Lightweight Multicast Authentication in NoC-based SoCs

Hansika Weerasena and Prabhat Mishra
Department of Computer & Information Science & Engineering
University of Florida, Gainesville, Florida, USA

***Abstract*—Network-on-Chip (NoC) is responsible for managing communication in modern SoCs. The ubiquity of NoC and its distributed nature across the chip has made it a focal point of attacks. Spoofing attacks by impersonating the nodes in SoC can lead to unauthorized information access and can also be employed to launch denial of service attacks. Modern software tends to use more parallelism among multiple cores, increasing multicast communication among cores to exchange cache coherence messages. Traditional multicast authentication solutions are not effective due to the resource-constrained nature of NoC-based SoCs. In this paper, we propose a lightweight multicast authentication mechanism that utilizes existing unicast authentication infrastructures in NoC using one-way accumulators. Experimental results demonstrate the effectiveness of our approach with required security while incurring minor area and performance overhead.**

***Index Terms*—Multicast Communication, Network-on-Chip, Security, Authentication**

I. Introduction

The advancement of manufacturing technologies has enabled the integration of more and more diverse intellectual property (IP) cores on the same System-on-Chip (SoC). Multiprocessor SoC (MPSoC) is dominated by a large number of computing cores that support parallel computation and muti-programming workloads. Commercial MPSoC such as Altra® multicore server processor has 128 cores [1]. Network-on-chip (NoC) has become the de facto standard in providing communication infrastructure among these core MPSoCs to eliminate communication bottlenecks. For example, leading MPSoC manufacturing companies such as Intel use Skylake Mesh [2] NoC in server-grade processors. In a typical MPSoC, NoC is mainly used for communicating cache coherence and other control messages between processors and memory subsystems.

Due to the steady increase in the number of cores in MPSoC, parallel programming has become a viable option for performance improvement in applications that can exploit parallelism. Therefore, modern programs and compiler optimizations are designed to exploit the parallelism provided by multiple cores. This trend is expected to improve over time [3]. Parallel workloads lead to increased one-to-many (multicast) communication inside MPSoCs. Multicast communication in NoC can be used in replication, barrier synchronization, cache coherency in distributed shared caches, and clock synchronization [4].

When considering cache coherence, the intensity of one-to-many communication in NoC depends on the specific cache coherence protocol. For example, the MESI directory-based cache coherence protocol has 5 - 13% multicast traffic when running SPLASH2 and PARSEC benchmarks [5]. Similarly, the multicast ratio can increase over 50% for broadcast-based protocols [6]. It is important to note that multicast intensity is expected to increase further with the increase in parallelisms in programs. Traditionally, NoC treats multicast traffic patterns as repeated unicast traffic, which is known as software multicast.

Software multicast can lead to hostspots and performance bottleneck in NoCs. Consider a simple example to understand the implications. Assume that there are only 5% multicast messages with 10 destinations of 100 total messages. If software-multicast is used there will be 95 unicast packets and 50 software-unicast packets. Therefore, software-unicast is not a scalable solution for multicast communication in NoC. A multitude of path-based and tree-based multicast routing schemes [7], [8], [9] are proposed in the literature for efficient multicast communication in NoC. Figure 1 illustrates XY tree-based multicast routing that is used in this paper.

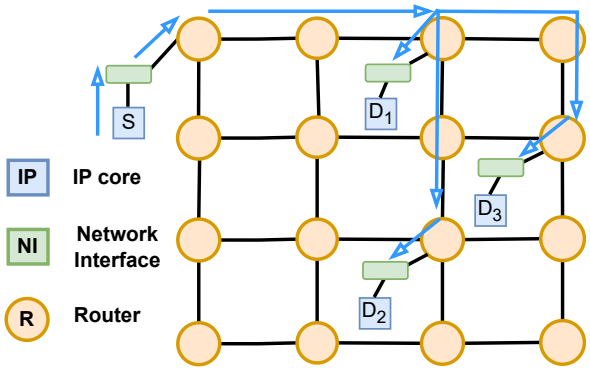

Fig. 1: Multicast communication from source (S) to three destinations (D_1, D_2, D_3). Multicast packets follow the same path according to XY protocol and branch out when necessary.

Due to cost and time-to-market constraints, it has become an industry norm to use third-party Intellectual Property (IP) blocks in designing SoCs. These third-party IPs pose security concerns as they can come with malicious implants, hidden backdoors, and undocumented bugs. Additionally, long supply chains and potentially untrusted vendors can increase security concerns in SoC. Since NoC has access to all the components in SoC, it has become the focal point of the attackers. Ensuring authenticity against spoofing attacks is recognized as one of the critical security concerns in NoC [10]. In a spoofing attack, a malicious node can impersonate another node to violate the security of SoC. Though there are existing lightweight unicast

security solutions [11], [12], [13] for authentication, they cannot be applied for multicast communication since they use shared keys or secret between source and destination. On the other hand, the existing multicast authentication solutions from traditional computer networks are not suitable for resource-constrained NoCs.

This paper tries to answer the following question: *Is there a way to provide lightweight multicast packet authentication in resource-constrained NoCs?* Specifically, this paper makes the following research contributions.

- We propose a lightweight multicast authentication scheme that utilizes the existing unicast authentication infrastructure in NoC.
- We show that our multicast authentication scheme can provide the desired level of security.
- Our multicast authentication can also provide reconfigurable security.
- Experimental results demonstrate that our proposed multicast authentication scheme has minimal performance and area overhead.

