CoNe Freiburg

IIF INSTITUT FÜR INFORMATIK FREIBURG

Systeme II

3. Die Datensicherungsschicht

Christian Schindelhauer
Technische Fakultät
Rechnernetze und Telematik
Albert-Ludwigs-Universität Freiburg

Version 15.05.2017

UNI FREIBURG

CoNe Freiburg

Fehlererkennung: CRC

- Effiziente Fehlererkennung: Cyclic Redundancy Check (CRC)
- Praktisch häufig verwendeter Code
  - Hoher Fehlererkennungsrate
  - Effizient in Hardware umsetzbar
- Beruht auf Polynomarithmetik im Restklassenring Z_2
  - Zeichenketten sind Polynome
  - Bits sind Koeffizienten des Polynoms

$a+b = b+a$ $\quad a*b$

Inverses Elmt

Distr.

Ass

$1+x = 0$

$1 = -1$

$Z_2 = \{0, 1\}$

xor	+	0	1
	0	0	1
	1	1	0

$1+1 = 0$

And	*	0	1
	0	0	0
	1	0	1

Rechnen in Z_2

$123 = 1 \cdot 10^2 + 2 \cdot 10^1 + 3 \cdot 10^0$

CxXIII

- Rechnen modulo 2:
- Regeln:
  - Addition modulo 2 = Xor = Subtraktion modulo 2
  - Multiplikation modulo 2 = And

```
  101101
xor 011001
-----------
  110100
```

A	B	A + B
0	0	0
0	1	1
1	0	1
1	1	0

A	B	A - B
0	0	0
0	1	1
1	0	1
1	1	0

A	B	A ⊙ B
0	0	0
0	1	0
1	0	0
1	1	1

- Beispiel: $0 + (1 \cdot 0) + 1 + (1 \cdot 1) =$ 0

0 1

CoNe Freiburg

Polynomarithmetik modulo 2

$a x + b = 0$

- Betrachte Polynome über den Restklassenring Z_2
  - $p(x) = a_n x^n + \ldots + a_1 x^1 + a_0$
  - Koeffizienten a_i und Variable x sind aus {0,1}
  - Berechnung erfolgt modulo 2

$a_i \in \{0,1\}$

- Addition, Subtraktion, Multiplikation, Division von Polynomen wie gehabt

$0 \cdot x^2 + 1 \cdot x + 1 + 1 \cdot x^3 + 1 \cdot x^2 + 0 \cdot x + 1$

$= 1 \cdot x^3 + 1 \cdot x^2 + 1x + \underbrace{1+1}_{0} = x^3 + x^2 + x$

$(x+1) \cdot (x+1) = x^2 + \underbrace{(1+1)}_{0} \cdot x + 1 = x^2 + 1$

$x^2 + x + 1 : x + 1 = x$, Rest 1

$x^2 + x$

1

$x^2 + x + 1 \bmod x + 1 = 1$

CoNe Freiburg

$4\ 3\ 2\ 1\ 0$
$X\ X\ X\ X\ X$

$X^4 + X^2 + 1 \hat{=} |10101|$

$|0101 \cdot |101| = |1101001|$

$|0101$
$|0101000$ ← Carry
$|1101001$

$10 = x$

- Irreduzibles Polynom $\hat{=}$ Primzahl

$x^2 + x + 1$:

$x^2 + 1 = (x+1)^2$

Zeichenketten und Polynomarithmetik

CoNe Freiburg

- Idee:
  - Betrachte Bitstring der Länge n als Variablen eines Polynoms
- Bit string: $b_n b_{n-1} \ldots b_1 b_0$

