Audio CAPTCHA for SIP-Based VoIP

Yannis Soupionis, George Tountas, and Dimitris Gritzalis

Information Security and Critical Infrastructure Protection Research Group
Dept. of Informatics, Athens University of Economics & Business, Greece
jsoup@aueb.gr, gtountas@aueb.gr, dgrit@aueb.gr

Abstract. Voice over IP (VoIP) introduces new ways of communication, while utilizing existing data networks to provide inexpensive voice communications worldwide as a promising alternative to the traditional PSTN telephony. SPam over Internet Telephony (SPIT) is one potential source of future annoyance in VoIP. A common way to launch a SPIT attack is the use of an automated procedure (bot), which generates calls and produces audio advertisements. In this paper, our goal is to design appropriate CAPTCHA to fight such bots. We focus on and develop audio CAPTCHA, as the audio format is more suitable for VoIP environments and we implement it in a SIP-based VoIP environment. Furthermore, we suggest and evaluate the specific attributes that audio CAPTCHA should incorporate in order to be effective, and test it against an open source bot implementation.

1 Introduction

A serious obstacle when trying to prevent Spam over Internet Telephony (SPIT) is identifying VoIP communications which originate from software robots (bots). Alan Turing's Turing Test" paper [1] discusses the special case of a human tester who attempts to distinguish humans from artificial intelligence (AI) computer programs. The research interest in this subject has spurred a number of proposals for CAPTCHA (Completely Automated Public Tests to tell Computers and Humans Apart) [2,3,4,5]. Commercial examples include major stakeholders in the field, such as Google and MSN, which require CAPTCHA (visual or audio), in order to provide services to users. However, more advanced computer programs can break a number of CAPTCHA that have been proposed to date.

In this paper, we develop an audio CAPTCHA suitable for use in VoIP infrastructures, which are based on the Session Initiation Protocol (SIP). We first illustrate some of the SIP-based SPIT characteristics and present background work. We then explain how a CAPTCHA can be utilized in a VoIP infrastructure. In section 3, we propose a classification of the characteristics/attributes of audio CAPTCHA. In section 4 we briefly introduce a bot that is currently publicly available and will be used for testing purposes. We also present an example of how this bot solves CAPTCHA. In section 5 we implement a new audio CAPTCHA, which is based on the attributes shown in section 3. Finally, we present the results of the tests performed to order to evaluate its performance.

D. Gritzalis and J. Lopez (Eds.): SEC 2009, IFIP AICT 297, pp. 25–38, 2009.

2 Background

SPIT constitutes a new, emerging type of threat in VoIP environments. It illustrates several similarities to email spam. Both spammers and spitters use the Internet to target a group of users to initiate bulk and unsolicited messages and calls, respectively. Compared to traditional telephony, IP telephony provides a more effective channel, since messages are sent in bulk, and at a low cost. Marketers can use spam-bots to harvest VoIP addresses. Furthermore, since call-route tracing over IP is more difficult, the potential for fraud is considerably greater.

A method that is widely used to uphold automated SPAM attacks is CAPTCHA. The same technique can be used in order to mitigate SPIT. Each time a callee receives a call from an unknown caller, an automated reverse Turing test would be triggered, which the spit-bot needs to solve in order to complete its attack. Integrating such a technique into a VoIP infrastructure raises two issues. Firstly, the CAPTCHA module should be combined with other anti-SPIT controls, i.e. not every call should pass through the CAPTCHA challenge, since each audio CAPTCHA requires considerable computational resources. A simultaneous triggering of numerous CAPTCHA challenges may eventually lead to denial of service. Challenges would also cause annoyance to users, if they had to solve one for every single call they make. Moreover, the CAPTCHA needs to be designed in a friendly way to humans and also remain solvable by them.

