Sergey Beryozkin, Talend

Apache CXF

Practical JOSE with Apache CXF

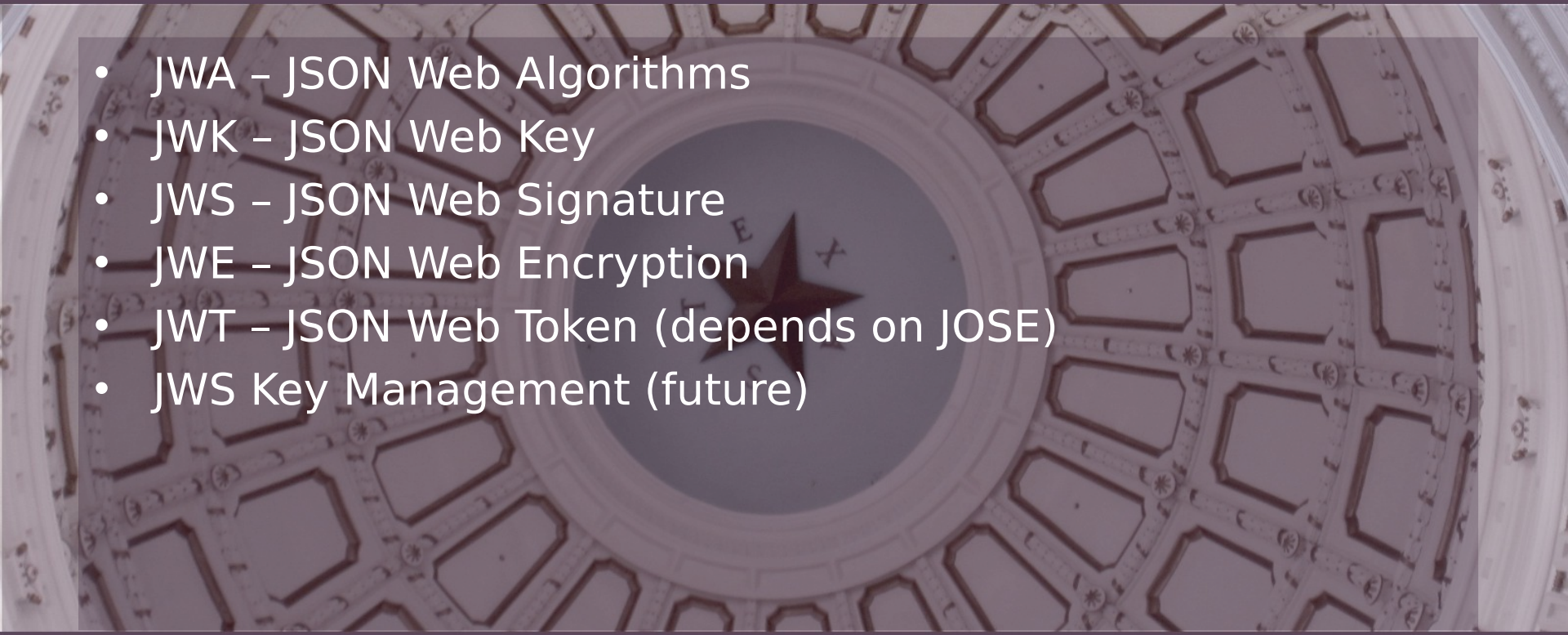
What Is Apache CXF

- Production quality JAXRS and JAXWS services framework
- Popular with small and large users (customers) alike
- Used by top Apache projects such as TomEE and Tika
- Runs in OSGi and standalone servlet containers
- JAX-WS 2.2, JAX-RS 2.0, JAX-RS 2.1 to be supported in time
- Major focus on supporting secure HTTP services
- WS-Security, advanced HTTPS, OAuth1 and OAuth2, SAML (Web SSO, Claim-based AC), and now – JOSE !
- Initial OpenIdConnect RP and IDP utility support

