SICHERHEITSERWEITERUNG EINES ISDN-GERÄTES DURCH REKONFIGURIERBARE LOGIK

Mathias Schmalisch, Hagen Ploog, 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: sm15@e-technik.uni-rostock.de

Kurzfassung: Das Integrated Services Digital Network (ISDN) ist der Standard für das leitungsgebundene Telefonieren. Dieses öffentliche Netzwerk wurde 1984 entwickelt, allerdings sind keine Möglichkeiten vorgesehen, die Informationen, die über das ISDN übertragen werden, gegen Angreifer abzusichern. Das Sichern der Informationen muß daher außerhalb des ISDN Protokolls erfolgen. In diesem Beitrag stellen wir unsere Erfahrungen dar, wie rekonfigurierbare Logik zum Sichern von existierenden ISDN-Endgeräten eingesetzt werden kann.

1 EINFÜHRUNG

In vielen Firmen wird das ISDN für Telekommunikationsdienste, wie Telefonieren, Faxen, Videokonferenzen, Datenübertragung und ähnliches eingesetzt. Da das ISDN keine Möglichkeiten zur sicheren Übertragung von Informationen unterstütz, muß dies außerhalb des normalen ISDN-Protokolls erfolgen. Diese kann entweder durch eine spezielle Hardware oder durch Software auf einem Computer mit einer ISDN Karte erfolgen. Die letztere Möglichkeit ist nicht so gut geeignet, wenn auf Mobilität Wert gelegt wird. Außerdem ist es recht unhandlich jedesmal einen Computer zu booten, um ein sicheres Telefongespräch zu führen. Falls auch noch die vorhandenen ISDN-Geräte weiterverwendet werden sollen, ist es nötig eine spezielle ISDN-Karte zu kaufen, da die normalen ISDN-Karten keine Möglichkeit haben eine Vermittlungsstelle zu simulieren, was dafür aber notwendig ist.

Aus diesen Gründen haben wir uns dafür entschieden eine spezielle Hardware zu entwickeln, welche die Sicherung der zu übertragenen Informationen vornimmt. In diesem Beitrag präsentieren wir unsere Erfahrungen mit dem Einsatz von rekonfigurierbarer für die Sicherung von ISDN-Kommunikation gegen Angreifern bei Punkt zu Punkt Verbindungen. Für diese Aufgabe stand uns ein ISDN Least Coast Router (LCR) zu Verfügung, welcher durch eine zusätzliche Platine mit einem FPGA erweitert wurde. In diesem FPGA werden dann die kryptographischen Funktion ausgeführt, da der Mirkocontroller nicht schnell genug ist, um die Algorithmen in Software auszuführen.

2 ARCHITEKTURBESCHREIBUNG

Der LCR, welchen wir erweitert haben, enthielt einen V25 Mikrocontroller, RAM, EPROM und zwei ISDN-Anschlüsse. Ein ISDN-Anschluß wird mit dem Endgerät verbunden und der andere mit der Vermittlungsstelle (VSt). Um das FPGA an den Mikrocontrollerbus anzuschließen wurde die Zusatzplatine mit dem FPGA über den EPROM-Sockel an den LCR angeschlossen, dadurch konnte zusätzliche Verdrahtung eingespart werden.

