Who do you sync you are?
Smartphone Fingerprinting via Application Behaviour

Tim Stöber
TU Kaiserslautern
t_stoebe@cs.uni-kl.de

Mario Frank
UC Berkley
mfrank@berkeley.edu

Jens Schmitt
TU Kaiserslautern
jens.schmitt@cs.uni-kl.de

Ivan Martinovic
University of Oxford
ivan.martinovic@cs.ox.ac.uk

ABSTRACT

The overall network traffic patterns generated by today's smartphones result from the typically large and diverse set of installed applications. In addition to the traffic generated by the user, most applications generate characteristic traffic from their background activities, such as periodic update requests or server synchronisation. Although the encryption of transmitted data in 3G networks prevents an eavesdropper from analysing the content, periodic traffic patterns leak side-channel information like timing and data volume. In this work, we extract such side-channel features from network traffic generated from the most popular applications, such as Facebook, WhatsApp, Skype, Dropbox, and others, and evaluate whether they can be used to reliably identify a smartphone. By computing fingerprints from ≈ 6 hours of background traffic, we show that 15 minutes of monitored traffic suffice to reliably identify a smartphone based on its behavioural fingerprint with a success probability of 90%.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and protection*

Keywords

Smartphone; Authentication; Measurement

1. MOTIVATION

Over the last decade, smartphones became omnipresent. According to [1], 419 million devices have been purchased all over the world during the second quarter of 2012, and half of all US mobile subscribers own a smartphone [2]. The main reasons for such a popularity of smartphone usage are the overall improvement in performance, battery life, and decreasing price of the mobile Internet access over 3G radio networks. In particular, the frequent Internet access is crucial for the success of application markets and a high number of downloaded applications (Apps), such as Email clients, Facebook, WhatsApp, Skype, or Dropbox. Most of these Apps generate traffic, which is not only initiated by the user, but also from the Apps itself to maintain the most current state, receive updates, or synchronise cloud services. For example, Figure 1 shows results of our measurements of network traffic transmitted to and from different smartphones during 24h. We identified that only 30% of the overall smartphone traffic can be attributed to user interactions (we refer to is as *interactive traffic*), and 70% of the traffic belongs to background activities generated by different installed Apps. Many of such background activities result in characteristic traffic patterns, especially in the time domain and in the volume of transmitted data. In addition, the generated traffic highly depends on the multitude of installed applications and their personal configuration.

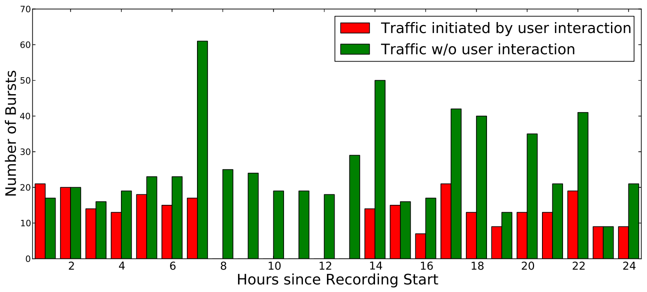

Figure 1: Interactive- and non-interactive smartphone traffic (24 hours). Only about 30% of transmissions are triggered by user interactions.

The 3G/UMTS radio access technology implements encryption at the data link-layer to guarantee the confidentiality of the users' data in the presence of a wireless eavesdropper. Yet, the resulting application-dependent traffic patterns may still pose a privacy risk. A wireless eavesdropper might be able to identify a particular smartphone by analysing only the side-channel information of encrypted traffic. Hence, this motivates the main research question of this work: is it possible to identify a smartphone based on the traffic behaviour of the installed applications, arising from their background activities? Importantly, we assume that an eavesdropper is completely agnostic to the content of the traffic and that security services of the current UMTS radio access network remain unaffected.

