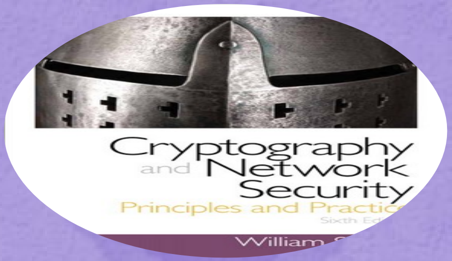


Other Public-Key Cryptosystems

Diffie-Hellman Key Exchange

- First published public-key algorithm
- A number of commercial products employ this key exchange technique
- Purpose is to enable two users to securely exchange a key that can then be used for subsequent symmetric encryption of messages
- The algorithm itself is limited to the exchange of secret values
- Its effectiveness depends on the difficulty of computing discrete logarithms

Primitive root of a prime number

The Diffie-Hellman algorithm depends for its effectiveness on the difficulty of computing discrete logarithms.

Definition:

The primitive root of a prime number $\boldsymbol{p}$ is one whose powers modulo p generate all the integers from 1 to $p - 1$. That is, if $\boldsymbol{a}$ is a primitive root of the prime number p, then the numbers

$$\boldsymbol{a \text{ mod } p,\ a^2 \text{ mod } p,\ \ldots,\ a^{p-1} \text{ mod } p}$$

are distinct and consist of the integers from 1 through $p - 1$ in some permutation.

For any integer b and a primitive root a of prime number p, we can find a unique exponent i such that

$$\boldsymbol{b = a^i \ (\text{mod } p) \text{ where } 0 \le i \le (p - 1)}$$

The exponent $\boldsymbol{i}$ is referred to as the discrete logarithm of $\boldsymbol{b}$ for the base $\boldsymbol{a}$, **mod** $\boldsymbol{p}$

$$\mathbf{dlog}_{a,p}(b)$$

- Is 3 a primitive root of 11?
- Is 3 a primitive root of 13?
- Is 3 a primitive root of 5?

