A Survey of Timing Channels and Countermeasures

Arnab Kumar Biswas, Indian Institute of Science, Bangalore, India.
Dipak Ghosal, University of California at Davis, CA, USA.
Shishir Nagaraja, Lancaster University, Lancaster, UK.

A timing channel is a communication channel that can transfer information to a receiver/decoder by modulating the timing behavior of an entity. Examples of this entity include the inter-packet delays of a packet stream, the re-ordering packets in a packet stream or the resource access time of a cryptographic module. Advances in the information and coding theory and the availability of high performance computing systems interconnected by high speed networks, have spurred interest in and development of various types of timing channels. With the emergence of complex timing channels, novel detection and prevention techniques are also being developed to counter them. In this paper we provide a detailed survey of timing channels broadly categorized into network timing channel in which communicating entities are connected by a network and in-system timing channel in which the communicating entities are within a computing system. This survey builds on the last comprehensive survey by [Zander et al. 2007] and considers all the three canonical applications of timing channels namely, covert communication, timing side channel, and network flow watermarking. We survey the theoretical foundations, the implementation, and the various detection and prevention techniques that have been reported in literature. Based on the analysis of current literature, we discuss potential future research directions both in design and application of timing channels and their detection and prevention techniques.

CCS Concepts:•**General and reference** → **Surveys and overviews;**•**Networks** → *Network security;*

Additional Key Words and Phrases: Timing Channel, Covert Timing Channel Communication, Timing Side Channel, Network Flow Watermarking, Timing Channel Categories, Timing Channel Countermeasures

ACM Reference Format:
Arnab Kumar Biswas, Dipak Ghosal, and Shishir Nagaraja, 2016. A Survey of Timing Channels and Countermeasures. *ACM Comput. Surv.* 0, 0, Article 0 (December 2016), 39 pages.
DOI: http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

According to [Department of Defense Standard 1985], a covert channel is defined as a "communication channel that allows a process to transfer information in a manner that violates the system's security policy." Here the term system may refer to a single computer system or a distributed system connected by a network. The concept of covert channels was first published in [Lampson 1973] with a characterization of the confinement problem and methods to block subtle information leakage paths. While the work was primarily conceptual in nature, the underlying concept has since grown in importance. Significant developments in information theory, coding theory, and the

This work was supported by the PhD scholarship from the MHRD, Government of India and US NSF grant CNS-101886.
Author's addresses: Arnab Kumar Biswas, Department of Electronic Systems Engineering, Indian Institute of Science, Bangalore 560012, India; email: akbiswas@dese.iisc.ernet.in; Dipak Ghosal, Department of Computer Science, University of California at Davis, CA 95616, USA; email: dghosal@ucdavis.edu; Shishir Nagaraja, School of Computing and Communications, Lancaster University, Bailrigg, Lancaster LA1 4YW, United Kingdom; email: s.nagaraja@lancaster.ac.uk.

DOI: http://dx.doi.org/10.1145/0000000.0000000

emergence of high performance systems interconnected by high-speed networks have enabled covert channels in general, and timing channels in particular, to evolve from a conceptual idea into a potentially useful and practical tool.

In this paper, we present a survey of timing channels and their countermeasures. A timing channel is defined as a channel that can transfer information based on the modulation of some timing behavior of an entity. An example of this entity is the inter-packet delay (IPD) of a packet stream generated by a distributed application. Specifically, the transfer of information is achieved by modulating the IPDs to carry the information bits. Other examples of timing channel entities include the packet ordering in network and the resource access time of a cryptographic module. In literature, the term covert timing channel (CTC) is used to refer to a timing channel that transfers covert messages over a network. In this paper, we use the term covert timing channel communication (CTCC) in place of CTC to emphasize its use as a communication channel to transfer messages. The term network flow watermarking (NFW) is used to refer to a timing channel that can mark network flows, while the term timing side channel (TSC) is used to refer to a timing channel that leaks information due to faulty/vulnerable operations. In this survey, we consider NFW and TSC along with CTCC as types of timing channels.

Timing channels have both legitimate and malicious applications. An example of a malicious application of CTCCs is its use by criminals to ex-filtrate secret information from an enterprise. Whereas, as an example of legitimate application, a network administrator can use CTCCs to hide network management related communication or to transfer authentication data. Further details of various CTCC applications (both malicious and legitimate) are given in [Zander et al. 2007; Archibald 2013]. An important application of NFW is to detect the stepping stones which refer to the compromised nodes that are used as intermediate nodes in a network to launch an attack. NFW can also be used to de-anonymize an anonymous network [Dingledine et al. 2004]. These applications can be considered malicious or legitimate depending on the actual application scenario. TSCs are mainly used in malicious applications e.g., to extract secret cryptographic keys.

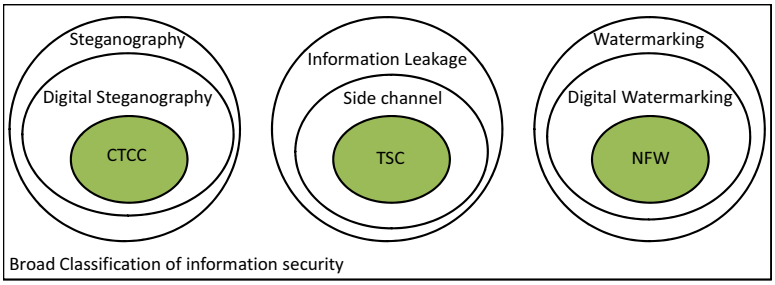
An earlier comprehensive survey on network covert channels including the CTCCs is given in [Zander et al. 2007]. The survey covered the period from 1987 through 2006 and reviewed all types of available implementations of network covert channels and their mitigation techniques. Two relatively new surveys on CTCC are reported in [Jaskolka et al. 2012] and [Goher et al. 2012]. Both these surveys are short and are limited to existence conditions for CTCC in [Jaskolka et al. 2012] and detection techniques in [Goher et al. 2012].

In this paper we build on [Zander et al. 2007] and review the literature on design, implementation, and detection/prevention techniques of timing channels since 2006. We consider the CTCC as only one type of timing channel and unlike in [Zander et al. 2007] include the review of the NFW and the TSC in this survey. We first categorize timing channels based on the location i.e., network timing channel and in-system timing channel. For network timing channel, the communication channel is established between systems that are connected by a LAN or a WAN. A timing channel established within a computer system is referred to as an in-system timing channel. In Section 3, we discuss network timing channel and review both CTCC and remote timing side channel (RTSC). While NFW is a type of network timing channel, we review it separately in Section 4. In-system timing channels are reviewed in Section 5. For in-system timing channel, both CTCC and TSC are relevant and we review their implementations, and detection/mitigation techniques. Due to space constraints, we do not review various programming language and Operating System (OS) related or Virtual Machine and

cloud related timing channels in this survey. We conclude and suggest future research directions in Section 6.

2. PRELIMINARIES

According to [Department of Defense Standard 1985], a covert timing channel is defined as "a covert channel in which one process signals information to another by modulating its own use of system resources (e.g., CPU time) in such a way that this manipulation affects the real response time observed by the second process." To the best of our knowledge, the first timing channel was reported in [Bell and Padula 1976]. It was an interval based timing channel where a long interval between two events indicated bit 1 and a short interval indicated bit 0. While the term timing channel was not used, it was described as an indirect communication path meant to disclose sensitive information.



CTCC: Covert timing channel communication, TSC: Timing side channel, NFW: Network flow watermarking

Fig. 1. Different applications of timing channels in the world of information security.

From a broad information security perspective, timing channels have been applied in steganography (information hiding), information leakage, and watermarking. As shown in Figure 1, CTCC is a type of digital steganography where a sender sends covert messages using a timing channel. In the context of information leakage, TSC is a type of side channel where information is leaked due to faulty or vulnerable operation of a system or by a timing attack. Finally, in the domain of digital watermarking, NFW can be implemented using a timing channel. Whereas for both CTCC and NFW there is a sender (embedder) and a receiver (decoder), in case of TSC a sender may not be present.

2.1. Key Entities and Definitions

Simmons presented CTCC as prisoners' problem in [1984]. It was proposed to resemble authentication without secrecy and can be described as follows. Alice and Bob are two prisoners who want to flee the prison. Wendy is a warden who suspects that the two prisoners are planning to flee but she does not have any proof. To get the proof, Wendy allows Alice and Bob to communicate with a requirement that the messages must be fully open to her i.e., an overt channel. It is Alice and Bob's goal to communicate secret messages hidden in the innocuous communication in a way that Wendy cannot detect the hidden secret messages.

We use the term *overt channel* when we refer to a communication channel where information transmitted between sender and receiver is accessible without barriers to any person-in-the-middle (warden). By a *covert channel* we refer to the hiding of information in the communication channel (for instance within network protocols). The information thus transmitted via a covert channel is termed as *hidden information*. Figure 2 shows the key entities of a CTCC. Only logical (not physical) locations of

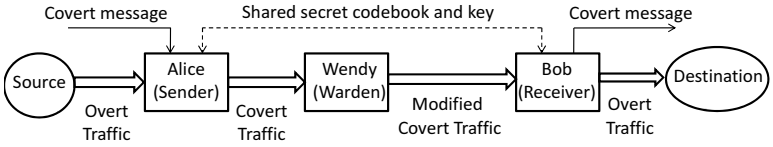


Fig. 2. A generic model of a covert timing channel showing the various entities. This figure is adapted from the figure in [Zander et al. 2007] where it is used to represent covert channels including CTCC.

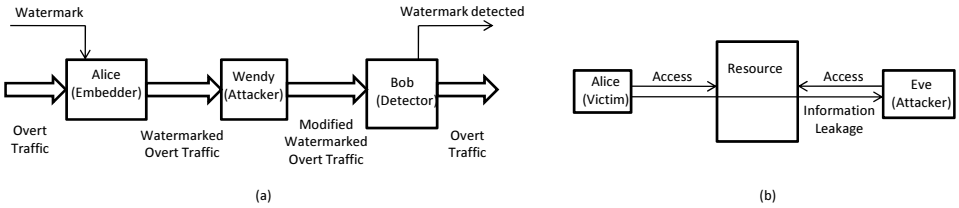


Fig. 3. The key entities in a) network flow watermarking (NFW) and b) timing side channel (TSC).

(covert) sender and (covert) receiver are shown with respect to the (overt) source and the (overt) destination. Moreover, a CTCC can be active or passive [Gianvecchio and Wang 2007]. In active CTCC, sender and source are same and hence the sender can generate overt traffic as per requirement. In passive CTCC, they are different and the source generates overt traffic independent of the sender's requirement. A passive CTCC typically has lower capacity than an active CTCC but is harder to detect. A warden can be located anywhere between source and destination i.e., co-located with the source, on a router in the path from the source to the destination, or at the destination.

Figure 3(a) shows the various entities of NFW. Here Alice represents the embedder that embeds watermarks into the overt traffic and Bob represents the watermark detector that decodes the presence of watermarks from the network traffic. The attacker Wendy attempts to thwart the ability of Bob to accurately detect the watermark and trace it back to Alice.

Figure 3(b) shows the key entities of TSC. Here Alice (victim) unintentionally leaks information to Eve (attacker). Alice accesses the resource for her own operation based on some secret information. The same resource is also accessed by Eve causing resource contention. Now Eve can interpret Alice's timing behavior by observing variation of her own timing characteristics. This leads to the leakage of Alice's secret information with high probability. In some cases (like in chosen plaintext attack) an input is also applied to the victim.

2.2. Timing Channel Requirements and Threat models

The basic requirements of a timing channel are **non-detectability** and **non-disruptability**. Non-detectability[1] implies that the warden cannot identify the existence of a timing channel. In the context of a CTCC, a timing channel is termed Polynomial non-detectable regarding a security variable δ provided that there exists a negligible function $\nu(\delta)$ such that $|T(d) - T(d_s)| \leq \nu(\delta)$ for some probabilistic polynomial-time statistical test T [Liu et al. 2010; C. 2004], where d and d_s are arbitrary N samples of the timing feature of the overt and covert traffic, respectively. Polynomial time statistical tests include Kullback-Leibler (KL) divergence test [C. 2004; Archibald and Ghosal 2014], Kolmogorov-Smirnov (KS) test [Cabuk et al. 2009] among others.

[1] In literature the terms invisibility and undetectability are also used. In this paper we use the term non-detectability to mean the same.

Another closely related property to non-detectability is **non-disclosure** that ensures a warden cannot decode the secret information if the timing channel is detected. A timing channel's non-detectability guarantees its non-disclosure[2]. Cryptographic encoding of covert message can enhance non-disclosure property.

A timing channel is exposed to both intentional and inherent channel noise. **Non-disruptability** refers to the property of correctly receiving the covert message at the receiver in spite of the presence of channel noise. In literature, robustness is also used to refer to the same term. In [Liu et al. 2010], the ability to achieve a bit error rate $P_e \leq \varepsilon$ is used to measure the robustness subject to a robustness condition $\varepsilon \in \mathbb{R}^+$. The Signal-to-Noise Ratio and P_e are inversely proportional to each other.

2.2.1. Threat models. The objective of the threat model is to clarify the capabilities of the warden. Inspired by attacks on encryption schemes, we consider the following.

(1) A passive *output-only* warden tries to detect a timing channel by only observing the channel output. This is the weakest threat model and any timing channel implementation vulnerable to this type of attack is completely insecure. In this scenario, the warden is unable or unwilling to exert any other control. Such an assumption may be unrealistic as a warden may have access to additional information about the communication channel.
(2) A *known-input* warden is aware of channel input and the corresponding channel output but does not control what goes into channel input. Such a warden observes arbitrary flows both before and after the encoding process and tries to detect the timing channel apart from the channel noise. Such attacks are proposed in [Lin and Hopper 2012].
(3) A *chosen-input* warden chooses the channel input and observes the corresponding output. The study in [Lin and Hopper 2012] first proposed these attacks in the context of watermarking.
(4) An *adaptive chosen-input* warden is a chosen-input warden wherin the choice of channel input may depend on the channel output received previously. Such attacks have not been proposed so far.
(5) A *chosen-output* warden is given the channel input corresponding to an observed output. The warden gains access to the decoder (but not any keys) and is able to convince the decoder to decode a given encoder output, any number of times. Having this capability already means the warden can detect whether a channel output has covert messages. However, the objective is to then uncover any keys used within the timing channel implementation so that the warden can then encode arbitrary messages to confuse the recipient. So far, no such attacks have been proposed.
(6) An *adaptive chosen-output* warden is a chosen-output warden where the choice of outputs the warden asks to be decoded are dependent on the results of previous requests to the decoder. Such attacks have not been proposed so far.

While all of the above mentioned threat models are applicable to NFW, only a subset may be applicable to CTCCs. In case of TSC, some of the threat models are used by attacker to extract secret information from victim's timing behavior. Knowledge of the threat models can help to devise more effective detection and prevention techniques.

[2]In literature the term non-decodability is also used. In this paper we use the term non-disclosure to mean the same.

3. NETWORK TIMING CHANNELS

Network timing channel can be of three types: 1) CTCC, 2) RTSC, and 3) NFW. In this section, we review various CTCC related theoretical works, implementation techniques, and detection and prevention methods. We also discuss various studies related to RTSC. NFW is discussed in Section 4.

3.1. Covert timing channel communication

In literature the term steganographic timing channel is also used in place of covert timing channel. From our perspective they are same because of their identical goal, characteristics, and implementation techniques. In this section steganographic timing channel and CTCC are used interchangeably. The goal of CTCC is to transfer messages between the sender and receiver without being detected by the warden. Consequently, both non-detectability and non-disruptability are important requirements. Additionally, high channel capacity (secret information transfer rate) is required for CTCC which depends on the overt traffic and the use of appropriate coding techniques that mitigate the impact of the channel noise.

3.1.1. Theoretical studies. There is a rich history of theoretical studies on the capacity and non-detectability (secrecy) of CTCC. One of the earliest studies on the capacity of a CTCC over a noisy channel based on Shannons information theory was reported in [Moskowitz and Miller 1992]. It is noted in [Anantharam and Verdu 1996] that the channel capacity of a queue having exponentially distributed service time is $e^{-1}\mu$ nats/sec and it has the minimum channel capacity considering all servers having service rate μ packets/sec. Various aspects of Exponential Service Timing Channels (ESTC) are discussed and studied in detail in [Anantharam and Verdu 1996; Sundaresan and Verdu 2000a; 2000b; Giles and Hajek 2002; Wagner and Anantharam 2005; Momcilovic 2006; Sundaresan and Verdu 2006] with the discrete-time counterpart present in [Bedekar and Azizoglu 1998; Thomas 1997]. Lower bounds of capacity for single-server timing channel are estimated in [Sellke et al. 2006] with the service time distributions of Uniform, Gaussian, and truncated Gaussian. Sellke et al. have given two lower bounds and one upper bound on the capacity of Bounded Service Timing Channels (BSTC) [2007]. BSTC has bounded support service time distributions which implies that the service times follow $P(a < S_k < a + \Delta) = 1$ where $a, \Delta > 0$ are some constants and service time S_k is independent and identically distributed (iid) random variable [Sellke et al. 2007]. It is shown that the uniform BSTC has the lowest capacity among different kinds of BSTCs with small support interval. This parallels the unbounded service time case in which the exponential service time has the lowest capacity as shown in [Anantharam and Verdu 1996]. Gorantla et al. have given an upper bound of the CTCC capacity from a high security level user (HU) to a low security level user (LU) through the full buffer NRL pump [2010; 2012].