The rest of the paper is organized as follows. Section II presents background on hashing-based authentication and surveys the related efforts. Section III outlines the threat model and provides the problem formulation. Section IV describes our proposed lightweight multicast authentication scheme. Section V presents the experimental results. Finally, Section VI concludes the paper.

II. Background and Related Work

In this section, we first introduce one-way accumulator as a cryptographic primitive and outline a scheme for fast one-way accumulation. Next, we survey prior efforts related to NoC security.

A. Fast One-way Accumulator

One-way accumulator was proposed as an alternative to digital signature by [14]. In accumulated hashing, items are combined and hashed to generate a unique hash code, where individual items can prove their membership in the accumulated hash code. A one-way accumulator is a one-way hash function with the quasi-commutative property. A function $f : A \times B \rightarrow A$ is said to be quasi-commutative if for all $a \in A$, and for all $b, c \in B$:

$$f(f(a,b),c) = f(f(a,c),b) \tag{1}$$

Simply, the order of accumulation of the items does not affect the final outcome. The value a is considered as the seed. Nyberg [15] defined a fast way of doing accumulated hashing which is elaborated in the rest of the section.

Let $N = 2^d$ be the upper bound of the number of possible accumulated items. Then $h : \{0,1\}^* \rightarrow \{0,1\}^l$ be a cryptographically secure hash function that has the tag length of $l = rd$, where r is an integer representing the length of the accumulated hash value. So if $X = x_1, \ldots, x_m$ be the items to be accumulated ($m \leq N$).

$$y_i = h(x_i), i = 1 \ldots m$$

Define $\alpha_r : \{0,1\}^l \rightarrow \{0,1\}^r$. α_r takes a input of length l and interpret as r of sub-strings of d. Then it will replace every all zero bit-strings with 0 and others with 1. So if we take y_i from $y_1, \ldots, y_m$, y_i is represented as $y_i = (y_{i,1}, \ldots, y_{i,r})$. Then for each y_{ij}, if $y_{ij} = \{0\}^d$ it is replaced by 0, or 1 otherwise. The resultant r length bit-string is the output of α_r which can be expressed as $a_i = (a_{i,1}, \ldots, a_{i,r})$.

The fast accumulation [15] can be mathematically represented as follows, where s is the seed, $\odot$ is the bitwise and operator and $\prod$ represent bitwise prod operator:

$$Tg = H(s, X) = s \odot \prod_{i=1}^{m} \alpha_r(h(x_i)) \tag{2}$$

Since the bitwise operator is commutative and the hash function has the property of one-wayness, H is quasi-commutative. Verification of membership of item x_i using the accumulated hash is straightforward. We need to calculate a_i of an item i as $a_i = \alpha_r(h(x_i))$. Then, check whether $a_i \odot H(s, X) = H(s, X)$ to see if the item x_i is in the accumulated hash.

The security of the accumulator depends on the ability of an adversary to forge a single accumulated value (a_i) of an item i. The security is compromised when forged a_i passes an item from membership validation. For a security level t, the length of the accumulated hash $r = N \times e \times t$, where e is the Napier's number. Our proposed modification of the fast accumulator for multicast authentication needs a different security guarantee discussed in Section IV-A.

B. Related Work

Security of NoC has been extensively studied across different security goals (confidentiality, authenticity, integrity, anonymity and freshness) due to the evolving threat landscape in SoCs [10]. Malicious NoC is a common threat model seen across many types of attacks [13], [16], [17], [18], [19], [20] since most SoC tend to use third-party NoC IPs. For example, the threat model in [13] assumes that NoC IP is malicious. Specifically, there are malicious routers that can eavesdrop, tamper packets and impersonate other nodes in SoC. Malicious routers have been used as a threat model in [16], [17] to launch denial of service (DoS) attacks using packet tampering and flooding. In [16], the authors introduced four types of hardware Trojans (HT) in the router, which can change header bits including address to launch spoofing attacks.

Several efforts have been made to secure NoC traffic against spoofing or node impersonating attacks. The solution for confidentiality with authenticated encryption provides protection against spoofing attacks through authentication [11], [12], [13], [21]. For example, [13] uses Siphash [22] for authentication which is a lightweight and fast hash function well suited for short inputs and is an ideal candidate for NoC-based SoCs. SipHash iteratively performs a series of add, rotation, and XOR operations to achieve fast MAC computation for short messages. It can introduce hardware overhead of 2% when compared to the entire baseline MPSoC. We assume that Siphash authentication is implemented in each network interface of the node. All of these solutions on authenticity

depend on a pre-shared secret (key) shared between two communicating parties, which is suitable for unicast traffic with one sender and receiver. Any of these solutions cannot be directly applied on multicast communication because there are multiple receivers.

Several efforts can be found in traditional computer networks to authenticate multicast traffic. Public key cryptography [23] has the natural asymmetric property that can be utilized for multicast authentication. Here, the sender can send an authentication tag by signing with the private key and receivers can use the public key to verify the authenticity. Although this method provides adequate security, it is not suitable for resource-constrained NoC. Several group key-based protocols [24] are proposed in traditional networks where a key is shared between the multicast group to generate a message authentication code. The main issue with this approach is it uses a weaker threat model where the members of the group are trusted. An adversary inside the group can easily impersonate another member in the group using the shared key. A MAC for each recipient is another approach used for trivial multicast authentication which has high communication overhead due to long packet sizes. μ-TESLA [25] is a lightweight multicast authentication scheme that uses hash-chains, but the delayed verification of this scheme makes it vulnerable to multitudes of attacks [26], [27], including DoS. An HMAC-based multicast authentication scheme is proposed in [28], which accumulates HMAC tags of all receivers to create the final tag. However, usage of HMAC makes it computationally intensive for NoC. To the best of our knowledge, our proposed approach is the first attempt in securing multicast communication to ensure the complete authenticity of NoC traffic.

