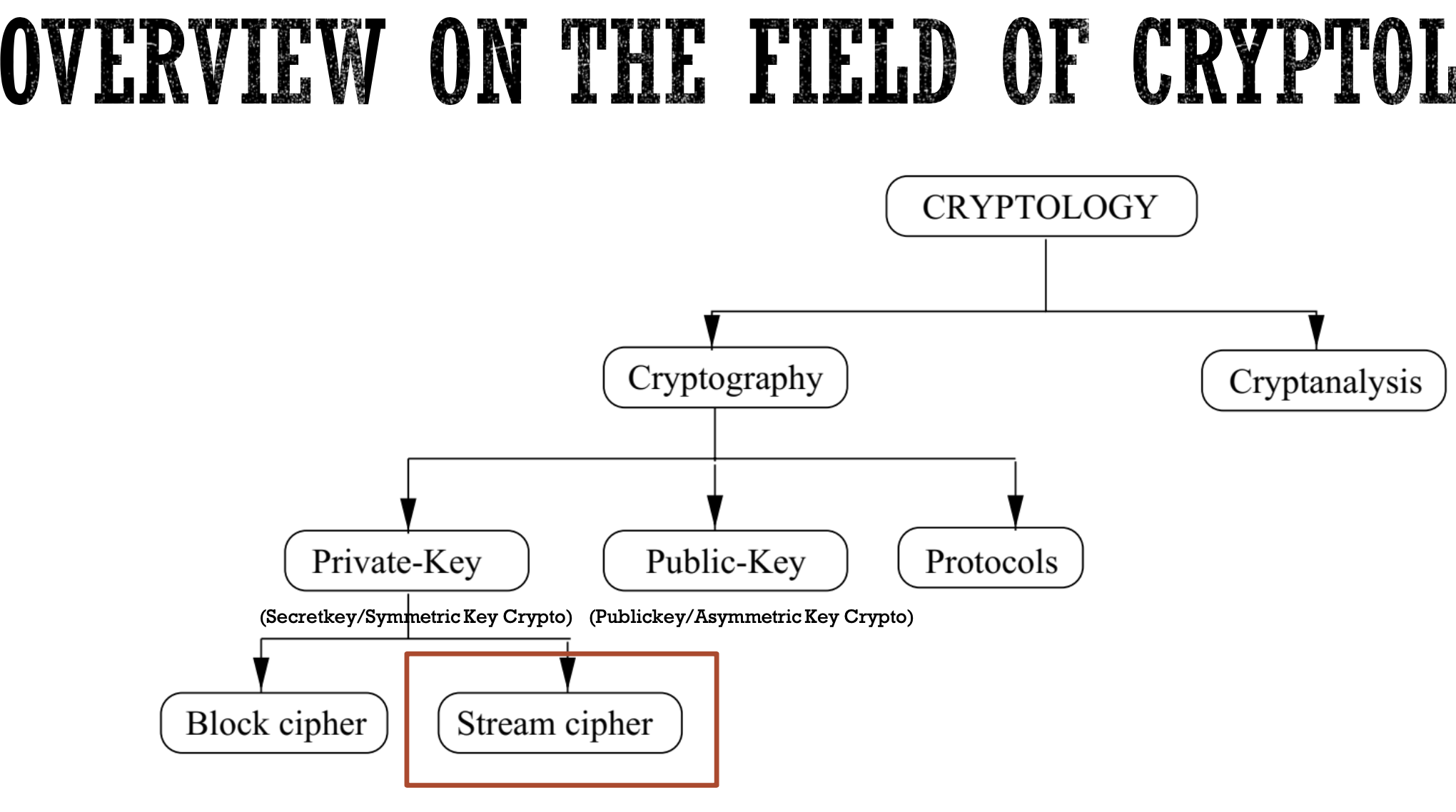
**Class Notes**

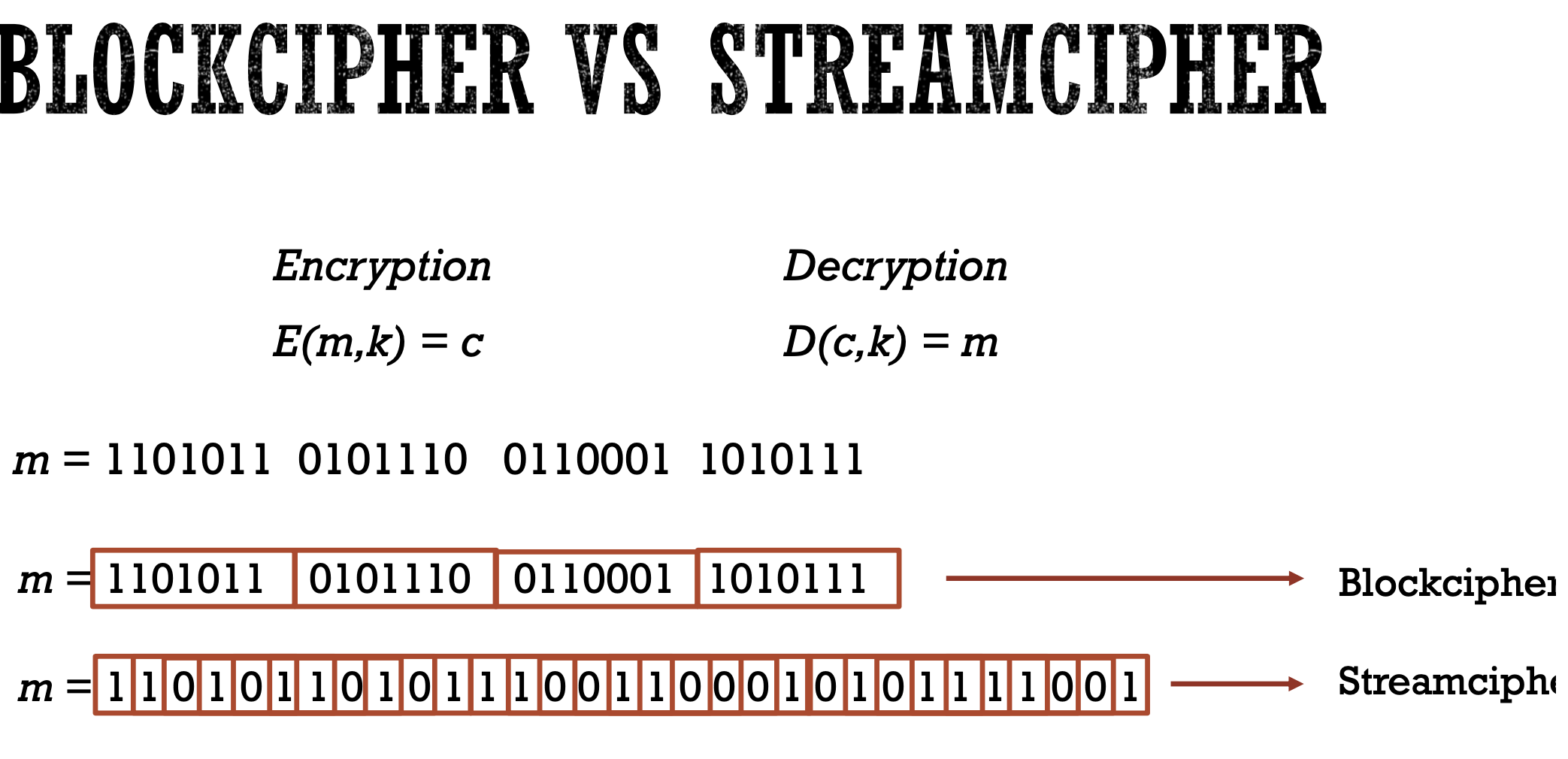
**123456**

**467912**

Both numbers have the same amount of randomness .Even though it looks like the top one has a sequence, in respect to randomness, it does not matter. Both have the same probability.

