(19) österreichisches patentamt

(10) **AT 508 649 A2 2011-03-15**

(12) Österreichische Patentanmeldung

(21) Anmeldenummer: **A 1166/2010**
(22) Anmeldetag: **09.07.2010**
(43) Veröffentlicht am: **15.03.2011**

(51) Int. Cl.: ***G06F 21/22*** (2006.01),
G06F 9/44 (2006.01),
G06F 12/14 (2006.01),
G06F 11/07 (2006.01)

(30) Priorität:

15.07.2009 DE 102009033211
beansprucht.

(73) Patentinhaber:

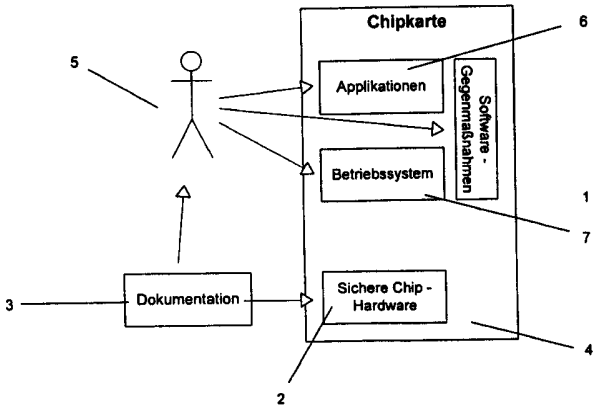
AUSTRIA CARD GMBH
A-1230 WIEN (AT)

(72) Erfinder:

AICHINGER THOMAS
WIEN (AT)

(54) CHIPKARTE MIT ÜBERWACHUNG DER INTEGRITÄT AUF SOFTWAREBASIS

(57) Chipkarte mit Maßnahmen zur Überwachung der Datenintegrität und Maßnahmen zur Gewährleistung der Ablaufintegrität auf Softwarebasis, wobei diese Maßnahmen automatisch durch ein Software-Modul erzeugt werden, wobei diese durch Parameter gesteuert wird und in eine oder alle Komponenten der Toolchain integriert ist oder eingreift.



AT 508 649 A2 2011-03-15

Zusammenfassung

Chipkarte mit Maßnahmen zur Überwachung der Datenintegrität und Maßnahmen zur Gewährleistung der Ablaufintegrität auf Softwarebasis, wobei diese Maßnahmen automatisch durch ein Software-Modul erzeugt werden, wobei diese durch Parameter gesteuert wird und in eine oder alle Komponenten der Toolchain integriert ist oder eingreift.

Chipkarte mit Überwachung der Integrität auf Softwarebasis

Gegenstand der Erfindung ist eine Chipkarte mit Maßnahmen zur Überwachung der Datenintegrität der auf der Chipkarte gespeicherten Daten und Maßnahmen zur Gewährleistung der Ablaufintegrität, wobei diese Maßnahmen in der auf der Chipkarte ablaufenden Software (Betriebssystem und Applikationen) realisiert sind und automatisch bei der Softwareerstellung erzeugt werden.

Erfindungsgemäß ist vorgesehen, dass auf der in der Chipkarte implementierten Software (Betriebssystem und Applikationen) Gegenmaßnahmen eingebaut werden müssen, um die Chipkarte gegen äußere Angriffe zu schützen.

Solche Angriffe werden Penetrationsversuche genannt, wo im konkreten Fall der Chip der Chipkarte physikalischen und/oder chemischen Prozessen ausgesetzt wird und hierbei versucht wird, gezielt Fehler auf der Karte zu erzeugen (im Folgenden „Fehlerinduktion" auch bekannt unter dem Begriff „fault induction"). Bekannt sind unter anderem Angriffe mit Hilfe von Laserstrahlen, Licht, ionisierender Strahlung und mit Spannungs- und Frequenzveränderungen sowie Stromimpulsen an den Anschlüssen des Chips, sowie physikalische Manipulationen mittels sehr feinen Sonden / Kontakten und chemische Manipulationen.

