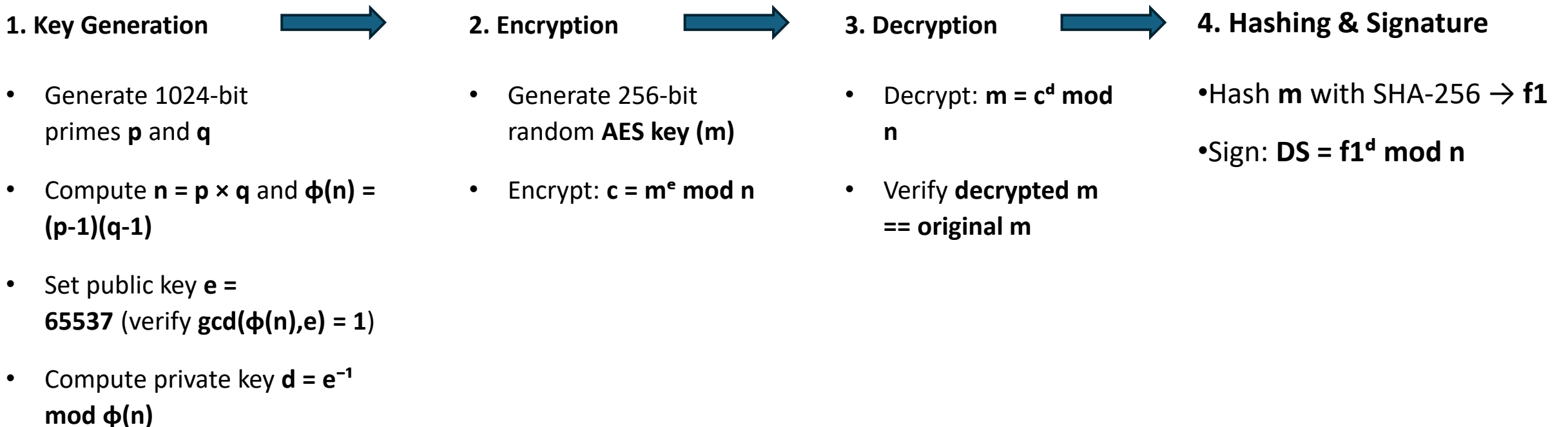


Programming Task (Using SageMath)

RSA Implementation

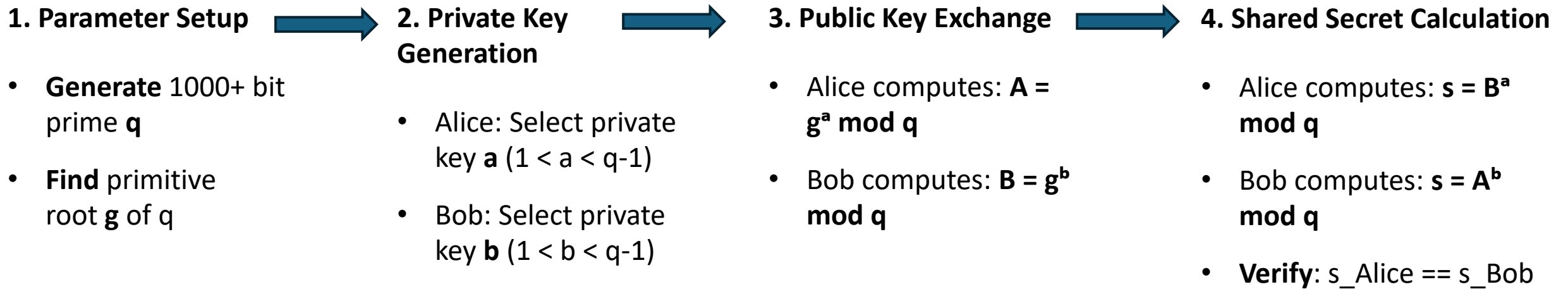
- Generate random primes p and q , each 1000+ bits in size.
- Use the public key $e = 65537$.
- Generate the private key d .
- Generate a random 256-bit AES key (treat this as the message m).
- Encrypt m using the public key: Compute ciphertext $c = m^e \bmod n$ (where $n = p \cdot q$).
- Hash the message m : Input m into SHA-256, producing a fingerprint $f1$.
- Sign $f1$ with the private key d : Compute digital signature $DS = f1^d \bmod n$.



Programming Task (Using SageMath)

Diffie-Hellman (DH) Key Exchange

- Perform a DH calculation similar to the RSA example above.
- Use 1000+ bit parameters (e.g., safe primes). random primes p and q , each 1000+ bits in size.



Programming Task (Using SageMath)

ECDSA (Elliptic Curve Digital Signature Algorithm)

- Perform an ECDSA calculation similar to the RSA example.
- Use ECC with at least 256-bit curves

NOTES:

1. **SageMath Reference:** Elliptic curve operations: [SageMath Finite Field Elliptic Curves Documentation](#)
2. **Curve Parameters:**
 - Prime modulus (**q**): From [SafeCurves Project](#)
 - Coefficients (**a, b**): Verified via [SafeCurves Equation Guide](#) to satisfy the elliptic curve equation: $y^2 = x^3 + ax + b$
 - Base point (**G**): Coordinates from [SafeCurves Base Points](#)
3. **Hash Assumption:** Message hash (**e = 13**): Simplified value for demonstration (based on slide 22 of "Digital Signature" lecture notes).

1. Parameter Setup

- **Prime (q):** Predefined 256-bit prime
- **Curve Coefficients:** $a = -3$, b (predefined)
- **Base Point (G):** Predefined (x, y) coordinates
- **Order (n):** Order of G



2. Key Generation

- **Private Key (d):** Random integer $\in [1, n-1]$
- **Public Key (Q):** Compute $Q = d \times G$



3. Signature Generation

- Generate random $k \in [1, n-1]$
- Compute $P = k \times G \rightarrow r = x_P \bmod n$ (repeat if $r=0$)
- Compute hash **e = SHA-256(m)** (simplified as $e=13$ here)
- Calculate $s = k^{-1} \times (e + dxr) \bmod n$ (repeat if $s=0$)
- Signature: Pair (r, s)



4. Signature Verification

- Compute $w = s^{-1} \bmod n$
- Calculate $u1 = exw \bmod n$, $u2 = rxw \bmod n$
- Compute $X = u1 \times G + u2 \times Q$
- **Verify:** $v = x_X \bmod n$ matches r