Personal Marks and Community Certificates: Detecting Clones in Wireless Mobile Social Networks

Marco V. Barbera
Computer Science Department
Sapienza University of Rome, Italy

Alessandro Mei
Computer Science Department
Sapienza University of Rome, Italy

***Abstract*—We consider the problem of detecting clones in wireless mobile ad-hoc networks. We assume that one of the devices of the network has been cloned. Everything, including saved passwords, certificates and secret keys. We propose a solution in networks of mobile devices carried by individuals–composed by nodes that can communicate by short-range technology like bluetooth or Wi-Fi, and links appear and disappear according to social relationships between users. Our idea is to use social physical contacts, securely collected by wireless personal smartphones, as a biometric way to authenticate the owner of the device and detect the clone attack. We introduce two mechanisms: Personal Marks and Community Certificates. Personal Marks is a simple cryptographic protocol that works well when the adversary is an insider, a malicious node in the network that tries to use the stolen credentials in the social community of the original device that has been cloned. Community Certificates works well when the adversary is an outsider, a node that has the goal of using the stolen credentials when interacting with other nodes that are far in the social network from the original device. When combined, these mechanisms provide an excellent protection against this very strong attack. We prove our ideas and solutions with extensive simulations in both simulated and real world scenarios—with mobility traces collected in a real life experiment.**

***Keywords*-Delay tolerant networks, pocket switched networks, clone detection, community authentication.**

I. Introduction

You have left your smartphone on the table at a cafeteria. You soon realize and go back to take it. Fortunately, it is still there! You feel safe—while you are not safe at all. An adversary has connected your smartphone to a laptop and dumped all of its memory, including public and secret cryptographic keys. It is a matter of seconds or, at most, minutes. You do not revoke your certificates and passwords (you feel safe!) but, a month later, you discover that your credentials have been used by someone else. If you think this cannot happen—people take very good care of their personal devices—consider that according to a fairly recent report (WTOP, 15 Nov 2006) 478 laptops have been lost or stolen from the IRS (the Internal Revenue Service is the United States federal government agency that collects taxes and enforces the internal revenue laws) between 2002–2006; 112 held sensitive taxpayer data, including SSNs.

Portable personal devices—smartphones, laptops, and PDAs—are more and more used in our everyday life. We use them to make phone calls, to plan our activities, to surf the web, to manage our banking account, and very soon to make purchases ([31]). We can easily envision a society in which almost everybody will carry a personal computing device of this kind. The system we consider is a network of personal smartphones that connects to each other by using short-range communication technology like bluetooth or Wi-Fi. The nodes are the devices and the links appear and disappear as people move and get in physical touch. As a consequence, the network is not a collection of randomly moving objects—it has a social structure that can be exploited to deliver new and effective solutions to classical networking and security problems.

These networks have been called with several different names—*pocket switched networks* [15], [5], *mobile social networks*, *opportunistic mobile ad-hoc networks*, among others—and have drawn the attention of many researchers in the community. Here, we focus on security. The attack that we have described in the first paragraph of this paper is the *clone attack*. Here we are looking for a solution to this problem. Of course, we do not want one that consists in typing a password every time we use our certificates. People do not like passwords and tend to forget them, or, even worse, choose trivial ones. Our idea is to exploit social physical contacts, securely collected by wireless personal smartphones, as a biometric way to authenticate the owner of the device: Users prove who they are not with something they know (passwords), or something they have (certificate), but with the proof that they actually meet physically and regularly the people they are supposed to meet.

Indeed, our physical contacts are not random. Rather, they are highly influenced by the fact that almost everything in our everyday life is guided by social relationships (e.g. friends, family) and interests (e.g. work, school, gym, or playing chess). If we think of our daytime, it usually follows a do-repeat cycle that might look like this: Go to work (or school); meet with colleagues; finish work and go out with the same old friends or do whatever we like to do during our free time; go back home; go to work again and so on. Of course, each day is somewhat different and we do not always meet the same people, but surely there is a regularity in the communities we live in and in circle of friends that we

978-0-7695-4707-7/12 $26.00 © 2012 IEEE
DOI 10.1109/DCOSS.2012.22

usually meet in person ([14], [13]). We will see how to use this regularity, complemented with other essential mechanisms, to detect the clone attack in wireless networks of smartphones or other personal devices and to prevent the misuse of stolen certificates.

In this paper we introduce two protocols: Personal Marks and Community Certificates. Personal Marks is a simple cryptographic protocol that provides a way to detect the attack when the adversary is an *insider*, a person that tries to use the cloned identity in the community of the victim. Community Certificates is a solution based on certificates that tells how the node is expected to behave in terms of social contacts. If the clone is an *outsider*, a node that wants to use the cloned identity outside the community of the victim, the certificates soon expire, and the clone cannot be used anymore. The *community* of node *i* is the set of nodes that get frequently in physical contact with node *i* (his "best friends"). Communities are a fundamental notion in social networks ([30]), people meet more often other people of the same communities and this intuitive property have been leveraged in the literature in several ways ([16], [17], [2]).

The Personal Marks and the Community Certificates protocols are meant to be used at the same time. According to a large set of experiments made with well-known synthetic and real traces of human contacts computed during real-life experiments, they collectively are able to protect the nodes against the clone attack.

II. Related Work

The detection of the clone attack is one of the most investigated security issue in wireless ad-hoc networks. As far as static wireless networks are concerned, there are three main approaches to the problem: Centralized, local and distributed random based techniques. The centralized techniques like [4] require a base station to collect the location information of the nodes and to check for anomalies (same node ID with different locations). Local-based schemes like [6] make use of voting mechanisms within nodes' neighborhoods to detect clones. Finally, the distributed and random based techniques like [10], [34] require nodes to send signed location information to randomly selected destinations on the network in a hop-by-hop fashion. All these techniques, by relying on fixed geographical position of the nodes in the network are not apt to be used in mobile scenarios such the one we consider [15], [5].

