

TPM – das Sicherheitsimplantat

Mit Trusted Platform Modules machen Embedded-Computersysteme nur das, was sie sollen

Trusted Platform Modules sind in vielen Computersystemen bereits eingebaut, werden aber nur selten genutzt. Dabei sind mit diesen Modulen viele Anwendungsszenarien denkbar, die für die Embedded-Welt von größtem Interesse sind. Ob Schutz vor Produktpiraterie oder Sicherung gegen Manipulation von Daten – die Library „Trousers“ bietet zahlreiche Hilfsfunktionen für Linux-Systeme.

Von Thomas Brinker

Zur Abwicklung und Unterstützung von Sicherheitsfunktionen in informationsverarbeitenden Systemen wurde das „Trusted Platform Module“ von der Trusted Computing Group (TCG), vormals Trusted Computing Platform Alliance (TCPA), spezifiziert. Desktop- und Server-PCs werden nun schon seit einiger Zeit mit diesem Bauteil bestückt – aber nur selten wird es auch wirklich benutzt. Grund ist die – wohl nicht ganz aus der Luft gegriffene – Besorgnis der Endbenutzer beziehungsweise Konsumenten bezüglich einer möglichen Bevormundung. In der Embedded-Welt ist TPM noch nahezu unbekannt und wird nur sehr selten eingesetzt.

Dabei sind mit TPM viele Anwendungsszenarien denkbar, die für Geräteentwickler von größtem Interesse sind. Mit I²C als Interface sind TPM-Bausteine verfügbar, die sich auch in bestehende Systeme leicht integrieren lassen. Atmel stellt das TPM AT97SC3203S her. Als Interface benutzt dieses den SM-Bus, der I²C stark ähnelt.

■ Starke Kryptographie

TPM setzt starke kryptographische Methoden zur Sicherung seiner eigenen und externer Daten ein. Dabei wird das asymmetrische Verschlüsselungsverfahren RSA angewendet. Asymmetrische Verfahren benötigen zwei

Schlüssel, die zusammen ein Schlüsselpaar bilden. Ein Schlüssel des Paares hat die Funktion eines Verschlüsselungsschlüssels. Mit ihm und der Verschlüsselungsfunktion werden die Nutzdaten verschlüsselt.

Die Entschlüsselung ist nur noch mit der Entschlüsselungsfunktion und dem passenden Entschlüsselungsschlüssel möglich. Der Verschlüsselungsschlüssel und die Kenntnisse über beide Funktionen sind nicht ausreichend, um die Originaldaten mit realistischem Aufwand wiederherzustellen. Daher ist es auch nicht erforderlich, Funktionen und Verschlüsselungsschlüssel geheim zu halten. Solange der Entschlüsselungsschlüssel geheim bleibt, sind auch die Originaldaten geheim. Bei asymmetrischen Verschlüsselungsverfahren ist es durchaus üblich, den Verschlüsselungsschlüssel offenzulegen oder sogar in einem öffentlichen Verzeichnis uneingeschränkt verfügbar zu machen. Der Entschlüsselungsschlüssel bleibt aber geheim. Der Verschlüsselungsschlüssel wird Public Key, der Entschlüsselungsschlüssel Private Key genannt. Beliebige Personen können mit Hilfe des Public Key geheime Daten an den Besitzer des Private Key versenden.

Durch Umkehrung des RSA-Algorithmus kann auch mit dem Private Key eine Datenmenge verschlüsselt werden. Nur der Public Key kann nun aus der verschlüsselten Datenmenge

wieder die Ursprungsdaten herstellen. Leider hat dieses Verfahren klare Grenzen bezüglich der Datenlänge. Nicht die Nutzdaten selbst, sondern nur ihr Hashwert werden verschlüsselt. Die Nutzdaten werden dagegen unverändert und parallel mit dem verschlüsselten Hashwert übertragen. Der verschlüsselte Hashwert wird Signatur genannt [1].

Bei Signaturen ist die Vertrauenswürdigkeit des Public Key von großer Bedeutung. „Ist das wirklich der öffentliche Schlüssel von Person XY?“ Auch dies kann wiederum mittels signierter Schlüssel geprüft werden. Das führt dann zu einem Anfangsproblem, welches jedoch mittels TPM lösbar ist, denn das TPM ist eindeutig identifizierbar. Außerdem können Public Keys vorab in das TPM geladen werden und sind somit vertrauenswürdig.

