a final synthesis.

2. Methodology and Scope

This survey is designed to reflect the most probable future trajectory of the OAuth protocol. The methodology focuses on documents that have achieved a formal level of consensus within the OAuth WG.

2.1. Defining the Scope of Active Documents

This survey is strictly limited to the 14 active Internet-Drafts listed in the IETF DataTracker [Hardt et al. 2025, Looker et al. 2025a, Kasselman et al. 2025, Fett et al. 2025, Terbu et al. 2025, Tulshibagwale et al. 2025a, Looker et al. 2025c, Parecki et al. 2025e, Parecki et al. 2025a, Parecki et al. 2025d, Parecki and Smith 2025b, Schwenkschuster et al. 2025, Sheffer et al. 2025, Jones et al. 2025] under the OAuth WG that have passed the WG Call for Adoption, and whose status was verified as of November 2025. The decision to exclude individual drafts (those not formally adopted by the WG) is justified because the WG adoption process signals that the proposal has reached a minimum consensus and is considered essential for the protocol's future.

[2]`https://datatracker.ietf.org/group/ace/about/`

[3]`https://datatracker.ietf.org/group/drip/about/`

Individual drafts, while numerous, represent ideas still in the initial discussion stages and have a much lower probability of becoming a final RFC. These documents represent the areas of greatest focus and time investment by the group. Each draft was categorized thematically to facilitate trend analysis.

3. Draft Categorization

To facilitate the identification of key trends and innovation areas, the 14 adopted Internet-Drafts were grouped into five main thematic categories. This categorization, presented in Table 1, allows for a structured view of the WG's primary areas of focus.

Table 1. Thematic Classification of Active Internet-Drafts in the OAuth WG

Category	Main Focus	Related Drafts
Core Consolidation and Evolution	Updating and simplifying the main framework (OAuth 2.0 to 2.1).	[Hardt et al. 2025]
Advanced Security and Authentication	Client authentication mechanisms and proof of possession.	[Looker et al. 2025a], [Kasselman et al. 2025]
Advanced Tokens and Credentials	Use of JSON Web Tokens (JWTs), verifiable credentials, and new token types.	[Fett et al. 2025], [Terbu et al. 2025], [Tulshibagwale et al. 2025a], [Looker et al. 2025c]
Specific Use Cases and Flows	Applications in browsers, devices, and first-party apps.	[Parecki et al. 2025e], [Parecki et al. 2025a]
Metadata and Interoperability	Discovery and chaining of identity/authorization, RFC updates.	[Parecki and Smith 2025b], [Schwenkschuster et al. 2025], [Sheffer et al. 2025], , [Parecki et al. 2025d], [Jones et al. 2025]

4. Detailed Analysis by Category

This section provides a detailed analysis of the drafts within each thematic category, describing each document's purpose, problem addressed, and proposed solution. The inclusion of each draft is justified based on its primary focus area, which drives the current evolution of OAuth 2.0.

4.1. Core Consolidation and Evolution

This category focuses on efforts to modernize and simplify the core OAuth 2.0 protocol, aiming for a single, unified standard. The OAuth 2.1 Authorization Framework [Hardt et al. 2025] represents the mandatory consolidation of security best practices developed since the publication of the core OAuth 2.0 specifications [Hardt 2012, Jones and Hardt 2012]. Its primary goal is to simplify the standard, enhance security by default, and eliminate historically insecure patterns.

The following elements of the original OAuth 2.0 specification are explicitly deprecated and removed from OAuth 2.1:

- Implicit Grant Flow (RFC 6749, Section 4.2): Removed entirely due to its vulnerability to token leakage via the URL fragment and its inability to enforce PKCE (Proof Key for Code Exchange).
- Resource Owner Password Credentials (ROPC) Grant Flow (RFC 6749, Section 4.3): Removed because it requires the client application to handle the user's plaintext credentials, presenting a high security risk. It should be replaced by more secure methods, such as the Authorization Code flow or OIDC.
- Bearer Token Usage in URL Query Parameters (RFC 6750): Transmitting Bearer Tokens via URI query parameters is forbidden as it leaks tokens into server logs and browser history. This prohibition, though existing, is made explicit and mandatory in OAuth 2.1.

It is worth mentioning that, while technically an extension (RFC 7636) in OAuth 2.0, PKCE is mandatory for all public clients using the Authorization Code flow in OAuth 2.1, effectively removing the option for public clients to omit it. Moreover, OAuth 2.1 essentially boils down the authorization process to two primary, secure flows: the Authorization Code Grant with PKCE (for clients with a user interface) and the Client Credentials Grant (for M2M authorization). We describe both flows below.

