KRYPTOGRAPHIECOPROZESSOR ZUR VERSCHLÜSSELUNG VON ISDN-DATEN

Mathias Schmalisch, Hagen Ploog, Frank Grassert, Dirk Timmermann

Institut für Angewandte Mikroelektronik und Datentechnik, Universität Rostock
Richard-Wagner-Str-31, 18119 Rostock
Tel.: 0381 / 498 35 36
Fax: 0381 / 498 36 01
Email: mathias.schmalisch@e-technik.uni-rostock.de

Kurzfassung: In vielen Fällen werden für ISDN (Integrated Services Digital Network) - Geräte Mikrocontrollersysteme eingesetzt. Damit ein solches Gerät genutzt werden kann, um die Informationen, die übers ISDN übertragen werden, gegen das Belauschen durch Dritte zu sicher, sind einige Erweiterungen notwendig. In diesem Beitrag sollen unsere Erfahrungen dargestellt werden, wie ein vorhandenes ISDN Gerät zu einem Verschlüsselungssystem erweitert werden kann.

1 EINFÜHRUNG

Das ISDN [1] ist ein weit verbreitetes öffentliches Netz, über das viele Arten von Informationen übertragen werden können. Die wichtigsten sind dabei vor allem die Telekommunikationsdienste, wie Telefonieren, Faxen und ähnliches. Als das ISDN im Jahr 1984 entwickelt wurde, sind allerdings keine Möglichkeiten vorgesehen worden, um das Belauschen durch Unbefugte zu verhindern. Das heißt, daß es mit geringem technischen Aufwand möglich ist Telefonate abzuhören, zu verändern und umzuleiten. Einen wirksamen Schutz dagegen bietet die Kryptographie, wobei die Daten auf der Senderseite verschlüsselt und auf der Empfängerseite wieder entschlüsselt werden.

Um die Daten für eine ISDN-Kommunikation zu verschlüsseln gibt es zwei Möglichkeiten, entweder eine Softwarelösung oder eine spezielle Hardware. Bei der Softwarelösung wird ein Computer mit einer ISDN-Karte benötigt. Ein solche Lösung ist aber durch Viren und Trojaner angreifbar, außerdem können die meist schon vorhandenen ISDN-Geräte nicht mehr weiterverwendet werden. Daher haben wir uns dafür entschieden, eine spezielle Hardware zu entwickeln, welche die Verschlüsselung der ISDN-Daten übernimmt.

Im weitern Verlauf werden der Aufbau des vorhandenen Mikrocontrollersystems und die Erweiterung durch ein FPGA, in welches die Verschlüsselungshardware implementiert wird, beschrieben. Daraufhin wird der Verbindungsaufbau zwischen zwei verschlüsselnden Endgeräten erläutert und auf das Schlüsselaustauschprotokoll näher eingegangen. Anschließend wird die Implementierung der Verschlüsselungshardware in das FPGA geschildert. Zum Abschluß erfolgt dann noch eine kurze Zusammenfassung und eine Ausblick auf fortzuführende Arbeiten.

2 AUFBAU UND ERWEITERUNG DES MIKROCONTROLLERSYSTEMS

Für unseren ISDN-Verschlüsseler (sog. ISDN-Krypter) stand als Basis ein Prototyp eines Least-Cost-Router zur Verfügung. Dieser verfügt über einen V25 Mikrocontroller und die entsprechenden ISDN-Bausteinen (ISAC), mit denen auf die ISDN-Daten zugegriffen werden kann. Außerdem befinden sich noch RAM und ein EPROM für den Mirkocontroller auf der Platine. Die beiden ISDN Bausteine tauschen die ISDN Nutzdaten, welche über den B-Kanal übertragen werden, über den IOM2-Bus aus. Dazwischen ist ein digitaler Switch (MT8981D) geschaltet, über den die ISDN Daten gelesen und manipuliert werden können.

