1.1 Miscellaneous simple key derivation mechanisms

Table 1, Miscellaneous simple key derivation Mechanisms vs. Functions

<table>
<thead>
<tr>
<th>Mechanism</th>
<th>Encrypt &amp; Decrypt</th>
<th>Sign &amp; Verify</th>
<th>SR &amp; VR</th>
<th>Digest</th>
<th>Gen. Key / Key Pair</th>
<th>Wrap &amp; Unwrap</th>
<th>Derive</th>
</tr>
</thead>
<tbody>
<tr>
<td>CKM_CONCATENATE_BASE_AND_KEY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
</tr>
<tr>
<td>CKM_CONCATENATE_BASE_AND_DATA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
</tr>
<tr>
<td>CKM_CONCATENATE_DATA_AND_BASE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
</tr>
<tr>
<td>CKM_XOR_BASE_AND_DATA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
</tr>
<tr>
<td>CKM_EXTRACT_KEY_FROM_KEY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
</tr>
<tr>
<td>CKA_PUB_KEY_FROM_PRIV_KEY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
</tr>
</tbody>
</table>

1.1.1 Definitions

Mechanisms:

- CKM_CONCATENATE_BASE_AND_DATA
- CKM_CONCATENATE_DATA_AND_BASE
- CKM_XOR_BASE_AND_DATA
- CKM_EXTRACT_KEY_FROM_KEY
- CKM_CONCATENATE_BASE_AND_KEY
- CKA_PUB_KEY_FROM_PRIV_KEY

1.1.2 Parameters for miscellaneous simple key derivation mechanisms

- **CK_KEY_DERIVATION_STRING_DATA**;
  - **CK_KEY_DERIVATION_STRING_DATA_PTR**

CK_KEY_DERIVATION_STRING_DATA provides the parameters for the CKM_CONCATENATE_BASE_AND_DATA, CKM_CONCATENATE_DATA_AND_BASE, and CKM_XOR_BASE_AND_DATA mechanisms. It is defined as follows:

```c
typedef struct CK_KEY_DERIVATION_STRING_DATA {
    CK_BYTE_PTR pData;
    CK_ULONG ulLen;
} CK_KEY_DERIVATION_STRING_DATA;
```

The fields of the structure have the following meanings:

- `pData` pointer to the byte string
- `ulLen` length of the byte string

CK_KEY_DERIVATION_STRING_DATA_PTR is a pointer to a CK_KEY_DERIVATION_STRING_DATA.
CK_EXTRACT_PARAMS; CK_EXTRACT_PARAMS_PTR

CK_EXTRACT_PARAMS provides the parameter to the CKM_EXTRACT_KEY_FROM_KEY mechanism. It specifies which bit of the base key should be used as the first bit of the derived key. It is defined as follows:

```c
typedef CK_ULONG CK_EXTRACT_PARAMS;
```

CK_EXTRACT_PARAMS_PTR is a pointer to a CK_EXTRACT_PARAMS.

1.1.3 Concatenation of a base key and another key

This mechanism, denoted CKM_CONCATENATE_BASE_AND_KEY, derives a secret key from the concatenation of two existing secret keys. The two keys are specified by handles; the values of the keys specified are concatenated together in a buffer.

This mechanism takes a parameter, a CK_OBJECT_HANDLE. This handle produces the key value information which is appended to the end of the base key’s value information (the base key is the key whose handle is supplied as an argument to C_DeriveKey).

For example, if the value of the base key is 0x01234567, and the value of the other key is 0x89ABCDEF, then the value of the derived key will be taken from a buffer containing the string 0x0123456789ABCDEF.

- If no length or key type is provided in the template, then the key produced by this mechanism will be a generic secret key. Its length will be equal to the sum of the lengths of the values of the two original keys.
- If no key type is provided in the template, but a length is, then the key produced by this mechanism will be a generic secret key of the specified length.
- If no length is provided in the template, but a key type is, then that key type must have a well-defined length. If it does, then the key produced by this mechanism will be of the type specified in the template. If it doesn’t, an error will be returned.
- If both a key type and a length are provided in the template, the length must be compatible with that key type. The key produced by this mechanism will be of the specified type and length.

