Key Encapsulation: A New Scheme for Public-Key Encryption

Most specifications of public-key encryption follow the original “encrypt/decrypt” model from 25 years ago.

New model is emerging, based on work of Shoup and others: *key encapsulation*, with better flexibility and security proofs.

Recommend transition to new model:
- Introduction of key encapsulation into XML Encryption v1.1.
Original Approach

- Bob has public key / private key pair
- Alice encrypts message $M$ with Bob’s public key to produce a ciphertext $C$:
  \[ C = E( \text{PubKey}_B, M ) \]
- Bob decrypts $C$ with his private key:
  \[ M = D( \text{PrivKey}_B, C ) \]
Limitations

- **Message length**: Length of $M$ may be limited
- **Malleability**: Encryption may not protect message integrity
- **Mathematical properties**: Encryption of related messages may be related
- **Modeling**: DH (ECDH) doesn’t fit well
Typically, some message padding is applied to address these limitations, but current approaches for RSA are less than ideal:

- PKCS #1 v1.5 padding is *ad hoc*, doesn’t provide integrity
- OAEP provides integrity and is provably secure, but bounds aren’t tight (e.g. knowledge of plaintext in RSA-OAEP reveals input to RSAEP; this is not the case with RSA-KEM)

Message length is still bounded, and DH needs its own method.
New Remedy: Two Layers

- *Public-key layer* establishes a random symmetric key

- *Symmetric-key layer* protects data with the established symmetric key and symmetric algorithm
  - data can be of any length

- Layers are independent
Addressing the Limitations

- **Modeling**: DH, RSA, other PKC all fit
- **Message length**: Length of $M$ not limited
- **Malleability**: Symmetric method can provide integrity protection
- **Mathematical properties**: Symmetric keys are unrelated; symmetric method avoids mathematical properties
Don’t We Do This Already?

Many specifications (including S/MIME) have two layers:
- message encrypted with symmetric key
- symmetric key encrypted with RSA public key

But the symmetric key is generated first *then* encrypted; more than needed, and results in a looser (or no) proof of security.
Encryption: Alice generates a symmetric key $W$ and a ciphertext $C$ that “encapsulates” $W$:

$$(C, W) = E(PubKey_B)$$

Decryption: Bob regenerates $W$ from $C$:

$$W = D(PrivKey_B, C)$$
Two Layers with Key Encapsulation

Symmetric-Key Layer

E → C → D

symmetric key

public key

private key
Encapsulation Using RSA

Encrypt with public key \((n, e)\):

- \( r \leftarrow_R [0, n-1] \)
- \( C_0 \leftarrow r^e \mod n \)
- \( W \leftarrow \text{KDF}(r) \)

Decrypt with private key \((n, d)\):

- \( r \leftarrow C_0^d \mod n \)
- \( W \leftarrow \text{KDF}(r) \)
Key Transport Using KEM

1. Generate a random integer $z$ ($0 \leq z \leq n-1$)
   
   $z = \text{RandomInteger}(0, n-1)$

2. Encrypt the random integer $z$ using the recipient's public key ($n, e$)
   
   $c = z^e \mod n$

3. Derive a key-encrypting key $KEK$ of length $kekLen$ bytes from $z$ using the underlying key derivation function
   
   $KEK = \text{KDF}(z, kekLen)$

4. Wrap the keying data $K$ with the key-encrypting key $KEK$ using the underlying key-wrapping scheme to obtain wrapped keying data $WK$
   
   $WK = \text{Wrap}(KEK, K)$

5. Concatenate the ciphertext $C$ and the wrapped keying data $WK$ to obtain the encrypted keying data $EK$
   
   $EK = C \ || \ WK$

6. Output the encrypted keying data $EK$
Key Transport in Two Layers
(similar for message encryption)

Symmetric-Key Key Transport

\[ K \rightarrow \text{Symmetric-Key Key Transport} \rightarrow K \]

\[ E \quad \text{symmetric key} \quad D \]

\[ \text{public key} \quad C \quad \text{private key} \]
## Key Encapsulation in Standards

<table>
<thead>
<tr>
<th>Standard</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSI X9F1 (X9.63, X9.44 draft)</td>
<td>✓</td>
</tr>
<tr>
<td>IEEE P1363 (P1363a draft, P1363b)</td>
<td>Proposed</td>
</tr>
<tr>
<td>ISO/IEC 18033-2 (draft)</td>
<td>✓</td>
</tr>
<tr>
<td>PKCS #11</td>
<td>Being proposed</td>
</tr>
<tr>
<td>XML Encryption</td>
<td>Proposed here…</td>
</tr>
<tr>
<td>S/MIME</td>
<td><em>In WG last-call</em></td>
</tr>
</tbody>
</table>
Conclusions & Proposal

- Key encapsulation is a convenient way of positioning public-key cryptography

- Specific suggestion for XMLSec: Include KEM as new key transport method in XMLEnc 2.0 (?)
  - RSA-KEM, ECDH-KEM

- Will entail: Defining schema for defining key encapsulation method (RSA-KEM, ECDH-KEM), key derivation function, key length and key wrapping scheme
Zheng-Seberry, Bellare-Rogaway proposed RSA-based schemes with two layers (early 1990s)

Shoup: KEM for ISO proposal (2001)


http://www.rsa.com/rsalabs/node.asp?id=3D2147

Jakob Jonsson's paper comparing security bounds of OAEP and KEM:
Key Agreement in Two Layers
(one key-pair case)

Symmetric-Key Key Agreement

$K \leftarrow \text{Symmetric-Key Key Agreement} \rightarrow K$

$E \quad \text{symmetric key} \quad D$

public key \quad C \quad private key
Encrypt with public key \((p, q, g, y)\):

- \(r \leftarrow_R [1, q-1]\)
- \(C_0 \leftarrow g^r \mod p\)
- \(Z \leftarrow y^r \mod p\)
- \(W \leftarrow \text{KDF}(C_0 || Z)\)

Decrypt with private key \((p, q, g, x)\):

- \(Z \leftarrow C_0^x \mod p\)
- \(W \leftarrow \text{KDF}(C_0 || Z)\)