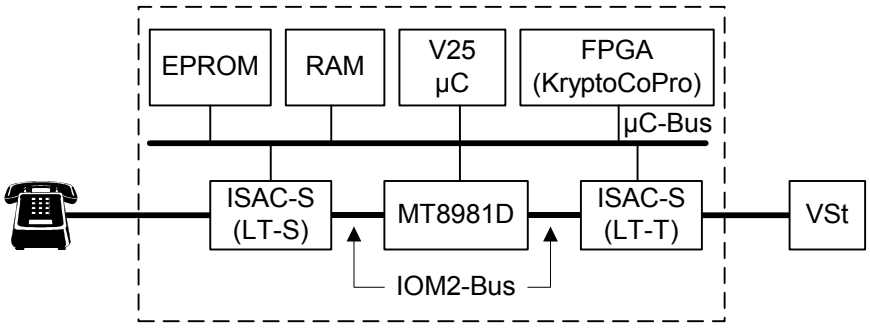

Abbildung 1: Aufbau des ISDN-Krypter

Da der Mirkocontroller nicht die Leistungsfähigkeit besitzt, die Verschlüsselung in der benötigten Zeit vorzunehmen, war es notwendig, dies in einer zusätzlichen Hardware durchzuführen. Dazu wurde ein FPGA mit einer Zusatzplatine über den EPROM-Sockel an den Mikrocontrollerbus angeschlossen. Es mußten nur noch die Takt- und Write-Enable-Leitung per Hand verdrahtet werden, da diese nicht zum EPROM hingeführt wurden. In dem so angeschlossenen FPGA konnte dann die notwendige Verschlüsselungsfunktionalität implementiert werden.

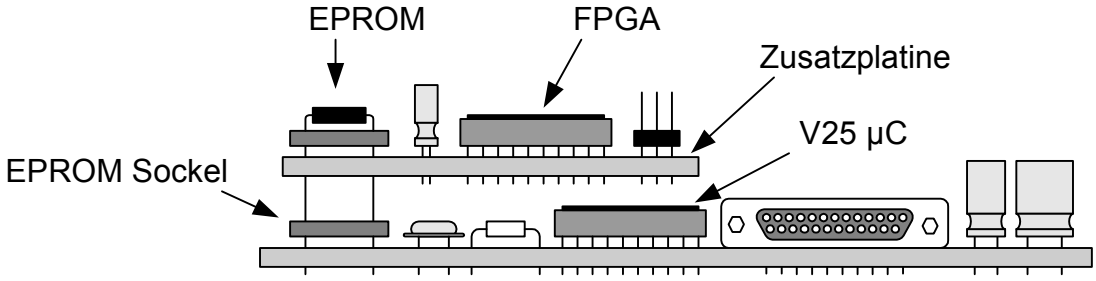

Abbildung 2: Erweiterung des Mikrocontrollersystems

3 VERBINDUNGSAUFBAU

Um einen Verbindungsaufbau zu dem angeschlossenen Endgerät zu erkennen, muß der Mikrocontroller die D-Kanaldaten des ISDN auswerten, diese werden durch die beiden ISAC's zu Verfügung gestellt. Dabei werden die D-Kanaldaten unverändert an den jeweils anderen

ISAC weitergegeben. Dadurch ist der ISDN-Krypter vollständig transparent zum eigentlichen Endgerät. Nach einem erfolgreichen Verbindungsaufbau zwischen zwei Endgeräten schaltet sich der ISDN-Krypter in den B-Kanal ein, um diese Verbindung zu sichern. Die Sicherung der Verbindung erfolgt dabei in drei Schritten.

Zuerst wird mit Hilfe einer Autodetect-Funktion festgestellt, ob sich auf der Gegenseite ein Verschlüsselungs-Partnergerät befindet, da sonst die verschlüsselten Daten nicht wieder entschlüsselt werden können. Dazu wird eine vier Byte lange Sequenz von der anrufenden Seite zu der gerufenen geschickt. Die Wahrscheinlichkeit, daß ausgerechnet diese Sequenz bei einer normalen Kommunikation auftritt ist 2^{-32}, außerdem wird sie nur kurz am Anfang einer Datenübertragung benötigt. Wenn sich auf der Gegenseite ebenfalls ein ISDN-Krypter befindet, empfängt er diese Sequenz und weiß damit, daß eine sichere Verbindung aufgebaut werden kann. Gleichzeitig schickt das Gerät die gleiche Sequenz mehrmals hintereinander zurück. Damit wird dem ISDN-Krypter auf der anrufenden Seite mitgeteilt, daß sich auf der Gegenseite ein Partnergerät befindet. Die Vier–Byte-Sequenz wird mehrmals gesendet, damit eine eventuelle Störung auf dem Übertragungskanal keinen Einfluß auf den Verbindungsaufbau hat.

Nach einem erfolgreichen Autodetect erfolgt dann im zweiten Schritt der Austausch des Kryptographie-Schlüssels zwischen den beiden ISDN-Kryptern. Darauf werden ich nachfolgend noch näher eingehen, da der Schlüsselaustausch einen entscheidenden Faktor bei der Sicherheit des ganzen Systems darstellt.

