https://doi.org/10.2352/EI.2023.35.4.MWSF-373

Audio CAPTCHA Breaking and Consequences for Human Users

Fabian Oberthür; TU Darmstadt; Germany
Martin Steinebach, Verena Battis; Fraunhofer SIT | ATHENE; Darmstadt; Germany

Abstract

On the Internet, humans must repeatedly identify themselves to gain access to information or to use services. To check whether a request is sent by a human being and not by a computer, a task must be solved. These tasks are called CAPTCHAs and are designed to be easy for most people to solve, while at the same time being as unsolvable as possible for a computer. In the context of automated OSINT, which requires automatic solving of CAPTCHAs, we investigate the solving of audio CAPTCHAs. For this purpose, a program is written that integrates two common speech-to-text methods. The program achieves very good results and reaches an accuracy of about 81 percent. As CAPTCHAs are also an important tool for Internet access security, we also use the results of our attack to make suggestions for improving the security of these CAPTCHAs. We compares human listeners with computers and reveal weaknesses of audio CAPTCHAs.

Introduction

Tools that can solve CAPTCHAs automatically offer many advantages for research in any discipline, as it facilitates access to vast datasets and information sources on the Internet. These data sets can then be downloaded and used through web scraping. However, many operators have a commercial interest in their websites, so they try to make their resources accessible only to people and not to computers. One solution to find out whether a request was made by a human or a computer is the ”completely automated public Turing test to tell computers and humans apart”, abbreviated by CAPTCHA [1]. Only after the test has been passed successfully, the requested resources are made available.

In principle, CAPTCHAs can be used wherever it is important to know that a certain action is performed by a human being. When a clear distinction can be made, both users of the Internet and website operators have significant advantages. Some of these are that users on the internet are protected from excessive SPAM, both in their email inboxes and on social media, because verification prevents robots (BOTs) from creating and abusing thousands of email addresses in a very short time. In this way, website operators can use their resources more sparingly and only grant actual people access to their information, data and applications. Only after the test has been successfully completed, the requested resources will be made available. Please note, CAPTCHAs do not serve to protect the user or their data. It only protects the operator's resources.

In general, visual CAPTCHAs were used in the vast majority of cases. This might be motivated by the fact that humans are able to intuitively understand visual contexts – a task that for a long time was at least very difficult, if not impossible, for computers.

To enable people with impaired visual perception to verify themselves as human beings on the Internet, there is a form of CAPTCHA called *audio CAPTCHA*. In this case, a sentence is read out to the user that has been distorted audiovisually. Usually, a noise is superimposed on a discretely spoken sentence. The task is now to understand the spoken sentence and enter it in the given field. If the given answer matches the stored solution, the CAPTCHA is considered passed. The user is authenticated as a human being.

With the advent of speech-to-text (STT) solutions, such as the Google STT API, a new vulnerability of audio CAPTCHAs has emerged: audio CAPTCHAs can be broken automatically using such STT methods. However, if audio CAPTCHAs are insecure, they are no longer offered as an alternative to visual CAPTCHAs. This puts users with visual impairments at a disadvantage, which in turn violates applicable law, as people may not be discriminated against on the basis of a certain characteristic – including a disability. Therefore, in the second part of the thesis, a scientific study will be conducted to investigate how audio CAPTCHAs can be modified to close the security gap without worsening the usability for people.

State of the Art

In this section, the topics and procedures relevant to the work are introduced and explained.

CAPTCHAs

The original CAPTCHA was invented by Reshef, Raanan and Solan in 1997 [2] and required to enter a sequence of letters or numbers that were bent or distorted in some other way. The term *CAPTCHA*, which is a homophone of the English word 'capture', was first introduced by Luis von Ahn, Manuel Blum, Nicholas J. Hopper and John Langford in 2000 [3]. One of the earliest uses and the first commercial application of CAPTCHA was in 2001 when *PayPal* used CAPTCHAs to prevent fraudster from using bots to create accounts [4].

The ”fully automatic public Turing test for distinguishing between computers and humans” must fulfil two criteria: First, it must be easy to solve for most people, and at the same time, second, it must be as unsolvable as possible for computers [5]. These tests usually represent so-called challenge-response tests. The user is given a task (challenge) to solve and then returns the result (response). If the response is correct, the test is considered passed.

