Secured Data Transmission over Insecure Networks-on-Chip by Modulating Inter-packet Delays

Jiaen Xu, Xiaohang Wang, *Member, IEEE,* Yingtao Jiang, *Member, IEEE,* Amit Kumar Singh, *Member, IEEE,* Chongyan Gu, Letian Huang, *Member, IEEE,* Mei Yang, *Member, IEEE,* and Shunbin Li

***Abstract*—As the network-on-chip (NoC) integrated into an SoC design can come from an untrusted third party, there is a growing risk that data integrity and security get compromised when supposedly sensitive data flows through such an untrusted NoC. We thus introduce a new method that can ensure secure and secret data transmission over such an untrusted NoC. Essentially, the proposed scheme relies on encoding binary data as delays between packets travelling across the source and destination pair. The maximum data transmission rate of this inter-packet-delay (IPD) based communication channel can be determined from the analytical model developed in this paper. To further improve the undetectability and robustness of the proposed data transmission scheme, a new block coding method and communication protocol are also proposed. Experimental results show that the proposed IPD-based method can achieve a packet error rate (PER) of as low as 0.3% and an effective throughput of 2.3×10^5 bps, outperforming the methods of thermal covert channel, cache covert channel, and circuit-based encryption, and thus is suitable for secure data transmission in unsecure systems.**

***Index Terms*—NoC, secure channel, inter-packet delay, block coding.**

Manuscript received April 07, 2022; revised June 11, 2022; accepted July 05, 2022. This work was supported in part by the National Natural Science Foundation of China under Grant 61971200, in part by Zhejiang Lab under Grants 2021LE0AB01 and 2021PC0AC01, in part by Major Scientific Research Project of Zhejiang Lab under Grant 2021LE0AC01, in part by the Open Research Grant of State Key Laboratory of Computer Architecture Institute of Computing Technology, Chinese Academy of Sciences under Grant CARCH201916, in part by the Key Technologies R&D Program of Jiangsu (Prospective and Key Technologies for Industry) under Grant BE2021003, in part by the Key Laboratory of Big Data and Intelligent Robot (South China University of Technology), Ministry of Education, and in part by the National Key Research and Development Program of China under Grant 2019QY0705. This article was presented in the International Conference on Compilers, Architectures, and Synthesis for Embedded Systems (CASES) 2022 and appears as part of the ESWEEK-TCAD special issue. *(Corresponding author: Xianghang Wang.)*

Jiaen Xu is with the School of Software Engineering, South China University of Technology, China (e-mail: jiaen_xu@mail.scut.edu.cn).

Xiaohang Wang is with the School of Software Engineering, South China University of Technology, and also with Zhejiang Laboratory and State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, China (e-mail: xiaohangwang@scut.edu.cn).

Yingtao Jiang and Mei Yang are with the Department of Electrical and Computer Engineering, University of Nevada at Las Vegas, Las Vegas, USA (e-mail: yingtao.jiang@unlv.edu; mei.yang@unlv.edu).

Amit Kumar Singh is with the School of Computer Science and Electronic Engineering, University of Essex, U.K. (e-mail: a.k.singh@essex.ac.uk).

Chongyan Gu is with the School of Electronics, Electrical Engineering and Computer Science at Queen's University Belfast, U.K. (email: C.Gu@qub.ac.uk).

Letian Huang is with the School of Electronic Science and Engineering, University of Electronic Science and Technology of China, China (e-mail: huanglt@uestc.edu.cn).

Shunbin Li is with Zhejiang Laboratory, China (e-mail: lishunbin@zhejianglab.com).

I. Introduction

PROTECTING data privacy and maintaining confidentiality during data transfer in a many-core chip are vital to ensure a system's overall security. Unfortunately, a many-core chip's network-on-chip (NoC) is perceived to be one of the most vulnerable spots in the security chain [25]. To reduce the time to market, a chip's NoC is often adopted from a third party vendor and there has been a great concern that it might have been compromised by hardware Trojans (HTs) [14], [49]. Unfortunately, completely removing all these hardware Trojans, particularly those conditionally triggered and switched on for a short duration of time, cannot be guaranteed at offline circuit testing stage. With the assistance of the HTs, highly sensitive information (*e.g.*, encryption key or password) can be compromised or leaked when it gets transferred between two cores across the NoC. In order to securely deliver data over an untrusted NoC, several approaches have been considered, noticeably authenticated encryption [1], [2] and network secure/covert channel [3], [4]. Of these two approaches, a secure/covert channel tends to preserve confidentiality better and incurs a lower information processing overhead. Essentially, a secure/covert channel is able to secretly transmit data on top of existing network traffic, rather than directly exposing data to adversaries as an encryption scheme does.

A secure channel can be built on vastly different media [20] and it has many transmission methods to choose from. Among these many diversified secure channels, timing covert channel [12], [13] has recently caught a great deal of attention. Over an NoC's timing covert channel, information can be encoded by varying packet order [5] or manipulating inter-packet delay (IPD). Defined as the time interval between two consecutive packets, IPD is one of the classic parameters determining covert transmission efficiency of data packets. In an IPD-based secure channel, the transmitter encodes bit '1' (bit '0') as a long (short) inter-packet delay. As shown in Fig. 1, the gap between packet 1 and packet 2 is 100 cycles (a long inter-packet delay), which corresponds to bit '1', and there is a gap of only 10 cycles (a short inter-packet delay) between

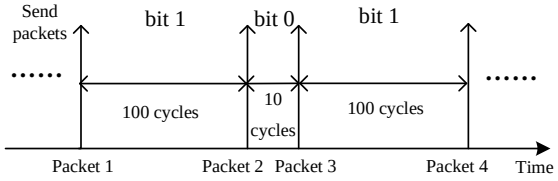


Fig. 1. An example showing how the time intervals between packet transmissions can be manipulated to embed binaries.

packet 2 and packet 3, which gives a bit '0'. The receiver recovers embedded binary data by comparing the IPDs against a predefined threshold.

Actually, IPD-based timing channels have been explored in wireless sensor networks or Internet as the cases in [6]–[8]. However, the network architecture, traffic pattern, and noise (data traffic created/transmitted by other applications/cores) of NoC are substantially different from those in wireless sensor network or Internet.

In this paper, an IPD-based channel is designed to transmit sensitive information in untrusted NoCs. Such an IPD-based channel, however, can be severely congested and thus blocked, if a malicious node in the NoC is capable to inject a large volume of useless traffic. To combat this problem, a block coding scheme is subsequently proposed in this paper that makes the packet delays of the IPD channel appear to be the result of the network traffic fluctuation. As a result, the malicious node will not be able to identify the existence of the IPD-based channel, and the data can still be safely transmitted.

The contributions of the paper are twofold. (1) To improve the stealthiness, we propose to adopt the $nBmT$ block coding to smooth out the delay distribution. Compared to codes seen in the IPD channels of the mobile wireless networks, the proposed $nBmT$ block coding is easy to implement with low cost and has a better channel capacity. (2) To obtain the optimal decision threshold on the receiver side, a novel BER model is presented.

The rest of the paper is organized as follows. Section II reviews the related work of the IPD-based secure channel. Sections III and IV introduce the IPD-based secure channel and the block coding scheme. In Section V, the performance of the IPD channel in NoC is evaluated and the paper is concluded in Section VI.

II. Related Work and Preliminary

A. Coding in IPD-based Secure Channels

IPD-based secure channels have been perceived to hide sensitive information in different network settings, such as in mobile network (*e.g.*, VoLTE [12]) and Internet [7], [11]. The first IPD-based covert/secure timing channel, presented in [11], uses different IPDs to represent different codes. However, this approach is sensitive to network jitter and has a high error rate when the network is congested. To address this problem, a variety of coding schemes have been developed. For instance, Liu *et al*. [15] adopted spreading code in the encoding process to reduce the impact of noise, but it has low transmission rate. In [13], the transferred data is coded using Huffman coding, which is found to improve both the channel capacity and its covertness. Houmansadr *et al*. [7] proposed CoCo which adjusts four different coding algorithms to encode the covert message bits. Liu *et al*. [17] reported a model-based covert channel using analog fountain code, and the model can be dynamically changed between the transmitter and the receiver. One drawback of using the fountain code is its increased complexity at both coding and decoding ends.

B. Transmission and Decoding in IPD-based Secure Channels

When an IPD packet leaves the transmitter and enters the network, it may be queued at the intermediate nodes, incurring extra routing and processing delays. When the delay caused by network congestion is much shorter than the IPD [11], the IPD will undergo a small deviation, which makes reliable data transmission over the IPD channel possible.

When receiving the encoded IPD sequence, in [10], a binary code is determined by checking whether a normal packet is received or not within a certain time interval. The receiver in [7], [15]–[17] adopts a thresholding method that compares IPDs with the threshold to determine the encoded message. This approach, however, suffers from degraded performance whenever there is network congestion or there is a packet loss.

C. Information Leakage Attacks and Countermeatures in NoC

A related but completely different work is the timing side channel. Both covert channel and side channel use media to establish channels for information transmission, but they are completely different. In side channel attacks, a spy infers the victim program execution trace by tracking power consumption or timing to infer the program's execution or password. Reinbrecht *et al*. [46] reported the Prime+Probe attack in NoC by measuring the throughput change to infer whether a cache access traffic traverses that sensitive path. If so, the adversary can infer the password with prior knowledge.