In our scenario, the adversary is a passive eavesdropper which is able to capture encrypted wireless 3G/UMTS data from a victim's smartphone and using that traffic to extract smartphone fingerprints. The fingerprints would then allow him to identify the device afterwards and make a point of whether the victim's smartphone is present within a certain UMTS radio cell. While the granularity of such UMTS cell-based tracking does not provide a very fine-grained physical position of the smartphone, it might still reveal important privacy information and help in launching more sophisticated attacks, for example, by detecting if the user is at home or at the workplace.

To assess the threat of identifying smartphones through analysing side-channel information of background traffic, we attempt to answer the following research questions:

- How discriminative are the features of smartphone traffic generated by applications' background activities?
- Are individual configurations of installed Apps sufficient to distinguish between different smartphones?
- How long does it take to identify a smartphone?

2. THREAT MODEL AND ASSUMPTIONS

We assume that the attacker is located within the range of UMTS transmissions. The inherent broadcast characteristic of wireless communications allows him to eavesdrop on UMTS physical signals. Hence, our main assumption is that the attacker is able to demodulate and demultiplex the physical layer and measure side-channel information such as the amount of transmitted data and timing information. In order to acquire such side-channel information, the attacker must extract the users' data streams from the superimposed signal on the corresponding wideband code division multiple access (WCDMA) air interface. In this section, we briefly discuss technical requirements and the complexity to gather side-channel information without the possession of spreading and scrambling codes used in UMTS.

The 3G/UMTS air interface is based on WCDMA, a code multiplexing technique to separate the medium into single channels and enable simultaneous transmissions over the same frequency. The users' data (payload) is transmitted over so-called Dedicated Physical Data Channels (DPDCH), and its correct demodulation requires the knowledge of scrambling and spreading codes as specified in [3]. The spreading codes increase the actual bandwidth of the signal, while the scrambling codes are multiplied chip by chip with the already spreaded signal to achieve orthogonal coding. The scrambling codes separate distinct base stations on the downlink and different user equipments (UEs) on the uplink. However, none of these coding techniques were designed for security reasons and the security services are offered by the higher layers of the UMTS network stack. In particular, both spreading and scrambling codes can be "brute-forced" as their search space is not large (and the assignment of spreading codes on the uplink is almost completely specified in [3]). For example, assuming one base station (i.e., Node-B) using one primary scrambling code for its cell, there are less than 1000 available spreading codes that could be employed after deducting codes for reserved channels [3]. The scrambling code is more expensive to find, as it is generated by 18 bit (downlink) and 24 bit (uplink) seeded shift registers, respectively. Yet, none of these lengths presents a significant computational burden for an adversary. Moreover, the Node-B's scrambling code for downlink is automatically determinable by the smartphone's cell search procedure. In our experiments, we therefore investigate how the availability of only the downlink traffic affects the success probability of correctly identifying a smartphone. Table 1 shows a cost estimate for obtaining the required codes in the downlink and uplink cases.

	Scrambling	**Spreading**
Up	2^{24} possibilities	max. 7 possibilities
Down	2^{18} possibilities[1]	max. 1000 possibilities

Table 1: Cost estimate for attaining scrambling- and spreading codes. Determination of Node-B's scrambling code is less expensive in contrast to the UEs'.

In summary, we believe it is reasonable to assume that there is a practical way for an adversary to capture the encrypted UMTS traffic and to use it for fingerprint acquisition (i.e., the training phase) and for fingerprint detection (i.e., the attack phase). During fingerprint acquisition, the adversary should know whether the extracted traffic belongs to the victim. One concrete approach to achieve this would be to inject known traffic markers by initiating a call, sending an email, or sending an SMS to the victim. However, the detailed analysis of this approach is out of the scope of this work.

3. DATA ACQUISITION

The first dataset consists of recorded traffic from five distinct users for whom all 3G network communication has been captured in the background for approximately eight hours. During this collection phase, the users interacted with their smartphones without any restrictions.

Due to the low scope of the user dataset, we recorded 8 hours from 20 user devices with different combinations of Apps installed. We call this dataset *non-interactive* because transmissions were captured without any user interactions that could cause traffic.

