BlackHat Japan '08

Karsten Nohl—Univ. of Virginia

Disclosing Secret Algorithms from Hardware

Source: *New Yorker*

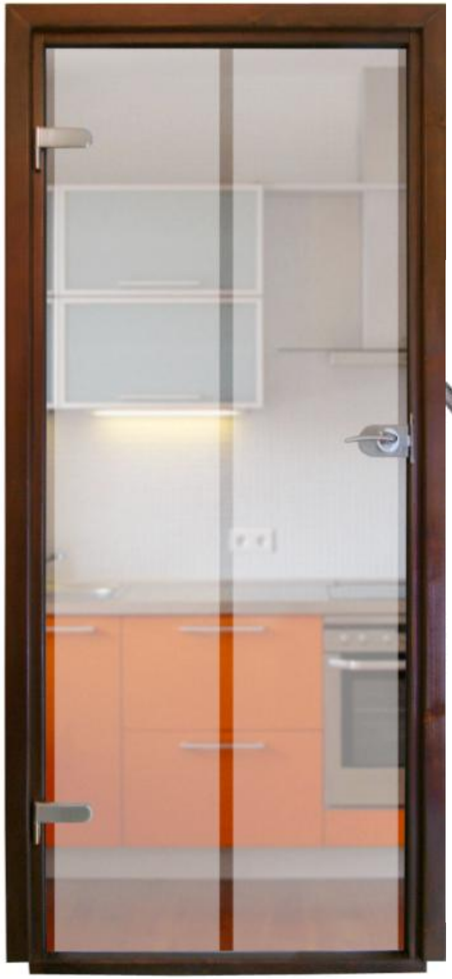
Some material courtesy bunnie or Flylogic.

Motivation

- Lots of critical infrastructure relies on secure hardware
  - Smartcards for access control, payment tokens
  - Satellite TV cards, car keys, printer cartridges, …
- Security often considered hard and expensive
  - Hence, often excluded from initial design
    - Protection added after problems arise
    - Patchwork security is harder and more expensive!

Accurate security estimates needed for risk assessment.