What Is JOSE

- JSON (JavaScript) Object Signing and Encryption
- Example of a productive cooperation between industry and community cryptography experts
- Essential for advanced OAuth2 applications
- Works well in regular HTTP client server communications
- JSON is only for describing the details of a cryptographic operation (algorithm, etc)
- Arbitrary formats for the secured payloads (plain text, JSON, binary data, even XML if needed)
- Compactness of JOSE representations is a priority

JOSE Building Blocks

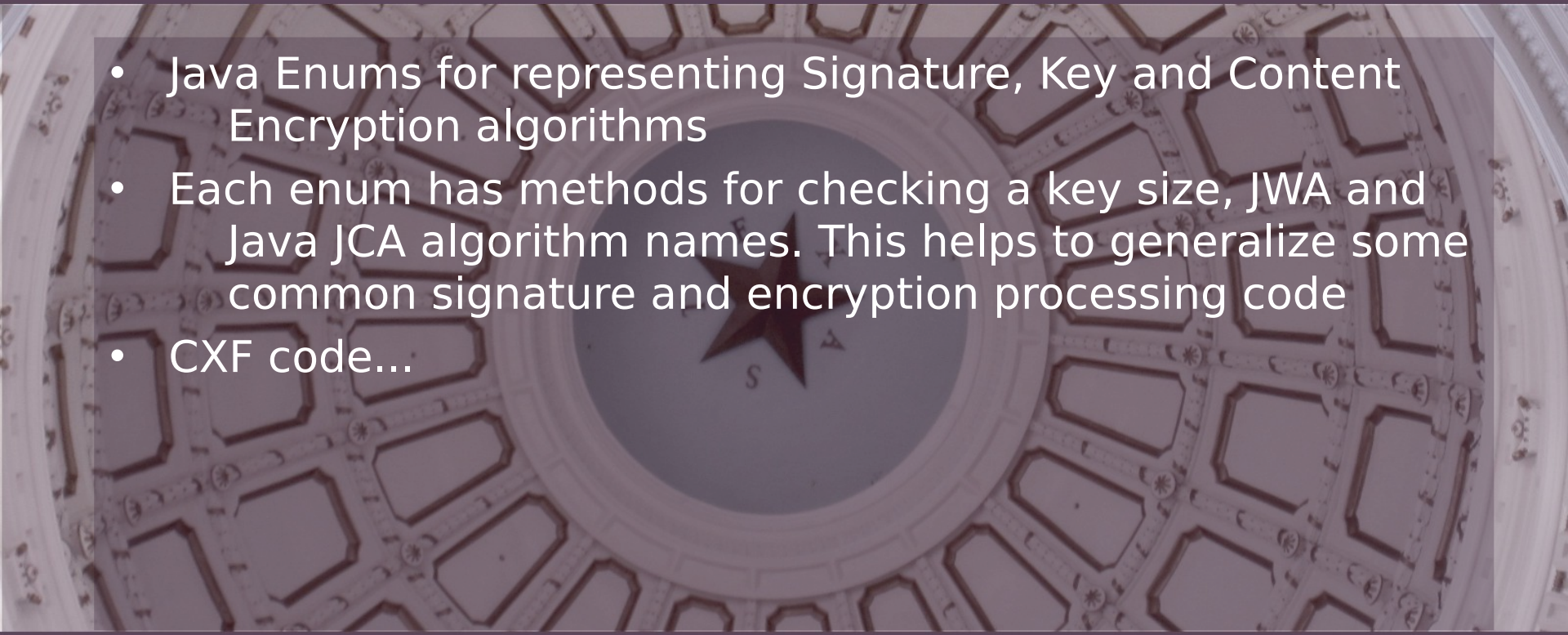


- JWA – JSON Web Algorithms
- JWK – JSON Web Key
- JWS – JSON Web Signature
- JWE – JSON Web Encryption
- JWT – JSON Web Token (depends on JOSE)
- JWS Key Management (future)

