A Defense Mechanism for Timing-based Side-Channel Attacks on IoT Traffic

Nelson Prates*, **Andressa Vergütz***, **Ricardo T. Macedo**†**, Aldri Santos***, **Michele Nogueira***
*NR2/CCSC - Federal University of Paraná, Brazil
†Dept. of Information Technology - Federal University of Santa Maria (Frederico Westphalen), Brazil
Emails: {ngpjunior, avergutz, aldri, michele}@inf.ufpr.br, rmacedo@inf.ufsm.br

GLOBECOM 2020 - 2020 IEEE Global Communications Conference | 978-1-7281-8298-8/20/$31.00 ©2020 IEEE | DOI: 10.1109/GLOBECOM42002.2020.9322070

***Abstract*—This work proposes FISHER: a deFense mechanIsm against timing-based Side-cHannel attack related to response time on the intERnet of things (IoT). IoT connects objects that support important applications, such as electronic health, smart homes, and Industry 4.0. However, timing-based side-channel attacks on IoT network traffic compromise user privacy. Related works present a limited view of side-channel leakages and as a solution, these works try to mask them. However, they ignore that devices have unique behaviors that intensify the problem of privacy leaks through response time. Hence, FISHER follows two modules: (i) vulnerability test and (ii) privacy protection. The vulnerability test module identifies timing-based side-channel leakages and reveals new vulnerabilities associated with the response time. The privacy protection module implements two methods that mask the identified time-based leakages on the network traffic. Results from an experimental scenario show that FISHER identifies precisely the side-channel leakages related to response time and efficiently masks them.**

***Index Terms*—IoT, Side-Channel Attacks, Network Traffic**

I. Introduction

Advances in communication technology and MultiProcessing Systems-on-Chips promote the connection of an ever-growing number of devices to the Internet and the Internet of Things (IoT) [1]. Such devices comprise multiple embedded processors and sensors, providing smart services and generating big data in real-time. IoT protocols, such as the Constrained Application Protocol (CoAP) and the IPv6 over Low power Wireless Personal Area Networks (6LoWPAN), motivate new services and create new resources to support smart cities. IoT attracts attention to new business markets, given its potential to understand human behaviors, personal interests, consumer intentions, among other issues.

However, side-channel attacks on IoT jeopardize user data privacy. These attacks employ sniffers to snoop wireless network traffic, probe side-channel leakages (e.g., packet length and headers) [2], and reveal sensitive information. In timing-based side-channel attacks, the attacker exploits traffic interference as timing channels to identify patterns that correspond to device behavior and infer sensitive information about users [3]. This attack infers confidential information analyzing the similarities between time intervals and response time, such as the type of the device in use, the embedded sensors, and how users interact with them. Crossing this information with human behavior, the attacker infers private user information (e.g., how many people live in a smart home and the number of rooms of the house [3], [4]).

In the literature, there are works to identify and mask the timing-based side-channel leaks on IoT traffic. Such works address device reactive behavior (e.g., motion sensors) and only mask timestamps [3], [5]–[7]. The most efficient method to hide timing-based leakages lies in transmitting fake packets. This method generates fake packets to keep the frequency of traffic and to hide the original traffic patterns from attack. However, this is not enough for passive behavior because there are crucial leaks in response time about the device embedded sensors. These leaks jeopardize user privacy. Hence, it is essential to design a new defense mechanism to mask timing-based leaks efficiently.

This paper presents FISHER, a defense mechanism against timing-based side-channel attacks on IoT network traffic. FISHER prevents this attack identifying and hiding timing leakages, improving user privacy. FISHER follows two modules, the vulnerability test and the privacy protection. The first module collects IoT network traffic, and it extracts the timestamp related to the request and response messages exchanged with the gateway. This module computes statistical measures from the collected network traffic to identify timing-based leakages. The second module defines rules for IoT devices to hide their behavior and possible leaks through two methods. Such methods insert delays and generate fake packets to force devices acting similarly in the same time instants.