Our testbed was a Samsung Galaxy Nexus running the Android operating system version 4.0.4. Instead of sniffing traffic on the UMTS link, we captured directly on the devices' 3G interface using `tcpdump`. We randomly picked 20 distinct combinations out of a universe of 14 Apps. Each combination is composed of seven Apps, whereas we assured the marginal cases to be present, meaning two combinations with six common applications and only one differing, as well as two combinations with completely disjoint subsets.

The 14 Apps were selected from the list of top free Android applications from the Google Play Store. We only picked Apps that actually produce background transmissions without user interactions, since otherwise they would have no effect on the traffic behaviour of the device. This comprises cloud services, several Messengers, or Email clients. The complete list of chosen Apps with some additional information from `https://play.google.com/store/apps` (accessed 26/09/2012) can be found in Table 2.

4. FINGERPRINTING

In this section we first introduce the notion of a burst, which plays a central role in our framework, and then we describe the process of creating smartphone fingerprints.

As an input, the process requires traffic extracted from a specific device. The traffic is a chronological sequence of incoming and outgoing packets. Each packet is represented as a vector $p_i = (t_i, s_i, d_i)$. Hence, the only information

[1]Node-B's scrambling code determinable on CPICH.

Index	Name	Downloads	Rank
1	Email	-	native
2	Facebook	100 - 500	4
3	WhatsApp	50 - 100	9
4	Skype	50 - 100	12
5	Twitter	50 - 100	14
6	Dropbox	10 - 50	15
7	Instagram	10 - 50	23
8	Flipboard	5 - 10	27
9	Viber	10 - 50	34
10	Evernote	10 - 50	156
11	Spotify	10 - 50	66
12	Wetter.com	5 - 10	not ranked
13	Skydrive	0.1 - 0.5	422
14	ChatON	10 - 50	50
15	Google Account	-	native

Table 2: Universe of applications from which the emulating combinations were chosen.

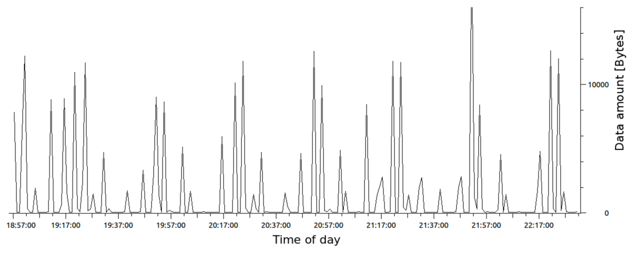

Figure 2: 3.5 hours of smartphone traffic. Idle phases alternating with transmissions illustrate the burstiness. As can be observed similar burst patterns occur in regular intervals.

needed about a packet is its arrival time t_i in the form of an absolute or relative time stamp, its size in bytes s_i and the direction d_i in terms of a Boolean flag distinguishing between incoming and outgoing packets.
Taking a closer look at smartphone traffic, one can observe alternating idle periods followed by short peaks of incoming and outgoing data transfers. This behaviour, which we refer to as burstiness, is illustrated in Figure 2 and has also been observed by Falaki et al. in [10]. A single burst, represented by the peaks of the black curve, consists of a sequence of packets that are mostly semantically connected like, e.g., packets from the same TCP connection.

4.1 Burst Separation

Since we cannot analyse the payload to identify and aggregate packets from the same application (using e.g., TCP flows), the bursts are extracted by only considering the timing information. We define a burst distance to be the minimum length of an idle interval between two packet arrivals. If the arrival time of two subsequent packets is larger, the respective packets are considered to belong to different bursts. Hence, extracting bursts from the captured traffic is equivalent to setting cuts in the packet sequence and aggregating all packets which are within these borders to one burst.
Clearly, selecting the burst distance is an important parameter. On the one hand, we may prefer a small distance because this avoids the aggregation of several unrelated transmissions. On the other hand, choosing a too small distance may result in splitting related transmissions.
In agreement to the observations by Falaki et al. [10], we observed that 95% of all packets arrive at most 4.43s after their predecessors (4.5s in [10]). Therefore, we selected a distance threshold of 4.5s for burst separation.