Als Funktionen zur Ver- und Entschlüsselung benutzt das TPM den RSA-Algorithmus mit 2048 bit Schlüssellänge. Asymmetrische kryptographische Operationen sind sehr aufwendig und rechenintensiv. TPM-Module enthalten deshalb zur Beschleunigung eine Hardware-Einheit und können so Ver- und Entschlüsselungsaufgaben sehr schnell durchführen. Außerdem hilft ein Hardware Key Generator, sichere Schlüsselpaare effizient herzustellen.

■ Kopierschutz für Embedded-Software

Wie bereits in [2] beschrieben, sind Plagiate allgegenwärtig. Auch Embedded-Systeme sind davon nicht ausgenommen, schließlich werden schon ganze Autos und sogar Busse „abgekupfert“. Mit einem TPM kann das Kopieren der Software nahezu ausgeschlossen werden. Im TPM-Baustein können Schlüssel gespeichert werden, die sich nicht mehr auslesen lassen. Ist die Software nun verschlüsselt im persistenten Speicher abgelegt, kann nur das TPM, das über den Schlüssel verfügt, die Software entschlüsseln. Eine zusätzliche Rückversicherung der

Software, ob das TPM über bestimmte Schlüssel verfügt, macht die Software dann hundertprozentig abhängig vom unkopierbaren Inhalt des TPM.

Die Ablage verschlüsselter Software eröffnet weitere Möglichkeiten: Nur die Person, die über den Schlüssel im TPM genaue Kenntnisse hat, kann Software so verschlüsseln, dass sie erfolgreich entschlüsselt werden kann. Personen ohne diese Kenntnisse sind nicht in der Lage, Software für dieses Embedded-System zu entwickeln.

Diese Sicherung wäre jedoch leicht zu umgehen, wenn es gelänge, die Software-Ebene, die das Entschlüsseln vornimmt, zu manipulieren. Doch auch dieser Fall kann durch das TPM ausgeschlossen werden. Der TPM-Baustein kann Prüfsummen aller bereits genutzten Software-Teile unabänderlich speichern. Diese können dann bei der Entschlüsselung anderer Software-Teile herangezogen werden. Diese Prüfsummen können anhand der vorhandenen Hardware erzeugt werden, so dass auch die Manipulation hieran erschwert wird.

Manipulationen an der Software eines Embedded-Linux-Systems sind durch die Veröffentlichungspflicht einiger Teile des Source-Codes leicht möglich. Die General Public License (GPL), welche z.B. für den Linux-Kernel gilt, verpflichtet zur Herausgabe des Codes wenigstens auf Nachfrage. Für einen Angreifer ist es also leicht möglich, an den Source-Code des Kerns zu gelangen. Weil der Kernel die zentrale Schnittstelle nahezu aller Daten ist, kann eine Manipulation die Sicherheit stark gefährden, selbst dann, wenn die Steuerapplikation nicht im Source-Code verfügbar ist.

■ Nutzungsbeschränkung von Software-Modulen

Erweiterungsmodule finden auch bei Embedded-Systemen häufigen Einsatz. Während Hardware-Module nicht ohne weiteres kopiert werden können, sieht dies bei Software-Modulen ganz anders aus. Ihre leichte Kopierbarkeit erschwert eine kommerzielle Vermark-

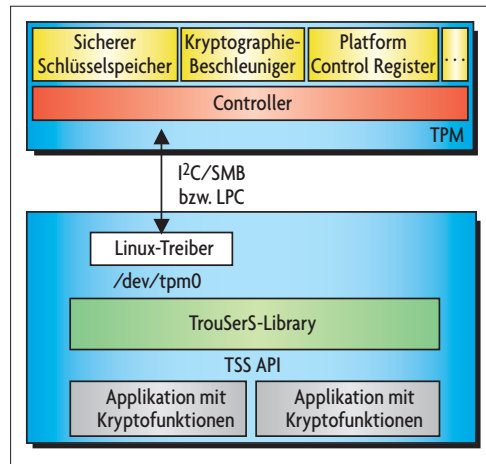


Bild 1. Trusted Platform Modules enthalten einen Controller, der die Kryptographie-Befehle steuert. Die TrouSerS-Bibliothek hilft den Applikationen bei der Nutzung der Befehle.

tung, da lediglich eine Kopie erforderlich ist, um beliebig viele Geräte mit dem Software-Modul aufzuwerten.

