This project works as follows:

1) the "transmitter" inputs a message up to 200 characters.

here, just change "original.txt"

in verilog, I will assume the inputs as 7-bit ASCII > [16:0] or use some I/O features

see the ASCII conversion in "TOasciiRaw.txt"

2) the message is transferred from a [200][7] array to a [200][16] array

see "TOasciiEXP.txt"

3) the unencrypted [200][16] array from step 2 is encrypted to a new [200][16] array

here, the array is encrypted by making the MSB of each row = 1

see "TOcrypt.txt"

4) at some point, the [200][16] encrypted array is transmitted to the "receiver"

the receiver hopefully has the circuitry to unlock the message

in this case, the receiver just needs a circuit to group x200 16-bit binary numbers and process them

all the receiver has to do is:

- get the decimal digit of the 16-bit number (or just use binary subtraction)

- subtract 2^15 from the decimal number

- find what ASCII character the result corresponds to

- output the result

- read the message

if "TOdecrypt.txt" matches "original.txt" then the message was successfully transmitted

in practice, the message can be transmitted multiple times and the receiver can see if all transmission are the same

in real life, the receiver will not have access to the original message

5) The verilog test bench will serve as the regular module stimulator that it is

for simplicity, a section of the test bench will serve as the input

also, the test bench will fulfill the decryption and output

the main module will have similar roles since it is stimulated by the test bench

if applied, the transmitter and receiver would have their own circuits

driving the cycle is a FSM

The encryption scheme is very weak for simplicity, and because we are short on time.

It can be anything you want as long as there is an algorithm or look up table to decrypt.

If decryption can't be done, then it defeats the purpose.

If someone can decrypt the message, then that also defeats the purpose.

But this scheme can be expanded or modified in any way.

This started to get very involved as soon as work began. There are some big numbers to work with.

Implementing on a circuit without any I/O or ability to demonstrate (during virus outbreak of 2020) makes it all theoretical.

C++ was a good way to get the idea in writing and provides a some what easy transition into verilog.

At least you can use some of the existing functions to generate some of the 1000's of inputs you will need.

The array index of C++ being reversed from the usual MSB...LSB can be negated if big endiness is always assumed.

That is what I used here, and what will be used in Verilog.

get your input

https://www.rapidtables.com/convert/number/ascii-to-binary.html

count your characters

https://www.lettercount.com/

here is a chart

https://www.ascii-code.com/

verilog I/O

http://chris.spear.net/pli/fileio.htm#$fread

https://www.verilogpro.com/verilog-arrays-plain-simple/

<http://inferno-cpp2v.sourceforge.net/>

The ability to securely communicate has always been needed. On average, for every person trying to transmit a message, at least 50 people are trying to intercept it.

Using a 4 digit PIN can be secure, although with enough time, a malicious person can simply enumerate the possibilities and gain access. Without encrypted data, all is compromised.

This project creates a system, implemented on electronic circuitry, to securely transmit and receive a plain text message. The sender and receiver both have two circuits. One circuit is for encrypting messages and the other for decrypting messages. In this way, both parties can transmit and receive.

The procedure begins with the sender entering a plain text message. The encryption circuitry arranges the message to any encryption scheme desired by the user.

The ability to change the encryption scheme forces any potential adversary to enumerate a mathematically impossible set of combinations. A token and password verify the encrypted message is sent to the correct decryption circuit. Once received, the message is decrypted and the process can continue indefinitely between either node.

So long as each party has the proper circuitry and agrees to changes concurrently, messages can be sent and received across any platform with standard input and output peripherals.

The intentions, details, distribution, and further applications of this system are not able to be disclosed to the general public.