```
### Import Necessary Libraries

from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.asymmetric import dsa
from cryptography.hazmat.backends import default_backend

### Define Functions

def generate_key_pair():
    private_key = dsa.generate_private_key(key_size=1024,
backend=default_backend())
    return private_key, private_key.public_key()

def sign_message(private_key, message):
    return private_key.sign(message.encode(), hashes.SHA256())

def verify_signature(public_key, message, signature):
    try:
        public_key.verify(signature, message.encode(), hashes.SHA256())
        return True
    except Exception:
        return False

### Generating User1'keys and verifying it with his public key

message = "Pay authors a bonus of $20,000."
privkey1, pubkey1 = generate_key_pair()
signature = sign_message(privkey1, message)
is_verified = verify_signature(pubkey1, message, signature)

if not is_verified:
    print("Signature not Verified")
else:
    print("Signature Verified")

### Generating User2'keys and verifying User1's Digital signature with User2's
public key

privkey2, pubkey2 = generate_key_pair()
is_verified = verify_signature(pubkey2, message, signature)

if not is_verified:
    print("Signature not Verified")
else:
    print("Signature Verified")
```

output:-

```
Signature not Verified
```