If a DES, DES2, DES3, or CDMF key is derived with this mechanism, the parity bits of the key will be set properly.

If the requested type of key requires more bytes than are available by concatenating the two original keys’ values, an error is generated.

This mechanism has the following rules about key sensitivity and extractability:

- If either of the two original keys has its CKA_SENSITIVE attribute set to CK_TRUE, so does the derived key. If not, then the derived key’s CKA_SENSITIVE attribute is set either from the supplied template or from a default value.
- Similarly, if either of the two original keys has its CKA_EXTRACTABLE attribute set to CK_FALSE, so does the derived key. If not, then the derived key’s CKA_EXTRACTABLE attribute is set either from the supplied template or from a default value.
- The derived key’s CKA_ALWAYS_SENSITIVE attribute is set to CK_TRUE if and only if both of the original keys have their CKA_ALWAYS_SENSITIVE attributes set to CK_TRUE.
- Similarly, the derived key’s CKA_NEVER_EXTRACTABLE attribute is set to CK_TRUE if and only if both of the original keys have their CKA_NEVER_EXTRACTABLE attributes set to CK_TRUE.

1.1.4 Concatenation of a base key and data

This mechanism, denoted CKM_CONCATENATE_BASE_AND_DATA, derives a secret key by concatenating data onto the end of a specified secret key.

This mechanism takes a parameter, a CK_KEY_DERIVATION_STRING_DATA structure, which specifies the length and value of the data which will be appended to the base key to derive another key.
For example, if the value of the base key is 0x01234567, and the value of the data is 0x89ABCDEF, then
the value of the derived key will be taken from a buffer containing the string 0x0123456789ABCDEF.

- If no length or key type is provided in the template, then the key produced by this mechanism will be a
generic secret key. Its length will be equal to the sum of the lengths of the value of the original key
and the data.
- If no key type is provided in the template, but a length is, then the key produced by this mechanism
will be a generic secret key of the specified length.
- If no length is provided in the template, but a key type is, then that key type must have a well-defined
length. If it does, then the key produced by this mechanism will be of the type specified in the
template. If it doesn’t, an error will be returned.
- If both a key type and a length are provided in the template, the length must be compatible with that
key type. The key produced by this mechanism will be of the specified type and length.

If a DES, DES2, DES3, or CDMF key is derived with this mechanism, the parity bits of the key will be set
properly.

If the requested type of key requires more bytes than are available by concatenating the original key’s
value and the data, an error is generated.

This mechanism has the following rules about key sensitivity and extractability:

- If the base key has its CKASensitive attribute set to CK_TRUE, so does the derived key. If not, then
the derived key’s CKASensitive attribute is set either from the supplied template or from a
default value.
- Similarly, if the base key has its CKAExtractable attribute set to CK_FALSE, so does the
derived key. If not, then the derived key’s CKAExtractable attribute is set either from the
supplied template or from a default value.
- The derived key’s CKAlwaysSensitive attribute is set to CK_TRUE if and only if the base key has its CKALWAYSSENSITIVE attribute set to CK_TRUE.
- Similarly, the derived key’s CKANeverExtractable attribute is set to CK_TRUE if and only if
the base key has its CKNEVEREXTRACTABLE attribute set to CK_TRUE.

1.1.5 Concatenation of data and a base key

This mechanism, denoted CKM_CONCATENATE_DATA_AND_BASE, derives a secret key by
prepending data to the start of a specified secret key.

This mechanism takes a parameter, a CK KEY DERIVATION STRING DATA structure, which
specifies the length and value of the data which will be prepended to the base key to derive another key.
For example, if the value of the base key is 0x01234567, and the value of the data is 0x89ABCDEF, then
the value of the derived key will be taken from a buffer containing the string 0x0123456789ABCDEF.

