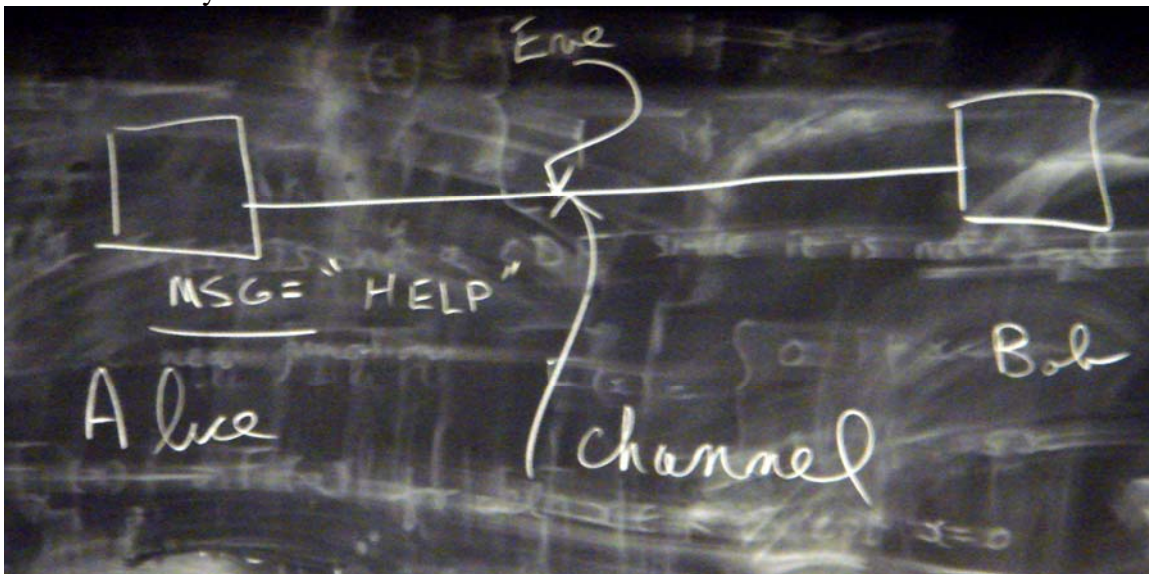


NP-Completeness is bad news. It means our computers are limited. There are many important problems that computers cannot solve quickly if $P \neq NP$. But there is a silver lining and it is in the field of cryptography.

Historically, codes have been incredibly important in wars, politics, banking, etc.

Here is the basic set up. Alice wants to communicate securely with Bob; however, Eve hears everything Alice says. Thus Alice wants to somehow hide her message from Eve. We call "the key" all information that Alice and Bob share but Eve does not.



Different types of codes:

Substitution cipher.

Monoalphabetic Cipher (Caesar Cipher). Represent every letter with the next (or the k^{th} next) letter in the alphabet. There are 26 possible keys. Or just represent every letter with some other letter. Then there are $26!$ possible keys. But it's still easy to break. So a large number of possible keys is a necessary but not sufficient condition for a code that is difficult to break.

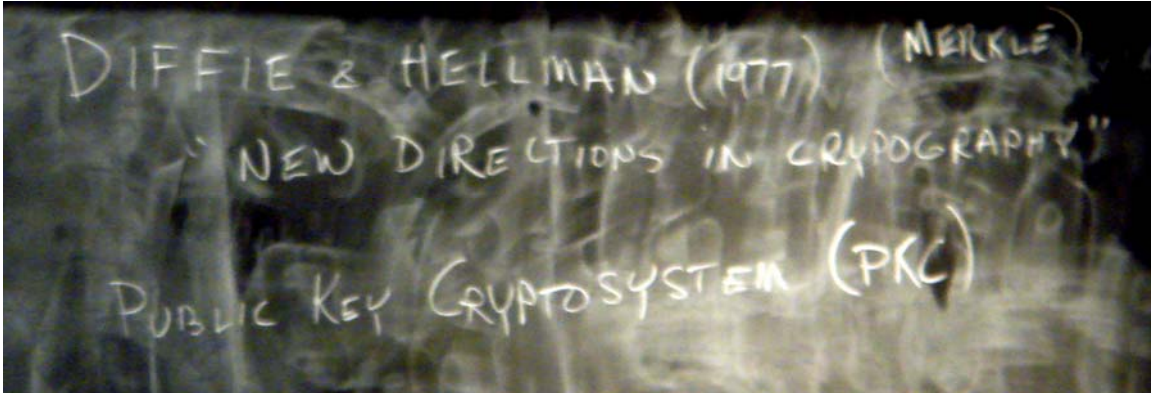
Polyalphabetic cipher (1467). Given a fixed number of alphabets, say m alphabets, represent the i^{th} letter of the message with another letter from the $(i \bmod m)^{\text{th}}$ alphabet. This cipher was also broken, this time by Babbage.

One-Time-Pad. An extension of the polyalphabetic cipher but every letter has a different alphabet. This cipher is provably unbreakable (proved by Shannon).

