

# Sicherheit ohne Performance-Einbuße

## Ein Kryptomodul entlastet den Prozessor bei den aufwendigen Chiffrier-Algorithmen

**Die Freescale-Prozessoren MCF5235, 5329 und 5373 sind mit einem integrierten Kryptomodul ausgestattet. Doch was bewirkt dieses Modul eigentlich? Am Beispiel des MCF5329 wird hier gezeigt, wie stark die Hardware einen Krypto-Algorithmus gegenüber der reinen Software-Lösung beschleunigt. Die Programmierung ist einfach – bei Verwendung einer Library kann die Software sogar unverändert bleiben.**

Von Thomas Brinker und Sebastian Heß

**D**atensicherheit ist in aller Munde – auch in Embedded-Systemen spielt sie eine immer größere Rolle. Verschlüsselte Datenströme sichern Kommunikationsverbindungen gegen Abhören. Datenpakete können durch Verschlüsselung signiert werden, um die Herkunft eindeutig zu verifizieren. Dies verbessert z.B. die Sicherheit von Software-Updates. Auch bestimmte Bereiche des Speichers können durch Verschlüsselung vor Zugriffen und Manipulationen von außen geschützt werden. Eine andere Anwendung für Krypto-Algorithmen ist die Bildung von Hash-Werten. Diese Werte werden aus großen Datenblöcken, wie z.B. Dateien, gebildet und dienen dazu, die Datenblöcke eindeutig zu identifizieren.

Anwender von kryptographischen Algorithmen in einem Embedded-System stehen vor der Entscheidung, ob es sich lohnt, kryptographische Funktionen selbst zu implementieren oder einer Library zu entnehmen. Viele Libraries sind verfügbar, erfüllen spezielle Aufgaben und sind an die Bedürfnisse kryptographischer Verfahren angepasst.

### ■ Das Ressourcenproblem

Das bekannteste Beispiel für kryptographische Bibliotheken dürfte Open-

SSL sein, welche nahezu alle Funktionen implementiert, die in kryptographischen Systemen typischerweise benötigt werden. Neben der Verwaltung von Secure Socket Layern gehören auch alle aktuell wichtigen symmetrischen und asymmetrischen Verschlüsselungsverfahren sowie Funktionen zur Zertifikats- und Sessionverwaltung dazu. Diese Vielzahl an Funktionen führt jedoch zu einer nicht zu unterschätzenden Größe, sowohl des Codes als auch der binären Dateien von OpenSSL. Deshalb eignet sich diese Library nur für Embedded-Systeme mit ausreichend Speicher. Ein großer Vorteil von OpenSSL sind die vielen, bereits vorhandenen Open-Source-Applikationen, die auf OpenSSL aufbauen, sowie der Support für OCF zur Nutzung von Beschleuniger-Hardware.

Eine schlankere Alternative zur OpenSSL-Bibliothek ist libtomcrypt, die beispielsweise vom „dropbear“-SSH-Server verwendet wird. Libtomcrypt ist als Public Domain lizenziert und kann daher lizenzkostenfrei eingesetzt werden. Die Liste der unterstützten Krypto-Algorithmen (Cipher) ist genauso vollständig wie die der unterstützten Hash-Funktionen. Die Mehrzahl der Algorithmen kann hinsichtlich der Größe oder der Laufzeit optimiert werden. Assembler-Optimierun-

gen für x86, x86\_64 und PowerPC sind bereits Bestandteil der Library. Libtomcrypt bietet sich für all jene an, die OpenSSL aufgrund der Größe nicht einsetzen können, aber dennoch auf eine verlässlich getestete Implementierung aktueller Krypto-Algorithmen Wert legen. Libtomcrypt bietet leider keinen Support für OCF.

Neben diesen beiden Bibliotheken für kryptographische Funktionen entwickelt sich noch yaSSL (yet another SSL). Diese Bibliothek steht unter einer Dual-Lizenz, was eine kommerzielle Nutzung genauso erlaubt wie die Verwendung in GPL-Software. Die Bibliothek bringt eine OpenSSL-Kompatibilitätsschicht mit und unterstützt alle gängigen Krypto- und Hash-Funktionen.