Example: Smart cards



Cryptographic cipher

Challenge-response protocol

Cryptographic cipher

Example: NFC Payment

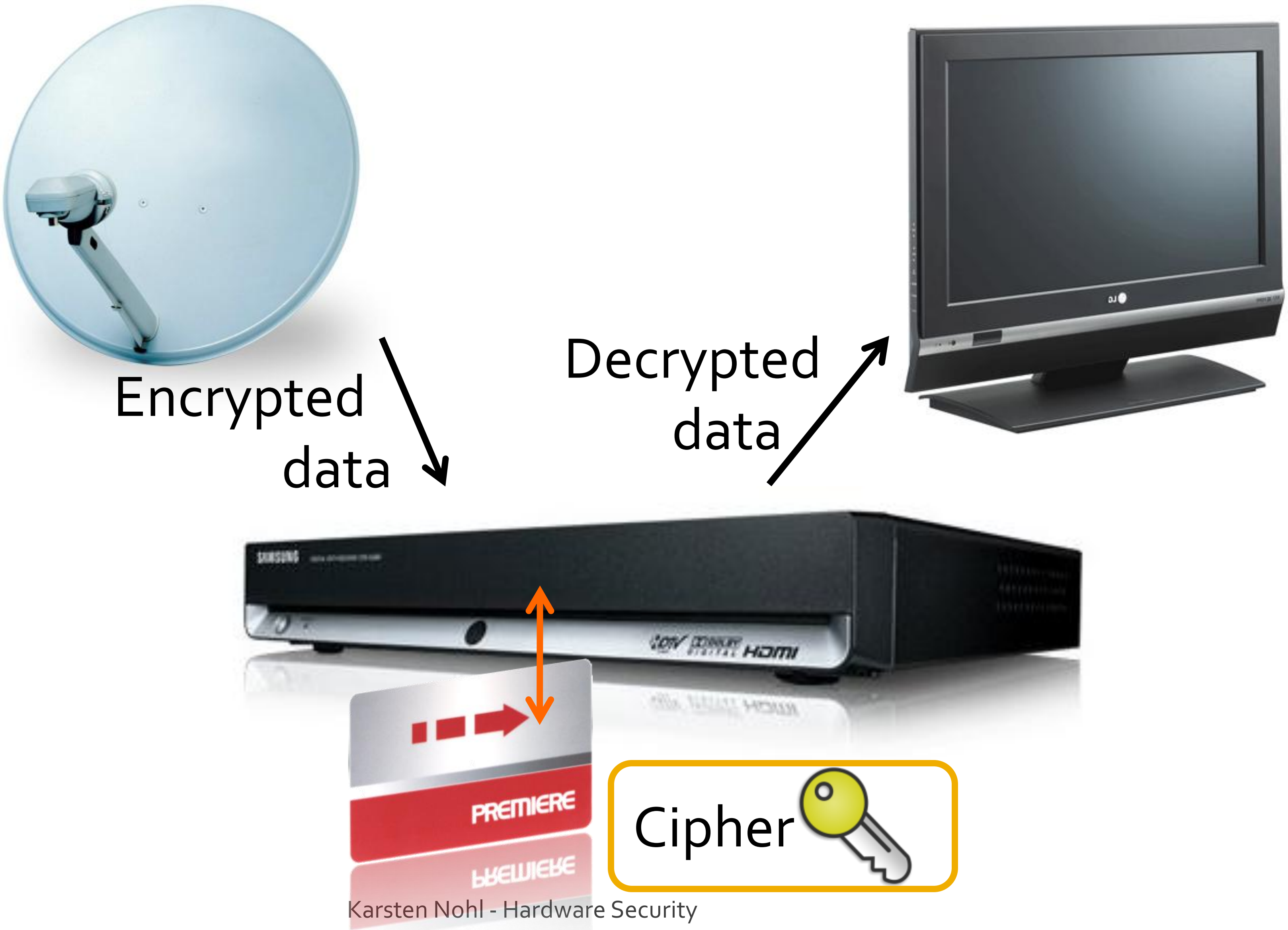
Cipher

Cipher

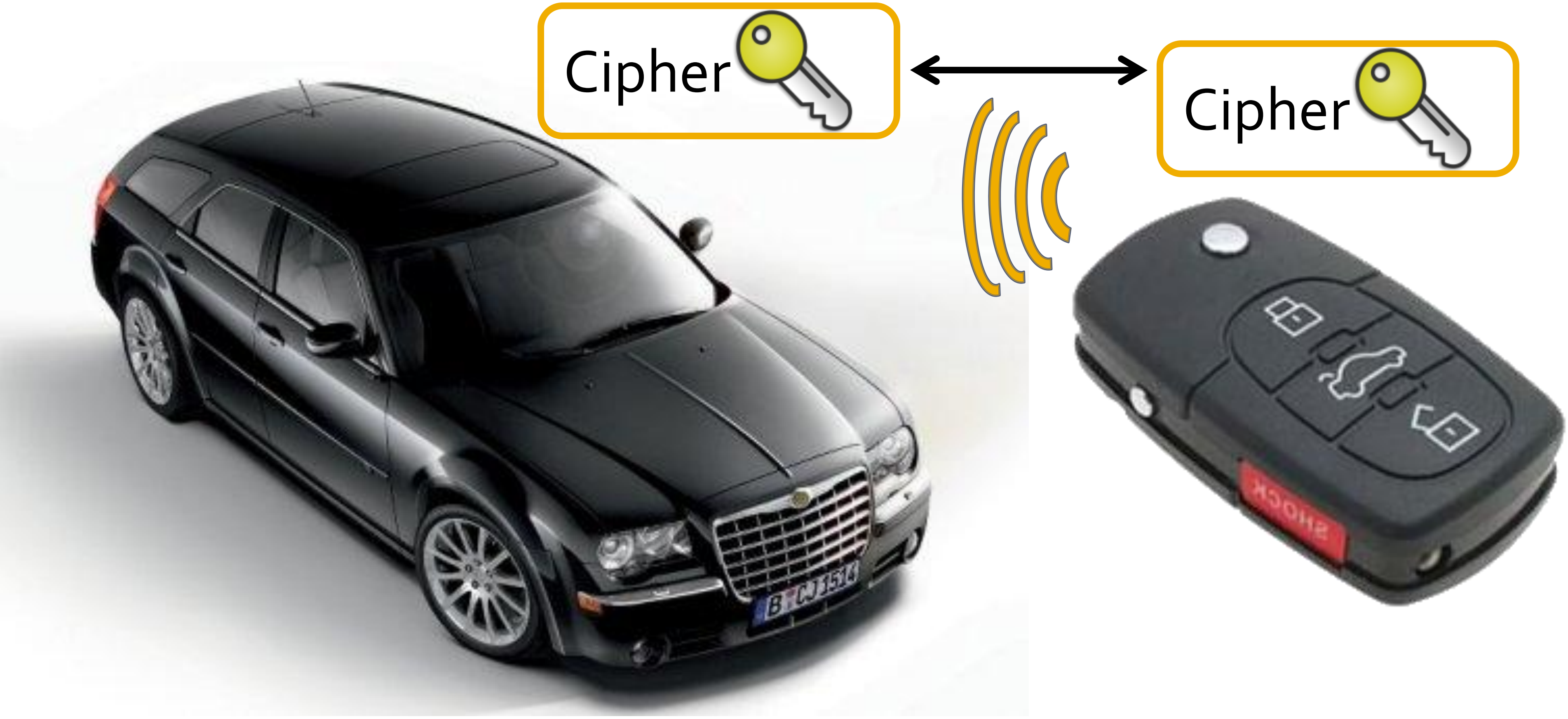
Backend Server

Near **F**ield **C**ommunication phone

Example: Satellite TV



Example: Car Key



Foundation of Hardware Security

