Secure Debug architecture on RISC-V

Yann Loisel, SiFive Principal Security Architect

3rd RISC-V Week
March 2021, 31st

use cases, the challenge

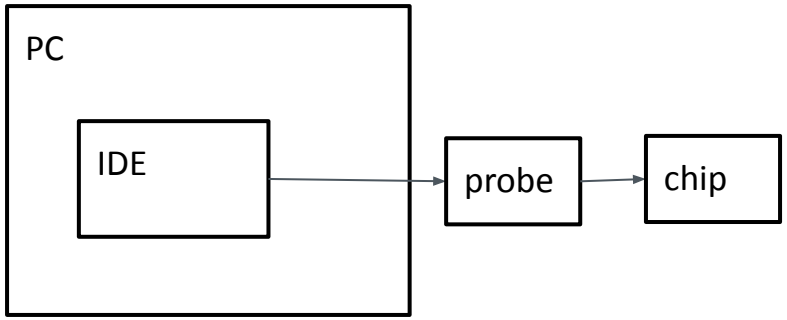
"To help bring up and debug low-level software and hardware, it is critical to have good debugging support built into the hardware"
(RISC-V debug spec)

The debug interface is the best and the worst of things; best for debugging,worst for security

scenario: A RISC-V-based platform owner does not want to disable the RISC-V debug interface but wants to securely control who has access to it.
The profiles of who can access are:

- a developer,
- a production operator,
- a maintenance operator,
- a technician,
- a RMA operator

