Kryptographiecoprozessor zur Verschlüsselung von ISDN-Daten

Mathias Schmalisch · Hagen Ploog · Frank Grassert · Dirk Timmermann

Universität Rostock

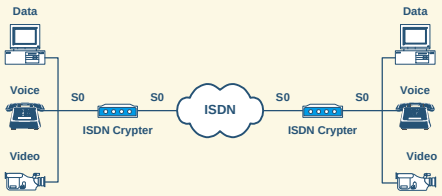
Übersicht

- Motivation
- Einsatz vorhandener Hardware
- Kryptographie
- Implementierung
- Zusammenfassung

© Institut für Angewandte Mikroelektronik und Datentechnik
Fachbereich Elektrotechnik und Informationstechnik, Universität Rostock
Dipl.-Ing. Mathias Schmalisch
mathias.schmalisch@e-technik.uni-rostock.de

Motivation

- Sichern von ISDN-Diensten gegen Belauschen
- Benutzerfreundlich und transparent zu gängigen ISDN-Geräten



© Institut für Angewandte Mikroelektronik und Datentechnik
Fachbereich Elektrotechnik und Informationstechnik, Universität Rostock
Dipl.-Ing. Mathias Schmalisch
mathias.schmalisch@e-technik.uni-rostock.de

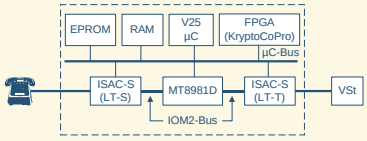
Übersicht

- Motivation
- **Einsatz vorhandener Hardware**
- Kryptographie
- Implementierung
- Zusammenfassung

© Institut für Angewandte Mikroelektronik und Datentechnik
Fachbereich Elektrotechnik und Informationstechnik, Universität Rostock
Dipl.-Ing. Mathias Schmalisch
mathias.schmalisch@e-technik.uni-rostock.de

Ausgangssituation

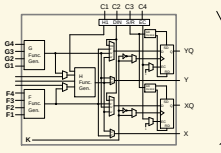
- Least-Cost-Router mit 16 bit Mikrocontrollersystem
- Problem:
  - Mikrocontroller zu langsam für sichere Ver-/Entschlüsselung
- Lösung:
  - Erweiterung durch Kryptographiecoprozessor im FPGA



© Institut für Angewandte Mikroelektronik und Datentechnik
Fachbereich Elektrotechnik und Informationstechnik, Universität Rostock
Dipl.-Ing. Mathias Schmalisch
mathias.schmalisch@e-technik.uni-rostock.de

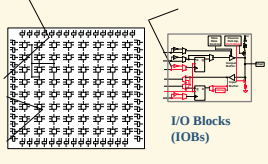
Einführung: FPGA Xilinx XC4000

Configurable Logic Blocks (CLBs)



Programmable Interconnect

- **Große Dichte → 1 Mio. System Gatter**
- **SRAM basierende LUT**
- **Array Struktur**
- **interne Tri-States**
- **beliebig rekonfigurierbar in wenigen ms**



I/O Blocks (IOBs)

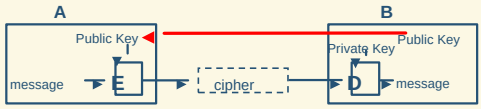
© Institut für Angewandte Mikroelektronik und Datentechnik
Fachbereich Elektrotechnik und Informationstechnik, Universität Rostock
Dipl.-Ing. Mathias Schmalisch
mathias.schmalisch@e-technik.uni-rostock.de

Übersicht

- Motivation
- Einsatz vorhandener Hardware
- **Kryptographie**
- Implementierung
- Zusammenfassung

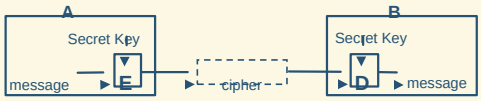
Asymmetrische Systeme

- Verschiedene Schlüssel auf jeder Seite
  - Verschlüsselung : $c = f(m, \text{public key})$
  - Entschlüsselung : $m = f^{-1}(c, \text{private key})$
- Langsamer als symmetrische Systeme (~ 100 mal)



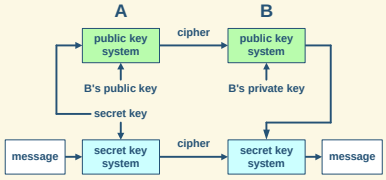
Symmetrische Systeme

- Ein geheimer Schlüssel auf beiden Seiten
  - Verschlüsselung : $c = f(m, k)$
  - Entschlüsselung : $m = f^{-1}(c, k)$
- Problem: sicherer Schlüsselaustausch



Hybride Systeme

- Benutzung des asym. Systems für den Sitzungsschlüssel
  - Während des Verbindungsaufbaus
  - Unkritische Geschwindigkeitsanforderung
- Benutzung des sym. Systems für den Datenstrom



Übersicht

- Motivation
- Einsatz vorhandener Hardware
- Kryptographie
- **Implementierung**
- Zusammenfassung

Asymmetrisches System: RSA

- RSA : Modulare Exponentiation
  - Verschlüsselung : $E(m) = m^E \bmod N = c$ • public key : E, N
  - Entschlüsselung : $D(c) = c^D \bmod N = m$ • private key : D

Sicherheit ist abhängig von der Länge der Zahl N (> 512 bits)

Realisation: "Square and multiply" (Knuth): $Y = B^E \bmod N$

```
Y = 1
FOR i = log2(E) DOWNTO 0 DO
    Y = Y * Y mod N
    Y = Y * B mod N if (Ei == 1)
END
```

