

Abwehrmaßnahmen gegen Angriffe auf Chipkarten oder Die erstaunlich vielen verschiedenen Arten eine PIN sicher zu prüfen.

von Wolfgang Rankl, München

25. Mai 2009

Seit ihrer Erfindung in den achtziger Jahren haben sich Chipkarten zu einem universellen Medium für die sichere Speicherung von Daten und die geschützte Ausführung von Programmen entwickelt. Dies wird recht eindrucksvoll durch die immense Zahl der in den Markt gehenden Chipkarten dokumentiert – es waren alleine im Jahr 2008 etwas über vier Milliarden Stück! Interessant dabei ist auch, dass dabei ein erheblicher Anteil an Chipkarten dabei ist, die außerhalb des als Haupttreiber dieser Technologie fungierenden Telekomumfeldes eingesetzt werden. Etwa ein viertel dieser Chipkarten finden nämlich ihren Einsatz in den Bereichen Zahlungsverkehr, Transportwesen und Identifizierung.

Die Gründe dieses erstaunlichen Erfolgs sind dabei gar nicht einmal so vielschichtig wie man meinen möchte. Die Grundvoraussetzung ist die bereits eingangs erwähnte Möglichkeit Daten sicher zu speichern und Programme geschützt auszuführen. Die Vielfalt an möglichen Speichergrößen, die zur Verfügung stehende Rechenleistung sowie die in weiten Bereich skalierbare Sicherheit erhöhen die Bandbreite der Einsatzmöglichkeiten. Eine wichtige Rolle spielt natürlich auch, dass Chipkarten breit etabliert und auch von Benutzern und Sicherheitsexperten weithin akzeptiert sind. Das gerade von Ingenieuren gerne übersehene abschließende Argument dieser Auflistung ist der Preis von Chipkarten, der den Einsatz als Massenprodukt sehr begünstigt.

Die unterschiedlichen Formfaktoren von Chipkarten sind weithin bekannt. Andererseits hat die Chipkarten-Technologie in den letzten Jahren auch in viele anderen Bauformen Einzug gehalten. Als Beispiele kann man hier kontaktlose Reisepässe, USB-Token, Karten mit Flash-Speicher (z.B. MicroSD-Karten), kontaktlose Tags und auch Karten mit Display und Tasten aufzählen. Alle diese Ausführungsformen von Chipkarten haben eines gemein - sie benutzen ähnliche oder sogar die gleichen Sicherheits-Mikrocontroller benutzen wie die klassischen Chipkarten.

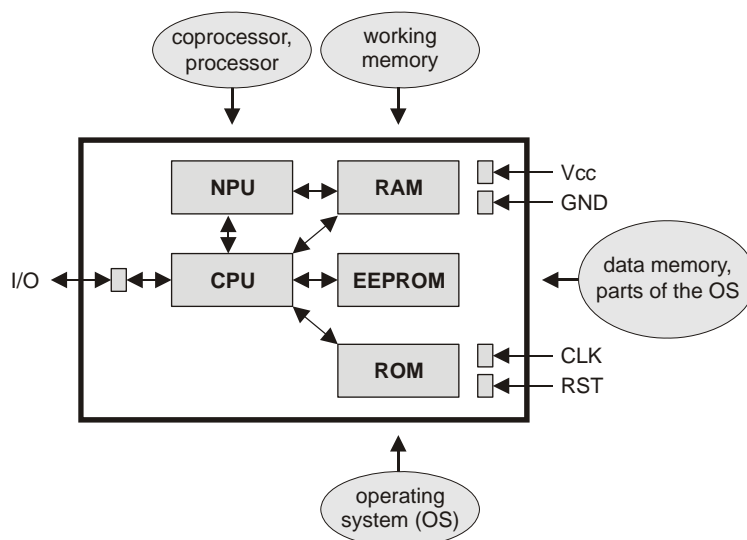


Bild 1 Typische Architektur eines Mikrocontrollers für Chipkarten mit maskenprogrammiertem ROM und numerischen Koprozessor (NPU).

Ein exemplarisches Beispiel für die Architektur solch eines Mikrocontrollers zeigt Bild 1. Dabei handelt es sich um einen Chip mit maskenprogrammiertem ROM und einen numerischen Koprozessor, wie er beispielsweise zur schnellen Berechnung von RSA-Algorithmen oder Elliptischen Kurven verwendet wird. Solche Mikrocontroller werden aktuell häufig im Umfeld des elektronischen Zahlungsverkehrs eingesetzt.

Die Sicherheit eines informationstechnischen Systems mit Chipkarten ist vielschichtig und hängt selbstverständlich nicht ausschließlich von den Chipkarten ab, allerdings sind diese ein wesentlicher Faktor im Gesamtsystem. Im Prinzip verhält es sich mit den einzelnen Elementen der Sicherheit eines Chipkarten-Systems wie mit den Gliedern einer Kette. Alle Glieder der Kette tragen gleichermaßen dazu bei, dass sie bei Beanspruchung nicht zerreißt. Sobald jedoch ein einziges Glied der Kette bricht, ist es vorbei mit der Stabilität. Analog verhält es sich mit der Sicherheit eines Chipkartensystems. Kann eines der Elemente gebrochen werden, dann ist auch die Gesamtsicherheit nicht mehr gewährleistet.

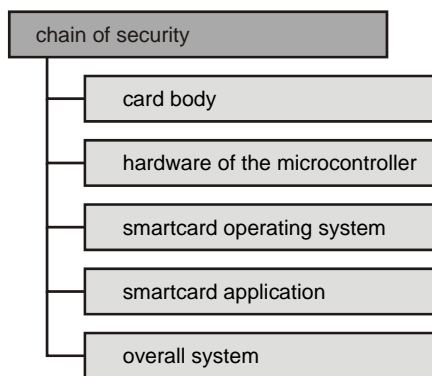


Bild 2 Die Kette der Sicherheitselemente in einem informationstechnischen System mit Chipkarten.

Die dabei zum Einsatz kommenden Sicherheitselemente sind selbstverständlich stark abhängig vom Gesamtsystem. Es lassen sich hier aber durchaus typische Architekturen von Chipkartensystemen erkennen, bei denen immer wieder die gleichen Sicherheitselemente zum Einsatz kommen.

Benutzt man Chipkarten in einem Umfeld, in dem ihre Echtheit von Menschen geprüft werden muss, wird man die Karten mit entsprechenden, meist visuellen Sicherheitsmerkmalen ausstatten. Beispiele dafür zeigen die aktuellen Zahlungsverkehrskarten mit ihren Hologrammen, Mikroschriften, im UV-Licht sichtbaren Elementen und ähnlichen direkt oder mit geringem Zusatzaufwand von Menschen überprüfbar Merkmalen.

