

Verlässliche Echtzeitsysteme - Übungen

Arbeiten mit Binärdateien und Assembler, Codierung

Wintersemester 2024

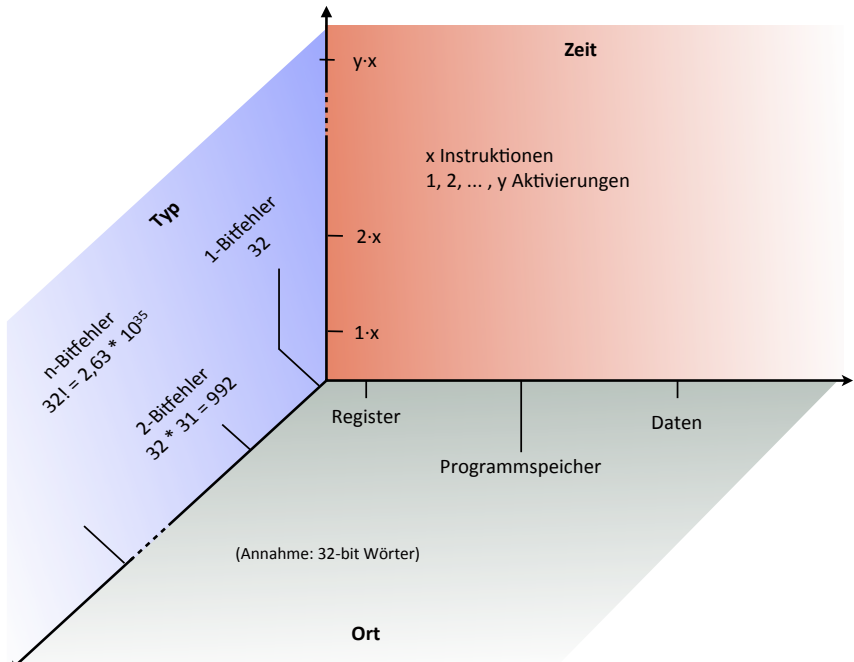
Eva Dengler, Peter Wägemann

Friedrich-Alexander-Universität Erlangen-Nürnberg

Lehrstuhl Informatik 4 (Systemsoftware)

<https://sys.cs.fau.de>

Fehlerraum



C-Code vs. Assembler-Code

C-Code

```
int a;  
int b = 1;  
const int c = 2;  
void main() {  
    static int s = 3;  
    int x, y;  
    char* p =  
        malloc(100);  
}
```

Assembler-Code

```
4004f0 <main>:  
4004f0: push    %rbp  
4004f1: mov     %rsp,%rbp  
4004f4: sub     $0x10,%rsp  
4004f8: movabs $0x64,%rdi  
400502: callq  4003e0 <malloc@plt>  
400507: mov     %rax,-0x10(%rbp)  
40050b: add     $0x10,%rsp  
40050f: pop     %rbp  
400510: retq
```

Wo können Datenfehler auftreten?

C-Code vs. Assembler-Code

C-Code

```
int a;  
int b = 1;  
const int c = 2;  
void main() {  
    static int s = 3;  
    int x, y;  
    char* p =  
        malloc(100);  
}
```

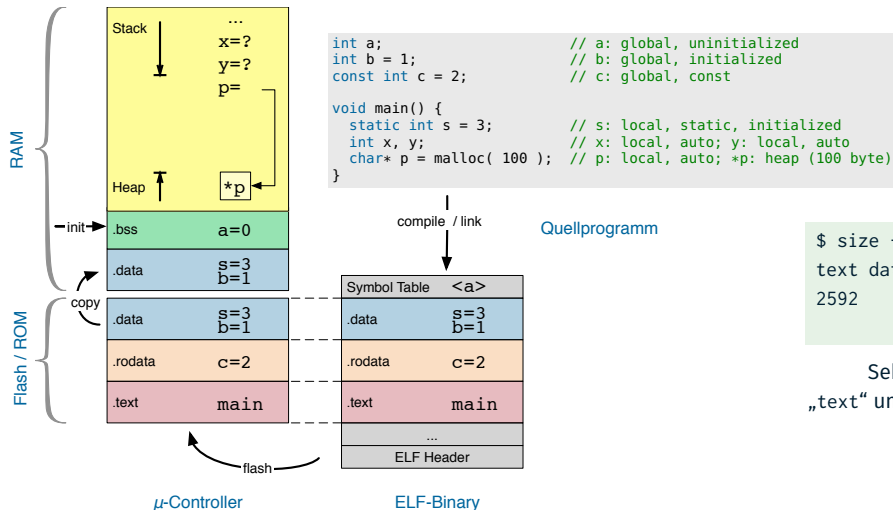
Assembler-Code

```
4004f0 <main>:  
4004f0: push    %rbp  
4004f1: mov     %rsp,%rbp  
4004f4: sub    $0x10,%rsp  
4004f8: movabs $0x64,%rdi  
400502: callq  4003e0 <malloc@plt>  
400507: mov    %rax,-0x10(%rbp)  
40050b: add    $0x10,%rsp  
40050f: pop    %rbp  
400510: retq
```