TPM kann jedoch die Anwendung eines Software-Moduls auf einen ganz bestimmten Kreis von Embedded-Systemen begrenzen. Das Software-Modul wird vom TPM hinsichtlich seiner Herkunft verifiziert. Soweit diese bestätigt wurde, versucht das TPM das Paket zu entschlüsseln. Dazu verwendet es einen individuellen Schlüssel. Nur wenn das Paket zuvor mit dem passenden Schlüssel verschlüsselt wurde, kann es auch benutzt werden. Das Software-Paket ist durch „Binding“ an ein bestimmtes TPM gebunden. Ein anderes TPM, das nicht über den Schlüssel verfügt, kann das Software-Modul nicht entschlüsseln. Es ist für das Embedded-System nicht nutzbar.

TPMs ermöglichen die eindeutige Zuordnung von Hardware und Software. Auf dieser Eineindeutigkeit kann Vertrauen gegründet werden. Ein solches

Vertrauen kann innerhalb eines Netzwerkes notwendig sein. Jeder Knoten im Netzwerk kann sich sicher sein, dass der andere integer ist, wenn der TPM-Baustein dieses bestätigt. Das TPM enthält dazu einen Schlüssel beispielsweise eines X.509-Zertifikates. Dieser Schlüssel kann dem TPM nicht entrisen werden, und er kann auch nicht unbemerkt ausgetauscht werden. Beispiele für solche Netzwerke sind etwa Geldautomaten und Supermarktkassen. Beide müssen sich auf die Integrität einer zentralen Instanz verlassen und umgekehrt.

Der gleiche Grad an Vertraulichkeit ist in vielen anderen Netzwerken notwendig; sei es in einer Produktion, wo sich ERP und Maschinen gegenseitig vertrauen sollen, oder in der Logistik, wo Fahrzeug und Zentrallager nur sehr selten miteinander kommunizieren.

▣ Datenpakete rein, Antwort raus

Jeder TPM-Baustein enthält einen eigenen Controller und kann komplexe Aufgaben selbstständig bearbeiten. Entsprechend einfach ist die Kommunikation mit einem TPM-Baustein. Befehle werden in Paketen zusammen mit den Daten an das TPM geschickt. Dieses arbeitet den Befehl ab und verpackt die Antwortdaten in ein Paket, welches wiederum abgeholt wird.

Als Controller in einem TPM-Baustein werden häufig kleine Prozessoren eingesetzt, deren Programm in einem ROM-Speicher unabänderlich festgelegt ist. Dieses Programm interpretiert die Befehle und stellt die Antwortpakete zusammen. Es verarbeitet sicherheitsrelevante Befehle nur bei korrekter Authentifizierung. So ist sichergestellt, dass auch bei manipulierter Software auf der Haupt-CPU das TPM und die enthaltenen Schlüssel sicher bleiben.

Entsprechend einfach ist die Entwicklung von Treiber-Software für TPM-Bausteine. In einem Embedded-Linux-System muss der Kernel ledig-

lich die Pakete mit dem TPM-Baustein austauschen. I²C und LPC sind hierbei häufig verwendete Busse zur Kommunikation. Das Erzeugen der Pakete kann außerhalb des Kernels in einer Applikation erfolgen. Die Library „Trousers“ bietet hierfür zahlreiche Hilfsfunktionen (Bild 1). Sie implementiert den, ebenfalls von TCG definierten „TCG Software Stack (TSS)“. Die Buchstabenfolge TSS findet sich ebenso in „Trousers“ wieder, weswegen dieser Name gewählt wurde. „Trousers“ kann nicht nur gegen Standard-Linux-Applikationen gelinkt, sondern auch von Bootloadern verwendet werden, da auch diese sicherheitsrelevant sind.

