Analyzing Key-Click Patterns of PIN Input for Recognizing VoIP Users

Ge Zhang

Analyzing Key-click Patterns of PIN Input for Recognizing VoIP Users

Ge Zhang

Karlstad University, Universitetsgatan 2,
65188, Karlstad, Sweden
ge.zhang@kau.se

Abstract. Malicious intermediaries are able to detect the availability of VoIP conversation flows in a network and observe the IP addresses used by the conversation partners. However, it is insufficient to infer the calling records of a particular user in this way since the linkability between a user and a IP address is uncertain: users may regularly change or share IP addresses. Unfortunately, VoIP flows may contain human-specific features. For example, users sometimes are required to provide Personal identification numbers (PINs) to a voice server for authentication and thus the key-click patterns of entering a PIN can be extracted from VoIP flows for user recognition. We invited 31 subjects to enter 4-digital PINs on a virtual keypad of a popular VoIP user-agent with mouse clicking. Employing machine learning algorithms, we achieved average equal error rates of 10-29% for user verification and a hitting rate up to 65% with a false positive rate around 1% for user classification.

1 Introduction

Current Internet users heavily rely on distributed networking intermediaries to transmit packets. These networking intermediaries might be compromised and thus cannot be simply trusted by users. Malicious intermediaries can wiretap their relayed packets for man-in-the-middle attacks. This threat is also an issue for Voice over IP (VoIP) services. Previous research [28] shows that it is easy for an intermediary to detect the availability of VoIP conversation flows between two hosts without reading the flow details. This actually reveals the VoIP calling records which include the IP addresses used by the conversation partners, the starting and ending time of the conversations. Other work [27,26] proposed more advanced method using watermark to increase the robustness and accuracy of the calling records detection. Calling records could reveal daily life of a user. For instance, spammers may infer the requirements and preference of a particular user from the calling records (e.g, a recent calling record showing that a user calls a dentist reveal that the user might have dental problems) so that they can send advertisements more effectively. Recently news report that third parties offer traditional telephone calling records for profit [1]. With the increasingly deployment and usage of VoIP services, it can be predicted that the confidentiality of calling records will be an important issue on VoIP as well.

Nevertheless, previous VoIP tracking methods [27,26] at most enable intermediaries to find out the calling records between two hosts identified with their IP addresses (IP-level calling records). The linkability between a VoIP user and a IP addresses is usually unstable: VoIP users may move from one network to another network with their laptops, or switch between devices (e.g., the user switches from the home computer to an office workstation). Therefore, a user may use different IP addresses at different times. In addition, even one IP address might be shared by several users due to current limited IP address space [18]. Thus, the IP-level calling records are not accurate enough to attack a particular user. In this case, attackers require user-level calling records. To solve this, attackers need to extract human-specific characteristics from flows for user recognition. Previous papers [20] [8] verified that speech features can be extracted to re-identify a user if the user employs a specific codec. This paper investigate another alternative by taking advantage of user key-click patterns: Some automated voice services require users to provide their Personal identification numbers (PINs) for authentication (e.g., access a voice mailbox or a configuration setup). In this situation, users enter their PINs on their VoIP user-agents so that the user-agents generate specific packets to indicate which keys have been clicked. Thus, the key-click pattern for PIN input is a potential characteristic for user recognition. We addresses the following research questions in this paper: (1) How can a malicious intermediary recover the key-click patterns from intercepted VoIP flows? (2) How to minimize the impact from networking conditions (e.g., jitter, packet loss)? (3) Is the recovered key-click patterns accurate enough for user recognition? To answer these questions, we invited 31 subjects to participate in the experiments. Each of them entered 4-digital PIN codes on a popular VoIP user-agent by mouse clicking. Employing machine learning algorithms, we achieved average equal error rates of 10-29% for user verification and a hitting rates up to 65% with a false positive rate around 1% for classification. Finally, we also discuss corresponding countermeasures to prevent user recognition.