- Hardware security relies on
  a) Key storage
  b) Cryptographic cipher (encryption)

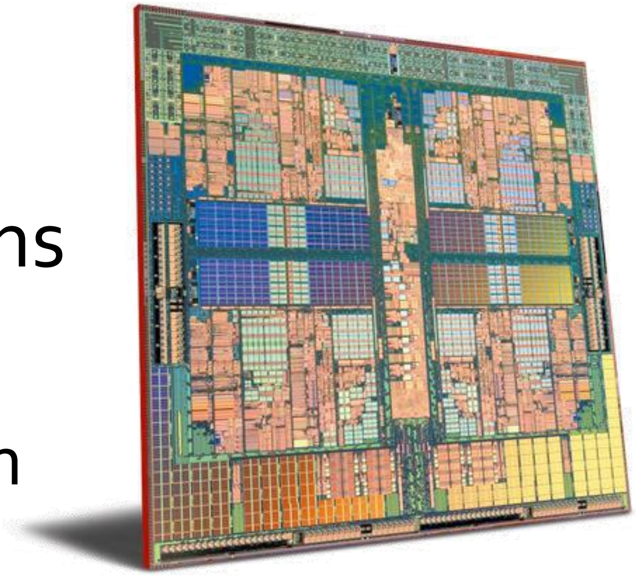
- Many systems fail to acknowledge lack of secrecy in hardware

  "There are no secrets in silicon." -bunnie

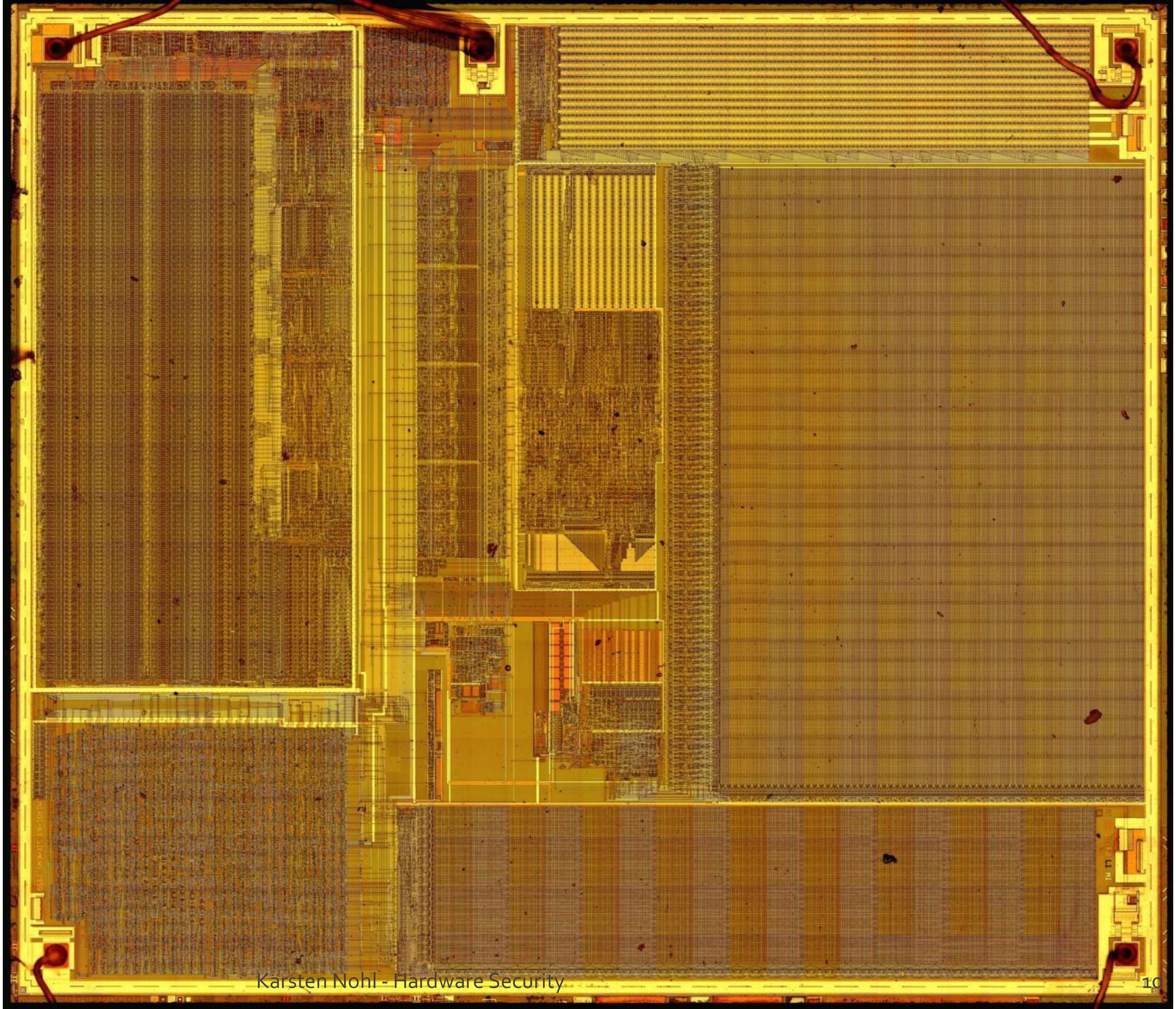
> This talk discusses common weaknesses in secure key storage and proprietary encryption.

