Neue Entwicklungen in der e-Banking Security

Roland Schmitz
Hochschule der Medien Stuttgart
schmitz@hdm-stuttgart.de

Shujun Li
Universität Konstanz
shujun.li@uni-konstanz.de

Abstract.

Wir zeigen, dass die in heutigen e-Banking Systemen eingesetzten CAPTCHAs unsicher sind, d.h. von einem Programm ausreichend schnell gelöst werden können, um praktische Attacken zu ermöglichen. Auch die von den Banken in letzter Zeit häufig propagierte Lösung mTAN (mTAN) ist aus unserer Sicht problematisch. Wir plädieren deshalb für den Einsatz kostengünstiger, Hardware-basierter Token, die transaktionsabhängige TANs erzeugen können und stellen einen Prototyp vor, der selbst auf einer nicht vertrauenswürdigen PC-Plattform sicher ist.

1 Einführung

Zurzeit nutzen ca. 27 Millionen Deutsche e-Banking zur Abwicklung Ihrer Bankgeschäfte, viele von ihnen sicherlich mit einem gewissen Unbehagen. Immer wieder liest man Berichte über gelungene Angriffe in den Medien, über konkrete Schadenssummen schweigen sich die betroffenen Banken jedoch zumeist aus. Für sie bedeutet e-Banking ein gutes Geschäft, lassen sich doch durch den elektronischen Zahlungsverkehr Personal- und Infrastrukturkosten einsparen. Umso wichtiger ist es, dass beim Kunden Vertrauen in die Sicherheit des Verfahrens besteht und die eingesetzten Sicherheitsmechanismen auf dem neuesten Stand sind.

In allen uns bekannten Systemen authentifiziert sich der Nutzer beim e-Banking Server mit Hilfe einer NutzerID und eines Passworts, welches häufig in einer Zahl (PIN) besteht. Hinzu kommt eine zusätzliche Transaktionsnummer (TAN) zur Authorisierung einer Transaktion. Die Herausforderung besteht darin, sicherzustellen, dass diese Credentials auch wirklich beim e-Banking Server landen und nicht bei einem Angreifer, der sich als Server ausgibt (Man-in-the-Middle, abgekürzt MitM). In einer 2010 erschienenen, vom BSI in Auftrag gegebenen Studie zum Identitätsdiebstahl im Internet [1] werden die Techniken für Man-in-the-Middle Angriffe beschrieben, von denen wir hier die wesentlichsten kurz zusammenfassen:

- **Klassische Phishing-Attacken**:

  Bei klassischen Phishing-Attacken erhalten die Opfer per Mail unter einem Vorwand die Aufforderung, sich bei Ihrer Bank einzuloggen. Die Mails enthalten dazu einen Link, der jedoch nicht zu der Bank des Opfers führt, sondern zum Server des Angreifers. Diese Form des Angriffs kommt jedoch kaum noch vor (s. [2] S.23), unter anderem wohl deshalb, da sie von den Nutzern zumeist leicht erkannt werden kann. In heutiger Zeit werden deshalb fortgeschrittene Angriffstechniken verwendet, die vom Opfer deutlich schwieriger zu erkennen sind.

- **Malware-basierte Attacken**:

  Diese Angriffe nutzen zumeist den Browser des Opfers als Einfallstor. Sehr häufig erfolgt die initiale Infektion durch den Besuch präparierter Webseiten per *Drive-by Download*. Weiterhin sind bösartige Browsererweiterungen in der Lage, die Eingaben in bestimmte Browserfenster wie „Nutzername“ „PIN“ oder „TAN“ mitzuprotokollieren, die Verbindung zum Bankenserver zu unterbrechen und dann an einen Angreifer zu versenden. Auch das Flash-Player-Plug-In der Firma Adobe bietet potentiellen Angreifern eine leichte Möglichkeit, beispielsweise mittels bösartig präparierter Flash-Applets in Webseiten ein Endsystem mit Malware zu infizieren. Andere Schadsoftware wiederum ist in der Lage, den DNS-Cache des Opfers zu manipulieren und Daten somit zu einer anderen IP-Adresse zu leiten, als die zu der vom

Opfer gewünschten URL gehörige (sog. *Pharming* oder *Cache-Poisoning*). Ist der Angreifer in der Lage, die Web-Präsenz des Banking-Servers originalgetreu nachzubilden, und zusätzlich die im Browser installierten Root-Zertifikate zu ersetzen, ist der Angriff auch für fortgeschrittene Nutzer kaum noch zu erkennen.

- **Cross-Site-Scripting Attacken**:

  Hierbei muss das Opfer dazu gebracht werden, auf einen präparierten Link zu klicken, der zwar tatsächlich zum Server der Bank führt, gleichzeitig aber dafür sorgt, dass Script-Code vom Server des Angreifers geladen wird und im Browser-Kontext des Opfers ausgeführt wird. So lassen sich beispielsweise Cookies, die den Authentisierungsstatus des Opfers halten, vom Opfer zum Angreifer übertragen. Ist das Opfer zur Zeit des Angriffs angemeldet, kann sich auch der Angreifer mit Hilfe des Cookies anmelden.

Um diesen MitM-Angriffen entgegen zu wirken, sind von den Banken etliche Verbesserungen des ursprünglichen PIN/TAN Verfahrens entwickelt worden, angefangen beim iTAN-Verfahren, bei dem der Nutzer TANs mit einem bestimmten Index aus einer Liste auswählt, über den Einsatz von CAPTCHAs zum Schutz des Login vor automatisierten Attacken und zur Transaktionsverifikation bis hin zum Versand transaktionsabhängiger TANs per SMS (mTAN). Forscher der Ruhr-Uni Bochum haben jedoch bereits 2005, kurz nach Einführung des iTAN-Verfahrens, demonstriert, dass es keinen Schutz gegen einen Man-in-the-Middle bietet, der in Echtzeit agiert. Immerhin wird der Angreifer durch das iTAN-Verfahren gezwungen, in Echtzeit zu agieren. Um dies zu erschweren bzw. unmöglich zu machen, wurden von einigen Banken Transaktions-CAPTCHAs, bei denen die Nummer der einzugebenden TAN aus einem CAPTCHA ausgelesen werden muss, eingeführt.

