A Bandwidth-Aware Authentication Design for Packet Integrity Detection in Untrusted Third-party NoCs

Mubashir Hussain Hui Guo

School of Computer Science and Engineering
University of New South Wales, Australia
{mhussain,huig}@cse.unsw.edu.au

Technical Report
UNSW-CSE-TR-201801
April 2018

UNSW
SYDNEY

School of Computer Science and Engineering
The University of New South Wales
Sydney 2052, Australia

Abstract

Bandwidth is a fundamental issue in the network-on-chip (NoC) design and increasing use of third-party NoC IPs adds the security as another design concern. The third-party NoC may have hardware Trojans (small malicious hardware modifications) that can break the confidentiality and integrity of the data transferred over the NoC, thereafter exposing the system to varied attacks. Basically, encryption can be applied to protect the data confidentiality; For the data integrity, authentication is often used. A general authentication approach for network communication is using a tag to identify the data; The tag is attached to the data at the source and the data is verified against the tag at the destination. If data is altered, its tag becomes invalid and the change can be detected. For the effectiveness of the authentication, it is desired that the tag be sufficiently large. But large tags, when transferred over the NoC, will consume a considerable network bandwidth, potentially conflicting with the system bandwidth requirement. In this work, we propose a packet data authentication design, where a packet is progressively authenticated against a set of small tag segments such that the authentication overhead is low yet the effective tag size is large. Our experiments on the 8x8 mesh NoC for a set of applications show that with an 8-bit tag size, our authentication design can achieve a detection rate of 99.99%, better than the traditional design of 32-bit tags. Meanwhile, the significant overheads, about 36% on bandwidth, 12% on packet latency, and 56% on area that have been incurred by the traditional design, can be saved from our approach.

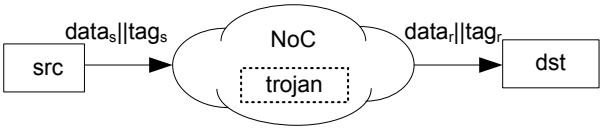


Figure 1.1: Traditional authentication approach: tagging data for integrity checking

1 Introduction

With the advancement in the semiconductor technology, many intellectual property (IP) cores can be integrated on a single chip. A typical example is the MPSoC (Multi-Processor System-on-Chip) design. MPSoC contains multiple processors, memory elements, power management blocks and I/O interfaces. To meet the high bandwidth needs of the MPSoCs, the router based scalable communication architecture, network-on-chip (NoC), is often preferred.

To reduce the cost and the time-to-market of MPSoCs, designers often use off-the-shelf components, for example, using NoC (Network-on-Chip) IP from the third-party vendors. However, concerns about the security of the third-party components become increasingly salient. The vendor can modify the hardware components in the design to leak sensitive information and/or tamper the system data for various attacks.

In this report, we target an MPSoC system that uses the third-party NoC IP. We assume that the NoC may contain a hardware Trojan (a malicious hardware component embedded in the NoC) and the Trojan can be activated and de-activated during the system operation. Once activated, the Trojan attempts to modify the packet data transferred over the NoC, such as changing the destination in a packet to divert the packet to a different node, or altering the packet data of an online money transaction.

For the integrity verification of a packet, a simple way is to tag the packet data at the source node; both the data and its tag are then transferred over the network to the destination, as illustrated in Figure 1.1. At the destination, the packet data is authenticated against the tag. If the data is altered, the tag becomes invalid. The authentication calculates the tag value of the received data and compares it with the tag in the packet. If they are different, the packet is deemed tampered. However, the attack can pass the authentication and go undetected if the tag of the altered data collides with the tag of the original data. Therefore, it is important to design tags that have no or low tag collisions, which entails a big tag size. However big tags will consume considerable packet space, greatly degrading the network traffic bandwidth.

In this report, we aim for a packet authentication design that is low in bandwidth overhead yet highly effective in verification of the packet data transferred over the third party NoC. Our main contributions are

- We proposed a novel progressive packet authentication design, where a packet is progressively authenticated during its transmission. For a given packet, it has a different tag segment for individual network node. When the packet comes to a node, the packet is verified against the specific tag segment. Each tag segment is small (hence incurs a low cost), but the effective tag used for the overall packet authentication is formed by a

chain of such tag segments and can be sufficiently large.

- We undertook a case study on the 2D mesh NoC. For the randomly distributed packets, we identified the most effective attack position for a Trojan. For the Trojan in this position, we investigated and verified the effectiveness of the proposed authentication design.

The rest of the report is organized as follows. Section 2 reviews existing work related to data authentication. Our progressive packet authentication design is proposed in Section 3. The case study based on the 2D-mesh NoC is presented in Section 4. Section 5 presents the experimental results and the report is concluded in Section 6.

2 Related work

The data integrity checking has been well studied, especially for the messages transferred over the network. The common approach for message authentication is to use a MAC (short for message authentication code, also called tag in the data authentication), to identify a message. Most MAC schemes divide messages into blocks and the MAC generation is made of operations on blocks. Depending on how those operations are structured between blocks, MAC schemes can be divided into iterated MAC [24, 4, 20, 18, 5, 15] and parallel MAC [2, 6, 21].

In the iterated MAC design, the operations on blocks are performed in sequence. The result of one block relies on the output of previous blocks. In the parallel MAC scheme, all blocks of a message are processed in parallel. In terms of data authentication, designs can be cryptographic hash function based (such as CRC [19]), Added Redundancy Explicit Authentication ((AREA) based[9], and MAC (Message Authentication Code) scheme based.

In [25], Yan et al. proposed an authentication and encryption design (**AE**) to protect the confidentiality and integrity of memory data with Galois/Counter Mode(GCM) operations proposed in [17].

In [22] Rogers and Milenkovic proposed another AE design aiming to protect both code and data in the memory. A PMAC [6] scheme is used for memory data authentication. The tag for a memory data is associated with the data address and a sequence number, and the sequence number is unique to each tag generation for a given memory location. Bellare and Namprempre in [3] pointed out that this design is not as secure as [25].

In [12], the authors proposed a tag generation design that is based on a sequence of randomization operations. The design aims for low tag storage consumption and on-chip area cost, and can be optimized for a given tag size.

