PGP (Pretty Good Privacy)

PGP (Pretty Good Privacy) is a popular encryption and digital signing system used for secure email communication. It provides both confidentiality and authenticity of the message. PGP is a hybrid cryptosystem, which means it uses both symmetric-key encryption and public-key encryption.

The PGP algorithm involves the following steps:

1. Key Generation:

    - Generate a symmetric key (session key), which is a random sequence of bits
    - Encrypt the session key with recipient's public key using an asymmetric algorithm (for example, RSA)
    - Attach the encrypted session key to the message

    Public key: The recipient's public key Private key: The recipient's private key

2. Encryption:

    - Generate a symmetric key (session key)
    - Encrypt the message with the session key using symmetric-key algorithm (for example, AES)
    - Attach the encrypted message to the message

3. Decryption:

    - Decrypt the session key using the recipient's private key
    - Decrypt the message using the session key

4. Digital Signing:

    - Generate a hash of the message using a hash function (for example, SHA-256)
    - Sign the hash with sender's private key, using an asymmetric algorithm
    - Attach the signature to the message

    Public key: The sender's public key
    Private key: The sender's private key

5. Verification:

    - Calculate the hash of the received message
    - Decrypt the signature using sender's public key
    - Compare the two hashes. If they are the same, the message is authentic.

The mathematical formulas used in PGP algorithm:

- Symmetric Key Encryption: This is a type of encryption that uses the same key to encrypt and decrypt a message.
- Public Key Encryption: This is a type of encryption that uses two keys - a public key and a private key. The public key is used to encrypt a message and the private key is used to decrypt it. For example, RSA is a public key encryption algorithm.
- Digital Signature: This is a mathematical technique used to ensure the authenticity, integrity and non-repudiation of data. It involves generating a hash of the message and then encrypting that hash with the sender's private key.

Example of JavaScript code for PGP Encryption and Decryption:

```
// node-js
const crypto = require('crypto');

// PGP Encryption
function pgpEncrypt(publicKey, plaintextMsg) {
  const sessionKey = crypto.randomBytes(32); // Generate random key
  const encryptedSessionKey = crypto.publicEncrypt(publicKey, sessionKey);
  const cipher = crypto.createCipher('aes-256-cbc', sessionKey);
  let ciphertext = cipher.update(plaintextMsg, 'utf8', 'base64');
  ciphertext += cipher.final('base64');
  return [encryptedSessionKey, ciphertext];
}

// PGP Decryption
function pgpDecrypt(privateKey, encryptedSessionKey, ciphertextMsg) {
  const sessionKey = crypto.privateDecrypt(privateKey, encryptedSessionKey);
  const decipher = crypto.createDecipher('aes-256-cbc', sessionKey);
  let plaintext = decipher.update(ciphertextMsg, 'base64', 'utf8');
  plaintext += decipher.final('utf8');
  return plaintext;
}

// Digital Signing
function pgpSign(privateKey, message) {
  const hash = crypto.createHash('sha256');
  hash.update(message);
  const hashedData = hash.digest();
  const signature = crypto.privateEncrypt(privateKey, hashedData);
  return signature;
}
```

```
// Verify Signature
function pgpVerify(publicKey, message, signature) {
  const hash = crypto.createHash('sha256');
  hash.update(message);
  const hashedData = hash.digest();
  const decryptedSignature = crypto.publicDecrypt(publicKey, signature);
  return decryptedSignature.equals(hashedData);
}
```

Note:

This is a basic implementation of PGP algorithm.