Im vorliegenden Bericht soll über neue Forschungsergebnisse im Bereich CAPTCHAs berichtet werden, die deren Einsatz im e-Banking Umfeld als nicht mehr zeitgemäß erscheinen lassen (s. Abschnitt 2). Auch die von vielen Banken favorisierte Alternativtechnologie mTAN ist aus unserer Sicht unsicher, wie wir in Abschnitt 3 darlegen. Wir plädieren deshalb für den Einsatz kostengünstiger, Hardware-basierter TAN Generatoren im e-Banking. Zwar sind solche Generatoren vielfach bereits im Markt, können jedoch häufig etwa bei einem mit einem Keylogger verseuchten PC keinen ausreichenden Schutz bieten. Nur wesentlich teurere, mit einer Tatstatur ausgestattete Hardware-Token sind dazu bislang in der Lage. In Abschnitt 4 stellen wir einen kostengünstigen Prototyp vor, der ohne Tatstatur auskommt und selbst bei einem komplett nicht vertrauenswürdigen PC, wie man ihn etwa in einem Internet-Cafe vorfindet, ein hohes Sicherheitsniveau bietet.

2 e-Banking CAPTCHAs sind unsicher[1]

Das Ziel eines CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) ist es, über ein für Menschen einfach lösbares Problem, das für eine Maschine jedoch unmöglich zu lösen ist, menschliche Nutzer zu identifizieren und nur diese zur Nutzung eines Systems zuzulassen.

CAPTCHAs werden beim e-Banking vor allem für zwei Zwecke eingesetzt: Zur Abwehr automatischer Login-Skripte (login-CAPTCHAs), die alle möglichen Passwörter bzw. PINs durchprobieren, sowie zur Transaktionsauthentisierung (Transaktions-CAPTCHAs). Bei einer CAPTCHA-basierten Transaktionsverifikation erzeugt der Server, nachdem er die Transaktionsdaten vom Client erhalten hat, ein CAPTCHA, in dem die Transaktionsdaten, die Nummer einer gewünschten TAN und weitere Informationen enthalten sind, mit deren Hilfe sich der Server gegenüber dem Nutzer authentifiziert. Der Nutzer prüft die in dem CAPTCHA enthaltenen Informationen und bestätigt die Transaktion durch Eingabe der gewünschten TAN. Die Sicherheit beruht in beiden Anwendungsszenarien auf der Annahme, dass ein automatisierter Man-in-Middle nicht in der Lage ist, die im CAPTCHA enthaltenen Informationen auszulesen.

Der Vorteil von CAPTCHA-basierten Lösungen besteht darin, dass sie nicht auf spezieller Hardware beruhen und die Implementations- und Wartungskosten für die Banken daher minimal sind.

[1] Dieses Kapitel basiert in wesentlichen Teilen auf dem Paper „Breaking e-Banking CAPTCHAs“ [3], © ACM 2010. Insbesondere sind alle Abbildungen in diesem Kapitel aus [3] übernommen.

2.1 Tools

Im Folgenden stellen wir kurz die wichtigsten der zum Brechen von e-Banking CAPTCHAs verwendeten Image Processing Algorithmen vor. Details zu den erwähnten morphologischen Algorithmen findet man in [4] sowie Verweise auf weitere Spezialliteratur in [3].

- ***k*-means Layer Segmentation**:

  Der erste Schritt bei jedem Angriff auf ein CAPTCHA besteht darin, bestimmte Objekte aus dem CAPTCHA zu extrahieren. Dazu muss das CAPTCHA in mehrere Schichten segmentiert werden. Eine klassische Segmentierungsmethode ist das *k-means clustering*: Die *k* Segmente bzw. Layer (in unserem Fall binäre Bilder) sind jeweils um einen Mittelpunkt, den Centroid, herum gruppiert. Dabei werden, ausgehend von einer zufälligen Anfangsauswahl der *k* Centroide die Pixel den Layern in jedem Schritt so zugeordnet, dass der durchschnittliche Abstand aller Punkte zum jeweils nächsten Centroid minimal wird. Im nächsten Schritt werden die Centroide als Durchschnittswerte jedes Layers neu berechnet und die Schritte wiederholen sich, bis die Layer stabil sind.

- **Morphological Image Processing**:

  Mathematische Morphologie ist die Theorie der Analyse und Verarbeitung geometrischer Strukturen. Dabei wird ein Bild durch die Anwendung morphologischer Operationen auf das Vorhandensein gewisser vordefinierter Strukturelemente geprüft. Typische morphologische Operationen sind Dilation, Erosion, Opening und Closing. Mit ihrer Hilfe kann Rauschen ausgefiltert und die Form gefundener Objekte weiter verfeinert werden.

- **Line Detection:**

  Einige e-Banking CAPTCHAs enthalten zufällige Linien, um die Segmentation zu erschweren. Um das CAPTCHA zu brechen, werden diese Linien detektiert und entfernt. Die Kantendetektion ist eine klassische Aufgabe der Bildanalyse. Typischerweise können sie mit Hilfe der Hough-Transformation gefunden werden. Bei e-Banking CAPTCHAs ziehen sich die Linien durch das gesamte Bild und haben nur zwei Orientierungen (horizontal oder vertikal), was die Detektion erleichtert.

- **Digital Image Inpainting**:

  Mit dieser Technik werden fehlende Teile in einem digitalen Bild ergänzt, indem die Grauwerte fehlender Pixel über ihre Nachbarn geschätzt werden. Bei den Angriffen auf Transaktions-CAPTCHAs wurden zunächst die echten Transaktions-Daten detektiert und entfernt, um danach gefälschte Transaktionsdaten via Image Inpainting einzusetzen.

- **Character Segmentation:**

  Das Endziel einer Attacke auf ein CAPTCHA ist die Erkennung der darin enthaltenen Zeichen. Als Vorstufe müssen zunächst die enthaltenen Zeichen segmentiert, also voneinander getrennt werden. Durch das oben erklärte *k*-means clustering erhält man zunächst ein Layer mit allen Zeichen. Die Zeichen können dann voneinander isoliert werden als separate, zusammenhängende Objekte. Enthält ein zusammenhängendes Objekt mehr als ein Zeichen, können diese voneinander getrennt werden, wenn sie verschiedene Farben besitzen. Um die Genauigkeit des Segmentierungsprozesses weiter zu erhöhen, können verschiedene morphologische Operationen eingesetzt werden.

