Week 9: PKI + Networks + DNS

Key Exchange Protocols:

Question 1: Recall that in a Diffie-Hellman key exchange, there are values a, b, g and p. Alice computes g^a mod p and Bob computes g^b mod p. Which of these values (a, b, g, and p) are publicly known and which must be kept private?

Question 2: Consider the following key exchange protocol which can be used by Alice (A) and Bob (B) to agree upon a shared key, K.

Diffie-Hellman Key Exchange

- Message 1 A → B: g^a mod p
- Message 2 A ← B: g^b mod p
- <Keys are now exchanged> K = g^{ab} mod p
- Message 3 A ← B: Enc(K, secret1)
- Message 4 A → B: Enc(k, secret2)

Some additional details:

- K, the Diffie-Hellman exponents a and b, and the messages themselves are destroyed once all messages are sent. I.e., these values are not stored on Alice and Bob's devices after they are done communicating.
- Eavesdropper Eve records all communications between Alice and Bob, but is unable to decrypt them.
- At some point in the future, Eve is lucky and manages to compromise Bob's computer. Is the confidentiality of Alice and Bob's prior Diffie-Hellman-based communication in jeopardy?

Digital Certificates

We've seen powerful techniques for securing communication such that the only information we must carefully protect regards "keys" of various sorts. Given the success of cryptography in general, arguably the biggest challenge remaining for its effective use concerns exactly those keys, and how to *manage* them. For instance, how does Alice find out Bob's public key? Does it matter?

We've seen one instance of key management in lab and class: the Chain-of-Trust model. Let's explore other models, and see the advantages and disadvantages of each.

Question 1: Using a Trusted Directory Service: In this approach to key management, some central organization maintains an association between the name of each participant and their public key. Suppose everyone trusts Dirk the Director to maintain this association. Then any time Alice wants to communicate with someone, say Bob, she can contact Dirk to ask him for Bob's public key.

Issues with the Trusted Directory Service

A. Trust
B. Scalability
C. Reliability
D. Being Online
E. Others… (there are many more!)

Question 2: Chain of Trust Model: In this approach we have a Certificate Authority (CA) that issues certificates to Alice and Bob. In order to verify the CA, we check the mapping from the CA to it's public key from the CA higher up in the chain. We continue to do so until we reach the root CA. Every browser in the world ships with a list of trusted CAs. Firefox currently ships with about 88 trusted CAs preconfigured in the browser. The browser manufacturers have decided that, whether you like it or not, those CAs are trusted. Is there an advantage to having many CAs configured?

A. Yes, because you get a choice depending on who you trust
B. No, it looks like the web browser will accept *any* certificate issued by *any* of these 88 CAs.

Question 3: Web of Trust Model: This approach was pioneered by PGP, a software package for email encryption. The idea is to democratize the process of public key verification so that it does not rely upon any single central trusted authority. In this approach, each person can issue certificates for their friends, colleagues, and

others whom they know. Suppose Alice wants to contact Doug, but she doesn't know Doug. In the simplest case, if she can find someone she knows and trusts who has issued Doug a certificate, then she has a certificate for Doug, and everything is easy.

If that doesn't work, things get more interesting. Suppose Alice knows and trusts Bob, who has issued a certificate to Carol, who has in turn issued a certificate to Doug. In this case, PGP will use this certificate chain to identify Doug's public key.

In the latter scenario can we say that Alice has securely obtained a copy of Doug's public key?

A. Yes
B. No