The above mentioned approaches are mainly focused to the attacks on the off-chip network. There are also approaches proposed for packet authentication in the NoC.

Yu et al. in [27] investigated the data integrity attacks by hardware Trojans on the NoC links, where a Trojan on a link can flip the bit value of the link. They proposed a three-stage design: 1) encoding/decoding, where every output flit (a basic flow control unit in NoC) from a router is encoded using the single error correction double error detection (SECDED) code and the flit is verified at the input of the next router; the flit will be retransmitted if a two-bit error is found; 2) bits reshuffle, where for the flit retransmission, data bits are reshuffled

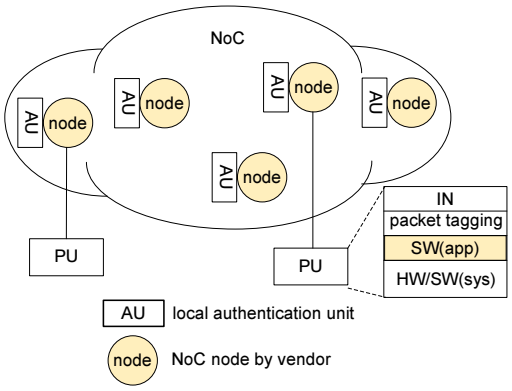


Figure 3.1: System overview

(odd bits swapped with even bits); with the reshuffle, the Trojan location on the link can be identified; 3) link isolation, where links with active HTs are isolated and a data flit is sent through HT-free links.

Similarly, Boraten et al. [8] also addressed the data integrity attacks by hardware Trojans that can inject error vectors on links. They proposed to use different error detection codes (CRC [19] or algebraic manipulation and detection) for the different level of security required by individual applications.

Frey et al. [10] proposed a dynamic flit permutation and error correcting code (ECC) techniques to protect against NoC bandwidth depletion attacks within a router, where the ECC value for a flit is generated and scrambled with flit data at the input port of NoC router. At the output port, the flit is verified using ECC.

However, those approaches are not effective in that if an altered packet goes undetected by the next authentication, it will never be detected. In addition, large error-code adds the computing burden to authentication on each node, hence degrading the network performance.

Instead of using a large authentication code or tag and attaching it directly to the data, our design distributes the authentication over the network. A packet is progressively authenticated along the transmission path from one node to another. On each node the authentication uses a small but different tag. With our design, there are no bandwidth overhead, and the only a small on-chip storage and performance overhead are incurred.

3 Progressive Packet Authentication Design

Figure 3.1 shows how our authentication design fits into the NoC provided by the third-party vendor. Each node in the NoC can connect to a designer's processing unit (PU). We assume that the hardware and the system software of the PU are trusted and the application software running on the PU is untrusted. The data is transferred over the NoC in packets. Each (alive) packet is identified by its unique ID.

For a new packet, the source node generates a set of tag segments. Each tag segment corresponds to a specific node and the tag value is created by a

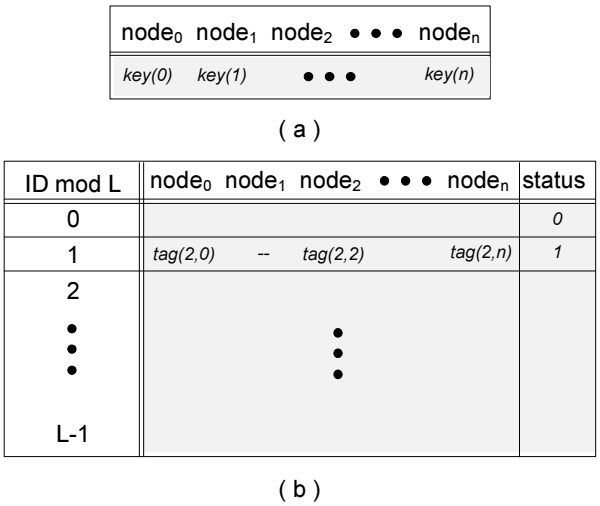


Figure 3.2: Centralized repository (a) key table (b) tag table

cryptographic function controlled by the key that is local to the node. If the packet routing algorithm used in the NoC is deterministic, only tag segments for the nodes on the transmission path are calculated and saved in the related position in the tag table. If the packet routing can be dynamically changed during the transmission, the table will hold tag segments for all nodes.

To detect whether a packet has been tampered, we attach an authentication unit (AU) to each node for an incoming packet. The authentication unit calculates the tag segment based on the received packet and compares it with the original one, if they are the same, the packet is passed to the next node; otherwise, the packet is dropped.

In our design, each node has a dedicated key for the local tag generation. Tags and keys are stored in a centralized repository that holds two tables, as shown in Figure 3.2. Both tables are accessible to PUs and AUs. At the source of a packet, all local tag segments for the nodes on the packet transmission path are generated with the related keys provided in the key table (Figure 3.2 (a)). Those tag segments are then saved in the tag table, as shown in Figure 3.2(b). The AU of a node can have a local copy of its key to avoid repeat access of the key table.

The tag table only needs to hold tags for packets that are alive in the NoC. For a NoC with n nodes, assume there are N_A alive packets, the tag table contains N_A entries and each entry holds n tag segments plus a state bit. We allow for all packets of the same mod ID (mod to N_A) to share one entry. For a packet with ID i, its related tag segments are saved in row (i (mod N_A)). Packet i and packet $i + N_A$ should be temporally far away from each other. When packet $i + N_A$ is generated, packet i would be off the NoC and the table entry is free to packet $i + N_A$.

We use the status bit (located in the last column of the tag table) to control the use of a tag entry in the table. The bit serves two functions: one, indicating whether the tag entry is available for a new packet and two showing whether authentications on the path nodes are required for the related packet. If the

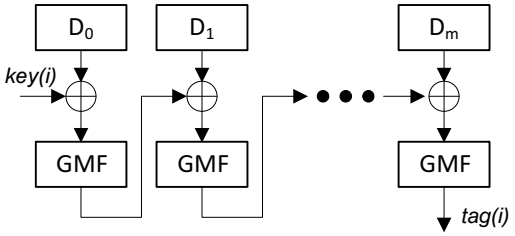


Figure 3.3: Tag segment generation

bit is 1, the tag segments in the entry are valid and the packet should be authenticated during the packet transmission. If the status bit is 0, then to a packet source this entry is free to use for a new packet that is mapped to the entry and to the network nodes there is no need to authenticate the related packet.

