Continuous Authentication using Biometric Keystroke Dynamics

Patrick Bours and Hafez Barghouthi
Norwegian Information Security Laboratories (NISlab),
Gjøvik University College, Teknologivegen 22, 2815 Gjøvik
patrick.bours@hig.no
http://www.nislab.no

Abstract

When talking about authentication in general and biometrics in particular, we envisage a situation where a user provides an identity and gives proof of this identity, in order to get access to certain services. This kind of authentication (called static authentication) is done when first accessing a service and will be valid throughout a full session, until the user logs off from that session. Most authentication methods are very well suited for this kind of (static) authentication, e.g. the well known username/password combination for access to computers or websites. In this article we will discuss a different kind of authentication, which will be applied after the start of a session, and will monitor if the current user is the same as the user who performed the initial static authentication. This new type of authentication is called dynamic authentication or continuous authentication. We will in particular look at sessions at a computer where we monitor the typing behavior of the user to detect anomalies. Normal techniques used in biometrics cannot be applied anymore. We will explain the differences between static and continuous keystroke dynamics and we will show, using the performance of an experiment, that continuous authentication using keystroke dynamics can be a valuable addition to computer security. In order to use continuous authentication we designed the concept of a *penalty-and-reward* function to measure the confidence that the user has not changed during a session.

1 Introduction

Person authentication is present in every day life. For getting access to a computer, a program on a computer or a website, we need to provide username and password. When getting money from an ATM machine we need a bankcard and a PIN code. At some airports fingerprints or iris scans are already used to authenticate people. Systems need to be able to rely on the fact that the person using the system is indeed the one he or she claims to be. All authentication methods (password, token,

This paper was presented at the NISK-2009 conference.

biometric features, and more) have their own advantages and disadvantages, and the environment determines which authentication method is best suited. However, there is one disadvantage that all static authentication methods share. They authenticate the user at the moment that the authentication mechanism is executed: any change of user after that will be unnoticeable to the system. Such authentication systems are referred to as static authentication. A completely different type of authentication is continuous authentication, which is used after an (authenticated) user has entered a system. The system will then continuously monitor if a change of user occurs [8, 18].

Not every authentication method can be used for continuous authentication. In particular, we are restricted to biometric methods, where again we will be restricted. Biometric features like fingerprints or iris scans are not suitable for continuous authentication on a computer. We can for example use continuous gait authentication on mobile phones [2]. On a computer system we can use for example a face recognition system [5], where a camera mounted on top of the screen monitors if the current user is still present. If the user disappears or another user appears, the computer can be locked. Another, probably more natural way for continuous authentication on a computer system, is monitoring the keyboard (and mouse) behaviour of the user. Keystroke (and mouse) dynamics have been proved viable for static authentication. In other words, we can authenticate people based upon their keyboard (and mouse) usage.

The basis for continuous keystroke dynamics can be found in static keystroke dynamics. In static keystroke dynamics we measure how a user types a fixed text, e.g. a username or a password. In general timing information is used as data, but other information, like key pressure can be used too. However collection of this kind of data requires special equipment. The timing information that is used in keystroke dynamics is called duration and latency. The duration of a key is the time elapsed between pressing down a key and releasing it. Latency is the elapsed time between releasing one key and pressing the next one. When typing N characters, we have N duration values and $N-1$ latency values. Most articles on static keystroke dynamics use these timing values [6, 7, 9, 10, 11, 12, 13, 15]. Analysis methods either based on neural networks on on statistical analysis. During enrollment a user must type a fixed text a number of times. The timing information collected is then averaged to construct the template. The template consist the mean of a duration or latency timing, often in combination with the variation of the different measurements. During the authentication phase the same text is typed again and the timing information is then compared to the information stored in the template to decide if it was a genuine or an impostor attempt. Different distance metrics can be used in this process, like Euclidean [12], Normalized Bayesian Classifier [17], or Time Classification [4].