2.1 CAPTCHA

A CAPTCHA challenge is a test that most humans should be able to pass, but current computer programs should not. Such a test is often based on hard, open AI problems, e.g. automatic recognition of distorted text, or of human speech against a noisy background. Differing from the original Turing test, CAPTCHA challenges are automatically generated and graded by a computer. Since only humans are able to return a sensible response, an automated Turing test embedded in the above protocol can verify whether there is a human or a bot behind the challenged computer. Although the original Turing test was designed as a measure of progress for AI, CAPTCHA is a human-nature-authentication mechanism. In this paper, we focus on audio CAPTCHA. These were initially created to enable people that are visually impaired to register or make use of a service that requires solving of a CAPTCHA. Nowadays, an audio CAPTCHA would also be useful to defend against automated audio VoIP messages, as visual CAPTCHA are hard to apply in VoIP environments due to the limitations of end-user devices (e.g. IP phones). If an adequate CAPTCHA is used, it should be hard for a spit-bot to respond correctly and, thus, manage to initiate a call. Audio CAPTCHA also seem attractive, as text-based CAPTCHA have been proven to be breakable [7,8,9,10,11]. We validate our results with user tests and with a bot that was configured in order to solve difficult audio CAPTCHA. The proposed CAPTCHA must be: (a). Easy for humans to solve, (b). Easy for a tester machine to generate and grade, (c). Hard for a software bot to solve.

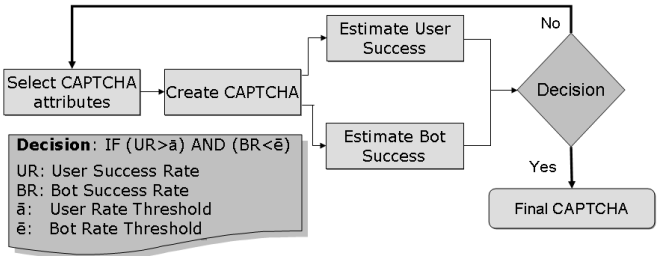


Fig. 1. A generic CAPTCHA development process

The first requirement implies that user studies are necessary in order to evaluate the effectiveness of CAPTCHA. The latter ones imply that a test with a new property is required: the test must be easy to generate, but intractable to pass without special knowledge available to humans and not computers. Audio recognition seems to fit in the category. Humans can easily identify words in a noisy environment, but this is not true for computers. Specification-wise, CAPTCHA do not need to be 100% effective at identifying software bots. A design goal for a CAPTCHA could be to prevent bots from having a success rate greater than 0.01% [6]. Since CAPTCHA use increases the cost of a software robot, the CAPTCHA can still be effective, as long as this increased cost remains higher than the cost of using a human. In order to develop a new audio CAPTCHA, we followed an iterative approach: (a) we selected a set of attributes appropriate for audio CAPTCHA, (b) we developed a CAPTCHA that is based on these attributes, and (c) we evaluated the CAPTCHA by calculating the success rates of a bot and a number of users until the results were satisfying and the attributes did not require further adjustment (see Fig 1).

3 CAPTCHA Attributes

High success rates by users is a key factor in deciding whether the CAPTCHA is effective or not. This is particularly important in the case of audio CAPTCHA, as it does not only refer to VoIP callers, but also to visually impaired users of a VoIP service. Equally important is the success rate of a bot, which should be kept to a minimum. Both factors depend on several attributes, which we classified into four categories (Fig. 2): (a). *vocabulary*, (b). *background noise*, (c). *time*, and (d). *audio production*.

Vocabulary attributes: CAPTCHA can vary based on the vocabulary used, by the following attributes:

1. *Adequate data field*: A data field (called alphabet) is used as a pool for selecting the characters to be included in an audio CAPTCHA. For the development of our CAPTCHA we used an alphabet of ten one-digit numbers, i.e. 0,,9. Such a choice allows the use of the DTMF method for answering the audio CAPTCHA. Other examples of audio CAPTCHA that use only digits are the MSN and the Google ones. A limited alphabet may make an audio method quite vulnerable to attacks.

Therefore, in order to make the CAPTCHA solution harder for a bot, a means that we adopted, is to use a number of different human speakers for each digit of the alphabet.

2. *Spoken characters variation*: Another drawback is the use of a fixed number of characters. Having a non variable number of characters in combination with a limited alphabet can make a CAPTCHA particularly vulnerable to attacks. For example, if only 3-digit CAPTCA are used and a bot can successfully recognize 2 of the digits, then it would easily reach 10% success.
3. *Language requirements*: Another important factor is the mother tongue of the users, as it plays a major role in achieving a high success rate by human users. This is particularly important in the case of audio methods, where there is a greater difficulty in identifying spoken characters when the mother tongue differs from that of the user. As a result, the language used should meet the scope of the specific CAPTCA application. As a good practice, the spoken characters should be few. The CAPTCHA we proposed can be adjusted for non English users, as the CAPTCHA are created dynamically and different characters can be added easily.

