Riddle me this! Context Sensitive CAPTCHAs

Tobias Urban[1], René Riedel[1], Ulrike Schmuntzsch[2], Norbert Pohlmann[1], and Matthias Rötting[2]
[1]Institute for Internet-Security, Westphalian University of Applied Sciences, Gelsenkirchen
{urban | riedel | pohlmann}@internet-sicherheit.de
[2]Human-Machine-Systems, Berlin Institute of Technology, Berlin
{ulrike.schmuntzsch | roetting}@mms.tu-berlin.de

Abstract — **In modern information society online transactions are an important part of our daily lives. In this work we propose a protocol that allows users to perform secure online transaction even if the used system is not trustworthy or infected with malware. We developed a user-centered protocol that uses a CAPTCHA like approach to prevent attackers from manipulating a transaction without the user or the corresponding server noticing. Therefore, we add context sensitive information about the transaction to a task that is set to the user. This task is designed to be hard to solve for computer programs but easy for humans. To evaluate our approach we conducted a user study and computed the probability by which an attacker can successfully attack the system. We show that a vast majority (>94%) of all transaction can be secured while the system itself remains useable.**

I. INTRODUCTION

Online banking and online transactions are a huge part of the modern information society and will even grow in importance. Let alone between 2007 and 2015 the usage of online banking grew from 25% to 46% in Europe [1]. Due to the rapid growth of applications that include *micro transactions* and the general digitalization of our society the amount of executed online transaction will rise further in the future [2]. Fraudsters already compromised this sector in various ways. An official report published in 2014 by the German Federal Criminal Police Office shows that the damage done by fraudsters in the online banking business rounds up to almost 30 million Euro [3] (in Germany). The actual damage is expected to be way higher due to the large number of unknown cases which haven't been reported by the finical institutions. Most financial institutions simply refund their customers without publishing the occurrence of a fraud (e.g. [4]). This is presumably done because the financial institutions want to avoid the bad publicity that comes along with this kind of press releases.

Modern security mechanisms (like *mobileTAN*) are successfully attacked by fraudsters [5]. *mobileTAN* is broadly used in the online banking business to secure transactions using a second channel, the user's smartphone. In 2014 56% of all online banking users in Germany used *mobileTAN* [6]. *mobileTAN* is a mechanisms that sends a SMS to the user's smartphone when the user wants to perform a transaction. The SMS contains aside the transaction data a transaction number (TAN) which is used to authorize the transaction [7]. Nowadays, this mechanism is broken by infecting the user's smartphone and stealing the authorization SMS allowing the attacker to automatically perform any transaction if she has access to the user's online banking account [5].

Successful attacks on online banking systems are enabled by *Man-in-the-Browser* (MitB) attacks [8]. A MitB attack allows the attacker to get full control over the user's browser giving her the opportunity to control all data sent from the server to the user and vice versa. This results in giving her full control over the website's presentation on the user's device. For example the attacker could manipulate the payee's account number that is sent to the banking server but still display the account number the user originally entered. To avoid this, a second channel is needed to verify the data the finical institution received since the user can't trust the display of his own device. If the second channel is successfully attacked, the entire system is broken.

In this work we introduce a protocol which allows the user to perform a transaction on an infected device without the need of an external 'trusted display' (e.g. a smartphone). The protocol ensures - with a high probability - that the attacker cannot manipulate the transaction without the user or server noticing. We adapt the principals of CAPTCHAs (*Completely Automated Public Turing Test To Tell Computers and Humans Apart*) [9] to the given problem (see section II).

CAPTCHAs have been developed to decide whether an application is used by a human or a computer program. In order to do so the user has to solve a task that *should* be easy to solve for humans but hard for computer programs.

We alter this approach by adding context sensitive information to the CAPTCHA to secure a transaction. The user has to answer an automatically generated question that is related to the transaction that he wants to perform. The idea behind this approach is if an attacker manipulates a transaction, there are two possible scenarios: (1) If the attacker modifies the user's view, the answer to the question won't be the answer the server expects (section VI) since it is the answer to the original transaction. (2) If the attacker doesn't manipulated the user's view, the user notices the manipulation since the wrong information are displayed.

Obviously the attacker will try to *guess* the correct answer to surpass the security mechanism.

Possible attacks on the system are discussed in section IV of this paper. A security inspection of the system, based on the expected attacks, is done in section VI. A very important part in the IT-security world is the acceptance of a security mechanism by the users. Therefore, we conducted a small usability

study ($n = 30$) of our developed system. The results of the study are discussed in section VII.
The main contributions of this work are:

- We present an adaptation of the widely used CAPTCHA mechanism and apply it to online transactions.
- We introduce a protocol which allows users to perform secure online transactions even if the system they use is infected with malware.
- We discuss the usability and security of our introduced approach.

II. The Problem

If one wants to perform secure online transactions most of the occurring problems can be sorted into three main categories. These are on the one hand technical security problems that are currently abused by attackers and on the other hand limitations that occur due to the tensions between security and usability.

A. *Stolen login information*

If an attacker wants to gain control over a user's account, it is inevitable to steal the user's login information. Nowadays, almost every web-application uses the combination of username and password in order to authenticate its' users. Hence, attackers need to steal the username and password to gain access to the user's account.

B. *Manipulating the transactions*

The main problem if one wants to perform online transactions on a device that is infected with specialized malware is that the attacker gains total control over all information that are processed by the browser. Thus, the attacker can manipulate all information displayed in the browser (e.g. account balances or the destinations of a transaction). The attacker can add, modify, or delete content just as she pleases [8]. This is utilized by the attacker to perform sophisticated social engineering attacks to steal sensitive information from the user (e.g. the cell phone number) [10]. Data send to the server, usually via HTTPs messages, can be manipulated by the attacker with only very little chance of the user to notice. Note that all manipulations are possible even if the client and server communicated via a correctly established, state of the art TLS connection because the manipulations take place after/before the content is decrypted/encrypted on the client. As a result of this problem the user cannot check if the server received the information the user intended to send and vice versa.

C. *Trusted Display*

Several problems emerge from the problem stated above. One solution to this problem is using a second independent communication channel as 'trusted display' (e.g. a smartphone). This display can be used to check the data received by the server. Online banking applications all over the world use a smartphone, an external card reader, or TAN-generators (these generate transaction specific TANs) as trusted displays [11–13]. Card readers and TAN-generators offer a huge security improvement since the attacker cannot install any malware on these devices. In a practical environment these devices lack of mobility and are mostly used at home. Due to this fact a lot of users prefer to use their smartphone which can be attacked more easily [6]. Successful attacks on smartphones are carried out more and more often by attackers which means that the secure channel is broken more often [5].

