9th International Conference on Security, Privacy and Applied Cryptographic Engineering

SPACE, December 5-7, 2019

Authors: Vidya Govindan, Sandhya Koteshwara, Amitabh Das, Keshab K. Parhi, Rajat Subhra Chakraborty

Affiliations: Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, Kharagpur, West Bengal, India and Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN 55455, USA

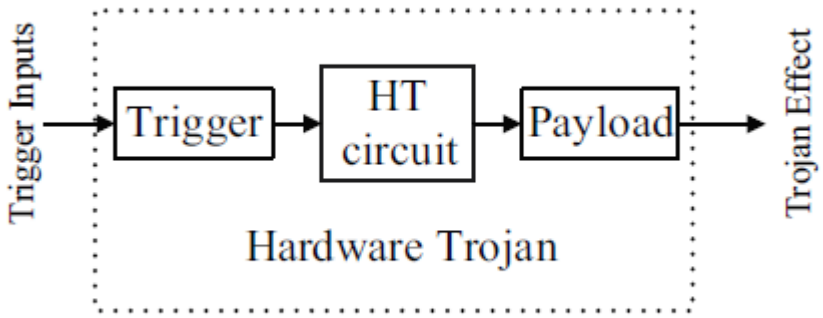
Table of Contents

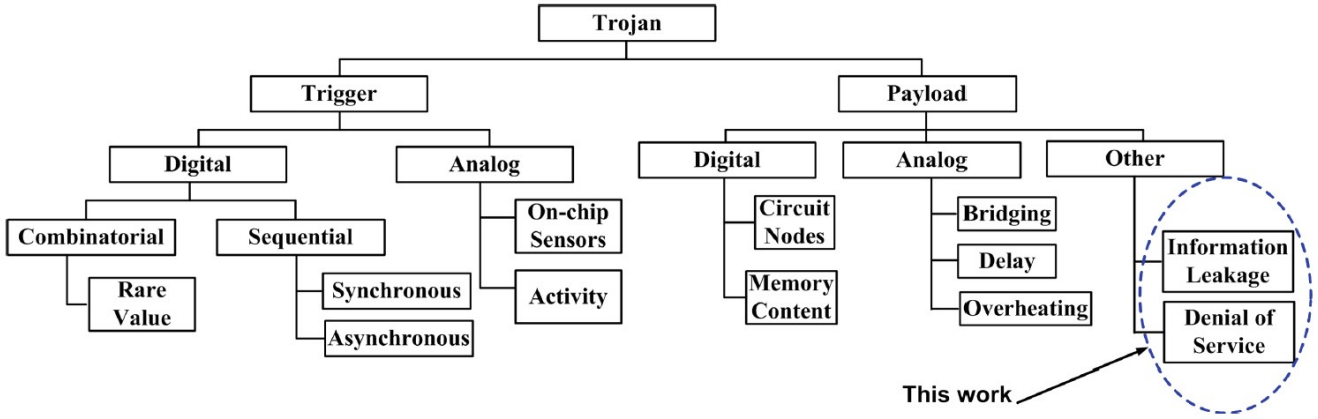
Motivation

- Wired Ethernet or Local Area Network (LAN): Backbone of Internet Infrastructure
- Internet of Things:
  - Home and Office automation, Data Centers, In-car automotive networks, Smart Grid etc.
  - Increased growth and demand of internet bandwidth
  - LAN capacity: 10Mbps to 10 Gbps from 1980s to 2018
- Higher BW ⇒ Larger Ethernet segments ⇒ Need for simple, effective Switches
- Architectural security of computer networks ⇒ Not included originally
- In the 7 layer OS [illegible] ayer ⇒ Security weak link

Hardware Trojan

- A Hardware Trojan (HT) is defined as a malicious, intentional modification of a circuit design that results in undesired behavior when the circuit is deployed.
  - Trojan trigger: circuit activation mechanism
  - Trojan payload: the part of the circuit or the functionality affected
- Types of HTs : classificatio



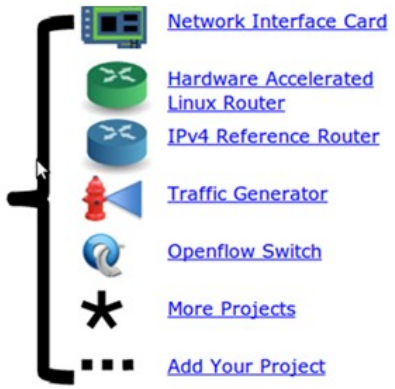


FPGA and Networking

- Software based solutions to securing networks ␣ overburdened
- Widening gap between execution speed of security software and amount of data to be processed
- Solution ␣ Hardware implementation of security functions
- Field Programmable Gate Arrays (FPGAs):
  - Reconfigurable hardware devices
  - Flexibility of software + Parallelism of hardware
  - Used in many network applications: protocol wrapper, packet classification, intrusion detection etc.
  - NetFPGA platform ␣ best example of incorporating FPGAs into network implementations