The maximum information transmission rate in an on/off CTCC where information is transfered by transmitting or not transmitting a packet within a pre-determined time period is proposed by Yao et al. in [2009]. The authors have divided the on/off CTCC into two types based on the characteristics of the IPD, namely, deterministic CTCC and non-deterministic CTCC. They have mainly analyzed the non-deterministic on/off CTCC considering stable or slowly varying network delay characteristic. Zi et al. have proposed a procedure to determine the maximum transmission rate of an IPD based CTCC without actually implementing it [2011]. The method requires the jitter characteristic in the network which is described by a group of probability distributions. Additional information on the packet loss probability and the maximum sending rate without any congestion or packet loss are also required. In their scheme, the overt and covert senders are same i.e., the scheme is applicable to active CTCCs only.

A CTCC is implemented in [Ezzeddine and Moulin 2009] using queue-based codes and Shannon's encoding functions. Sparse graph coset codes over non-binary finite fields can be used to approach Shannon capacity for queuing timing channels [Coleman and Kiyavash 2008a; 2008b]. Authors in [Wang and Moulin 2008] have studied the theoretical limits of steganographic channel capacity and the required coding structure to achieve those limits. They have also noted that the perfect security requirement of steganographic timing channel is that the probability distribution of stegotext must be same as legitimate traffic distribution. Works on achievable secrecy rate and coding scheme to achieve maximum information rate are available. In particular, Dunn et al. have derived achievable secrecy rate of IPD based CTCC involving parallel queues [2009]. They have also given the necessary and sufficient condition to achieve a certain secrecy rate for a deterministic encoder.

3.1.2. Implementations. CTCC implementations can be of two types: IPD based and packet reordering based. IPD based CTCC is implemented by modulating the inter-packet delay of overt traffic. Both active and passive CTCC can be implemented using IPD modulation. On the other hand, packet reordering based CTCC is implemented by encoding the covert messages in the ordering of packets in a single flow or multiple flows. In case of multiple flows, the set of flows themselves also can be used to encode messages. Only active CTCC can be implemented using this type because passive CTCC will require a lot of buffer space to hold packets from one or multiple flows.

A) IPD based CTCC. Probably the first published work on CTCC in a LAN is [Girling 1987]. The author has analyzed the capacity of CTCC and has provided methods to reduce the bandwidth of CTCC. Different requirements of CTCC i.e., non-detectability, non-disclosure, robustness, and high capacity are not considered in this early study. A TCP-based CTCC called TCPScript proposed in [Luo et al. 2008] embeds covert messages into the TCP burst size (i.e., number of packets in a burst). In order to ensure non-detectability, TCP's normal burstiness patterns are maintained. While authors have given results for capacity as well for robustness against packet loss, packet reordering, and traffic shaping, no solution is given to improve them. Another IPD based CTCC over TCP/IP connection is proposed in [Sellke et al. 2009] with a non-detectable version with low capacity and another version without the guarantee on non-detectability but with high capacity. Kiyavash et al. have proposed an IPD based CTCC targeting interactive SSH traffic which is modeled using two-state Markov Modulated Poisson Process (MMPP) [2009]. The study addresses only the non-detectability requirement which is guaranteed for statistical tests that consider an MMPP model of the overt traffic.

A time-replay CTCC is proposed in [Cabuk 2006] where covert messages are transmitted by replaying a previously recorded sequence of timing intervals. They have partitioned the prerecorded sequence and the number of partitions are equal to the size of the message alphabet. Each partition is then associated with a symbol. A timing interval from a partition is randomly chosen to send the associated symbol. An example of time-replay CTCC is given in Appendix A. While non-detectability and non-disclosure requirements are considered, robustness and high capacity requirements are not addressed in the study.

In order to improve the robustness of the CTCC many studies have employed various types of coding scheme. Liu et al. have proposed an IPD based CTCC using spreading codes [2009; 2010]. The use of spreading code makes the CTCC robust but with low capacity. Furthermore, the proposed approach is only applicable to iid IPDs. In CoCo [Houmansadr and Borisov 2011a], different error correcting codes are used to improve robustness of the scheme. Apart from robustness, other requirements i.e., non-detectability, non-disclosure, and high capacity are also considered in the work.

Huffman coding to compress the CTCC data is proposed in [Wu et al. 2012]. Though experimental results of non-detectability, and capacity are given, robustness is not considered in their work. Stillman et al. have proposed to use a concatenated code (convolutional outer code over an inner copy code) for a CTCC implementation through Mix-firewalls [2008].

A model-based CTCC to mimic the statistical properties of legitimate traffic is proposed in [Gianvecchio et al. 2008]. The proposed framework consisting of filter, analyzer, encoder, and transmitter is discussed in more detail in Appendix B. Though they have considered non-detectability and capacity requirements, robustness requirement is not considered. Liu et al. have proposed a CTCC with distribution matching [2009; 2012] in which the overt traffic is divided into fixed-length fragments and all IPDs in a fragment are used to derive the IPD histogram. This histogram distribution is matched after the covert messages are encoded into IPDs. Authors have considered all requirements of an efficient CTCC. Model-based CTCC with polynomial-time non-detectability and high capacity is proposed in [Ahmadzadeh and Agnew 2013]. They have used trellis structure at the modulation stage of the transmitter and at the iterative demodulation stage of the receiver. They have also used an adaptive modulation scheme to improve the channel robustness without any loss of non-detectability. Archibald et al. have proposed a model based CTCC using Fountain codes [2012; 2013]. Whereas the model based CTCC helps to achieve non-detectability requirement, Fountain codes generate encoded symbols continuously until the receiver successfully receives the message. The receiver needs to signal the completion to the sender using a reverse channel. Model based CTCCs typically consider iid IPDs. It is shown that these CTCCs can be detected if the real IPDs are not iid [Zander et al. 2011]. Zander et al. have proposed a non-detectable CTCC using a portion of each IPD to embed a covert message. This causes the CTCC to be highly sensitive to channel noise and hence lack robustness.

A Phase Locked Loop (PLL) based synchronization scheme for CTCC implementation is proposed in [Chen et al. 2011]. With respect to robustness, the decoding rate of the scheme is shown to be high despite packet loss and out-of-order packet arrivals. Lee et al. have proposed a CTCC in the physical layer using sub-microsecond modulation [2014]. They have shown that their CTCC named Chupja is non-detectable and robust with high capacity. CTCC implementation targeting a particular application is also available in literature. Sun et al. have proposed a CTCC based identity authentication mechanism [2012]. They have used overt IPDs to encode the authentication tags: long IPD as bit 1 and short IPD as bit 0. They have only tried to achieve the non-detectability requirement of the CTCC. Zi et al. have proposed a passive CTCC where the covert sender dwells in an intermediate node (e.g., router, gateway) and modulates incoming IPDs to embed covert messages [2010]. They have also proposed to use synchronization and error correction techniques based on frames for covert communication. They have only considered the robustness requirement in their study.

Various detection resistant CTCCs are also proposed in literature. IPDs of active CTCC exhibit statistical regularity since they are generated using a specific IPD distribution. To avoid detection because of regularity, Kothari et al. have proposed an active irregular CTCC called Mimic [2013]. They have used two modules called “shape modeler” and “regularity modeler” to learn about shape and regularity properties of legitimate traffic and to generate Mimic traffic. Walls et al. have proposed a CTCC called Liquid that is resistant to entropy detection tests [2011]. They have proposed to use a number of IPDs to smooth out the shape distortions apart from using some IPDs to embed covert messages. Both detection resistant studies consider only non-detectability and capacity requirements. Different CTCC detection and prevention techniques are discussed later. A summary of the literature on IPD based CTCC is given in Table I.

Table I. A summary of the literature on IPD based CTCC. The letters C and I indicate behavioral **c**haracteristic and **i**mplementation detail respectively.

Main attributes		**IPD based CTCC works**
Non-detectability	General (C)	[Liu et al. 2010]
	Model based (C)	[Liu et al. 2009; Gianvecchio et al. 2008] [Liu et al. 2009; Liu et al. 2012] [Ahmadzadeh and Agnew 2013; Zander et al. 2011] [Archibald and Ghosal 2012; Archibald 2013]
	Time replay (C)	[Cabuk 2006]
	Physical layer (I)	[Lee et al. 2014]
	Resistant to Regularity test (I)	[Kothari and Wright 2013]
	Resistant to Entropy test (I)	[Walls et al. 2011]
Robustness (use of Error Correction Code) (I)		[Houmansadr and Borisov 2011a] [Stillman 2008; Liu et al. 2009]
Channel capacity	Coding scheme (C)	[Coleman and Kiyavash 2008a; 2008b; Wu et al. 2012] [Wang and Moulin 2008; Ezzeddine and Moulin 2009]
	Theoretical bounds (C)	[Sellke et al. 2007] [Gorantla et al. 2010; Gorantla et al. 2012]
	Maximum rate (C)	[Yao et al. 2009; Zi et al. 2011]
Secrecy rate (C)		[Dunn et al. 2009]
CTCC general model (C)		[Shrestha et al. 2013; Shrestha et al. 2016]
Protocol specific (e.g., TCP, SSH) CTCC (I)		[Girling 1987; Luo et al. 2008] [Sellke et al. 2009; Kiyavash and Coleman 2009]
PLL based synchronization (I)		[Chen et al. 2011]
Identity authentication (I)		[Sun et al. 2012]
Passive CTCC (I)		[Zi et al. 2010]

Here the main attributes column gives an attribute for each IPD based CTCC work that is the focus of that work. All the works whose main focus is same, are grouped together in the same row corresponding to their main attribute. The sub-rows to the channel capacity and non-detectability, show more specific attributes under these two attributes. The main attributes are mainly of two types i.e., behavioral characteristic and implementation detail, which are indicated in each row after every attribute.

B) Packet reordering based CTCC:. Ahsan et al. have proposed a CTCC based on packet ordering within the IPSec framework [2002]. The different requirements of an efficient CTCC are not considered in this early paper. Another packet reordering based CTCC is proposed in [El-Atawy and Al-Shaer 2009] using specific permutations of consecutive packets to increase robustness. Additionally the CTCC imitates real traffic distribution to increase non-detectability. They have also considered the high capacity requirement of CTCC. Another similar work reported in [Chakinala et al. 2007] have proposed polynomial time optimal encoding and decoding algorithms to achieve maximum channel capacity. They have proposed formal models for packet re-ordering channels based on an information-theoretic game between the covert sender and the adversary (jamming process) and have proved the existence of a Nash equilibrium for the mutual information rate. In this theoretical work no solution is proposed to improve different requirements (except capacity) of a practical CTCC.

Packet reordering among multiple flows can also be used to implement a CTCC. In [Luo et al. 2007; Luo et al. 2012a], authors have proposed a CTCC called Cloak by sending N packets in every round over X flows to transfer a message. They have proposed ten encoding and decoding schemes using different combinations of N and X. Their proposal is to exploit TCP's reliable transmission mechanism to provide robustness against packet losses. They have also considered non-detectability and high capacity requirements of CTCC. Inter-socket packet arrival order is also used to build a CTCC. In [Khan et al. 2009], authors have proposed a high capacity CTCC based

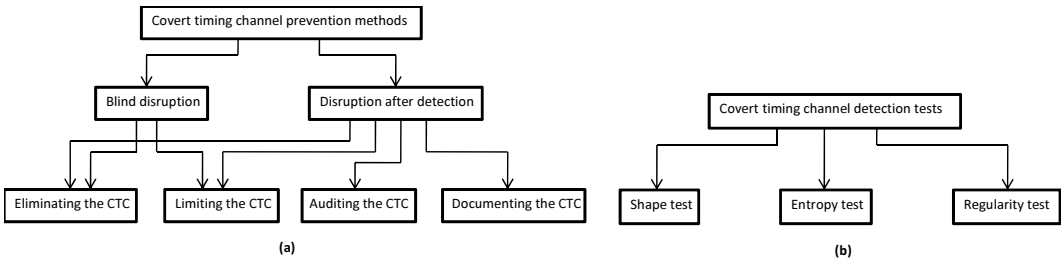


Fig. 4. Different types of (a) CTCC prevention methods, and (b) CTCC detection tests.

on inter-socket packet arrival order using multiple active connections. Their proposed CTCC is shown to be non-detectable by regularity based tests. The reason is that the covert message is embedded in the order and sequence of different connections and not in the IPDs of a connection. Robustness requirement is not considered in the study.

C) CTCC implementations in wireless network:. CTCCs are implemented not only in wired networks but in other types of networks also. For example, CTCC can be implemented in wireless network [Kiyavash et al. 2013; Radhakrishnan et al. 2013; Yue et al. 2014]. Kiyavash et al. have implemented an IPD based CTCC in carrier sense multiple access with collision avoidance (CSMA/CA) protocol with a triggering mechanism [2013]. They have proposed to build spyware circuit components within original circuit components during synthesis so that the spyware components cannot be distinctly located. Radhakrishnan et al. have implemented a CTCC on IEEE 802.11 network exploiting the random back-off in distributed coordination function (DCF) [2013] calling it Covert-DCF. They have used off-the-shelf wireless cards to implement their proposed technique. Wireless CTCC is also implemented by varying CPU speed in mobile devices exploiting Android OS's performance governor [Yue et al. 2014]. Yue et al. have demonstrated their technique (1) across existing network services, (2) across ICMP pings, and (3) via a trojanized chat application. There is a legitimate application using wireless CTCC reported in [Edwards et al. 2012]. They have proposed to use CTCC to detect wormhole attacks in Mobile ad hoc networks (MANETs). It is shown that the CTCC capacity is reduced under wormhole attack. They have used error correcting codes and have used the number of errors corrected to detect the presence of wormhole attack.

3.1.3. Detection and Prevention. There are two types of CTCC prevention methods - blind disruption and disruption after detection [Archibald 2013; Archibald and Ghosal 2014]. Blind disruption means disrupting (eliminating or limiting) the possible presence of CTCC without detecting it. According to [Zander et al. 2007] there are mainly four ways to counter CTCC after detecting it - 1) eliminating, 2) limiting, 3) auditing, and 4) documenting the CTCC. Different CTCC prevention methods are shown in Figure 4(a). Authors in [Archibald 2013; Archibald and Ghosal 2014] have categorized the CTCC detection tests into three categories as shown in Figure 4(b): 1) shape test like Kolmogorov Smirnov (KS) test, 2) entropy test like Kullbacke Leibler (KL) divergence test, and simple entropy test, and 3) regularity test like auto-correlation test. A test may belong to more than one category like corrected conditional entropy (CCE) test which measures both the entropy and the regularity.

Archibald et al. have compared three CTCC detection test performances against three types of CTCCs: 1) IPD based CTCC, 2) model-based CTCC, and 3) time-replay CTCC [2013; 2014]. They have also proposed a shape test called Welch's t-test and multi-feature Support Vector Machine (SVM) classifier for increasing the classification

rate. Mou et al. have also proposed a CTCC detection method using SVM and wavelet transform [2012]. They have also proposed a detection and classification method using a sliding window to detect different CTCCs in a single traffic. Gianvecchio et al. have suggested that the change of entropy in a flow due to CTCC can be used to detect a CTCC [2007; 2011]. They have proposed an entropy test to detect CTCC by observing abnormal shape. Mason et al. have presented a CTCC detection mechanism using entropy measurements on a Massively Parallel Processing Architecture (MPPA) called Ambric [2010]. Cabuk et al. have shown that IPDs in CTCC are more regular than legitimate channel [2006; 2009]. They have proposed to use compressibility to detect CTCC by detecting regularity in traffic. Details of compressibility metric are given in Appendix C.

There are few CTCC blind disruption techniques reported in literature. Wang et al. have proposed a traffic controller to control IPD based CTCC by randomly delaying traffic [2009]. Their method does not detect the presence of any CTCC. Insertion of delays causes performance degradation of all traffic including legitimate ones. Another timing mitigator to limit the CTCC capacity is proposed in [Askarov et al. 2010]. They have also shown the trade-off between the system performance and the achievable CTCC limit.

IPD based CTCC and packet reordering based CTCC can exist in a virtual private network (VPN) [Sadeghi et al. 2012]. To mitigate IPD based CTCC, [Sadeghi et al. 2012] have proposed to implement traffic reshaping inside the Linux kernel. They have implemented IPsec anti-replay window as a packet buffer and packets are sorted using their Encapsulated Security Payload (ESP) sequence numbers. This helps to mitigate packet reordering based CTCC. Another work on CTCC mitigation in case of VPN is [Herzberg and Shulman 2013]. Here authors have assumed that man-in-the-end (MitE) malware runs within both ends of protected network and a man-in-the-middle (MitM) attacker is located on the path of communication. They have suggested to use traffic shaper similar to [Sadeghi et al. 2012] to modify the IPDs of sender's packets. They have also suggested to use error correcting codes in the sender to limit packet reordering based CTCC from MitM to MitE. The receiver can recover original packets removing most if not all MitM covert messages.

Luo et al. have proposed methods to detect a TCP based CTCC called TCPScript [2008]. The first method is based on the TCP burst size because in TCPScript the burst size is dependent on the covert message. The second method is based on the delay between an arriving ACK packet and the next departing packet. This is because a TCPScript encoder sends two TCP bursts with a separation that is much higher than a normal inter-TCP burst separation. They have also proposed a third method that is a combination of the two earlier methods.

