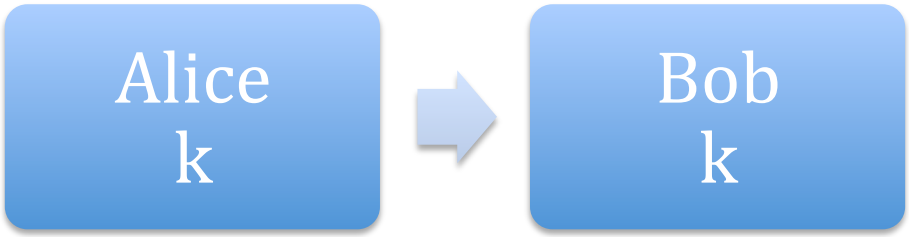
CS 558 Lecture 1-26-2017

Continuation of Encryption

Review:

- **Schemes - how we do encryption and/or decryption**
- **Definition of what it means to be secure(sometimes use to analyze a system)**
- **Proof that the scheme satisfies the definition**

One Time Pad (OTP) for one bit messages is an example of a scheme.



k = key
m = plain text message
c = cipher text
such that
|k| = 1 bit
|m| = 1 bit

The absolute value means length therefore the length of k is 1 bit.

Remember:

k	m	k ⊕ m
0	0	0
0	1	1
1	0	1
1	1	0

Encryption:
$Enc_k(m) = k \oplus m$

Decryption:
$Dec_k(c) = c \oplus k$

Kerckhoff's Principle

"Enemy knows your system" A crypto system should be secure even if everything about the system is public except the secret key.

Ex) in the OTP everything is known (the algorithm of encryption and decryption) expect the key.

We should make systems and expect the system to be secure for this fact, it should be secure with only the key hidden.

In fact if you should want everything about the system to be made public because

1) we don't want a system's security to rest on the fact that others don't know the system(since the assumption is that someone can figure it out)
2) we want experts to test the system for vulnerabilities in order strengthen the scheme.
3) If respected cryptographers fail to decrypt the schemes, this means the system will be widely considered to be secure

If we use OTP "twice"

ie) $c_1 = k \oplus m_1$

$c_2 = k \oplus m_2$

If I want to use the OTP to encrypt an n bit message

$m = m_1, m_2, \ldots..m_n$

↑ the 1st bit ↑ the nth bit

You should choose an n-bit random key

$k = k_1, k_2, \ldots\ldots.. k_n$

$c = (m_1 \oplus k_1, m_2 \oplus k_2, \ldots\ldots m_n \oplus k_n)$

if we use OTP "more than once"

$c_1 = k \oplus m_1$

$c_2 = k \oplus m_2$

.

.

.

.

$c_n = k \oplus m_n$

Brute force – trying every possible key

Consider a brute force attack on this scheme. How many keys do you need to try?

Answer: only 2 tries

Also: what happens if $c_1 \oplus c_2 = (m_1 \oplus k) \oplus (m_2 \oplus k)$?

$= m_1 \oplus m_2 \oplus k \oplus k$ since $k \oplus k = 0$

$= m_1 \oplus m_2 \oplus 0$

$= m_1 \oplus m_2$

Why is this a problem?

1) The key is eliminated which is where the randomness should be coming from
2) Since the messages have structure, it is vulnerable to attacks using
   - knowledge about the plaintext
   - frequency analysis

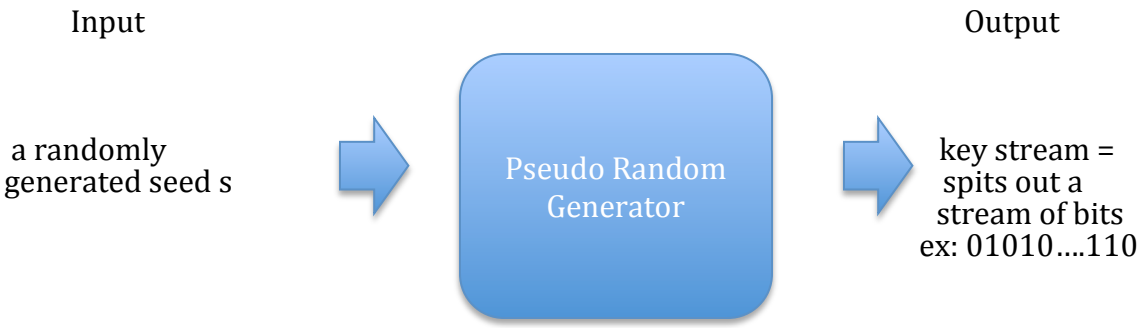
ex) If we know that a message will always start with a 0, you can use the information you do have, to solve the rest of the message that is unknown. There is a good example of this in the film The Imitation Game, when they realize the same words are used at the end of every message.

