Is It Really You? User Identification Via Adaptive Behavior Fingerprinting

Paul Giura
AT&T Security Research Center
New York, NY
paulgiura@att.com

Ilona Murynets
AT&T Security Research Center
New York, NY
ilona@att.com

Roger Piqueras Jover
AT&T Security Research Center
New York, NY
roger.jover@att.com

Yevgeniy Vahlis[*]
Bionym Inc
Toronto, ON Canada
yvahlis@bionym.com

ABSTRACT

The increased popularity of mobile devices widens opportunities for a user either to lose the device or to have the device stolen and compromised. At the same time, user interaction with a mobile device generates a unique set of features such as dialed numbers, timestamps of communication activities, contacted base stations, etc. This work proposes several methods to identify the user based on her communications history. Specifically, the proposed methods detect an abnormality based on the behavior fingerprint generated by a set of features from the network for each user session. We present an implementation of such methods that use features from real SMS, and voice call records from a major tier 1 cellular operator. This can potentially trigger a rapid reaction upon an unauthorized user gaining control of a lost or stolen terminal, preventing data compromise and device misuse. The proposed solution can also detect background malicious traffic originated by, for example, a malicious application running on the mobile device. Our experiments with annonymized data from 10,000 users, representing over 14 million SMS and voice call detail records, show that the proposed methods are scalable and can continuously identify millions of mobile users while preserving data privacy, and achieving low false positives and high misuse detection rates with low storage and computation overhead.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*security and protection*; D.4.6 [**Operating Systems**]: Security and Protection—*authentication, access controls*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection—*authentication, unauthorized access*

Keywords

Identification; Behavior fingerprint; Mobile security

[*]Dr. Vahlis participated in this work while being a member of the AT&T Security Research Center.

1. INTRODUCTION

Mobile devices, such as smartphones and tablets, are increasingly being lost or stolen due to their widespread use, tremendous popularity and price [8]. In turn, loss of a mobile device increases the likelihood of account misuse and stolen or compromised data, which calls for user identification schemes superior to traditional use of passwords, which are very often not strong enough [3].

In parallel, data theft and compromise are often also linked to a malware or trojan infection in a mobile terminal. There have been reports of malicious applications posting sensitive phone information, such as the IMSI (International Mobile Subscriber Identity) and the IMEI (International Mobile Equipment Identity), on remote servers [15]. Cell-phones are becoming the latest platform for malware spreading in data networks [10], with malicious applications constantly being reported in the media and industry [2, 4, 5, 24].

Device misuse can potentially have an economic impact on the device user, which leads to customer dissatisfaction. This is the case of known instances of malware that send messages to premium rate numbers [1], with the resulting spike in the customer's monthly bill, or identity theft as a result of an unauthorized access to an email account through a stolen cell-phone [17]. Note that most user credentials, including online banking, can be reset via an email sent to the email account that an attacker could access in the case of compromising a lost or stolen device [27].

There are currently no means for rapid network-based detection of mobile device misbehavior, either originated by a piece of malware running in the background or as a result of misuse of the device after being lost/stolen. Most schemes addressing this problem rely on heavy processes running on the device, with the resulting limitations in terms of battery life impact and data storage. In the case of a lost or stolen mobile terminal, all preventive measures to avoid misuse or data compromise rely on a rapid reaction by the end user to, for example, change email passwords and, if necessary, contact the cellular provider to block the SIM (Subscriber Identity Module). In certain situations it might take a long time from the moment the device is lost until the user reports it. This would give a potential attacker plenty of time to compromise the device.

Cellular infrastructures need to provide means of network-based misuse detection independent - yet transparent - from the user that do not rely on any local computation or processing on the device. For example, if a cell-phone is lost or stolen at an airport prior to boarding a 14 hour intercontinental flight, the network should be able to independently detect and block any misbehavior originating from the stranded device. And eventually block the terminal if appropriate. Relying on the user reporting the incident would give the attacker 14 hours of time to compromise the device.

In this paper we tackle the problem of non-intrusive identification of a user that legitimately interacted with the device previously. In general there are several ways to identify a device user based on:

1) *what the user knows* (i.e. password, passphrase, etc.),
2) *what the user has* (i.e. biometrics, token, etc.),
3) *what the user does* (i.e. behavior, tasks patterns, etc.)
4) *who user knows* (i.e. user contacts).

This work focuses on finding a method that covers the last two categories: *what the user does* and *who the user knows*. More specifically, we analyze the network Call Detail Records (CDR) generated by the device in present and in the recent past, continuously extracting a behavior fingerprint for each time period under consideration (i.e. hour of day, day of week, etc). Then, for each new observation time window, we compute the distance, in a behavior fingerprint metric, between the current fingerprint sample and the one observed in the past. If the distance is larger than a threshold, that user is identified as illegitimate (not the one interacting with the device in the past). That triggers an alert sent that could be sent to the legitimate user on an alternate channel, or could be handled by the network operator which can potentially decide to block user access until her identification is confirmed. To the best of the author's knowledge, this is the first time that such an analysis on real data has been performed and shown to work.

The proposed methods are generic and can be applied to multiple attack scenarios. However, our solution is aimed to detect unauthorized access to user resources and to provide a detection mechanism in two possible scenarios: 1) stolen mobile devices (i.e. smart-phones, tablets, etc) and 2) mobile malware compromised devices. The same methods could be implemented on other portable computers and even desktop computers. The only difference would rely in the nature of the fingerprint features being extracted and processed. Note that the proposed system is not intended to detect a phone or tablet that is turned off after being lost or stolen, but for the case when the attacker does use the device. With this work we make the following key contributions:

- We formalize the problem of user identification based on behavior fingerprint and propose several methods to represent and compare user behavior in order to identify illegitimate users for different scenarios: normal different user, random attacker, informed attacker and compromised device.
- We implemented and tested seven possible user identification methods and reported the experiments with 10,000 users SMS and data calls from a tier-1 wireless carrier, representing over 14 million records. Our results show that in most cases the proposed methods can detect with over 90% accuracy most types of attacks within an hour with less than 5% false positives.
- We analyze the privacy properties of our method and show that in many cases, the information about raw user data that must be given to the identification system is minimal. In particular, privacy of contact information is always preserved, and even the communication pattern can be hidden to some degree.

The remainder of this paper is organized as follows. An overview of our methods is presented in Section 2. The behavior fingerprints extracted in each one of the methods under analysis are described in Section 3. Section 4 discusses the potential information leakage about each user profile from the information that we store. Our experimental results are summarized in Section 5 and related work is discussed in Section 6. Finally, the paper is concluded in Section 7.

2. METHOD OVERVIEW

2.1 Adversarial Model

Throughout this paper we consider the following adversarial models. In all cases, the goal of the adversary is to continue using the device for as long as possible while avoiding triggering the authentication mechanism.

2.1.1 User Swap

In this scenario we assume that an unsophisticated attacker has gained possession and control of the device. An unsophisticated attacker completely ignores the prior behavior of the user, and simply starts using the device as her own. The device usage behavior of the attacker in this case is independent from the behavior of the legitimate owner of the device. We believe that this scenario is applicable in many practical situations, where the attacker has gained possession of a device and has no familiarity with the legitimate owner as may be the case when the phone is lost or stolen by a pickpocket.