According to [Stillman 2008], bits that are being exfiltrated from a system using CTCC likely reside in memory at some point during the transmission and any correlation between memory content and IPDs suggests a CTCC. The memory must contain a portion of covert message before the covert encoder encodes them in IPDs. They have proposed to decode the possible covert bit strings from IPDs which are then matched against the memory content. They have shown that their detection method can detect IPD based CTCC even in the presence of noise. A summary of different CTCC detection and prevention studies is given in Table II.

3.2. Remote timing side channel

Remote timing side channel (RTSC) is a type of TSC attack launched in a network. Here the adversary tries to exploit a vulnerability of a cryptographic implementation remotely. In TSC, non-detectability and non-disclosure ensure that a warden cannot detect a TSC's presence and decode the leaked secret information, respectively. The

Table II. A summary of the literature on CTCC detection and prevention methods.

Main attributes	**CTCC detection and prevention works**
Comparison between detection tests	[Archibald 2013; Archibald and Ghosal 2014]
Use of wavelet transform and SVM	[Mou et al. 2012]
Entropy test	[Gianvecchio and Wang 2007; 2011; Mason et al. 2010]
Regularity test	[Cabuk 2006; Cabuk et al. 2009]
Blind disruption	[Wang et al. 2009; Askarov et al. 2010]
Targeting VPN based CTCC	[Sadeghi et al. 2012; Herzberg and Shulman 2013]
Targeting TCP based CTCC	[Luo et al. 2008]
Memory correlation based detection	[Stillman 2008]

observer cannot improve these properties because the secret information is leaked by a faulty operation. High TSC capacity is hard to obtain because a large number of repeated observations are required to reduce the error probability to improve robustness.

Two weaknesses in SSH is demonstrated in [Song et al. 2001] through which an adversary can extract passwords remotely during an SSH session. First, the original data size can be approximately revealed by the eight-byte boundary of transmitted packets. Second, the inter-keystroke timing information is leaked as every keystroke information is sent to the remote machine during the interactive mode. They have shown that significant information about users' typing can be obtained by using some advanced statistical techniques. Key sequences can be predicted from the inter-keystroke timings by Hidden Markov Model and their key sequence prediction algorithm.

There are studies on RTSC attacks against OpenSSL. In OpenSSL, square-and-multiply algorithm is optimized using Sliding Window Exponentiation technique. In this technique a block of bits (window) of the private decryption exponent are considered in every iteration compared to one bit in normal square-and-multiply algorithm. A multiplication table is usually pre-computed for sliding window technique. Brumley et al. have shown that the extraction of the private keys is possible from a local network web server (based on OpenSSL) [2003; 2005]. Authors in [Aciiçmez et al. 2005] have proposed an attack that improves the efficiency of the attack in [Brumley and Boneh 2003] by an order of magnitude. They have exploited the Montgomery multiplications' timing behavior during the initialization of table. This allows them to get one of the prime factors of RSA moduli by increasing the number of multiplications. Another RTSC attack against OpenSSL is shown in [Brumley and Tuveri 2011; B. Brumley 2011]. They have used a vulnerability in OpenSSL to recover the private key of a TLS server. The vulnerability is in the Montgomerys ladder implementation in elliptic curve cryptosystem (ECC). They have used messages, signatures, and timing of those messages to launch the attack.

A RTSC attack on AES based on cache use, is reported in [Aciiçmez et al. 2007] to get the remote cryptosystem's secret keys. The requirement is that the victim server must be a multitasking or simultaneous multithreading system running large workload. Crosby et al. have discussed the limits of RTSC attacks by measuring network response times and jitter. They have conducted their research both on the Internet and a local network [2009]. They have proposed filters to minimize the effects of jitter and increase the timing measurement accuracy across the network. Gong et al. have shown a RTSC attack based on a scheduler between two users [2013]. Utilizing Shannon equivocation as a privacy metric, they have proved that one user can learn the complete traffic pattern of the other user if the scheduler employs a first come first serve (FCFS) policy. Moreover, they have shown the feasibility of a RTSC attack exploiting the TSC inside a home digital subscriber line (DSL) router. Using the attack an attacker can get victim's passwords, and voice over IP (VoIP) conversations. A summary of RTSC studies is shown in Table III.

Table III. A summary of literature on RTSC.

Main attributes	Remote timing side channel works
SSH based attack	[Song et al. 2001]
OpenSSL based attack	[Brumley and Boneh 2003; 2005; Aciiçmez et al. 2005] [Brumley and Tuveri 2011; B. Brumley 2011]
Cache based attack on AES	[Aciiçmez et al. 2007]
Limits of attack	[Crosby et al. 2009]
Scheduler based attack	[Gong and Kiyavash 2013]

3.3. Summary

IPD based CTCC is the main focus of most of the theoretical studies and also most of the implementation studies on CTCC. Studies on packet reordering based CTCC and RTSC are few in number. Maximum transmission rate using non-deterministic on/off timing channel has been reported in [Yao et al. 2009]. However, results are obtained under ideal conditions. In particular the study doesn't take into account the security of the channel or the synchronization cost. Consequently, the real achievable transmission rate is likely to be less than reported. There is a scope of further work to determine the maximum transmission rate by taking various channel factors into account. A method to calculate the transmission rate of an IPD based active CTCC is reported in [Zi et al. 2011]. An important research direction is to extend this for passive CTCC. This includes incorporating a buffer to account for transient mismatches between the rate at which the overt source generates packets and the rate at which they are serviced by the covert sender. Additionally, the impact of any interfering traffic must be considered to ensure robustness of the passive CTCC. A simple error correction approach is proposed in [Zi et al. 2010] to implement a robust passive CTCC. As error correction schemes add to the overhead, further research is required to explore different robust communication paradigms that can provide tradeoff between robustness and capacity particularly for passive CTCCs. An encoding scheme for IPD based CTCC is proposed in [Zander et al. 2011] to increase the robustness. Future work is required to analyze the effects of different network conditions including packet loss over CTCC.

A preliminary mathematical model of CTCC operation is reported in [Shrestha et al. 2013; Shrestha et al. 2016]. Further work is possible that can provide mathematical models for detecting covert channels. Although, literature on packet reordering based CTCC are few, the growth of distributed network applications that exploit multiple flows will enable packet reordering based CTCC such as those proposed in [Ahsan and Kundur 2002]. A future research direction is to develop a formal theoretical framework for packet reordering based CTCC. Such a framework could incorporate various coding strategies that take into account the real distributed applications and network impairments. Another possible future research can be to dynamically adjust CTCC parameters to adapt to network changes. The initialization process can be equipped with learning capabilities to know the host connection properties.

Spyware communication channel that exploits the CSMA/CA medium access control protocol is proposed in [Kiyavash et al. 2013]. A possible future work is to design and implement methods to detect the start of covert communication or the "trigger" time. A wireless communication, both in WiFi and LTE based cellular networks, have many timing aspects such as rate adaptation and sleeping mode. Detailed research is required to examine if spyware can be enhanced using these timing features. A covert timing channel for IEEE 802.11 networks is reported in [Radhakrishnan et al. 2013]. Impact on the BER due to change in the number of surrounding wireless users and sending rate is a possible future work. Additionally, detailed research is required to study the CTCC performance in different sender-receiver scenarios like multiple-sender single-receiver and multiple-sender multiple-receiver.

A model-based covert timing channel framework is presented in [Gianvecchio et al. 2008]. Detection of model-based CTCCs using different goodness-of-fit tests like the Cramer-Von Mises and Anderson-Darling tests, is a possible future research direction. While these tests are less general than the Kolmogorov-Smirnov test, they maybe more effective in detecting CTCC in certain types of traffic.

4. NETWORK FLOW WATERMARKING

NFW is an emerging field where secret information (i.e., a watermark) is embedded in a flow by modulating its timing characteristics. While conceptually it is very similar to CTCC, an important difference is that in NFW non-disruptability is considered as the main requirement whereas in CTCC non-detectability is the main requirement. NFW is an emerging technique to link flows in different parts of the network to carry out traffic analysis applied to network anonymity and privacy [Gong et al. 2013]. Stepping stones refer to compromised hosts that are exploited as intermediate nodes to launch various types of attacks. By matching flows, NFW can be used to detect stepping stones. Another important application of NFW is to de-anonymize an anonymous network like Tor [Dingledine et al. 2004]. VoIP calls using an anonymous network can also be tracked using NFW. The purpose of an anonymous network is to map incoming flows randomly to outgoing flows. NFW can correlate an outgoing flow with the corresponding incoming flow (and vice versa) and attack an anonymous network. So, depending on the application scenario, the role of different entities in NFW as shown in Figure 3(a) changes (from protector to attacker).

In applications that rely only on detecting a multi-bit watermark, the output of detector is only one bit of information denoting the presence or absence of the watermark. However there are applications where multiple flows and their sources have to be detected distinctly resulting in multiple output bits of information. Recently this scenario is termed as flow fingerprinting [Houmansadr and Borisov 2013b; Elices and Perez-Gonzalez 2013]. Nevertheless, flow fingerprinting is a special case of NFW; in both cases the watermarking technique used are very similar. It is to be noted that capacity is not an important requirement for NFW as only a small watermark is embedded. Non-disclosure is also not relevant because once the watermark's presence is detected, it can be removed or replicated. Disclosing the watermark will not do any additional harm. Non-disclosure is relevant in flow fingerprinting as the information can be used to distinguish each marked flow.

To the best of our knowledge, the first study on NFW was published in [Wang and Reeves 2003] and since then there has been a lot of research activity in this area. NFW techniques can be divided into two categories: blind and hybrid. While NFW is an active approach, there exist non-blind or passive flow analysis techniques (e.g., [Zhang and Paxson 2000; Donoho et al. 2002]) to correlate traffic flows. These techniques may use different flow timing characteristics like IPD to analyze flows but they do not modify the traffic flow (by embedding a watermark) secretly.

4.1. Blind NFW

Unlike passive or non-blind flow analysis techniques, in blind NFW, the detector does not require the original incoming flow to correlate with an outgoing flow. All the information required to detect the watermark are embedded in the flow itself. There are three types of blind NFW techniques - 1) interval based, 2) IPD based, and 3) Direct Sequence Spread Spectrum (DSSS) based.

4.1.1. Interval based NFW. In this type of NFW, a packet flow is divided into equal length intervals. Then some transformations are done to the packets in each selected interval to embed the watermark bits. Wang et al. have presented an Interval Centroid

Based Watermarking (ICBW) scheme [2007]. Their scheme is shown to be resilient to various flow transformations like flow merging/splitting and packet dropping. Common flow transformation techniques are discussed in Appendix D and centroid of an interval is defined in Appendix E.

In the context of NFW, various detection mechanisms are known as attacks or threats to watermarking. So non-detectable watermarks are called attack resistant watermarks. One such attack is Multi-flow Attack (MFA) and few MFA resistant interval based watermarking schemes are available in literature. Houmansadr et al. have presented a Multi-flow Attack Resistant Interval Centroid Based Watermarking (MAR-ICBW) method [2009a]. MAR-ICBW removes correlations between flows by embedding the watermarks on arbitrary locations over multiple flows. Another MFA resistant scheme called Scalable Watermark Invisible and Resilient to packet Losses (SWIRL) is proposed in [Houmansadr and Borisov 2011b]. In this scheme each flow is marked with a different pattern to resist multi-flow attack. We will discuss more about Multi-flow Attack in Section 4.5.1.

Interval Based Watermarking (IBW) method [Pyun et al. 2007; Pyun et al. 2012] utilizes the passed time of the flows to trace the traffic during repacketization, timing perturbation, chaff packets, and flow splitting. Every flow is divided in small intervals and packet timing is modified to control the packet number in particular intervals. Clock synchronization among the watermark decoder and encoder is not required. A variation of IBW called BotMosaic is proposed by [Houmansadr and Borisov 2013a] to counter Internet Relay Chat (IRC)-based botnets. The watermark encoder places a specific pattern in the flow that will be identified by client organizations. The watermark is collaboratively inserted into multiple flows at once to enable easy detection.

A sequential watermark detection model (SWDM) is proposed in [Wang et al. 2012; Wang et al. 2013]. Three sequential detectors are also proposed to traceback network attack flows using the proposed model SWDM. The authors have proposed the "single-interval-based optimum watermark detector" (SOWD) and the "paired-intervals-based optimum watermark detector" (POWD) assuming known parameters of the perceived flow. They have also proposed the sequential sign watermark detector (SSWD) for non-parametric watermark detection.

4.1.2. IPD based NFW. In this type of NFW, IPD values are first quantized and then some transformation is applied to embed watermark bits. An IPD based watermarking method that adjusts the timing of selected packets is proposed by [Wang and Reeves 2003; 2011]. They have quantified the trade-offs between the attainable correlation and the timing perturbation's attributes. They have also derived an upper bound on the number of packets to achieve an effective correlation. A watermarking-based host correlation detection method is proposed in [Pan et al. 2009]. Authors have divided the IPDs into two different groups randomly, and have used the average value of the IPDs in different groups to embed the watermark. An IPD based watermarking technique is also used to correlate encrypted, peer-to-peer VoIP calls passing through low latency anonymizing networks [Wang et al. 2005]. In this scheme timing of selected packets are adjusted to embed a watermark into the encrypted VoIP flow. Neither robustness nor non-detectability of watermark is shown to be satisfied in their work.

An NFW method that inserts watermarks in IPDs using quantization-index modulation is proposed in [Gong et al. 2012; 2013]. They have proposed to use error-correcting codes and a maximum likelihood decoding (ML) method to deal with the watermark desynchronization and substitution errors.

4.1.3. DSSS based NFW. In this type of NFW, a Pseudo-Noise (PN) code is used to implement the Direct Sequence Spread Spectrum (DSSS). Secret spread spectrum signal is embedded in the sender's traffic following some method based on the specific scheme.

Yu et al. have proposed a DSSS based watermarking technique where the traffic rate is varied to embed a hidden spread spectrum signal in the sender's traffic [2007]. Detailed description of the proposed system is given in Appendix F. This system does not require any offline training for detection. However, there is a limitation that the target flow must have a fixed traffic rate which may not be true in practical networks. The study has not considered the requirements of non-detectability and robustness of the watermark. It is to be noted that mere secrecy of the PN codes is not sufficient to satisfy the non-detectability of the watermark. Zhang et al. have improved the scheme proposed in [Yu et al. 2007] by modulating the statistical characteristics of arrival times of packets instead of traffic rate [2009]. They have also not considered the non-detectability and robustness requirements. Another DSSS based watermarking technique by varying sender's traffic rate is proposed in [Huang et al. 2011]. Their scheme uses long PN code to track anonymous flows over an anonymous network. They have used a short portion of a long PN code to modulate each signal bit to provide resistance to Mean-Square Autocorrelation (MSAC) based detection.

MSAC and Multi-flow Attacks Resistant Spread Spectrum Watermarking (MMAR-SSW) technique is proposed in [Zhang et al. 2010b] for tracing multiple network flows. The technique embeds watermark bits in randomly selected interval positions using multiple orthogonal PN codes. We will discuss more about MSAC attack in Section4.5.1.

A) DSSS and other type combined. Apart from the above mentioned DSSS techniques, there are some hybrid techniques that use DSSS in combination with some other technique to embed watermarks into flows. Interval Centroid Based Watermarking (ICBW) technique is combined with Spread Spectrum (SS) technique to obtain the "Interval Centroid Based Spread Spectrum Watermarking" scheme (ICBSSW) [Wang et al. 2009; Luo et al. 2012b]. Authors have used this technique to track multiple flows simultaneously. They have used many PN codes to randomize the inserted watermark location over many flows. A "Double Interval Centroid-Based Watermark" (DICBW) scheme is proposed in [Wang et al. 2010] for network flow traceback. DICBW considers every pair of adjacent intervals and collaboratively modulates their packet timing. They have also formed a hybrid watermarking scheme combining DICBW with SS scheme.

Zhang et al. have proposed an "Interval-Based Spread Spectrum Watermarking" (IBSSW) scheme by joining Interval-Based Watermarking (IBW) and SS schemes to trace multiple network flows [2010a]. Their scheme modulates the packet arrival time and uses multiple orthogonal PN codes to randomize the inserted watermark locations over many traffic flows. The main attributes of different blind NFW studies is shown in Table IV.

4.2. Hybrid NFW

As mentioned before, we characterize a passive flow analysis as non-blind because this technique does not have any similarity with blind watermarking. There are only a few self-described non-blind watermarking schemes. These so-called non-blind techniques actually consist of various characteristics of blind watermarking (like active NFW) and passive flow analysis. It is our view that these techniques should be called hybrid NFW. They require the original input flow characteristics to be shared with/sent to all the detectors similar to passive or non-blind techniques.

The method proposed in [Peng et al. 2005] uses packet matching and timing-based active watermarking to correlate stepping stone connections. They have proposed different algorithms with different computation cost, false positive rate, and detection rate. It is a hybrid scheme because the detector needs all the input flow information.

Table IV. A summary of the studies related to blind NFW.