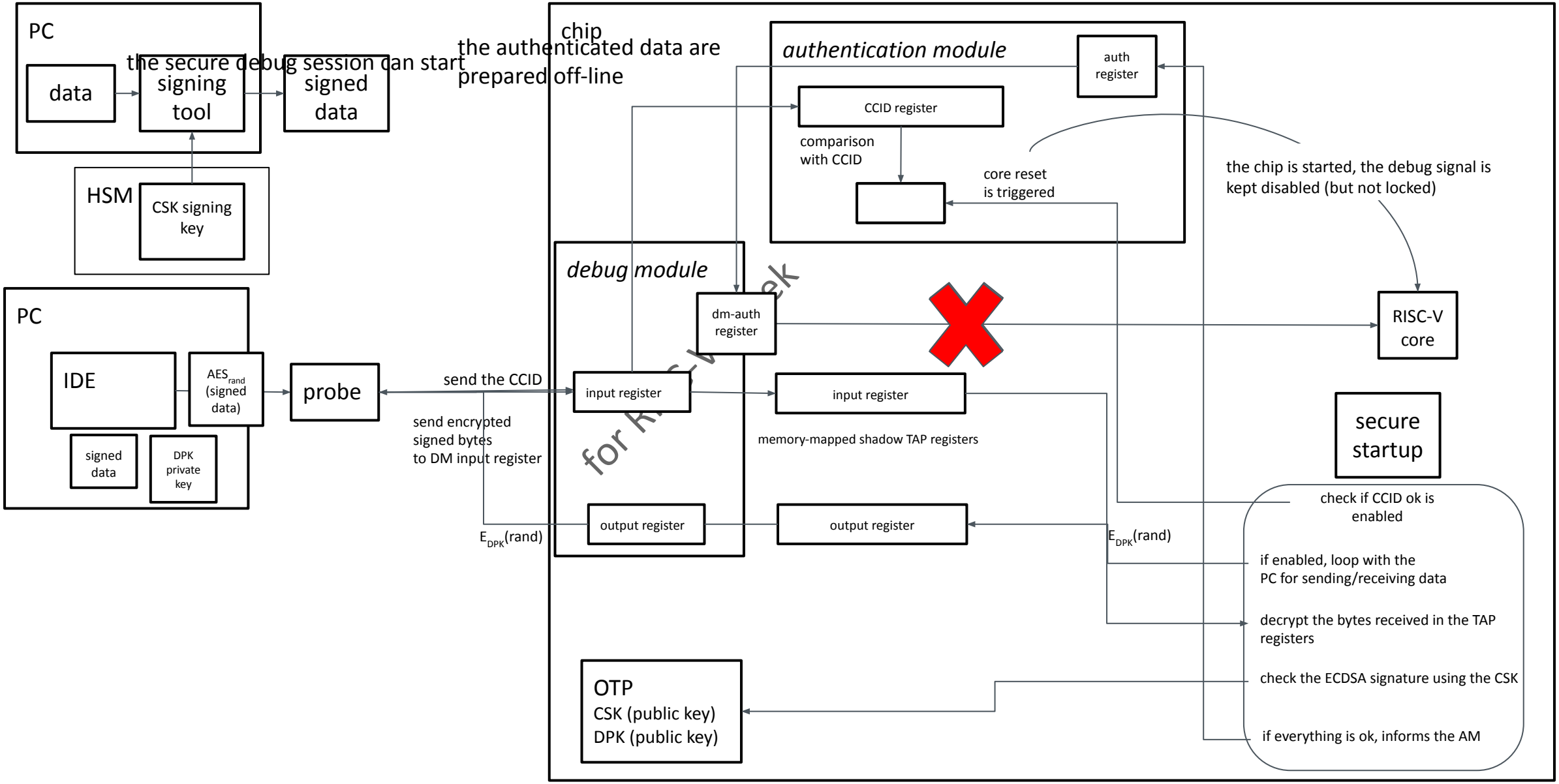
introduction



- a secure protocol is proposed to control the core debug using the chip debug interface between the PC and the chip via the probe
- this protocol is mandatory to enable the communication on the debug interface
- this secure debug proposal benefits from the availability of the Secure Startup firmware on the chip
- the protocol is using authentication with public crypto and symmetric encryption (optional)
- the probe initially sends a magic sequence, the CCID, to the debug module to inform the chip about a debug session opening request: the CCID is not a secret, only an identification value
- Once the identification is performed, the chip firmware is able to manage the communication
- if the authentication is successful, the debug mechanism is enabled.

- CCID: Customer/Chip Identification: magic word
- IDE: integrated development environment
- DM: debug module
- AM: authentication module
- CCID flag: a memory-mapped flag, read-only by the software

the authentication process



summary

1. **there is a dedicated public keypair, DPK (Debug public keypair)**
   a. the public key value is stored in the chip's OTP
   b. the private key value could be stored in the host PC or could be in the probe or in a HSM

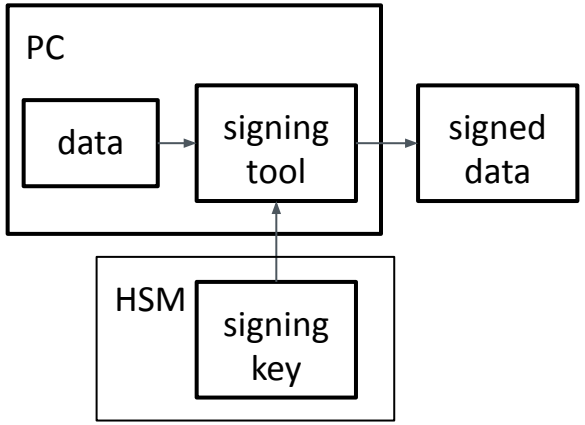
2. **instead of only sending the *ready* information to the host, the chip performs a random number generation and send this number, encrypted with DPK (public key)**
3. **the host decrypts the received data thanks to the DPK private key and extracts the random value**
4. **the host encrypts the signed data with a AES using the random value as the encryption key**
5. **the chip decrypts the encrypted data with the random value and checks the signed data**

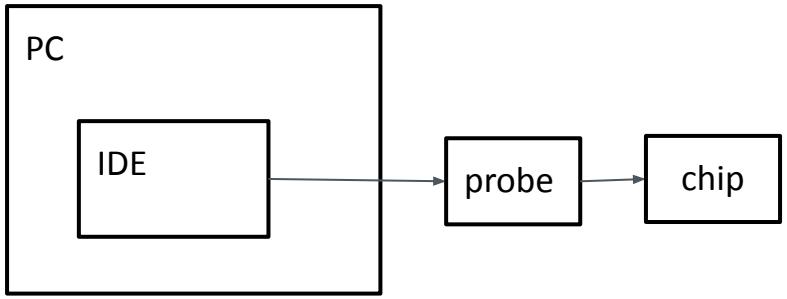
This exchange is dynamic because of the random value, so a replay attack is not possible

secure debug proposal: conclusion and benefits

- **the solution innovates in using the firmware and the core to manage the security: there is no dedicated IP block, attached to the debug/authentication modules**
- **the purpose is to have these modules and the hardware state machine as simple as possible and as flexible as possible**
- **the enabling is under the customer control, thanks to cryptography and the CSK**
- **there is a high level of customization and granularity, regarding the authentication means (customer level, chip level)**
  - CCID and signed data can contain P/N, S/N, or any other customer information useful for limiting the risk
- **there is no embedded secret, so no risk for extracting and re-using this value**
- **the solution is RISC-V-debug spec-compliant**
- **the solution is more secure than 1-nothing, 2-a password-based mechanism**
- **the solution is not a simple challenge-response, it is a complete protocol that establishes a secure channel over the debug interface**

secure debug proposal: generic scenario





Data preparation:

- the data to be transferred are prepared off-line, using a PC tool connected to the signing key, (optionally customized with the chip S/N for a better chip targeting)
- the data and their signature can be stored within the probe or stay on the PC (they're not sensitive as they don't contain any secret value)

debug secure activation:

1. the chip is started, the debug signal is kept disabled (but not locked)
2. the CCID identification pattern is sent by the probe to the chip
3. the *debug module* transfers it to the *authentication module*
4. the AM sets the CCID-ok flag if the CCID is ok, otherwise lock the debug signal in disabled state and exit
5. the AM resets the core
6. the Secure Startup code detects the CCID-ok flag and starts the secure debug routine
7. the Secure Startup performs the host authentication process and validates the signed data
8. if successful, the Secure Startup informs the AM
9. the AM instructs the DM to enable the debug

- CCID: Customer/Chip Identification: 128-bit magic, unique, random word
- IDE: integrated development environment
- DM: debug module
- AM: authentication module
- CCID flag: a memory-mapped flag, read-only by the software