4.2 Burst Characterisation

In the next step, we identify the most discriminative features that distinguish well between bursts generated by different smartphones. In addition to the mean values of the packet inter-arrival times and packet sizes, we also take into account the 20%, 50% (median) and 80% quantiles of the timing and size distributions. In the presence of outliers and non-Gaussian distributions, these measures are more robust in comparison to the simple arithmetic mean. For the same reason, we apply the median absolute deviation (MAD) in terms of packet sizes and inter-arrival times. The MAD is the median of the deviations from the median [14]. Let X be a sample set vector, it is computed as

$$MAD = median(|X - median(X)|). \quad (1)$$

In the following two subsections we examine the individual features in terms of their importance. The purpose of this analysis is not primarily to minimize computational costs in the training phase or to address the curse of dimensionality; we are rather interested in gaining a better understanding of this particular kind of data. The complete list of features is listed in Table 3.

Feature	rMI
Median absolute deviation (MAD) packet size	16.6%
50% quantile packet size	15.7%
Standard deviation packet size	15.6%
80% quantile packet size	14.7%
20% quantile packet size	14.6%
Mean packet size	13.8%
Byte ratio	13.8%
Distance to next burst	8.5%
Number of outgoing bytes	7.2%
80% quantile packet interarrival time	7.0%
Mean packet interarrival time	6.9%
Number outgoing packets	6.9%
Packet ratio	6.5%
50% quantile packet interarrival time	6.2%
Throughput	5.9%
Duration	5.8%
Number of incoming packets	5.7%
Number of incoming bytes	5.6%
Median absolute deviation packet interarrival time	5.5%
Standard deviation packet interarrival time	5.5%
Mean consecutive outgoing packets	4.8%
20% quantile packet interarrival time	4.5%
Mean consecutive incoming packets	4.2%
Random feature	0.9%

Table 3: Feature list with respective relative mutual information (rMI) for the non-interactive dataset.

To determine the importance of individual features, we compute the mutual information $I(F_i; U) = H(U) - H(U|F_i)$ between a feature F_i and the target variable (the user ID) U. To account for the fact that the entropy of target variables can vary, we compute the relative mutual information (rMI) that can be computed as a fraction of entropies:

$$\text{rMI}(F_i; U) = I(F_i; U)/H(U) = 1 - H(U|F_i)/H(U) \quad (2)$$

The entropy $H(U)$ quantifies the uncertainty about the ID. The conditional entropy $H(U|F_i)$ quantifies the remaining uncertainty if the value of feature i is known. The difference of $H(U)$ and $H(U|F_i)$ becomes maximal if the feature fully determines the user ID.
Before computing rMI, it is necessary to quantise the continuous feature values. To account for outliers, we divided the 0%- to the 90%-quantile into bins and accounted all outliers to the last quantile. The results for rMI are shown in Table 3. To offer a point of reference, we introduced an artificial feature with random values that should hardly provide any information about the user.
We refrain from choosing only the best ranked features as a result of our analysis. The reason is that even variables with a small independent informativeness can provide rich information when combined [13]. In contrast, pairs of variables with large rMI could be fully redundant.

4.3 Classifiers

As it is common practice in supervised learning, we turn this multi-class classification problem with n user IDs into n individual binary classification problems. In each individual problem, a classifier must decide if the current data comes from the respective phone or not. We consider each phone's classifier as the fingerprint of the phone. For each such problem, the observation matrix, used as an input for training the classifier, consists to 50% of bursts from the legitimate user. The remaining 50% are filled by an equal number of bursts from other users. This way, we only have two class labels, namely *user* and $\neg$*user*. Each observation or burst in the observation matrix is represented by a row whereas each of the 23 columns corresponds to a feature.
In terms of classification, we use the k-nearest neighbors algorithm (kNN) as well as a support vector machine (SVM). The kNN is capable of directly solving the original multi-class classification problems since it simply stores the feature vector of every observation and its corresponding categorial label. A new object is then classified by taking the majority label of the k nearest neighbors in the feature space. To estimate the optimal setting for the parameter k, we perform 5-fold cross-validation on the training data testing all odd numbers from 1 to 13. In contrast to kNN, the SVM does not store all feature vectors but instead computes one or several hyperplanes that divide the set of observations according to their class labels, whereas the distance of the hyperplane to the nearest objects (called support vectors) is maximised.