2.1.2 Informed Attacker

The second scenario that we consider is when the attacker knows the behavior pattern of the legitimate user, but is unable to fake certain activities. For example, the attacker can dial the same numbers as the owner, but is unlikely to be able to maintain a conversation for the same length of time because the person on the other end will recognize that the caller is not the owner of the device, and will refuse to continue the conversation. More precisely, the informed attacker generates events from a distribution which is a function of the real distribution of the owner behavior, but with an additional "noise" component which represents the actions taken by the adversary for her own purposes.

2.1.3 Compromised Device

In this scenario we assume the device is used in parallel by the legitimate user and a piece of malware or a trojan running in the background without the user knowledge. For

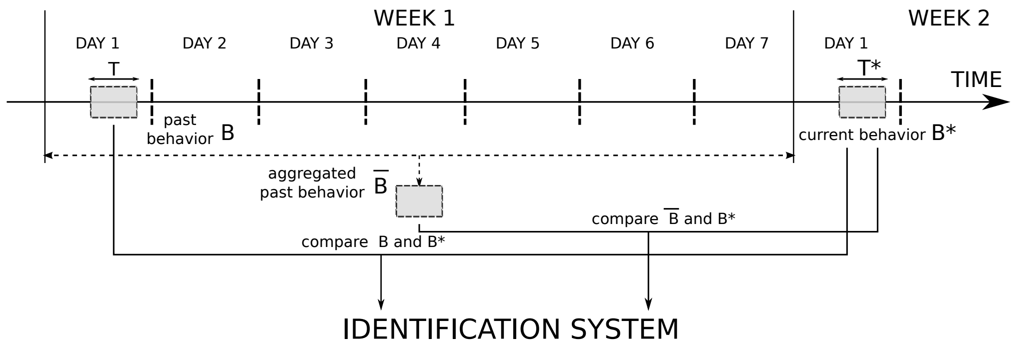

Figure 1: Schematic representation of a behavior based identification method.

example, the well known Android trojan GGTracker [1] attempts to dial or send SMS messages to premium numbers without user consent, incurring unexpected charges. In this scenario, the victim's device would be generating the exact same fingerprint as in the past, given the continued legitimate interaction of the user with the phone. In parallel, network traces would be created from the malicious connections that the malware or trojan operates. For our measurements, we generated adversarial events randomly according to what we believe is a reasonable distribution (described in detail in Section 5).

2.2 Problem Statement

In this section we focus on defining the behavior identification problem and identifying the properties of a robust and efficient solution. More formally, suppose that we have n different types of session (e.g., phone calls, SMS, data, etc), and that session of type i has m_i different features. For example, a feature in the "phone calls" session may correspond to a dialed number. We define user behavior over a period $T = [t_0, t_1]$ by a set of recorded sessions $\{S_1(T), \ldots, S_n(T)\}$, where each recorded session $S_i(T)$ is a subset of the features $f_{i,j}$ that were measured for the j-th feature of session type i, for a user during time period T. Then, given a user with behavior $S(T^*) = S_1(T^*), \ldots, S_n(T^*)$ over a period T^* we want to find an efficient, reliable and robust way to determine whether the same users' behavior recorded during another period T^* is *sufficiently similar* to the behavior during the period T. To do so we define two functions: $\rho(S(T))$ is some representation of the behavior $S(T)$ that is sufficient to compute a score, and $\beta(\rho(S(T^*)), \rho(S(T)))$ is a measure of similarity between behavior $S(T^*)$ and $S(T)$. To determine whether $S(T^*)$ is sufficiently close to $S(T)$, we check whether

$$|\beta(\rho(S(T)), \rho(S(T^*)))| \leq \varepsilon \tag{1}$$

For some identification threshold $\varepsilon > 0$. Note that to perform the test, we must only store $\rho(S(T^*))$ and not the entire behavior record $S(T^*)$.

Besides being robust and accurate, the behavior fingerprint should preserve the privacy of the user behavior and should not reveal any information that can lead to the reconstruction of past user activity. A key challenge is to define the function ρ to retain enough information about the original recorded behavior to compute the score, while removing enough unneeded information to guarantee privacy.

Figure 1 shows an intuitive representation of how a behavior fingerprint method could be used by an identification system. The behavior fingerprint can be recorded for a rolling time window T and sent to the identification system. Whenever an identification decision should be made, the current behavior fingerprint could be compared to the behavior recorded in the corresponding time window of the previous day, or the previous week day. If the difference is above the identification threshold then, based on the application, an action will be triggered. For example, in the case of mobile user behavior identification the system might react by requiring the user to identify herself using an alternate channel such as a web questionnaire or another factor of authentication.

3. BEHAVIOR FINGERPRINT

In this section we present the features used to generate the behavior fingerprint and three base methods to represent it along with the metrics to assess the behavior distance. The goal is to choose only methods that store fingerprints in encoded format so that they maintain the privacy of the user data even when the data is stored for longer periods of time in an adversarial environment. The aim behind considering privacy preservation using the suggested methods is to be compliant with the privacy regulations on the network providers. Having a privacy preserving identification system can also allow for the provision of such a service by a third party using the cellular network as a platform. We consider using the following three base methods to represent the set of features for each period of time: 1) a hash set , 2) a standard Bloom filter and 3) a counting Bloom filter. Starting from these three methods we experimented with other three derived combinations of them and reported the experimental results in Section 5.

3.1 Features

In general, a feature set should include measurable elements that have the potential to uniquely identify a user. For example, the set of numbers that the user calls on a regular basis, the duration of the calls with those numbers, the set of numbers that the user sends SMS to, base station identification numbers that serve the device, a list of URLs visited in the data transactions, a set of IP address that the device contacted, etc. However, for the scope of this paper we limited the feature set to the CDR records representing details from SMS and voice call transactions, used by most wireless carriers for billing. Thus, for each user, we consider a set of features containing data from the CDR records corresponding to the SMS and voice transactions within each observation time interval. Table 1 lists the set of features extracted from the CDRs. The left column represents the type of session, the center column contains the feature description and the right column lists the data type for that specific feature. In the next subsections we will describe the three methods to represent the set of features along with the metric used to compute the difference between any two sets of features.

Session	Feature	Type
SMS	# of outgoing msgs	int
	outgoing destinations	list
	SMS timestamps	list
	# of outgoing msgs for destination	list
	base station IDs	list
Voice	# of outgoing calls	int
	outgoing destinations	list
	call timestamps	list
	call duration for destination	list
	base station IDs	list

Table 1: Features for SMS and Voice sessions.

3.2 Hash Set

The first and most simplistic method to represent a set and be able to answer membership queries is to use a hash set. That is, instead of storing each element in the clear we use a one-way cryptographic hash function (e.g., SHA-1) [11] and store only a salted hash value of each element. For each user U and for each time T interval we store a set of hashes of the recorded feature values $S(T)$ together with a per-user nonce n_U that is generated once for each user. Specifically, let H be a cryptographic hash function. Then we define $\rho(S(T)) = \{H(f, n_U)|f \in S(T)\}$.