The authors of [3] consider keystroke dynamics on free text. They consider statistical properties of combinations of two, three or four letters. Their approach is a first step toward real continuous authentication, but in [3] they merely look at specific character combinations from various users. Furnell et al. published a series of articles on continuous keystroke dynamics [1, 16, 14], where they also consider properties of two and three letter combinations, using both statistical analysis and a neural network approach. They took the approach that a genuine user should never be locked out of the system, which is an approach we adapted in this work. They however expressed the performance of the system in terms of False Match Rate

(FMR) and False Non-Match Rate (FNMR). In Section 4 we will use a different approach which is much more suitable for continuous authentication.

2 Techniques used for static biometric authentication

In static biometric authentication, each participant provides his biometric features during enrollment. These features are stored in a so-called template. Whenever a person tries to authenticate himself, he will provide the same biometric feature again and this new input is compared to the template. In case the template and the new input are "close together", then we say that the input matches the template, otherwise it does not match it. A function is used express the distance between the template and the input in a numerical value. This function is called the distance metric $d(.,.)$. Furthermore, a threshold value T is used to determine if the resulting distance represents a match or a non-match. If the distance is below the threshold, we say that the template and the input match, otherwise they do not match.

Two types of error can now occur. First a person may provide an input that is too far away from his own template, i.e. it results in a non-match. This is called a False Non-Match. On the other hand a person might provide an input that is so close to another person's template that these two match. This is called a False Match. The probability of occurrence of these errors is expressed in the False Non-Match Rate (FNMR) and the False Match Rate (FMR). Obviously, given a specific distance metric, these two error rates depend on the chosen threshold. In general we can say that if the threshold T grows, then the FMR will grow too, while the FNMR will decrease. If T decreases, then the FMR will decrease and the FNMR will increase. Depending on the application, it might be desirable to have a low FMR (e.g. for access to high security areas) or a low FNMR (e.g. to keep customers happy for access in an internet cafe). We can display the performance of the system for various threshold values in a Receiver Operating Curve (ROC) where we display the FMR on the horizontal axis and the FNMR on the vertical axis. For each value of the threshold, we will find one point in the plane, and connecting these points gives the curve. A general point to express the quality of the biometric system (i.e. the combination of the biometric feature, distance metric and more) is the so-called Equal Error Rate (EER) point. This is the point of the curve where FMR=FNMR, so it is the point where the ROC curve crosses the $x = y$ line.

3 Static keystroke dynamics authentication

In keystroke dynamics we measure how a user uses a keyboard. In principle a lot of information can be measure, for example the force with which particular keys are pressed during typing or the finger that is used to type a particular key. We do however need extra equipment to record this type of information. The easiest information to collect is related to timing, in particular when a key is pressed down and when it is released again. This information can easily be collected using a small piece of software that checks if a key is pressed down or released and in either of these two occasions collects information on what key was used and what the time was. In this way we can determine how long a key is pressed down, called the duration of that key, and how long it takes between the release of one key and the pressing down of the next key, called the latency. Obviously the duration depends on the

key that is used. It is even more obvious that the latency depends on both the first and the second key that are used. For example the latency between often used key combinations like 'NG', 'IS', or 'ND' is much shorter than the latency between "strange key combinations" like 'Q9' or '2)'. We can explain this by noting that in these rarely used key combinations, it takes time for the user to locate the second key resulting in a larger latency timing.

For many biometric features, like fingerprints or iris, a user provides the biometric features only once to create the template, i.e. to enroll into the system. In static keystroke dynamics however, there is a larger variation between the various times the same text is typed. For enrollment in a static keystroke dynamics system, the user is asked to type a fixed text a number of times (in general between 5 and 20 times) and then an "average" way of typing this text is calculated. There are many ways to calculate such an average and also different kind of information can be stored in the template, but this is outside the scope of this article. When trying to authenticate the user types the text once again while measuring duration and latency timings again. These new values are then again compared to the average values in the template and based on the distance, a user is authenticated or not. Again, many different distance metrics can be used here, not in the last place because of the different kind of information that can be stored in the template. If a template is compared to an input the user is authenticated or not, meaning that he will get access to the system or not. Whenever access is granted, this will be valid throughout the full session!

