Volume 5, Issue 10, October-2015 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Enfold RSA Check on Web Executable Using Reconfigurable Hardware

Rahul Jassal

Department of Computer Science & Applications, Panjab University Regional Centre, Hoshiarpur, India

Abstract: Perhaps unsurprisingly in light of the high degree algorithm or available tapping data centers, security forces the rise of regional or standalone dongle for data threats. The paper presents a secure system RSA de-cryptography model[1] using a FPGA based hardware to prevent data stealing. A part is wrapped under hardware side[2] and further is decrypted software side.

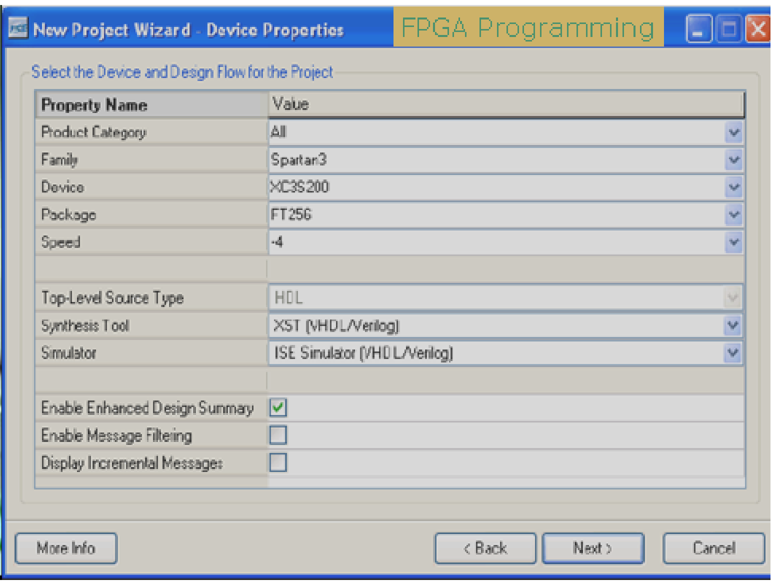
Keywords: - Virtex FPGA, RSA, moxa_usbcable, Spartan, Killing_Thread

I. INTRODUCTION

The application is based on reconfigurable Hardware (FPGA) utility in such a way that Web Pages can only be accessed when the Dongle is connected to USB Port. The link has been established between Dongle (FPGA) and a dummy Website with the help of Communication Assembly Protocols. At the Front End, the interface carries two sections like Admin and the end user, Admin can assign usernames/passwords, view the roll-calls, update e the amendments made by him, on the other hand end user can only mark his/her attendance. All this is managed in VB.NET [3][4] environment under the presence of Dongle connected at USB Port. The application is designed in such a way that Software Piracy can be stopped .At the back end of the application a task manager is tracing out all the processes whose status is set to "Running" and re- filter out all that processes saved by User within database. The database is maintaining a log carrying all the processes provided by User, and during startup of the application it blocks them all if required hardware is missing. If the application finds presence of dongle, it stops all its back end routines and set the status of the processes to "Running". Besides this, the application is communicating with virtual COM Ports created through Mosa Utility The application needs user side feedback for determining the COM Port and one initiative to run the hardware simulation and rest of the task is done through .NET[5].

II. FPGA USED

The diagram depicts a hardware side, accepting input message with encrypted keys. The all system is synchronized with clock and handover to module for fast execution. And this in continuation is repeated for getting equivalent cipher text and the closing figure depicts pin used for this dongle. The Figure 3 shown here is a Xilinx operated Spartan3 kit where the bit file is tuned [6].