With the spreading of the mobile pay systems (72.8 million of users at the end of 2009, expected 220 million by 2011 in China only [28]), and especially those based on credit card transactions, phone cloning has become a vicious threat to the users portfolio [32]. Thus a lot of research on detecting such attacks have been done, most of which is based on the use of neural networks by the carrier to detect possible anomalies. For a good survey see [7].

The idea of exploiting information regarding social ties between nodes is not new, actually it is common to a good part of the literature on pocket switched networks (PSN) and similar social networks. Much research has been dedicated to the analysis of the data collected during real-life experiments, to compute statistical properties of human mobility, and to uncover its structure in sub-communities [15], [5], [19]. Later on, most of the work in the field focused on message forwarding and to find the best strategy to relay messages in order to route them to destination as fast as possible (see [11], [16], among many others). Also security problematics such as node capture [9] or selfishness [23], [27] have been solved by making use of social relationships among nodes.

The use of biometrics to authenticate has become a well-known area of investigation in computer security. Several biological measurements have been used to identify people to computer systems: voice, face, iris, keystroke dynamics (the way we type is often sufficiently unique to identify ourselves), and others. As an excellent reference and starting point on computer biometrics, see [3].

The authors in [1] propose two intrusion detection systems that have similar biometrics ideas: The first one is build upon Radio Frequency Fingerprinting (RFF), whereas the second one leverages User Mobility Profiles (UMP). However both detection systems are not distributed, and rely on the fact that the intruder (the clone) behaves substantially differently from the real user in terms of geographical movements. Thus, compared with the solutions proposed in this paper, both systems are not able to detect anomalies when the clone behaves similarly to the original node (for example, when the clone attack happens in a building).

In [8], Chaw et al. propose a framework for authentication based on behavioural information collected by a portable device like a smartphone. There are some key differences with our solution. First, they require an always online Data Aggregator that must be invoked during the authentication phase, while our solution is distributed. Second, they are not able to detect an anomaly when the adversary is able to reproduce the behaviour of the victim. Finally, they don't leverage the social characteristics of the user.

To the best of our knowledge, this is the first paper that presents biometric authentication techniques based on the social contacts of the owner of the device to detect the clone attack in mobile wireless networks of smartphones. The techniques, namely *Community Certificates* and *Personal Marks*, are *only* built upon the socially-guided meeting patterns of users in the mobile social network. They are thoroughly orthogonal to trust or voting mechanisms, or geography based techniques to detect cellphone theft or credit card fraud, that are based on different ideas and can be used at the same time.

The rest of this paper is structured as follows: In Section III we give a detailed description of our system and of the model of the adversary; In Section IV we describe the details of the Personal Marks and Community Certificates protocols; In Section VI we show some experimental results; Finally, in Section VII we discuss some interesting extensions to our system.

III. The System

Our network setting is made of last generation smartphones. Smartphones are not-so-small devices that can easily handle video/audio streaming, 3D games, web surfing and SSL sessions, and other applications. Therefore, we can safely assume that nodes are able to perform public key cryptography. The nodes are equipped with public/private key pairs, and the former is signed by a trusted authority CA.

Nodes are loosely time synchronized. Loose time synchronization is very easy to get, if a precision in the order of the second is enough, like in our protocols. We also assume that the trusted authority is able to send a message to any node in the system, for example using the cellular network. When a clone is present, the message is received by the original node and by the clone as well (of course it is perfectly possible that the clone has turned off its interface to the cellular network).

We also assume that the users have access to an alternative way to authenticate to the authority. There are several examples of such mechanisms. One example is GMail: If you forget your password, you can still authenticate by responding to a list of personal questions that, most probably, only you can respond. In other systems, you might be able to authenticate by using a smart-card at your desktop at home. Another example is the PUK code used in GSM mobile phones. In any case, we assume that the alternative mechanism to authenticate is secure but long, burdensome, and we definitely want to use it only in rare and exceptional circumstances like when we need to recover from a clone attack.

In the rest of this paper, we use $\langle m \rangle_E$ to denote a message m signed by the entity E (e.g. a node or the CA).

A. The Adversary and the Problem

We consider a scenario in which an adversary has cloned a device by copying the relevant contents of its memory, including passwords, cryptographic keys and certificates. By storing the stolen data on his device, the adversary is now able to interact in the network by using the victim's credentials.

We assume that the victim is unaware of the attack and keeps on using his device in a normal way. This makes the problem more difficult to solve, since, if the user knew that his identity has been cloned, he could renew his keys and passwords to stop the attack.

Our adversary can be an outsider or an insider. An outsider tries to use the cloned identity outside of the victim's community. On the other hand, an insider is interested in using the cloned identity inside the victim's community. We fight outsiders with the Community Certificates subsystem, and insiders with the Personal Marks subsystem.

Our model of the adversary is rather general: He can turn the cloned device on and off at will, refuse to follow any security protocol and try to eavesdrop any data transmitted or received by any other device (if he is in its transmission range). Also, we don't make any assumption on the way he moves. For instance, he could start following the victim or any other node at all times, though this would require some effort.

A powerful adversary may also infect the victim's device with a malware that is able to continuously steal the certificates and the keys from it. We believe that this kind of attack should be tackled with malware detection techniques ([18]) and we don't address it directly.

IV. The Protocol

In this Section we are going to give a detailed description of the Personal Marks protocol and the Community Certificates and show how, used together, they make it possible to detect the clone attack in mobile networks of smartphones.

A. Personal Marks

Personal Marks is a simple cryptographic protocol that can be used by a node to check if it has been interacting with two different nodes claiming to be the same node. Personal Marks is executed each time a node, say node i, meets a node j in its community. The aim of the protocol is to make it possible for node j to realize if there is another node claiming to be node i. Personal Marks is made of three parts: channel-creation, mark-check and mark-exchange.

