

Verlässliche Echtzeitsysteme

Übungen zur Vorlesung

Codierung

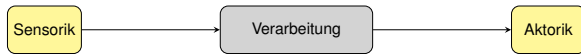
Phillip Raffeck, Florian Schmaus, Simon Schuster

Friedrich-Alexander-Universität Erlangen-Nürnberg
Lehrstuhl Informatik 4 (Verteilte Systeme und Betriebssysteme)
<https://www4.cs.fau.de>

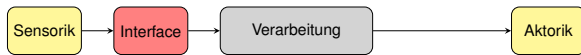
Wintersemester 2020



Klassische "Triple Modular Redundancy" (TMR)



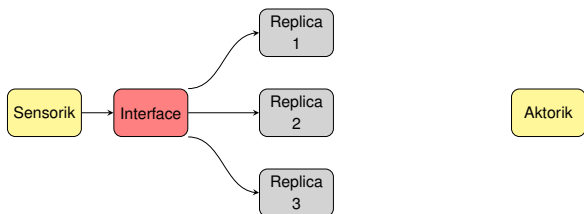
Klassische “Triple Modular Redundancy” (TMR)



- Schnittstelle sammelt Eingangsdaten (Replikdeterminismus)



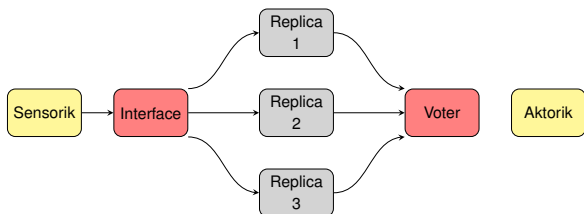
Klassische "Triple Modular Redundancy" (TMR)



- Schnittstelle sammelt Eingangsdaten (Replikdeterminismus)
- Verteilt Daten und aktiviert Replikate



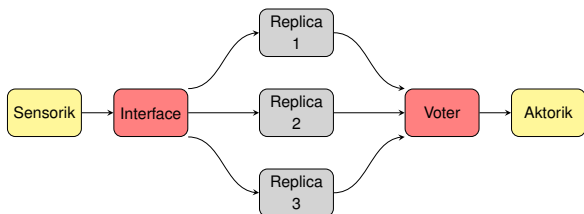
Klassische “Triple Modular Redundancy” (TMR)



- Schnittstelle sammelt Eingangsdaten (Replikdeterminismus)
- Verteilt Daten und aktiviert Replikate
- Mehrheitsentscheider (Voter) wählt Ergebnis



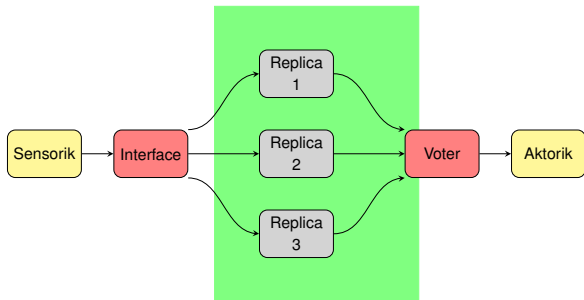
Klassische "Triple Modular Redundancy" (TMR)



- Schnittstelle sammelt Eingangsdaten (Replikdeterminismus)
- Verteilt Daten und aktiviert Replikate
- Mehrheitsentscheider (Voter) wählt Ergebnis
- Ergebnis wird an Aktuator versendet



Klassische "Triple Modular Redundancy" (TMR)

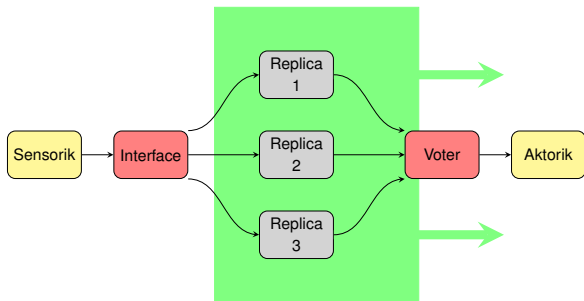


Redundanzbereich

Ausschließlich Replikatausführung



Klassische "Triple Modular Redundancy" (TMR)