- If no length or key type is provided in the template, then the key produced by this mechanism will be a
generic secret key. Its length will be equal to the sum of the lengths of the data and the value of the
original key.
- If no key type is provided in the template, but a length is, then the key produced by this mechanism
will be a generic secret key of the specified length.
- If no length is provided in the template, but a key type is, then that key type must have a well-defined
length. If it does, then the key produced by this mechanism will be of the type specified in the
template. If it doesn’t, an error will be returned.
- If both a key type and a length are provided in the template, the length must be compatible with that
key type. The key produced by this mechanism will be of the specified type and length.

If a DES, DES2, DES3, or CDMF key is derived with this mechanism, the parity bits of the key will be set
properly.

If the requested type of key requires more bytes than are available by concatenating the data and the
original key’s value, an error is generated.

This mechanism has the following rules about key sensitivity and extractability:
If the base key has its `CKA_SENSITIVE` attribute set to `CK_TRUE`, so does the derived key. If not, then the derived key's `CKA_SENSITIVE` attribute is set either from the supplied template or from a default value.

Similarly, if the base key has its `CKA_EXTRACTABLE` attribute set to `CK_FALSE`, so does the derived key. If not, then the derived key's `CKA_EXTRACTABLE` attribute is set either from the supplied template or from a default value.

The derived key's `CKA_ALWAYS_SENSITIVE` attribute is set to `CK_TRUE` if and only if the base key has its `CKA_ALWAYS_SENSITIVE` attribute set to `CK_TRUE`.

Similarly, the derived key's `CKA_NEVER_EXTRACTABLE` attribute is set to `CK_TRUE` if and only if the base key has its `CKA_NEVER_EXTRACTABLE` attribute set to `CK_TRUE`.

### 1.1.6 XORing of a key and data

XORing key derivation, denoted `CKM_XOR_BASE_AND_DATA`, is a mechanism which provides the capability of deriving a secret key by performing a bit XORing of a key pointed to by a base key handle and some data.

This mechanism takes a parameter, a `CK_KEY_DERIVATION_STRING_DATA` structure, which specifies the data with which to XOR the original key's value.

For example, if the value of the base key is 0x01234567, and the value of the data is 0x89ABCDEF, then the value of the derived key will be taken from a buffer containing the string 0x88888888.

- If no length or key type is provided in the template, then the key produced by this mechanism will be a generic secret key. Its length will be equal to the minimum of the lengths of the data and the value of the original key.
- If no key type is provided in the template, but a length is, then the key produced by this mechanism will be a generic secret key of the specified length.
- If no length is provided in the template, but a key type is, then that key type must have a well-defined length. If it does, then the key produced by this mechanism will be of the type specified in the template. If it doesn't, an error will be returned.
- If both a key type and a length are provided in the template, the length must be compatible with that key type. The key produced by this mechanism will be of the specified type and length.

If a DES, DES2, DES3, or CDMF key is derived with this mechanism, the parity bits of the key will be set properly.

If the requested type of key requires more bytes than are available by taking the shorter of the data and the original key's value, an error is generated.

This mechanism has the following rules about key sensitivity and extractability:

- If the base key has its `CKA_SENSITIVE` attribute set to `CK_TRUE`, so does the derived key. If not, then the derived key's `CKA_SENSITIVE` attribute is set either from the supplied template or from a default value.
- Similarly, if the base key has its `CKA_EXTRACTABLE` attribute set to `CK_FALSE`, so does the derived key. If not, then the derived key's `CKA_EXTRACTABLE` attribute is set either from the supplied template or from a default value.
- The derived key's `CKA_ALWAYS_SENSITIVE` attribute is set to `CK_TRUE` if and only if the base key has its `CKA_ALWAYS_SENSITIVE` attribute set to `CK_TRUE`.
- Similarly, the derived key's `CKA_NEVER_EXTRACTABLE` attribute is set to `CK_TRUE` if and only if the base key has its `CKA_NEVER_EXTRACTABLE` attribute set to `CK_TRUE`.

### 1.1.7 Extraction of one key from another key

Extraction of one key from another key, denoted `CKM_EXTRACT_KEY_FROM_KEY`, is a mechanism which provides the capability of creating one secret key from the bits of another secret key.

