

# JavaScript Message Syntax (JSMS)

`draft-rescorla-jsms-00`

IETF 80

Eric Rescorla\*

Joe Hildebrand

`ekr@rtfm.com`   `jhildebr@cisco.com`

\* Presenting

# Overview

- Lots of need for cryptographically protected (signed/encrypted) messages
  - XMPP, OAuth, RELOAD, ...
- Empirically implementors (and designers) don't want to use CMS
  - Fear of protocol complexity
  - ASN.1 allergy
  - Especially bad fit for JavaScript, which does badly with binary encodings
- Result is people avoid secure messaging entirely (XMPP, OAuth) or invent their own formats (RELOAD)
- We need a format people are actually willing to implement

## Current Efforts

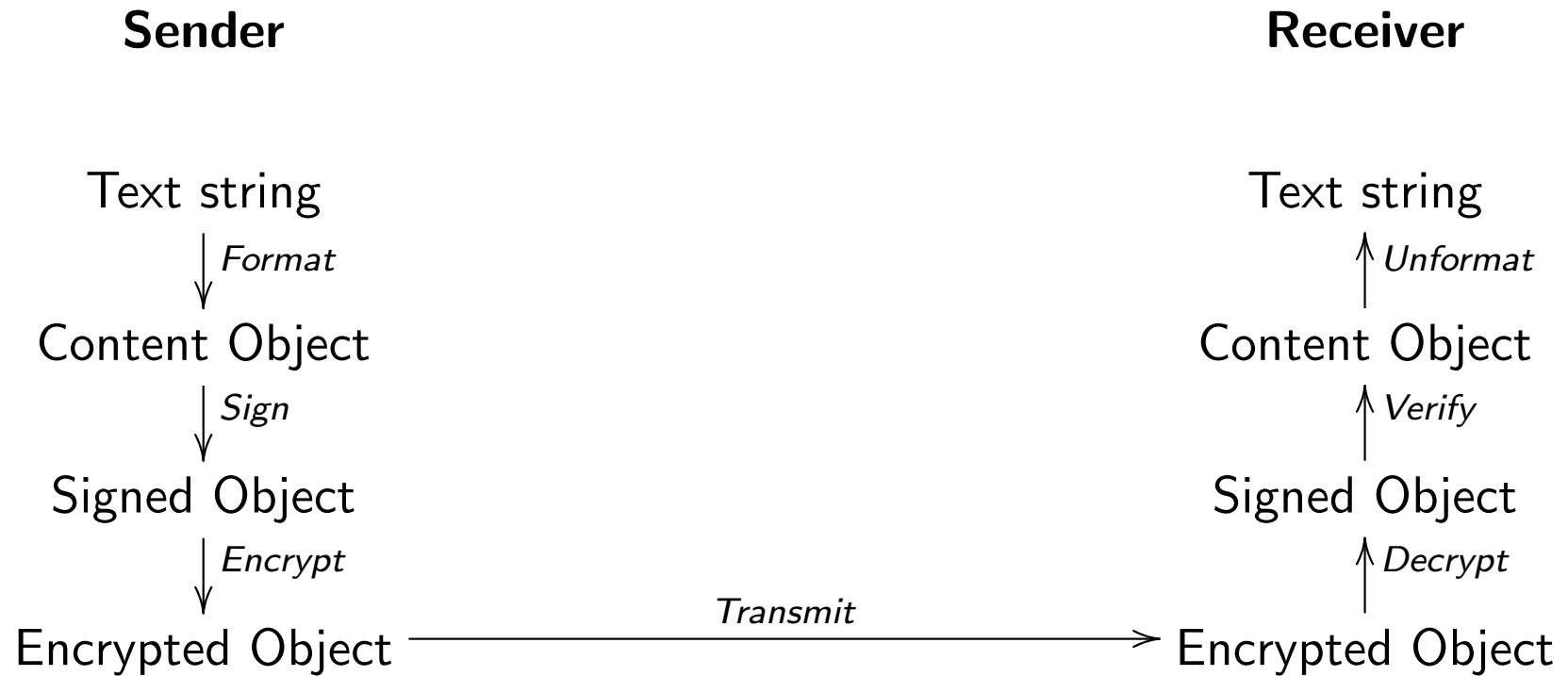
- WebToken (draft-jones-json-webtoken-01)
- JSMS (draft-rescorla-jsms-00, this talk)
- Web Object Encryption and Signing (WOES) bar BOF (tonight at 2000 in Karlin I)

# JSMS: The Basic Idea

- Use JSON encoding
  - Very convenient for working in JavaScript
  - JSON libraries are readily available for other languages
- Pick the simplest and most common use cases
  - Digital signature
  - Encryption under recipient's public key (+ MAC for integrity)
  - Encryption with a shared symmetric key (+ MAC for integrity)
- Design for maximum implementation simplicity
  - No canonicalization
  - Base-64 anything difficult to represent as a string
  - In-memory processing (no streaming operation)

\* WARNING: Hard hat area

# Sample Workflow



# Content Objects

```
{  
  "ContentType": "text/plain; charset=UTF-8",  
  "Type": "content",  
  "Data": "SGVsbG8sIFdvcmxkCg==",  
  "ID": "746a4c9f-8e84-4313-b669-81590ee2949e",  
  "Created": "2011-03-07T16:17Z"  
}
```

- Wrapper around whatever the original content was
- Content-type to identify the format
- Base64 to protect potentially dangerous characters
- Datestamp and ID for anti-replay

# Signed Objects

```
{  
  "SignedData": "ewogICAgIkNvb3RlbnRUeXB1IjoidGV4dC9wbGFpbjsY2hhcn  
    ... IKfQ==",  
  "DigestAlgorithm": "SHA-256",  
  "SignatureAlgorithm": "RSA-PKCS1-1.5",  
  "Signer": "xmpp:romeo@example.net",  
  "Signature": "sNsxJltUaz4pSzAtJiPZagUMV4SwWugWexGbffK/WJRD  
    ... SJfIdiAJNA+nEnk=",  
  "CertChain": {  
    "Type": "PKIX",  
    "Chain": [ ... ]  
  }  
}
```

- Signature computed over binary representation of Contents
  - Base64-encoded to prevent damage in transit
- Support for PKIX certificates\*

---

\*But wait, aren't certificates in ASN.1? More on this shortly

## Wait, aren't PKIX certs in ASN.1/DER?

- Answer 1: Do without
  - Can potentially use raw public keys (not supported yet)
- Answer 2: Certificates are easier to isolate
  - Stand up a Web service to verify/decode (natural in a Web 2.0 app)
  - ... remember that the JS probably came from the server anyway
- Answer 3: Replace
  - Natural to have the contents of a Signed object be a key/identity binding
  - Eventually expect to have a simple JSMS-based certificate format



# What's next?

Come to the WOES bar bof: Tonight at 2000 in Karlin I