Type	Main attributes	Blind flow watermarking works
Interval based	Interval Centroid Based	[Wang et al. 2007; Houmansadr et al. 2009a] [Houmansadr and Borisov 2011b]
	Interval Based	[Pyun et al. 2007; Pyun et al. 2012]
	Sequential detection	[Wang et al. 2012; Wang et al. 2013]
	Countering IRC-based botnets	[Houmansadr and Borisov 2013a]
IPD based	Generic	[Wang and Reeves 2003; 2011] [Pan et al. 2009; Wang et al. 2005]
	Quantization-index modulation	[Gong et al. 2012; 2013]
DSSS based	Varying traffic rate	[Yu et al. 2007; Huang et al. 2011]
	Modulating packet arrival time	[Zhang et al. 2009]
	Resistant to MSAC and Multi-flow Attacks	[Zhang et al. 2010b]
DSSS and other type combined	Combining with ICBW	[Wang et al. 2009; Luo et al. 2012b] [Wang et al. 2010]
	Combining with IBW	[Zhang et al. 2010a]

However it is not clear how the detectors will obtain the required information. Authors have not considered robustness and non-detectability requirements of the watermark.

A hybrid watermarking scheme called RAINBOW proposed in [Houmansadr et al. 2009b; 2014] uses smaller delays compared to blind techniques by eliminating the interference caused by the flow. Their extended scheme is shown to be robust against packet drops and repacketization using selective correlation and longer observation periods. Apart from the active watermarking like blind techniques their scheme requires an IPD database to hold all unwatermarked input flow IPD information. This requirement of sharing input flow properties is similar to passive/non-blind flow analysis techniques. In RAINBOW, all detectors must have access to all the entries in the database to make correct decision. The authors have also proposed to use the Repeat-Accumulate codes to improve the detection performance of RAINBOW [Houmansadr and Borisov 2011c].

4.3. Flow fingerprinting

To the best of our knowledge, flow fingerprinting problem was first introduced in [Houmansadr and Borisov 2013b] with clear distinction to NFW. They have proposed a hybrid fingerprinting technique called Fancy based on RAINBOW watermark and coding theory. A game-theoretic framework for network flow linking problem is proposed in [Elices and Perez-Gonzalez 2013] and authors have used it to derive the Nash Equilibrium. They have compared the optimal strategies with different fingerprinting schemes to study the limits of flow correlation based on packet timings against an active attacker.

Table V shows which NFW methods satisfy non-disruptability, and/or non-detectability requirements. It can be observed that many methods do not consider the non-detectability requirement. Houmansadr et al. have not considered the non-disclosure requirement of flow fingerprinting in [2013b].

4.4. Cost analysis of different categories

Consider a network with m incoming and n outgoing flows. If multiple detectors are used to detect relayed flows, continuous communication among the detectors is required for passive traffic analysis schemes to transmit flow statistics. This requires $\bigcirc(n)$ communication overhead for one detector [Houmansadr and Borisov 2011b]. Furthermore, a detector has to correlate each outgoing flow against all the original input flow candidates causing $\bigcirc(mn)$ computation overhead. The hybrid technique like RAINBOW also requires similar communication and computation overhead like pas-

Table V. Non-disruptability, and/or non-detectability requirements, addressed by different NFW methods.

Requirements	NFW methods
Non-disruptability	[Houmansadr et al. 2009a; 2009b; 2014] [Pyun et al. 2007; Pyun et al. 2012; Wang and Reeves 2003; 2011] [Luo et al. 2012b; Pan et al. 2009; Gong et al. 2012; 2013] [Wang et al. 2007; Wang et al. 2009; 2010] [Houmansadr and Borisov 2011b; 2011c; 2013a; 2013b]
Non-detectability	[Houmansadr et al. 2009a; Houmansadr and Borisov 2011b] [Huang et al. 2011; Zhang et al. 2010a; Zhang et al. 2010b] [Luo et al. 2012b; Wang et al. 2009; Gong et al. 2012; 2013]

sive or non-blind schemes [Houmansadr et al. 2014]. There is no communication cost other than the shared key ($\bigcirc(1)$ communication overhead) for blind schemes. Blind detectors process each outgoing flow individually to detect watermarks requiring $\bigcirc(n)$ computation overhead [Houmansadr and Borisov 2011b].

4.5. Threat models

Three types of threats are reported against NFW schemes based on the strength of the attacker - A) isolated attacker, B) partially known flow attack, and C) fully known flow attack [Gong et al. 2013].

4.5.1. Isolated attacker. In this type of attack an attacker only observes a watermarked flow. The attacker may also have some knowledge of original flow like distribution of IPDs. Kiyavash et al. have shown that interval based watermarking approach creates time dependent correlations that can be exploited using a multiple flow attack (MFA) [2008]. An attacker can detect the watermark, extract the secret parameters, and even delete the watermark pattern using MFA. The attack can not be defeated by using different watermarks in separate flows to carry different messages. They have assumed a Markov-modulated Poisson process (MMPP) model of interactive traffic to analyze their attack model. They have also proposed to use multiple watermark positions to defeat MFA.

A scheme to detect homogeneous PN code based watermarks in single flow is proposed in [Jia et al. 2009]. They have used traffic rate time series of a single modulated flow to get the mean-square autocorrelation (MSAC). This MSAC is then used to detect DSSS watermarks without knowing the applied PN code. Another spread-spectrum flow watermark detection mechanism using PN codes is proposed in [Luo et al. 2010]. They have also used TCPs flow-control mechanism to remove spread-spectrum flow watermarks.

4.5.2. Partially known flow attack. In this type of attack, an attacker has a distorted version of the original flow and observes a watermarked flow for detection. The imperfect version of original flow is more informative than the information available to an isolated attacker. Peng et al. have proposed an attack method by studying the packet delays among two close stepping stones [2006]. They have proposed to send packets with controlled timing. Then a detector can detect uncommon extra delays by statistical methods. An attacker can extract secret watermark parameters, and can duplicate the watermarks using their attack. Some trade-offs of watermarking becomes apparent from their work (like trade-off between detection rate and invisibility). Another partially known flow attack called BACKLIT is proposed in [Luo et al. 2011]. It is based on the fact that any timing-based flow watermark causes noticeable alterations of a TCP flow's intrinsic timing features. They have shown successful attack against IBW, ICBW, RAINBOW and SWIRL watermarking schemes.

Table VI. Main attributes of different threats toward NFW techniques.

Type of threat	Main attributes	Reference
Isolated attacker	Multi-flow based	[Kiyavash et al. 2008]
	Single-flow based (targeting DSSS based flow watermarking)	[Jia et al. 2009; Luo et al. 2010]
Partially known flow attack	Generic	[Peng et al. 2006; Luo et al. 2011]
Fully known flow attack	Generic	[Lin and Hopper 2012]

4.5.3. Fully known flow attack. In this type of attack, an attacker observes both the unwatermarked original flow and the watermarked flow. So the process of detection is easier than the other threat models. It must be noted that the assumptions involved in this specific attack may not be true for all practical cases and the attack may not be applicable also [Gong et al. 2013]. In [Lin and Hopper 2012], authors have introduced a known/chosen flow attack model and a copy attack. They have shown successful attacks against IBW, ICBW, RAINBOW and SWIRL. The literature on different types of threats to various NFW techniques is summarized in Table VI.

4.6. Summary

NFW techniques are quite new compared to the field of digital watermarking or multimedia watermarking. Many of the network flow watermarks are inspired by the multimedia watermarking techniques [Kiyavash et al. 2008]. Still there is a need to model various effects of network traffic on watermarks. More work can be done toward threat analysis of different watermarking schemes considering different practical aspects of the network such as the link layer protocol and the topology.

An active watermark-based correlation scheme to tackle arbitrary timing perturbations is proposed in [Wang and Reeves 2011]. Future research is possible to develop a NFW technique that is optimal from coding theoretic perspective. Development of robust NFW technique which requires only few packets is another future research direction. Effect of various flow transformations (like flow merging and splitting, packet drop, packet reordering, chaff packet addition etc) on NFW techniques can be analyzed more elaborately in future research.

A blind detection method for malicious DSSS traceback is proposed in [Jia et al. 2009]. Future work is possible in the design of detection mechanism when various bits of same signal are spread using many PN codes. Same sequence of PN codes can be used to despread the signal. The mean square autocorrelation may not be helpful for detection if the PN codes are chosen to be orthogonal. Detailed investigation is also necessary to see if design of flow watermark elimination system is possible such that the normal traffic characteristics (like throughput) do not get affected.

5. IN-SYSTEM TIMING CHANNELS

In-system timing channel refers to those channels that reside inside a system which can be a computer system or a chip (like SoC, MP-SoC). In-system timing channel can either be a TSC or a CTCC. The difference between the two types is that CTCC always has a covert source like a Trojan process but the TSC is unintentional information leakage by timing behavior. In this paper we consider only the hardware architecture based in-system timing channels. We will not discuss programming language and OS related or Virtual Machine and cloud computing related channels. Though insider attack is most prominent for hardware architecture related channels, both insider and external attacks are possible using in-system channels. A Trojan can be installed by an external party to obtain the required timing information.

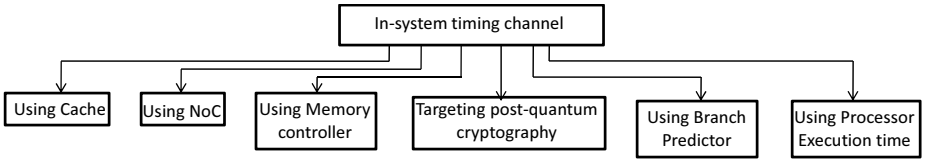


Fig. 5. In-system timing channel categories.

5.1. Architecture based in-system timing channel techniques

An architecture based in-system timing channel can be implemented using cache, network-on-chip (NoC), memory controller, branch predictor, or processor execution time of different cryptographic algorithms. Even some timing channel based attacks are reported against post-quantum cryptographic algorithms like McEliece public key cryptography (PKC). Among all these types of timing channels, cache based timing channels are most studied in literature. Figure 5 shows different in-system timing channel categories.

5.1.1. Cache based timing channel. Cache based attacks can be classified into three categories, namely, access-driven, trace-driven, and timing-channel attack [Savaş 2013]. In this paper we discuss about cache based timing channels only. An attacker can observe the variations in execution times of cryptographic operations and can infer the approximate number of cache hits or misses. In some attacks, an attacker can even locate the exact cache line that is accessed by the victim. There are many variations of cache timing attacks like chosen plaintext, chosen ciphertext or attack without any knowledge of plaintext or ciphertext.

The study in [Bernstein 2005] has shown that full AES key retrieval is possible from a network server using the timings of known-plaintext. Author has pointed out that the vulnerability is in the AES design itself due to difficulty in writing efficient AES code having fixed runtime. Several other timing attacks against the table-based software implementation of the AES cipher is proposed in [Bonneau and Mironov 2006]. They have predicted timing variation due to cache-collisions during encryption process and they have proposed a patch that can reduce the vulnerability to their attacks. Xinjie et al. have also proposed a cache timing attack against 128-bit AES using first two rounds access [2008]. They have used a malicious process to get the 128-bit full AES key by obtaining table lookup indices during encryption. Rebeiro et al. have proved that a similar third round cache based timing attack does not work [2010]. A differential cache-collision timing attack on AES software implementation is proposed in [Bogdanov et al. 2010]. They have treated pairs of AES executions differentially and have demonstrated the attack on ARM9 processor. Aly et al. have reproduced Bernstein's attack toward AES on Pentium Dual-Core and Core 2 Duo processors [2013]. They have targeted OpenSSL's AES implementation on Windows and Linux. They have used two-way measurements to improve Bernstein's first round attack and have also used the above minimum timing information to improve the results.

Percival has shown that memory caches with shared access, allows threads to create high bandwidth side channel [2005]. The shared access allows a malicious thread to observe another thread's execution. This may allow stealing of cryptographic keys. Some recommendations to mitigate or eliminate this attack entirely are also provided in [Percival 2005]. Authors in [Osvik et al. 2006; Tromer et al. 2010] have described inter-process leakages revealing memory access patterns through cache memories. They have also described an attack that does not require any knowledge of specific plaintexts or ciphertexts. They have targeted Linux's encrypted partition by dm-crypt,

and OpenSSL. They have also presented some countermeasures to mitigate the attacks.

First micro-architectural attack using Instruction Cache (I-Cache) is discussed in [Aciiçmez 2007]. A trace-driven timing attack toward RSA algorithm by observing the whole I-Cache is proposed in [Chen et al. 2013b]. They have also proposed an error detection mechanism to detect erroneous decisions to reduce the number of erroneous recovered bits. Vector quantization and hidden Markov models can be used to improve I-Cache data analysis techniques as proposed in [Aciiçmez et al. 2010; B. Brumley 2011]. This is demonstrated by recovering keys by attacking OpenSSL's DSA implementation. They have also proposed kernel or compiler level software countermeasures. Vector quantization and hidden Markov model cryptanalysis to recover secret key is also shown in [Brumley and Hakala 2009; B. Brumley 2011]. They have targeted the timing attack against the ECC in OpenSSL.

Hund et al. have presented the limitations of kernel space Address Space Layout Randomization (ASLR) against a local attacker with restricted privileges [2013]. They have shown that an adversary can implement a generic side channel attack against the memory management system to deduce information about the privileged address space layout.

5.1.2. NoC based timing channel. Network-on-chip (NoC) is a shared resource utilized by different applications running on a multiprocessor-system-on-chip (MP-SoC). One application can affect another application's timing characteristics through interference in some NoC resources like links and buffers. A malicious application can observe another application's data dependent timing characteristics and can obtain secret information resulting in TSC attack. Again two applications can communicate covertly using intentional modifications of timing characteristics resulting in a covert timing channel attack [Wang and Suh 2012].

Timing channel based watermarking methods are proposed in [Biswas 2016] to authenticate the source of the configuration packets for an NoC router. Hence router attacks due to malicious router configuration can be prevented in an MP-SoC. Timing channel watermarking methods, a stream authentication technique and different combinations of them are used to transfer the authentication data to a receiver router.

5.1.3. Memory controller based timing channel. A memory-based timing channel attack is launched by exploiting the fact that the memory latencies of one program depend on memory accesses by other programs sharing the same memory [Wang et al. 2014]. In case of a shared memory controller, a memory timing channel exists between software modules in different security domains. The reason is that the time to schedule a memory request depends on other competing requests. Sources of interference in a memory controller can be categorized into three groups: queuing structure interference, scheduler arbitration interference, and DRAM device resource contention. Wang et al. have demonstrated that shared memory controllers are vulnerable to both side channel and covert channel attacks that exploit memory interference as timing channels [2014].

5.1.4. Branch predictor based timing channel. First branch prediction based software side channel attack is proposed in [Aciiçmez et al. 2006]. It is shown that cryptanalysis can be conducted on a cryptographic primitive employing a data-dependent program flow. Cryptanalysis uses the extra clock cycle requirement for a mis-predicted branch. The attack is independent of memory protection mechanisms, sandboxing or virtualization. They have targeted the attack against RSA. Some countermeasures are also suggested to mitigate branch prediction attacks. Aciicmez et al. have proposed a Simple Branch Prediction Analysis (SBPA) attack [2007]. Simply by observing a quasi-parallel task, the states of the CPU's Branch Predictor can be analyzed. They have attacked an

OpenSSL RSA implementation by SBPA attack. A side channel timing attack against the MSP430 serial bootstrap loader (BSL) is proposed in [Goodspeed 2008]. Version 2.12 is the only version reported to be vulnerable. The attack is based on an unbalanced branch that takes two cycles longer to execute than other branch.

5.1.5. Execution time based timing channel. An attacker can obtain secret information used in a cryptographic algorithm by observing the time taken to execute different operations of that algorithm. Kocher has presented an attack that can factor RSA keys and get Diffie-Hellman exponents by computing private key execution times [1996]. He has proposed some techniques to prevent the attack for RSA and Diffie-Hellman. To the best of our knowledge, this is the first published work that considers execution time based timing attack to break cryptographic algorithms. A basic description of the cryptanalysis method is given next. Both Diffie-Hellman and RSA private-key operations require to compute $R = y^x \ mod \ n$, where n is public and y can be obtained by eavesdropping. The attack's main aim is to find the secret key x. For a successful attack, the victim is required to compute $y^x \ mod \ n$ for different values of y. Here it is assumed that the attacker knows the values of y, n, and the computation time. In fact attacker can record all the messages received by the target and can calculate the response time to each y. A timing attack to recover secret keys from a smart card using Kocher's proposed ideas is reported in [Dhem et al. 2000].

Kelsey et al. have designed timing attack toward a product block cipher called IDEA, Hamming weight attack toward DES, and processor-flag attack toward RC5 [1998]. Two DES implementations are analyzed in [Hevia and Kiwi 1999] against timing attack and authors have obtained the Hamming weight of the key used in both implementations. Furthermore they have shown that necessary design characteristics for the attack can be obtained from timing measurements. A timing attack against Rijndael is proposed in [Koeune and Quisquater 1999] using few thousand measurements per byte of the key.

A timing attack against RSA is proposed in [Schindler 2000]. The condition is that Chinese Remainder Theorem (CRT) and Montgomery's algorithm must be used for exponentiation with the secret exponent. Walter et al. have shown that conditional subtractions at the end of Montgomery modular multiplications can enable an attack on RSA without knowing the input plaintext [2001]. The attack proposed by [Walter and Thompson 2001] is generalized in [Schindler 2002] where it is shown that the original attack is not applicable for 4-bit tables. Author in [Schindler 2002] has also shown that this optimized attack can be launched against other table based methods, other multiplication algorithms and in-exact timings. Chen et al. have proposed a timing attack on RSA-CRT using t-test [2013a] which is enhanced with an error detection and correction mechanism to detect and correct the erroneous decision.