**Stream Ciphers**





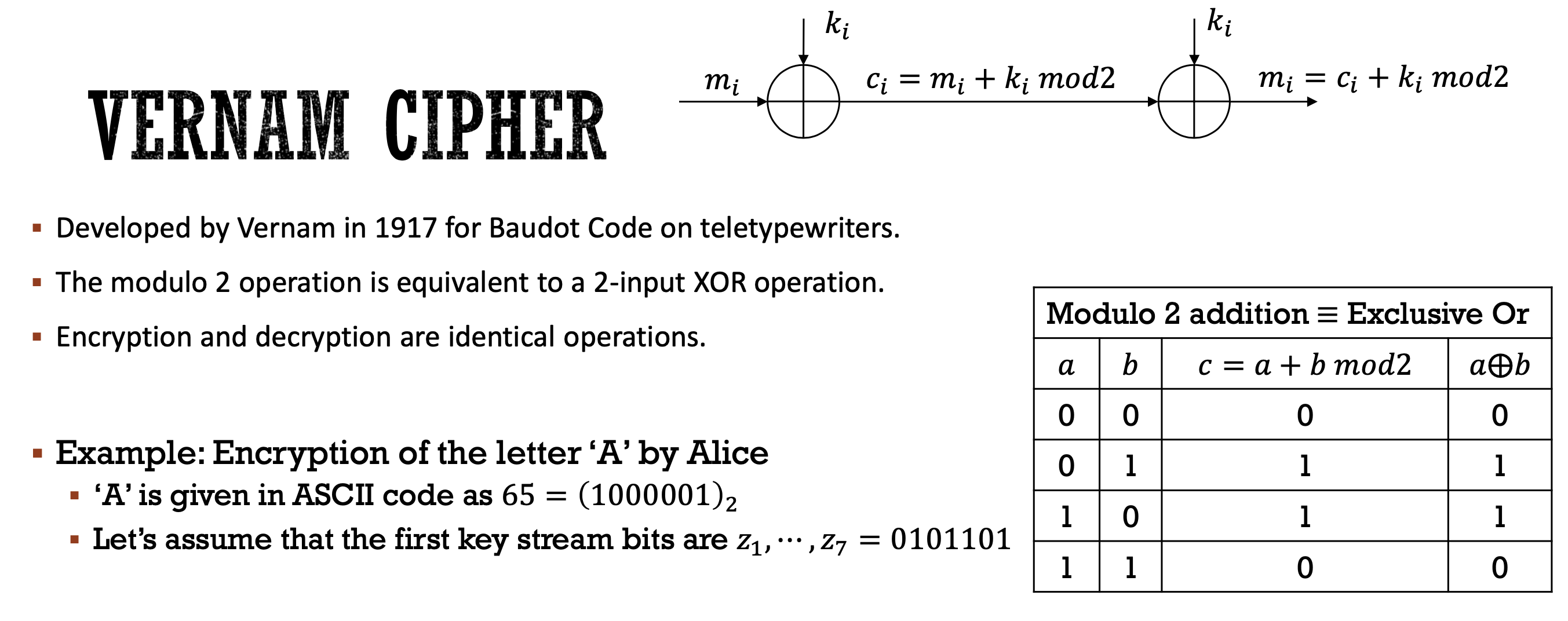
-Encrypting bit by bit vs in blocks is the main dfiferation between Blockcipher and Streamcipher

-The block cipher allows you to suffle up the data better because you're not just encrypting a single bit

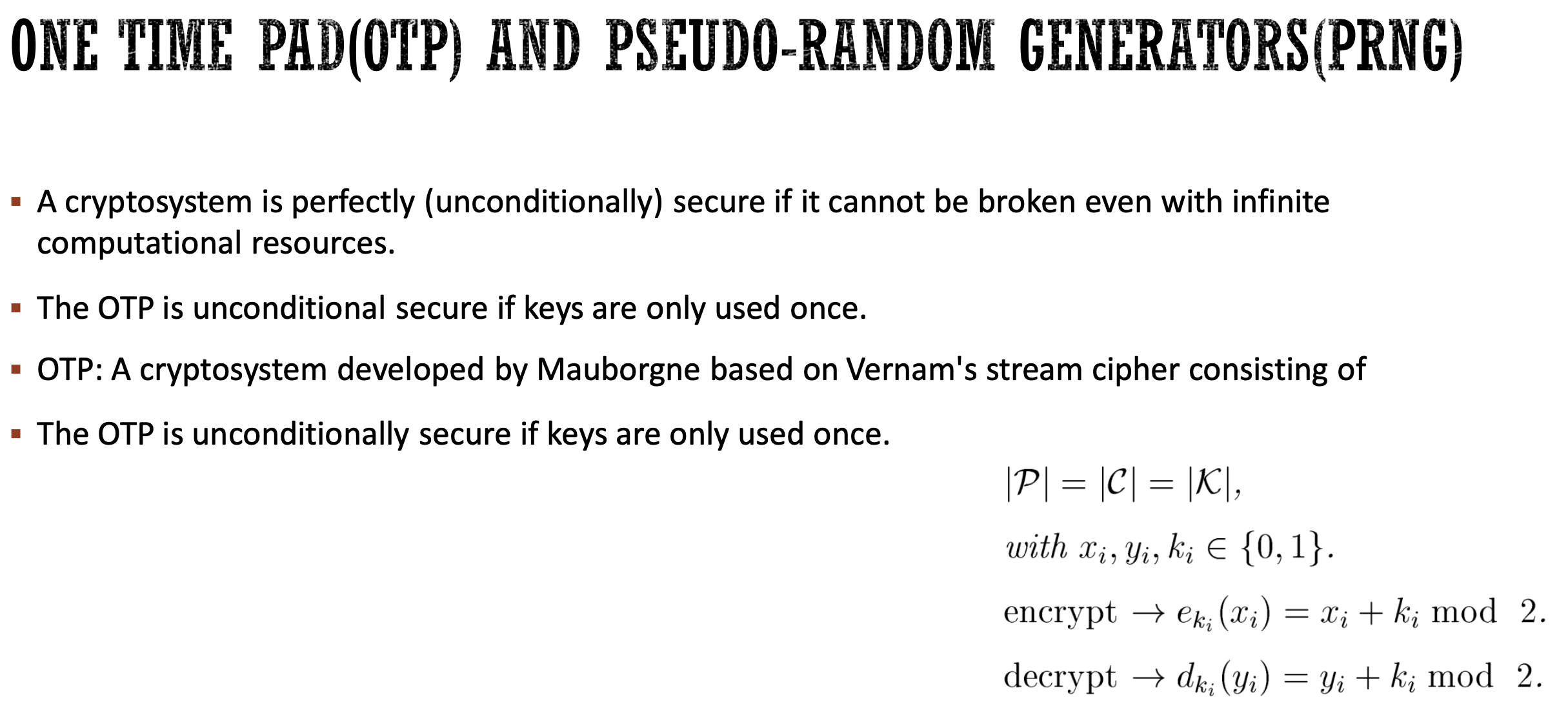
-If you’re using a block cipher you need your data ahead of time

-Stream Cipher does not know its input data, its more for live encrypting of a feed coming over the wire