The Authorization Code flow is the recommended standard for clients capable of interacting with the resource owner's User Agent (UA) (browser), such as Single-Page Applications (SPAs), mobile apps, and traditional web applications. The inclusion of PKCE ensures security even for public clients that cannot maintain a confidential secret. The flow, detailed in Figure 1, operates as follows:

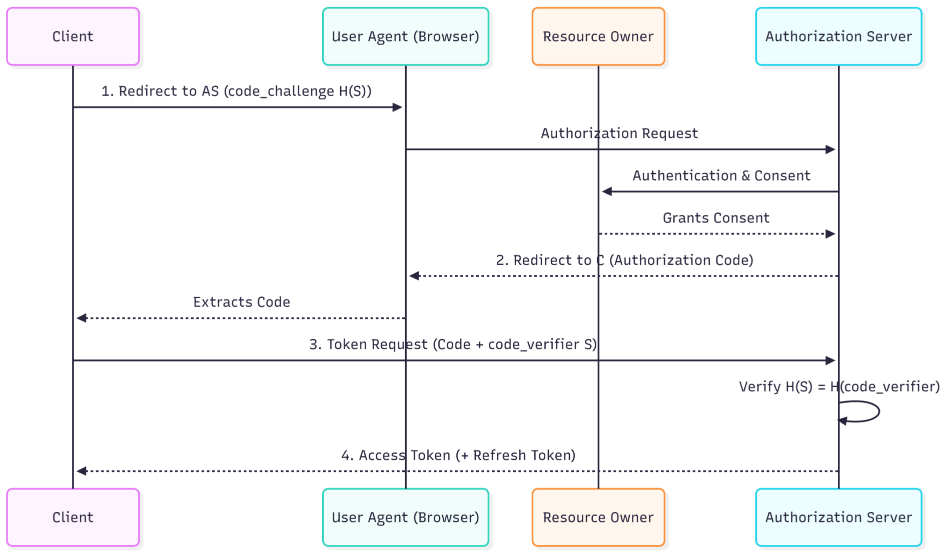

Figure 1. Authorization Code Grant Flow with PKCE: The standard flow for delegated authorization in OAuth 2.1, securing public clients via the use of `code_verifier` and `code_challenge`.

1. C $\rightarrow$ Authorization Server (AS): Authorization Request with Code Challenge (H(S)). The Client (C) initiates the flow by redirecting the UA to the AS. The request includes the generated `code_challenge` (H(S)) and a `redirect_uri`.

2. AS → C: Authorization Code (Code). After the Resource Owner (RO) authenticates and grants consent, the AS issues a short-lived Authorization Code and redirects the UA back to the Client's `redirect_uri`.
3. C → AS: Token Exchange Request with Code Verifier (S). The Client sends a back-channel request directly to the AS's token endpoint. This request contains the Code received in Step 2 and the original, unhashed `code_verifier` (S).
4. AS → C: Access Token. The AS validates the Code, calculates the hash of the received S, and verifies it matches the stored H(S). If validation succeeds, the AS issues the Access Token (and optionally a Refresh Token).

Finally, the Client Credentials flow is designed exclusively for M2M authorization, where the client (typically a server or backend service) is acting on its own behalf, controlling resources it owns, or acting as an application layer. This flow is always used by Confidential Clients (those capable of protecting a secret) and is not used for delegated user authorization. Figure 2 illustrates this flow. It operates as follows:

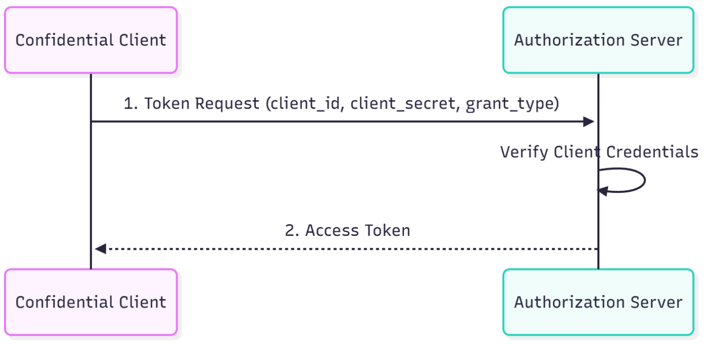

Figure 2. Client Credentials Grant Flow. A simple M2M flow where the Confidential Client authenticates itself directly to the AS to obtain an Access Token.

1. C → AS: Token Request with Client Credentials. The Confidential Client (C) sends a request directly to the AS's token endpoint. This request is authenticated using the Client's own credentials (`client_id` and `client_secret`) and specifies the grant type as `client_credentials`.
2. AS → C: Access Token. The AS verifies the Client's credentials. If valid, it immediately issues an Access Token corresponding to the permissions granted to the client application itself (and not a user).

4.2. Advanced Security and Authentication

These drafts seek to raise the security baseline by introducing robust client authentication mechanisms and improving practices for complex, high-risk flows, such as those involving multiple devices. This category includes drafts that go beyond the basic PKCE requirement.