4 Problems and solutions for continuous keystroke dynamics

Part of the above described techniques can be also used in continuous authentication, or in particular in continuous keystroke dynamics authentication. In particular will the users, during the enrollment phase, provide input to create a template. During the authentication phase, which in this case is the full time that a user is logged on to a system, the input of the user will be compared to the template and based upon that comparison a decision will be made if it is still the same user or if the user has changed. In case of keystroke dynamics, the template for static is completely related to the fixed text, which is no longer the case for continuous keystroke dynamics. However, for various keys or key combinations, the template will contain average duration and latency timings. The way a template is created is different for continuous keystroke dynamics. Instead of typing a fixed text a number of times, the user needs to use the system as normal during a longer period of time. Over that period of time (can be one day or even more), information is collected on how the user types on the keyboard. The template will now in general contain timing information on those keys and key combinations that are used often by the user and are also typed in a more or less stable way. Obviously, if the timing information related to a specific key or key combination various too much, then such information cannot be used to recognize this user. During the full enrollment session, we measure the timing information for each and every typed key and key combination. Creating the template, this collected information is analyzed and for each key and key combination, the average duration and latency timing is calculated, together with information about the stability. The average and stability can be represented by the mean (μ) and standard deviation (σ) of the collected timings.

The decision to include a particular key or key combination in the template depends on the number of occurrences N during the enrollment phase (e.g. the 'E' will be typed numerous times, while the occurrence of the '@' will be much lower) and on the ratio between σ and μ. The number of occurrences should be above a predetermined threshold while the value of $\frac{\sigma}{\mu}$ should be below another predetermined threshold. Also the number of features in a template could be limited from below and above if desired.

The major difference between static and continuous keystroke dynamics is in the authentication phase. In static keystroke dynamics the complete typing of the fixed text in the input and template is compared, be this cannot be applied to continuous keystroke dynamics. In continuous keystroke dynamics, we use confidence levels. At each point in time, we must determine how confident we are that the user has not changed, based upon previous typing behavior. At any point in time this confidence can increase or decrease, but once the confidence becomes below a certain level, actions must be taken, e.g. the user needs to provide a password in order to prove that he has not changed.

One way to implement the confidence levels is to use a "penalty and reward function". In this case, when a session starts, a value C is initialized as 0. For each key (or key combination) that is typed by the user, the C value is adjusted, based upon the information in the template and the timing of the just typed key (or key combination). If the timing information is correct, meaning it has a small distance to the information stored in the template, then the user is rewarded by reducing the value of C. In case the timing information is not correct, meaning it has a large distance to the information stored in the template, then the user is punished by increasing the value of C. If the C value stays below a predetermined threshold, then we are confident that the user has not changed and no action will be undertaken. If however the C value becomes too high, then the system will need to take action to (re-)confirm the identity of the user.

Some extra restrictions need to be applied to the penalty and reward function. The C value should not become negative. If we would allow that, then an attacker could more easily take over a system, when the C value has become extremely low, which would give him more time to use the system before the change of user would be detected. Furthermore, we need to determine what to do with the value of C in case the typed key (or key combination) does not appear in the template. One solution might be to increase the C value slightly, in order to prevent attackers from using many keys that do not appear in the template. The reward and penalty values should also be determined. These can be fixed (e.g. decrease C by 1 in case of a reward and increase it by 1 in case of a penalty) or they could depend on how good or how bad the timing information matches the values in the template, or a combination of both. One example could be to determine the distance d between the information in the template and the timing of the input. If the distance is below the threshold T, the the reward will be always equal to 1, while the penalty will be equal to $d - T$ in case the distance is above the threshold. In that case the penalty and reward function could be defined as follows:

$$C := \begin{cases} 0 & \text{at startup;} \\ max(C-1,0) & \text{if } d \leq T; \\ C + d - T & \text{if } d > T. \end{cases} \quad (1)$$

The *max* function makes sure that C will not become negative. Besides this penalty and reward function, a decision function is needed to determine when the system needs to action to assure the identity of the user. This can simply be a check if the value of C is above a threshold T_{action}.