5. EXPERIMENTATION

5.1 Single Burst Classification

The goal of this experiment is to investigate the feasibility of our identification approach. This requires to test for possible collisions between each user's fingerprint with the other fingerprints.
When referring to a particular user from the non-interactive dataset, we always mean the App combination on the phone of this user.
For each user u out of the dataset, a fingerprint was constructed by training a classifier with 70% of u's bursts and the same amount of bursts from other users, both randomly chosen. After the fingerprints were generated, we started classifying the remaining 30% of u's bursts as well as equally

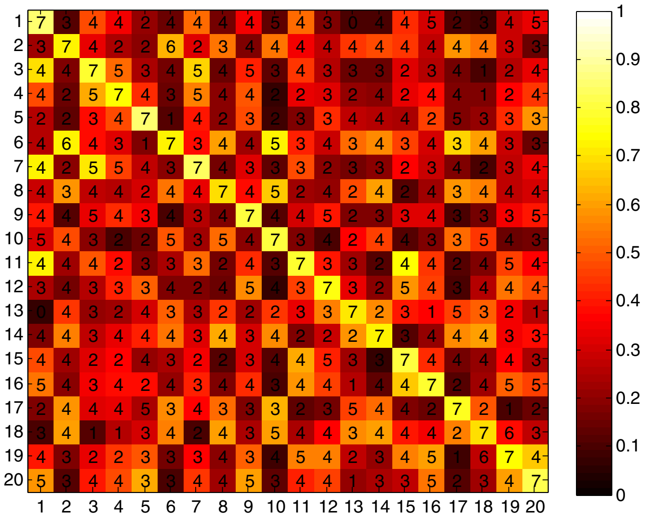

Figure 3: kNN results for matching users against every fingerprint (Non-interactive dataset). The brighter the color, the more bursts were classified to belong to the fingerprint. The numbers inside the cells indicate the count of common Apps of two combinations.

	User dataset		Non-interactive dataset	
FNR	13.60%	16.99%	17.44%	22.85%
FPR	24.02%	31.41%	32.60%	33.46%

Table 4: False negative- and false positive rates for feasibility experiment (SVM left, kNN right).

many random bursts from every single other user, assuring not to employ any burst that has been used for building the classifier. Each classification combination was repeated 20 times. In every round, the training- and test set were populated by newly, randomly selected bursts.
As a result, we obtain the number of false negative- (FN) and false positive (FP) class assignments. FNs for the cases where we match users on their own fingerprints and FPs for the remaining cases. Fig. 3 depicts the results for the kNN classification of the non-interactive dataset. The cells of the colour matrix show how well the traffic of the column-user matched the fingerprint of the row-user. The brighter the colour, the more bursts have been classified to match the fingerprint. The numbers within the cells indicate how many applications the two compared phones have in common.
In general, SVM performs better, especially in terms of the false negative rate (diagonal). The precise false positive- and false negative rates for both classifiers and both datasets are given in Table 4.
The results for SVM on the user dataset are best. This is not only due to the minor scope of the dataset but also because the event of two real users having nearly the same combination of installed Apps and similar configurations is rather unlikely. This makes the UMTS application attractive to an attacker, since he would only have to compare the traffic with a fixed number of users per cell.
Obviously it becomes more and more difficult for the classifier to distinguish users with many shared Apps and, consequently, similar traffic. However, there are also other influences. E.g. the App Spotify tends to generate many bursts. Therefore, matching two phones against each other that both contain Spotify will exacerbate the decision of the