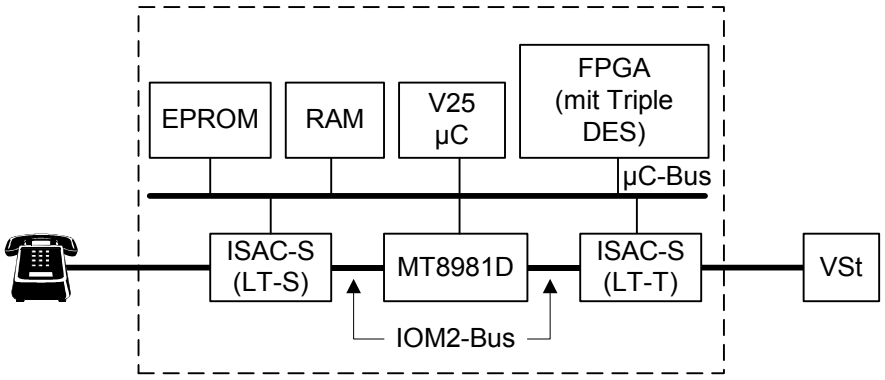

Abbildung 1: Aufbau des ISDN-Krypters

Der Aufbau einer sicheren Verbindung erfolgt in drei Schritten. Als erstes wird mit Hilfe des Autodetects die Existenz eines baugleichen Gerätes an der Gegenseite überprüft. Als nächstes wird ein Schlüsselaustauschprotokoll durchgeführt, mit dem die Sitzungsschlüssel sicher an die Gegenseite übertragen werden. Im dritten Schritt erfolgt dann die Verschlüsselung der eigentlichen Nutzdaten.

Für den Autodetect wird eine 4 Byte Sequenz vom Anrufer zum Gerufenen geschickt. Die Wahrscheinlichkeit, daß diese Sequenz während einer normalen Kommunikation auftritt ist 2^{-32}, diese Sequenz wird aber nur kurz am Anfang eines Verbindungsaufbaus benötigt. Wenn auf der Empfängerseite diese Sequenz empfangen wird, sendet der ISDN-Krypter als Antwort die selbe Sequenz mehrmals hintereinander zurück, um so den Empfang zu bestätigen. Durch das mehrfache Zurücksenden der Sequenz wird sichergestellt, daß bei Störungen auf dem Übertragungskanal die Sequenz trotzdem empfangen werden kann.

Nach einem erfolgreichen Autodetect kann eine verschlüsselte Verbindung aufgebaut werden, diese wird dem Nutzer durch eine optische Anzeige verdeutlicht. Daraufhin kann dann der Schlüsselaustausch erfolgen.

Da der Schlüsselaustausch und die Ver-/Entschlüsselung der Daten zeitkritisch ist, verwenden wir dafür das FPGA zweimal hintereinander. Zuerst wird das FPGA mit dem asymmetrischen Algorithmus für den Schlüsselaustausch geladen. Nach dem erfolgereichen Übertragen der Sitzungsschlüssel wird das FPGA mit dem symmetrischen Algorithmus für die Ver-/Entschlüsselung der eigentlichen Nutzdaten neugebootet.

3 BESCHREIBUNG DES SICHERHEITSMODELLS

Die symmetrischen Algorithmen habe den großen Vorteil, daß sie wesentlich schneller sind als asymmetrische Algorithmen. Daher setzen wir auch einen solchen Algorithmus ein, um die Nutzdaten zu Verschlüsseln. Allerdings haben diese Algorithmen den Nachteil, daß auf beiden Seiten der selbe Schlüssel benötigt wird. Das Problem besteht darin, diesen Schlüssel sicher an die Gegenseite zu übermitteln. Für diese Aufgabe kommt ein asymmetrischer Algorithmus zu Einsatz. Da hierbei zwei verschiedene Verschlüsselungsarten zum Einsatz kommen, wird dieses Verfahren auch als hybrides Verfahren bezeichnet.

Um bei einem hybriden System sicher zu gehen, daß jede Seite den öffentlichen Schlüssel der Gegenseite erhält, und nicht den Schlüssel eines Angreifers, ist es notwendig eine Authentifizierung durchzuführen. Normalerweise wird diese Authentifizierung durch ein Trustcenter vorgenommen. Da wir aber nur eine Punkt zu Punkt Verbindung aufbauen wollen, muß eine andere Methode gewählt werden. Für den ISDN-Krypter haben wir eine Methode gewählt, die in Abbildung 2 dargestellt ist. Dabei erzeugt die Zertifizierungsstelle, das kann ein Trustcenter oder auch ein Administrator sein, das Schlüsselpaar für jeden Nutzer. Dabei wird der öffentliche Schlüssel jedes Nutzers von der Zertifizierungsstelle signiert, in dem er mit dem privaten Schlüssel der Zertifizierungsstelle verschlüsselt wird. Jeder Nutzer erhält dann seinen privaten Schlüssel, seinen signierten öffentlichen Schlüssel und den öffentlichen Schlüssel der Zertifizierungsstelle. Beim Austausch der öffentlichen Schlüssel wird nur der signierte Schlüssel übermittelt. Aus diesem kann dann mit dem öffentlichen Schlüssel der Zertifizierungsstelle, der öffentlichen Schlüssel der Gegenseite berechnet werden. Somit bräuchte ein möglicher Angreifer ebenfalls einen signierten Schlüssel, um die Kommunikation erfolgreich abhören zu können.