3.2.1 Jaccard Distance

To measure the distance between two hash sets S and S^*, we use the Jaccard distance, defined as follows:

$$J(S, S^*) = \frac{|S \cup S^*| - |S \cap S^*|}{|S \cup S^*|} \quad (2)$$

With this method, all set membership queries have a precise answer. This comes at the expense of storing the same number of bits for each hash value of each feature in the set, regardless of the size of the element. Moreover, computation time for the Jaccard distance is linear in the size of the sets. This aspect becomes crucial when assessing the scalability of the fingerprinting method to hundreds of millions of users.

3.3 Standard Bloom Filter

A different more space efficient method to represent a set and support membership queries is to use a Bloom filter. A standard Bloom filter is a space-efficient probabilistic data structure used for representing a set in order to support membership queries. It was first introduced by Burton Bloom in 1970 [7]. A Bloom filter is identified by a bit array of size m with all the bits initially set to 0, and k independent hash functions with the range $\{1,\ldots,m\}$. When an element of a set is inserted into the Bloom filter, it is first hashed with all k hash functions and all the corresponding zero bits in the bit array are flipped to 1. If one of the corresponding hash functions bits is already set to 1, then a *collision* occurs and the bit is not changed. When an element is tested for membership in a set, the Bloom filter is queried as follows. First, the element is hashed with all k hash functions and all the corresponding bits in the bit array are checked. If all bits checked are 1 then we say the element was inserted in the Bloom filter with some probability, called the *false positives* rate, introduced because of the collisions in the insertion process. If at least one corresponding bit is 0 then we know precisely that the element was not inserted in the Bloom filter. Thus, a Bloom filter has no false negatives, and space-efficiency is achieved at the cost of a small probability of false positives. After inserting n elements in a Bloom filter, the false positives (FP) rate for subsequent membership queries is given by Equation 3.

$$FP = \left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^k \approx (1 - e^{-kn/m})^k \quad (3)$$

Our second method for representing the feature set is to use a Bloom filter for each time interval for each user. Because a Bloom filter can represent only a discrete set, from the feature set we generate a set of labels and insert them into the Bloom filter. The representation $\rho(S(T))$ is then simply the bit array of the Bloom filter and the hash functions.

3.3.1 Hamming Distance

We compute the difference between two Bloom filters corresponding to two different time intervals by using the Hamming distance [21]. Thus, for any two Bloom filters with their corresponding bit arrays B and B^* the Hamming distance is given by Equation 4:

$$H(B, B^*) = \sum_{i=1}^{m} B(i) \oplus B^*(i) \quad (4)$$

where $B(i)$ represents the bit at index i for bit array B. Note that, unlike the Jaccard distance, the computation time for the Hamming distance is independent from the number of elements inserted in each Bloom filter and is dependent only on the size of the Bloom filter. Since standard Bloom filters can be represented as binary strings, computing the distance can be implemented as a simple XOR operation and a count of set bits.

3.4 Counting Bloom Filter

One limitation of using a standard Bloom filter is given by the fact that the insert operation is indempotent, that is, if the same element is inserted multiple times, the Bloom filter byte array will not change because of the way Bloom filter handles collisions. In many occasions knowing the number of times an element was inserted into a Bloom filter could be a useful feature. For example, a Bloom filter that stores phone numbers dialed by a user in a time interval will not show any difference if the user dials a number once or multiple times. The number of times a user calls a number could represent a feature that one might want to capture to define the usage pattern and user behavior.

In our method we use a variation of the standard Bloom filter, called *counting* Bloom filter [22], introduced for the first time in [14], that can capture the number of times an element was inserted into a Bloom filter. Essentially, a counting Bloom filter uses an array of counting bins rather than a single bit for each array position, representing the number of matches that are encountered for each corresponding position. Thus, there is a tradeoff between the storage savings achieved by the counting Bloom filter and the information stored in the filter. For instance, if the application requires counting bins with a maximum counter up to 255 then an array with bins of size 8 bits will suffice to represent the counters. However, this causes an increase in the storage by a factor of 8 compared to the space required for a standard Bloom filter. Another important property of a counting Bloom filter is the ability to remove elements from the Bloom filter by decrementing the counters corresponding to a given element. This property can also be used when inserting elements in overlapping time intervals. In this case, one can only insert the elements that are new in the new interval, and remove the elements that are no longer present. This avoids the need to re-insert every element into the Bloom filter.

The third method to represent the feature set is to use counting Bloom filters. Similar to inserting into the standard Bloom filter, we insert a set of labels derived from the set of features into the counting Bloom filter. The stored representation $\rho(S(T))$ is then the array of bins along with the definition of the hash functions used.

3.4.1 Euclidian Distance

We compute the difference between two counting Bloom filters corresponding to two different time intervals by using the Euclidian distance. Thus, for any two counting Bloom filters bin arrays CB and $CB*$ the Euclidian distance is given by Equation 5:

$$E(CB, CB^*) = \sqrt{\sum_{i=1}^{m}(CB(i) - CB^*(i))^2} \tag{5}$$

where $CB(i)$ represents the value stored in the i-th bin of the bins array CB. Similarly to the case of standard Bloom filter, the computation time for the Euclidian distance is independent from the number of elements inserted in each counting Bloom filter and it is dependent only on the size of the bins array.

3.5 Contact-based Method

This section proposes a network-based method for behavior identification that improves the one in [26]. The idea is to compare statistical distributions of current and past user's communication activities. Let t_j and c_{t_j} represent a timestamp of a communication activity and a hash of a phone number contacted at time t_j, respectively. Then the pair (t_j, c_{t_j}) is the only information needed to analyze communication activity j.

The training period for this method has two steps. The first step makes the list $L^i = \cup_{t_j \in [\tau_0, \tau_1]}\{c_{t_j}\}$ of hashed phone numbers that user i contacts (calls and messages) during time period $[\tau_0, \tau_1]$. The second step monitors behavior of user i during time period $(\tau_1, \tau_2]$ and stores timestamps of user communications with existing and new contacts:

$$T_E^i = \cup_{t_j \in (\tau_1, \tau_2], c_t \in L^i}\{t_j\} \tag{6}$$

and

$$T_N^i = \cup_{t_j \in (\tau_1, \tau_2], c_t \notin L^i}\{t_j\}, \tag{7}$$

respectively. For user i, the further analysis uses the following three statistics:

(a) maximum number, M^i, of communications with new contacts per time interval, e.g. day, hour, 15 mins, etc.;

(b) α_1-percentile of the distribution of intervals between each two consecutive communications with existing contacts, p^i, where $75 \leq \alpha_1 \leq 100$; and

(c) α_2-percentile of the distribution of intervals between each two consecutive communications with new contacts, P^i, where $0 \leq \alpha_2 \leq 25$.

For each pair (t_k, c_k), the testing stage checks whether contact c_k is in the contact list L^i. If $c_j \notin L^i$ and the time elapsed since the last communication activity with any new contact is less than P^i, the alert score is raised. Also if the total number of communication events with new contacts exceeds M^i, the alert score is raised as well. In other words, the alert score is raised when a user communicates either with new contacts more frequently or with the existing contacts less frequently compared to the past communication history. Specifically, the alert score is raised whenever the time δ_t^e passed since the last communication with the existing contact exceeds p^i and there have been at least h communications with a new contact during this time (the last condition assures that the user is not simply inactive).