The channel-creation is done first and it's meant to make it impossible for an adversary that has cloned node i (i.e. has node's i private key) to eavesdrop the data exchanged between the two nodes. The protocol is inspired by the SSL handshake ([33]): Node i randomly generates a symmetric key and sends it to node j encrypted with the public key of node j. Starting from this moment until the end of the Personal Marks protocol, the two nodes will encrypt their messages using this key. Since the key is encrypted using node's j public key, an adversary who has cloned node i cannot read it. The adversary, however, could also try to perform a man-in-the-middle attack to make node i believe it is node j and eavesdrop the channel. To make sure that he is using the public key of node j, node i can, for example, use the certificate signed by the CA to node j. Another possibility could be that of making node i and node j exchange keys the first time they meet through either the wireless channel (assuming that an attack doesn't happen in that short moment), or, for example, by means a Near Field Communication ([29]).

The mark-exchange consists on the two nodes exchanging a *mark*, that is an object in the form $\langle \text{MARK}, t \rangle_{ji}$, signed by both nodes, where t is a timestamp. To exchange the mark, node j first sends $\langle \text{MARK}, t \rangle_j$ to node i, and node i replies with $\langle \text{MARK}, t \rangle_{ji}$. Note that if the protocol is not completed for any reason (may be for lack of continuous connectivity, or because one of the two nodes is malicious), the peers can safely assume that it has not happened.

The mark-check is done before the mark-exchange protocol when it is not the first time that node j meets node i. Node j sends a request to which node i has to reply with $\langle \text{MARK\_RPL}, t, m \rangle_i$, where t is a timestamp and m is the latest version of the mark node i received from node j. Note than m was signed by both nodes in the previous contact, so node i cannot forge it. The mark-check continues with node j checking if the mark m received by node i is the same as the mark exchanged during the previous contact with node i. If this is not the case, then node j can be sure that the node i met during the previous contact was not the same as the current one. Therefore, one of them must be a clone of the other.

To understand why this is true, assume that node i has been cloned at time t_1 and that node j gets in physical contact with node i for the first time after the attack at time $t_2 > t_1$. In this case both node i and its clone have the same mark $m_0 = \langle \text{MARK}, t_0 \rangle_{ji}$ in memory, where $t_0 < t_1$. Therefore, independently on whether node i is the clone or the original node, the mark-check protocol doesn't fail, and, at the end of the mark-exchange protocol, node i gets a new mark $m_1 = \langle \text{MARK}, t_2 \rangle_{ji}$. The detection is triggered when eventually node j meets at time $t_3 > t_2$ the *other* node, that, during the mark-check protocol, sends the message $r = \langle \text{MARK\_RPL}, t_3, m_0 \rangle_i$ which doesn't pass the test made by node j. Now, node j cannot tell *which* node is a clone, but it can *prove* that node i has been cloned. The proof consists on the pair of messages r and m_1. The reason this is a proof of the attack is that, in r, node i declares that at time t_3 the latest mark was m_0 (generated at time t_0), but at the same time node j has a mark m_1 (generated at time $t_1 > t_0$) signed by node i that contradicts this declaration.

Note that the messages r and m_1 are signed with the key of node i, thus the proof cannot be forged by node j. This is an important detail because it makes it impossible for an adversary to build fake reports and threaten the honest nodes. This proof can be either sent to the authority by using GSM or broadcasted in the network, and the credentials of node i are thus revoked. Then, the legitimate node i is invited to re-authenticate and get new credentials.

B. Community Certificates

A community certificate is a cryptographic object used by a node i to prove that he hasn't abruptly changed his social behavior. When node i joins the system, it automatically enters a training period during which it securely collects signed and timestamped logs of the physical contacts with the other nodes. At the end of the period the logs are reported to the authority. The authority uses the logs to build a signed certificate $ComC_i$ that is sent back to node i. All these messages are encrypted and authenticated. The certificate is of the form $ComC_i = \langle \langle FS_i, FI_i, k_i \rangle_{CA}, SU_i \rangle$.

In the certificate, FS_i is the community of node j; FI_i is a mapping that tells the "strength of the relationship" between node i and his best friends. More specifically, for every $j \in FS_i$, $FI_i(j)$ is a value computed as a function of the inter-contact times between nodes i and j observed during the training period. Moreover, for every $j \in FS_i$, $SU_i(j)$ is an object $\langle \text{TIMESTAMP}, t \rangle_j$ signed by node j where t is a timestamp. This object, called *signed timestamp*, certifies at what time there has been the last contact between node i and j. Signed timestamps are similar to the marks of Personal Marks. However, the two objects *cannot* be used interchangeably.

Before sending a signed timestamp to node i, a node j in the set FS_i must first check node i by using the Personal Marks protocol described in the previous Section. Upon completing the Personal Marks protocol with success, node j sends the signed timestamp to node i. This is an important detail we will talk about in a short time.

Node i receives signed timestamps through an encrypted channel created with the same channel-create procedure used in the Personal Marks protocol (it could even be the same channel created during the execution of the Personal Marks protocol). This prevents the adversary with node i's private key to get a copy of the timestamp through eavesdropping or a man-in-the-middle attack.

Given node $j \in FS_i$, we say that the timestamp $SU_i(j)$ is *fresh* at time t if $t < SU_i(j) + FI_i(j)$. Certificate $ComC_i = \langle \langle FS_i, FI_i, k \rangle_{CA}, SU_i \rangle$ is valid at time t if and only if at least k signed timestamps in set SU_i are fresh at time t. In other words, the certificate is valid only if the node has been able to collect enough fresh signed timestamps by physically meeting people in his circle of friends.