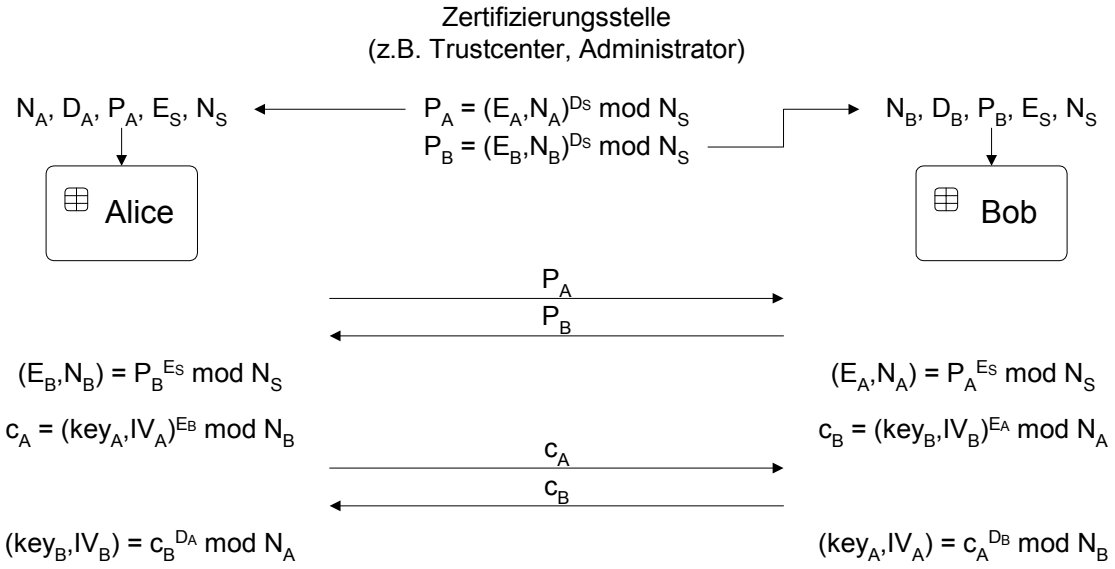

Abbildung 2: Schlüsselaustauschprotokoll

4 IMPLEMENTATIONSDETAILS

4.1 Zufallszahlen

Einige Dinge, die nicht zeitkritisch sind können vom Mikrokontroller erledigt werden, so auch die Erzeugung von Zufallszahlen. Für unseren ISDN-Krypter benötigen wir drei 64 Bit Schlüssel für die Triple-DES Verschlüsselung und ein 64 Bit Zahl für den Initialisierungsvektor. Diese Zahlen bilden dann den Sitzungsschlüssel für das nächste Telefongespräch. Für die Erzeugung der Zufallszahlen wir ein Linearer Kongruenz Generator der wie folgt aussieht verwendet:

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

Dieser Sitzungsschlüssel wird nach dem Verbindungsaufbau mit dem öffentlichen Schlüssel der Gegenseite verschlüsselt und dann übertragen. Jede Seite erzeugt ihren eigenen Sitzungsschlüssel, so daß für die Hin- und Rückrichtung unterschiedliche Schlüssel zum Einsatz kommen, dadurch erhöht sich die Sicherheit das Systems.

4.2 Modulare Exponentiation

Für den asymmetrischen Algorithmus benutzen wir das RSA Verfahren [1]. Bei diesem 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 dem selben Modulus N und dem Exponenten D, der folgendes erfüllt $D = E^{-1} \bmod (p\text{-}1)(q\text{-}1)$. 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, den 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 dann gleich noch den öffentlichen Schlüssel signiert. Da die Ver- und Entschlüssel mit der selben Funktion ausgeführt wird, allerdings mit unterschiedlichen Exponenten, kann der RSA Algorithmus zur Authentifizierung und Verschlüsselung eingesetzt werden.