III. Concept

The protocol we introduce in this paper uses aside a digital identity no further entities (software or hardware), and could therefore be used on any device with an installed web browser. This includes devices that are not fully trustworthy or infected with specialized banking malware.
From now on in this paper we use online bank wire transfers as example of online transactions and describe our approach according to this example. In general the protocol can be used for any online transaction (e.g. shopping in an online shop).

A. *Usage of digital identities*

Within the protocol we use a digital identity to authenticate the user (2-factor-authentication) and to authorize the transaction by signing it digitally. An authentication with username and password is also possible but the 2-factor-authentication prevents the usage of a stolen username and password to authenticate the user. The transaction is digitally signed using the digital identity to protect it cryptographically and to authorize it. We cannot assume that a digital identity comes with a trustworthy display to verify the transaction data. Therefore, another trustworthy way to check the transaction data is needed. In this work we introduce a CAPTCHA like question that refers to the transaction. The user needs to answer the question to authorize the transaction. The question is explained in more detail in section III.B. Thus, the transaction is secured by answering the question and a digital signature. Since physical interaction with the identity is needed to create the signature it is ensured that an attacker cannot create his own transactions and answer the questions semi automatically once the user logged into his account.
Examples for digital identities that could be used for our approach are electronical identity cards (e.g. the German or Estonian identity card) or the YubiKey Token [14]. Credit or debit cards (if they support NFC) could also be used for this purpose. In general all digital identities can be used if they support 2-factor-authetication and digital signatures.

B. *Adaption of the CAPTCHA principle*

As stated above the user cannot trusted the display of his device. In order to create a trustworthy display we adapted the CAPTCHA principle. In order to authorize a transaction the user has to perform two tasks. The first task is to answer the automatically generated question. The second task is signing the transaction digitally. The question consists of two different types of information (1) information the user needs to answer the question and (2) information that make it harder for the attacker to parse and understand the question automatically (Example: *Add the 3rd number and the 4th number of the account number. The subtraction of the 8th number from the 5th is not of any interest.*). It can be assumed that the attacker has no access to the web-application (or its server), and therefore cannot influence the result the server expects. Hence, an attack is only successful if the attacker can determine the answer to the question (this scenario is discussed in section VI).
When generating the question it is inevitable to refer directly to parts of the transaction that will most likely be manipulated and

which are most important to perform the transaction (e.g. the payee's account number). In the upper example the question is directly tied to the payee's account number which ensures that it cannot be manipulated without changing the answer the server expects. Only if the manipulated account number has the same numbers at the relevant digits our approach has no effect. The probability for this scenario is 2% $(({}^{1}/_{10} * {}^{1}/_{10}) * 2)$. The account number was chosen since a manipulation of the payee's account number is the only way to transfer money to an account the attacker controls.

Our approach does *not* protect every part of the transaction. Only if one would construct a question that refers to all parts of the transaction the whole transaction would be secured. An implication of this fact is that an attacker might manipulate some parts of the transaction (e.g. the amount or reference) without the user noticing it. Ultimately this implies if the questions are stated as above, the user can only be sure that the payee was *not* manipulated. If the question would relate to more parts of the transaction these parts would also be protected but the question would become more complex which would presumably result in a significant decrease of the system's usability. Including the payee in the question implies that the attacker cannot transfer money to an account she controls. This is in contrast to current solutions where the attacker can transfer money to any bank account she controls. Since the possible financial gains are lowered for the attacker her motivation to attack the system is also reduced. A detailed security analysis is presented in section VI.

To support the user four different potential solutions to the question are presented. Hence, answering the question is a multiple choice 'test'. The potential solutions are chosen in a way that these candidates are possible solutions to the confusing numbers in the question (see section V). This is done so that the attacker can't conclude anything about the question from the presented answers. Example: If the question names the numbers 3, 4, and 8 there are different tasks that can be generated using these numbers (e.g. $3 + 4$, $8 - 4$, etc.). Each potential solution should be the solution to one of these tasks. By choosing the solutions in this way the attacker cannot exclude any of the presented solutions, since all of them fit to a potential task for the given question. Offering potential solutions has the upside that if the transaction was manipulated, the correct solution might not be among the presented solutions. This increases the system's security since the user can abort the transaction if no correct solution is presented to him.

The reason why exactly four potential solutions are presented and which influence the amount of potential solutions has on the security of the system is discussed in section VI.

The schematic process flow of our protocol is shown in Figure 1 and is described briefly in the following. After a successful 2-factor-authentification against the web-application (1) the user creates the transaction (2). The web-application generates a question for the transaction and sends it to the client (3). The question is displayed on the user's device and the user solves the question (5). Afterwards the user signs the transaction digitally (6). In the last step the web-application checks the answer (7a) and the digital signature (7b). If both are correct, the transaction is executed.

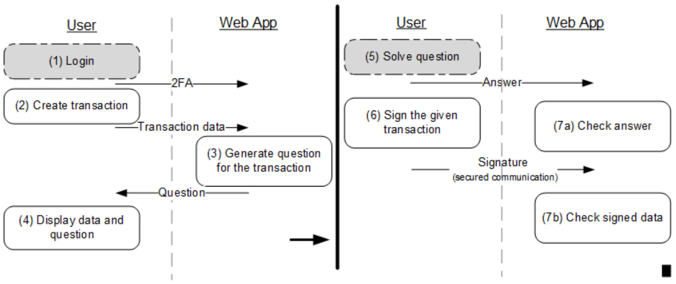


Figure 1 – Schematic process of the protocol

By combining a digital identity and a question, which is generated for a specific context, it is assured that only the person who physically holds the digital identity can perform a transaction. If the transaction was manipulated, either the user or server can detect it. This allows the user to perform transactions even if the used device is untrusted or infected.

The attacker, or someone he hired (e.g. a "captcha solving company" [15], or persons from low-cost countries), could solve the questions by hand. But since the question has to be solved just in time and due to the fact that there *might* be some language barriers the system would still increase the attacker's efforts greatly to successfully attack it. Note that our system targets large scale fraud. An attacker monitoring one transaction in real time has greater chances to successfully attack the system.

