Lightweight Anonymous Routing in NoC based SoCs

Subodha Charles, Megan Logan, Prabhat Mishra
Department of Computer and Information Science and Engineering
University of Florida, Gainesville, Florida, USA

***Abstract*—System-on-Chip (SoC) supply chain is widely acknowledged as a major source of security vulnerabilities. Potentially malicious third-party IPs integrated on the same Network-on-Chip (NoC) with the trusted components can lead to security and trust concerns. While secure communication is a well studied problem in computer networks domain, it is not feasible to implement those solutions on resource-constrained SoCs. In this paper, we present a lightweight anonymous routing protocol for communication between IP cores in NoC based SoCs. Our method eliminates the major overhead associated with traditional anonymous routing protocols while ensuring that the desired security goals are met. Experimental results demonstrate that existing security solutions on NoC can introduce significant (1.5X) performance degradation, whereas our approach provides the same security features with minor (4%) impact on performance.**

I. Introduction

The growth of general purpose as well as embedded computing devices has been remarkable over the past decade. This was mainly enabled by the advances in manufacturing technologies that allowed the integration of many heterogeneous components on a single System-on-Chip (SoC). Manufacturers of modern SoCs typically outsource multiple intellectual property (IP) cores from potentially untrusted third-party vendors. Therefore, the trusted computing base of the SoC should exclude the third-party IPs. In fact, measures should be taken since malicious third-party IPs (M3PIP) can launch passive as well as active attacks on the SoC. Such attacks are possible primarily because the on-chip interconnection network that connects SoC components together, popularly known as Network-on-Chip (NoC), has visibility of the entire SoC and the communications between IP cores. Previous efforts have developed countermeasures against stealing information [1], snooping attacks [2], and even causing performance degradation by launching denial-of-service (DoS) attacks [3]. This paper presents an effective solution to provide secure communication in the presence of M3PIPs operating under the following threat model.

Threat Model: Figure 1 shows an SoC with heterogeneous IPs integrated on a Mesh NoC. The two nodes marked as S (source) and D (destination) are trusted IPs communicating with each other through an M3PIP A. During design time, a third party IP provider can insert a hardware Trojan in a router that duplicates packets transferred through its ports and leaks data to a malicious IP running a program which has the following capabilities: (1) steal information if data is sent as plaintext. (2) If data is encrypted and header information is kept as plaintext, gather packets generated from the same source and intended to the same destination and launch complex attacks such as linear/differential cryptanalysis. The Trojan is able to act upon commands sent by the malicious program. A similar threat model was explored in [1], [4].

There is a classical trade-off between energy efficiency [5] and security/trust [6]. While computer networks can accommodate strong security techniques, utilizing them in resource-constrained NoCs can lead to unacceptable overhead. Previous work on lightweight encryption proposed smaller block and key sizes, less rounds of encryption and other hardware optimizations [7]. These optimizations can weaken the security guarantees. Furthermore, the header information that is sent as plaintext allows an M3PIP to gather packets from one communication session and launch more complex attacks to break the cipher. The requirement to hide the header information motivates the need for an anonymous routing scheme. However, using traditional anonymous routing schemes such as onion routing [8], [9] is not feasible in resource-constrained NoCs. Onion routing introduces significant performance overhead since the sender has to do several rounds of encryption before sending the packet to the network and each intermediate router has to perform decryption before forwarding it to the next hop. Adopting this in resource-constrained NoCs can lead to unacceptable performance overhead. While optimized anonymous routing is promising in MANETS (e.g., [10]), it cannot address the unique communication requirements of an NoC. *To the best of our knowledge, our approach is the first attempt in developing an anonymous NoC routing protocol.*

This work was partially supported by the NSF grant SaTC-1936040.

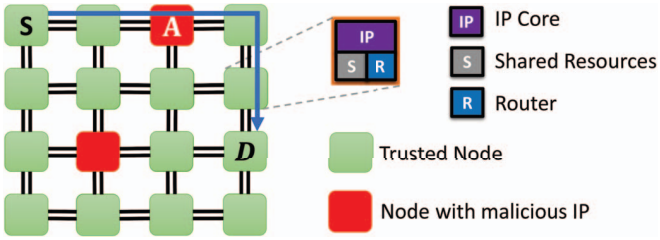


Fig. 1: A typical SoC architecture with a Mesh NoC.