5.1.6. Targeting post-quantum cryptography. Security of modern public key cryptography (PKC) algorithms depends on some mathematical problems that cannot be solved by modern computers in acceptable times. But theses problems are assumed to be solvable by a quantum computer with sufficient number of qubits. So, cryptographic algorithms are required that can provide system and network security in the presence of practical quantum computers. McEliece PKC is one example of such cryptography. Strenzke et al. have proposed a timing attack toward a McEliece PKC code [2008]. They have also suggested some countermeasures to prevent these attacks. Avanzi et al. have also presented timing side channel attacks against the Niederreiter and McEliece PKCs and have proposed countermeasures against such attacks [2011].

The McEliece PKC uses Patterson Algorithm during decryption operation for error correction. A timing attack toward the Patterson Algorithm proposed in [Shoufan et al. 2010] can extract the secret error vector. A similar attack is proposed in [Strenzke

Table VII. Main attributes of different architecture based in-system timing channel works.

Type	**Main attributes**	**Architecture based in-system timing channel works**
Cache based	Targeting AES	[Bernstein 2005; Bonneau and Mironov 2006] [Xinjie et al. 2008; Rebeiro et al. 2010] [Bogdanov et al. 2010; Aly and ElGayyar 2013]
	Inter-process leakage	[Percival 2005; Osvik et al. 2006] [Tromer et al. 2010]
	Using Instruction Cache	[Aciiçmez 2007; Chen et al. 2013b] [Aciiçmez et al. 2010; B. Brumley 2011]
	Use of vector quantization and hidden Markov model	[Aciiçmez et al. 2010] [Brumley and Hakala 2009] [B. Brumley 2011]
	Targeting ASLR	[Hund et al. 2013]
NoC based	Inter-application interference	[Wang and Suh 2012]
	Source authentication	[Biswas 2016]
Memory controller based	Memory interference	[Wang et al. 2014]
Branch predictor based	Generic	[Aciiçmez et al. 2006; 2007]
	Against serial bootstrap loader	[Goodspeed 2008]
Execution time based	Generic	[Kocher 1996; Dhem et al. 2000]
	Toward a block cipher	[Kelsey et al. 1998; Hevia and Kiwi 1999] [Koeune and Quisquater 1999]
	Against RSA CRT and Montgomery's algorithm	[Schindler 2000; Walter and Thompson 2001] [Schindler 2002; Chen et al. 2013a]
Targeting post-quantum cryptography	Against public-key cryptosystems (like McEliece)	[Strenzke et al. 2008; Avanzi et al. 2011]
	Against Patterson Algorithm	[Shoufan et al. 2010; Strenzke 2010]

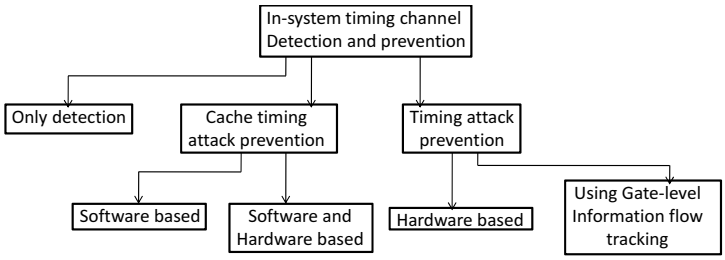


Fig. 6. In-system timing channel detection and prevention categories.

2010] to gather secret permutation information which is useful in a brute force attack against the secret key. A summary of the literature on architecture-based in-system timing channel is shown in Table VII.

5.2. Detection and Prevention

Figure 6 shows different categories of in-system timing channel detection and prevention works. Whereas some studies propose to prevent cache timing attack or execution time based timing attack, other studies exist that only try to detect timing channels.

5.2.1. Detection. Kemmerer has outlined a methodology for discovering storage and timing channels [1982; 1983; 2002]. The methodology known as the Shared Resource Matrix Methodology can be used through all phases of the software life cycle to increase the assurance that all channels have been identified. Another method to identify all timing channels in a computer system is presented in [Wray 1991]. Chen et al.

have also proposed an algorithm to identify shared processor timing channels [2014]. They have tested their algorithm on memory-bus/Quick-Path-Interconnect (QPI) based and integer divider based timing channels.

5.2.2. Cache timing attack prevention. Cache timing attack can be prevented by software-only or software-hardware combined approach.

A) Software solution. Software solution provides a timing attack resistant cryptographic algorithm implementation or employ randomization technique like masking. Masking is the use of random values to mask the input to a cryptographic algorithm to uncorrelate the intermediate results against the input. This process can stop useful information leakage from the timing side channel.

An AES randomization technique (masking) is proposed in [Blomer et al. 2005] and authors have formally proven its security against side channel attacks. Ksper et al. have presented a timing attack protected bit-sliced implementation of AES encryption [2009]. The implementation is a counter mode AES on 64-bit Intel processor. They have also proposed a constant-time implementation of AES-GCM that is resistant to timing attack. Another timing attack protected bit-sliced implementation of AES-128 is presented in [Konighofer 2008]. The implementation is immune to cache-timing attacks as it does not need table-lookups.

B) Software-hardware combined solution. To counter cache based timing channel, different cache architectures with their software control interfaces are proposed. There is a proposal to limit the timekeeping granularity to reduce the timing channel information bandwidth. Instruction sets supporting AES operations in hardware are also proposed.

Two security-alert caches called "Random Permutation Cache" (RPcache) and "Partition-Locked cache" (PLcache) are proposed in [Wang and Lee 2006; 2007]. They have shown that the partition-based PLcache can eliminate cache interference and the randomization based RPcache can randomize cache interference to minimize information leakage. Kong et al. have analyzed both PLcache and RP-cache and have shown vulnerabilities of those designs [2008]. They have proposed some modifications of those cache designs to overcome the vulnerabilities. Pre-loading is also proposed to secure the PLcache [Kong et al. 2009]. They have leveraged informing loads to protect the RPcache. They have also replaced the random permutation hardware in the RPcache with a software permutation technique.

Another cache architecture is proposed in [Wang and Lee 2008] to reduce cache miss rate, to overcome access-time overhead and to thwart information leakage. They have used a security-aware cache replacement algorithm to thwart information leakage. Liu et al. have tested the security of a security enhancing cache called Newcache using cache side channel attacks for conventional set-associative caches [2013]. They have shown that Newcache can thwart most of the attacks. They have also modified the vulnerable replacement algorithm in Newcache design to secure against the remaining possible attacks.

Techniques to curb the adherence of performance counters and accurate timekeeping are proposed in [Martin et al. 2012]. They have shown that their proposed methods can thwart timing attacks by confusing the attacker in distinguishing different micro-architectural events.

It is shown in [Mowery et al. 2012] that data-cache side channel attacks on AES has become more difficult because of five new developments in software and hardware. These developments are: physically tagged caches, sophisticated software, new prefetchers, multi-core processors with caches, and the AES-NI instructions. The Intel AES New Instructions (NI) are presented in [Gueron 2012]. The proposed Intel archi-

Table VIII. Main attributes of different cache timing attack prevention works.

Type	Main attributes	Cache timing attack prevention works
Software based	Masking	[Blomer et al. 2005]
	Immune implementation	[Ksper and Schwabe 2009; Konighofer 2008]
Software-hardware combined	Cache design	[Wang and Lee 2006; 2007; 2008] [Kong et al. 2008; Kong et al. 2009; Liu and Lee 2013]
	Limiting fidelity of timekeeping	[Martin et al. 2012]
	Hardware support	[Mowery et al. 2012; Gueron 2012]

tecture consists of 6 instructions with hardware support for AES. The table-less and data-independent runtime AES implementation can prevent timing attacks.

Main attributes of different cache timing attack prevention works are presented in Table VIII.

5.2.3. Timing attack prevention. There are mainly two different solutions to prevent timing attacks: hardware based solution and gate level information flow tracking (GLIFT). Both of these solutions are described next separately.

A) Hardware based solution. Different hardware implementations of cryptographic algorithms are proposed that can prevent execution time based timing attack. There is a proposal to make all clocks available to a process noisy (fuzzy time) and thus limiting timing channel bandwidth [Hu 1991].

A parallel architecture to counteract the timing attack is proposed in [Ciet et al. 2003]. They have used Montgomery Multiplication derived from Residue Number Systems (RNS) to design the architecture. It can perform an RSA signature in parallel on a set of identical and independent co-processors. A timing attack resistant elliptic curve processor is proposed in [Hodjat et al. 2005]. They have modified the point multiplication algorithm (double-add-subtract) to prevent timing attack. The processor is based on the Galois Field of GF(2n) where n is configurable. Ghosh et al. have proposed another programmable GF(p) arithmetic unit to perform modular addition, subtraction, multiplication, inversion, and division [2011]. They have also designed an elliptic curve scalar multiplication hardware to perform point doubling and point addition in each iteration concurrently on two cores of programmable GF(p) arithmetic unit.

An 128-bit CCA2-secure McEliece cryptoprocessor is proposed in [Ghosh and Verbauwhede 2014] incorporating BLAKE-512 module into the architecture. They have shown that their design is resistant against existing timing attacks. They have also introduced a binary-XGCD algorithm for Goppa field. Dual-spacer dual-rail delay-insensitive Logic (D^3L) is proposed in [Cilio et al. 2010; Cilio et al. 2013] to mitigate timing attacks. They have inserted random delays to mitigate the timing-data correlation.

Different hardware based solutions are available in literature to prevent timing channel attack in Network-on-chip (NoC) based MP-SoC. Wang et al. have proposed a technique called “Reversed Priority with Static Limits” (RPSL) that requires changes to the router hardware [2012]. In RPSL scheme high priority is given to the low-security traffic. So low-security application cannot imply any information about high-security application using congestion in NoC and hence timing channel is mitigated. An on-chip network called SurfNoC is proposed in [Wassel et al. 2013] to reduce the latency due to temporal partitioning. They have introduced a scheduling policy and router micro-architecture to allow data from different domains to flow in a strictly non-interfering manner.

A memory controller proposed in [Wang et al. 2014] allows mutually mis-trusting parties to share main memory securely by eliminating memory timing channels. They

Table IX. A summary of the literature on different timing attack prevention studies.

Type	Main attributes	Timing attack prevention works
Hardware based	Fuzzy time	[Hu 1991]
	Modified architecture	[Ciet et al. 2003; Hodjat et al. 2005] [Ghosh et al. 2011; Ghosh and Verbauwhede 2014]
	Random delay	[Cilio et al. 2010; Cilio et al. 2013]
	NoC architecture	[Wang and Suh 2012; Wassel et al. 2013]
	Memory controller	[Wang et al. 2014]
Using GLIFT	Implementation	[Tiwari et al. 2009; Tiwari et al. 2010] [Tiwari et al. 2011; Oberg et al. 2011]
	Theoretical analysis	[Oberg et al. 2010; Hu et al. 2012]

have proposed two changes to eliminate the interference across security domains. The changes are security domain specific queuing structure, and time slots statically allocated in the scheduling algorithm.

B) Using gate level information flow tracking (GLIFT). GLIFT is used to track all implicit, explicit and timing information movements within a system. The proposed hardware solutions are costly (in terms of area, time etc.) compared to the original implementations.

Tiwari et al. have proposed a GLIFT based architecture implementing Shadow Logic capable of tracking all explicit and implicit information flows within a machine [2009; 2010]. They have termed the logic "Shadow Logic", because it co-exists with normal logic and helps to understand the information flow propagation throughout a system. Their proposed implementation is not general-purpose programmable but supports some programming to handle public-key encryption and authentication. Another configurable architectural skeleton is proposed in [Tiwari et al. 2011] that combines a microkernel with a hardware realization. They have statically verified whole structure's information flow attributes at the gate-level implementation. Oberg et al. have proposed a methodology for testing information flows in I2C and USB using GLIFT [2011]. They have shown that unintended information flows are present in those two protocols. They have also proposed a method to isolate devices on the bus using time division multiple access (TDMA).

Theoretical analysis of GLIFT is presented in [Oberg et al. 2010] and authors have explained the Shadow Logic theory for GLIFT implementing systems. They have shown that for Shadow Logic, the minterms increase exponentially with the increase of inputs. Hu et al. have given a formal basis for deriving GLIFT logic [2012] and have proved that generation of precise GLIFT logic is NP-complete.

A summary of the literature on different timing attack prevention studies is shown in Table IX.

5.3. Summary

Most works consider only malicious applications for in-system timing channels. More elaborate studies are required on possible legitimate application aspects of in-system channels. Most of the timing attack related works target cache but there are other resources in a system like interconnection modules (routers) in a multiprocessor system. Detailed studies about possible timing attack vulnerabilities related to other resources are required. It is true that specific modification in an algorithm is required if it is found vulnerable. Nevertheless, general purpose preventive measures that is application- and architecture-independent can be developed to protect from some, if not all, timing attacks. Mitigation techniques can be implemented solely on software, solely on hardware, or on a mixture of both software and hardware.

Various attack scenarios targeting RSA is reported in [Aciiçmez et al. 2006] exploiting CPU's branch prediction unit. Exploration of efficient mitigation techniques

against such attacks is a viable future research direction. Further investigation is required to see if such attacks are possible against symmetric ciphers like DES. The branch prediction based attacks depend on Simultaneous Multi-threading (SMT) ability of a CPU but it will be of great importance to see if the attacks can be effective against non-SMT capable CPUs. Two cache architectures i.e., the RPcache and the PLcache are proposed in [Wang and Lee 2007] to prevent cache based TSC attacks by eliminating or randomizing cache interference. Future research is required to explore the design space of security-aware cache and computer architectures without sacrificing performance, cost, and energy consumption.

A formal notion of security for randomized maskings is presented in [Blomer et al. 2005] for cryptographic algorithms. Future research can be conducted to find solutions where the requirement of randomness is minimized without any adverse effect on security. The reason is that true randomness generation is very costly. A timing-attack resistant crypto-processor for computing McEliece post-quantum PKS is proposed in [Ghosh and Verbauwhede 2014]. Future research can be directed to develop efficient crypto-processors for different code-based crypto-systems like Niederreiter PKS.

6. CONCLUSIONS

In this paper we have provided a detailed survey of timing channels. Both network and in-system timing channels have been considered. We have given pictorial models for CTCC, NFW and TSC. Both network and in-system timing channels are categorized and each category is discussed in detail. We have also categorized and provided detailed discussions of different detection and prevention techniques of different types of timing channels.

We have also indicated the various possibilities of future research directions. For example, innovative legitimate and malicious applications of timing channels are a possible future research direction. Till now timing channels are primarily considered as a threat to information security. However it must be noted that no technology is good or bad by itself. It is the application of that technology that makes it good or bad. This is more relevant in current scenarios of high speed networks and massively parallel multiprocessor chips. Malicious application scenarios are also needed to be examined so that effective detection and prevention techniques can be devised for them.

REFERENCES

O. Aciiçmez. 2007. Yet Another MicroArchitectural Attack:: Exploiting I-Cache. In *Proceedings of the 2007 ACM Workshop on Computer Security Architecture (CSAW '07)*. ACM, New York, NY, USA, 11–18.

O. Aciiçmez, B. Brumley, and P. Grabher. 2010. New Results on Instruction Cache Attacks. In *Cryptographic Hardware and Embedded Systems, CHES 2010*, Stefan Mangard and Franois-Xavier Standaert (Eds.). Lecture Notes in Computer Science, Vol. 6225. Springer Berlin Heidelberg, 110–124.

O. Aciiçmez, Ç.K. Koç, and J. Seifert. 2006. Predicting Secret Keys Via Branch Prediction. In *Topics in Cryptology CT-RSA 2007*, Masayuki Abe (Ed.). Lecture Notes in Computer Science, Vol. 4377. Springer Berlin Heidelberg, 225–242.

O. Aciiçmez, Ç.K. Koç, and J. Seifert. 2007. On the Power of Simple Branch Prediction Analysis. In *Proceedings of the 2Nd ACM Symposium on Information, Computer and Communications Security (ASIACCS '07)*. ACM, New York, NY, USA, 312–320.

O. Aciiçmez, W. Schindler, and Ç.K. Koç. 2005. Improving Brumley and Boneh Timing Attack on Unprotected SSL Implementations. In *Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS '05)*. ACM, New York, NY, USA, 139–146.

O. Aciiçmez, W. Schindler, and Ç.K. Koç. 2007. Cache Based Remote Timing Attack on the AES. In *Topics in Cryptology CT-RSA 2007*, Masayuki Abe (Ed.). Lecture Notes in Computer Science, Vol. 4377. Springer Berlin Heidelberg, 271–286.

S.A. Ahmadzadeh and G. Agnew. 2013. Turbo covert channel: An iterative framework for covert communication over data networks. In *INFOCOM, 2013 Proceedings IEEE*. 2031–2039.

K. Ahsan and D. Kundur. 2002. Practical Data Hiding in TCP/IP. In *Proc. Workshop on Multimedia Security at ACM Multimedia '02* (2002-12-01). Juan Les Pins, France.