The rest of this paper is organized as follows. Section 2 introduces some background information about VoIP. Section 3 introduces the general idea of the attack method. Section 4 presents the preparation, procedure and results of the experiments that we conducted. Section 5 lists related work. Finally, we summarize this paper in Section 6.

2 Background in VoIP flows

The Realtime Transport Protocol (RTP) [24] standardizes the packet format for VoIP conversations. RTP provides end-to-end delivery schemes for data with real-time characteristics over IP networks. It supports a variety of payload types, two of which are especially related to this paper.

- Voice payload: In a conversation, a user-agent constantly encodes acoustic signal into digital data as voice payloads using a codec. The user-agent on the other side recovers the acoustic signal by decoding the payloads from

the received RTP packets. We name this kind of RTP payloads as *RTP voice packets*. In many cases, a user-agent continuously generates RTP voice packets at a constant time interval (e.g., 20 ms) in a conversation unless other types of RTP packets with higher priority are triggered.

– Event payload [25]: When a user clicks a phone key in a conversation, the user-agent generates RTP packets with event payloads to indicate which key has been clicked. The RTP packets with event payloads are called *RTP event packets*. RTP event packets have higher priority than RTP voice packets.

The Secure Realtime Transport Protocol (SRTP) scheme [10] has been widely applied to protect RTP packet payloads by encrypting. However, it does not protect RTP headers[1]. This means that RTP header fields are available to intermediaries despite of the protection. Several RTP header types are introduced as follows:

– Marker bit: It indicates the beginning of a new event. For instance, a user presses a key "4" may span a series of RTP event packets, but only the first packet has the marker bit set. We define that a RTP event packet with marker bit set is a *key-down* packet and the others are *key-holding* packets.
– Payload type: It identifies the type of the RTP payload.
– Sequence number: The RTP sequence number is incremented by one in each successive RTP packet sent. The sequence numbers are assigned to RTP packets to allow the receiver to restore the original sequence in case of unreliable transmission.

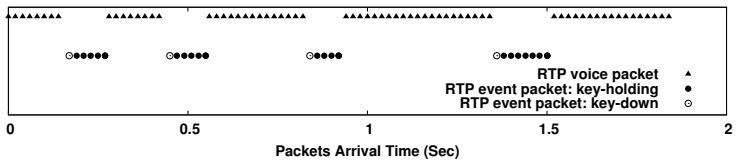


Fig. 1. The packet inter arrival time of an example RTP flow

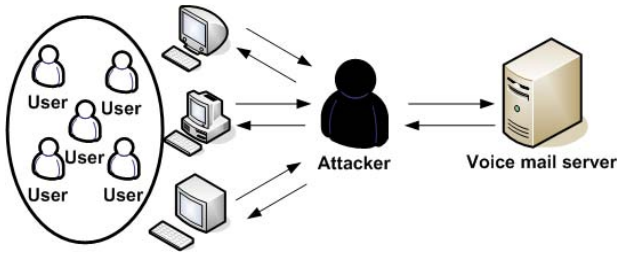
Figure 1 plots a typical RTP flow with both event and voice packets. The packet inter-arrival time is around 20 ms. It can be predicted that 4 events are represented in this flow. Each event contains 1 key-down packet and a set of key-holding packets.

[1] RTP headers are sent in the clear to allow for billing purposes and header compression.

3 Attacking method

Image the following scenario illustrated in Figure 2(b): There are several users whose user-agents share the same IP address. An attacker is a malicious intermediary which intercepted a number of RTP flows originated from this IP address. The packets arrival time in the flows are recorded. Some flows have been already correctly labeled with their originator users (we call these flows as *testing flows*). The rest without being labeled are called *testing flows*. The attacker aims to further profile user-level calling records using the testing flows. Thus, the attacker may want to (1) label the flows for a particular user; or (2) label the flows for all users. We assume that the users occasionally access their voice mail by providing their PINs on a virtual keypad of a user-agent using a mouse[2] (see Figure 2(a)). In addition, all RTP flows in the environment are encrypted by using SRTP [10].