IV. Threat Model

Before conducting the security inspection we define the attack model which is used as basis for the inspection. We make the following assumptions about the attacker:

(A1) The attacker *cannot* automatically parse *and* understand the question. The automated understanding of human langue is up till now still a hard problem (see section VIII or for example [16]). Especially Winogard questions [17] are still hard to understand for computer programs.

(A2) The attacker is capable of extracting all words and numbers from the question without understanding their context.

(A3) The questions are *not* generated by a small set of patterns. Thus, the attacker cannot learn these few patterns and try to answer the questions based on that (see also section IX.A).

(A4) The System only uses addition and subtraction in the generated questions/tasks. This is assumed to keep the tasks easy to solve for humans.

(A5) The attacker has *no* access to the server side web-application. If the attacker could manipulate the server there would be no need to attack the client since she can manipulate the transaction on the server.

Based on this assumptions the following attack vector on the introduced protocol is expected.

A. Expected attack vector on the protocol

A transaction can only be manipulate by the attacker if and only if he can present the correct answer to the server and if the user signs the transaction, optimally by physical interaction with the identity. To successfully break the introduced mechanism the attacker has to compute the answer to the question. The attacker can use different resources that help him to do that. These resources are:

- (I1) Single words and numbers she extracted from the question (see A1).
- (I2) All information the user entered for the original transaction (e.g. account number and amount).
- (I3) All information about the manipulated transaction.
- (I4) The user’s answer to the original transaction, if available.

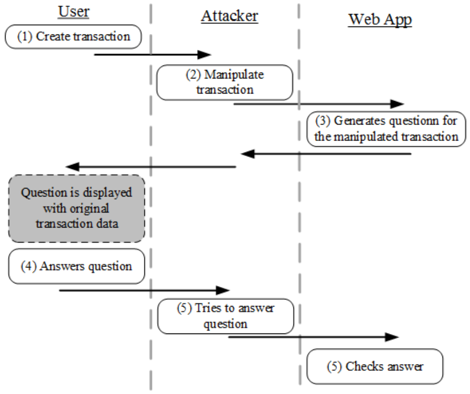

User
Attacker
Web App
(1) Create transaction
(2) Manipulate transaction
(3) Generates questionn for the manipulated transaction
Question is displayed with original transaction data
(4) Answers question
(5) Tries to answer question
(5) Checks answer

Figure 2 – Schematic display of the expected attack

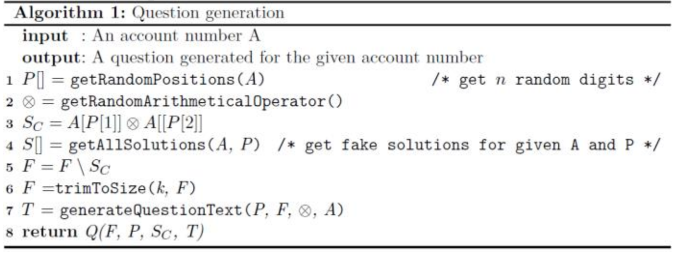
After receiving the question the attacker can extract the *n* digits from the account number (I1). With these *n* digits the attacker can identify the *n* numbers, from (I3), that are candidates to solve the question. Remember that there are some numbers in the question to confuse the attacker. Based on these *n* numbers $3 * \binom{2}{n}$ different tasks can be generated ($2 * \binom{2}{n}$ subtractions and $\binom{2}{n}$ additions). Since all presented potential solutions in the multiple choice test match at least one of those tasks the attacker cannot unambiguously identify the correct solution. But with the information from (I2) the attacker could present potential solutions that would fit to the original transaction. If the user chooses one of these answers the attacker gets new information that she can use to solve the task (I4). Figure 2 illustrates the expected attack on the question.

In order to reduce the information the attacker gains from the question’s text the text could be distorted (like classical CAPTCHAs – see Figure 5). Thus, it is harder for the attacker to automatically extract the information. This is not a reliable way since many CAPTCHAs have already been broken (e.g. [18]) but raises the effort of the attacker. Distorting the text also decreases the usability of the question.

V. Question generation

Given the presented attack vector it is essential that the attacker can extract as few usable information as possible from the question. In this context a question can be defined as the following tuple $Q = (F, P, S_C, T)$ with $F = (F_1; F_2; \dots; F_k)$ different fake solutions, $P = (P_1; \dots; P_n)$ different positions that refer to the transaction, S_C the correct solution, and T the question’s text. To compute these four parameters Algorithm 1 is used. First the algorithm chooses *n* random digits from the given account number *K*. S_C is computed by choosing a random arithmetical operator $\otimes$ and the first two chosen numbers (at the chosen digit). Subsequently the possible fake solutions for the given account number and chosen digits are computed. This ensures that all fake solutions are solutions for at least one task. As mentioned above this is done to reduce the information the attacker can gain from parsing the presented potential solutions.

In the last step the question’s text is generated by using the determined values. When choosing the question’s text it is necessary that different verbalizations are used so that the attacker cannot predict the structure of the question. See also section IX for further improvements.



```
Algorithm 1: Question generation
  input  : An account number A
  output: A question generated for the given account number
  $P[] = $ getRandomPositions($A$)                /* get $n$ random digits */
  $\otimes = $ getRandomArithmeticalOperator()
  $S_C = A[P[1]] \otimes A[[P[2]]$
  $S[] = $ getAllSolutions($A$, $P$)  /* get fake solutions for given A and P */
  $F = F \setminus S_C$
  $F = $trimToSize($k$, $F$)
  $T = $ generateQuestionText($P$, $F$, $\otimes$, $A$)
  return $Q(F, P, S_C, T)$
```

Algorithm 1 – Question generation algorithm

The following example shows exemplary the generation of a question. Input: account number $A = 15324$. The following potions are chosen at random: $P = 0, 4, 2, 1, 3$ the solution S_C is $1 + 4 = 5$ the following *shortened* list of fake solutions is determined $F = 3,\ 7,\ -2, -3$. With these parameters the question can be crafted: *Add the 1st and 5th digit of the account number. The subtraction of the 3rd and 2nd digit are not of any interest.*

VI. Security inspection

In this section we compute the probability by which a motivated attacker can successfully attack the proposed protocol. The following assumptions are made for the computation of the probability:

- (S1) The question is only related to the account number.
- (S2) The question contains five digits from the account number.
- (S3) Only additions and subtractions occur in the questions (see (A4)).

