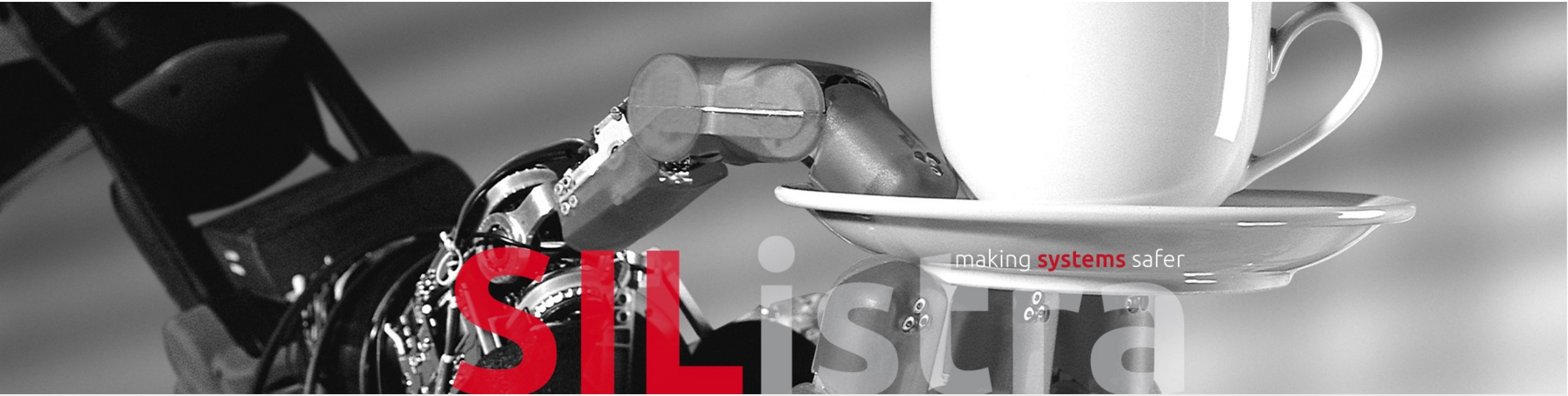

# INTEROPERABILITY OF DIVERSIFIED ENCODING

Dr. Martin Süßkraut – Dr. Andreas Ecke – Mario Schwalbe

# Agenda

- software-based diagnosis of random hardware errors (diversified encoding)
- architecture of a generic safety application with software-based diagnosis
- generic component with software-based diagnosis
- summary

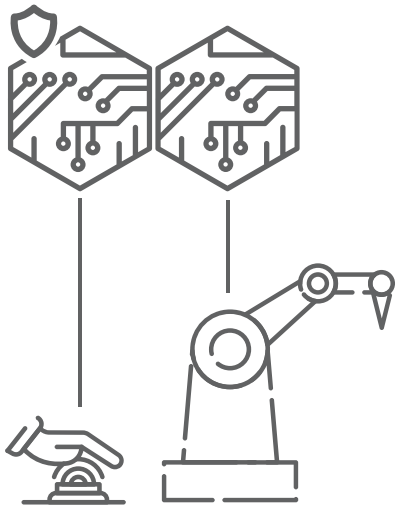
# Software-based diagnosis of random hardware errors