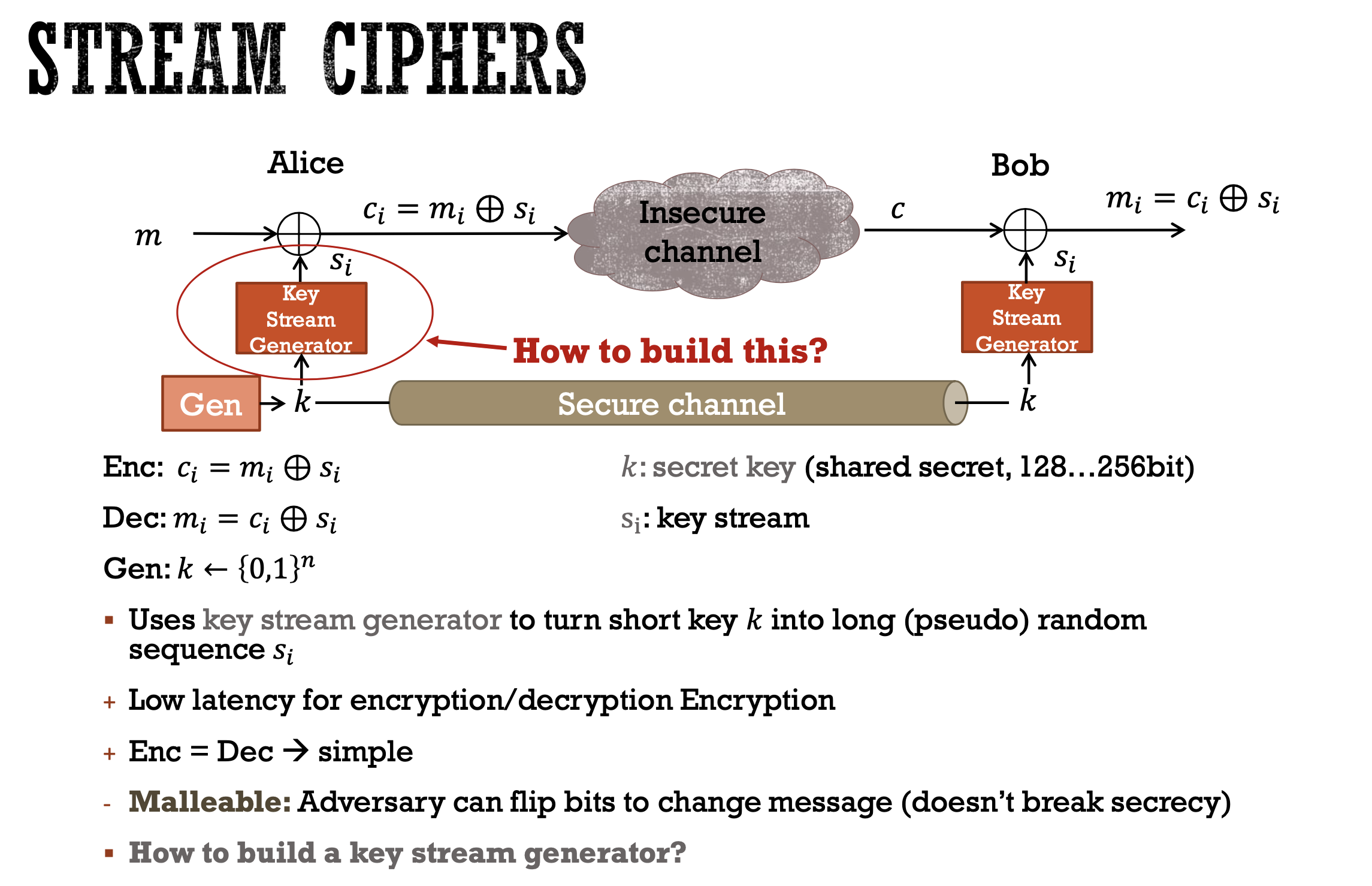
**Vernam Cipher**



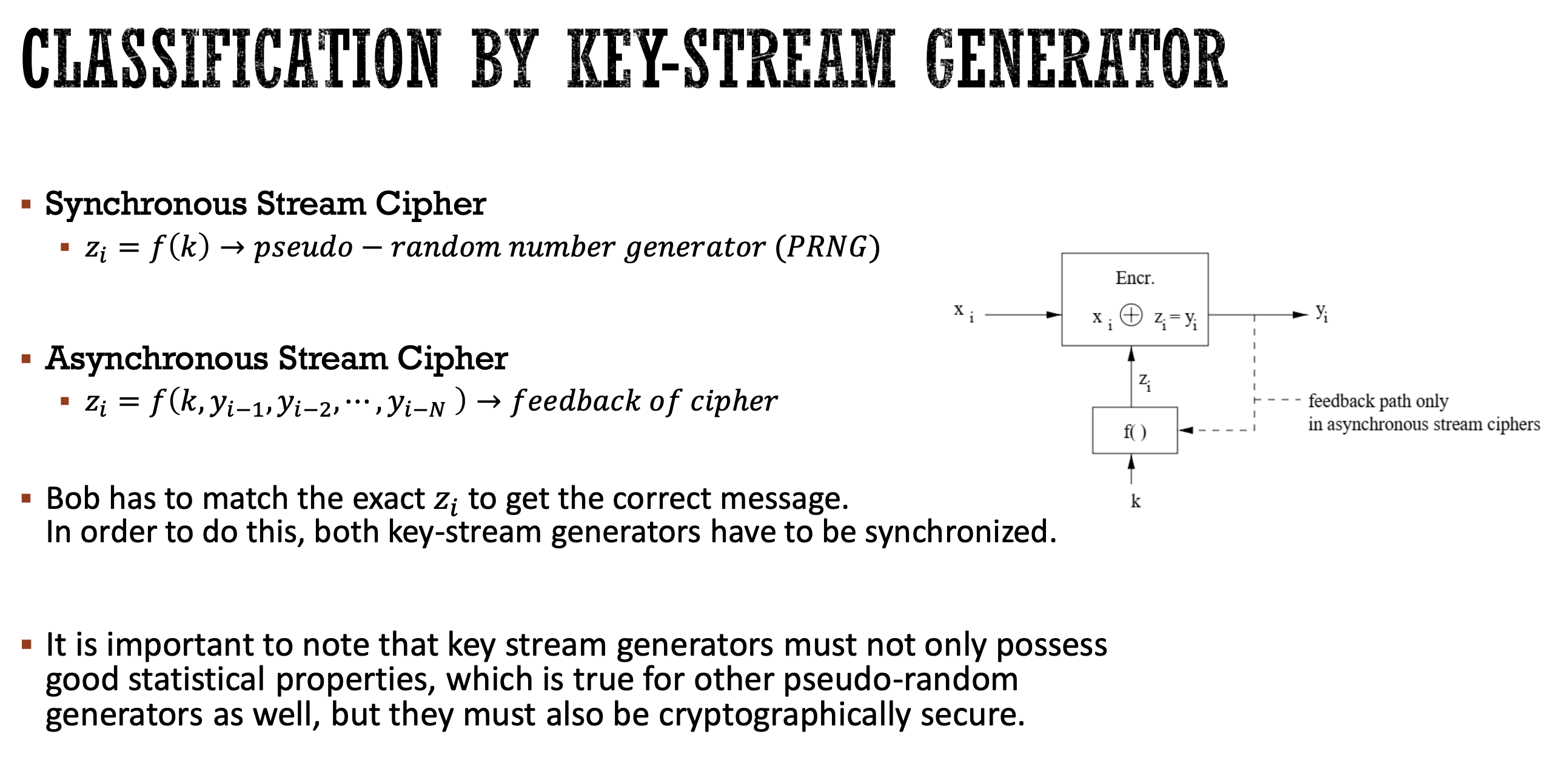
**One-Time Pad**



The definition of security means you should never have a good guess on the content of the message.