WordCAPTCHAs

The WordCAPTCHA is the original version of the CAPTCHA as developed by Reshef et al. [2]. The task is to uniquely recognise two different, distorted words and enter them into a given input field. If the answer given matches the stored solution, the person has successfully authenticated themselves as such.

ImageCAPTCHAs

With this type of CAPTCHA, the user receives an image that is divided into equal-sized squares. The challenge here is to understand a task that is in the following form: Select all squares with [traffic lights, bicycles, cars, etc.]. The answer is to click only on the grids in which the object you are looking for is partially or completely contained. If all fields with the object are selected, the CAPTCHA is considered to have been successfully solved.

audio CAPTCHAs

Lastly, the audio CAPTCHA is introduced. In order to enable people with visual impairments to verify their identity on the internet, there is the audio CAPTCHA. The audio CAPTCHA works very similarly to the two CAPTCHA variants already described. Here, however, a sentence is read out to the user, which has been distorted audiovisually. If the user is able to understand the sentence read out and enter it correctly in the input field, the corresponding audio CAPTCHA is considered solved.

Further development of CAPTCHAs

In addition to the actual task of protecting resources from misuse by bots, large companies such as *Google LLC* (now *Alphabet*) saw another advantage in CAPTCHAs, which is why they acquired the CAPTCHA service *reCAPTCHA* in 2009. The billions of data points labeled each day, free of charge, by millions of internet users have been used by Google in several projects. E.g., 2009 in the Google Books project [6, 7], or in 2012 for Google Street View [6, 8]. In 2014, the company started to use CAPTCHAs, or rather the answers to the CAPTCHAs, to train their artificial intelligence. This all happened without the knowledge of the users. In addition to the CAPTCHA, which was used for authentication as a human being, the users were given another task to solve - disguised as part of the CAPTCHA. Here, words that were not clearly recognized by a computer program were shown to several users as part of a CAPTCHA. With the help of the bundled answers, Google was able to assign a probability-weighted meaning to the unidentifiable word.

Since 2009, *reCAPTCHA* has been constantly developed and improved. In the process, the nature of the tasks to be solved has changed again and again and become more challenging. In 2014, Google published the second version of reCAPTCHA. Here, it was no longer necessary to recognize words, but to select pictures that had a special characteristic. For example: "Click on all pictures that show a traffic light". In 2019, the third version of *reCAPTCHA* appeared, which contains a number of improvements, including a score indicating how likely the user is to be classified as human. This score is calculated in the background and depends on a number of factors [9], including

- IP address
- mouse and keyboard usage
- fingerprint of the device
- Referrer URL
- date and time

However, it is not clear whether and to what extend other data is included in the evaluation. Data protectionists sharply criticize Google for this procedure. If the score is sufficiently high, the user no longer has to solve a CAPTCHA. In this so-called "customer-friendly" design, a simple click on a button is sufficient for authentication from now on.

In 2020, the version *reCAPTCHA Enterprise* was finally released, in which the "customer-friendly" design was expanded. This means that more data is collected and evaluated in the background. However, since it is not clear what data is collected and how it is processed, there is repeatedly criticism from experts who express concerns about data protection. The subject of this work is the third version, *reCAPTCHA v3*.

Security vulnerabilities

In the digital age, every implemented verification procedure triggers a race between the developers and people or groups of people who have an interest in circumventing the verification procedure. This is also the case with CAPTCHAs, which is why they have to be constantly improved and changed.

In 2008, a group of *spammers* succeeded in developing a method that managed to circumvent between 20 and 30 percent of the CAPTCHAs. These were CAPTCHAs used by Google to protect their free mail program Google-Mail. As a result, the percentage of spam mails sent by Google mails increased from about 1.3 percent to 2.6 percent [10].

Another breakthrough was made in 2017 by four researchers at the University of Maryland who developed the "low-resource attack" *unCAPTCHA* [11]. Only one laptop and one IP address were used. The results obtained are thus much scarier than those published by large collectives that have almost unlimited resources, since the resources needed for the unCAPTCHA attack are available to almost everyone. The developers have successfully solved about 85 percent of 450 of Google's audio CAPTCHAs. The authors, of course, informed Google about unCaptcha, whereupon they updated their audio challenges. From then on, the user had to correctly identify phrases or sentence fragments instead of digits to authenticate as a human. Shortly thereafter, the authors showed that their original approach, with some modifications, could also crack Google's reCAPTCHAv2 - and this time even with an accuracy of 90% [12]. After the release of *unCAPTCHA v2* on Jan. 18, 2019, Google responded by revising their challenge again.

