Cryptography 1- Dan Boneh (Stanford)

# Week 1:

## Introduction:

Fundamentally, the first part of the course is going to focus on symmetric key crpytosystems and the idea of how the key exchanges happen is discussed later. Within the scope of Symmetric Key Cryptosystems, there is the single use key where a new key is generated to encrypt every message (or whatever) in contrast to the multi use keys where the same key is used to encrypt multiple things. This is useful in things like file systems where you don’t want to use multiple keys to encrypt many files in a given directory. However, care must be taken to ensure that this doesn’t make it less secure. (There are more steps to this apparently)

## Types and Areas of Cryptography:

Bunch of areas dealt with at a very high level.

Most interesting thing was this point where he talks about a 3rd party or a trusted authority and claims that there exists a theorem that states that “ Everything that can be done with such an authority can also be done without the authority”

Other interesting areas are Zero Knowledge Proof of Knowledge and Privately outsourcing Computation (where you compute on encrypted data)

Three Steps of Cryptography:



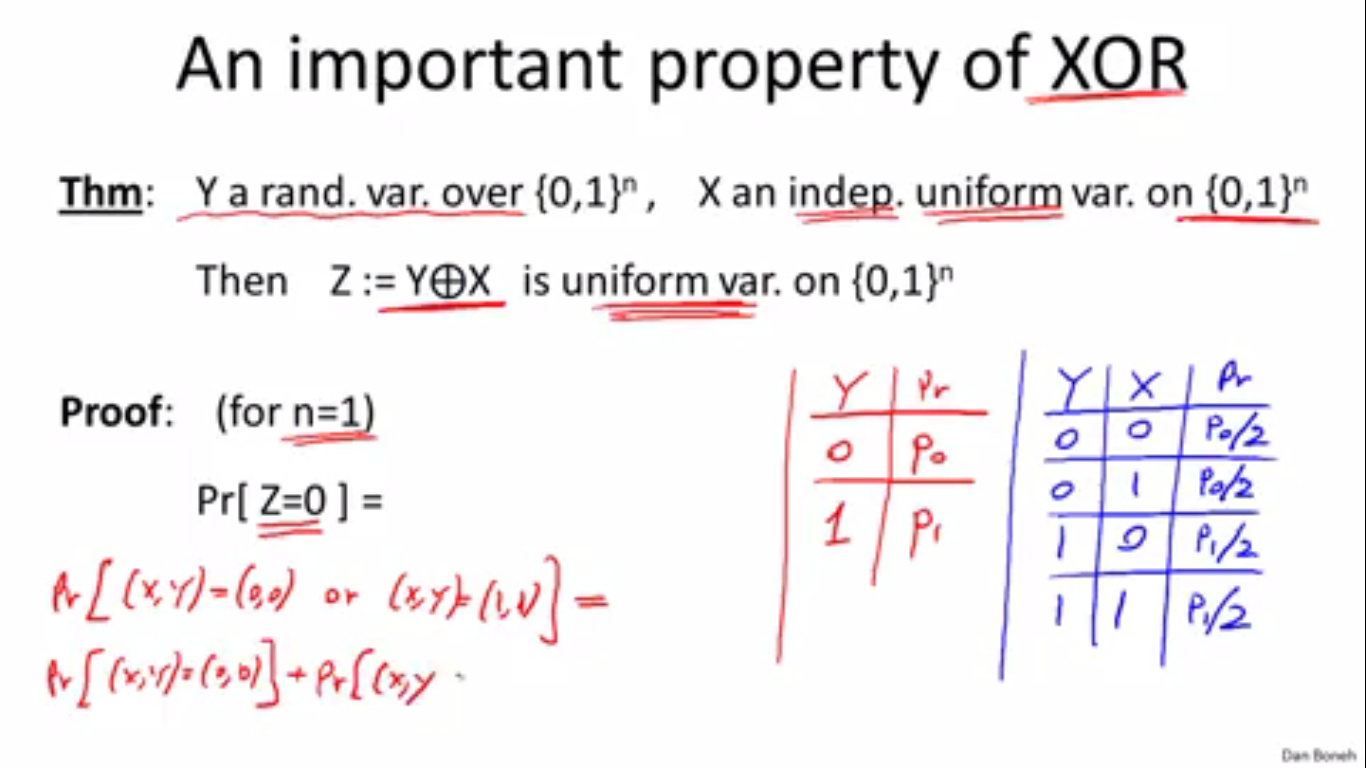
## History of Cryptography:

At 15:19 he argues that Hebern machines and early rotor machines were broken by frequency analysis. This is because after 26 letters, the substitution table resets itself. So the frequency matching will hold on sets of lettes like (1,27,53 etc and 2, 28, 54 etc). You would need a lot of ciphertext, but still easily doable.