Redundanzbereich


Ausschließlich Replikatausführung

- Erweiterung der Ausgangsseite mit Informationsredundanz
- Mehrheitsentscheid über codierte Prüfsumme



nach Forin 1989: "Vital coded microprocessor principles and application for various transit systems" [1]

- Arithmetisch codierter Wert V_C
- Ausgangswert


$$V_C = V$$

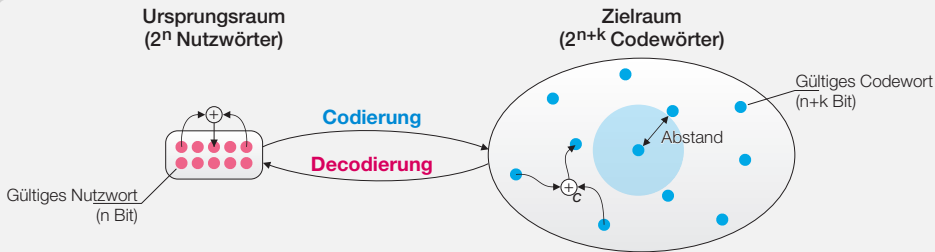


Erweiterte arithmetische Codierung

nach Forin 1989: "Vital coded microprocessor principles and application for various transit systems" [1]

- Arithmetisch codierter Wert V_C
- Ausgangswert

$$V_C = V * A$$



Erweiterte arithmetische Codierung

nach Forin 1989: "Vital coded microprocessor principles and application for various transit systems" [1]

- Arithmetisch codierter Wert V_C
- Ausgangswert

$$V_C = V * A$$

- Schlüssel

Bitfehlererkennung
(Restfehlerwahrscheinlichkeit
 $P = 1/A$)



Erweiterte arithmetische Codierung

nach Forin 1989: "Vital coded microprocessor principles and application for various transit systems" [1]

- Arithmetisch codierter Wert V_C
- Ausgangswert

$$V_C = V * A + B_V$$

- Schlüssel
- Variablenspezifische Signatur

Adressierungsfehlererkennung



Erweiterte arithmetische Codierung

nach Forin 1989: "Vital coded microprocessor principles and application for various transit systems" [1]

- Arithmetisch codierter Wert V_C
- Ausgangswert

$$V_C = V * A + B_V + D$$

- Schlüssel
- Variablenspezifische Signatur
- Zeitstempel

Erkennung
veralteter Daten

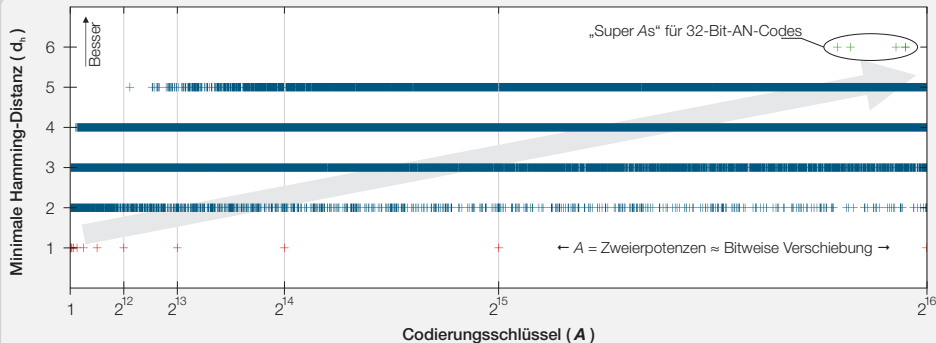


- Schlüssel A sollte so groß wie möglich sein:
 - ↪ Möglichst geringe Restfehlerwahrscheinlichkeit ($P = 1/A$)
- Wertebereich des dynamischen Zeitstempels
 - $D = \{x | x \in \mathbb{N}_0 \wedge x \leq D_{max}\}$
 - Zeitstempel darf überlaufen: $D_{max} + 1 = 0$
- Für jede Signatur $B_* \in \mathbb{N}$ muss dann gelten
 - $B_* + D_{max} < A$
 - Die minimale Distanz zwischen jeweils zwei Signaturen im System muss größer D_{max} sein: $\forall i, j : |B_i - B_j| > D_{max}$
 - Sämtliche Distanzen zwischen jeweils zwei Signaturen müssen unterschiedlich sein: $\forall i : \forall j, k : |B_i - B_j| = |B_k - B_j| \Rightarrow i = k$