=> Exponentiation reduziert auf ~ 1.5 n modulare Multiplikationen, $n = \log_2(N)$

Multiple precision

- Zwischenergebnisse müssen in externen RAM der Breite w gespeichert werden
  - Neue Basis W=2^w anstelle der Basis 2
  - Transformiere jede Integerzahl X in die neue Basis

$$X = \sum_{i=0}^{x-1} x_i 2^i = \sum_{i=0}^{s-1} d_i (2^w)^i = \sum_{i=0}^{s-1} d_i W^i$$

- Herz des Algorithmus ist eine $w \cdot w$ Bit Multiplikation
  - z Anzahl Takte für die Ausführung einer $w \cdot w$ Bit Multiplikation
- VHDL-Model kann parametrisiert werden (n, w, z)

Ergebnisse: RSA

Zeit für die Berechnung einer mod. Exponentiation: $\frac{3 \times z \times (n^3 - n^2)}{f_{clk} \times w^2}$

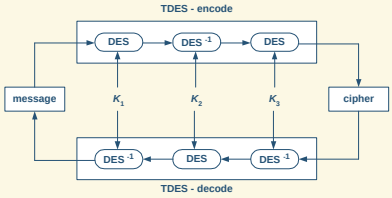
Ergebnisse [s] für 3.5 MHz

w \ n	256	512	1024
8	0.22	1.79	14.37
16	0.056	0.45	3.59
32	0.014	0.11	0.90
64	0.0035	0.028	0.22

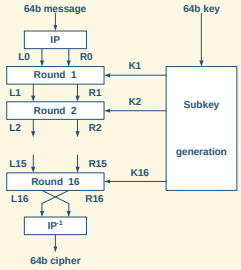
Realistische Werte

Symmetrisches System: Triple DES (TDES)

- TDES : kaskadiertes DES, zur Zeit noch nicht gebrochen

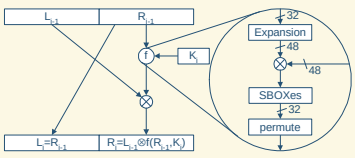


DES



- 64 Bit Input
- 64 Bit Output
- 16 Runden
  - $L_i = R_{i-1}$
  - $R_i = L_{i-1} \otimes f(R_{i-1}, K_i)$
- Jede Runde hat eigenen Teilschlüssel K_i
- Runden sind unabhängig voneinander

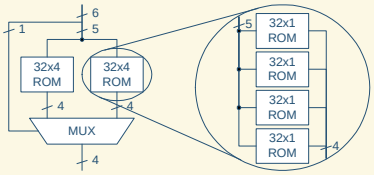
Eine Runde



- Die f-Funktion ist das kryptographische Herz beim DES
  - Erweitere R auf 48 Bit
  - XOR Ergebnis mit K_i
  - Ausführen der SBoxen (Reduzieren auf 32 Bits)
  - Permutiere Ergebnis der SBoxen

SBox

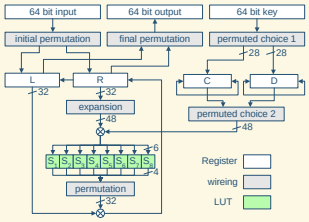
- Eine SBox läßt sich am besten mit LUT's realisieren



- 10 CLB's / SBox
  - insgesamt 80 CLB's

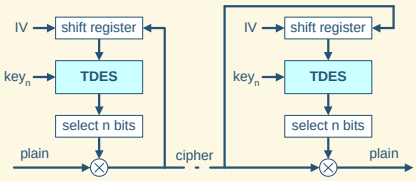
DES: Rekursive Implementation

• Rekursive Implementierung für kleine Architektur



Autosynchronisation mit CFB Modus

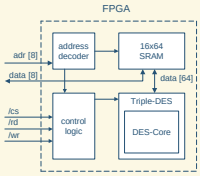
• Cipher Feed Back Mode (CFB)



Ergebnisse: TDES

- FPGA: XC4010e-3
  - 380 CLB von 400 = 95% Auslastung
  - Aber: 209 FF von 800 = 26%
- TDES: 55 Takte – 8 Bit CFB-Modus
  - Datenrate @ 8 MHz: 1.16 MBits/s

ISDN B-Kanal: 64 KBit/s pro Richtung



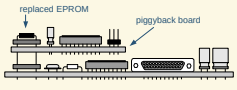
=> TDES-Core kann theoretisch bis zu 8 B-Kanäle parallel verschlüsseln

Ausnutzung der Rekonfigurierbarkeit

- Ziel: Vorhandene Hardwareresourcen ausnutzen
- Benutzen von zeitliche getrennten Algorithmen
- Zwei Arbeitsphasen
  - RSA nach dem Neustart
  - FPGA rebooten nach erfolgreichem Schlüsselaustausch
  - Triple DES für die Verschlüsselung des Datenstroms
- Rebooten des FPGA mit neuem Inhalt in ungefähr 35 ms

Prototype

- Erweiterung des Systems über den EPROM-Sockel
- FPGA XC4010e-3



Übersicht

- Motivation
- Einsatz vorhandener Hardware
- Kryptographie
- Implementierung
- **Zusammenfassung**

Zusammenfassung

- ISDN Echtzeitanforderungen lasten FPGAs nicht aus
- Kryptographische Algorithmen passen auch in kleine FPGAs
- Rebooten des FPGAs hält die Kosten des Systems niedrig
- Verringerte zusätzliche Verdrahtung durch Huckepack-Board
- Durch Einsatz von FPGAs Update auf neu Algorithmen möglich