The Attestation-Based Client Authentication draft [Looker et al. 2025a] addresses the problem that traditional client authentication methods fail to prove the integrity of the client software itself. Its proposed solution defines a method using software attestation to provide cryptographic proof of client integrity, ensuring the client is genuine and not a malicious imitation. This elevates authentication from proving identity to proving the integrity of the client environment.

The Cross-Device Flows: Security Best Practices draft [Kasselman et al. 2025] targets the security challenges inherent in multi-device authorization flows, which introduce attack vectors like phishing and Denial-of-Service (DoS). The solution outlines best practices for mitigating these risks in cross-device authorization flows, thereby enhancing overall user security when interacting across multiple devices.

4.3. Advanced Tokens and Credentials

This category encompasses innovations focused on the format, use, and management of access tokens, emphasizing privacy, fine-grained control, and efficient revocation in distributed systems.

The SD-JWT draft [Fett et al. 2025] solves the privacy problem where traditional JWTs exposes all claims, even those unnecessary for the verification step. SD-JWT enables selective disclosure of claims by the token holder, allowing the verifier to validate only the necessary claims without accessing the full token payload, thereby preserving privacy.

The SD-JWT-based VCs draft [Terbu et al. 2025] builds upon the SD-JWT work by specifying formats and validation rules for using SD-JWT to structure VCs. This addresses the need for a secure, standardized format that integrates selective disclosure capabilities into VCs.

The TTs draft [Tulshibagwale et al. 2025a] addresses the limitation of broad authorization scopes by defining a new token type. These tokens represent specific actions (e.g., transfer $X to account Y) rather than broad scopes, enabling fine-grained, context-aware authorization policies.

Finally, the Token Status List (TSL) draft [Looker et al. 2025c] tackles the issue of inefficient token revocation in distributed environments. The proposed solution defines a standardized format for publishing lists of revoked or suspended tokens, streamlining the revocation status check.

4.4. Specific Use Cases and Flows

These drafts provide tailored security guidance and protocol definitions for specific application environments, acknowledging that different client types have unique security requirements and usage patterns.

The OAuth for Browser-Based Applications draft [Parecki et al. 2025e] specifically addresses the unique security challenges faced by SPAs running in web browsers. It describes best practices and mandatory configurations for OAuth flows tailored to SPAs.

The OAuth for First-Party Applications draft [Parecki et al. 2025a] recognizes that applications owned by the resource provider (first-party apps) have different security needs and dynamics compared to third-party applications. The draft provides best practices tailored to secure these first-party apps.

4.5. Metadata and Interoperability

This category encompasses efforts aimed at improving the discovery of client and AS capabilities, enhancing interoperability, and updating existing security-critical specifications to reflect current best practices.

The Client ID Metadata Document draft [Parecki and Smith 2025b] addresses the lack of a standardized way for clients to publish their metadata. The draft defines a standard format and discovery mechanism for client metadata, which aids authorization servers in client registration and management.

The Identity and Authorization Chaining draft [Schwenkschuster et al. 2025] focuses on the complexity involved in chaining identity and authorization across multiple distinct domains or services. It specifies methods for securely linking and chaining these processes.

The JWT Client Authentication and Assertion Grants Update draft [Jones et al. 2025] updates RFC 7523 with improvements and clarifications regarding the usage of JWTs in client authentication and assertion grants. This refinement is decisive for systems relying on JWTs for client identity and delegation.

The Identity Assertion JWT Authorization Grant (ID-JAG) draft [Parecki et al. 2025d] defines a standardized grant type using JWT identity assertions. This addresses the need for a standardized flow that allows a client to request a token based on an existing, verified identity assertion.

Finally, the JWT Best Practices Update draft [Sheffer et al. 2025] serves to update RFC 8725 with new best practices and guidance related to the security of JWTs, reflecting evolving security challenges.

5. Key Technical Concepts

The active drafts introduce or formalize several technical concepts fundamental to the next generation of OAuth 2.0. The primary mechanisms driving innovation in security and privacy are described in detail below.

5.1. PKCE (Proof Key for Code Exchange)

PKCE is a security mechanism introduced by RFC 7636 [Sakimura et al. 2015] and is now mandatory in OAuth 2.1 [Hardt et al. 2025]. Its purpose is to mitigate authorization code interception attacks common in public clients, such as mobile or SPAs. Figure 3 shows how PKCE works.