In the literature, insecure and unreliable NoCs were found to enable sinkhole/blackhole and data tampering HT attacks [50], and there are generally two methods to ensure secure computation/transmissions over an insecure SoC/NoC: 1) data encryption-based protection [45] that encrypts sensitive information directly based on AES or other encryption algorithms, and 2) information hiding-based computing [48], where sensitive information is hidden in the data packets or other media.

Alternatively, secure data transmission over an NoC can follow the method detailed in the next section.

III. IPD Channel in NoC

Secure data communication can be established between two cores in NoC through an IPD channel. The packets transmitted in NoC are referred as normal data packets. A covert (secure) packet is composed of multiple covert (secure) bits, which are represented by the time intervals between normal data packets. In this context, a transmitter is a program that is responsible for assembling and sending out data packets, and it also has an ability to set the time interval between two consecutive packets. The receiver, at the other end of the transmission, is a program that can figure out the time intervals between the normal packets received, from which it can recover the original covert data to be transmitted.

A. Threat Model

The proposed channel runs in an NoC designed for real time applications or mixed critical systems that a core can initiate direct packet communication with another core, as the case in Intel Singlechip Cloud Computer (SCC) [22], systems with scratchpad memory (SPM) [24], and Tilera TILE64 [23], *etc*. Each normal packet generated by the transmitter in the network is routed through a series of intermediate nodes before reaching the receiver. However, some of these intermediate nodes may be embedded with hardware Trojans (HTs), which

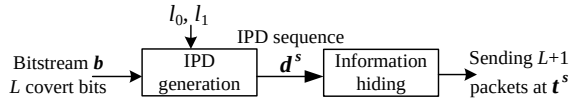


Fig. 2. The flow of an IPD channel at the transmitter.

can keep a local copy of any packet [14], or use hardware-supported debugging, such as the trace buffer [9], to export traffic traces to an unintended external analyzer. Either way there is a great danger of information leakage.

For each packet passing through the malicious node, the core or the external analyzer connected to its detector records the arrival time of each arriving packet, from which the detector determines the packet delay information. Once an IPD channel is detected, the detector injects useless packets or drops a few passing packets to disturb the network delays of the current transmission flow. In this case, the information transmitted in terms of IPD shall no longer be accurately extracted. However, if the malicious node constantly floods the network with packets, it might lead to network congestion and consequently performance degradation, which can be sensed by the user. To deal with this potential problem, a malicious node can be conditionally triggered [44] such that it cannot be easily detected and thus poses a great threat to the system security.

B. The Transmitter and Receiver Pair in NoC

As illustrated in Fig. 2, there are L covert bits, denoted as $\boldsymbol{b} = \{b_1, b_2, ..., b_L\}$, that need to be transmitted over the IPD channel. The data transmission goes through two steps.

Step 1. An IPD sequence of length L, denoted as $\boldsymbol{d^s} = \{d_1^s, d_2^s, ..., d_L^s\}$, is generated from $\boldsymbol{b}$, and $d_i^s \in \boldsymbol{d^s}$ is defined as:

$$d_i^s = \begin{cases} l_0 & b_i = 0 \\ l_1 & b_i = 1 \end{cases} \quad 1 \le i \le L \tag{1}$$

where $l_0(l_1)$ is the duration of IPD needed for the transmission of one single covert bit of '0' (bit '1').

Step 2. To transmit the IPD sequence $\boldsymbol{d^s}$, $(L+1)$ normal data packets need to be sent. The i-th bit of the covert message to be transmitted, b_i, is sent out at the time instance of $t_i^s \in \boldsymbol{t^s} = \{t_1^s, t_2^s, ..., t_{L+1}^s\}$, and it is determined as

$$t_i^s = t_{i-1}^s + d_{i-1}^s \quad 2 \le i \le L+1 \tag{2}$$

where t_{i-1}^s is the time instance to send the previous bit b_{i-1}.

As illustrated in Fig. 3, the receiver decodes the IPDs to obtain the original bit stream with three steps.

Step 1. The receiver receives a stream of normal data packets at different time instances, and records the arrival time of each packet, denoted as $\boldsymbol{t^R} = \{t_1^R, t_2^R, ..., t_{L+1}^R\}$.

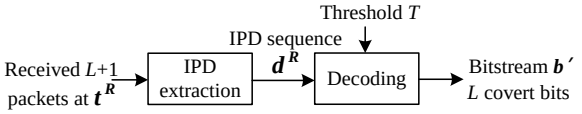


Fig. 3. Recovering the covert bits transmitted over an IPD channel in an NoC at the receiver.

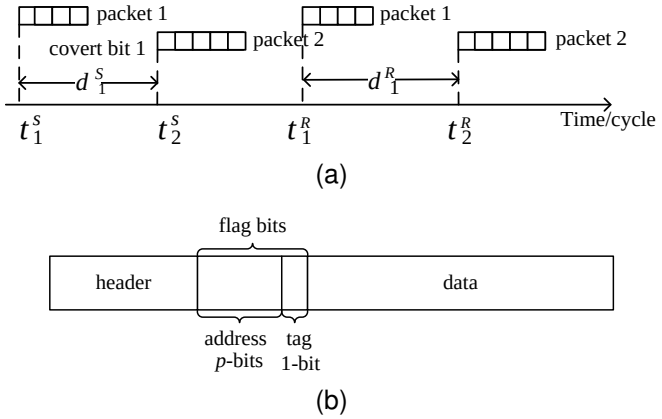


Fig. 4. (a) Illustration of a complete data transmission session: from the first covert bit that leaves the transmitter to the last covert bit that is received by the receiver, and (b) the data packet format that includes the flag bits.

Step 2. The received IPD sequence $\boldsymbol{d^R} = \{d_1^R, d_2^R, ..., d_L^R\}$ is obtained by computing the intervals between the arrival times of two consecutive packets:

$$d_i^R = t_{i+1}^R - t_i^R \quad 1 \le i \le L \tag{3}$$

For instance, if the data packets are received at time instances of {40, 50, 106, 171, 178}, the corresponding IPD's will be {10, 56, 65, 7}.

Step 3. The received covert binaries, $\boldsymbol{b'} = \{b'_1, b'_2, ..., b'_L\}$, are determined by comparing the IPD's against a threshold T_0. That is,

$$b'_i = \begin{cases} 0 & d_i^R < T_0 \\ 1 & d_i^R \ge T_0 \end{cases} \quad 1 \le i \le L \tag{4}$$

For example, if the predefined threshold is 55 cycles, the covert bit stream decoded from an IPD sequence {10, 56, 65, 7} is {0110}. Network congestion might add additional delay to the data transmission, causing a discrepancy between $\boldsymbol{d^s}$ at the transmitter and $\boldsymbol{d^R}$ observed by the receiver. Selection of the threshold must take into account of the extra delays induced by network congestion and buffering at various places of the NoC.

Fig. 4(a) shows a complete data transmission session. The transmitter sets the time interval d_1^s between the two normal data packets, packet 1 and packet 2, and it is determined by the logic value of the first covert bit needs to be transmitted, and the receiver is responsible for parsing the received IPD d_1^R. In order to notify the receiver that it needs to check IPDs for a secret message, the transmitter sets a fixed number of flag bits in a number of packets. As shown in Fig. 4(b), flag bits can be decomposed in two parts: the tag and the address. A one-bit tag determines if the normal packet carries secure information. If it is set to be '1', it means that the arrival time of the normal data packet should be parsed by the receiver. The remaining p flag bits indicate the transmitter's address, with $p = \log_2$(network size). For example, if the transmitter is the second node in an 8×8 network, the p flag bits for address are set to be '00010'.

The size of a covert packet shall be chosen by taking the decoding overhead into account. There are two cases to consider.

- When the decoding overhead is greater than l_0 and l_1, decoding covert packets may not keep pace with the

arrival of the normal data packets. To minimize the negative impact of decoding overhead, S_l covert bits should be sent together. These S_l covert bits will be broken down to S_p covert packets and each covert packet consists of S_n covert bits.

- On the other hand, when the decoding overhead is much lower than l_0 and l_1, information can be transmitted directly in the form of covert bits.

Multiplexed streams (that is, multiple IPD channels between different pairs of transmitter and receiver share the same physical channels/paths) can be sent in a similar manner, which are distinguished by the transmitters' addresses embedded in the normal data packets. Suppose node A sends two covert packets to nodes B and C. The times to send the covert bits (normal data packets) to both nodes will be governed by (2).

C. Modeling the IPD Channel

The effective throughput of an IPD-based channel, denoted as R, is given as

$$R = (1 - P_e) \times H \tag{5}$$

where H is the number of covert bits received per second, and P_e is bit error rate (BER). H can be taken as the average transmission rate over t^a time for a total of L bits, that is,

$$t^a = \sum_{i=1}^{L} d_i^R \tag{6}$$

$$H = \frac{L}{t^a} \tag{7}$$

A closer inspection of (6) to (7) reveals that the smaller values l_0 and l_1 can take, the less amount of time it takes to send the same number of bits, which is translated to a greater value of H, and correspondingly, a higher effective throughput.