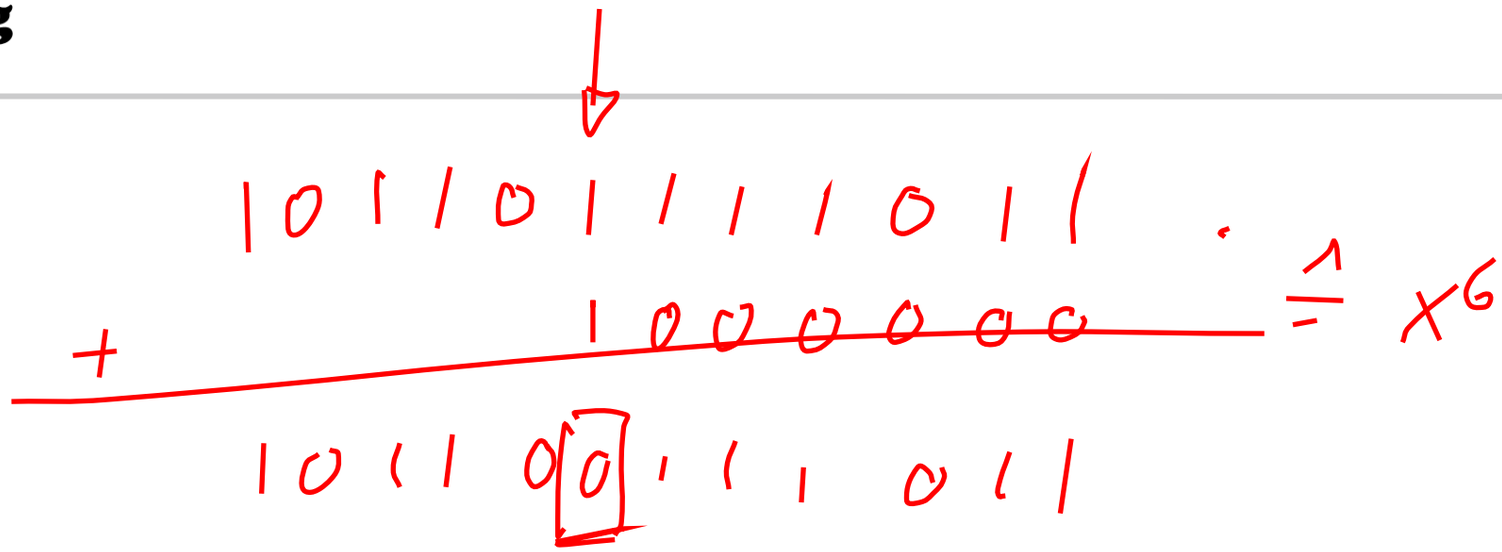
  Polynom: $b_n x^n + \ldots + b_1 x^1 + b_0$
  - Bitstring mit (n+1) Bits entspricht Polynom des Grads n
- Beispiel
  - A xor B = A(x) + B(x)
  - Wenn man A um k Stellen nach links verschiebt, entspricht das
    - $B(x) = A(x)\, x^k$
- Mit diesem Isomorphismus kann man Bitstrings dividieren

$1101\,00000 \triangleq (x^3 + x^2 + 1) \cdot x^5$

$A(x) \bmod x^5$

$1101\,1101 : 100000 = 110 + \frac{11101}{100000}$

CoNe
Freiburg



UNI FREIBURG

Polynome zur Erzeugung von Redundanz: CRC

CoNe Freiburg

1 1 0 1 0 0 1 1 1 Bluetooth

- Definiere ein Generatorpolynom *G(x)* von Grad *g*
  - Dem Empfänger und Sender bekannt
  - Wir erzeugen g redundante Bits
- Gegeben:
  - Frame (Nachricht) M, als Polynom M(x)
- Sender
  - Berechne den Rest der Division $r(x) = x^g M(x) \bmod G(x)$
  - Übertrage $T(x) = x^g M(x) + r(x)$
    - Beachte: $x^g M(x) + r(x)$ ist ein Vielfaches von G(x)
- Empfänger
  - Empfängt m(x)
  - Berechnet den Rest: m(x) mod G(x)