NetFPGA

- NetFPGA = Networked FPGA
  - A line-rate, flexible, open networking platform for teaching and research
  - NetFPGA 1G(2006)-NetFPGA-1G-CML(2014)
  - NetFPGA 10G(2010)-NetFPGA SUME(2014)

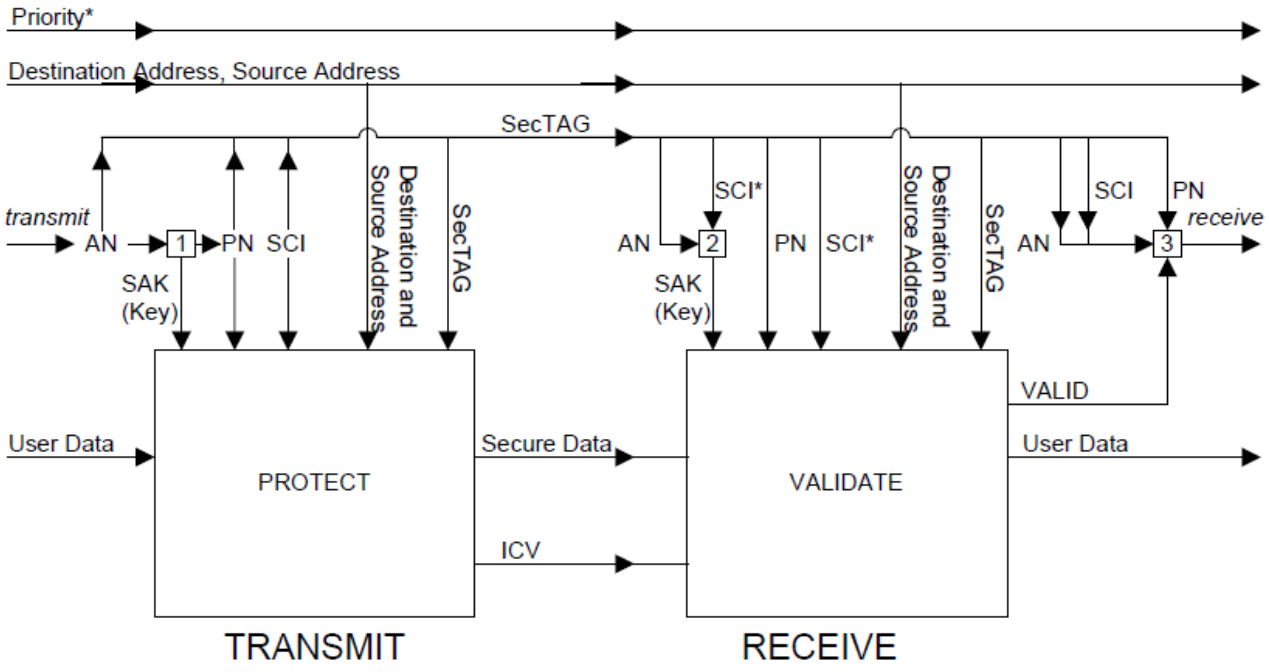


- NetFPGA SUME
  - Virtex-7 XC7V690T FPGA Development Board
  - 4 SFP+ (10Gbps) ports, 2 SATA III ports,
  - three 72 MBit QDRII+ SRAMs and two 4GB DDR3 SODIMMs etc.

Media Access Control Security (MACSec)

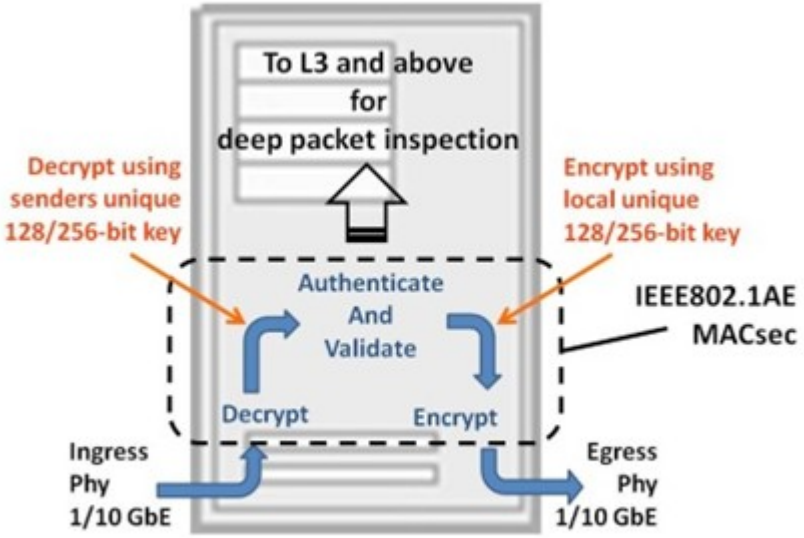
- IEEE 802.1 standard – secure communication between stations connected to the same LAN forming a set of trusted entities. MACSec was first published in 2006 followed by three subsequent amendments:
  - 802.1AEbn-2011 , 802.1AEbw-2013 and 802.1AEcg-2017
- It involves two standards:
  1. IEEE 802.1AE – specifies provisions for connectionless user data confidentiality, frame data integrity, and data origin authenticity. (data plane)
  2. IEEE 802.1X – specifies a general method for port-based network access control and defines MACSec Key Agreement protocol (MKA). (control plane)
- Every node capable of participating in an instance of the secure MAC Service comprises both a MAC Security Key Agreement Entity (KaY) and a MAC Security Entity (SecY).