Outline

- Microchip Basics
- Reverse-engineering algorithms
  - Finding secret ciphers
  - Exploiting proprietary encryption
- Security of key storage
- Demo / Hands-on lab



Microchip Basics

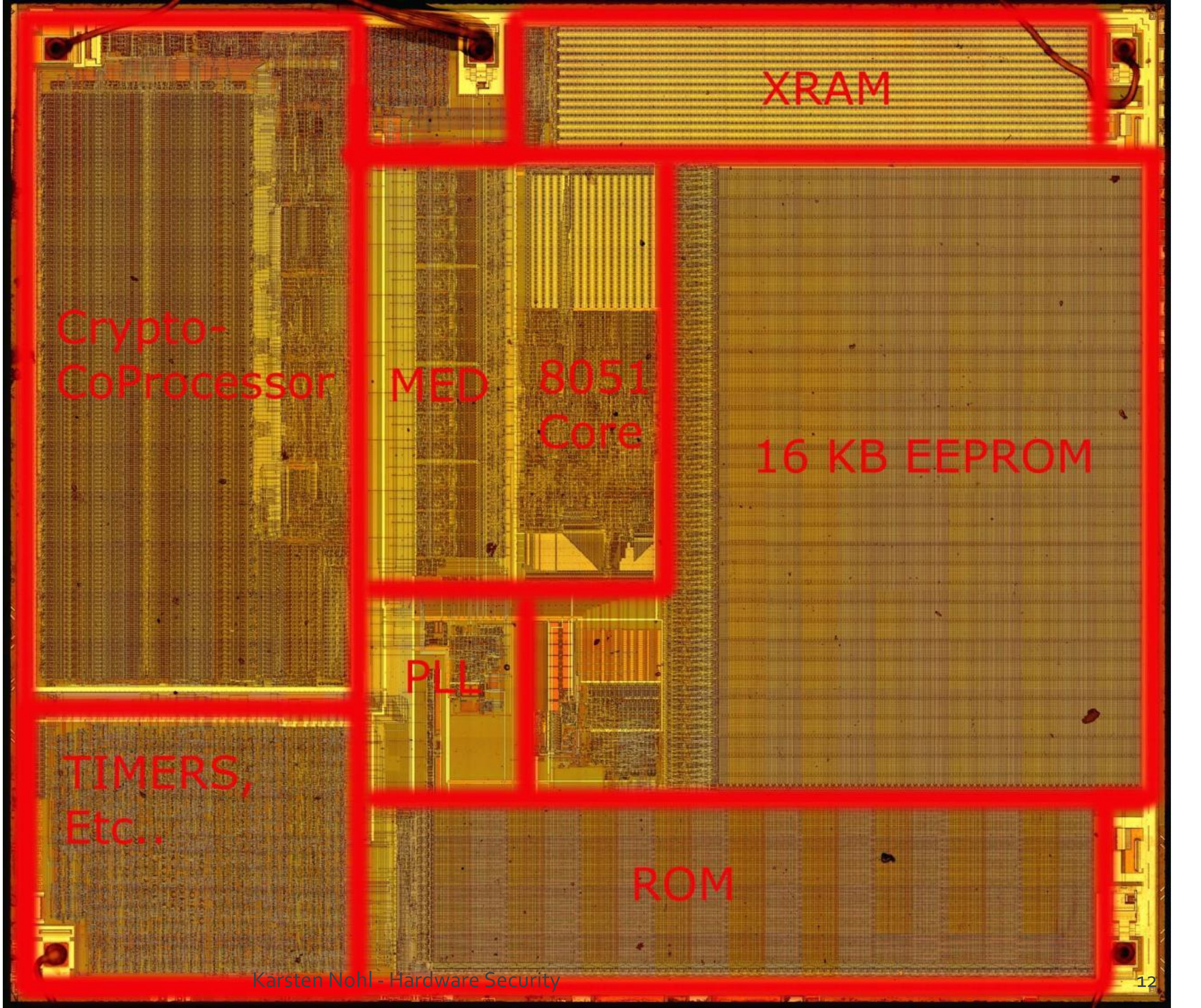


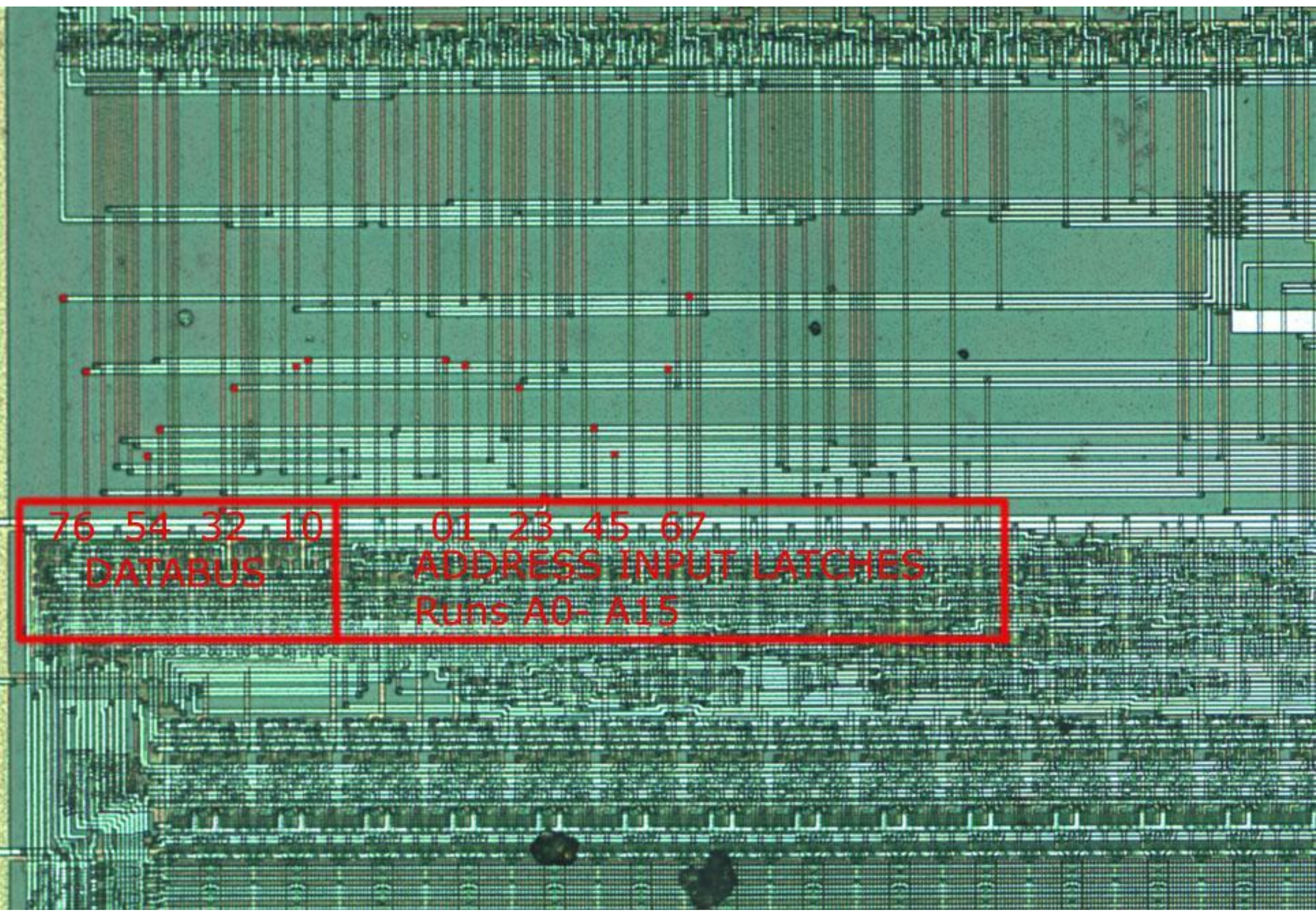
Infineon SLE66, courtesy Flylogic

Understanding Chip Layout

- Analyze chips using "last principles"
  - Principle #1: Chips are structured
    - Crucial for design partitioning and refactoring
  - Principle #2: Chips are designed to be read back
    - Enables prototyping and debugging
- Complement analysis with "first principle"
  - Principle #3: Nothing can be hidden in silicon
    - Chips are self-contained; hence all data, programs, and algorithms are available on the chip

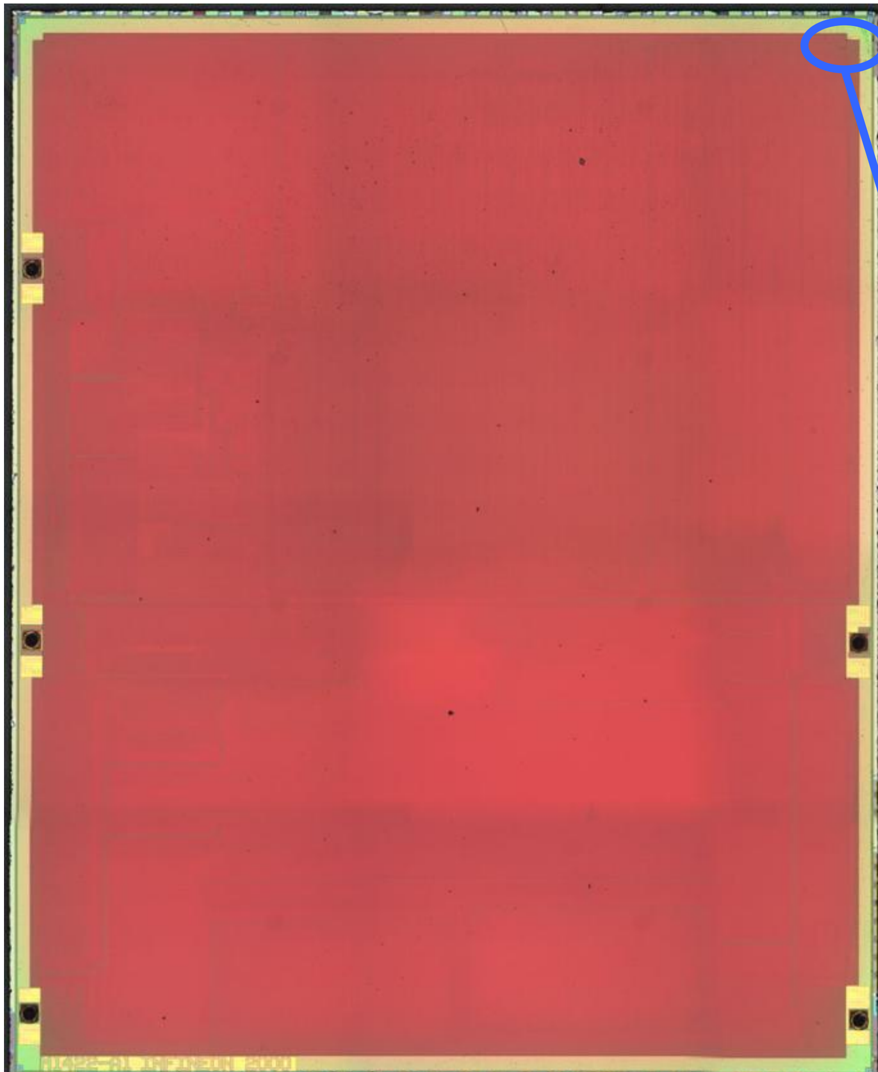
Infineon SLE66, courtesy Flylogic

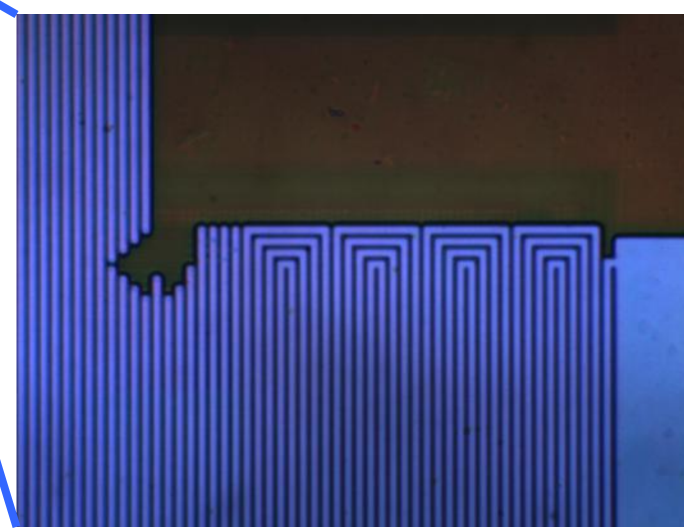




Infineon SLE66 address/data bus, courtesy Flylogic

Protection Meshes





- Can sometimes protect data, but not algorithms

Proprietary Encryption

Weak Cryptography

- Most security systems use cryptography
  - Too many use proprietary ciphers
  - Many are weak, but secret
- We find cipher implementations from silicon
  - Cheap approach, no crypto knowledge required
  - We want to enable you to do the same