Das wesentliche Kartenelement einer Chipkarte ist natürlich der eingebaute Chip. Hier gibt es drei Aspekte, welche für die Sicherheit verantwortlich zeichnen. Dies ist zum einen der verwendete Mikrocontroller. Er muss speziell auf die Sicherheitsanforderungen abgestimmt sein und alle notwendigen Schutzmaßnahmen gegen die bekannten Angriffe aufweisen. Eine umfangreiche Beschreibung dieser Maßnahmen findet sich im Handbuch der Chipkarten (Smart Card Handbook) [Rankl 09], auf das an dieser Stelle verwiesen sei.

Für einen überblicksartigen Eindruck der mittlerweile sehr ausgefeilten Schutzmechanismen seitens der Hardware seien an dieser Stelle einige interessante Neuerungen der letzten Jahre kurz aufgeführt. Die Kommunikation auf dem Chip zwischen Prozessor und den unterschiedlichen Speichern (z.B. RAM, ROM, EEPROM) findet in verschlüsselter Form statt. Die Benutzung von sitzungsindividuellen Schlüsseln ist hier durchaus bereits Stand der Technik, so dass nach jedem Reset der Chipkarte die interne Kommunikation mit neuen abgeleiteten Schlüsseln geschützt wird. Im Regelfall sind auch die Inhalte der einzelnen Speicher chipindividuell verschlüsselt, um die Daten zusätzlich zu den Maßnahmen des Chipkarten-Betriebssystems zu absichern.

Als Abwehrmaßnahme gegen viele Arten von invasiven Angriffen besitzen eine Reihe von Mikrocontrollern über der eigentlichen Schaltung liegende Metallschichten (*shield*), deren Unversehrtheit durch spezielle Logikelemente laufend geprüft wird. Die mittlerweile im Repertoire jedes besseren Labors befindlichen SPA/DPA-Analysen (*single power analysis/differential power analysis*) führten ebenfalls zu einer Reihe recht wirksamer Abwehrmaßnahmen seitens der Hardware. Als Beispiele sei hier die komplementäre Logik genannt, bei der jeder Pegeländerung auf dem Chip immer eine gegenläufige Pegeländerung entgegensteht, so dass die Schwankung des Stromverbrauchs minimiert

wird. Ein weiterer Schutzansatz sind auf dem Chip integrierte Stromglätter, die jegliche Stromschwankungen des Mikrocontrollers nach außen hin wegnivelieren.

Zur Erkennung und Abwehr der zumindest in Laborumgebung recht starken Angriffe mit energiereichen Laserstrahlen wurden in den letzten Jahren ebenfalls eine Vielzahl von Maßnahmen erdacht. Dies umfasst zum einen eine größere Zahl von auf dem gesamten Chip verteilten Lichtsensoren, die sogar auch noch Streulicht von stark fokussierten Lasern detektieren können. Neben dieser direkten Detektion von Fehlereinstreuungen befinden sich mittlerweile sogar Mikrocontroller auf dem Markt, die mit zwei Prozessoren in unterschiedlicher Implementierung jeden Programmablauf parallel zweifach ausführen und auf diese Weise Fehlereinstreuungen gut erkennen können. Diese Schutzmaßnahme der sich gegenseitig kontrollierenden Prozessoren ist gegenwärtig nicht weit verbreitet, hat aber durchaus einen gewissen technischen Reiz. Hingegen haben sich per Zufall gesteuerte Wartezyklen des Prozessors (*Random-Wait-States*) zum Erschweren der Synchronisation von Angriffsequipment mit den internen Chipabläufen in den letzten Jahren als Standardfeature nahezu auf allen Chipkarten-Mikrocontrollern etabliert.

Unabhängig von den mathematisch orientierten Angriffen auf Kryptoalgorithmen kristallisierte sich in den letzten Jahren eine ziemlich gut anwendbare Systematik zur Einteilung der Angriffe auf Chipkarten heraus. Diese beruht auf der Invasivität des jeweiligen Angriffs und sieht drei verschiedene Abstufungen vor. Ein invasiver Angriff bedeutet, dass ein physischer Eingriff auf dem Chip vorgenommen wird. Klassisch zu nennende Beispiele sind hier FIB (*focused ion beam*) oder Laser Cutter. Bei den non invasiven Angriffen hingegen wird kein physischer Einfluss auf den Chip ausgeübt, sondern nur das Verhalten des Chips von außen messtechnisch beobachtet. Die Beispiele hierzu sind Strommessungen (SPA/DPA), Abstrahlungsmessungen (SEMA/DEMA - *simple electromagnetic analysis/ differential electromagnetic analysis*) und Zeitmessungen (*timing attack*).

In der Mitte zwischen diesen beiden Polen bewegen sich die semi invasiven Angriffe. Bei ihnen werden zwar keine physischen Änderungen auf dem Chip durchgeführt, allerdings wird der Chip trotzdem von außen beeinflusst. Dies kann etwa das fokussierte Licht eines Lasers sein oder auch energiereiche Strahlung. Die beabsichtigte Wirkung ist dabei in der Regel die Einstreuung von Fehlern in den Programmablauf des Mikrocontrollers. Mit dieser Dreiteilung lassen sich alle derzeit bekannten Angriffe auf Chipkarten gut zuordnen, was ganz gewiss eine Voraussetzung für umfassende Schutzmaßnahmen ist.

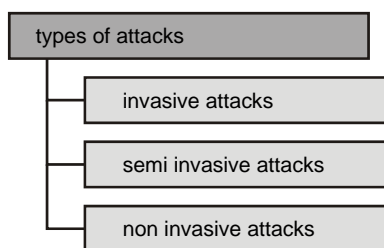


Bild 3 Die aus der Praxis entstandene und auch gut einsetzbare Systematik der Angriffe auf Chipkarten.

Die Entwicklung der sicheren PIN-Überprüfung als Praxisbeispiel

Im vorangehenden Text sind einige wichtige Maßnahmen zum Schutz der Daten und Programmabläufe einer Chipkarte beschrieben. Der Sinn all dieser Mechanismen erschließt sich aber am besten, wenn man diese anhand ihrer Anwendung in der Praxis betrachtet. Ein sehr anschauliches Beispiel dazu ist eine eigentlich ziemlich kleine Funktion im Chipkarten-Betriebssystem, welche den Vergleich der vom Benutzer eingegebenen PIN (*personal identification number*) mit der in der Chipkarte gespeicherten PIN durchführt. Das dazugehörige Kommando hat je nach zugrundeliegender Norm unterschiedliche Namen wie VERIFY nach ISO/IEC 7816-4 oder VERIFY CHV nach ETSI TS 51.011. Die Funktionalität ist jedoch immer annähernd identisch. Es ist dies der Vergleich der vom Benutzer am Terminal eingegebenen PIN zu einer in der Chipkarte gespeicherten Referenz-PIN.