The probability that the manipulated account number has the same numbers at the relevant digits and therefore the attacker can simply adopt the solution is $2 * ({}^{1}\!/_{10} * {}^{1}\!/_{10}) = 0.02$ for addition, the order of numbers doesn’t matter, and ${}^{1}\!/_{10} * {}^{1}\!/_{10} = 0.01$ for subtraction.

We consider two different attacker types and estimate their ability to automatically solve the question: (1) *Simple attacker*: the attacker has not extracted any information from the question, and (2) *Specialized attacker*: the attacker extracted the relevant digits from the question.

A. Simple Attacker

The simple attacker has not extracted any information from the question and therefore the attacker can only blindly guess the correct answer. For each k-digit account number $\binom{k}{2} + 2 \cdot \binom{k}{2} = \Omega$ different tasks can be generated. $\binom{k}{2}$ additions and $2 \cdot \binom{k}{2}$ substractions. The probability that a solution is unambiguous related to the question is: $\frac{n}{\Omega}$ (with n the amount of unique solutions for an account number). A solution ϵ is unique if there is only one task in Ω that has ϵ as solution. Example: With $k = 18$ and $n = 6$ we get $\frac{6}{459} = 0.013$. To compute the total probability that an unambiguous question is proposed we need to define the function $\pi(j)$ that returns the amount of account numbers that have exactly *j* unambiguous questions. The probability that an unambiguous task is selected is computed using the function $\sigma(j) \coloneqq \frac{j}{\Omega}$. Combined with the percentage occurrence of an n-digit number with j unambiguous solutions ($\omega(j) \coloneqq \frac{\pi(j)}{10^k}$) the weighted probability can be computed as follows: $\omega(j) \cdot \sigma\ (j)$. Note that this is the probability that an unambiguous task is selected if an account number has *j* different unambiguous tasks. Thus, the total probability that an unambiguous question is selected can be computed by summing all weighted probabilities $\sum_j\ \sigma(j) \cdot \omega(j)$.

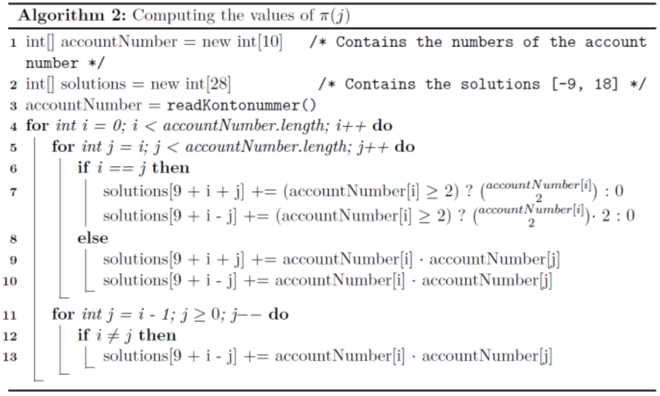
We computed the total probability that an unambiguous task is chosen for $k = 18$.We chose k = 18 so that we can compute $\pi(j)$ in a reasonable time while still using a realistic account number length. European account numbers are between 15 and 32 digits long [19]. The probability of choosing an unambiguous task, for $k = 18$, is almost zero (0.02%). The extreme large amount of different tasks is mostly responsible for this low probability. It must be assumed that the probability will decrease if *k* increases since the number of considered digits increases.

The function $\pi(j)$ is computed using Algorithm 2. The array *accountNumber* contains the occurrence of all numbers in a given account number (e.g. account number = 100333 → *accountNumber* = [2, 1, 0, 3, 0, 0, 0, 0, 0, 0]. Thus, the account number contains two zeros, one one, no two and three threes.

The algorithm fills the array *solutions* in a way that it contains the amount of how often a solution appears for the given account number. Thereby, *solutions[0]* represents the smallest potential solution (-9) and *solutions[27]* represents the biggest potential solution (18). For the previous example we get: account number = 100333 → *accountNumber* = [2, 1, 0, 3, 0, 0, 0, 0, 0, 0] → *solutions* [0, 0, 0, 0, 0, 0, 4, 2, 2, 5, 4, 2, 8, 2, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]. Therefore, the following solutions exists: (-3) → 2 times; (-2) → 2 times; (-1) → 2 times; 0 → 5 times; 1 → 4 times; 2 → 2 times; 3 → 8 times 4 → 2 times and 6 → 1 times. In order to compute all solutions Algorithm 2 iterates over the *accountNumber* array and multiples the occurrence of the given numbers (e.g. If the account number contains three threes and one 1 we get $3 * 1 = 3$ times the result $3 + 1 = 4$). Accordingly, we get 3 times the solution $3 - 1 = 2$ and $1 - 3 = -2$. After termination $solutions[j]\ = \pi(j)$ applies.

B. Specialized attacker

We assume that the specialized attacker parsed the question and extracted all digits of interest but didn't understand the context in which they were mentioned. The attacker also sniffed the answer the user send to the bank. Thus, the attacker knows P, which she extracted from T, and sniffed S_C for the original transaction. Using these information the attacker has a much higher probability to break the system. The total probability that a question is chosen to which the attacker can determine the correct answer can be compute as follows. Since the attacker could extract all five positions of interest from the question (S2) the amount of possible tasks shrinks drastically. There are only $\Omega = \binom{5}{2} + \binom{5}{2} * 2 = 30$ different tasks in this scenario. Analogously to the computation of total probability for the simple attacker we get a total probability of 21.07% that this attacker can successfully attack the system.



Algorithm 2: Computing the values of $\pi(j)$

```
int[] accountNumber = new int[10]    /* Contains the numbers of the account number */
int[] solutions = new int[28]        /* Contains the solutions [-9, 18] */
accountNumber = readKontonummer()
for $int\ i = 0; i < accountNumber.length; i{+}{+}$ do
    for $int\ j = i; j < accountNumber.length; j{+}{+}$ do
        if $i == j$ then
            solutions[9 + i + j] += (accountNumber[i] ≥ 2) ? $\binom{accountNumber[i]}{2}$ : 0
            solutions[9 + i - j] += (accountNumber[i] ≥ 2) ? $\binom{accountNumber[i]}{2} \cdot 2$ : 0
        else
            solutions[9 + i + j] += accountNumber[i] · accountNumber[j]
            solutions[9 + i - j] += accountNumber[i] · accountNumber[j]
    for $int\ j = i - 1; j \geq 0; j{-}{-}$ do
        if $i \neq j$ then
            solutions[9 + i - j] += accountNumber[i] · accountNumber[j]
```

Algorithm 2 – Determines all solution for a given account number.