The optimal threshold T^* corresponding to the maximium effective throughput is computed as follows. The derivative of (5) with respect to T, is given in (8). Since H is independent of T, it is treated as a constant in the derivation.

$$\frac{\partial R}{\partial T} = \frac{\partial (1 - P_e) H}{\partial T} = -H \frac{\partial P_e}{\partial T} \tag{8}$$

Let $P(1)$ be the prior probability of the transmitter sending bit '1' and $P(0)$ be the prior probability of sending bit '0'. Due to the existence of channel noise (*e.g.*, traffics generated by other applications), IPD encountered at the receiver end is likely to be different from that at the transmitter end. Let $P(0/1)$ be the probability of receiving bit '0' when bit '1' was actually sent, and $P(1/0)$ be the probability of receiving bit '1' while bit '0' was sent. Correspondingly, P_e in (5) can be calculated as:

$$P_e = P(1)P(0/1) + P(0)P(1/0) \tag{9}$$

Let $d_0(T)(d_1(T))$ be the probability distribution function of packet delays observed by the receiver when sending a bit '0' (bit '1'). As shown in Fig. 5, when T_0' is chosen as the threshold, $P(0/1)$ and $P(1/0)$ can be expressed as:

$$P(0/1) = 1 - \int_{-\infty}^{T_0'} d_0(T) \mathrm{d}T \tag{10}$$

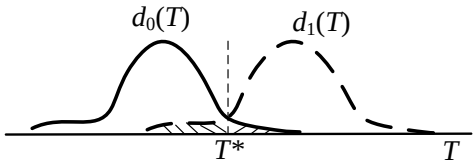


Fig. 5. Distributions of $d_0(T)$ and $d_1(T)$

$$P(1/0) = 1 - \int_{T_0'}^{+\infty} d_1(T) \mathrm{d}T \tag{11}$$

Substituting (10)(11) into (9), we can obtain P_e, and P_e achieves its minimum by setting its derivative with respect to the threshold equal to zero:

$$\frac{\partial P_e}{\partial T} = P(1) d_1(T) + P(0) d_0(T) \tag{12}$$

$$\frac{P(1)}{P(0)} = -\frac{d_0(T)}{d_1(T)} \tag{13}$$

Through extensive experiments, $d_0(T)$ and $d_1(T)$ are fitted as Gaussian distribution, and their expectations are l_0 and l_1. When $P(0) = P(1)$, the optimal threshold T^* at the receiver side is at the intersection of $d_0(T)$ and $d_1(T)$. $T^* = \frac{l_0 + l_1}{2}$ as given in [16].

D. Packetization in the IPD Channel

Many things can go wrong in transmitting data over an IPD channel. One big concern is related to packet loss that actually may occur in any network. As in the case of untrusted NoCs, an HT-infected node can intentionally drop packets passing the node. In this paper, we propose a communication protocol that supports reliable transmission over a secure channel. The protocol is depicted in Fig. 6 and works as follows.

1) The transmitter first needs to send a request packet (REQ) to inform the receiver to start a communication session and next waits for the receiver's reply. Once the receiver receives the REQ packet, it replies an acknowledgement (ACK) back to the transmitter. The transmitter does not send covert packets until it receives the ACK packet. After all the covert packets embedded into the time intervals of normal data packets are sent, the transmitter sends a terminate packet (TER) to the receiver to terminate the transmission.
2) The transmitter gathers the bits to be transmitted over the secure channel and then groups them into multiple

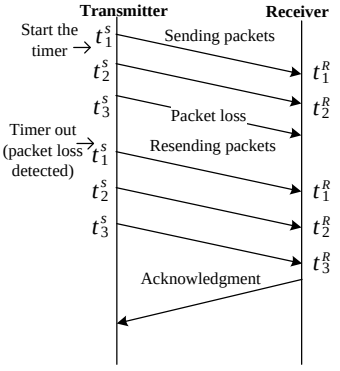


Fig. 6. The communication protocol to handle packet loss.

packets designated to the receiver. All covert packets have the same fixed size and each packet is assigned with a lifetime. The transmitter needs to keep track of each packet's lifetime, which can be achieved by setting a timer. Once the timer is expired, the packet will be discarded.

3) Before sending out a covert packet, the transmitter evaluates the congestion level of the network by sending probe packets and evaluating the roundtrip delays. If the network is found not congested, the transmitter is allowed to send packets. Since there might be a lot of noise in the network, the IPD channel communication may suffer packet error or loss. The receiver sends a NACK if an error is detected in the received covert packet. The transmitter, upon receiving a NACK or receiving no packet for a certain amount of time, will retransmit the packets since the last NACK. If the network is congested for a long time, the IPD channel can still work by increasing the decision threshold at the receiver.
4) Each covert packet is sent in the same way as any other normal data packets. For each covert packet, if no packet loss occurs during the transmission, the delays of all the received normal data packets shall be shorter than a packet's lifetime. On the other hand, the covert packet transmission fails if any packet gets lost. As shown in Fig. 6, the first three normal data packets constitute a covert packet. When the transmitter does not receive an acknowledgment within the lifetime of the covert packet, it is considered that a covert packet is lost, and the lost packet gets retransmitted.
5) After receiving all normal data packets that can form a covert packet, the receiver sends back an acknowledgement. The transmitter holds off its transmission of the next covert packet until it receives the acknowledgement coming from the receiver. In this case, the receiver will not reply to every single covert bit received. Instead, the transmitter sends a group of covert bits, and it will stop and wait for the acknowledgement from the receiver before it can send another group of covert bits.

E. Detection Scheme

A malicious node sitting in the path that links the transmitter to the receiver obtains the IPD distribution. As a packet passes through the malicious node, its arrival time gets recorded and analyzed along with other packets' arrival times, from which the packet delay distribution will be determined. Data traffics generated from running normal applications in the PARSEC and SPLASH-2 benchmark suits can be viewed as the noise to the IPD channel, and the delay distribution of such noise is found to follow a uniform-like distribution, as shown in Fig. 7. Since the time intervals of data packets in the IPD channel are used to encode bit '0' and bit '1', the probability distribution of IPD corresponding to bit '0' and bit '1' is drastically different from that of noise (two peaks corresponding to bits '0' and '1' as shown Fig. 7). The detector can draw a conclusion that there exist an IPD channel if there are big differences between the packet delay distribution of one port and that of other ports.

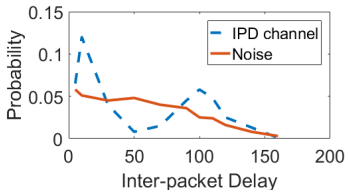


Fig. 7. The distribution of IPD in an 8×8 NoC, where l_1 = 100 cycles, l_0 = 10 cycles.

Given that the packet delay of the IPD-based channel has two peaks, it can be detected by a statistical test, like the Kullback-Leibler (KL) Divergence test [17]. Here the difference between the IPD distribution of noise $N(t)$ and that of the IPD channel $I(t)$, $D_{KL}(N||I)$, is given as:

$$D_{KL}(N||I) = \sum_t N(t) \log \frac{N(t)}{I(t)} \tag{14}$$

To detect an IPD-based channel signal in a malicious node, the following steps are performed.

Step 1. $N(t)$ is computed by checking the data traffic generated by various applications.

Step 2. For each transmission stream, all packets are recorded and then $I(t)$ is computed. If there is a significant difference between $N(t)$ and $I(t)$, *i.e.*, $D_{KL}(N||I)$ is greater than a threshold D_T, the existence of an IPD channel is confirmed.

Once the detector (a malicious node) finds an ongoing secure transmission, it floods the network channel with useless packets to alter the latencies of IPD traffic or selectively drops packets, which essentially makes data recovery from the IPD data impossible.

IV. Block Coding Based IPD Channel

In order to make IPDs less likely detectable, the IPD distribution of the secure channel signals should appear to be similar to noise, which can be achieved by the use of the $nBmT$ based channel coding [21]. The advantage of the $nBmT$ code is that it increases the diversity of packet intervals such that the IPD distribution is "smoothed" out. Since this coding scheme guarantees the minimum Hamming distance, it tends to have a low overhead and produce high effective throughput.

The receiver needs multiple thresholds to make decisions on the received bit stream. However, using multiple thresholds may actually result in higher BER. To keep the BER at a reasonably low level, the codes should have ingenuous error resilience. As long as the minimum Hamming distance d_M of code groups meets the requirement of $d_M \geq 2e + 1$ [19], it is guaranteed that up to e-bit errors in a block code can be corrected.

A. Mapping Binaries to $nBmT$ Codes

For an $nBmT$ code, it transforms n-bit binary symbols into m-bit ternary symbols for a total of $C(^n_m)$ possibilities. It is important to find the optimal mapping in terms of the minimum Hamming distance, and the distributions of ternary values '0', '1', and '2', denoted as p_0, p_1, and p_2, are ideally to be equal. As shown in (15), we use the variance, denoted as $var(p_0, p_1, p_2)$, to measure the actual differences among p_0, p_1, p_2. The variance needs to be minimized so that p_0, p_1,

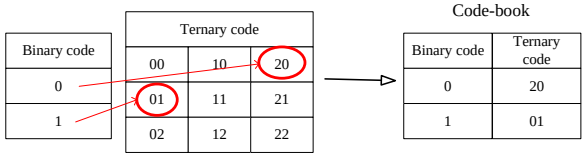


Fig. 8. Using $1B2T$ to generate a code-book.

and p_2 are poised to be as close to each other as possible. That is,

$$\begin{aligned} \min\{var(p_0,p_1,p_2)\} = \min\{\frac{\sum_{i=0}^{2}(p_i-\mu)}{3}\} \\ s.t. \quad p_0+p_1+p_2=1 \\ hd \geq min\_hd \end{aligned} \tag{15}$$

where μ is the average of p_0, p_1, p_2, and min_hd is the expected minimum Hamming distance.