OTP is not used in practice because you could not feasibly have enough storage to secure all the forms of communication.

Most schemes that are used in practice are similar to OTP and shift ciphers.

Stream Cipher

Let's define s as the seed.



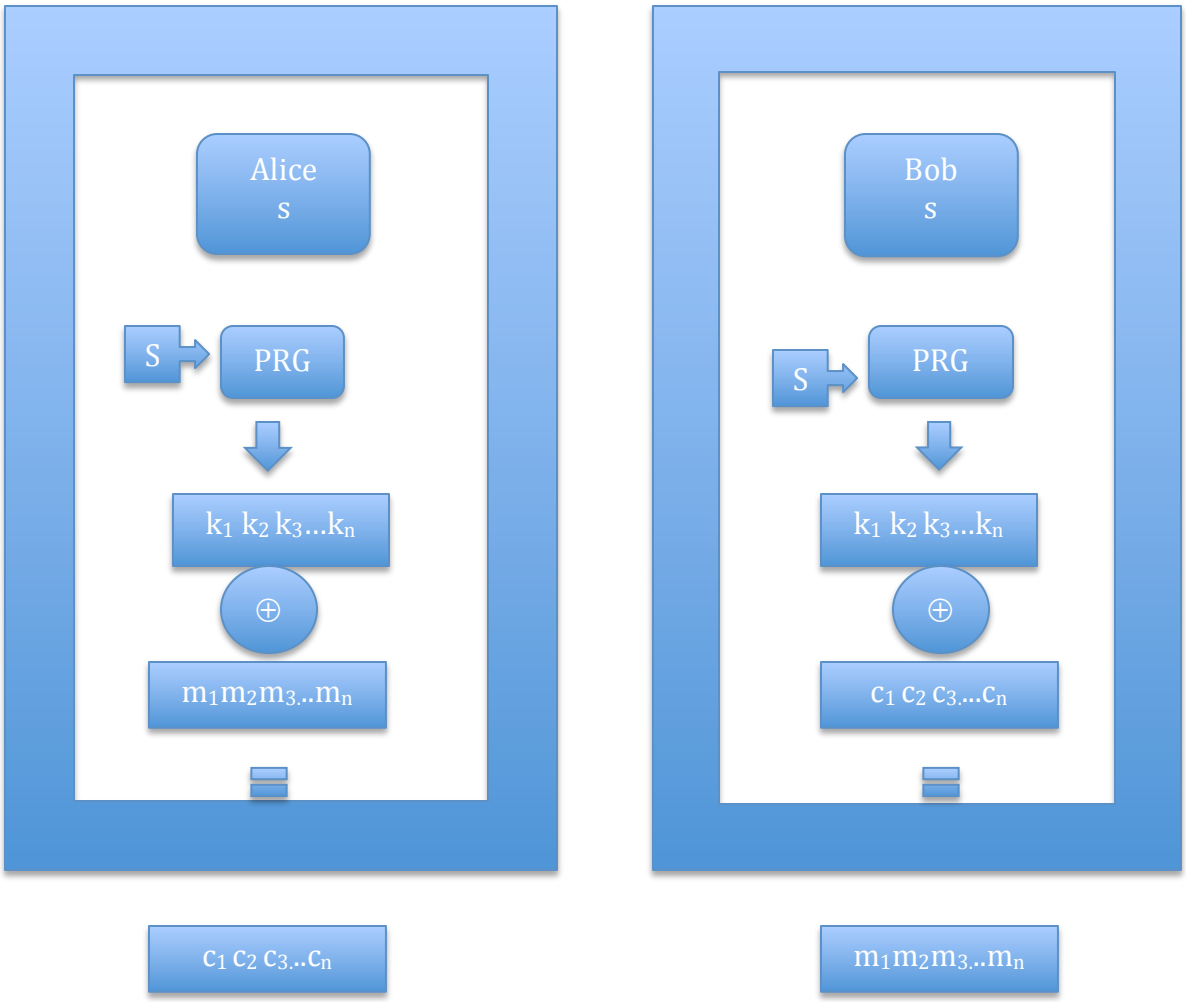
Uniformly random 128 bit string
(note: there are 2^{128} possibilities for S, which are all equally likely)