4. PRIVACY ANALYSIS

In this section we present an analysis of the privacy and information leakage of user data in our systems. Recall that for user behavior $S(T)$ during time period T, we must record $\rho_T = \rho(S(T))$ in order to later compute the similarity measure $\beta(\rho_T, \rho_{T^*})$ with some other time period T^*. We therefore concentrate on analyzing the information leakage from the stored representation ρ_T of the user data for the four different approaches that we described in Section 3. We consider each case separately. Below, we denote by $\rho_T(U)$ the representation of the recorded behavior for user U during time period T.

Hash sets.

In the hash set solution we store for each recorded feature f, a hash $H(f, n_U)$, where n_U is a random nonce generated for each user, and H is a cryptographic hash function. To analyze the privacy of the scheme we follow the common convention of modeling H as a random function (known as the random oracle model [6]). The representation ρ_T is therefore a set of random strings that is independently chosen for each user. Indeed, if H is random, then the values $H(f, n_U)$ and $H(f, n_{U'})$ are independently random as long as $n_U \neq n_{U'}$. This implies that, for example, one cannot determine whether the same feature appears in both $\rho_T(U)$ and $\rho_T(U')$. However, our representation does reveal whether the same feature f appears in two different time periods T and T' for the same user U. In fact, revealing this information is necessary to compute the Jaccard distance between $S(T)$ and $S(T')$. Additionally, $\rho_T(U)$ reveals the number of features that were recorded for user U during period T.

More formally, one can easily show that the representations of any two multi-user traces with the same feature counts per user are identically distributed, and therefore indistinguishable.

Bloom Filters.

A Bloom Filter is a concise and lossy representation of a set. Therefore, by storing the hash set in a Bloom Filter, we gain all the privacy benefits of the hash set, and in addition we remove a large amount of information due to the conciseness of the Bloom Filter representation. In our solution, we dynamically determine the size of the Bloom Filter for each user during the training period T^* to be an n-bit long array where $n = |S(T^*)|$ is the total number of recorded features. Consequently, we store n bits of information about a set of n features.

To determine the amount of information leakage about the underlying hash set, we must consider the entropy of the recorded features. Recall that an array of n bits has entropy at most n, and suppose that a feature has entropy γ and that all features are independent. Then, the amount of uncertainty left after exposing the Bloom Filter is $n(\gamma - 1)$. For example, for data of 10,000 users, we calculated the average entropy of CDR data to be roughly $\gamma n \approx 3.49n$. Therefore, an adversary that is given the Bloom Filter representation of the user behavior for the training period, has a probability of roughly $1/2^{3.49} \approx 8\%$ of correctly guessing a specific element of the hash set. We calculated the average number of CDR features for a customer is 44, and therefore in expectation the adversary will be able to recover roughly $1/2^{3.49} \cdot 44 \approx 4$ features from the hash set. For the elements that are recovered, we fall back to the basic privacy guarantees of the hash set as described above.

Counting Bloom Filters.

The analysis of privacy when counting Bloom Filters are used is very similar to the analysis for regular Bloom Filters. However, here each bucket is a counter instead of a bit. Suppose that each counter is represented by ℓ bits, then the amount of entropy left after exposing the counting array to the adversary is $\max\{0, n(\gamma - \ell)\}$. Unfortunately, for CDR data, this value is zero. Note that this does not mean that the user data is leaked, but rather that counting Bloom Filters may offer a privacy guarantee which is only as strong as the guarantee provided by the hash set representation[1].

Contact based method.

For the contact based solution, we must store hashed feature descriptions, as in the hash set representation. We additionally store the intervals between phone calls and messages made by the user, and the maximal number of communications with new contacts during the time period. However, if we set $\alpha_1 = 100$ and $\alpha_2 = 0$, we need only store the minimum and maximum intervals between communications to existing contacts. We then obtain essentially the same level of privacy as the hash set.

5. EXPERIMENTS

5.1 Data Set

The study presented in this paper uses 1 month of voice and text message communication data (April 2012) of over 10,000 post-paid cell-phone accounts. The data is obtained from more than 14 million anonymized Call Detail Records (CDR) of one of the major tier-1 cellular providers in the United States. CDRs are records that are logged each time a phone call or text message is sent over the network. In the case where the two communicating entities are connected to the same provider, a duple of records is stored. The Mobile Originated (MO) record logs the data of the transmitting party, while the Mobile Terminated (MT) record stores information of the receiver. We use MO CDRs in this work.

Time	Transmission/Reception time and date
Orig	Originating number
Term	Terminating number
Call type	Mobile originated/terminated SMS/call
Duration	Duration of a voice call
LACCI	Location Area Code and Cell Id

Table 2: Call Detail Record Fields

Table 2 lists the CDR fields handled in our analysis. The originating and terminating phone numbers are completely anonymized. Each transaction is marked with a time stamp and we also record the Location Area Code and Cell Id (LACCI) of the base station that handled the traffic in the uplink (MO) or in the downlink (MT). The data is sampled across US population.

Attribute	Value
# of users	10,000
# of CDRs	14,874,731
duration of data	30 days
# of sessions	2
training data	14 days
testing data	16 days

Table 3: Data characteristics.

Data processing and anonymization

All the data utilized in this analysis is fully anonymized. No personal identifiers, such as phone numbers, are processed, substituting them for hashed values. The results presented in this paper do not include any details or information from the actual processed data, but just aggregated results on the performance of the algorithms *on* the anonymized data.

5.1.1 Training Data for Hash Set and Bloom Filter methods

We use the first 14 days of the original data as the training set for establishing the thresholds for each metric, and the remaining 16 days for generating the testing data. We assume there is no malicious activity in the training data.

5.1.2 Training Data for Contact based method

We use the first 9 days of the original data to make the list of contacts L^i and the following 5 days to store timestamps of user communications with existing and new contacts and compute the three statistics M^i, P^i and p^i and the remaining 16 days for generating the testing data.

5.1.3 Testing Data

We have generated several testing data sets in order to test our methods in different scenarios from the original data collected for the last 16 days, from day 15 to day 30.

Original data: The first testing set is represented by the original data collected from day 15 to day 30 of all data set. We use this data set to assess the False Positives rates, that represent instances when the user was required to authenticate with the system even though she was not supposed to. We will call this data set the original data and the remaining testing sets were generated from this data.

[1]This is a purely information theoretic property. We believe that in practice, it is unlikely that an adversary will be able to extract the entire hash set representation from the counting Bloom Filter.