▣ Das TPM – ein elektronischer Schlüsselkasten

Das TPM enthält neben dem Prozessor zur Befehlsbearbeitung einen Schlüsselspeicher. Dieser Speicherbereich kann nur vom internen Prozessor gelesen und beschrieben werden. Direkte Zugriffe über das I²C- bzw. LPC-Interface sind nicht möglich, da dies die Sicherheit untergraben würde. Auf

Schlüssel kann nur mittels Befehlen an das TPM zugegriffen werden. Dies ist jedoch nur selten notwendig, da Ver- und Entschlüsselungsoperationen direkt im TPM möglich und üblich sind. Der Schlüssel selbst braucht das TPM nicht zu verlassen. Bei einigen Schlüsseln ist ein Auslesen gar nicht vorgesehen; sie verlassen das TPM also niemals.

Zwei besondere Schlüssel, die im TPM hinterlegt sind, haben feste Aufgaben. Das „Endorsement Key Pair“ wird durch den TPM-Befehl „TPM_CreateEndorsementKeyPair“ oder „TPM_CreateRevocableEK“ erzeugt. Der private Teil des Schlüssels, „privEK“, verlässt das TPM niemals. Der öffentliche Teil des Paares, „pubEK“, kann nur aus dem TPM gelesen werden, solange er im Zustand „unowned“ ist. Die TCG definiert unter anderem die Zustände „owned“ und „unowned“ und sieht vor, hiermit einen echten (juristischen) Besitzerwechsel des Systems zu modellieren. In der Produktion wird also der „Endorsement Key“ erzeugt, und das TPM ist „unowned“. Nachdem das System verkauft wurde, kann der neue Besit-

Wofür lässt sich ein TPM in einem Embedded-System nutzen?

Software-Kopiersperre

Beispiel: Firma E. stellt ein sehr erfolgreiches Steuergerät für die Hausautomatisierung her. Die Firma S. möchte illegale Kopien des Steuergerätes herstellen und auf den Markt bringen. Sie kopiert die Hardware dazu Eins zu Eins, ebenso die Software. Diese ist jedoch verschlüsselt und nicht direkt lauffähig. Der Schlüssel zur Entschlüsselung befindet sich im TPM. Und diesen kann Firma S. nicht auslesen.

Verifikation der Herkunft von Updates

Beispiel: Fleischer S. hat an seiner Verkaufstheke drei Waagen, die mit einem Embedded-Linux-Betriebssystem ausgerüstet sind. Hobbymäßig beschäftigt er sich mit Linux und hat mittlerweile viel Erfahrung sammeln können. Leider läuft die Metzgerei nicht so gut, und er lässt sich die Sources des Kernels für seine Waagen vom Hersteller zuschicken. Auf Grund der GPL ist dieser dazu verpflichtet. Fleischer S. modifiziert den Kernel und präpariert hiermit das letzte Update der Waagen so, dass die Eichwerte immer leicht zu seinen Gunsten manipuliert werden. Leider kann er das Update nicht installieren, da seine Signatur vom TPM-Chip zurückgewiesen wird. Nachdem er zum JTAG-Debugger gegriffen hat, startet die Waage nicht mehr vollständig, da der TPM-Chip das System nicht verifizieren konnte.

Beschränkung von Software-Komponenten

Beispiel: Der Getränkeautomat der Firma B. hat ein grafisches Display. Es sind zwei verschiedene Designs des Displayinhaltes verfügbar. Ein rudimentäres, ohne aufwendige Grafik für wenig repräsentative Aufstellungsorte und ein sehr aufwendiges Design, welches von einem Künstler geschaffen wurde. Der Automat mit dem aufwendigen Displayinhalt ist 10 % teurer als der andere, obwohl die Hardware sich nicht unterscheidet. Automatenaufsteller H. möchte nun sparen und trotzdem ein aufwendiges Display haben. Er installiert die Software-Updates mit aufwendigem Display. Der TPM-Chip weist das Update jedoch zurück.

Aufbau einer Authentifizierungshierarchie

Beispiel: In der Produktion der Firma H. werden Maschinen mit Embedded-Linux und TPM eingesetzt. Sie dienen der Authentifizierung der einzelnen Maschinen bei einem zentralen Server. Auf diesem Server sind die Zugangsdaten aller Mitarbeiter gespeichert, welche sich an den Maschinen einloggen müssen, an denen sie gerade arbeiten. Verlässt ein Mitarbeiter die Firma oder kommt ein neuer hinzu, so muss dieses nur auf dem Server vermerkt werden und nicht auf jeder Maschine, an der der Mitarbeiter arbeiten wird. TPM stellt sicher, dass manipulierte Boards oder Computer im Netz nicht zu Sicherheitsproblemen führen, da diese nicht mit dem Server kommunizieren können.

zer den „pubEK“ auslesen und mit „TPM_TakeOwnership“ den Besitz ergreifen. Nun kann niemand mehr den „pubEK“ auslesen. Befehle, die eine Freigabe mit dem „pubEK“ benötigen, können nur noch vom „Owner“ gegeben werden. Gleichzeitig wird bei der Übernahme ein Passwort im TPM gespeichert, mit dem sich der Besitzer authentifiziert.