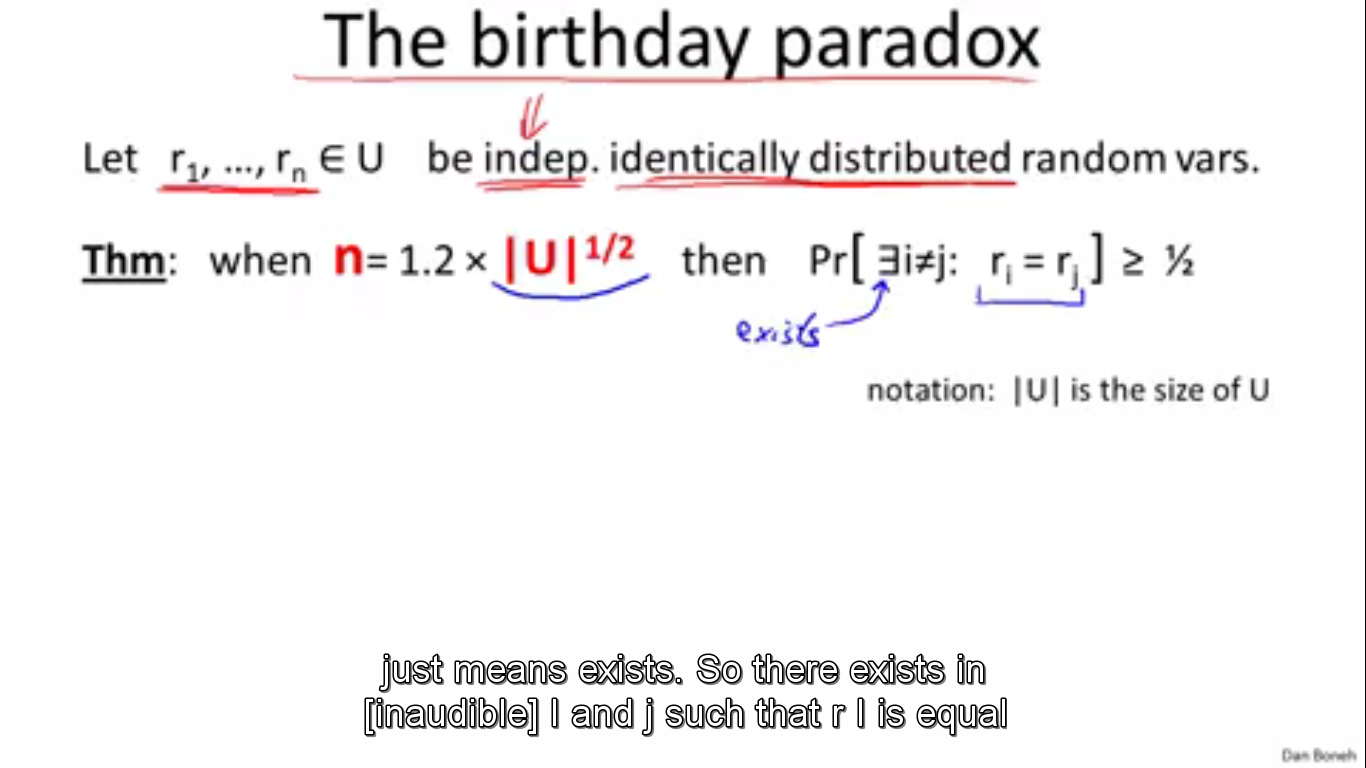
Question: Does this mean that these substittion ciphers etc are secure in cases where you do not know anything about the plaintext, including the language etc. What non empirical knowledge can be used to break these?

DES is considered weak before 2^56 is considered too small a key space and it can be brute forced. This is insightful of the world of cryptography in general. You are not that concerned with efficiency as long as you can break it. (If a specialised machine can be built for about 200000 $ and takes a few days, this is considered VERY VERY WEAK) Ref: <http://www.freeswan.org/freeswan_trees/freeswan-1.5/doc/DES.html>

## Discrete Probability:



This theorem of XOR. Search for a more rigorous proof.



General theorem of the birthday paradox

## Information Theoretic Security and the One Time Pad:

For symmetric ciphers, there is what is called the consistency equation that every symmetric cipher must satisfy.

**D(k, (E(k,m))) = m**

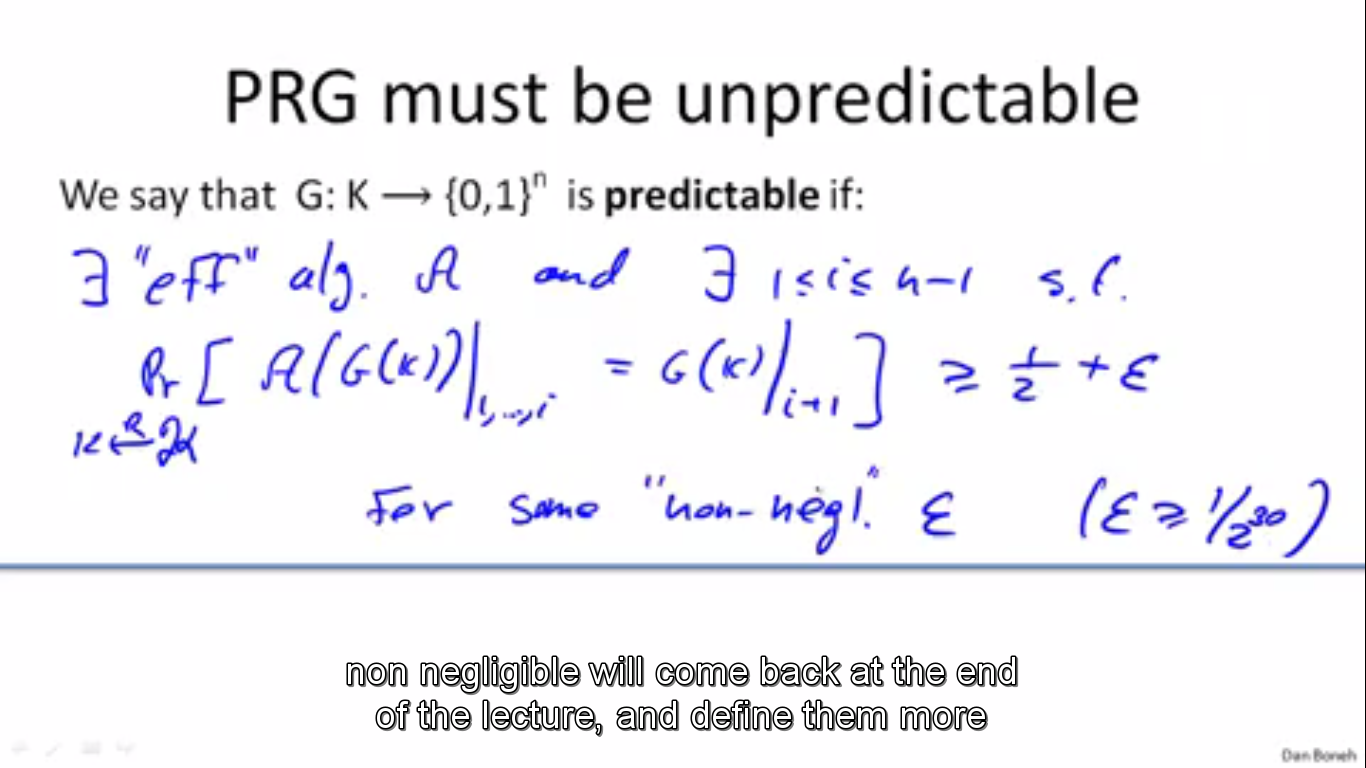
No proof of theorem that any cipher satisfying perfect secrecy will have Key length >= message length.