(a) The virtual keypad



(b) The network topology

Fig. 2. The environment of experiments

If the attacker detects a VoIP flow for PIN input and recognizes its originator user, it is highly possible that the calls at the time around are done by the same user. In this way, we consider to recognize a user by key-click patterns. The challenge is how to recover the key-click patterns from RTP flows? Given an encrypted RTP flow, the attacker firstly needs to distinguish the RTP event packets and RTP voice packets. Actually it is rather simple: Despite of the protection by SRTP, the RTP headers are still in plain text. Thus, the attacker can distinguish them directly by reading the "Payload-Type" header fields. After picking RTP event packets from a flow, the next step is to restore the key-click behavior. To input 4-digital PIN code, 4 key-click events are generated. The attack can observe 4 key-down packets with following key-holding packets from the marker bits in headers. Moreover, we define the last key-holding packet for each event as the *key-up* packet. The attacker can restore a key-click pattern by guessing 4 key-holding periods (The period between a key-down packet and its following key-up packet) and 3 key-switching periods (the period between a key-up packet and the next key-down packet). Let us take the example flow in

[2] Some user-agents may also support keyboard input, but in this paper we only consider virtual keypad input.

Figure 1, the 4 key-holding periods are: 0.17-0.28s, 0.45-0.58s, 0.85-0.92s and 1.3-1.5s. The 3 key-switching periods are 0.28-0.45s, 0.58-0.85s and 0.92-1.3s.

The 4 key-holding periods and 3 key-switching periods are taken as variables for training and testing. Then, the attacker can employ a learning algorithm to construct a classifier, which builds key-click pattern using the training data and classifies the testing flows into correct classes. We did several experiments and the detailed work will be introduced in the next section.

Nevertheless, current Internet does not guarantee the quality of packet transmitting. Since this attack needs exact inter-packet arrival time of RTP event packets, the varying network quality (namely **jitter** and **packet loss**) could lead to an inaccurate observation. (1) Jitter indicates latency variations for different packets in a flow. For instance, a packet sent earlier may arrive the destination later. A large jitter on RTP event packets could make the key-click pattern recovering unreliable. Nevertheless, the sequence number on packet header field can help attackers to restore the original packet sequence. Moreover, attackers know what the fixed time interval between two successive packets should be (e.g., 20 ms). In this way, attackers can restore the packet inter-arrival time and sequence. Thus, the impact of jitter is not vital. (2) Packet loss indicates the amount of packets which are accidentally dropped in the transmission. Although attackers can detect packet loss rate by reading sequence number, they do not know the type of the lost packet for key-click pattern recovery. However, the attacker can heuristically guess it by the types of its neighbor packets. For example, if the lost packet is a key-holding packet in the middle, it is also easy for attackers to guess since the packets before and after are the same type.