The PKCE is a multi-step process involving the Client (C) and the AS. The client starts the process by dynamically generating a highly secret, single-use string, the `code_verifier` (S) (Step 1). The client then calculates a hashed version of this secret, the `code_challenge` (H(S)), and sends this hash to the AS along with the authorization request (Step 2). The AS securely stores the H(S) (Step 3) and redirects the user back to the client with the Authorization Code (Code) (Step 4). To mitigate the risk of interception, when the client attempts to exchange the code for an access token, it must send the Code along with the original, unhashed `code_verifier` (S) to the AS (Step 5). The AS performs a validation step by hashing the received `code_verifier` (S) and

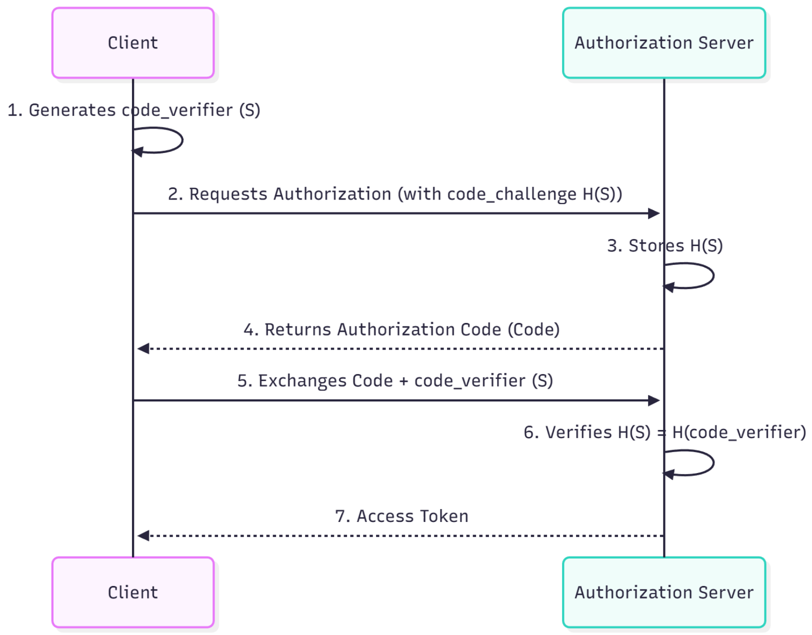
Client
Authorization Server
1. Generates code_verifier (S)
2. Requests Authorization (with code_challenge H(S))
3. Stores H(S)
4. Returns Authorization Code (Code)
5. Exchanges Code + code_verifier (S)
6. Verifies H(S) = H(code_verifier)
7. Access Token
Client
Authorization Server

Figure 3. PKCE (Proof Key for Code Exchange) Flow: The client proves key possession to prevent authorization code interception.

comparing the result with the stored H(S) (Step 6). Only if the hashes match does the AS issue the Access Token to the client (Step 7), proving that the client receiving the token is the same client that initiated the flow.

5.2. SD-JWT (Selective Disclosure for JWTs)

SD-JWT, detailed in draft [Fett et al. 2025], allows JWT token holders to selectively reveal only the necessary claims to a verifier, preserving privacy regarding the token's remaining claims.

The Figure 4 illustrates the emission and selective disclosure process of an SD-JWT. The flow begins when the Issuer creates the SD-JWT with certain claims cryptographically hidden, then signs and emits it to the Holder (Step 1). The Holder decides which specific claims are required by the Verifier and generates cryptographic proofs, known as disclosures. The Holder then presents the SD-JWT along with these Selective Disclosures to the Verifier (Step 2). The Verifier first checks the integrity of the credential by verifying the SD-JWT Signature (Step 3). Simultaneously, the Verifier uses the presented Disclosures to reconstruct and validate the hashes of the selectively revealed claims (Step 4). If both the signature and the integrity of the disclosed claims are proven to be valid, the Verifier grants Access based on the necessary claims. If the proof is invalid, access is denied.

This is particularly relevant for use in SD-JWT-based VCs [Terbu et al. 2025], where privacy is paramount.

5.3. Attestation-Based Client Authentication

Attestation-Based Client Authentication, proposed in draft [Looker et al. 2025a], addresses the challenge of securely authenticating public clients by ensuring not only their