When the authority generates the certificate, SU_i can be prepared with timestamps signed by the authority, just to make the certificate immediately valid. Note that all this process is totally automatic. Note as well that we assume that no attack occurs during the training period. This is quite reasonable indeed, if such an attack takes place, the authority would receive training logs from both the victim and the clone at the same time, thus revealing the attack.

C. Personal Marks and Community Certificates

Personal Marks and Community Certificates, when used together, are able to defend any honest node i from a clone attack. Let us see how: In our system, having a valid community certificate is a requirement for the authentication of a node i to the other nodes. Indeed, when node i wants to set up a communication or to use one of the services provided by a node j, it will first be asked to show a valid certificate $ComC_i$. This applies also in the case when node i needs to get a signed timestamp from node j.

Note that node i should never send the certificate $ComC_i$ directly since this would make it possible for the node receiving it to use node i's identity for the remaining validity time of the certificate. Node i should instead sign and send an object like $\langle ComC_i, t, j \rangle_i$ where t is a timestamp and j is the public key of the node to which the certificate is being sent. This would bind the certificate to both the moment it is being used and to the node that is receiving it, thus making it impossible for any node to reuse it at a later time. Also, the certificate should always be sent through an encrypted channel created in such a way that an adversary who copied node i's private key cannot eavesdrop it (see the channel-create protocol described in Section IV-A).

Let us now consider the scenario where an adversary has just cloned node i, getting, along with all the keys stored in the node, the valid community certificate $ComC_i$. By using the certificate, the adversary can interact with any node j of the system pretending to be node i. Now, if the adversary is an insider, then the attack will eventually be detected by node j

thanks to the Personal Marks protocol. On the other hand, if the adversary is an outsider, he will be able to use node i's identity only until the certificate $ComC_i$ expires.

The only way an outsider adversary can try to keep on using the cloned identity is to get enough fresh signed timestamps from the nodes in the community of node i and use them to maintain $ComC_i$ up to date. However, as explained earlier, before giving a new signed timestamp, a node j in the community of node i will require the adversary to go through the Personal Marks protocol and, therefore, will detect the attack.

This is the *key idea* behind Personal Marks and Community Certificates: While Personal Marks discourages the adversary from using the cloned identity to interact with the nodes in the community of the victim, at the same time Community Certificates requires him to interact with them. Therefore the adversary will either be detected with Personal Marks or will be able to use the cloned identity for a limited amount of time.

In the experiments with real traces, we show that it is possible to choose FS_i, FI_i, and k in such a way that the certificates are continuously and consistently valid when carried by the legitimate owner and expire quickly when carried by outsiders. Therefore, a combined use of Community Certificates and Personal Marks is a good way to provide efficient and secure authentication and to thwart the clone attack.

D. Dynamic Community Certificates

So far, we have described Community Certificates as a static system. However, in real life it may be possible, even if it is not common, that we change our own community. In general, it is reasonable to imagine the following scenarios: (i) our community changes completely since, for example, we move to another town; and (ii) one of our friends moves away, or a new node is our new best friend. Here, we see that it is easy to design protocols to dynamically change the community certificate in a secure way.

In case (i), it is enough to start off a new training phase and to get a new certificate. In case (ii), we can initiate a selective update of the certificate to remove one node, or to add a new one, or to update the parameter of a node that is already part of our ring. Of course, the addition and/or the removal can change all the parameters of the certificate, like mapping FI_i or k_i. The procedure can be easily secured. Indeed, when the procedure starts, the authority sends a GSM message to the node. If a clone requests the procedure to change the certificate according to his own communities, then the message is received by the original owner as well, that promptly detects the attack and sends to the authority a signed request of certificate revocation.

V. Multiple, Coordinated Clone Attacks

In this paper, we consider the problem of detecting a single clone attack. Generally speaking, if the adversary is very powerful, it is however possible that it clones a whole set of mobile nodes. While we do not explicitly deal with this case in this work, it is still useful to see what is going to happen with Personal Marks and Community Certificates.

Let us start with Personal Marks. An insider adversary who clones more than one node has one main advantage: He can use the stolen private keys either to eavesdrop on the encrypted channels that two victims establish during the Personal Marks protocol, or to make the cloned nodes forge marks to each other. In most of the cases, however, this doesn't stop Personal Marks to detect the attack anyway. The only thing the adversary can do is to clone a node i and a node j in node i's community and make sure that, whenever he uses a clone of node i to communicate with node j, at the exact same time he uses the clone of node j to exchange a mark with node i. This would make it impossible for the two honest nodes to detect the attack since they will store the same mark. Moreover, whenever the adversary wants to repeat this procedure, he will first have to eavesdrop a mark-exchange between the honest node i and node j in order to be sure he has the latest version of the mark.

Though in principle this attack is possible, we think that the amount of effort it requires is big enough to drastically reduce the probability that the adversary performs it.

A similar argument can be made for Community Certificates: By using the stolen private keys, an outsider adversary can either eavesdrop on the encrypted channels that two victims establish to exchange the signed timestamps, or make the cloned nodes forge signed timestamps to each other. This means that if the adversary clones a node i and some of the nodes in its community, he could now get signed timestamps for node i without having to exchange marks with the honest nodes. If the number of cloned nodes in the community of node i is high enough, then the adversary can build a valid certificate $ComC_i$ (Section IV-B).

There are two possible ways we can try to fight back this kind of attack. The first one is to use a fairly big value of k in the $ComC_i$. This would tradeoff between the risk for false positives for the node i and the number of nodes the adversary has to clone in order to generate a valid certificate. A second possibility would be requiring that, to be considered valid, $ComC_i$ has to contain a valid $ComC_j$ for each fresh signed timestamp $SU_i(j)$. This extra requirement, however, shouldn't be applied recursively to the internal $ComC_j$'s so as to avoid the certificate to become too big.