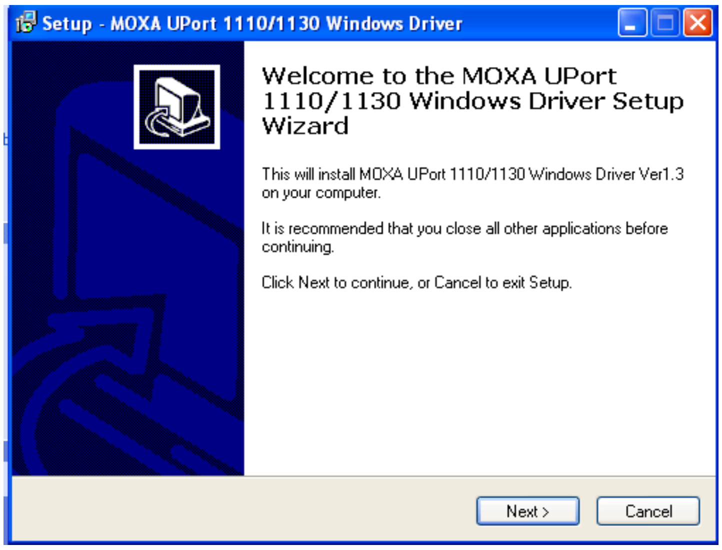


Select the Device and Design Flow for the Project

Property Name	Value
Product Category	All
Family	Spartan3
Device	XC3S200
Package	FT256
Speed	-4
Top-Level Source Type	HDL
Synthesis Tool	XST (VHDL/Verilog)
Simulator	ISE Simulator (VHDL/Verilog)
Enable Enhanced Design Summary	✓
Enable Message Filtering	☐
Display Incremental Messages	☐

III. INSTALLING MOXA_UPORT 1110/1130 CABLE

The following will install MOXA UPort 1110/1130 driver on the machine, and it is recommended that all applications kept closed before continuing it for installation, specify the path where you want to install the moxa and finally click on finish button.

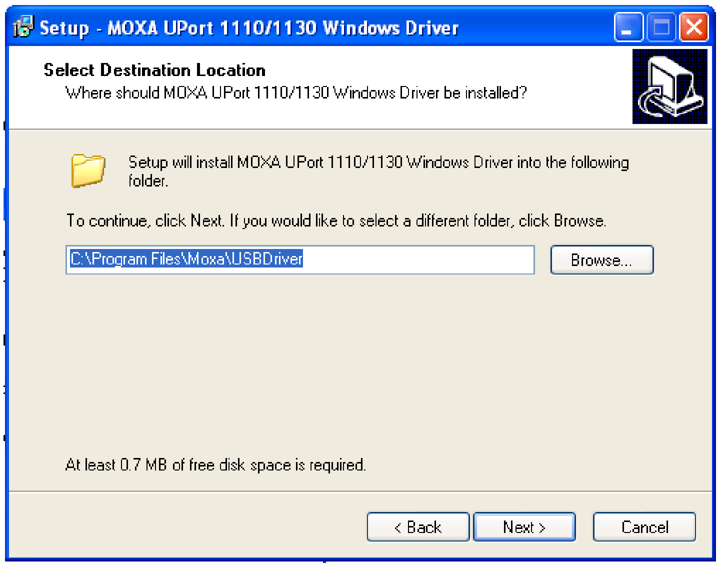

Setup - MOXA UPort 1110/1130 Windows Driver

Welcome to the MOXA UPort 1110/1130 Windows Driver Setup Wizard

This will install MOXA UPort 1110/1130 Windows Driver Ver1.3 on your computer.

It is recommended that you close all other applications before continuing.

Click Next to continue, or Cancel to exit Setup.


Setup - MOXA UPort 1110/1130 Windows Driver

Select Destination Location

Where should MOXA UPort 1110/1130 Windows Driver be installed?

Setup will install MOXA UPort 1110/1130 Windows Driver into the following folder.

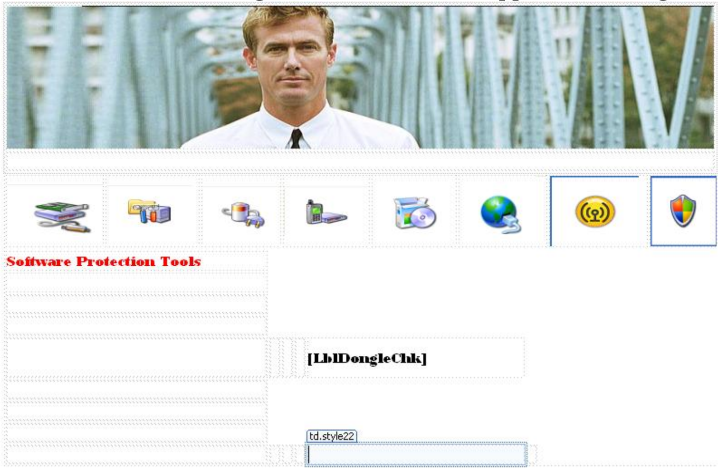
To continue, click Next. If you would like to select a different folder, click Browse.