In 2021, Nikolai Tschacher then developed the program *unCAPTCHA v3*. This further development of *unCAPTCHA v2* achieved an accuracy of 91 percent [13] with its adapted mode of operation.

Meanwhile, Google no longer serves all requests for audio CAPTCHAs, but only if the request is deemed trustworthy. The evaluation that runs in the background plays a decisive role here. If the rating is too low, Google identifies a supposed bot and blocks further requests.

The last major security vulnerability became known in 2022. It affected Google's audio CAPTCHAs. The user had to play an audio CAPTCHA and enter a sentence consisting of at least ten words as a response. The choice of words did not matter. However, Google was able to close this loophole within a few hours of it becoming known [14].

Concept

For audio CAPTCHA breaking, we use existing speech-to-text solutions. The task is to transcribe spoken text and put it

into a written form. In general, STT procedures consist of two parts. In the first step, preprocessing takes place. In this process, the audio files are converted from an analog signal into a digital bit sequence. The background noise is filtered out, and the time signal is then converted into a frequency spectrum using a Fast Fourier Transform. In the second step, the actual recognition takes place. Either hidden Markov models or neural networks are often used for this purpose. These try to generate words from the given signals. Language models are then used to determine the probability of words to follow a given word. Today, the most common methods provide results that are quite comparable to human performance.

Implementation

In this section we describe our implementation to automatically solve audio CAPTCHAs.

The program was written in Python version 3.8. The implementation was done using the *PyAutoGui* library, which allows the cursor to be moved and input to be generated via the keyboard [15]. To edit the MP3 audio files, *pydub* is used. It allows to open, edit and save MP3's in a suitable format [16]. The format required is dictated by the STT procedures used.

In this work, two different methods are used to transform audio files to text files. To integrate Google's STT API, the module *SpeechRecognition* is utilized [17]. The implementation of a local solution is done with the module *Vosk* [18].

Furthermore, the modules *time*, *json*, *random* and *wave* already included in Python are employed.

Structure

The aim of the program is to solve audio CAPTCHAs automatically without being detected as a bot. To achieve this, the following processing structure was implemented.

1. The page where the audio CAPTCHA to be solved is located is accessed in private mode.
2. A new audio CAPTCHA is loaded.
3. Before the audio file is downloaded and converted to a suitable format, it is checked whether the program has been detected as a bot. If this is the case, the program jumps to **step nine**.
4. Once the file is converted to the desired format, the program checks if the file is undamaged. If the file is undamaged, it is transcribed using the STT method and saved as a string. If the file is not OK, the program jumps to **step two**.
5. The saved string is entered in the field provided.
6. Finally, it checks whether the CAPTCHA was successfully solved.
7. If the CAPTCHA is solved, the program saves the results and jumps to **step one**.
8. If the CAPTCHA is not solved, the program tests whether it has been exposed as a bot. If not, the programs jumps to **step two**, otherwise to **step nine**.
9. The program is terminated.

The steps described above are processed one after the other until the program is terminated.

Results for four consecutive test runs. NoT shows the number of attempts needed until a CAPTCHA was solved. Analyzed shows the total number of CAPTCHAs requested in the respective run to solve 100 CAPTCHAs using Google's STT API.

Run	NoT 1	NoT 2	NoT 3	NoT 4	Analyzed
1	82	14	3	1	123
2	77	19	3	1	128
3	86	12	1	1	117
4	29	5	1	0	42
$\sum$	274	50	8	3	410

Evaluation

The goal is to find out how many audio CAPTCHAs can be successfully solved without the program being detected as a bot. Four consecutive runs are performed in which as many CAPTCHAs are requested until 100 CAPTCHAs have been solved successfully. If the program is detected as a bot, the current run is terminated. Table 1 depicts the series of tests performed with Google's STT method. It shows that in the first three test runs, 66% of the attacks were successful in the first attempt. Run four was terminated after 35 successfully solved CAPTCHAs because the program was detected as a bot.

Overall, it can be concluded that the automated program was able to solve 335 CAPTCHAs out of 410 requested CAPTCHAs, resulting in a success rate of 81.7%. Although the program was ultimately detected as a bot, the experiment revealed two important facts:

- the in 2017 proposed unCAPTCHA attack is still valid and
- it is possible to use STT solutions to bypass audio CAPTCHAs in order to crawl the Internet or create fake accounts.