In this way we would make it harder for the adversary to forge a valid $ComC_i$ because, in order to do that, he would also have to forge valid $ComC_j$ for at least k_i of the nodes in FS_i. The adversary would therefore be forced to search for a set of nodes which form a clique and can mutually certificate each other. Considering that, in general, communities are not closed sets of nodes ([30]), it should be hard find such a clique. While studying this attack is out of the scope of this paper, it is certainly a open and interesting research direction.

VI. Experimental Results

To validate our mechanisms we use four different data-sets. We first start with describing them and their properties. Then,

TABLE I
DETAILS ON THE DATASETS (DS) AND TRAINING PERIODS (TR).

Data set	Dartmouth	UCSD	Reality	SWIM
Total nodes	1099	32	45	1500
DS AVG active nodes/day	1034	27	37	1500
TR AVG active nodes/day	980	28	38	1500
DS AVG contacts/node/day	283	49	15	132
TR AVG contacts/node/day	263	51	16	131

we show the effectiveness of Personal Marks and Community Certificates in detecting insiders and outsiders in all datasets.

A. Datasets

For the experiments we use an event-driven simulator, fed with 4 traces of three types: WLAN (real)—*Dartmouth* [21] and *UCSD* [24]; bluetooth (real)—*Reality* [12], and *SWIM* [26] simulated traces. In the simulated scenario we are able to select the number of nodes and the length of the experiment. Rather, in the real scenarios (WLAN and bluetooth), the datasets suffer from data loss due to periods of low node activity (e.g. people not always have their devices on), thus we are forced to select a trace period in which nodes are reasonably active.

1) Dartmouth: It contains SNMP logs of the access points across the Dartmouth College campus from April 2001 to June 2004. To infer the contacts between the nodes we follow the assumption widely used in the literature: two nodes can communicate whenever associated to the same access point [25] [5]. From this trace we have selected a time span of 8 weeks, from January 5, 2004 to March 1, 2004, during which 1099 nodes have at least 50 contacts a day for at least the 80% of the days. This ensures us that these nodes remain fairly active during the whole period.

2) UCSD: This trace is part of the Wireless Topology Discover project (WTD) [24]. It contains logs extracted from PDAs carried around campus by a set of about 275 freshmen students of the University of California San Diego. The trace spans a period of 11 weeks between September 22, 2002 and December 8, 2002, during which each PDA periodically recorded the signal strength of all the APs in its range. To infer contacts between nodes we again use the assumption that two nodes communicate when they are associated to the same access point. As reported in [25] the trace is characterized by a steady decline of the user population that especially affects the last two weeks. Thus, we restrict our tests to the first 8 weeks and use the nodes that are fairly active during this period by discarding nodes that record no contact at all for more than 20% of the time. This selection yields a set of 32 nodes.

3) Reality: Differently from UCSD and Dartmouth which are WiFi–based, the *Reality* [12] trace contains the *bluetooth* records collected by 94 cellphones distributed to student and faculty on MIT campus during 9 months (from Sept. 2004 to June 2005). This trace is one of the few bluetooth–based existing traces this long encompassing a relatively large node number. Nonetheless, it includes many nodes which have recorded very few to no sightseeings for long periods of time. In order to keep the amount of active nodes high we thus restrict ourselves to a period of 8 weeks (Oct. 18th–Dec. 13th 2004) and discard the nodes that record no contacts (do not appear) for more than 30% of the time. This selection yields a final set of 45 nodes.

4) SWIM: This trace is generated by the SWIM model ([26], [20]), shown to simulate well human mobility in conference and university campus environments and to properly scale a reference scenario by keeping the nodes density constant. With SWIM we replicate the statistical and social properties of the Cambridge Campus bluetooth data set [22] (only 11 days long) increasing both the number of nodes and the time span while preserving the dynamics of the original real trace in terms of average number of contacts per user: The generated trace contains 1500 nodes and is 8 weeks long. The simulated nodes do not suffer from battery insufficiency or other problematics that generate periods of low activity in the network. Thus, with SWIM we use the whole trace with all the 1500 nodes in it.

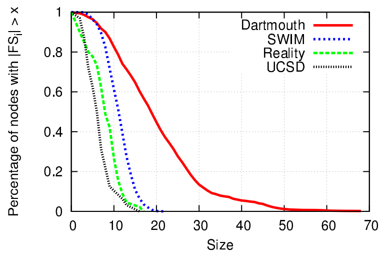
B. Training Period

In order to work, Personal Marks and Community Certificates need that nodes “know” their Friends Set (FS_i) and the values FI_i to be put in $ComC_i$ so that the Authority can incorporate this information in the certificates. For this reason nodes go through a *training period* during which they collect logs of contacts with other nodes. In our experiments we use the first part of each trace as the training period, whereas the remaining of the trace is used to test the performance of our security mechanisms. We have experimentally observed that the training period doesn’t need to be long: Only the first 2 weeks of each trace are enough for our system to quickly detect both insiders and outsiders. Table I includes statistical details of traces and training periods. It is interesting to note that the characteristics of each trace are similar to those of the respective training period, even though the training period is just a subset of 2 weeks of the whole trace.

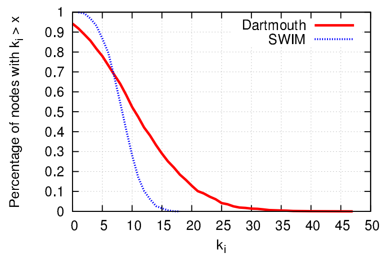
C. Selecting the right friends is good for your phone

Our mechanisms are built upon a given node’s i friend set (FS_i)—the set of nodes whose signed timestamps are collected by node i—and the respective FI_i telling the “strength of the relationship” between the node and his friends in FS_i. As we anticipated in Section IV-B, $FI_i(j)$ defines the period of time in which a signed timestamp given to node i by a node $j \in FS_i$ remains fresh (valid). The choice of FS_i and FI_i is crucial: Large FS_i (too many “best” friends) and big values of FI_i (too long time before expiration) clearly reduce the chances of a honest node to fail proving its identity (the false positive rate). Nonetheless, they also increase the chances that the adversary can use cloned certificates for a long time. Also, a too big FS_i could include nodes that don’t have regular contacts with node i. Thus, this would reduce the effectiveness of Personal Marks in detecting an attempt of the adversary to refresh the community certificate of the victim.