Im dritten Schritt wird dann der komplette Datenaustausch zwischen den Endgeräten auf der einen Seite verschlüsselt und auf der anderen Seite wieder entschlüsselt. Somit ist es einem Angreifer nicht mehr möglich, die Kommunikation direkt zu belauschen.

4 SICHERHEITSMODELL

In der modernen Kryptographie gibt es im großen und ganzen zwei Arten von Verschlüsselungsalgorithmen. Das sind zum einen die symmetrischen und zum anderen die asymmetrischen Algorithmen. Beiden habe jeweils ihre Vor- und Nachteile. Der große Vorteil der symmetrischen Algorithmen besteht darin, daß sie sich etwa 100 bis 1000 mal so schnell ausführen lassen wie asymmetrische Algorithmen, aber sie haben den Nachteil, daß beide Seiten den selben Schlüssel benötigen. Anders sieht es bei den asymmetrischen Algorithmen aus. Dort gibt es zwei Schlüssel, einen privaten und einen öffentlichen. Dabei kann der öffentliche Schlüssel an alle Nutzer verteilt werden, ohne daß dies ein Sicherheitsrisiko darstellt. Mit diesem Schlüssel kann eine geheimzuhaltende Nachricht verschlüsselt werden, und nur mit dem privaten Schlüssel kann diese Nachricht wieder entschlüsselt werden. Daher muß der private Schlüssel auch geheimgehalten werden.

Um die Vorteile der beiden System zu verbinden wird in unserem ISDN-Krypter ein hybrides Verschlüsselungssystem eingesetzt. Dabei wird der asymmetrische Algorithmus dazu benutzt, den symmetrischen Schlüssel sicher zum Kommunikationspartner zu befördern. Mit dem schnellen symmetrische Algorithmus werden dann die eigentlichen ISDN-Daten verschlüsselt. Aber auch dieses System ist angreifbar durch sogenannte Man-in-the-middle Attacken [2]. Um dies zu verhindern ist es notwendig, eine Authentifizierung der Nutzer vorzunehmen. Dazu wird normalerweise eine Authentifizierungsstelle verwendet, welche die Identität der Nutzer überprüft. Da wir aber nur eine Punkt-zu-Punkt-Verbindung aufbauen wollen, ist eine andere Lösung notwendig, eine Möglichkeit ist in Abbildung 3 dargestellt.

Dabei signiert die Zertifizierungsstelle die öffentlichen Schlüssel der Nutzer mit ihrem privatem Schlüssel. Dann werden jedem Nutzer sein signierter öffentlicher Schlüssel, sein privater Schlüssel und der öffentliche Schlüssel der Zertifizierungsstelle überreicht, zum Beispiel auf einer Smartcard.

Wenn nun nach dem Verbindungsaufbau zwischen zwei Endgeräten mit ISDN-Kryptern der Autodetect erfolgreich verlaufen ist, werden die signierten öffentlichen Schlüssel der Nutzer ausgetauscht. Diese können dann mit dem öffentlichen Schlüssel der Zertifizierungsstelle wieder entschlüsselt werden. Mit dem so gewonnenen öffentlichen Schlüssel werden dann die symmetrischen Schlüssel (Sitzungsschlüssel) verschlüsselt und ausgetauscht. Mit den privaten Schlüssel des jeweiligen Nutzers können dann der Sitzungsschlüssel wieder entschlüsselt werden. So erhält jede Seite auf sicherem Wege den Sitzungsschlüssel der anderen Seite. Da die Sitzungsschlüssel durch einen Zufallszahlengenerator erzeugt werden, ist bei jeder neuen Kommunikation ein neuer Sitzungsschlüssel im Einsatz, was die Sicherheit weiter erhöht. Durch das Signieren der öffentlichem Schlüssel wird es einem Angreifer wesentlich erschwert die Kommunikation zwischen den beiden Teilnehmern zu belauschen.