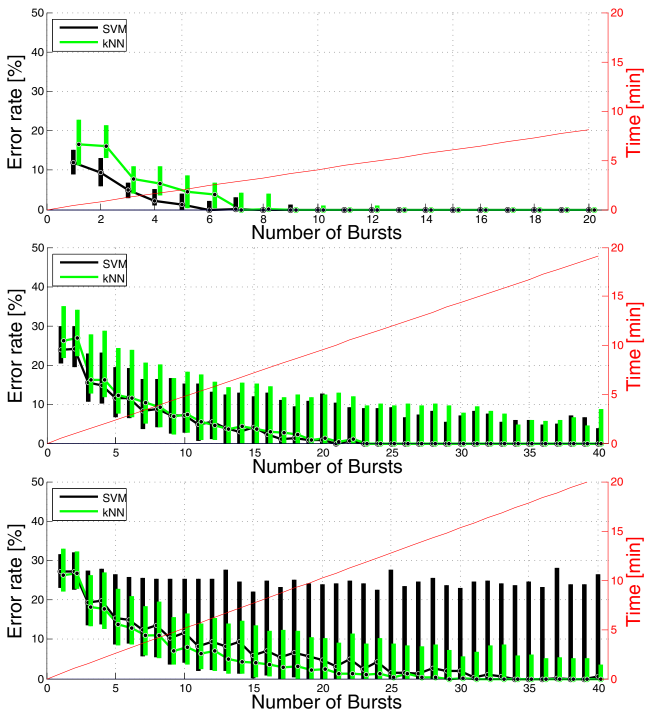

Figure 4: Error rate versus the number of bursts used for classification. To achieve a median classification error rate of 0%, an attacker would have to wait approximately 3 min in case of the user dataset (upper chart), 12 min for the non-interactive dataset (middle chart), and 15.5 min when only non-interactive downlink data is used (lower chart).

classifier. However, we always used Spotify with the same configuration which made the task unrealistically hard. Distinct configurations (Playlists etc.) might affect the traffic and render it more unique. Overall, the investigated setting can be considered rather conservative, since the emulated users have on average 3.5 out of only 7 installed Apps in common. We assume to observe more variety in the wild.

5.2 Effects of Capturing Duration

The objective of this experiment is to find out how much traffic one must capture to make a reliable statement whether it belongs to the victim or not. This means we have to investigate classification results for varying amounts of available traffic. To that end, we must evaluate the burst inter-arrival times to find out how long an attacker should wait to gather a certain amount of traffic.

As in the last experiment, we generate a fingerprint for each user with a training set containing 70% of its bursts and equally many random bursts from other users. Yet, unlike last time, the training set is filled with the remaining 30% of the users bursts as well as the same amount of randomly chosen bursts from other users, assuring to not employ those that have been used for training.

Instead of classifying each single burst separately, we apply a sliding window of size $k \in \{1, 2, \dots 40\}$ such that we classify k bursts and take the majority vote of the k labels. This procedure is repeated in 20 times for every user to quantify random effects.

The two upper charts in Fig. 4 depict the results for the user dataset and the non-interactive dataset applying the kNN algorithm and the SVM. The bars reach from the 25%-quantile to the 75%-quantile. The dots inside the bars represent the median values. As can be seen, the SVM generally out performs kNN. After 6 bursts for the user dataset and after 23 bursts for the non-interactive data, a median error rate of 0% is reached. In terms of time, depicted by the linear function, an attacker would have to wait merely ∼3 minutes and ∼12 minutes, respectively.