# Motivation

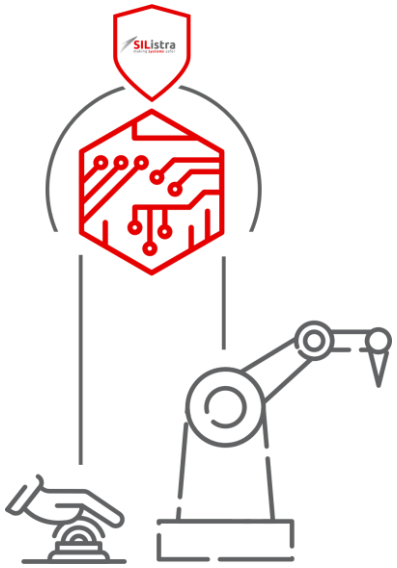
current state of safety

- safety-critical applications **run only on** safety-critical h**ardware**

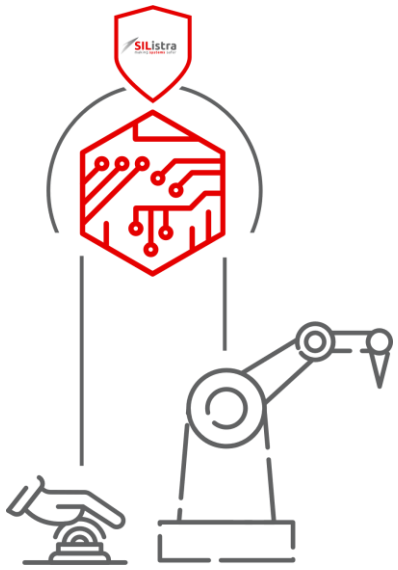


SIListrified safety-critical application

- safety in software with safety-critical hardware

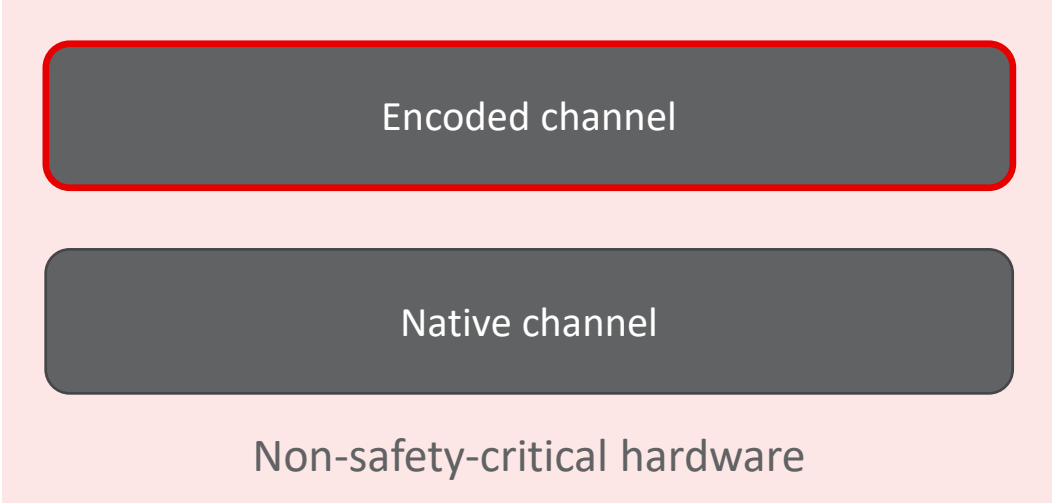


# Advantages and challenges of 1-channel hardware



- **no separate hardware for safety required**
  - no complex development of 2-channel hardware
  - safety applications can run on existing 1-channel hardware together with non-safety applications
- **safety mostly hardware independent**
  - better CPU performance
  - simple change to new hardware generation at end-of-life-notice
- **challenges**
  - diagnosis and handling of random hardware errors
  - architecture

# Software-based diagnosis with diversified encoding




- 2 **diverse** software channels
  - run on **same** hardware
  - channels functional equivalent
  - encoded channel:
    - uses software coded processing
    - can be generated by a tool

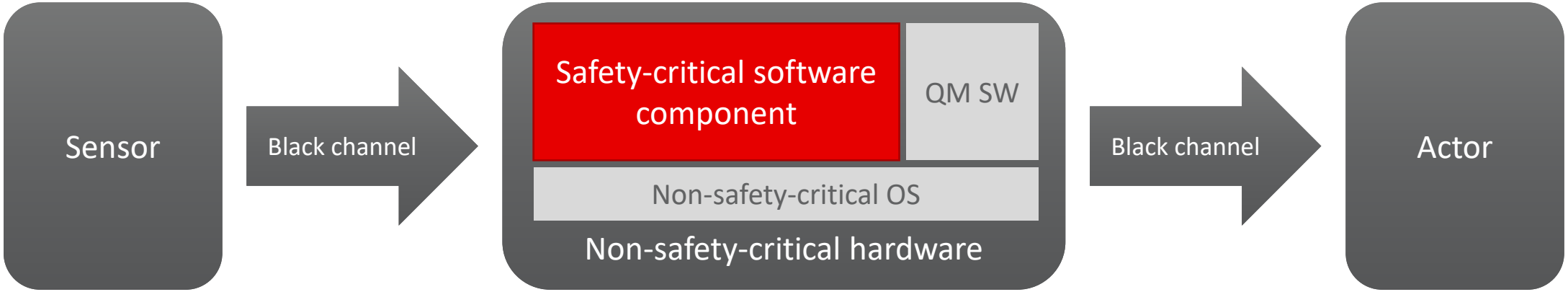
# Software coded processing

Encoded channel works with encoded code on encoded data.

- example: AN encoding
  - variables, constants and operators are arithmetically encoded
  - every variable and constant is a multiple of A: $x \rightarrow x_{enc} = A \cdot x$
  - correctness check: $x_{enc} \bmod A \equiv 0$
  - error: result is not a multiple of A
- encoded C-code can be automatically generated from the C-code of the native channel

# Architecture of a generic safety application with software-based diagnosis

# Generic safety-critical application with software-based diagnosis



# Requirements on components

Sensor

- should be intrinsically safe
- other concepts possible

Actor

- **must** be intrinsically safe
- must safely detect and handle all failures of the black channel

Black channel

- **safety-critical communication protocol**

Non-safety-critical hardware

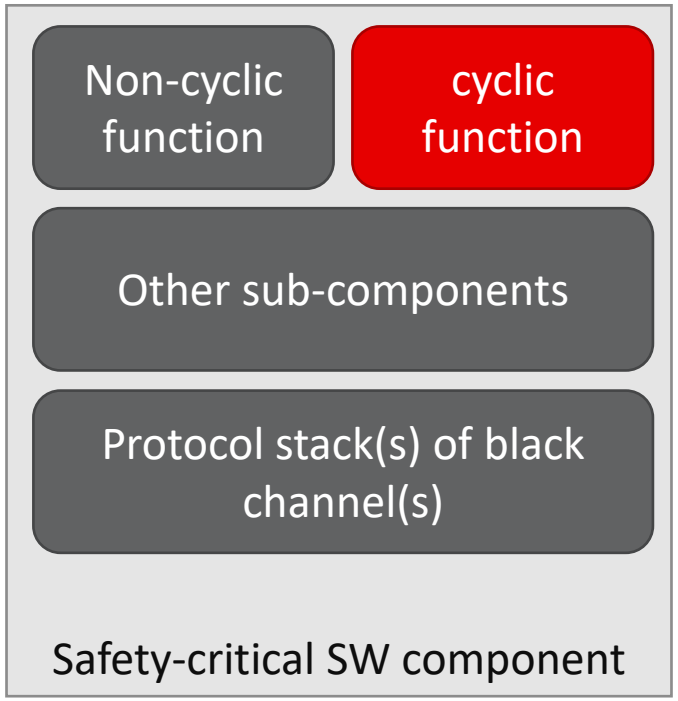
- 1-channel hardware that runs the safety functions