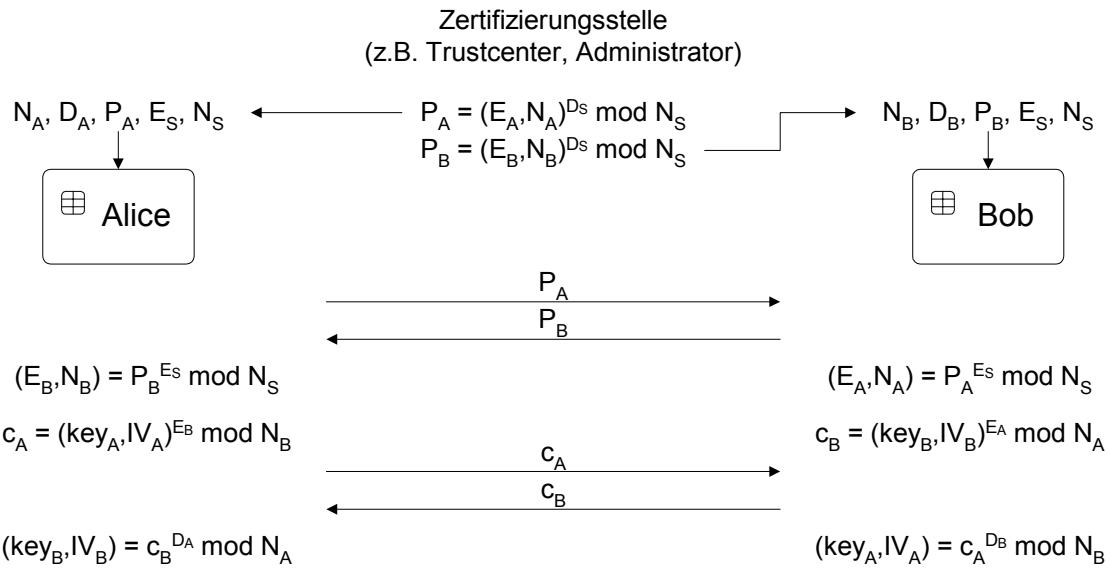

Abbildung 3: Schlüsselaustauschprotokoll

5 IMPLEMENTIERUNG

Nachdem im vorangegangenen Abschnitt der Schlüsselaustausch und die beiden Arten von kryptographischen Algorithmen erläutert wurden, sollen im folgenden Abschnitt die Wahl und Umsetzung der einzelnen Algorithmen beschrieben werden.

5.1 Tripel-DES

Beim symmetrischen Algorithmus haben wir uns für den DES-Algorithmus [5] entschieden, da dieser schon als VHDL-Beschreibung vorlag [3]. Weil Algorithmus aber schon mehrfach

durch sog. Brute-Force-Angriffe gebrochen wurde [4], haben wir uns dazu entschieden, die Triple-DES Erweiterung einzusetzen. Dabei wird der DES-Algorithmus dreimal nacheinander mit unterschiedlichen Schlüsseln auf den selben Klartext angewendet. Bei der Verschlüsselung wird der Klartext zuerst mit dem ersten Schlüssel verschlüsselt, dann mit dem zweiten Schlüssel entschlüsselt und noch einmal mit dem dritten Schlüssel verschlüsselt. Auf der Empfängerseite wird dies dann in umgekehrter Reihenfolge ausgeführt, um den Klartext wieder zu erhalten.

Um den DES-Algorithmus effektiv in Hardware umsetzen zu können, muß die genaue Funktionsweise bekannt sein. Der DES-Algorithmus verschlüsselt einen 64 Bit Klartextblock mit einem 64 Bit Schlüssel, daher wird dieser Algorithmus auch als Blockchiffre bezeichnet. Also sind längere Nahrichten in 64 Bit große Blöcke zu unterteilen. Der Klartextblock und der Schlüssel werden jeweils einer Eingangspermutation unterzogen, dabei verkürzt sich die wirksame Schlüssellänge auf 56 Bit. Daraufhin wird in 16 Durchläufen der Klartext mit dem Schlüssel verschlüsselt. Ein solcher Durchlauf ist in Abbildung 4 dargestellt. Der 64 Bit Klartextblock wird nach der Eingangspermutation in zwei 32 Bit Blöcke R und L unterteilt. Aus dem Schlüssel wird in jeder Runde ein neuer 48 Bit Teilschlüssel K generiert. Dieser Teilschlüssel wirkt in der Funktion f auf den R Block ein, das Ergebnis wird dann noch mit dem L Block XOR verknüpft. So wird der nachfolgende R Block gebildet, der nächst L Block besteht aus dem vorherenden R Block.