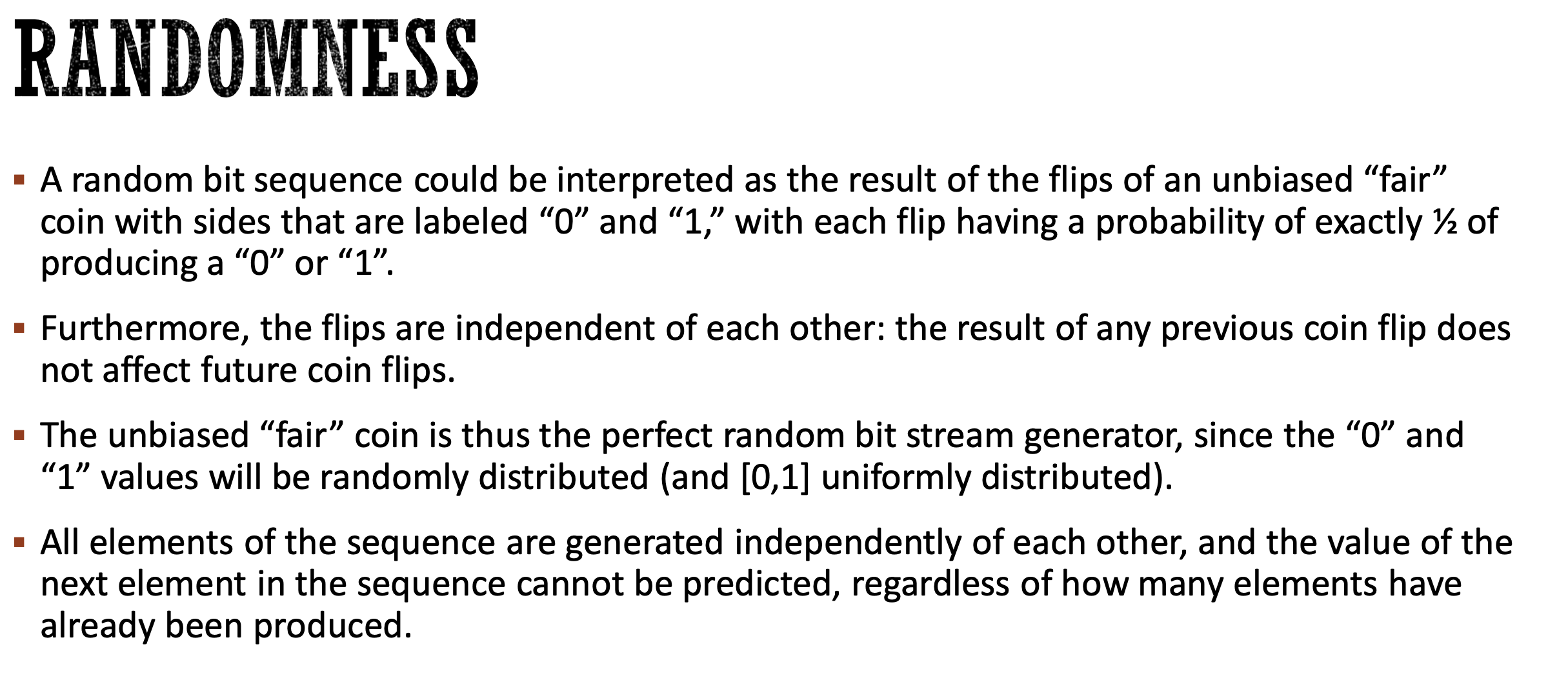
We need to create a random looking key string with pseudo random number generators in order to secure this. However, because the random generator is different on both ends, we would never get the same key string twice.



**PRNG:** A cryptographiically secure pseduo-randodm generator

**Pseudo Random Generator:** (key string generator) is cryptographiicaly secure if it is unpredictable. That is, given the first n output bits of the generator, it is computationally infeaseable to compute the following bits.

The idea of “Randomness” is still unsolved. We are close but do not have full randomness



PRNG’s use one or more inputs and generate multiple pseudorandom numbers

Inputs to PRNGs are called seeds

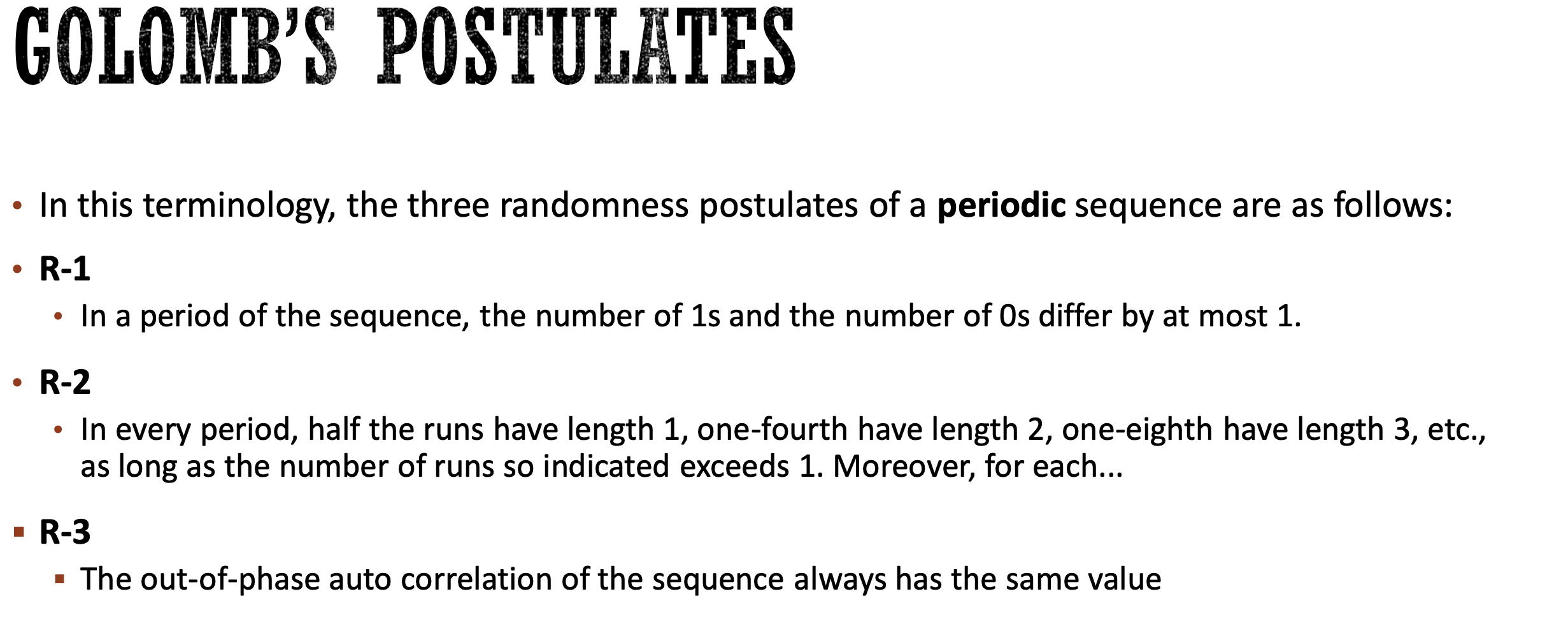
The outputs of PRNG are typically deterministic functions of seed.

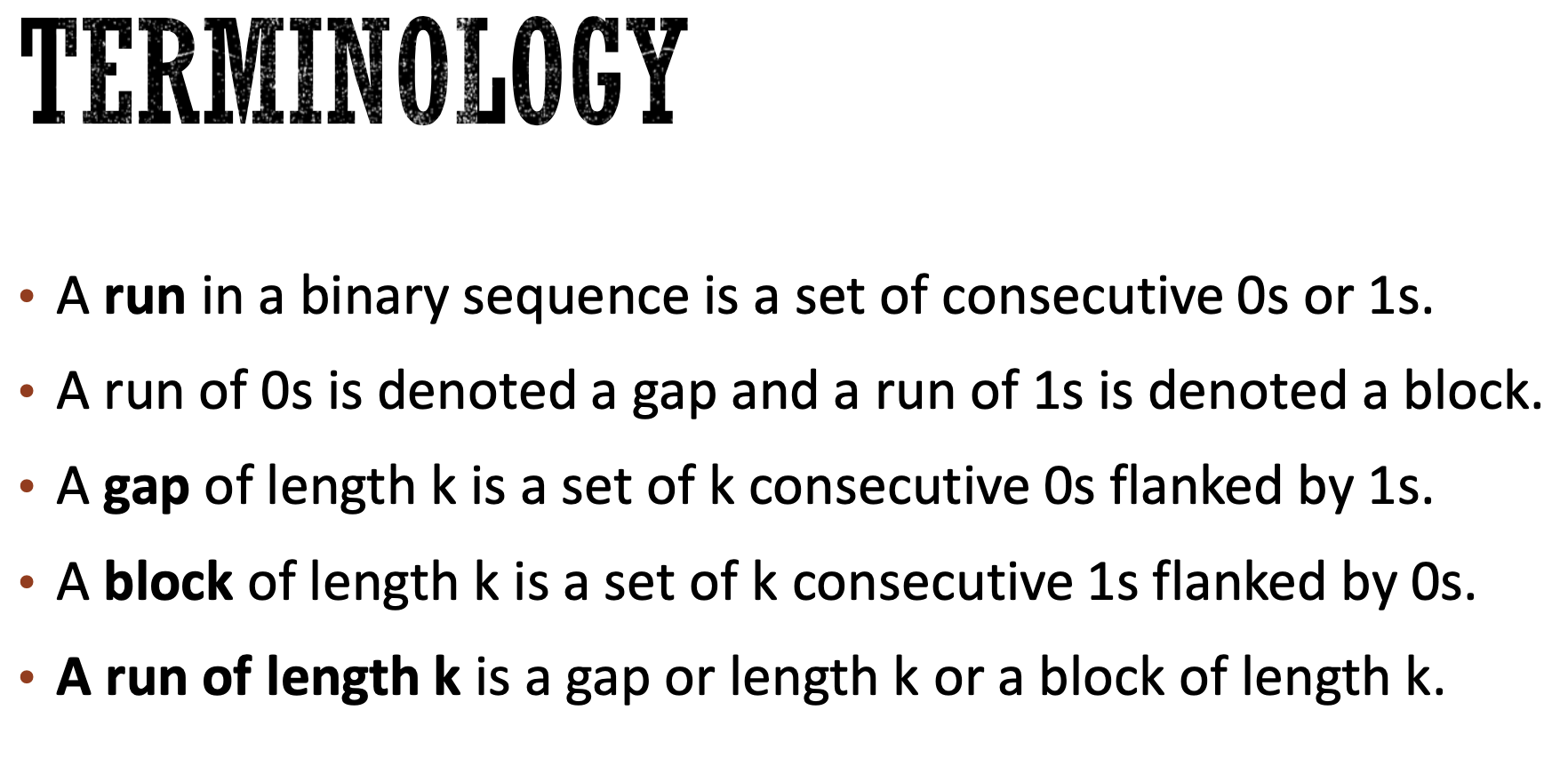
Since it is deterministic, thus is the reason its called “Pseudo” randomness.