When authenticating packet i at node j, the AU on the node first calculates the tag for the packet, and then compares the tag value with $tag(i \pmod{N_A}, j)$ in the tag table. If they are the same, the node passes the packet to the next node; otherwise, the packet is deemed tampered and should be discarded.

Along the transmission path, a packet, if required by the source, will go through authentications on each path node. If the packet passes all the authentications and arrives to the destination, the related entry in the tag table will be reset and free to the next packet with the same mod ID value.

As can be seen, unlike the traditional error-code based design, where the authentication is one-off (namely, if a modified packet cannot be immediately identified based on the error code, the subsequent detections are just in vain but only to waste energy), our approach is progressive. With our design, a packet undergoes a series of authentications. If a modified packet escaped the authentication on one node, it may be caught in the subsequent authentications on the downstream nodes.

For the authentication on the local node, we adopt a block-based CBC approach similar to [24] for tag generation, as shown in Figure 3.3.

We divide the packet into blocks with the block size equal to the tag segment size. Each block is randomized with a GFM (Galois-field Multiplication) function. The operations are chained into all blocks and the tag segment value is produced from the last block. For each node, the tag segment generation is controlled by a key dedicated to the node, as mentioned earlier.

Table 3.1: GMF costs for 128-bit packet

Block Size (bit)	**Area** (um^2)	**Rel. Sav**	**Delay** (ns)	**Rel. OH**	**OH/Sav**
4	3616.0	21.3	20.8	5.5	0.26
8	6096.0	12.6	12.2	3.2	0.26
16	11102.4	6.9	9.0	2.4	0.35
32	21067.2	3.7	6.8	1.8	0.50
64	40446.0	1.9	4.9	1.3	0.68
128	76987.1	1.0	3.8	1.0	1.0

Algorithm 1: Packet tagging at the source

```
/* ------ For any new packet D, generated at PU ------    */
j = getPacketID();
foreach node, i, on transmission path do
    key(i) = getKey(i);
    tag(i, j) = getTag(key(i), D);
    saveTag(tag(i,j));
end foreach
```

Algorithm 2: Packet authentication on a node in NoC

```
/* - On a node, i, over the NoC, for a coming packet, D -  */
j = getPacketID();
tag_ori = readTag(i, j);
tag_cal = calcuateTag(key(i), D);
if tag_cal = tag_ori then
    passPacket(D);
else
    dropPacket(D);
end if
```

Since the authentication should be implemented on each node, we aim to reduce the cost of AU, which is mainly related to GFM. To see the hardware cost of GFM, we modelled the GFM of varied sizes, ranging from 4 to 128 bits. Based on the estimations from Synopsis Design Compiler [1], we found that the cost of GFM is exponentially increased with its size. Table 3.1 shows the related GFM costs in the tag generation for 128-bit packet under different block sizes. From the table, we can see that the small block-based design has low area cost but long delay. The last column in the table shows the relative performance overhead per area saving for different block sizes, which shows the design is most cost effective when block size is 4 or 8 bits and among two, the design with 8-bit block (8-bit tag segment) offers a better performance.

We would emphasize that though the tag generated based on the small blocks is small, a sequence of such tags used along the transmission path essentially form a large tag, hence making the authentication effective.

The packet tagging at the source and authentication on a path node are summarized in Algorithm 1 and Algorithm 2, respectively.

For a packet to be sent to the NoC, Algorithm 1 generates an ID and a tag segment for each network node[1]. For authentication, Algorithm 2 first reads the packet ID from the packet and the related tag segment from the tag table, then calculates the tag segment based on the received data and compares the two tag values. If they are the same, the packet transmission continues; otherwise, the transmission is stopped.

[1]This is for the general case with a dynamic routing scheme. If a deterministic routing is used, only the tag segments for the related transmission path are needed.

4 Case Study: 2D-Mesh NoC

To obtain an insight of the effectiveness of the progressive packet authentication, we investigate the NoC with a 2D-mesh topology, popular in MPSoCs. We assume that the XY routing is used in the packet transfer. For a packet transferred from source (x_s, y_s) to destination (x_d, y_d), it first flows horizontally to (x_s, y_d) then vertically to (x_d, y_d).

The location of the Trojan may affect its ability to capture a target packet for attack. For the 2D-mesh with the fixed routing scheme, some packets may never pass through the Trojan node. Those packets are immune from the Trojan attack. Furthermore, the location of the Trojan also affects the effective tag size used for a packet authentication.

To make our evaluation well-founded and effective, we target the Trojan that holds the maximal advantages in terms of attack possibility and attack success probability, which is discussed below.

4.1 Advantageous Trojan Locations

Without targeting specific-applications, we assume sources and destinations of packets are randomly distributed over the whole network. We also assume that the Trojan is located at (x_t, y_t) in the mesh that has m rows and n columns, and we call the row and column that the Trojan is located, the **Trojan row** and **Trojan column**.

1) Trojan Locations for High Attack Possibility

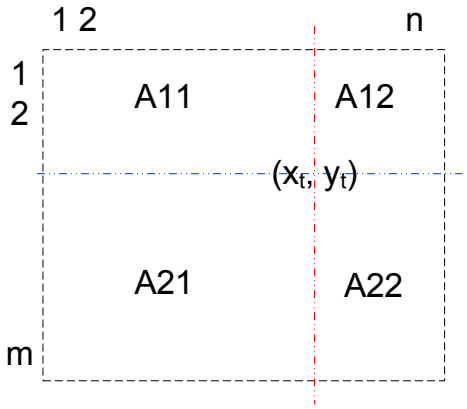


Figure 4.1: Trojan in mesh: for high attack possibility

Intuitively, a Trojan can gain maximal attack advantage when it is located in the center of the network, which can be verified below.