C:\Program Files\Moxa\USBDriver

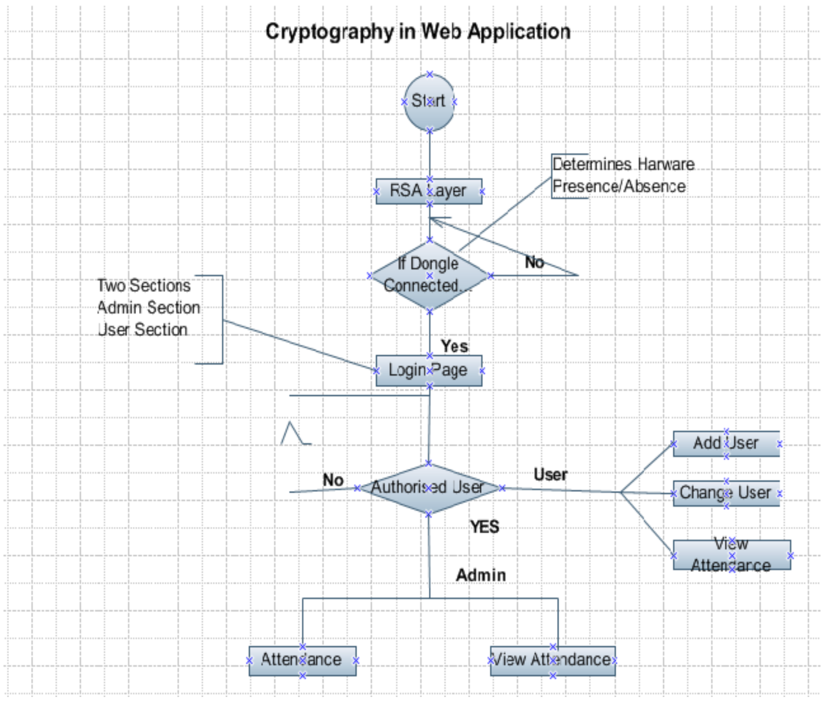
At least 0.7 MB of free disk space is required.

IV. DESIGN INTERFACE

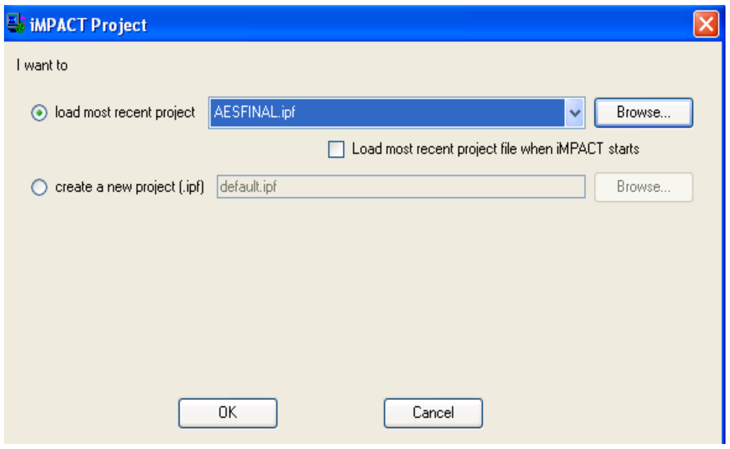
This screen is designed through ASP technology and a XML based task manager that is executing a thread after some seconds interval for the presence of dongle using serial cable, if the dongle is connected with the computer the screen forwards you to the relevant web page otherwise it continuously making a check on the same page after some interval for detection of the dongle. The flow chart is describing the flow of the web application using RSA Algorithm.