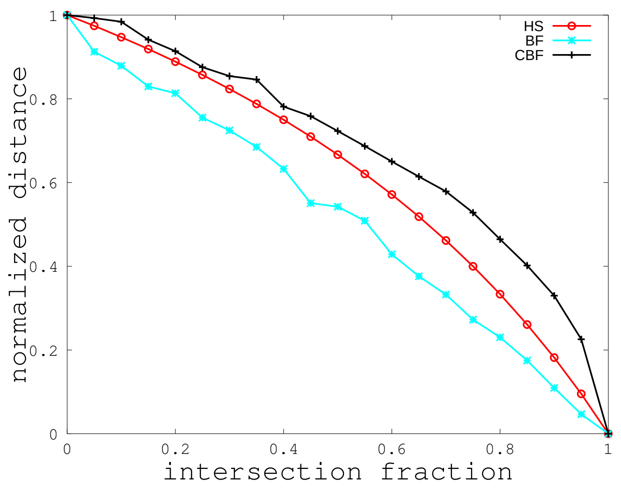

Figure 2: Normalized distance for random sets with variable intersection size.

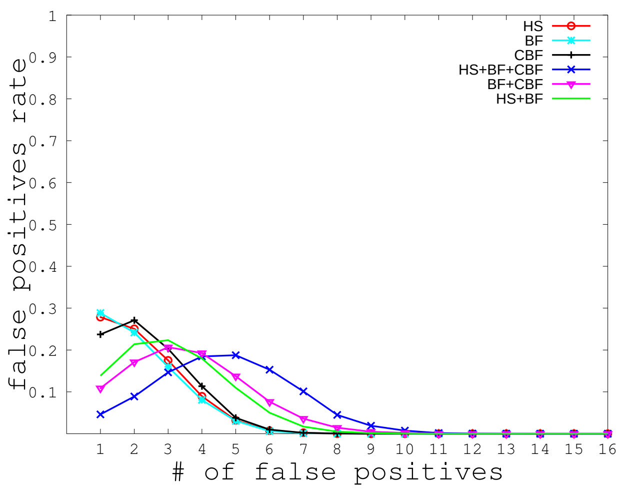

Figure 3: False positives for all methods.

Splicing data: The second testing data set was generated by swapping the original data between any random pairs of two users for the data from day 15 to day 30. This data set was used to test the detection rates when the device is used by an illegitimate normal user.

Random attacker data: Third testing data set was obtained from the original data by generating completely random communication activity for all users. This case was used to measure the detection rate for the cases when, for example, the device is stolen by an attacker that wants to get the most out of using the device in a short period of time. We assume the device connected to the same base stations.

Informed attacker data: Fourth testing data set was obtained from the original data by judiciously generating communication activity that is similar to the original activity. As such, for each testing day we simulated the case when a fraction of the SMS and voice calls, 10% in our experiments, were to the original numbers with the voice calls having zero duration. Note that if the attacker dials the contacts of the victim's phone, she will not be able to impersonate the victim. Therefore, the calls will be of zero duration or, at most, a couple of seconds if the attacker remains silent. We believed is reasonable to assume, an informed attacker cannot withstand a longer duration call with a victims contact in order to not be detected. We assume the device connected to the same base stations.

Malware communication data: This data set was generated to simulate a piece of malware communication activity. For this case we assumed the malware injects some random communication activity within the legitimate user activity. Thus, we generated a fraction of random communication activity, 50% in our experiments, on top of the original user activity. We assume the device connected to the same base stations.

5.2 Hash Set and Bloom Filter Methods

5.2.1 Feature Labels

For each session we consider a set of features that has the potential to uniquely identify a user behavior. For each feature of each session we derive a label that will represent the value of that feature in an encoded and discrete set of session features. Then we will use each of the first three methods described in Section 3 to store the features sets for all sessions and to define the behavior fingerprint. Our behavior fingerprint design can accommodate any number of sessions but for the scope of this paper we are concerned to use features from the CDR records of SMS and voice calls that represent the SMS session and the voice session.

SMS Session: We consider the set of features and the corresponding labels listed in Table 4. For each day time window we generate a label with *zero*, *low* or *high* if the number of SMSes is either zero, smaller or larger than the average training data. For each destination, *dest*, we generate two labels, first with the destination itself and second with the destination concatenated with *high* or *low*, if the number of SMSes to that destination is smaller or larger than the average to that destination in the training data. From the timestamp of each communication activity we create a label from the destination concatenated with the corresponding day time shift. We partition the 24 hour day into three shifts, shift 1 from hour 0 to hour 8, shift 2 from hour 8 to hour 16 and shift 3 from hour 16 to hour 24. Finally, for each time window we create a label with the ID of each base station that the device contacted.

SMS Feature	Labels
# of outgoing msgs	zero, low, high
outgoing dsts w. counts	{dest} & ({dest}+low or {dest}+high)
day time shift	{dest}+1 or {dest}+2 or {dest}+3
list of base station IDs	{ID}

Table 4: SMS features and corresponding labels.

Voice Session: For the voice session we consider a similar set of features and we generate the labels in a similar way for number of outgoing destinations, day time shift and the list of base stations. However, for the voice session we take into account the duration of the voice calls and we generate the label *low* if the voice call has zero duration and *high* otherwise.

Voice Feature	Labels
# of outgoing voice calls	zero, low, high
outgoing dsts w. duration	{dest}, {dest}+low, {dest}+high
day time shift	{dest}+1 or {dest}+2 or {dest}+3
list of base station IDs	{ID}

Table 5: Voice features and corresponding labels.

5.2.2 Implementation, Parameters and Thresholds

We implemented the first three methods presented in Section 3 and several combinations of them. More specifically, we represent the behavior fingerprint using a hash set (HS), a Bloom filter (BF), a counting Bloom filter (CBF) as the base methods. From the base methods, we built other methods by using all three methods together (HS+BF+CBF), only standard Bloom filter and counting Bloom filter (BF+CBF), and hash set and standard Bloom filter (HS+BF).

We used the training data in order to decide on the parameters used and the identification threshold. For each user, we train on the first 14 days of communication activity (SMS and voice call) aggregated in daily slots, therefore 14 slots for training. When building the hash set we applied SHA-256 on each label generated from the data in each time window and we keep the first 128 bits as the hash value for each label.

For building the behavior fingerprints with Bloom filters we use two hash functions. We compute the SHA-256 on each label and then split it into two parts, each part representing a value of a hash function. We decide the size of the Bloom filters by choosing the size that has the potential to yield the most accurate distance between two Bloom filters and counting Bloom filters. We did so, by using Bloom filters to represent two generated sets of random elements that have a variable intersection size and observing the distance function. The size of each set is the average number of labels that we expect to get for each time window for all the users. Figure 2 shows, for each base method, how the distance between the fingerprints varies with the variation of the intersection size for the two random sets. The figure represents the results for using a Bloom filter size equal with the average number of elements inserted. That is, we use 1 bit to represent each label using a standard Bloom filter. We used 1 bin to represent each label in a counting Bloom filter for a total of 16 bits per element of storage for counters with maximum value of 65536 (2 bytes).

In the next step, we get the identification thresholds by computing the day maximum distance D_M, day minimum distance D_m between the fingerprint of the current time window and the fingerprint of the previous day, and the day maximum distance W_M and the day minimum distance W_m compared to the previous week day respectively for all the days in the training data in order to get the day distance variation interval $[D_m, D_M]$ and week variation interval $[W_m, W_M]$. For example for a day of Tuesday, we first get the fingerprint difference between Tuesday and Monday as well as the fingerprint between Tuesday and Tuesday from the previous week and then we compute the maximum and minimum differences among all the day and week distances for all the days.