"No more weak ciphers. No more paranoia."
Sean O'Neil

Motivating Example: RFID tags

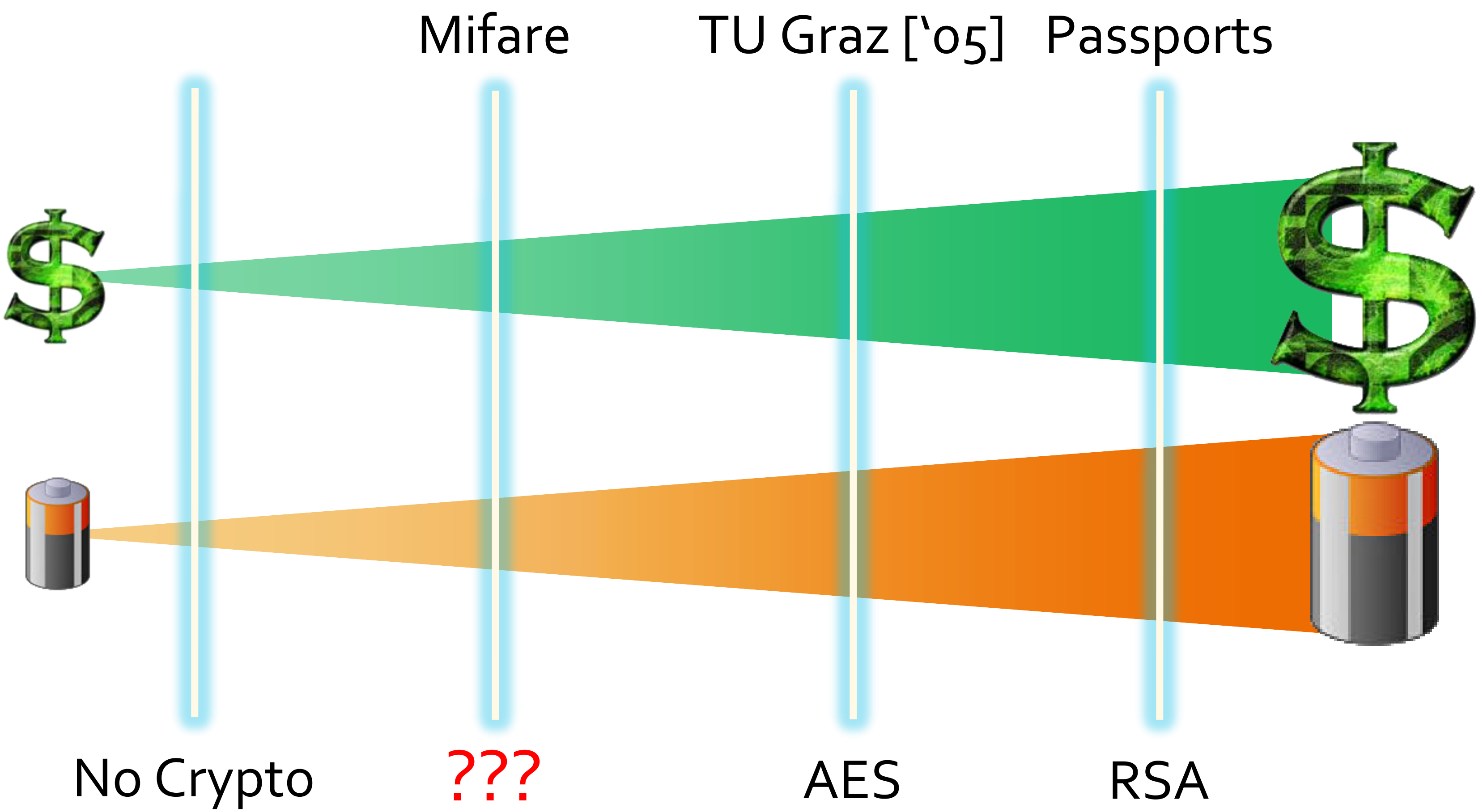
- **R**adio **F**requency **ID**entification
- Tiny computer chips
- Passively Powered

RFID Applications

- RFIDs are becoming ubiquitous
- Integrated in many *security* applications
  - Payment, Access Control
  - Passports, Car Ignition
  - Implants, …

RFIDs will be *universal identifier*. Might replace passwords, PINs, and fingerprints.

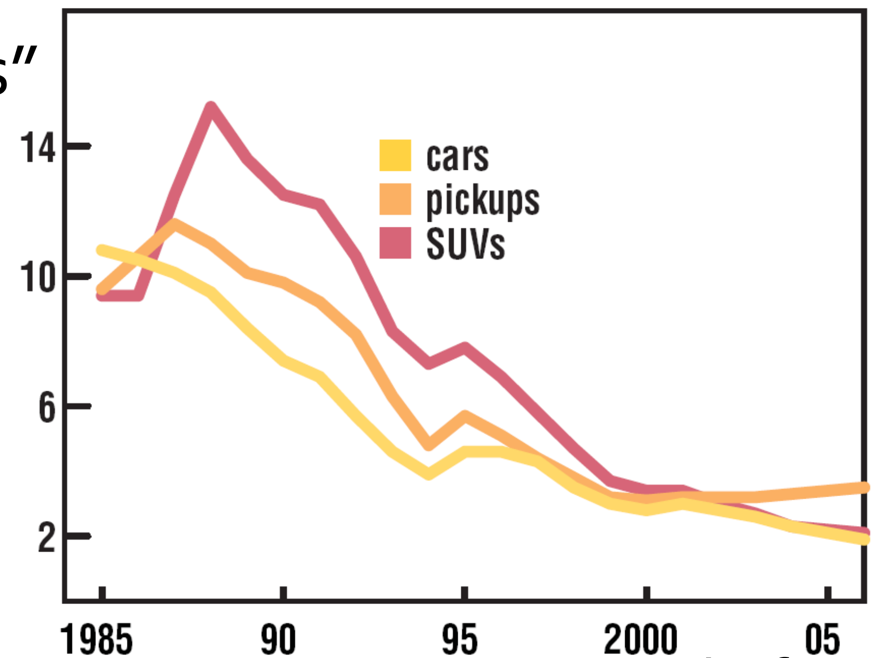
RFID-Crypto Mismatch



Mifare Security

- NXP claimed high security:
  - "approved authentication"
  - "advanced security levels"

- Specs suggested mediocre security at best:
  - Stream cipher with small 48 bit key



Car thefts

(source: hldi.org)