Die Zahl n repräsentiert die Anzahl Bitstellen, die für die Darstellung von N notwendig sind. Die einfachste Methode um eine modulare Exponentiation auszuführen ist die "quadrier und multiplizier" Technik. Um $b^E \bmod N$ 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.

1985 hat P. L. Montgomery eine Algorithmus für die modulare Multiplikation vorgeschlagen $A \cdot B \bmod N$ [2].

Montgomery Multiplikation: Die Zahlen A, B sind Elemente von Z_N, wobei Z_N eine Menge von ganzen Zahlen zwischen [0, N-1] ist. R ist eine ganze Zahl relativ prim zu N, daß heißt gcd(R, N) = 1 und $R > N$. Dann kann der Montgomery Algorithmus berechnet werden

$$\text{ModProd}(A, B) = A \cdot B \cdot R^{-1} \bmod N \tag{2}$$

Wenn R eine Potenz von 2 ist, dann kann die Division durch eine einfache Schiebeoperation ersetzt werden. Für die Berechnung des Montgomery Produkts wird noch eine weiterer Wert benötigt N' mit $1 < N' < R$ und $N' = -N^{-1} \bmod R$. Der Algorithmus ModProd sieht wie folgt aus:

Algorithmus 2

Input : $\overline{a}, \overline{b}, R^{-1}, N, N'$

Output: $\overline{a} \cdot \overline{b} \cdot R^{-1} \bmod N$

t = $\overline{a} \cdot \overline{b}$

m = t N' mod R

u = (t + m · N) / R

IF u ≥ N THEN u = u – N

RETURN u

Eine VHDL Beschreibung für einen Coprozessor für die modulare Exponentiation basierend auf der optimierten Montgomery Multiplikation ist gegeben in [5]. 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. Wobei der Coprozessor mit einem Takt von f = 16 MHz arbeitet. Für unsere Aufgabe benötigen wir drei Modulare Exponentiationen mit (n = 256, w = 16), dies kann also in ~37ms 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

4.3 Triple-DES

Für den symmetrischen Algorithmus haben wir den Data Encryption Standard (DES) gewählt [3], mit der Erweiterung auf Triple-DES. Mit Triple-DES ist gemeint, daß der DES Algorithmus drei mal nacheinander angewendet wird. Bei einer Verschlüsselung wird die Nachricht zuerst mit dem ersten Schlüssel verschlüsselt, mit dem zweien Schlüssel entschlüsselt und zum Schluß noch einmal mit dem dritten Schlüssel verschlüsselt. Die Entschlüsselung erfolgt in umgekehrter Reihenfolge.

Der DES Algorithmus verschlüsselt eine 64 Bit Klartextblock mit einem 64 Bit Schlüssel. Der Klartextblock und der Schlüssel werden zuerst auf eine Eingangspermutation angewendet, dabei verkürzt sich der wirksame Schlüssel auf 56 Bit. Dann wird in 16 Runden ein Teilschlüssel generiert, der auf den Klartextblock einwirkt. Eine Runde ist in Abbildung 3 dargestellt. Dabei wird der 64 Bit Eingansblock in zwei 32 Bit Teile R und L aufgeteilt. Der nachfolgende R Block wird mit der Funktion f und dem 48 Bit Teilsschlüssel K gebildet, der dann noch mit dem L Block XOR verknüpft wird. Der nachfolgende L Block besteht aus dem vorhergehenden R Block.