We divide the mesh into four sections A11, A12, A21, A22, as shown in Figure 4.1. Given the XY-routing, a packet from the source (x_s, y_s) to the destination (x_d, y_d) that can be attacked should pass the Trojan. Therefore, the following two types of packets are vulnerable:

- the packet source on the Trojan row (i.e. $x_s = x_t$) and the source and the destination are separated by the Trojan column; namely, if the source

on the left section, the destination should on the right section, and vice versa.

- the packet destination on the Trojan column (i.e. $y_d = y_t$) and the source and the destination are separated by the Trojan row; namely, if the source on the top section, the destination should be on the bottom section, and vice versa.

For a Trojan at the location (x_t, y_t), the number of the packets from the left Trojan row to the right section is

$$(y_t - 1) * (n - y_t + 1) * m,$$

and from the right Trojan row to the left section is

$$(n - y_t) * y_t * m.$$

Similarly, the number of the packets from the bottom section to the top Trojan column is

$$(m - x_t) * n * x_t,$$

and from the top section to the bottom Trojan column is

$$(x_t - 1) * n * (m - x_t).$$

The total number of possible packets that can be attacked by the Trojan is

$$\begin{aligned} P &= (y_t - 1) * (n - y_t + 1) * m + (n - y_t) * y * m + \\ &\quad (m - x_t) * n * x_t + (x_t - 1) * n * (m - x_t) \end{aligned} \tag{4.1}$$

From[1]

$$\frac{dP}{dy_t} = 2n - 4y_t + 1 = 0$$

and

$$\frac{dP}{dx_t} = 2m - 4x_t + 1 = 0,$$

we can obtain that when $x_t = (m/2 + 1/4)$ and $y_t = (n/2 + 1/4)$, the value of P reaches to the maximum. Namely, when the Trojan is positioned at the center of the mesh network, it has the highest probability to intercept the packets for the attack.

2) Trojan Locations for High Attack Success Probability

The success probability is closely related to the effective tag size, which is determined by the Trojan path length to the packet destination. To calculate such a path length, we divide the mesh vertically into three sections: two sections of the same size, symmetrical to the Trojan column and the rest non-symmetric section, as shown in Figure 4.2. The width of the symmetric section is $S_x = min(x_t - 1, n - x_t)$ or 0 if the Trojan is on the either of two edge columns. Similarly, we can divide the mesh horizontally, where $S_y = min(y_t - 1, m - x_t)$ or 0 if the Trojan is on the edge rows.

[1]To reduce the complexity of the derivation, we treat all discrete functions as continuous.

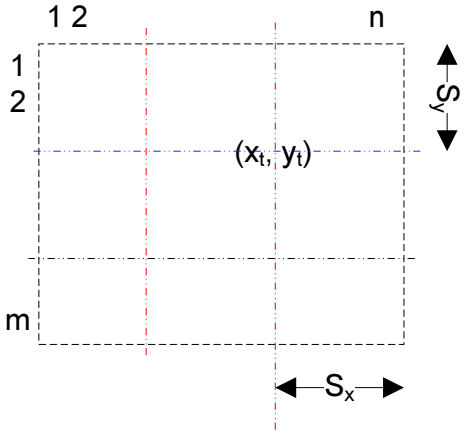


Figure 4.2: Trojan in mesh: high attack success probability

The total path length of the Trojan to all nodes on the Trojan row in a symmetric section is

$$\sum_{i=0}^{S_x} i,$$

and for the non-symmetric section, it is

$$S_x * (m - 2S_x - 1) + \sum_{i=0}^{m-2S_x-1} i.$$

The total path length of the Trojan to all nodes on the same row is

$$\begin{aligned} Lx &= 2 * \sum_{i=0}^{Sx} i + Sx(m - 2Sx - 1) + \sum_{i=0}^{m-2Sx-1} i \\ &= 2 * \frac{1 + Sx}{2} * Sx + Sx(m - 2Sx - 1) + \\ &\quad \frac{1 + (m - 2Sx - 1)}{2} * (m - 2Sx - 1) \\ &= Sx^2 - (m - 1)Sx + \frac{m(m - 1)}{2} \end{aligned} \tag{4.2}$$

Given the path length for the Trojan to all nodes on the same row, we can easily find the total length to other rows. For the next neighbour row, the length is $m + Lx$, and for a row that is k row away, the length is $m * k + Lx$. We can use similar way to calculate the distance of all rows for the symmetrical sections, and the third non-symmetrical section. And the total path length of the Trojan

to all other nodes is

$$\begin{aligned} L &= Lx * n + m * (2 * \sum_{i=0}^{Sy} i + Sy * (n - 2Sy - 1) + \sum_{i=0}^{n-2Sy-1} i) \\ &= Lx * n + m * (Sy^2 - (n-1)Sy + \frac{n(n-1)}{2}) \\ &= n * (Sx^2 - (m-1)Sx + \frac{m(m-1)}{2}) + \\ &\quad m * (Sy^2 - (n-1)Sy + \frac{n(n-1)}{2}) \end{aligned} \tag{4.3}$$

With $dL/dS_x = 0$ and $dL/dS_y = 0$, we can obtain that when $S_x = (m-1)/2$ and $Sy = (n-1)/2$ (namely, the mesh center), L is minimal.

As can be seen from above derivations, when Trojan is placed at the center of the mesh, it gains the highest chance and success probability to attack a packet.

5 Experiment Evaluation and Results

To verify the analytical findings in Section 4, we built a simulation platform written in C to simulate the packet flows over the 2D-mesh NoCs. The packets' sources and destinations are uniformly distributed over the whole NoCs. For the Trojan sitting in a different location, the effective authentication path length of all packets are obtained.

We further evaluate the overheads of our detection design with the Synopsys Design Compiler [1]. To see the impact of traditional authentication on performance of the NoC, we run application traces collected by Netrace 1.0 [11] on a cycle-accurate NoC simulator Booksim 2.0 [16]. Our target NoC has 64 nodes connected together using 8x8 2D-mesh topology, a 3-stage pipelined routers with matrix arbitration, wormhole switching and dimension order routing (XY) algorithm, which are commonly used in NoC designs. The experiment results are discussed below.