Die PIN ist neben den geheimen Schlüsseln das wesentliche Geheimnis einer Chipkarte, denn mit ihr wird sichergestellt, dass die Chipkarte nur vom autorisierten Benutzer verwendet werden kann. Den dazugehörigen Ablauf der Authentisierung des Benutzers durch eine PIN-Überprüfung zeigt Bild 5. Die auslösende Aktion ist dabei die Aufforderung des Terminals an den Benutzer seine PIN einzugeben. Hat er dies gemacht, so wird die PIN in das Kommando VERIFY verpackt und an die Chipkarte gesendet. Die Übertragung kann dabei auch optional in kryptografisch gesicherter Form erfolgen (Secure Messaging), so dass ein Abhören der Datenübertragung einem Angreifer keinen Erfolg bringt.

Coding of the Command-APDU

CLA	INS	P1	P2	Lc	Data	Le
'00'	'20'	'00'	'00'	8	'30 30 30 30 31 32 33 34'	'00'

Coding of the Response-APDU

SW1	SW2
'90'	'00'

Bild 4 Die Kodierung des Kommandos VERIFY und des dazugehörigen Returncodes. Die eingetragenen Beispielwerte gelten für eine globale PIN mit dem ASCII-kodierten Wert 1234 für den Gutfall.

Ist die PIN mit VERIFY und einer typischen Länge von vier numerischen Zeichen bei der Chipkarte angelangt, dann prüft diese als erstes ob der Fehlerzähler (*error counter*) seinen Maximalwert erreicht hat. Dieser hat in der Regel den Wert drei und schützt vor einem Angriff durch Ausprobieren aller möglichen PINs. Hat der Fehlerzähler diesen Maximalwert erreicht, dann wird ein PIN-Vergleich abgeblockt und die Chipkarte sendet einen entsprechenden Returncode an das Terminal zurück. Dieses informiert dann den Benutzer, dass weitere PIN-Eingaben nicht möglich sind und er seine Karte entblocken muss. Dies kann er typischerweise durch die Eingabe einer PUK (*personal unblocking key*) und einer neuen PIN erreichen.

Hat der Fehlerzähler seinen Maximalwert jedoch noch nicht erreicht, dann kopiert die Chipkarte aus dem nichtflüchtigen Speicher (EEPROM oder Flash) die Referenz-PIN in das RAM und vergleicht diese mit der vom Benutzer eingegebenen PIN. Sind die beiden PINs nicht identisch, wird der Fehlerzähler um eins erhöht und dem Terminal eine entsprechende Information gesendet, welche in der Regel auch den aktuellen Wert des Fehlerzählers enthält.

Ist der Maximalwert des Fehlerzählers noch nicht erreicht und entspricht die an die Chipkarte übergebene PIN der gespeicherten Referenz-PIN, dann prüft die Software der Chipkarte ob der Fehlerzähler auf null steht und setzt ihn bei Bedarf auf null zurück. Der Benutzer hat damit wieder seine üblichen drei Versuche bei der PIN-Eingabe. Anschließend wird der durch die erfolgreiche PIN-Prüfung erreichte Sicherheitszustand im flüchtigen Speicher der Chipkarte (RAM) gesetzt. Typischerweise ist dann ein lesender oder schreibender Zugriff auf bestimmte Dateien der Chipkarte möglich oder bestimmte Kommandos dürfen dann vom Terminal auf der Chipkarte ausgeführt werden. Anschließend informiert die Chipkarte mit einem bestimmten Returncode ('9000') das Terminal über die erfolgreiche Überprüfung der PIN.

Dies ist der grundlegende Ablauf der Überprüfung einer PIN oder allgemein ausgedrückt, eines Geheimnisses in einer Chipkarte. Falls man aus Implementierungssicht exakt diesen Ablauf haben möchte, so muss die Routine in Assembler und nicht wie bei Chipkarten üblich in der Programmiersprache C ausgeführt werden. Damit umgeht man die Gefahr, dass der C-Compiler bestimmte Anweisungen umorganisiert oder sogar wegoptimiert. Bei den im Anschluss gezeigten Abwehrmaßnahmen gegen die unterschiedlichen Angriffe ist dies von essentieller Bedeutung, da die Ablaufpfade des Programms zur Verwirklichung von Schutzmaßnahmen oftmals Redundanzen aufweisen, die ansonsten den Optimierungsmaßnahmen des Compilers zum Opfer fallen würden .

Das Thema sichere Prüfung einer PIN ist kein wirklich neues Thema, da es seit den ersten Chipkarten Ende der achtziger Jahre ständig weiterentwickelt und verbessert wird. Da es sehr hardwarenah ist, sind viele der Mechanismen auch patentierbar, was auch vielfach geschehen ist.

Das Bild 5 zeigt den grundsätzlichen Ablauf einer PIN-Überprüfung. Dieser dürfte auch ungefähr dem Ablauf entsprechen, den ein sicherheitstechnisch unbedarfter Entwickler implementieren würde. Technisch gesehen entspricht dies ungefähr dem Wissenstand von 1990. In den folgenden Absätzen wird nun gezeigt, wie die Sicherheitstechnik in den folgenden Jahren vorangeschritten ist und welche Gegenmaßnahmen ergriffen werden, um Angriffe erfolgreich abzuwehren.

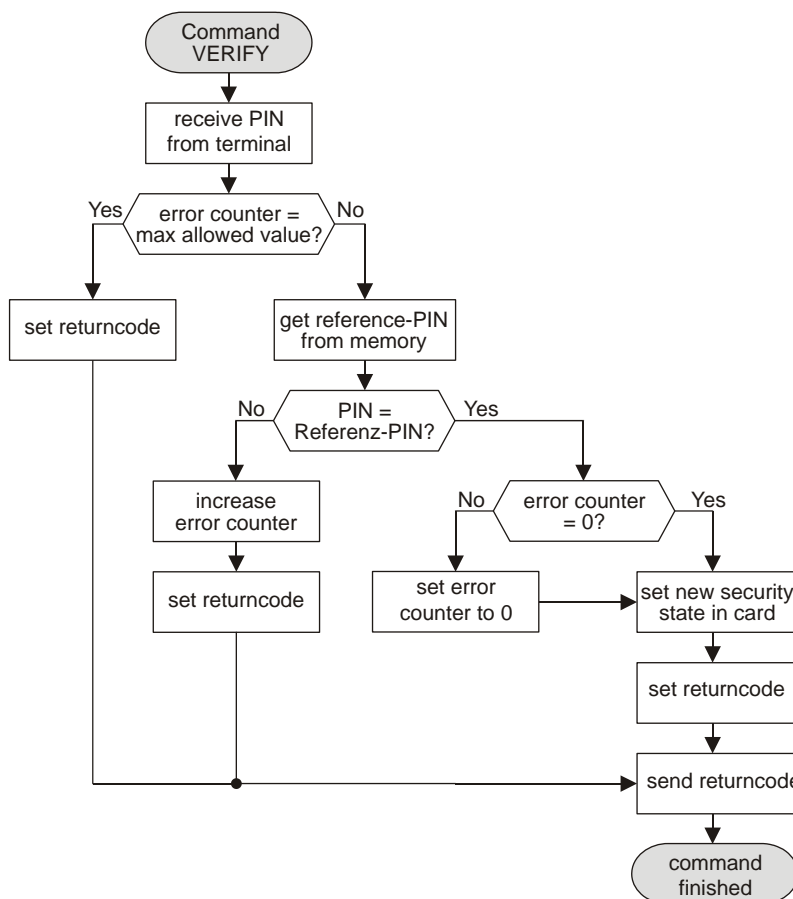


Bild 5 Der Programmablauf bei der Ausführung des Chipkarten-Kommandos VERIFY zur Authentisierung eines Benutzers mittels PIN. Die hier dargestellte Funktion weist bis auf den Fehlerzähler keine Maßnahmen zum Schutz gegen Angriffe auf.