This mechanism has a parameter, a `CK_EXTRACT_PARAMS`, which specifies which bit of the original key should be used as the first bit of the newly-derived key.
We give an example of how this mechanism works. Suppose a token has a secret key with the 4-byte value 0x329F84A9. We will derive a 2-byte secret key from this key, starting at bit position 21 (i.e., the value of the parameter to the CKM_EXTRACT_KEY_FROM_KEY mechanism is 21).

1. We write the key's value in binary: 0011 0010 1001 1111 1000 0100 1010 1001. We regard this binary string as holding the 32 bits of the key, labeled as b0, b1, ..., b31.

2. We then extract 16 consecutive bits (i.e., 2 bytes) from this binary string, starting at bit b21. We obtain the binary string 1001 0101 0010 0110.

3. The value of the new key is thus 0x9526.

Note that when constructing the value of the derived key, it is permissible to wrap around the end of the binary string representing the original key's value.

If the original key used in this process is sensitive, then the derived key must also be sensitive for the derivation to succeed.

- If no length or key type is provided in the template, then an error will be returned.
- If no key type is provided in the template, but a length is, then the key produced by this mechanism will be a generic secret key of the specified length.
- If no length is provided in the template, but a key type is, then that key type must have a well-defined length. If it does, then the key produced by this mechanism will be of the type specified in the template. If it doesn’t, an error will be returned.
- If both a key type and a length are provided in the template, the length must be compatible with that key type. The key produced by this mechanism will be of the specified type and length.

If a DES, DES2, DES3, or CDMF key is derived with this mechanism, the parity bits of the key will be set properly.

If the requested type of key requires more bytes than the original key has, an error is generated.

This mechanism has the following rules about key sensitivity and extractability:

- If the base key has its CKA_SENSITIVE attribute set to CK_TRUE, so does the derived key. If not, then the derived key's CKA_SENSITIVE attribute is set either from the supplied template or from a default value.
- Similarly, if the base key has its CKA_EXTRACTABLE attribute set to CK_FALSE, so does the derived key. If not, then the derived key's CKA_EXTRACTABLE attribute is set either from the supplied template or from a default value.
- The derived key’s CKA_ALWAYS_SENSITIVE attribute is set to CK_TRUE if and only if the base key has its CKA_ALWAYS_SENSITIVE attribute set to CK_TRUE.
- Similarly, the derived key’s CKA_NEVER_EXTRACTABLE attribute is set to CK_TRUE if and only if the base key has its CKA_NEVER_EXTRACTABLE attribute set to CK_TRUE.

1.1.8 Public key from private key

Public key from private key, denoted CKM_PUB_KEY_FROM_PRIV_KEY, is a mechanism which creates a matching public key from and existing private key. This mechanism takes no parameters. The public key matches the private key's type, and all the key specific parameters are generated by the mechanism. If the following parameters are not set in the template, they will default as follows:

1. CKA_TOKEN – CK_FALSE
2. CKA_PRIVATE – CK_FALSE
3. CKA_MODIFIABLE – CK_TRUE
4. CKA_LOCAL – CK_FALSE
5. CKA_SENSITIVE – CK_FALSE
6. CKA_ALWAYS_SENSITIVE – CK_FALSE
7. CKA_EXTRACTABLE – CK_TRUE
8. CKA_NEVER_EXTRACTABLE – CK_FALSE
9. CKA_LABEL – NULL
10. CKA_ENCRYPT – to CKA_DECRYPT of the private key.
11. CKA_VERIFY – to CKA_SIGN of the private key.
12. CKA_VERIFY_RECOVER – to CKA_SIGN_RECOVER of the private key.
13. CKA_WRAP – to CKA_UNWRAP of the private key.
14. CKA_DERIVE – to CKA_DERIVE of the private key.
15. CKA_ID – copy from private key.
16. CKA_START_DATE – copy from private key.
17. CKA_END_DATE – copy from private key.
18. CKA_SUBJECT – copy from private key.

This mechanism should be supported for all private key types supported by the token.