In this paper, we propose a lightweight ***A**nonymous **R**outing protocol for **NoC**s (ARNoC)* that provides the desired security while staying within the power and performance budgets. As a result of using ARNoC, an intermediate node can neither detect the origin nor the destination of a packet. Major contributions of this paper can be summarized as follows:

- Our proposed anonymous routing hides both source and destination information making the packets untraceable.
- Our approach is lightweight compared to existing anonymous routing methods such as onion routing.

The remainder of the paper is organized as follows. Section II describes our anonymous routing protocol. Section III presents the experimental results considering both performance and security. Finally, Section IV concludes the paper.

II. ARNoC: Anonymous Routing for NoCs

Our proposed approach, ARNoC, negates the need for multiple layers of encryption and per-hop decryption when transferring data and as a result, achieves better performance and energy efficiency. First, we provide an overview of our framework in Section II-A. Next, Section II-B and Section II-C describe the two major components of ARNoC.

A. Overview

ARNoC has two main phases as shown in Figure 2. When an IP wants to communicate with another IP, it first completes the "Route Discovery" phase which sends a packet to discover the route and distributes some parameters among participants. This is done using a three-way handshake between the sender and the destination nodes. The handshake uses three (out of 4) types of packets sent over the network with the fourth type being used in the second phase. The four packet types are:

1) RI (Route Initiate) - flooded packet from sender S to destination D to initialize the communication session.
2) RA (Route Accept) - packet sent from D to accept new connection with S.
3) RC (Route Confirmation) - confirmation packet sent from S to D to indicate successful route discovery.
4) DT (Data) - the data packet from S to D that is routed anonymously through the NoC.

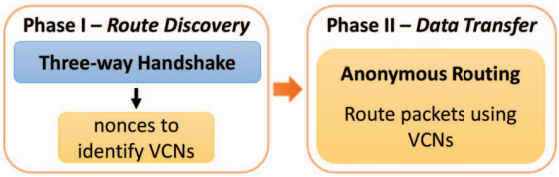


Fig. 2: Our proposed framework (ARNoC) consists of two phases: route discovery and data transfer.

During the three-way handshake, each router along the routing path is assigned with random nonces to represent preceding and following routers. The second phase, "Data Transfer", uses these parameters to forward the message through the route anonymously. Anonymous routing is achieved by using the random nonces which act as *virtual circuit numbers* (VCN). When transferring data packets, the intermediate routers will only see the VCNs corresponding to the preceding router and the following router which reveals no information about the source or the destination.

TABLE I: Notations used to illustrate ARNoC

$OPK_S^{(i)}$	one-time public key (OPK) used by S to uniquely identify an RA packet
$OSK_S^{(i)}$	the private key corresponding to $OPK_S^{(i)}$
ρ	a random number generated by S
PK_D	the global public key of D
SK_D	the private key corresponding to PK_D
$TPK_A^{(i)}$	temporary public key of node A
$TSK_A^{(i)}$	the private key corresponding to $TPK_A^{(i)}$
K_{S-A}	symmetric key shared between S and A
υ_A	a randomly generated nonce by node A
$E_K(M)$	a message M encrypted using the key K

The route discovered at the route discovery stage will remain the same for the lifetime of a task (until the task execution is complete), which is considered as one communication session. In case of context switching and/or task migration, a new communication session will start and the first phase will be repeated before transferring data. Each IP in the SoC that uses the NoC to communicate with other IPs follows the same procedure. The next two sections describe these two phases in detail. A list of notations used to illustrate the idea is listed in Table I. The superscript "i" is used to indicate that the parameter is changed for each communication session.