Angriff und Abwehr: Zeitmessung des PIN-Vergleichs

Der Vergleich zwischen der vom Benutzer am Terminal eingegebenen PIN und der in der Chipkarte gespeicherten Referenz-PIN ist aus informationstechnischer Sicht ein simpler Vergleich von zwei Strings. Ein normaler Entwickler wird dazu entweder die von der C-Bibliothek zur Verfügung gestellte Funktion benutzen oder sich diesen Vergleich mit einigen wenigen Zeilen Code selber erstellen. Da es programmtechnisch am einfachsten und zudem auch noch im Programmablauf am schnellsten ist wird man vernünftigerweise den Vergleich bei der ersten nicht übereinstimmenden Zahl abbrechen. Das Bild 6 zeigt so einen üblichen Vergleich von zwei Strings.

Diese Zeitabhängigkeit beim Vergleich lässt sich jedoch recht gut für einen Angriff nutzen. Dazu sendet man eine geratene PIN zur Chipkarte und misst die Zeit bis die Antwort der Karte eintrifft. Diese Zeitspanne ist umso kleiner, je weiter vorne im PIN-Vergleich die erste Abweichung zwischen der eingegebenen PIN und der Referenz-PIN aufgetreten ist. Damit kann man aufgrund des Fehlerzählers zwar nicht alle PINs durchprobieren, aber es lässt sich der Zahlenraum der möglichen PINs doch erheblich einschränken. Allgemeiner

ausgedrückt öffnet diese etwas zu einfache Vergleichsroutine einen Seitenkanal mit Zeitinformationen, den ein Angreifer nutzen kann.

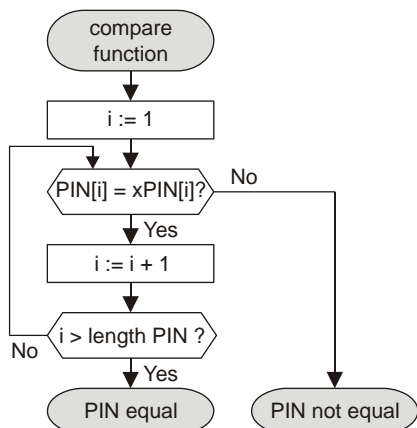


Bild 6 Beispiel für eine unsichere PIN-Vergleichsroutine, deren Zeitdauer abhängig von den übereinstimmenden Stellen der vom Benutzer eingegebenen PIN und der in der Chipkarte gespeicherten Referenz-PIN ist. Diese Vergleichsroutine bricht den Vergleich bei der ersten ungleichen Stelle des Geheimnisses ab. Dies ist der übliche Ansatz bei laufzeitminimierten Abläufen, kann aber gut als Ansatzpunkt für einen Angriff genutzt werden. Die folgenden Variablen werden verwendet: i – Zähler für die Anzahl der verglichenen Stellen, PIN – Feld in der Chipkarte mit dem gespeicherten Geheimnis (Referenz-PIN), $xPIN$ – Feld mit dem zum Vergleich übergebenen Geheimnis.

Die Konsequenz daraus ist, dass man prinzipiell nur zeitunabhängige Vergleichsroutinen bei PINs verwenden darf. Ein Beispiel dafür findet sich in Bild 7. Die Abarbeitungszeit dieser Routine ist immer gleich und damit unabhängig bis zu welcher Stelle die übergebene und die Referenz-PIN übereinstimmen. Eine Zeitabhängigkeit zwischen den beiden zu vergleichenden Strings ist damit nicht mehr gegeben.

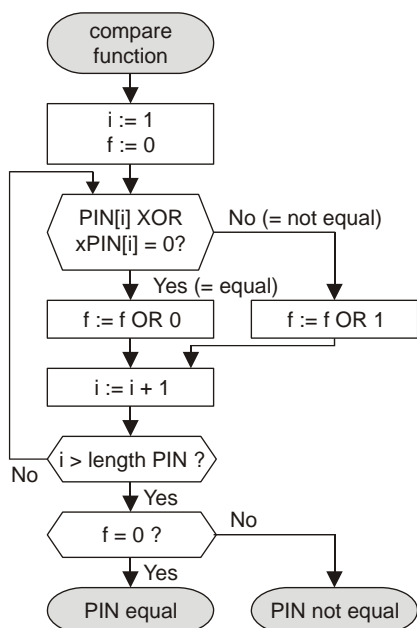


Bild 7 Beispiel für eine sichere Vergleichsroutine, deren Zeitdauer unabhängig von den übereinstimmenden Stellen der vom Benutzer eingegebenen PIN und der in der Chipkarte gespeicherten Referenz-PIN ist. Diese Routine vergleicht immer alle Stellen des Geheimnisses und besitzt zudem zeitlich ausbalancierte Abläufe, so dass sie unabhängig vom Vergleichsergebnis immer die gleiche Zeitspanne für einen Vergleich benötigt. Die folgenden Variablen werden verwendet: i – Zähler für die Anzahl der verglichenen Stellen, f – Variable als Merker des stellenweisen Vergleichs, PIN – Feld in der Chipkarte mit dem gespeicherten Geheimnis (Referenz-PIN), $xPIN$ – Feld mit dem zum Vergleich übergebenen Geheimnis.

Diese verbesserte Routine weist zudem noch die Besonderheit auf, dass der Vergleich nicht mit einem direkten Vergleichsbefehl („=“) gemacht wird, sondern mit einer

XOR-Operation. Dies umgeht elegant das Problem, dass die meisten Prozessoren einen Vergleichsbefehl abhängig vom Ergebnis des Vergleichs mit unterschiedlicher Taktzahl und auch Stromverbrauch abarbeiten.

Angriff und Abwehr: Abschalten der Spannungsversorgung der Chipkarte

Beim Schreib- oder Löschvorgang ins EEPROM oder Flash benötigt der Chipkarten-Mikrocontroller deutlich mehr Strom. Dies kommt dadurch zustande, dass man dafür eine Spannung von knapp 18 Volt benötigt, die von der externen Stromversorgung (typisch 5 Volt oder 3 Volt) nicht bereitgestellt wird. Sie muss deshalb auf dem Chip mit einer Ladungspumpe erzeugt werden. Das Anschalten und der Betrieb dieser Ladungspumpe ist jedoch mit einer Strommessung gut zu sehen, was Bild 8 deutlich zeigt.