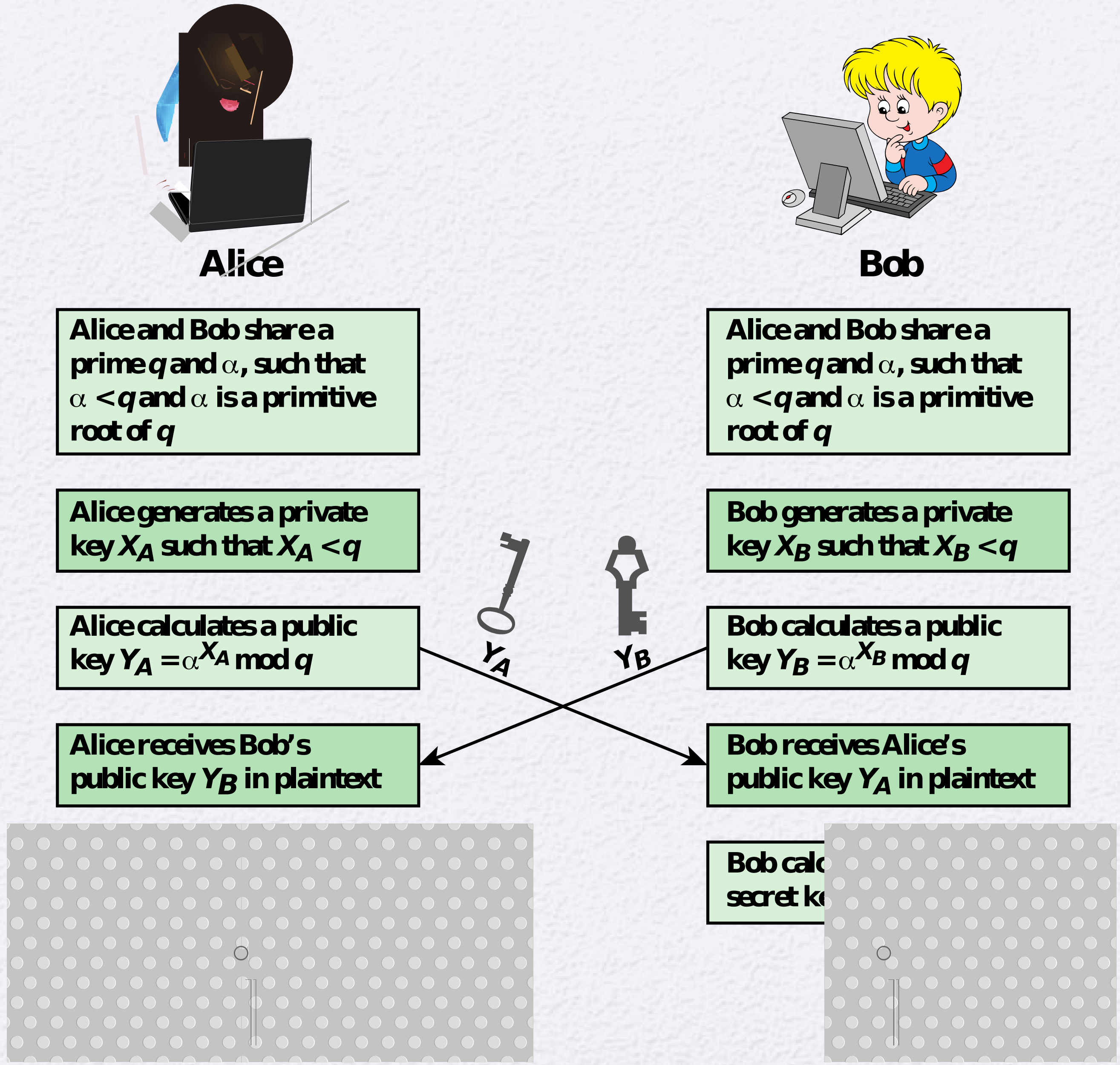


Figure 10.1 Diffie-Hellman Key Exchange

$$
\begin{aligned}
K &= (Y_B)^{X_A} \bmod q \\
&= (\alpha^{X_B} \bmod q)^{X_A} \bmod q \\
&= (\alpha^{X_B})^{X_A} \bmod q && \text{by the rules of modular arithmetic} \\
&= \alpha^{X_B X_A} \bmod q \\
&= (\alpha^{X_A})^{X_B} \bmod q \\
&= (\alpha^{X_A} \bmod q)^{X_B} \bmod q \\
&= (Y_A)^{X_B} \bmod q
\end{aligned}
$$

To determine the private key of user B, an adversary must compute:

$$\mathbf{X_B = dlog_{\alpha,q}(Y_B)}$$

Numeric Example

$q = 353$, and the primitive root $\alpha = 3$

Private Keys are chosen as $X_A = 97$, and $X_B = 233$

A computes $Y_A = 3^{97} \bmod 353 = 40$

B computes $Y_B = 3^{233} \bmod 353 = 248$

After exchanging public keys each uses their own private key to compute the common secret key