Also, note that perfect secrecy only means that it is safe from cyphertext only attacks. It is apparently possible to break the One Time Pad, with other attacks.

The above lemma renders the One Time Pad impractical for everyday use. As a result, we must move to stream ciphers and to a new definition of security (since the OTP is the most efficient cipher that satisfies perfect secrecy)

## Stream Ciphers and Pseudo-Random Generators:

One necessary property for a generator to generate a secure cipher is for the PRG to be ‘**unpredictable’**



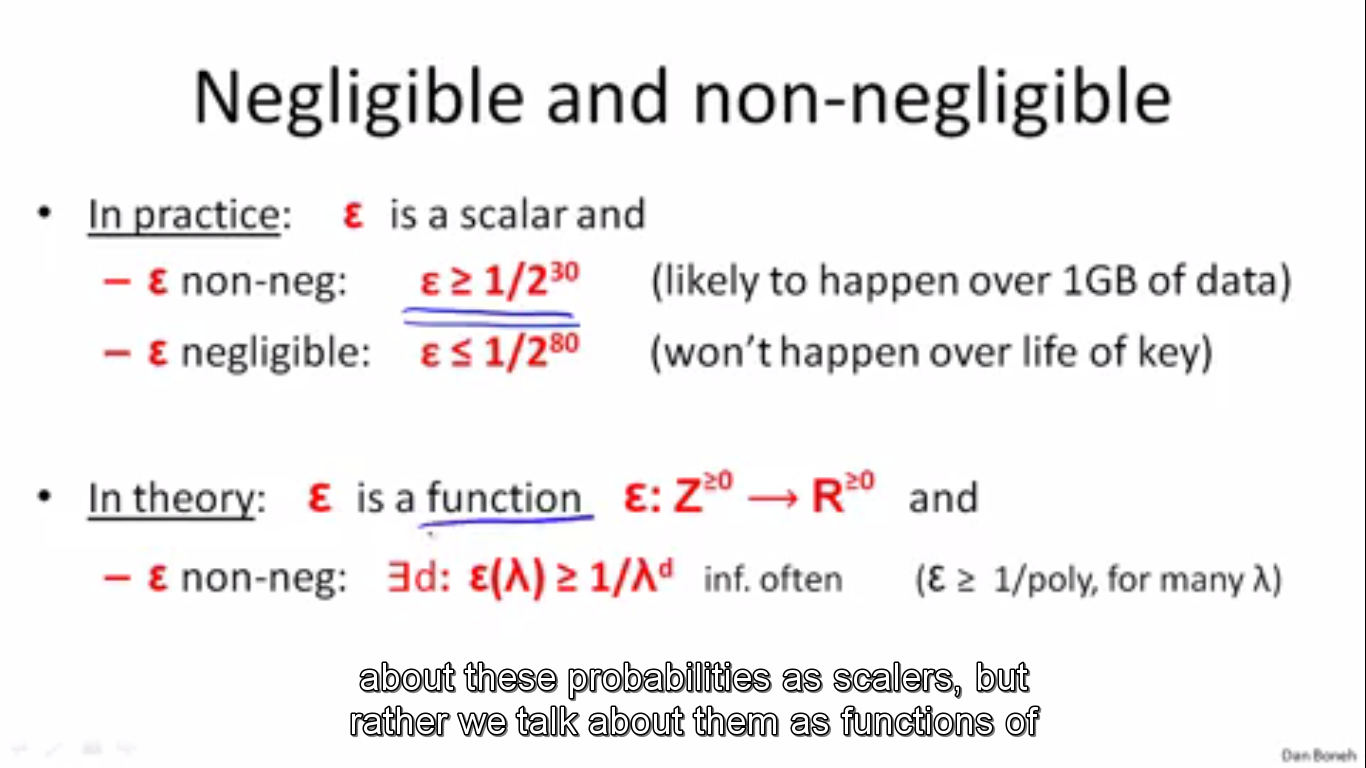
Definition of predictability

Why is unpredictability important?

If a PRG is predictable, then given the first ‘I’ bits of the key, the next bits (or even bit) can be recovered. Now, let us say that for a message encrypted with a key from such a PRG, an attacker intercepts the ciphertext. Based on some knowledge of the plain-text, like a default header etc, they can easily get the corresponding bits of the key.

With this and the predictability property, they can get the rest of the key, and thus decrypt the entire message.

Weak PRGs which are predictable exist, like Linear Congruential Generator and the random function in the glibc library. These are extremely easy to predict (although I have no idea how to go about doing that. Should look at that) and so should never be used for crypto.



So the theoretical definition of neglible is rather similar to the big oh used when talking about functions. There are apparently problems with absolute values (although I am guessing they are more practical and useful when talking about a specific protocol)