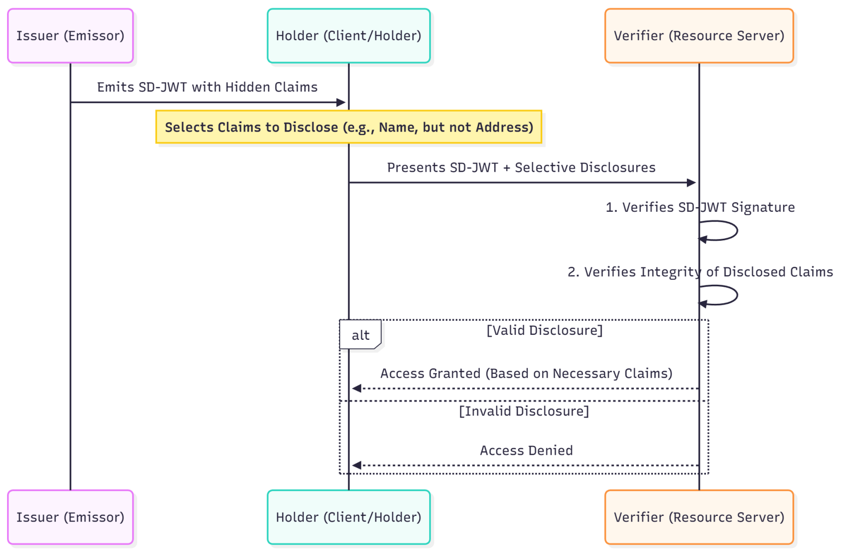
Issuer (Emissor)
Holder (Client/Holder)
Verifier (Resource Server)
Emits SD-JWT with Hidden Claims
Selects Claims to Disclose (e.g., Name, but not Address)
Presents SD-JWT + Selective Disclosures
1. Verifies SD-JWT Signature
2. Verifies Integrity of Disclosed Claims
alt
[Valid Disclosure]
Access Granted (Based on Necessary Claims)
[Invalid Disclosure]
Access Denied

Figure 4. SD-JWT (Selective Disclosure for JWTs) Flow: Demonstrates the process of issuing and selectively disclosing claims to preserve privacy.

identity but also their integrity. Traditional clients (like mobile or desktop apps) can have their identity spoofed. Software attestation uses operating system security mechanisms (e.g., Apple App Attest, Android Play Integrity) to generate cryptographic proof that the client software is genuine and has not been tampered with. This proof is then used as part of the client's authentication to the AS. It ensures that the client is genuine and not a malicious imitation.

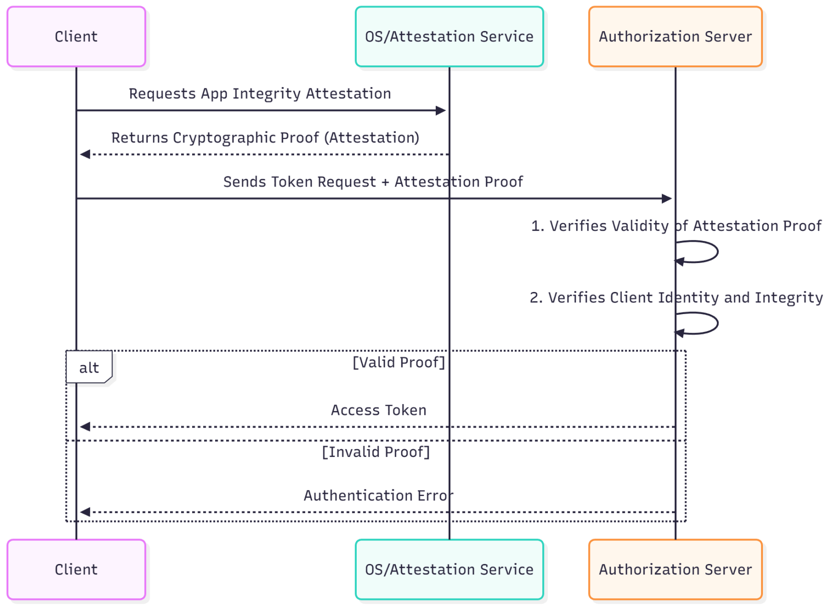
Client
OS/Attestation Service
Authorization Server
Requests App Integrity Attestation
Returns Cryptographic Proof (Attestation)
Sends Token Request + Attestation Proof
1. Verifies Validity of Attestation Proof
2. Verifies Client Identity and Integrity
alt
[Valid Proof]
Access Token
[Invalid Proof]
Authentication Error

Figure 5. Attestation-Based Client Authentication Flow: Use of cryptographic proof of software integrity (Attestation) to authenticate the client.

The Figure 5 shows how attestation is incorporated into the authentication flow.

The flow starts with the Client (C) requesting an attestation from the operating system or a trusted Attestation Service (OS) to confirm the application's integrity (Step 1). The OS returns a Cryptographic Proof (Attestation) to the Client (Step 2). The Client then includes this Attestation Proof when requesting an access token from the AS (Step 3). The AS first validates the authenticity of the Attestation Proof itself using the OS/Attestation Service's public key (Step 4). Next, the AS verifies that the proof specifically validates the registered Client's identity and software integrity (Step 5). If the integrity is proven, the Access Token is issued; otherwise, an Authentication Error occurs, protecting the system from spoofed clients.

5.4. Transaction Tokens TTs

The concept of *TTs* [Tulshibagwale et al. 2025a] is a response to the limitations of broad scopes. Instead of a token granting general access (e.g., `account.read` or `account.write`), a TT is tied to a specific action with exact parameters within a contextualized transaction (e.g., "transfer $100.00 to account X"). This enables fine-grained, context-aware authorization policies. This allows the Resource Server (RS) to apply much more granular and contextual authorization policies, raising the security and auditability level of operations.