B. Route Discovery

The route discovery phase performs a three-way handshake between the sender S and destination D. This includes broadcasting the first packet - RI from S with the destination D, getting a response (RA) from D acknowledging the reception of RI, and finally, sending RC to complete route setup. Figure 3 shows an illustrative example of parameters (using only four nodes) shared and stored during the handshake. The initial route initiate packet (RI) takes the form:

$$\{RI \parallel OPK_S^{(i)} \parallel E_{PK_D}(OPK_S^{(i)} \parallel \rho) \parallel TPK_S^{(i)}\} \quad (1)$$

The first part of the message indicates the type of packet being sent, RI in this case. $OPK_S^{(i)}$ refers to the one-time public key associated with the sender node. This public key together with its corresponding private key $OSK_S^{(i)}$ change with each new communication session or RI. This change allows for a particular communication session to be uniquely identified by these keys, which are saved in its *route request table*. ρ is a randomly generated number by the sender that is concatenated with the $OPK_S^{(i)}$ and then encrypted with the destination node's public key PK_D as a global trapdoor [11]. Since PK_D is used to encrypt, only the destination is able to open the trapdoor using SK_D. Then the $TPK_S^{(i)}$ is attached to show the temporary key of the forwarding node, which is initially the sender. The next node, $r1$, to receive the RI messages goes through a few basic steps. First, it checks for the $OPK_S^{(i)}$ in its *key mapping table*, which would indicate a duplicated message. Any duplicates are discarded at this step. Next, $r1$ will attempt to decrypt the message and retrieve ρ. Success would indicate that $r1$ was the intended recipient D. If not, $r1$ replaces $TPK_S^{(i)}$ with its own temporary public key $TPK_{r1}^{(i)}$ and broadcasts:

$$\{RI \parallel OPK_S^{(i)} \parallel E_{PK_D}(OPK_S^{(i)} \parallel \rho) \parallel TPK_{r1}^{(i)}\} \quad (2)$$

$r1$ also logs $OPK_S^{(i)}$ and $TPK_S^{(i)}$ from the received message and $TSK_{r1}^{(i)}$ corresponding to $TPK_{r1}^{(i)}$ in its key mapping table. This information is used later when an RA message is received from D. D will eventually receive the RI message and will decrypt using SK_D. This will allow D to retrieve $OPK_S^{(i)}$ and ρ from $E_{PK_D}(OPK_S^{(i)} \parallel \rho)$. Then to verify that RI has not been tampered with, D will compare the plaintext $OPK_S^{(i)}$ and the recently decrypted $OPK_S^{(i)}$. If they are different, RI is simply discarded. Otherwise, D sends an RA (route accept) message:

$$\{RA \parallel E_{TPK_{r2}^{(i)}}(E_{OPK_S^{(i)}}(\rho \parallel \upsilon_D \parallel K_{S-D}))\} \quad (3)$$

RA, like RI in the previous message, is there to indicate message type. D generates a random nonce, υ_D, to serve as a VCN and a randomly selected key K_{S-D} to act as a symmetric key between S and D. D stores υ_D and K_{S-D} in its key mapping table. It also makes an entry in its routing table indexed by υ_D, the VCN. The concatenation of ρ, υ_D, and K_{S-D} is then encrypted with the $OPK_S^{(i)}$, so that only S can access that information. Then the message is encrypted again by $TPK_{r2}^{(i)}$, $r2$'s temporary public key, with $r2$ being the node that delivered RI to D. Once $r2$ receives the RA, it decrypts it using its temporary private key, $TSK_{r2}^{(i)}$, and follows the same steps as D. It generates its own nonce, υ_{r2}, and shared symmetric key, K_{S-r2}, to be shared with S. Both the nonce and symmetric key are then concatenated to the