JWA Overview

- References all JOSE algorithms: signature algorithms, content and key encryption algorithms
- Describes how some of JOSE algorithms work in cases where JCA (or BouncyCastle, etc) does not offer a 1 to 1 support, example, AES-CBC-HMAC-SHA2
- Algorithm name is a type + hint: HS256 (HMac with SHA-256), RSA-OAEP-256 (RSA OAEP key encryption with SHA-256, etc)
- Offers security considerations common to all or specific to some of algorithms

JWA in CXF



- Java Enums for representing Signature, Key and Content Encryption algorithms
- Each enum has methods for checking a key size, JWA and Java JCA algorithm names. This helps to generalize some common signature and encryption processing code
- CXF code...

JWK Overview

- JSON Object for representing a cryptographic key, ex:
  {"kty":"oct",
  “kid”:”AesKeyWrapKey”,
  "alg":"A128KW",
  "k":"GawgguFyGrWKav7AX4VKUg"}
- Keys for all of JOSE algorithms can be in JWK format
- JWK is light-weight and easy to process
- JWK can describe X509 chains if needed
- JWK 'kid' is a useful property to indicate a key rotation

JWK in CXF

- Support for representing a single JWK key or JWK key sets
- Reading keys from InputStream/URI, writing to OutputStream
- Conversion from JWK to Java JCA RSA or EC Public/Private keys or SecretKey and vice versa
- Getting a JWK key from a key set by its kid, use, type, etc
- JWK key and key sets can be JWE-encrypted (PBES2 password-based algorithm is default, accessed at runtime with a password callback)
- CXF code...

JWS Overview

- Arbitrary payload (JSON, etc) is Base64URL-encoded
- Metadata (signature algorithm, etc) are in JOSE headers (JSON object) and Base64URL-encoded too
- Metadata + “.” + Payload is passed to a JWS signature function and is signed with HMac key or RSA or EC private key, signature is Base64URL-encoded
- Compact JWS: Metadata + “.” + Payload + “.” + Signature
- JSON JWS: JSON Object with one or more signatures
- JWS Payload can be detached
- JWS sequence (it is just a string) can be JWE-encrypted

JWS Example

- Input: Headers: {“alg”:”HS256”}, Data: “Hello”
- Compact JWS:
  eyJhbGciOiJIUzI1NiJ9.SGVsbG8.urVE_lxKKKtaqV4mFxuKWty
  S4fMGs34edqwDxyh50mo
- JSON JWS: {
  “payload”:”SGVsbG8”, “signatures”: [
- {“protected”:”eyJhbGciOiJIUzI1NiJ9”,
  “signature”:”urVE_lxKKKtaqV4mFxuKWtyS4fMGs34edqw
  Dxyh50mo”}
  ]}

JWS in CXF

- JWSSignatureProvider supports creating signatures
- JWSSignatureVerifier supports validating signatures
- Providers and verifiers for all JWS JWA algorithms
- JWS Producer and JWS Consumer help with creating and analyzing JWS Compact and JSON sequences
- JAX-RS JWS filters can stream while signing
- Support for creating Providers and Verifiers from JWKs and JCA RSA/EC/HMac keys
- Single Verifier instance supports a single algorithm only
- CXF Code...

JWE Overview

- All JWE content algorithms create authentication tags
- Content encryption keys (CEKs), IVs are usually generated
- CEKs are encrypted/wrapped
- Direct encryption is possible (CEK is known to both parties)
- Compact JWE: Metadata + “.” + Encrypted CEK + “.” + IV + “.” + CipherText + “.” + Authentication Tag
- JSON JWE: JSON Object with CEK encrypted by one or more algorithms - support for multiple recipients
- Metadata is integrity protected as additional authentication data