MACSec Operation



* Priority can be changed by media access method or receiving system and is not protected
* The SCI is extracted from the SCI field of the SecTAG if present. A value conveyed by key agreement (point-to-point only) is used otherwise.

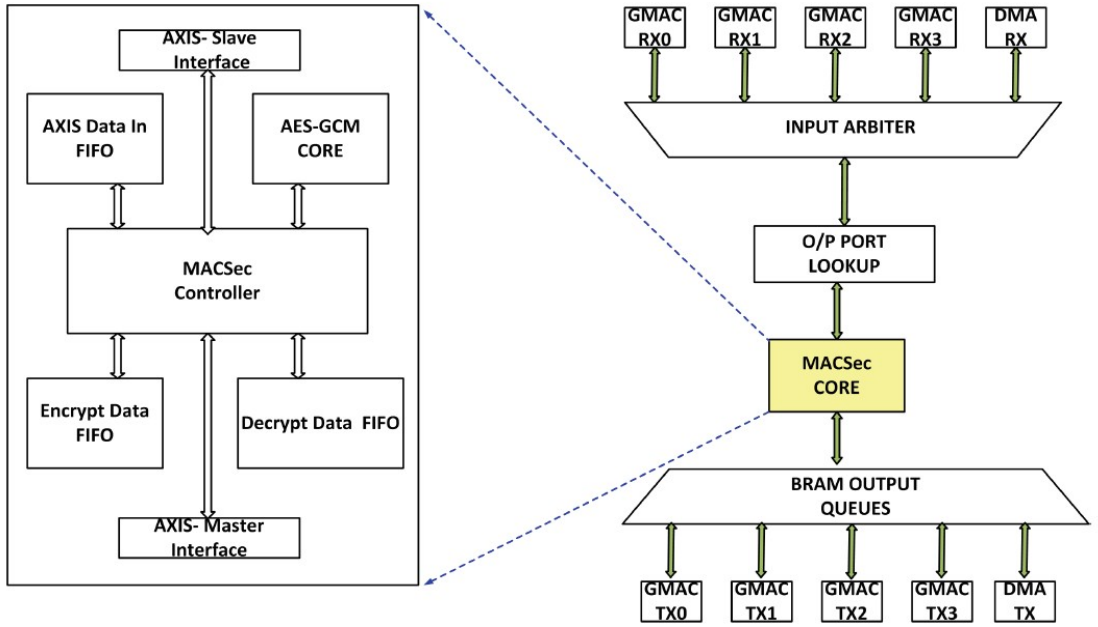
Functions
1 Lookup Key and next PN for transmit SA identified by AN
2 Lookup Key PN for receive SA identified by SCI, AN
3 Discard if received frame not VALID. Discard if replay check of PN for receive SA identified by SCI, AN fails. Updated replay check.