4 Experiments

4.1 Data Collecting

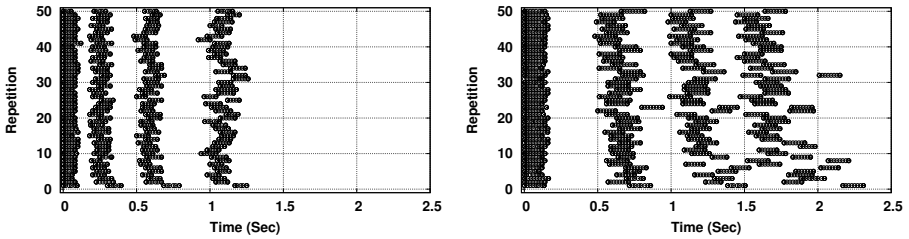
To test the performance of this kind of attack, we did a series of experiments. We invited 31 students as test subjects who are denoted by $S = \{s_1, s_2, \cdots, s_{31}\}$. All of them have experience with using a computer, a mouse and a VoIP client. Each subject was asked to input two kinds of PIN codes. First, we randomly generated 31 different 4-digital PIN codes and thus assigned these PIN codes to the subjects one by one. Each subject was asked to input his/her unique PIN code using the mouse on the virtual-keypad of the X-Lite [7], a popular user-agent for 50 repetitions. We call these repetitions as *unique input repetitions* and let $du(s_i)_j$ to denote repetition j done by subject s_i. Furthermore, some users may have the same PIN code since there is only a 10^4 space for a 4-digital PIN code. Taking this into account, we arbitrarily selected a particular PIN code ("9913") and again asked each subject to input it for 50 repetitions. Similarly, we call the repetitions as *a shared input repetitions* and use $ds(s_i)_j$ to denote these repetitions.

We employed two computers: one ran X-Lite (version 3.0) as the user-agent. It is equipped with a DELL 19-inch flat panel LCD screen with 1024x768 pixel

resolution. The mouse is a HP USB optical wheel mouse with the default speed setup on Windows XP platform. Another ran TCPDump [5] to simulate attackers to intercept RTP flows.

4.2 Data Processing

Following the method introduced in Section 3, we first extract RTP event packets from each flow. Figure 3(a) and Figure 3(b) illustrate the arrival time of RTP event packets of the unique input repetitions done by s_{12} and s_{17}. ($du(s_{12})_j$ and $du(s_{17})_j$, $1 \leq j \leq 50$). At a glance, readers can find the general key-click patterns are different for the two users. Then, we restore the 4 key-holding periods and 3 key-switching periods for each repetition using the method introduced in Section 3.



(a) $du(s_{12})_j$, $1 \leq j \leq 50$ (b) $du(s_{17})_j$, $1 \leq j \leq 50$

Fig. 3. Generated RTP event packets of the unique input repetitions done by s_{12} and s_{17}: one dot indicates one RTP event packet and the x-axis indicates packets inter-arrival time.

4.3 Learning algorithms

We employ three popular learning algorithms in this paper since these algorithms have been widely tested and performed well in previous work on keystroke biometrics [23,22]:

- Supporting Vector Machine (SVM) [11]: SVM works by constructing an N-dimensional hyperplane that groups the data into two spaces. In principle, it only solves the two-class problems. For multi-class problems, the algorithm can repeatedly perform the operations over all the possible two-class pairs and then find the suspect class by a voting mechanism.

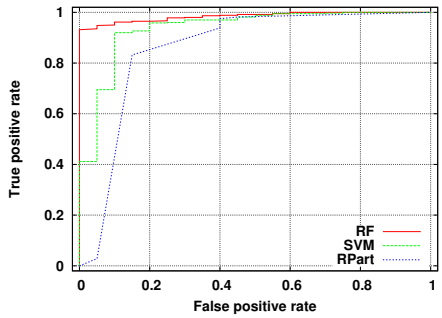
– Random Forest (RF) [12]: Random forest is an ensemble learning method by generating a large number of bootstrapped classification trees and aggregating them during the training. Different to SVM, random forest can perform variable selection by itself and thus it is robust against noise. In a previous comparison, random forest provides a better predictive accuracy than other learning algorithms [14].
– Recursive partitioning (RPart) [13]: Recursive partitioning is a tree-based method for classifying data. It creates a classification tree and further splits the tree based on the condition of variables. The split process will be repeated for each leaf node until a certain stop splitting condition is met. Recursive partitioning is a fast classification algorithm.

We implemented the classifiers in the R platform (version 2.12.0) [6], which provides a wide variety of statistical functions including classification based on the S language. In this paper, we implemented our classifiers using e1071 (SVM) [2], random forest [3] and recursive partitioning [4] packages. By using these packages, we can focus on our classification problems rather than the detail implementation of the algorithms.