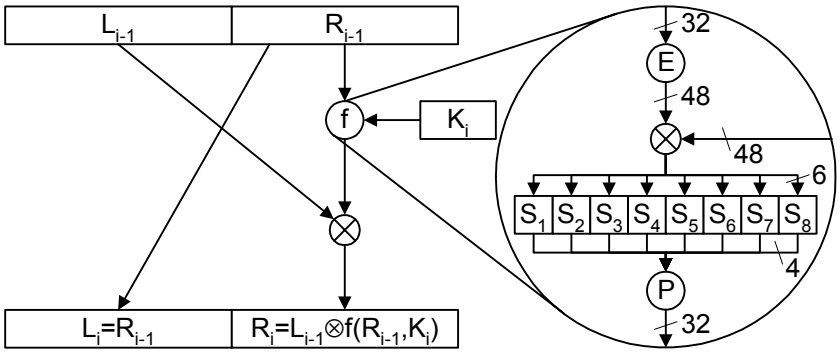

Abbildung 4: Eine Durchlauf beim DES Algorithmus

In der Funktion f wird der 32 Bit R-Block auf 48 Bit erweitert, dieser wird dann mit dem 48 Bit Teilschlüssel XOR verknüpft. Das Ergebnis wird an die acht sog. S-Boxen (S_1 bis S_8) verteilt, wobei jede S-Box aus dem 6 Bit Eingang einen 4 Bit Ausgangswert generiert. Die S-Boxen bestehen aus Tabellen die in [5] nachzulesen sind, wobei das erste und letzt Eingangsbit die Zeile und die mittleren 4 Bit die Spalte bestimmen, in der das 4 Bit Ergebnis für den Ausgang steht. Solche Tabellen lassen sich besonders gut in ROM-Zellen in einem FPGA umsetzen, ein Beispiel für eine S-Box ist in Abbildung 5 dargestellt. Die Ergebnisse der acht S-Boxen werden außerdem noch einer Permutation unterworfen.

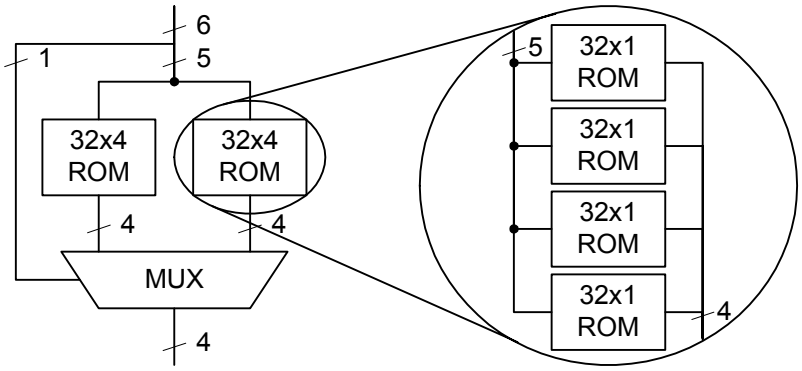

Abbildung 5: Aufbau einer S-Box

Da in dem verwendeten FPGA nur begrenzter Platz für dem Triple-DES-Coprozessor zur Verfügung stand haben wir uns dazu entschieden, die 16 Durchläufe nicht zu parallelisieren, sonder nacheinander auszuführen. Die Struktur dieser Implementierung ist in Abbildung 6 zu sehen. Diese Lösung benötigt 17 Takte für eine einfache DES Verschlüsselung, wobei der zusätzliche Takt zum Laden des Schlüssels benötigt wird.

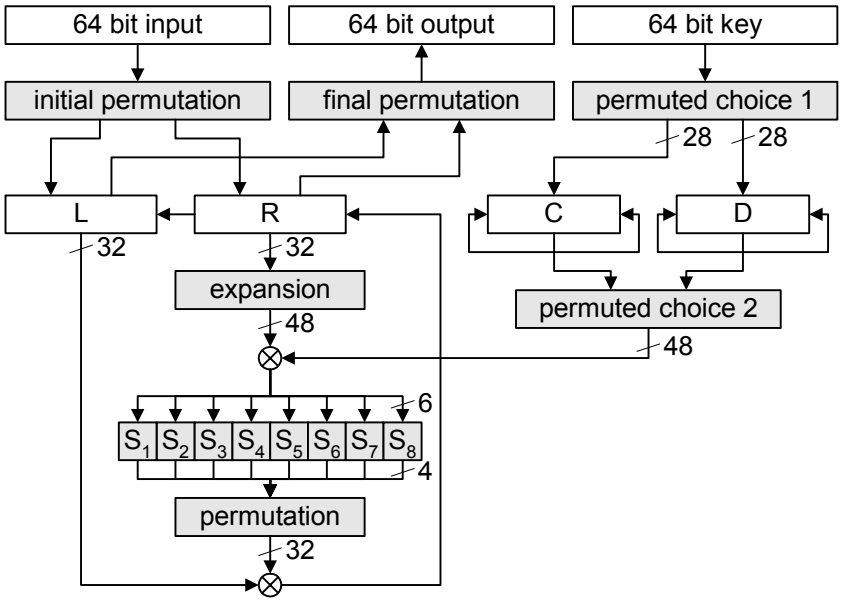

Abbildung 6: Struktur der DES Implementation

Wie schon erwähnt, haben wir aus Sicherheitsgründen eine Erweiterung von DES auf Triple-DES vorgenommen. Dazu wurde eine Zustandmaschine verwendet, welche die drei Verschlüsselungen mit den unterschiedlichen Schlüsseln steuert.