**Golomb Postulates for PRNG Binary Sequences**

Golomb introduced the notion of PR sequence of periodic binary sequence that satieties three randomness postulates

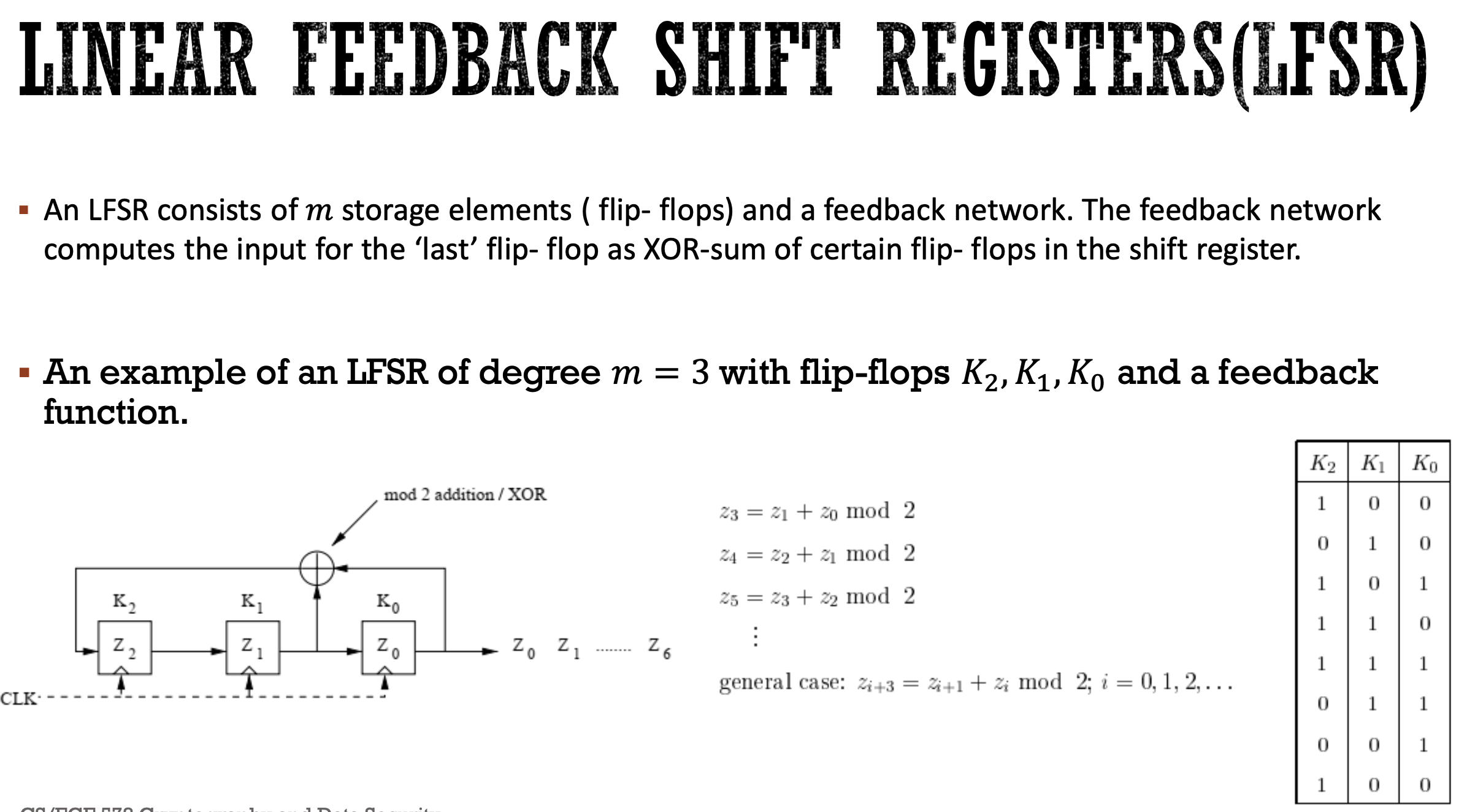
These postulates reflect properties one would expect to find in a random sequence





R3 Example: 11100010

Shift it to the left – 01110001 then XOR them ( = 4). Check all possible shift and they should all give you the same number of 1’s in the XOR process.



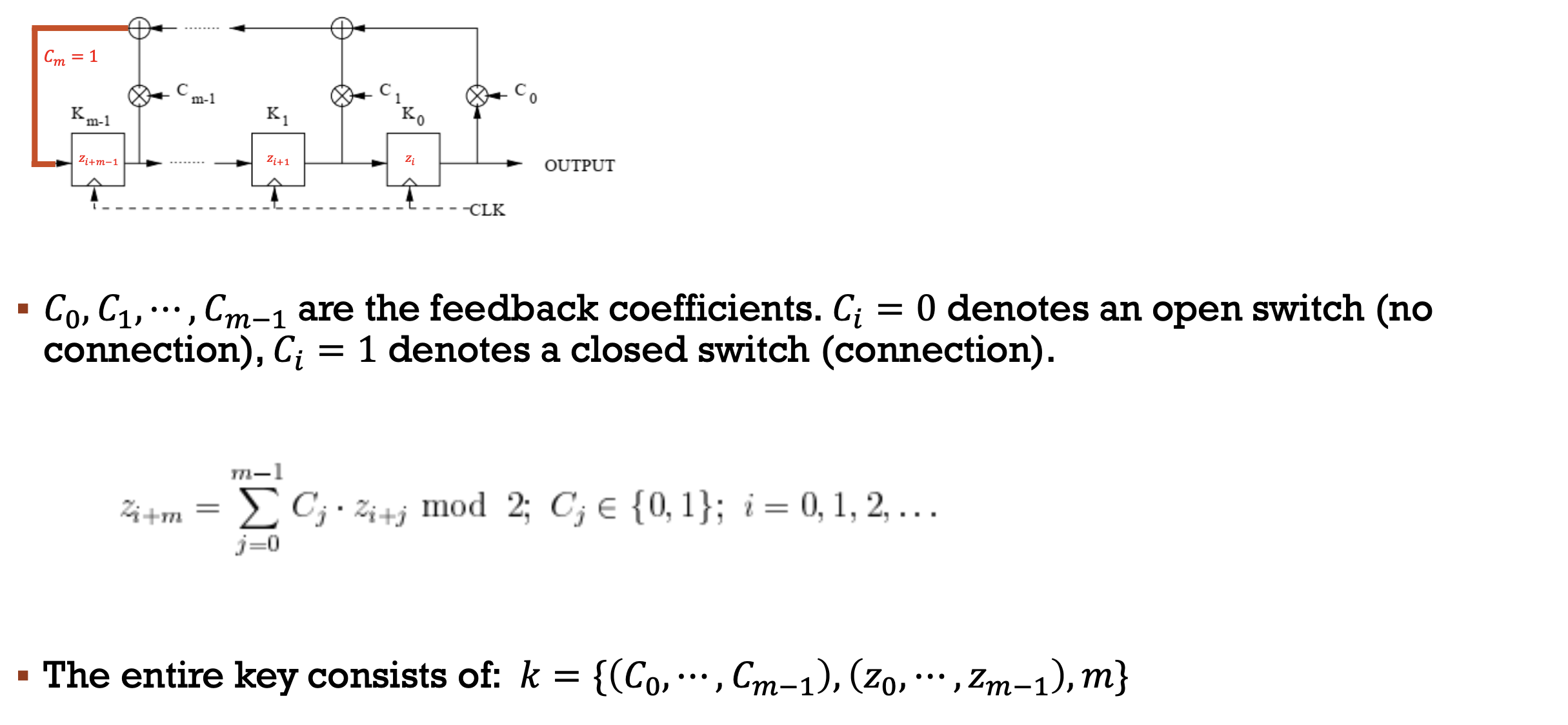
K0 is always the output, and the XOR of K0 and k1 is the next value of K2