In case of static authentication we can express the performance of the biometric systems in terms of FMR, FNMR, EER and ROC curve. These terms are harder to express in continuous authentication. Here we can however express the performance of the system in terms of how long it takes before an impostor is detected. Specifically in case of continuous keystroke dynamics, we measure the system performance by the number of keystrokes that an impostor can do before his value C goes over the threshold T_{action}. Obviously, the better a system performs, the lower this number of keystrokes will be.

5 Experimental results

Experiment setup

In this section we will discuss the results of an experiment we conducted. In this experiment 25 people participated, providing typing information over a period of at least 6 days. The users who participated in this experiment did not get any restrictions on the way to use their computers. A small program was installed that registered all typed keys and related timing information. The program was run over various days. Users were asked to run the program for at least 6 days, but in some cases keystroke information for up to 15 days was provided. As there were no restrictions for the users, we measured the normal behavior of the user's typing rhythm. For each keystroke the program recorded two lines in a log-file. The first of these lines was recorded when the user pressed a particular key and the recorded line started with the word "KeyDown", followed by the name of the key that was pressed and the time when it was pressed. The other line, which was recorded at the moment the pressed down key was released again, contained the word "KeyUp", followed again by the name of the key and the time it was released. In both cases the time was given in milliseconds after the start of the small program. Furthermore, both of the lines contained at the end a reference to the type of application in which the key was typed. This information will not be used in the analysis in this paper, but will be used in future research to see if the typing rhythm depends on the application. An example of this data is given below, where the particular user has typed the text "`he hasn't `":

```
KeyDown | H | 187.077976041648 | Conversation
KeyUp | H | 187.151446552565 | Conversation
KeyDown | E | 187.209934712373 | Conversation
KeyUp | E | 187.301092177916 | Conversation
KeyDown | Space | 187.334812804421 | Conversation
KeyUp | Space | 187.408780096353 | Conversation
KeyDown | H | 187.554834203788 | Conversation
KeyDown | A | 187.600848920743 | Conversation
KeyUp | H | 187.610930858531 | Conversation
KeyDown | S | 187.700653295321 | Conversation
KeyUp | A | 187.77763479716 | Conversation
KeyUp | S | 187.87070479628 | Conversation
```

```
KeyDown | N | 187.93017674034 | Conversation
KeyUp | N | 188.001366933507 | Conversation
KeyDown | Oem7 | 188.465398674971 | Conversation
KeyUp | Oem7 | 188.546950939295 | Conversation
KeyDown | T | 188.654562032325 | Conversation
KeyUp | T | 188.753510292509 | Conversation
KeyDown | Space | 188.833326512168 | Conversation
KeyUp | Space | 188.890464570218 | Conversation
```

Analysis method

We can easily calculate the duration of a key from the two lines described above, by subtracting the time related to the "KeyDown" line from the time related to the "KeyUp" line. The latency between two consecutive keys is derived as follows. First notice that we know the order in which the keys are typed, by looking only at the "KeyDown" lines. If we then have identified two consecutive keys, then the latency between those keys equals the difference between time related to the "KeyDown" time of the second key and the time related to the "KeyUp" line of the second key. So the latency more or less measures the time that the keyboard is not used, although it can be negative. A negative latency means that the second key is pressed before the first key is released. Although this can happen for any two keys, it mostly occurs in combinations of keys that are used often, and it always occurs when using special keys, like the Shift key or the CTRL key.

The original log files of the users are preprocessed first such that the lines in the new files contain 4 items:

1. The name of the current key;
2. The duration of the current key;
3. The latency between the current key and the next key;
4. The name of the next key.

Note that in these transformed files, the name of the next key in one line equals the name of the current key in the line right after it. Only the last line of these files do not contain a next key, and thus no latency timing. The results look as follows:

```
H|0,0734705109170193|0,0584881598080074|E
E|0,0911574655429774|0,0337206265050156|Space
Space|0,0739672919320071|0,146054107434992|H
H|0,0560966547429871|-0,0100819377879873|A
A|0,176785876417|-0,0769815018390148|S
S|0,170051500959005|0,0594719440599931|N
N|0,0711901931670127|0,464031741463998|Oem7
Oem7|0,0815522643239888|0,107611093030016|T
T|0,0989482601839882|0,0798162196589942|Space
```

Notice in the example above the negative latencies between the H and the A and the A and the S, which can be traced back to the fact that the A was pressed before

the H was released, and the S was pressed before the A was released. Notice also the relatively long latency between the N and the ' character (represented here by "Oem7", probably due to the fact that this last key is not used very often, so the user has to "search" for it.

