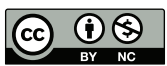
Introduction to Cryptography

Cryptographic Protocols



What are Cryptographic Protocols?

- To use cryptographic primitives properly, we need to embed them in a *protocol*: a stylized set of messages
- We've already seen one example: the hash-based coin-flipping protocol
- All real uses of cryptography involve a protocol
- Not surprisingly, protocol design is subject to subtle errors

A Warning

This warning is from the first published paper on cryptographic protocols, in 1978:

> Finally, protocols such as those developed here are prone to extremely subtle errors that are unlikely to be detected in normal operation. The need for techniques to verify the correctness of such protocols is great, and we encourage those interested in such problems to consider this area.

The warning was prescient.

Basic Concepts

Assumptions What do you assume is secure? What are the enemy's powers?

Environment Can you assume accurate clocks? Secure local storage? Memorized keys? Local computing capability? Limits on transmissions or latency?

Nonce A random value generated by one party and used only once

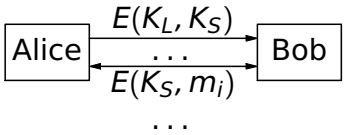
Trusted Party An entity who will carry out its functions honestly

Honest but Curious An entity who will carry out its functions honestly, but who will observe and perhaps use anything unencrypted

Often, changing assumptions or environment will require a very different protocol. It's also how academic cryptography generates so many papers—just change the assumptions or environment. . .

A Toy Protocol

Transmitting too much with a single key is a bad idea. Alice and Bob therefore want to use a separate key for each communication. Here's a toy protocol.



1. Alice and Bob share a long-term key K_L
2. Alice generates a random *session key* K_S
3. She then encrypts K_S using K_L and sends it to Bob:
   $A \to B : E(K_L, K_S)$
4. Bob decrypts the message and now knows K_S
5. Alice and Bob have a conversation protected by K_S
   $A \to B : E(K_S, m_0)$
   ...
   $B \to A : E(K_S, m_i)$
   ...

What's Wrong?

- The protocol doesn't scale—what if Alice, Bob, Carol, Dave, etc., all need to talk to each other? You'd need $O(n^2)$ keys
- If they all shared the same secret K_L, no one would have any cryptographic assurance of the identity of the other party
- No one knows if the key K_S (and hence the subsequent session) is *fresh*—maybe some attacker is replaying old traffic

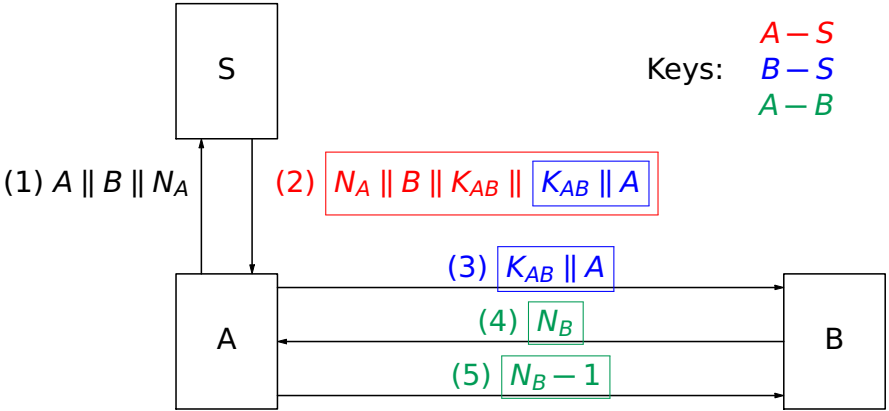
The Needham-Schroeder Protocol

- The oldest cryptographic protocol in the open literature
- Assumption: there is a universally trusted server S; Alice, Bob, etc., share a key K_{AS}, K_{BS}, etc., with S
- (S is sometimes called a *Key Distribution Center* (KDC))
- Everyone wants cryptographic assurance of the other party's identity
- All keys must be fresh

The Needham-Schroeder Protocol

1. $A \rightarrow S : A \parallel B \parallel N_A$
2. $S \rightarrow A : E(K_{AS}, N_A \parallel B \parallel K_{AB} \parallel E(K_{BS}, K_{AB} \parallel A))$
3. $A \rightarrow B : E(K_{BS}, K_{AB} \parallel A)$
4. $B \rightarrow A : E(K_{AB}, N_B)$
5. $A \rightarrow B : E(K_{AB}, N_B - 1)$

The Needham-Schroeder Protocol Pictorial



Boxes show encryption; colors show keys

The Needham-Schroeder Protocol: Explanation

1. Alice tells the server she wants to talk to Bob. She includes a nonce—a newly generated random number—to provide assurance that the server's response is fresh
2. The server sends Alice a complex message
   - Alice knows it's from the server, because it's encrypted with a key only she and S know.
   - The nonce assures freshness.
   - (Why is Bob's name there?)
   - The session key to talk to Bob is K_{AB}.
   - There is also a package encrypted to Bob that Alice can't read
3. Alice forwards that package to Bob
4. Bob decrypts it. He knows it's from S, because it's encrypted with K_{BS}; it provides the session key K_{AB} and Alice's identity
5. To ensure that K_{AB} is fresh, Bob sends Alice a nonce N_B
6. Alice subtracts 1 from it, to show that she could read it

Is it Correct?