Wertebereichseinschränkungen

- Schlüssel A sollte so groß wie möglich sein:
~> Möglichst geringe Restfehlerwahrscheinlichkeit ($P = 1/A$)
- Wertebereich des dynamischen Zeitstempels

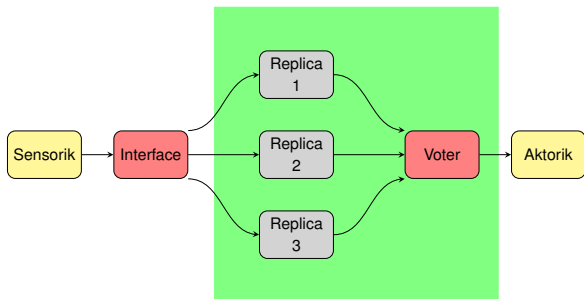


Quelle: Schiffel, U.: Hardware Error Detection Using AN-Codes, Diss., 2011



- Schlüssel A sollte so groß wie möglich sein:
 - ↪ Möglichst geringe Restfehlerwahrscheinlichkeit ($P = 1/A$)
- Wertebereich des dynamischen Zeitstempels
 - $D = \{x | x \in \mathbb{N}_0 \wedge x \leq D_{max}\}$
 - Zeitstempel darf überlaufen: $D_{max} + 1 = 0$
- Für jede Signatur $B_* \in \mathbb{N}$ muss dann gelten
 - $B_* + D_{max} < A$
 - Die minimale Distanz zwischen jeweils zwei Signaturen im System muss größer D_{max} sein: $\forall i, j : |B_i - B_j| > D_{max}$
 - Sämtliche Distanzen zwischen jeweils zwei Signaturen müssen unterschiedlich sein: $\forall i : \forall j, k : |B_i - B_j| = |B_k - B_j| \Rightarrow i = k$





- Replikate liefern arithmetisch codierte Ergebnisse
- Mehrheitsentscheid auf codierten Prüfsummen
- Übertragung codierter Ergebnisse

Vereinfachung für diese Übungsaufgabe

- Kein Zeitstempel

- Voting basiert auf codierter Vergleichsoperation:

$$\rightsquigarrow X_C = Y_C \Rightarrow X * A + B_X = Y * A + B_Y$$

- Im fehlerfreien Fall gilt:

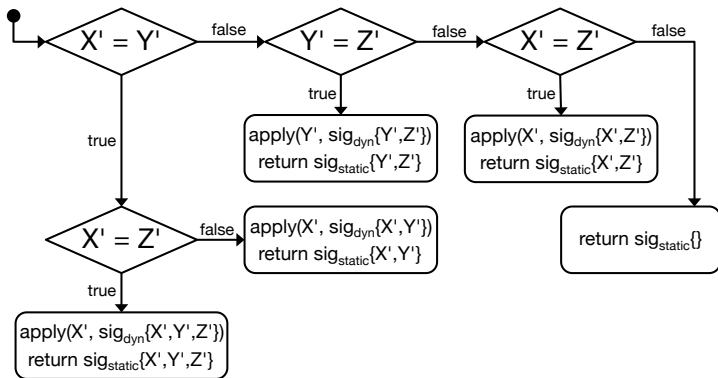
- Rohwerte sind identisch
- Schlüssel ist per Definition identisch
- Signaturen sind unterschiedlich (aber konstant!)

$$X = Y, \quad A = A \quad \text{aber} \quad B_X \neq B_Y !$$

Bestimmung der Gleichheit durch Differenzbildung:

$$\rightsquigarrow X_C - Y_C = B_X - B_Y = \text{const.}$$





■ Bestimmung von dynamischer und statischer Signatur:

→ $sig_{dyn}(X', Y') : X' = Y' \Rightarrow X' - Y'$

→ $sig_{static}(X', Y') : X' = Y' \Rightarrow B_X - B_Y$

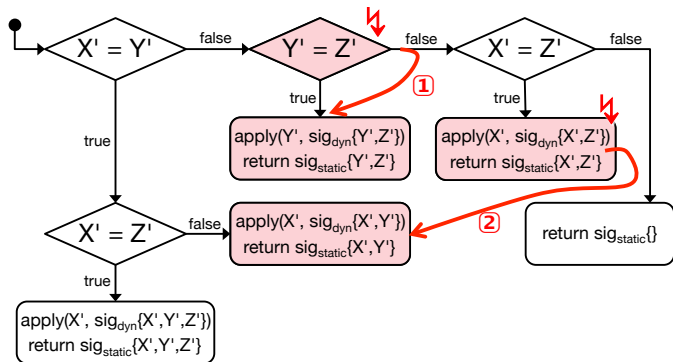
■ Für die Signaturen muss gelten: $B_X > B_Y > B_Z$

1. Vergleichsoperation wird durchgeführt (z. B. $X' = Y' \wedge X' = Z'$)
 - Berechnung von sig_{dyn}
 - Vergleich mit sig_{static}
2. Verzweigungsentscheidung wird nachberechnet:
 - Wiederholte (redundante) Berechnung von sig_{dyn}
 - Erneuter Vergleich: $sig_{dyn} = sig_{static}$
 - Addiere sig_{dyn} (*apply*) zum gewählten Ergebnis
3. Konstante Signatur des durchlaufenen Zweiges identifiziert Gewinner (Rückgabewert: sig_{static})
 - Akteur wählt entsprechendes Replikatergebnis
 - Führt inverse Operation zu *apply* durch

Im Voter wurde die *dynamisch berechnete Signatur der Verzweigungsentscheidung* hinzu addiert. Im Akteur wird mit der entsprechenden *konstanten Signatur zurückgerechnet*.



Codierter Mehrheitsentscheid - Fehlerfall

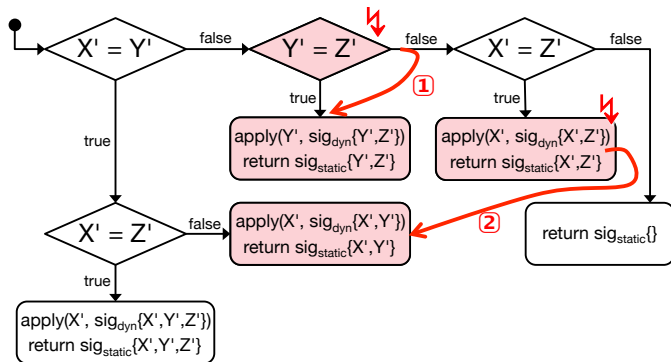


1. Falsche Verzweigungsentscheidung: ($Y' \neq Z'$)

- Y' wird als korrekt angenommen, sig_{dyn} wird berechnet
- allerdings ist sig_{dyn} tatsächlich $\neq sig_{static}$
- Fehler wird vor dem `apply` erkannt



Codierter Mehrheitsentscheid - Fehlerfall



2. Falscher (plötzlicher) Sprung

- X' wird als korrekt erkannt, sig_{dyn} wird erneut berechnet
- Ein fehlerhafter Sprung zu einem anderen Block führt zu einem inkonsistenten Rückgabewert $sig_{static}\{X', Z'\}$
- $sig_{dyn}\{X', Y'\} \neq sig_{static}\{X', Z'\}$ wird beim Dekodieren erkannt

Ausgangslage

- Wähle $A = 601$
- Initiale Ergebniskodierung in den Replikaten:

	X	Y	Z
Wert	7	5	7
B	37	23	5



Ausgangslage

- Wähle $A = 601$
- Initiale Ergebniskodierung in den Replikaten:

	X	Y	Z
Wert	7	5	7
B	37	23	5
Kodiert	$X' = A * 7 + 37$ $= 4244$	$Y' = A * 5 + 23$ $= 3028$	$Z' = A * 7 + 5$ $= 4212$



- Wähle $A = 601$
- Initiale Ergebniskodierung in den Replikaten:

	X	Y	Z
Wert	7	5	7
B	37	23	5
Kodiert	$X' = A * 7 + 37$ $= 4244$	$Y' = A * 5 + 23$ $= 3028$	$Z' = A * 7 + 5$ $= 4212$

- Berechnung der statischen Signaturen **vorab, statisch**:

$$\text{sig}_{\text{static}} \{X', Y', Z'\} = (B_X - B_Y) + (B_X - B_Z) = 46$$

$$\text{sig}_{\text{static}} \{X', Y'\} = (B_X - B_Y) = 14$$

$$\text{sig}_{\text{static}} \{Y', Z'\} = (B_Y - B_Z) = 18$$

$$\text{sig}_{\text{static}} \{X', Z'\} = (B_X - B_Z) = 32$$



- Das eigentliche Voting geschieht dann zur Laufzeit:

1. $X' = Y'?$

$$X' - Y' \stackrel{?}{=} \text{sig}_{\text{static}} \{X', Y'\} \Leftrightarrow 4244 - 3028 = 1216 \stackrel{?}{=} 14 \Leftrightarrow \text{false}$$

2. $Y' = Z'?$

$$Y' - Z' \stackrel{?}{=} \text{sig}_{\text{static}} \{Y', Z'\} \Leftrightarrow 3028 - 4212 = -1184 \stackrel{?}{=} 18 \Leftrightarrow \text{false}$$

3. $X' = Z'?$

$$X' - Z' \stackrel{?}{=} \text{sig}_{\text{static}} \{X', Z'\} \Leftrightarrow 4244 - 4212 = 32 \stackrel{?}{=} 32 \Leftrightarrow \text{true}$$

4. Berechnung der dynamischen Signaturen zur Laufzeit:

$$\text{sig}_{\text{dyn}} \{X', Z'\} = (X' - Z') = (4244 - 4212) = 32$$



- $sig_{dyn} \{X', Z'\} \stackrel{?}{=} sig_{static} \{X, Z\} \Leftrightarrow 32 \stackrel{?}{=} 32 \Leftrightarrow true$
- $X' = apply(X', sig_{dyn} \{X', Z'\}) = 4276$
- $B_E \leftarrow sig_{static} \{X', Z'\} = 32$
- $return(32)$
- Nachschlagen der zugrundeliegenden Variable mittels $B_{dyn} = 32$ (erste Variable der Konsensmenge), basierend auf den vorberechneten statischen Werten:

$$(result_{variable}, result_{extrasignature}) \leftarrow (X', B_X)$$

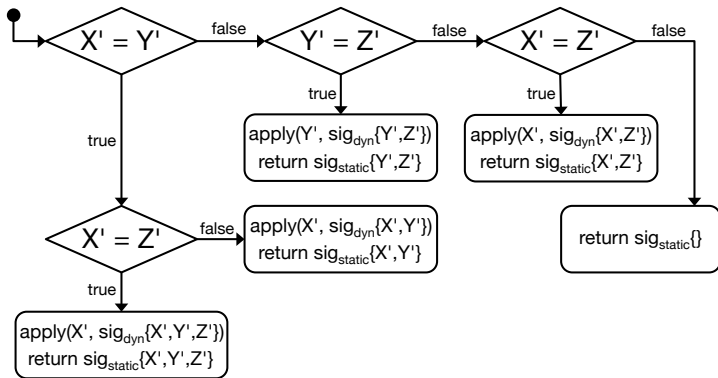
- $inv_apply(result_{variable}, B_{dyn}) = inv_apply(4276, 32) = 4276 - 32 = 4244$



11. Signaturverifikation:

$$\text{check}(4244, A, B_X): \frac{4244 - B_X}{A} = 7 \text{ Rest: } 0$$

12. Ergebnis erfolgreich dekodiert: 7



Es hat somit eine erfolgreiche Einigung auf die Konsensmenge $\{X', Z'\}$ stattgefunden.