Solving (15) requires all the possibilities are exhaustively checked, which is done beforehand. Since one ternary digit (*a.k.a.*, a trit) encodes three values, and 2 trits in $1B2T$ in Fig. 8 are equivalent to nine binary symbols, two of them are selected to represent the codes for a single binary bit. In this case, any 1-bit error code can be corrected and the minimum variance is 0.013. Before a data transmission session is established, the code-book has to be shared between the transmitter and the receiver.

The transmitter and receiver that use $nBmT$ coding are introduced as follows.

B. The Transmitter

As shown in Fig. 9, the transmitter first needs to divide L bits into L/n n-bit codes, denoted as $\boldsymbol{p^s} = \{p_1^s, p_2^s, ..., p_{L/n}^s\}$. Zeroes are padded if the bit length is not divisible by n. Each code in $\boldsymbol{p^s}$ is mapped to an m-bit ternary code. Let $\boldsymbol{ts^s}$ be the ternary stream, $ts_i^s \in \boldsymbol{ts^s}$, where $0 \leq i \leq L \times m/n$. The IPD generation module creates an IPD sequence $\boldsymbol{pd^s} = \{pd_1^s, pd_2^s, ..., pd_{L\times m/n}^s\}$ from $\boldsymbol{ts^s}$. For $pd_i^s \in \boldsymbol{pd^s}$, we have

$$pd_i^s = \begin{cases} l_0' & ts_i^s = 0 \\ l_1' & ts_i^s = 1 \\ l_2' & ts_i^s = 2 \end{cases} \quad 1 \leq i \leq L \times m/n \tag{16}$$

where l_0', l_1', and l_2' correspond to the three levels of IPD duration between two normal data packets. Next, all the normal data packets with the intervals stipulated as $\boldsymbol{pd^s}$ are injected into the network, following a flow similar to what

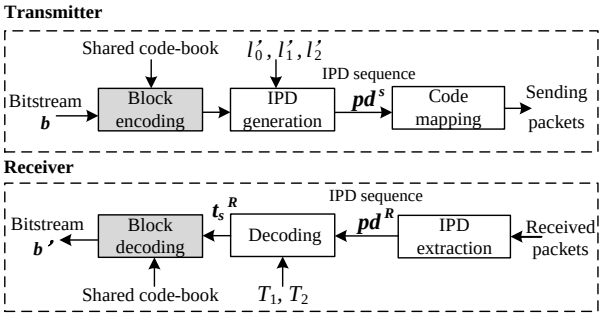


Fig. 9. The transmitter and receiver of a secure channel with signals in $nBmT$ block codes

is described in Section III-B. Given the mapping between the binary code and the ternary code shown in Fig. 8, a binary bit string of {010} can be converted into 2-digit ternary codes of {20,01,20}. The IPD sequence will then be $\{l_2', l_0', l_0', l_1', l_2', l_0'\}$.

C. The Receiver

The receiver extracts the IPD sequence $\boldsymbol{pd^R} = \{pd_1^R, pd_2^R, ..., pd_{L\times m/n}^R\}$ from its received packets. Two pre-set thresholds T_1 and T_2 are used to determine the ternary value of $ts_i^R \in \boldsymbol{ts^R}$:

$$ts_i^R = \begin{cases} 0 & pd_i^R < T_1 \\ 1 & T_1 \leq pd_i^R < T_2 \\ 2 & T_2 \leq pd_i^R \end{cases} \quad 1 \leq i \leq L \times m/n \tag{17}$$

As shown in Fig. 9, the block decoding module divides $\boldsymbol{ts^R}$ into L/n m-bit codes $\boldsymbol{q^R} = \{q_1^R, q_2^R, ..., q_{L/n}^R\}$. The next step is to look up the code-book and find the corresponding binary code of $\boldsymbol{q^R}$. As discussed above, the ternary codes in the code-book have error correction capability. The last step is to extract the binary codes to recreate the original bit stream. On the receiver side, for a received IPD sequence of {10, 56, 105, 7} and with the predefined thresholds T_1 and T_2 set to be 32 and 78 cycles, respectively, the parsed ternary stream is {0120}, which is divided into two 2-digit ternary codes, and the ternary codes are mapped to the original two binary 1-bit codes {1,0} according to the code-book shown in Fig. 8.

V. Experimental Results

A. Experimental Setup

Our experiments are performed on a cycle accurate NoC simulator [18] with all the configurations listed in Table I. The traffic traces are obtained from randomly generated traffic mixed with uniform, shuffle, butterfly, transpose, and hotspot patterns [29] as well as real traffic traces, which altogether pose as the noise to the IPD channels. Part of real traffic is generated by running gpgpu-sim [34] with the ispass benchmarks; the NoC configuration of gpgpu-sim is the same as the one used in [18] and reproduced in Table I. The other part of real traffic is extracted from an 8×8 NoC-based many-core simulator [28] running the PARSEC [27] and SPLASH-2 benchmarks, and the NoC configuration for this simulator is also the same as the one used in [18] and reported in Table I. Each node has a core, a router, an L1 D-cache, an L1 I-cache, an L2 cache bank, and a network interface. Two cores are picked as the transmitter and the receiver, and the malicious detector is a node sitting in the path that connects the transmitter and the receiver.

We evaluate the performance of the baseline scheme and the block coding scheme with noises, that is, traffics from both randomly generated benchmarks and real benchmarks. Instead of BER, we use packet error rate (PER) as the figure of merit to measure the error rate of covert data packets. PER is the ratio of the number of packets not successfully received to the number of packets transmitted. That is,

$$PER = \frac{N_e}{N_{all}} \times 100\% \tag{18}$$

TABLE I
PARAMETERS OF THE EXPERIMENTAL PLATFORMS

Configuration of the NoC simulator	
Baseline topology	8×8 mesh
Flit size	256 bits
Packet size (number of flits)	5
NoC latency	router: 2 cycles; link: 1 cycle
Number of virtual channels	4
Input buffer size (number of flits)	4
Routing algorithm	XY routing
Number of covert bits per packet	8
Number of covert packets	1000
Random traffic patterns	uniform, shuffle, butterfly, transpose, hotspot
Packet injection rate	0.002-0.1 packets/cycle/router
Network sizes	4×4 / 8×8 / 12×12 / 16×16
Configuration of gpgpu-sim for extracting traffic traces	
Core / cluster	16
Warp size	32
Shared memory / core	48KB
L1 D-cache size	16KB
L1 I-cache size	2KB
Texture cache size	8KB
Constant cache size	12KB
Bandwidth / memory module	8 bytes / cycle
Benchmarks	RAY, STO, WP, LIB, LPS
Configuration of the many-core simulator for extracting traffic traces	
Core architecture	64-bit Alpha 21264
Core frequency	3GHz
Fetch / decode / commit size	4 / 4 / 4
ROB size	64
L1 D cache (private) size	16KB
L1 I cache (private) size	32KB
L2 cache (shared) size	64KB
Main memory size	2GB
Benchmarks	barnes, blackscholes, canneal, dedup, ferret, freqmine, raytrace, swaptions

where N_e is the number of error packets received, and N_{all} is the total number of packets sent from the transmitter. The effective throughput defined in (5) is also used to evaluate performance of the IPD channels. The detection accuracy is used to evaluate the undetectability of the IPD channels. Assume that every time an IPD channel transmits information, the malicious node detects the existence of the IPD channel. Let f_r be the number of times the IPD channels are transmitting information, and f_d be the number of times the malicious node successfully detects the IPD channels, which is obtained by the method introduced in Section III-D. A successful detection of the malicious node means that an IPD channel is detected when transmitting information, and this number is counted. The detection accuracy P_{acc} is defined as:

$$P_{acc} = \frac{f_d}{f_r} \times 100\% \tag{19}$$

Here we set f_r to be 1000 experimentally.

B. Selection of Parameters

1) Parameters for the baseline IPD channel: Selection of the threshold T_0 and short/long IPDs (l_0, l_1) is essential to tune the performance of the baseline IPD channel. From Fig. 10(a) and 10(b), one can see that PER decreases as the difference between l_0 and l_1 becomes larger. As such, in our simulation, we set l_0 and l_1 to be 10 and 50 cycles. Fig. 10(c) measures the PER with different values of T_0. One can see that when

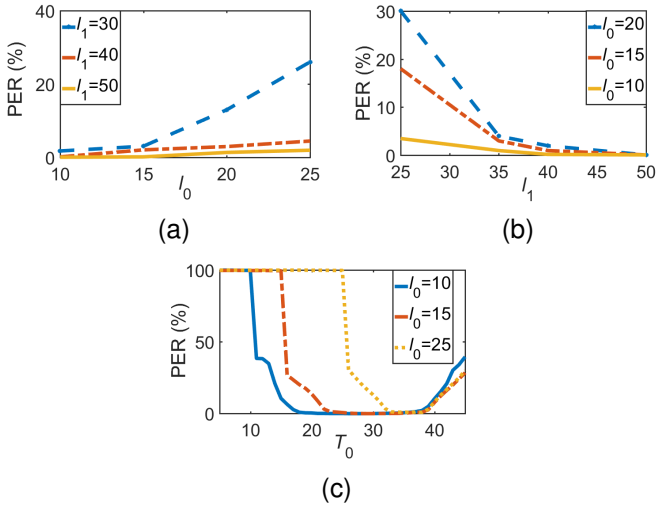


Fig. 10. The PERs of the baseline IPD channel by varying (a) l_0, (b) l_1, and (c) T_0. In (a), T_0=20, in (b), T_0=25, and in (c) l_1=50.

