Purpose. This document outlines the procedure for encapsulating a fully-CAESAR HW API-compliant authenticated cipher in the FOBOS Device Under Test (DUT) Wrapper, and generating CAESAR-API and FOBOS-compliant test vectors.

1. Generate test vectors compliant with the CAESAR HW API using the test vector generator, aeadtvgen.py, or other user-defined method of generating compliant test vectors. The procedure for generating test vectors using aeadtvgen.py is outlined in [x] and [y].

2. The products generated by aeadtvgen.py, and designed for use with the CAESAR HW API protocol and the AEAD test bench (AEAD\_TB.vhd), located at [x] and described in [y], are pdi.txt, sdi.txt, and do.txt.

3. (Recommended) It is highly recommended that the user ensure that the test vectors in a set of pdi.txt, sdi.txt and do.txt perform correctly during behavioral simulation using AEAD\_TB.vhd, or user-defined behavioral test bench.

4. The user should then encapsulate a valid set of pdi.txt, sdi.txt, and do.txt into a FOBOS test vector, typically called dinFile.txt. A FOBOS test vector consists of an application-specific test vector (in this case, a CAESAR HW API-compliant test vector), with the addition of FOBOS protocol. FOBOS protocol consists of a number of 16-bit words and is summarized in Table X. In this table, quantities for “number of bytes” are expressed in four-digit hexadecimal words.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sequence | Code (Hex) | Definition | Remarks | Requirement |
| 1 | 00C0 | PDI FIFO | Load FIFO Register (Input) | Required |
| 2 | P3P2P1P0 | # of Bytes | # of bytes in pdi.txt | Required |
| 3 | 00C1 | SDI FIFO | Load FIFO Register (Input) | Required |
| 4 | S3S2S1S0 | # of Bytes | # of bytes in sdi.txt | Required |
| 5 | 00C2 | RDI FIFO | Load FIFO Register (Input) | Optional |
| 6 | R3R2R1R0 | # of Bytes | # of bytes of random data | Optional |
| 7 | 0081 | DO FIFO | Load FIFO Register (Output) | Required |
| 8 | D3D2D1D0 | # of Bytes | # bytes of expected output | Required |
| 9 | 0080 | Command | Load Command Register | Required |
| 10 | 0001 | Start | Start Command | Required |

5. A formal FOBOS test vector encapsulator is not included in this FOBOS release. However, two examples of python scripts use to encapsulate a set of test vectors consisting of pdi.txt, sdi.txt, and do.txt in FOBOS protocol are included in the appendix. The first script, dinFileTBGen1.py, generates a FOBOS test vector designed for a “fixed-versus-random” (FVR) t-test on an unprotected authenticated cipher. The FVR t-test, as well as other methodologies using Test Vector Leakage Assessment (TVLA)m is described in [x] and [y]. The second script, dinFileTBGen2.py, generates a FOBOS test vector designed for a FVR t-test on a protected authenticated cipher. The difference in the second script is that dinFileTBGen2.py encapsulates random data, required for initial share separation or masking, through pre-generated random data from a separate file rdi.txt. The amount of random data to be encapsulated is dependent on the user-defined masking scheme or requirement.

6. Once a valid dinFile.txt has been created, it is highly recommended to test for proper operation of the cipher using the provided FOBOS test bench, FOBOS\_DUT\_TB.vhd. The user should insert the proper file and path for dinFile.txt in the FOBOS\_DUT\_TB.vhd source code, along with a corresponding destination file, doutFile.txt. If the FOBOS-instantiated authenticated cipher runs correctly in behavioral simulation, the text output into doutFile.txt should match that contained in the corresponding do.txt. No method of automatic verification of doutFile.txt with do.txt is provided – the user should perform this step manually or design a custom verification script.