Background noise attributes: The background noise is another important attribute of an audio CAPTCHA, as it increases the difficulty for an automated procedure to solve it [12]. The main noise attributes are the following:

1. *Noise patterns*: The noise, which can be added during the production of a voice message, can make CAPTCHA particularly resistant to attacks by automated bots. Application of background noise requires a great variety of such noises to be available. These noises should be rotated in an erratic manner. In our proposal, instead of developing a repository with noises, we chose to proceed with a dynamic production of noises, which are distorted in a random manner. The way various noises are produced should prevent the easy elimination of them by automated programs that use learning techniques. An example of an audio CAPTCHA which is vulnerable to attacks, because of the static nature of the added noise is the MSN audio CAPTCHA (violated by the J. van der Vorms bot, success rate of 75%) [13]. The Google audio CAPTCHA, with a noise production different than the MSN and with different announcers for each character, appears to be harder for the same bot (33%) [13]. In any case, the final version of the audio message, resulting from the combined use of different distortion techniques and added noise, should be such that the majority of users can recognize it as easily as possible.
2. *Sound distortion techniques*: Sound distortion techniques may prevent an automated program to isolate the spoken characters from a voice message correctly. One needs to select the scale in which a distortion method will be applied to the spoken characters, the background noise added, or both. The deformation can last for the entire duration of the voice message, or it can be applied only when characters are announced, or even appear at random time intervals. In our CAPTCHA the distortion is applied between the characters, as there appears to be no effective method for evaluating how people understand digits with distortion.

Time attributes: During the production of an audio snapshot, a set of variables should be defined. These variables refer to the length of the audio message, which depends on: (a). the number of characters spoken, (b). the characters chosen, and (c). the time required for each character to be announced, which depends on the announcer of each character. The beginning and the end of each character spoken should also be defined. This depends on the duration of each character, as well as to the duration of the pause between the spoken characters, which could vary for each CAPTCHA. If the above time parameters follow specific patterns, then the resistance of the audio CAPTCHA to an automated program will decrease. In our CAPTCHA, we tried to eliminate such time-related patterns.

Audio production attributes: An audio CAPTCHA production procedure should be automated. An acceptable human interference refers only to the adjustment of various thresholds.

1. *Automated production process*: The automation of the CAPTCHA production process is a desirable but hard to achieve property. The various elements that compose an audio CAPTCHA, such as the number of characters of a message, the different announcers of each character, the different background sound, the timing and the distortion of the message, make the process time-costly and demanding of hardware resources. Our choice is to produce audio CAPTCHA periodically in order: (a) not to produce them in real-time, and (b) not to produce identical snapshots for extended time periods.
2. *Audio CAPTCHA reappearance*: An audio CAPTCHA should reappear rarely. In any case, especially with short alphabets, every CAPTCHA is expected to reappear after a while. Due to the attributes of the voice messages (e.g., technical distortion, added noise, language, speakers, etc.), as well as to the context of the user (e.g., noisy environment, etc.), a voice message may not be identified by the user on the first attempt. Therefore, a second chance may be given. In this case, a different CAPTCHA should be used.

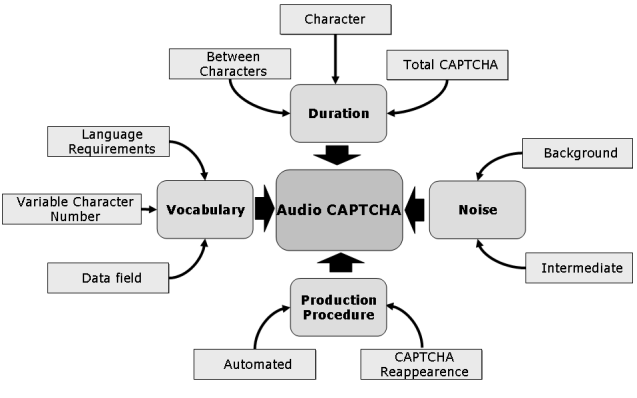


Fig. 2. Audio CAPTCHA attributes

4 CAPTCHA Bot

Frequency and energy pick detection bots
There are various methods/tools to recognize the words spoken in an audio file, such as the HTK toolkit [16] and the Sphinx [17]. These methods are demanding in hardware and time resources, because they use combinations of speech recognition methods. Moreover, they do not focus on how quick they reach a result but on how correct the result is. Therefore, we selected a bot category which employs frequency and energy peak detection methods and can be used to solve audio CAPTCHA for the following reasons:

- *Such bots have been proven effective*: Demonstrative (though perhaps not thorough enough) tests of such bots against popular audio CAPTCHA implementations have been successful [13,14,15] (e.g., SPIT prevention infrastructures, registrations for visually impaired people, etc.)
- *Such bots are easy to implement*: Frequency and energy peak detection bots are comparatively easy to implement using open-source software.
- *Such bots require limited time to solve a CAPTCHA*: Fast CAPTCHA solving is required because most services leave a small time frame for their users to solve the tests (5-15 sec), especially if VoIP services are considered. The CAPTCHA solving bot must analyze and reform the solution to the desired form (SIP message, DTMF, etc.), in a limited time frame.
- *Such bots occupy a small amount of system recourses*: An automated spam attack is selected when its cost is lower than employing humans. Also, a spitter performs multiple attacks simultaneously (e.g. the goal is to initiate SIP calls or messages in parallel). Thus, a bot must be inexpensive in terms of system recourses, which will allow the spammer/spitter to run several instances of the bot at the same time. Regarding time constraints, frequency and energy peak detection processes are less demanding than other approaches, which use different methods such as Hidden Markov Models (HMM) [16].

On the other hand, there are drawbacks when using these bots, mainly due to the fact that they require a training session. In this session a human identifies a number of selected CAPTCHA. The human recognizes the announced characters and records them in a file, from which the bot receives the data to solve the CAPTCHA. The set of training audio CAPTCHA might be extensive if the CAPTCHA data field (alphabet) is long. However, in the VoIP domain, the available alphabet is relatively small as it contains only digits (0-9), which increases the applicability of the mechanism.

The bot used
For the purpose of this paper we used the bot developed by J. van der Vorm [13]. This bot uses frequency analysis and energy peak detection, in order to segment and solve an audio CAPTCHA. The bot works as follows: it first reads the audio file. It skips as many starting bytes as the user has predefined (to avoid the starting bells that many services have, e.g. Google). Then, the samples are treated with a hamming window defined by the user. Each block is transformed into the frequency domain using Discrete Fourier

Transformation. Then, the frequencies are put in a predefined number of bins (the bins are not equally wide, the higher the frequency, the larger the band). After that, the bot looks at the highest frequency bin. Every block that has more energy in a window than the predefined threshold energy is considered a peak (see Fig. 3). These peaks are used to segment the audio file in the different spoken digits. Then the bot looks for a number of windows around the peaks and prints all the frequency bins. This is the profile of the digit. The profiles of the digits are then compared with the ones in the training file, and the closest match is chosen as a possible guess for each digit.

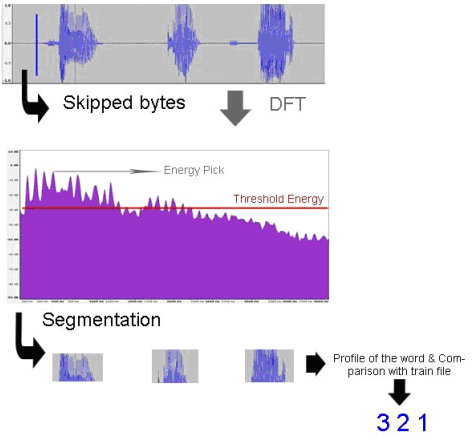


Fig. 3. Audio analysis of the bot

During the training session of the bot, the user gives as input to the bot an audio CAPTCHA. Then, for each profile of the digit that the bot chooses, the user enters which digit it actually was (this procedure can be automated if the user names the audio files accordingly, i.e., if an audio CAPTCHA file includes the digits 3, 2 and 1, the file name can be 321.wav). Obviously, the larger the number of audio CAPTCHA in the training set is, the higher the success ratio of the bot would be.

Bot applicability to SIP-based VoIP

In order to implement the bot in a SIP-based VoIP environment and examine its applicability, it was decided that the implementation procedure should consist of three stages. The procedure and the exchanged SIP messages between the participating entities are presented in Fig. 4.

Stage 0: It is dominated by the administrator of the callees domain (Domain2). When the callees domain receives a SIP INVITE message, there are three possible distinct outcomes: (a) forward the message to the caller, (b) reject the message, and (c) send a CAPTCHA to the caller (UA1).