Safety-critical software component

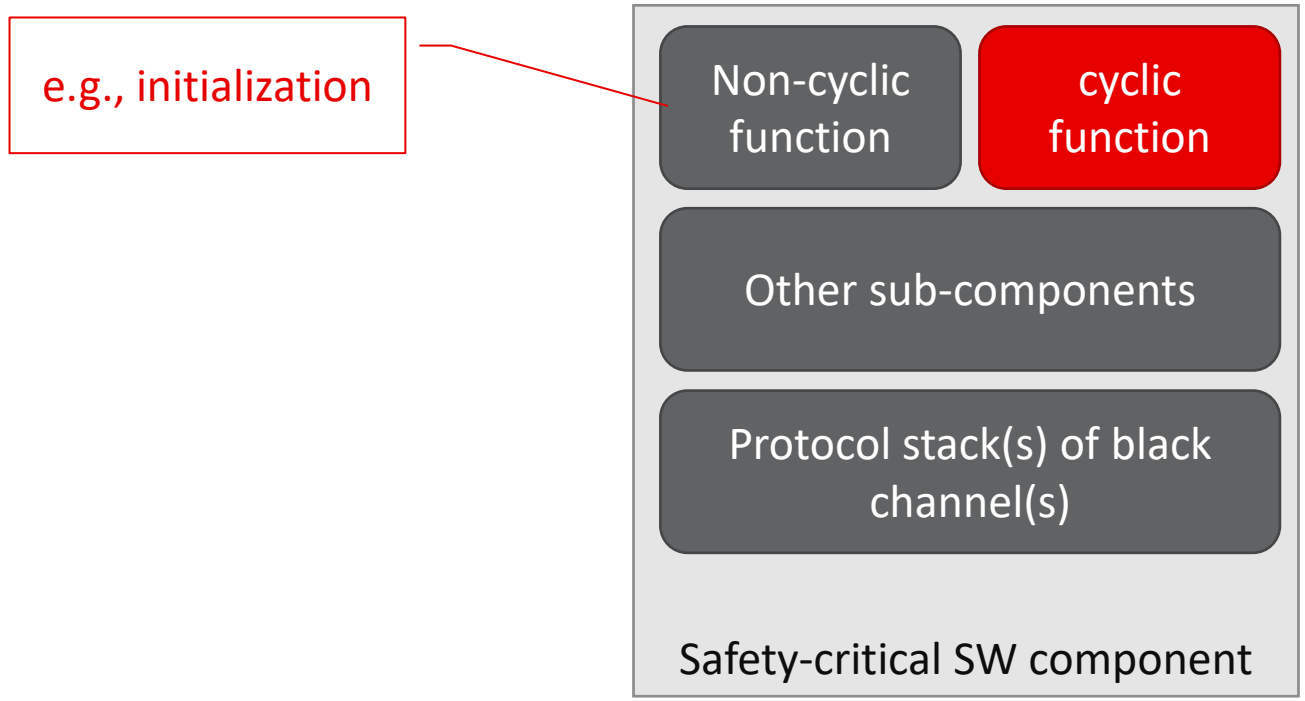
- diagnoses random errors with diversified encoding
- native channel must be developed following an appropriate safety process

# Generic component with software-based diagnosis

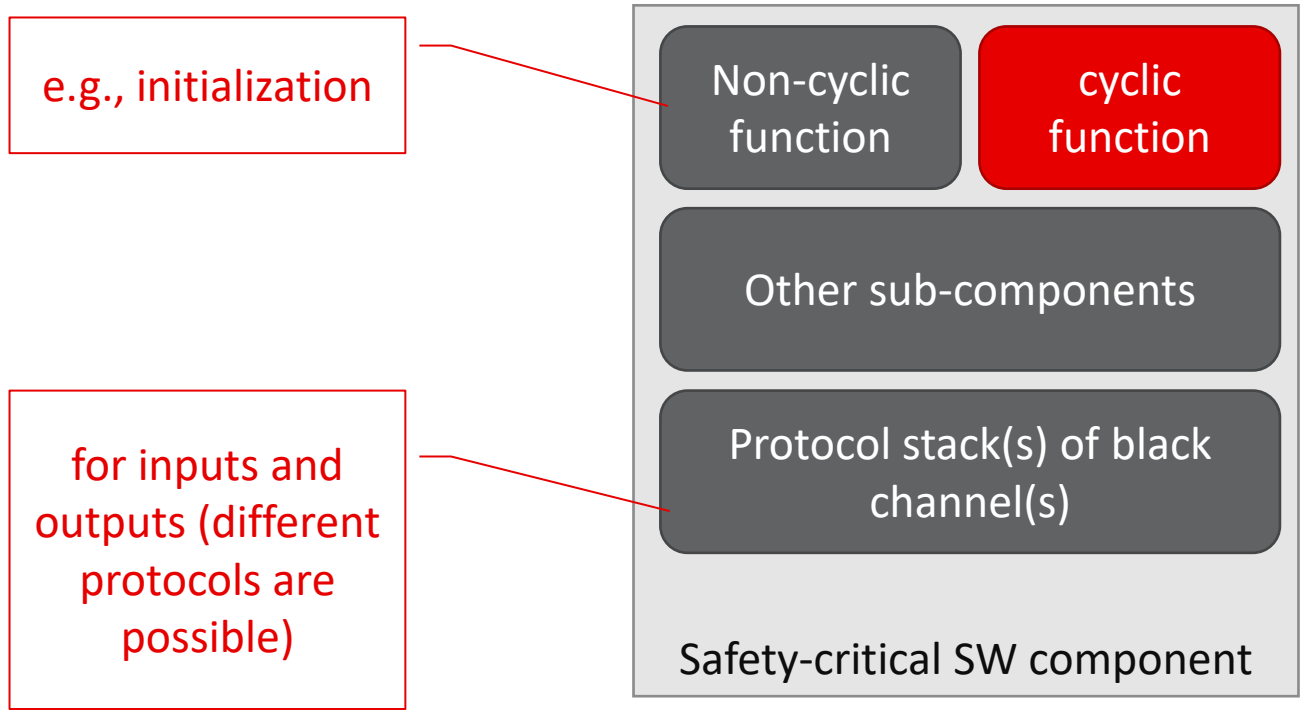
# Architecture of the safety-critical SW component