a short seed is passed to the pseudo random generator (PRG) and it outputs a long key stream of what seems like a randomly order bits.

What we can do is label the stream of bits
0 1 1 0 1 0….
$k_1\ k_2\ k_3\ k_4\ k_5\ k_6$

and then the assign the appropriate message bit to the appropriate key stream bit. And once given the key stream, you essentially used the OTP scheme to encrypt the entire message.

$c_1 = k_1 \oplus m_1$
$c_2 = k_2 \oplus m_2$
:
etc



The PRG algorithm is deterministic therefore a stream of s will be the same every time. The randomness comes from S no the stream the PRG algorithm produces. Some well known PRG's are : RC4, Salsa, Chacha

If the key stream is to leak the scheme would also fail.

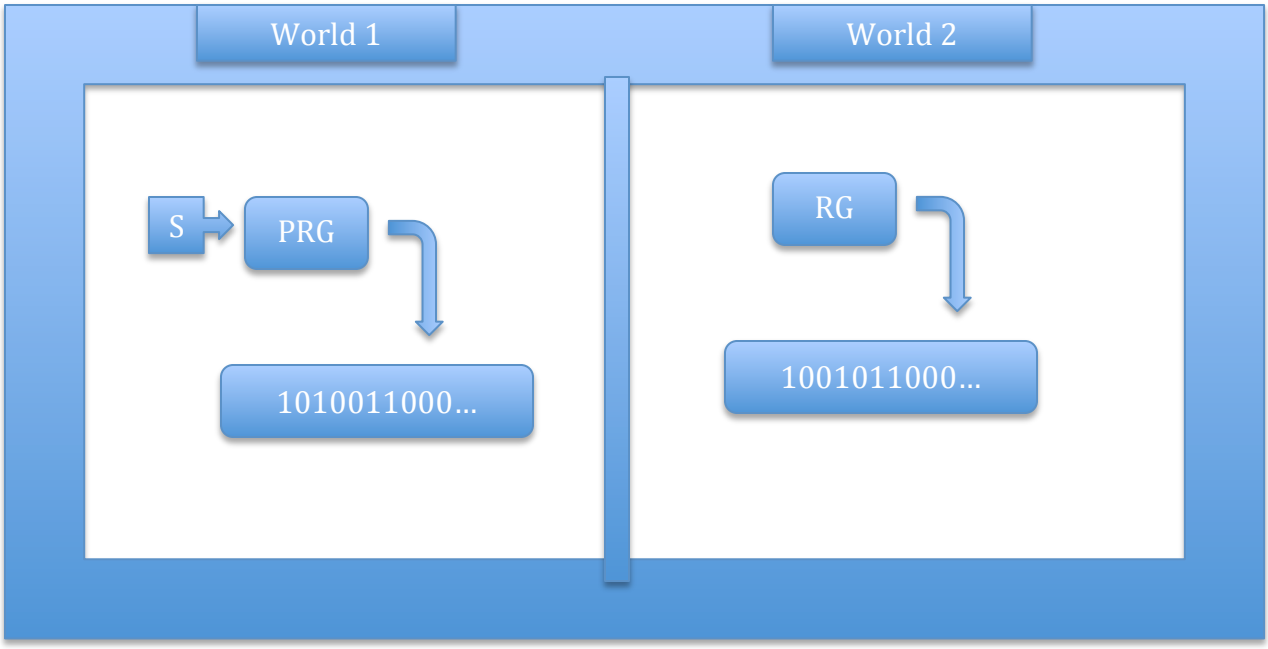
What does it mean for something to look random?