Um solchen Angriffen zu entgegnen ist es im Stand der Technik bekannt, dass man bei der Programmierung der Chipkartensoftware zur Sicherung der Datenintegrität Daten, Adresswerte und Rücksprungadressen mit Prüfsummen oder redundanten Daten versieht und dazu benutzt die Datenintegrität vor und oder nach der Datenverwendung zu überprüfen. Bezüglich der Ablaufintegrität besteht ein weiteres bekanntes Verfahren darin, zum Beispiel die Ausführung von Codes dadurch abzusichern, dass die Abfrage von Bedingungen doppelt (ggf. in komplementärer Weise) ausgeführt wird oder bestimmte Code-Abschnitte redundant vorhanden sind. Eine weitere Möglichkeit besteht darin die Ablaufintegrität dadurch sicherzustellen, dass logische Programmabfolgen (z.B. Unterprogrammaufrufe) durch die Weitergabe zusätzlicher redundanter Informationen auf ihre logische Korrektheit geprüft werden. Weiters ist auch bekannt, dass Daten zunächst geschrieben werden, danach zurück gelesen werden und dann verglichen wird, ob die Daten richtig geschrieben wurden

oder nach mathematischen Operationen zusätzlich die inversen Operationen durchgeführt und die Ergebnisse überprüft werden. Weiters kann durch die Dokumentation der Chip-Hardware vorgegeben sein, welche Hardware-Gegenmaßnahmen in bestimmten Fällen zu Aktivieren oder zu Deaktivieren sind.

Die Grundidee der vorliegenden Erfindung besteht nun darin, dass man den vorher beschriebenen Sicherungsprozess, gemeint ist die Implementierung der Gegenmaßnahmen in die Chipkartensoftware, automatisiert und durch Parameter steuerbar macht.

Bisher war es bei der Programmierung einer solchen Software notwendig auf den speziellen Programmablauf abgestimmte - Sicherungsmechanismen (Gegenmaßnahmen) – sozusagen „von Hand" - einzubauen.

Weiterer Nachteil ist, dass bei einer umfangreichen Programmierung von Chipkartensoftware, wie es beim Stand der Technik der Fall ist, ein außerordentlich hoher Programmieraufwand entsteht, dessen Prüfung schwierig ist, der fehleranfällig ist und der einem großen Entwicklungsaufwand führt. Durch die implementierten Gegenmaßnahmen steigen weiters der Speicherbedarf und die Berechnungszeit der Software.

Der Erfindung liegt deshalb die Aufgabe zugrunde eine Chipkarte der eingangs genannten Art mit einer Absicherung der Software über Software-Gegenmaßnahmen so weiter zu bilden, dass die Implementierung der Gegenmaßnahmen in der Chipkartensoftware automatisiert und über Parameter steuerbar erfolgt.

Zur Lösung der gestellten Aufgabe ist die Erfindung dadurch gekennzeichnet, dass die Gegenmaßnahmen durch ein Software-Modul erzeugt werden, welches in die Toolchain eingreift oder integriert ist.

Die Erfindung sieht deshalb eine Chipkarte mit Maßnahmen zur Überwachung der Datenintegrität und Maßnahmen zur Gewährleistung der Ablaufintegrität auf Softwarebasis (Software – Gegenmaßnahmen vor, wobei diese Maßnahmen in der

auf der Chipkarte ablaufenden Software und Applikationen realisiert sind, um die Chipkarte gegen äußere Angriffe zu schützen.

Solche äußeren Angriffe auf die Chipkarte sind insbesondere

- Angriffe mit Hilfe von
  - Elektromagnetischen Wellen (Laserstrahlen, Licht, ionisierender Strahlung)
  - Spannungs- und Frequenzveränderungen an den Kontakten des Chips
  - physikalische Manipulationen des Chips mittels sehr feine Sonden / Kontakten
  - chemische Manipulationen

Das Resultat eines solchen Angriffes ist im Erfolgsfall, dass

- der Inhalt einer Speicherzelle (im flüchtigen oder nicht-flüchtigen Speicherbereich) geändert wird oder
- der Inhalt eines Prozessor-Registers (eines Haupt- oder Co-Prozessors oder ein spezielles Hardware-Register) geändert wird. Handelt es sich hierbei beispielsweise um den so genannten Programm Counter, welcher den als nächsten auszuführenden Befehl bezeichnet, so wird das Programm an einer anderen Stelle fortgesetzt.

