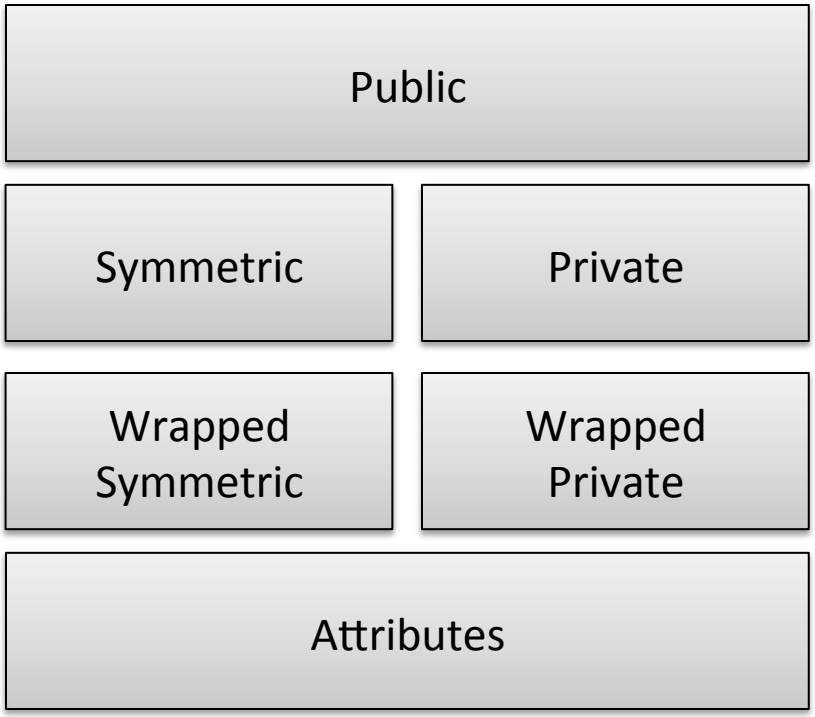
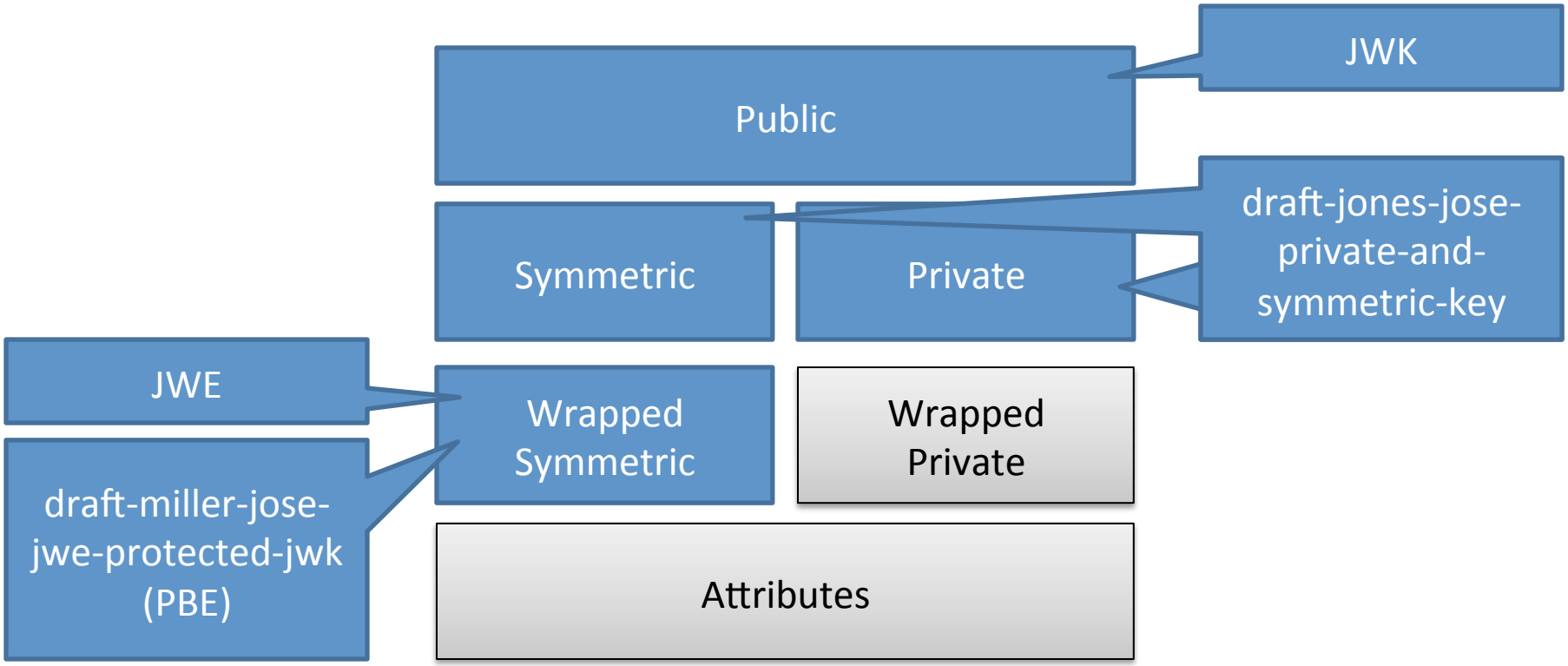
A Unified Theory of JOSE Keys