- **Character Recognition**:

  Nachdem ein Zeichen isoliert wurde, kann es automatisch erkannt werden. Bei den Attacken auf Transaktions-CAPTCHAs haben wir zwei Erkennungsmethoden benutzt, die beide ohne eine Trainingsphase auskommen. Beide Methoden vergleichen das Input-Zeichen mit einer Anzahl von Referenz-Bildern (sog. Templates) und selektieren das Template mit der besten Übereinstimmung.

2.2 Brechen von Transaktions-CAPTCHAs

Die folgende Diskussion beschränkt sich auf in Deutschland eingesetzte Transaktions-CAPTCHAs; in [3] wurden außerdem auch Transaktions-CAPTCHAs von zwei chinesischen Banken gebrochen. Die Beispielabbildungen sind dergestalt anonymisiert, dass direkte Hinweise (Firmenlogos etc.) auf die betreffende Bank entfernt wurden (s. Abb. 1).

Abb. 1: Anonymisiertes Transaktions-CAPTCHA (GeCAPTCHA)

In diesem deutschen Transaktions-CAPTCHA (im Folgenden als GeCAPTCHA bezeichnet) fungiert der Geburtstag des Nutzers als gemeinsames Geheimnis zwischen Nutzer und Bankenserver. Somit kann der Nutzer die Identität des Bankenserver authentifizieren. Um eine automatische Erkennung zu erschweren, werden auf die Ziffern des Geburtstags folgende Operationen angewandt:

- jede Ziffer ist um einen bestimmten Zufallswinkel gedreht
- der Font jeder Ziffer ist zufällig gewählt
- die Position der Ziffern ist zufällig
- alle Ziffern sind zwischen den Transaktionsdaten und einem zufälligen Gitter platziert

Die Transaktionsdaten befinden sich oberhalb der anderen Layer, um eine Manipulation ohne eine merkliche Beeinflussung der restlichen Bilddaten zu erschweren. Um eine erfolgreiche MitM-Attacke auf ein GeCAPTCHA durchzuführen muss der Angreifer bzw. die Malware die Transaktionsdaten in Echtzeit manipulieren können, ohne dass dies vom Nutzer bemerkt wird. Angenommen, der Nutzer schickt seine gewünschten Transaktionsdaten TDU zum Server. Die Daten werden vom MitM abgefangen, manipuliert und als TDA zum Bankenserver weiter gesandt. Der MitM erhält vom Server ein GeCAPTCHA zurück, welches TDA enthält. Um den Nutzer zu täuschen und zu veranlassen, die richtige TAN anzugeben, muss der MiTM dem Nutzer ein GeCAPTCHA präsentieren, das den Geburtstag und die ursprünglichen Transaktionsdaten TDU enthält. Dazu gibt es zwei grundsätzliche Möglichkeiten:

- **Manipulation des bestehenden CAPCHA (Manipulation Attack):**

  Dabei werden die Transaktionsdaten TDA im CAPTCHA detektiert, entfernt und durch die Daten TDU ersetzt, oder

- **Erzeugung eines neuen CAPTCHA (Generation Attack)**:

  Zunächst wird der Geburtstag des Nutzers detektiert und damit ein neues CAPTCHA mit TDU erzeugt

Bei beiden Ansätzen müssen zunächst die verschiedenen Objekte im CAPTCHA (Transaktionsdaten, Geburtstag des Nutzers und der geforderte TAN Index) voneinander isoliert werden. Da die Anzahl der Layer ($k=3$) in GeCAPTCHA bekannt ist, kann hierfür die *k*-means Layer Segmentation Technik angewandt werden. Darauf aufbauend, können beide Ansätze in automatischen Attacken umgesetzt werden. Abb. 2 zeigt die drei Layer des GeCAPTCHA: Gitterebene, Textebene und Geburtstagsebene.

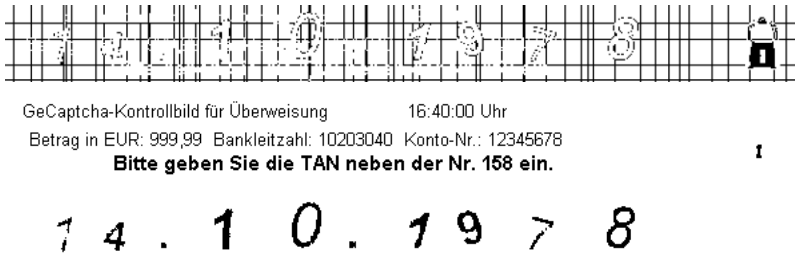


Abb. 2: Die drei Layer des GeCAPTCHA

2.2.1 Manipulation des bestehenden CAPCHA (Manipulation Attack)

Zunächst müssen die Transaktionsdaten innerhalb des Text Layers gefunden werden. Der Text Layer enthält drei Zeilen, die die Transaktionsdaten, den TAN Index und die Zeit der Transaktion enthalten. Die Reihenfolge der Zeilen variiert, so dass die Transaktionsdaten auf andere Weise gefunden werden müssen. Beispielsweise könnte mit einem handelsüblichen OCR Tool nach den Schlüsselwörtern „Betrag in EUR“, „Bankleitzahl“ und „Konto-Nr“ gesucht werden. Es geht aber sogar noch einfacher: Die Zeile, die den TAN Index enthält, ist immer fett gesetzt, während die Zeile mit der Zeit weniger Zeichen als die anderen Zeilen und dafür eine große weiße Fläche enthält. Dadurch unterscheidet sich das sogenannte *Average Font Weight* (AFW) der drei Zeilen beträchtlich voneinander und kann dazu benutzt werden, die Zeile mit den Transaktionsdaten zu identifizieren.

Nachdem die Zeile mit den Transaktionsdaten gefunden ist, kann sie mit Hilfe von Digital Image Inpainting Methoden entfernt werden. Dabei müssen Pixel auf den Gitterlinien separat behandelt werden, d.h. ihr Wert sollte nur anhand der nächsten Pixel auf dem Gitter geschätzt werden, ansonsten kann es zu sichtbaren Verzerrungen in der Nähe der Gitterlinien kommen. Die Lage der Gitterlinien kann mit Hilfe der Hough-Transform bestimmt werden. Das Ergebnis nach dem Entfernen der Transaktionsdaten zeigt Abbildung 3.