Zusätzlich wird durch die Besitzübernahme durch „TPM_TakeOwnership“ der „Storage Root Key“ (SRK) erzeugt. Auch hier verlässt der private Teil des Schlüssels das TPM niemals. Der SRK wird benutzt, wenn der interne Schlüsselspeicher des TPM nicht ausreicht und Schlüssel in den Haupt- oder Festspeicher des Rechners „entladen“ werden müssen. Alle relevanten Daten der zu entladenden Schlüssel werden mit dem SRK verschlüsselt und können so – vor unbefugtem Zugriff gesichert – im unsicheren Hauptspeicher abgelegt werden.

Neben „Endorsement Key“ und „Storage Root Key“ kann ein TPM noch weitere Schlüsselpaare speichern. Entweder sind diese anderweitig erzeugt und vom Besitzer geladen worden, oder das TPM hat sie selber erzeugt. Beispiele für geladene Keys sind „Storage Keys“ zur Verschlüsselung geheimer Daten und „Signing Keys“ zum Signieren.

■ Schutz vor Software-Manipulation

Die Integrität und damit die Vertrauenswürdigkeit eines Systems können zusammen mit dem TPM bestimmt werden. Dabei werden von allen Programmteilen, die auf die Sicherheit Einfluss haben, Prüfsummen mit dem SHA-1-Algorithmus berechnet. Das Ergebnis wird in einem Register des TPM gespeichert – jedoch erst, nachdem der vorherige Wert des Registers mit dem neuen verknüpft wurde, so dass die komplette Historie der Registerwerte abgebildet wird (Bild 2).

Für solche Zwecke stellt ein TPM mehrere so genannte Platform Control Register (PCR_x) zur Verfügung. Soll ein Programm, das Einfluss auf die Sicherheit nehmen kann, neu gestartet werden, so berechnet die aufrufende

Software die Prüfsumme und erweitert mit ihr eines der PCR_s. Für den Systemstart bedeutet dies, dass der Bootloader die Prüfsumme des Linux-Kernels bestimmt und dem PCR₄ hinzufügt. PCR₄ ist von TCG dem Betriebssystem-Kernel zugeordnet.

Nun ist es durch Prüfen des PCR₄ möglich, herauszufinden, ob der erwartete Kernel gestartet wurde oder ob es einen Tausch gegeben hat. Stimmen die Werte, kann man dem Kernel vertrauen und insbesondere seiner Fähigkeit, selber PCR_s zu erweitern. Denn genau das macht der Kernel, wenn er sicherheitsrelevante Programme startet oder Kernel-Module lädt. Was genau und in welcher Reihenfolge den PCR_s hinzugefügt wurde, wird in einem Log gespeichert, dem Stored Measurement Log (SML). Der Sicherheitsprüfer kann nun mit dem Log die erwarteten PCR-Werte vorausberechnen und verifizieren.

Die Echtheit der PCR-Werte belegt das TPM durch Signatur. Somit kann ein entfernter Prüfer ausschließen, dass ihm „geschönte“ PCR-Werte untergeschoben werden. Auf diesem Wege ist es also möglich, festzustellen, ob die Software eines Embedded-Systems „echt“ ist und ob „unerlaubte“ Programme ausgeführt werden.

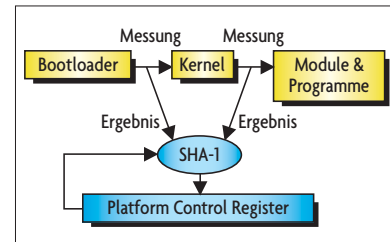
■ Bindung von Daten an ein TPM

Der TPM-Befehlssatz ermöglicht sehr komplexe, miteinander verbundene Befehlsabläufe. So kann etwa ein Entschlüsselungsbefehl von den Werten der PCR_s abhängig gemacht werden. Enthalten die PCR_s nicht die vorgegebenen Werte, so gibt das TPM eine Fehlermeldung zurück, ohne eine Entschlüsselung vorgenommen zu haben.