5.1 Effectiveness of Trojan Location

To demonstrate that the center location offers the advantage to the Trojan for the highest possible attack, Table 5.1 shows a small set of experiment results on NoCs with 3x3 mesh. Each row lists the Number of Attacked packets (NoA) among all packets (Row 1) transferred over the network and the average Authentication Path Length (APL) per attacked packet for a given Trojan location (Column 1).

A Trojan can only attack a packet if the Trojan is on the packet's transfer route. For example, if a Trojan sits at the corner node (1,1), a packet sent from node (2,2) to (3, 4) cannot be intercepted by the Trojan and hence cannot be diverted to a different location. Therefore, among all random packets generated, only a portion of them can be attacked. For each test (namely, for a given Trojan location and a given number of random packets), the table provides two values in each cell.

Table 5.1: Passing Trojan packet number and average attack path length 3x3 mesh under different test sizes

Trojan location	**100**		**1000**		**5000**		**10000**		**50000**		**100000**		**500000**	
	NoA	**APL**	**NoA**	**APL**	**NoA**	**APL**	**NoA**	**APL**	**NoA**	**APL**	**NoA**	**APL**	**NoA**	**APL**
(1,1)	27	2	283	2	1517	2	3033	2	15306	2	30336	2	152186	2
(1,2)	25	1	405	1	1878	1	3902	1	19086	1	38086	1	191275	1
(1,3)	33	2	342	2	1452	2	3000	2	15203	2	30386	2	151462	2
(2,1)	41	2	384	1	1942	1	3854	1	19103	1	38871	1	192158	1
(2,2)	51	1	472	1	2348	1	4609	1	23139	1	46343	1	232339	1
(2,3)	46	2	406	1	1919	1	3812	1	19042	1	38331	1	191304	1
(3,1)	35	2	326	2	1555	2	2989	2	15210	2	30531	2	151774	2
(3,2)	42	2	378	1	1923	1	3793	1	19284	1	38405	1	192484	1
(3,3)	24	2	284	2	1496	2	2963	2	15217	2	30426	2	151945	2

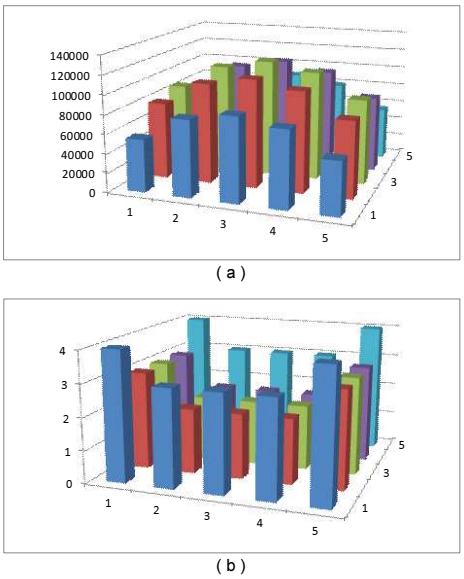


Figure 5.1: Experiment results on 5x5 mesh (a) passing Trojan packet number (b) average attack path length

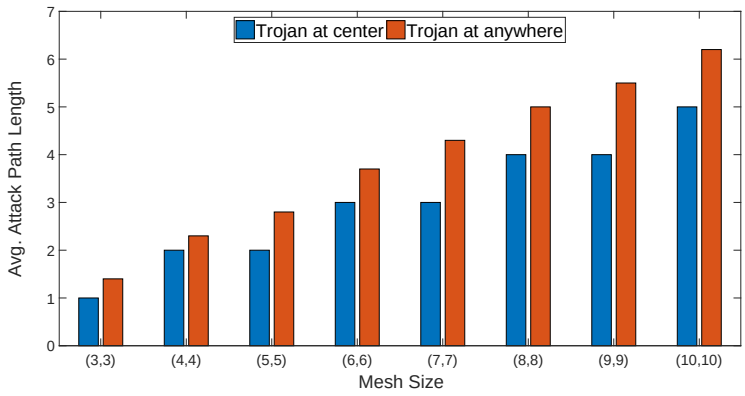


Figure 5.2: Average attack path length for different mesh sizes

The first is the number of packets that can be attacked by the Trojan and the second is the average authentication length in term of number of nodes. As can be seen from the table, at the center location (2,2), the Trojan can intercept a maximal number of packets and the related attacks will undergo authentication with a minimal tag size.

Similar results can be observed for other sizes of meshes, as shown in Figure 5.1, for the 5x5 mesh with 5000 random packet. The attack path length for different mesh sizes are summarized and shown in Figure 5.2. The average effective tag size with the 8-bit tag segment is given in Table 5.2. As can be seen, with the small 8-bit tag segment, on average, an equivalent 32-bit tag is used in the authentication when the Trojan is located in the network center,

and with the randomly located Trojan, tags of average 41 bits are used.

Table 5.2: Average effective tag size

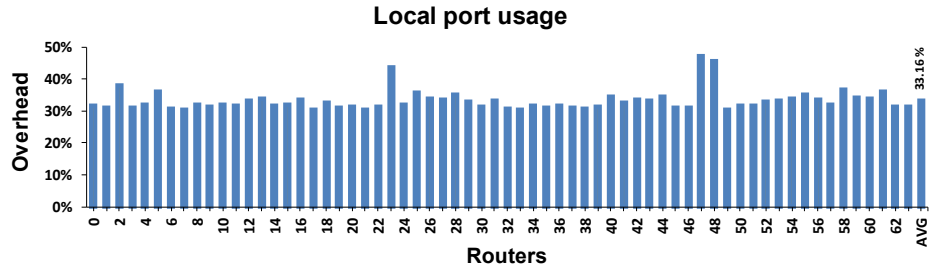
Mesh	Trojan at center		Trojan at anywhere	
	Attack Path Length	**Effective Tag Size**	**Attack Path Length**	**Effective Tag Size**
(3,3)	1	8	1.4	12
(4,4)	2	16	2.3	19
(5,5)	2	16	2.8	23
(6,6)	3	24	3.7	30
(7,7)	3	24	4.3	35
(8,8)	4	32	5.0	41
(9,9)	4	32	5.5	45
(10,10)	5	40	6.2	50