### ■ Kryptomodul entlastet Prozessor

Die kryptographischen Funktionen aller Libraries sind sehr aufwendig und benötigen leistungsstarke Prozessoren, um schnell abgewickelt werden zu können. Um die Geschwindigkeit der Verschlüsselung zu erhöhen, bringen immer mehr moderne Embedded-Prozessoren ein eigenes Kryptographie-modul mit. Die Hauptaufgabe dieser Module ist die Anwendung von kryptographischen Operationen auf Datenblöcke, um den Prozessor zu entlasten. Die Mehrzahl dieser Module erlaubt es, Hash-Werte von Datenblöcken zu bilden oder die Daten zu verschlüsseln.

Der Prozessor wird dann nicht mit der Aufgabe des Verschlüsseln belastet, sondern nur dazu genutzt einige Register mit den passenden Werten zu beschreiben und so die entsprechende Aktion zu starten. Sobald die Kryptomodule ihre Arbeit getan haben, lösen sie einen Interrupt aus, sodass der Prozessor die fertigen Ergebnisse verarbeiten kann. Während das Modul die

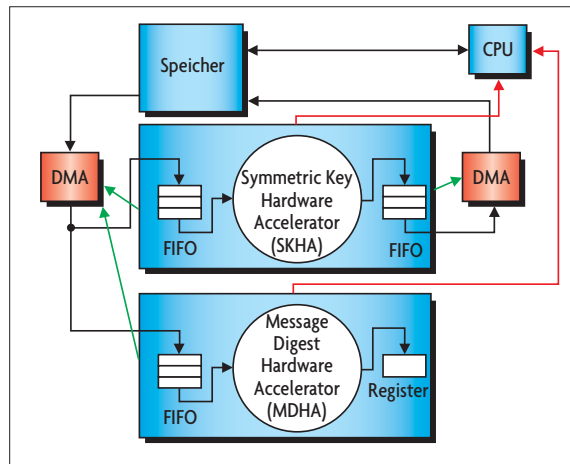
Berechnungen anstellt, kann der Prozessor in der Zwischenzeit andere Aufgaben ausführen. Als konkretes Beispiel für einen Embedded-Prozessor mit Kryptomodul dient im Folgenden der Coldfire MCF5329 von Freescale. Das Modul unterstützt alle üblichen Verschlüsselungsverfahren.

Um den Prozessor vollständig zu entlasten, muss das Kryptographiemodul selbstständig in der Lage sein, Daten aus dem Speicher zu lesen bzw. in den Speicher zu schreiben. Für diese Aufgabe werden direkte Speicherzugriffe genutzt, so genannte DMAs (Direct Memory Access). Die eDMA-Engine (Enhanced DMA) des MCF5329 überträgt immer dann Daten, wenn ein DMA-Request ausgelöst wurde. Diese DMA-Requests lösen die Module aus, sobald ihr Eingangs-FIFO eine bestimmte Anzahl an freien Plätzen aufweist oder das Ausgangs-FIFO einen bestimmten Füllstand erreicht hat. Die eDMA-Engine kümmert sich um das automatische Hochzählen der Speicheradresse und führt Konvertierungen zwischen verschiedenen Speicherbreiten durch, um den Datenfluss zu optimieren.

Zur Verwaltung der DMA-Aufträge werden „Transfer Control Descriptors“ (TCD) verwendet, die an bestimmte Funktionseinheiten gebunden sind. Diese Deskriptoren enthalten neben der Start- und der Zieladresse im Speicher auch Angaben über die Transfergröße pro Request, mögliche Verkettungen mit anderen DMA-Aufgaben sowie einige Steuer- und Statusbits. Durch diesen Aufbau können Daten „in-place“ verschlüsselt werden, was bedeutet, dass die Eingangsdaten vom verschlüsselten Ergebnis überschrieben werden.