MACSec Switch

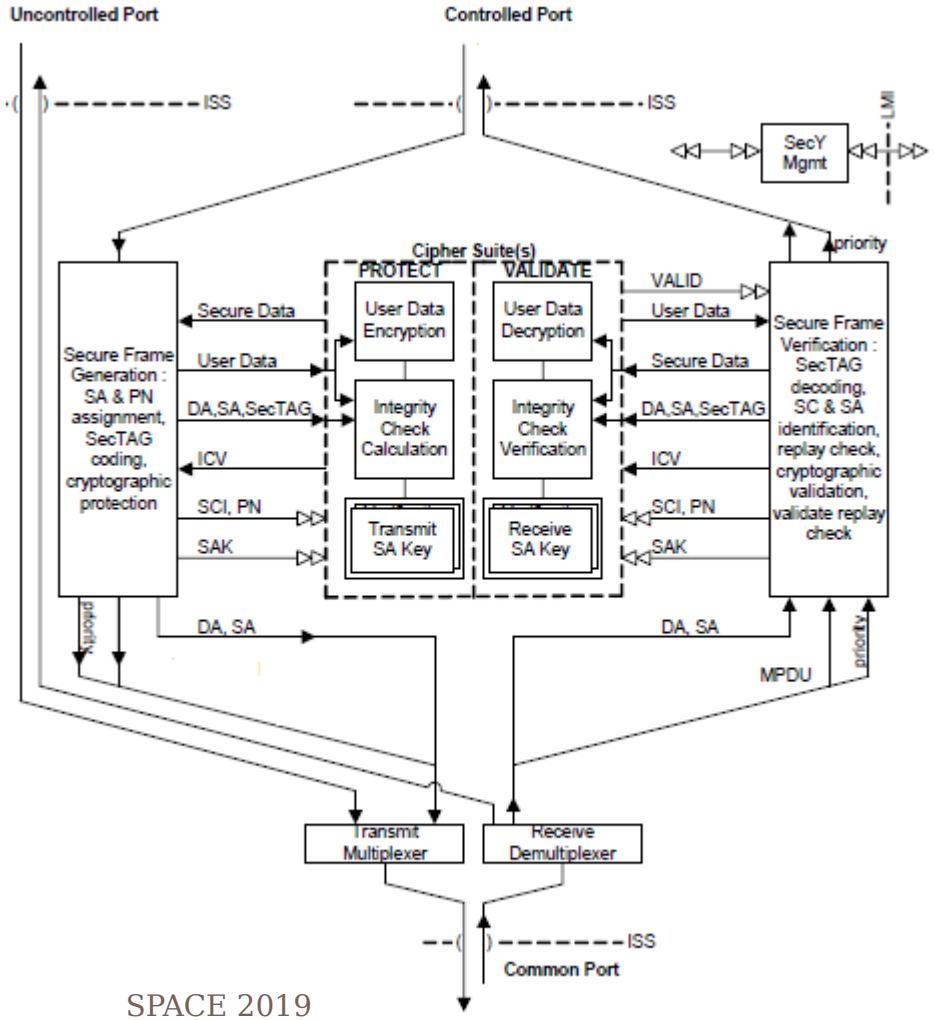
System Architecture

- Based on the NetFPGA SUME's reference switch design
  - Packet-based module interface
  - Pluggable design with register infrastructu
  - Five stages :
    - Input ports
    - Input Arbiter
    - Output Port Lookup
    - BRAM Output Queues
    - Output ports
  - User defined module : MACSec Core
    - AES-GCM-128bit Core
    - MACSec controller
    - Encrypt and Decrypt FIFOs



- Xilinx Microblaze subsystem, including a BRAM block and its controller.

MACSec Controller



Procedure 1 MACSec Controller

Input: Ingress Ethernet Packet with MACSec Header

Output: Egress Ethernet Packet with MACSec Header

1: MACSec Header Extraction
2: User data decryption with secure frame verification
3: User data encryption with secure frame generation
4: AXI4-Stream Master Interface signal generation

Threat Model

- Assumptions:
  - When ? Hardware design can be covertly compromised by HTs inserted into Register Transfer Level (RTL) source code or gate level netlist.
  - Who ? HTs are introduced by one or more rogue designers in the in-house design team, or integrated into the design with third-party IP (3PIP) cores.
  - How ? An attacker having physical access to the board or an attacker who can reconfigure the FPGA bitstream remotely through partial reconfiguration during a design update could introduce the proposed HT.
- Attack Scenario:
  - The simple ethernet switch is deployed in a data center.
  - To support higher speed and better security policies, data centers are moving towards Layer-2 secure connectivity inside the center.
  - The Layer-2 security policy facilitates updating the ethernet switch core with MACSec functionality. While adding the MACSec functionality, the new design also opens up the possibility of a HT being introduced that can modify the flow of packets through the switch.
- From a network perspective, this can be viewed as an example of a network packet interception and removal, or network traffic redirection attack leading to Denial of Service (DoS), while from a security perspective, this is an information leakage attack.