7. For the behavioral simulation, the user-defined AEAD external I/O bus widths for pdi, sdi, and do (defined in [x]) are specified as generics in the FOBOS\_DUT\_TB.vhd, which is the top-level file. Note that the bus width for do\_data is defined to be the same as pdi\_data, per reference [x]. The bus width for rdi\_data (not defined in [x] but similar to pdi), is likewise defined to match the width of pdi\_data

In FOBOS\_DUT Version 4\_0 and higher, the widths of the FIFO automatically match their corresponding destination ports, i.e., the width of PDI\_FIFO is the same as pdi\_data; the width of SDI\_FIFO is the same as sdi\_data, etc.

Caution! The user must specify the desired FIFO depth for all FIFOs, including PDI\_FIFO, SDI\_FIFO, RDI\_FIFO (if used), and DO\_FIFO. The required FIFO depth is expressed by the following formula:

For example, in the case where G\_W = pdi\_data bus width = 16 bits = 2 bytes, and where the total number of bytes expected from pdi.txt = 140 bytes, then . Therefore, the FIFO depth for PDI\_FIFO should be set to 7.

Caution! The generics for FIFO depth are not available in FOBOS\_DUT\_TB.vhd; they must be set in the top module for implementation, namely FOBOS\_DUT.vhd.

Caution! It is recommended to set a minimum depth of 1 (one word) for all FIFOs, even if not in use. The FOBOS\_DUT\_Controller (finite state machine) has not been tested with FIFO depths of 0 (zero).

Caution! It is the user’s responsibility to ensure that FIFO depths are sufficient for the size of test vectors, and to ensure that the test vectors are of sufficient size to prevent FIFO starvation. If FIFO starvation occurs in simulation, the simulation will not complete, but will not report errors. If FIFO starvation occurs in actual hardware, the effect is unpredictable. Note: A common cause of starvation is insufficient random data provided to RDI\_FIFO.

Appendix

# dinFileTBGen1.py

# William Diehl

# FOBOS Team George Mason University

# Updated 02-04-2018

# creates FOBOS-ready dinFile.txt of input width NUMCHAR

# Expects pdi.txt, sdi.txt, and do.txt are 1) generated from aeadtvgen v2.0.0, and 2) are in same directory

# Creates one (1) dinFile.txt, containing NUMTRACES number of test vectors, in a "fixed versus random" format

# Creates the fvrchoice.txt file, which has a 0 for fixed vectors and one for random vectors

import os

import sys

import random

NUMCHAR = 4 # Expects input files (e.g., pdi.txt) with mimimum width of 16 bits = 4 characters

# Caution! User update is required for minimum width less than 16 bits

NUMRNDLINES = 250 # only used if RDI input is included

NUMTRACES = 2000 # Must be set to 1 for FOBOS\_DUT\_TB simulation; unlimited for actual HW test vectors

pdifilename = "pdi.txt"

sdifilename = "sdi.txt"

dofilename = "do.txt"

rdifilename = "rdi.txt" # optional

dinFileName = "dinFile.txt"

fvrchoicefilename = "fvrchoicefile.txt"

hexlist = ['A','B','C','D','E','F']

def returnHex(x):

if (x < 10):

s = str(x)

else:

s = hexlist[x - 10]

return s

# return a 2-digit hex string

def returnHex2(x):

if (x < 16):

s = '0' + returnHex(x)

else:

s = returnHex(x/16) + returnHex(x%16)

return s

# return a 4-digit hex string

def returnHex4(x):

if (x < 16):

s = '000' + returnHex(x)

else:

if (x < 256):

s = '00' + returnHex(x/16) + returnHex(x%16)

else:

if (x < 4096):

y1 = x/256

y2 = x - 256\*y1

s = '0' + returnHex(y1) + returnHex(y2/16) + returnHex(y2%16)

else:

y1 = x/4096

y2 = x - 4096\*y1

y3 = y2/256

y4 = y2 - y3\*256

s = returnHex(y1) + returnHex(y3) + returnHex(y4/16) + returnHex(y4%16)

return s

def dinFileGen(tracenum):

