HTTP Message Signatures and OAuth Proof of Possession (PoP)

Justin Richer
IETF OAuth WG Interim
May 10, 2021
A brief history

• OAuth 1.0
• OAuth 2.0 “MAC Token”
• OAuth 2.0 “Proof of Possession Architecture”
• OAuth 2.0 “Signed HTTP Requests”
Existing HTTP signing drafts weren’t useful
- No WG support
- Limited coverage
- Limited flexibility

Let’s make our own
- How hard could it be?
How the old draft worked

1. Mash together covered headers, hash it
2. Mash together covered query, hash it
3. Hash the body
4. Add a few extra variables (tokens, timestamp, method, URI, …)
5. Wrap everything up in a JWS and send it
draft-ietf-httpbis-message-signatures

• Official document of HTTP WG
• Builds on several community drafts
• HTTP-native
• Built to be profiled and flexible
How HTTP Message Signing works

1. Choose covered content and crypto parameters
2. Normalize the HTTP message components
3. Generate a signature input string
4. Sign the string creating a signature output
5. Add the signature output and parameters as structured HTTP headers
Example HTTP Message

POST /foo?param=value&pet=dog HTTP/1.1
Host: example.com
Date: Tue, 20 Apr 2021 02:07:55 GMT
Content-Type: application/json
Content-Length: 18

{"hello": "world"}
POST /foo?param=value&pet=dog HTTP/1.1
Host: example.com
Date: Tue, 20 Apr 2021 02:07:55 GMT
Content-Type: application/json
Content-Length: 18

{"hello": "world"}
Signature Base

@request-target": post /foo?param=value&pet=dog
"host": example.com
"date": Tue, 20 Apr 2021 02:07:55 GMT
"content-type": application/json
"@signature-params": ("@request-target" "host" "date"
   "content-type");created=1618884475;keyid="test-key-rsa-pss"
Signature Bytes

NtIKWuXjr4SBEXml97gbick4095ff378I0CZ0a2VnIeEXZ1itzAdqTpsvG91XYrq5CfxCmk8zz1Zg7ZGYD+ngJyVn805r73rh2eFCPO+ZXDS45Is/Ex8srzGC9sfVZfqeEfApRFFe5yXDmANVUwzFWCEmGM6+SJVmWl1/jyEn45qA6Hw+ZDHbrbp6qvD4N0S92jLPyVVEh/SmCwnkeNiBgnbt+E0K5wCFNHPbo4X1Tj406W+bTtnKzaoKxBWKW8aIQ7rg92zqE1oqBRjqtRi5/Q6P5ZYGGINKzNyV3UjZtxeZNnNJ+MANWS0mofFqcZHVgSU/1wUzP7Mhz0KLca1Yg==
Signed Request

POST /foo?param=value&pet=dog HTTP/1.1
Host: example.com
Date: Tue, 20 Apr 2021 02:07:55 GMT
Content-Type: application/json
Content-Length: 18

Signature-Input: sig1=\"host\" "date\" "content-type\");created=1618884475;keyid="test-key-rsa-pss"
Signature:
sig1=:NtIKWuXjr45BEXj97gbick4095ff378I0CZOa2VnIeEXZ1itzAdqTpsvG91XYrq5CfxCmk8zz1Zg7ZGYD+ngJyVn805r73rh2eFCPO+ZXD5Is/Ex8srzGC9sfVZfqeEfApFfe5yXDmANVUwzFWCEnGM6+SJVmWl1/jyEn45qA6Hw+ZDHbrbp6qvD4N0S92j1PyVVEh/SmCwnkeNiBgnbt+E0K5wCFNHpbo4X1Tj406W+bTtnKzaoKkxkwKw8a1Q7tg92zqE1oqBRjqtRi5/Q6P5ZYGGINKzNyV3UjZtxeZNnNJ+MANW5mofFqcZHvSU/1wUzP7MhzOKLca1Yg==:

{"hello": "world"}
How to apply it to OAuth 2.0

• New token type: PoP
• Requirement to present Signature, Signature-Input, and Authorization headers together to the RS
• Minimum request coverage requirements
• Key and algorithm determined by client context
  – Pre-registered, generated, negotiated, other?
  – Communicated in JWT or introspection to RS
Signed Request

POST /foo?param=value&pet=dog HTTP/1.1
Host: example.com
Date: Tue, 20 Apr 2021 02:07:55 GMT
Content-Type: application/json
Content-Length: 18
Authorization: PoP 3ZM-B0XGPQTR31UOH6XKG.WEM1N3G98L
Signature-Input: sig1="host" "date"
  "content-type" "authorization";created=1618884475;keyid="test-key-rsa-pss"
Signature: sig1=NTIKwuXjr4SBEXj97gbick4095ff378I0CZ0a2VnIeEXZ1itzAdqTpsvG91XYrq5CfxCmk8zz
1Zg7ZGYD+ngJyVn805r73rh2eFCPO+ZXDs45Is/Ex8srzGC9sfVZfqcEfApRFFe5yXDmANVUwzFWCEn
GM6+SVJmWl1/jyEn45qA6Hw+ZDHbrbp6qvD4N0S92j1PyVVEh/SmCwnkeNiBgnbt+E0K5wCFNHPbo4X
1Tj406W+bTtnKzaoKxBWkW8aIQ7rg92zqElqqBRjqtRi5/Q6P5ZYYGGINKzNyV3UjZtxeZNnNJ+MAnW
S0mofFqcZHvGSU/1wUzP7Mhz0KLca1Yg==:

{"hello": "world"}
Missing Features

• Can’t partially-sign query parameters
  – Could be added to HTTP Message Signatures as a new specialty type

• Doesn’t directly protect content body
  – Relies on HTTP Digest header
What about DPoP?

• Let them co-exist!
• Two different flavors
  – DPoP: SPA, minimalism
  – DPoP: dynamic asymmetric keys made by client
  – PoP: all clients, flexibility
  – PoP: various forms of key distribution and types
Why not just use JOSE?

- Facilitate interaction with non-JOSE systems
- JOSE excels at self-contained crypto
  - Need to either duplicate components or wrap whole message inside JOSE object
- HTTP Message Signatures already supports JWA for algorithm resolution
- HTTP Message Signatures supports advanced use cases like multiple chained signatures
Next Steps

• Update the draft-ietf-oauth-signed-http-request spec to use draft-ietf-httpbis-message-signatures

• Open questions
  – Registration of keys (static, dynamic, transactional)
  – Key distribution (per-token, AS-generated)
  – Algorithm/feature discovery
  – Client authentication with message signatures