The experimental evaluation conducted on FIT-IoT Lab showed the efficiency of FISHER vulnerability test and privacy protection modules [8]. We evaluate the modules on classifying the timing-based side-channel leakages and compare results from two defense approaches. We employed machine learning algorithms to classify IoT traffic. Results showed that there are timing-based side-channel leaks on IoT devices traffic since the vulnerability test module efficiently identifies similar sensors embedded in different devices. Furthermore, the privacy protection module masked the side-channel leakages efficiently by reducing identification performance.

This paper proceeds as follows. Section II overviews related works. Section III details the FISHER mechanism. Section IV describes the performance evaluation methodology and Section V discusses results. Section VI concludes the paper.

This work was supported by CAPES under grants #88882.381944/2019-01 and #88882.461738/2019-01, and CNPq under grants #309129/2017-6 and #432204/2018-0.

II. Related Work

In the literature, the timing-based side-channel attacks on network traffic follow different terminologies, such as traffic classification attacks [9], traffic behavior-based inference [10], or Fingerprint And Timing-based Snooping (FATS) attack [3]. Side-channel information leaks are well-known for disclosing insights about the communications. Since the targeted attack exploits wireless communication timestamps [3], [5], [6], [9], we called it as Timing-based Side-Channel Attack on network traffic. In the IoT context, several works addressed timing-based side-channel attacks to protect user privacy, but ignoring the differences in the device behaviors.

A device presents a reactive or passive behavior. In the reactive behavior, the device monitors and shares the detected environment changes (e.g., motion sensor). In contrast, in passive behavior, sensors activate only when they receive a request message (e.g., a periodic temperature reading). In [3], [5], [6], the authors revealed a significant privacy issue by analyzing only timestamps to infer the Activity of Daily Living (ADL) of smart home users. However, they ignored crucial leaks in response time about the device embedded sensors. Differently, this work shows that response time reveals, in passive behavior, information about specific sensors endangering the privacy of several IoT applications.

The most efficient method to hide timing-based leakages lies in transmitting fake packets [2], [3], [5]–[7], [9]. This method generates dummy traffic to keep the frequency of sending packets and to hide the original traffic patterns from attack, making it challenging the device identification. Hence, the studies shaped the timestamp distributions by observing the real network behavior, to then mask the timing leaks. However, response time also enables device identification, and the works ignore it. Thus, different from the literature, we show that an attacker identifies similar sensors embedded in different devices with passive behavior. Moreover, we propose a mechanism to obfuscate, device and sensor, timing-based side-channel leaks by inserting delays and adding fake packets in the network traffic, protecting user privacy.

III. A Defense Mechanism Against Timing-Based Side-Channel Attack on IoT Traffic

This section details FISHER, a deFense mechanIsm against timing-based Side-cHannel attack related to response time on the intERnet of things. It follows the vulnerability test and privacy protection modules. The next subsections describe the network and attack models (following the terminology in Table I), and the proposed mechanism.

A. Network and Attack Models

The design of FISHER assumes a personal area wireless network, according to the IEEE 802.15.4 standard under the 6LoWPAN and an UDP based CoAP protocol. A graph $\vartheta(N, L)$ represents this personal area network, where the set of devices is $N = \{G\} \cup \{D\}$, and the set of links $L = \{l_1, l_2, l_3, ..., l_{|D|}\}$ (as shown in Fig. 1). The gateway G is the border router and a network coordinator device (sink) that intermediates the connection between devices and the Internet. Each link $l \in L$ represents a direct communication between gateway G and a device d. Thus, the set of devices N has subsets G and D, where $|G| = 1$ and $|D| \geq 1$.

Symbol	Value
ϑ	IoT network
G	Gateway
D	Subset of devices d
L	Set of links l
$Traf$	Network traffic
req	Requests
$resp$	Responses
Tr	Transmissions
M	Statistical measures set
p_ν	Response time distribution
$\mathcal{P}$	Distribution family
P_{p_ν}	Probability *a priori* of $Tr \sim p_\nu$
A	Timestamp T and response time τ samples set
X	Response time variable
M	Metrics mean
$threshold$	Vulnerable devices threshold
V	Traffic observed by the attacker

TABLE I: Terminology

The gateway G sends all requests messages to devices $d \in D$ that, in turn, reply with response messages. The requests require environmental data. As each device has a set of embedded sensors $s \in S$, the device reads its embedded sensor s, and the response message payload lies in the sensed data. These messages transmissions $Tr = (req_i, resp_i)$ are independent and identically distributed (i.i.d.) with distribution $p_\nu(Tr)$ on network traffic $Traf$. The network traffic per device $Traf^{d_i} = \{Tr_1, Tr_2, ..., Tr_n\} \in Traf$, where all req_i and $resp_i$ have timestamps of a sent request T_{req} and a received response T_{resp}, respectively.