NUMBYTES = 0

# parse pdi.txt

# first pass

pdifile = open(pdifilename,'r')

t = pdifile.readline()

while (t != ""):

if ((t.find('#')==0) or (t.lstrip()=="")):

t = pdifile.readline()

else:

delim = t.find(' = ')

t = t[delim+3:len(t)-1]

NUMBYTES = NUMBYTES + len(t)/2

t = pdifile.readline()

pdifile.close()

# write pdi.txt header and number of bytes

writefile.write('00C0')

writefile.write(returnHex4(NUMBYTES))

# second pass

pdifile = open(pdifilename,'r')

if (random.random()<0.5):

# fixed pdi

fvrchoicefile.write('0')

t = pdifile.readline()

while (t != ""):

if ((t.find('#')==0) or (t.lstrip()=="")):

# print t

t = pdifile.readline()

else:

delim = t.find(' = ')

t = t[delim+3:len(t)-1]

for i in range (0,len(t)/NUMCHAR):

u = t[i\*4:i\*4+4]

#print u

writefile.write(u)

# print t

t = pdifile.readline()

else:

# random pdi

fvrchoicefile.write('1')

t = pdifile.readline()

while (t != ""):

if ((t.find('#')==0) or (t.lstrip()=="")):

t = pdifile.readline()

else:

if (t.find('DAT = ')<0):

# no DAT statement, replace the text with pdi.txt

delim = t.find(' = ')

t = t[delim+3:len(t)-1]

for i in range (0,len(t)/NUMCHAR):

u = t[i\*4:i\*4+4]

#print u

writefile.write(u)

else:

# DAT statement - replace DAT with random txt of same length

delim = t.find(' = ')

t = t[delim+3:len(t)-1]

for i in range(0,len(t)/NUMCHAR):

u = returnHex2(int(random.random()\*256))

u = u + returnHex2(int(random.random()\*256))

writefile.write(u)

t = pdifile.readline()

pdifile.close()

print ("NUMBYTES of pdi.txt = " + str(NUMBYTES))

# parse sdi.txt

# first pass

NUMBYTES = 0

sdifile = open(sdifilename,'r')

t = sdifile.readline()

while (t != ""):

if ((t.find('#')==0) or (t.lstrip()=="")):

t = sdifile.readline()

else:

delim = t.find(' = ')

t = t[delim+3:len(t)-1]

NUMBYTES = NUMBYTES + len(t)/2

t = sdifile.readline()

sdifile.close()

# write sdi.txt header and number of bytes

writefile.write('00C1')

writefile.write(returnHex4(NUMBYTES))

# second pass

sdifile = open(sdifilename,'r')

t = sdifile.readline()

while (t != ""):

if ((t.find('#')==0) or (t.lstrip()=="")):

# print t

t = sdifile.readline()

else:

delim = t.find(' = ')

t = t[delim+3:len(t)-1]

for i in range (0,len(t)/NUMCHAR):

u = t[i\*4:i\*4+4]

#print u

writefile.write(u)

# print t

t = sdifile.readline()

sdifile.close()

print ("NUMBYTES of sdi.txt = " + str(NUMBYTES))

# parse rdifile

# uncomment this section to include RDI input

# write preamble

# writefile.write('00C2')

# writefile.write(returnHex4(NUMRNDLINES\*2))

# copy next seciton of rdi.txt to dinFile.txt

# for i in range(tracenum\*NUMRNDLINES, (tracenum\*NUMRNDLINES) + NUMRNDLINES):

# t = rdifile.readline()

# writefile.write(t[0:len(t)-1])

# parse do.txt

# only one pass

NUMBYTES = 0

dofile = open(dofilename,'r')

t = dofile.readline()

while (t != ""):

if ((t.find('#')==0) or (t.lstrip()=="")):

t = dofile.readline()

else:

delim = t.find(' = ')

t = t[delim+3:len(t)-1]

NUMBYTES = NUMBYTES + len(t)/2

t = dofile.readline()

dofile.close()

# write do.txt header and number of bytes

print ("NUMBYTES of do.txt = " + str(NUMBYTES))

writefile.write('0081')

writefile.write(returnHex4(NUMBYTES))

# write start command

writefile.write('0080')

#writefile.write('0001\n') # uncomment if producing test vectors to run on actual hardware

writefile.write('0001') # uncomment if using FOBOS\_DUT\_TB simulation

print ("Test Vector # " + str(tracenum) + " created.")