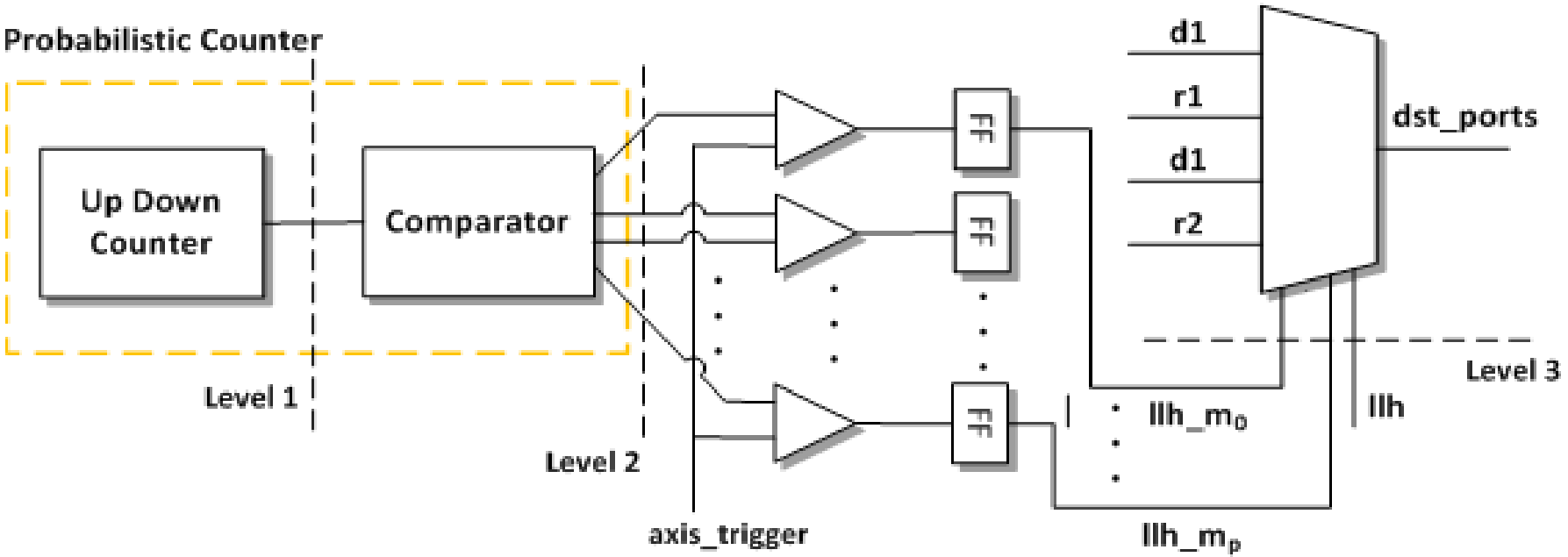
Hardware Trojan Design

- Trojan Location :
  - HT is embedded in the Output Port Lookup module of the reference switch design.
  - The Output Port Lookup module consists of two main submodules:
    1. Ethernet parser: extracts the source MAC address, destination MAC address and the source port fields from the input packet.
    2. Learning Content Addressable Memory (CAM) :
       - These fields are then looked up in the CAM, implementing a lookup table (LUT).
       - If the result is a hit, the packet is sent to the destination ports indicated by the lookup (except for the source port), if the result is a miss, the destination ports are set to broadcast to all output ports (except for the source port).
       - If the source MAC is not found in the LUT, it is learned for future lookups.

Hardware Trojan Design

- Trojan Premise:
  - Traditionally packet counting has been employed in network devices such as routers for characterizing and controlling traffic. For diagnostic and record keeping purposes, it may be desirable for the router to keep a running total of the number of dropped packets.
  - We assume that one such counter pre-existed in the network switch design, and has been replaced by a malicious probabilistic counter (as part of the inserted HT).
  - In this design, the probabilistic counter is enabled whenever an incoming packet has one of the following EtherType field values:
    - MACSec packet - 0x88E5
    - IPv4 packet - 0x0800
    - IPv6 packet - 0x86DD
    - ARP packet - 0x0806