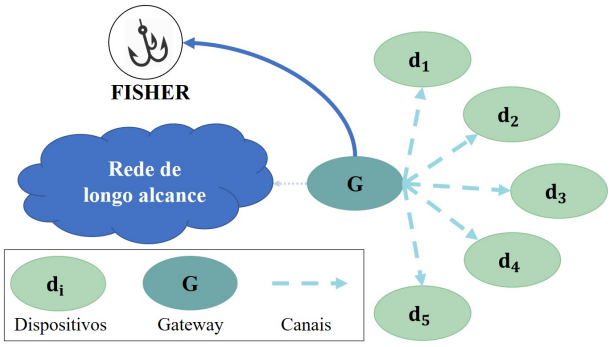

Fig. 1: IoT Network

We assume non-invasive timing-based side-channel attacks on network traffic model following [11]. The attacker only observes network traffic to classify the Timestamps T and response times τ statistical distributions computed by p_ν. Thus, attackers capture the traffic timestamps on a set of samples $V = \{T_1, T_2, ..., T_x\}$ of x transmissions and they seek to classify the differences between each p_ν by identifying patterns. These distributions p_ν come from a possible family of density functions $\mathcal{P} = \{p_\nu\}_{\nu=1}^{s}$, where we denote the

probability *a priori* of a d_i device has sent messages according to a given p_ν, such as $P_{p_\nu}(V) \triangleq P(a_k \sim p_\nu)$. Each $\mathcal{P}$ represents IoT device types and ν represents a given device d_i; or $\mathcal{P}$ represents a set of operating states, and ν the current operating state. Attackers, by unsupervised and supervised classifiers, model a given p_ν. Unsupervised classifiers identify the possible sets of $Traf^d \in V$, i.e., the probability of belonging to a given d_i. Supervised classifier requires samples to be labeled. Hence, they use some attributes through any previously known ν and set of sampled transmissions Tr, linked to some d_i that transmitted messages in V to train the supervised classifier and identify the probability $P_{p_\nu}(Tr_x)$ of each transmission Tr_x came from a distribution p_ν.

B. The mechanism

FISHER operates as a virtual service running into the gateway to observe the traffic per device, identify vulnerable behaviors, and mask them. Therefore, FISHER employs supervised classifiers to model device behaviors through response time side-channel leaks on IoT traffic and identify vulnerable devices. Then, this employs privacy protection methods to generate new indistinguishable behavior. Thus, FISHER acts through cycles of vulnerability testes, privacy protection, and testing of protections following two modules: vulnerability test and the privacy protection, as illustrated in Fig. 2.

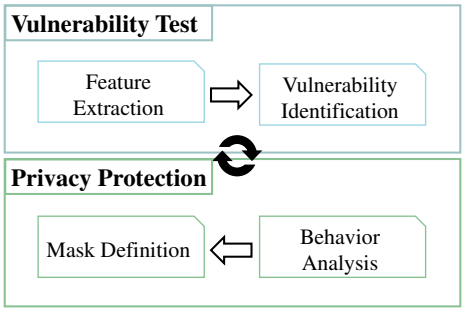

Fig. 2: The FISHER Mechanism Architecture

1) Vulnerability Test Module: This module searches for side-channel leaks per device on network traffic. It collects network traffic, extracts timestamps, and computes samples to train and test a supervised classifier. The module captures the timestamps of traffic per device and follows two main functions: *feature extraction* and *vulnerability identification*. The *feature extraction function* pre-processes traffic timestamps per device. This function computes the response times (τ) by the difference between the timestamps of receiving a response $T.resp$ and sending a request $T.req$ under the gateway perspective, i.e., $\tau_j^d = resp_j - req_j, (\forall d \in D \quad \& \quad \forall j \in Traf^d)$. Then, it extracts the statistical features (Table II) from $T.req$ and tau for each set of t samples. These statistical features optimizes the classification algorithms used in the next function. Thus, this function returns a set of statistical features samples per device of size $|Traf^d|/t$, composed by device address $a_i.id$, and the statistical features $a.min$, $a.sum$, $a.mean$, $a.upper$, $a.lower$, $a.mode$, and $a.pearson$.