Für Blockchiffren gibt es unterschiedliche Betriebsarten [6] welche zusätzliche Sicherheit und andere Vorteile bieten. So hat der CFB-Modus den Vorteil, daß er selbstsynchronisierend ist. Das heißt, daß am Anfang der Kommunikation keine Einsynchronisierung der beiden Gesprächspartner erfolgen muß. Des weitern werden bei einem Übertragungsfehler die beiden Seiten wieder automatisch synchronisiert. Diese Betriebsart bietet auch noch die Möglichkeit, Bitbreiten die kleiner als 64 Bit sind zu verschlüsseln. So müssen keine Daten zwischengespeichert werden, bis die nötigen 64 Bit vorhanden sind.

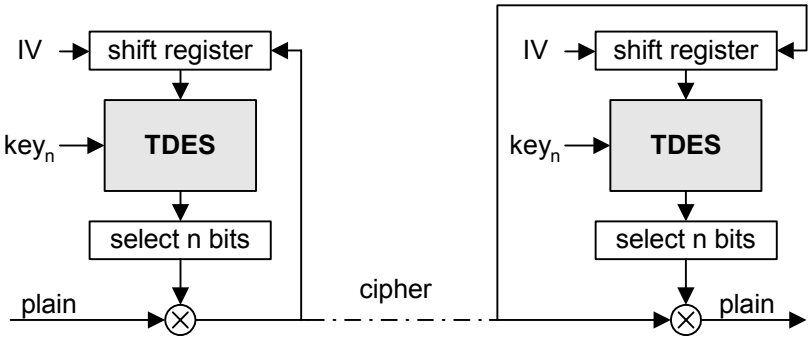

Abbildung 7: CFB-Modus

Aus diesen Gründen haben wir in unseren Triple-DES-Coprozessor den 8 Bit CFB-Modus implementiert, da die ISDN-Daten alle 125 µs in einem 8 Bit Block am digitalen Switch anliegen und auch in dieser Zeit verschlüsselt werden müssen. Wie in Abbildung 7 zu sehen ist, wird der Triple-DES Algorithmus im CFB-Modus als Pseudozufallszahlengenerator eingesetzt. Dadurch kann die Zahl, mit der die ISDN-Daten XOR-verknüpft werden schon vorausberechnet werden, was zur Abarbeitungsbeschleunigung beim Mikrocontroller führt, da dieser nach dem Schreiben der ISDN Daten in den Triple-DES-Coprozessor die verschlüsselten Daten im nächsten Schritt schon wieder auslesen kann, und nicht erst die gesamte Verschlüsselung abwarten muß. Für eine Triple-DES Verschlüsselung werden 51 Takte benötigt, was bei einer internen Taktfrequenz von f = 8 MHz immerhin 6,4 µs dauert. Da aber zeitgleich Daten für die Hin- und Rückrichtung zu bearbeiten sind, würde dies schon immerhin 12,8 µs dauern.

Damit der Mikrocontroller auf den Coprozessor zugreifen kann, mußte noch eine Interfaceschaltung implementiert werden, wie sie in Abbildung 8 zu sehen ist. Der Mikrocontroller hat Zugriff auf einen RAM Bereich im FPGA, wo er Daten schreiben und lesen kann. Hier werden die Schlüssel und die zu verschlüsselnden ISDN Daten abgelegt, nach der Verschlüsselung können die Daten wieder aus diesem Bereich gelesen werden.

