Modular Exponentiation in Cryptography

Cybersecurity Specialization
-- Hardware Security

Diffie-Hellman Key Exchange

- Alice:
  - Generate random $X_A < q$
  - Calculate $Y_A = a^{X_A}$ (mod q)
  - Send Y_A to Bob
  - keep X_A as a secret
- Key generation
  - $K = (Y_B)^{X_A}$ (mod q)
- Bob:
  - Generate random $X_B < q$
  - Calculate $Y_B = a^{X_B}$ (mod q)
  - Send Y_B to Alice
  - keep X_B as a secret
- Key generation
  - $K = (Y_A)^{X_B}$ (mod q)

$$\boxed{(a^{X\_B})^{X\_A} = (a^{X\_A})^{X\_B} \pmod{n}}$$

RSA: Public Key Encryption

- Rivest-Shamir-Adelman algorithm (1998)
- Asymmetric encryption
- How public key encryption system works?
  - Each user has a pair of keys (K_{PUB}, K_{PRIV})
  - Encryption: $C = E_{K\_PUB}(P)$
  - Decryption: $P = D_{K\_PRIV}(C)$
  - Requirement: $P = D_{K\_PRIV}(E_{K\_PUB}(P))$
  - Current RSA keys are 1024-bit, but will be 2048-bit or 4096-bit very soon.

RSA Algorithm

- Select n = p * q, p and q are large primes
- Choose e relatively prime to (p-1)*(q-1)
- Select d, s.t. e*d = 1 mod (p-1)*(q-1)
- Public key (e,n), private key (d,n)

- Encryption: $C = P^e \pmod{n}$
- Decryption: $P = C^d \pmod{n}$

if $e*d=1 \pmod{(p-1)(q-1)}$

$(x^e)^d=x \pmod{n}$

RSA Algorithm: Example

- Set up the parameters:
  - p=11, q=17, n=11*17=187, (p-1)*(q-1)=160
  - Choose e=7, gcd(7, 160) = 1
  - d=23, 7*23 = 1 (mod 160)
  - public key (7, 187), private key (23, 187)
- Encryption and decryption:
  - Message x=88, encrypted as 88^7=11 (mod 187)
  - Ciphertext y=11, decrypted to 11^{23} = 88 (mod 187)
- How to compute 88^7 and 11^{23}(mod 187), modular exponentiations when RSA's key length is 1024- or 2048-bit?