The *vulnerability identification* function receives the samples A^d as input and employs a supervised classifier to identify vulnerable devices $D' \in D$. Hence, each device d represents an input class for the classifier. The samples in A^d are independent and identically distributed (i.i.d.) by the distribution function $p_\nu(A^d)$. This distribution represents the behavior of IoT devices, when exchanging messages, based on response times. The module splits the set A^d into two subsets of train A^d_{train} and test A^d_{test} per device.

Statistical Measure	Equação
Minimum (τ)	$min = min(x_i)$
Sum (τ and T)	$sum = \sum_{i=1}^{N} x_i$
Mean (τ)	$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$
Upper Bound (τ)	$L = \mu + 1.96\sigma$
Lower Bound (τ)	$U = \mu - 1.96\sigma$
Mode (τ and T)	$mode = freq(X)$
Median (τ and T)	$Md = \frac{1}{2}(X_{\frac{N}{2}} + X_{\frac{N}{2}+1})$ of $sort(X)$
Pearson's C. C. (τ and T)	$cp = \frac{\sum_{i=1}^{N}(x_i-\mu_x)(y_i-\mu_y)}{\sqrt{\sum_{i=1}^{N}(x_i-\mu_x)^2 \sum_{i=1}^{N}(y_i-\mu_y)^2}}$

X = data set; x_i = samples i of the data; N = number of samples; σ = variance; $freq$ = most frequent value; $sort$ = ordered values.

TABLE II: Statistical Measures [12]

The classifiers receive the sample set A^d_{train} as input and train a model of p_ν distribution for each class d. The model identifies the probability of each a_i belonging to a certain class of devices d from the test entries A^d_{test}. The classifiers evaluation consists of the average M of each performance metric (accuracy, precision, recall, and F-Score). Thus, FISHER employs a $threshold$ to classify devices as vulnerable or masked. Each device $d' \in D'$ receives a binary value, where 0 represents a safe device, i.e., there is no leak when compared to other distributions, and 1 a vulnerable device. The rules are applied according to Eq. 1. Finally, if there are vulnerable devices, this module activates the Privacy Protection Module.

$$F3(V^{d_{i+1}}) = \begin{cases} 1 & if \quad M^d \geq threshold \\ 0 & else \end{cases}, (\forall d \in D) \quad (1)$$

2) Privacy Protection Module: This module receives the identified leaks to mask them and to reduce the effectiveness of timing-based side-channel attacks. It assumes that the attacker has a previously trained classifier with a distribution of the $\mathcal{P}$ family, which allows the pre-knowledge of the $P_{p\nu}$ probability. This probability reveals which device or the purpose it is exchanging messages on the desired network. Thus, to modify this identifiable behavior, the module considers two methods, the delay insertion and fake packet methods, to modify devices behavior. The delay insertion method changes the average differences between all response time (τ) distributions per device, i.e., $\tau' = \tau + \gamma$. The second method generates fake packets sending a false request req_f. These fake packets are directed to a real device that implements the delay insertion method as an original req packet and sends a fake response $resp_f$ to change the sending and receiving timestamps.

Denoting the privacy protection, the observation of IoT traffic represents each device behavior defined by the family of distribution functions $\mathcal{P} = \{p_\nu\}_{\nu=1}^{s}$. The sets A^d are a discrete sequence of features where the variable X takes a value of of all pre-determined possible average response times $x \in M = \{m_1, m_2, ..., m_{|M|}\}$. The module needs to modify x in order to masks the original $p_\nu(\chi)$ distribution. Therefore, for each new request, the mechanism manipulates the future behavior defining new timestamps or response times and generates a fake $p_{\nu'}$ distribution from the $p_\nu(A^d)$ not representing a unique distribution feature per device $d \in D$. Finally, this module activates the *Vulnerability Test Module* and verify the masked leaks.

