- on embedded sys, code is deployed on thousands of systems so anyone can get physical access, although designer has more control over what applications are running

Attack Vectors

- ways in which the system can be attacked

Threat Model

- what are we assuming the attacker has access to?

Firmware Security

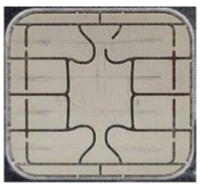
- code is stored in flash, if someone has access to flash, can read all code
- when programming the board, JTAG interface is used
  - custom protocol used for programming microcontrollers
  - If you add probes to these pins on the board, you can just read and write flash the same way Keil does when you program your board.
- if system uses **separate** flash chip, likely uses SPI
  - from pinout for the flash chip, can R/W the signals
- **defence** STM allows enabling *readout protection RDP*, no R/W program access to flash/SRAM while connected to a computer

Supply Chain Vulnerability

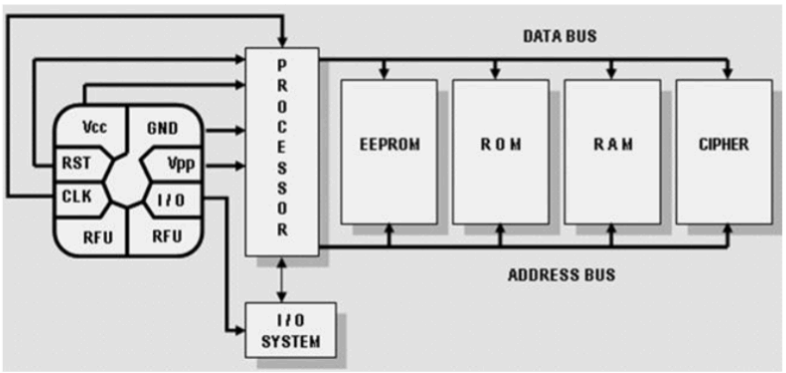
- **component authenticity** make sure each component on chip is sourced from reliable company
  - hidden bluetooth commands in ESP32
  - vendor-specific commands in bluetooth firmware; allow low-level control over bluetooth functions
  - 29 of them; memory manipulation, MAC address spoofing, packet injection

Smart Card Security

- Smart cards (debit or credit cards) are actually embedded systems.
- A card transaction is just messages sent between the chip in your card and the card reader/terminal.



RFU – Reserved for Future Use



MITM Attack

- intercept and alter communication
  - pin correct, card sends verification code (0×9000) to reader to approve the transaction

Side-Channel Attacks

- leak info by monitoring the execution of a program
- **power side channel attack** measure the power of the system while it is running this code to recover each bit of the key
  - no need for connection to Vdd line

        - can measure EM emanations from chip

Defense

- constant time algorithm; randomize order of execution

Privacy

- essential to handle data carefully
    - camera → hackers could log in and get access

Differential Privacy

- computational technique when dealing with large amounts of private data
    - only care about group statistics and protect each individual datapoint
    - DP adds small noise to each data point; protects data, no effec on statistics

DP - Example

- Computing the average for an assessment.

$$Average = \frac{x_1 + x_2 + x_3 + \ldots + x_N}{N}$$

- But this means all your grades must be stored in a single place (Quercus).
  - If Quercus gets hacked, all your grades are made public.

- Instead, if we ONLY wanted to calculate statistics, we could add some random noise $r \in [-K, K]$ to each grade.
  - Now, what is stored is $(x_1 + r)$, $(x_2 + r)$ etc.
  - Even if Quercus is hacked, they don't see your 'real' grades.

DP - Example

- The average calculation then becomes:

$$Average = \frac{(x_1+r) + (x_2+r) + (x_3+r) + \ldots + (x_N+r)}{N}$$

- If the random noise $r \in [-K, K]$, then, the average of r is 0.
  - Adding r to each grade does not affect the final average but anonymizes *each* grade.

- Caveat: The magnitude of the noise you can add, depends on N.
  - The more samples you have, the more noise you can add, which adds more privacy.

Ethical Design

- prevent harm, protect privacy, promote sustainability, ensure fairness