Wo können Datenfehler auftreten?

1. RAM: `-0 x10 (% rbp)`
 ↪ Stack, globale Daten, Heap, (Programmcode)
2. Allgemeine CPU-Register: `% rsp`
3. Sonstige CPU-Register: `% rip , % rflags`

Speicherorganisation und Sektionsgrößen



```

$ size --format=berkeley a.out
text data bss dec hex filename
2592 8 4 2604 a2c a.out
    
```

Sektionsgrößen in Byte,
„text“ umfasst hier: .text + .rodata

Umgang mit Assembly-Code I

nm: Ausgabe der Symboltabelle

```
00000000000004028 D var_initialized
...
00000000000201028 B __bss_start
00000000000201028 b var_uninitialized
...
0000000000000061a T main
00000000000000580 t register_tm_clones
00000000000000510 T _start
...
0000000000000066d t int max<int,0u>(int,int)
0000000000000067c t int max<int,1u>(int,int)
```

■ Nützliche Optionen

- -C, --demangle: Dekodieren der C++-Namensmangelung:
`_Z3maxIiLj0EET_S0_S0_` \Rightarrow `int max<int, 0u>(int, int)`
- -S, --print-size: Ausgabe der Symbolgrößen
`0000000000000061 a 00000028 T main`

Umgang mit Assembly-Code II

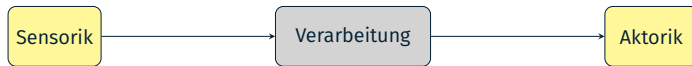
objdump: Ausgabe von Informationen über Objektdateien

```
int main(){
4007cd: 55                push   %rbp
4007ce: 48 89 e5         mov    %rsp,%rbp
4007d1: 48 83 ec 10      sub    $0x10,%rsp
int a = max<int>(23U, 42);
4007d5: be 2a 00 00 00   mov    $0x2a,%esi
4007da: bf 17 00 00 00   mov    $0x17,%edi
4007df: e8 04 01 00 00   callq 4008e8 <_Z3maxIiET_S0_S0_>
4007e4: 89 45 fc         mov    %eax,-0x4(%rbp)
std::cout << a << "\n";
4007e7: 8b 45 fc         mov    -0x4(%rbp),%eax
...
```

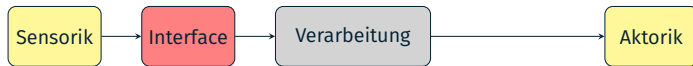
■ Nützliche Optionen

- -S: Ausgabe von Quell-Code im Assembly-Code (Debug-Symbole notwendig)
- -D: alle Sektionen disassemblieren

Klassische “Triple Modular Redundancy” (TMR)

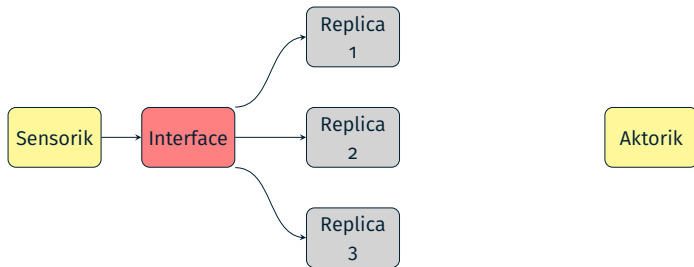


Klassische “Triple Modular Redundancy” (TMR)