IV. Performance Evaluation Methodology

As the FISHER mechanism aims at identifying and defending against side-channel attacks, the performance evaluation focuses on analyzing timing-based leakage of network traffic. Analyzing and identifying timing side-channel leaks reveal characteristics from network structure and IoT devices. Fig. 3 shows the main steps followed for the analyses.

Fig. 3: Steps of Timing-based Leakage Analysis

The **Experimental Traffic Generation** step implements an IoT infrastructure based on the platform FIT-IoT Lab [8] as a real network to evaluate FISHER. FIT-IoT provides an experimentation interface for IoT devices, each one embedded with lighting, pressure, accelerometer, magnetometer, and gyroscope sensors. Particularly, the experiment infrastructure employs ARM Cortex M3 devices embedded with AT86RF231 chips following the IEEE 802.15.4 specifications [13]. The IoT devices run Contiki-NG operational system and follow IETF standardized network protocols specific for personal area networks, such as 6LoWPAN, RPL, UDP e CoAP. One device acts as a gateway, i.e., it receives and sends external requests and simulates a CoAP client. Moreover, the gateway observes requests and responses by the Wireshark tool. Hence, the gateway is connected by a wireless link with eight IoT devices, each one embedded with five sensors. Each device *Node j* contains five attributes: lighting *Node j-i*, pressure *Node j-p*, accelerometer *Node j-a*, magnetometer *Node j-m*, and gyroscope *Node j-g*.

The configuration of the traffic generation was set empirically and divided into two moments. In the first moment, the normal traffic generation sent requests respecting random intervals between 3 and 10 ms from receiving an answer. In the second moment, the privacy protection module generated the defense traffic. Then, the defense traffic is the gateway sending a fake packet for each request, respecting the random intervals mentioned above. Also, devices started the delay insertion selecting randomly between 0 and 10 ms. This setup enabled the vulnerability test to evaluate more than one side-channel leakage in the same sniffed traffic and detailed privacy protection information.

The **Traffic Sampling** step performs pre-processing of data to optimize input for classifieds. Therefore, by Python3 scripts, FISHER splits the network data collected into 40 subsets (1 subset for each sensor of a given device). Each subset contains 10,000 requests. Afterward, FISHER selects and makes samples of sending timestamps and packet response time. From the subsets, FISHER computes statistical measures to improve the details about the network traffic. The measures involve min, sum, $mean$, $upperbound$, $lowerbound$, $mode$, $median$, and $PearsonCorrelationCoefficient$. As shown in Table II (Section III), the statistical measures receive as input the timestamp values or the response time values, according to [12]. For instance, the min, $mean$, $upperbound$, $lowerbound$ receive as input only timestamp values, while the sum, $mode$, $median$, and Pearson Coefficient receive both timestamp and response time values. The mechanism project defines feature extraction, training the classification algorithms, and activating the defense modules as online activities. However, evaluations of these activities were conducted **offline** to support a detailed analysis of privacy protection.

The generated samples follows two evaluation scenarios C1 and C2. C1 aims to identify different devices, while C2 aims at identifying patterns from similar sensors. Hence, there are some particularities in each scenario. C1 identifies different devices based on all statistical measures (presented in Table II) extracted from both response time and timestamp. From the 40 subsets containing the requests, every 10 requests, FISHER computes the statistical measures and generates 1000 samples. Afterward, each device has a subset of data containing statistical measures and labels. The label is crucial to mark the device class and assist in the IoT device identification since it points out which network traffic belongs to each IoT device. FISHER extracts the label during the network traffic collection in the experimental scenario. Scenario C2 identifies patterns from similar sensors embedded in different devices, considering only the response time as the main network characteristic and the min, sum, $mean$, $upperbound$, and $mode$ as statistical measures. Thus, every 10 requests from the 40 subsets, FISHER computes the measures. Then, each sensor has 1 subset, labeled by the names of the sensors (*i, p, a, m*) containing 8,000 samples of statistical measures.

The *Vulnerability Identification* step analyses the unique behavior from each device by the generated samples. Hence, in the python Scikit-learn data-mine tool [14], the identification deals with three multi-classification machine learning algorithms well known in the literature, K-Nearest Neighbors (KNN), Random Forest (RF), and the neural network Multilayer Perceptron (MLP) [15]. The evaluation was conducted per scenario (C1 and C2). Each algorithm received 70% of the samples by device as a training input and generated a model for each one. Then, to evaluate the models, the algorithms received the remaining 30% of samples by device.