III. Problem Formulation and Threat Model

In directory-based cache coherence protocols, multicast messages are responsible for communicating cache invalidation messages. These messages are short control messages that are sent from the owner of the cache data to all the cores that are sharing that cache data. Our threat model assumes the possibility of one of the multiple malicious routers with HT that can impersonate nodes and send fake invalidation messages, as shown in Figure 2. These fake invalidation messages will invalidate cache blocks that are in use by several nodes, which will result in performance degradation due to multiple reasons. (1) All the users of the cache blocks need to send read requests to reacquire the cache blocks, resulting in increased traffic. (2) Application execution may be halted due to an invalidated cache block. (3) Traffic hotpots may happen around the owner of the cache block with simultaneous requests. Kim et al.[29] shows severe performance degradation (more than 100%) due to fake cache invalidation requests. Therefore, authenticating cache invalidation packets is critical.

Our trust model assumes network interface (NI) to be trustworthy because they are fabricated in-house. A similar trust model has been used in [17], [16], [13]. Furthermore, we assume Siphash [22] is available for unicast authentication at each NI. This Siphash scheme uses the shared key between the sender and receiver to generate the message authentication code and then uses the same key to validate the authenticity of the message. Since the invalidation is sent as a single multicast message targeting multiple destinations, the existing authentication scheme cannot be used to authenticate the origin of the message. This allows a malicious adversary to send a fake cache invalidation impersonating the actual owner of the cache block. Section IV describes our proposed lightweight authentication scheme that utilizes the existing unicast authentication implementation.

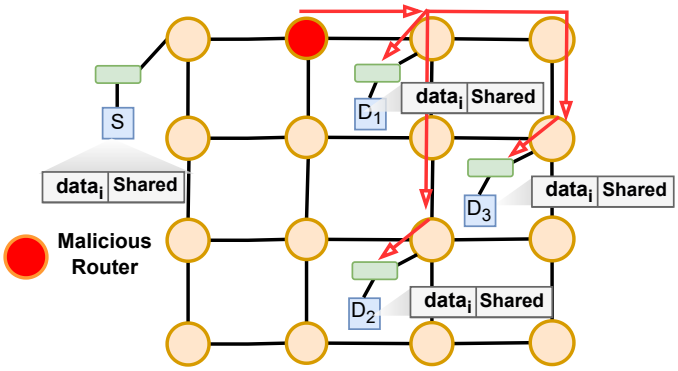

Fig. 2: Fake invalidation message is sent by a malicious router impersonating cache block owner (S). The shared data in the cache of D_1, D_2, and D_3 will be invalidated.

IV. Lightweight Multicast Authentication

Figure 3 shows an overview of our lightweight muticast authentication scheme. Our proposed scheme has two major components: (i) multicast MAC tag generation at sender, and (ii) multicast tag verification at receiver. The tag generation and verification procedures are shown in Algorithm 1 and Algorithm 3, respectively. Table I summarizes the notations used in these algorithms.

Algorithm 1 Multicast MAC tag generation at sender

1: **Input:** $\{M, k_{s,1} \dots k_{s,m}\}$
2: **Output:** multicast authenticated packet
3: $Tg \leftarrow \{1\}^r$
4: **for** i=1 to m **do**
5: $\quad tag \leftarrow siphash(k_{s,i}, M)$
6: $\quad ltag \leftarrow prng(tag)$
7: $\quad Tg \leftarrow Tg \odot \alpha_r(ltag)$
8: $pkt \leftarrow M||Tg$
9: **Return** pkt

Algorithm 1 highlights the major steps at the sender to generate accumulated MAC Tag (Tg). Tg is initialized to the '1's of length r. This is to ensure that it will adhere to the upper bound on a number of zeros to make multicast MAC tag unforgeable, which is elaborated in section IV-A. Lines 5-7 show steps to be conducted targeting each receiver of the multicast packet. There are m number of receivers where