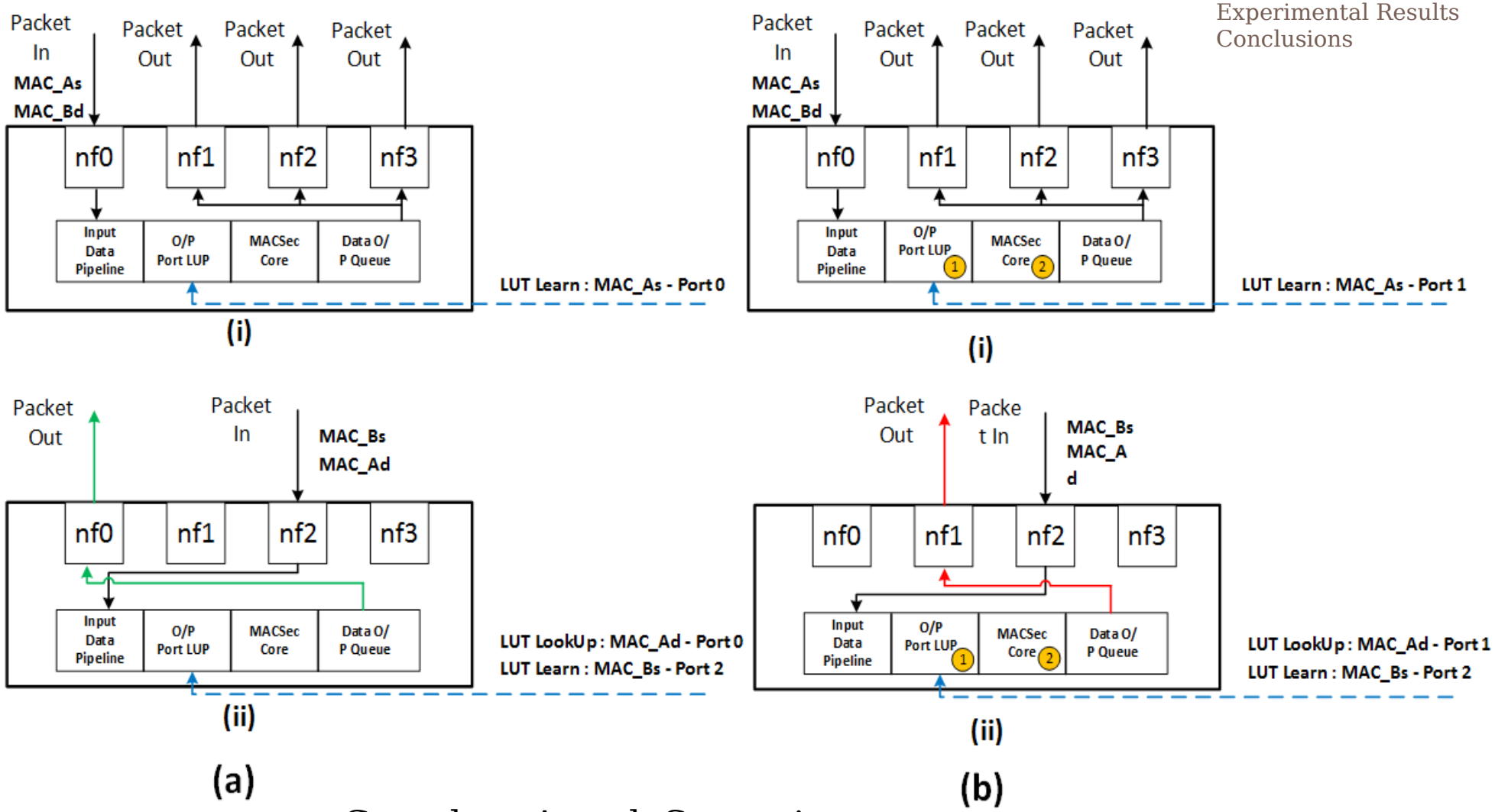
Hardware Trojan Design



Probabilistic counter based **Tro**jan (ProTro) Circuit

Hardware Trojan Design

- Probabilistic Counter based Trojan Circuit (ProTro):
  - Trigger: Multi-level trigger mechanism
    - Level 1 trigger : probabilistic counter + PRNG
    - Level 2 trigger : axis_trigger , AXI4 stream slave signals of Output port lookup module
      axis_trigger = s_axis_tvalid & s_axis_tready & (s_axis_tlast)
    - Level 3 trigger : llh_m, i.e., the malicious LUT lookup hit signal forms
  - Payload: The HT payload comprises of the registers lut and dst_ports of the Output port Looking module which are responsible for updating the output ports.
- Counter : 4-tuple *(n, I, D, T)* ␣ (8, 1, 1, 0x7C1FE0)
  - *n : number of bit in counter*
  - *I : Increment value* ␣ *when* switch receives an IPv4 or IPv6 packet
  - *D : Decrement value* ␣ when switch receives a MACSec or ARP packet
  - *T : Threshold / Trigger value* ␣ counter is updated only when the PRNG output value is greater than the threshold value
- PRNG ␣ a 23-bit Fibonacci linear feedback shift register (LFSR) with primitive polynomial $x^{23} + x^{18} + 1$.



Complete Attack Scenario at the Network Switch

Experimental Results

- The proposed architecture is implemented in Verilog on Xilinx's Virtex-7 based NetFPGA SUME board.
- Xilinx's Vivado 2016.4 tool's ISIM simulator was used to carry out the simulation of the design.
- A python based testing environment is used based on the NetFPGA reference switch project to verify the functionality of the design. Packet stimuli and verification scripts are built upon scapy, using which a MACSec packet is constructed.

Experimental Results

Resource Overhead

Resource	Without HT	With HT	HT Insertion Overhead (%)
LUT	48650	48744	0.19
LUTRAM	1880	1880	0.00
FF	70264	70376	0.16
BRAM (kbit)	213.50	213.50	0.00
BUFG	22	22	0.00
MMCM	3	3	0.00
PCIe	1	1	0.00
Power (watt)	6.90	6.90	0.00