Our Project

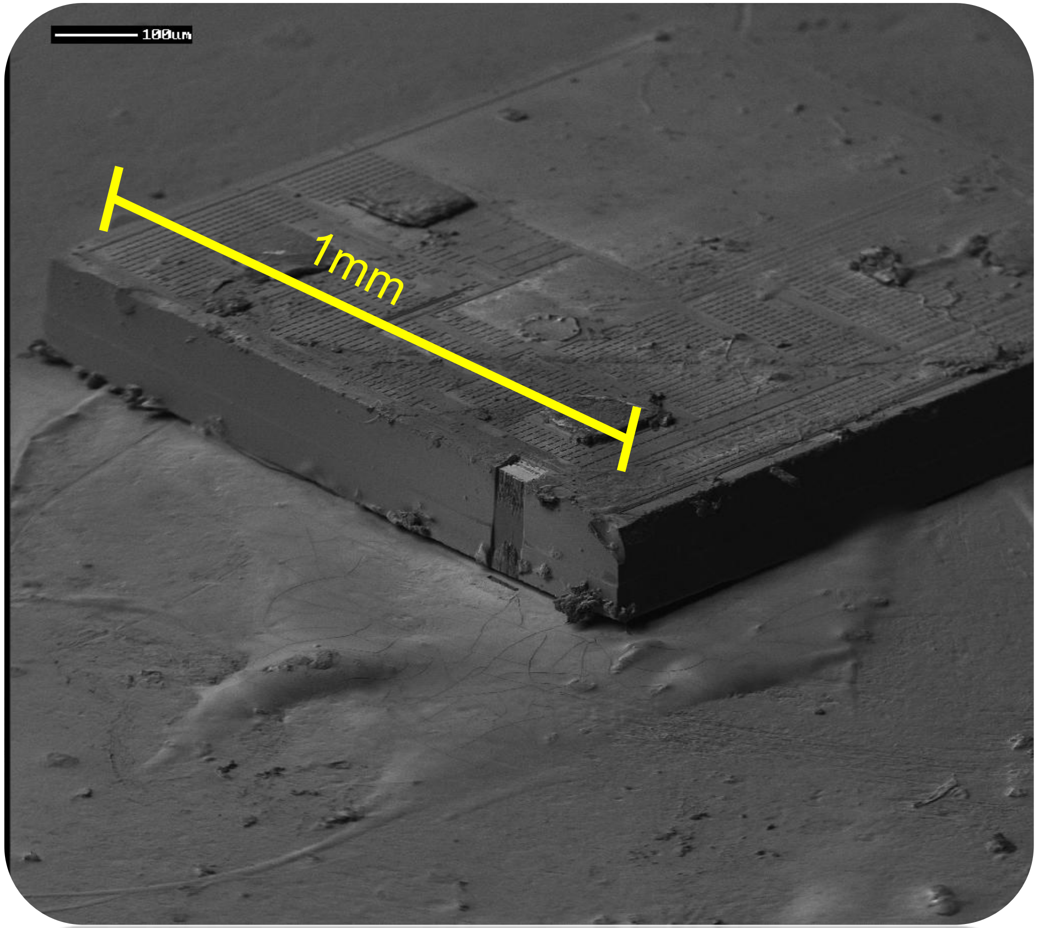
Reverse-engineered the secret cipher of the *Mifare Classic* RFID tag and evaluated its security

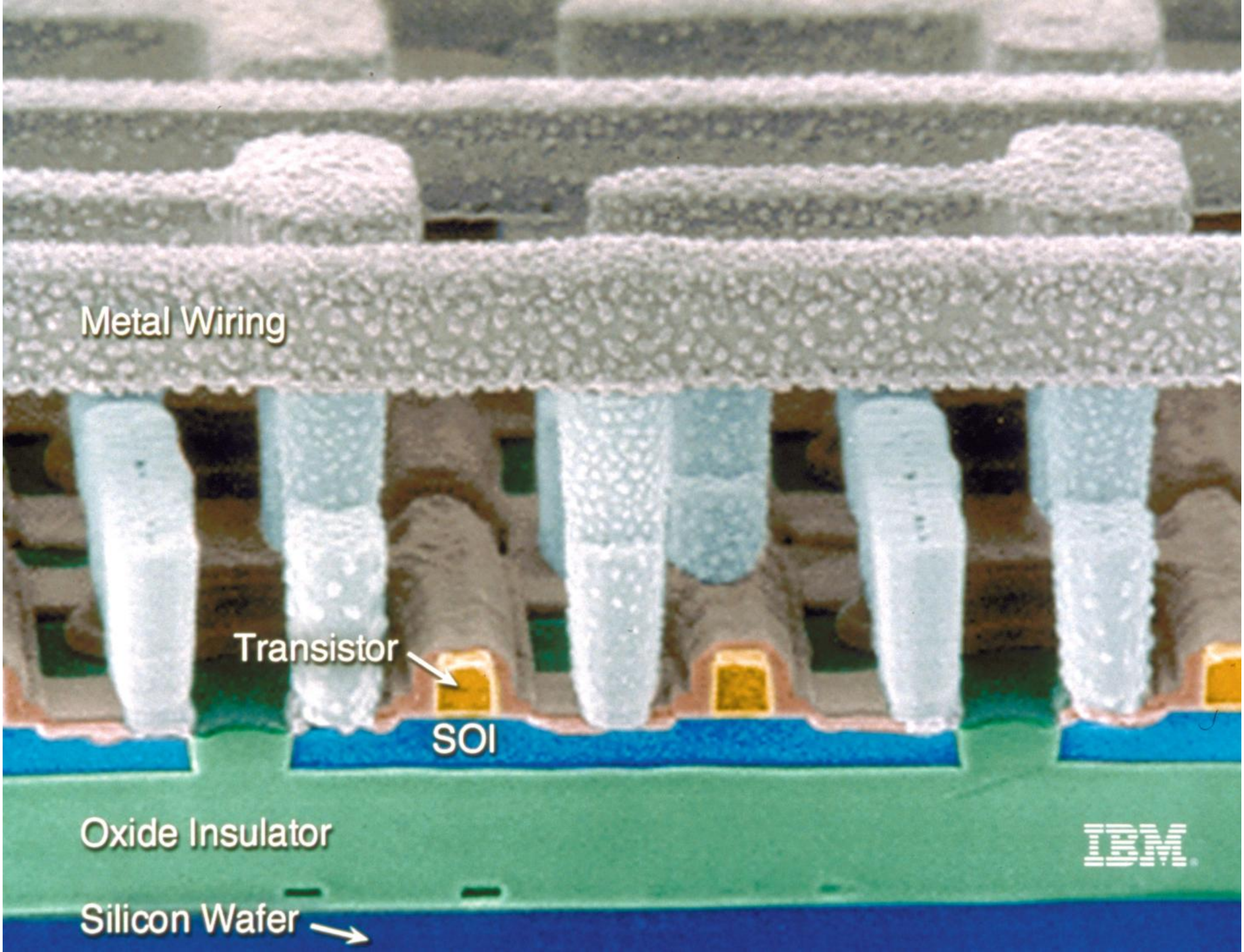
Proprietary Encryption: Reverse-Engineering

Obtaining Chips

- Chemically extract chips:
  - Acetone
  - Fuming nitric acid
- Shortcut: buy blank chips!