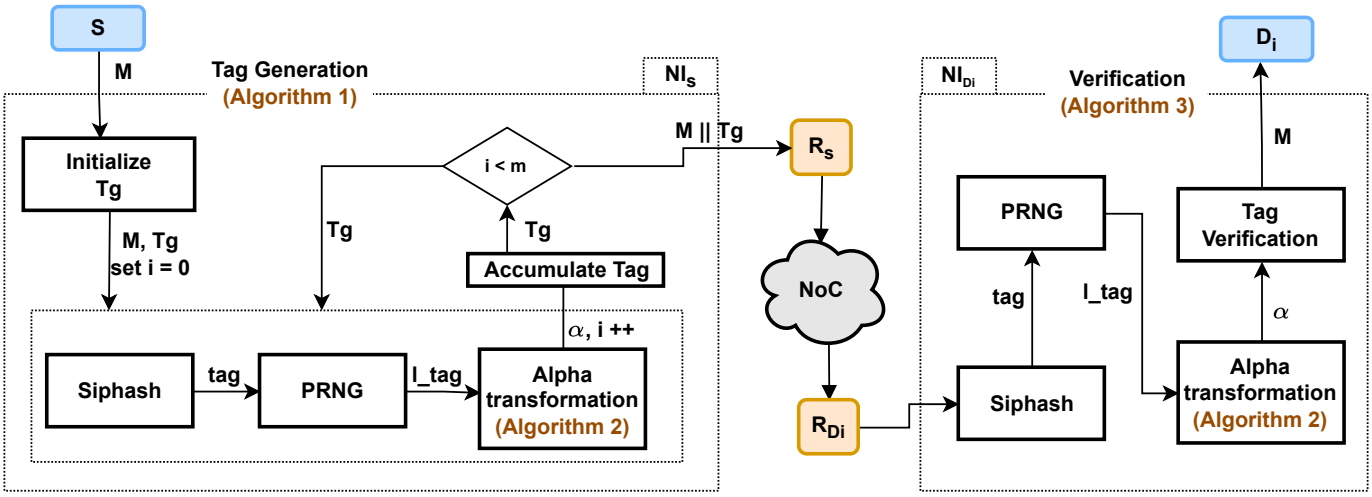

Fig. 3: Overview of our proposed lightweight multicast authentication. It consists of two major components: (i) tag generation by source(S) and (ii) verification by one of the multicast destinations (D_i). Both tag generation and verification are implemented in the network interface of the sender and receivers.

$m \leq N$. First, MAC tag is generated by the existing MAC algorithm for unicast which is Siphash. Siphash is a keyed hash function, where the sender uses the pre-shared symmetric key between sender and receiver ($k_{S,D}$) and the message payload as inputs to Siphash. This tag is short (64 bits), but the α_r needs a bit sequence of length $l = r \times d$. Therefore, we use a pseudorandom number generator that will map 64 bit short MAC to a l length long random bit sequence (line 6). Then α_r function will be applied on to the expanded tag where it will map it to a bit sequence of length r (line 7).

TABLE I: Table of notations.

$\odot$	Bitwise and operation
Tg	Accumulated Hash/Tag
h	Cryptographically secure hash function
s	Seed value
N	Upper bound of number of items accumulated
M	Message payload
$k_{s,i}$	Symmetric key between node s and i
d	$log_2(N)$
m	number of multicast destinations
r	length of Tg
t	Desired security level
z	Minimum number of 1's in Tg
$prng$	Pseudorandom number generator
X	Set of items to be accumulated
$\|\|$	Concatenation operator
α_r	alpha transformation: $\{0,1\}^l \rightarrow \{0,1\}^r, r < l$

The implementation of α_r is described in Algorithm 2, which is based on the procedure outlined in Section II-A. It takes the bit-sequence of length l and generates r-length bit-sequence (α value) of a receiver. First, the bit-string is parsed as r of d length sub-strings (line 3). Then it will replace every all zero bit-sub-strings with 0 and others with 1 (lines 5 - 9).

Finally, the α value of the receiver is accumulated to the Tg by applying bit-wise AND operation (line 7 in Algorithm 1). The generation of the accumulated MAC Tag can be summarized by modifying Equation 2 as follows.

$$Tg = \{1\}^r \odot \prod_{i=1}^{m} \alpha_r(prng(siphash(k_{s,i}, M))) \quad (3)$$

Algorithm 2 Alpha Transformation

1: **Input:** bit sequence $in \in \{0,1\}^l$
2: **Output:** bit sequence $out \in \{0,1\}^r$
3: Parse in as $y_1||\ldots||y_r$ where length(y_j) = d
4: $out \leftarrow$ empty bit sequence
5: **for** j=1 to r **do**
6: **if** $y_j = \{0\}^d$ **then**
7: $out \leftarrow out||0$
8: **else**
9: $out \leftarrow out||1$
10: **return** out

Equation 3 highlights that we have modified the original fast accumulation proposed by [15] by replacing the hash function by using a cryptographically secure hash (Siphash) and a pseudorandom generator. This allows the modified accumulation to be used for multicast authentication with bounds discussed in Section IV-A

Algorithm 3 is used by the receiver to verify the authenticity of the message using the accumulated MAC tag. First, the receiver parses the packet to separate out the actual message payload and the tag (line 3). Then Siphash is applied with the key between sender and receiver ($k_{s,i}$) and payload to generate a short tag (line 4). The short tag is extended to a long random bit sequence using a pseudorandom number generator (prng) by providing the short tag as the seed (line 5). Finally, a conditional check is done (line 6) to verify that the authenticity of the message is from the correct sender. This

Algorithm 3 Multicast MAC tag verification at receiver

1: **Input:** $\{pkt, k_{s,i}\}$
2: **Output:** authenticity of the packet
3: parse pkt as $M||Tg'$
4: $tag \leftarrow siphash(k_{s,i}, M)$
5: $ltag \leftarrow prng(tag)$
6: **if** $\alpha_r(ltag) \odot Tg'$ is Tg' **then**
7: valid pkt
8: **else**
9: spoofed pkt

is done by doing bitwise AND between $\alpha_r(tag)$ and Tg' and then comparing it again with Tg'.

A. Security Analysis

There are some group key-based protocols where a symmetric key is shared between a group. They pose a security threat of using the same key across multiple nodes where there's more possibility of key being leaked and a malicious node inside the group to launch attacks. Therefore, we did not consider lightweight symmetric group key sharing protocols for multicast authentication.