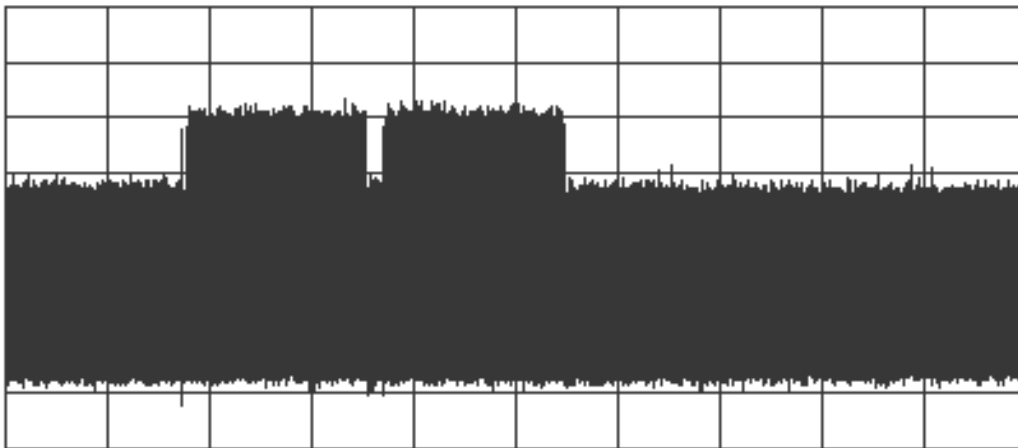


Bild 8 Die X-Achse dieses Diagramm zeigt die Zeit bei einer Dauer von 1 ms pro Kästchen und die Y-Achse die über einen Vorwiderstand gemessene Stromaufnahme des Chipkarten-Mikrocontrollers. Der erste Anstieg ist das Anschalten der Ladungspumpe zum Löschen der Speicherseite und der zweite Anstieg das Schreiben des eigentlichen Wertes in das EEPROM. Im Beispiel wird der Wert 0x55 in eine EEPROM-Zelle geschrieben, die vorher den Wert 0xAA enthielt. Die Dauer des Lösch- bzw. Schreibvorgangs beträgt ca. 2 ms.

Schaltet man nun die Stromversorgung der Chipkarte unmittelbar beim Erkennen eines Stromanstiegs ab oder sendet ein Reset-Signal zur Chipkarte, so findet der Schreib- bzw. Löschvorgang in den nichtflüchtigen Speicher nicht statt. Dieser Sachverhalt lässt sich nun in Kombination mit dem Ablauf der PIN-Prüfung aus Bild 5 für einen Angriff auf die PIN-Vergleichsroutine einsetzen.

Dazu sendet man der Reihe nach alle möglichen PINs zur Chipkarte und misst parallel dazu laufend den aufgenommenen Strom. Sobald ein Stromanstieg für eine Schreib- oder Löschoperation auf den nichtflüchtigen Speicher erkennbar ist schaltet man die Chipkarte ab. Damit kann man zuverlässig das Schreiben des Fehlerzählers unterbinden. Gleichzeitig zeigt dieser Stromanstieg dem Angreifer, dass die geratene PIN falsch ist. Ohne funktionsfähigen Fehlerzähler kann man jedoch alle PINs durchprobieren, was bei einem automatisierten Ansatz weniger als 5 Minuten in Anspruch nimmt. Sobald der Angreifer die korrekte PIN errät, wird der Gutfall im Programmfluss durchlaufen und er erhält von der Chipkarte eine entsprechende positive Rückmeldung (Returncode '9000'). Damit kann der Angreifer weitere Versuche einstellen, da er erfolgreich war.

Dieser Ansatz würde bei unbedarft programmierten PIN-Prüfungen tatsächlich funktionieren. Dies sollte aber seit nahezu 20 Jahren nicht mehr der Fall sein, da es dagegen sehr gute Schutzmassnahmen gibt, die in den folgenden Absätzen vorgestellt werden.

Ein Ansatz diesen Angriff abzuwehren besteht darin, dem Angreifer keine nutzbare Information hinsichtlich des Schreibvorgangs des Fehlerzählers zukommen zu lassen. Dazu wird auch beim erfolgreichen PIN-Vergleich der Fehlerzähler in jedem Fall geschrieben, d.h. er wird auch dann auf null zurückgesetzt wenn er bereits auf null steht und dies eigentlich nicht notwendig wäre. Zusätzlich balanciert man die beiden

Ablaufpfade nach dem Vergleich zeitlich aus, so dass kein zeitlicher Unterschied mehr erkennbar ist. Bild 9 zeigt diesen Ansatz anhand des Kerns der PIN-Prüfung. Eine Zeitanalyse weist dann keine Unterschiede mehr in Abhängigkeit des Ergebnisses des PIN-Vergleichs auf und der oben beschriebene Angriff lässt sich somit abwehren.

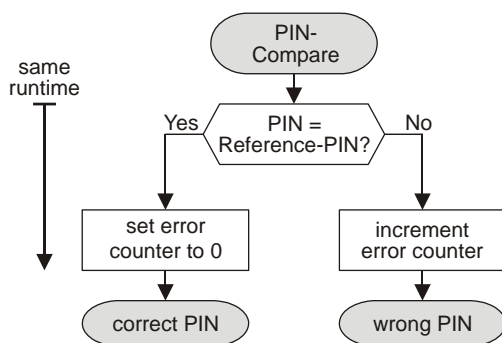


Bild 9 Der Kern des Programmablaufs bei der Ausführung des Chipkarten-Kommandos VERIFY. Die beiden Ablaufpfade nach dem Vergleich bis zur Schreiboperation des Fehlerzählers in den nichtflüchtigen Speicher zeitlich ausbalanciert, so dass von außerhalb der Chipkarte kein Unterschied im Ablauf feststellbar ist.

Der Ablauf aus Bild 9 hat allerdings einen kleinen Schönheitsfehler. Es ist technisch nicht so einfach die beiden Zweige im Programmablauf zeitlich ganz exakt auszubalancieren. Zusätzlich erfordert dies auch noch relativ aufwendige Zeitmessungen zur Prüfung der Implementierung. Deshalb wurde schon recht frühzeitig eine Alternative erdacht, die sich international auch auf breiter Basis durchgesetzt hat. Sie ist in Bild 10 vorgestellt und funktioniert folgendermaßen:

Vor der eigentlichen Vergleichsoperation zwischen der vom Benutzer eingegebenen PIN und der Referenz-PIN der Chipkarte wird der Fehlerzähler in jedem Fall prophylaktisch hoch gezählt. Erst anschließend findet der Vergleich statt und im Fall einer gleichen PIN wird der Fehlerzähler auf null zurück gesetzt und dem Terminal eine entsprechende Meldung geschickt. Im anderen Fall wird ohne weitere Schreiboperation in den nichtflüchtigen Speicher der entsprechende Status an das Terminal zurückgegeben.