How long can it be?

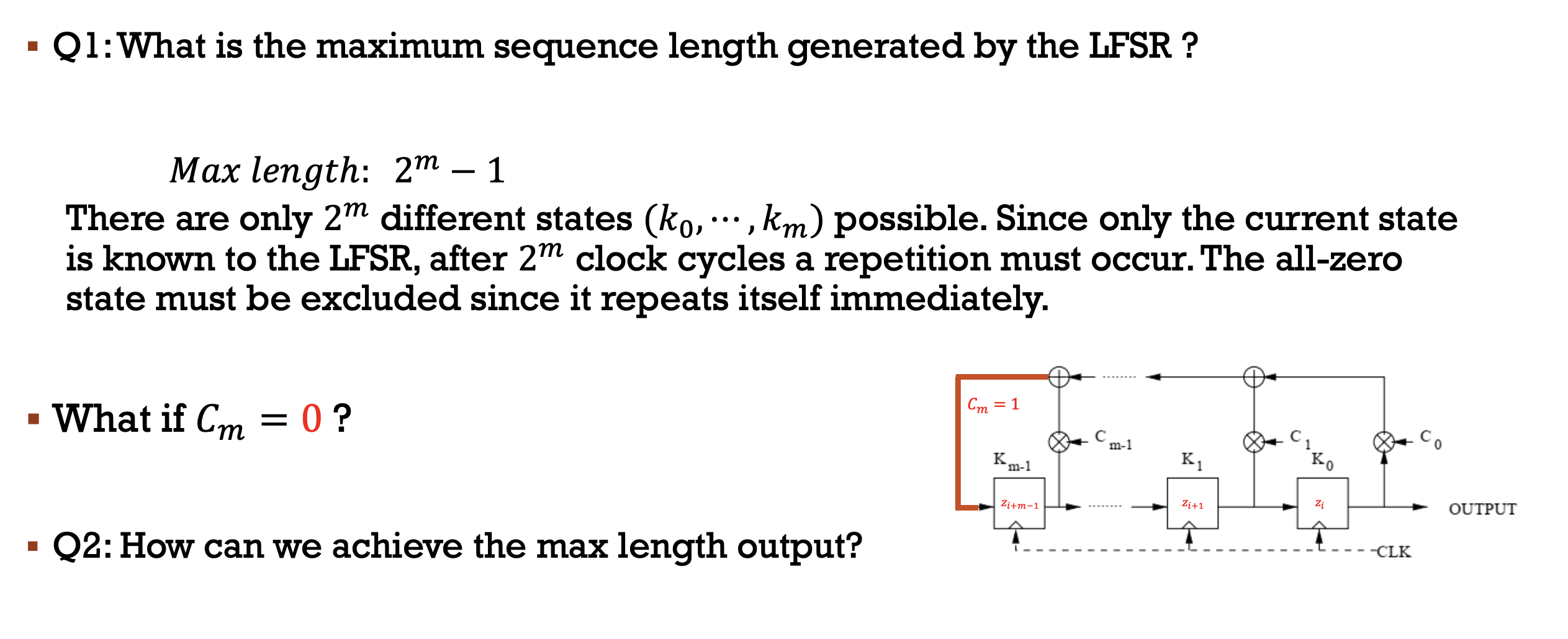
How can we decide the tap points?

How do we choose the initial state?

Initial state, number of registers, and tap point will let us know what type of feedback function we have.



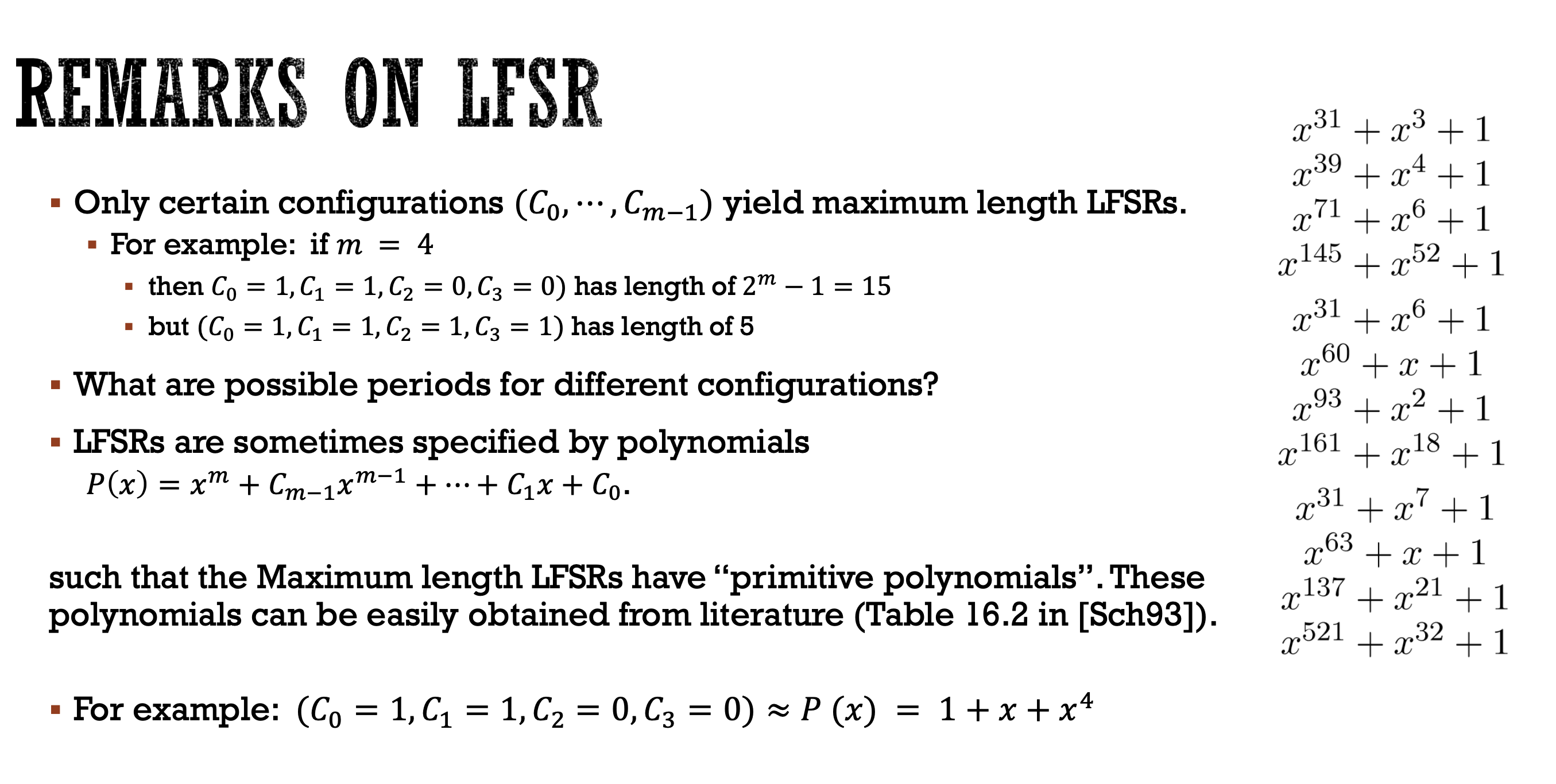
Tap points are always defined by the linear feedback function.