If we consider a unicast message authentication code, it should have the property of unforgery to be secure. If the hash function is proven to be secure, it cannot be forged. Since our approach uses a cryptographically secure has function (siphash), we can rely on the same security properties. Similarly, since prng uses siphash tag as the seed, the l length tag cannot be forged. Therefore, we can claim that individual accumulated hashes cannot be forged. We need to show that the accumulated hash cannot be forged by an adversary. Since the validation of the authenticity is done through bitwise AND operation ($\odot$), an attacker can send a forged accumulated signature with large number of 0s to increase the probability of the validation. For example, if the actual accumulated hash is $Tg = 11001000$, a_i is 11101100, and if adversary used all zero $\{0\}^8$ as the forged accumulated hash (Tg'), the tag will be verified because $\{0\}^8 \odot a_i = \{0\}^8$. Therefore, we need to define an upper bound on number of 0s in Tg depending on the security requirement (t). On the other hand, we need to define a lower bound on number of 1s to be on a valid Tg.

Assume that the number of ones in the r-length accumulated hash is z. We can think of the bit composition of the final accumulated MAC tag as a binomial distribution associated with two possible outcomes: (1) a bit being 1 and (2) a bit being 0. We can define discrete random variable Z as the number of 1's in the tag Tg. Then the probability of having z ones in a r-bit long tag is given by the probability mass function of a binomial distribution:

$$Pr(Z = z) = \binom{r}{z} p^k (1-p)^{r-z} \tag{4}$$

We can find the probability p by looking at the accumulation process. Since the probability of $a_{i,j}$ to be 0 after applying α_r is 2^{-d}, $Pr(a_{i,j} = 1) = 1 - 2^{-d}$. If we consider one bit of accumulated MAC tag Tg_i that accumulates individual MAC tags for m destinations:

$$p = Pr(Tg_i = 1) = (1 - 2^{-d})^m \tag{5}$$

By substituting probability p to the equation, we get the probability mass function as follows:

$$Pr(Z = z) = \binom{r}{z} (1 - 2^{-d})^m \left(1 - (1 - 2^{-d})^m\right)^{r-z} \tag{6}$$

Equation 6 gives the probability of having exactly z 1's in tag Tg. Our goal is to find a lower bound on number of 1's. Since, the probability of getting z or more 1's in bit sequence of Tg is equal to 1 minus the probability of getting z or fewer 1s ($1 - Pr(Z \leq z)$).

$$P(Z > z) = 1 - \sum_{i=0}^{z} \binom{r}{i} (1 - 2^{-d})^m \left(1 - (1 - 2^{-d})^m\right)^{r-i} \tag{7}$$

By examining Equation 7, we can observe that probability $Pr(Z > z)$ depends on r, d, and m. The discussion in Section II-A highlights that r depends on security strength t. For a particular configurations of SoC (NoC size and cache coherence protocol) of multicast authentication to secure cache invalidations, we can fix d and m. For example, when using MESI in a 4×4 mesh, the maximum possible m is 8 and d is 3 [30]. Therefore, the minimum value for z depends on r.

TABLE II: Minimum multicast tag length for increasing security levels when $N = 8$.

security level (t)	Minimum 1s in tag (z)	Minimum tag length (r)
4	32	128
6	48	196
8	64	262
10	80	330
15	120	500
20	160	672

Now let us focus on the verification of the proposed authentication scheme in a scenario when Tg' is a forged MAC by an adversary. Since the verification now has a lower bound on number of 1s on Tg', it cannot have 1s less than $min(z)$. Let there be z 1s in the final MAC tag. Then the probability of an arbitrary bit to be 1 on Tg is $Pr(Tg_j = 1) = z/r$. If we focus on α_r calculation of the i^{th} receiver, probability of $a_{i,j}$ to be 0 after applying α_r is 2^{-d} ($Pr(a_{i,j} = 0) = 2^{-d}$). If $Tg'_j \odot a_{i,j} \neq Tg'_j$, the receiver will detect forged MAC. For a successful forge, the attacker need to avoid this scenario for all r bits. The probability of doing so can be written as follows :

$$Pr_{forge} = 1 - \left(2^{-d} \times \frac{z}{r}\right)^r = \left(1 - \frac{z}{N \cdot r}\right)^r \tag{8}$$

For any real number, inequality $1 + x \leq e^x$ holds, therefore, for $x = -\frac{z}{N \times r}$, we deduce that $1 - \frac{z}{N \times r} \leq e^{-z/Nr}$. By raising both sides to the power of r, we obtain:

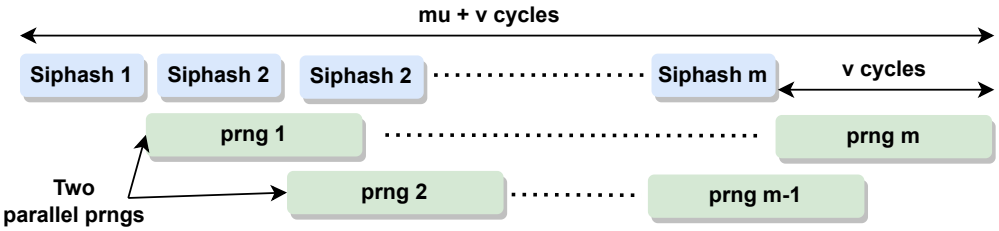

Fig. 4: Pipelining siphash and two prng for speedup.

$$Pr_{forge} = \left(1 - \frac{z}{N \cdot r}\right)^r \leq e^{-z/N} \tag{9}$$

Now for fixed N and security level t we can calculate q using Equation 9. Then Equation 7 can be used to calculate respective r values. Table II shows calculated r and z values for increasing t values for $M = 8$.