What do we want in a key format?



What do we already have?

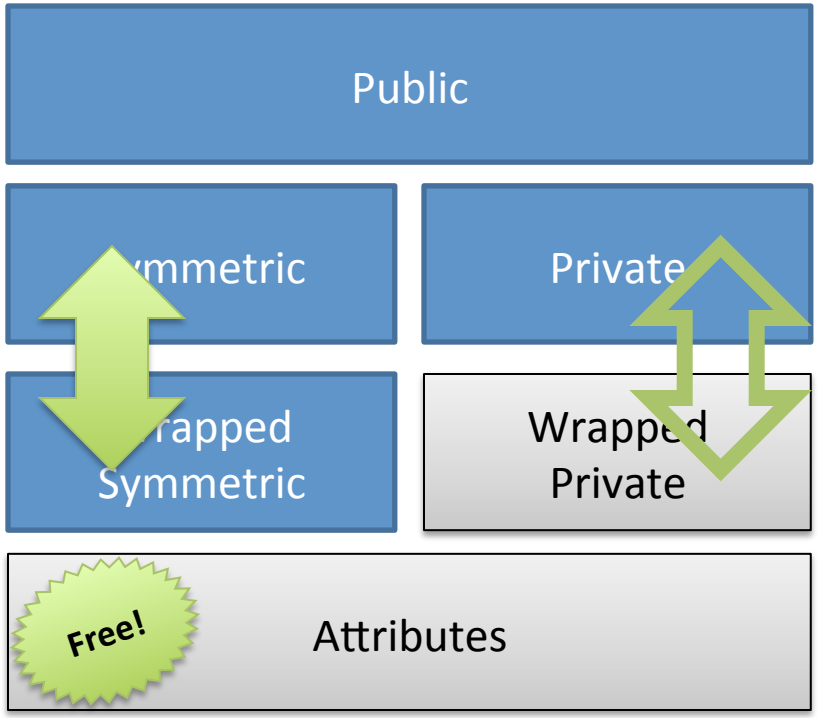


Goals

- Use key encryption algorithms compatible with validation processes (e.g., FIPS)
- Have one unified approach to wrapping keys
  - Produce JWE key wrapping as a special case (symmetric with no wrapped attributes)
- Minimize size overhead imposed by wrapping

What do we need to do?

Understand the wrapping system we already have…



… so that it can be applied to other kinds of JSON keys

One we can handle JSON keys, we get attributes for free!

The Wrapping Algorithm (symm)

```
// INPUT
{
  "k": "s5R6jlO4xuA"
}
```


```
// OUTPUT
{
  "alg": "A128KW",
  "kid": "outer",
  "wk": "5vJa77gb6lPI
         widHNP_cYw"
}
```

The Wrapping Algorithm (attr)

```
// INPUT
{
  "kid": "inner"
  "k": "s5R6jlO4xuA"
}
```

```
// OUTPUT
{
  "alg": "A128KW",
  "kid": "outer",
  "kat": {
    "kid": "inner",
  },
  "wk": "5vJa77gb6lPI
         widHNP_cYw"
}
```

