```
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives import padding
import os
import binascii

# Function to encrypt data using AES
def encrypt_data(data, key):
    # Generate a random IV
    iv = os.urandom(16)

    # Initialize the AES cipher in CBC mode
    cipher = Cipher(algorithms.AES(key), modes.CBC(iv), backend=default_backend())
    encryptor = cipher.encryptor()

    # Pad the data to ensure it is a multiple of block size (16 bytes)
    padder = padding.PKCS7(128).padder()
    padded_data = padder.update(data.encode()) + padder.finalize()

    # Encrypt the padded data
    ciphertext = encryptor.update(padded_data) + encryptor.finalize()

    # Return the IV and ciphertext in hexadecimal format
    return binascii.hexlify(iv).decode('utf-8'),
binascii.hexlify(ciphertext).decode('utf-8')

# Function to decrypt data using AES
def decrypt_data(iv, ciphertext, key):
    # Convert hex to bytes
    iv = binascii.unhexlify(iv)
    ciphertext = binascii.unhexlify(ciphertext)

    # Initialize the AES cipher in CBC mode with the given IV
    cipher = Cipher(algorithms.AES(key), modes.CBC(iv), backend=default_backend())
    decryptor = cipher.decryptor()

    # Decrypt the data
    decrypted_data = decryptor.update(ciphertext) + decryptor.finalize()

    # Unpad the decrypted data
    unpadder = padding.PKCS7(128).unpadder()
    unpadded_data = unpadder.update(decrypted_data) + unpadder.finalize()

    return unpadded_data.decode()

# Example usage
if __name__ == "__main__":
    key = os.urandom(32)  # 32-byte key for AES-256
    data = "This is a secret message"
```

```
print("Original Data:", data)

# Encrypt data
iv, ciphertext = encrypt_data(data, key)
print("IV:", iv)
print("Ciphertext:", ciphertext)

# Decrypt data
decrypted_data = decrypt_data(iv, ciphertext, key)
print("Decrypted Data:", decrypted_data)
```