This is the probability that the attacker can identify the question based on the given information. As expected the probability that a transaction can be attacked successfully rises drastically. But around $^4/_5$ of all transactions could still be executed safely even if the user's system was infected by the attacker.

A highly motivated attacker will also try to learn more about the structure of the generated questions in order to increase her chances to find the correct answer. To counter this either the questions have to be more challenging, which would negatively affect the usably of the system, or the question has to be generated in a way that makes it harder for computer programs to interpret. Examples for such question are so called *Winogard* questions [17]. Winogard questions are multiple choice questions that consist of two sentences that contain an ambiguity that is resolved in opposite ways (see section IX). It can be assumed that these kind of questions wouldn't increase the cognitive workload of the user since the ambiguity in Winogard questions is resolved by common world knowledge that humans naturally have.

C. Adding multiple choice questions

As mentioned in section III the user has to choose the correct answer in a multiple choice like test. The security impact of using such a test is presented in this section. In this section we focus mainly on the security gain by using such a test rather than the user experience. The main advantage of a multiple choice test is that the correct solution to the question might *not* be presented to the user if the transaction was manipulated. In this case the user can abort the transaction and prevent that any damage is done. Obviously the solution might randomly be among the presented potential solutions. Therefore, it is crucial to decide how many potential solutions should be presented to the user. We still assume that the attacker extracted the five relevant digits from the question.

Let *j* be the amount of different solutions for a 5-digit number and let *k* be the amount of displayed solution candidates (with $1 \leq\ k \leq 28$). Since the attacker knows the five relevant digits the amount of different solutions for a 5-digit number needs to be computed. Essentially the question: “How many different solutions has a 5-digit number” needs to be answered. We found no 5-digit number with $i > 18$.

Let $\pi_{\%}(i)$ be the function that returns the percentage amount of 5-digit numbers that have *j* different solutions. In the following we compute the probability that the correct solution for the original transaction is among this *k* pictures.

The probability that the correct solution is *not* displayed can be computed using $\delta(k) := \max(\frac{j-k}{j}; 0)$. For a 5-digit number with exactly j solutions and k displayed pictures the formula $\delta(k) * \pi_{\%}(i)$ computes the probability that the solution for the original transaction is *not* displayed. The total probability that the solution for the original transaction is *not* displayed can be computed as follows: $\varphi(k) := \sum_i \delta(k) * \pi_{\%}(i)$. The probability that the attacker can simply guess the correct solution can be computed using $h(k) = \frac{1}{k}$.

The probability that the user chooses a solution and digitally signs the transaction, when the transaction was manipulated, can be computed using the function $f(k) = \max\big(1 - \delta(k); h(k)\big)$. In order to increase the security of the system we need to minimize $f(k)$ for all *k* (min $\bigcap_k$ f(k)).

Figure 3 displays the curves of the functions $h(k)$ (—o—), $1 - \varphi(k)$ (-△-), and $f(k)$ (·+··). $f(k)$ reaches at $k = 4$ its absolute minimum with $f(4) = 0.25$ at the point of itersection of $h(k)$ and $\varphi(k)$. The curve $f(k)$ (·+··) displays depending on different values of *k* the probability that the correct solution is displayed – if the transaction was manipulated.

By combining the probability that an attacker can determine the correct solution (21.02%), and the probability that the user answers the question (25%) we get an overall probability of 5.93% ($0.25 * 0.2102 = 0.0593$) that the attacker will successfully attack the system.

D. Limitations

Presumably the attacker controls multiple accounts to make it more difficult to track him and to lower his damage, if an account gets closed by authorities. This does not directly affect the security of our proposed system since the attacker has to choose which account he uses before the question is generated. To increase her chances to successfully attack she could choose an account number that is *similar* to the user’s account number. In other words it has multiple equal numbers at different digits (e.g. if 4 out of 18 numbers are equal there is a chance of $^4/_{18} = 0.\bar{2}$ that these digits are chosen).

Our proposed system only protects the payee’s account number. Thus, if the attacker could lure the victim into transferring a small amount of money to an account she controls she could change this small amount into a much bigger amount. However, in the wild we see that if the attacker convinced the user, using sophisticated Social Engineering attacks (e.g. [5]), to transfer money to an account she controls the user would also transfer larger amounts. In this scenario the attacker manipulates the website in a way that the user thinks the he falsely received money from a “credible” source (e.g. the customs office). These manipulations go as far as manipulating the account balance, transaction history, and ‘disabling’ all functions of the website. The attacker prompts that the user has to transfer the money back to an account of the “credible” source in order to be able to use his online banking system again.

Our proposed system doesn’t protect against this attack vector because the user intentionally transfers the money since he was tricked into doing so. Also in this scenario the attacker doesn’t has to manipulate the transaction amount because the user transfers the money intentionally.

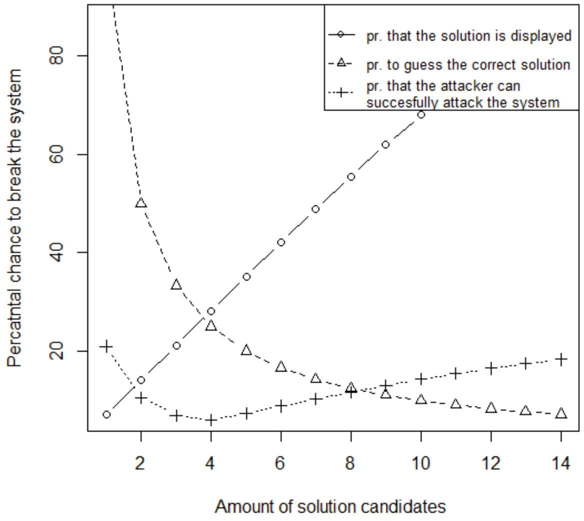
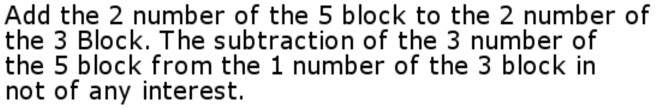


Figure 3 – pr. to successfully attack the system.