$T(x) \bmod G(x)$

$= M(x) \cdot x^g + r(x) \bmod G(x)$

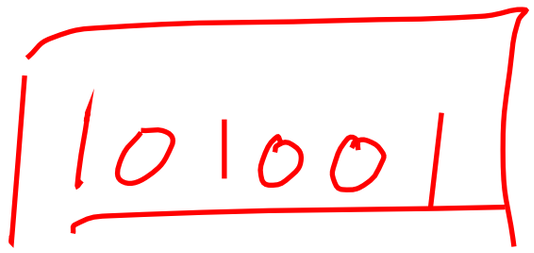
$= r(x) + r(x) \bmod G(x)$

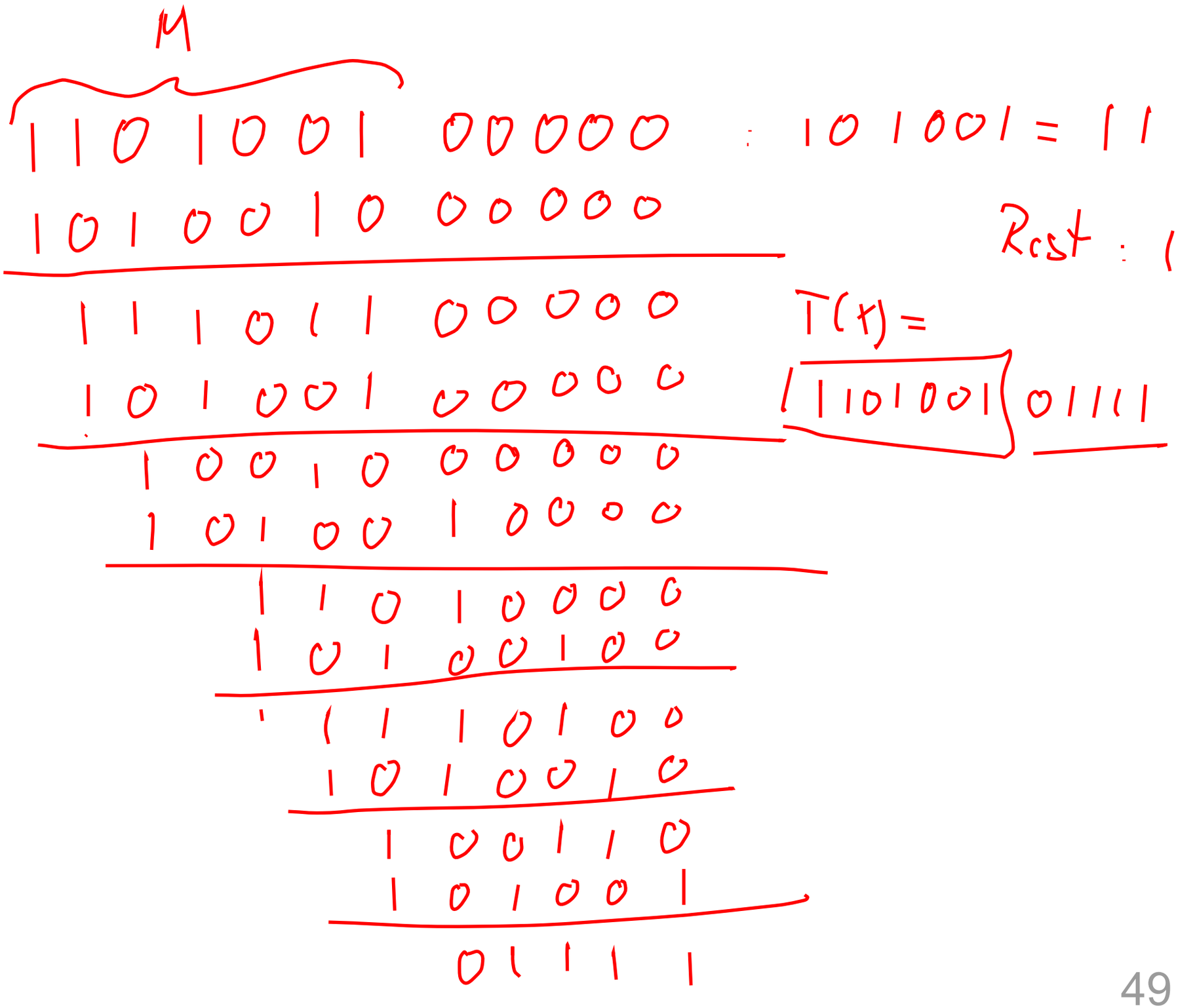
$= 0 \bmod G(x)$

UNI FREIBURG

CoNe Freiburg

RFID





UNI FREIBURG

CRC Übertragung und Empfang

- Keine Fehler:
  - T(x) wird korrekt empfangen
- Bitfehler: T(x) hat veränderte Bits
  - Äquivalent zur Addition eines Fehlerpolynoms E(x)
  - Beim Empfänger kommt T(x) + E(x) an
- Empfänger
  - Empfangen: m(x)
  - Berechnet Rest m(x) mod G(x)
  - Kein Fehler: m(x) = T(x),
    - dann ist der Rest 0
  - Bit errors: m(x) mod G(x) = (T(x)+ E(x)) mod G(x)
    = T(x) mod G(x) + E(x) mod G(x)