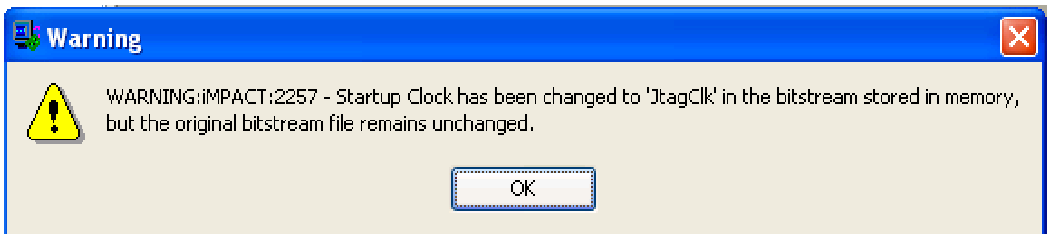


4.1 Flowchart

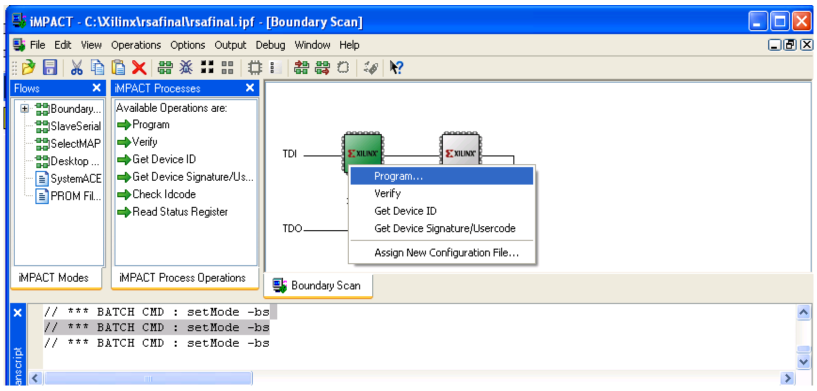


4.2 RSA FPGA TUNNING





WARNING:iMPACT:2257 - Startup Clock has been changed to 'JtagClk' in the bitstream stored in memory, but the original bitstream file remains unchanged.


iMPACT - C:\Xilinx\rsafinal\rsafinal.ipf - [Boundary Scan]
File Edit View Operations Options Output Debug Window Help
Flows
Boundary...
SlaveSerial
SelectMAP
Desktop ...
SystemACE
PROM Fil...
iMPACT Modes
iMPACT Processes
Available Operations are:
Program
Verify
Get Device ID
Get Device Signature/Us...
Check Idcode
Read Status Register
iMPACT Process Operations
TDI
TDO
XILINX
XILINX
Program...
Verify
Get Device ID
Get Device Signature/Usercode
Assign New Configuration File...
Boundary Scan

```
// *** BATCH CMD : setMode -bs
// *** BATCH CMD : setMode -bs
// *** BATCH CMD : setMode -bs
```

V. PROTOTYPES FOR ENCRYPTION AND DECRYPTION

```
Encoder_Encryption(int random_number)
Encryption_started()[window based]
processlist(List of Exe's)
Ecomponent(cnt, phi)
Dcomponent(cnt, phi)
Encript = System.Math.Pow(Double.Parse(carry1), Ecom_Arr(0)) Mod n
```

5.1 Encoder Encryption (int random number)

This Subroutine[6] is the core part of the application it contains the basic subroutine calls for generating random number, converting normal text to cipher text and major system calls to transfer data from software end to Hardware end, the interrupts like serial write or read checks whether particular hardware is attached to the system through its properties like CTS Holding. In addition with this all this function is having subroutine for comparing the results of the software end hardware end, and if results matched required web page is displayed to the User.

5.2 Encryption_started()

Like above this function is built for window application repeats the same steps as specified above the differences lies in window or web application. Various methods help in generating random numbers, preparing the cipher text and ultimately comparing the results if results matched you have access to particular software otherwise you are transformed to form requesting for connection the hardware device.

5.3 Processlist(List of Exe's)

Actually this function is working at back end in Window Application to filter out the various processes specified by the User for restricting their access to the third person. It's using
XML for maintain a proper log for all that processes which is running at back end, and to terrace them it is referring to class system.management whose object properties help in determine Process Status and Ultimately destroyed the process that is restricted and it gets all these restricted processes details from Database working at back end.

5.4 Ecomponent(cnt, phi)

This function creates "e" component Private key that is further provided to other subroutines for creating cipher text.

5.5 Dcomponent(cnt, phi)

This function creates "d" component Public key that is further provided to other subroutines for creating cipher text.

