

# Public-key Cryptography II

Practical instantiations

# Practical Constructions

- ElGamal
- D-H Key Exchange
- ElGamal Encryption/Signatures
- Schnorr Signatures

# ElGamal

- Defined by 3 algorithms:
- $(PK, SK) \leftarrow \text{KeyGen}(1^n)$ 
  - run  $\text{GroupGen}(1^n) \rightarrow (G, g, q)$
  - pick  $x \leftarrow \mathbb{Z}_q$ , find  $h = g^x$
  - return  $PK = (G, g, q, h)$ ,  $SK = (G, g, q, x)$
- $C \leftarrow \text{Encrypt}(PK, m \in G)$ 
  - pick  $y \leftarrow \mathbb{Z}_q$
  - return  $C = (g^y, h^y \cdot m)$

# ElGamal

- $m \leftarrow \text{Decrypt}(\text{SK}, C)$ 
  - return  $m = (h^y \cdot m) / g^{yx}$
- If DDH is hard, ElGamal is CPA-secure
- Proof intuition:
  - Reduction-based proof
  - Assume ElGamal CPA-adversary A exists
  - Show DDH adversary B exists too
  - By interacting with A, B can break DDH

# ElGamal Implementation Issues

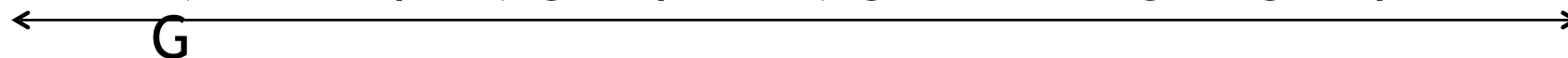
- $(G, g, q)$  usually fixed, shared among receivers
- Each receiver chooses individual  $x \leftarrow \mathbb{Z}_q$
- Choose  $|G| = q$ , prime-order subgroup of  $\mathbb{Z}_p^*$ , or elliptic curves
- $m \in G$ , unfortunately.
  - Either map actual message  $m' \leftrightarrow m$ ;  $m \in G$
  - Or just use hybrid encryption ( $K = H(m)$ ,  $m \in G$ )

# Diffie-Hellman Key Exchange



Set up three public values:

1) Prime  $p$ , 2) group  $G$ , 3) generator  $g$  of group



1. Picks secret key,  $SK_A$ ,  $SK_A < p$

2. Computes  $PK_A = g^{SK_A} \bmod p$

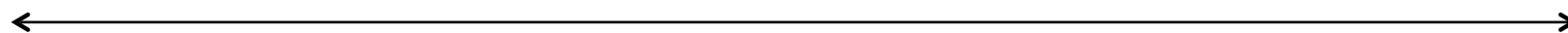
3. Picks secret key,  $SK_B$ ,  $SK_B < p$

4. Computes  $PK_B = g^{SK_B} \bmod p$

5. Exchange  $PK_A$ ,  $PK_B$

6. Compute shared key:  $K = (PK_B)^{SK_A} \bmod p$

7. Compute shared key:  $K = (PK_A)^{SK_B} \bmod p$



# Diffie-Hellman Key Exchange

- Hard for Charlie to find  $K$ , knowing only  $PK_A$  and  $PK_B$ . Why?
- Because for computing  $K$ , Charlie would need to find either  $SK_A$  or  $SK_B$ :

$$SK_A = \log_g PK_A, \text{ or}$$

$$SK_B = \log_g PK_B$$

- Then compute  $K$  similar to Alice or Bob
- Since we assume discrete logs are hard to find, secret keys hard for Charlie to compute

# D-H Man-in-the-Middle Attack

- Public values:  $p$ ,  $g$ ,  $G$  known to all



1. Picks 2 secret keys:  
 $SK_1, SK_2 < p$
2. Computes  $PK_1 = g^{SK_1} \bmod p$ ,  $PK_2 = g^{SK_2} \bmod p$

3. Picks secret key,  $SK_A < p$
4. Computes  $PK_A = g^{SK_A} \bmod p$

5. Send  $PK_A$  to Bob  
→



# D-H Man-in-the-Middle Attack



6. Intercept  $PK_A$
7. Compute  $K_2 = PK_A^{SK_2} \mod p$

8. Send  $PK_1$  to Bob  
→

9. Compute  $K_1 = PK_1^{SK_B} \mod p$

# D-H Man-in-the-Middle Attack



10. Send  $PK_B$  to Alice  
←

11. Intercept  $PK_B$   
12. Compute  
 $K_1 = PK_B^{SK_1} \bmod p$

13. Send  $PK_2$  to Alice  
←

14. Compute  $K_2 = PK_2^{SK_A} \bmod p$

# D-H Man-in-the-Middle Attack

- Alice-Charlie share key  $K_2$  ,
- Bob-Charlie share key  $K_1$
- What could Charlie do?
  - Intercept all messages between Alice, Bob
  - Just read and pass messages along
  - Or send spurious messages to either party
- Intuitive way to fix this: Authenticate Alice, Bob to each other
  - But need PKI, CA, etc. in place

# ElGamal Signatures



Set up four public values:

1) Prime  $p$ , 2) group  $G$ , 3) generator  $g$  of group  $G$ , 4) Message  $M$  to be signed by Alice



1. Picks secret key,  $SK_A < p-1$

2. Computes  $PK_A = g^{SK_A} \bmod p$

3. Send  $PK_A$  to Bob. Also send  $E_{PK_B}(M)^2$

4. Compute  $h = H(M)$ ;  
 $0 \leq h \leq p-1$

1:  $h$  is the hash of the message  $M$

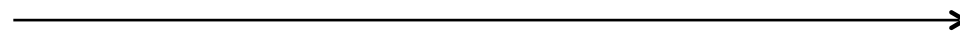
2: We assume Bob has already transmitted his public key  $PK_B$  to Alice

# ElGamal Signatures



5. Pick  $K$ ; such that  $1 \leq K \leq p-1$ ,  $\gcd(K, p-1) = 1$
6. Compute  $S_1 = g^K \bmod p$
7. Compute  $S_2 = K^{-1} (h - (SK_A \cdot S_1)) \bmod p^{-1}$
8.  $\text{Sig} = (S_1, S_2)$

9. Send Sig to Bob



10. Compute  $V_1 = g^h \bmod p$
11. Compute  $V_2 = (PK_A)^{S_1} (S_1)^{S_2} \bmod p$
12. Is  $V_1 \stackrel{?}{=} V_2$ ? If yes, accept Sig as valid, else reject