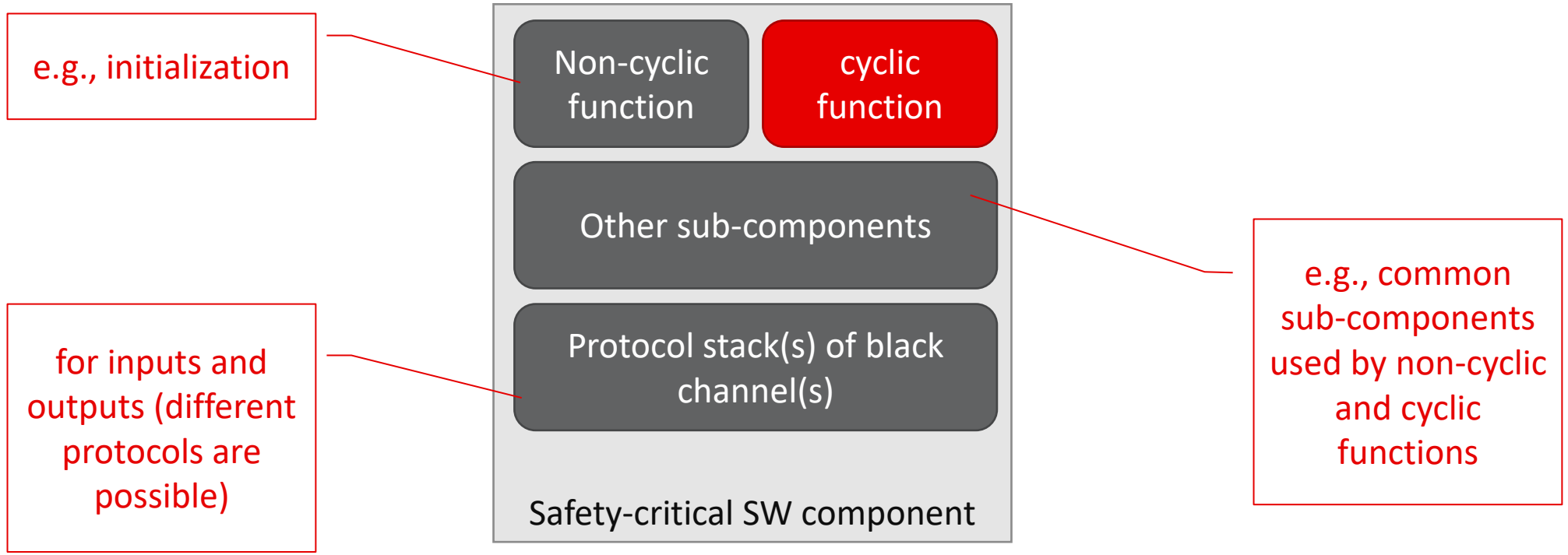
# Architecture of the safety-critical SW component



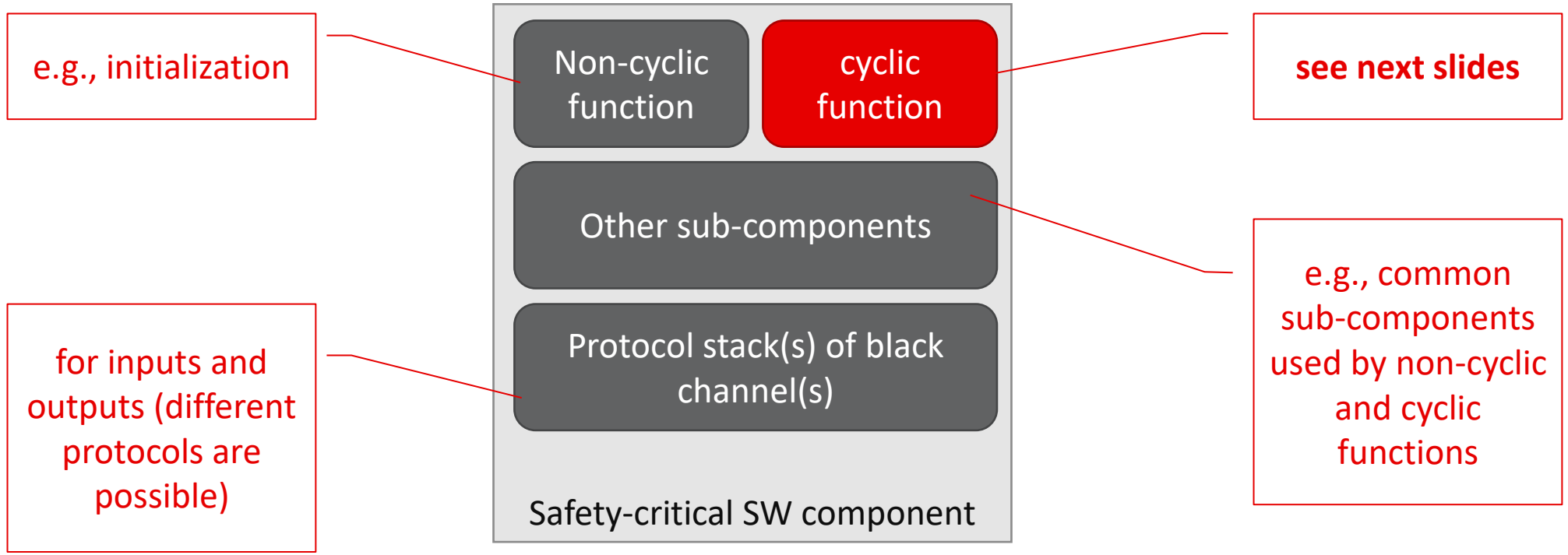
# Architecture of the safety-critical SW component