VII. User Study

Besides the security inspection we also conducted a user study of the proposed system. Therefore, we implemented a prototypical online banking environment that uses the proposed concept. 30 experimental subjects (15 men and 15 women) participated in our user study. The subject group consisted of three different age brackets: 16-29 (young), 30-49 (mid), and 50-69 (old). Each group consisted of 10 different subjects. The mean aged of our subjects is 38.68 years. When choosing the subjects we made sure that they are familiar with online banking. 20 out of 30 subjects stated that they use online banking at least once a week. 80% of the subjects hold at least a high school diploma. During the study we used two different scenarios (1) *not* manipulated transactions and (2) manipulated transactions, meaning that the correct solution was not displayed. The aim was to check if the subjects would notice the manipulation. Only 15 of the subject where informed that manipulations might take place during the experiment and how the questions might help to prevent fraud. The questions where displayed in two different ways: as simple black continuous text (“black” – see Figure 4); and as distorted text in different colors (”color” – see Figure 5). Note that the account number was divided into blocks to make it more readable for the subjects. Each subject had to solve two questions of each type. One question related to a manipulated transaction and the other transaction to a normal transaction. The order of all four questions was randomized for each subject.



Add the 2 number of the 5 block to the 2 number of the 3 Block. The subtraction of the 3 number of the 5 block from the 1 number of the 3 block in not of any interest.

Figure 4 – simple black continuous text

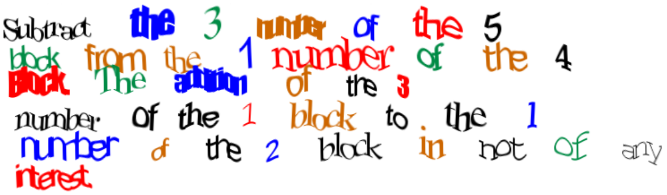

Subtract the 3 number of the 5
block from the 1 number of the 4
block. The addition of the 3
number of the 1 block to the 1
number of the 2 block in not of any
interest.

Figure 5 – Distorted question

One used metric was to determine how many subjects could solve the question on their first try without any help. Figure 6 shows that 80% of the subjects could solve the "black" question on their first try. Older people had bigger problems solving the question than younger people.

The different scenarios have a significant impact on the time the subjects needed to complete or abort a transaction. The results are listed in Table 1. All timings are the timings for the whole transaction process, starting from entering the transaction details.

	black	color
not manipulated	30s	37s
manipulated	48s	52s

Table 1 – Timings of the two scenarios

To measure the subjective stress of the subjects we used the NASA-TLX Score [20]. The NASA-TLX Score is a widely used, multidimensional scale which measures the workload of human agents performing a given task. After determining the NASA-TLX score we found that the colored question was rated subjectively to be mentally more challenging than the black question (black question: 4.20 vs. colored question: 5.25).

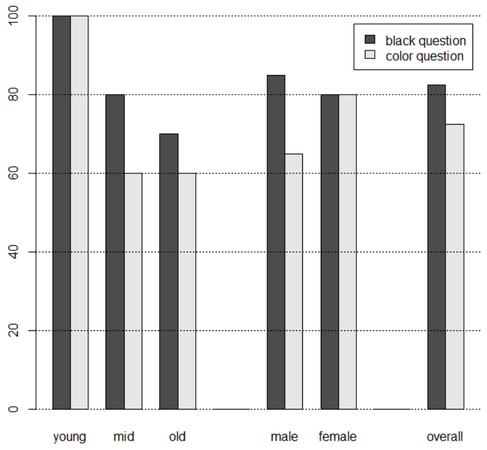


Figure 6 – Percentile amount of subjects that solved the question on their first try

VIII. Discussion

Finally, 73% of the subjects could imagine completing every online banking transaction with a CAPTCHA like question. However, nearly all subjects wished for more information about the associated increase in security. As a main disadvantage of the question that was mentioned by the subjects is the time requirement. In this specific user study 70% of the subjects criticize the wording of the question as overly complicated. Furthermore, subjects complained about the poor legibility of the distorted text in different colors which leads to unrecognized characters. Consequently, 90% of the subjects preferred the black question since it was seen as more user-friendly.

Generally, this user study showed that further development should focus on the simplification of the question and to inform user information about recognizing a manipulation and how to deal with one. To the latter, it would help to implement a "right answer is not available"-button since users have inhibitions to cancel a transaction. Another advantage would be that users do not necessarily have to understand that a phishing attack took place but just realize that the right answer is not available and the system itself would take care of everything else. Implementing a "right answer is not available"-button seems to be useful in two ways. On the one hand, it demonstrates users the opportunity that the right answer might be not available which then eliminates the inhibition to press a "cancel"-button. On the other hand, it bypasses the problem that users more carefully read the answers than explanatory text.

Apart from recognizing and dealing with manipulation further development should focus on the simplification of the question. As mentioned, users find the distorted text in different colors extremely complicated and very difficult to read. Moreover, because of the loud colors and distortion it appears dubious to users and provokes erroneous input. Furthermore, it contradicts the principle of accessibility for people with defective vision such as color-blindness. Consequently, a strong majority wishes for simple black text. Apart from the distorted color design, user criticize that two of the three sentences where in their view needless and therefore distracting. Furthermore, it was difficult to subtract, especially for younger users. So users wish for an easier wording and a summation instead of a subtraction.

IX. Future Work

The security of the proposed system mostly relies on questions that cannot be answered by computer programs but intuitively by humans. Thus, questions should be selected that are hard for computer to parse and interpret. Due to this, questions that are based on the *Winogard Shema* [17] or the *Pronoun Disambiguation Problem* (PDP) should be used. Winogard questions are multiple choice questions that consist of two sentences that contain an ambiguity that is resolved in opposite ways. Example (borrowed from [16]):

*The trophy doesn't fit in the brown suitcase because it's too [**big/small**]. What is too [**big/small**]? Answer 0: the trophy - Answer 1: the suitcase.*

In this case the word *big* refers to the trophy but the word *small* refers to the suitcase. In order to answer such questions one needs to do some common sense reasoning and needs to have some "world knowledge". Winogard questions have been proposed by Levesque in [16] as an alternative to the Turing test. Answering Winogard questions is up till now an open and unsolved AI problem [21–23].

PDPs resemble Winogard questions but are more general. They consist of at least one pronoun which has at least one ambiguous relationship in the sentence. Example (borrowed from [24]):

Do you suppose that Peter is responsible for the captain's illness? Maybe **he** *bribed the cook to put something in his food. Snippet: "***he** *bribed the cook" Answers: (a) Peter (b) the captain*

To improve the systems' security the used questions should follow the Winogard or PDP schema. Obviously these question still need to reference the transaction.