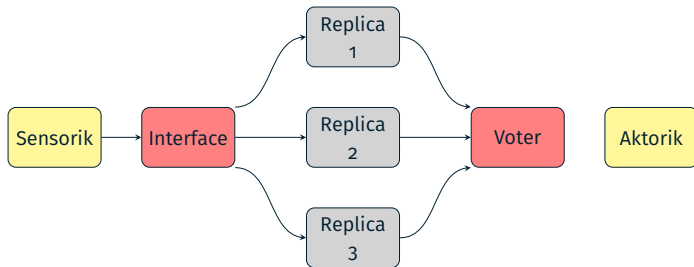
- Schnittstelle sammelt Eingangsdaten (Replikdeterminismus)

Klassische “Triple Modular Redundancy” (TMR)



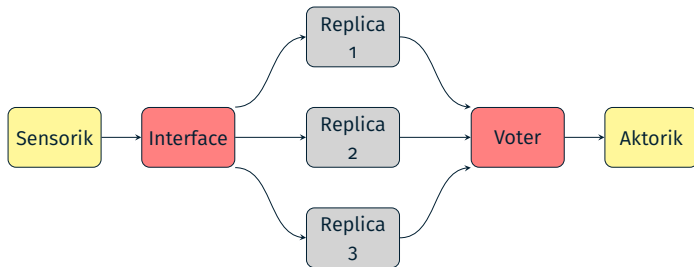
- Schnittstelle sammelt Eingangsdaten (Replikdeterminismus)
- Verteilt Daten und aktiviert Replikate

Klassische “Triple Modular Redundancy” (TMR)



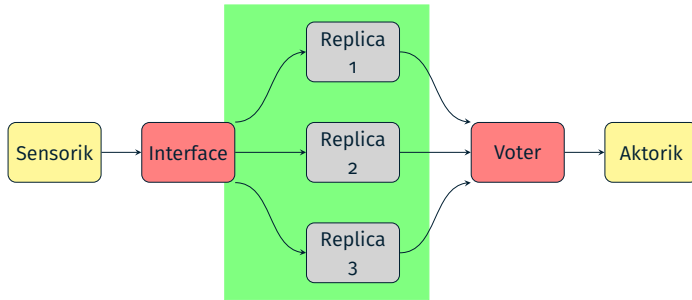
- Schnittstelle sammelt Eingangsdaten (Replikdeterminismus)
- Verteilt Daten und aktiviert Replikate
- Mehrheitsentscheider (Voter) wählt Ergebnis

Klassische “Triple Modular Redundancy” (TMR)



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

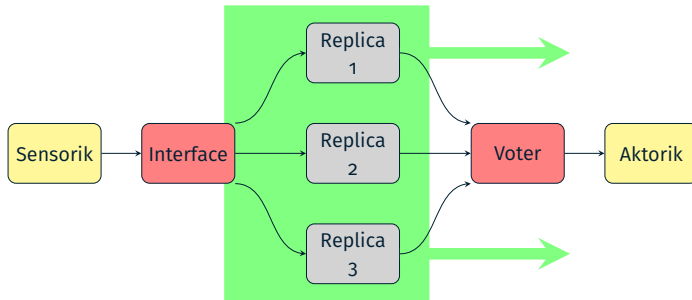
Klassische “Triple Modular Redundancy” (TMR)



Redundanzbereich

Ausschließlich Replikatausführung

Klassische “Triple Modular Redundancy” (TMR)



Redundanzbereich

Ausschließlich Replikatausführung

- ~ Erweiterung der Ausgangsseite mit Informationsredundanz
- ~ Mehrheitsentscheid über codierte Prüfsumme