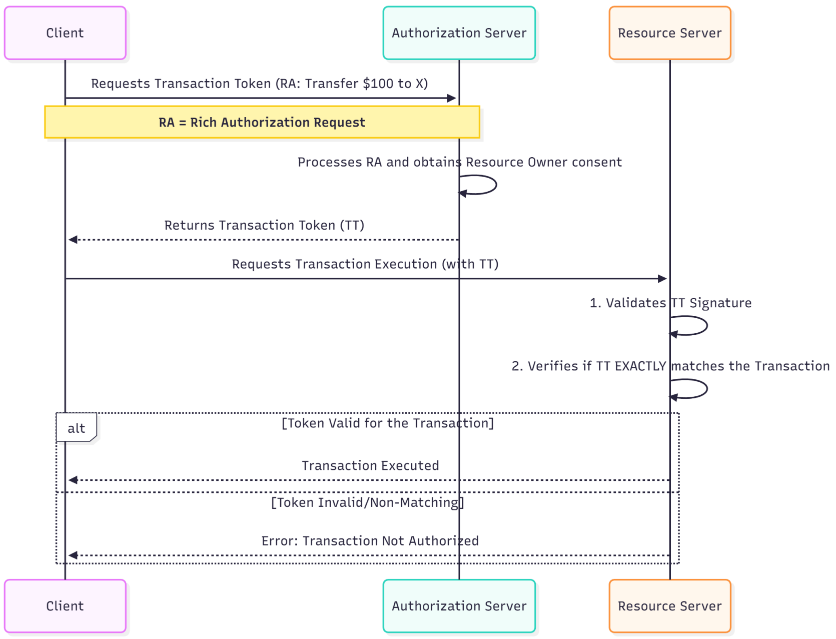
Client
Authorization Server
Resource Server
Requests Transaction Token (RA: Transfer $100 to X)
RA = Rich Authorization Request
Processes RA and obtains Resource Owner consent
Returns Transaction Token (TT)
Requests Transaction Execution (with TT)
1. Validates TT Signature
2. Verifies if TT EXACTLY matches the Transaction
alt
[Token Valid for the Transaction]
Transaction Executed
[Token Invalid/Non-Matching]
Error: Transaction Not Authorized
Client
Authorization Server
Resource Server

Figure 6. TT Life Cycle: Life cycle of a TT, from the request (Rich Authorization Request (RA)) to the transaction execution at the RS.

Figure 6 illustrates the life cycle of a TT. The TT flow enables granular, action-specific authorization using a RA. The Client (C) initiates the process by sending an RA (detailing the specific transaction, e.g., "Transfer $100 to X") to the AS (Step 1). The AS processes the RA and obtains the RO's consent specifically for the detailed transaction (Step 2). The AS then mints a TT containing the precise authorized transaction details and returns it to the Client (Step 3). The Client uses the TT to request the execution of the transaction at the RS (Step 4). The RS validates the token's signature (Step 5) and then performs a critical check to verify that the transaction requested EXACTLY matches the authorization coded in the TT (Step 6). Only if the token is fully valid for the transaction is the transaction executed; otherwise, an error indicates the transaction is Not Authorized due to a mismatch.

5.5. ID-JAG

The ID-JAG [Parecki et al. 2025d] defines a new OAuth 2.0 grant type crucial for delegated authorization in complex, multi-domain environments. This grant type standardizes the process where a client uses a JWT-based identity assertion (often an ID Token obtained via Single Sign-On (SSO), e.g., OIDC) to request an Access Token from a separate AS (typically a RS AS). ID-JAG is a profile of the Identity and Authorization Chaining specification, designed to facilitate secure and standardized cross-domain delegation. Key architectural challenges addressed by this proposal include ensuring Proof-of-Possession (PoP) binding across the entire assertion and token chain, and supporting Multi-tenancy and Multi-instance environments [Parecki et al. 2025c].

6. Trends and Gaps

The analysis of the active Internet-Drafts clearly identifies the main development directions for the OAuth 2.0 protocol and, simultaneously, areas that still require more attention from the WG.

6.1. Emerging Trends

The analysis of the adopted drafts reveals three major developmental trends reflecting the industry's response to security vulnerabilities and the demand for enhanced user control in decentralized identity environments.