Abb. 3: GeCAPTCHA ohne Transaktionsdaten

Der letzte Schritt, das Einfügen der vom Nutzer erwarteten Transaktionsdaten TDU in das CAPTCHA, ist nun trivial (s. Abb. 4). Man beachte die gegenüber Abb. 1 geänderten Transaktionsdaten.

Abb. 4: GeCAPTCHA mit geänderten Transaktionsdaten

2.2.2 Erzeugung eines neuen GeCAPTCHA (Generation Attack)

Die automatische Erkennung des Geburtstages des Nutzers ist der aufwändigste Schritt bei der Erzeugung eines neuen GeCAPTCHA. Er wird offline, d.h vor der eigentlichen MitM-Attacke, durchgeführt. Zunächst müssen dazu genügend GeCAPTCHAs verschiedener Nutzer gesammelt werden, um eine Datenbank mit hinreichend vielen Referenz-Zahlsymbolen zu erzeugen. Danach wird ein GeCAPTCHA vom Opfer kopiert und an den Angreifer geschickt. Aus diesem kann mit Hilfe der Datenbank der Geburtstag extrahiert werden. Zunächst wird das GeCAPTCHA segmentiert und der Birthday Layer isoliert. Darauf wird ein Wavelet-basierter Algorithmus [5] angewendet, der ursprünglich zur Bewertung der Bildqualität entwickelt wurde. Mit Hilfe von 60 Referenzsymbolen (10 Zahlzeichen in drei verschiedenen Rotationswinkeln und zwei verschiedenen Fonts) konnte eine Erkennungsrate von 91% erzielt werden, d.h. aus 9 von 10 CAPTCHAs konnte erfolgreich der Geburtstag extrahiert werden. Nachdem der Geburtstag des Opfers bekannt ist, muß im Online-Teil des Angriffs nur noch die Zeile bestimmt werden, die den geforderten TAN-Index enthält. Danach kann ein manipuliertes GeCAPTCHA mit den gewünschten Transaktionsdaten erzeugt werden (s. Abb. 5).

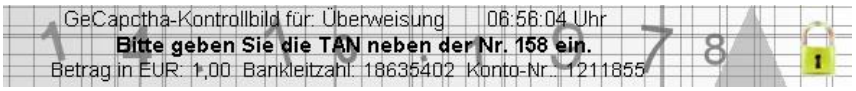


Abb. 5: Gefälschtes GeCAPTCHA

2.3 Laufzeiten

Beide Angriffsvarianten wurden jeweils für 100 GeCAPTCHAs getestet. Die durchschnittliche Laufzeit der Inpainting Attack war 250 Millisekunden, während bei der Generation Attack die Laufzeit des Online-Teils durchschnittlich 190 Millisekunden betrug. Die Laufzeit wurde gemessen ab dem Einlesen des ursprünglichen CAPTCHA von der Festplatte bis zum Speichern des veränderten bzw. neu erzeugten CAPTCHA. Das Auslesen des Geburtstags im Offline-Teil dauerte im Schnitt 5 Sekunden. Hierzu ist zu bemerken, dass alle Angriffe in

MATLAB realisiert wurden, eine interpretierte Sprache, die im Hinblick auf Geschwindigkeit sicher nicht optimal ist.

2.4 Login-CAPTCHAs

Zusätzlich zu den drei Transaktions-CAPTCHAs wurden in [3] auch 41 Login-CAPTCHAs von Banken weltweit untersucht. Zwei dieser CAPTCHAs sind auch in Deutschland weit verbreitet. Jedes Login-CAPTCHA wurde anhand von mindestens 60 Beispielen untersucht. Die Ergebnisse sind alarmierend: Alle 41 Login-CAPTCHAs können mit den oben dargestellten Methoden automatisch segmentiert werden. Darauf aufbauend, ist die automatische Erkennung der segmentierten Zeichen nicht mehr schwierig. Abb. 6 zeigt die untersuchten CAPTCHAs in Deutschland und die dazu gehörigen Segmentierungsergebnisse.

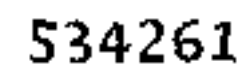
CAPTCHA	Segmentierung	Verwendete Methoden
		3-means Clustering, morphologische Operationen
		2-means Clustering, Line Detection, Image Inpainting

Abb. 6: Segmentierung von Login-CAPTCHAs

3 mTANs – ein guter Ersatz für CAPTCHAs?

Die in Kapitel 1 und 2 beschriebenen Angriffe legen den Schuss nahe, dass es für eine wirklich sichere e-Banking Lösung zweierlei Zutaten bedarf:

- transaktionsabhängige TANs: So können die TANs nicht mehr losgelöst von den Transaktionen verwendet werden, die sie autorisieren
- ein vertrauenswürdiger Offline-Kanal, auf dem die TANs vom Server zum Client übermittelt werden

Einige Banken in Deutschland unterstützen deshalb seit kurzer Zeit das iTAN-Verfahren nicht mehr. Stattdessen führen sie für Ihre online-Kunden mandatorisch das mobile TAN-Verfahren (kurz. mTAN) ein. Andere Banken bieten dieses Verfahren bereits seit längerem optional als Alternative zum iTAN-Verfahren an. Beim mTAN-Verfahren füllt der Kunde zunächst wie gewohnt einen Überweisungsträger am PC aus und sendet ihn an den Bankenserver. Der Bankenserver generiert daraufhin eine transaktionsabhängige TAN aus den Transaktionsdaten und sendet diese, zusammen mit den ursprünglichen Transaktionsdaten, per SMS zurück an den Kunden. Der vertrauenswürdige Kanal wird also in diesem Fall durch das Mobilfunknetz realisiert.

3.1 Usability-Aspekte

Die mTAN-Lösung besitzt einige auf den ersten Blick erkennbare Usability-Nachteile. So ist jeder Online-Kunde automatisch auch zur Handy-Nutzung gezwungen. Bei Verlust, Defekt bzw. Diebstahl des Handys oder auch bei mangelnder Netzabdeckung ist zunächst kein Online-Banking mehr möglich. Teilweise treten auch längere Wartezeiten auf die TAN-SMS auf, die sogar so lang sein können, dass die übermittelte TAN schon wieder ungültig geworden ist. Das Beschaffen einer neuen SIM-Karte mit der gleichen Nummer oder auch der Providerwechsel unter Mitnahme der Nummer ist zwar heutzutage kein Problem mehr, auf der anderen Seite kann aber auch genau dieses Feature für Attacken ausgenutzt werden (s. Abschnitt 3.2). Zudem fallen bei Nutzung der mTAN im Ausland erhöhte Kosten für den SMS-Versand an.