Erweiterte arithmetische Codierung

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

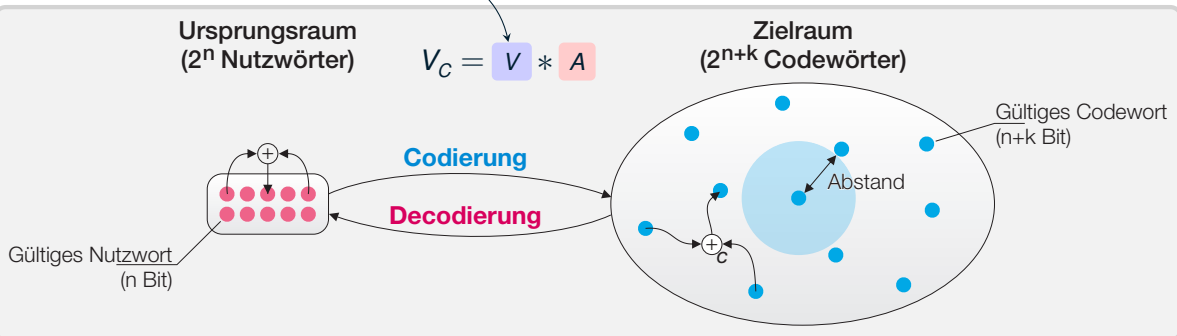
- Arithmetisch codierter Wert V_C
- Ausgangswert


$$V_C = V$$

Erweiterte arithmetische Codierung

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

- Arithmetisch codierter Wert V_C
- Ausgangswert



Erweiterte arithmetische Codierung

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

- 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"

- 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"

- Arithmetisch codierter Wert V_C
- Ausgangswert

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

- Schlüssel
- Variablenspezifische Signatur
- Zeitstempel

Erkennung
veralteter Daten

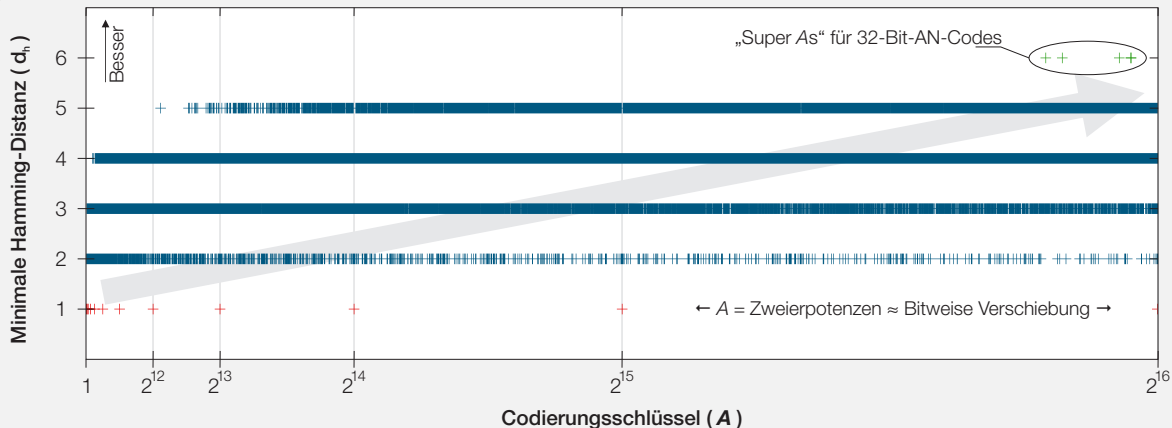
Wertebereichseinschränkungen

- Schlüssel A sollte so groß wie möglich sein:
 - ↷ Möglichst geringe Restfehlerwahrscheinlichkeit ($P = 1/A$)
- Wertebereich des dynamischen Zeitstempels
 - $D = \{x \mid x \in \mathbb{N}_0 \wedge x \leq D_{max}\}$
 - 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:

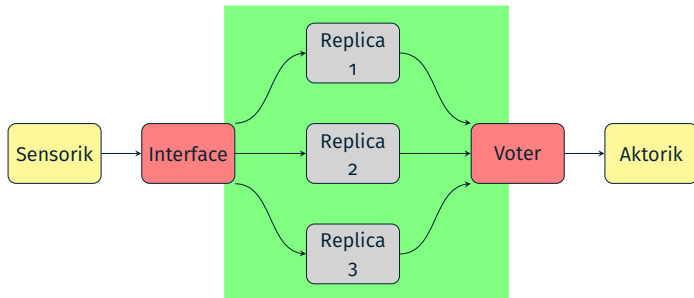
↪ Möglichst geringe Restfehlerwahrscheinlichkeit ($P = 1/A$)



Wertebereichseinschränkungen

- 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: $\forall i : \forall j, k : |B_i - B_j| = |B_k - B_j| \Rightarrow i = k$

Erweiterung I – codierte Ausgangswerte



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

EAN Vergleichsoperator

Vereinfachung für diese Übungsaufgabe

- Kein Zeitstempel

- Voting basiert auf codierter Vergleichsoperation:

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

- Im fehlerfreien Fall gilt:

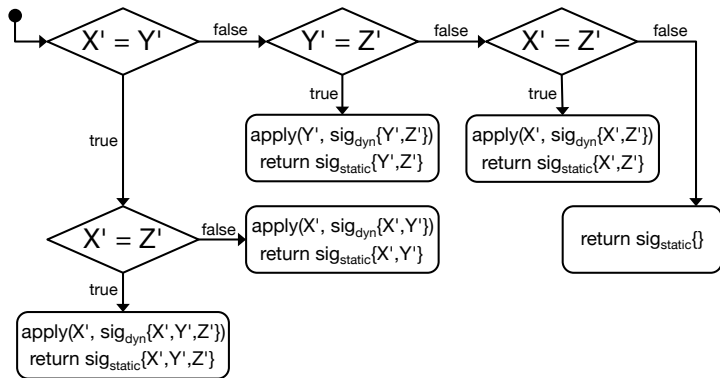
- Rohwerte sind identisch
- Schlüssel ist per Definition identisch
- Signaturen sind unterschiedlich (aber konstant!)

$X = Y$, $A = A$ aber $B_X \neq B_Y$!

Bestimmung der Gleichheit durch Differenzbildung:

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

Codierter Mehrheitsentscheid



- Bestimmung von dynamischer und statischer Signatur:

→ $\text{sig_dyn}(X', Y') : X' = Y' \Rightarrow X' - Y'$

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

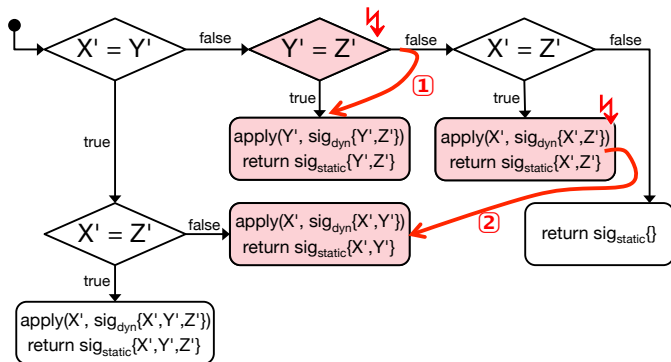
- Für die Signaturen muss gelten: $B_X > B_Y > B_Z$

Codierter Mehrheitsentscheid (Forts.)

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*.

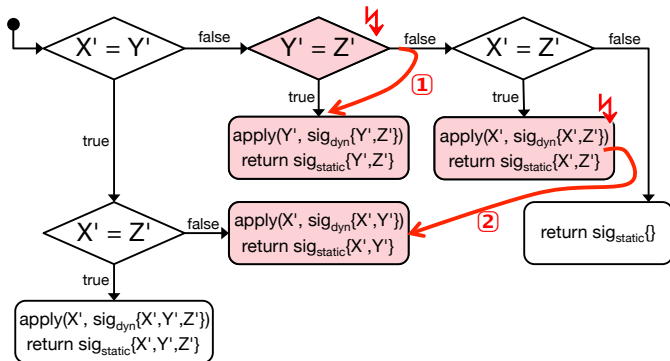
Codierter Mehrheitsentscheid - Fehlerfall



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

Ausgangslage

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

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

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

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

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

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

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

Regulärer Durchlauf

- 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:

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

Regulärer Durchlauf (II)

5. $sig_{dyn} \{X', Z'\} \stackrel{?}{=} sig_{static} \{X, Z\} \Leftrightarrow 32 \stackrel{?}{=} 32 \Leftrightarrow true$

6. $X' = apply(X', sig_{dyn} \{X', Z'\}) = 4276$

7. $B_E \leftarrow sig_{static} \{X', Z'\} = 32$

8. $return(32)$

9. Nachschlagen der zugrundeliegenden Variable mittels $B_{dyn} = 32$ (erste Variable der Konsensmenge), basierend auf den vorberechneten statischen Werten:

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

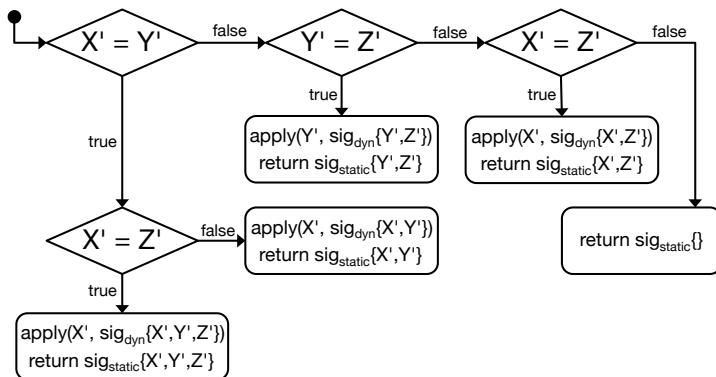
10. $inv_apply(result_{variable}, B_{dyn}) = inv_apply(4276, 32) = 4276 - 32 = 4244$

Regulärer Durchlauf (III)

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**:

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

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

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

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

Fehlerszenario ①

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

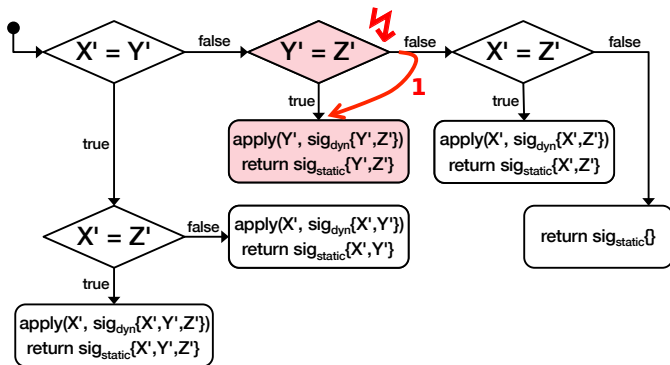
4. Berechnung der dynamischen Signaturen zur Laufzeit:

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

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

Fehlerszenario ① (II)

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:**

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

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

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

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

Fehlerszenario ②

1. $X' = Y'?$

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

2. $Y' = Z'?$

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

3. $X' = Z'?$

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

4. Berechnung der dynamischen Signaturen zur Laufzeit:

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

Fehlerszenario ②

- $sig_{dyn} \{X', Z'\} \stackrel{?}{=} sig_{static} \{X, Z\} \Leftrightarrow 32 \stackrel{?}{=} 32 \Leftrightarrow true$
- $X' = apply(X', sig_{dyn} \{X', Z'\}) = 4276$
- 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:

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

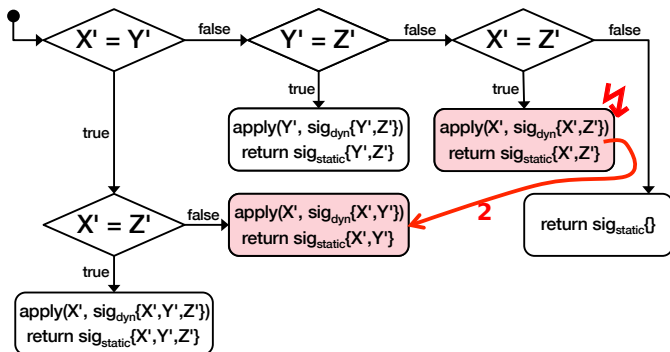
- $inv_apply(result_{variable}, B_{dyn}) = inv_apply(4276, 14) = 4276 - 14 = 4262$

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



Analyse der Toleranzmaßnahmen

- Fehlerräumenanalyse

- Einfluss des Speichers
- Einfluss der Laufzeit

- Aufrufgraph

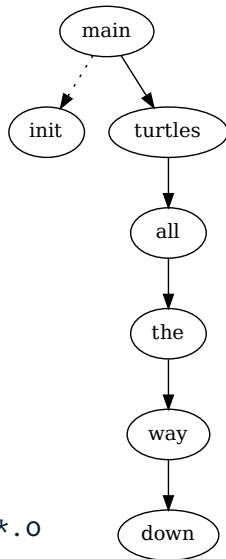
`make callgraph_{baseline,tmr,ean}`

- ELF

`build/{baseline,tmr,ean}.elf`

- Objektdateien

`build/CMakeFiles/{baseline,tmr,ean}.dir/src/*.o`



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 ==