B. Pipelined Multicast Authentication

The discussion from the previous section highlights that the length of the long tag after applying prng is relatively high. For example, when $N = 8$ and $t = 10$, the value of l is 652 bits. Generating long bit sequence takes more cycles than generating MAC using Siphahsh. Therefore, we use two prngs working simultaneously in pipelined manner as shown in Figure 4.

Assume that we need u cycles for generating single unicast MAC and v cycles for generating long MAC where $u < v$. When there are m multicast recipients, this approach will result in total of $mu+v+1$ cycles in total for generating multicast MAC tag. A non-pipelined approach will take $u+mv+1$ cycles.

V. Experiments

In this section, we first describe the experimental setup. Next, we present the results and demonstrate the performance and overhead of our approach.

A. Experimental Setup

We used the gem5 [31] simulator, which is a cycle-accurate full system simulator to evaluate our approach. The "GARNET2.0" model was used as on-chip interconnection model [32]. We modified the network interface (NI) and routers of gem5 source to simulate the XY-tree based multicast routing. For securing unicast traffic, we model SipHash-2-4 [22] which produces a 64bit tag. For pseudo-random number generator in multicast signature, we model xoroshiro128+ [33]. Then, the proposed countermeasure was implemented on the NI of the gem5 source. We modified garnet synthetic traffic to generate multicast packets for evaluating our approach on synthetic traffic. Multiple benchmarks from SPLASH-2 and PARSEC benchmarks were run as applications to capture performance on actual traffic. The configuration parameters used in our experiments are outlined in Table III.

TABLE III: gem5 configuration parameters.

Synthetic traffic configuration	
Topology	4 x 4 mesh
Packet length	unicast : 1 flit and 5 flits multicat : 1 flit
Multicast ratio	10%
Multicast destination count	4-8 (uniformly random)
No. of nodes	16
Vnets	0: 1 flit multicast, 1: 1 flit unicast 2: 5 flit unicast
Full System Configuration	
Cahe Coherence protocol	MESI Two Level
Topology	4 x 4 mesh
No. of directories	16
Core frequency	2GHz
Instruction Set Architecture	x86
L1 Cache (I & D)	16KB
L2 Cache	256KB

The configurations were carefully chosen considering real multicast traffic characterization by [30]. For example, in 4x4 mesh, the destinations per multicast message in directory-based coherence vary between 2 and 8. We compare our approach (**MulAuth**) against two scenarios:

- **No-MulAuth**: NoC without supporting authentication of multicast packets. Note that unicast authentication uses Siphash 2-4 algorithm.
- **Pub-MulAuth**: Since there are no previous efforts of multicast authentication on NoC, we compare it with traditional public/private key multicast authentication. Here, multicast packets are authenticated with private key and validated with public key.

To evaluate the area overhead of our approach, we synthesized the network interface with proposed multicast authentication using Synopsys Design Compiler with ASAP7nm [34] library.

B. Performance Analysis

Figure 5 shows normalized packet latency for increasing packet injection rates when 10% of the total packets are multicast packets. The packet latencies are normalized against Pub-MulAuth packet latencies. For MulAuth, we fixed the security level to be 10 ($t = 10$). Our lightweight multicast authentication scheme incurs significantly less overhead compared to public key multicast authentication. For example, at packet injection rate 0.001 multicast authentication scheme introduces

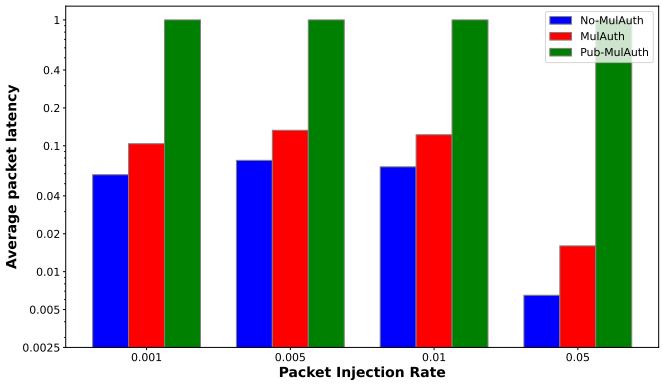

Fig. 5: Comparison of normalized average packet latency across increasing packet injection rates for synthetic traffic.

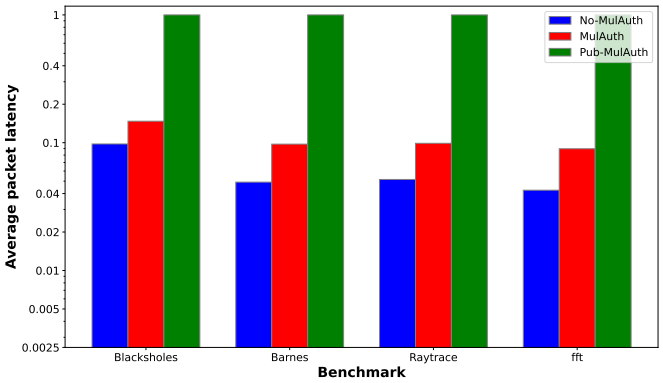

Fig. 6: Comparison of normalized average packet latency across SPLASH-2 and PARSEC benchmarks.

performance overhead of 0.7x while traditional public key multicast results in 15x performance overhead compared to no authentication on multicast packets. At higher injection rate of 0.1, our approach incurs relatively high overhead (1.4x times compared to no authentication) due to buffering of the multicast authentication process. At the same injection rate (0.1), traditional public key authentication incurs overhead of $\approx$ 400x times.