3.2 Security-Aspekte

Entscheidend für die Akzeptanz der Lösung ist natürlich in erster Linie ihre Sicherheit. Die bereits erwähnte Studie [1] schätzt das mTAN-Verfahren als sicher gegen MitM-Angreifer ein, denn:

> „*Selbst ein Online-Angreifer kann dieses Verfahren nicht brechen: Wenn er die an die Bank übermittelten Transaktionsdaten verändert, so übermittelt die Bank diese veränderten Daten per SMS*

an den Kunden. Dieser kann die Veränderung sofort erkennen und wird die mTAN demzufolge nicht an die Bank übermitteln: Die gefälschte Transaktion wird so nicht ausgeführt."

Selbst wenn man aber davon ausgeht, dass die transaktionsabhängigen TANs serverseitig mit Hilfe einer kryptografischen Hashfunktion und eines geheimen Schlüssels erzeugt und somit nicht gefälscht werden können, ist das mTAN-Verfahren in der Praxis angreifbar. So kann die Luftschnittstelle des GSM-Mobilfunksystems mittlerweile in Echtzeit belauscht werden (s. [2], S. 34). Für die Praxis relevanter erscheinen aber Angriffe, bei denen die Zuordnung eines Kunden zu seiner Mobilfunknummer unterbrochen wird. Hier lassen sich Betrüger neue SIM-Karten im Namen eines Opfers zusenden (sog. *SIM Swap Fraud*, s. [6], vgl. auch [7] zu einem ähnlich gelagerten Fall in Australien). Zwar sind solche Angriffe bislang nur sehr vereinzelt bekannt geworden, in jedem Fall sollte man aber beachten, dass diese Angriffe nur deshalb möglich sind, weil die für die Sicherheit des Gesamtsystems essenzielle Zuordnung einer Mobilnummer zu einer Person nicht durch die Banken, sondern durch den Mobilfunkbetreiber geleistet wird. Auch im Hinblick darauf, dass eine TAN-SMS tatsächlich beim Kunden ankommt und nicht umgeleitet wird, muss dem Mobilfunkbetreiber bzw. deren Angestellten vertraut werden.

In letzter Zeit sind auch Malware-basierte Attacken gegen das mTAN-Verfahren bekannt geworden, die alle größeren mobilen Plattformen betreffen (für Windows Mobile s. [8], für Symbian s. [9], für Android s. [10]). Bei diesen Angriffen wird das Smartphone des Opfers derart manipuliert, dass es eine ankommende TAN-SMS der Bank an den Angreifer weiterleitet, der wiederum die Transaktionsdaten in der SMS anpasst und die so manipulierte SMS an das Opfer zurück schickt. Derartige Angriffe auf mobile Plattformen werden vermutlich noch zunehmen, da Smartphones als Zielscheibe von Malware-basierten Attacken immer attraktiver werden. Zwar könnte man argumentieren, dass zum Brechen des mTAN-Verfahrens sowohl der PC als auch das Smartphone des Nutzers kompromittiert sein müssen, was die Wahrscheinlichkeit eines solchen Angriffs deutlich verringert. Es ist jedoch davon auszugehen, dass viele Kunden, wenn sie ohnehin zur Handynutzung bei der Erledigung ihrer Bankgeschäfte gezwungen sind, diese komplett über ihr Smartphone durchführen werden, ohne auf einen PC zurück zu greifen. Der eigentlich der Sicherheit dienende „Medienbruch" zwischen Smartphone und PC ist dann hinfällig. Letztlich wird durch die mTAN nur eine unsichere Plattform – der Opfer-PC – durch eine andere unsichere Plattform – das Smartphone – ersetzt. Zwar könnte man mit technischen Mitteln sicher stellen, dass die User tatsächlich zwei voneinander unabhängige Geräte für das e-Banking benutzen, dies wird aber sicher keine Akzeptanz beim User finden. Die Transaktionsabhängigkeit der TAN, obwohl ein Schritt in die richtige Richtung, reicht offenbar nicht aus, um die Sicherheit des Gesamtsystems zu gewährleisten.

4 Das hPIN/hTAN Token

Eine weitere Alternative zur Erzeugung transaktionsabhängiger TANs bildet der Einsatz von Hardware-basierten TAN Generatoren. Derartige Generatoren sind bereits von einigen Banken in Deutschland eingeführt worden, haben aber noch eine relativ geringe Verbreitung, da sie häufig durch die Kunden käuflich erworben werden müssen. Eine lobenswerte Ausnahme bildet der TAN-Generator der BW-Bank, der – mit eigener Tastatur und Display ausgestattet – sogar vom Fraunhofer Institut für sichere Telekooperation (SIT) zertifiziert wurde und kostenlos an die Kunden abgegeben wird (s. [11]). Bedingt durch die Tastatur, besitzt dieser Generator allerdings bereits eine Größe, die seine Portabilität im Alltag einschränkt.

Erfahrungsgemäß lässt sich eine weit reichende Nutzung von TAN-Generatoren nur erreichen, wenn die Geräte möglichst billig, wenig komplex und einfach zu bedienen sind, wie die geringe Verbreitung von HBCI zeigt. Zudem muss aus unserer Sicht gewährleistet sein, dass die e-Banking Transaktionen auch bei einem vollständig kompromittierten Computer nicht zu manipulieren sind. Das schließt den Einsatz des Computers als Ein/Ausgabemedium weitgehend aus, so dass ein sicheres Hardware-Token eigentlich zwingend über eigene Tastatur und Display verfügen muss, was wiederum die Kosten in die Höhe treibt. In diesem Spannungsfeld gegensätzlicher Anforderungen wurde das hPIN/hTAN Token (s. [12]) entwickelt mit dem Ziel, trotz minimalistischer Ausstattung (Zweizeiliges Display und OK Button) und einer Größe, die nicht wesentlich über übliche USB-Token hinaus geht, komplette Sicherheit gegen kompromittierte PC zu gewährleisten. Zur Nutzung des Tokens muss clientseitig lediglich ein Browser-Plugin installiert werden. Zudem kommt das Token komplett mit symmetrischer Kryptografie (eine sichere Hashfunktion genügt) aus, wodurch Implementationskosten und

Ausführungszeiten gering bleiben. Bei großen Stückzahlen dürften die Herstellungskosten für das Token im Bereich zwischen drei und fünf Euro liegen.