T_0 is $\frac{l_0+l_1}{2}$, PER reaches its minimum, which validates what is presented in Section III-C. So we set T_0 to be 30 cycles experimentally. In the following experiments, l_0, l_1, and T_0 are thus so set.

2) Parameters for the block coding based IPD channel: Compared with the baseline scheme, the channel based on block coding needs to determine values of a few additional parameters, including l_0', l_1', l_2', and two thresholds T_1 and T_2. In order to make the throughput of the block coding scheme closer to that of the baseline, we choose l_0' and l_2' to be 10 and 50 cycles, respectively. Yet we still need to determine the value of l_1'. From Fig. 11(a), one can see that when l_0' and l_2' are fixed, PER reaches its minimum with l_1' set to be $\frac{l_0'+l_2'}{2}$. As such, in our simulation, we set l_0', l_1', and l_2' to be 10, 30, and 50 cycles respectively. PER increases as T_1 and T_2 are close to l_1', as shown in Fig. 11(b) and 11(c). Similar to the setting of T_0 in the baseline IPD channel, we set T_1 and T_2 to be $\frac{l_0'+l_1'}{2}$ and $\frac{l_1'+l_2'}{2}$; that is, $T_1 = 20$ and $T_2 = 40$ cycles.

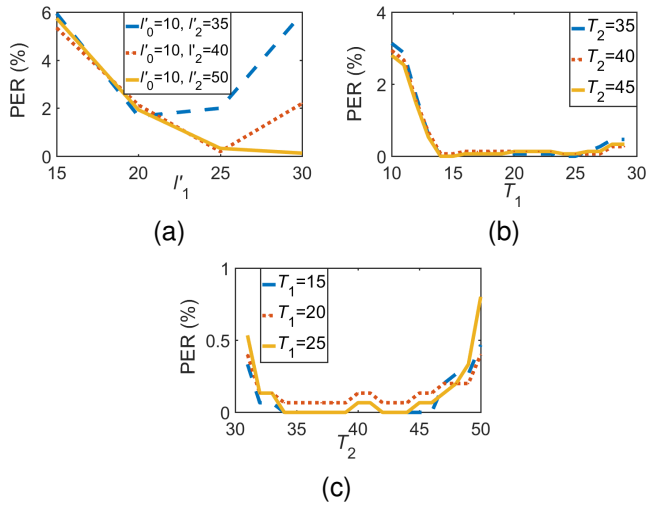


Fig. 11. The PERs of the block coding based IPD channel by varying (a) l_1', (b) T_1, and (c) T_2. In (a), l_0'=10, T_1=15, T_2=30, in (b) and (c), l_0'=10, l_1'=30, l_2'=50.

C. Performance Comparison

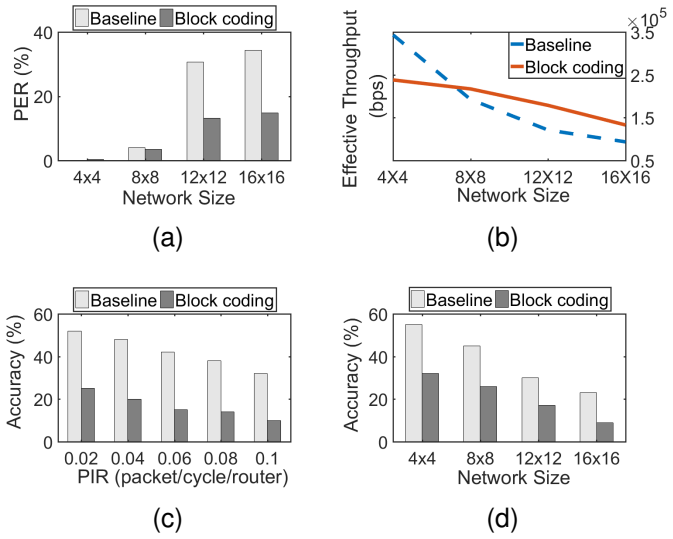


Fig. 12. (a) PER and (b) effective throughput with different network sizes, and (c)(d) detection accuracy against the two IPD channels by varying (c) PIR of other traffics (noise) and (d) network size. PIR is defined as the number of packets sent per cycle per router. The malicious node only detects but does not flood useless packets.

1) Experiments with random benchmarks: Fig. 12(a) shows the PERs obtained by the baseline and block coding based secure channels for different network sizes without flooding noise (*i.e.*, the malicious node detects but does not flood useless packets). For the small sized network (*e.g.*, 8×8), the block coding scheme reduces PER by 8% compared to that of the baseline method. As the network size gets larger, the effect of block coding scheme on reducing PER is more significant. This is particularly true when the system size is 16×16. In this case, thanks to the use of block coding, PER is reduced by 20%. With the increase of network size, there are more data packets transferred per cycle, which leads to the queuing of packets and increased network delay. As shown in Fig. 12(b), the block coding scheme has much lower PER and thus the effective throughput of the block coding scheme is 40% higher than the baseline with a network size of 16×16. The reason is that the error resilience of the block coding scheme plays an important role when the network is congested. When the network size is small (*e.g.*, 4×4), the block coding scheme contributes to less than 10% reduction in PER, but it takes 1.5× more packets to transmit the same amount of data. In this case, the effective throughput of the block coding scheme tends to be lower.

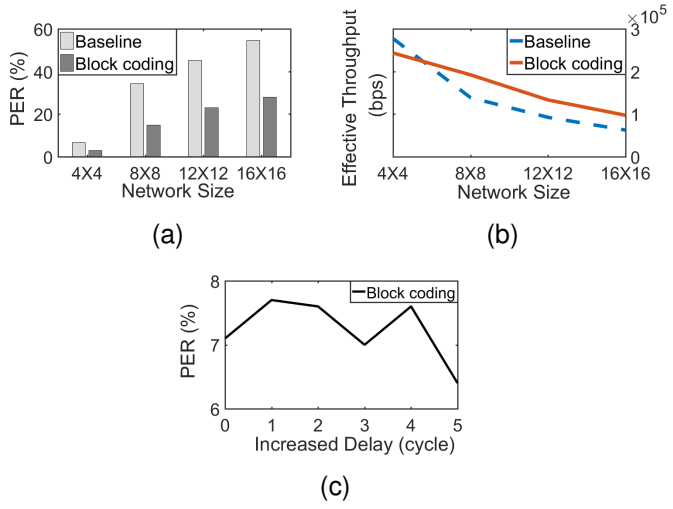


Fig. 13. (a) PER and (b) effective throughput with different network sizes. The malicious node floods useless packets. (c) PER of the block coding based method with malicious nodes that can increase IPD delays.

Fig. 12(c) shows the detection accuracy of the malicious nodes against the block coding scheme on average is half of the baseline scheme. From Fig. 12(d), one can see that when the network size is 16×16, the detection accuracy against the block coding based IPD channel decreases by 22% over the baseline. The reason is that both high packet injection rate (PIR) and large network size can increase the network delay and PER, resulting in irregular time intervals after network transmission.

Once the detector in the malicious node finds the existence of an IPD channel, it injects useless packets into the network. From Fig. 13(a), one can see that when the network size is 16×16, the PER of the block coding scheme is 30% lower than that of the baseline scheme due to the error resilience, and the effective throughput is 33% higher than that of the baseline as shown in Fig. 13(b). To increase the risk posed by malicious nodes, in another experiment, malicious nodes can add randomized delays while forwarding packets from the output port of the router. As shown in Fig. 13(c), these malicious nodes have little impact on the transmission of the IPD channel, since delayed forwarding mostly adds extra delays to the packets, and has little impact on the time intervals between packets. These results have confirmed the robustness of the block coding scheme even with the presence of the malicious node.

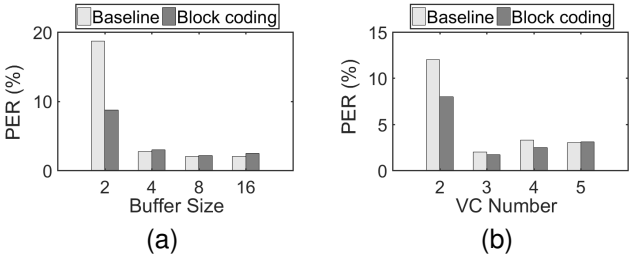


Fig. 14. The PER with different (a) buffer sizes and (b) virtual channel numbers of the two methods.

We measure the impacts of the network parameters (input buffer size and virtual channel number) on the performance of the baseline IPD secure channel and the block coding scheme. Fig. 14(a) shows that when the buffer size is 2, the PER of the block coding scheme is almost half of that of the baseline IPD channel. When the buffer size continues to increase to 4, 8, 16, the PER of the two methods stays flat. Similar to Fig. 14(a), Fig. 14(b) shows that when the number of virtual channels exceeds 3, the PER of the two methods is almost unchanged. A smaller number of input buffers and virtual channels leads to network congestions more easily, causing deviations from the set time intervals, thereby increasing the PER.