Mifare Classic RFID tag

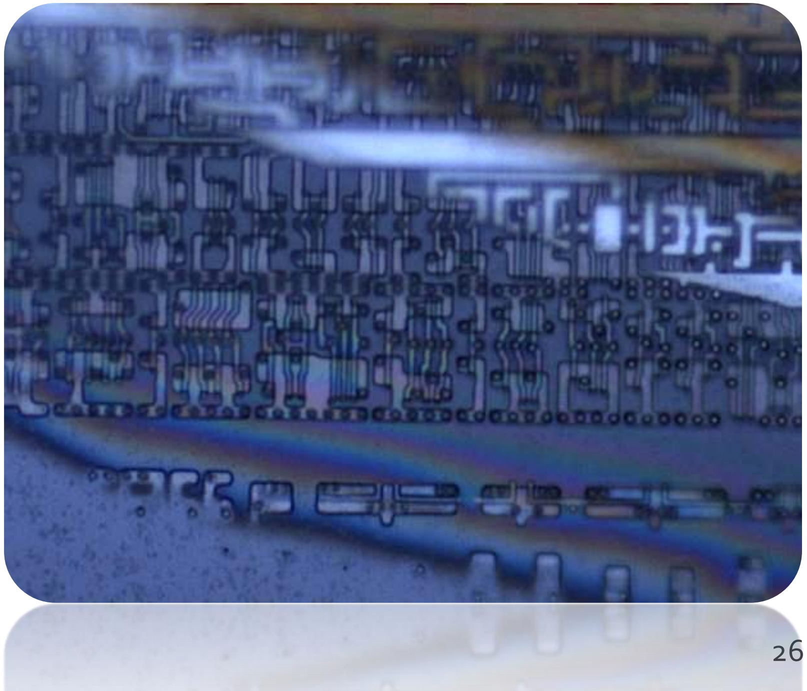



Metal Wiring
Transistor
SOI
Oxide Insulator
IBM
Silicon Wafer

Revealing Circuits

- Automated polishing with machine
  –or–
- Manually with sand paper
  - "On your kitchen table"

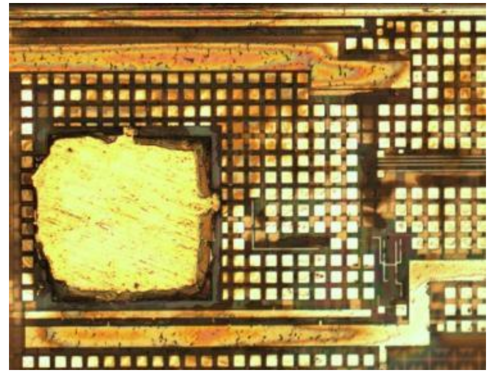
    -Starbug
- Potential problem: tilt
  - Solution: imbed chip in block of plastic



Imaging Chip

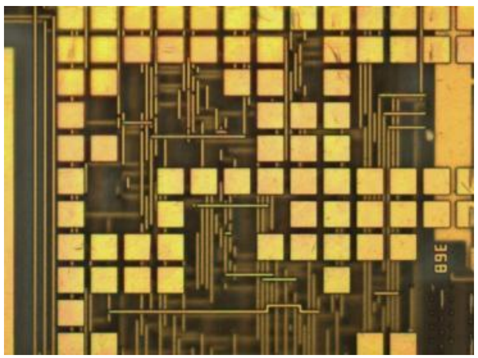
- Simple optical microscope
  - 500x magnification
  - Camera 1 Mpixel
  - Costs < $1000, found in most labs
- Stitching images
  - Panorama software (hugin)
  - Each image ~100x100 µm
- Align image layers