5.2 Performance Evaluation

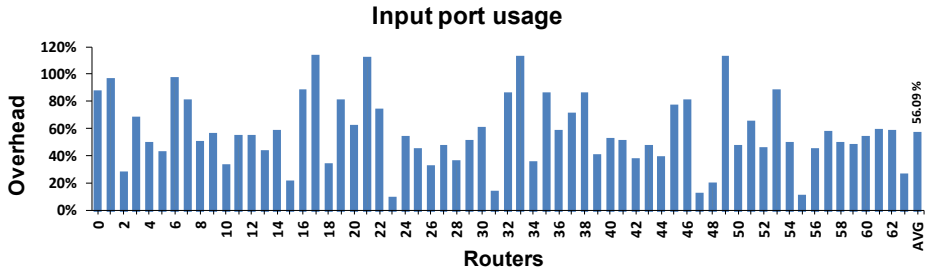
To see the impact of the traditional tag-based authentication on the NoC performance, we first run an application (blackschole) on the NoC simulator, Booksim 2.0, for two designs: without authentication and with the authentication where the tag is transferred with the packet. The application's tasks are scheduled on 64 cores, connected through NoC routers. We investigate the utilization of network resources of the application, in terms of the local port (to the local PU), input ports, and output ports of the router. The usage overhead of each type of resources of all 64 routers are shown in Figure 5.3(a), Figure 5.3(b) and Figure 5.3(c), respectively.

As can been seen from the three figures, resource usages of all routers are increased due to the tag transmission. The average overhead (AVG) on the local ports is about 33% as compared to the baseline design without authentication, and routers $R2$, $R23$, $R47$ and $R48$ are mostly affected (see Figure 5.3(a)), which means the processing units attached to these routers may have to wait for longer time to inject data in NoC. Figure 5.3(b) shows the usage overhead of input ports. From this figure, we can see that the usage is doubled on $R17$, $R21$, $R33$ and $R49$. Therefore, these routers very likely create bottlenecks in the network. Any packet travelling through these routers may experience a high buffer delay and hence overall long packet latency. Furthermore, if the packet belongs to a real time task, the task may miss its deadline due to the unexpected long communication delay. In terms of the output port usage, as shown in Figure 5.3(c), all routers have more than 30% overhead, and this overhead exceeds 45% on routers $R23$, $R47$, $R48$ and $R55$.

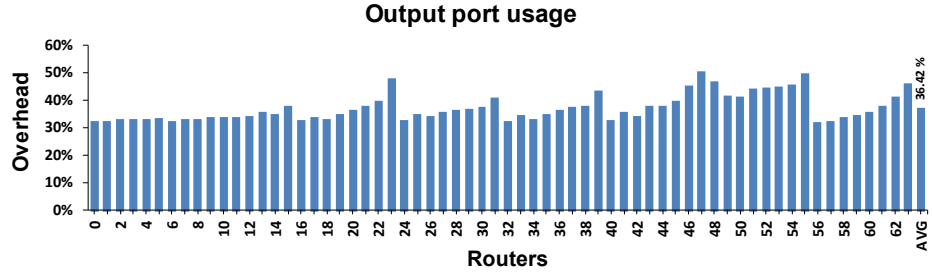
These overheads will increase resource competition and lead to performance degradation. Table 5.3 shows the performance overhead incurred by the traditional design for a set of applications as compared to our design. On average, a 36.06% increase on bandwidth, 11.66% on buffer delay, and 28.16% on packet latency are caused due to the packet space consumption. Our authentication technique, on the other hand, does not send the tag with the packet and only incurs a small overhead on packet latency (due to packet authentication).



(a)



(b)

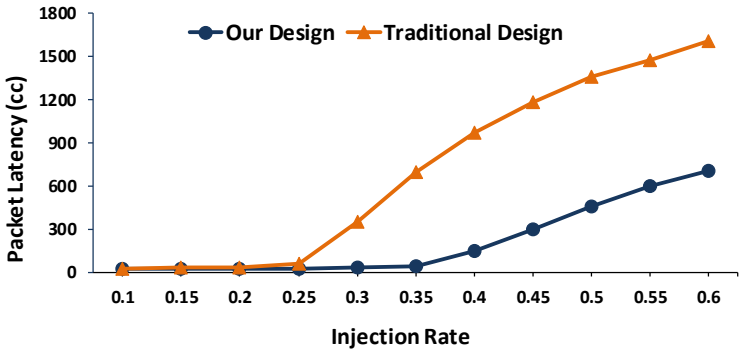


(c)

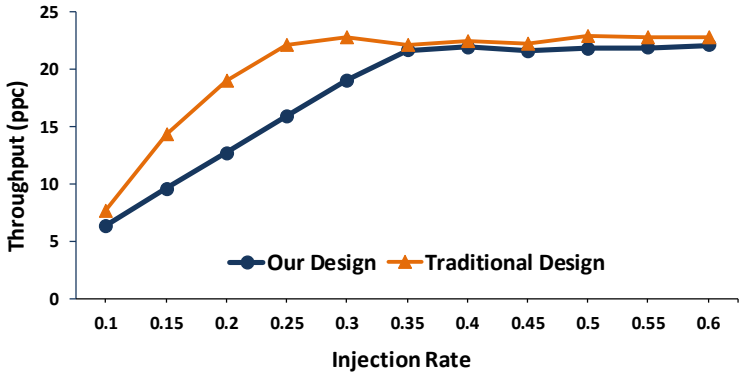
Figure 5.3: Overhead by traditional tag based authentication design by blackschole on (a) local port usage (b) input port usage (c) output port usage

Table 5.3: Overhead by the traditional design scheme

Applications	Bandwidth (%)	Buffer delay (%)	Packet Latency (%)
blackscholes	36.02	27.62	11.59
bodytrack	36.05	28.71	11.67
canneal	36.05	28.8	11.68
dedup	36.08	27.96	11.63
ferret	36.05	27.67	11.66
fluidanimate	36.05	28.8	11.68
swaptions	36.05	27.77	11.62
vips	36.12	28.25	11.72
x264	36.05	27.83	11.67
AVG	**36.06**	**28.16**	**11.66**



(a)



(b)