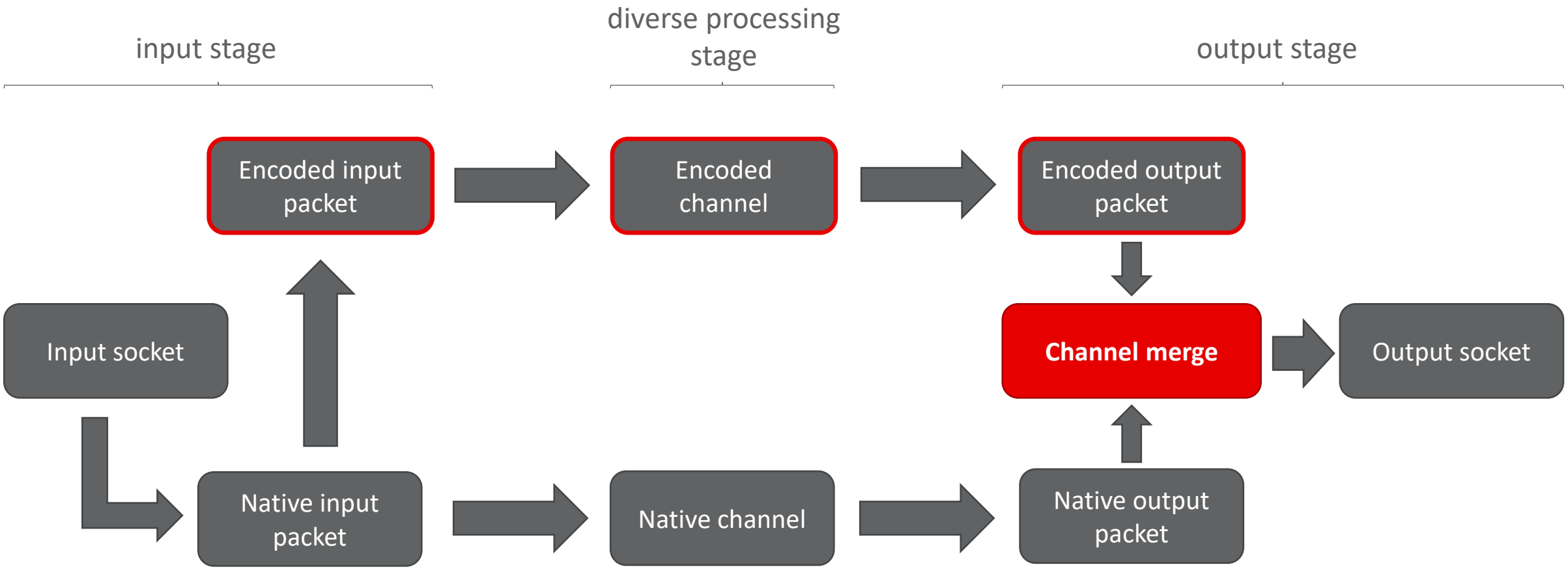
# Architecture of the safety-critical SW component



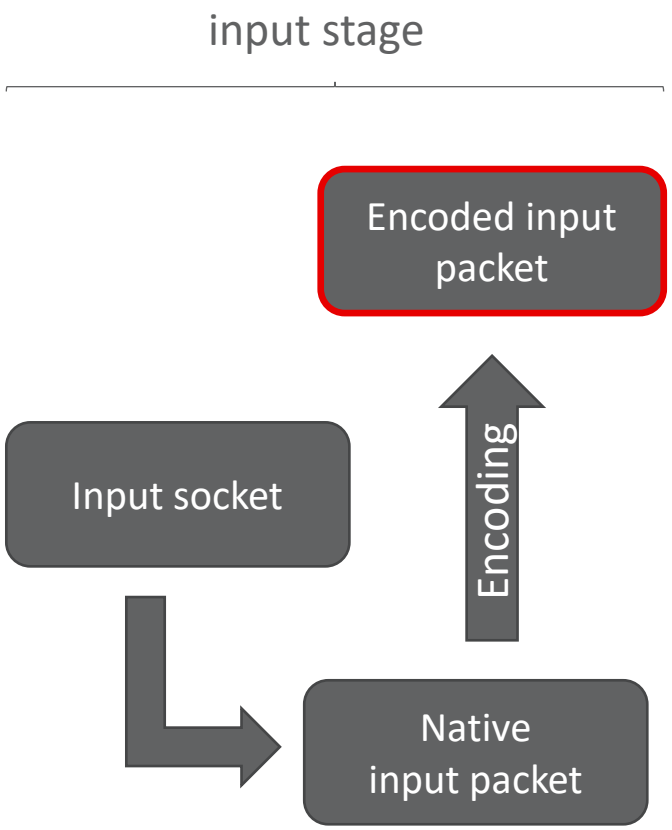
# Architecture of the safety-critical SW component