H. Aly and M. ElGayyar. 2013. Attacking AES Using Bernsteins Attack on Modern Processors. In *Progress in Cryptology AFRICACRYPT 2013*, Amr Youssef, Abderrahmane Nitaj, and AboulElla Hassanien (Eds.). Lecture Notes in Computer Science, Vol. 7918. Springer Berlin Heidelberg, 127–139.

V. Anantharam and S. Verdu. 1996. Bits through queues. *Information Theory, IEEE Transactions on* 42, 1 (jan 1996), 4 –18.

R. Archibald. 2013. *Design and Detection of Covert Communiation: Timing Channels and Application Tunneling*. Ph.D. Dissertation. Davis, CA., USA.

R. Archibald and D. Ghosal. 2012. A Covert Timing Channel Based on Fountain Codes. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on*. 970–977.

R. Archibald and D. Ghosal. 2014. A comparative analysis of detection metrics for covert timing channels. *Computers & Security* 45 (2014), 284 – 292.

A. Askarov, D. Zhang, and A.C. Myers. 2010. Predictive Black-box Mitigation of Timing Channels. In *Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS '10)*. ACM, New York, NY, USA, 297–307.

R. Avanzi, S. Hoerder, D. Page, and M. Tunstall. 2011. Side-channel attacks on the McEliece and Niederreiter public-key cryptosystems. *Journal of Cryptographic Engineering* 1, 4 (2011), 271–281.

B. Brumley. 2011. *Covert Timing Channels, Caching, and Cryptography*. Ph.D. Dissertation. Espoo, Finland.

A.S. Bedekar and M. Azizoglu. 1998. The information-theoretic capacity of discrete-time queues. *Information Theory, IEEE Transactions on* 44, 2 (Mar 1998), 446–461.

D.E. Bell and L.J.L. Padula. 1976. *Secure Computer System: Unified Exposition and MULTICS Interpretation*. Technical Report 522B. Deputy for Command and Management System, Hanscom Air Force Base, Bedford, Massachusetts.

D.J. Bernstein. 2005. *Cache-timing attacks on AES*. Technical Report. Department of Mathematics, Statistics, and Computer Science, The University of Illinois at Chicago, Chicago, IL 606077045.

Arnab Kumar Biswas. 2016. Source Authentication Techniques for Network-on-Chip Router Configuration Packets. *J. Emerg. Technol. Comput. Syst.* 13, 2, Article 28 (Nov. 2016), 31 pages. DOI:http://dx.doi.org/10.1145/2996194

J. Blomer, J. Guajardo, and V. Krummel. 2005. Provably Secure Masking of AES. In *Selected Areas in Cryptography*, Helena Handschuh and M.Anwar Hasan (Eds.). Lecture Notes in Computer Science, Vol. 3357. Springer Berlin Heidelberg, 69–83.

A. Bogdanov, T. Eisenbarth, C. Paar, and M. Wienecke. 2010. Differential Cache-Collision Timing Attacks on AES with Applications to Embedded CPUs. In *Topics in Cryptology - CT-RSA 2010*, Josef Pieprzyk (Ed.). Lecture Notes in Computer Science, Vol. 5985. Springer Berlin Heidelberg, 235–251.

J. Bonneau and I. Mironov. 2006. Cache-Collision Timing Attacks Against AES. In *Cryptographic Hardware and Embedded Systems - CHES 2006*, Louis Goubin and Mitsuru Matsui (Eds.). Lecture Notes in Computer Science, Vol. 4249. Springer Berlin Heidelberg, 201–215.

B. Brumley and R. Hakala. 2009. Cache-Timing Template Attacks. In *Advances in Cryptology ASIACRYPT 2009*, Mitsuru Matsui (Ed.). Lecture Notes in Computer Science, Vol. 5912. Springer Berlin Heidelberg, 667–684.

B. Brumley and N. Tuveri. 2011. Remote Timing Attacks Are Still Practical. In *Computer Security ESORICS 2011*, Vijay Atluri and Claudia Diaz (Eds.). Lecture Notes in Computer Science, Vol. 6879. Springer Berlin Heidelberg, 355–371.

D. Brumley and D. Boneh. 2003. Remote Timing Attacks Are Practical. In *Proceedings of the 12th Conference on USENIX Security Symposium - Volume 12 (SSYM'03)*. USENIX Association, Berkeley, CA, USA, 1–1.

D. Brumley and D. Boneh. 2005. Remote timing attacks are practical. *Computer Networks* 48, 5 (2005), 701 – 716.

Christian C. 2004. An information-theoretic model for steganography. *Information and Computation* 192, 1 (2004), 41 – 56.

S. Cabuk. 2006. *Network Covert Channels: Design, Analysis, Detection, and Elimination*. Ph.D. Dissertation. West Lafayette, IN, USA.

S. Cabuk, C.E. Brodley, and C. Shields. 2009. IP Covert Channel Detection. *ACM Trans. Inf. Syst. Secur.* 12, 4, Article 22 (April 2009), 29 pages.

R.C. Chakinala, A. Kumarasubramanian, R. Manokaran, G. Noubir, C.P. Rangan, and R. Sundaram. 2007. Steganographic Communication in Ordered Channels. In *Information Hiding*, JanL. Camenisch, Chris-

tianS. Collberg, NeilF. Johnson, and Phil Sallee (Eds.). Lecture Notes in Computer Science, Vol. 4437. Springer Berlin Heidelberg, 42–57.

C. Chen, M. Song, G. Hsieh, and C. Xin. 2011. A PLL Based Approach to Building an Effective Covert Timing Channel. In *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*. 1–5.

C. Chen, T. Wang, Y. Kou, X. Chen, and X. Li. 2013b. Improvement of trace-driven I-Cache timing attack on the {RSA} algorithm. *Journal of Systems and Software* 86, 1 (2013), 100 – 107.

C. Chen, T. Wang, and J. Tian. 2013a. Improving timing attack on RSA-CRT via error detection and correction strategy. *Information Sciences* 232 (2013), 464 – 474.

J. Chen and G. Venkataramani. 2014. An Algorithm for Detecting Contention-based Covert Timing Channels on Shared Hardware. In *Proceedings of the Third Workshop on Hardware and Architectural Support for Security and Privacy (HASP '14)*. ACM, New York, NY, USA, Article 1, 8 pages.

M. Ciet, M. Neve, E. Peeters, and J-J. Quisquater. 2003. Parallel FPGA implementation of RSA with residue number systems - can side-channel threats be avoided?. In *Circuits and Systems, 2003 IEEE 46th Midwest Symposium on*, Vol. 2. 806–810 Vol. 2.

W. Cilio, M. Linder, C. Porter, J. Di, S. Smith, and D. Thompson. 2010. Side-channel attack mitigation using dual-spacer Dual-rail Delay-insensitive Logic (D3L). In *IEEE SoutheastCon 2010 (SoutheastCon), Proceedings of the*. 471–474.

W. Cilio, M. Linder, C. Porter, J. Di, D.R. Thompson, and S.C. Smith. 2013. Mitigating power- and timing-based side-channel attacks using dual-spacer dual-rail delay-insensitive asynchronous logic. *Microelectronics Journal* 44, 3 (2013), 258 – 269.

T.P. Coleman and N. Kiyavash. 2008a. Practical codes for queueing channels: An algebraic, state-space, message-passing approach. In *Information Theory Workshop, 2008. ITW '08. IEEE*. 318–322.

T.P. Coleman and N. Kiyavash. 2008b. Sparse graph codes and practical decoding algorithms for communicating over packet timings in networks. In *Information Sciences and Systems, 2008. CISS 2008. 42nd Annual Conference on*. 447–452.

S.A. Crosby, D.S. Wallach, and R.H. Riedi. 2009. Opportunities and Limits of Remote Timing Attacks. *ACM Trans. Inf. Syst. Secur.* 12, 3, Article 17 (Jan. 2009), 29 pages.

Department of Defense Standard. 1985. Trusted Computer System Evaluation Criteria. *Tech. Rep. DOD 5200.28-STD* (1985).

J. Dhem, F. Koeune, P. Leroux, P. Mestre, J. Quisquater, and J. Willems. 2000. A Practical Implementation of the Timing Attack. In *Smart Card Research and Applications*, Jean-Jacques Quisquater and Bruce Schneier (Eds.). Lecture Notes in Computer Science, Vol. 1820. Springer Berlin Heidelberg, 167–182.

R. Dingledine, N. Mathewson, and P. Syverson. 2004. Tor: The Second-generation Onion Router. In *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13 (SSYM'04)*. USENIX Association, Berkeley, CA, USA, 21–21.

D.L. Donoho, A.G. Flesia, U. Shankar, V. Paxson, J. Coit, and S. Staniford. 2002. *Multiscale Stepping-Stone Detection: Detecting Pairs of Jittered Interactive Streams by Exploiting Maximum Tolerable Delay*. Springer Berlin Heidelberg, Berlin, Heidelberg, 17–35.

B.P. Dunn, M. Bloch, and J.N. Laneman. 2009. Secure bits through queues. In *Networking and Information Theory, 2009. ITW 2009. IEEE Information Theory Workshop on*. 37–41.

J.J. Edwards, J.D. Brown, and P.C. Mason. 2012. Using covert timing channels for attack detection in MANETs. In *MILITARY COMMUNICATIONS CONFERENCE, 2012 - MILCOM 2012*. 1–7.

A. El-Atawy and E. Al-Shaer. 2009. Building Covert Channels over the Packet Reordering Phenomenon. In *INFOCOM 2009, IEEE*. 2186–2194.

J.A. Elices and F. Perez-Gonzalez. 2013. The flow fingerprinting game. In *Information Forensics and Security (WIFS), 2013 IEEE International Workshop on*. 97–102.

I. Ezzeddine and P. Moulin. 2009. Achievable rates for queue-based timing stegocodes. In *Information Theory Workshop, 2009. ITW 2009. IEEE*. 379–383.

S. Ghosh, D. Mukhopadhyay, and D. Roychowdhury. 2011. Petrel: Power and Timing Attack Resistant Elliptic Curve Scalar Multiplier Based on Programmable GF(p) Arithmetic Unit. *Circuits and Systems I: Regular Papers, IEEE Transactions on* 58, 8 (Aug 2011), 1798–1812.

S. Ghosh and I. Verbauwhede. 2014. BLAKE-512-Based 128-Bit CCA2 Secure Timing Attack Resistant McEliece Cryptoprocessor. *Computers, IEEE Transactions on* 63, 5 (May 2014), 1124–1133.

S. Gianvecchio and H. Wang. 2007. Detecting covert timing channels: an entropy-based approach. In *CCS '07: Proceedings of the 14th ACM Conference on Computer and Communications Security*. 307–316.

S. Gianvecchio and H. Wang. 2011. An Entropy-Based Approach to Detecting Covert Timing Channels. *Dependable and Secure Computing, IEEE Transactions on* 8, 6 (nov.-dec. 2011), 785–797.

S. Gianvecchio, H. Wang, D. Wijesekera, and S. Jajodia. 2008. Model-Based Covert Timing Channels: Automated Modeling and Evasion. In *RAID '08: Proceedings of the 11th international symposium on Recent Advances in Intrusion Detection*. Springer-Verlag, Berlin, Heidelberg, 211–230.

J. Giles and B. Hajek. 2002. An information-theoretic and game-theoretic study of timing channels. *Information Theory, IEEE Transactions on* 48, 9 (sep 2002), 2455 – 2477.

C.G. Girling. 1987. Covert Channels in LAN's. *IEEE Transactions on Software Engineering* 13, 2 (1987), 292–296.

S.Z. Goher, B. Javed, and N.A. Saqib. 2012. Covert channel detection: A survey based analysis. In *High Capacity Optical Networks and Enabling Technologies (HONET), 2012 9th International Conference on*. 057–065.

X. Gong and N. Kiyavash. 2013. Timing side channels for traffic analysis. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. 8697–8701.

X. Gong, M. Rodrigues, and N. Kiyavash. 2012. Invisible flow watermarks for channels with dependent substitution and deletion errors. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*. 1773–1776.

X. Gong, M. Rodrigues, and N. Kiyavash. 2013. Invisible Flow Watermarks for Channels With Dependent Substitution, Deletion, and Bursty Insertion Errors. *Information Forensics and Security, IEEE Transactions on* 8, 11 (Nov 2013), 1850–1859.

T. Goodspeed. 2008. A Side-channel Timing Attack of the MSP430 BSL. In *Black Hat USA 2008*.

S.K. Gorantla, S. Kadloor, T.P. Coleman, N. Kiyavash, I.S. Moskowitz, and M.H. Kang. 2010. Directed information and the NRL Network Pump. In *Information Theory and its Applications (ISITA), 2010 International Symposium on*. 343–348.

S.K. Gorantla, S. Kadloor, N. Kiyavash, T.P. Coleman, I.S. Moskowitz, and M.H. Kang. 2012. Characterizing the Efficacy of the NRL Network Pump in Mitigating Covert Timing Channels. *Information Forensics and Security, IEEE Transactions on* 7, 1 (FEBRUARY 2012), 64–75.

S. Gueron. 2012. *Intel Advanced Encryption Standard (AES) New Instructions Set*. Technical Report 323641-001, Revision 3.01. Intel Corporation, Intel Architecture Group, Israel Development Center.

A. Herzberg and H. Shulman. 2013. Limiting MitM to MitE Covert-Channels. In *Availability, Reliability and Security (ARES), 2013 Eighth International Conference on*. 236–241.

A. Hevia and M. Kiwi. 1999. Strength of Two Data Encryption Standard Implementations Under Timing Attacks. *ACM Trans. Inf. Syst. Secur.* 2, 4 (Nov. 1999), 416–437.

A. Hodjat, D.D. Hwang, and I. Verbauwhede. 2005. A scalable and high performance elliptic curve processor with resistance to timing attacks. In *Information Technology: Coding and Computing, 2005. ITCC 2005. International Conference on*, Vol. 1. 538–543 Vol. 1.

A. Houmansadr and N. Borisov. 2011a. CoCo: Coding-Based Covert Timing Channels for Network Flows. In *Information Hiding (Lecture Notes in Computer Science)*, Toms Filler, Toms Pevn, Scott Craver, and Andrew D. Ker (Eds.), Vol. 6958. Springer, 314–328.

A. Houmansadr and N. Borisov. 2011b. SWIRL: A scalable watermark to detect correlated network flows. In *In Network and Distributed System Security Symposium. Internet Society*.

A. Houmansadr and N. Borisov. 2011c. Towards improving network flow watermarks using the repeat-accumulate codes. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*. 1852–1855.

A. Houmansadr and N. Borisov. 2013a. BotMosaic: Collaborative network watermark for the detection of IRC-based botnets. *Journal of Systems and Software* 86, 3 (2013), 707 – 715.

A. Houmansadr and N. Borisov. 2013b. The Need for Flow Fingerprints to Link Correlated Network Flows. In *Privacy Enhancing Technologies*, Emiliano De Cristofaro and Matthew Wright (Eds.). Lecture Notes in Computer Science, Vol. 7981. Springer Berlin Heidelberg, 205–224.

A. Houmansadr, N. Kiyavash, and N. Borisov. 2009a. Multi-flow attack resistant watermarks for network flows. In *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*. 1497–1500.

A. Houmansadr, N. Kiyavash, and N. Borisov. 2009b. RAINBOW: A robust and invisible non-blind watermark for network flows. In *In: Proceedings of the Network and Distributed System Security Symposium (NDSS 2009). The INTERNET Society*.

A. Houmansadr, N. Kiyavash, and N. Borisov. 2014. Non-Blind Watermarking of Network Flows. *Networking, IEEE/ACM Transactions on* 22, 4 (Aug 2014), 1232–1244.

W.M. Hu. 1991. Reducing timing channels with fuzzy time. In *Proceedings of IEEE Computer Society Symposium on Research in Security and Privacy*. 8–20.

W. Hu, J. Oberg, A. Irturk, M. Tiwari, T. Sherwood, D. Mu, and R. Kastner. 2012. On the Complexity of Generating Gate Level Information Flow Tracking Logic. *Information Forensics and Security, IEEE Transactions on* 7, 3 (June 2012), 1067–1080.

J. Huang, X. Pan, X. Fu, and J. Wang. 2011. Long PN code based DSSS watermarking. In *INFOCOM, 2011 Proceedings IEEE*. 2426–2434.

R. Hund, C. Willems, and T. Holz. 2013. Practical Timing Side Channel Attacks against Kernel Space ASLR. In *Security and Privacy (SP), 2013 IEEE Symposium on*. 191–205.

J. Jaskolka, R. Khedri, and Q. Zhang. 2012. On the Necessary Conditions for Covert Channel Existence: A State-of-the-Art Survey. *Procedia Computer Science* 10 (2012), 458 – 465.

W. Jia, F.P. Tso, Z. Ling, X. Fu, D. Xuan, and W. Yu. 2009. Blind Detection of Spread Spectrum Flow Watermarks. In *INFOCOM 2009, IEEE*. 2195–2203.

J. Kelsey, B. Schneier, D. Wagner, and C. Hall. 1998. Side channel cryptanalysis of product ciphers. In *Computer Security ESORICS 98*, Jean-Jacques Quisquater, Yves Deswarte, Catherine Meadows, and Dieter Gollmann (Eds.). Lecture Notes in Computer Science, Vol. 1485. Springer Berlin Heidelberg, 97–110.

R. Kemmerer. 1982. A Practical Approach to Identifying Storage and Timing Channels. In *Security and Privacy, IEEE Symposium on*.