5.2.3 False Positives

We tested the false positives rate by using the original data test set representing the last 16 days of user CDRs. For each day we compute the difference between the day fingerprint and the previous day and previous week day fingerprints into F_D and F_W respectively. In the analysis, we count a false positive when we obtain that both $F_D \notin [D_m, D_M]$ and $F_W \notin [W_m, W_M]$. More specifically, we count a false positive if the difference between the current day behavior fingerprint and the behavior fingerprint observed in the previous day and in the previous week day are not in the corresponding variation interval. Figures 3 shows the number of false positives that users get for the 16 days of original testing data. The y-axis represents the percentage of users, out of the total of 10,000 that have encountered the number of false positives on the x-axis, for each tested method. For example, the point (1, 0.22) represents that 22% or the users got exactly one false positive in the 16 days of testing data. For the derived methods, HS+BF+CBF, BF+CBF and HS+BF we count a false positive if the day fingerprint distance is not in the variation interval for each base method. By doing so we sum the probabilities of false positives for each method, resulting in larger false positives for derived methods than for the base methods. The figure shows that for the base methods the expected false positive rates for different number of false positives declines with the increase in the number of false positives. For derived methods the false positives rates are low for small numbers and climb up to 22% for 3,4 and 5 false positives in the 16 days of testing with original data.

5.2.4 Detection Rate

We tested the detection rate by using the generated splicing data, the random attacker data, the informed attacker data and the compromised device data for each method. We used the same training data as in the case of false positives testing in order to build the variation interval for each user behavior fingerprint. Figure 4 shows the detection results on splicing data. The x-axis represents the time window when the user was detected as not being the legitimate user, and the y-axis represents the percentage of users not detected for that window. For example the point (1, 0.98) on the HS+BF+CBF curve represents that 98% of the illegitimate users were detected as not being the legitimate users in the first testing time window. The derived method, HS+BF+CBF that combines all the base methods provides the best detection rate of more than 98% from the first testing day, for the splicing and random attacker cases shown in Figure 4 and Figure 5. Figure 6 and Figure 7 show the detection rates for informed attacker and compromised device. In these cases, as the behavior of the illegitimate users becomes more similar to the legitimate user behavior, the detection rate in the first time window starts to decrease to 87% and 45% respectively compared to the random attacker case. However, it increases significantly as more time windows are processed to identify the legitimate user with detection rates over 90% starting with time window 2 in the informed attacker case and time window 6 in the compromised device case.

Note that, because the counting Bloom filter retains more information and the detection rates for counting Bloom filter method are lower than those for standard Bloom filter, the results seem counter-intuitive. These results are explained by the use of Euclidean distance metric to compute the variation interval for counting Bloom filter which yields larger values than the Hamming distance for standard Bloom filter thus, larger variation intervals, hence lower detection rates.

If, in a real deployment each time window is shifted one hour these results show that the illegitimate user that exhibits a completely random network communication behavior can be identified with 98% accuracy in the first hour. Consequently, an informed attacker can be identified with 95% accuracy in the first 2 hours, and a compromised device with 90% accuracy within the first 6 hours. However,

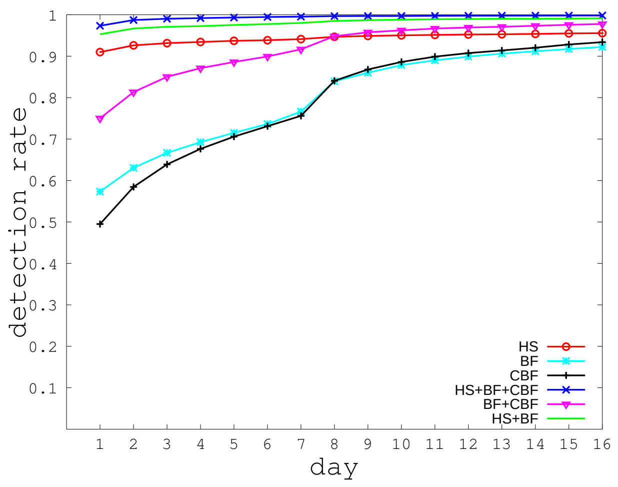

Figure 4: Detection rates for splicing.

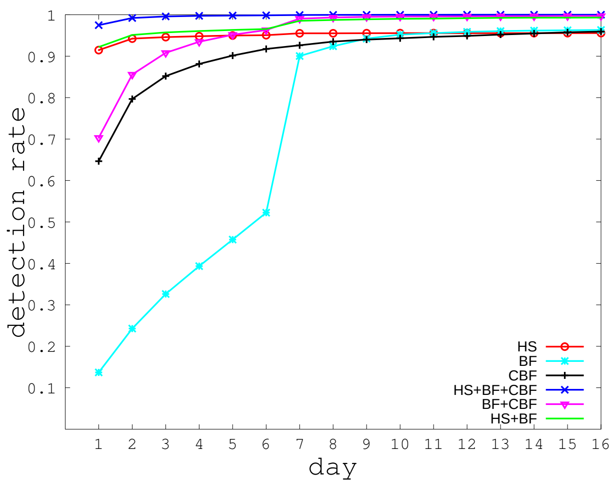

Figure 5: Detection rates for random attacker.

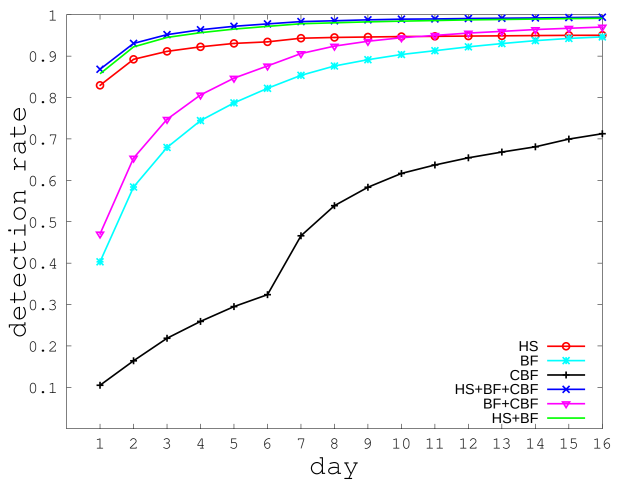

Figure 6: Detection rates for informed attacker.

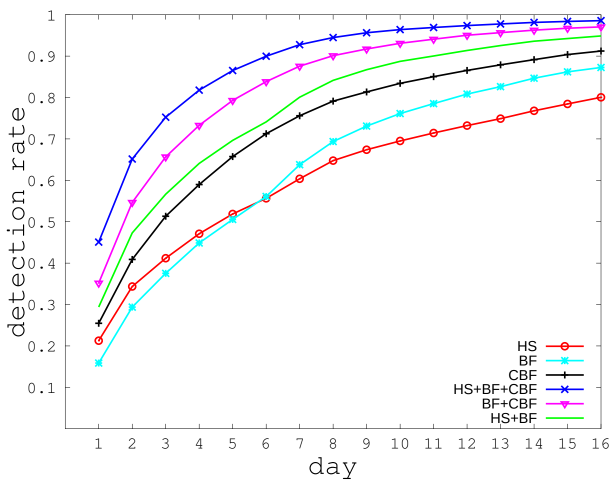

Figure 7: Detection rates for compromised device.

the results presented could be influenced by the parameters used such as Bloom filter sizes, the time window size, the amount of storage and computation available, etc. One could set the expected detection rate to be 99% and then be concerned on the higher number of false positives that could be mitigated by other means, by for example taking into account information other than the CDR records such as data call records, applications, communication content, etc. Nonetheless, our proposed methods could be used to represent other features as well, not only CDR records.