Bei diesem Ablauf hat ein Angreifer keine Möglichkeit durch zeitgenaues Abschalten der Chipkarte vor stattfindenden Schreiboperationen einen Vorteil zu erlangen, da der Fehlerzähler bereits vor dem Vergleich erhöht wurde.

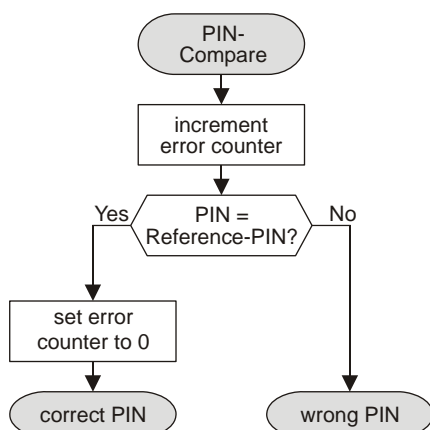


Bild 10 Der Kern eines sicheren Programmablaufs bei der Ausführung des Chipkarten-Kommandos VERIFY. Vor der eigentlichen Vergleichsroutine wird dabei der Fehlerzähler erhöht und nur bei positiven Vergleichsergebnis auf null zurückgesetzt.

Technisch gesehen hat diese einfache, robuste und sichere Lösung nur einen klitzekleinen Pferdefuß. Es wird deutlich öfter in den nichtflüchtigen Speicher geschrieben als notwendig, was sich nachteilig auf die Lebensdauer auswirkt. Da jedoch die Anzahl der PIN-Eingaben im normalen Einsatzfall selbst über Jahre hinweg nicht allzu hoch ist und moderne nichtflüchtige Speicher durchaus 500 000 Schreib-/Löschzyklen überstehen können ist dies kein in der Praxis relevantes Problem.

Angriff und Abwehr: Störangriff auf den PIN-Vergleich

Die PIN-Vergleichsroutine wie in Bild 10 gezeigt galt für viele Jahre als die perfekte Lösung. Allerdings wurde etwa um die Jahrtausendwende bekannt (siehe dazu beispielsweise [Skorobogatov 02]), dass man mit Lichtblitzen, Spikes auf der Versorgungsleitung oder energiereichen Strahlen in ungünstigen Fällen bestimmte Abfragen im Programmablauf verändern kann. Dazu ist neben vielen anderen Aspekten auch eine genaue zeitliche Synchronisation mit der Abarbeitung der Maschinenbefehle durch den Prozessor der Chipkarte notwendig.

Ist jedoch diese Synchronisation gegeben, dann kann man in gewissen Fällen beispielsweise durch einen energiereichen fokussierten Lichtblitz an eine bestimmte Stelle auf der Chipoberfläche eine bedingte Verzweigung ändern.

Aber auch gegen diesen mächtigen Angriff gibt es wirksame Gegenmaßnahmen, die zwar den Programmcode etwas komplizierter machen, aber dafür auch ganz gut schützen. Ein Beispiel dazu zeigt Bild 11. Als Schutzmassnahme eignet sich beispielsweise eine Wartezeit zufälliger Dauer vor dem eigentlichen Vergleich der beiden PINs. Ein Angreifer kann damit seine Störquelle nicht mehr mit der Vergleichsoperation synchronisieren und auch nicht durch simples ausprobieren den richtigen Zeitpunkt herausfinden, da dieser bei jedem Vergleich der PINs ein anderer ist.

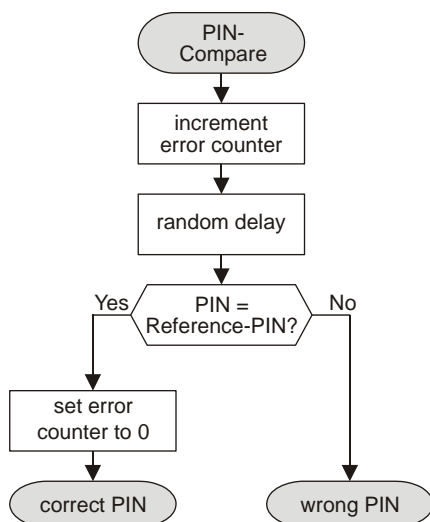


Bild 11 Der Kern eines sicheren Programmablaufs bei der Ausführung des Chipkarten-Kommandos VERIFY. Hier wird vor der eigentlichen Vergleichsroutine eine zufällige Wartezeit eingefügt um eine zeitgenaue Synchronisation von äußeren Störquellen zu verhindern.

Diese Schutzmassnahme lässt sich noch weiter verbessern indem man die „Random-Wait-States“ des Mikrocontrollers aktiviert. Diese spezielle Hardwarefunktion fügt zufallsgesteuert Wartezyklen in den normalen Programmablauf ein. Dies für sich alleine würde bereits die für den beschriebenen Angriff notwendige taktgenaue Synchronisation verhindern, doch zusammen mit der zufälligen Wartezeit vor der Vergleichsoperation unterbindet es zuverlässig jegliche Synchronisation zur Aktivierung einer Störquelle.

Ein kleiner Einschub ist an dieser Stelle angebracht. Natürlich können moderne Chipkarten-Mikrocontroller die Verwendung von äußeren Störquellen wie Lichtblitze oder ähnliches erkennen und entsprechend reagieren. Dies kann beispielsweise eine Vielzahl von Lichtdetektoren auf der Chipoberfläche sein oder auch ein Chipkarten-Mikrocontroller mit zwei parallel arbeitenden Prozessoren unterschiedlicher Implementierung. Die hier vorgestellten Abwehrmaßnahmen setzten deshalb bereits auf ein sehr hohes Sicherheitsniveau auf. Allerdings verlässt man sich in diesem Metier grundsätzlich nicht nur auf eine einzelne Schutzmassnahme.

Ein Dilemma all dieser softwaretechnischen Schutzmassnahmen vor Angriffen ist natürlich die zunehmende Kompliziertheit im Programmcode, was unter anderem die Gefahr von Programmierfehlern gegenüber den einfachen Lösungen ansteigen lässt. Die erforderliche Codequalität kann man neben den üblichen umfangreichen Tests nach dem

Vier-Augen-Prinzip zusätzlich mit entsprechenden Reviews oder auch einer Evaluierung nach Common Criteria sicherstellen.

Angriff und Abwehr: Verschlüsselte Speicherung von Geheimnissen auf der Chipkarte

Die wichtigste Eigenschaft von Chipkarten ist bekanntermaßen, dass die gespeicherten Daten nicht unbefugt von außen gelesen werden können. Die logische Konsequenz daraus ist, dass es eigentlich überhaupt nicht notwendig ist, vertrauliche Daten in der Chipkarte verschlüsselt zu speichern. Die Referenz-PIN in der entsprechenden Datei (EF_{PIN}) kann deshalb auch unverschlüsselt im Klartext abgelegt sein. Den Schutz, dass die Referenz-PIN nicht unbefugt gelesen werden kann übernimmt die Hardware und das Betriebssystem der Chipkarte.