R.A. Kemmerer. 1983. Shared Resource Matrix Methodology: An Approach to Identifying Storage and Timing Channels. *ACM Trans. Comput. Syst.* 1, 3 (Aug. 1983), 256–277.

R.A. Kemmerer. 2002. A practical approach to identifying storage and timing channels: twenty years later. In *Computer Security Applications Conference, 2002. Proceedings. 18th Annual*. 109–118.

H. Khan, Y. Javed, F. Mirza, and S.A. Khayam. 2009. Embedding a Covert Channel in Active Network Connections. In *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*. 1–6.

N. Kiyavash and T. Coleman. 2009. Covert timing channels codes for communication over interactive traffic. In *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*. Taipei, Taiwan, 1485–1488.

N. Kiyavash, A. Houmansadr, and N. Borisov. 2008. Multi-flow Attacks Against Network Flow Watermarking Schemes. In *17th USENIX Security Symposium*.

N. Kiyavash, F. Koushanfar, T.P. Coleman, and M. Rodrigues. 2013. A Timing Channel Spyware for the CSMA/CA Protocol. *Information Forensics and Security, IEEE Transactions on* 8, 3 (March 2013), 477–487.

P.C. Kocher. 1996. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *Advances in Cryptology CRYPTO 96*, Neal Koblitz (Ed.). Lecture Notes in Computer Science, Vol. 1109. Springer Berlin Heidelberg, 104–113.

F. Koeune and J. Quisquater. 1999. *A timing attack against Rijndael*. Technical Report CG-1999/1. Universite catholique de Louvain, Departement d Electricite (DICE), Belgium.

J. Kong, O. Aciiçmez, J. Seifert, and H. Zhou. 2008. Deconstructing New Cache Designs for Thwarting Software Cache-based Side Channel Attacks. In *Proceedings of the 2Nd ACM Workshop on Computer Security Architectures (CSAW '08)*. ACM, New York, NY, USA, 25–34.

J. Kong, O. Aciicmez, J.P. Seifert, and H. Zhou. 2009. Hardware-software integrated approaches to defend against software cache-based side channel attacks. In *High Performance Computer Architecture, 2009. HPCA 2009. IEEE 15th International Symposium on*. 393–404.

R. Konighofer. 2008. A Fast and Cache-Timing Resistant Implementation of the AES. In *Topics in Cryptology CT-RSA 2008*, Tal Malkin (Ed.). Lecture Notes in Computer Science, Vol. 4964. Springer Berlin Heidelberg, 187–202.

K. Kothari and M. Wright. 2013. Mimic: An active covert channel that evades regularity-based detection. *Computer Networks* 57, 3 (2013), 647 – 657.

E. Ksper and P. Schwabe. 2009. Faster and Timing-Attack Resistant AES-GCM. In *Cryptographic Hardware and Embedded Systems - CHES 2009*, Christophe Clavier and Kris Gaj (Eds.). Lecture Notes in Computer Science, Vol. 5747. Springer Berlin Heidelberg, 1–17.

B.W. Lampson. 1973. A Note on the Confinement Problem. *Commun. ACM* 16, 10 (Oct. 1973), 613–615.

K.S. Lee, H. Wang, and H. Weatherspoon. 2014. PHY Covert Channels: Can You See the Idles?. In *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation (NSDI'14)*. USENIX Association, Berkeley, CA, USA, 173–185.

Z. Lin and N. Hopper. 2012. New Attacks on Timing-based Network Flow Watermarks. In *Proceedings of the 21st USENIX Conference on Security Symposium (Security'12)*. USENIX Association, Berkeley, CA, USA, 20–20.

F. Liu and R.B. Lee. 2013. Security Testing of a Secure Cache Design. In *Proceedings of the 2Nd International Workshop on Hardware and Architectural Support for Security and Privacy (HASP '13)*. ACM, New York, NY, USA, Article 3, 8 pages.

G. Liu, J. Zhai, and Y. Dai. 2012. Network covert timing channel with distribution matching. *Telecommunication Systems* 49, 2 (2012), 199–205.

G. Liu, J. Zhai, Y. Dai, and Z. Wang. 2009. Covert Timing Channel with Distribution Matching. In *Proceedings of the 2009 International Conference on Multimedia Information Networking and Security - Volume 01 (MINES '09)*. IEEE Computer Society, Washington, DC, USA, 565–568.

Y. Liu, F. Armknecht, D. Ghosal, S. Katzenbeisser, A. Sadeghi, and S. Schulz. 2009. Hide and Seek in Time - Robust Covert Timing Channels. In *14th European Symposium on Research in Computer Security (ESORICS)*.

Y. Liu, D. Ghosal, F. Armknecht, A.R. Sadeghi, S. Schulz, and S. Katzenbeisser. 2010. Robust and Undetectable Steganographic Timing Channels for i.i.d. Traffic. In *Information Hiding*, Rainer Bhme, PhilipW.L. Fong, and Reihaneh Safavi-Naini (Eds.). Lecture Notes in Computer Science, Vol. 6387. Springer Berlin Heidelberg, 193–207.

J. Luo, X. Wang, and M. Yang. 2012b. An interval centroid based spread spectrum watermarking scheme for multi-flow traceback. *Journal of Network and Computer Applications* 35, 1 (2012), 60 – 71.

X. Luo, E.W.W. Chan, and R.K.C. Chang. 2007. Cloak: A Ten-Fold Way for Reliable Covert Communications. In *Computer Security ESORICS 2007*, Joachim Biskup and Javier Lpez (Eds.). Lecture Notes in Computer Science, Vol. 4734. Springer Berlin Heidelberg, 283–298.

X. Luo, E. Chan, and R. Chang. 2008. TCP covert timing channels: Design and detection. In *Dependable Systems and Networks With FTCS and DCC, 2008. DSN 2008. IEEE International Conference on*. 420 –429.

X. Luo, E.W.W. Chan, P. Zhou, and R.K.C. Chang. 2012a. Robust Network Covert Communications Based on TCP and Enumerative Combinatorics. *Dependable and Secure Computing, IEEE Transactions on* 9, 6 (Nov 2012), 890–902.

X. Luo, J. Zhang, R. Perdisci, and W. Lee. 2010. On the Secrecy of Spread-Spectrum Flow Watermarks. In *Computer Security ESORICS 2010*, Dimitris Gritzalis, Bart Preneel, and Marianthi Theoharidou (Eds.). Lecture Notes in Computer Science, Vol. 6345. Springer Berlin Heidelberg, 232–248.

X. Luo, P. Zhou, J. Zhang, R. Perdisci, W. Lee, and R.K.C. Chang. 2011. Exposing Invisible Timing-based Traffic Watermarks with BACKLIT. In *Proceedings of the 27th Annual Computer Security Applications Conference (ACSAC '11)*. ACM, New York, NY, USA, 197–206.

R. Martin, J. Demme, and S. Sethumadhavan. 2012. TimeWarp: Rethinking timekeeping and performance monitoring mechanisms to mitigate side-channel attacks. In *Computer Architecture (ISCA), 2012 39th Annual International Symposium on*. 118–129.

B.C. Mason, D. Ghosal, and C. Corbett. 2010. Evaluation of a Massively Parallel Architecture for Network Security Applications. In *Parallel, Distributed and Network-Based Processing (PDP), 2010 18th Euromicro International Conference on*. 85–91.

P. Momcilovic. 2006. Mismatch Decoding of a Compound Timing Channel. In *Forty-Fourth Annual Allerton Conference on Communication, Control, and Computing*.

I.S. Moskowitz and A.R. Miller. 1992. The channel capacity of a certain noisy timing channel. *IEEE Transactions on Information Theory* 38, 4 (1992), 1339–1344.

S. Mou, Z. Zhao, S. Jiang, Z. Wu, and J. Zhu. 2012. Feature extraction and classification algorithm for detecting complex covert timing channel. *Computers and Security* 31, 1 (2012), 70–82.

K. Mowery, S. Keelveedhi, and H. Shacham. 2012. Are AES x86 Cache Timing Attacks Still Feasible?. In *Proceedings of the 2012 ACM Workshop on Cloud Computing Security Workshop (CCSW '12)*. ACM, New York, NY, USA, 19–24.

J. Oberg, W. Hu, A. Irturk, M. Tiwari, T. Sherwood, and R. Kastner. 2010. Theoretical Analysis of Gate Level Information Flow Tracking. In *Proceedings of the 47th Design Automation Conference (DAC '10)*. ACM, New York, NY, USA, 244–247.

J. Oberg, W. Hu, A. Irturk, M. Tiwari, T. Sherwood, and R. Kastner. 2011. Information Flow Isolation in I2C and USB. In *Proceedings of the 48th Design Automation Conference (DAC '11)*. ACM, New York, NY, USA, 254–259.

D. Osvik, A. Shamir, and E. Tromer. 2006. Cache Attacks and Countermeasures: The Case of AES. In *Topics in Cryptology CT-RSA 2006*, David Pointcheval (Ed.). Lecture Notes in Computer Science, Vol. 3860. Springer Berlin Heidelberg, 1–20.

Z. Pan, H. Peng, X. Long, C. Zhang, and Y. Wu. 2009. A Watermarking-Based Host Correlation Detection Scheme. In *Management of e-Commerce and e-Government, 2009. ICMECG '09. International Conference on*. 493–497.

P. Peng, P. Ning, and D.S. Reeves. 2006. On the Secrecy of Timing-Based Active Watermarking Trace-Back Techniques. In *SP '06: Proceedings of the 2006 IEEE Symposium on Security and Privacy*. Washington, DC, 334–349.

P. Peng, P. Ning, D.S. Reeves, and X. Wang. 2005. Active timing-based correlation of perturbed traffic flows with chaff packets. In *Distributed Computing Systems Workshops, 2005. 25th IEEE International Conference on*. 107–113.

C. Percival. 2005. Cache missing for fun and profit. In *Proc. of BSDCan 2005*.

Y.J. Pyun, Y. Park, D.S. Reeves, X. Wang, and P. Ning. 2012. Interval-based flow watermarking for tracing interactive traffic. *Computer Networks* 56, 5 (2012), 1646 – 1665.

Y.J. Pyun, Y.H. Park, X. Wang, D.S. Reeves, and P. Ning. 2007. Tracing Traffic through Intermediate Hosts that Repacketize Flows. In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*. 634–642.

S.V. Radhakrishnan, A.S. Uluagac, and R. Beyah. 2013. Realizing an 802.11-based covert timing channel using off-the-shelf wireless cards. In *Global Communications Conference (GLOBECOM), 2013 IEEE*. 722–728.

C. Rebeiro, M. Mondal, and D. Mukhopadhyay. 2010. Pinpointing Cache Timing Attacks on AES. In *VLSI Design, 2010. VLSID '10. 23rd International Conference on*. 306–311.

A.R. Sadeghi, S. Schulz, and V. Varadharajan. 2012. The Silence of the LANs: Efficient Leakage Resilience for IPsec VPNs. In *Computer Security ESORICS 2012*, Sara Foresti, Moti Yung, and Fabio Martinelli (Eds.). Lecture Notes in Computer Science, Vol. 7459. Springer Berlin Heidelberg, 253–270.

E. Savaş. 2013. Attacks on Implementations of Cryptographic Algorithms: Side-channel and Fault Attacks. In *Proceedings of the 6th International Conference on Security of Information and Networks (SIN '13)*. ACM, New York, NY, USA, 7–14.

W. Schindler. 2000. A Timing Attack against RSA with the Chinese Remainder Theorem. In *Cryptographic Hardware and Embedded Systems CHES 2000*, etinK. Ko and Christof Paar (Eds.). Lecture Notes in Computer Science, Vol. 1965. Springer Berlin Heidelberg, 109–124.

W. Schindler. 2002. A Combined Timing and Power Attack. In *Public Key Cryptography*, David Naccache and Pascal Paillier (Eds.). Lecture Notes in Computer Science, Vol. 2274. Springer Berlin Heidelberg, 263–279.

S.H. Sellke, N.B. Shroff, S. Bagchi, , and C.C. Wang. 2006. Timing Channel Capacity for Uniform and Gaussian Servers. In *Forty-Fourth Annual Allerton Conference on Communication, Control, and Computing*.

S.H. Sellke, C. Wang, S. Bagchi, and N. Shroff. 2009. TCP/IP Timing Channels: Theory to Implementation. In *INFOCOM 2009. The 28th Conference on Computer Communications. IEEE*. 2204–2212.

S.H. Sellke, C. Wang, N. Shroff, and S. Bagchi. 2007. Capacity Bounds on Timing Channels with Bounded Service Times. In *IEEE International Symposium on Information Theory*. 981–985.

A. Shoufan, F. Strenzke, H.G. Molter, and M. Stottinger. 2010. A Timing Attack against Patterson Algorithm in the McEliece PKC. In *Information, Security and Cryptology ICISC 2009*, Donghoon Lee and Seokhie Hong (Eds.). Lecture Notes in Computer Science, Vol. 5984. Springer Berlin Heidelberg, 161–175.

P.L. Shrestha, M. Hempel, M. Alahmad, and H. Sharif. 2013. Modeling packet rate covert timing channels. In *Innovations in Information Technology (IIT), 2013 9th International Conference on*. 54–59.

P. L. Shrestha, M. Hempel, H. Sharif, and H. H. Chen. 2016. An Event-Based Unified System Model to Characterize and Evaluate Timing Covert Channels. *IEEE Systems Journal* 10, 1 (March 2016), 271–280.

G.J. Simmons. 1984. The Prisoners Problem and the Subliminal Channel. In *Advances in Cryptology*, David Chaum (Ed.). Springer US, 51–67.

D.X. Song, D. Wagner, and X. Tian. 2001. Timing analysis of keystrokes and timing attacks on SSH. In *SSYM'01: Proceedings of the 10th Conference on USENIX Security Symposium*. Berkeley, CA, USA, 25–25.

R.M. Stillman. 2008. Detecting IP covert timing channels by correlating packet timing with memory content. In *Southeastcon, 2008. IEEE*. 204–209.

F. Strenzke. 2010. A Timing Attack against the Secret Permutation in the McEliece PKC. In *Post-Quantum Cryptography*, Nicolas Sendrier (Ed.). Lecture Notes in Computer Science, Vol. 6061. Springer Berlin Heidelberg, 95–107.

F. Strenzke, E. Tews, H.G. Molter, R. Overbeck, and A. Shoufan. 2008. Side Channels in the McEliece PKC. In *Post-Quantum Cryptography*, Johannes Buchmann and Jintai Ding (Eds.). Lecture Notes in Computer Science, Vol. 5299. Springer Berlin Heidelberg, 216–229.

Y. Sun, X. Guan, T. Liu, and Y. Qu. 2012. An Identity Authentication Mechanism Based on Timing Covert Channel. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on*. 832–836.

R. Sundaresan and S. Verdu. 2000a. Robust decoding for timing channels. *Information Theory, IEEE Transactions on* 46, 2 (Mar 2000), 405–419.

R. Sundaresan and S. Verdu. 2000b. Sequential decoding for the exponential server timing channel. *Information Theory, IEEE Transactions on* 46, 2 (Mar 2000), 705–709.

R. Sundaresan and S. Verdu. 2006. Capacity of queues via point-process channels. *Information Theory, IEEE Transactions on* 52, 6 (June 2006), 2697–2709.

J.A. Thomas. 1997. On the Shannon capacity of discrete time queues. In *Information Theory. 1997. Proceedings., 1997 IEEE International Symposium on*. 333–.

M. Tiwari, X. Li, H.M.G. Wassel, B. Mazloom, S. Mysore, F.T. Chong, and T. Sherwood. 2010. Gate-Level Information-Flow Tracking for Secure Architectures. *Micro, IEEE* 30, 1 (Jan 2010), 92–100.

M. Tiwari, J.K. Oberg, X. Li, J. Valamehr, T. Levin, B. Hardekopf, R. Kastner, F.T. Chong, and T. Sherwood. 2011. Crafting a usable microkernel, processor, and I/O system with strict and provable information flow security. In *Computer Architecture (ISCA), 2011 38th Annual International Symposium on*. 189–199.

M. Tiwari, H.M.G. Wassel, B. Mazloom, S. Mysore, F.T. Chong, and T. Sherwood. 2009. Complete Information Flow Tracking from the Gates Up. In *Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS XIV)*. ACM, New York, NY, USA, 109–120.

E. Tromer, D.A. Osvik, and A. Shamir. 2010. Efficient Cache Attacks on AES, and Countermeasures. *J. Cryptol.* 23, 2 (Jan. 2010), 37–71.

A.B. Wagner and V. Anantharam. 2005. Zero-rate reliability of the exponential-server timing channel. *Information Theory, IEEE Transactions on* 51, 2 (Feb 2005), 447–465.

R.J. Walls, K. Kothari, and M. Wright. 2011. Liquid: A detection-resistant covert timing channel based on IPD shaping. *Computer Networks* 55, 6 (2011), 1217 – 1228.

C.D. Walter and S. Thompson. 2001. Distinguishing Exponent Digits by Observing Modular Subtractions. In *Topics in Cryptology CT-RSA 2001*, David Naccache (Ed.). Lecture Notes in Computer Science, Vol. 2020. Springer Berlin Heidelberg, 192–207.

X. Wang, S. Chen, and S. Jajodia. 2005. Tracking Anonymous Peer-to-peer VoIP Calls on the Internet. In *Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS '05)*. ACM, New York, NY, USA, 81–91.

