

Here's a protocol for playing **Rock Paper Scissors** between Alice and Bob in a secure and cheat-resistant manner:

### **Rock Paper Scissors Protocol:**

#### **1. Player Initialization:**

- Alice and Bob each generate a secret cryptographic key for the match. The keys should be kept confidential.
- Both players exchange public keys securely, ensuring that each player knows the other's public key.

#### **2. Game Setup:**

- Define the moves for Rock, Paper, and Scissors.
- Each player privately selects their move (Rock, Paper, or Scissors) based on their secret key.

#### **3. Move Exchange:**

- Players exchange their moves simultaneously using a secure communication channel.
- Moves are encrypted using the recipient's public key, ensuring confidentiality during transmission.

#### **4. Move Decryption:**

- Upon receiving the opponent's encrypted move, each player decrypts the move using their private key.

#### **5. Move Reveal:**

- Players simultaneously reveal their decrypted moves.
- Moves are publicly disclosed to ensure transparency.

#### **6. Game Outcome Determination:**

- Determine the winner based on the standard Rock Paper Scissors rules:
  - Rock crushes Scissors (Rock wins)
  - Scissors cuts Paper (Scissors wins)

- Paper covers Rock (Paper wins)
- If both players choose the same move, it's a tie.

#### **7. Match Progression:**

- Record the outcomes of each game within the match.
- Proceed to the next game until the predefined length of the match is reached.

#### **8. Security Measures:**

- The use of cryptographic keys adds an additional layer of security, preventing cheating by keeping the moves confidential until the reveal stage.
- Public key exchange ensures that players can verify each other's moves.
- Transparency in move reveal prevents any manipulation during the game.

#### **9. Reporting:**

- Maintain a log of moves and game outcomes for auditing purposes.
- Report the final results of each match, including the winner.

This protocol leverages cryptographic principles to ensure confidentiality, integrity, and transparency during the Rock Paper Scissors game. The use of public and private keys adds an extra layer of security, making it difficult for players to cheat and ensuring a fair and trustworthy gaming experience.

### **Generating Private and Public Keys:**

#### **# Generate private key for Bob**

```
openssl genpkey -algorithm RSA -out bob_private_key.pem
```

### # Extract public key from Bob's private key

```
openssl rsa -pubout -in bob_private_key.pem -out bob_public_key.pem
```

### # Generate private key for Alice

```
openssl genpkey -algorithm RSA -out alice_private_key.pem
```

### # Extract public key from Alice's private key

```
openssl rsa -pubout -in alice_private_key.pem -out alice_public_key.pem
```

Below is a basic Python implementation for the "Game Setup" part of the Rock Paper Scissors protocol. This implementation uses the **cryptography** library for cryptographic functions.

### Game Setup Implementation:

```
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives.kdf.pbkdf2 import PBKDF2HMAC
from cryptography.hazmat.primitives.asymmetric import rsa
from cryptography.hazmat.primitives import serialization

def generate_key_pair():
    # Generate RSA key pair
    private_key = rsa.generate_private_key(
        public_exponent=65537,
        key_size=2048,
        backend=default_backend()
    )

    # Extract public key
    public_key = private_key.public_key()

    # Serialize public key to bytes
    public_key_bytes = public_key.public_bytes(
```

```

        encoding=serialization.Encoding.PEM,
        format=serialization.PublicFormat.SubjectPublicKeyInfo
    )

    return private_key, public_key_bytes

def derive_secret_key(master_key, salt=b'salt'):
    # Derive a secret key from a master key using PBKDF2
    kdf = PBKDF2HMAC(
        algorithm=hashes.SHA256(),
        salt=salt,
        iterations=100000,
        length=32,
        backend=default_backend()
    )
    secret_key = kdf.derive(master_key)
    return secret_key

def select_move(secret_key, move_options):
    # Select a move based on the secret key
    move_index = int.from_bytes(secret_key, byteorder='big') %
len(move_options)
    return move_options[move_index]

# Game setup
move_options = ['Rock', 'Paper', 'Scissors']

# Alice's setup
alice_private_key, alice_public_key = generate_key_pair()
alice_secret_key = derive_secret_key(alice_private_key.private_bytes(
    encoding=serialization.Encoding.DER,
    format=serialization.PrivateFormat.TraditionalOpenSSL,
    encryption_algorithm=serialization.NoEncryption()
))

alice_move = select_move(alice_secret_key, move_options)

# Bob's setup
bob_private_key, bob_public_key = generate_key_pair()
bob_secret_key = derive_secret_key(bob_private_key.private_bytes(
    encoding=serialization.Encoding.DER,
    format=serialization.PrivateFormat.TraditionalOpenSSL,
    encryption_algorithm=serialization.NoEncryption()
))

bob_move = select_move(bob_secret_key, move_options)

# Print moves
print(f"Alice's move: {alice_move}")
print(f"Bob's move: {bob_move}")

```

In the protocol we discussed, Alice and Bob generate a new secret key for each round based on their private keys. This process ensures that each round's move is independently determined and not predictable by the opponent. The steps involved include:

**1. Key Generation:**

- Before each round, Alice and Bob generate a new pair of private and public keys.

**2. Secret Key Derivation:**

- They use their newly generated private keys to derive a secret key for the specific round. The secret key is a unique input for that round.

**3. Move Selection:**

- The derived secret key is then used to deterministically select a move for that round. This move is unpredictable for the opponent due to the secrecy of the key.

**4. Round Execution:**

- The players exchange their moves and follow the rules of the Rock Paper Scissors game to determine the winner of that round.

By generating a new secret key for each round, the protocol adds an additional layer of unpredictability to the game, making it more challenging for opponents to guess or exploit patterns. It enhances the security and fairness of the game by ensuring that each round's move is uniquely tied to a specific key.

**Role of the Secret Key:**

- The secret key acts as a unique and confidential input to the move selection algorithm. Each player uses their secret key to determine their move, and since the secret key is not known to the opponent, the opponent cannot reliably predict the player's move.

**Element of Unpredictability:**

- While the process is deterministic, the secrecy of the key introduces an element of unpredictability from the opponent's perspective. The opponent cannot predict the outcome of the deterministic process because they do not know the input (secret key) used in that process.

i can't understand how alice and bob can play this game after generate secret key

Let's walk through the steps for Alice and Bob to play the Rock Paper Scissors game after generating their secret keys:

### 1. Key Generation:

- Before the game starts, both Alice and Bob generate a new pair of private and public keys.

### 2. Secret Key Derivation:

- Each player uses their newly generated private key to derive a secret key specifically for the upcoming round of the game.

### 3. Move Selection:

- With the derived secret key, each player uses the **select\_move** function (as defined earlier) to deterministically select a move (Rock, Paper, or Scissors) for that round.

pythonCopy code

```
# Assuming move_options is a list ['Rock', 'Paper', 'Scissors']
alice_move = select_move(alice_secret_key, move_options)
bob_move = select_move(bob_secret_key, move_options)
```

### 4. Exchange Moves:

- Alice and Bob exchange their moves securely. The actual exchange method will depend on the communication channel used in the protocol.

### 5. Determine Round Winner:

- The moves are then compared to determine the winner of that round based on the Rock Paper Scissors rules.

## **6. Repeat for Next Round:**

- If the game consists of multiple rounds, players repeat the process. Before each round, they generate a new pair of private and public keys, derive a new secret key, and select a new move.

## **Summary:**

- The game proceeds in rounds, with each round involving the generation of new keys and the selection of moves based on the secret keys. The use of secret keys ensures that each round's move is unpredictable for the opponent, adding an element of randomness and security to the game.