Eine solche Veränderung des Inhaltes kann grundsätzlich persistent oder transient sein. Folgende Fälle sind zu unterscheiden:

- persistente Veränderung im nicht flüchtigen Speicher: alle nachfolgenden Lese-Zugriffe liefern den geänderten Wert zurück.
- Persistente Veränderung im flüchtigen Speicher: alle nachfolgenden Lese-Zugriffe bis zum nächsten Reset des Chips liefern den veränderten Wert zurück (nach dem Reset ist der Inhalt des flüchtigen Speichers im Allgemeinen undefiniert).
- Persistente Veränderung eines Prozessor-Registers: alle nachfolgenden Lese-Zugriffe bis zu dem Zeitpunkt an dem ein neuer Wert in das Register geschrieben wird (das kann auch selbsttätig von der Chip-Hardware verursacht werden), liefern den geänderten Wert zurück

- Transiente Veränderung einer Speicherzelle oder eines Registers: nur der aktuelle Lese-Zugriff auf die Speicherzelle oder ein Register liefert einen geänderten Wert zurück. Alle nachfolgenden Lese-Zugriffe liefern wieder den korrekten Wert zurück. Dieser Angriff funktioniert im Allgemeinen nur dann, wenn der Angriff exakt zum Zeitpunkt des Lese-Zugriffs ausgeführt wird. Es handelt sich hier um den Angriff mit der derzeit höchsten – da auf realer Hardware am einfachsten ausführbaren - Erfolgsrate.

Die derzeit bekannten Angriffe auf reale Hardware haben gemeinsam, dass das Setzen einer Speicherzelle auf einen bestimmten Wert im Allgemeinen sehr schwierig ist, sondern ein zumeist ein – von außen für den Angreifer aufgrund von Speicherverschlüsselungsmechanismen – nicht vorhersehbares Bit-Muster gesetzt wird oder alle Bits der zu verändernden Speicherzelle auf den Wert Null oder alle Bits auf den Wert Eins gesetzt werden.

Erfindungsgemäß werden die Gegenmaßnahmen (redundante Daten und Prüfroutinen) von einem Software-Modul erzeugt, insbesondere folgende Gegenmaßnahmen

- Versehen der Daten, Adresswerte und Rücksprungadressen mit Prüfsummen oder redundanten Daten und deren Verarbeitung durch Prüfroutinen
- Einbringen von redundanten Codeabschnitten (z.B. doppeltes und ggf. komplementäres Abfragen von Bedingungen, Weitergabe redundanter Information bei Unterprogrammaufrufen) zur Sicherung der Ablaufintegrität logischer Programmabfolgen
- Einbringen von zusätzlichem Code und Daten zur Erkennung von unbeabsichtigten Sprüngen (z.B. von einem Angreifer induziert) innerhalb des Programmablaufes.
- Das Schreiben und nachfolgende Zurücklesen von Daten
- Durchführen von mathematisch inversen Operationen und Kontrolle der Ergebnisse
- Aktivieren und Deaktivieren von Hardware-Gegenmaßnahmen gemäß den Vorgaben in der Dokumentation der Chip-Hardware

Mit der Verwendung eines Software-Moduls besteht der Vorteil, dass dieses automatisch (selbsttätig) arbeitet und der Benutzereingriff nur beim Start und möglicherweise bei der Auslesung und Auswertung der Ergebnisse notwendig ist. Ein solches Software-Modul ist vorteilhaft in die sogenannte Toolchain integriert.

Die Toolchain (deutsch: Werkzeugkette) ist ein Gesamtbegriff für die Programmier- und insbesondere Übersetzungswerkzeuge, die in einem Projekt eingesetzt werden um den Source-Code zu Erstellen und in den ablauffähigen Binärcode zu übersetzen. Diese Produkte bilden ein Gesamtsystem oder eine Werkzeugkette, die für die Programmierung von sowohl Anwendungen als auch Betriebssystemen genutzt werden. Der Begriff Toolchain ist ein wichtiger Bestandteil bei der Entwicklung des Linux Kernels, der Entwicklung der BSD und ein Standardtool bei der Entwicklung von Eingebetteten Systemen.

Demzufolge versteht man unter einer Toolchain (Toolkette) die gesamte Kette von Softwareentwicklungswerkzeugen, beginnend von der Programmerstellung und Quellkodeverwaltung über den Pre- Prozessor, Compiler, Assembler und den Linker bis hin zum Simulator und Emulator. In manchen Fällen werden die Werkzeuge für die Anforderungs- und Änderungsverwaltung ebenfalls zur Toolchain gezählt.