VI. GENERATING THE PAIR OF KEYS

```
Generate two distinct prime numbers [p,q]
n = p * q
phi = (p - 1) * (q - 1)
Dim q1 As Integer = 0
While (q1 <> range)
Ecom_Arr(q1) = ecomponent(cnt, phi)
cnt = cnt + 1
q1 = q1 + 1
End While
While (flag)
If (phi Mod cnt <> 0) Then
flag = 0
carry = lcm(cnt, phi)
gcd1 = cnt * phi / carry
d = Dcomponent(cnt, phi)End While
```

where p and q are prime numbers

After generating the pair of public and private keys the next step is to create a cipher text that can be transferred to the decryption module.

6.1 ENCRYPTED DATA

```
Encript = System.Math.Pow(Double.Parse(carry1), ecomponent) Mod n
Dim arr(1) As Byte
arr(0) = Byte.Parse(encript)
SerialPort1.Write(arr, 0, arr.Length)
```

6.2 DECRYPTED DATA FROM FPGA

```
Dim decript_byte As Byte
decript_byte = SerialPort1.ReadByte
```

After this a comparison is made if decrypted data and the data before encryption results in same then the user is allowed to switch over to various web pages.

6.3 TRACING "RUNNING"STATUS OF EXE'S AT BACK END

```
Dim class1 As ManagementClass = New ManagementClass("Win32_process")
writer = New StreamWriter("c:\process3.xml", False)
writer.Write("<?xml version=""1.0""?>")
writer.Write("<processes>")
For Each ob As ManagementObject In class1.GetInstances
Dim Caption As String = ob.GetPropertyValue("Caption").ToString
Container(index) = Caption
index = index + 1
Dim Description As String = ob.GetPropertyValue("VirtualSize").ToString
Dim Name As String = ob.GetPropertyValue("WorkingSetSize").ToString
writer.Write("<process>")
writer.Write("<Caption>" + Caption + "</Caption>")
writer.Write("<VirtualSize>" + Description + "</VirtualSize>")
writer.Write("<UserModeTime>" + ob.GetPropertyValue("UserModeTime").ToString + "</UserModeTime>")
writer.Write("<WorkingSetSize>" + Name + "</WorkingSetSize>")
writer.Write("<WriteOperationCount>"      +      ob.GetPropertyValue("WriteOperationCount").ToString      +
"</WriteOperationCount>")
```

6.4 KILLING PROCESS

```
For index = 0 To cntr - 1
If Trim(Container1(j)) = Container(index) Then
Dim plist As Process() = Process.GetProcesses()
For Each p As Process In plist
If p.MainModule.ModuleName.ToLower = ListBox1.Items(j) Then
process_str = ListBox1.Items(j).ToString()
p.Kill()
End If
```

VII. CONCLUSION

The paper describes a paradigm for enclosing a system under RSA cryptography check using intervention of FPGA programming and Hardware dongle using MOxa cable, one can make use of this simulation for initiator with web based application for making them executable only when user or admin permits under wrapped simulator.

REFERENCES

[1] William Stallings ,"Cryptography and Network Security, Principles and practices" Edition 3d, Pearson Education

[2] Rahul Jassal, "Wrapped RSA Cryptography Check on window executable using reconfigurable hardware"

[3] Daniel Cazzulino, "Web Programming using VB.NET"

[4] Ganelon, "Visual Basic . Net Black Book"

[5] msdn2.microsoft.com/en-us/library/a9910312.aspx

[6] Computer Security Objects Register (CSOR): http://csrc.nist.gov/csor/.[19] For javax.comm package: http://www.stanford.edu/~bsuter/javax.comm-v2 win32.zip[20] R. Rivest, A. Shamir, L. Adleman. A Method for Obtaining Digital Signatures andPublic-Key Cryptosystems. Communications of the ACM, Vol. 21 (2), pp.120–126.1978. Previously released as an MIT "Technical Memo" in April 1977. Initial Publication of the *RSA* scheme

AUTHORS PROFILE

Rahul Jassal is working as Assistant Professor in Department of Computer Science & Application, Panjab University Regional Centre, Hoshiarpur, India. He is working in the same profession since 2007, his area of interest lies with algorithms designs.