Asymmetric Key Token Type for OAuth 2.0

Proof-of-Possession Tokens using Asymmetric Keys

Background

- OAuth 2.0 initially standardized only bearer tokens
- MAC tokens are proposed in draft-ietf-oauth-v2-http-mac to expand token types to include symmetric Proof-of-Possession (PoP) tokens
- To avoid canonicalization issues draft-tschofenig-oauth-hotk moved this to JWT and proposed an asymmetric key scheme based on TLS.
- MAC tokens:
  - Can be more efficient than asymmetric cryptography for the PoP
  - However, they require encryption to distribute the MAC key, which in turn requires pre-sharing of encryption keys between authorization server and resource server

Asymmetric Key Token Design Goals

- Proof-of-Possession token type that:
  - Does not require encryption and associated key management
  - Can be used as a refresh token to require PoP when obtaining short-lived bearer or MAC access tokens
  - Otherwise, mostly similar to the MAC token
- Asymmetric computation will be less efficient than MAC computation, however:
  - This may be acceptable for infrequent operations
  - Asymmetric computations can be made less frequent by using this as a refresh token

Asymmetric Key Token Type

- Key Distribution
  - Client provides signing public key to server in the authorization request
  - To client: Server includes two parameters with the access token:
    - pub_key: Signing public key for token
    - kid: Key ID
  - To resource server: Server includes above parameters in the access token
- Authenticator
  - Input to signature computation is similar to the MAC token
    - However, canonicalizing the signature input was a major issue with OAuth 1.0
    - This is avoided by using a JWT, rather than signing HTTP headers and body parts
    - The attribute value is a JWT
  - Verification is simply JWT signature validation

Asymmetric Key Token Type

- Proposed algorithms
  - ECDSA P256 SHA256
  - RSASSA-PKCS1-V1_5 SHA-256
- Security considerations
  - The authorization server does not do key generation, which would require entropy
  - Authorization server must ensure the authenticity of pub_key during request
  - Client may wish to use different pub_key for different resource servers (and also for different sessions to same resource server) for privacy reasons
- Next steps – update draft-tschofenig-oauth-hotk
  - Generic solution approach