5.3 Contact-based Method Results

Since the contact-based method described in Section 3.5 uses statistical distributions of intervals between consecutive communication activities, it is effective only when the training set for each user contains at least a certain minimum number of events. To test the algorithm we consider only customers with at least 20 network activities (SMS or voice) to compute the probability distributions. This is opposed to hash set and Bloom filter methods, which can be used even when we have a few SMS or voice records available per customer at the training stage. To test this method we set $\alpha_1 = 90$ and $\alpha_2 = 25$.

As it can be seen in Figures 8 and 9, this method achieves very high detection rates in all the scenarios while still performing with acceptable false positive rate, always below 23% of users having one false positive in 16 days. The best detection rate is achieved for splicing, with just 95% of the users detected correctly. In the toughest scenario, a device compromised with malware, the method still achieves a very good result, with 1% users without false negatives.

5.4 Storage and Computation Requirements

The higher the detection rate a method has, the higher the storage and computation requirements. Specifically, the detection accuracy of each method is highly influenced by the information retained by its corresponding fingerprint representation. As such, the BF and CBF methods taken alone seem to provide the lowest detection performance, consistently lower than the hash set method. However, by combining the HS and BF methods into HS+BF seems to improve the HS method in all four scenarios. In other words, by adding an extra bit to the hash value per element for the Bloom filter representation has the potential to significantly improve the detection rate. In all but the compromised device experiments, the HS+BF method ranked the second

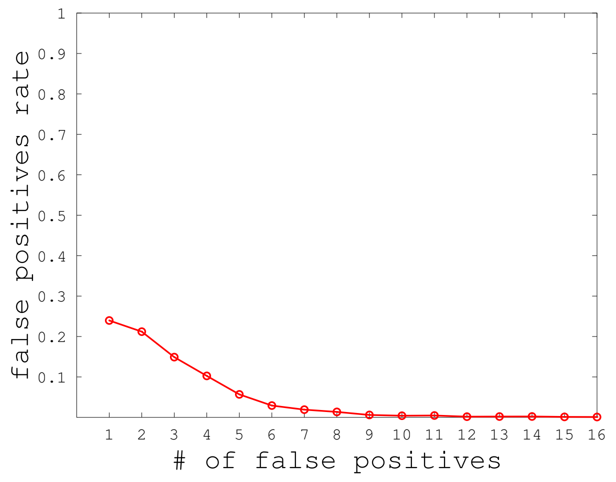

Figure 8: False positive rate for Contact-based Method.

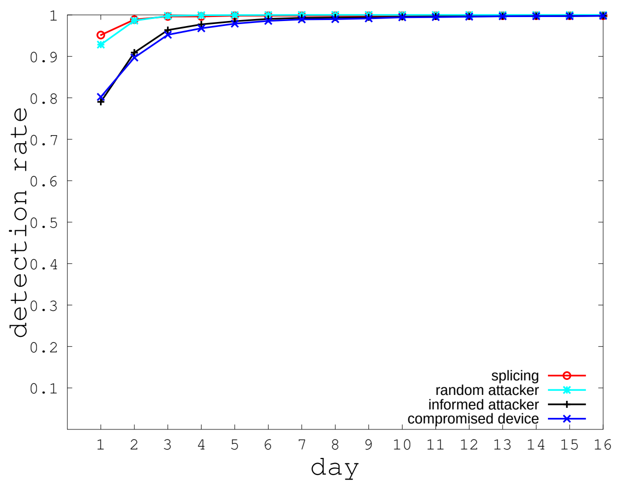

Figure 9: Detection rates for Contact-based Method.

best in terms of detection rate while increasing the storage requirements for the base HS method by a bit per element. Table 6 shows roughly how all the methods compare in terms of storage footprint, fingerprint construction and fingerprint testing.

Method	Storage (bits)	Construction	Testing
HS	128 x N	O(N)	O(N)
BF	N	O(N)	O(1)
CBF	16 x N	O(N)	O(1)
HS+BF+CBF	145 x N	O(N)	O(N)
BF+CBF	17 x N	O(N)	O(1)
HS+BF	129 x N	O(N)	O(N)
Contact-based	128 x N	O(N)	O(N)

Table 6: Storage and processing requirements as a function of the average number N of features for each time window.

Based on the amount of information retained, each method requires a different amount of storage. The standard Bloom filter requires the least while the method where all the sub-methods are combined requires the most storage. Constructing the behavior fingerprint requires roughly the same computational resources for all methods. However, the Bloom filter based methods, because they use the same fixed amount of storage for all time windows, require a constant number of operations when performing the fingerprint checking as compared to the other methods. In a real deployment, depending on the application and on the resource limitation, one could use the combination of HS+BF+CBF methods if the storage and computation requirements can be relaxed while the detection rate is high priority. On the other hand, if computation and storage are limited one could use BF+CBF and sacrifice from the accuracy of the detection but having lower false positives. Finally, if most of the entities to be monitored are active and have at least a certain number of transactions, one could use the Contact-based method that provides both low false positives and detection rates and reasonable storage requirements.

The implementation of the proposed methods can be highly parallelized and optimized based on the fingerprinting method used. For example, when using overlapping time windows, when computing the subsequent fingerprints, one can just discard the features that don't belong to the new time window and insert into the set representation only the new ones. In this way the fingerprint creating time might be reduces.

Finally, because user processing is independent from the rest of the computation for each user, all the users identification fingerprints could be processed in parallel in a separate thread. Moreover, with the increase popularity of distributed computing frameworks, such as MapReduce [12], the computation of data that belongs to million of users could be split and processed independently in parallel by different workers without the need to recombine the results as in the traditional MapReduce Implementation. However, the distributed computation aspect was not the main focus of our paper therefore we don't elaborate it here.

6. RELATED WORK

Several approaches exist in the literature to address the user management, user authentication and user identification problems for mobile terminals. In general, a user is identified on a device either based on something she knows (password), something she has (biometrics) or something she does (behavior analysis). Using passwords is the most common technique for user authentication in telecommunication systems. However, often users choose weak passwords consisting on simple words, short and simple sequences of numbers or easily available information about themselves (i.e. date of birth) [30]. Some solutions have been proposed aiming to protect all the passwords a user requires. A strengthened cryptographic hash function is used to compute secure passwords for arbitrarily many accounts while requiring the user to memorize only a single short password [18]. Biometrics is an alternative scheme for user authentication used in all kinds of applications and devices [25]. In this context, certain efforts aim to achieve highly accurate authentication systems by combining multiple biometric signatures [28]. Other schemes combine biometrics and behavior analysis. These are efficient authentication systems based on the fact that device usage varies significantly from person to person [13]. Implicit authentication systems have been proposed previously in the literature. These are transparent to the user, who is identified based on typing patterns [23] and keystroke dynamics [19]. The approach introduced in our paper resembles these in the sense that user identification is implicit and performed in a transparent way to the user.