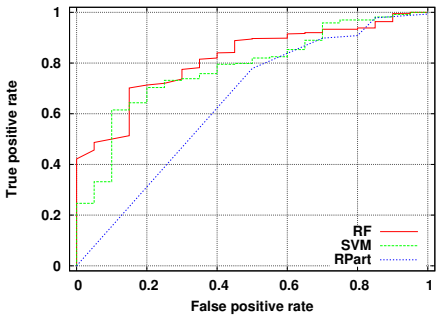
4.4 Analysis and results

We recovered the 7 variables (4 key-holding periods and 3 key-switching periods) from each repetition. For each subject, we selected the first 30 repetitions ($du(s_i)_j$ and $ds(s_i)_j$, $1 \leq i \leq 31 \wedge 1 \leq j \leq 30$) for training and took the rest 20 repetitions ($du(s_i)_j$ and $ds(s_i)_j$, $1 \leq i \leq 31 \wedge 31 \leq j \leq 50$) for testing. In this paper, we focus on two problems, namely user verification and user classification.

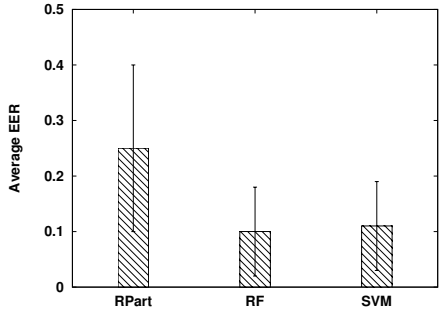
– User verification: Given a testing repetition and a specific user s_x, the attacker asks the classifier whether the testing repetition was done by s_x. Thus, it is a binary classification problem (the real user s_x and imposters $s_{i \neq x}$). In this way, we split the training repetitions into 2 classes. One contains the repetitions done by user s_x ($du(s_i)_j$ or $ds(s_i)_j$, $i = x \wedge 31 \leq j \leq 50$) and another contains all the remaining repetitions ($du(s_i)_j$ or $ds(s_i)_j$, $i \neq x \wedge 31 \leq j \leq 50$). Given the testing repetitions ($du(s_i)_j$ or $ds(s_i)_j$, $1 \leq i \leq 31 \wedge 31 \leq j \leq 50$), the classifiers calculate the scores showing how likely these repetitions were done by the user s_x. Finally, attackers can set a *decision threshold* to distinguish the real user and the imposters.
– User classification: In this case, there are 31 classes, each of which represents one subject. The attacker trains the classifiers with the training repetitions which have been correctly labeled their classes. Given the testing repetitions ($du(s_i)_j$ or $ds(s_i)_j$, $1 \leq i \leq 31 \wedge 31 \leq j \leq 50$), the classifiers distinguish them into the 31 classes using the default decision threshold. Finally, we create a 31 by 31 dimensional confusion matrix in which the element in row s_i, column s_j is a count of the number of times the subject with true ID s_i was classified into ID s_j.



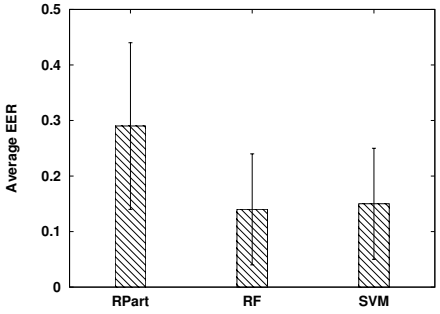
(a) The ROC curve for verifying s_{10} (unique input repetitions)



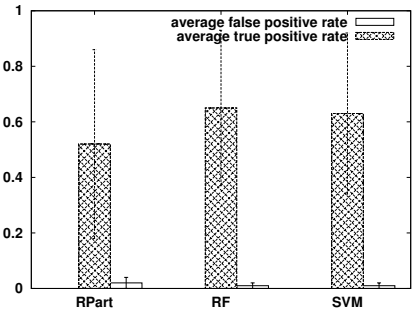
(b) The ROC curve for verifying s_{10} (shared input repetitions)