4.1 hPIN Protokoll

Das hPIN Protokoll dient zum einen zur sicheren Authentifikation des Nutzers beim Token, zum Anderen aber auch der gegenseitigen Authentifikation von Token und Server über ein symmetrisches Challenge-Response Protokoll. Da beim hPIN/hTAN Token auf eine eigene Tastatur verzichtet wurde, muss die PC Tastatur als Eingabemedium benutzt werden. Trotzdem kann die zum Token gehörige PIN auf sichere Weise, d.h. ohne dass der PC die PIN erfährt, eingegeben werden. Dazu erzeugt das Token bei jeder Initialisierung, d.h. bei Einstecken in den USB-Port des PC, einen zufälligen Code, der die PIN-Ziffern durch andere Zeichen ersetzt. Dieser Code wird im Display angezeigt (s. Abb. 7). Auf der PC Tastatur gibt der Nutzer die zu den PIN-Ziffern gehörigen codierten Zeichen ein. Nach jeder Zeicheneingabe wird ein neuer Code P_i vom Token erzeugt und angezeigt, so dass gleiche PIN-Ziffern nicht durch gleiche Codezeichen erkennbar sind.

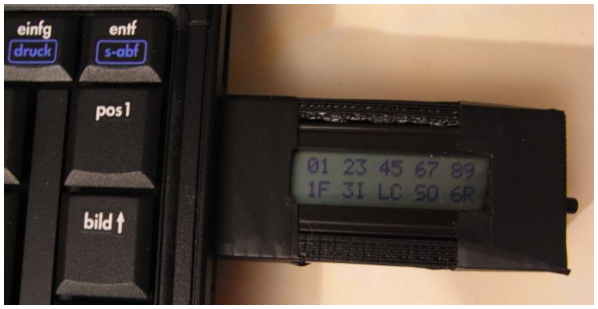


Abb. 7: Das hPIN/hTAN Token

Nach erfolgreicher Authentifikation des Nutzers beim Token führen Token und Server ein gegenseitiges symmetrisches Authentifikationsprotokoll durch. Der dem Protokoll zu Grunde liegende, nutzerindividuelle Schlüssel K_T sowie die PIN werden von der Bank bei der Personalisierung des Token in gehashter Form auf dem Token abgelegt, so dass diese Daten auch bei Verlust des Tokens nicht ausgelesen werden können. Genauer gesagt, sind auf dem Token die folgenden Daten gespeichert:

- NutzerID ID_U,
- Salt s,
- $K_T^* = K_T \oplus hash(PIN||s)$
- $PIN^* = HMAC(K_T, PIN||s)$
- Fehlbedienungszähler C_T

Somit kann K_T nicht ohne Kenntnis der PIN berechnet werden. Serverseitig werden lediglich ID_U und der dazu gehörige Schlüssel K_T gespeichert. Nachdem der Nutzer das Token eingesteckt und den OK Button gedrückt hat, laufen folgende Schritte ab:

- Das Token erzeugt für jede der n PIN-Ziffer einen zufälligen Code zur sicheren Eingabe der PIN auf der PC-Tastatur, wie in Abb. 7 gezeigt.
- Nach der Eingabe aller PIN-Ziffern berechnet das Token $K_T = K_T^* \oplus hash(PIN||s)$ und verifiziert, ob $PIN^* = HMAC(K_T, PIN||s)$ gilt. Falls ja, ist der Nutzer authentifiziert. Falls nein, wird der Fehlbedienungszähler um eins erhöht und der Nutzer erhält einen weiteren Versuch der PIN-Eingabe, sofern die maximale Anzahl an Fehlversuchen noch nicht überschritten ist.
- Nach erfolgreicher Nutzerauthentifizierung legt das Token K_T zur weiteren Verwendung in seinem flüchtigen Speicher ab. Auf Basis von K_T kann nun ein beliebiges beidseitiges Authentifikationsprotokoll

zwischen Token und Server ablaufen. In unserer prototypischen Implementation haben wir das leichtgewichtige SKID3-Protokoll [13] verwendet.

Abb. 8 zeigt den Ablauf des hPIN-Protokolls noch einmal in vereinfachter Form. Dabei wird davon ausgegangen, dass die PIN aus *n* Zeichen *PIN(1), PIN(2),…, PIN(n)* besteht.