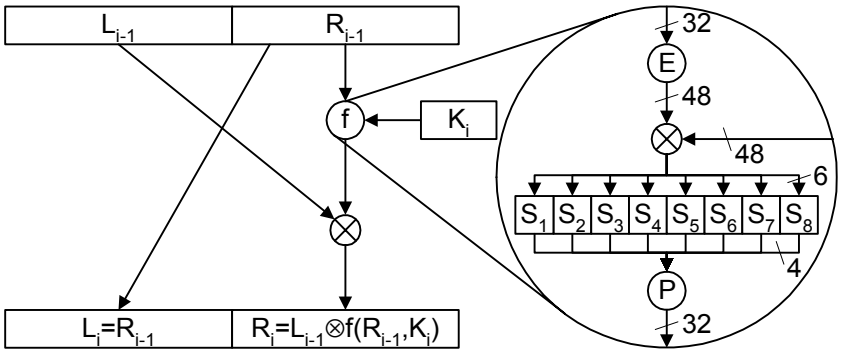

Abbildung 3: Eine Runde beim DES Algorithmus

In der Funktion f wird der R Block auf 48 Bit erweitert, danach wird dieser mit dem 48 Bit Teilschlüssel K XOR verknüpft. Diese Ergebnis wird dann an die acht S-Boxen weitergegeben. Wobei in jeder S-Box der 6 Bit Eingang auf einen 4 Bit Ausgang abgebildet wird. Der Aufbau einer S-Box in einem FPGA ins in Abbildung 4 dargestellt.

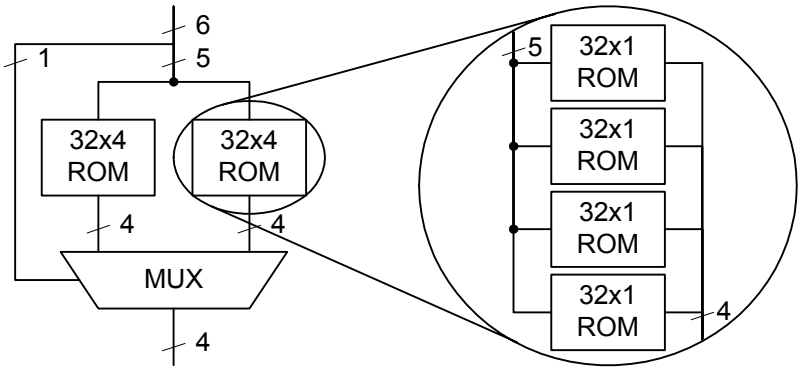

Abbildung 4: Aufbau einer S-Box

Jede S-Box benötigt 10 konfigurierbare Logikblöcke (CLB), wobei acht auf die RAM-Zellen fallen und zwei auf den Multiplexer.

Da für das ISDN nur geringe Anforderungen an die Datenrate gestellt werden, ist eine Pipelining nicht notwendig. Das heißt, daß die 16 Runden nacheinander ausgeführt werden können, wodurch sich ein kleineres Design erzielen läßt. Die Struktur dieser Implementation ist einmal in Abbildung 5 dargestellt.