Liegen also bestimmte Daten nur verschlüsselt vor, so kann der Zugriff nur einem vollständig vertrauenswürdigen System erlaubt sein. Unsicheren Systemen verweigert das TPM den Zugriff. Damit die vorgegebenen PCR-Werte im Befehl nicht verfälscht werden können, wird der Befehl so verschlüsselt, dass nur ein bestimmtes TPM ihn entschlüsseln kann. Es erfolgt also ein so genanntes „Binding“ an das TPM. Die Nutzdaten haben durch das „Binding“ und Prüfen der

PCR_s ein „Sealing“ an das System erfahren.

Eine „versiegelte“ Datenmenge kann mit der Funktion `Tspi_Data_Unseal(...)` nur auf dem TPM entschlüsselt werden, auf dem die Daten mit `Tspi_Data_Seal(...)` „versiegelt“ wurden – und natürlich auch nur dann, wenn der Systemzustand „passt“. Die Funktionen sind Bestandteil des TCG Software Stack und werden von der



! Bild 2. Von allen sicherheitsrelevanten Programmteilen werden Prüfsummen berechnet und mit SHA-1 verschlüsselt. Das Ergebnis wird in einem Platform Control Register gespeichert und vorher mit dem Inhalt des Registers verknüpft. So beeinflusst die komplette Historie aller Software seit dem Systemstart den Inhalt des Registers.

Library „Trousers“ implementiert. Das abgebildete Listing erläutert die Parameter.

Die Nutzdaten, die etwa mit „Seal“ geschützt wurden, sind in den meisten Fällen wiederum Schlüssel, jedoch für symmetrische Verschlüsselungsverfahren. Diese sind deutlich weniger zeitaufwendig und sogar oft hardwarebeschleunigt entschlüsselbar. Nutzdaten wie etwa ein Linux-Kernel von ca. 1 Mbyte Größe befinden sich also symmetrisch mit AES verschlüsselt im Flashspeicher (Bild 3)

Der Schlüssel befindet sich ebenso im Flashspeicher, ist aber asymmetrisch mit RSA verschlüsselt. Der RSA Private Key zur Entschlüsselung befindet sich unerreichbar in genau diesem TPM, und die Entschlüsselungsoperation kann nur von diesem TPM vorgenommen werden – und auch nur dann, wenn alle sonstige Software vertrauenswürdig ist.

Wird also der AES-Schlüssel des Kernels erfolgreich vom TPM entschlüsselt, so kann nun der Ker-

```

TSS_RESULT Tspi_Data_Seal(
    TSS_HENCDATA    hEncData,           //zu verschlüsselnde Daten
    TSS_HKEY        hEncKey,           //Kennung des Schlüssels
    UINT32          ulDataLength,      //Länge der Daten
    BYTE*          rgbDataToSeal,     //Ausgangsdatenpuffer
    TSS_HPCRS      hPcrComposite       //einzubeziehende PCRs
);

TSS_RESULT Tspi_Data_Unseal(
    TSS_HENCDATA    hEncData,           //verschlüsselte Daten
    TSS_HKEY        hKey,              //Kennung des Schlüssels
    UINT32          puUnsealedDataLength, //Größe des Klarschriftpuffers
    BYTE**          prgbUnsealedData   //Klarschriftpuffer
);
    
```

I Durch den Befehl `Tspi_Data_Seal` können Daten an ein bestimmtes TPM gebunden werden. Nur von diesem TPM und einem intakten, nicht manipulierten System können die Daten mit `Tspi_Data_Unseal` wieder entschlüsselt werden.

nel entschlüsselt werden – entweder beschleunigt mit Hardware oder auf der Haupt-CPU. [3] empfiehlt bei der für einen Linux-Kernel typischen Datenmenge eine Hardware-Beschleunigung, soweit eine solche verfügbar ist.

■ Mehr Sicherheit durch TPM

Mit TPM lässt sich jeder Computer von einem „General Purpose“-Computer in einen „Specific Purpose“-Computer umwandeln. Auf einem Desktop oder Server ist eine solche Umwandlung wenig populär. In einem Embedded-System, das durch die Einbettung ohnehin ein „Specific Purpose“-Computer ist, geht durch TPM keine Freiheit verloren. Es wird jedoch sehr viel Sicherheit hinzugewonnen.