2) Experiments with real benchmarks: In this set of experiments, we take the traces generated from the real applications as the noise to evaluate the PER and the detection accuracy. The malicious node only detects but does not flood useless packets. Fig. 15(a) and 15(b) show the results with the benchmarks' traces collected from gpgpu-sim as the noise.

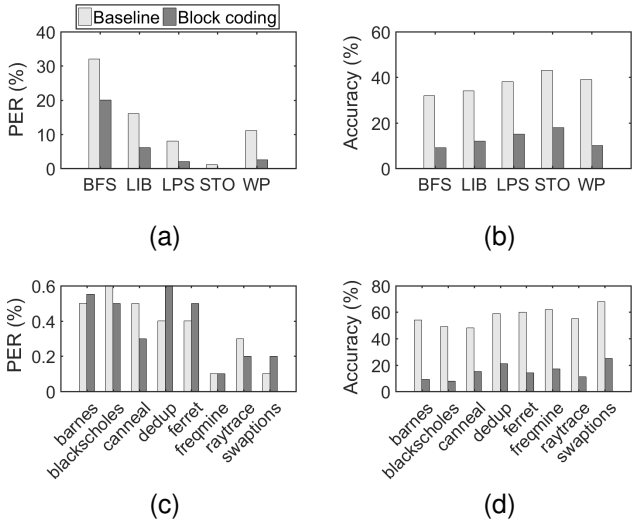


Fig. 15. (a) and (c) are PERs with different noises (real applications from the ispass and PARSEC benchmarks), (b) and (d) are detection accuracy comparison of the two methods.

Fig. 15(a) shows that, compared with the baseline scheme, the block coding scheme reduces average PER by 15%. Fig. 15(b) shows that the detection accuracy of the malicious node against the block coding scheme is reduced by 20% on average. Fig. 15(c) and 15(d) show the results with the benchmark traces collected from the many-core simulator taken as the noise. Fig. 15(c) shows there is a minor difference between the PER of the baseline scheme and that of the block coding scheme, but the average detection accuracy against the block coding drops by nearly 50%, as shown in Fig.15(d), and the low detection accuracy of the malicious node indicates that most sensitive information is successfully transmitted. The reason is that the packet injection rate of most cases of PARSEC is lower than 0.01 packets per cycle per router, resulting in a very low PER. Putting things together, one can see that the proposed method is applicable to most traffic scenarios.

To find the worst case of the proposed block coding based secure channel, Fig. 16 shows the PER of the channel with respect to the noise traffics' PIR. The PER of the block coding scheme increases with the increase in PIR. When the PIR is 0.1 packets per cycle per router, the PER of the block coding scheme is close to 100%, whereas the average latency of the network is over 1000 cycles. In this case, it is safe to say that the network is already congested, which indicates that the proposed method does not work under this extremely congested network workload. As described in Section III-D, to tackle such scenarios, we need to wait until the network becomes less congested to handle reasonable amount of data traffic flow.

To demonstrate the scalability of the IPD channel, we further performed experiments on a chiplet-based system [41].

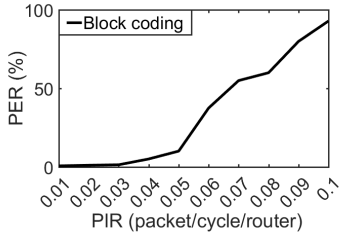


Fig. 16. The PER of the block coding scheme by varying PIR of other traffics (noise).

TABLE II
MULTI-CHIPLET NETWORK CONFIGURATION

Configurations of the inter- and intra-chiplet networks	
Intra-chiplet network topology	2×2 mesh
Inter-chiplet network topology	8×8 mesh
Intra-chiplet latency	router 2 cycle, link 1 cycle
Inter-chiplet latency	link latency model [43], PHY latency model [42]
Virtual channel number	4
Input/output buffer size	4
Routing algorithm	XY routing for inter- and intra-chiplet networks
Traffic pattern	uniform

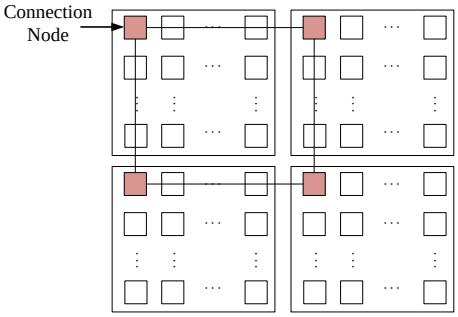


Fig. 17. The network topology of the chiplet-based architecture.

The chiplet-based system features an inter-chiplet topology shown in Fig. 17, and its inter-chiplet network is a 2×2 mesh. The network configurations are listed in Table II. Both the intra- and inter-chiplet networks use XY routing, whereas, in the inter-chiplet network, each chiplet is treated as a meta-node in routing. The upper left node in each chiplet connects to the neighboring chiplets and the PHY latency is 2 ns as reported in [42], and the interposer wire delay model is adopted from [43]. The floorplanning and size of each node in the chiplets are adopted from [47]. Fig. 18 shows that when PIR is higher than 0.004 packets per cycle per router, the PER of the block coding scheme decreases by about 16% compared to that of the baseline scheme. The effective throughput of the block coding scheme on average is 24% higher than that of the baseline scheme.

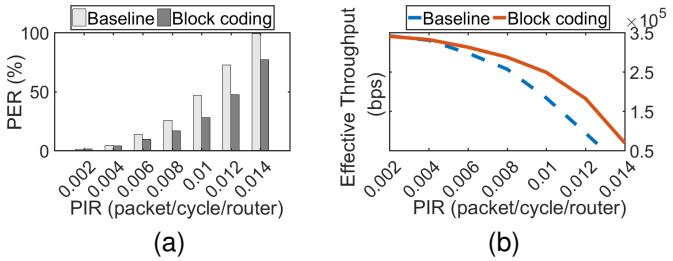


Fig. 18. (a) PER and (b) effective throughput by varying PIR of other traffics (noise) in the chiplet-based system.

D. Overhead of the Proposed Method

The transmitter and receiver are programs running in different cores. If there is no normal data packet to be transmitted to the receiver, the transmitter needs to send dummy packets to hide information. To reduce unnecessary overhead, the proposed method is more applicable to scenes with moderate or heavy traffic. For the baseline scheme, the IPD coding overhead of the transmitter and decoding overhead of the receiver are about 11 and 9 cycles per covert bit, respectively.

TABLE III
PERFORMANCE COMPARISON OF DIFFERENT ENCODINGS

Method	PER	Effective throughput ($\times 10^5$bps)
Huffman code	44.6%	1.91
Golay code	5.8%	1.26
The proposed method	7.1%	2.38

There is additional overhead associated with coding and decoding the $nBmT$ block codes. This overhead is fairly modest. In our experiments, it takes 29 cycles, which is broken down to 16 cycles of the code-book checkup time and 13 cycles for code conversion.

The energy consumptions of sending and receiving 1000 bits are 1.92 mJ and 1.67 mJ, respectively, as reported by the GPUWattch [26] power simulator. Compared to running the benchmark applications, *e.g.*, the 32×32 matrix multiplication, the energy overhead is as small as only 0.16%.

E. Comparison with Other Encoding Algorithms

We have included a few experiments comparing the proposed method with two more encoding methods, Huffman code [13] and Golay code [7]. The ternary Golay code (11, 6) is used, and each code contains 6 information bits. As shown in Table. III, the PER of the IPD channel encoded by Huffman code is 37% higher that of the proposed method. Although Huffman code reduces the number of bits in the code group, its length varies and it does not have error correction mechanism. The PER of Golay code is almost equal to that of the proposed method, but 5 check bits have to be added to each code group, which effectively cuts the actual throughput by half.

F. Comparison with Other Secured Transmission Methods

In this set of experiments, the proposed block coding based IPD channel is compared against several methods for secure data transmission, including (1) thermal covert channel (TCC) [20], [35]–[37], (2) cache covert channel [30]–[32], and (3) hardware-based encryption [33].

Experiments of TCC were performed on the many-core simulator and Hotspot version 6.0 [40], which is used as the temperature simulator, and the configurations are listed in Table IV. TCC works by heat transfer, therefore, in our experimental setting, it only works within 2 hops. That is, the distance between the transmitter and the receiver of the TCC should be within 2 hops. The core and cache parameters are set to be the same as the many-core simulator configuration in Table I. As shown in Fig. 19(a), the PER of the TCC is 1.5 times higher than the proposed IPD channel on average when the PIR is 0.06 packets per cycle per router. Fig. 19(b) shows

TABLE IV
TCC PARAMETERS

Configurations of TCC	
Chip thickness	0.00015m
Specific heat capacity	$1.75 \times 10^6 J/(m^3 \cdot K)$
Silicon thermal conductivity	$100\ W/(m \cdot K)$
Temperature threshold for DTM	373.15K
Heat sink side	0.06m
Heat sink thickness	0.0069m
Heat sink thermal conductivity	$400\ W/(m \cdot K)$
Specific heat capacity of heat sink	$3.55{\times}10^6 J/(m^3 \cdot K)$
Thermal sensor resolution	0.1°C

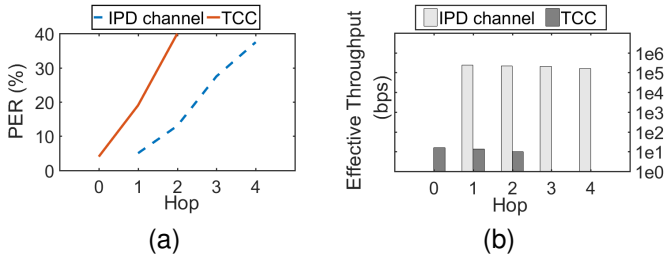


Fig. 19. (a) PERs and (b) effective throughputs of TCC and the proposed block coding based IPD channel with respect to the distance between the transmitter and the receiver. TCC works within 2 hops.

that the effective throughput of the TCC is 0.006% of the proposed IPD channel. Therefore, the proposed IPD channel has a lower error rate and higher transmission throughput than the TCC.