Using these transformed files, it is easy to create the template of each user. These templates consists of two parts. The first part is related to single keys and the second part is related to key combinations. The data used to create the template is from the first day of typing of the user. For the single key k, all the lines with that particular key as the current key are regarded. From the duration timings in the corresponding lines we calculated the mean μ_k and standard deviation σ_k for that particular key, and we also recorded the number of occurrences N_k. If the number of occurrences was above 50 and the ratio between the standard deviation and the mean was below 0.35, then the particular key was added to the template, together with the mean and the standard deviation. For a double key combination k_1k_2 only those lines are considered with k_1 as current key and k_2 as next key. For the mean and standard deviation of the duration of k_1 the values in the second column are used and for the mean and standard deviation of the latency the values of the third column are used and finally for the mean and standard deviation of the duration of k_2 the values in the second column of the "next lines" are used.

For example if the following is part of the transformed file:

```
A 0.1723 0.2231 B
B 0.1811 0.3109 C
```

then, for the calculation on the statistic for the key combination $k_1k_2 = AB$, the value 0.1723 is used in the calculation of the duration of A, the value 0.2213 is used in the calculation of the latency between A and B and the value 0.1811 is used in the calculation of the duration of B. The conditions for adding a key combination to the template are slightly different from the conditions for a single key. As before the number of occurrences $N_{k_1k_2}$ should be at least 50, but now the ratio between the standard deviation (σ_{k_1} and σ_{k_2}) and the mean (μ_{k_1} and μ_{k_2}) for both durations should be below 0.5, and the ratio between the standard deviation ($\sigma_{k_1k_2}$) and the mean ($\mu_{k_1k_2}$) should be at most 0.6. The reason that these limits are higher is because otherwise the number of key combinations in the template would be too low. Furthermore, the limit for the latency is slightly higher than the limit for the durations is because of the higher variance that naturally occurs in latency timings as opposed to duration timings. We include the key combination k_1k_2 in the template if the number of occurrences is at least 50 and at least 2 out of the three ratios are below the threshold. So in this case we do not require all three ratios to satisfy the threshold. The reason behind this is again that we do not want to exclude too many key combinations.

Once the templates have been created for all user we can start the remainder of the analysis, to find the performance of this continuous keystroke dynamics system. The data that has been used for the creation of the user templates is excluded from this part of the analysis. We first want to define a penalty and reward function, similar to Equation (1). In order to do so, we need a distance metric, which in our case will be:

$$d = d((\mu, \sigma), t) = \left| \frac{t - \mu}{\sigma} \right| \tag{2}$$

In this equation t represents the duration timing related to a specific key k or the latency between two keys, and μ and σ are the corresponding mean and standard deviation values in the template. This distance metric measures the distance between the measured timing t and the expected value μ in units of σ. This function well defines the distance in case of a single key. For key combinations k_1k_2 we define the distance as the average between the distances of the two durations and the latency, i.e. we have:

$$d = \frac{1}{3} \cdot \left(\left| \frac{t_{k_1} - \mu_{k_1}}{\sigma_{k_1}} \right| + \left| \frac{t_{k_1k_2} - \mu_{k_1k_2}}{\sigma_{k_1k_2}} \right| + \left| \frac{t_{k_2} - \mu_{k_2}}{\sigma_{k_2}} \right| \right) \tag{3}$$

Once the distance metric has been defined, we can also define the penalty and reward function:

$$C := \begin{cases} 0 & \text{at startup;} \\ max(C - R, 0) & \text{if } d \leq T; \\ C + d - T & \text{if } d > T. \\ C + \alpha & \text{if not in template} \end{cases} \tag{4}$$

Initial testing showed that good results were obtained when using $T = 0.5$, $R = 1.3$, and $\alpha = 0.01$.