# Runtime view of the cyclic function for one cycle

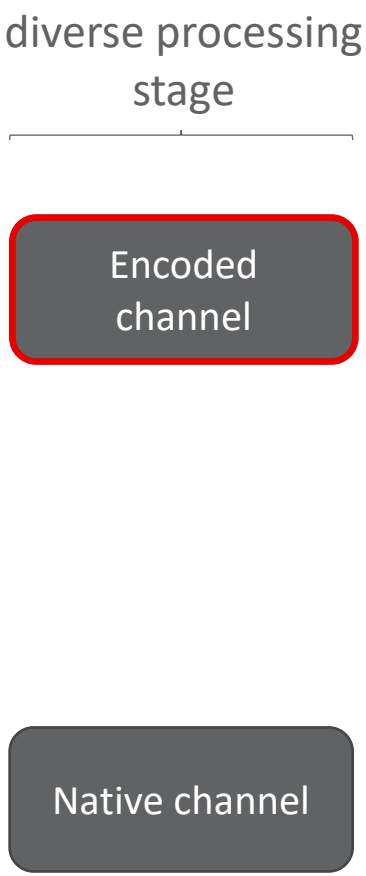


# Input stage: transition from 1 channel to 2 channels



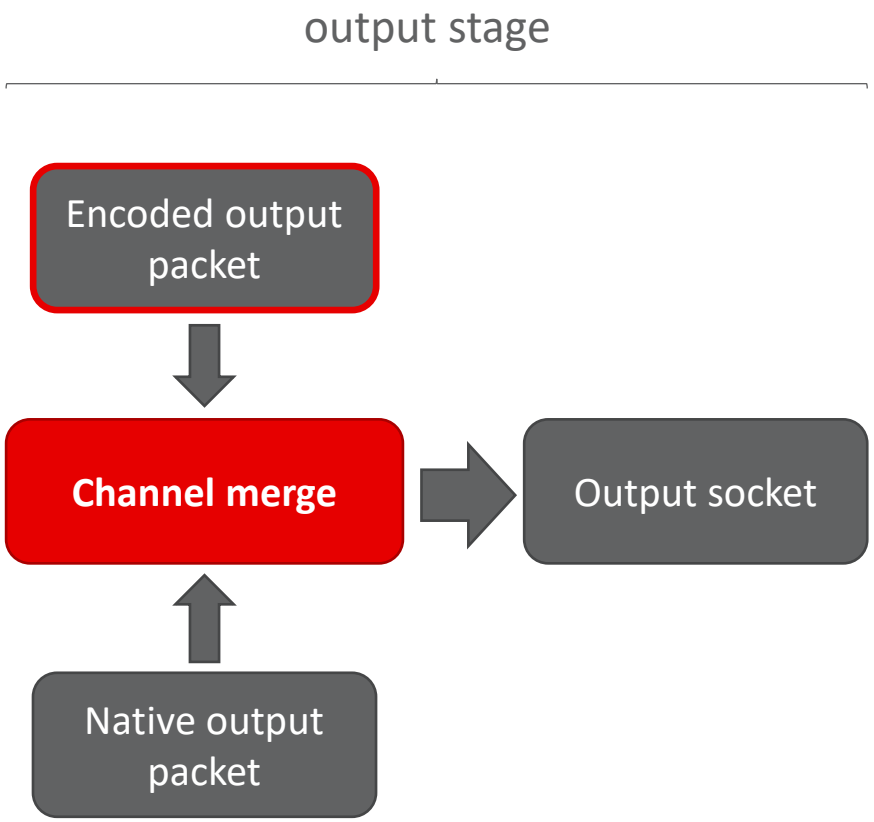

- input stage itself does not check the input packet
  - check must be done in 2-channel code in the diverse processing stage
- encoding of the input packet does not need to know the packet structure or the protocol
  - encoding can be done byte-by-byte

# Diverse processing stage



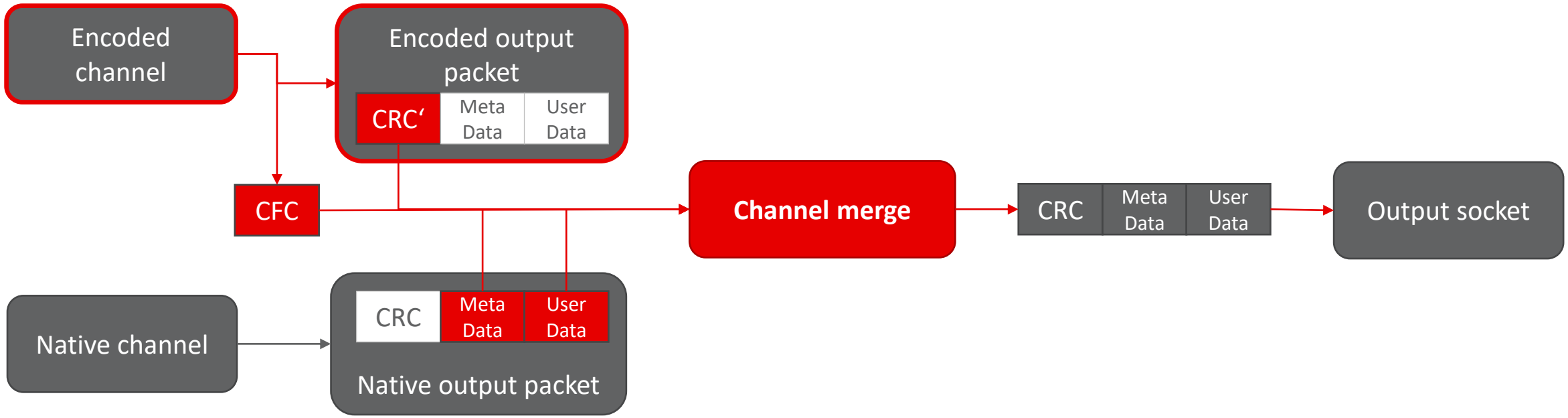

- channels are independent of each other and can run in parallel or sequentially
- the diverse processing stage **must** check each input packet in both channels
  - diagnosed errors must be propagated to the output packet on application level
- encoded channel computes:
  - encoded complete output packet
  - uses a dynamically calculated control-flow signature to check for control-flow errors
- native channel computes incomplete output packet (e.g. without checksum)
- only the channel merge will compute the final complete output packet