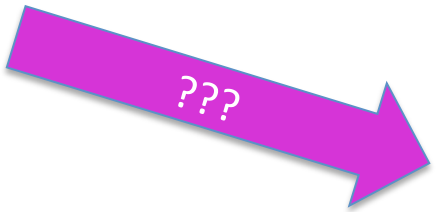
The Wrapping Algorithm (priv)

```
// INPUT
{
  "kid": "inner"
  "n": "YRBve...o3g",
  "e": "AQAB",
  "d": "mWHnC..._9s"
}
```

```
// OUTPUT
{
  "kty": "RSA",
  "alg": "A128KW",
  "kid": "outer",
  "kat": {
    "kid": "inner",
    "n":…, "e":…,
  }
  "wk": "5vJa7...cYw"
}
```

The Wrapping Algorithm (priv2)

```
// INPUT
{
  "kid": "inner"
  "n": "YRBve...o3g",
  "e": "AQAB",
  "d": "mWHnC..._9s",
  "p": "wHhqA...7J0",
  "q": "QZ81X...Qo8",
  "use": enc
}
```




```
// OUTPUT
{
  "kty": "RSA",
  "alg": "A128KW",
  "kid": "outer",
  "kat": {
    "kid":…,"n":…,
    "e":…
  }
  "wj": true,
  "wk": "5vJa7...cYw"
}
```

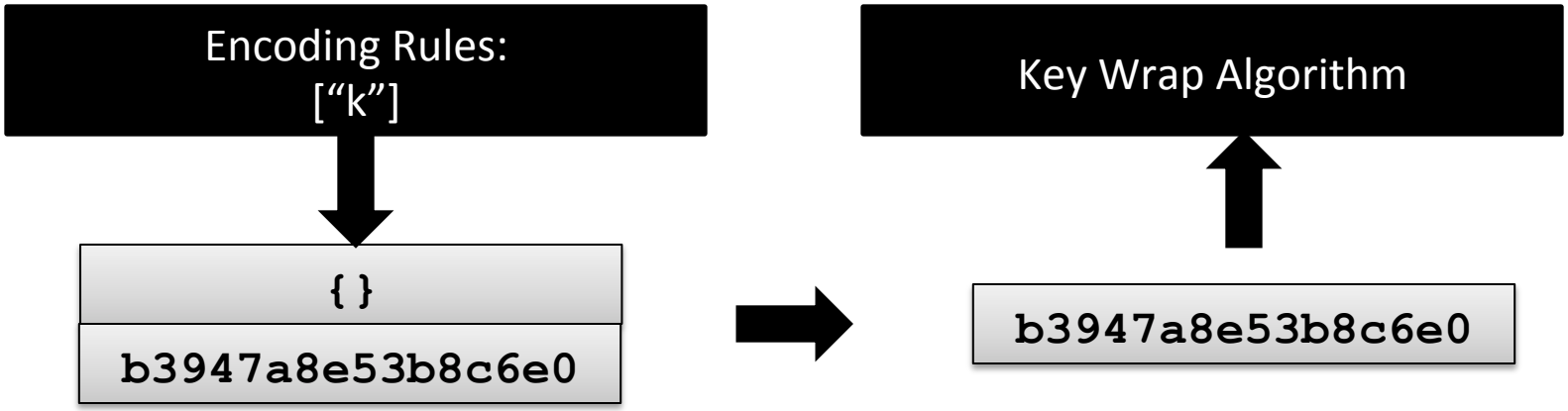
JSON with Efficient Binary (JEB)

- Encoding Rules: Order of binary fields
  - JWE = [“key”, “iv”, “ciphertext”, “mac”]
  - RSA = [“d”, “p”, “q”, “dp”, “dq”, “qi”]
- Encoding operation:
  - Input: Encoding rules, JSON object with binary fields
  - Remove fields in encoding rule from object
  - Concatenate: lenJSON || JSON || len1 || field1 || …
    - Don’t bother with length if there’s one field and no JSON
- Decoding operation:
  - Split JSON and binary fields
  - Add binary fields back to JSON object
- Example implementation: http://ipv.sx/ietf86/jeb.html

JEB Example (symm)

```
// INPUT
{
  "k": "s5R6jlO4xuA"
}
```

```
// OUTPUT
{
  "alg": "A128KW",
  "kid": "outer",
  "wk": "5vJa77gb6lPI
         widHNP_cYw"
}
```