A computes $K = (Y_B)^{X_A} \bmod 353 = 248^{97} \bmod 353 = 160$

B computes $K = (Y_A)^{X_B} \bmod 353 = 40^{233} \bmod 353 = 160$

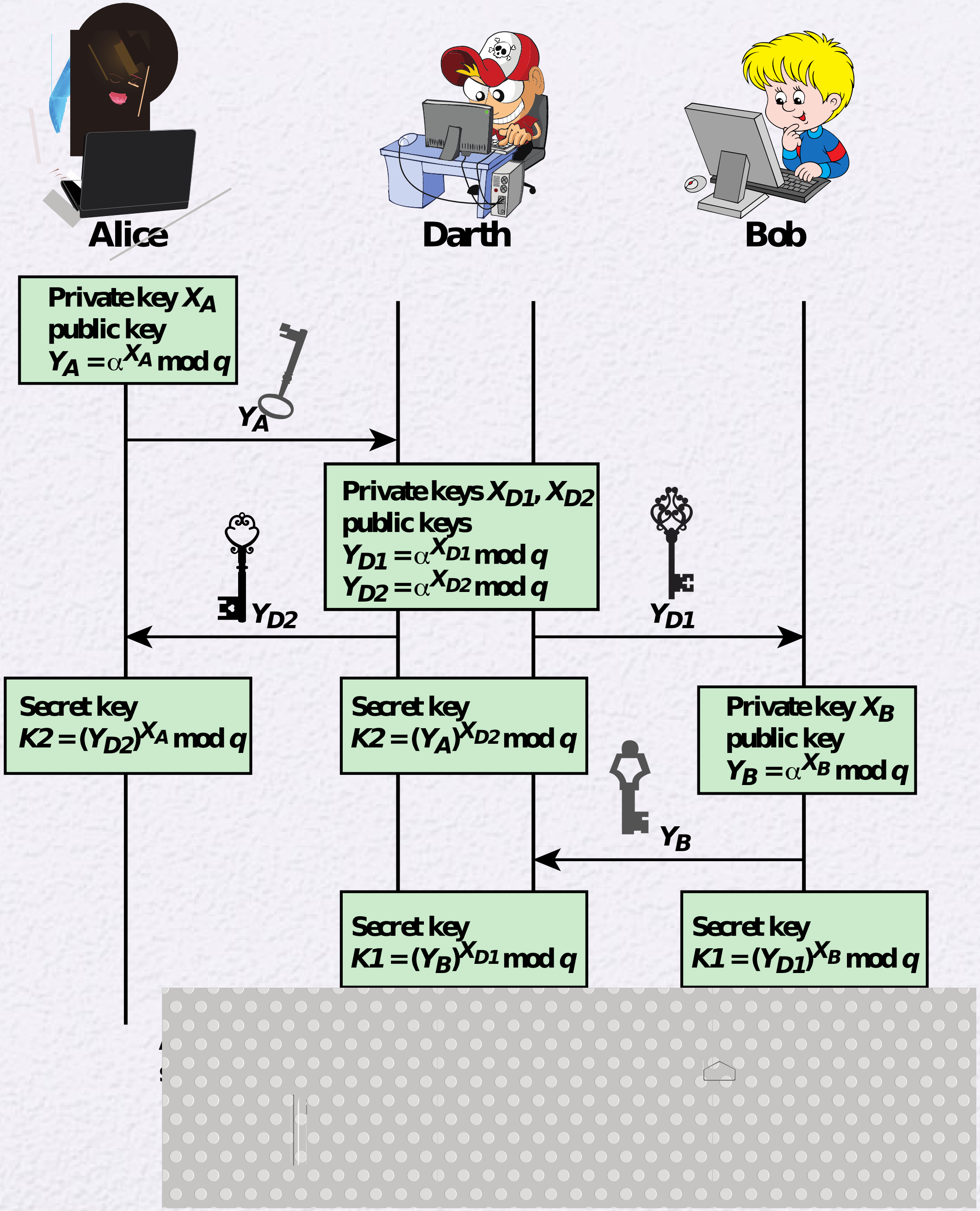
What information would any attacker have available?

In this example would it be possible by brute force to determine

Key Exchange Protocols

- Users could create random private/public Diffie-Hellman keys each time they communicate
- Vulnerable to Man-in-the-Middle-Attack
- Authentication of the keys is needed

Man in the Middle Attack



The protocol is insecure against a man-in-the-middle attack.
Suppose Alice and Bob wish to exchange keys, and Darth is the adversary.
The attack proceeds as follows:

1. Darth prepares for the attack by generating two random private keys X_{D1} and X_{D2} and then computing the corresponding public keys Y_{D1} and Y_{D2}
2. Alice transmits Y_A to Bob.
3. Darth intercepts Y_A and transmits Y_{D1} to Bob. Darth also calculates $K2 = (Y_A)^{X_{D2}} \bmod q$
4. Bob receives Y_{D1} and calculates $K1=(Y_{D1})^{X_B} \bmod q$
5. Bob transmits Y_B to Alice.
6. Darth intercepts Y_B and transmits Y_{D2} to Alice. Darth calculates $K1=(Y_B)^{X_{D1}} \bmod q$
7. Alice receives Y_{D2} and calculates $K2=(Y_{D2})^{X_A} \bmod q$.

At this point, Bob and Alice think that they share a secret key, but instead Bob and Darth share secret key K1 and Alice and Darth share secret key K2. All future communication between Bob and Alice is compromised in the following way:

Comparable Key Sizes in Terms of Computational Effort for Cryptanalysis (NIST SP-800-57)

Symmetric key algorithms	**Diffie-Hellman, Digital Signature Algorithm**	**RSA (size of *n* in bits)**	**ECC (modulus size in bits)**
80	L = 1024 N = 160	1024	160–223
112	L = 2048 N = 224	2048	224–255
128	L = 3072 N = 256	3072	256–383
192	L = 7680 N = 384	7680	384–511
256	L = 15,360 N = 512	15,360	512+

Note: L = size of public key, N = size of private key