JWE Example

- {“enc”:”A128GCM”,”alg”:”RSA-OAEP”}, Data: “Hi”
- Compact JWE (headers + CEK + IV + Cipher + Tag):
  EyJhbGciOiJSU0EtT0FFUCIsImVuYyI6IkExMjhHQ00ifQ
  .RceDjhyuL6...lm_w
  .48V1_ALb6US04U3b.R4U.19ePGJBOpy7ZwTK63LxFtw
- JSON JWE:
  {“protected”:”EyJhbG...ifQ”,
  “recipients”:[{“encrypted_key”:”RceDjhyuL6...lm_w”}],
  “iv”:”48V1_ALb6US04U3b”, “ciphertext”:”R4U”,
  “tag”:”19ePGJBOpy7ZwTK63LxFtw”}

JWE In CXF

- JWEEncryptionProvider produces JWE encryptions with KeyEncryptionProvider and ContentEncryptionProvider
- JWEDecryptionProvider decrypts JWE
- All of JWA JWE algorithms are supported
- Jwe Producer and Jwe Consumer help with creating and processing JWE compact and JSON sequences
- JAX-RS JWE filters can stream while encrypting
- Support for creating Encryptors/Decryptors from JWKs or JCA RSA/EC/Secret keys
- CXF code...

JWT Overview

- JWT is simply a JSON object for holding standard or custom claims. SAML Assertion is an XML alternative.
- Not part of JOSE but uses it to get signed and/or encrypted
- Used most often in OAuth2: as internal access token representation or (assertion) grant, id_token in OIDC, etc
- Might be used as a standard JSON wrapper in non OAuth2 services or as JWT HTTP Authorization scheme (CXF)
- Example of claims: {"iss":"joe","exp":1300819380}
- The above JSON text is JWS signed and/or JWE encrypted

JWT in CXF

- JwtToken and JwtClaims helper beans
- JwsJwtCompactProducer and Consumer for JWS signing
- JweJwtCompactProducer and Consumer for immediate JWE encryption (skipping the signature process)
- 'JWT' HTTP Authorization scheme where a signed and/or encrypted JWT is linked to a signed and/or encrypted HTTP payload
- CXF Code...

CXF JAXRS JOSE Filters

- JAX-RS filters support a case where client and server work with plain Java beans but the data which goes on the wire is JWS-signed and/or JWE-encrypted
- The data secured by filters can be linked to an authenticated user with a JWT authorization scheme
- JWS and JWE Writers and JWE Readers and JWS readers can be chained (sign-then-encrypt on the output, decrypt-then-verify on the input)
- JWS and JWE writers can do the best effort at streaming
- Filters supported by Java KeyStores or JWK stores

CXF JOSE Configuration

- Main configuration is about supporting JAX-RS JOSE filters with traditional Java Key Stores or JWK stores
- In most cases a filter reads a Java properties file, which points to either a Java Key Store or JWK store
- JWK store is usually a file where an array of JWK keys (JWK key set) is kept. The file can be JWE-encrypted
- Alternatively, a key set or individual JWK can be inlined directly inside the Properties file – in the JWE-encrypted form
- Many options for optimizing the configuration when possible: ex, Properties can specify an algorithm name but it is not needed if a JWK key has it too, etc, etc

JOSE and OAuth2

- At the moment JOSE is primarily utilized in the OAuth2 world, though using JOSE in a non-OAuth2 world will inevitably become more wide-spread over time.
- JWT may represent an access token or JWT Bearer grant and signed and/or encrypted
- JWT can be used as part of a secured authorization code request
- JWT is a secured OIDC id_token, etc...
- JWKs are used in many places, example, for distributing OAuth2 PoP token secret keys, for validating OIDC id_token, etc, etc

What is next for JOSE

- Final optimizations to the specification texts
- Possible interoperability events
- Key Management for JWS (example, using HMAC to do JWS is effectively a direct key signature where both parties need to know a key in advance, similar to direct JWE encryption)
- COSE – optimized version of JOSE
- JSON Clear Signature (Anders Rundgren)