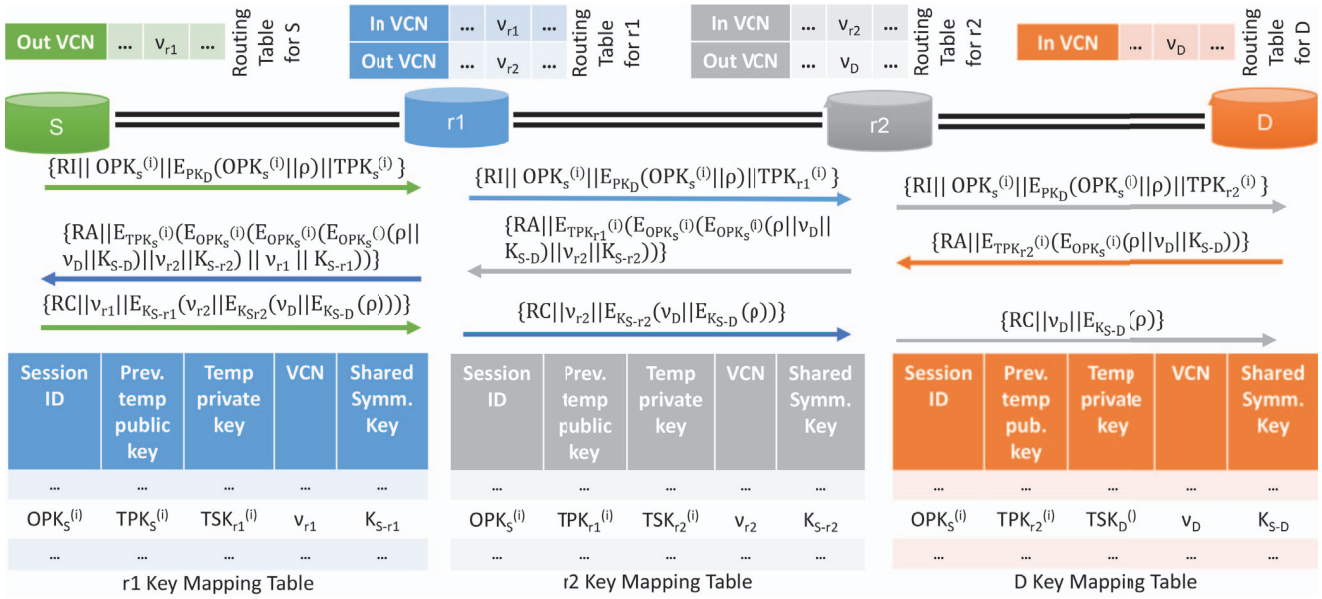


Fig. 3: Steps of the three-way handshake and the status of parameters at the end of the process.

RA message and encrypted by S's public key, $OPK_S^{(i)}$, so that only S can retrieve that data. This adds another layer of encrypted content to the message for S to decrypt using $OSK_S^{(i)}$. Similar to D, $r2$ also stores v_{r2} and K_{S-r2} in its key mapping table and routing table. It then finds the temporary public key for the previous node in the path from its key mapping table - $TPK_{r1}^{(i)}$ and encrypts the message. The message sent out by $r2$ looks like:

$$\{RA \parallel E_{TPK_{r1}^{(i)}}(E_{OPK_S^{(i)}}(E_{OPK_S^{(i)}}(\rho \parallel v_D \parallel K_{S-D}) \parallel v_{r2} \parallel K_{S-r2}))\} \quad (4)$$

This process is repeated at each node along the path until the RA packet makes its way back to S. The entire message at that point is encrypted with $TPK_S^{(i)}$, which is stripped away using $TSK_S^{(i)}$. Then S can "peel" each layer of the encrypted message by $OSK_S^{(i)}$ to retrieve all the VCNs, shared symmetric keys, and ρ. ρ is used to authenticate that the entire message came from the correct destination and was not changed during the journey. Once S completes authentication of the received RA packet, S constructs an RC packet:

$$\{RC \parallel v_{r1} \parallel E_{K_{S-r1}}(v_{r2} \parallel E_{K_{S-r2}}(v_D \parallel E_{K_{S-D}}(\rho)))\} \quad (5)$$

Similar to the case in RA and RI, RC in the packet refers to the packet type. The rest of the message is layered much like the previous RA packet. Each layer contains the v_* for each node concatenated with information that is encrypted with the shared key K_{S-*}, where * corresponds to $r1, r2$ or D in our example (Figure 3). The (v_*, K_{S-*}) pair was generated by each node during the RA packet transfer phase and the values were stored in the key mapping tables as well as entries indexed by the VCNs created in the routing table. Therefore, each node can decrypt one layer, store incoming and outgoing VCNs, and pass it on to the next node to do the same. For example, $r1$ receiving the packet observes that the incoming VCN is v_{r1}. It decrypts the first layer using the symmetric key K_{S-r1}, that is stored in the key mapping table, and recovers the outgoing VCN v_{r2}. It then updates the entry indexed by v_{r1} in its routing table with the outgoing VCN. Similarly, each router from S to D builds its routing table.