We evaluated our approach across multiple benchmarks of SPLASH-2 and PARSEC, namely, Blacksholes, Barnes, Raytrace, and fft. Figure 6 shows normalized average packet latency across the benchmarks for No-MulAuth, MulAuth, and Pub-MulAuth scenarios. For MulAuth, we fixed the security level to be 10 ($t = 10$). The packet latencies are normalized against Pub-MulAuth packet latencies. It can be observed that our proposed scheme behaves similarly across all benchmarks. When benchmarked against an unprotected multicast traffic scenario, our method exhibits an average performance overhead of 0.87x across all benchmarks. In contrast, traditional public key multicast authentication has a significantly higher average overhead of 17.3x. Synthetic and real traffic experimental results highlight that our proposed approach is suitable when multicast authentication is needed to secure NoC traffic.

Figure 7 shows average packet latency of synthetic traffic when we increase security of multicast authentication. When security level increases, the accumulated MAC tag length (r) increases according to Table II. This results in increase of packet latency in two ways: (1) prng need to generate longer bit-streams will incur more cycles, and (2) increase in r results in more flits per packet resulting in increase of NoC traffic congestion. This reconfigurability allows security designer to trade-off between security and performance of the system considering both threat model of the adversary and performance requirement.

C. Overhead Analysis

Our approach shows an area overhead of only 2.2% compared to the baseline network interface with unicast authentication. This is because baseline NoC has a network interface with logic for siphash and key tables. Our approach only needs additional logic for xoroshiro128+ prng and fast accumulation which are both lightweight. Therefore, our approach is ideal for resource-constrained NoC architectures.

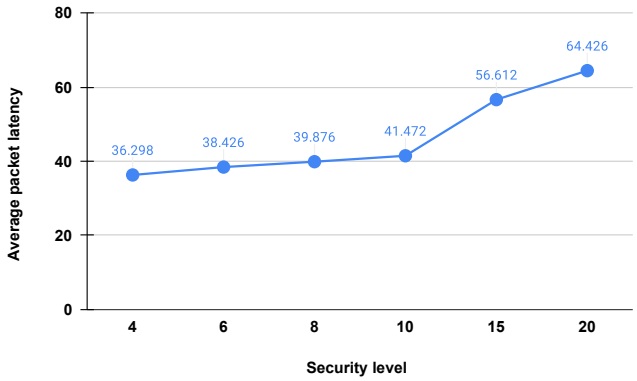

Fig. 7: Average packet latency across increasing security levels for 10% multicast packet percentage.

VI. Conclusion

Network-on-Chip (NoC) is a widely used solution for communication between IP cores in modern SoCs. The ubiquity of NoC and its distributed nature across the chip has made it a focal point of attacks. While there are existing solutions for protecting unicast traffic, they cannot be used for protecting multicast traffic. Moreover, traditional multicast authentication solutions can lead to unacceptable performance overhead. In this paper, we developed a lightweight multicast authentication scheme by leveraging state-of-the-art unicast authentication and accumulation techniques. Our approach is configurable and provides adequate security. Experimental results demonstrate the effectiveness of our approach with an acceptable area and performance overhead.

VII. Acknowledgments

This work was partially supported by the National Science Foundation (NSF) grant SaTC-1936040.

References

[1] Ampere, "Ampere Altra Max 64-Bit Multi-Core Processor," https://amperecomputing.com/processors/ampere-altra/, 2022.

[2] Intel, "Intel® xeon® processor scalable family technical overview," https://www.intel.com/content/www/us/en/developer/articles/technical/xeon-processor-scalable-family-technical-overview.html, 2022.

[3] C. E. Leiserson, N. C. Thompson, J. S. Emer, B. C. Kuszmaul, B. W. Lampson, D. Sanchez, and T. B. Schardl, "There's plenty of room at the top: What will drive computer performance after moore's law?" *Science*, vol. 368, no. 6495, p. eaam9744, 2020.

[4] M. Palesi and M. Daneshtalab, *Routing algorithms in networks-on-chip*. Springer, 2014.

[5] A. Karkar, T. Mak, K.-F. Tong, and A. Yakovlev, "A survey of emerging interconnects for on-chip efficient multicast and broadcast in many-cores," *IEEE Circuits and Systems Magazine*, vol. 16, no. 1, pp. 58–72, 2016.

[6] T. Krishna, L.-S. Peh, B. M. Beckmann, and S. K. Reinhardt, "Towards the ideal on-chip fabric for 1-to-many and many-to-1 communication," in *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, 2011, pp. 71–82.

[7] P. Abad, V. Puente, and J.-A. Gregorio, "Mrr: Enabling fully adaptive multicast routing for cmp interconnection networks," in *2009 IEEE 15th International Symposium on High Performance Computer Architecture*. IEEE, 2009, pp. 355–366.