**Bild 1** veranschaulicht das Zusammenspiel zwischen den kryptographischen Funktionseinheiten, der CPU und der DMA-Engine. Auf dem Diagramm ist die Richtung des Informationsflusses zu erkennen. Dabei ist zu beachten, dass der MDHA (Message Digest Hardware Accelerator) nur Daten von der DMA-Engine empfängt, aber keine versendet. Der SKHA

(Symmetric Key Hardware Accelerator) hingegen benutzt DMA-Transfers zum Füllen seines Eingangs-FIFOs und zum Leeren des Ausgangs-FIFOs. Die Request-Leitungen der Einheiten sind in Grün, die Interrupt-Leitungen an die CPU in Rot dargestellt. Zu erkennen ist, dass die CPU nach dem Programmieren der DMAs und des SKHA bzw. MDHA keine weitere Verbindung mit diesen beiden unterhält. Das Abfragen des Hash-Wertes,



**! Bild 1.** Daten zur Verschlüsselung (SKHA) werden durch DMAs gelesen und geschrieben. Daten zur Hashwert-Bildung werden durch DMA nur gelesen.

welcher vom MDHA gebildet wird, erfolgt über ein internes Register.

Die Aktionen, um einen Datenblock symmetrisch zu verschlüsseln, lassen sich wie folgt zusammenstellen:

- ▶ Einstellen der DMAs auf die Startadresse des Klartext- und des verschlüsselten Textblocks. Diese beiden fallen möglicherweise auf dieselbe Adresse, falls in-place verschlüsselt wird.
- ▶ Programmieren des symmetrischen Schlüssels sowie des Krypto-Algorithmus.
- ▶ Falls „Cipher Block Chaining“ verwendet wird, muss noch ein Initialisierungsvektor angegeben werden (normalerweise der vorhergehende Datenblock).
- ▶ Einstellen der Nachrichtenlänge,
- ▶ Starten des Moduls.

Ab diesem Zeitpunkt ist die CPU wieder frei für weitere Aktivitäten. So kann in der Zwischenzeit beispielsweise eine Netzwerkanfrage beantwortet oder die grafische Oberfläche neu auf-

## ■ Steuerung über Open Cryptographic Framework

Wie in modernen Betriebssystemen üblich, soll der Zugriff auf die Hardware möglichst abstrakt und generisch erfolgen. Mit dem Open Cryptographic Framework (OCF), das ursprünglich für OpenBSD entwickelt wurde, steht ein Framework aus zwei Schichten zur Verfügung. Die oberste Schicht, die die Anwendungen direkt anspricht, besteht aus einem „character device“ – dem so genannten „cryptodev“. Bei „cryptodev“ handelt es sich um eine Pseudodatei. Mit Zugriff auf diese Datei wird OCF gesteuert. Alle Einstellungsmöglichkeiten des OCF sind dem Benutzer über das „cryptodev“ zugänglich. Die Anwendungsschicht validiert alle Eingaben des Nutzers, um eine fehlerhafte Verwendung des OCF zu verhindern.

Die Treiberschicht liegt unter der Anwendungsschicht und steuert direkt die Hardware. OCF für Linux enthält bereits Unterstützung für verschiedene Kryptographiemodule. Aktivitäten wie das Einstellen der DMAs oder das Programmieren der Hardware-Register erfolgen vollständig in dieser Treiberschicht. **Listing 1** zeigt beispielhaft die Benutzung von OCF zur Erzeugung der MD5-Summe. Nachdem einige benötigte Variablen angelegt wurden, wird die Pseudo-Datei `/dev/crypto` geöffnet. Ab Zeile 21 werden dann die Variablen mit passenden Werten für eine MD5-Sitzung gefüllt. Der `ioctl`-Befehl in Zeile 25 legt die Session dann an. Die Sessionnummer wird dabei in das Feld `sess.ses` geschrieben. Im nächsten Schritt werden die Daten mit der Session verknüpft (Zeile 27). Der `ioctl`-Befehl in Zeile 35 führt dann die Operation aus. Es werden nur Zeiger auf die Daten übergeben, eine Kopie wird nicht erzeugt.