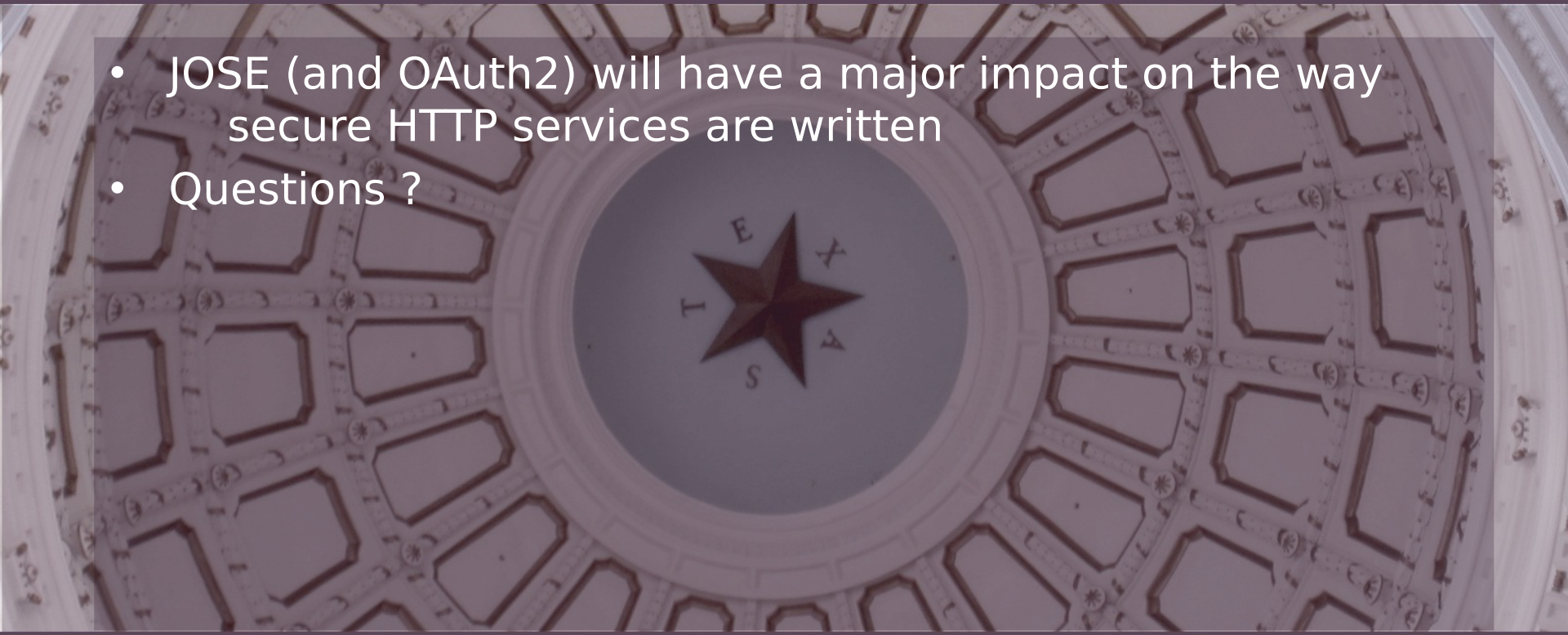
Demo

- Shows a WebCrypto (http://www.w3.org/TR/WebCryptoAPI/) Java Script client sending a JWS-signed payload to Apache CXF server (JWS interoperability)
- Original demo was created by Anders Rundgren, available at https://mobilepki.org/WCPPSignatureDemo/home
- Anders explained how to build a demo, one of original demo servlets was replaced by CXFServlet and CXF JAX-RS server with the CXF JOSE JwsCompactConsumer.
- WebCrypto demo client has not been modified
- The actual demo...

Alternatives to CXF JOSE

- Jose4J
- Apache Oltu
- RestEasy
- Spring Security
- Only a start...

Conclusion



- JOSE (and OAuth2) will have a major impact on the way secure HTTP services are written
- Questions ?


Thank You !
users@cxf.apache.org
sberyozkin.blogspot.com
APACHECON
NORTH AMERICA