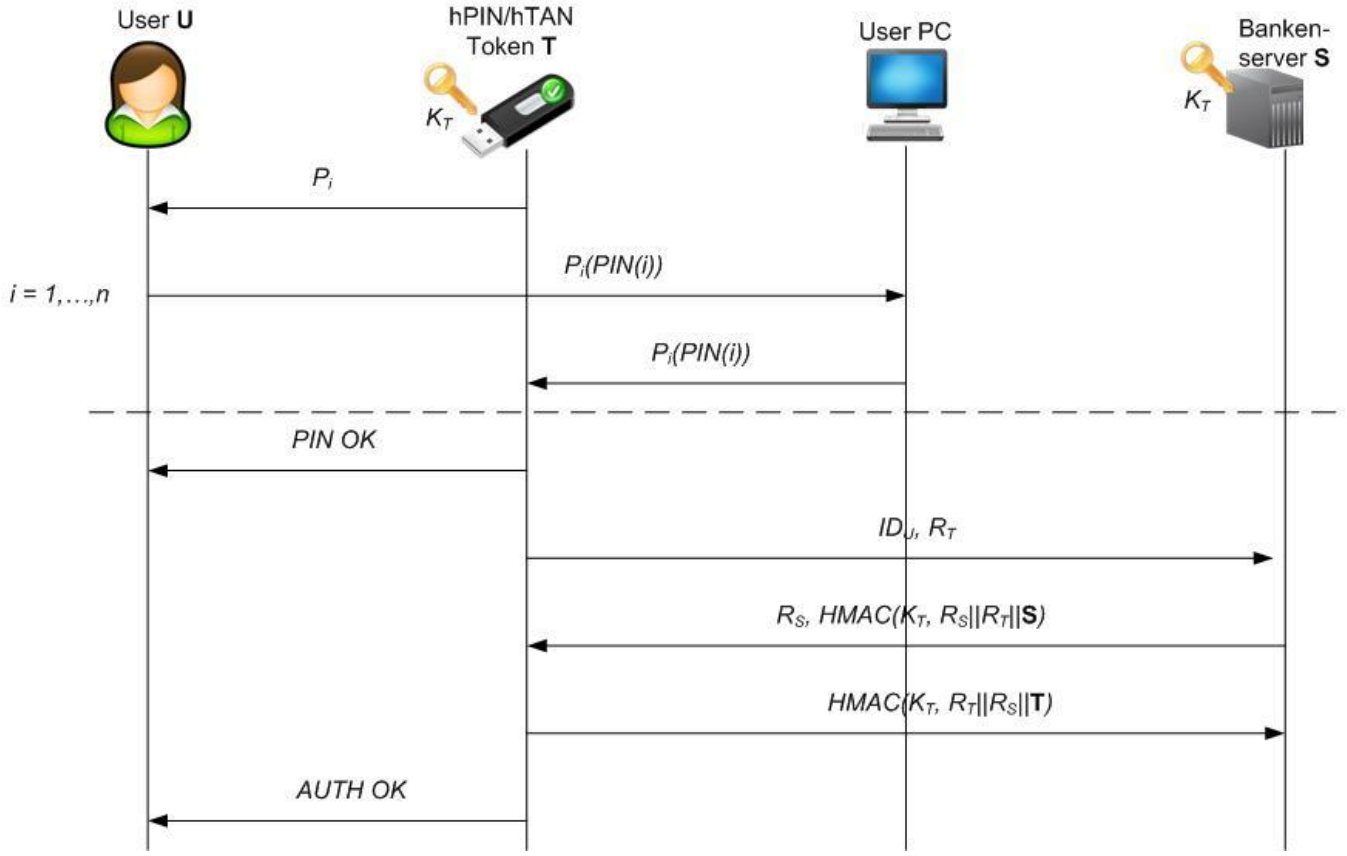


Abb.8: Das hPIN-Protokoll (vereinfachte Darstellung)

4.2 hTAN Protokoll

Das hTAN Protokoll dient der Erzeugung transaktionsabhängiger TANs sowie zur gegenseitigen Authentifizierung von Token und Server nach der Übermittlung der Transaktionsdaten. Da der Browser des Nutzers diese Daten manipulieren und verfälscht darstellen könnte, muss der Nutzer diese einzeln auf dem Display des Token überprüfen und durch den OK Button authorisieren. Um zu verhindern, dass der Nutzer sich bei der Überprüfung der Transaktionsdaten auf das PC-Display verlässt, werden die Transaktionsdaten am PC verdeckt dargestellt und sind nur auf dem Token-Display sichtbar. Nachdem alle Transaktionsdaten eingegeben und vom Nutzer authorisiert sind, laufen die folgenden Schritte ab:

- Das Token schickt die Transaktionsdaten TD zusammen mit der NutzerID ID_U und einer neuen Random Challenge R_T* an den Server.
- Der Server berechnet als Response einen vom gemeinsamen Schlüssel K_T abhängigen Hashwert (HMAC) über die Transaktionsdaten und die Challenge R_T*. Zudem schickt der Server eine eigene neue Challenge R_S* an das Token.
- Kann die Response durch das Token erfolgreich verifiziert werden, ist sicher gestellt, dass die Transaktionsdaten auch wirklich beim Server angekommen sind. Das Token erzeugt dann seinerseits

einen schlüsselabhängigen Hashwert über die Transaktionsdaten, R_S* sowie R_T* und sendet den Hashwert als Response an den Server. Dieser Hashwert dient gleichzeitig auch als transaktionsabhängige TAN.

- Mit Hilfe der TAN kann der Server das Token und die im ersten Schritt erhaltenen Transaktionsdaten authentifizieren und die gewünschte Transaktion durchführen. Gleichzeitig wird eine weitere, wiederum per HMAC gesicherte Kontrollnachricht an das Token als Benachrichtigung geschickt.

Abb. 9 zeigt den gesamten Ablauf des hTAN-Protokolls noch einmal in vereinfachter Form.