Ausgangslage (unverändert)

- Wähle $A = 601$
- Initiale Ergebniskodierung in den Replikaten:

	X	Y	Z
Wert	7	5	7
B	37	23	5
Kodiert	$X' = A * 7 + 37$ $= 4244$	$Y' = A * 5 + 23$ $= 3028$	$Z' = A * 7 + 5$ $= 4212$

- Berechnung der statischen Signaturen **vorab, statisch**:

$$sig_{static} \{X', Y', Z'\} = (B_X - B_Y) + (B_X - B_Z) = 46$$

$$sig_{static} \{X', Y'\} = (B_X - B_Y) = 14$$

$$sig_{static} \{Y', Z'\} = (B_Y - B_Z) = 18$$

$$sig_{static} \{X', Z'\} = (B_X - B_Z) = 32$$

1. $X' = Y'?$

$$X' - Y' \stackrel{?}{=} \text{sig}_{\text{static}} \{X', Y'\} \Leftrightarrow 4244 - 3028 = 1216 \stackrel{?}{=} 14 \Leftrightarrow \text{false}$$

2. $Y' = Z'?$

$$Y' - Z' \stackrel{?}{=} \text{sig}_{\text{static}} \{Y', Z'\} \Leftrightarrow 3028 - 4212 = -1184 \stackrel{?}{=} 18 \Leftrightarrow \text{false}$$

3. Hier tritt nun der Operatorfehler ein, die falsche Verzweigung ① wird ausgewählt

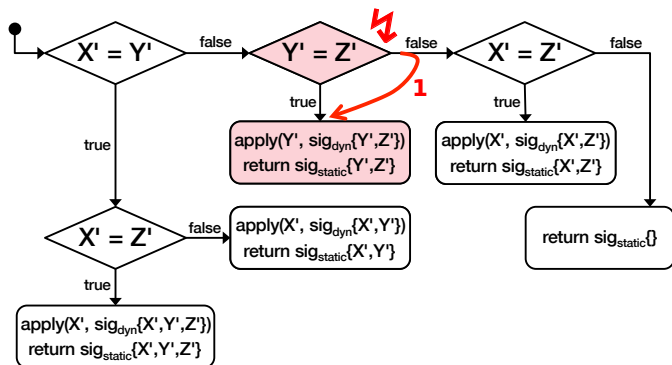
4. Berechnung der dynamischen Signaturen zur Laufzeit:

$$\text{sig}_{\text{dyn}} \{Y', Z'\} = (Y' - Z') = (3028 - 4212) = -1184$$

5. $\text{sig}_{\text{dyn}} \{Y', Z'\} \stackrel{?}{=} \text{sig}_{\text{static}} \{Y, Z\} \Leftrightarrow -1184 \stackrel{?}{=} 18 \Leftrightarrow \text{false}$



6. Fehler erfolgreich detektiert: Vergleich zwischen sig_{dyn} und sig_{static} für Konsensmenge $\{Y', Z'\}$ fehlgeschlagen



Ausgangslage (unverändert)

- Wähle $A = 601$
- Initiale Ergebniskodierung in den Replikaten:

	X	Y	Z
Wert	7	5	7
B	37	23	5
Kodiert	$X' = A * 7 + 37$ $= 4244$	$Y' = A * 5 + 23$ $= 3028$	$Z' = A * 7 + 5$ $= 4212$

- Berechnung der statischen Signaturen **vorab, statisch**:

$$\text{sig}_{static} \{X', Y', Z'\} = (B_X - B_Y) + (B_X - B_Z) = 46$$

$$\text{sig}_{static} \{X', Y'\} = (B_X - B_Y) = 14$$

$$\text{sig}_{static} \{Y', Z'\} = (B_Y - B_Z) = 18$$

$$\text{sig}_{static} \{X', Z'\} = (B_X - B_Z) = 32$$



1. $X' = Y'?$

$$X' - Y' \stackrel{?}{=} \text{sig}_{\text{static}} \{X', Y'\} \Leftrightarrow 4244 - 3028 = 1216 \stackrel{?}{=} 14 \Leftrightarrow \text{false}$$