After verifying, that the threat Bock et al. [11] unveiled in 2017 is still valid, we took a closer look at a Google reCAPTCHAv3. Since all examined CAPTCHAs in this work are reCAPTCHAv3s, an exemplary representation of a Google reCAPTCHAv3 is depicted in Figure 1. Utilizing the spectrographic view and with the help of a frequency analysis, we can conclude that all examined CAPTCHAs are structured in the same way. Namely,

- The audio file is divided into two distinguishable parts.
- First part: noise. Second part: a spoken sentence.
- The spoken sentence is recited by different readers.
- The noise is always brown noise.
- There is little to no background noise in the second part.

Especially this last finding poses a severe security risks which – as we demonstrated previously – can be used to bypass the audio CAPTCHA. Which leads us to the second part of this work - a human-computer comparison on how to improve the security-utility trade-off of audio CAPTCHAs without degrading their utility for visually impaired people.

Study: Human-Computer Comparison

The aim of the study is to create a direct comparison between the capabilities of humans and the capabilities of STT-

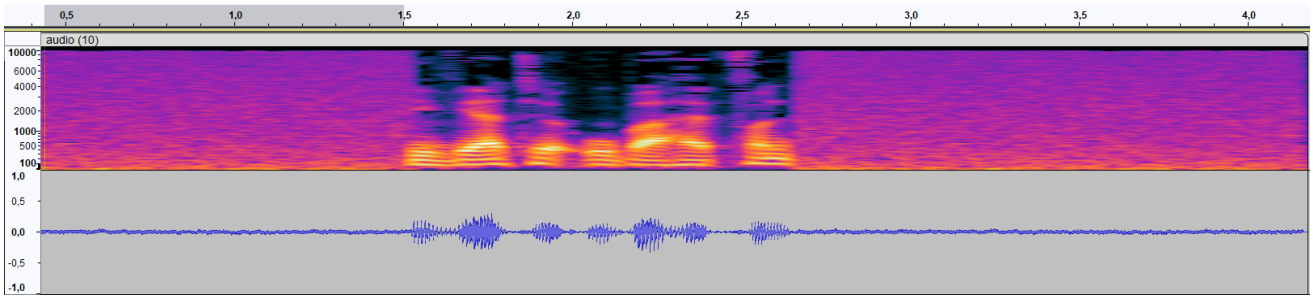

Figure 1. *Exemplary detail view of a Google reCAPTCHAv3. Top: spectrographic view, bottom: wave form*

algorithms in understanding and reproducing differently distorted words. These words were exclusively common words of the German language, selected by means of a random generator and recorded by the authors themselves. In this study, seven participants and 30 audio files were observed to compare the effect of audio interference on humans and algorithms. For this, the previously used STT API from Google, as well as a local solution from *Alpha Cephei Inc.*, *Vosk* [18], are used. It should be noted that due to the small number of participants and the small amount of audio files, the results obtained by this study cannot be considered representative. In this work they only serve to get a first impression of the current state. To evaluate the performance of humans and algorithms, the two evaluation measures *Jaro Distance* and *Levenshtein Distance* are employed.

In order to apply these two measures, the results of the human participants and those of the STT methods must be converted into a lower case string without spaces. For example, the string ***An Example Sentence*** becomes ***anexamplesentence***. The generated strings are then examined for similarity to the original string of the audio files using the two metrics.

Levenshtein Distance [19, 20] counts the number of operations needed to transform the first string X into the second string Y. To achieve this, characters can be either inserted, deleted or replaced with another character. If the two strings are identical, the Levenshtein distance is zero, while the maximum possible distance is equal to the total number of characters in the longer string. But the distance is always at least as large as the difference in the length of the two strings.

Jaro Distance [21] returns a number between zero and one, where zero means no similarity between the strings and one indicates that the two strings are identical.

Set-up: Participants and algorithms receive the identical audio files, 30 each in total. Each audio file is between two and five seconds long and contains either two, three or four words. In total, there are ten files each with two, three and four words. These 30 files are divided into six equal-sized blocks and are always distorted in the same order: no noise, white noise, pink noise, brown noise and mixed noise. The first file of each block is intentionally not distorted to ensure that the setup in which the study is conducted is correct and not disturbed by external influences.