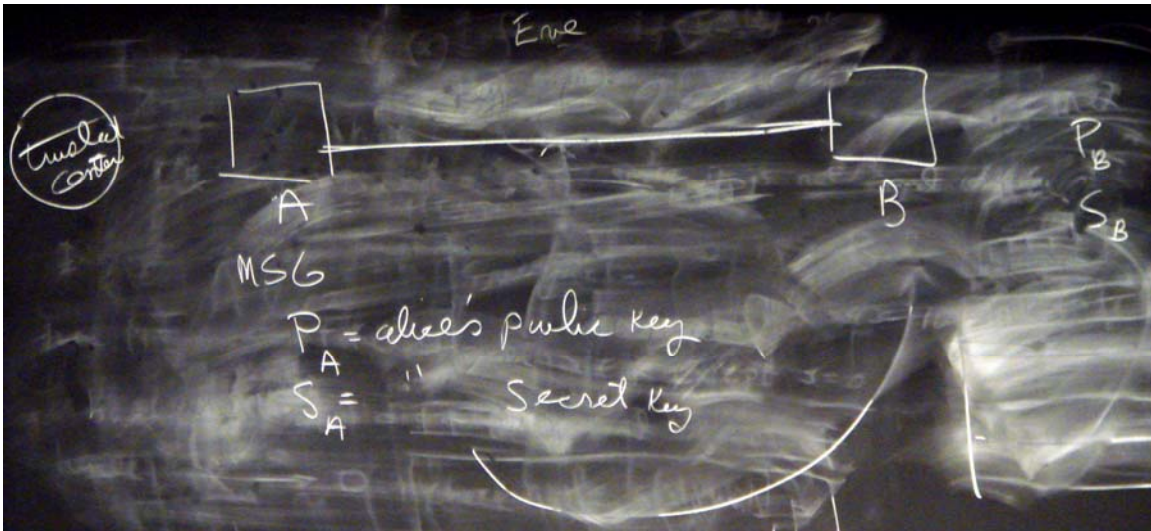
Transposition cipher (permutation cipher). Scramble the letters.

Key distribution problem.

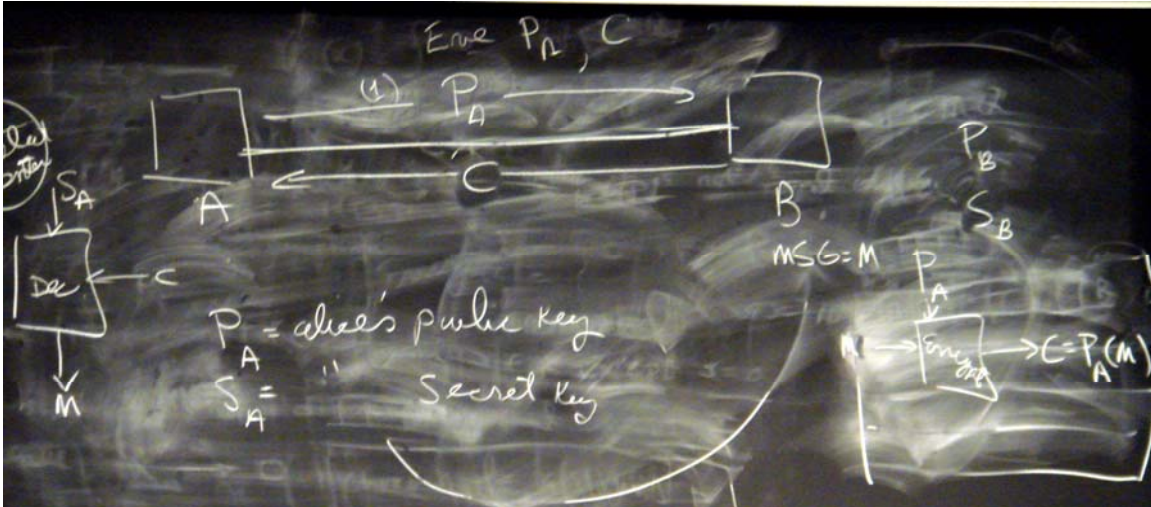
Sending the key over the channel is a problem because Eve gets the key and can decrypt messages. One possible solution is a common trusted center.



Diffie and Hellman (1977) wrote “New Directions in Cryptography” (also working with Merkle). They proposed the idea of a Public Key Cryptosystem (PKC).



Public Key Cryptosystem. Alice has a public key P_A and a secret key P_B . Bob wants to send Alice a message M , so he gets Alice's public key P_A over the channel (Eve hears this key) and uses the publicly known encrypt algorithm, which inputs two arguments, P_A and M , and outputs $C = P_A(M)$, known as a ciphertext. Bob sends C to Alice over the channel (Eve hears C) and Alice uses the publicly known decrypt algorithm, which inputs two arguments, S_A and C , and outputs M . Eve knows P_A and C but not M .



For this system to work we need (informally):

Public and secret keys have to be different.

One's public and secret key must be somehow related.

The secret key should not be easily reverse engineerable from the public key and encryption and decryption algorithms.

The message should not be easily reverse engineerable from the public key and encryption and decryption algorithms.

Problem:

Input: $P_A, P_A(M)$

Output: M

Note that we can try all possible messages, encrypt them using P_A , and compare the result to C . If they match, we found our message. But that algorithm probably doesn't run in polynomial time. If all algorithms that solve this problem run in non-polynomial time, then we're on to something!

More on this next class.