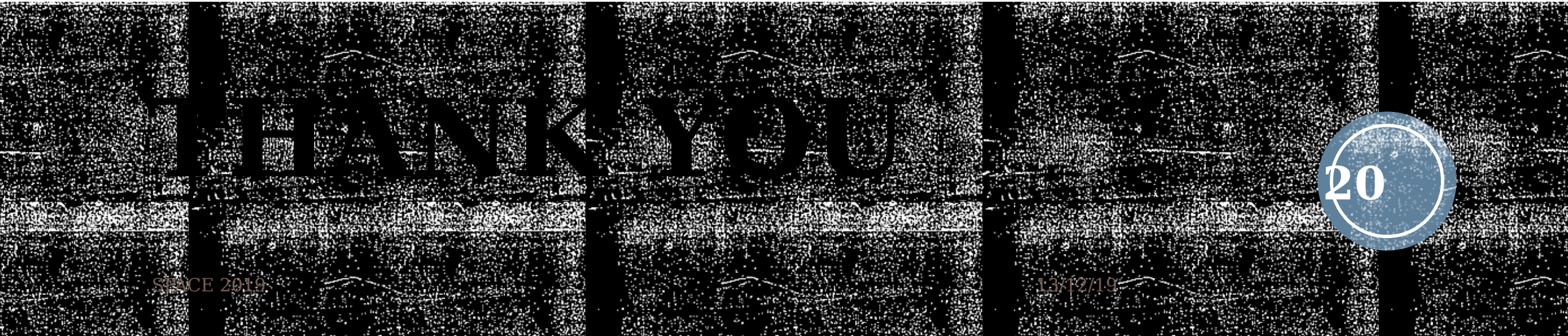
Trust- Hub Benchmark Overhead

Trust-Hub Trojans	Resource Overhead (%)			
	LUT	LUTRAM	FF	BRAM
MC8051T200	0.80	0	0	0
MC8051T300	0.19	0	0	0
MC8051T400	15.30	0	1.45	0
MC8051T500	2.82	0	0	0
MC8051T600	0.50	0	0	0
MC8051T700	3.74	0	0	0
MC8051T800	0.32	0	0	0
Average	3.38	0	0.20	0
PIC16F84T100	6.25	0	0	0
PIC16F84T200	1.30	38.40	3.10	0
PIC16F84T300	1.32	38.40	0.37	0
PIC16F84T400	1.20	38.40	37.00	0
Average	2.51	28.80	10.12	0

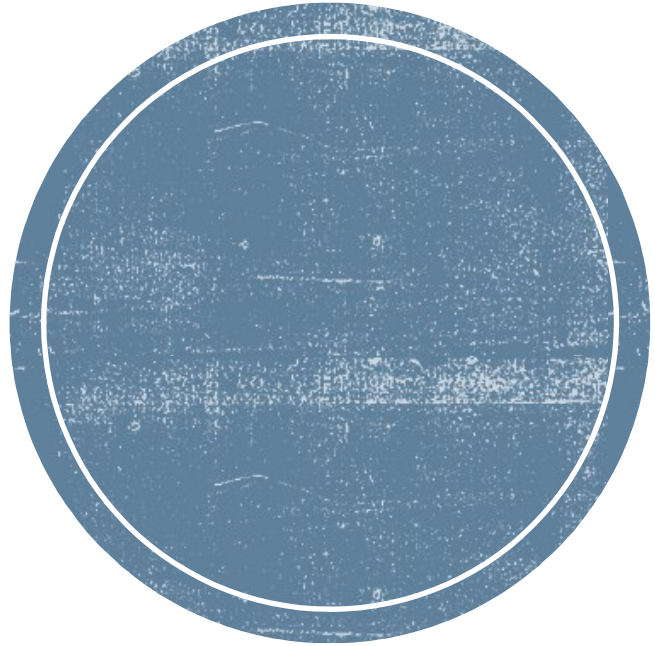
Conclusion

- A hardware implementation of the MACSec protocol over an ethernet switch reference design targeting the NetFPGA-SUME platform.
- Proposed a probabilistic counter based HT attack on MACSec enabled ethernet switch design.
- Normal functionality of the switch is altered such that an eavesdropper user attached to one of the ports of the switch can receive the ethernet traffic intended for another user.
- HT has negligible overhead and detect-ability as compared to TrustHub benchmark trojans.
- Our future work will involve integrating the MKA protocol to develop a complete hardware-software co-design for FPGA based ethernet security, and development of advanced HT detection techniques to detect HTs of the type proposed in the paper.