Stage 1: An audio CAPTCHA is sent (in the form of a 182 message) to the caller (UA1). In the proposed implementation, the caller is replaced by a bot. It must record

the audio CAPTCHA, reform it to an appropriate audio format (wav, 8000Hz, 16 bit), and identify the announced digits. The procedure depends mainly on the time needed to reform the message. Moreover, the particular bot needs approximately 0.10 sec to identify a 3-digit CAPTCHA and 0.15 sec to identify a 4-digit one.

Stage 2: When the bot has generated an answer, it forms a SIP message by using SIPp [14], which includes the DTMF answer. This answer is sent as a reply of the CAPTCHA puzzle. If the caller does not receive a 200 OK message, a new CAPTCHA is sent and the bot starts to record again (Stage 1).

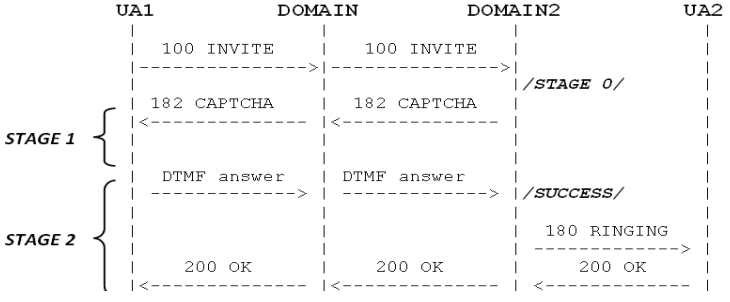


Fig. 4. SIP message exchange for CAPTCHA

The procedure above should be completed within a specific time frame. This time slot opens when the audio file is received by the caller and closes when the timeout of the users input expires (defined by the service CAPTCHA provider). The duration of the CAPTCHA playback does not affect the time frame because the waiting time for an answer starts when the playback is complete. If an answer arrives before the timeout, then it is validated by CAPTCHA service (and if it is correct the call is established), otherwise the bot has another try. In our proposal, the bot is given 6 sec to respond to the CAPTCHA, whereas the maximum number of attempts is set to three (3).

Table 1 illustrates the time required by the various stages in the proposed implementation. The selected bot is able to answer properly to CAPTCHA puzzle in much less time than the proposed time frame. Since a CAPTCHA is desired to be easy for humans, we suggest that the time frame, in which the caller should answer the CAPTCHA puzzle, should not be less than 3 sec. That is because many groups of users, such as physically impaired or elderly people, may not be able to respond promptly.

5 Audio CAPTCHA Implementation

Selected Attributes

In order to develop an effective audio CAPTCHA, we decided upon the following attributes:

Different announcers: The announcer of each and every digit is selected randomly.

Table 1. Stage duration

Stage	Step	Duration (sec)
1	Reform audio	∼1.00
	Identify digits	∼ 0.15
2	Create SIPp message	∼0.40
	Send SIPp message	∼0.00
Total (sec)		∼1.55

Random positioning of each digit in the CAPTCHA: The digits of the CAPTCHA are physically distributed randomly in the available space.

Background noise of each digit: Background noise, randomly selected, is added to each and every digit of the audio CAPTCHA. The audio noise files are segments (from 1 to 3 seconds) of randomly selected music files and not auto-generated by other methods (e.g. creation of white noise). We wanted to ensure that they will be less annoying for the user to listen to. The automatic generation of background and intermediate noises would require statistical analysis. Moreover, the volume level of the noise is lower than the level of the digits so that they remain audible to humans.

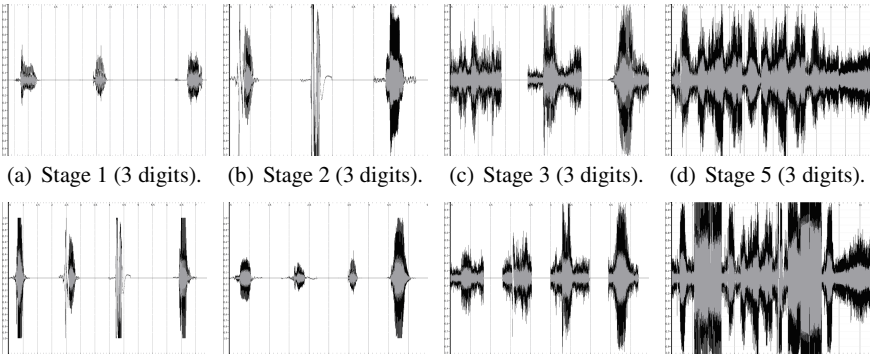
Loud noise between digits: Loud noise is introduced between the digits (the noise is not very loud, to minimize the discomfort of the user).