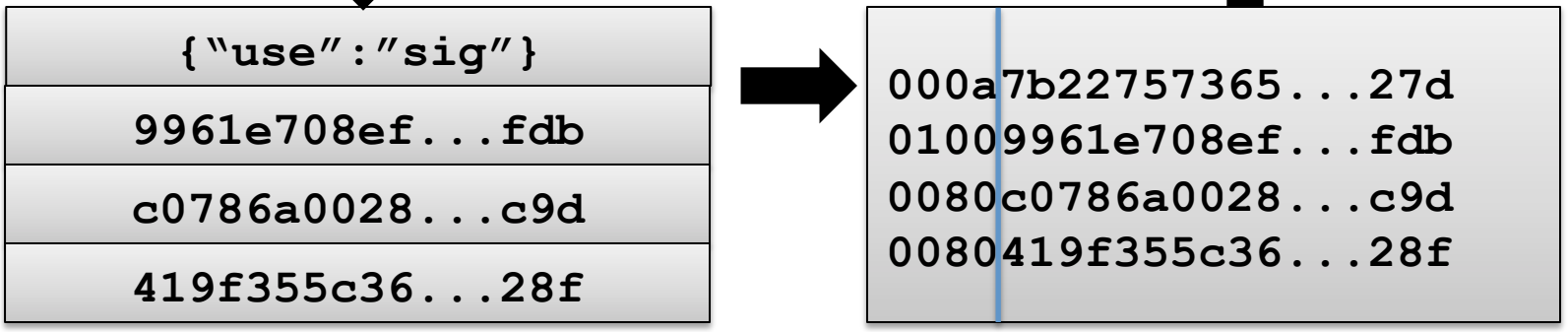
JEB Example (priv)

```
// INPUT
{
  "kid": "inner"
  "n": "YRBve...o3g",
  "e": "AQAB",
  "d": "mWHnC..._9s",
  "p": "wHhqA...7J0",
  "q": "QZ81X...Qo8",
  "use": "sig"
}
```

```
// OUTPUT
{
  "kty": "RSA",
  "alg": "A128KW",
  "kid": "outer",
  "kat": {
    "kid":…,"n":…,
    "e":…,"d":…
  }
  "wj": true,
  "wk": "5vJa7...cYw"
}
```

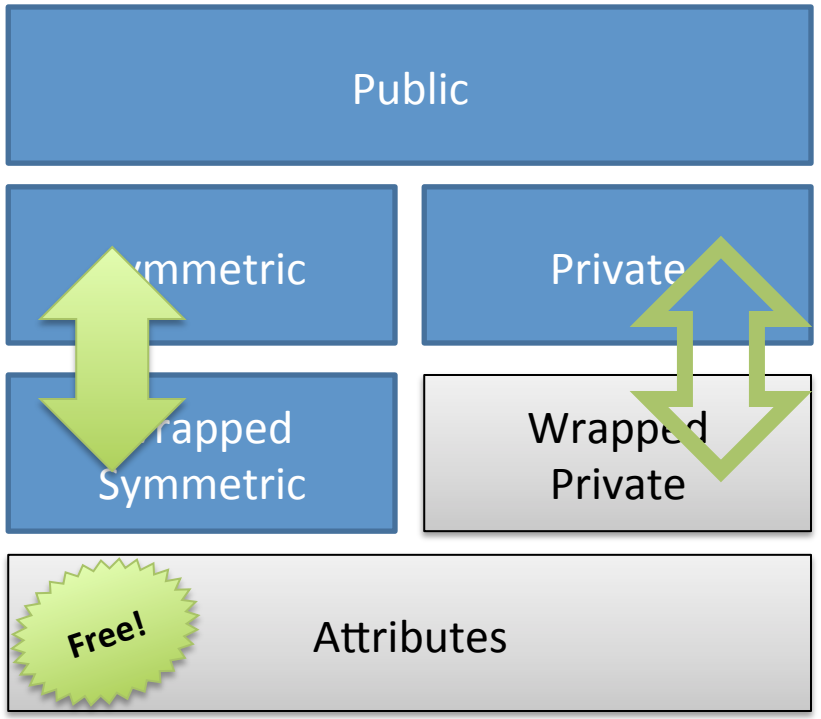
Encoding Rules:
["d", "p", "q", "dp", "dq", "qi"]

Key Wrap Algorithm



Recap

- Want a full-featured key representation
- Have several partial solutions already
- Need to define
  - General key wrapping algorithm
  - Efficient way to pack JSON with binary (?)



Questions

1. What framework should we allow for key protection?
   A. Keys == Data (JWK within JWE)
   B. Keys != Data (Key wrapping)
2. How should we encode the key to be protected?
   A. Simple JSON
   B. Efficient JSON