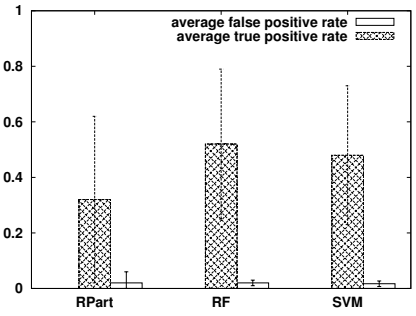
(c) Average EER with standard deviation for verifying $s_i, 1 \leq i \leq 31$ (unique input repetitions)



(d) Average EER with standard deviation for verifying $s_i, 1 \leq i \leq 31$ (shared input repetitions)



(e) Average TPR and FPR with standard deviation for classifying (unique input repetitions)



(f) Average TPR and FPR with standard deviation for classifying (shared input repetitions)

Fig. 4. Statistical results of our experiments on verification and classification using the 3 algorithms

Like most previous work on classification problems, we evaluate the performance of the implementation by the classification errors, in more detail, false positive, which mistakenly takes the imposter's data as the real user's; and false negative, which mistakenly classifies the real user's data into the imposter's class. For evaluating the user verification problem, we also concern the Equal Error Rate (EER). In a binary classification problem, false positive rate and false negative rate usually varies depending on the decision threshold. EER, as the crossover point at which the false positive rate equals the false negative rate, is an important value to judge the classifier. The lower the EER, the better performance for the classifier. We run the implementation of our classifiers several times for the experiments. Figure 4 shows the result.

Figure 4(a) and 4(b) illustrate the Receiver Operating Characteristic (ROC) curves with subject s_{10} as the genuine user when we did user verification. The repetitions are $du(s_{10})_j$ and $ds(s_{10})_j$ respectively. The True Positive Rate (TPR) is the frequency with the repetitions of subject s_{10} has been correctly detected. The False Positive Rate (FPR) is the frequency with which the imposters are mistakenly detected as the genuine users. Both of them varies depending on the decision threshold. We observe that the RF gives the best result and the RPart gives the worst result for both unique and shared input repetitions. The EER in unique input repetitions is from 0.08 to 0.18 and that in shared input repetitions is from 0.3 to 0.4. Figure 4(c) and 4(d) show that the average EER with standard deviation for all the subjects. The RF gives the average EERs around 0.1 and 0.14 for unique and shared input repetitions. The EERs given by SVM are 0.12 and 1.5 respectively. RPart gives the highest EER, around 0.25 and 0.29. As said, the lower the EER, the better performance for the classifier. Therefore, RF gives the best result. Figure 4(e) and 4(f) show the result of the classification problem. The best performance is still given by RF, with the lowest average TPR around 0.65 and FPR around 0.01 for unique input repetitions. The TPR for unique and shared input repetitions are 0.52 with 0.02 FPR. SVM has the similar results to RF. The worst case is still given by the RPart: its average TPR is around 0.52 for unique input repetitions and only 0.32 for shared input repetitions. The results show that VoIP user recognition by key-click patterns is possible. RF algorithm gives better performance than the other two. It is easier to recognize users if they have different PIN codes to input.

4.5 Discussion on countermeasure

One countermeasure is to insert random delay between key-click events. When a user-agent receives a key-click event, it does not immediately process the event. Instead, it puts all information of the event (e.g., the holding time) in a First In First Out (FIFO) queue. The user-agent constantly checks the queue and fetches a event from it after a random time delay for processing. It obscures real key-click patterns. Yet another defending method is to encrypt the whole RTP packet using IPSec. We know that the attacks take advantage of the factor that SRTP does not encrypt RTP headers, which enable attackers to restore key-click patterns. The attacks do not work if the RTP headers are encrypted.