Different duration and file size: Each audio CAPTCHA file has different duration and different size.

Vocabulary:The vocabulary was limited to digits 0,,9, since the audio CAPTCHA was designed for a SIP-based VoIP infrastructure, where DTMF signals need to be sent.

CAPTCHA development

The audio CAPTCHA development was carried out in five incremental Stages (Stage 1 to Stage 5), in terms of the number of attributes adopted. Each development stage was tested and evaluated upon its efficiency according to the success rate of the bot and the success rate of human users. The audio CAPTCHA produced in Stage 1 were pronounced by one sole announcer without including additional features. Furthermore, the first digit of every word started at the exact same point as the other ones, and the time difference between each digit was equal. The waveforms of the resulting 3- and 4-digit CAPTCHA appear in Fig. 5(a) and 5(e). In such a simple audio CAPTCHA, a bot can use a detection method, such as energy peak detection, and easily segment and recognize the digits. An important factor in this process is the number of audio CAPTCHA that were used during the training of the bot. If a small number was used, then there is a high chance that not all digits are given as an input in the training process; thus, the bot may have a low success rate. That is the case with the 4-digit CAPTCHA (Fig. 5(e)). The random training sequence did not involve many instances of some digits (such us 8 and 9), therefore, the bot resulted in a relatively low (69%) success rate.



(a) Stage 1 (3 digits). (b) Stage 2 (3 digits). (c) Stage 3 (3 digits). (d) Stage 5 (3 digits).

(e) Stage 1 (4 digits). (f) Stage 2 (4 digits). (g) Stage 3 (4 digits). (h) Stage 5 (4 digits).

Fig. 5. Audio files' waveforms in implementation stages

During Stage 2, the audio CAPTCHA were produced by using 7 different announcers. Each digit was pronounced by a randomly selected announcer. This modification affected the success of the bot, but it mainly depended on the training set. When a larger training set was provided in the case of 4-digit audio CAPTCHA, the bot was able to maintain a relatively high success rate. However, when the same number of audio files, as in the 3-digit CAPTCHA process, was provided, the success rate decreased slightly. Moreover, we should mention that 4-digit CAPTCHA offer more digits to the training procedure. For example, if we use 100 3-digit CAPTCHA for training, we have 300 digits recorded, whereas with the same number of 4-digit CAPTCHA we get 400 digits. Figs. 5(b) and 5(f) show the waveforms of the produced digits.

In Stage 3 background noise was added against each digit. This way we managed to suppress the success rate of the bot, but it still remained relatively high (30% for the 3-digit CAPTCHA and 55% for the 4-digit ones). Figs. 5(c) and 5(g) show the waveforms of the produced digits with the background noise. The high success rate is due to the ability of the bot to cut off the low energy sounds (i.e., the noise) by checking above a certain threshold energy. The difference between the success of 3- and 4-digit CAPTCHA is due to the difference in the training sets. In this case we allowed a training of 50 audio CAPTCHA for the 3-digit ones and 150 for the 4-digit ones. As a result, the available digits taking part in the training process were 150 and 600, respectively.

In Stage 4 we raised the volume of the background noise of each digit. Although the bot success rate fell noticeably (10-15% success), the produced audio CAPTCHA were too difficult to solve for humans, as the loud background noise made it hard for the users to distinguish the digits spoken.

In Stage 5 we introduced loud noise between the digits (intermediate noise) (Fig. 5(d), 5(h)). This resulted in the bot being unable to segment the audio file correctly. This happened because there were more energy peaks than the digits spoken. The loud intermediate noises were recognized as additional digits, because they produce high energy peaks as well, when transformed with the Discrete Fourier Transformation. As a consequence, the bot could not be trained, as it failed to recognize successfully any digits.

