### SYSC 5104. Methodologies for Discrete Event Modelling and Simulation

### Project Report

### Simulating Encryption of QR Code based on ECA

### Team Members:

### Saima Hidayatullah Khan (Student# 8511041)

### Zahra(Leili) Parishani (Student# 101060131)

### 

**Part I - Overview:**

Barcode technology is an emerging technology including coding, printing, data acquisition and processing. Two-dimensional code as a kind of effective mode of carrying and transmission information has certain security function. The project is about the simulation of the encryption of the QR Code codes. The idea is to implement cryptographic algorithm to hide the information contained in an image to transmit it securely over the network.

We have used a DES (Data Encryption Standard) algorithm for encryption. The DES encryption algorithm is used to encrypt the pepper-and-salt region of the QR code image. Then, the unencrypted dates of the white region around the QR are combined with encrypted data of the pepper-and-salt region in the middle of the QR. The algorithm requires a key to be selected by the formula *N\*N/8* so we have a key value as *128* for *32x32* dimension QR.

To simulate the encryption, the cell space is divided into two zones as below:

* A 32x32 dimension cell space for QR Input
* 32x32 dimension cell space to hold QR Encrypted result.

**Part II – Formal Specification:**

The formal specification for the Cell DEVS space defined is as under:

N ={(0,1),(0,2),(0,3),(0,4),(0,5),(0,6),(0,7),(0,8),(0,9),(0,10),(0,11),(0,12),(0,13),(0,14),(0,15),

(0,16),(0,17),(0,18),(0,19),(0,20),(0,21),(0,22),(0,23),(0,24),(0,25),(0,26),(0,27),(0,28),(0,29),

(0,30),(0,31),(0,32),(0,0),(0,-1),(-1,0),(1,0)}

Select {(0,1),(0,2),(0,3),(0,4),(0,5),(0,6),(0,7),(0,8),(0,9),(0,10),(0,11),(0,12),(0,13),(0,14),(0,15),(0,16),(0,17),(0,18),(0,19),(0,20),(0,21),(0,22),(0,23),(0,24),(0,25),(0,26),(0,27),(0,28),(0,29),(0,30),(0,31),(0,32),(0,0),(0,-1),(-1,0),(1,0)}

**Part III – Cell-DEVS Rules:**

The following rules govern the simulation results:

**Encrypted QR Rules:**

* If the QR Input cell has a value less than or equal to 100, encrypt the QR Encrypted cell with value x key i.e. value x 128.
* If the QR Input cell has a value less than or equal to 200, encrypt the QR Encrypted cell with value x key i.e. value x 128 / 2.
* If the QR Input cell has a value less than or equal to 300, encrypt the QR Encrypted cell with value x key i.e. value x 128 / 3.
* If the QR Input cell has a value less than or equal to 400, encrypt the QR Encrypted cell with value x key i.e. value x 128 / 4.
* If the QR Input cell has a value equal to zero, encrypt the QR Encrypted cell with value equal to key i.e. 128.

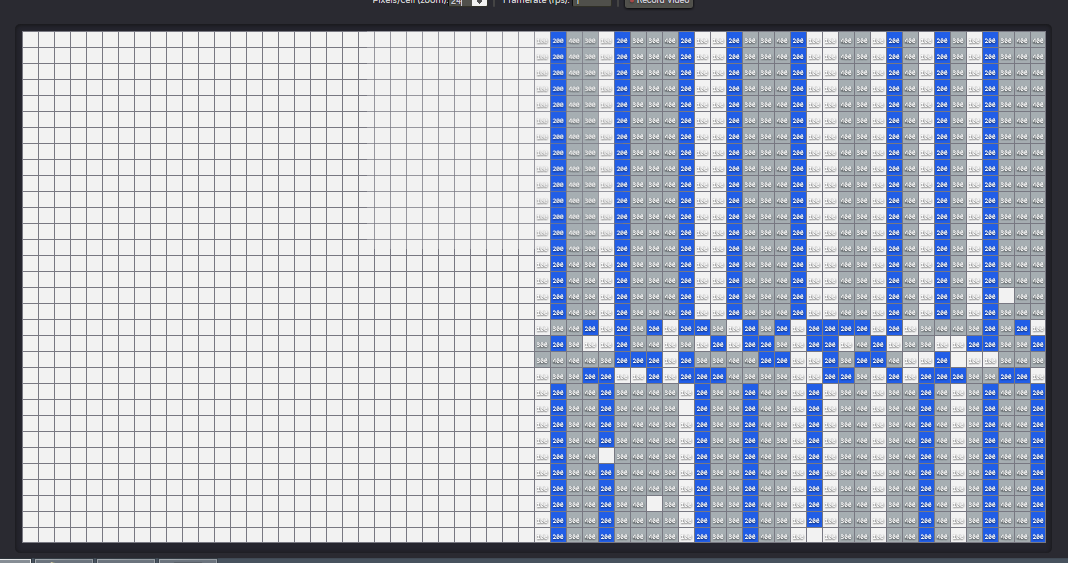
**QR Input Rules:**

* If the QR Input cell has a value not equal to zero, put the value same as initial values in *init.val* file.
* If the QR Input cell has a value equal to zero, put *zero* as the input value.

**Simulation Results:**

Below are the results of the simulation observed on the online Cell-DEVS tool:

* Initial State:

The QR Input is initialized at the right hand-side of the figure with the input QR data values as shown below:

QR Input

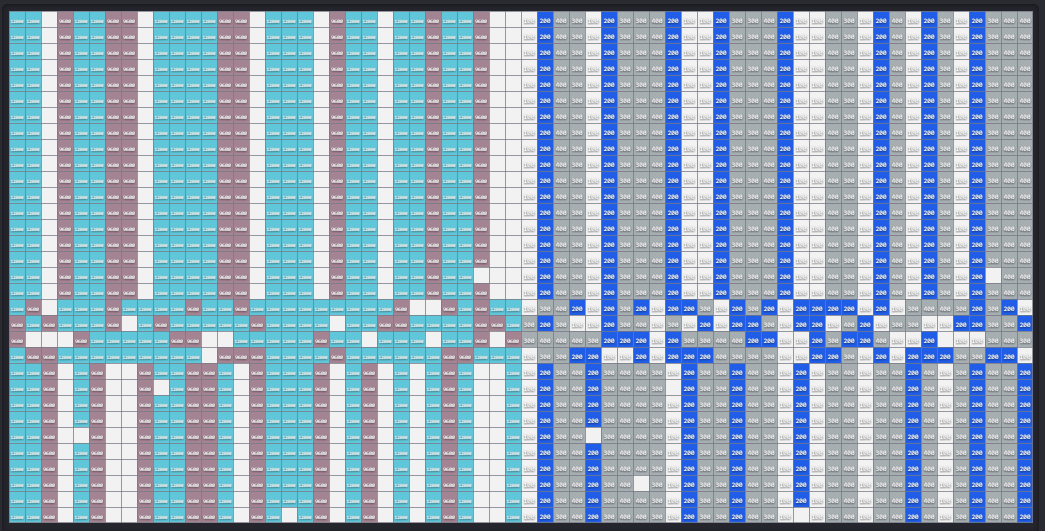
Encrypted QR

* Intermediate State:

The QR is being encrypted using the rules as shown in the left hand-side of the figure below:

QR Input

Encrypted QR



* Final State:

The QR Encrypted zone is completely converted to encrypted data values in the end of the simulation. The QR Input and Encrypted QR results can be observed in the figure below:

QR Input

Encrypted QR