# Output stage: transition from 2 channels to 1 channel



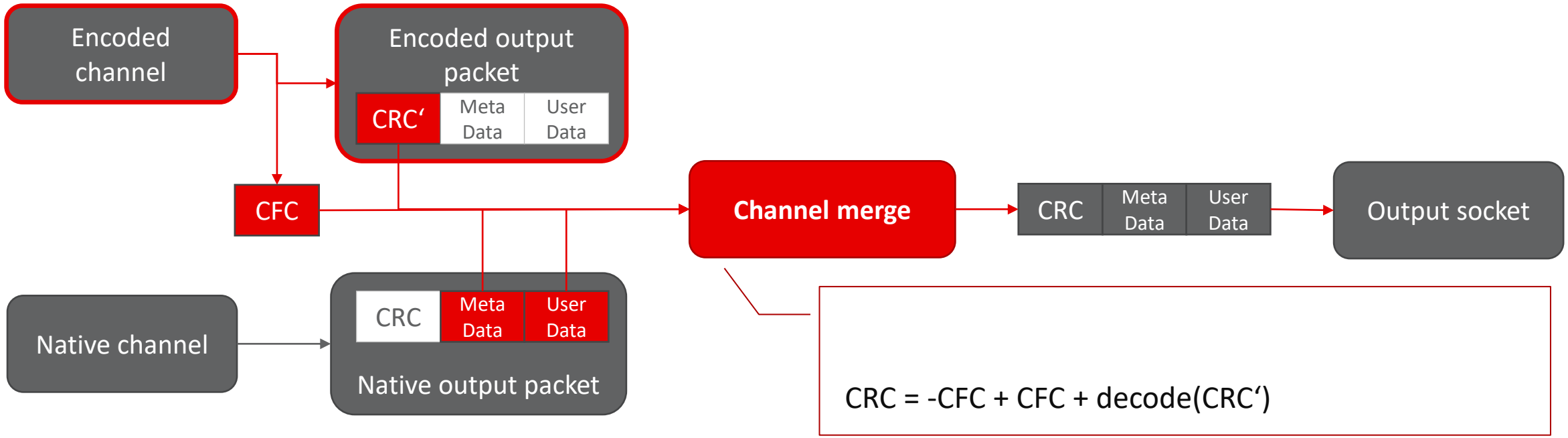

- only the merge computes the final output packet
- every diagnosed error must be propagated as channel error:
  - if diverse processing stage computes invalid output
    - e.g., because of invalid input packet
  - if random error was diagnosed
    - encoded output packet and native output packet to not match
    - encoded output packet not validly encoded
    - control-flow error diagnosed in encoded channel