Figure 5.4: Comparison of our design vs traditional design when injection rates are increased (a) packet latency in clock cycles (cc) and (b) throughput in packets per clock cycle (ppc)

We also study the impact of packet injection rate on the network performance. Figure 5.4 (a) shows the average packet latency of two designs: with the traditional authentication scheme and with our authentication scheme. It can be seen that at low injection rates (hence no resource competition), both

designs have the same stable packet latency. The packet latency rises if the injection rate continually increases and eventually causes network traffic jam. But this rise starts earlier and much faster in the traditional design than in our design. This is due to the exacerbated network congestion imposed by the packet space consumption of the authentication tags in the traditional design. When the utilization of the network resources reaches to 100%, it starts to build a back pressure in NoC, causing link blockage, and finally the network saturation. For the same reason, the network throughput with the traditional design also reaches to saturation fast and early, as shown in Figure 5.4 (b). With our design, on the other hand, no extra traffic is generated by authentication; the network traffic linearly increases with the injection rate and gracefully becomes saturated.

5.3 Security Evaluation

To evaluate the security effectiveness of our aprpoach, we implemented three types of Trojans [10, 26, 8, 7, 13, 23, 14] that focus on changing packet data for different attack purposes. The first type of Trojans ($Tro1$) modify the packet addresses to leak information or to degrade the system performance. The second type of Trojans ($Tro2$) alter the packet ID, type and control bits to create denial-of-services. The third type of Trojans ($Tro3$) only modify the data bits to perform data integrity and possibly performance degradation attacks.

For each Trojan type, we run the simulation for 10 million attack packets. The detection rates of both the traditional design (of different tag sizes, 8-32 bits) and our design (8 bits for tag segment) on three types of Trojan are given in Table 5.4. As can be seen from Table 5.4, for $Tro1$, when 8-bit tag is used, the detection rate is 57.35%. The detection rate increases with the tag size. When the tag size is 32 bits, the detection rate reaches to 98.54%. Similarly, for $Tro2$ and $Tro3$, the detection rates are related to the tag size and reach to 99.89% and 99.9%, respectively, when 32-bits tags are used. However, with our authentication approach, even if the 8-bit tag is used, the overall tag size is effectively large and the detection rates for the three Trojans are above 99.99%, as shown in the last column of the table.

Table 5.4: Detection rates (%) of traditional authentication vs progressive authentication

Application	**Traditional Authentication**				**Progressive Authentication**
	8	**16**	**24**	**32**	
Tro1	57.35	68.46	92.33	98.54	99.9989
Tro2	57.67	68.88	93.02	99.89	99.9999
Tro3	57.55	68.91	95.81	99.99	99.9899
AVG	**57.52**	**68.75**	**93.72**	**99.47**	**99.9962**

5.4 Implementation Overheads

For the case of Trojan located in the center on the 8x8-mesh network, we have investigated the hardware implementation costs of the authentication design. The overheads are summarized in Table 5.5. Given the NoC size (64 nodes) and the tag segment size (8 bits), the area for the key table size is fixed at 6412 μm^2 (see Column 4), but for each application its tag table size varies with the number of alive packets (N_A), as shown in Column 3. Column 5 lists the total area cost that is the sum of the tag table, key table and the authentication logic on each network node. For comparison, we also estimated the cost of the traditional design. With the traditional design, a fixed tag size is used and the tag is attached and transferred with the packet; therefore, its area cost and packet space consumption (PSC) of the authentication are fixed, as shown in Columns 6 & 7. The related savings of our design as compared to the traditional design are given in the last two columns. As can be seen, the progressive authentication design not only incurs zero packet space consumption, but also reduces the area overhead as compared to the traditional authentication design. On average, an overhead of 56% area can be saved.

Table 5.5: Cost overhead comparison of traditional authentication vs progressive authentication

Application	N_A	**Area of Progressive Authentication** (μm^2)			**Traditional Authentication**		**Relative Saving**	
		Tag table	**Key table**	**Total Area**	**Area** (μm^2)	**PS** (bits)	**Area** (μm^2)	**PS** (bits)
blackscholes	82	525803	6412	2677230.841	5971230.72	32	0.55	1.00
bodytrack	66	423207	6412	2593871.851	5971230.72	32	0.57	1.00
canneal	69	442444	6412	2587459.621	5971230.72	32	0.57	1.00
dedup	72	461681	6412	2606696.311	5971230.72	32	0.56	1.00
fluidanimate	89	570688	6412	2715704.221	5971230.72	32	0.55	1.00
swaptions	58	371909	6412	2516925.091	5971230.72	32	0.58	1.00
vips	94	602750	6412	2747765.372	5971230.72	32	0.54	1.00
x264	91	583513	6412	2728528.681	5971230.72		0.54	1.00
AVG							**0.56**	**1.00**

6 Conclusion

In this report, we presented a progressive authentication approach for the packet data authentication on the third-party NoC, where a packet is tagged at the source for all path nodes it may pass. When the packet travels to a node, it is authenticated against the tag for this node. Different from other packet error detection designs, our design has two special features: one, the tag for a packet is not carried by the packet, hence avoiding the bandwidth overhead, and two, for a given packet, the authentication tag is different from node to node. If an altered packet went undetected on one node, it may be caught at the next node. Therefore, the effectiveness of the detection is greatly improved.

Our experiment on an 8x8 mesh NoC shows that, given a small 8-bit tag used in the authentication unit, an average of 32-bit tag is effectively used for the highly advantageous Trojan and the effective tag size can grow when the Trojan is randomly located and the network size is increased. It must be pointed out that the cost of the tag table and key table used in the progressive authentication can be moderated by the savings from the small tag-based authentication on the network nodes. Our result even shows 56% of area saving when we use an 8-bit tag for authentication as compared to the traditional authentication design with a similar attack detection rate (over 99%).

Our design incurs little performance overhead, which would, otherwise, be significant if the traditional authentication design approach is used. According to our experiments on a set of applications, the overheads on the bandwidth, buffer delay, and packet latency from the traditional design as compared to our design are 36%, 28%, and 12%, respectively.

Bibliography

[1] Synopsys Design Compiler. http://www.synopsys.com.

