Function	Supported Algorithms
asymmetric_sign()	RSA, DSA
asymmetric_verify()	RSA, DSA
create_asymmetric_priv_key()	RSA, DSA, DH
create_asymmetric_pub_key()	RSA, DSA, DH
create_dh_parameters()	DH

Note

Although you can create keys using any of the RSA, DSA, or DH encryption algorithms, other functions that take key arguments might accept only certain types of keys. For example, asymmetric_encrypt() and asymmetric_decrypt() accept only RSA keys.

The following descriptions describe the calling sequences for MySQL Enterprise Encryption functions. For additional examples and discussion, see Section 6.6.2, "MySQL Enterprise Encryption Usage and Examples".

- asymmetric_decrypt(*algorithm*, *crypt_str*, *key_str*)

  Decrypts an encrypted string using the given algorithm and key string, and returns the resulting plaintext as a binary string. If decryption fails, the result is NULL.

  key_str must be a valid key string in PEM format. For successful decryption, it must be the public or private key string corresponding to the private or public key string used with asymmetric_encrypt() to produce the encrypted string. *algorithm* indicates the encryption algorithm used to create the key.

  Supported *algorithm* values: 'RSA'

  For a usage example, see the description of asymmetric_encrypt().

- asymmetric_derive(*pub_key_str*, *priv_key_str*)

  Derives a symmetric key using the private key of one party and the public key of another, and returns the resulting key as a binary string. If key derivation fails, the result is NULL.

  pub_key_str and *priv_key_str* must be valid key strings in PEM format. They must be created using the DH algorithm.

  Suppose that you have two pairs of public and private keys:

```
SET @dhp = create_dh_parameters(1024);
SET @priv1 = create_asymmetric_priv_key('DH', @dhp);
SET @pub1 = create_asymmetric_pub_key('DH', @priv1);
SET @priv2 = create_asymmetric_priv_key('DH', @dhp);
SET @pub2 = create_asymmetric_pub_key('DH', @priv2);
```

  Suppose further that you use the private key from one pair and the public key from the other pair to create a symmetric key string. Then this symmetric key identity relationship holds:

```
asymmetric_derive(@pub1, @priv2) = asymmetric_derive(@pub2, @priv1)
```

- asymmetric_encrypt(*algorithm*, *str*, *key_str*)

  Encrypts a string using the given algorithm and key string, and returns the resulting ciphertext as a binary string. If encryption fails, the result is NULL.