The cache covert channel [31] transmits sensitive information by encoding the bits with the latencies of accessing last level cache (LLC), *i.e.*, the latency difference in cache miss and hit, which is implemented in our many-core simulator. The configurations of the simulator are detailed in Table I. Note that due to the NoC transmission delay, the throughput and PER of the cache covert channel in the many-core system are no longer the same as reported in [31] which has no NoC transmission delay. Fig. 20 shows that with different network sizes, the PER of the cache covert channel on average is 1.43 times higher than the proposed IPD channel, and the effective throughput of the proposed IPD channel is on average 78% higher than that of the cache covert channel.

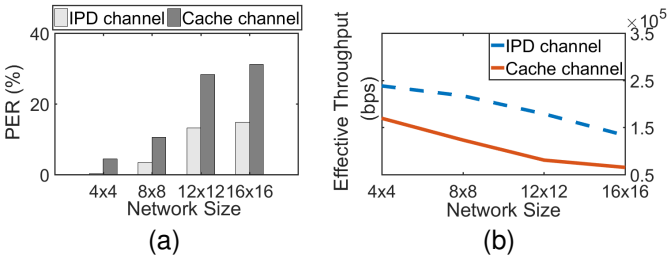


Fig. 20. (a) PERs and (b) effective throughputs of the cache covert channel and the proposed block coding based IPD channel with different network sizes.

Compared with the proposed block coding based IPD channel, the hardware-based encryption scheme (*e.g.*, AES) encrypts data before secure transmission, which can achieve a PER of 0.03% [51], but it requires additional hardware circuit for AES coding and decoding. As reported from [33], the AES encoder circuit using 28nm TSMC technology has an area of 0.0028 mm^2 and consumes 48 mJ when encrypting 1000 bits. As a comparison, the block coding based IPD channel requires no additional circuit, and its energy consumption is only 7.5% of the encryption scheme.

G. Evaluation of the IPD Channel on an FPGA-based Multi-core System

We evaluated the block coding based IPD channel on a 4×4 multi-core system on Xilinx vcu113p FPGA. The multi-core system uses PULPino [39] as the processor cores, and OpenPiton [38] as the uncore (NoC and memory *etc.*). The configurations of OpenPiton are detailed in Table V. We

TABLE V
OPENPITON'S PARAMETERS

Number of cores	16
L1 D-cache	8KB
L1 I-cache	16KB
L1.5 cache	8KB
TLBs (number of entries)	16
Bootloading	SD/SDHC Card, UART, JTAG
On-chip network	4×4 mesh
Input buffer size	4
Routing algorithm	XY routing

inject random traffics as noise. Extensive experiments on this platform revealed that the PER of the block coding based IPD channel is 1.38% and the effective throughput of the block coding based IPD channel reaches 1.44×10^5 bps on average.

VI. CONCLUSION

In this paper, an IPD-based channel was proposed for secure transmission over an untrusted NoC. Essentially, the bits of sensitive information are represented by different time intervals between successive packets. In addition, to enhance confidentiality and reduce PER, the block coding scheme was also proposed to achieve PER as low as 0.3% and effective throughput of 2.3×10^5 bps. Experimental results showed that the application of block coding reduces the PER by as much as 30%, and it is less detectable than the baseline IPD channel, particularly true when the network size is large. The proposed IPD channel has lower PER and overhead, and higher throughput than competing thermal covert channel, cache covert channel, and circuit-based encryption scheme, making it suitable for secure data transmission over unsecured NoC-based systems.

REFERENCES

[1] Sajeesh K, Kapoor H K, "An authenticated encryption based security framework for NoC architectures," in *Proc. Int'l Symp. Electronic System Design*, 2011, pp.134-139.

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

[3] Wang Y, Suh G E, "Efficient timing channel protection for on-chip networks," in *Proc. IEEE/ACM Int'l Symp. Networks-on-Chip*, 2012, pp. 142-151.

[4] Boraten T, DiTomaso D, Kodi A K, "Secure model checkers for network-on-chip (NoC) architectures," in *Proc. Int'l Great Lakes Symp. VLSI*, 2016, pp. 45-50.

[5] Biswas A K, Ghosal D, Nagaraja S, "A survey of timing channels and countermeasures," *ACM Computing Surveys*, vol. 50, no. 1, 2017, pp. 1-39.

[6] Berk V, Giani A, Cybenko G, "Detection of covert channel encoding in network packet delays," *Technical Report*. TR2005-536, 2005.

[7] Houmansadr A, Borisov N, "CoCo: coding-based covert timing channels for network flows," in *Proc. Int'l Workshop on Information Hiding*, 2011, pp. 314-328.

[8] Sun Y, Guan X, Liu T, *et al*., "An identity authentication mechanism based on timing covert channel," in *Proc. IEEE Int'l Conf. Trust, Security and Privacy in Computing and Communications*, 2012, pp. 832-836.

[9] Rout S S, Basu K, Deb S, "Efficient post-silicon validation of network-on-chip using wireless links," in *Proc. Int'l Conf. VLSI Design and Int'l Conf. Embedded Systems*, 2019, pp: 371-376.

[10] Cabuk S, Brodley C E, Shields C, "IP covert timing channels: design and detection," in *Proc. ACM Conf. Computer and Communications Security*, 2004, pp. 178-187.

[11] Sellke S H, Wang C C, Bagchi S, *et al*., "TCP/IP timing channels: theory to implementation," in *Proc. Int'l Conf. Computer Communications*, 2009, pp. 2204-2212.

[12] Zhang X, Tan Y A, Liang C, *et al*., "A covert channel over VoLTE via adjusting silence periods," *IEEE Access*, vol. 6, 2018, pp. 9292-9302.

[13] Wu J, Wang Y, Ding L, *et al*., "Improving performance of network covert timing channel through Huffman coding," *Mathematical and Computer Modelling*, vol. 55, no. 1-2, 2012, pp. 69-79.

[14] Weber I, Marchezan G, Caimi L, *et al*., "Open-source NoC-based many-core for evaluating hardware trojan detection methods," in *Proc. IEEE Int'l Symp. Circuits and Systems*, 2020, pp. 1-5.

[15] Liu Y, Ghosal D, Armknecht F, *et al*., "Robust and undetectable steganographic timing channels for iid traffic," *Int'l Workshop on Information Hiding*, 2010, pp. 193-207.

[16] Ziemer R E, Tranter W H. "Principles of communications," *John Wiley & Sons*, 2014.

[17] Liu W, Liu G, Zhai J, *et al*., "Designing analog fountain timing channels: undetectability, robustness, and model-adaptation," *IEEE Trans. Information Forensics and Security*, vol. 11, no. 4, 2016, pp. 677-690.

[18] Popnet, 2015. [Online].
Available: https://github.com/karellincoln/popnet.git

[19] Mazda, Fraidoon, "Telecommunications engineer's reference book," *Butterworth-Heinemann*, 1993.

[20] Tian S, Szefer J, "Temporal thermal covert channels in cloud FPGAs," in *Proc. Int'l Symp. Field-Programmable Gate Arrays*, 2019, pp. 298-303.

[21] Osman O, Uçan O N. "Contemporary coding techniques and applications for mobile communications," *CRC Press*, 2009.

[22] Intel. Intel single-chip cloud computer product overview. Available: http://techresearch.intel.com/spaw2/uploads/files/SCC-Overview.pdf.

[23] Wentzlaff D, Griffin P, Hoffmann H, *et al.*, "On-chip interconnection architecture of the tile processor," *IEEE micro*, vol. 27, no. 5, 2007, pp. 15-31.

[24] Banakar R, Steinke S, Lee B S, *et al*., "Scratchpad memory: a design alternative for cache on-chip memory in embedded systems," in *Proc. Int'l Symp. Hardware/Software Codesign*, 2002, pp. 73-78.

[25] Diguet J P, Evain S, Vaslin R, *et al*., "NoC-centric security of reconfigurable SoC," in *Proc Int'l Symp. Networks-on-Chip*, 2007, pp. 223-232.

[26] Leng J, Hetherington T, ElTantawy A, *et al*., "GPUWattch: enabling energy optimizations in GPGPUs", in *Int'l Symp. Computer Architecture*, 2013, pp. 487-498.

[27] Bagrodia R, Meyer R, Takai M, *et al*. "PARSEC: a parallel simulation environment for complex systems," *Computer*, vol. 31, no. 10, 1998, pp. 77-85.

[28] Wang X, Yang M, Jiang Y, *et al*. "On self-tuning networks-on-chip for dynamic network-flow dominance adaptation," *ACM Trans. Embedded Computing Systems*, 2014, vol. 13, pp. 1-21.