C. Data Transfer

The path set up can now be used to transfer messages from S to D anonymously. To transfer a message with sensitive content, M, from S to D anonymously, S constructs data transfer (DT) packet $\{DT \parallel v_{r1} \parallel E_K(M)\}$. DT, like every other packet, has an indicator of packet type at the front of the packet - DT. v_{r1} is the VCN of the next node. $E_K(M)$ is the encrypted sensitive data. Once $r1$ receives the DT packet, it uses its routing table to find the VCN of the next node and replaces the incoming VCN by the outgoing VCN in the DT packet. Therefore, the message received by $r2$ has the form $\{DT \parallel v_{r2} \parallel E_K(M)\}$. Next, $r2$ repeats the same process and forwards the packet $\{DT \parallel v_D \parallel E_K(M)\}$ to D. Eventually, D will receive the message and decrypt to recover the plaintext. Using this method, neither an intermediate node nor an eavesdropper in the middle will be able to see the routing path of the packet.

III. Experimental Results

A. Performance Evaluation

ARNoC was tested on an 8×8 Mesh NoC-based SoC with 64 IPs using the gem5 cycle-accurate full-system simulator [12]. The NoC was built using the "GARNET2.0" model that is integrated with gem5 [13]. Each encryption/decryption is modeled with a 12-cycle delay [7]. Our NoC model implements a credit-based flow control mechanism that captures congestion in the NoC during route discovery as well as data transfer phase and models relevant delays. A 4-port, 4-input buffer router with a 3-cycle pipeline delay was modeled at each node. Each link is assumed to take one cycle to transfer packets between two adjacent routers. Memory access was modeled with a 300-cycle delay. The delays were chosen to be consistent with the components in the gem5 simulator.

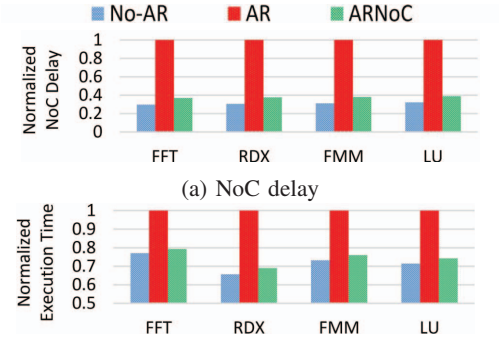
We ran 4 real benchmarks (FFT, RADIX, FMM, LU) from the SPLASH-2 benchmark suite [14] to test ARNoC. Out of the 64 cores, 16 IPs were chosen at random and each one of them instantiated one instance of the task. The packets injected into the NoC when running the benchmarks were the memory requests/responses. We used 8 memory controllers that provided

the interface to off-chip memory which were placed on the boundary of the SoC. The memory controller placement we used adheres to commercial SoC architectures such as Intel's Knights Landing (KNL) [15]. These choices were motivated by the architecture/threat model and the behavior of the gem5 simulator. However, ARNoC can be used with any other NoC topology and task/memory placement.

Figure 4 shows performance improvement that ARNoC can provide when running real benchmarks. We compare the results from ARNoC against two scenarios:

- **No-AR:** NoC without implementing anonymous routing.
- **AR:** Onion routing implemented as anonymous routing.

The values in Figure 4 are normalized to the scenario that consumes the most time. AR shows 70% (69% on average) increase in NoC delay (total NoC traversal delay for all packets) and 34% (28% on average) increase in execution time compared to the No-AR implementation. ARNoC improves NoC delay by 63% (62% on average) and total execution time by 30% (25% on average) when compared with AR. Overall, ARNoC can provide anonymous routing with only 4% performance overhead compared to the NoC that does not implement anonymous routing (No-AR).



(a) NoC delay

(b) Execution time

Fig. 4: NoC delay and execution time comparison across different security levels using real benchmarks.

The performance improvement of ARNoC comes from the fact that once the path has been set up for the communication between any two IPs, the overhead caused to securely communicate between the two IPs (data transfer phase) while preserving route anonymity is small. The notable overhead occurs at the route discovery phase due to complex cryptographic operations. Once the routing path is setup, packets can be forwarded from one router to the other by a simple table look-up. No per-hop encryption and decryption is required to preserve anonymity. The cost during the route discovery phase gets amortized over time since the route discovery phase occurs only once during the lifetime of a task (except context switching and/or task migration). This leads to a significant performance improvement compared to the traditional methods of anonymous routing.