Erfindungsgemäß ist nun vorgesehen, dass man das Softwaremodul zur automatischen Generierung der Gegenmaßnahmen in diese Toolchain einfügt.

Es bestehen bei der vorliegenden Erfindung also wesentliche Vorteile gegenüber der manuellen Programmierung der Gegenmaßnahmen.
Die Erfindung vermeidet diese aufwendige Programmierung und sieht stattdessen ein Software-Modul vor, welches in die Toolchain (in dem Bereich zwischen dem Source-Code und dem resultierenden Maschinencode) dazwischen geschaltet ist.

Hier bestehen besondere Vorteile, denn der Hardware-Hersteller legt dem Verwender der Chipkarte (Softwareentwickler) besondere Sicherungsmaßnahmen auf, wie beispielsweise das Aktivieren und Benutzen von Schutzfunktionen, die in der Chip-Hardware eingebaut sind sowie von Software- Sicherungsmaßnahmen.

Hier setzt die Erfindung ein, die vorsieht, dass die Software- Sicherheitsfunktionalität automatisch und über Parameter steuerbar in der Toolchain generiert wird.

Es handelt sich also um eine Software-Anwendung, welche in die einzelnen Elemente der Toolchain (insbesondere Pre-Prozessor, Compiler, Linker, Assembler) integriert ist, d.h. um eine spezielle Ausführung der Toolchain bzw. der Compiler-Software.

Das Software-Modul zur Erzeugung der Gegenmaßnahmen wird durch Parameter gesteuert, welche die Art und den Umfang der Gegenmaßnahmen definieren, insbesondere

- Welche Arten von Daten durch redundante Daten abgesichert werden sollen (z.B. Daten im RAM oder EEPROM, Adresswerte, Registerinhalte)
- Wie die redundanten Daten gestaltet sein sollen (z.B. Art der zu benutzenden Prüfsummen)
- Welche Arten von Prüfroutinen erzeugt werden sollen (Erstellen und Prüfen von Prüfsummen vor/nach der Datenverwendung, doppeltes Abfragen von Bedingungen ggf. in komplementärer Weise, redundante Code-Abschnitte, Weitergabe redundanter Information zur Kontrolle der logischen Programmablauffolgen, Schreiben und nachfolgendes Lesen und Vergleichen von Daten, Durchführen von mathematisch inversen Operationen)

Wie viel Prüfroutinen eingebracht werden sollen (absolute oder relative Maßzahl der Codegröße

Das folgende Beispiel illustriert die Vorgehensweise:
Wenn in der Smartcard-Software z. B. ein Speicherbereich angelegt wird und es handelt sich dabei um einen kritischen Speicherbereich, wie z. B. einen Schlüssel, dann wird üblicherweise vom Entwickler gefordert, dazu auch eine Prüfsumme abzuspeichern. Der Entwickler muss die Berechnung der Prüfsumme programmieren, muss einen Speicherbereich für diese Prüfsumme reservieren und muss dann auch an bestimmten Stellen des von ihm entwickelten Programms die Prüfsumme wieder prüfen. Die Erfindung besteht nun darin, dass dieser Schritt, das Berechnen der Prüfsumme, das Reservieren des Speicherplatzes, das Abspeichern und das nachträgliche Wiederprüfen automatisch in diesen Software-Code

automatisch durch die Toolchain eingefügt, wird und damit für den Entwickler auch weitgehend transparent abläuft.

Zusätzlich kann nach einer Weiterbildung das Softwaremodul dynamische Parameter erzeugen, welche zur Laufzeit des Chipkartenbetriebssystem und der Applikationen vorgeben, wie ausführlich und wie zeitaufwendig die Gegenmaßnahmen auszuführen sind.

Aus dem beigefügten Blockschaltbild ergibt sich, dass die Sicherheit einer Chipkarte durch das Zusammenspiel sowohl der sicheren Chip-Hardware als auch der sicheren Software (Applikationen, Betriebssystem und Software-Gegenmaßnahmen, wie von der Dokumentation gefordert) erzeugt werden muss.

Der Erfindungsgegenstand der vorliegenden Erfindung ergibt sich nicht nur aus dem Gegenstand der einzelnen Patentansprüche, sondern auch aus der Kombination der einzelnen Patentansprüche untereinander.