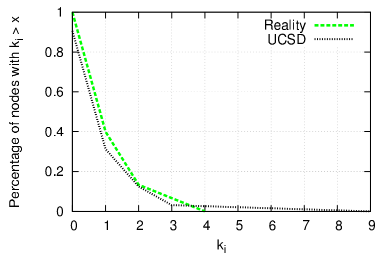
That said, it is meaningful to restrict FS_i to the set of nodes that i is likely to encounter more often. More in details, we define the set FS_i as follows: We compute the average number



(a) Distribution of the sizes of the sets FS_i.



(b) Distribution of the optimal k_i values on Dartmouth and SWIM.



(c) Distribution of the optimal k_i values on UCSD and Reality.

Fig. 1. Distribution of FS_i sizes and parameter k for all traces.

of encounters between i and other network nodes. Then we put j in FS_i if the number of encounters with i is larger than the average value. Simultaneously, we set $FI_i(j)$ to be the average time that usually passes from two consecutive encounters between i and j (their average inter-contact time). In this way, we implicitly include in the certificate i's habitual behavior in the network, that is, his "best friends" and how often he meets them. Note that both FS_i and FI_i are computed taking into account the training period only.

The distribution of the size of FS_i's for all traces is shown in Figure 1(a). In Reality, UCSD and SWIM traces, most of the sets' sizes are between 7 and 15. Instead, in Dartmouth, FS_i sets' sizes are a bit larger—between 10 and 30 nodes. This happens for two reasons: (1) Dartmouth is more than 20 times bigger than Reality and UCSD, in terms of number of nodes; (2) even though SWIM is bigger than Dartmouth, the longer range of the Wi-Fi communication in Dartmouth (vs the bluetooth in SWIM) induces more contacts among nodes. Even so, we want to stress that the size of the FS_i's in Dartmouth is still much smaller than the total number of nodes in the network (at most 2%), meaning that our selection strategy scales well.

D. Inter-contacts and detection

Suppose node i has been cloned, and that the adversary is an insider that can easily meet node i's friends. Take, for example, node $j \in FS_i$. Either node i meets j before the clone does, or vice versa. If the former happens, node i exchanges a mark with friend node j thus making obsolete the clone's mark. As a consequence, the next time the clone meets node j it will be detected. Rather, if it is the clone who first meets node j after the cloning and exchanges marks with it, victim node's i mark will result outdated and the anomaly will eventually be detected during the next meet between nodes i and j. Thus, the amount of time the clone can get away inside i's community is bounded by the time-period between two consecutive meets of i with his friends—the inter-contact time.

In a similar way, the validity of the certificate $ComC_i$ depends again on the inter-contacts of i and its friends. Indeed, FI_i is the average inter-contact time between i and its friends in FS_i. Overall, the better the friends in FS_i of a node i are selected, i.e., the more frequently i meets them, the faster is the detection of insiders (through Personal Marks) and outsiders (through Community Certificates). In the real-life we would expect that each of us has in his FS work colleagues, family members etc.. People that we meet daily or even more often (e.g. office co-workers). And this is what our mechanisms are based upon. However, the real-traces we use for our experiments are not exactly representing the real life: The people taking part in the experiment might not be friends, thus, might not frequent each other with the same rate as it happens to us with our best friends or family. This is confirmed when looking at the inter-contacts of nodes with their friends (nodes in FS) of the real-life traces. The average of such inter-contact time distribution is little more than 2 days in UCSD and Reality, whereas it is little more than 1 day in Dartmouth and little less than 1 day in SWIM. We would expect then that this is also reflected in the clone detection time in each trace.

E. Personal Marks vs Insiders

An insider is one that clones a victim i's device and hangs out in victim's community to keep refreshing his certificate. It can attempt to do so at any moment after the cloning. Suppose, without loss of generality, that the cloning happens after the meeting of i with friend node j. As we already discussed, through the Personal Marks mechanism the anomaly will be detected at most after both the clone and the victim meet with j again (no matter in what order this happens). We don't know when the attacker will meet j, nor we do know when exactly the cloning happened (if right after i meets j or later on). However, the clone is interested in seeing j again, to keep renewing the certificate $ComC_i$ (remind that j is one of the nodes in FS_i, and that $ComC_i$ expires as dictated by FI_i–the averages of inter-contacts between i and $j \in FS_i$). So, to test the effectiveness of Personal Marks we measure, for each node i and each $j \in FS_i$, the expected amount of time it will take for them to meet again starting from any point in time in order to count for the possible instants the attacker could meet j, and possible cloning times. Then we average the results over all network nodes and communities. Figure 2(a) shows, for all traces, the distribution of the detection time of an insider with Personal Marks. As expected, the detection time is shorter with SWIM, where nodes have, on average, shorter inter-contacts. A little bit longer detection times are yielded by the Dartmouth trace, where again inter-contacts are longer than in SWIM. However, in both traces the detection for more than 90% lasts less than 1 day and a half. Accordingly, UCSD and Reality–the two traces with longer inter-contacts among nodes–yield longer detection times than Dartmouth: less than 2.7 days in UCSD and less than 3.5 days in Reality for more than 90% of nodes.