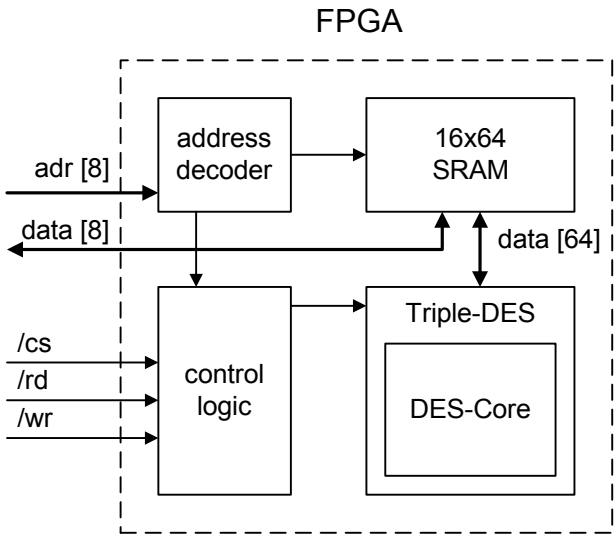

Abbildung 8: Mikrocontrollerinterface

5.2 Zufallszahlen

Um es einem Angreifer weiter zu Erschweren, die Kommunikation zu belauschen, werden die Sitzungsschlüssel nach jeder Kommunikation neu gebildet. Dazu verwenden wir einen Zufallszahlengenerator. Da wir für den symmetrischen Algorithmus das Triple-DES-Verfahren einsetzen, benötigen wir drei 64 Bit Schlüssel und einen 64 Bit Initialisierungsvektor. Diese Zahlen bilden den Sitzungsschlüssel für die nächste Kommunikation. Für die Erzeugung des Sitzungsschlüssel benutzen wir einen Linearen-Kongruenz-Generator, der wie folgt aussieht:

$$x_n = \left(2^{34} + 1\right) \cdot x_{n-1} + 1 \bmod 2^{64} \tag{1}$$

Da die Erzeugung der Zufallszahlen nicht zeitkritisch ist, wird dieser Algorithmus in Software auf dem Mikrocontroller ausgeführt. Jede Seite erzeugt ihren eigenen Sitzungsschlüssel, daher sind die Schlüssel für die Hin- und Rückrichtung unterschiedlich, wodurch die Sicherheit des Systems gesteigert wird.

5.3 RSA

Beim asymmetrischen Algorithmus haben wir uns für das RSA Verfahren [7] entschieden. In einer ersten Version wurde der Algorithmus in Software auf dem Mikrocontroller ausgeführt, allerdings wurden für eine Verschlüsselung mehrer Sekunden benötigt. Daher habe wir uns dazu entschlossen auch diesen Algorithmus in Hardware umzusetzen, denn für den Schlüsselaustausch wird dieser Algorithmus immerhin dreimal benötigt. Da das gewählte FPGA nicht groß genug war, um zwei Verschlüsselungs-Algorithmen parallel zu implementieren, haben wir die Rekonfigurierbarkeit des FPGA ausgenutzt und den jeweils benötigten Algorithmus zum Anwendungszeitpunkt in das FPGA geladen. Durch diese Maßnahme konnte auf zusätzliche Hardware verzichtet werden.

Beim RSA-Algorithmus stellt die modulare Exponentiation von langen Zahlen die wichtigste Funktion dar. Damit dieser Algorithmus sicher ist, werden wesentlich größere Bitbreiten als bei normalen Computern verwendet (bis zu 2048 Bit). Beim RSA Algorithmus besteht der öffentliche Schlüssel aus dem Exponenten E und dem Modulus N, wobei N aus dem Produkt zweier großer Primzahlen p und q gebildet wird, $N = p \cdot q$. Der private Schlüssel besteht aus demselben Modulus N und dem Exponenten D, der folgendes erfüllt $D = E^{-1} \bmod (p-1)(q-1)$. Die Zahlen E und N können ohne Sicherheitsrisiko an andere Nutzer verteilt werden, da sie ja den öffentlichen Schlüssel bilden. Die Zahl D dagegen muß geheim gehalten werden, sie bildet den privaten Schlüssel des Nutzers.

Um eine Nachricht m zu verschlüsseln, muß $c = m^E \bmod N$ berechnet werden. Die Entschlüsselung von c erfolgen dann mit $m = c^D \bmod N$. Die Erzeugung der Zahlen (p, q, N, D, E) ist sehr rechenaufwendig, daher erfolgt dies auch nur einmal für jeden Nutzer. Dies kann von der Zertifizierungsstelle vorgenommen werden, die dabei gleich den öffentlichen Schlüssel signieren kann. Da die Ver- und Entschlüssel mit derselben Funktion ausgeführt wird, allerdings mit unterschiedlichen Exponenten, kann der RSA-Algorithmus zur Authentifizierung und Verschlüsselung eingesetzt werden.