The next step is to determine the threshold T_{action}. This will be a user specific value and in our experiment we define this value based upon the template and the analysis data of the user. The procedure is to compare the analysis data of a user to his own template and then keep track of the highest C value resulting from the penalty and reward function in Equation (4). This value will be used in this experiment as the T^i_{action} threshold for user i. In this way we assured that a genuine user would not be locked out of the system during a period of "not so consistent" typing. The reason to use user specific threshold values is that we noticed that these values are rather different for different users and in this way we optimize the performance of the system. The system performance could furthermore be optimized by also using user specific distance functions and penalty and reward functions (in particular the values of T, R and α in Equation (4)), but that is left for future research.

Now we need to compare the analysis data of user i to the template of user j for all pairs (i, j), where $i \neq j$. We will measure how many keystrokes (on average) it takes before the C value from the penalty and reward function becomes higher than T^j_{action}.

User	Lockout Time	User	Lockout Time	User	Lockout Time
1	197	10	157	18	183
2	94	11	205	19	150
3	174	12	348	20	97
4	258	13	168	21	221
5	79	14	164	22	260
6	146	15	200	23	213
7	261	16	93	24	171
8	169	17	211	25	129
9	186				

We see from the above table that the average number of keystrokes needed to lockout an intruder varies between 79 and 348. This shows that an intruder will be locked out fairly quickly.

6 Conclusion and future research

We have described a way to use a biometric feature not just for static authentication, but for continuous authentication. In order to do so, we had to adapt the use of the distance function and we came up with a penalty and reward function. This penalty and reward function will keep track of the behaviour of the user over time and will decide on locking out a user or not.

In our experiment we used 25 participants doing their normal daily business on their own computers, without any restrictions. In this way we measured their normal typing behaviour. Using that data, we optimized some parameters and at the end found that the system would never lock out a genuine user, while intruders were locked out fairly quickly. The argument behind not locking out a genuine user is that the system should not be a burden on the genuine user, so it should in principle be invisible to him. However, in order to make the system more secure, the parameters could be changed in such a way that intruders are locked out earlier, at the cost of occasionally also locking out the genuine user. In settings where security is of the utmost importance, this will probably not be a problem.

The results in the previous section are promising, in the sense that an intruder will be, on average, locked out fairly quickly. Obviously this average should be as low as possible to make the system more secure. One way of improving the security would be to look at the typing behaviour for different applications. It is not hard to see that informal chatting on media like MSN, Yahoo or Skype will have a completely different typing behaviour compared to programming in C++, entering data in MS Excel, preparing slides for presentations or using a web browser. In our experiment we already collected information on the used application, so a next step in the analysis will be to factor the application into the penalty and reward function.

One more thing that needs to be mentioned here is that, in our experiment, there were absolutely no restrictions on the behaviour of the participants. This also included their freedom of choice of the hardware to use, so some users could have been using a normal keyboard, while others where using a laptop, where the typing behaviour will probably be very different. Also the layout of the keyboard (US, NO, or any other) might have been different for different participants in the experiment. This might imply that if an intruder will try to use the computer of a victim, his typing behaviour might change from his own normal typing behaviour because he has to adapt to the hardware and layout of the keyboard of the victim. It is not clear beforehand if such a change of typing behaviour would work in favor of the intruder or against him. This again needs to be investigated in future research.

Finally in the future, we might improve the performance of the continuous authentication system, by combining keystroke dynamics with mouse usage. This can be done at two levels. First we can measure the particular statistics of the typing rhythm and the mouse usage and use this kind of information for authentication. This is more or less the fusion of a continuous keystroke dynamics and a continuous mouse dynamics system. Second, we can also measure the interaction between keyboard and mouse, meaning that we model the user behaviour at the computer and use that for authentication.

References

[1] Furnell SM Dowland PS, Singh H. A preliminary investigation of user authentication using continuous keystroke analysis. In *Proceedings of the IFIP 8th Annual Working Conference on Information Security Management & Small Systems Security*, September 2001.