OCF muss aber nicht direkt von der Applikation aus bedient werden. In den meisten Fällen wird eine Library zwischengeschaltet, die als Bindeglied zwischen Applikation und Treiber dient. Diese Aufgabe kann unter anderem auch von OpenSSL übernommen werden (siehe **Bild 2**). Zum Zeitpunkt der Compilation von OpenSSL muss lediglich angegeben werden, dass OCF verfügbar ist. Alle Befehle, die sich an

```

01 #include <stdio.h>
02 #include <unistd.h>
03 #include <fcntl.h>
04 #include <sys/ioctl.h>
05 #include <crypto/cryptodev.h>
06
07 int main (void)
08 {
09     unsigned char* in = (unsigned char*)"md5 crypto testing string";
10     unsigned char out[16];
11     int i, fd, cfd;
12     struct session_op sess;
13     struct crypt_op cryp;
14
15     fd = open("/dev/crypto", O_RDWR, 0);
16     ioctl(fd, CRIOGET, &cfd);
17     fcntl(cfd, F_SETFD, 1);
18
19     /* Variablen mit passenden Werten einer */
20     /* MD5-Sitzung belegen */
21     sess.cipher = 0;
22     sess.mac = CRYPTO_MD5;
23     sess.mackeylen = 16;
24     sess.mackey = 0;
25     ioctl(cfd, CIOCGSESSION, &sess);
26
27     cryp.ses = sess.ses;
28     cryp.len = strlen(in);
29     cryp.src = in;
30     cryp.dst = out;
31     cryp.iv = 0;
32     cryp.op = COP_ENCRYPT;
33
34     /* Do the MD5 */
35     ioctl(cfd, CIOCCRYPT, &cryp);
36
37     //out beinhaltet nun MD5-Summe
38
39     return 0;
40 }

```

Listing 1. Benutzung von OCF zur Erzeugung der MD5-Summe. Nachdem einige benötigte Variablen angelegt wurden, wird die Pseudo-Datei `/dev/crypto` geöffnet.

gebaut werden. Sobald das Kryptomodul seine Arbeit verrichtet hat, signalisiert es dem Prozessor seine Bereitschaft per Interrupt. Die CPU prüft danach den Status des Moduls, um Fehlfunktionen zu korrigieren oder die entstandenen Daten als gültig zu markieren und den nächsten Einsatz vorzubereiten. Durch dieses Vorgehen bleibt der CPU immer wieder Zeit für andere Aufgaben. Das System wirkt

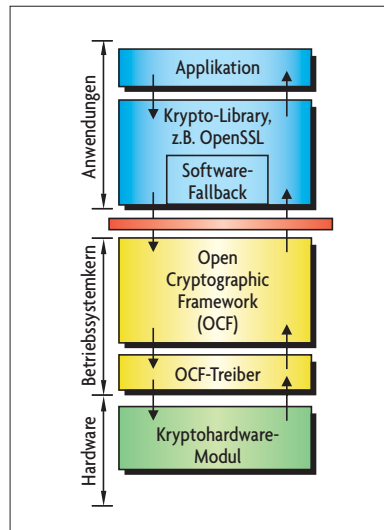
flüssiger und kann so schneller auf Benutzereingaben reagieren.

Das Kryptomodul des MCF5329 muss für jeden Datenblock, der verschlüsselt werden soll, recht kompliziert neu initialisiert werden. Bei kleinen Datenblöcken macht sich das negativ bemerkbar.

Wie fast alle Kryptomodule, bietet auch das des MCF5329 keine Unterstützung bei der Durchführung eines Diffie-Hellman-Algorithmus. Insbesondere die hier erforderliche Modulo-Berechnung großer Zahlen muss der Prozessor selbst durchführen. Dies wirkt sich negativ auf die Dauer der Initialisierungsphase einer verschlüsselten Kommunikation aus. Dieser Umstand hat im späteren Verlauf der Sitzung aber keine weiteren Auswirkungen, da später nur noch symmetrisch verschlüsselt wird.

die Hardware ausgelagern lassen, werden von OpenSSL dann über OCF an die Hardware weitergegeben. Die Applikation bemerkt davon nichts – die kryptographischen Funktionen bleiben unverändert. Auch bereits vorhandene Applikationen können davon profitieren, ohne dass Anpassungen notwendig sind. Es muss lediglich sichergestellt werden, dass auf eine OpenSSL-Version zugegriffen wird, die OCF unterstützt.