[8] W. Hu, Z. Lu, A. Jantsch, and H. Liu, "Power-efficient tree-based multicast support for networks-on-chip," in *16th Asia and South Pacific Design Automation Conference (ASP-DAC 2011)*. IEEE, 2011, pp. 363–368.

[9] F. A. Samman, T. Hollstein, and M. Glesner, "Multicast parallel pipeline router architecture for network-on-chip," in *Proceedings of the conference on Design, automation and test in Europe*, 2008, pp. 1396–1401.

[10] H. Weerasena and P. Mishra, "Security of electrical, optical and wireless on-chip interconnects: A survey," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 2023.

[11] H. K. Kapoor, G. B. Rao, S. Arshi, and G. Trivedi, "A security framework for noc using authenticated encryption and session keys," *Circuits, Systems, and Signal Processing*, vol. 32, no. 6, pp. 2605–2622, 2013.

[12] K. Sajeesh and H. K. Kapoor, "An authenticated encryption based security framework for noc architectures," in *2011 International Symposium on Electronic System Design*. IEEE, 2011, pp. 134–139.

[13] J. Sepúlveda, A. Zankl, D. Flórez, and G. Sigl, "Towards protected mpsoc communication for information protection against a malicious noc," *Procedia computer science*, vol. 108, pp. 1103–1112, 2017.

[14] J. Benaloh and M. De Mare, "One-way accumulators: A decentralized alternative to digital signatures," in *Workshop on the Theory and Application of Cryptographic Techniques*. Springer, 1993, pp. 274–285.

[15] K. Nyberg, "Fast accumulated hashing," in *International Workshop on Fast Software Encryption*. Springer, 1996, pp. 83–87.

[16] M. K. JYV, A. K. Swain, S. Kumar, S. R. Sahoo, and K. Mahapatra, "Run time mitigation of performance degradation hardware trojan attacks in network on chip," in *2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 2018, pp. 738–743.

[17] R. JS, D. M. Ancajas, K. Chakraborty, and S. Roy, "Runtime detection of a bandwidth denial attack from a rogue network-on-chip," in *Proceedings of the 9th International Symposium on Networks-on-Chip*. ACM, 2015, p. 8.

[18] H. Weerasena, S. Charles, and P. Mishra, "Lightweight encryption using chaffing and winnowing with all-or-nothing transform for network-on-chip architectures," in *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2021, pp. 170–180.

[19] H. Weerasena and P. Mishra, "Breaking on-chip communication anonymity using flow correlation attacks," *arXiv preprint arXiv:2309.15687*, 2023.

[20] S. Charles and P. Mishra, "Securing network-on-chip using incremental cryptography," in *2020 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 2020, pp. 168–175.

[21] S. Charles and P. Mishra, "Reconfigurable network-on-chip security architecture," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 25, no. 6, pp. 1–25, 2020.

[22] J.-P. Aumasson and D. J. Bernstein, "Siphash: a fast short-input prf," in *International Conference on Cryptology in India*. Springer, 2012, pp. 489–508.

[23] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[24] D. Boneh, G. Durfee, and M. Franklin, "Lower bounds for multicast message authentication," in *Advances in Cryptology—EUROCRYPT 2001: International Conference on the Theory and Application of Cryptographic Techniques Innsbruck, Austria, May 6–10, 2001 Proceedings 20*. Springer, 2001, pp. 437–452.

[25] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. Tygar, "Spins: Security protocols for sensor networks," in *Proceedings of the 7th annual international conference on Mobile computing and networking*, 2001, pp. 189–199.

[26] P. Ning, A. Liu, and W. Du, "Mitigating dos attacks against broadcast authentication in wireless sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 4, no. 1, pp. 1–35, 2008.

[27] Q. Dong, D. Liu, and P. Ning, "Pre-authentication filters: providing dos resistance for signature-based broadcast authentication in sensor networks," in *Proceedings of the first ACM conference on Wireless network security*, 2008, pp. 2–12.

[28] X. Yao, X. Han, X. Du, and X. Zhou, "A lightweight multicast authentication mechanism for small scale iot applications," *IEEE Sensors Journal*, vol. 13, no. 10, pp. 3693–3701, 2013.

[29] M. Kim, S. Kong, B. Hong, L. Xu, W. Shi, and T. Suh, "Evaluating coherence-exploiting hardware trojan," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*. IEEE, 2017, pp. 157–162.

[30] S. Abadal, R. Martínez, J. Solé-Pareta, E. Alarcón, and A. Cabellos-Aparicio, "Characterization and modeling of multicast communication in cache-coherent manycore processors," *Computers & Electrical Engineering*, vol. 51, pp. 168–183, 2016.

[31] N. Binkert *et al.*, "The gem5 simulator," *SIGARCH Computer Architecture News*, 2011.

[32] N. Agarwal *et al.*, "GARNET: A detailed on-chip network model inside a full-system simulator," *ISPASS*, 2009.

[33] D. Blackman and S. Vigna, "Scrambled linear pseudorandom number generators," *ACM Transactions on Mathematical Software (TOMS)*, vol. 47, no. 4, pp. 1–32, 2021.

[34] L. T. Clark, V. Vashishtha, L. Shifren, A. Gujja, S. Sinha, B. Cline, C. Ramamurthy, and G. Yeric, "Asap7: A 7-nm finfet predictive process design kit," *Microelectronics Journal*, vol. 53, pp. 105–115, 2016.