RFC 3711 [10] suggests IPSec (ESP method) [19] if users would like both the RTP headers and contents to be protected. Nevertheless, users may particularly worry about the performance overhead and configuration complexity by using IPSec. Although it is possible to effectively reduce the performance overhead by using packet header compression [9], configuration complexity might be a barrier to widely deploy IPSec in VoIP.

5 Related work

keystroke dynamics is a method to recognize individual users by using their typing characteristics, with the time stamps of key-down and key-up. Many previous work has been done in this domain and a broad overview can be found in [23]. This section only summarizes the work most relevant in the context of ours. Maxion et al. [22] asked 28 volunteers to type 200 repetitions of the same 10-digital code using only the index finger on the number pad of a standard keyboard. They intercepted the keystroke information on key-down and key-up time locally and analyze them using random forest classifier. Half of the data were selected for training and the rest were used for testing. The classifier achieved a hitting rate of 99.54% and false alarm rate of 12.50%. Kotani, et al., [21] performed experiments using a special pressure sensitive keypad with 9 subjects. Each subject typed 20 repetitions of the same 4-digital PIN code. Besides key-down and key-up times, stroking force was chosen as a third element. Their classifier gives a equal error rate at 2.4%. Clarke et al., [17,15,16] performed several tests in which they asked different number of subjects (from 16 to 32) to type the same 4-digital PIN code for 30 repetitions on the keypad of a mobile phone. 20 repetitions are used for training and the rest is used for testing. Their neural network classifier gives an equal error rate from 5.5% to 8.5%. Our work is on a different scenario. We recover key-click pattern from VoIP flows and the subjects use a mouse and virtual keypad for input. Moreover, our method needs to take the networking condition impact into account.

On VoIP user recognition, Khan et al. [20] proposed a scheme exploiting patterns on the sizes distribution of RTP voice packets. Their classifier achieved a hitting rate of 75% for 10 speakers and 51% for 20 speakers. Similarly, Backes et al. [8] proposed an approach using the periods of speech and silence of a speaker in a conversation. This speaker specific pattern is modeled using the speaker's talking speed and frequency. The identification rates obtained were 65% for 13 speakers and 48% for 20 speakers. Nevertheless, their method requires specific codec being applied. Our work achieves the same goal but in another way: We exploit the side channels in RTP event flows rather than RTP voice flows.

6 Conclusion

This paper proposed an attacking method to recognize VoIP users for user-level calling records profiling. It takes advantage of user-specific key-click patterns. Even if a VoIP flow are protected by SRTP, attackers can still recover key-click

patterns from the flow by reading packet header fields. The impact introduced by varying network conditions (jitter, packet loss) can be minimized. In an empirical setup with 31 users our analysis is able to correctly classify unknown RTP flows in about 65%. For user verification, the average equal error rate is from 10% to 29%. The result raises serious concerns about anonymity for VoIP users. To prevent this attack, users can consider to either randomize the time interval between key-clicks or use another security scheme (e.g., IPSec) which not only encrypts the whole part of a RTP packet, but also pads all RTP packets to an equal size.

There are still some limitations on our currant experiments. First, we only asked subjects to enter PINs using the virtual keypad by mouse clicking so far. Nevertheless, some user-agents also support standard keyboard input. Second, we let all subjects to enter the same PIN "9913" for collecting shared input repetitions. The results with only one PIN instance may difficult to be generalized. In future work, we will investigate this problem further not only using virtual keypad, but also standard keyboard. In addition, we will try several shared PIN candidates for collecting shared input repetitions.