- Security by Default - Raising the Minimum Baseline: This trend is centered on transforming optional security extensions into mandatory features, drastically reducing the attack surface. The core driver is the OAuth 2.1 [Hardt et al. 2025] consolidation, which mandates PKCE [Sakimura et al. 2015] for all public clients and removes insecure flows (Implicit Grant, ROPC). Furthermore, security is being extended to the client's execution environment through *attestation-based authentication* [Looker et al. 2025a], a step that provides cryptographic proof of the client software's integrity against tampering. This shifts the responsibility from developers manually implementing BCPs to making secure practices the protocol's default setting;
- Privacy and Decentralized Identity - User Control over Data Disclosure: The OAuth ecosystem is adapting to the privacy demands of decentralized identity models. The primary mechanism is the SD-JWT [Fett et al. 2025], which allows the data owner (Holder) to reveal only the minimal claims required by a Verifier, thereby preventing unnecessary data exposure common with standard JWTs. This capability is directly applied to VCs through SD-JWT-based VCs [Terbu et al. 2025], facilitating secure and privacy-preserving data exchange in contexts like digital identity wallets;
- Contextual Authorization - Moving Beyond Coarse Scopes: This trend addresses the limitation of broad OAuth scopes, which historically provided excessive access. The focus is on delivering permissions that are granular and tied directly

to the action being performed. The main development here is the TT draft [Tulshibagwale et al. 2025a]. These tokens encode specific, rich authorization details (often requested via a RA), ensuring that the resulting token grants permission only for that exact action (e.g., a specific monetary transfer), making the authorization process inherently more secure and easily auditable than traditional generic scopes.

6.2. Gaps and Opportunities

While the WG is active on multiple fronts, important security and architectural gaps remain open issues for future work, often related to cross-protocol dependencies and refinement of foundational mechanisms. These gaps were frequently highlighted during the IETF 124 discussions (November 2025). They are:

- Authorization Code Invalidation and Downgrade Attacks: The primary security gap identified in Browser-Swapping Attack analysis is that PKCE alone fails to prevent code theft after suspected CSRF injection [Primbs 2025]. This is exacerbated by the lack of a mechanism for a client to actively notify the AS to invalidate codes suspected of CSRF attacks after initial issuance. Furthermore, the inability to enforce the correct `response_mode` makes the protocol vulnerable to downgrade attacks (e.g., to `response_mode=query`), which can lead to code leakage via logs and referrer headers [Primbs 2025];
- PoP Binding and Token Complexity: Achieving end-to-end security binding, especially in complex delegated flows, presents architectural difficulties. For instance, the ID-JAG draft [Parecki et al. 2025d] requires a clearer definition of PoP binding that spans the entire chain from initial SSO to the final Access Token [Parecki et al. 2025c]. Additionally, while introducing separate DPoP bindings for access and refresh tokens could mitigate large-scale compromise, it raises concerns about implementation complexity. Furthermore, for TTs [Tulshibagwale et al. 2025a], open issues remain regarding the need to define formal mechanisms for extending the lifetime or modifying the context (`tctx`) of the token [Tulshibagwale et al. 2025b];
- Metadata Trust and SSRF Mitigation: The mechanism defined in the Client ID Metadata Document draft [Parecki and Smith 2025b] introduces new security risks related to fetching external data, primarily the need for robust mitigation against SSRF vulnerabilities during metadata fetching [Parecki and Smith 2025a]. Mitigation strategies include prohibiting redirects and checking DNS resolution against private IP addresses. Furthermore, the WG needs to provide explicit guidance on incrementally establishing client ID reputation and mandating that all URI properties within the metadata (e.g., `logo_uri`) be restricted to HTTPS URLs [Parecki and Smith 2025a];
- Architectural Scalability and Deployment Challenges: Issues related to the broad deployment and migration of OAuth standards in enterprise environments remain open, particularly concerning architectural scalability. The primary challenge is supporting Multi-tenancy and Multi-instance RS AS, which requires explicit clarification on how user and client IDs are mapped across these disparate systems [Parecki et al. 2025c]. A key, unresolved architectural migration hurdle is how an AS can support both OAuth 2.0 and 2.1 clients simultaneously with-

out breaking older behavior [Parecki et al. 2025b]. Moreover, the ambiguity surrounding whether the `client_id` should represent a family (class) or an individual instance requires immediate clarification to stabilize dependent standards [Looker et al. 2025b];

- Client Authentication Scheme Definition and Compliance: The specification faces difficulty in clearly defining how an AS should signal supported authentication methods on an `invalid_client` error response (HTTP 401). This ambiguity stems from the lack of defined challenge schemes for newer authentication methods, such as `private_key_jwt`, which leaves the compliance path unclear for implementers.
- Bridging JSON Object Signing and Encryption (JOSE) and CBOR Object Signing and Encryption (COSE) for IoT: The security standards defined by the OAuth WG are adapted by the ACE WG to be viable in resource-limited IoT contexts. This adaptation is fundamentally cryptographic, as the ACE WG utilizes the COSE[4] (Concise Binary Object Representation (CBOR) Object Signing and Encryption) format — significantly more efficient in size and processing than JOSE[5] (used by the OAuth WG) — to implement the same security concepts. This technical boundary reveals an opportunity: the OAuth WG must ensure its advanced security concepts (like DPoP and Attestation) are designed with portability in mind, actively collaborating with the ACE WG to bridge the JOSE/COSE gap.