2. $Y' = Z'?$

$$Y' - Z' \stackrel{?}{=} \text{sig}_{\text{static}} \{Y', Z'\} \Leftrightarrow 3028 - 4212 = -1184 \stackrel{?}{=} 18 \Leftrightarrow \text{false}$$

3. $X' = Z'?$

$$X' - Z' \stackrel{?}{=} \text{sig}_{\text{static}} \{X', Z'\} \Leftrightarrow 4244 - 4212 = 32 \stackrel{?}{=} 32 \Leftrightarrow \text{true}$$

4. Berechnung der dynamischen Signaturen zur Laufzeit:

$$\text{sig}_{\text{dyn}} \{X', Z'\} = (X' - Z') = (4244 - 4212) = 32$$



- $sig_{dyn} \{X', Z'\} \stackrel{?}{=} sig_{static} \{X, Z\} \Leftrightarrow 32 \stackrel{?}{=} 32 \Leftrightarrow true$
- $X' = apply(X', sig_{dyn} \{X', Z'\}) = 4276$
- Hier tritt nun der Kontrollflussfehler ② ein
- $B_E \leftarrow sig_{static} \{X', Y'\} = 14$
- $return(14)$
- Nachschlagen der zugrundeliegenden Variable mittels $B_{dyn} = 14$ (erste Variable der Konsensmenge), basierend auf den vorberechneten statischen Werten:
$$(result_{variable}, result_{extrasignature}) \leftarrow (X', B_X)$$
- $inv_apply(result_{variable}, B_{dyn}) = inv_apply(4276, 14) = 4276 - 14 = 4262$

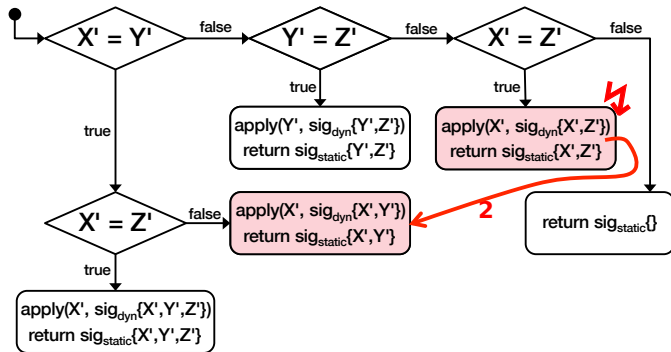


Fehlerszenario ② (II)

12. Signaturverifikation:

$$\text{check}(4262, A, B_X): \frac{4262 - B_X}{A} = 7 \text{ Rest: } 18$$

13. Fehler erfolgreich detektiert: Dekodieren schlägt fehl



Aufgabe

- Absichern des Voters per EAN mittels CoRed-Ansatz
- Berechnungen mit codierten Werten
 - Berücksichtigung der (statischen) Signaturen
 - ↪ Eigene Operationen mit konstanten Signaturwerten notwendig
 - ↪ bspw. eigenes equals anstatt ==
- Jedes Replikat hat genau einen Ausgabewert (integer \mapsto enc_t): eine codierte Prüfsumme des Ergebnisses
 - ↪ Festlegung für jeden der drei Ausgabewerte (X' , Y' , Z') jeweils *unterschiedliche* aber *konstante* Signatur (SIG_X , SIG_Y , SIG_Z)
- Nutzung des nächstgrößeren Datentyps X für den ursprünglichen Wert X'
 - ↪ Wahl einer Zahl A mit möglichst großem Hamming-Abstand unter *Vermeidung* möglicher *Überläufe bei der Codierung*
 - ↪ Handhabbarkeit: **nicht** Super-As





Forin.

Vital coded microprocessor principles and application for various transit systems.

IFA-GCCT, pages 79–84, 1989.