B. Security Analysis

Security of messages: The security of message content is preserved by using encryption [7]. We are using existing NoC encryption methods to encrypt secure data of each packet.

Anonymity of nodes in the network: ARNoC preserves the anonymity of nodes in the network during all of its operational phases. When the source sends the initial RI packet to initiate the three-way handshake, it does not use the identity of the destination. Instead, the source uses the global public key of the destination (PK_D) and sends a broadcast message on the network. When the RI packet propagates through the network, each intermediate node saves a temporary public key of its predecessor. This temporary public key is then used to encrypt data when propagating the RA packet so that unicast messages can be sent to preceding nodes without using their identities. Random nonces and symmetric keys are assigned to each node during the RA packet propagation which in turn is used by the RC packet. Data transfer is done by looking up the routing table that consists of the nonces representing incoming and outgoing VCNs. Therefore, the identities of the nodes are not revealed at any point during communication.

Anonymity of routes taken by packets: In addition to preserving the anonymity of nodes, ARNoC also ensures that the path taken by each packet is anonymous. The routing table contains only the preceding and following nodes along the routing path. An M3PIP compromising a router will only reveal information about the next hop and the preceding hop. Therefore, the routing paths of all packets remain anonymous.

IV. Conclusions

Security and privacy are paramount considerations during electronic communication. Unfortunately, we cannot implement well-known security solutions from computer networks on resource-constrained NoC based SoCs. Specifically, these security solutions can lead to unacceptable performance overhead in embedded systems as well as IoT devices. In this paper, we proposed a lightweight anonymous routing protocol that addresses the classical trade-off between security and performance. Our anonymous routing protocol achieves superior performance compared to traditional anonymous routing methods such as onion routing by eliminating the need for per-hop decryption. Experimental results demonstrated that implementation of traditional anonymous routing solutions on NoC can introduce significant (1.5X) performance degradation, whereas our approach can provide the desired security requirements with minor (4%) impact on performance.

References

[1] J. Sepúlveda *et al.*, "Towards Protected MPSoC Communication for Information Protection against a Malicious NoC," *Proc. Comp. Sci.'17*.
[2] S. V. R. Chittamuru *et al.*, "Soteria: Exploiting process variations to enhance hardware security with photonic noc architectures," *DAC*, 2018.
[3] S. Charles *et al.*, "Real-time Detection and Localization of DoS Attacks in NoC based SoCs," *DATE*, 2019.
[4] D. M. Ancajas *et al.*, "Fort-nocs: Mitigating the threat of a compromised noc," in *DAC*, 2014.
[5] S. Charles *et al.*, "Efficient cache reconfiguration using machine learning in noc-based many-core cmps," *TODAES*, 2019.
[6] Y. Huang *et al.*, "Scalable test generation for trojan detection using side channel analysis," *TIFS*, 2018.
[7] K. Sajeesh and H. K. Kapoor, "An authenticated encryption based security framework for NoC architectures," *ISED*, 2011.
[8] J. Kong and X. Hong, *ANODR: anonymous on demand routing with untraceable routes for mobile ad-hoc networks*. MobiHoc'13, 2013.
[9] W. Yuan, "An anonymous routing protocol with authenticated key establishment in wireless ad hoc networks," *IJDSN*, 2014.
[10] Y. Qin *et al.*, "Olar: On-demand lightweight anonymous routing in manets," *ICMU*, 2008.
[11] J. Katz *et al.*, *Handbook of applied cryptography*. CRC press, 1996.
[12] N. Binkert *et al.*, "The gem5 simulator," *SIGARCH CA News*, 2011.
[13] N. Agarwal *et al.*, "GARNET: A detailed on-chip network model inside a full-system simulator," *ISPASS*, 2009.
[14] S. C. Woo *et al.*, "The splash-2 programs: Characterization and methodological considerations," *SIGARCH Computer Architecture News*, 1995.
[15] S. Charles *et al.*, "Exploration of memory and cluster modes in directory-based many-core cmps," in *NOCS*, 2018.