Die meisten Standard-Netzwerkanwendungen nutzen zur Ver- bzw. Entschlüsselung eine Library, oftmals OpenSSL, seltener libtomcrypt. So können zum Beispiel Websites mit dem Secure Socket Layer (SSL) geschützt werden, ohne dass der Prozessor sehr viel stärker belastet wird, weil das Webserver-Programm auf die OpenSSL zurückgreift. Weitere Möglichkeiten bietet die Secure Shell (SSH). SSH baut eine Telnet-ähnliche Verbindung auf. Alle Daten inklusive der Passwörter werden verschlüsselt



**Bild 2.** Die kryptographischen Berechnungen werden von der Library über OCF an die Hardware ausgelagert. Funktionen, die die Hardware nicht unterstützt, werden vom Software Fallback abgefangen.

und damit sicher versendet. Mit SSH lassen sich aber auch ganze Datenströme verschlüsselt übertragen, wie es

zum Beispiel das Secure-Copy-Programm (scp) macht.

### ■ Krypto-Hardware spielt bei großen Datenblöcken Vorteile voll aus

Das Beispiel in **Listing 2** zeigt die Verwendung der OpenSSL Library, um die MD5-Summe eines einfachen Textes zu berechnen. Die komplette Berechnung ist in der Funktion MD5 gekapselt. Für größere Datenblöcke stehen Funktionen zur Verfügung, um diese stückweise zu berechnen. Die **Tabelle** zeigt die Benchmark-Ergebnisse des OpenSSL-Testprogramms für AES- und DES-Verschlüsselung. Es wird die Verschlüsselungsleistung in Kbyte pro Sekunde gezeigt. Die Daten sind dabei in unterschiedlich große Blöcke eingeteilt. Bei kleiner Datenblockgröße sind entsprechend häufiger Aufträge durch die Hard- bzw Software abzuarbeiten. Je kleiner also die Datenblockgröße, umso größer ist der

```
#include <openssl/md5.h>

int main (void)
{
    unsigned char* in = "md5 crypto testing string";
    unsigned char out[16];
    int i;

    MD5(in, strlen(in), out);

    //out enthält nun MD5-Summe

    return 0;
}
```

**Listing 2. Berechnung einer MD5-Summe eines einfachen Textes unter Verwendung der OpenSSL Library. Die komplette Berechnung ist in der Funktion MD5 gekapselt.**

Verwaltungsaufwand. Es ist deutlich zu erkennen, dass für kleine Datenblöcke die Verwendung der Hardware nicht sinnvoll ist. Der Verwaltungsaufwand kehrt die Beschleunigung durch die Hardware ins Gegenteil um. Während die Verschlüsselung durch die Software mit etwa 1270 Kbyte/s ihr Maximum erreicht, ist bei der Hardware auch bei 8 Kbyte Datenblockgröße das Maximum noch nicht erreicht.

Sehr viel praxisnäher kann die Leistungsfähigkeit des Systems mit dem Secure-Copy-Program (scp) getestet werden. Mittels scp wurden große Datenmengen über das Ethernet-Netzwerk kopiert. Bei deaktivierter Hardware-Beschleunigung beträgt der Durchsatz 546,2 Kbyte/s. Gleichzeitig ist der MCF-5329-Prozessor so ausgelastet, dass eine Benutzerinteraktion nicht mehr sinnvoll möglich ist. Wird die Hardware-Beschleunigung verwendet, steigt der Durchsatz um 52 % auf 832,9 Kbyte/s, gleichzeitig bleibt das System voll benutzbar.

Zufallszahlen bilden einen weiteren wichtigen Grundstein in der Kryptographie. Der MCF5329 enthält zur ihrer Erzeugung einen Random-Number-Generator (RNG). Um Zufallszahlen zu generieren, gibt es verschiedene Wege. Man unterscheidet dabei zwischen echten Zufallszahlen und Pseudozufallszahlen. Echte Zufallszahlen werden meistens durch physikalische

Effekte wie z.B. Rauschen generiert. Pseudozufallszahlen lassen sich sehr einfach über Hash-Funktionen erzeugen. Sie entstehen also nicht zufällig, sondern folgen einem bestimmten Muster.