The study aims to answer the following three hypotheses:

H1: Humans solve each type of distorted CAPTCHA better than the algorithms tested.

H2: The type of distortion has no influence on the human result.

H3: The more words used, the better the average performance of the algorithms compared to humans.

The participants are played the 30 audio CAPTCHAs one after the other. Between two files, they have enough time to enter what they have heard into the study sheet. After one block, they are asked to solve a simple math problem to clear their minds so that the next block of trials is not distorted by the test-sentences they heard before.

As can be seen from the table 2, we cannot find clear support for hypothesis H1. On the one hand, we find that humans are better at dealing with mixed noise than the two algorithms under investigation. On the other hand, we also observe that Google can handle both white and pink noise much better than the human survey participants.

Furthermore, we have to reject hypothesis H2. The results presented in 2 clearly show that the type of distortion used to superimpose the spoken sentence influences human perception. We observe that people can abstract better from brown and pink noise than from white noise.

Finally, we tested whether the trade-off between security and utility of audio CAPTCHAs can be improved by increasing the number of words read to the user. While we indeed observed an increase in performance by increasing the number of words from two to three for both Vosk and human participants (see table 3), this increase was followed by a sharp decrease once four words within a sentence were tested. The Google STT API, on the other hand, showed a constant drop in performance throughout this complete series of tests. Therefore, we also had to reject hypothesis H3.

Over all 30 audio files, the two algorithms achieved an average similarity (Jaro) of 0.6. The average used operations to transfer from one string to another is 11.94 (Levenshtein). The subjects of the study achieved an average similarity (Jaro) of 0.65. The best result was 0.79, the worst 0.61. On average, 10.26 operations had to be performed (Levenshtein).

Discussion

Automated breaking of audio CAPTCHAs has been shown in previous research [1], [3] and tools are available e.g., on github [2]. To our knowledge, a systematic comparison of human performance of CAPTCHA solving and automated CAPTCHA breaking to discuss the cost of improving the security of audio CAPTCHAs for human users has not been executed so far. We

Comparison of white, pink and brown noise for Google's STT API, the Vosk API and human study participants, where LS = Levenshtein, J = Jaro.

	Google		Vosk		Human	
Noise	$Ø_{LV}$	$Ø_J$	$Ø_{LV}$	$Ø_J$	$Ø_{LV}$	$Ø_J$
white	13.33	0.59	18.17	0.24	15.38	0.44
pink	7.33	0.76	9.17	0.74	9.62	0.72
brown	5.83	0.85	5.17	0.87	2.36	0.93
mixed	18.17	0.40	18.33	0.30	13.67	0.51
Ø	11.17	0.65	12.71	0.54	10.26	0.65

Average Jaro distance results for Google's STT API, the Vosk API, the mean performance of both algorithms and for the human study participants. 2W, 3W and 4W are the shorthands for two, three and four words respectively.

	ØGoogle	ØVosk	ØAlgo	ØHuman
2W	0.718	0.542	0.630	0.690
3W	0.666	0.708	0.687	0.764
4W	0.644	0.638	0.641	0.674

see this work as an update on recent developments in the field of speech-to-text related to CAPTCHA breaking, and a look ahead at what may happen in the future for attackers and users.

Although the program was ultimately exposed as a bot that prevented further use, a total of 410 audio CAPTCHAs could be requested in succession. Of these, 335 were successfully solved. This corresponds to a success rate of 81.7%. According to the evaluation criteria for a low-resource attack, a CAPTCHA is considered broken if the automated solution has a success rate of more than 70% [22, 23]. Thus, the isolated audio CAPTCHA is considered broken.

This poses a significant security risk due to the nature of the attack. A setup available to every standard user was used for the attack. This means that anyone can circumvent the audio CAPTCHA of *reCAPTCHA* with little effort and use it for any purpose [3].

Our study comparing humans and computers included 30 different audio CAPTCHAs that had to be reproduced. The evaluation shows that the STT method of *Google* with an average Jaro-similarity of 65% performed worse than most participants (five out of seven). However, it is already in a similar range (less than 10 % deviation from the average result of the participants) to humans.
The STT method of *Vosk* with an average Jaro-similarity of 54% , achieved worse results than most participants (six out of seven). Furthermore, the study showed that both, the method chosen to distort the audio files and the number of words, are non-trivial for security (difficult to solve for the computer) and usability (easy to solve for humans).