Also the usability of the system should be improved based on the results and feedback of the user study. The improved interface should be tested for suitability for daily use in a longitudinal field study. This study would also investigate the ability of users to answer the improved questions which follow the Winogard schema.

A. Generating more complex questions

Automated crafting of Winogard or PDP questions is crucial for system's security. However the generation of these questions that include arithmetical operations is not straight forward. Example 1 gives a simple example how such questions could look like. Basically a Winogard question is extended by the two tasks and the answer is determined by the answer to the Winogard question. This mechanism can be used for all Winogard question. Thus, there is a broad variety of different questions that are hard to parse for computers.

Example 1: `Frank and Bill took part in an algebra competition.`

`Frank had to compete` $a+b$ `and felt [`**`vindicated/crushed`**`] when his longtime rival Bill, who had to compute` $c+d$`, revealed that` **`he`** `was the winner of the competition.`

<u>Question:</u> `What is the solution computed by the winner of the competition?`

An example how to alter a PDP that is given in Example 2. Example 2 also gives an idea how this type of questions might be used in different areas.

Example 2: `Babar wonders how` **`he`** `can buy new clothing and get them delivered to A. Luckily, a very rich old man understands right away that` **`he`** `is longing for a fine suit. As he likes to make people happy, he buys a new suit but mistakenly delivers it to B.`

<u>Question:</u> `Should your order be delivered to the address the person used [`**`who likes to make people happy / is longing for a fine suit`**`]?`

X. Related Work

Breaking CAPTCHAs. The idea of using CAPTCHAs in an online banking environment is not new. In the past a transaction number was placed in a picture, the CAPTCHA, and sent to the client. These pictures also contain the transaction data and sometimes some personal information about the user. The pictures have been distorted so machines cannot read them, the basic idea of CAPTCHAs. Successful attacks on such systems have been presented [25]. Li et al. use image recognition algorithms in order identify the parts of the image that relate to the transaction and manipulate these parts. Our approach bypasses this attack since the attacker has to understand the question in order to manipulate it. Also manipulating the question is pointless since the answer, the server expects, doesn't change.

In [26] Shirali-Shahreza et al. examine the use of questions in the CAPTCHA context. Thereby, they used a combination of pictures and text. The intention was to prevent the CAPTCHA from pure letter recognition. The difference between these questions and our approach is that they are not in any relation to the context they are used in.

There is plenty of work focusing on breaking different CAPTCHAs (e.g. [18, 27, 28], [29]). All these papers focus on detecting and identifying objects and letters within different types of CAPTCHAs. Most of the approaches have very good detection rates and manage to break most of the examined CAPTCHAs. These approaches are not a problem for our proposed system since the challenge is understanding the question rather than detecting the letters.

Winogard questions. Levesque examines in [30] the generation of questions that cannot be understood and answered by state of the art computer programs. The author uses Winogard questions [17] as example how hard it is for computer programs to answer such questions.

The automated answering of questions by computer programs is an active area of research. To answer general questions most approaches use huge knowledge databases [31, 32] or results from search engines [33]. The proposed approach doesn't allow to use huge information sources ("big data") to answer the questions since there are no public information on the web that will help to understand and answer them.

Phishing attacks. Mechanisms to avoid password phishing by shoulder surfing attackers is an active field of research. In [34] Han Yan et al. present a system that allows users to enter passwords - leakage resilient - on mobile devices. Therefore, they present a hidden messages to the user in order to break the correlation between the password and the interaction observable to an adversary. The concept of providing hidden information to the user, in order to prevent password leakage, was introduced in [35] for pressure sensitive screens. Both approaches present the hidden information on the screen of the device. This information has to be shielded by the user so that the adversary cannot see it.

Limitations and design principles for systems that do not use a second channel to communicate a secret between two parties are discussed by Qiang Yan et al. in [36]. Qiang Yan et al. describe the tradeoffs between security and usability, for their developed framework, and conclude that either a high memory demand or high cognitive workload are unavoidable for users. Their work also focuses on leakage-resilient password systems.

XI. Conclusion

Our presented approach allows to secure a vast majority (>94%) of online bank wire transfers even if they are done on an infected system. Thus, we reduce the potential financial gains of the attackers hugely which will ultimately lower their motivation to attack the system.

As a main result of the user study it can be stated that a strong majority successfully solved the CAPTCHA and is willing to use them in order to increase the security of online banking transactions. Further improvements should address the simplification of the CAPTCHA and the user information about it. As to the first one, users strongly prefer simple black CAPTCHA instead of distorted colored ones and an easy wording as well as summation instead of a subtraction. As to the latter one, it would be help full to provide additional information and to implement a “right answer is not available”-button since users have inhibitions to cancel a transaction.

Overall our system increases the security of online transaction but our prototype lacks of usability and needs further improvement.

XII. Acknowledgments

This work was supported by the German Federal Ministry of Education and Research under grand 13N13252 and 13N13251 (“BOB”). We want to thank A. Brunstein, J. Conradi, N. Döllinger, E. Hod, M. Makaranets, C. Pawelke, S. Uhrig, M. Wenig und R. Zechner for conducting the user study. We would like to thank R. Widdermann for his efforts developing the online banking environment. We would like to thank the anonymous reviewers for their helpful and constructive comments that greatly contributed to improving the final version of the paper. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the sponsors or supporters.

References

[1] 1. Eurostat, the statistical office of the European Union: Individuals using the internet for internet banking, http://ec.europa.eu/eurostat/tgm/table.do?tab=table&init=1&language=en&pcode=tin00099

[2] 2. Mäntymäki, M., Salo, J.: Why do teens spend real money in virtual worlds? A consumption values and developmental psychology perspective on virtual consumption. International Journal of Information Management 35, 124–134 (2015)

[3] 3. Federal Criminal Police Office (Germany): Bundeslagebild Cybercrime 2014 2014 (2014)

[4] 4. Lloyds Bank: Our Fraud Guarantee, https://www.lloydsbank.com/security.asp?WT.ac=OBOSFOM#tab-row-3

[5] 5. Golovanov, S., Makrushin, D. and Monastyrsky, A.: Staying safe from virtual robbers, https://securelist.com/analysis/user-advice/58328/staying-safe-from-virtual-robbers/

[6] 6. Initiative D21: Online - Online Banking 2014 Sicherheit zählt! (GERMAN) (2014)