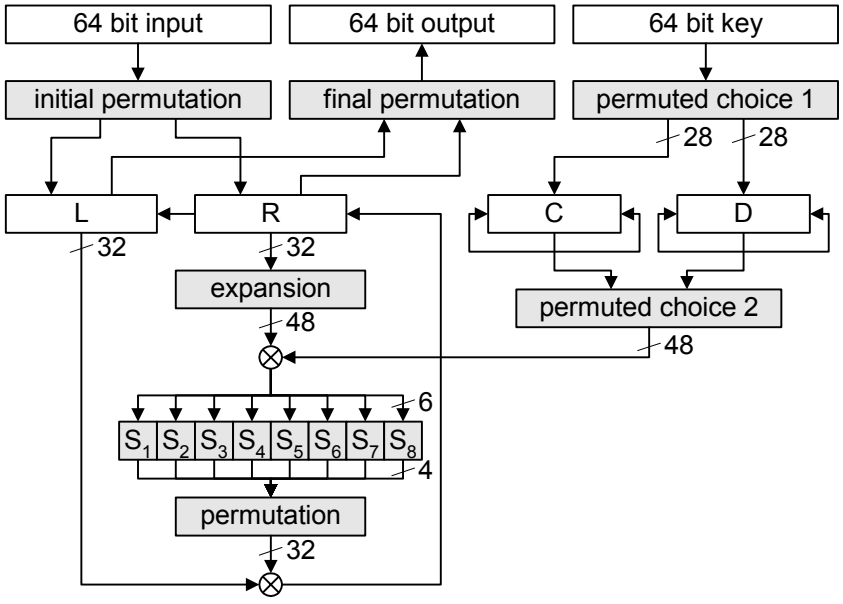

Abbildung 5: Struktur der DES Implementation

Es ist weithin bekannt, das der einfache DES Algorithmus schon mehrfach durch Brute Force Angriffe berochen wurde [6]. Daher haben wir uns auch dazu entschieden, die Triple-DES Erweiterung einzusetzen. Dazu werden drei 64 Bit Schlüssel benötigt, diese können vom Mikrokontroller aus in einem Zwischenspeicher im FPGA geschrieben werden. Außerdem habe wir uns für den 8 Bit Cipher Feedback Modus (CFB) als Betriebsart [4] für den Triple-DES entschieden, da die ISDN Daten in 8 Bit Paketen ankommen. In diesem Modus wird der Triple-DES Algorithmus als Pseudeozufallszahlengenerator eingesetzt.

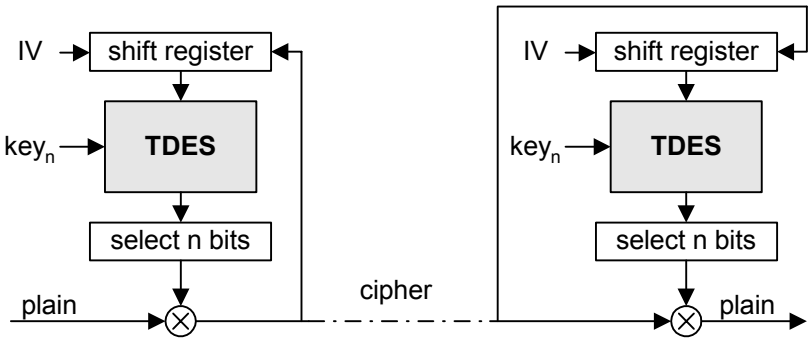

Abbildung 6: CFB Modus

Dieser Modus hat den Vorteil, das er selbstsynchronisierend ist, daher braucht am Anfang keine Einsynchronisierung der beiden Seiten vorgenommen werden. Des weiteren kann die 8 Bit Zahl, mit der die ISDN Daten XOR verknüpft werden, schon vorausberechnet werden.

Die Implementation des einfachen DES Algorithmus benötigt 17 Takte für die Verschlüsselung, da wir diesen aber drei mal hintereinander anwenden vergehen für den Triple-DES also 51 Takte. Bei einer Taktfrequenz von f = 8 MHz, vergehen 6,4 µs für eine Ver-/Entschlüsselung. Dies reicht für das ISDN vollkommen aus, da nur alle 125 µs neu ISDN Daten zu verarbeiten sind. Mit diesem Triple-DES Coprozessor könnten sogar etwa 8 ISDN Kanäle parallel verschlüsselt werden. Da der ISDN Basisanschluß aber nur zwei Kanäle bereitstellt reicht dies also aus um unsere Anforderungen zu erfüllen.

5 ZUSAMMENFASSUNG

Wie in diesem Beitrag aufgezeigt werden konnte, ist mit einer einfachen Erweiterung eines vorhanden ISDN Gerätes möglich, die Informationen, welche über das ISDN übertragen werden, gegen das Belauschen von Dritten zu sichern.

In unserem Prototypen benutzten wir nur ein RSA Schlüssel der Länge 256, um aber wirklich sicher gehen zu können, daß dieser nicht gerochen wird, sind mindestens 1024 Bit notwendig. An dieser Stelle können noch einige Verbesserungen vorgenommen werde.

6 LITERATURANGABE

[1] 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

[2] Montgomery, P.L.: Modular Multiplication without Trial Division, Mathematics of Computation, Vol.44, No.170, pp.519-521, 1985.

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

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

[5] Wienke, C.: Hardwareoptimale Implementierung der Montgomery-Multiplikation, University of Rostock, student work (in german), 1999

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