References

1. 40 websites offering telephone calling records and other confidential information. http://epic.org/privacy/iei/attachment_a.pdf, visited at 15th-Nov-2010.
2. e1071: Misc Functions of the Department of Statistics (e1071), TU Wien. http://cran.r-project.org/web/packages/e1071/index.html, visited at 18th-Sep-2010.
3. randomForest: Breiman and Cutler's random forests for classification and regression. http://cran.r-project.org/web/packages/randomForest/, visited at 18th-Sep-2010.
4. rpart: Recursive Partitioning. http://cran.r-project.org/web/packages/rpart/, visited at 18th-Sep-2010.
5. TCPDump. http://www.tcpdump.org/, visited at 20th-July-2010.
6. The R project for statistical computing. http://www.r-project.org/, visited at 18th-July-2010.
7. X-Lite. http://www.counterpath.com/x-lite.html, visited at 18th-July-2010.
8. Michael Backes, Goran Doychev, Markus Dürmuth, and Boris Köpf. Speaker Recognition in Encrypted Voice Streams. In *Proceedings of ESORICS '10*, LNCS. Springer-Verlag, 2010.
9. R. Barbieri, D. Bruschi, and E. Rosti. Voice over ipsec: Analysis and solutions. In *Proceedings of ACSAC '02*. IEEE, 2002.
10. M. Baugher, D. McGrew, M. Naslund, E. Carrara, and K. Norrman. The Secure Real-time Transport Protocol (SRTP), 2004. RFC 3711.
11. K.P. Bennett and C. Campbell. Support vector machines: hype or hallelujah? *SIGKDD Explor. Newsl.*, 2(2):1–13, 2000.
12. L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.
13. L. Breiman, C. J. Stone J. Friedman, and R. A. Olshen. *Classification and Regression Trees*. Chapman & HallCRC, 1984.
14. R. Caruana and A. Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of ICML '06*. ACM, 2006.

15. N. Clarke and S. Furnell. Advanced user authentication for mobile devices. *Computer & Security*, 26:109–119, 2007.
16. N. Clarke and S. Furnell. Authenticating mobile phone users using keystroke analysis. *International Journal of Information Security*, 6:1–14, 2007.
17. N. Clarke, S. Furnell, B. Lines, and P. Reynolds. Using keystroke analysis as a mechanism for subscriber authentication on mobile handsets. In *Proceedings of SEC'03*. Kluwer, 2010.
18. K. Egevang and P. Francis. The IP Network Address Translator (NAT), 2006. RFC 1631.
19. S. Kent and K. Seo. Security Architecture for the Internet Protocol, 2005. RFC 4301.
20. L.A. Khan, M.S. Baig, and A. M. Youssef. Speaker Recognition from Encrypted VoIP Communications. *Digital Investigationg*, 2009.
21. K. Kotani and K. Horii. Evaluation on a keystroke authentication system by keying force incorporated with temporal characteristics of keystroke dynamics. *Behaviour & IT*, 24(4):289–302, 2005.
22. R. A. Maxion and K. S. Killourhy. Keystroke biometrics with number-pad input. In *Proceedings of DSN'10*. IEEE, 2010.
23. A. Peacock, X. Ke, and M. Wilkerson. Typing patterns: A key to user identification. *IEEE Security and Privacy*, 2(5):40–47, 2004.
24. H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A transport protocol for real-time applications, 2003. RFC 3550.
25. H. Schulzrinne and T. Taylor. RTP Payload for DTMF Digits, Telephony Tones, and Telephony Signals, 2006. RFC 4733.
26. H. Sengar, Z. Ren, H. Wang, D. Wijesekera, and S. Jajodia. Tracking skype voip calls over the internet. In *Proceedings of INFOCOM '10*. IEEE, 2010.
27. X. Wang, S. Chen, and S. Jajodia. Tracking anonymous peer-to-peer VoIP calls on the Internet. In *Proceedings of CCS '05*. ACM, 2005.
28. C. Wu, K. Chen, Y. Chang, and C. Lei. Speaker Recognition in Encrypted Voice Streams. In *Proceedings of IPTCOMM '08*, LNCS. Springer-Verlag, 2008.