# Channel merge using a checksum



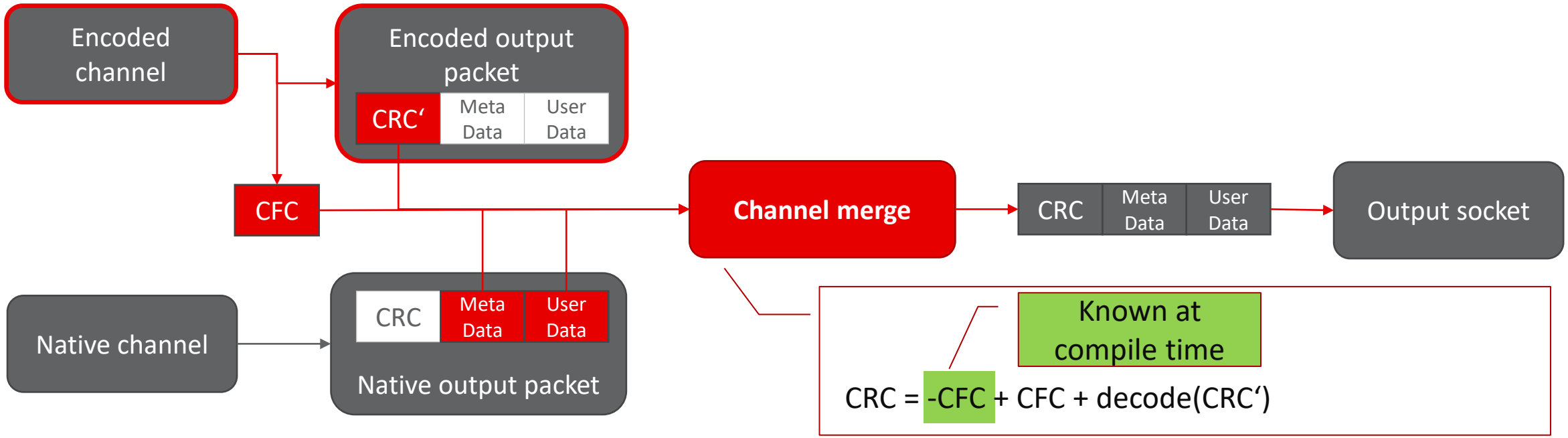

CFC = control-flow check signature
CRC = checksum

# Channel merge using a checksum



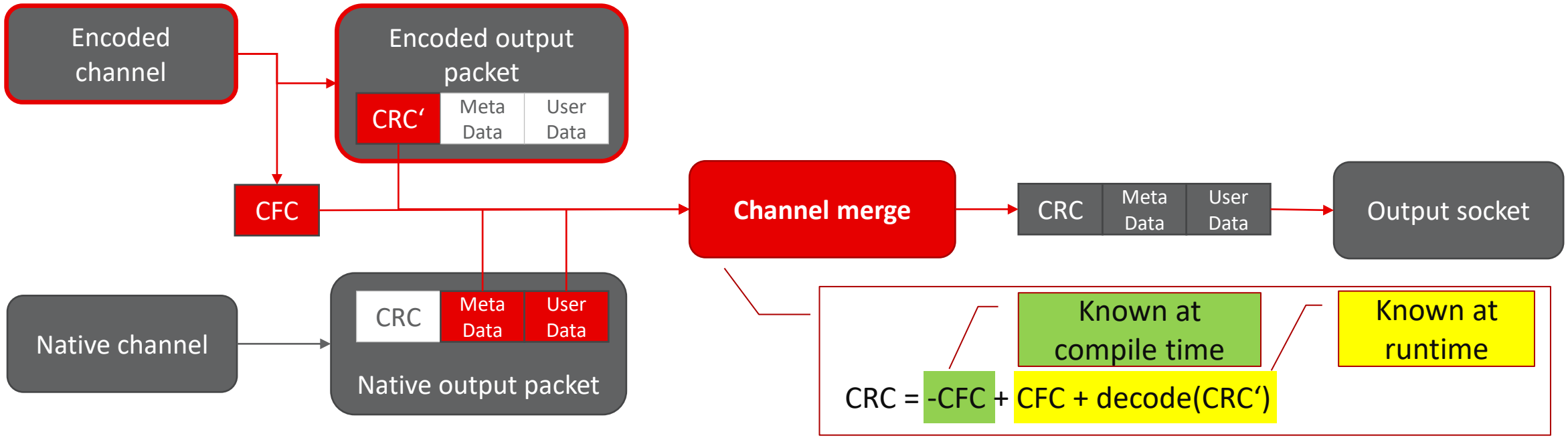

CFC = control-flow check signature
CRC = checksum

# Channel merge using a checksum




CFC = control-flow check signature
CRC = checksum

# Channel merge using a checksum




CFC = control-flow check signature
CRC = checksum

# (Partial) error analysis

| Error | Diagnosis | Handling |
|---|---|---|
| on application level | application must compute invalid output packet | actor detects and handles fault as channel failure |
| data-flow error in one channel | CRC‘ from encoded channel will not match the packet data computed by the native channel | actor detects and handles fault as channel failure |
| data-flow error in both channels | because of diversity: CRC‘ of encoded channel will not match the packet data computed by the native channel & CRC‘ invalid encoded | actor detects and handles fault as channel failure |
| control-flow error | CFC falsified → CRC falsified | actor detects and handles fault as channel failure |
| cycle not executed | no output packet generated | actor detects and handles timeout as channel failure |

# Advanced topics: initialization & clocks

**Initialization**

- example for a non-cyclic function
- store result of initialization in a global variable
- cyclic function must check this global variable in every cycle in both channels
  - error = application level error
- error propagation

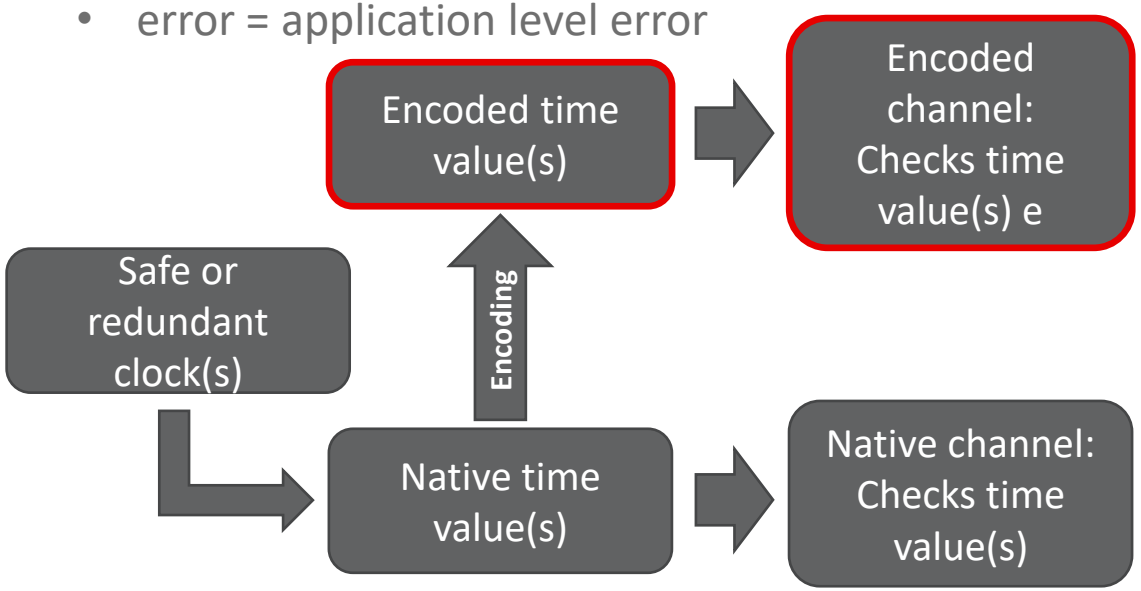
# Advanced topics: initialization & clocks

**Initialization**

- example for a non-cyclic function
- store result of initialization in a global variable
- cyclic function must check this global variable in every cycle in both channels
  - error = application level error
- error propagation


**Time as input**

- local clock alone is not safe
- safe clock or second redundant clock required
- time value must be checked by both channels:
  - error = application level error



# Summary

**Interoparability**

- sensors and actors do **not** need to know diversified encoding or coded processing
- connection via safe protocols
  - industry standards (e.g. PROFIsafe, CANopen safety, …)
  - proprietary protocols

**Main Idea**

- diagnosed errors are propagated as channel error

**Safety in Software**

- diversified encoding & coded processing enables diagnosis of random errors on 1-channel hardware
- safety-critical applications can run on the same HW as non-safety-critical applications