Alle in den Unterlagen, einschließlich der Zusammenfassung offenbarten Angaben und Merkmale, insbesondere die in den Zeichnungen dargestellte räumliche Ausbildung, werden als erfindungswesentlich beansprucht, soweit sie einzeln oder in Kombination gegenüber dem Stand der Technik neu sind.

Es zeigen:

Figur 1: Blockschaltbild mit allgemeiner Darstellung des Zusammenspiels von Software-Entwicklung, sicherer Chip-Hardware und Software Gegenmaßnahmen in einer Chipkarte.
Figur 2: Implementierung des Sicherheitscodes in der Toolchain.

Die Chipkarte 4 besteht im Prinzip aus einer sicheren Chip-Hardware 2, Applikationen 6, einem Betriebssystem 7 und Software Gegenmaßnahmen 1. Die zur sicheren Chip-Hardware gehörige Dokumentation 3 schreibt vor, welche Software – Gegenmaßnahmen 1 zu implementieren sind, um mir dieser sichere Chip-Hardware 2 ein bestimmtes Produkt (Chipkarte 4) in sicherer Weise umzusetzen.

Der Programmierer 5 implementiert Applikationen 6, Betriebssystem 7 und die Software – Gegenmaßnahmen 1 gemäß den Vorgaben der Dokumentation 3 für die sichere Chip-Hardware 2.

Erfindungsgemäß soll nun die auf der Chipkarte ablaufende Software (Betriebssystem 7 und Applikationen 6) durch eine automatisch Erzeugung von Gegenmaßnahmen 15 in der Toolchain 8 abgesichert werden, ohne dass es der manuellen Einfügung von bestimmten Gegenmaßnahmen bedarf.

Der Source-Code 9, der vom Entwickler 5 verfasst wurde, wird durch die Toolchain 8, die z. B. aus einem Preprozessor 10, einem Compiler 11, einem Assembler 12 und einem Linker 13 besteht in den Maschinencode 14 umgewandelt.

Das Wichtige der vorliegenden Erfindung ist nun in der zweiten Abbildung, Figur 2 dargestellt, dass herkömmlich vom Programmierer 5 verlangt wird, Software Gegenmaßnahmen gemäß Dokumentation 3 zu implementieren, beispielsweise bei Datenelementen redundante Daten zu definieren und ferner Prüfroutinen, um beispielsweise diese redundanten Daten auf Integrität zu prüfen oder Maßnahmen, welche die Ablaufintegrität sicherstellen.

Erfindungsgemäß soll dies nun vermieden werden und stattdessen sollen die Erzeugung der Gegenmaßnahmen 15 (z.B. redundanten Daten und die Prüfroutinen) in der den Sourcecode verarbeitenden Software (Toolchain 8) unter Verwendung des Softwaremoduls 17 erzeugt werden.

Damit wird der Programmieraufwand wesentlich verringert, denn es ist nicht mehr notwendig, im Source-Code 9 die Erzeugung der Gegenmaßnahmen 15 durch den Programmierer 5 durchzuführen, sondern dies wird nachfolgend in der nachgeschalteten Toolchain 8 zusammen mit dem Softwaremodul 15 gesteuert durch die Parameter 18 durchgeführt.

Die Gegenmaßnahmen werden somit mit wesentlich geringerem Entwicklungsaufwand generiert, wobei auch die Fehlerträchtigkeit minimiert wird.

Das Software-Modul (17) zur Erzeugung der Gegenmaßnahmen (15) ist vorteilhaft in die Toolchain (Bereich zwischen dem Source-Code (9), dem Preprozessor (10), dem Compiler (11), dem Assembler (12), dem Linker (13)) und dem ausführenden Organ für die Ausführung des Maschinencodes (14) dazwischen geschaltet.

Damit besteht der Vorteil, die manuelle und damit nur aufwendig anpassbare Implementierung der Gegenmaßnamen zu vermeiden und in die Toolchain zu verlagern, wobei dynamische Parameter definieren und damit der Toolchain vorgeben, wie und in welchem Ausmaß die Umsetzung der Gegenmaßnahmen stattzufinden hat. Beispielsweise können hier der zur Verfügung stehende Speicherplatz für die redundanten Daten sowie der gewünschte Abarbeitungsaufwand der Prüfroutinen definiert werden. Die Implementierung eines dynamischen Parameters hat also den Vorteil, dass man über diese Variablen frei entscheiden kann, wie die Sicherheitsprozeduren und in welcher Zeit sie ablaufen, während dies bei einer manuellen Source-Code-Programmierung nicht oder nur mit großen Aufwand möglich ist.