5.3 Effect of Using Downlink Data Only

In particular for UMTS, where it is easier to capture traffic on the downlink from the Node-B to the user equipments instead of the uplink, it is interesting to restrict the fingerprinting to downlink data. Fig. 4(bottom) illustrates the results of the experiment from Section 5.2 in the downlink-only case. The outcome indicates that the median error only impairs slightly. To achieve an error rate of 0%, the attacker must monitor ∼15.5 minutes compared to 12 minutes for the bidirectional case. In this scenario, kNN outperforms SVM by far. This is most likely due to the fact that the SVM performance relies on particular features that are not available anymore in the downlink case like *Packet ratio, #Packets out, Byte ratio, #Bytes out* and *Mean consecutive out.*

6. RELATED WORK

There has been much research in the area of device fingerprinting. In most cases, side-channel information, like hardware and manufacturing inconsistencies, or differences in driver implementations, were exploited as a discriminative component. Probably, one of the first publications in this direction was [15] by Kohno et al. in 2005. They introduced a mechanism to remotely fingerprint and identify devices based on their clock skews, which were in turn based on information from TCP and ICMP timestamps. In [11], the authors succeeded in fingerprinting and identifying wireless device drivers on the basis of a statistical analysis of a device's interarrival rate of IEEE 802.11 probe request frames. Since the standard does not provide a specific value for the scanning intervals of these management frames, distinct drivers tend to differ in their implementations. Based on this work, Loh et al. extended the discriminability of this feature to be even capable of distinguishing between single devices instead of device drivers [7].

In [12, 19, 5, 4], authors exploit manufacturing inconsistencies and hardware imperfections that have effect on the resulting transmission signal in order to differentiate between single entities. In addition, one can find a very detailed review of physical-layer identification systems and state-of-the-art techniques in [6].

All previously mentioned approaches are aiming at remote identification of devices, however the fingerprint does not include any form of application behaviour.

Another related area of research includes various approaches to traffic classification. Some of them use transport layer statistics like packet size, connection duration and ratio of bytes sent in each direction, combined with unsupervised machine learning algorithms [9, 21]. Even though the features are solely from the time and byte dimension, both techniques still rely on the notion of flow or connection, respectively. Clearly, traffic analysis countermeasures like padding can be used to conceal user identities. However, Dyer et al. showed in [8] that bandwidth-efficient, general-purpose traffic analysis countermeasures mostly fail. In [22], the authors use a packet-level classification, only resorting to link layer features which makes their approach applicable even under payload encryption. Similarly to our work, their methodology uses supervised learning algorithms like SVM and neural networks. Yet, their application and network environment is not focused on wireless networks. Related work on detecting network applications discusses more information that can be leaked through traffic analysis, e.g., visited web pages [16], language and spoken phrases [20] or watched videos [18]. Recently, in [17], the authors analysed the security of the TLS Record Protocol and identified attacks against TLS, even if variable length padding (as a countermeasure against SSL/TLS side-channel information leakage) is used. While many smartphone Apps establish an SSL/TLS connection with their servers, in our work, we do not consider SSL/TLS flows, but the overall aggregated traffic from all Apps and without any requirement to identify the SSL/TLS traffic.

7. CONCLUSIONS

In this work, we investigated the question of whether background traffic generated by smartphone applications can be used as a fingerprint to identify and discriminate different smartphones. We based the fingerprint features only on features available as side-channel information such as timing and data volume. These features can be extracted through monitoring of wireless channels used by the UMTS radio technology and without assuming any knowledge of the payload. Our results show that the multitude of installed applications and their background communication generates a unique behaviour that allows an eavesdropper to accurately identify smartphones. To that end, we designed extensive experiments including traffic monitoring from the most popular Apps and demonstrated that even if the smartphones have a large number of the same applications installed, they still can be successfully identified with a very high accuracy. In particular, after the fingerprint is generated, the eavesdropper requires only ≈15 min. of the captured traffic to achieve more than 90% classification accuracy.
These results have direct impact on the user's privacy as they justify that an adversary is able to detect whether a smartphone is associated with a certain UMTS radio cell.

Acknowledgments

This research was partially supported by the Swiss National Science Foundation and by Intel through the ISTC for Secure Computing.

8. REFERENCES

[1] Gartner. `http://tinyurl.com/d7ptpqc`, 2012. [Online; accessed 11/10/2012].