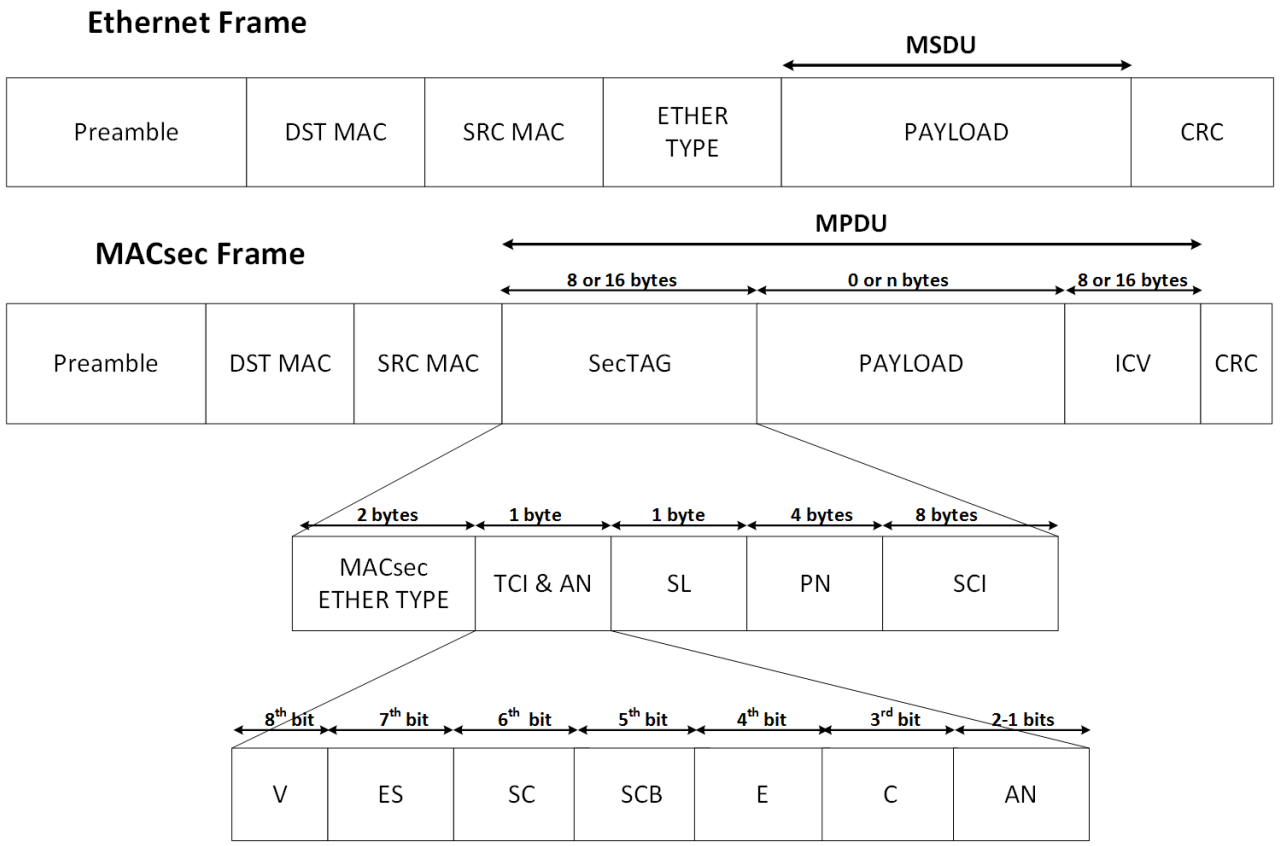
Questions?



BACKUP SLIDES



MACSec Data Frame



MACSec Definitions

- MAC Security Entity (SecY): The entity that operates the MAC Security protocol within a system.
- secure Connectivity Association (CA): A security relationship, established and maintained by key agreement protocols, that comprises a fully connected subset of the service access points in stations attached to a single LAN that are to be supported by MACsec.
- Secure Association (SA): A security relationship that provides security guarantees for frames transmitted from one member of a CA to the others. Each SA is supported by a single secret key, or a single set of keys .
- Secure Channel Identifier (SCI): A globally unique identifier for a secure channel, comprising a globally unique MAC Address and a Port Identifier, unique within the system allocated that address.

MACSec Definitions

- Association Number (AN): A number that is concatenated with the Secure Channel Identifier to identify a Secure Association.
- Secure Association Key (SAK): The secret key used by an SA.
- Secure Channel (SC): A security relationship used to provide security guarantees for frames transmitted from one member of a CA to the others. An SC is supported by a sequence of SAs thus allowing the periodic use of fresh keys without terminating the relationship.
- MAC Security TAG (SecTAG): A protocol header, comprising a 8 or 16 bytes and beginning with an EtherType, and is used to provide security guarantees.
- Integrity check value (ICV): A value that is derived by performing an algorithmic transformation on the data unit for which data integrity services are provided.

Verilog Code Modifications

MAC Address learning

```
//MAC Address learning
always @ (posedge clk) begin
 if (reset)
  lut[i] <= {(56){1'b0}};
 else if (lookup_req) begin
   if ((lut[i][47:0] == src_mac) ||
   (~lut_learn_hit[15] && (lut_wr_addr == i)))
//   lut[i] <= {(src_port), src_mac};
     lut[i] <= {src_port[7:3],src_port[0],src_port[1]
     ,src_port[2], src_mac};
   end
 end
```

Destination Port Lookup Update

```
//Destination Port Lookup Update
always (posedge clk) begin
 if(reset)
  dst_ports  <= {8{1'b0}};
 else begin
  if (lookup_req) begin
   //dst_ports <= (lut_lookup_hit[15]) ?
  // (rd_oq[15][7:0] & ~(src_port)) :
  // (DEFAULT_MISS_OUTPUT_PORTS & ~src_port);
     case ({llh_m,llh})
      2'b00: begin dst_ports <= d1; end
      2'b01: begin dst_ports <= r1; end
      2'b10: begin dst_ports <= d1; end
      2'b11: begin dst_ports <= r2; end
     endcase
    end
  end
end
```

AXI-4 Streaming Waveform