However, we note that in real life typically inter-contacts with the people we meet more often are even shorter than in SWIM: we meet our office co-workers or family members at least twice a day, vs 1 per day as in SWIM. This makes us think that in a real deployment the clone would be detected much faster.

F. False Positives

As we already discussed in Section IV-B a certificate $ComC_i = \langle\langle FS_i, FI_i, k\rangle_{CA}, SU_i\rangle$ is valid at time t if and only if at least k signed timestamps in set SU_i are fresh at time t. That is, if node i manages to refresh his certificate by meeting with at least k of its friends. So, the value k is crucial: It determines the trade-off between high detection performance (high k) and low false positives (low k). We first study the lower bound of the false positives in each trace by setting $k = 1$, the minimum possible value. In our tests we assume that, when the community certificate of an honest node expires, then the user asks the Certificate Authority for a community certificate with a validity of one day. The results of this first test are encouraging. In Reality, only for 8 over 45 nodes we get false positives: The certificate expires once for 4 of them, while for the remaining 4 it expires twice. In UCSD only 3 nodes over 32 show false positives: the certificate expires only once for one node, whereas it expires respectively 2 and 3 times for the remaining 2 nodes. In the case of the Dartmouth trace (1099 nodes) only 63 (less than 6%) show false positives. For all of them, the certificate expires at most 3 times during the whole two months span of the trace. Finally, in SWIM no node at all suffers from false positives.

In Figures 1(b) and 1(c) we show the distributions of the values k_i that we have found to be the largest we can use on each node without generating false positives. As we can see, the optimal values found for the Dartmouth and SWIM traces are bigger than those found in the Reality and UCSD traces. The reason is that, as already discussed, in Dartmouth we have bigger sets FS_i (more nodes) and the range of the nodes is higher, while, in SWIM, nodes have more contacts and, consequently, more chances to exchange signed timestamps with respect to the Reality and UCSD traces. By comparing the values k_i with the sizes of the corresponding sets FS_i, we have noted that in the UCSD and Reality traces, choosing a value of k_i around the 10% of the size of FS_i is good enough for the 90% of the nodes. In Dartmouth, for 90% of the nodes, k_i can be chosen to be around the 25% of the size of FS_i, while in SWIM it can be set to the 50%. Taking these values into consideration, we believe that a good initial choice of the value k_i could be the 10% of the size of FS_i and, in case this doesn't produce false positives, k_i could be safely raised up to values around the 20% or 25% using a tuning procedure like the one described in Section IV-D.

G. Community Certificates vs Outsiders

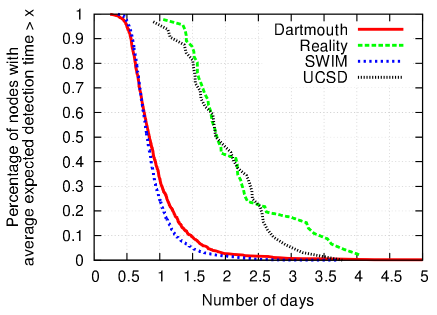
After we studied the lower-bound of the false positive rate with the Community Certificates we are ready to study the effectiveness of the mechanism in detecting outsiders, that is, attackers that try to get away with cloning a victim's device by using the device far from the victim's community, where the Personal Marks mechanism would promptly detect the attack. To do so we compute the average expiring time of a cloned certificate (e.g. from a legitimate node i) in absence of contacts with nodes in the set FS_i. This gives us a measure of the expected amount of time the adversary has available to use a cloned identity outside the victim's community. In the experiment, we set k_i to be, for each node, the highest value for which the node is not affected by false positives. The results of our measurements are plotted in Figure 2(b). The first observation we make is that in all traces the duration of the certificate is never more than 2 or 3 days at worst (UCSD and Reality, respectively), and roughly between 0.5 and 1.5 days at best (SWIM and Dartmouth, respectively). Again, the detection is faster in traces with shorter average inter-contacts among nodes (SWIM), which makes us think that in real-life the performance of the mechanism would be even better.

VII. Add-ons and Caveats

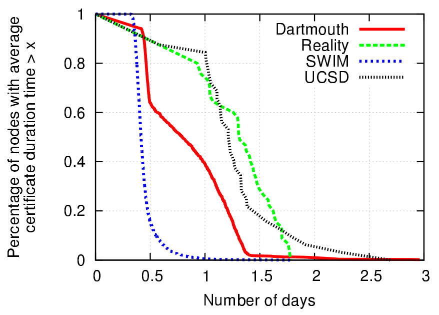
Community Certificates can be complemented and extended in such a way to design more sophisticated and complete versions. Here we discuss two of these addons.

a) Traveling: If we travel, especially alone, our community certificate is going to expire soon. While this is generally true, we can easily design a solution. The user can prepare two different *profiles*, in the same certificate, that can be selected when changing community. Of course, it is fundamental that these solutions can be used only by the the legitimate owner by performing the burdensome alternative authentication that is used in case of exceptional events.

b) Fixed Infrastructure: The certificate authority may potentially put into the set FS_i some *fixed nodes* like Wi-Fi access points in a University Campus. In such a scenario it would be perfectly reasonable for a student to have stored in the set FS_i of his/her smartphone also the public key of the access points of the university buildings he/she frequents the most. There would be two good reasons for doing that. The first one is that the access points have fixed locations and are, usually, always on. The second one is that it is supposedly harder to tamper with an access point and to move it to another location rather than a mobile device. This would make it more difficult for the adversary to refresh the community certificate of a cloned identity i by compromising a given number of nodes in the set FS_i (see Section V). For these reasons we found it interesting to test the performance of our system in a scenario where there is a set of fixed Wi-Fi access points implements both the Personal Marks and Community Certificates protocols. Due to lack of space however, we will not present here the results we obtained in this case. Nevertheless, our experiments have shown that using fixed



(a) Distribution of the average expected detection time of insiders.