Die Zahl n repräsentiert die Anzahl der Bitstellen, die für die Darstellung von N notwendig sind. Die einfachste Methode, um eine modulare Exponentiation auszuführen, ist die

Quadrier-und-Multiplizier-Technik. Um b^E mod N zu berechnen, kann die Exponentiation in eine Serie von modularen Multiplikationen aufgeteilt werden, wobei b, E und N drei n-Bit breite ganze Zahlen sind mit $b, E \in [0, N\text{-}1]$.

Algorithmus 1

```
Input : b, E, N; 0 < b, E < N
Output: b^E mod N
Y = 1
FOR i = (n - 1) DOWNTO 0 LOOP
    Y = Y*Y mod N
    Y = Y*B mod N IF (E_i == 1)
RETURN Y
```

Für die Berechnung dieses Algorithmus werden durchschnittlich 1,5 n modulare Multiplikationen benötigt, im schlechtesten Fall sogar 2 n.

Für die Implementierung des RSA-Algorithmus konnte auf eine Studienarbeit [8] zurückgegriffen werden, in der die modulare Exponentiation basierend auf der optimierten Montgomery-Multiplikation in eine VHDL-Beschreibung umgesetzt wurde. Dabei kann diese Beschreibung in den Werten w und n parametrisiert werden, wobei mit w die Bitbreite des Multiplizierers und mit n die Bitbreite des Modulus N gemeint ist. Tabelle 1 zeigt die Zeit, die benötigt wird, um eine modulare Exponentiation auszuführen. Der Coprozessor arbeitet dabei mit einem Takt von f = 16 MHz. Für unsere Aufgabe benötigen wir drei modulare Exponentiationen mit (n = 256, w = 16), dies kann also in ca.37 ms ausgeführt werden.

w	$\log_2(n)$		
	256	512	1024
8	48.96	392.4	3143
16	12.24	98.1	785.6
32	3.06	24.5	196.4

Tabelle 1: Zeit [ms] für die Berechnung der Modularen Exponentiation

6 ZUSAMMENFASSUNG

In diesem Beitrag konnte gezeigt werden wie ein vorhandenes ISDN-Gerät durch einfache Erweiterungen auch für andere Aufgaben eingesetzt werden konnte. Durch die Implementierung der Verschlüsselungsalgorithmen in ein FPGA ist es außerdem sehr einfach, diese Algorithmen durch andere Verfahren zu ersetzen, so daß auch in Zukunft dieses System gegen Angriffe gesichert werden kann.

Damit unser Prototyp wirklich gegen Angriffe sicher ist, müßte der RSA Schlüssel mindestens auf 1024 Bit erweitert werden, denn erst ab diese Schlüssellänge sind erfolgreiche Angriffe unwahrscheinlich. Das kann allerdings in einigen Jahren mit fortschreitender Entwicklung der Computertechnik anders aussehen . Außerdem ist es zur Zeit nicht möglich, zwei Telefongespräche gleichzeitig zu verschlüsseln, obwohl der ISDN Basisanschluß zwei Datenkanäle zur Verfügung stellt. Das liegt daran, daß bei einem laufendem Gespräch das FPGA nicht mit dem asymmetrischen Algorithmus geladen werden kann, ohne das erste Gespräch zu unterbrechen. Um dies zu ermöglichen müßte das System entweder um ein zweites

FPGA erweitert werden, oder ein größeres FPGA zum Einsatz kommen, in dem dann beide Algorithmen gleichzeitig zur Ausführung kommen können.

7 LITERATURANGABE

[1] Kanbach, A., Körber, A.: ISDN - Die Technik, Hüthig Verlag, Heidelberg, 1999

[2] Schneier, B.: Angewandte Kryptographie, Addison Wesley, Bonn, 1996

[3] Ploog, H., Schmalisch, M., Timmermann, D.: Kryptomodul für schnelle DES basierende Verschlüsselungssysteme, 2. IuK Tage, Rostock, 1999

[4] http://www.rsasecurity.com/rsalabs/des3/

[5] National Bureau of Standards FIPS Publication 46: Data Encryption Standard, 1977

[6] National Bureau of Standards FIPS Publication 81: DES modes of operation, 1980

[7] Rivest, R.L., Shamir, A., Adleman, L.: A Method of obtaining digital signature and public key cryptosystems, Comm. Of ACM, Vol.21, No.2, pp.120-146, Feb.1978

[8] Wiencke, C.: Hardwareoptimale Implementierung der Montgomery-Multiplikation, Universität Rostock, Studienarbeit, 1999