[7] 7. Commerzbank AG: mobileTAN: Tried and tested, https://www.commerzbank.de/portal/en/englisch/products-offers/services/secure-internet-banking/banking.html

[8] 8. Dougan, T., Curran, K.: Man in the Browser Attacks. International Journal of Ambient Computing and Intelligence 4, 29–39 (2012)

[9] 9. Ahn, L. von, Blum, M., Hopper, N.J., Langford, J.: CAPTCHA: Using Hard AI Problems for Security. In: Biham, E. (ed.) Advances in Cryptology - EUROCRYPT 2003. International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003 Proceedings, 2656, pp. 294–311. Springer, Berlin, Heidelberg (2003)

[10] 10. Abraham, S., Chengalur-Smith, I.: An overview of social engineering malware. Trends, tactics, and implications. Technology in Society 32, 183–196 (2010)

[11] 11. Australia and New Zealand Banking Group: ANZ security measures, http://www.anz.co.nz/personal/ways-bank/protect-banking/security-measures/

[12] 12. Raiffeisen e-force GmbH: The cardTAN, http://www.raiffeisen.at/oesterreich/1171457369322123440_1171457740299924723-677938155780761544-NA-30-NA.html

[13] 13. DBS Bank Singapore: DBS iBanking Secure Device, http://www.dbs.com.sg/personal/ibanking/faq/newtoken.page

[14] 14. Yubico: YubiKey. Trust the Net with YubiKey Strong Two-Factor Authentication, https://www.yubico.com/

[15] 15. 2Captcha: Human-powered CAPTCHA-solving service, https://2captcha.com/

[16] 16. Levesque, H.J., Davis, E., Morgenstern, L.: The Winograd schema challenge. In: AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning, 46, p. 47 (2011)

[17] 17. Winograd, T.: Understanding natural language. Cognitive Psychology 3, 1–191 (1972)

[18] 18. Bursztein, E., Martin, M., Mitchell, J.: Text-based CAPTCHA strengths and weaknesses. In: Chen, Y., Danezis, G., Shmatikov, V. (eds.) the 18th ACM conference, p. 125

[19] 19. The Society for Worldwide Interbank Financial Telecommunication: IBAN REGISTRY (ISO 13616). This registry provides detailed information about all ISO 13616 - compliant national IBAN formats (2016)

[20] 20. Hart, S.G., Staveland, L.E.: Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research. In: Meshkati, N., Hancock, P.A. (eds.) Human Mental Workload, 52, pp. 139–183. Elsevier textbooks, s.l. (1988)

[21] 21. Peter Schüller: Tackling Winograd Schemas by Formalizing Relevance Theory in Knowledge Graphs, http://www.aaai.org/ocs/index.php/KR/KR14/paper/view/7958 (2014)

[22] 22. Arpit Sharma, Vo, N.H., Somak Aditya, Chitta Baral: Towards addressing the winograd schema challenge - Building and using a semantic parser and a knowledge hunting module. In: IJCAI International Joint Conference on Artificial Intelligence, vol. 2015-Januaryvol. , pp. 1319–1325. International Joint Conferences on Artificial Intelligence (2015)

[23] 23. Rahman, A., Ng, V.: Resolving Complex Cases of Definite Pronouns: The Winograd Schema Challenge. In: Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 777–789. Association for Computational Linguistics, Stroudsburg, PA, USA (2012)

[24] 24. Morgenstern, L.: Commonsense Reasoning. Sample Pronoun Disambiguation Problems, http://commonsensereasoning.org/disambiguation.html

[25] 25. Li, S., Shah, S.A.H., Khan, M.A.U., Khayam, S.A., Sadeghi, A.-R., Schmitz, R.: Breaking e-banking CAPTCHAs. In: Gates, C., Franz, M., McDermott, J. (eds.) the 26th Annual Computer Security Applications Conference, p. 171

[26] 26. Shirali-Shahreza, M., Shirali-Shahreza, S.: Question-Based CAPTCHA. In: International Conference on Computational Intelligence and Multimedia Applications (ICCIMA 2007), pp. 54–58

[27] 27. Suphannee Sivakorn, Iasonas Polakis, Angelos D. Keromytis: I Am Robot: (Deep) Learning to Break Semantic Image CAPTCHAs. In: Proceedings of the 1st IEEE European Symposium on Security and Privacy (2016)

[28] 28. El Ahmad, A.S., Yan, J., Marshall, L.: The robustness of a new CAPTCHA. In: Costa, M., Kirda, E. (eds.) the Third European Workshop, pp. 36–41

[29] 29. Yan, J., Ahmad, A.S.E.: Breaking Visual CAPTCHAs with Naive Pattern Recognition Algorithms. In: Twenty-Third Annual Computer Security Applications Conference (ACSAC 2007), pp. 279–291

[30] 30. Levesque, H.J.: On our best behaviour. Artificial Intelligence 212, 27–35 (2014)

[31] 31. Qingqing Cai, Alexander Yates: Large-scale Semantic Parsing via Schema Matching and Lexicon Extension. In: In Proceedings of the Annual Meeting of the Association for Computational Linguistics (2013)

[32] 32. Fader, A., Zettlemoyer, L., Etzioni, O.: Open question answering over curated and extracted knowledge bases. In: Macskassy, S., Perlich, C., Leskovec, J., Wang, W., Ghani, R. (eds.) the 20th ACM SIGKDD international conference, pp. 1156–1165

[33] 33. Kwok, C., Etzioni, O., Weld, D.S.: Scaling question answering to the web. ACM Trans. Inf. Syst. 19, 242–262 (2001)

[34] 34. Yan, Q., Han, J., Li, Y., Zhou, J., Deng, R.H.: Designing leakage-resilient password entry on touchscreen mobile devices. In: Chen, K., Xie, Q., Qiu, W., Li, N., Tzeng, W.-G. (eds.) the 8th ACM SIGSAC symposium, p. 37

[35] 35. Kim, D., Dunphy, P., Briggs, P., Hook, J., Nicholson, J., Nicholson, J., Olivier, P.: Multi-touch authentication on tabletops. In: Mynatt, E., Schoner, D., Fitzpatrick, G., Hudson, S., Edwards, K., Rodden, T. (eds.) the 28th international conference, p. 1093

[36] 36. Yan, Q., Han, J., Li, Y., Huijie, R.D.: On Limitations of Designing Usable Leakage-Resilient Password Systems: Attacks, Principles and Usability. 19th Network and Distributed System Security Symposium (NDSS) (2012)