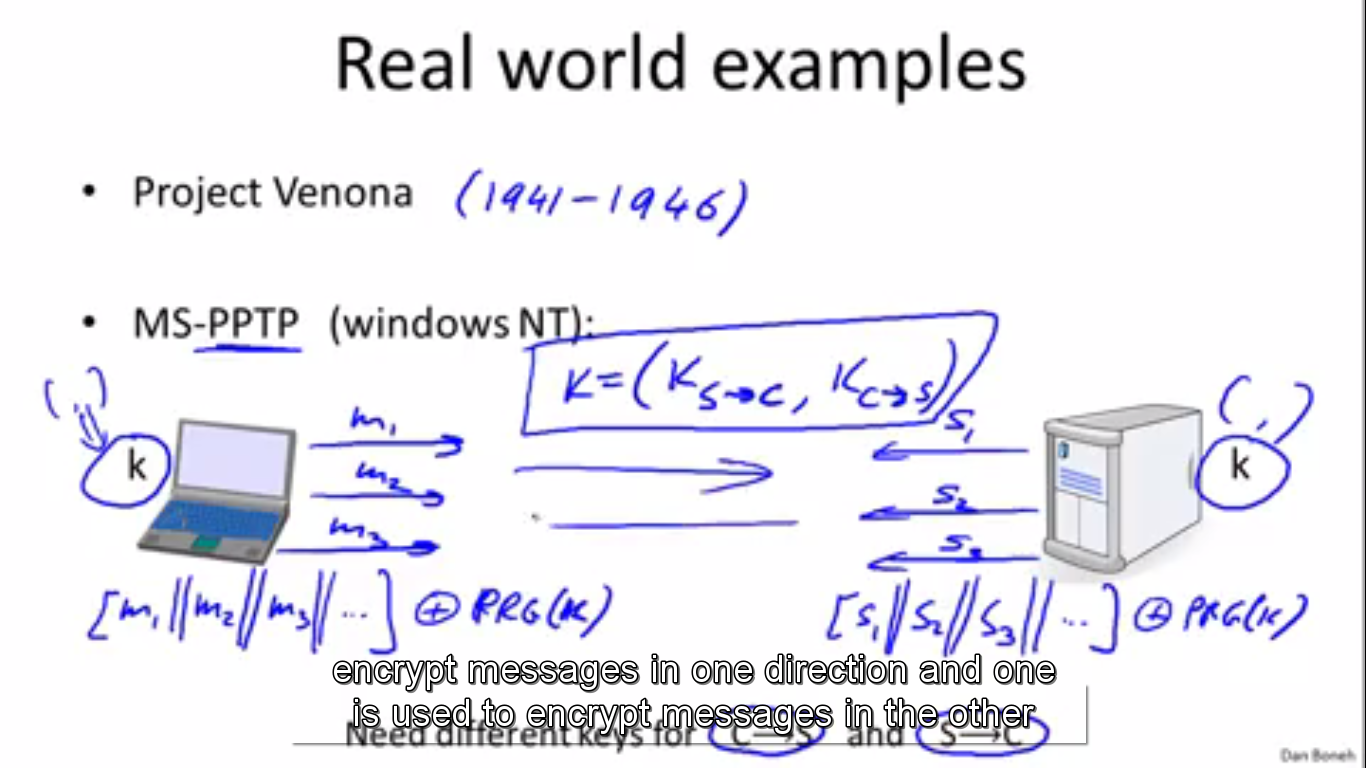
## Attacks on OTP and Security of Stream Ciphers:

### Many Time Pad:

This is a common attack. By XORing the two ciphertexts, the keys cancel out, leaving the xor of the plaintext. Using redundancy of the English alphabet, this can be broken.

Details: <https://crypto.stackexchange.com/questions/2249/how-does-one-attack-a-two-time-pad-i-e-one-time-pad-with-key-reuse>

(How would a non heuristic attack work on this?)



#### Real World Examples:

##### Project Venona:

TLDR: USSR was lazy and so used the same key often. US intercepted lots of messages.

##### MS-PPTP(Windows NT)

They used the same key to encrypt messages from both sides (client to server & server to client)

##### WEP:

Bunch of issues. Two time pad arises after sufficient number of frames. Keys are also very closely related, which makes them insecure.

##### Disk Encryption:

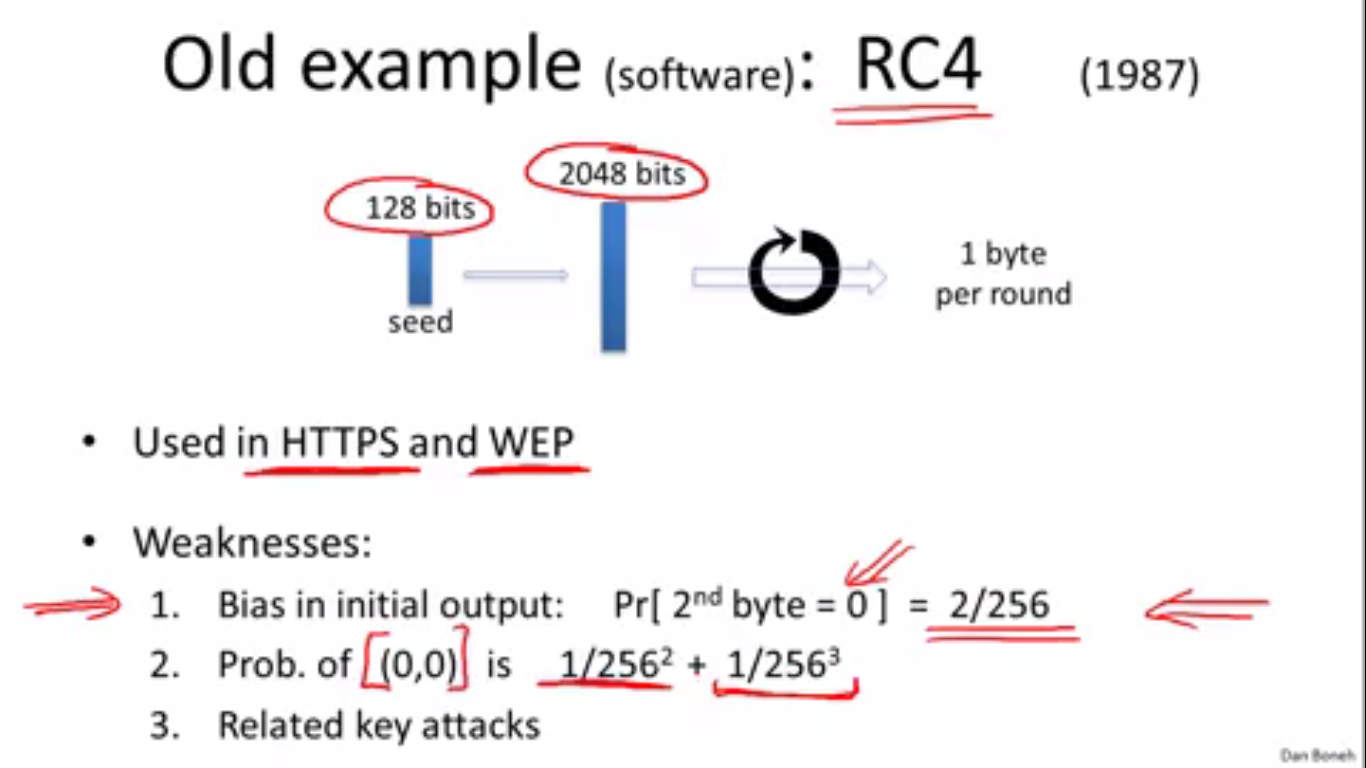
Stream ciphers are generally a bad idea here. If you encrypt a file, and later change it, you have a 2 time pad.