[29] Duato J, Yalamanchili S, Ni L. "Interconnect networks: an engineering approach," *IEEE Comp. Soc. Press*, 1998.

[30] Maurice C, Neumann C, Heen O, *et al*. "C5: cross-cores cache covert channel," in *Int'l Conf. Detection of Intrusions and Malware, and Vulnerability Assessment*, 2015, pp. 46-64.

[31] Saileshwar G, Fletcher C W, Qureshi M. "Streamline: a fast, flushless cache covert-channel attack by enabling asynchronous collusion," in *Proc. Int'l Conf. Architectural Support for Programming Languages and Operating Systems*, 2021, pp. 1077-1090.

[32] Xu Y, Bailey M, Jahanian F, *et al*. "An exploration of L2 cache covert channels in virtualized environments," in *Proc. ACM Workshop on Cloud Computing Security Workshop*, 2011, pp. 29-40.

[33] Lu M, Fan A, Xu J, *et al*. "A compact, lightweight and low-cost 8-bit datapath AES circuit for IoT applications in 28nm CMOS," in *Int'l Conf. Trust, Security and Privacy in Computing and Communications/ Int'l Conf. Big Data Science and Engineering*, 2018, pp. 1464-1469.

[34] Bakhoda A, Yuan G L, Fung W L, *et al*. "Analyzing CUDA workloads using a detailed GPU simulator," in *Proc. Int'l Symp. Performance Analysis of Systems and Software*, 2009, pp. 163-174.

[35] Masti R J, Rai D, Ranganathan A, *et al*. "Thermal covert channels on multi-core platforms," in *USENIX Security Symp.*, 2015, pp. 865-880.

[36] Bartolini D B, Miedl P, Thiele L. "On the capacity of thermal covert channels in multicores," in *Proc. Conf. Computer Systems*, 2016, pp. 1-16.

[37] Iakymchuk T, Nikodem M, Kepa K. "Temperature-based covert channel in FPGA systems," in *Int'l Workshop on Reconfigurable Communication-centric Systems-on-Chip*, 2011, pp. 1-7.

[38] Balkind J, McKeown M, Fu Y, *et al*. "OpenPiton: an open source manycore research framework," *ACM SIGPLAN Notices*, vol. 51, no. 4, 2016, pp. 217-232.
Available: https://github.com/PrincetonUniversity/openpiton

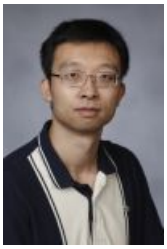
[39] Traber A, Zaruba F, Stucki S, *et al*. "PULPino: a small single-core RISC-V SoC," in *RISCV Workshop*, 2016.
Available: https://github.com/pulp-platform/pulpino

[40] Huang W, Ghosh S, Velusamy S, *et al*. "HotSpot: a compact thermal modeling methodology for early-stage VLSI design," *IEEE Trans. Very Large Scale Integration (VLSI) Systems*, vol. 14, no. 5, 2006, pp. 501-513.

[41] Bharadwaj S, Yin J, Beckmann B, *et al*. "Kite: a family of heterogeneous interposer topologies enabled via accurate interconnect modeling," in *Proc. ACM/IEEE Design Automation Conf.*, 2020, pp. 1-6.
[42] Universal chiplet interconnect express (UCIe), Available: https://www.uciexpress.org/general-8
[43] Kabir M D A, Peng Y. "Chiplet-package co-design for 2.5 D systems using standard ASIC CAD tools," in *Proc. Asia and South Pacific Design Automation Conf.*, 2020, pp. 351-356.
[44] Bhunia S, Hsiao M S, Banga M, *et al.* "Hardware Trojan attacks: threat analysis and countermeasures," in *Proc. IEEE*, 2014, vol. 102, no. 8, pp. 1229-1247.
[45] Oliveira B, Reusch R, Medina H, *et al.* "Evaluating the cost to cipher the NoC communication," in *Proc. Latin American Symp. Circuits & Systems*, 2018, pp. 1-4.
[46] Reinbrecht C, Susin A, Bossuet L, *et al*. "Side channel attack on NoC-based MPSoCs are practical: NoC Prime+ Probe attack," in *Proc. Symp. Integrated Circuits and Systems Design*, 2016, pp. 1-6.
[47] Wang X H, Liu P, Yang M, *et al*. "Energy efficient run-time incremental mapping for 3-D networks-on-chip," *J. Computer Science and Technology*, vol. 28, no. 1, 2013, pp. 54-71.
[48] Wang Y, Xu Q, Qu G, *et al.* "Information hiding behind approximate computation," in *Proc. Great Lakes Symp. VLSI*, 2019, pp. 405-410.
[49] Rajendran J, Dhandayuthapany A M, Vedula V, et al. "Formal security verification of third party intellectual property cores for information leakage," in *Proc. Int'l VLSI Design and Int'l Conf. Embedded Systems*, 2016, pp. 547-552.
[50] Zhang L, Wang X, Jiang Y, et al. "Effectiveness of HT-assisted sinkhole and blackhole denial of service attacks targeting mesh networks-on-chip," *J. Systems Architecture*, 2018, vol. 89, pp. 84-94.
[51] Jeon S, Choi J P. "CFB-AES-turbo: joint encryption and channel coding for secure satellite data transmission," in *Proc. Int'l Conf. Communications*, 2019, pp. 1-7.

Jiaen Xu received the bachelor's degree in Hunan Normal University, China. She is working toward the master's degree in the school of software engineering, South China University of Technology. Her research interests include hardware security, and covert channel.

Xiaohang Wang received the B. Eng. and Ph. D. degree in communication and electronic engineering from Zhejiang University, in 2006 and 2011, respectively. He is currently a Professor at South China University of Technology. He was the receipt of PDP 2015 and VLSI-SoC 2014 Best Paper Awards. He was the special session organizer of NoCS 2018, steering committee member of NoCArc 2014-2018, and TPC chair of ICCS 2021. He also served as the guest editor of the Mathematics, Integration, the VLSI Journal, Microelectronics Journal, and Computers and Electrical Engineering. His research interests include many-core architecture, power efficient architectures, optimal control, and NoC-based systems.



Yingtao Jiang received his Ph. D. in Computer Science from the University of Texas at Dallas in 2001, and joined the Department of Electrical and Computer Engineering (ECE), University of Nevada, Las Vegas (UNLV) as an assistant professor in the same year. He was promoted to the rank of full professor at the same department in 2013. He served as the ECE department chair between 2015 and 2018, and he is currently associate dean of UNLVs college of engineering. Besides STEM education, his research interests span a wide array of areas, including VLSI integrated circuit design, computer architectures, wireless networks, machine learning, cloud computing, biomedical engineering, and nanotechnologies.

Amit Kumar Singh is an Associate Professor at University of Essex, UK. He received the B.Tech. degree in Electronics Engineering from Indian Institute of Technology (Indian School of Mines), Dhanbad, India, in 2006, and the Ph.D. degree from the School of Computer Engineering, Nanyang Technological University (NTU), Singapore, in 2013. He has published over 110 papers in reputed journals/conferences, and received several best paper awards, e.g., IEEE TC February 2018 Featured Paper, ICCES 2017, ISORC 2016, PDP 2015, HiPEAC 2013 and GLSVLSI 2014 runner up. He is associate editor of IEEE Embedded Systems Letters, Design Automation for Embedded Systems, Journal of Low Power Electronics and Applications, and Frontiers in Neuroscience. He also edited a book for JLPEA journal and currently editing a special issue for JLPEA. He served as the publication chair of ESWeek-2021, Publicity co-chair of CF-2021, Local Co-chair of NASA/ESA Conference on Adaptive Hardware and Systems 2019, organized a special session at ESWeek-2021 and a tutorial at ESWeek-2018. He has served on the TPC of IEEE/ACM conferences like DAC, DATE, ICCAD, CASES and CODES+ISSS.

Chongyan Gu received the Ph.D. degree from Queen's University Belfast, Belfast, U.K., in 2016. She is currently a Lecturer in the School of EEECS at Queen's University Belfast, and a member of Center for Secure Information Technologies (CSIT) with in the Institute of Electronics Communications and Information Technologies (ECIT). She was invited to give tutorial/talks to international conferences, such as, IEEE ASP-DAC 2020 on the topic of practical PUF design on FPGA. Her current research interests include PUFs, security in/for approximate computing, true random number generator (TRNGs), hardware Trojan detection and machine learning attacks.

Letian Huang received the MS and Ph. D. degrees in communication and information system from the University of Electronic Science and Technology of China (UESTC), Chengdu, China in 2009 and 2016, respectively. He is an associate professor with UESTC. His scientific work contains more than 40 publications including book chapters, journal articles and conference papers. His research interests include heterogeneous multi-core system-on-chips, network-on-chips, and mixed signal IC design.

Mei Yang received her Ph. D. in Computer Science from the University of Texas at Dallas in Aug. 2003. In Aug. 2004, she joined in the Department of Electrical and Computer Engineering, University of Nevada, Las Vegas, where she was promoted to full professor in 2016. Her research interests include computer architectures, interconnection networks, machine learning, and embedded systems.

Shunbin Li received the PhD degree in information and communication engineering from Zhejiang University, Hangzhou, China, in 2018. He is currently working as an associate research fellow in Zhejiang laboratory. His research interests include VLSI circuits, security, and reconfigurable computing.