0 Fehlerindikator

CoNe Freiburg

$E(x)$

$G(x)$

$1000\,00000 : 10\,1001 = 11010$

2^5

$1\,0\,1\,0\,0\,1\,0\,0$

$1\,0\,0\,1\,0\,0\,0$

$1\,0\,1\,0\,0\,1$

$1\,1\,0\,1\,0$

$E(x) = 10100\,01000 = G(x) \cdot x^3$

$G(x) \cdot (1+x)$

$G(x) \cdot p(x)$

$\frac{1}{2^5}$

UNI FREIBURG

CRC – Überblick

Sender

Original Frame M(x)

Generator Polynom G(X)

$r(x) = x^g M(x) \bmod G(x)$

Sende $T(x) = x^g M(x) + r(x)$

Addiert Fehlerpolynom E(x)

Kein Fehler:
E(x) = 0

Kanal

Empfänger

Empfang: m(x) = T(x) + E(x)

Berechnet Rest (T(x) + E(x)) mod G(x)

möglicherweise

Falls Rest = 0 : Kein Fehler

If Rest ≠ 0: Fehler!

CoNe Freiburg

Der Generator bestimmt die CRC-Eigenschaften

- Bit-Fehler werden nur übersehen, falls E(x) ein Vielfaches von G(x) ist
- Die Wahl von G(x) ist trickreich:
- Einzel-Bit-Fehler: $E(x) = x^i$ für Fehler an Position i
  - G(x) hat mindestens zwei Summenterme, dann ist E(x) kein Vielfaches von G(x) ist
- Zwei-Bit-Fehler: $E(x) = x^i + x^j = x^j (x^{i-j} + 1)$ für i>j
  - G(x) darf nicht $(x^k + 1)$ teilen für alle k bis zur maximalen Frame-Länge
- Ungerade Anzahl von Fehlern:
  - E(x) hat nicht (x+1) als Faktor
  - Gute Idee (?): Wähle (x+1) als Faktor von G(x)
    - Dann ist E(x) kein Vielfaches von G(x)
- Bei guter Wahl von G(x):
  - kann jede Folge von r Fehlern erfolgreich erkannt werden
- Häufig:
  - G(x) wird als irreduzibles Polynom gewählt, dass heißt es ist kein Vielfache eines anderen (kleineren) Polynoms

CoNe Freiburg

CRC in der Praxis

802.3

- Verwendetes irreduzibles Polynom gemäß IEEE 802:
  - $x^{32} + x^{23} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$
- Achtung:
  - Fehler sind immer noch möglich
  - Insbesondere wenn der Bitfehler ein Vielfaches von G(x) ist.
- Implementation:
  - Für jedes Polynom x^i wird $r(x,i)= x^i \bmod G(x)$ berechnet
  - Ergebnis von B(x) mod G(x) ergibt sich aus
  - $b_0\, r(x,0) + b_1\, r(x,1) + b_2\, r(x,2) + \ldots + b_{k-1}\, r(x,k-1)$
  - Einfache Xor-Operation
- Oder rückgekoppelte Schieberegister

Fehlerkontrolle

- Zumeist gefordert von der Vermittlungsschicht
  - Mit Hilfe der Frames
- Fehlererkennung
  - Gibt es fehlerhaft übertragene Bits?
- Fehlerkorrektur
  - Behebung von Bitfehlern
  - Vorwärtsfehlerkorrektur (Forward Error Correction)
    - Verwendung von redundanter Kodierung, die es ermöglicht Fehler ohne zusätzliche Übertragungen zu beheben
  - Rückwärtsfehlerkorretur (Backward Error Correction)
    - Nach Erkennen eines Fehlers, wird durch weitere Kommunikation der Fehler behoben