### Lack of Integrity (Malleability)

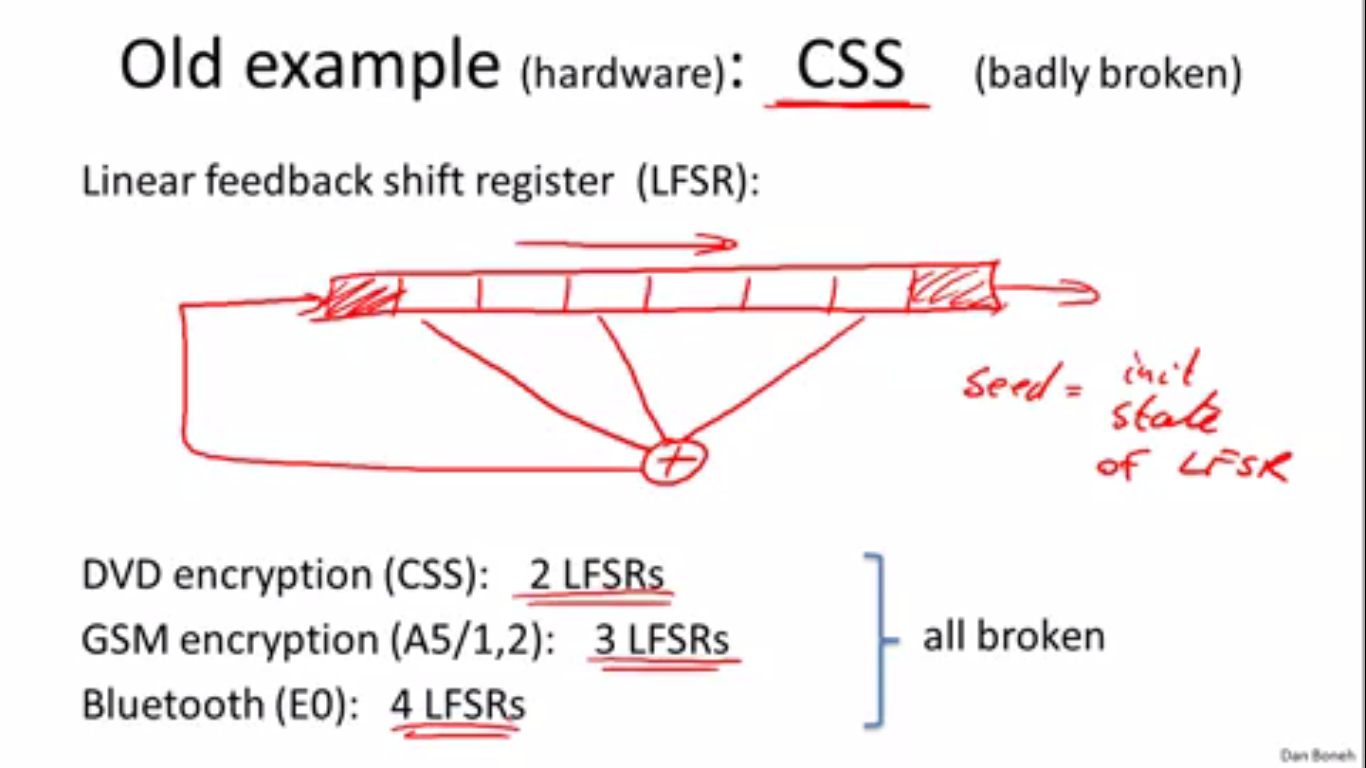
We can modify the plaintext by xoring the ciphertext with some other values. At the very least, you can garble communications a lot (though I don’t think any security protocol can prevent that. Maybe Error Correcting Codes) but you can also (with a little knowledge of the plain-text) cause noticeable changes on the plaintext.

## Real World Stream Ciphers:

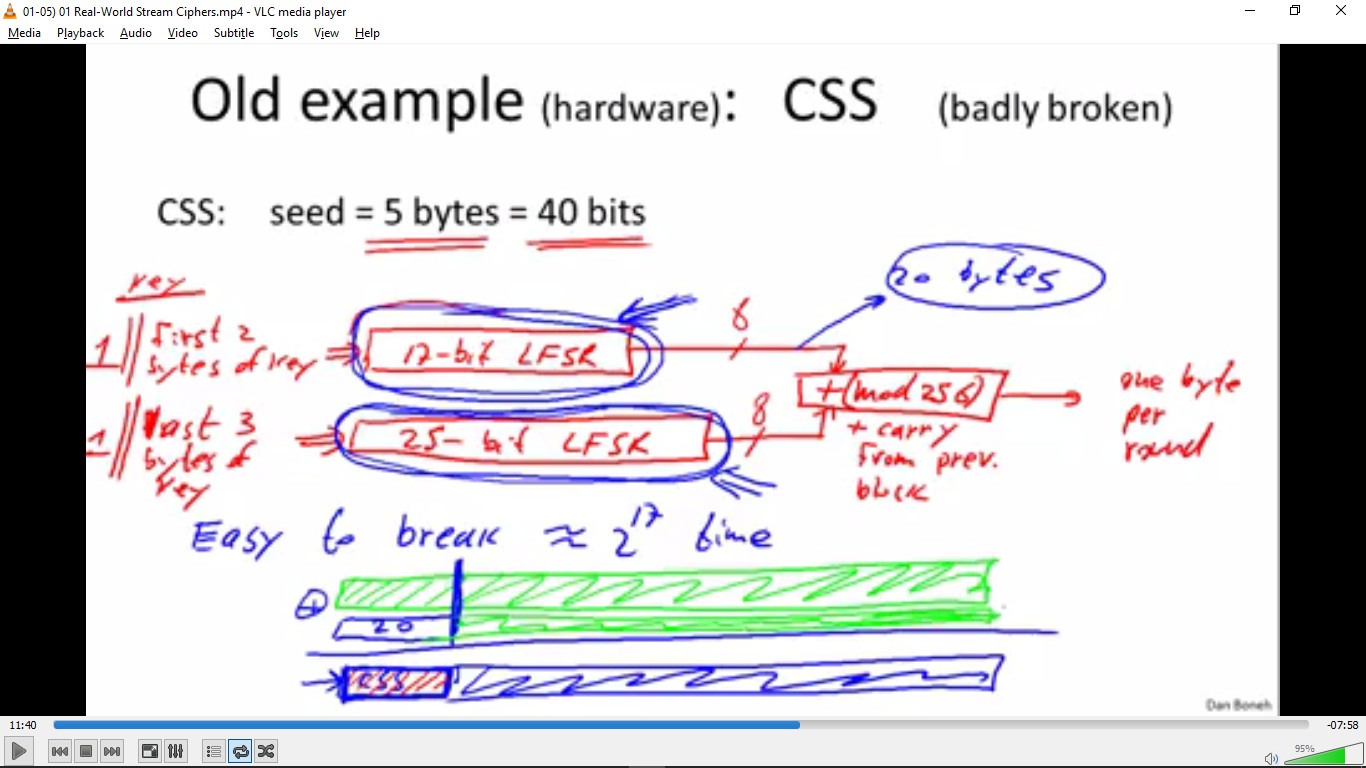
### RC4:



This is still used fairly often. It is advised now that if you use it, you skip the first 256 bytes and go from there. The attacks seem to be heuristic. Has anyone shown a way to exploit that info?



An attack effectively uses some knowledge of the protocol and a brute forcing mechanism to break the encryption in time 2^17.



Recently, 5 stream ciphers and PRGs were approved by the eStream project that ended in 2008. Salsa and Sosemenuk are examples of these and they exploit both software and hardware advantages and are so very fast.

The specific details of the implementation are not particularly important to me.

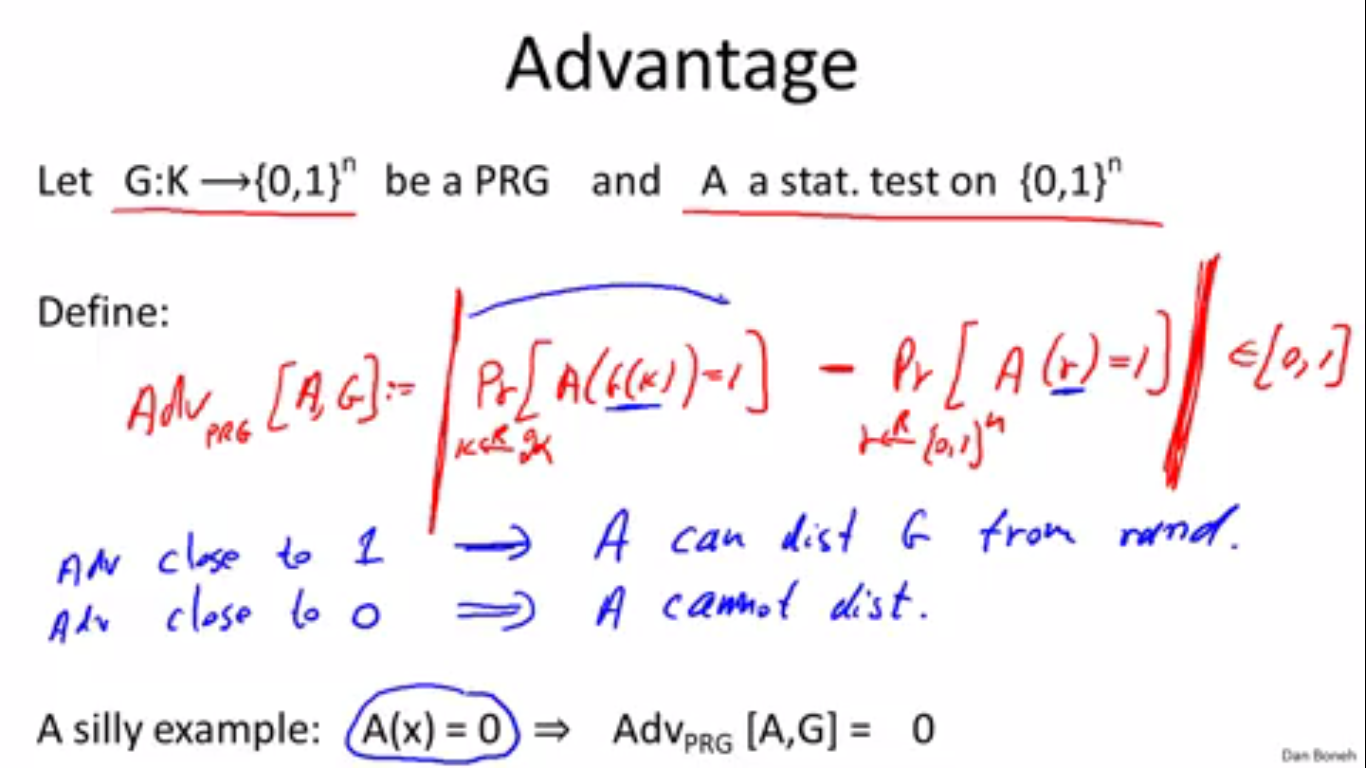
## Defining Security of a PRG:

### Statistical Test:

A statistical test is simply an algorithm that takes an input string and decides whether or not it looks random, based on some property. If it looks random, it returns a 1, if it doesn’t it returns a 0.

My problem here is the idea of some individual string looking random. Given a distribution, we can see if there is a pattern or if there isn’t one we can say it is random. But for a single string, what does randomness mean? Isnt it perfectly possible for a good PRG to output a string like 1111111…111? A part of this suspicion is allayed by the idea that you have a battery of tests and a reasonable string should pass most of them. However, would a PRG be dismissed as bad simply because it returns the all-string?

### Advantage:



This idea of advantage answers my previous question since we look at the probability distribution.

### Definition of Secure PRGs:

A PRG is said to be secure if there exists no **‘efficient’** statistical test that can break it with a non negligible advantage.

Boneh argues that the restriction to efficient statistical tests is necessary. **Why is that?**

To prove that a PRG is secure is equivalent to proving that P != NP. Thus it is not known if any provably secure PRGs exist. We have heuristic candidates which have presumably passed every test we have thought of. ~

#### Property 1: A secure PRG is Unpredicatable:

We prove the contrapositive.