[2] Nielsen. `http://tinyurl.com/8y2e773`, 2012. [Online; accessed 11/10/2012].

[3] 3GPP. TS 25.213, Spreading and modulation (FDD). Technical report, 1999.

[4] K. Bonne Rasmussen and S. Capkun. Implications of radio fingerprinting on the security of sensor networks. In *Third International Conference on Security and Privacy in Communications Networks*, SecureComm'07, 2007.

[5] V. Brik, S. Banerjee, M. Gruteser, and S. Oh. Wireless device identification with radiometric signatures. In *Proceedings of the 14th ACM international conference on Mobile Computing and Networking*, MobiCom'08, 2008.

[6] Danev B., Zanetti D., Capkun S. On physical-layer identification of wireless devices. `http://www.syssec.ethz.ch/research/OnPhysId.pdf`, 2012. [Online; accessed 22/10/2012].

[7] L. C. C. Desmond, C. C. Yuan, T. C. Pheng, and R. S. Lee. Identifying unique devices through wireless fingerprinting. In *Proceedings of the first ACM conference on Wireless Network Security*, WiSec'08, 2008.

[8] 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 *IEEE Symposium on Security and Privacy*, SP'12, 2012.

[9] J. Erman, M. Arlitt, and A. Mahanti. Traffic classification using clustering algorithms. In *Proceedings of the 2006 SIGCOMM workshop on Mining Network Data*, MineNet'06, 2006.

[10] H. Falaki, D. Lymberopoulos, R. Mahajan, S. Kandula, and D. Estrin. A first look at traffic on smartphones. In *Proceedings of the 10th annual conference on Internet Measurement*, IMC'10, 2010.

[11] J. Franklin, D. McCoy, P. Tabriz, V. Neagoe, J. Van Randwyk, and D. Sicker. Passive data link layer 802.11 wireless device driver fingerprinting. In *Proceedings of the 15th conference on USENIX Security Symposium*, USENIX-SS'06, 2006.

[12] R. M. Gerdes, T. E. Daniels, M. Mina, and S. F. Russell. Device identification via analog signal fingerprinting: A matched filter approach. In *In Proceedings of the Network and Distributed System Security Symposium*, NDSS'06, 2006.

[13] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 2003.

[14] F. R. Hampel. The breakdown points of the mean combined with some rejection rules. *Technometrics*.

[15] T. Kohno, A. Broido, and K. Claffy. Remote physical device fingerprinting. *IEEE Transactions on Dependable and Secure Computing*, 2005.

[16] M. Liberatore and B. N. Levine. Inferring the source of encrypted http connections. In *Proceedings of the 13th ACM conference on Computer and Communications Security*, CCS '06, 2006.

[17] K. G. Paterson, T. Ristenpart, and T. Shrimpton. Tag size does matter: attacks and proofs for the tls record protocol. In *Proceedings of the 17th international conference on The Theory and Application of Cryptology and Information Security*, ASIACRYPT'11, 2011.

[18] T. S. Saponas, J. Lester, C. Hartung, S. Agarwal, and T. Kohno. Devices that tell on you: privacy trends in consumer ubiquitous computing. In *Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*, SS'07, 2007.

[19] O. Ureten and N. Serinken. Wireless security through rf fingerprinting. *Canadian Journal of Electrical and Computer Engineering*, 2007.

[20] C. Wright, L. Ballard, S. Coull, F. Monrose, and G. Masson. Spot me if you can: Uncovering spoken phrases in encrypted voip conversations. In *IEEE Symposium on Security and Privacy*, SP'08, 2008.

[21] S. Zander, T. Nguyen, and G. Armitage. Automated traffic classification and application identification using machine learning. In *The IEEE Conference on Local Computer Networks*, 2005.

[22] F. Zhang, W. He, X. Liu, and P. G. Bridges. Inferring users' online activities through traffic analysis. In *Proceedings of the fourth ACM conference on Wireless Network Security*, WiSec'11, 2011.