Mit den Maßnahmen der Erfindung wird individuell notwendiger Programmieraufwand eingespart, weil die Implementierung von Gegenmaßnahmen in die selbsttätig ablaufende Toolchain verlagert ist.

Zeichnungslegende

1	Software-Gegenmaßnahmen
2	Sichere Chip-Hardware
3	Dokumentation
4	Chipkarte
5	Programmierer
6	Applikationen
7	Betriebssystem
8	Toolchain
9	Source-Code
10	Preprozessor
11	Compiler
12	Assembler
13	Linker
14	Maschinencode
15	Erzeugung von Gegenmaßnahmen
16	Analyseinformation
17	Software-Modul
18	Parameter

Patentansprüche

1. Chipkarte mit Maßnahmen zur Überwachung der Datenintegrität und Maßnahmen zur Gewährleistung der Ablaufintegrität auf Softwarebasis (Software – Gegenmaßnahmen (1)), wobei diese Maßnahmen in der auf der Chipkarte (4) ablaufenden Software (Betriebssystem (7) und Applikationen (6)) realisiert sind, um die Chipkarte (4) gegen äußere Angriffe zu schützen, insbesondere durch Software – Gegenmaßnahmen (1) wie

- Versehen der Daten, Adresswerte und Rücksprungadressen mit Prüfsummen oder redundanten Daten und deren Verarbeitung durch Prüfroutinen
- Einbringen von redundanten Codeabschnitten (z.B. doppeltes und ggf. komplementäres Abfragen von Bedingungen, Weitergabe redundanter Information bei Unterprogrammaufrufen) zur Sicherung der Ablaufintegrität logischer Programmabfolgen
- Einbringen von zusätzlichem Code und Daten zur Erkennung von unbeabsichtigten Sprüngen (z.B. von einem Angreifer induziert) innerhalb des Programmablaufes.
- Das Schreiben und nachfolgende Zurücklesen von Daten
- Durchführen von mathematisch inversen Operationen und Kontrolle der Ergebnisse
- Aktivieren und Deaktivieren von Hardware-Gegenmaßnahmen gemäß den Vorgaben der Dokumentation (3)

dadurch gekennzeichnet, dass diese Erzeugung von einer oder mehreren der Gegenmaßnahmen (15) (redundante Daten und Prüfroutinen) von einem Software-Modul (17) durchgeführt wird.

2. Chipkarte nach Anspruch 1, **dadurch gekennzeichnet, dass** das Software-Modul (17) selbsttätig abläuft.

3. Chipkarte nach Anspruch 1 oder 2, **dadurch gekennzeichnet, dass** das Software-Modul (17) zur Erzeugung der Gegenmaßnahmen in eine oder alle Komponenten der Toolchain (bestehend aus zumindest Preprozessor (10), dem Compiler (11), dem Assembler (12), dem Linker (13)) eingreift oder integriert ist.

4. Chipkarte nach einem der Ansprüche 1 bis 3, **dadurch gekennzeichnet, dass** das Software-Modul (17) zur Erzeugung der Gegenmaßnahmen durch Parameter (18) gesteuert wird, welche die Art und den Umfang der Gegenmaßnahmen definieren, insbesondere