The formal proof is presented in the slides, but the idea is that if a PRG is predictable, portions of it can be guessed with a much higher probability than for a truly random string. Thus a statistical test can be devised which gives it a non-negligible advantage, thereby rendering it insecure.

#### Property 2: An Unpredictable PRG is Secure:

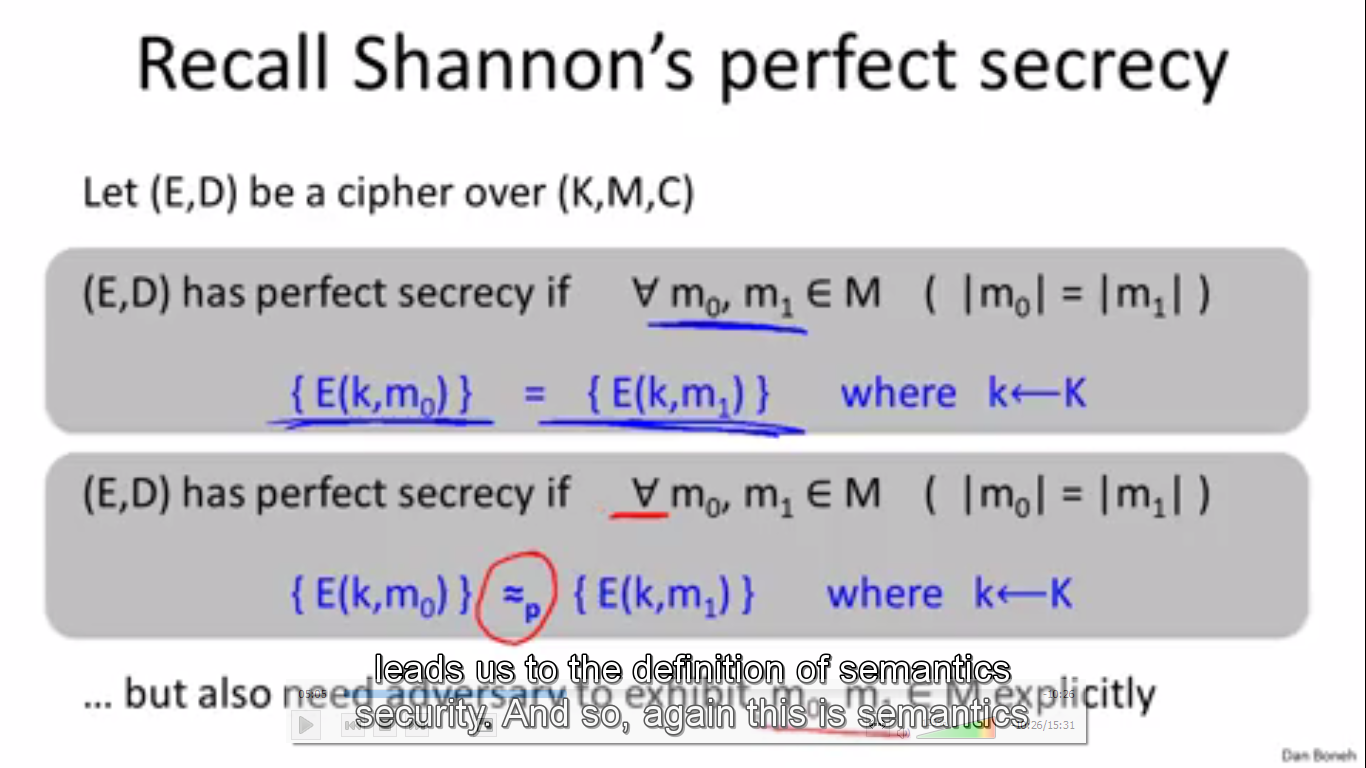
Note: Since it is unknown if truly secure PRGs exist, it stands to reason that no PRG is proven to be unpredictable.

Note 2: The theorem states that the PRG must be unpredictable for all bit positions. (obviously, because otherwise a statistical test can be constructed)

Proof is by Yao in 1982. Not done by Boneh.

## Semantic Security:

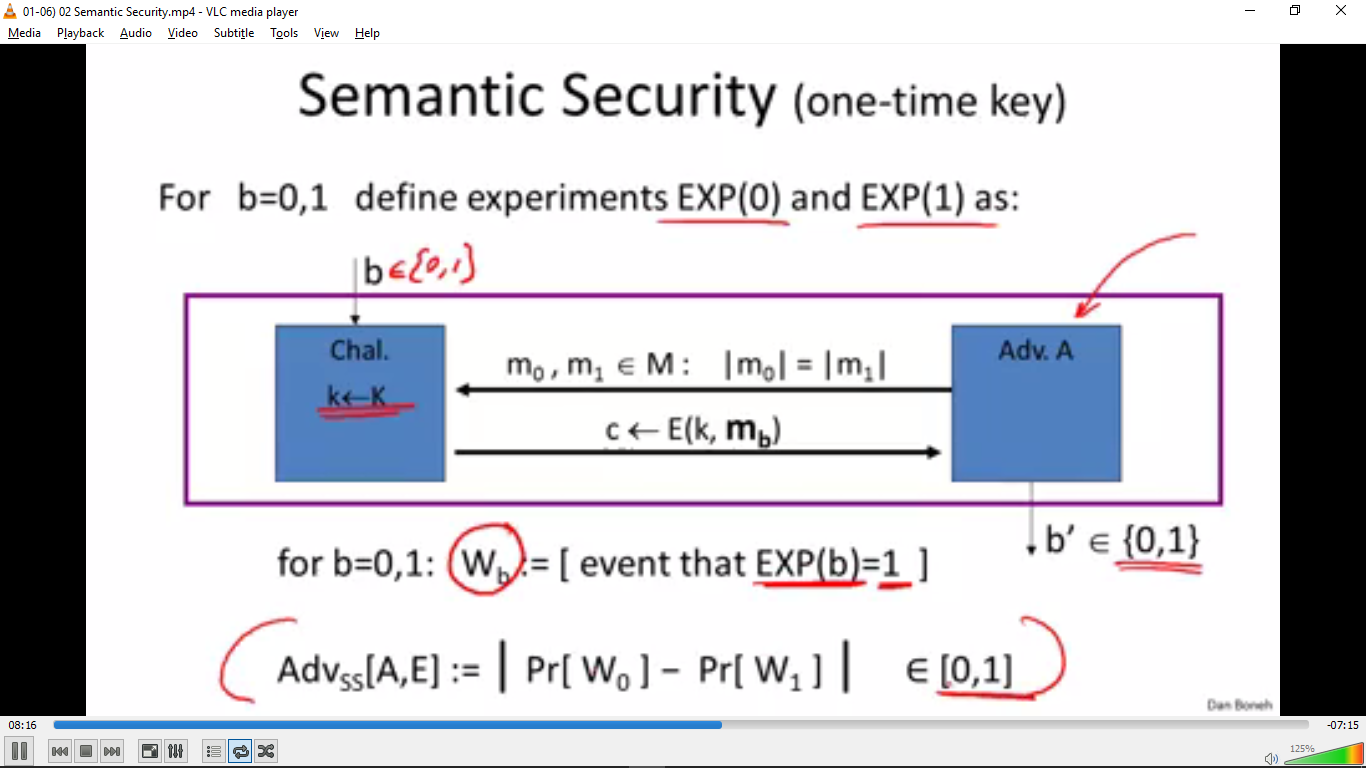
We weaken Shannon’s definition of perfect secrecy so that we can satisfy it with stream ciphers made with PRGs.