Im Sicherheitsbereich sollte man sich aber prinzipiell nie auf einen oder wenige Schutzmechanismen verlassen, sondern immer auch Vorkehrungen für den Fall treffen, wenn diese versagen sollten. So ist es zumindest denkbar, dass ein Angreifer den Zugriffsschutz auf die Datei mit der Referenz-PIN aushebelt. Ein möglicher Ansatz wären hier Fehlereinstreuungen durch Lichtblitze, obwohl moderne Chipkarten-Betriebssysteme und deren Mikrocontroller dagegen gehärtet sind.

Unter der Annahme, dass ein unbefugter lesender Zugriff auf die Referenz-PIN möglich wäre, ist es sinnvoll hier noch eine weitere Sicherheitsmassnahme vorzusehen. Dazu verschlüsselt man alle in der Chipkarte gespeicherten Geheimnisse. Klugerweise benutzt man dazu einen Schlüssel, der getrennt von den Geheimnissen, am besten noch in einer anderen Speicherart auf dem Chipkarten-Mikrocontroller abgelegt ist. Das Chipkarten-Betriebssystem hat nun die Aufgabe vor der Ausführung des eigentlichen PIN-Vergleichs zuerst die Referenz-PIN zu entschlüsseln und erst dann zu vergleichen.

Auf der Chipkarte verschlüsselte Geheimnisse schützen sicher davor, falls es ein Angreifer doch irgendwie schafft alle anderen Schutzmassnahmen zu umgehen und lesenden Zugriff auf in der Chipkarte gespeicherte Geheimnisse zu erlangen. Deshalb ist diese Abwehrmaßnahme mittlerweile weit verbreitet.

An dieser Stelle sei noch eine kleine Ergänzung angebracht. Diese Gegenmaßnahme ist recht wirkungsvoll, kann jedoch bei ungeschickter Implementierung in der Chipkarte durchaus ein Sicherheitsloch herbeiführen. Die Referenz-PIN befindet sich nach dem entschlüsseln nicht mehr im nichtflüchtigen Speicher, sondern im RAM. Dort wird sie dann mit der vom Terminal gesendeten und im Empfangspuffer befindlichen PIN verglichen. Sollte es nun einem Angreifer gelingen einen Pufferüberlauf der Senderoutine bei der Antwort der Chipkarte auf das VERIFY-Kommando zu erzeugen, dann könnte es im ungünstigsten Fall dazu führen, dass neben der beabsichtigten Antwort auch die Referenz-PIN an das Terminal gesendet wird. Dies ist einer der Gründe dafür, warum die Senderoutine einer Chipkarte gegen Angriffe ähnlich gehärtet sein muss wie die Vergleichsroutine für PINs. Allerdings ist es auch angeraten prinzipiell alle Geheimnisse unmittelbar nach Verwendung aus dem RAM wieder zu löschen. Damit würden selbst bei einem Pufferüberlauf keine Geheimnisse nach außen gesendet.

Angriff und Abwehr: Vergleichende Strommessung beim PIN-Vergleich

Der Seitenkanal des Stromverbrauchs von Chipkarten wird seit langem [Kocher 98 a] für unterschiedliche Angriffe wie SPA/DPA (*single power analysis/differential power analysis*) genutzt. Eine Variante davon ist die vergleichende Messung von zwei Chipkarten beim PIN-Vergleich, was auch Template-Angriff genannt wird. Dazu sind jedoch einige Voraussetzungen zu erfüllen. Man benötigt zwei Chipkarten mit dem gleichen Mikrocontroller und dem gleichen Betriebssystem. Zusätzlich muss die PIN auf einer der Karten frei änderbar sein. Diese Voraussetzungen klingen schwieriger als sie in der Praxis wirklich sind, da von den Kartenemittenten zu einem bestimmten Zeitpunkt nicht allzu viele unterschiedliche Chiptypen inklusive Betriebssystem ausgegeben werden.

Das Angriffsszenario gestaltet sich nun folgendermaßen: Die erste Chipkarte, deren PIN bekannt und beliebig änderbar ist, verwendet man um für alle möglichen PINs (d.h. „0000“ bis „9999“) jeweils ein Stromreferenzmuster zu ermitteln. Dies kann man auch mit überschaubarem Aufwand automatisieren, so dass es sich in kurzer Zeit durchführen lässt. Die gemessenen Stromsignaturen sind abhängig vom Mikrocontroller, dem Betriebssystem und natürlich von den verglichenen PINs. Die Zeitspanne der Messung

kann dabei beispielsweise die Kopierfunktion umfassen, die die Referenz-PIN vom nichtflüchtigen Speicher (EEPROM, Flash) zum darauf folgenden Vergleich ins RAM kopiert.

Nun sendet man eine beliebige PIN an die zweite Chipkarte mit der unbekanntem PIN und misst dabei mit genügend hoher Auflösung den Stromverbrauch während des gleichen Zeitraums wie bei der Referenzkarte. Das dabei gemessene Strommuster vergleicht man anschließend mit den Stromsignaturen der ersten Karte und ermittelt daraus durch einfachen Vergleich die unbekanntem PIN.

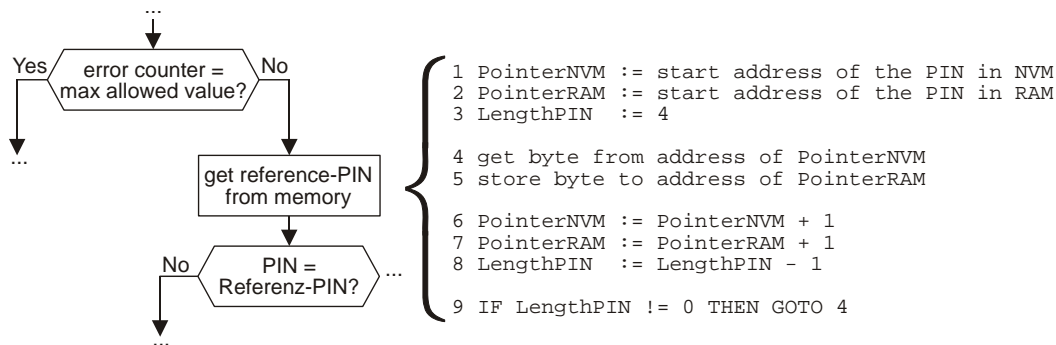


Bild 12 Vereinfachtes Prinzip der Ermittlung einer unbekanntem PIN mit 4 Byte Länge durch vergleichende Messung des Stromverbrauchs. In diesem Beispiel wird die Strommessung beim Kopieren der Referenz-PIN vom EEPROM in das RAM gewählt, da dabei der Stromverbrauch des Prozessors bei der Abarbeitung dieses Programmteils ausschließlich von den zu verarbeitenden Daten abhängt. Der Gesamtzusammenhang ist in Bild 5 zu sehen.