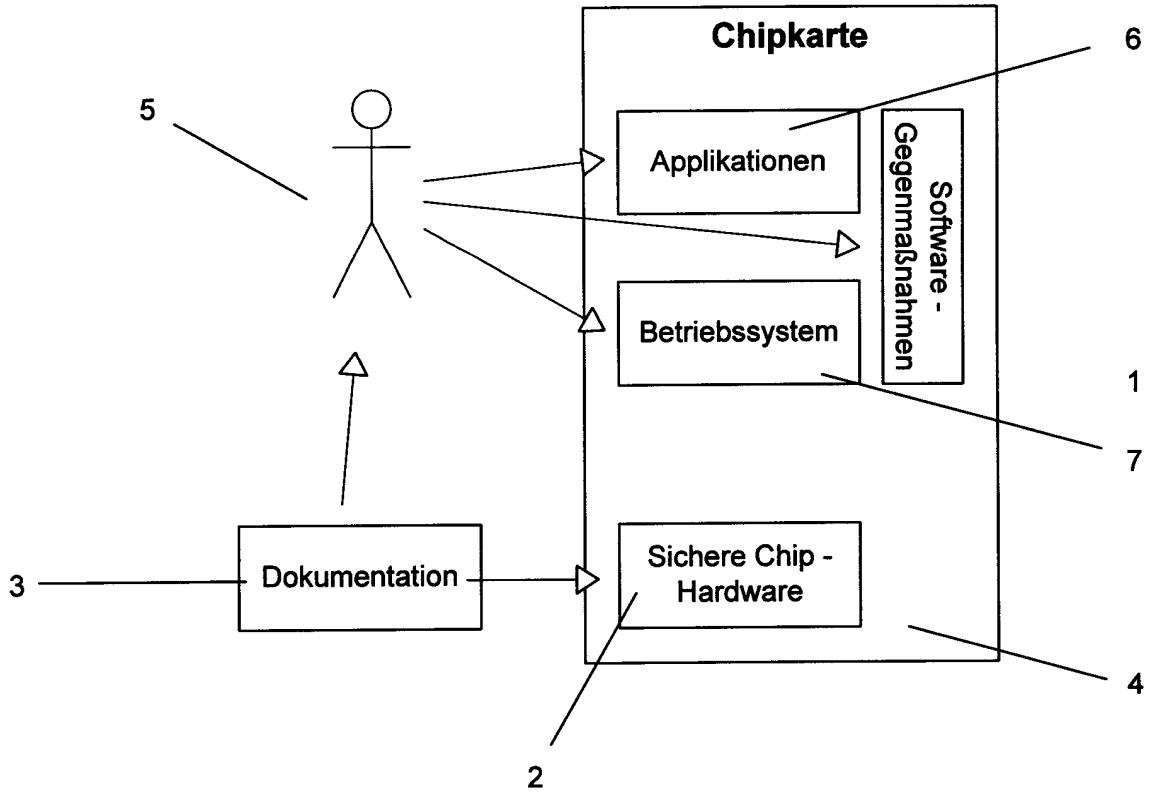
- Welche Arten von Daten durch redundante Daten abgesichert werden sollen (z.B. Daten im RAM oder EEPROM, Adresswerte, Registerinhalte)
- Wie die redundanten Daten gestaltet sein sollen (z.B. Art der zu benutzenden Prüfsummen)
- Welche Arten von Prüfroutinen erzeugt werden sollen (Erstellen und Prüfen von Prüfsummen vor/nach der Datenverwendung, doppeltes Abfragen von Bedingungen ggf. in komplementärer Weise, redundante Code-Abschnitte, Weitergabe redundanter Information zur Kontrolle der logischen Programmablauffolgen, Schreiben und nachfolgendes Lesen und Vergleichen von Daten, Durchführen von mathematisch inversen Operationen)
- Wie viel Prüfroutinen eingebracht werden sollen (absolute oder relative Maßzahl der Codegröße)

5. Chipkarte nach einem der Ansprüche 1 bis 4, **dadurch gekennzeichnet, dass** die Toolchain (8) zusätzlich die Werkzeuge für das Anforderungsmanagement (insbesondere Sicherheitsanforderungen, welche automatisch in Gegenmaßnahmen umgesetzt werden) umfasst.

6. Chipkarte nach einem der Ansprüche 1 bis 5, **dadurch gekennzeichnet, dass** das Software-Modul (17) zur Erzeugung der Gegenmaßnahmen (15) in einen speziellen Bereich der Toolchain, konkret den Bereich zwischen dem Source-Code (9) und dem Preprozessor (10, dazwischen geschaltet ist.