We weaken it further to say that this must hold only for pairs of messages that the attacker can exhibit. By this we mean that this needs to hold only for pairs of messages that can be computed by some efficient algorithm. (Boneh-Shoup Textbook <https://crypto.stanford.edu/~dabo/cryptobook/draft_0_3.pdf> : Page 13)

This definition Is for the case of the one time key. This means that all the attacker can do is intercept one ciphertext.

### Formal Definition of Advantage:

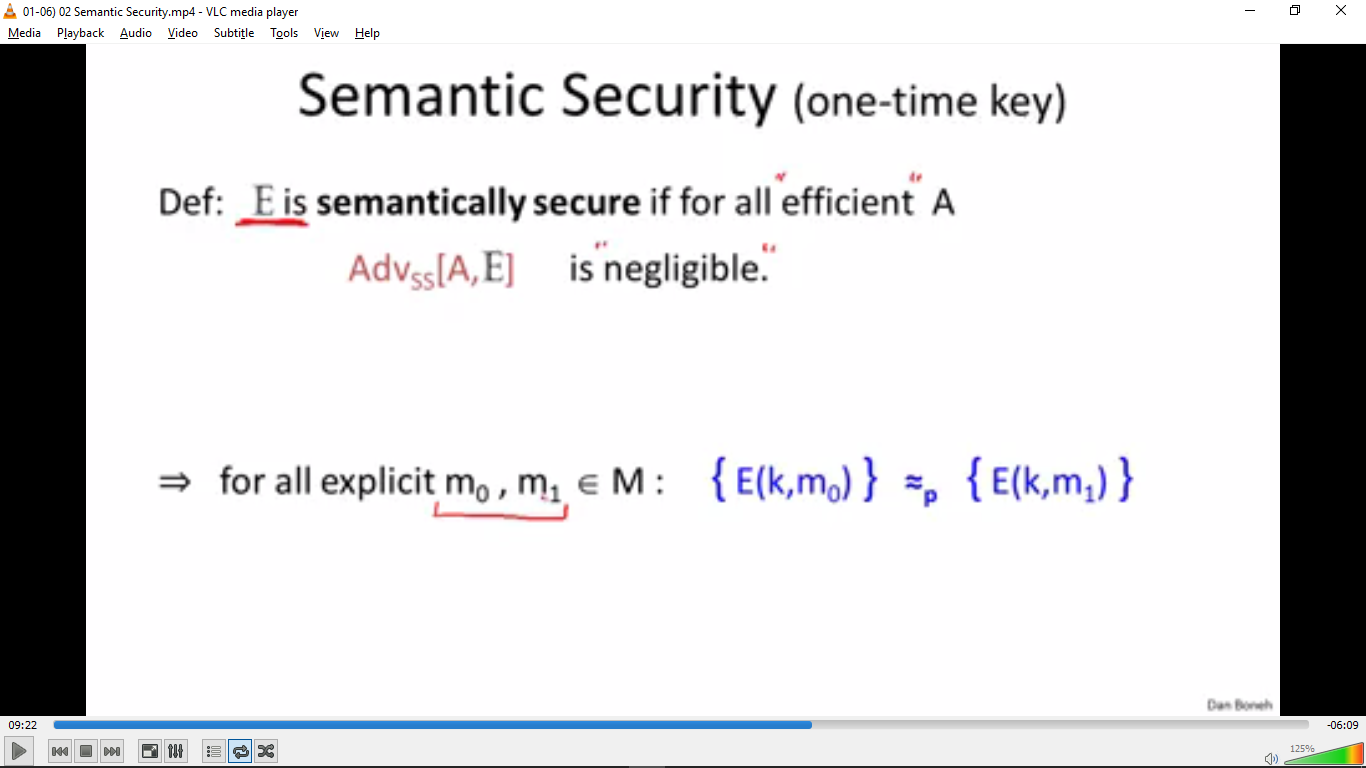


#### Process:

Effectively there are two experiments: Exp(0) and Exp(1). An adversary presents two messages m0 and m1 to the challenger. (equal lengths). The Challenger outputs either the encryption of m0 or the encryption of m1 and the job of the adversary is to guess which message has been encrypted. (Exp(0) encrypts m0 while Exp(1) encrypts m1.)

The rest of the definition is obvious from the image.

### Formal Definition:



## Stream Ciphers are Semantically Secure (with a secure PRG):

#### Proof Idea:

We swap out the PRG key for a truly random key. Since the PRG is secure, the adversary cant really tell that the swap took place. Now since the adversary has no advantage against the OTP, he will have negligible advantage vs the PRG key as well, thereby making the Stream Cipher secure.

#### Actual Proof:

Too hard to type out in Word. Refer to the video. We effectively try to prove that Advantage of the adversary vs the stream cipher <= 2\* Adv vs the PRG. Since the LHS is negligible, we can say that the RHS is also negligible.