Random VS Pseudo Random

S is assumed to be produced randomly. However the key stream that the PRG produced based on S is pseudo random meaning that it is indistinguishable from a randomly produced bit stream. With knowledge of the seed, the key stream does not look random because you can just plug it into the PRG to recover the key stream. Outputs only look random if s is random and unknown.
(note: We assume key agreement between Alice and Bob before analyzing schemes)

Attacks on stream ciphers are on the weaknesses within Alice's or Bob's box. We must never reuse the same seed and therefore key stream meaning it is a stateful.

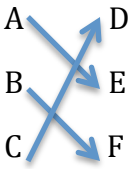
Security Definition: A Good PRG is secure if for all polynomial time adversaries D play the following game. There are two worlds both of which D can see the key stream of bits. However they need to figure out if they are in the world where they have the PRG or if they have the RG. If D cannot distinguish between which world they are in, this is a good PRG.



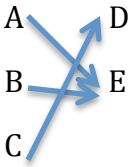
Block Cipher

Pseudo random permutation instead of a stream, it's a permutation.

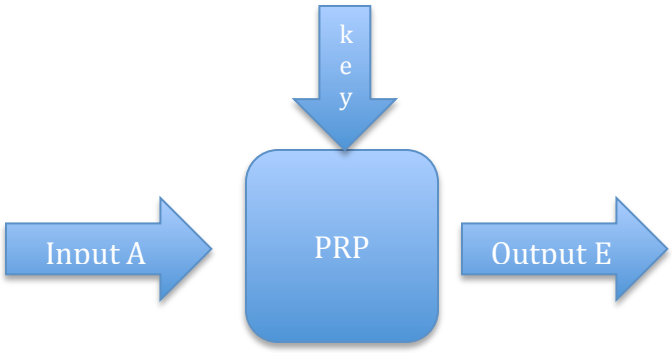
Permutation – n! where all outputs have a distinct and uniquely associated input. For example the following is a permutation

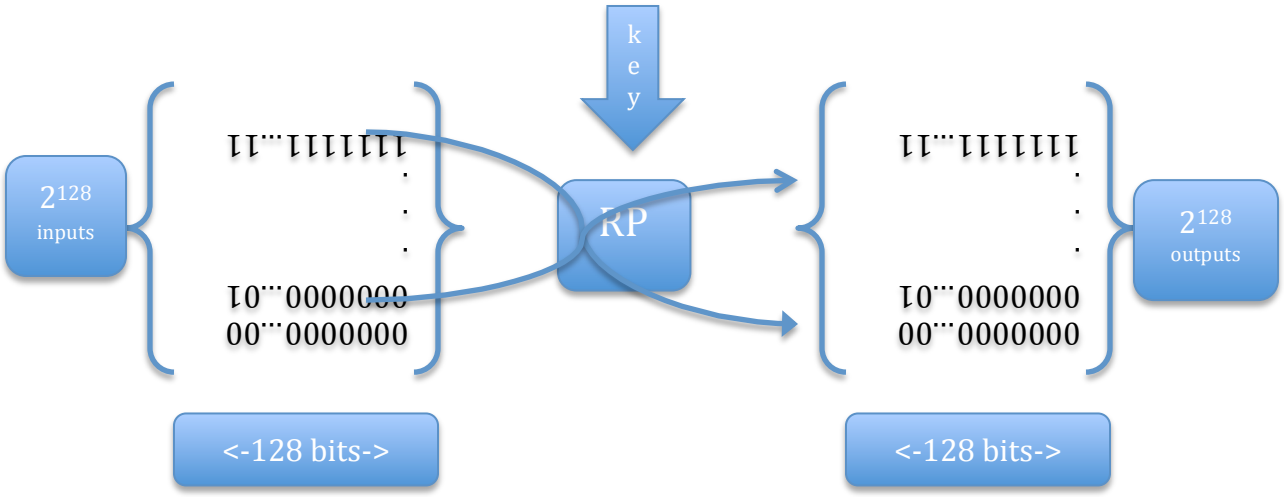


But the following is NOT a permutation because E has both A and B as possible inputs.



So permutation of size 4 is 4!





Q: How many random permutations do we have here?
A: $2^{128}!$