Zu diesem Angriff existieren eine Reihe unterschiedlicher Abwehrmaßnahmen. Verhältnismäßig einfach ist die Einfügung von zufallsgesteuerten Wartezeiten (z.B. *random wait states* des Prozessors) während der gesamten Kommandoabarbeitung. Da der Kopiervorgang dann nicht mehr zu einem festen Zeitpunkt nach Empfang des VERIFY-Kommandos stattfindet erschwert dies die Synchronisation der Strommessung erheblich. Eine weitere sinnvolle Maßnahme ist es, die in der Chipkarte gespeicherte PIN nicht nur mit einem einheitlichen Schlüssel zu verschlüsseln, sondern mit einem kartenindividuellen Schlüssel. Damit ist die PIN unabhängig von ihrem eigentlichen Wert immer kartenindividuell und der Stromvergleich beim Kopiervorgang läuft ins Leere, da jede Karte damit eine eigene Stromsignatur bekommt. Die von außen übergebene PIN muss dann vom Betriebssystem vor dem Vergleichsvorgang selbstverständlich ebenfalls mit demselben kartenindividuellen Schlüssel verschlüsselt werden.

Die gezeigten Angriffe und dazugehörigen Gegenmaßnahmen zeigen die wesentlichen Entwicklungsschritte der letzten zwei Jahrzehnte hinsichtlich des sicheren PIN-Vergleichs. Selbstverständlich weisen moderne Chipkarten-Betriebssysteme noch zahlreiche weitere Schutzmaßnahmen auf, die teilweise auch eng mit der Chiphardware verknüpft sind. Die hier gezeigten Mechanismen sollen einen Überblick und auch einige Besonderheiten zeigen und vor allem auch das Bewusstsein schaffen, dass ein sicheres und robustes Chipkarten-Betriebssystem auch einen erheblichen Erfahrungsschatz der Entwickler widerspiegelt.

Die Beispiele zeigen auch, dass bei einem Angriff nicht immer der direkte Weg zum Ziel führen kann. Wie dargestellt kann man versuchen unmittelbar eine Schwachstelle bei der eigentlichen Vergleichsroutine auszunutzen wie etwa das unterschiedliche Laufzeitverhalten beim eigentlichen Vergleich. Alternativ dazu ist es aber auch denkbar, dass ein Angreifer probiert, an der Integrationsschnittstelle der Vergleichsroutine zum Betriebssystem einen Vorteil für sich herauszuholen. Eine weitere Möglichkeit ist natürlich, einen ganz anderen Weg zu gehen und zu ausprobieren, die geheimen Daten über eine Schwachstelle des Chipkarten-Betriebssystems auszulesen.

Ein gutes Chipkarten-Betriebssystem schützt die ihm anvertrauten Geheimnisse gegen alle diese Angriffe und erfüllt zusätzlich auch noch die vielen anderen Kriterien, welche ebenfalls für einen Erfolg wichtig sind, wie beispielsweise geringer Speicherverbrauch, kurze Antwortzeiten und einfache Portabilität.

Resümee

Es ist eine Tatsache, dass sich perfekte Sicherheit eigentlich nie erreichen lässt. Allerdings sollte man in jedem Fall anstreben den Aufwand für die potentiellen Angreifer so hoch wie möglich zu treiben, um so die Angriffattraktivität niedrig zu halten. Damit kann man bereits im Vorfeld manche Angreifer abschrecken.

Ausschlaggebend ist auch, dass das gesamte System sicher ist – eine Komponente alleine reicht nicht aus. So lassen sich die oben beschriebenen Schutzmassnahmen für die PIN-Prüfung ganz leicht aushebeln, wenn beispielsweise das Terminal die PIN zwischengespeichert. Dann kann man diese nämlich anschließend mit einer entsprechend modifizierten Chipkarte abgreifen. Ein Fehler, der im wahren Leben nicht nur einmal vorgekommen ist! Wie eingangs beschrieben, es verhält sich hierbei wie mit einer Kette, sie hält nur, wenn alle Glieder gleichermaßen stabil bleiben.

Hinsichtlich des Systems und seiner Komponenten seien hier am Schluss auch einige Hinweise angebracht, die für die Praxis ganz nützlich sind. Man sollte grundsätzlich in einem System nur Mechanismen einsetzen, die man auch versteht und die auch eine gewisse Einfachheit aufweisen. Komplizierten Schutzmassnahmen, deren Funktionsweise nicht klar verständlich ist sollte man immer ein erhebliches Misstrauen entgegenbringen.

Weiterhin darf man nicht den Fehler machen, sich auf einzelne Schutzmassnahmen vollständig zu verlassen. Es könnte immer ein Fall eintreten, beispielsweise durch einen neuen noch unbekanntem Angriff, in dem die vorgesehene Maßnahme nicht greift. In genau diesen Situationen ist es wichtig, zusätzliche Vorsorgen getroffen zu haben, damit das System nicht diskreditiert werden kann.

Der letzte Hinweis sollte eigentlich selbstverständlich sein, wird aber trotzdem immer wieder missachtet. Sowohl bei dem Chipkarten-Mikrocontroller, beim Betriebssystem, der darauf aufsetzenden Anwendung und beim Gesamtsystem muss man auf eine professionelle und klare Architektur und Ausführung achten. Dazu gehört auch, dass dies von kompetenten Experten mit dem entsprechenden Erfahrungsschatz erstellt wird. Dies sind die wichtigsten Voraussetzungen um ein sicheres System zu erstellen und auch zu betreiben, wobei aus meiner ganz persönlichen Sicht, das wichtigste Element darin natürlich die Chipkarte ist.

Anhang

- [Biham 96] Eli Biham, Adi Shamir: A new cryptanalytic attack on DES, Internet, 1996
- [Boneh 96] Dan Boneh, Richard A. DeMillo, Richard J. Lipton: On the Importance of Checking Computations, Internet, 1996
- [Gandolfi 01] Karine Gandolfi, Christophe Mourtel, Francis Oliver: Electromagnetic Analysis: Concrete Results, Workshop CHES 2001, 2001
- [Kocher 95] Paul C. Kocher: Timing Attacks on Implementations of Diffie-Hellmann, RSA, DSS, and Other Systems, Internet, 1995
- [Kocher 98 a] Paul C. Kocher, Joshua Jaffe, Benjamin Jun: Introduction to Differential Power Analysis and Related Attacks, Internet, 1998
- [Kömmerling 99] Oliver Kömmerling, Markus G. Kuhn, Design Principles for Tamper-Resistant Smartcard Processors, USENIX Workshop on Smartcard Technology, Chicago USA, 10–11 Mai 1999
- [Lamla 00] Michael Lamla: Hardware Attacks on Smart Cards - Overview, Eurosmart Security Conference, Marseille, 13–15 Juni 2000
- [Rankl 09] Wolfgang Rankl, Wolfgang Effing: Handbuch der Chipkarten, 5. Auflage, Carl Hanser Verlag, 2008
- [Skorobogatov 02] Sergei Skorobogatov, Ross Anderson: Optical Fault Induction Attacks, Internet, Mai 2002