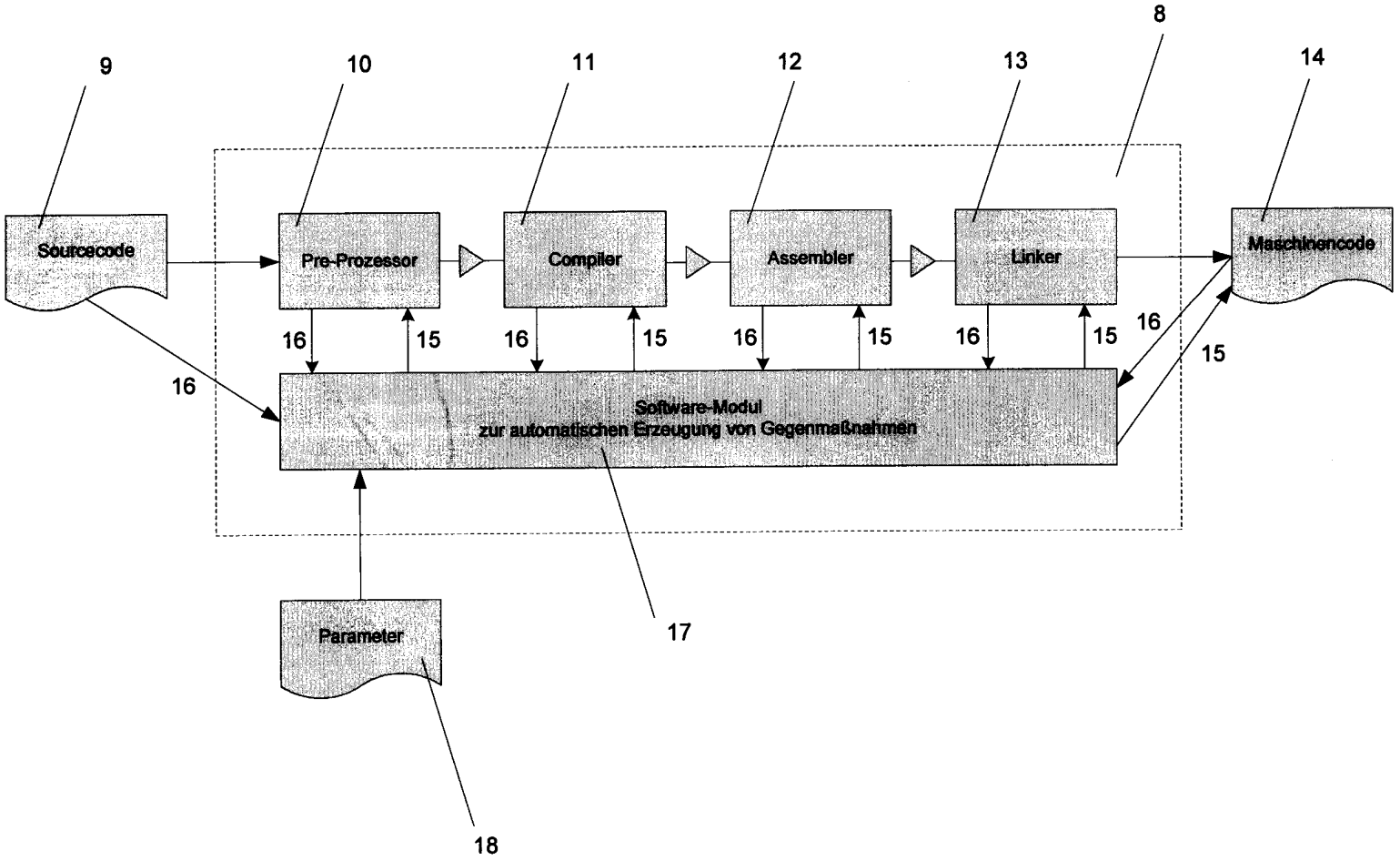
7. Verfahren zum Betrieb einer Chipkarte nach einem der Ansprüche 1 bis 6, **dadurch gekennzeichnet, dass** das Softwaremodul (17) weiters dynamische Parameter erzeugt, welche zur Laufzeit des Chipkartenbetriebssystem und der Applikationen vorgeben, wie ausführlich und wie zeitaufwendig die Gegenmaßnahmen auszuführen sind.

Figur 1:


Chipkarte
5
6
Applikationen
Software - Gegenmaßnahmen
Betriebssystem
1
7
3
Dokumentation
Sichere Chip - Hardware
4
2

5

Figur 2:


9
10
11
12
13
8
14
Sourcecode
Pre-Prozessor
Compiler
Assembler
Linker
Maschinencode
16
15
16
15
16
15
16
15
16
15
Software-Modul
zur automatischen Erzeugung von Gegenmaßnahmen
17
Parameter
18