Summary and Conclusion

Audio CAPTCHAs are an important alternative to visual CAPTCHAs for visually impaired users. They use distortion, especially the addition of noise, of the audio material to distinguish computers from humans. Due to the progress of speech-to-text solutions, this concept can be attacked with limited resources. Currently, the performance of humans and computers is already comparable when solving audio CAPTCHAs.

From a security perspective, such low discrimination between computer and human is problematic and fails the requirements for a CAPTCHA. For a user, an increase of distortion to prevent computers from successfully performing the task will lead to frustration as the human is also likely to fail at the required level of distortion.

As access protection mechanisms rely more on behavioral analysis of the user and less on audiovisual CAPTCHAs, this problem becomes less pressing. Still, for a limited number of bypasses, CAPTCHA breaking could still be important in the future. Our experiments show that we could access a system using audio CAPTCHA breaking several hundred times before being blocked.

Acknowledgment

This research work has been funded by BMBF and the Hessen State Ministry for Higher Education, Research and the Arts within their joint support of the National Research Center for Applied Cybersecurity ATHENE.

References

[1] Luis von Ahn, Manuel Blum, Nicholas J. Hopper, and John Langford. Captcha: Using hard ai problems for security. In Eli Biham, editor, *Advances in Cryptology — EUROCRYPT 2003*, pages 294–311, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.
[2] Eran Reshef, Gil Raanan, and Eilon Solan. Method and system for discriminating a human action from a computerized action, May 26 2005. US Patent App. 10/790,611.
[3] Luis von Ahn, M. Blum, N. J. Hopper, and J. Langford. Captcha:using hard ai problems for security. *Journal of Cryptology*, 2656.
[4] History Computer Staff. Max levchin - complete biography, history, and inventions, Nov 2022.
[5] Mathias Rauer. Captchas – gratwanderung zwischen sicherheit und usability.
[6] Luis von Ahn and Will Cathcart, 09 2009.
[7] Google. Google books.
[8] Google.
[9] Team datenschutzexperte.de. recaptcha & datenschutz: Ist die nutzung dsgvo konform?, 08 2021.
[10] Daniel Bachfeld. Spammer hebeln googles captchas aus, 03 2008.
[11] Kevin Bock, D. Patel, G. Hughey, and D. Levin. uncaptcha: A low-resource defeat of recaptcha's audio challenge. *woot'17*, 2017.
[12] A low-resource defeat of recaptcha's audio challenge.
[13] Nikolai Tschacher. Uncaptcha3, 2021.
[14] Robert McMillan. Wiederholung bricht google audio captcha, 2022.
[15] Al Sweigart. Pyautogui 0.9.53.
[16] James Robert. pydub, 2011.
[17] Anthony Zhang. Speechrecognition 3.8.1.
[18] Alpha Cephei Inc. Offline open source speech recognition api based on kalid and vosk.

[19] Vladimir Iosifovich Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Dokl. Akad. Nauk SSSR*, 163(4):845–848, 1965.
[20] mathe.zone. Levenshtein-distanz. 10 2020.
[21] Matthew A. Jaro. Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. *Journal of the American Statistical Association*, 84(406):414–420, 1989.
[22] Jennifer Tam, Jiri Simsa, Sean Hyde, and Luis Von Ahn. *NIPS*, 2008.
[23] Elie Bursztein and Steven Bethard. Decaptcha: breaking 75% of ebay audio captchas. 01 2009.

Author Biography

Prof. Dr. Martin Steinebach *is the manager of the Media Security and IT Forensics division at Fraunhofer SIT. In 2003 he received his PhD at the Technical University of Darmstadt for this work on digital audio watermarking. In 2016 he became honorary professor at the TU Darmstadt.*

Fabian Oberthür *completed his B.Sc. in Computer Science at the Technical University of Darmstadt (2022). He is now enrolled as a Master's student in Computer Science at Darmstadt Technical University.*

Verena Battis *completed her Master's degree in Statistics at the University of Trier in 2017. As a research assistant at the Fraunhofer Institute for Secure Information Technology, she investigates the risks that machine learning can pose to individual privacy. Detecting and preventing malicious attacks on trained models, as well as generating privacy-compliant synthetic data, are also among her research areas.*