Some recent work identifies users based on data from the sensors of a smart phone [20]. Following the same approach and proving its potential, the authors of [29] showed that user input on a smart phone can be inferred from measurement of the sensors on the device, such as the accelerometer and the gyroscope. One may assume that users prefer simple authentication schemes. Indeed, this would explain the frequency in which weak passwords are chosen. Surprisingly, the authors of [16] found that mobile users expect a transparent authentication that increases security and is performed continuously/periodically throughout the day. In this context, the authors of [26] present a method of implicit user authentication for mobile devices. However, in contrast to our methods, their approach is tested with data from just over 50 users and requires large storing of data in the device itself. Further, authentication schemes are also being proposed for different contexts other than mobile device access as it is the case presented in [9] for cloud access.

The major difference between our approach and most of the related methods is that in this work we are not focused on finding another method of authentication but rather a novel non-invasive robust, privacy preserving, accurate and network-based method of user identification that can work in conjunction with and confirm the authentication.

7. CONCLUSION

In this paper we introduce a set of methods to provide implicit and transparent user identification for mobile terminals. Based on the proposed networks, a cellular network can automatically detect device misuse as a result of a lost or stolen terminal or a piece of malicious software running in the background. This provides means for a rapid network-based detection and reaction that can potentially protect a mobile user from economic losses and identity theft.

As future work we intend to investigate using different combinations of contact-based method with the other methods, additional network feature sets and metrics to achieve lower false positive and higher detection rates. We seek to extend the recorded feature sets to the devices themselves and combine events on the device with events on the network. Moreover, we investigate using similar methods to identify users or entities of other devices or systems based on the past observed behavior.

Finally, because user data is very sensitive, an important direction is to design new metrics with very strong privacy guarantees that leak essentially no information about past user behavior, yet allow accurate identification. A promising direction is split the data among several servers, and rely on cryptographic techniques of secure computation to compute the identification outcomes.

8. REFERENCES

[1] GGTracker Android Trojan. `http://goo.gl/apq7eV`.

[2] Security Alert - SpamSoldier. The Lookout Blog, December 2012. `http://goo.gl/7lkRM`.

[3] The most common passwords used online in the last year revealed (and 'password' STILL tops the list). Daily Mail, October 2012. `http://goo.gl/tN9Yhr`.

[4] McAfee Threats Report: Second Quarter 2013. McAfee Labs, 2013. `http://goo.gl/qJPh3e`.

[5] C. Arthur. More than 50 android apps found infected with rootkit malware, March 2011. `http://www.guardian.co.uk/technology/blog/2011/mar/02/android-market-apps-malware`.

[6] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *Proceedings of the 1st ACM conference on Computer and communications security*, CCS '93, pages 62–73, New York, NY, USA, 1993. ACM.

[7] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13:422–426, July 1970.

[8] B. Chen. Cellphone Thefts Grow, but the Industry Looks the Other Way. New York Times, May 2013. `http://goo.gl/DWZwmV`.

[9] R. Chow, M. Jakobsson, R. Masuoka, J. Molina, Y. Niu, E. Shi, and Z. Song. Authentication in the clouds: A framework and its application to mobile users. 2010.

[10] D. Dagon, T. Martin, and T. Starner. Mobile phones as computing devices: the viruses are coming! *IEEE Pervasive Computing*, 3(4):11 – 15, Oct 2004.

[11] C. De Canniere and C. Rechberger. Finding sha-1 characteristics: General results and applications. *Advances in Cryptology–ASIACRYPT 2006*, pages 1–20, 2006.

[12] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. In *Proceedings of the 6th conference on Symposium on Operarting Systems Design & Implementation - Volume 6*, pages 10–10, Berkeley, CA, USA, 2004. USENIX Association.

[13] H. Falaki, R. Mahajan, S. Kandula, D. Lymberopoulos, R. Govindan, and D. Estrin. Diversity in smartphone usage. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pages 179–194. ACM, 2010.

[14] L. Fan, P. Cao, J. Almeida, and A. Z. Broder. Summary cache: a scalable wide-area web cache sharing protocol. *IEEE/ACM Trans. Netw.*, 8:281–293, June 2000.

[15] FortiBlog. Android droiddream uses two vulnerabilities, March 2011. `http://blog.fortinet.com/android-droiddream-uses-two-vulnerabilities/`.

[16] S. Furnell, N. Clarke, and S. Karatzouni. Beyond the pin: Enhancing user authentication for mobile devices. *Computer Fraud & Security*, 2008(8):12–17, 2008.

[17] C. Guo, H. Wang, and W. Zhu. Smart-phone attacks and defenses. In *HotNets III*, 2004.

[18] J. A. Halderman, B. Waters, and E. W. Felten. A convenient method for securely managing passwords. In *Proceedings of the 14th international conference on World Wide Web*, WWW '05, pages 471–479, New York, NY, USA, 2005. ACM.

[19] R. Joyce and G. Gupta. Identity authentication based on keystroke latencies. *Communications of the ACM*, 33(2):168–176, 1990.

[20] J. Kwapisz, G. Weiss, and S. Moore. Cell phone-based biometric identification. In *Biometrics: Theory Applications and Systems (BTAS), 2010 Fourth IEEE International Conference on*, pages 1–7. IEEE, 2010.

[21] J. H. v. Lint. *Introduction to Coding Theory.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1982.

[22] M. Mitzenmacher. Compressed bloom filters. In *Proceedings of the twentieth annual ACM symposium on Principles of distributed computing*, PODC '01, pages 144–150, New York, NY, USA, 2001. ACM.

[23] M. Nisenson, I. Yariv, R. El-Yaniv, and R. Meir. Towards behaviometric security systems: Learning to identify a typist. *Knowledge Discovery in Databases: PKDD 2003*, pages 363–374, 2003.

[24] P. Pachal. Google removes 21 malware apps from android market, March 2011. `http://www.pcmag.com/article2/0,2817,2381252,00.asp`.

[25] S. Pankanti, R. Bolle, and A. Jain. Biometrics:the future of identification. *Computer*, 33(2):46 –49, feb 2000.

[26] E. Shi, Y. Niu, M. Jakobsson, and R. Chow. Implicit authentication through learning user behavior. 2010.

[27] R. Siciliano. Recycled, Lost, Stolen Phones, Equal Identity Theft. FinExtra, February 2009. `http://goo.gl/OVgidl`.

[28] R. Snelick, U. Uludag, A. Mink, M. Indovina, and A. Jain. Large-scale evaluation of multimodal biometric authentication using state-of-the-art systems. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(3):450 –455, march 2005.

[29] Z. Xu, K. Bai, and S. Zhu. Taplogger: Inferring user inputs on smartphone touchscreens using on-board motion sensors. 2012.

[30] J. Yan, A. Blackwell, R. Anderson, and A. Grant. Password memorability and security: empirical results. *Security Privacy, IEEE*, 2(5):25 –31, sept.-oct. 2004.