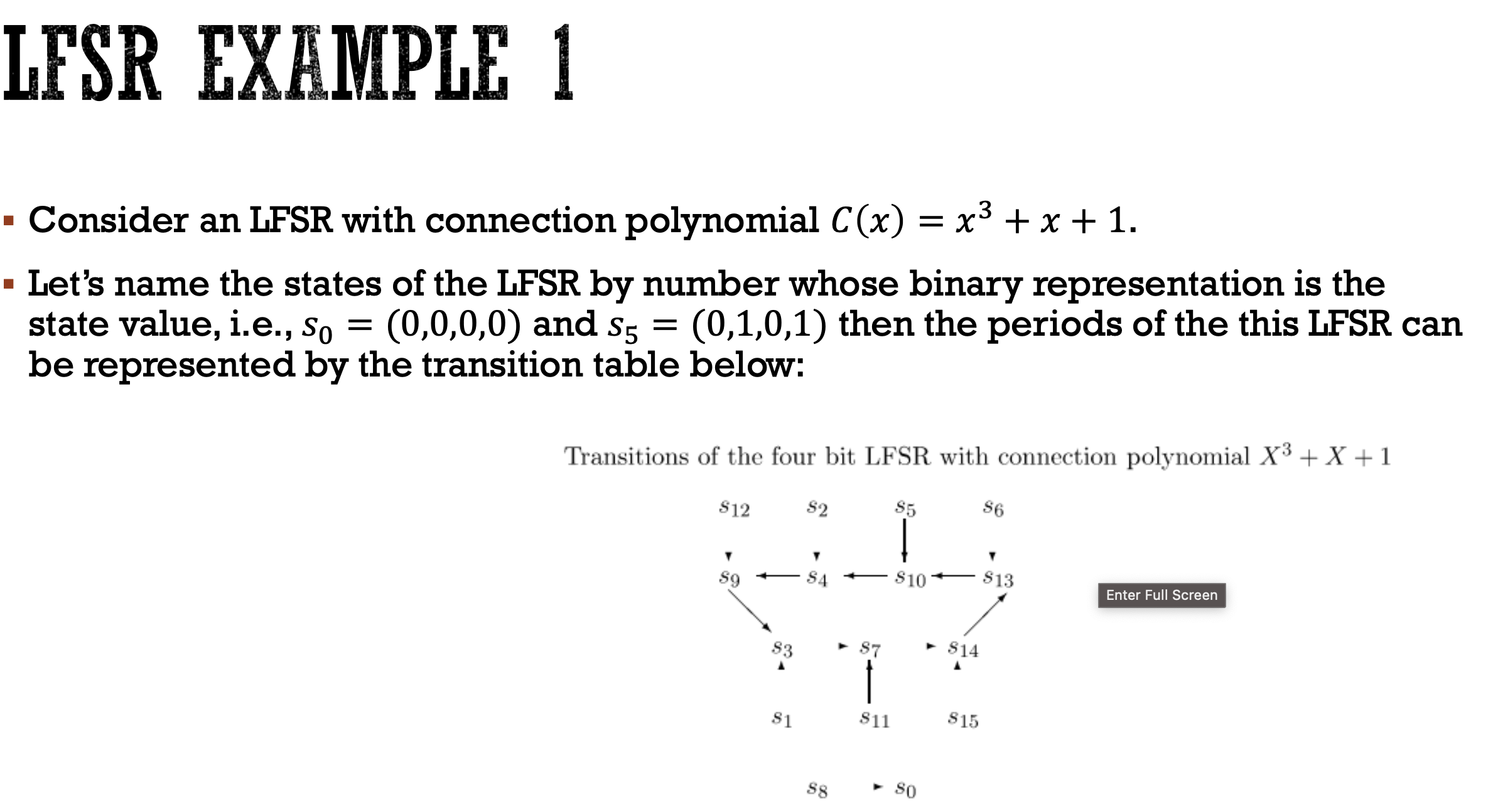
Max length is not 2 to the power m because you must subtract 1 to account if all starting points are 0, the output would be 0.

If the feedback is 0, it will not go back to the start of the of the feedback loop.

C0 is always one, because it needs to be in order for the feedback to work because without it, there would be one less register in the system.

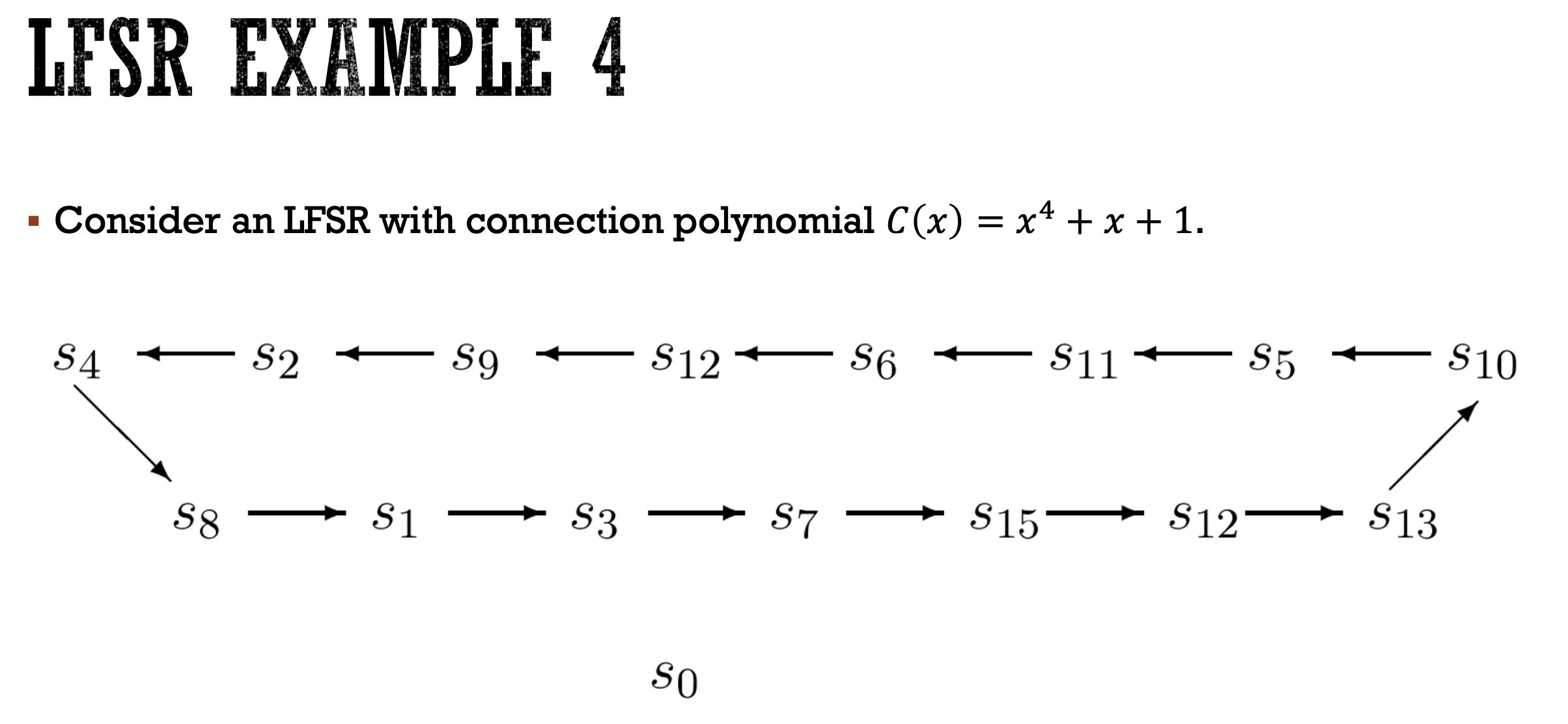


If it is a primitive polynomial, it an only divided by 1 and itself, meaning it has the maximum length for the tap point sequence. Therefore you can tell just by looking at a polynomial whether or not it provides the maximum length.