For performance evaluation, we employ as metrics *ac-*

curacy, *precision*, *recall*, and *F-Score* (also known as F-Measure). These metrics consider the true positive rate (TP), true negative rate (TN), false positive (FP), and false negative (FN). Accuracy ($\alpha = (TP+TN)/(TP+TN+FP+FN)$) defines the classified data traffic proportion regarding all samples from traffic. Precision ($p = TP/(TP+FP)$) estimates the true positive rate among all traffic samples classified as true positive. Recall consists in the true positive rate among all samples, where the expected class is positive. Finally, F-Score estimates the relation between precision and recall measures through the harmonic mean estimation. FISHER evaluates these metrics to define if a device is vulnerable.

V. Results

This section presents the results for the FISHER performance evaluation, considering both FISHER modules, the vulnerability test and the privacy protection.

A. Identification of Timing-based Side-Channel Leaks

Results point out timing-based side-channel leaks found in the network traffic from eight IoT devices, each one equipped with five sensors: lighting, pressure, accelerometer, magnetometer, and gyroscope. Fig. 4a presents the behavior of network devices from the C1 scenario and Fig. 4b shows the behavior of sensors in the C2 scenario. In the results, the average response time overlaps, making difficult to visually identify device pattern. Devices present a similar behavior related to response time. However, pressure and gyroscope sensors have greater data variability, which represents a specific behavior when monitoring different phenomena. In general, the average response time per sensor varies between 27.73 ms (*accelerometer*) and 38 ms (pressure). The behaviors, although visually overlapping, when observed in detail, they demonstrate unique characteristics between devices and sensors.

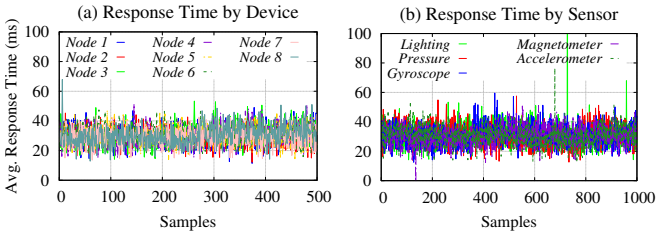

Fig. 4: Devices and Sensors Network Behavior

Figs. 5a and 5b show the device and sensor identification results obtained in both scenarios. In Fig. 5a, KNN and MLP have achieved the worst performance results when compared to RF. The similarities and approximation of data distribution values justify the performance decrease from them (as shown in Fig. 4), because both classifiers consider data proximity in their rules. For instance, KNN identifies the proximity between input data, while MLP separates data based on linear classification. In contrast, RF finds the best characteristics to build a decision tree sequence. It permits then to analyze input data particularities. Nonetheless, the rates are still high, reaching values around 80% of accuracy.

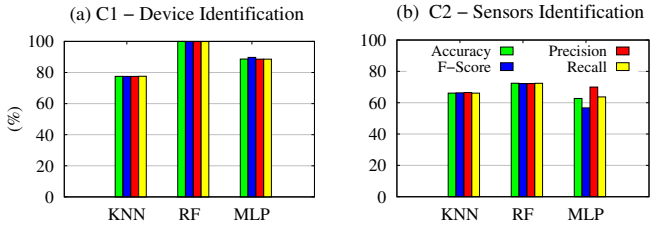

Fig. 5: Classifiers Results from C1 and C2 Scenarios

In Fig. 5b, RF reaches 72.2% of accuracy, while KNN and MLP present 66.1% and 63.7%, respectively. These results confirm the relationship between response time and similar sensors, impacting, even more, in user privacy. Identifying the sensor embedded in the device allows the attacker to infer information about the monitored data. The motivation to analyze similar sensors embedded in different devices was based on the results obtained in device identification. Table III presents a part of the KNN confusion matrix according to device identification. When data input is classified incorrectly, generally, it is assigned to a class that represents an equal sensor, but from a different device (e.g., *Node 2-l* with *Node 1-l* - highlighted in bold). This pattern demonstrates that the time taken by a sensor to capture data from the environment reveals information about what data is monitored, i.e., sensors have timing-based side-channel leaks, being susceptible to attacks.