Die TPM-Funktionen Schlüssel-speicher, kryptographische Operationen und Schlüsselerzeugung haben dem TPM den Namen „aufgelötete SmartCard“ eingebracht. Der Ver-

gleich ist gut gewählt, aber die Funktionalität eines TPM geht mit der Prüfung des aktuellen Systemzustandes weit darüber hinaus.

Die deutlich verbesserte Sicherheit durch TPM gegenüber einem Embedded-Linux-System ohne TPM birgt jedoch auch Gefahren. Bei einem Update der Software ist peinlichst genau darauf zu achten, dass die neue Software vom TPM auch als richtig und zulässig erkannt wird. Ein ausführlicher Roll-Out-Test ist erforderlich. Zudem sollte eine Fall-Back-Möglichkeit vorgesehen werden. Denn im schlimmsten Fall werden alle Geräte, die mit einem fehlerhaften Update versehen werden, nicht mehr korrekt arbeiten, da das TPM die Zusammenarbeit verweigert.

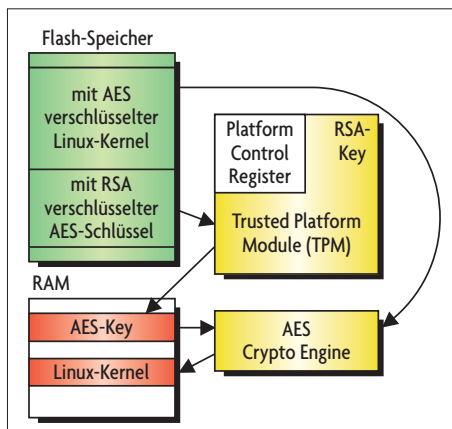
Die Sicherheit eines Embedded-Systems mit TPM kann durch Bekanntwerden des Schlüssels komplett verlorengehen. Die Integrität aller beteiligten Personen ist obligatorisch, genauso wie eine abschließende Überprüfung der auszuliefernden Software-Teile. Allzu schnell kann die Sicherheitskopie eines Schlüssels übersehen und versehentlich mit ausgeliefert werden.

Der Aufbau ganzer Vertrauensnetzwerke und die Integration von Embedded-Systemen bedarf einer genauen Planung und sollte durch den erfahrenen „Fachmann“ erfolgen. Mit dem TPM-Erweiterungsboard von EBV Elektronik und dem Linux-BSP von Emlix für das Board COBRA5329 gibt es ein umfangreiches Evaluations-Kit,

das die Möglichkeiten eines TPM in Embedded-Systemen umfassend darstellt. Anpassungen an andere Prozessorarchitekturen sind leicht möglich und können durch Emlix vorgenommen werden. *jk*

Literatur

- [1] Buchmann, J.: Einführung in die Kryptographie. 3. Auflage 2004, Springer-Verlag Berlin, Heidelberg, New York.
- [2] Kroll, J.: Bunte Waren auf grauen Märkten. Produkt- und Markenfälschung – Bestandsaufnahme und Abwehrmaßnahmen. *Elektronik* 2007, H. 14, S. 30 bis 37.
- [3] Brinker, T.; Heß, S.: Sicherheit ohne Performance-Einbuße. Ein Kryptomodul entlastet den Prozessor bei den aufwendigen Chiffrier-Algorithmen. *Elektronik* 2006, H. 22, S. 90 bis 95.



I Bild 3. Das TPM entschlüsselt den AES-Schlüssel nur auf vertrauenswürdigen Systemen. Die AES Crypto Engine entschlüsselt danach den Linux-Kernel.



Dipl.-Ing. Thomas Brinker

studierte an der Technischen Universität Berlin Technische Informatik. Seit 2005 ist er bei der emlix GmbH in Göttingen als Embedded Linux Systemengineer beschäftigt und dort an der hardwarenahen Entwicklung von Produkten für die Medizintechnik, Automatisierungstechnik und Datentechnik beteiligt. Er arbeitet im Bereich der Kernel- und Treiberentwicklung und leitet seit 2006 die Emlix-Niederlassung in Berlin. tb@emlix.com