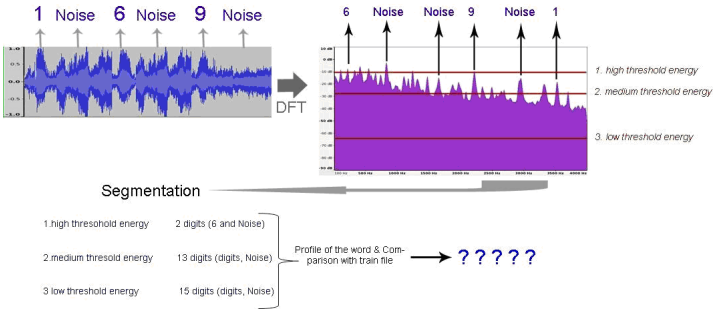


Fig. 6. Demonstration of a bot failing to solve a CAPTCHA

Stage 5 is described, in more detail in Fig. 6. When the bot transforms such an audio into the frequency domain, the energy peaks that can be found are both digits and noise. Therefore, the bot recognizes more digits than those which are actually included in the file. One possible solution for the bot would be to raise or lower the threshold of the energy. In that case (see Fig 6), the bot would still fail. If the threshold energy is very high, then the bot would not recognize some of the digits in the CAPTCHA, while at the same time it would recognize some intermediate noise as digits. On the other hand, if the threshold energy is lowered, then the bot would recognize all digits, but at the same time all intermediate noises would also be considered digits, as well. As a result, the bot would assume that there were 12 or 15 digits in the CAPTCHA.

CAPTCHA testing

User and bot success rates are the main factors that form the decision of whether a CAPTCHA is efficient or not. The corresponding success rates, regarding the CAPTCHA we described in section 5.2, appear on Fig. 7. Each attribute added efficiency to the CAPTCHA and directly affected the user and bot success rates.

The CAPTCHA developed in Stage 5 had an average user success rate of 83%, with an average bot success rate less than 1%.[1]

CAPTCHA implementation

During the implementation of the proposed audio CAPTCHA, the audio files had the following attributes:

1. They were created automatically; therefore they can be updated at random time-periods, without human intervention. The overall process for creating a full set of 3-digit CAPTCHA took 8 sec, whereas for creating a full set of 4-digit CAPTCHA it took 107 sec. Thus, the reproduction of the whole set of CAPTCHA does not

[1] The users who were invited to solve the CAPTCHA (sample) were 22 in number and mainly between 20-30 years old. Most of them were university students. All CAPTCHA were in english, which was not the mother tongue of any of the participants (there was a requirment for them to speak english).

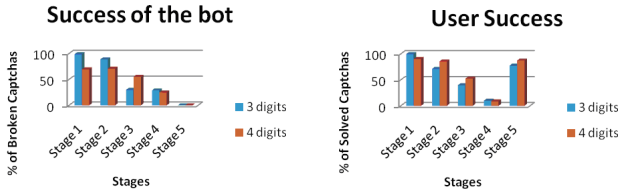


Fig. 7. Bot's and users' success rates

cause significant overhead to our VoIP infrastructure (the VoIP server was a 2GHz Core2Duo, with 2GB RAM).

2. All constituting parts of the audio CAPTCHA, such as the digits and the noise, lay in different folders. Moreover, each time a set of CAPTCHA is produced, the program selects randomly each digit from a different announcer, as well as a random background noise.
3. The noise between the digits is random and has different volume and energy.
4. The pronounced digits and the noise have random duration, which results in a random duration of each audio CAPTCHA.

6 Conclusions, Limitations and Future Plans

CAPTCHA are expected to play a key role for preventing email spam and voice spam (SPIT) in the near future. In order for them to be effective, they must be easy to solve for humans, while at the same time very hard for bots to pass. The CAPTCHA we proposed incorporated several attributes, such us different digit announcers, background noise against each digit, noise between digits and all of them in a random1 and automated way. We produced audio CAPTCHA, which are regularly refreshed, with a limited chance of creating the same instance of an audio CAPTCHA more than once. The production of the CAPTCHA was done in five discrete stages. Each time the CAPTCHA were tested both with a frequency and energy peak detection bot, as well as by a number of users. The bot managed to achieve a high success rate during the first four stages (up to 98%), but that rate dropped dramatically at the last one (less than 2%).

Additionally, we determined an appropriate level of background noise of each digit, in order for them to solvable by humans and difficult to break by bots. However, each attribute alone is not enough for making CAPTCHA robust; it is the combination of the features that make the CAPTCHA resistant. Needless to say, every CAPTCHA is efficient, as long as there is a high rate of success for humans and a low one for bots.