1-a	*1-g*	***1-l***	*1-m*	*1-p*	*2-a*	*2-g*	***2-l***	*2-m*	*2-p*	
261	0	**0**	0	0	39	0	**0**	0	0	*1-a*
0	248	**0**	0	0	0	48	**0**	0	0	*1-g*
0	0	**232**	0	0	0	0	**44**	**0**	**0**	***1-l***
0	0	**0**	188	0	0	0	0	56	0	*1-m*
0	0	**0**	0	269	0	0	0	0	47	*1-p*
39	0	**0**	0	0	239	0	0	0	0	*2-a*
0	35	**0**	0	0	0	210	0	0	0	*2-g*
0	0	**37**	**0**	**0**	**0**	**0**	**270**	**0**	**0**	***2-l***
0	0	0	28	0	0	0	0	264	0	*2-m*
0	0	0	0	48	0	0	0	0	245	*2-p*

Note that 1 and 2 refers to $Node1$ and $Node2$, respectively.
a, g, l, p, m refers to the sensors embedded in each node (device).

TABLE III: KNN Confusion Matrix - C1

Moreover, results suggest a relationship between accelerometer, magnetometer, and gyroscope sensors, as shown in Table IV. Algorithms have classified them in the same class because their components are magnetometer-based and programmed similarly. Although they collect data differently, the sensors present timing-based side-channel leaks. Through these leaks, one infers crucial information about devices and sensors.

a	g	i	m	p	
177	8	0	94	0	Accelerometer (a)
29	130	0	55	0	Gyroscope (g)
0	0	251	0	0	Lighting (i)
103	55	0	100	0	Magnetometer (m)
0	0	0	0	246	Pressure (p)

TABLE IV: Random Forest Confusion Matrix - C2

B. Privacy Protection Results

We have evaluated the methods of the fake packets and delay insertions, implemented by the FISHER privacy protection module. The first method sends a false request, and the second one inserts a delay in the core of the device operating system. Both methods force two or more devices to act similarly in the same time instant, i.e., they mask timing-based side-channel leaks on network traffic. The achieved results do not present any visible network behavior pattern. Also, they show that FISHER increases the average response time in 7 ms. However, the performance of the classifiers significantly decreases when implementing the privacy protection module, as shown in Fig. 6. Particularly, Figs. 6a and 6b show the results according to the device (C1 scenario) and sensor identification (C2 scenarios), respectively.

In Fig. 6a, the classifiers reduce around 20% all performance metrics, when compare to the results shown in Fig. 5a. RF and MLP reach 58% and 40% of F-Score rate, respectively. In the C2 scenario, the performance metrics reached values smaller than 40% when compared to the results shown in Fig. 6b. RF and MLP achieve 38% and 27% of precision, respectively, making it difficult sensor identification. For a better view of the results and improvements obtained from FISHER, Table V shows the RF confusion matrix using the privacy protection module. When compared to Table IV, one observes the effects of FISHER. Fake packets and delay insertion make difficult for the classifiers to identify devices and sensors, obfuscating the timing-based side-channel leaks. FISHER identifies such leaks and masks them to protect user privacy.

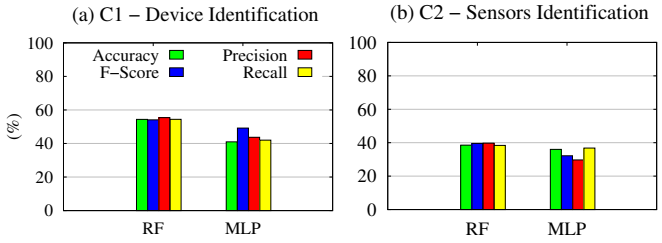

Fig. 6: Classifiers Results with Privacy Module

a	g	i	m	p	
137	122	116	133	15	Accelerometer (a)
145	130	137	115	4	Gyroscope (g)
110	107	154	100	11	Lighting (i)
129	109	123	119	12	Magnetometer (m)
29	24	19	24	408	Pressure (p)