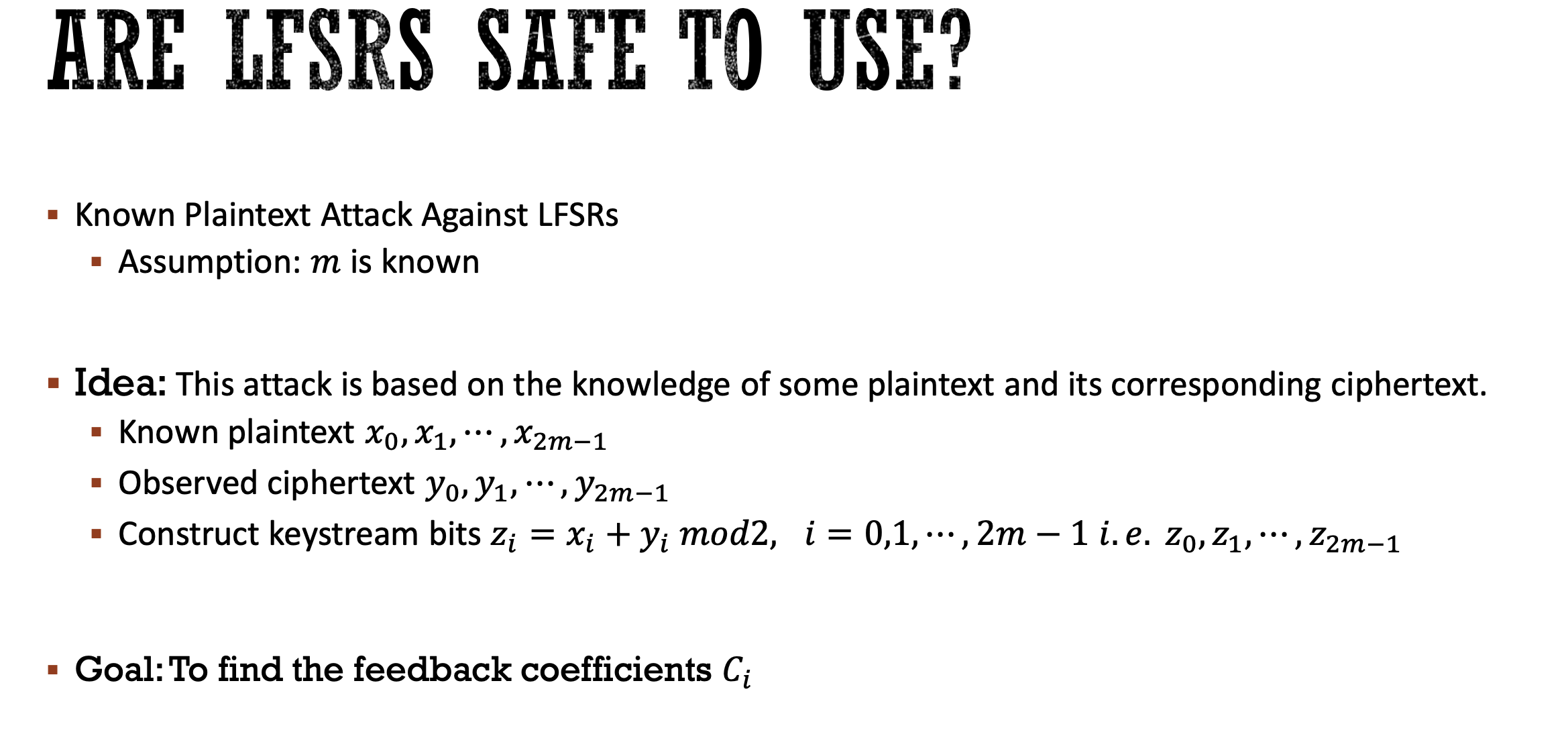


S0 should always be in a different group when the states are broken into groups. S0 will always be separate.

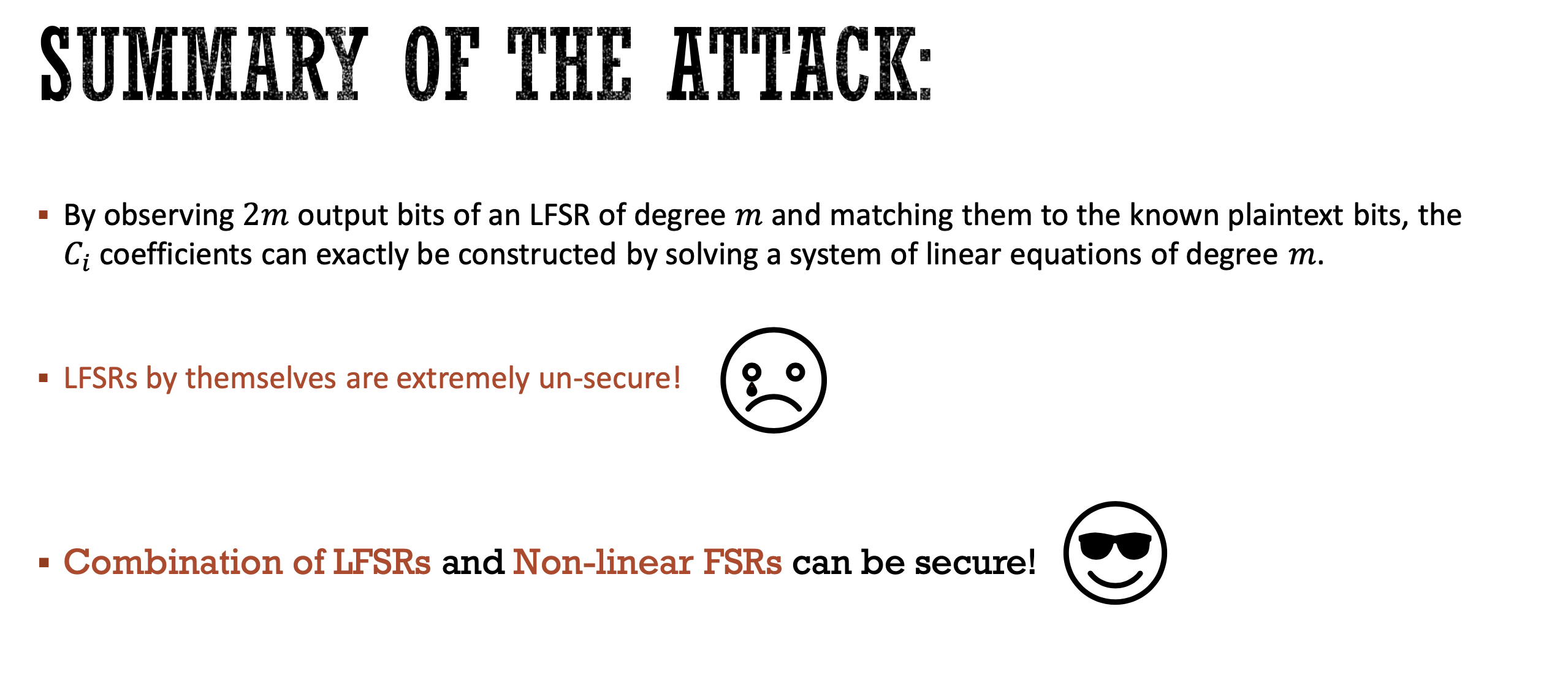
If you don’t have a primitive polynomial, then the starting point is important



This is an example of a primtiive. It has 2^4 – 1 state, with s0 being sperate. No matter where we start, it will continue through all the states.



All you need to find is 2m in order to find all of the coefficients. Unknowns bits + Known equations will give us the coefficients.



To solve this we can use a combination generator, which uses the Output of LSFRs with a Non linear function to create the output

.