# main

writefile = open(dinFileName,'w')

fvrchoicefile = open(fvrchoicefilename,'w')

#rdifile = open(rdifilename,'r') # uncomment to include rdi.txt

TRACENUM = 0

for i in range (0, NUMTRACES):

dinFileGen(TRACENUM)

TRACENUM = TRACENUM + 1

#rdifile.close() # uncomment to include rdi.txt

fvrchoicefile.close()

writefile.close()

# dinFileTBGen1.py

# William Diehl

# FOBOS Team George Mason University

# Updated 02-04-2018

# creates FOBOS-ready dinFile.txt of input width NUMCHAR

# Expects pdi.txt, sdi.txt, and do.txt are 1) generated from aeadtvgen v2.0.0, and 2) are in same directory

# Creates one (1) dinFile.txt, containing NUMTRACES number of test vectors, in a "fixed versus random" format

# Creates the fvrchoice.txt file, which has a 0 for fixed vectors and one for random vectors

import os

import sys

import random

NUMCHAR = 4 # Expects input files (e.g., pdi.txt) with mimimum width of 16 bits = 4 characters

# Caution! User update is required for minimum width less than 16 bits

NUMRNDLINES = 250 # only used if RDI input is included

NUMTRACES = 2000 # Must be set to 1 for FOBOS\_DUT\_TB simulation; unlimited for actual HW test vectors

pdifilename = "pdi.txt"

sdifilename = "sdi.txt"

dofilename = "do.txt"

rdifilename = "rdi.txt" # optional

dinFileName = "dinFile.txt"

fvrchoicefilename = "fvrchoicefile.txt"

hexlist = ['A','B','C','D','E','F']

def returnHex(x):

if (x < 10):

s = str(x)

else:

s = hexlist[x - 10]

return s

# return a 2-digit hex string

def returnHex2(x):

if (x < 16):

s = '0' + returnHex(x)

else:

s = returnHex(x/16) + returnHex(x%16)

return s

# return a 4-digit hex string

def returnHex4(x):

if (x < 16):

s = '000' + returnHex(x)

else:

if (x < 256):

s = '00' + returnHex(x/16) + returnHex(x%16)

else:

if (x < 4096):

y1 = x/256

y2 = x - 256\*y1

s = '0' + returnHex(y1) + returnHex(y2/16) + returnHex(y2%16)

else:

y1 = x/4096

y2 = x - 4096\*y1

y3 = y2/256

y4 = y2 - y3\*256

s = returnHex(y1) + returnHex(y3) + returnHex(y4/16) + returnHex(y4%16)

return s

def dinFileGen(tracenum):

NUMBYTES = 0

# parse pdi.txt

# first pass

pdifile = open(pdifilename,'r')

t = pdifile.readline()

while (t != ""):

if ((t.find('#')==0) or (t.lstrip()=="")):

t = pdifile.readline()

else:

delim = t.find(' = ')

t = t[delim+3:len(t)-1]

NUMBYTES = NUMBYTES + len(t)/2

t = pdifile.readline()

pdifile.close()

# write pdi.txt header and number of bytes

writefile.write('00C0')

writefile.write(returnHex4(NUMBYTES))

# second pass

pdifile = open(pdifilename,'r')

if (random.random()<0.5):

# fixed pdi

fvrchoicefile.write('0')

t = pdifile.readline()

while (t != ""):

if ((t.find('#')==0) or (t.lstrip()=="")):