TABLE V: Random Forest Confusion Matrix FISHER - C2

VI. Conclusions

This paper presented FISHER, a new mechanism to defend against timing-based side-channel attacks on IoT traffic. FISHER identifies the timing side-channel leaks, obfuscates them and protects IoT user privacy by inserting delays and generating fake packets. Results have indicated the possibility of identifying IoT devices and identical sensors through timing leaks, where accuracy ranges from 80% to 100% in different classifiers. Through FISHER, we reduced the identification performance for 40% of accuracy, and then, masked timing-based leaks. Therefore, this assist the research community in designing network mechanisms to improve IoT user privacy. As future directions, we will improve FISHER privacy protection module considering device power consumption and implement differential privacy.

References

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Commun. Surveys & Tuts.*, vol. 17, no. 4, pp. 2347–2376, 2015.

[2] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton, "Peek-a-boo, I still see you: Why efficient traffic analysis countermeasures fail," in *Symposium on Security and Privacy*. IEEE, 2012, pp. 332–346.

[3] V. Srinivasan, J. Stankovic, and K. Whitehouse, "Protecting your daily in-home activity information from a wireless snooping attack," in *International Conference on Ubiquitous Comput. (UbiComp)*. ACM, 2008, pp. 202–211.

[4] W. Zhang, Y. Meng, Y. Liu, X. Zhang, Y. Zhang, and H. Zhu, "Homonit: Monitoring smart home apps from encrypted traffic," in *Conference on Comp. Commun. Security (SIGSAC)*. ACM, 2018, pp. 1074–1088.

[5] J. He, Q. Xiao, and M. S. Pathan, "A method for countering snooping-based side channel attacks in smart home applications," in *International Conference on Commun. and Netw.* Springer, 2016, pp. 200–207.

[6] J. He, Q. Xiao, P. He, and M. S. Pathan, "An adaptive privacy protection method for smart home environments using supervised learning," *Future Internet*, vol. 9, no. 1, p. 7, 2017.

[7] N. Apthorpe, D. Y. Huang, D. Reisman, A. Narayanan, and N. Feamster, "Keeping the smart home private with smart (er) IoT traffic shaping," *Proc. Privacy Enhancing Technol.*, vol. 2019, no. 3, pp. 128–148, 2019.

[8] C. Adjih, E. Baccelli, E. Fleury, G. Harter, N. Mitton, T. Noel, R. Pissard-Gibollet, F. Saint-Marcel, G. Schreiner, J. Vandaele *et al.*, "FIT IoT-LAB: A large scale open experimental IoT testbed," in *Proc. World Forum Internet of Things (WF-IoT)*. IEEE, 2015, pp. 459–464.

[9] B. Copos, K. Levitt, M. Bishop, and J. Rowe, "Is anybody home? inferring activity from smart home network traffic," in *Security and Privacy Workshops (SPW)*. IEEE, 2016, pp. 245–251.

[10] B. Saltaformaggio, H. Choi, K. Johnson, Y. Kwon, Q. Zhang, X. Zhang, D. Xu, and J. Qian, "Eavesdropping on fine-grained user activities within smartphone apps over encrypted network traffic," in *Proc. 10th Workshop Offensive Technol. (WOOT)*. USENIX, 2016, pp. 1–10.

[11] S. Xiong, A. D. Sarwate, and N. B. Mandayam, "Defending against packet-size side-channel attacks in IoT networks," in *Proc. International Conference Acoust., Speech, Signal Process. (ICASSP)*. IEEE, 2018, pp. 2027–2031.

[12] V. Selis and A. Marshall, "A fake timing attack against behavioural tests used in embedded IoT M2M communications," in *Proc. 1st Cyber Security in Netw. Conference*. IEEE, 2017, pp. 1–6.

[13] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, "Transmission of IPv6 packets over IEEE 802.15.4 networks," IETF, Tech. Rep., 2007.

[14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *J. of Mach. Learn. Research*, vol. 12, pp. 2825–2830, 2011.

[15] F. Pacheco, E. Exposito, M. Gineste, C. Baudoin, and J. Aguilar, "Towards the deployment of machine learning solutions in network traffic classification: A systematic survey," *IEEE Commun. Surveys & Tuts.*, vol. 21, no. 2, pp. 1988–2014, 2018.