- In 1981, Denning and Sacco pointed out that if an attacker ever recovers a single K_{AB}, they can forever talk to Bob and impersonate Alice by replaying messages 3–5.
- They suggest using timestamps to guarantee freshness
- They then revise Needham and Schroeder's public key variant with one using timestamps and *certificates*
  1. $A \rightarrow S : A, B$
  2. $S \rightarrow A : C_A, C_B$
  3. $A \rightarrow B : C_A, C_B, E(P_B, D(S_A, K_{AB} \parallel T))$

  where C_A and C_B are certificates for A and B, P_B is Bob's public key, S_A is Alice's secret key, and T is the current time
- Is this correct?

(What's a Certificate?)

- A certificate is a digitally signed statement linking an identity to a public key
- Thus, $C_A = D(S_S, A \parallel P_A \parallel T)$ is S attesting that as of time T, P_A is Alice's public key.
- Anyone who wants to believe this statement must trust S and have a copy of S's public key
- More on certificates next class

Nope!

- In 1994, Abadi and Needham found a flaw in the Denning-Sacco protocol
- Bob can copy Alice's digitally signed statement about the session key K_{AB}

  $$A \rightarrow B : C_A, C_B, E(P_B, D(S_A, K_{AB} \parallel T))$$

  into a new message to Carol:

  $$B \rightarrow C : C_A, C_C, E(P_C, D(S_A, K_{AB} \parallel T))$$

- The presence of C_A means that the messages claims to be from A, and the session key is signed by A, so it appears to be genuine
- As long as this happens around time T, it will be accepted
- And—in 1996, Lowe used automated tools to find a new, previously unnoticed flaw in the original Needham-Schroeder protocol
- Cryptographic protocols are hard...

Cryptoagility and APIs

- All real-world systems feature *cryptoagility*: the ability to support multiple algorithms
- In interactive situations, the algorithms to use are negotiated by the parties
- For things like encrypted files, there's always a header field that says what algorithm was used to encrypt the file
- Why? Because algorithms *age* and become less secure, or because better ones are developed
- Example: in 1995, systems used 1024-bit RSA, MD5, SHA-1, RC4, and DES. RSA is still secure with larger moduli—but we now have elliptic curve, SHA-2, and AES, and we'll soon have post-quantum algorithms

How Do We Handle This?

- DO NOT WRITE YOUR OWN NEGOTIATIONS
- Negotiating crypto algorithms is part of cryptographic protocols—and like every other part, it's hard to get right
- In 2006, a friend and I showed that *every* IETF protocol did hash function negotiation incorrectly—and these were Internet standards (https://www.cs.columbia.edu/~smb/papers/new-hash.pdf)
- Let's instead worry about the code

General Negotiation Style

- One side proposes a set of algorithms; the other side selects and announces its choices
- (Implementations only announce options they've implemented, of course, but the choices negotiated may depend on cost, perceived security, or even legal restrictions)
- Different types of algorithms—hash functions, public key and symmetric ciphers, digital signatures, integrity checks—can be negotiated independently; alternatively, some systems negotiate suites of algorithms
- Regardless: at the end of the process, the program knows which algorithm to use for which purpose

Hash Functions

- Hash functions are straightforward: they all have the same functionality
- They do differ in output blocksize; that's easily paramaterized

Public Key Encryption

- Public key algorithms all use key pairs; however, keys may be composed of several elements
- Example: RSA public keys are the pair $\langle e, n \rangle$; depending on implementation choices, the private key may be the pair $\langle d, n \rangle$ or the triple $\langle d, p, q \rangle$
- Post-quantum algorithms may require more complex keys

Digital Signatures

- Some algorithms, e.g., RSA, provide “message recovery”—when you do the public operation, you get back the original message
- Example: if Alice sends $\{m, D(S_A, H(m))\}$, Bob would calculate $H(m)$ and $E(P_A, D(S_A, H(m)))$ and see if the two matched
- For other algorithms, $H(m)$ is not the result of signature verification; you have to do a more complex calculation

Integrity Checks and Ciphers

- Integrity checks can be done as a separate operation, e.g., HMAC, or they can be part of encryption with the proper mode of operation
- Make sure your design is flexible enough to handle either
- In the past, this was a problem for negotiation—the concept of a combined mode did not exist

Questions?

(Red-bellied woodpecker, Morningside Park, December 1, 2019)