# print t

t = pdifile.readline()

else:

delim = t.find(' = ')

t = t[delim+3:len(t)-1]

for i in range (0,len(t)/NUMCHAR):

u = t[i\*4:i\*4+4]

#print u

writefile.write(u)

# print t

t = pdifile.readline()

else:

# random pdi

fvrchoicefile.write('1')

t = pdifile.readline()

while (t != ""):

if ((t.find('#')==0) or (t.lstrip()=="")):

t = pdifile.readline()

else:

if (t.find('DAT = ')<0):

# no DAT statement, replace the text with pdi.txt

delim = t.find(' = ')

t = t[delim+3:len(t)-1]

for i in range (0,len(t)/NUMCHAR):

u = t[i\*4:i\*4+4]

#print u

writefile.write(u)

else:

# DAT statement - replace DAT with random txt of same length

delim = t.find(' = ')

t = t[delim+3:len(t)-1]

for i in range(0,len(t)/NUMCHAR):

u = returnHex2(int(random.random()\*256))

u = u + returnHex2(int(random.random()\*256))

writefile.write(u)

t = pdifile.readline()

pdifile.close()

print ("NUMBYTES of pdi.txt = " + str(NUMBYTES))

# parse sdi.txt

# first pass

NUMBYTES = 0

sdifile = open(sdifilename,'r')

t = sdifile.readline()

while (t != ""):

if ((t.find('#')==0) or (t.lstrip()=="")):

t = sdifile.readline()

else:

delim = t.find(' = ')

t = t[delim+3:len(t)-1]

NUMBYTES = NUMBYTES + len(t)/2

t = sdifile.readline()

sdifile.close()

# write sdi.txt header and number of bytes

writefile.write('00C1')

writefile.write(returnHex4(NUMBYTES))

# second pass

sdifile = open(sdifilename,'r')

t = sdifile.readline()

while (t != ""):

if ((t.find('#')==0) or (t.lstrip()=="")):

# print t

t = sdifile.readline()

else:

delim = t.find(' = ')

t = t[delim+3:len(t)-1]

for i in range (0,len(t)/NUMCHAR):

u = t[i\*4:i\*4+4]

#print u

writefile.write(u)

# print t

t = sdifile.readline()

sdifile.close()

print ("NUMBYTES of sdi.txt = " + str(NUMBYTES))

# parse rdifile

# uncomment this section to include RDI input

# write preamble

writefile.write('00C2')

writefile.write(returnHex4(NUMRNDLINES\*2))

# copy next seciton of rdi.txt to dinFile.txt

for i in range(tracenum\*NUMRNDLINES, (tracenum\*NUMRNDLINES) + NUMRNDLINES):

t = rdifile.readline()

writefile.write(t[0:len(t)-1])

# parse do.txt

# only one pass

NUMBYTES = 0

dofile = open(dofilename,'r')

t = dofile.readline()

while (t != ""):

if ((t.find('#')==0) or (t.lstrip()=="")):

t = dofile.readline()

else:

delim = t.find(' = ')

t = t[delim+3:len(t)-1]

NUMBYTES = NUMBYTES + len(t)/2

t = dofile.readline()

dofile.close()

# write do.txt header and number of bytes

print ("NUMBYTES of do.txt = " + str(NUMBYTES))

writefile.write('0081')

writefile.write(returnHex4(NUMBYTES))

# write start command

writefile.write('0080')

#writefile.write('0001\n') # uncomment if producing test vectors to run on actual hardware

writefile.write('0001') # uncomment if using FOBOS\_DUT\_TB simulation

print ("Test Vector # " + str(tracenum) + " created.")

# main

writefile = open(dinFileName,'w')

fvrchoicefile = open(fvrchoicefilename,'w')

rdifile = open(rdifilename,'r') # uncomment to include rdi.txt

TRACENUM = 0

for i in range (0, NUMTRACES):

dinFileGen(TRACENUM)

TRACENUM = TRACENUM + 1

rdifile.close() # uncomment to include rdi.txt

fvrchoicefile.close()

writefile.close()