A limitation of the proposed CAPTCHA is that there has been no evaluation of its effectivenss and its attributes by audio/speech recognition tools, such as HTK which uses Hidden Markov Model (HHM). Most audio recognition systems are based on identifing conversations, where each word should have a connection with other words. However,

an extensive research with an Automated Speech Recognition (ASR) system could support a more reliable evaluation of our implementation Also it would be interesting to compare our CAPTCHA with other audio CAPTCHA implementations [18], that have not been tested with the particular bot yet.

Another possible extension is to consider different populations of users and take into consideration the specific requirements of each set. This could be done if a major SIP provider can provide personalized services to its clients and, as a result, provide various CAPTCHA types for each specific user.

Acknowledgement. This work has been performed within the SPIDER (COOP-32720) Project, which is partly funded by the European Commission. The authors would like to acknowledge the contribution of our colleagues from Athens University of Economics & Business (GR), Fokus (D), Eleven (D), Voztelecom (E), Telio (NO) and IPTEGO (D).

References

1. Turing, A.: Computing machinery and intelligence. Mind LIX(236), 433–460 (1950)
2. Blum, M., von Ahn, L., Langford, J., Hopper, N.: The CAPTCHA Project (November 2000)
3. von Ahn, L., Blum, M., Hopper, N., Langford, J.: CAPTCHA: Using hard AI problems for security. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 294–311. Springer, Heidelberg (2003)
4. von Ahn, L., Blum, M., Langford, J.: Telling Humans and Computer Apart Automatically. Com. of the ACM 47(2), 57–60 (2004)
5. von Ahn, L., Maurer, B., McMillen, C., Abraham, D., Blum, M.: CAPTCHA: human-based character recognition via Web security. Science 321(5895), 1465–1468 (2008)
6. Chellapilla, K., Larson, K., Simard, P., Czerwinski, M.: Building Segmentation Based Human Friendly Human Interaction proofs. In: Proc. of the SIGCHI conference on Human Factors in Computing Systems, pp. 711–720. ACM Press, New York (2005)
7. Chew, M., Baird, H.: Baffletext: A Human Interactive Proof. In: Proc. of the 10th SPIE/IS&T Document Recognition & Retrieval Conference, USA, pp. 305–316 (January 2003)
8. Mori, G., Malik, J.: Recognizing objects in adversarial clutter: Breaking a visual CAPTCHA. In: Proc. of the Computer Vision and Pattern Recognition Conference, pp. 134–141. IEEE Press, Los Alamitos (2003)
9. Defeated CAPTCHA (retrieved May 18, 2008), `http://libcaca.zoy.org/wiki/PWNtcha`
10. Yan, J., El Ahmad, A.: Breaking Visual CAPTCHA with Naive Pattern Recognition Algorithms. In: Samarati, P., et al. (eds.)Proc. of the 23rd Annual Computer Security Applications Conference (ACSAC 2007), pp. 279–291. IEEE Computer Society, Los Alamitos (2007)
11. Yan, J.: A El Ahmad, A Low-cost attack on a Microsoft CAPTCHA, Technical Report, School of Computer Science, Newcastle University, United Kingdom (February 2008)
12. Jurafsky, D., Martin, J.: Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition, 2nd edn. Prentice-Hall, Englewood Cliffs (2008)
13. Defeating Audio (Voice) CAPTCHA (retrieved October 10, 2008), `http://vorm.net/captchas/`

14. SIPP Traffic Generator for the SIP Protocol (retrieved September 30, 2008), http://sipp.sourceforge.net/
15. Breaking Gmails Audio CAPTCHA (retrieved October 10, 2008), http://blog.wintercore.com/?p=11
16. HTK: Hidden Markov Model Toolkit (retrieved October 10, 2008), http://htk.eng.cam.ac.uk/
17. SPHINX: The CMU Sphinx Group Open Source Speech Recognition Engines (retrieved January 2, 2009), http://cmusphinx.sourceforge.net/html/cmusphinx.php
18. Tam, J., Simsa, J., Huggins-Daines, D., von Ahn, L., Blum, M.: Improving Audio CAPTCHA. In: Proc. of the Symposium on Usable Privacy and Security (SOUPS 2008), USA (2008)