(b) Distribution of the average community certificates durations.

nodes can improve the detection time of both outsider and insider nodes and also decrease the number of false positives.

VIII. Conclusions

In this paper we introduce Personal Marks and Community Certificates. The fundamental idea of Community Certificates is that, in networks of mobile people, authentication can be based on the notion of community, and nodes can authenticate by showing that they indeed meet the people that are part of their community. While the social structure of these networks has been extensively used in networking, to the best of our knowledge this is the first time that this has been used as a biometric to authenticate device. We also present Personal Marks, a way a community can use to protect itself against insiders performing a clone attack. The combined used of these mechanisms deliver an excellent protection of the social mobile network against the clone attack. Indeed our experiments show that the detection is fast enough considered the slow dynamics of the trace we have used.

References

[1] M. Barbeau, J. Hall, and E. Kranakis. Detecting impersonation attacks in future wireless and mobile networks. In *MADNES*, 2005.

[2] M. V. Barbera, J. Stefa, A. Carneiro Viana, M. Dias de Amorim, and M. Boc. Vip delegation: Enabling vips to offload data in wireless social mobile networks. In *DCOSS*, 2011.

[3] R. Bolle, S. Pankanti, and A. K. Jain. *Biometrics, Personal Identification in Networked Society: Personal Identification in Networked Society*. Kluwer Academic Publishers, 1998.

[4] R. Brooks, P. Govindaraju, M. Pirretti, N. Vijaykrishnan, and M. Kandemir. On the detection of clones in sensor networks using random key predistribution. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 2007.

[5] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott. Impact of human mobility on opportunistic forwarding algorithms. *IEEE Transactions on Mobile Computing*, 2007.

[6] H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks. In *IEEE Symposium on Security and Privacy*, 2003.

[7] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 2009.

[8] 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. In *CCSW*, 2010.

[9] M. Conti, R. Di Pietro, A. Gabrielli, L. V. Mancini, and A. Mei. The smallville effect: social ties make mobile networks more secure against node capture attack. In *MobiWac*, 2010.

[10] M. Conti, R. Di Pietro, L. V. Mancini, and A. Mei. Distributed detection of clone attacks in wireless sensor networks. *IEEE Transactions on Dependable and Secure Computing*, (2010).

[11] E. M. Daly and M. Haahr. Social network analysis for routing in disconnected delay-tolerant manets. In *MobiHoc*, 2007.

[12] N. Eagle and A. S. Pentland. CRAWDAD data set mit/reality (v. 2005-07-01). Downloaded from http://crawdad.cs.dartmouth.edu/mit/reality, July 2005.

[13] N. Eagle and A. S. Pentland. Eigenbehaviors: identifying structure in routine. *Behavioral Ecology and Sociobiology*, 2009.

[14] M. C. Gonzalez, C. A. Hidalgo, and A.-L. Barabasi. Understanding individual human mobility patterns. *Nature*, 2008.

[15] P. Hui, A. Chaintreau, J. Scott, R. Gass, J. Crowcroft, and C. Diot. Pocket switched networks and human mobility in conference environments. In *SIGCOMM*, 2005.

[16] P. Hui, J. Crowcroft, and E. Yoneki. Bubble rap: social-based forwarding in delay tolerant networks. In *MobiHoc*, 2008.

[17] P. Hui, E. Yoneki, S. Y. Chan, and J. Crowcroft. Distributed community detection in delay tolerant networks. In *MobiArch*, 2007.

[18] M. Jakobsson and K.-A. Johansson. Retroactive detection of malware with applications to mobile platforms. In *HotSec*, 2010.

[19] T. Karagiannis, J.-Y. Le Boudec, and M. Vojnović. Power law and exponential decay of inter contact times between mobile devices. In *MobiCom*, 2007.

[20] S. Kosta, A. Mei, and J. Stefa. Small world in motion (SWIM): Modeling communities in ad-hoc mobile networking. In *Proceedings of The 7th IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON 2010)*, 2010.

[21] D. Kotz, T. Henderson, I. Abyzov, and J. Yeo. CRAWDAD data set dartmouth/campus (v. 2009-09-09). Downloaded from http://crawdad.cs.dartmouth.edu/dartmouth/campus, Sept. 2009.

[22] J. Leguay, A. Lindgren, J. Scott, T. Riedman, J. Crowcroft, and P. Hui. CRAWDAD trace upmc/content/imote/cambridge (v. 2006–11–17), November 2006.

[23] Q. Li, S. Zhu, and G. Cao. Routing in socially selfish delay tolerant networks. In *INFOCOM*, 2010.

[24] M. McNett. Wireless topology discovery. http://my.url.com/, 2008.

[25] M. McNett and G. M. Voelker. Access and mobility of wireless pda users. *SIGMOBILE*, 2005.

[26] A. Mei and J. Stefa. Swim: A simple model to generate small mobile worlds. In *INFOCOM'09*, 2009.

[27] A. Mei and J. Stefa. Give2get: Forwarding in social mobile wireless networks of selfish individuals. In *ICDCS*, 2010.

[28] Mobile payment in china. http://www.mobilepaymentchina.com/mobile-payment/.

[29] Near field communication. http://www.nfc-forum.org/home/.

[30] G. Palla, I. Derényi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 2005.

[31] Google wallet. http://www.google.com/wallet/.

[32] R. Singh, P. Bhargava, and S. Kain. Cell phone cloning: a perspective on gsm security. *Ubiquity*, 2007.

[33] D. Wagner and B. Schneier. Analysis of the ssl 3.0 protocol. In *USENIX*, 1996.

[34] B. Zhu, S. Setia, S. Jajodia, S. Roy, and L. Wang. Localized multicast: Efficient and distributed replica detection in large-scale sensor networks. *IEEE Transactions on Mobile Computing*, 2010.