7. Conclusion

Analysis of the active IETF OAuth WG drafts indicates that OAuth 2.0's future rests on three pillars: Mandatory Security (via OAuth 2.1 and Attestation), Privacy by Design (using SD-JWT for selective disclosure), and Contextual Authorization (through TT for granular permissions). However, IETF 124 discussions highlighted critical gaps, including the lack of a mechanism for proactive authorization code invalidation to mitigate browser-swapping attacks, risks of SSRF in metadata fetching, and significant multi-tenancy scalability challenges. In essence, the protocol is evolving from a simple framework to a complex, context-aware security layer, and resolving these gaps is decisive for its ability to protect increasingly distributed and privacy-sensitive digital ecosystems.

References

Fett, D., Campbell, B., Bradley, J., Lodderstedt, T., Jones, M. B., and Waite, D. (2023). OAuth 2.0 Demonstrating Proof of Possession (DPoP). RFC 9449.

Fett, D., Yasuda, K., and Campbell, B. (2025). Selective Disclosure for JWTs (SD-JWT). Work in Progress.

Hardt, D. (2012). The OAuth 2.0 Authorization Framework. RFC 6749.

Hardt, D., Parecki, A., and Lodderstedt, T. (2025). The OAuth 2.1 Authorization Framework. Work in Progress.

Jones, M. B., Campbell, B., Mortimore, C., and Skokan, F. (2025). Updates to OAuth 2.0 JSON Web Token (JWT) Client Authentication and Assertion-Based Authorization Grants. Work in Progress.

[4]https://datatracker.ietf.org/wg/cose/about/
[5]https://datatracker.ietf.org/wg/jose/about/

Jones, M. B. and Hardt, D. (2012). The OAuth 2.0 Authorization Framework: Bearer Token Usage. RFC 6750.

Kasselman, P., Fett, D., and Skokan, F. (2025). Cross-Device Flows: Security Best Current Practice. Work in Progress.

Looker, T., Bastian, P., and Bormann, C. (2025a). OAuth 2.0 Attestation-Based Client Authentication. Work in Progress.

Looker, T., Bastian, P., and Bormann, C. (2025b). OAuth 2.0 Attestation-Based Client Authentication (IETF 124 OAuth Session). Presentation Slides. Presented at IETF 124, Montreal.

Looker, T., Bastian, P., and Bormann, C. (2025c). Token Status List (TSL). Work in Progress.

Parecki, A., Fletcher, G., and Kasselman, P. (2025a). OAuth 2.0 for First-Party Applications. Work in Progress.

Parecki, A., Hardt, D., and Lodderstedt, T. (2025b). OAuth 2.1 (IETF 124 OAuth Session). Presentation Slides. Presented at IETF 124, Montreal.

Parecki, A., McGuinness, K., and Campbell, B. (2025c). Identity Assertion Authorization Grant (IETF 124 OAuth Session). Presentation Slides. Presented at IETF 124, Montreal.

Parecki, A., McGuinness, K., and Campbell, B. (2025d). Identity Assertion JWT Authorization Grant. Work in Progress.

Parecki, A., Ryck, P. D., and Waite, D. (2025e). OAuth 2.0 for Browser-Based Applications. Work in Progress.

Parecki, A. and Smith, E. (2025a). Client ID Metadata Document (IETF 124 OAuth Session). Presentation Slides. Presented at IETF 124, Montreal.

Parecki, A. and Smith, E. (2025b). OAuth Client ID Metadata Document. Work in Progress.

Primbs, J. (2025). Browser-Swapping Attacks (IETF 124 OAuth Session). Presentation Slides. Presented at IETF 124, Montreal.

Sakimura, N., Bradley, J., and Agarwal, N. (2015). Proof Key for Code Exchange by OAuth Public Clients. RFC 7636.

Schwenkschuster, A., Kasselman, P., Burgin, K., Jenkins, M. J., and Campbell, B. (2025). OAuth Identity and Authorization Chaining Across Domains. Work in Progress.

Sheffer, Y., Hardt, D., and Jones, M. B. (2025). JSON Web Token Best Current Practices. Work in Progress.

Terbu, O., Fett, D., and Campbell, B. (2025). SD-JWT-based Verifiable Credentials (SD-JWT VC). Work in Progress.

Tulshibagwale, A., Fletcher, G., and Kasselman, P. (2025a). Transaction Tokens. Work in Progress.

Tulshibagwale, A., Fletcher, G., and Kasselman, P. (2025b). Transaction Tokens (IETF 124 OAuth Session). Presentation Slides. Presented at IETF 124, Montreal.