[2] M. Bellare, R. Guérin, and P. Rogaway. Xor macs: New methods for message authentication using finite pseudorandom functions. In *Annual International Cryptology Conference*, pages 15–28. Springer, 1995.

[3] M. Bellare and C. Namprempre. Authenticated encryption: relations among notions and analysis of the generic composition paradigm. In *Advances in Cryptology - ASIACRYPT 2000. 6th International Conference on the Theory and Application of Cryptology and Information Security. Proceedings (Lecture Notes in Computer Science Vol.1976)*, pages 531 – 45, Berlin, Germany, 2000.

[4] A. Berendschot, J. P. Boly, A. Bosselaers, J. Brandt, D. Chaum, I. Damgård, P. de Rooij, M. Dichtl, W. Fumy, C. Jansen, et al. Integrity primitives for secure information systems. final report of race integrity primitives evaluation (ripe-race 1040). *Lecture Notes in Computer Science*, 1007, 1995.

[5] J. Black and P. Rogaway. CBC MACs for arbitrary-length messages: the three-key constructions. In *Advances in Cryptology - CRYPTO 2000. 20th Annual International Cryptology Conference. Proceedings (Lecture Notes in Computer Science Vol.1880)*, pages 197–215, Berlin, Germany, 2000.

[6] J. Black and P. Rogaway. A block-cipher mode of operation for parallelizable message authentication. In *Advances in Cryptology - EUROCRYPT 2002. International Conference on the Theory and Applications of Cryptographic Techniques. Proceedings (Lecture Notes in Computer Science Vol.2332)*, pages 384 —- 97, 2002.

[7] T. Boraten and A. Kodi. Mitigation of hardware trojan based denial-of-service attack for secure nocs. *Journal of Parallel and Distributed Computing*, 111:24–38, 2018.

[8] T. Boraten and A. K. Kodi. Packet security with path sensitization for nocs. In *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1136–1139. IEEE, 2016.

[9] R. Elbaz, L. Torres, G. Sassatelli, P. Guillemin, M. Bardouillet, and A. Martinez. Block-Level Added Redundancy Explicit Authentication for Parallelized Encryption and Integrity Checking of processor-memory transactions. *Transactions on Computational Science X. Special Issue on Security in Computing, Part I*, 6340(PART 1):231–260, 2010.

[10] J. Frey and Q. Yu. A hardened network-on-chip design using runtime hardware trojan mitigation methods. *Integration, the VLSI Journal*, 56:15–31, 2017.

[11] J. Hestness, B. Grot, and S. W. Keckler. Netrace: dependency-driven trace-based network-on-chip simulation. In *Proceedings of the Third International Workshop on Network on Chip Architectures*, pages 31–36. ACM, 2010.

[12] M. Hong, H. Guo, and S. X. Hu. A cost-effective tag design for memory data authentication in embedded systems. In *Proceedings of the 2012 International Conference on Compilers, Architectures and Synthesis for Embedded Systems (CASES 2012)*, pages 17–26, 2012.

[13] M. Hussain and H. Guo. Packet leak detection on hardware-trojan infected nocs for mpsoc systems. In *Proceedings of the 2017 International Conference on Cryptography, Security and Privacy*, pages 85–90. ACM, 2017.

[14] M. Hussain, H. Guo, and S. Parameswaran. A customized authentication design for traffic hijacking detection on hardware-trojan infected nocs. *Journal of Computer and Communications*, 6(01):135, 2017.

[15] T. Iwata and K. Kurosawa. OMAC: one-key CBC MAC. In *Fast Software Encryption. 10th International Workshop, FSE 2003. Revised Papers (Lecture Notes in Comput. Sci. Vol.2887)*, pages 129 —- 53, 2003.

[16] N. Jiang, J. Balfour, D. U. Becker, B. Towles, W. J. Dally, G. Michelogiannakis, and J. Kim. A detailed and flexible cycle-accurate network-on-chip simulator. In *Performance Analysis of Systems and Software (ISPASS), 2013 IEEE International Symposium on*, pages 86–96. IEEE, 2013.

[17] D. A. McGrew and J. Viega. The security and performance of the Galois/counter mode (GCM) of operation. In *Progress in Cryptology - INDOCRYPT 2004. 5th International Conference on Cryptology in India. Proceedings (Lecture Notes in Computer Science Vol.3348)*, pages 343 – 55, Berlin, Germany, 2004.

[18] NIST. Recommendation for block cipher modes of operation: The cmac mode for authentication. http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf.

[19] W. W. Peterson and D. T. Brown. Cyclic codes for error detection. *Proceedings of the IRE*, 49(1):228–235, 1961.

[20] F. Petrank and C. Rackoff. CBC MAC for real-time data sources. *J. Cryptol. (USA)*, 13(3):315 – 38, 2000.

[21] P. Rogaway. Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC. In *Advances in Cryptology-ASIACRYPT 2004. 10th International Conference on the Theory and Application of Cryptology and Information Security. Proceedings (Lecture Notes in Computer Science Vol.3329)*, pages 16–31, Berlin, Germany, 2004.

[22] A. Rogers and A. Milenkovic. Security extensions for integrity and confidentiality in embedded processors. *Microprocess. Microsyst. (Netherlands)*, 33(5-6):398–414, 2009.

[23] H. Salmani, M. Tehranipoor, and R. Karri. On design vulnerability analysis and trust benchmarks development. In *Computer Design (ICCD), 2013 IEEE 31st International Conference on*, pages 471–474. IEEE, 2013.

[24] A. X9.9. American national standard for financial institution message authentication. 1981.

[25] C. Yan, B. Rogers, D. Englender, D. Solihin, and M. Prvulovic. Improving cost, performance, and security of memory encryption and authentication. In *Proceedings. 33rd International Symposium on Computer Architecture*, page 12 pp., 2006.

[26] Q. Yu, J. Dofe, Y. Zhang, and J. Frey. Hardware hardening approaches using camouflaging, encryption, and obfuscation. In *Hardware IP Security and Trust*, pages 135–163. Springer, 2017.

[27] Q. Yu and J. Frey. Exploiting error control approaches for hardware trojans on network-on-chip links. In *2013 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFTS)*, pages 266–271. IEEE, 2013.