Linux generiert ständig Zufallswerte und stellt sie den Anwendungen über definierte Schnittstellen zur Verfügung. Um nun eine möglichst hohe Zufälligkeit zu erreichen, muss der Kernel Quellen besitzen, die sich nicht deterministisch verhalten. Das Signal eines Zeitgebers ist somit nicht geeignet. Es ist jedoch möglich, bestimmte Interruptquellen als nicht deterministisch zu markieren. Ein gutes Beispiel hierfür ist die Computer-Maus. Bewegungen und Aktionen hängen einzig vom Benutzer ab und sind nicht vorhersagbar. In Embedded-Systemen besteht nun das Problem, dass Maus und Tastatur nicht immer zum Einsatz kommen und so nicht als Quelle für Zufallszahlen dienen können. Um dennoch verlässliche Zufallszahlen zu erhalten, nutzen RNGs physikalische Effekte und umgehen das Problem damit elegant und sicher. Die Zufallszahlen

Algorithmus	Engine	Datenblockgröße				
		16 byte	64 byte	256 byte	1024 byte	8192 byte
aes-128-cbc	Software	984,95 Kbyte/s	1194,38 Kbyte/s	1260,76 Kbyte/s	1278,98 Kbyte/s	1269,49 Kbyte/s
aes-128-cbc	Hardware	54,37 Kbyte/s	214,38 Kbyte/s	763,59 Kbyte/s	2090,05 Kbyte/s	4261,47 Kbyte/s
des-cbc	Software	722,32 Kbyte/s	817,61 Kbyte/s	846,58 Kbyte/s	854,12 Kbyte/s	849,04 Kbyte/s
des-cbc	Hardware	55,44 Kbyte/s	216,48 Kbyte/s	774,70 Kbyte/s	2098,18 Kbyte/s	4294,02 Kbyte/s

Die Beschleunigung durch die Hardware ist effektiver, je größer die Datenblöcke sind. Bei kleinen Datenblöcken erfordert die jeweilige Initialisierung des Kryptomoduls so viel Verwaltungsaufwand, dass eine reine Software-Lösung schneller ist.

werden meistens vom Jitter der PLL abgeleitet und in einem FIFO gespeichert. Von dort holt der Linux-Kernel die Zahlen ab und stellt Sie über sein Standard-Interface zur Verfügung. Applikationen können die Zufallszahlen aus der Pseudodatei /dev/random auslesen.

## ■ Sicherheit bei guter Performance

Der Stellenwert leistungsstarker Kryptographie wird heute wohl von niemandem mehr unterschätzt. Dass Sicherheit nicht zu Lasten der Performance gehen muss, auch wenn ein Embedded-Prozessor verwendet wird, zeigen die Mess-Ergebnisse. Die Verwendung von Kryptomodulen durch die Software wird durch OCF und OpenSSL ganz erheblich vereinfacht, in vielen Fällen ist nicht einmal eine Anpassung erforderlich. Die Entwicklung eigener

Software unter Verwendung von OpenSSL ist leicht möglich, auch wenn die Dokumentation nicht die übersichtlichste ist. Möchte man jedoch auf andere Crypto-Libraries zurückgreifen oder ganz ohne solche arbeiten, dann bietet OCF ein umfangreiches Interface zum Zugriff auf die Hardware an. Die Entwicklung von Software mit OCF ist leicht möglich, die Verwendung einer Library ist jedoch fast immer anzuraten.

Das Hardware-Kryptomodul des MCF5329 lässt kaum Wünsche offen. Alle gängigen Algorithmen sind integriert. Der Datendurchsatz ist, wie man es von Hardware-Modulen erwartet, sehr groß. Die beschriebenen Limitierungen wirken sich in der Realität von Embedded-Systemen nur sehr wenig aus, dennoch sollte im einzelnen eine genaue Bewertung vorgenommen werden. Die Freescale-Prozessoren MCF5373 und MCF5235 verfügen über das gleiche Modul wie MCF5329, so dass eine Anpassung von OCF leicht möglich ist und Anwendungen auch auf diesen Prozessoren von der Beschleunigung profitieren können. *jk*



**Thomas Brinker**

studierte an der Technischen Universität Berlin Technische Informatik. Seit 2005 ist er bei der emlix GmbH in Göttingen als Embedded Linux Systemingenieur 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 unterstützt Kunden bei deren Applikationsentwicklung als Berater.  
[tb@emlix.com](mailto:tb@emlix.com)



**Sebastian Heß**

studiert technische Informatik an der Hochschule Mannheim. Im Auftrag der Emlix GmbH entwickelte er Software für die Kryptographie-Einheiten der MCF5329-CPU sowie weitere Treiber für das COBRA5329-Referenzboard.