[2] Davrondzhon Gafurov. *Performance and Security Analysis of Gait-based User Authentication*. PhD thesis, University of Oslo, 2008.

[3] Daniele Gunetti and Claudia Picardi. Keystroke analysis of free text. *ACM Trans. Inf. Syst. Secur.*, 8(3):312–347, 2005.

[4] S. Hocquet, J.-Y. Ramel, and H. Cardot. Fusion of methods for keystroke dynamic authentication. pages 224–229, Oct. 2005.

[5] R. Janakiraman, S. Kumar, Sheng Zhang, and T. Sim. Using continuous face verification to improve desktop security. In *Application of COmputer VIsion, 2005. WACV/MOTIONS'05 Volume 1. Seventh IEEE Workshops on*, volume 1, pages 501–507, Jan 2005.

[6] Jarmo Ilonen,Keystroke dynamics,Lappeenranta University of Technology, Skinnarilankatu 34, 53850 Lappeenranta, Finland . 'http://www.it.lut.fi/kurssit/03-04/010970000/seminars/ilonen.pdf.',last visited 30.05.2009.

[7] J. A. Michael Chambers John A. Robinson, Vicky M. Liang and Christine L. MacKenzie. Computer user verification using login string keystroke dynamics. *IEEE Transactions on systems, Man, and cybernetics*, 28(2):236–241, 1998.

[8] Jie Liu, F.R. Yu, Chung-Lung, and H. Tang. Optimal combined intrusion detection and biometric based continuous authentication in high security mobile ad hoc networks. *Wireless Communications, IEEE Transactions on*, 8(2):806–815, Feb. 2009.

[9] Miguel G.Lizarraga Lee L.Ling Livia C.F.Araujo, Luiz H.R.Sucupira Jr. and Joao B.T. Yabu-Uti. User authentication through typing biometrics features. *IEEE Transactions on Signal Processing*, 53(2):851–855, 2005.

[10] M. S. Obaidat and B. Sadoun. Keystroke dynamics based authentication ,'http://www.cse.msu.edu/ cse891/sect601/textbook/10.pdf', last visited 30.05.2009.

[11] Fabian Monrose, Michael K. Reiter, and Susanne Wetzel. Password hardening based on keystroke dynamics. In *CCS '99: Proceedings of the 6th ACM conference on Computer and communications security*, pages 73–82, New York, NY, USA, 1999. ACM.

[12] Fabian Monrose and Aviel Rubin. Authentication via keystroke dynamics. In *CCS '97: Proceedings of the 4th ACM conference on Computer and communications security*, pages 48–56, New York, NY, USA, 1997. ACM Press.

[13] Fabian Monrose and Aviel D. Rubin. Keystroke dynamics as a biometric for authentication. *Future Gener. Comput. Syst.*, 16(4):351–359, 2000.

[14] Steven M. Furnell Paul S. Dowland. A long-term trial of keystroke profiling using digraph, trigraph and keyword latencies. In *Security and Protection in Information Processing Systems, IFIP 18th WorldComputer Congress, TC11 19th International Information Security Conference*, pages 275–290, August 2004.

[15] A. Peacock, Xian Ke, and M. Wilkerson. Typing patterns: a key to user identification. *IEEE Security and Privacy Magazine*, 2(5):40–47, September-October 2004.

[16] M. Papadaki P.S. Dowland, S.M. Furnell. Keystroke analysis as a method of advanced user authentication and response. In *Security in the Information Society: Visions and Perspectives, IFIP TC11 17th International Conference on Information Security (SEC2002)*, pages 215–226, May 2002.

[17] Charles Slivinsky Saleh Bleha and Bassam Hussien. Computer-access security systems using keystroke dynamics. *IEEE Transactions on Pattern Analysis and Machine Intelligenc*, 12(12):1217–1222, 1990.

[18] R.H.C. Yap, T. Sim, G.X.Y. Kwang, and R. Ramnath. Physical access protection using continuous authentication. In *Technologies for Homeland Security, 2008 IEEE Conference on*, pages 510–512, May 2008.