X. Wang, S. Chen, and S. Jajodia. 2007. Network Flow Watermarking Attack on Low-Latency Anonymous Communication Systems. In *Security and Privacy, 2007. SP '07. IEEE Symposium on*. 116–130.

X. Wang, J. Luo, and M. Yang. 2009. An interval centroid based spread spectrum watermark for tracing multiple network flows. In *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*. 4000–4006.

X. Wang, J. Luo, and M. Yang. 2010. A Double Interval Centroid-Based Watermark for network flow traceback. In *Computer Supported Cooperative Work in Design (CSCWD), 2010 14th International Conference on*. 146–151.

X. Wang, J. Luo, and M. Yang. 2012. An efficient sequential watermark detection model for tracing network attack flows. In *Computer Supported Cooperative Work in Design (CSCWD), 2012 IEEE 16th International Conference on*. 236–243.

X. Wang and D.S. Reeves. 2003. Robust Correlation of Encrypted Attack Traffic Through Stepping Stones by Manipulation of Interpacket Delays. In *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS '03)*. ACM, New York, NY, USA, 20–29.

X. Wang and D.S. Reeves. 2011. Robust Correlation of Encrypted Attack Traffic through Stepping Stones by Flow Watermarking. *Dependable and Secure Computing, IEEE Transactions on* 8, 3 (May 2011), 434–449.

X. Wang, M. Yang, and J. Luo. 2013. A novel sequential watermark detection model for efficient traceback of secret network attack flows. *Journal of Network and Computer Applications* 36, 6 (2013), 1660 – 1670.

Y. Wang, P. Chen, Y. Ge, B. Mao, and L. Xie. 2009. Traffic Controller: A Practical Approach to Block Network Covert Timing Channel. In *Availability, Reliability and Security, 2009. ARES '09. International Conference on*. 349–354.

Y. Wang, A. Ferraiuolo, and G.E. Suh. 2014. Timing channel protection for a shared memory controller. In *High Performance Computer Architecture (HPCA), 2014 IEEE 20th International Symposium on*. 225–236.

Y. Wang and P. Moulin. 2008. Perfectly Secure Steganography: Capacity, Error Exponents, and Code Constructions. *Information Theory, IEEE Transactions on* 54, 6 (June 2008), 2706–2722.

Y. Wang and G.E. Suh. 2012. Efficient Timing Channel Protection for On-Chip Networks. In *Networks on Chip (NoCS), 2012 Sixth IEEE/ACM International Symposium on*. 142–151.

Z. Wang and R.B. Lee. 2006. Covert and Side Channels Due to Processor Architecture. In *Computer Security Applications Conference, 2006. ACSAC '06. 22nd Annual*. 473–482.

Z. Wang and R.B. Lee. 2007. New Cache Designs for Thwarting Software Cache-based Side Channel Attacks. In *Proceedings of the 34th Annual International Symposium on Computer Architecture (ISCA '07)*. ACM, New York, NY, USA, 494–505.

Z. Wang and R.B. Lee. 2008. A novel cache architecture with enhanced performance and security. In *Microarchitecture, 2008. MICRO-41. 2008 41st IEEE/ACM International Symposium on*. 83–93.

H.M.G. Wassel, Y. Gao, J.K. Oberg, T. Huffmire, R. Kastner, F.T. Chong, and T. Sherwood. 2013. SurfNoC: A Low Latency and Provably Non-interfering Approach to Secure Networks-on-chip. In *Proceedings of the 40th Annual International Symposium on Computer Architecture (ISCA '13)*. ACM, New York, NY, USA, 583–594.

J.C. Wray. 1991. An analysis of covert timing channels. In *Research in Security and Privacy, 1991. Proceedings., 1991 IEEE Computer Society Symposium on*. 2 –7.

J. Wu, Y. Wang, L. Ding, and X. Liao. 2012. Improving performance of network covert timing channel through Huffman coding. *Mathematical and Computer Modelling* 55, 1–2 (2012), 69–79.

Z. Xinjie, W. Tao, M. Dong, Z. Yuanyuan, and L. Zhaoyang. 2008. Robust First Two Rounds Access Driven Cache Timing Attack on AES. In *Computer Science and Software Engineering, 2008 International Conference on*, Vol. 3. 785–788.

L. Yao, X. Zi, L. Pan, and J. Li. 2009. A study of on/off timing channel based on packet delay distribution. In *Computers & Security*.

W. Yu, X. Fu, S. Graham, D. Xuan, and W. Zhao. 2007. DSSS-Based Flow Marking Technique for Invisible Traceback. In *Security and Privacy, 2007. SP '07. IEEE Symposium on*. 18–32.

M. Yue, W.H. Robinson, L. Watkins, and C. Corbett. 2014. Constructing Timing-based Covert Channels in Mobile Networks by Adjusting CPU Frequency. In *Proceedings of the Third Workshop on Hardware and Architectural Support for Security and Privacy (HASP '14)*. ACM, New York, NY, USA, Article 2, 8 pages.

S. Zander, G. Armitage, and P. Branch. 2007. A Survey of Covert Channels and Countermeasures in Computer Network Protocols. *Communications Surveys & Tutorials, IEEE* 9, 3 (2007), 44–57.

S. Zander, G. Armitage, and P. Branch. 2011. Stealthier Inter-packet Timing Covert Channels. In *NETWORKING 2011*, Jordi Domingo-Pascual, Pietro Manzoni, Sergio Palazzo, Ana Pont, and Caterina Scoglio (Eds.). Lecture Notes in Computer Science, Vol. 6640. Springer Berlin Heidelberg, 458–470.

L. Zhang, J. Luo, and M. Yang. 2009. An Improved DSSS-Based Flow Marking Technique for Anonymous Communication Traceback. In *Ubiquitous, Autonomic and Trusted Computing, 2009. UIC-ATC '09. Symposia and Workshops on*. 563–567.

L. Zhang, Z. Wang, Q. Wang, and F. Miao. 2010b. MSAC and Multi-flow Attacks Resistant Spread Spectrum Watermarks for network flows. In *Information and Financial Engineering (ICIFE), 2010 2nd IEEE International Conference on*. 438–441.

L. Zhang, Z. Wang, Y. Wang, and H. Liu. 2010a. Interval-Based Spread Spectrum Watermarks for tracing multiple network flows. In *Communication Technology (ICCT), 2010 12th IEEE International Conference on*. 393–396.

Y. Zhang and V. Paxson. 2000. Detecting Stepping Stones. In *Proceedings of the 9th Conference on USENIX Security Symposium - Volume 9 (SSYM'00)*. USENIX Association, Berkeley, CA, USA, 1–15.

X. Zi, L. Yao, X. Jiang, L. Pan, and J. Li. 2011. Evaluating the transmission rate of covert timing channels in a network. *Computer Networks* 55, 12 (2011), 2760 – 2771.

X. Zi, L. Yao, L. Pan, and J. Li. 2010. Implementing a passive network covert timing channel. *Computers and Security* 29, 6 (2010), 686 – 696.

APPENDIX

A. AN EXAMPLE OF TIME-REPLAY CTCC

Let us assume that Alice and Bob communicate using a time-replay covert channel where Alice is the sender. Let us also assume that Alice transfers a code C that has symbols s_1 and s_2. Alice divides a prerecorded event sequence into two partitions denoted by p_1 and p_2. Next, the p_1-partition is used to get a timing data τ_{s1} for sending symbol s_1. The symbol is embedded by delaying an event for τ_{s1} time. Similar procedure is followed to send symbol s_2 using a timing value from the p_2-partition. A specific timing value is not used more than once. At the receiver, Eve i) observes the time τ between events, ii) decides the exact partition that contains τ, and iii) decodes the symbol s_1 or s_2 based on the result of second step.

B. MODEL BASED CTCC FRAMEWORK

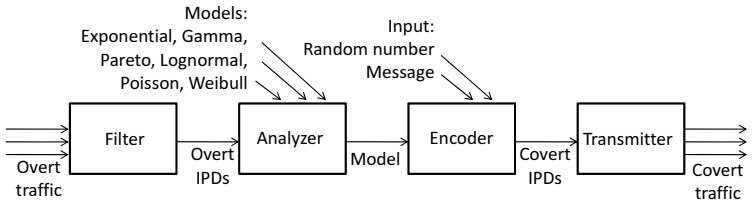


Fig. 7. A framework for implementing model-based CTCC (sender side) [Gianvecchio et al. 2008].

The model based CTCC framework consists of filter, analyzer, encoder, and transmitter as shown in Figure 7. The filter first extracts specific type of traffic from legitimate traffic and then measures the IPDs from each flow. Next the filter results are forwarded to the analyzer that can decide the best traffic model. The analyzer tries to fit the legitimate traffic with the Weibull, Poisson, Lognormal, Pareto, Gamma, and Exponential distributions. Both offline and online execution is possible for the filter and analyzer stages. Both the online and the offline modes has their advantages and disadvantages. Fewer resources (like computing and storage) are required in the offline mode, but the actual network traffic model may be different. The online mode requires more resources and a startup protocol to send the online model and parameters to the decoder, but the live network flow is closely approximated by the model. The encoder generates IPDs based on the model, a random number series, and a covert message. Then the transmitter sends packets according to the generated IPDs. The generated covert traffic approximates the legitimate traffic distribution to achieve non-detectability.

C. COMPRESSIBILITY METRIC

Let $S \in \Sigma^s$ is a string to be compressed with Σ denoting the alphabet of symbols, and the length of S is s. Let $C \in \Sigma^c$ denotes the final compressed string. The compressibility of S is defined in [Cabuk et al. 2009] as

$$\kappa(S) = \frac{|S|}{|C|} \tag{1}$$

where $\kappa(.)$ denotes the compressibility operator, and $|.|$ denotes the length operator. It is reported that the compressibility generated by the overt channel IPDs is less than that generated by noiseless IP CTCC IPDs. It is to be noted that IPDs are numerical values and strings only can be compressed. So IPDs are converted to strings for compression. Cabuk et al. have considered the two significant non-zero digits of an IPD value for conversion and they have used a letter before the digits [2009]. This letter is decided

by counting the zeros after the decimal point. Following the procedure 0.00247 becomes B25, and 0.0247 becomes A25.

The CTCC sender may insert noise to make the regularity metric comparable to legitimate channel. For this special case, Two detection methods are proposed in [Cabuk et al. 2009] called "compressibility-walk" and "CosR-walk". They have shown that the two methods are appropriate for offline measurement because of their high algorithmic complexity. In compressibility-walk a mixed data set is considered whose size is s and compressibility scores are calculated for every w size window. These windows are overlapped if $s < w$. If a window contains IPDs embedded with covert message, it generates higher compressibility scores compared to overt IPDs. In the case of CosR-walk, CosR scores are calculated instead of compressibility scores for consecutive windows. Again, windows can be overlapped. CosR score is lower when a window contains overt IPDs and the other window contains IPDs with covert message compared to the score when both windows contain overt IPDs. CosR metric between two strings S and T is defined as [Cabuk et al. 2009]

$$CosR(S,T) = 1 - \frac{\Im(S) + \Im(T) - \Im(S|T)}{\sqrt{\Im(S)\Im(T)}} \tag{2}$$

where $\Im$ is a compressor and $|$ is a concatenation operator.

D. VARIOUS FLOW TRANSFORMATION TECHNIQUES

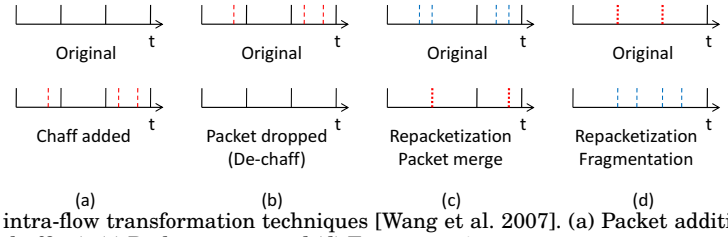


Fig. 8. Various intra-flow transformation techniques [Wang et al. 2007]. (a) Packet addition (chaffing), (b) Packet drop (de-chaffing), (c) Packet merge, and (d) Fragmentation.

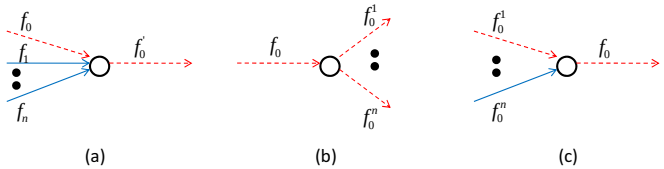


Fig. 9. Various inter-flow transformation techniques [Wang et al. 2007]. (a) Flow mixing, (b) Flow splitting, and (d) Flow merging.

Figure 8 shows various intra-flow transformation techniques. Figure 8(a) shows addition of chaff packet to the flow. Figure 8(b) shows packet dropping. Figure 8(c) and (d) shows repacketization techniques i.e., packet merging and fragmentation, respectively. Figure 9 shows various inter-flow transformation techniques. Figure 9(a) shows flow mixing where a mixed flow f_0' is created by mixing a flow f_0 with separate flows $f_1, ..., f_n$. Figure 9(b) shows flow splitting where subflows $f_0^1, ..., f_0^n$ are created by splitting a flow f_0. These split subflows can again be merged as in Figure 9(c).

E. CENTROID OF AN INTERVAL

The following definition of the centroid of an interval is adopted from [Wang et al. 2007]. We assume that there is a flow of duration $T_f > 0$ and l-bit watermark is embedded in the flow with redundancy $r > 0$. We also assume that a time period T_d can be divided among $2n$ (here $n = r \times l$) periods $I_0, ..., I_{2n-1}$ each having a positive duration T. We consider that the $2n$ periods contain packets $P_1, ..., P_{n_p}$ (total n_p packets) and $t_i (i = 1, ..., n_p)$ is the packet P_i's time instance. Next $t'_i = t_i - t_0$ represents P_i's time instance relative to the initial period t_0. Next the relative position of P_i within its period (i.e., the difference relative to beginning of its period) is calculated by Δt_i

$$\Delta t_i = t'_i \bmod T \tag{3}$$

If period $I_i (i = 0, ..., 2n-1)$ contains packets $P_{i_0}, ..., P_{i_{n_i-1}}$ (total n_i packets), then the "centroid of interval[3]" $I_i (i = 0, ..., 2n-1)$ is defined as

$$Cent(I_i) = \frac{1}{n_i} \sum_{j=0}^{n_i-1} \Delta t_{i_j} \tag{4}$$

F. DSSS BASED NETWORK FLOW WATERMARKING SYSTEM

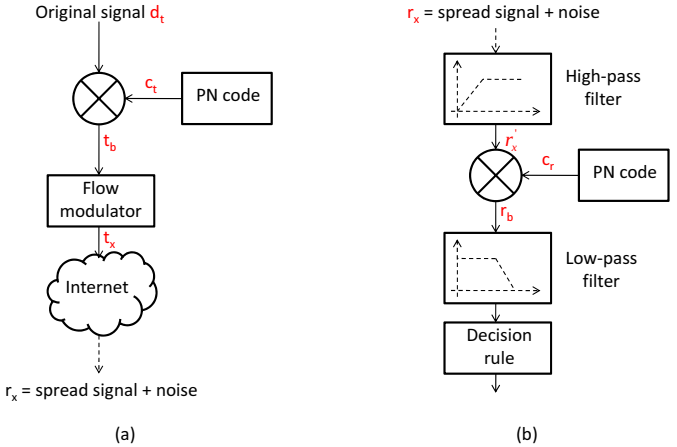


Fig. 10. (a) Transmitter side DSSS based mark creation module, and (b) Receiver side DSSS based mark identification module. [Yu et al. 2007]

Figure 10(a) shows the transmitter side DSSS based mark creation module and Figure 10(b) shows the receiver side DSSS based mark identification module as presented in [Yu et al. 2007]. The transmitter generates "+1" (logical bit 1) or "-1" (logical bit 0) of signal d_t. Next, a PN code c_t having T_c chip duration (a chip is a PN code bit) is multiplied with the original signal d_t to obtain baseband signal t_b.

$$t_b = d_t c_t \tag{5}$$

The baseband signal t_b is then passed to the flow modulator for modulating the flow. For +1 chip, the flow traffic rate D is increased by applying weak interference against the flow for T_c seconds and obtain high rate (D + A). Similarly, for -1 chip, strong interference is applied and a low traffic rate of (D - A) is obtained for T_c seconds. Here

[3]The terms interval and period are used interchangeably.

A denotes the mark amplitude. The transmitted signal t_x from the flow modulator can be expressed as

$$t_x = Ad_t c_t + D \tag{6}$$

The transmitted signal gets affected by network noise which is generated due to intentional noise injection or due to interfering traffic. If we denote noise by w, the received signal r_x is given by

$$r_x = Ad_t c_t + D + w \tag{7}$$

The received signal r_x is passed through a high-pass filter to remove the DC component D which yields

$$r'_x \approx Ad_t c_t + w \tag{8}$$

The filtered signal r'_x is multiplied with the PN code c_r to despread and to derive the baseband signal r_b which is given by

$$r_b = Ad_t c_t c_r + wc_r \tag{9}$$

Note that the PN code c_r is identical to the PN code used in the transmitter. The baseband signal r_b is then passed through a low-pass filter to get the low frequency signal. Next a decision rule helps to interpret the transmitted information from the filtered signal.