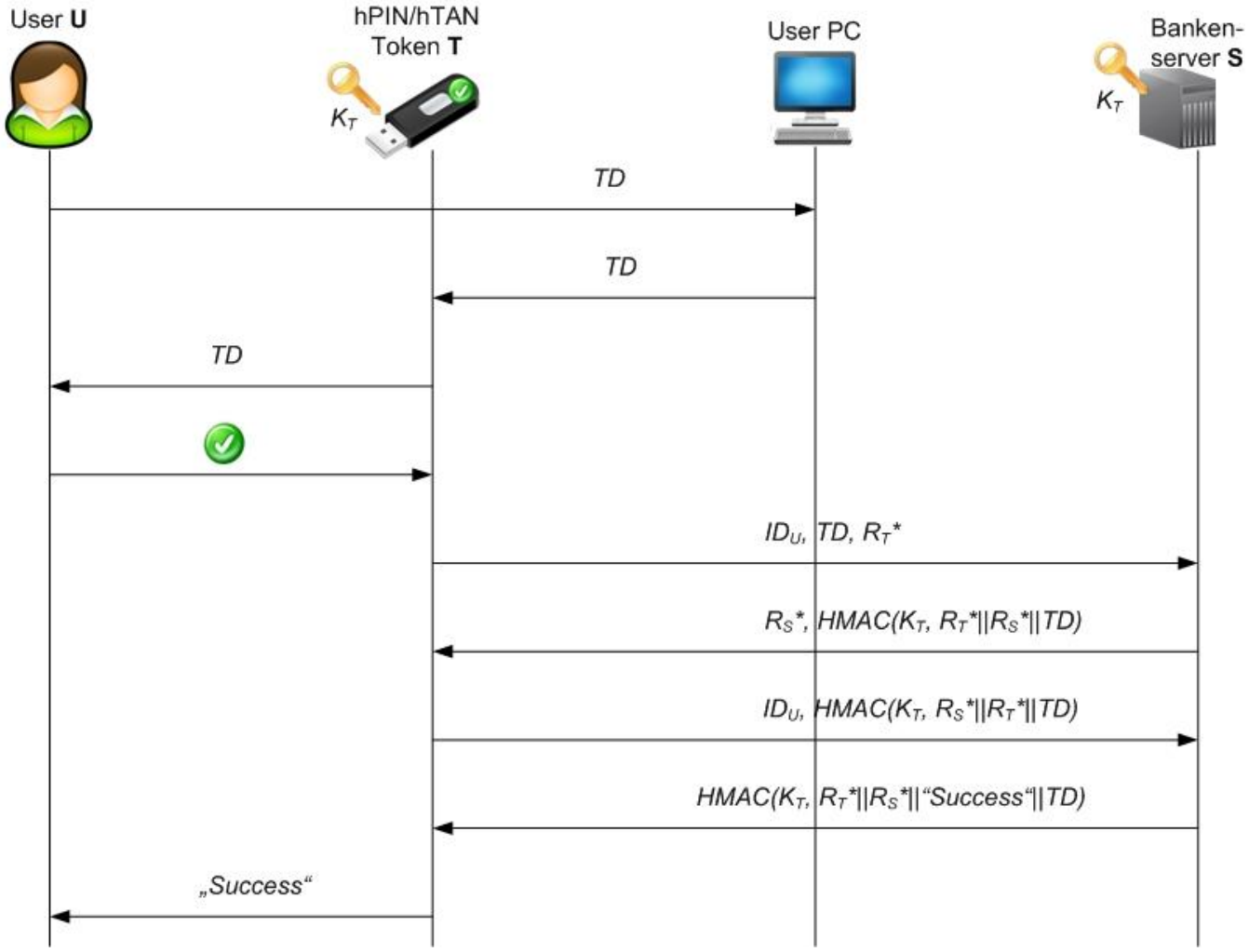


Abb.9: Das hTAN-Protokoll (vereinfachte Darstellung)

4.3 Nutzerstudie

Die Usability des Prototyps wurde in einer ersten Nutzerstudie mit 20 Teilnehmern getestet. Nach einer kurzen Einführung wurden die Teilnehmer, die zuvor keinen Kontakt mit dem Token gehabt hatten, gebeten, sich über das hPIN Protokoll beim Token zu authentifizieren und danach eine Beispieltransaktion durchzuführen. Hierfür wurde eine virtuelle Banken-Webseite aufgesetzt (http://www.hPIN-hTAN.net).

Die Median Login-Zeit betrug bei einer vierstelligen PIN und einer Erfolgsrate von 91% ca. 27,5 Sekunden. Die Beispieltransaktion mit insgesamt 55 Zeichen konnte in ca. 70 Sekunden durchgeführt werden, was 1,27 Sekunden pro Zeichen entspricht. Bedenkt man, dass bei einer papierbasierten Lösung wie iTAN die TANs erst gefunden werden müssen und bei der mTAN-Lösung die Wartezeit auf die TAN-SMS hinzu kommt, sind dies durchaus akzeptable Werte. Eine Umfrage unter den Teilnehmern zeigte, dass keiner von ihnen Schwierigkeiten

hatte, das System zu verstehen. Die Usability des Systems wurde im Schnitt mit 3,65 auf einer 5-Punkte Skala bewertet.

5 Fazit

Im vorliegenden Beitrag wurden CAPTCHA-basierte Sicherheitslösungen für das Online-Banking untersucht. Dabei hat sich heraus gestellt, dass keines dieser Verfahren Sicherheit gegen einen Malware-basierten Man-in-the-Middle Angriff bieten kann. Wir empfehlen daher, die Sicherheit von e-Banking Lösungen in Zukunft nicht mehr auf CAPTCHAs aufzubauen.

Aufgrund der Komplexität heutiger mobiler Betriebssysteme scheint es nur eine Frage der Zeit zu sein, bis sich das Bedrohungspotenzial der Smartphone-basierten mobilen TAN – Lösung dem reiner PC-basierten Lösungen angepasst hat. Wir propagieren daher Lösungen, die transaktionsabhängige TANs auf Basis eines sicheren HW-Tokens generieren. Existierende Lösungen sind jedoch häufig zu komplex und damit zu teuer für eine flächendeckende Anwendung. Mit dem hPIN/hTAN Token haben wir den Prototyp eines sicheren, kostengünstigen USB-Tokens vorgestellt, der nur mit einem Display und einem OK-Button auskommt und trotzdem maximale Sicherheit gegen einen komplett nicht vertrauenswürdigen Computer bietet.

Literatur

[1] G. Borges , J. Schwenk, C. F. Stuckenberg, C. Wegner , *Identitätsdiebstahl im Internet: Rechtliche und technische Aspekte*, Springer-Verlag 2011

[2] Bundesamt für Sicherheit in der Informationstechnik, *Die Lage der IT-Sicherheit in Deutschland 2011*, https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Lageberichte/Lagebericht2011.pdf

[3] S. Li, S. Shah, M. Khan, S. Khayam, A. R. Sadeghi, R. Schmitz: *Breaking e-Banking CAPTCHAs*, in: Proc. ACSAC 2010, pp. 171 - 180

[4] J. Steinmüller: *Mathematische Bildanalyse*, Springer-Verlag 2008

[5] Z. Wang, E. P. Simoncelli. *Translation insensitive image similarity in complex wavelet domain*, in Proc. ICASSP 2005, pp. 573 -576.

[6] http://www.it-online.co.za/content/view/1092105/142/

[7] http://www.heise.de/newsticker/meldung/mTAN-in-Australien-ausgehebelt-828221.html

[8] http://www.heise.de/newsticker/meldung/Online-Banking-Trojaner-attackiert-Smartphones-mit-Windows-Mobile-1195455.html

[9] http://www.heise.de/security/meldung/Angriffe-auf-deutsche-mTAN-Banking-User-1221951.html

[10] http://www.heise.de/security/meldung/ZeuS-Trojaner-befaellt-Android-1278449.html

[11] http://www.bw-bank.de/privatkunden/1000006911-de.html

[12] S. Li, A. R. Sadeghi, S. Heisrath, R. Schmitz, J. A. Ahmad: *hPIN/hTAN: A Lightweight and Low-Cost e-Banking Solution against Untrusted Computers*, in Proc. Financial Cryptography 2011, Springer-Verlag 2011

[13] A. Bosselaers, B. Preneel: *SKID*, in: Integrity Primitives for Secure Information Systems: Final Report of RACE Integrity Primitives Evaluation RIPE-RACE 1040, pp. 169-178, Springer –Verlag 1995