**Unit 3: Key Distribution**

[ draw title ]

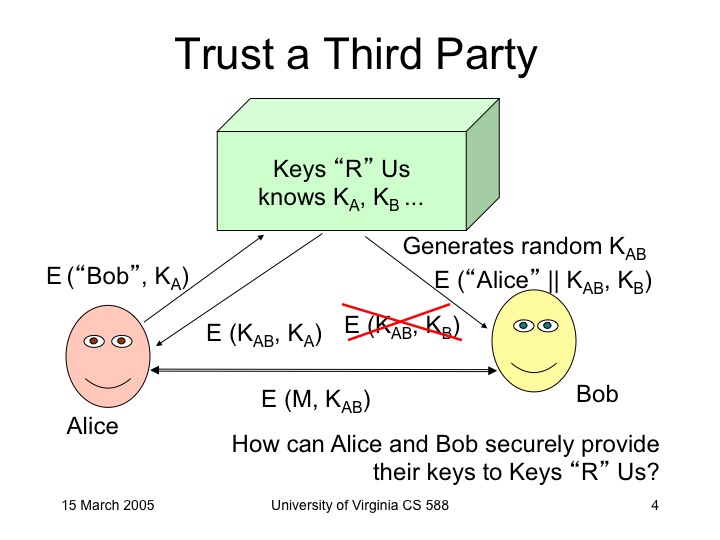
In Units 1 and 2 – learned about symmetric ciphers and how to use them. Symmetric ciphers assume there is a way for both parties to know the key – in some scenarios that is easy: for the encrypted file example, the same party is encrypting and decrypting the file so they can create and keep the key.

[ draw two parties sending messages ]

For sending messages, though, we need a way for the sender and recipient to agree on a key beforehand. In military applications, this can often be done by physically distributing a key. In the days of Colossus, this was done using a physical codebook that would list the keys for each day. Captains were under strict orders to destroy these codebooks before their ship is captured – highest priority!

For civilian applications, in the days of the Internet, physically distributing a key is not an acceptable solution. In this unit, we’re going to look at some ways two parties might establish a shared, secret key.

[ Solution #1: Use a Trusted Third Party ]



Alice Keys “R” Us Bob

knows K\_A, K\_B

(m1)

E\_K\_A(“Bob”) =>

t = D\_K\_A(m) = “Bob”

k\_AB <- {0, 1}^n

<=(m2) (m3)=>

E\_K\_A(“Bob:” || k\_AB) E\_K\_B(“Alice: ” || k\_AB)

Quiz: What could go wrong?

[x] Keys “R” Us can read all messages between Alice and Bob.

[x] Keys “R” Us can impersonate any customer.

[-] Colleen could intercept m1, change it to E\_K\_A(“Colleen”). KRU would generate K\_AC and send it to Alice and Colleen, and then Colleen would be able to read messages Alice intends to send to Bob.

[-] Colleen could tamper with m3, changing the key between Alice and Bob into a key Colleen knows.

This would only work if everyone really trusts KRU completely! The other big issue is it hasn’t really solved the key distribution problem – we still need to share K\_A with Alice and KRU, and K\_B with Bob and KRU, etc.

[ Solution #2 : Merkle’s Puzzles ]

Ralph Merkle, 1974

A

puzzles = []

for i in range(2 \*\* 30):

k\_i <- {0, 1}^s + 0^(n-s) - n-bit key, only s random bits

s\_i <- {0, 1}^n

puzzles.append(encrypt\_k\_i(“Puzzle: “ + str(i) + “ Secret: “ + s\_i))

random.shuffle(puzzles)