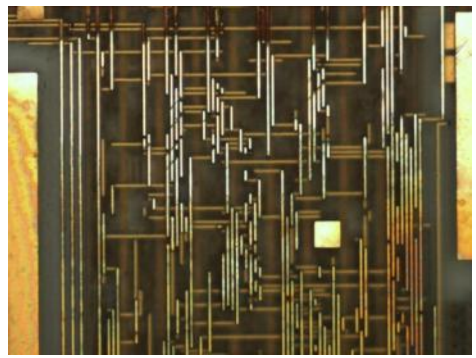
Chip Layers

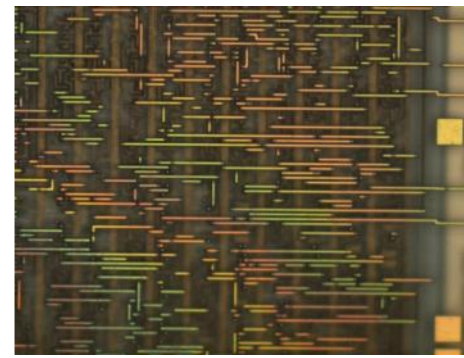


Cover layer

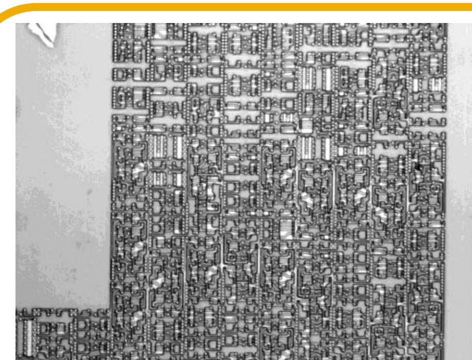
3 Interconnection layer





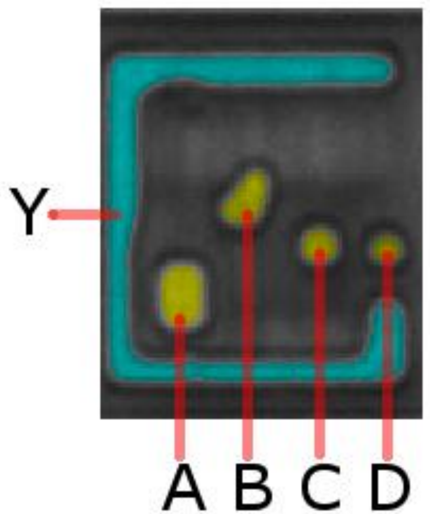


Logic layer

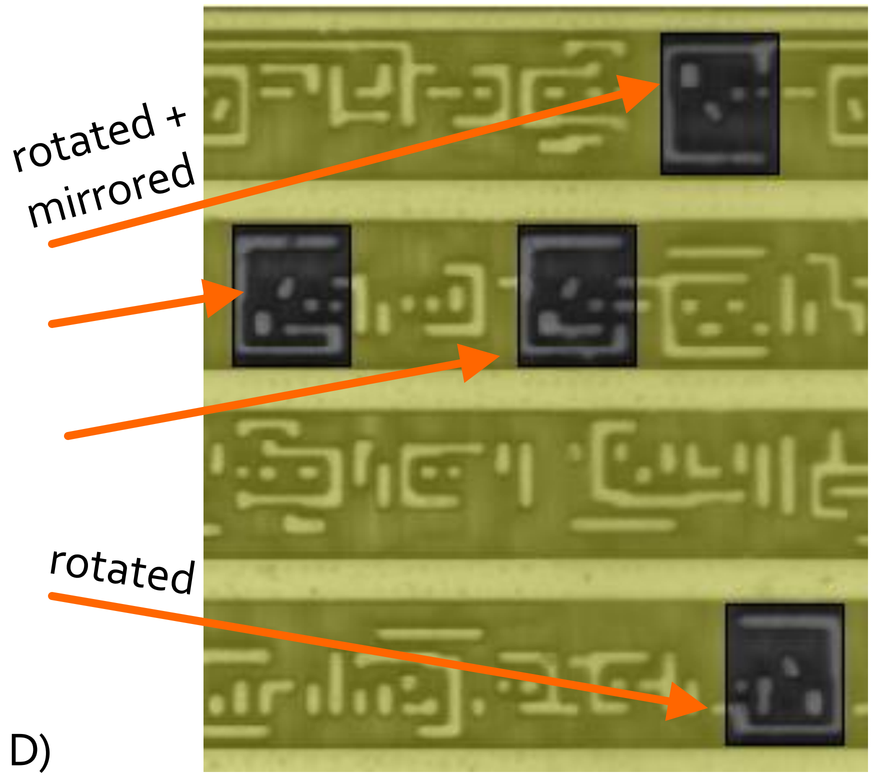


Transistor layer

Logic Cells

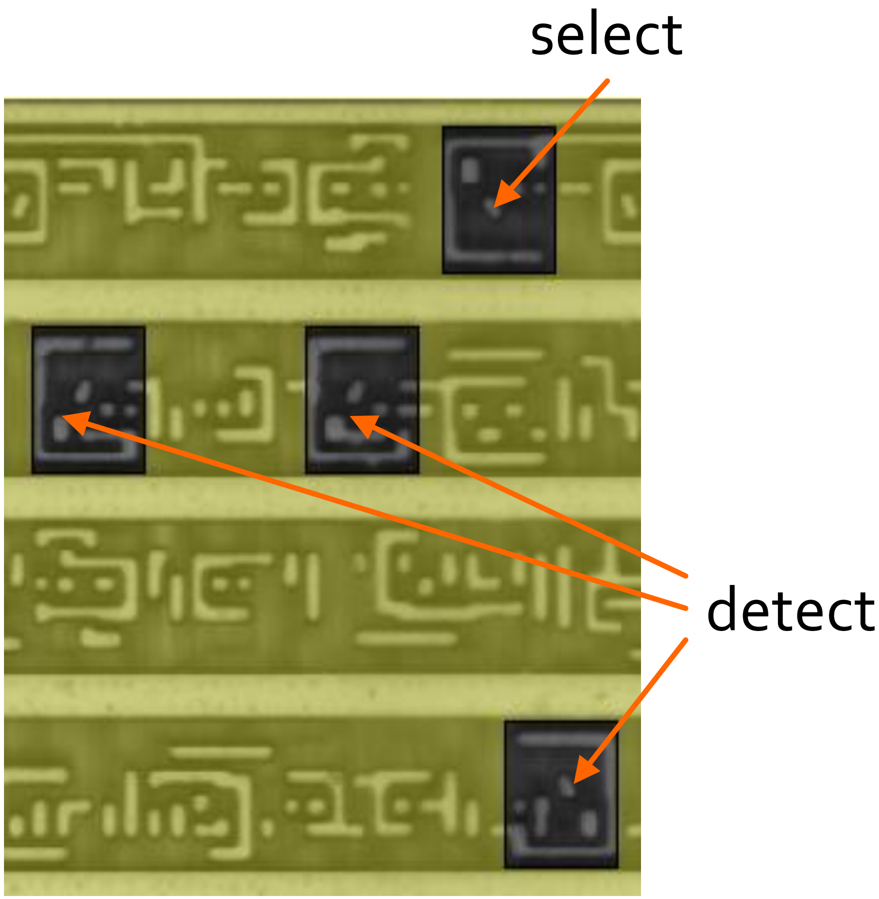


4 NAND: Y = !(A & B & C & D)

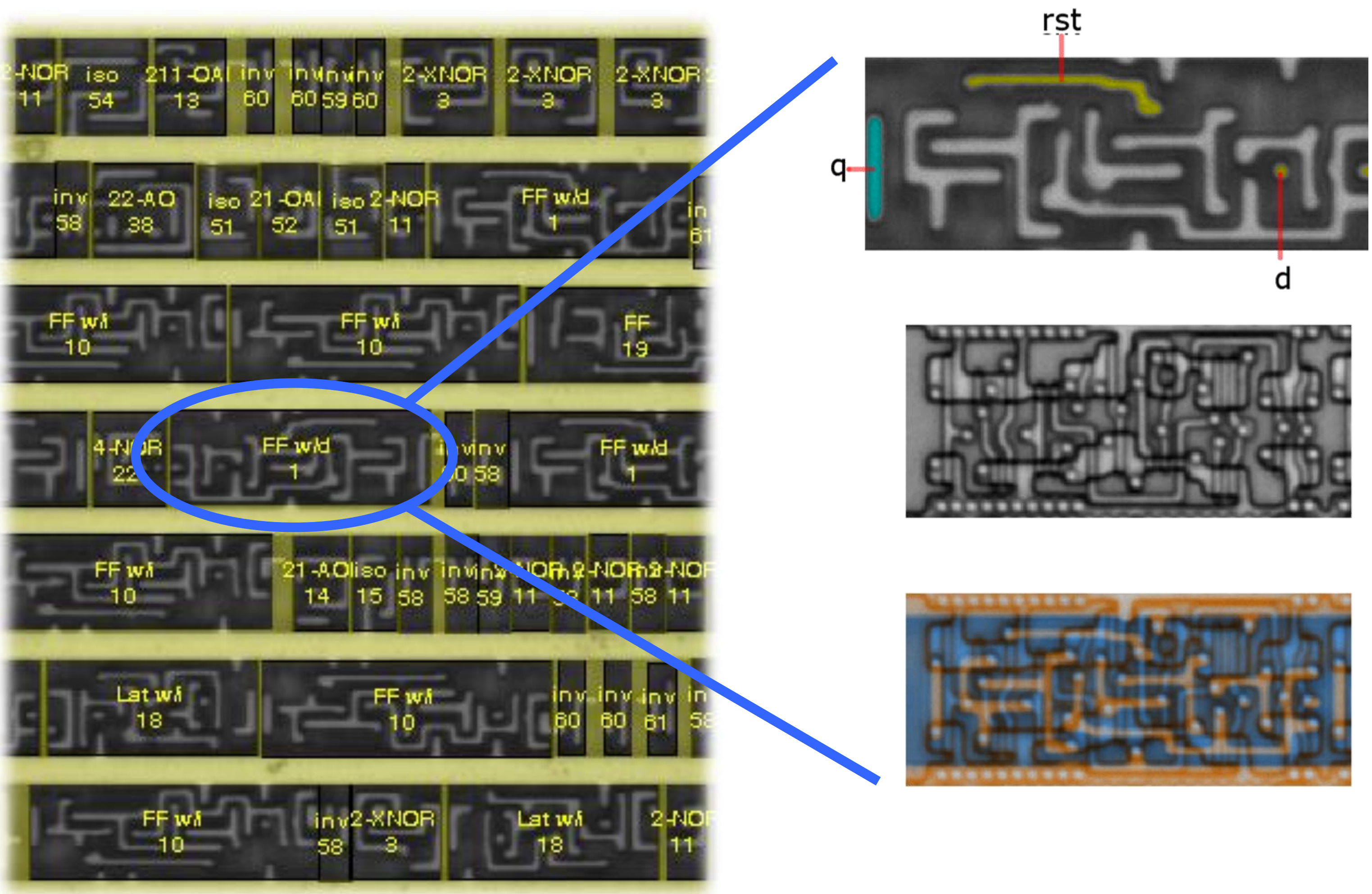


Standard Cell Library

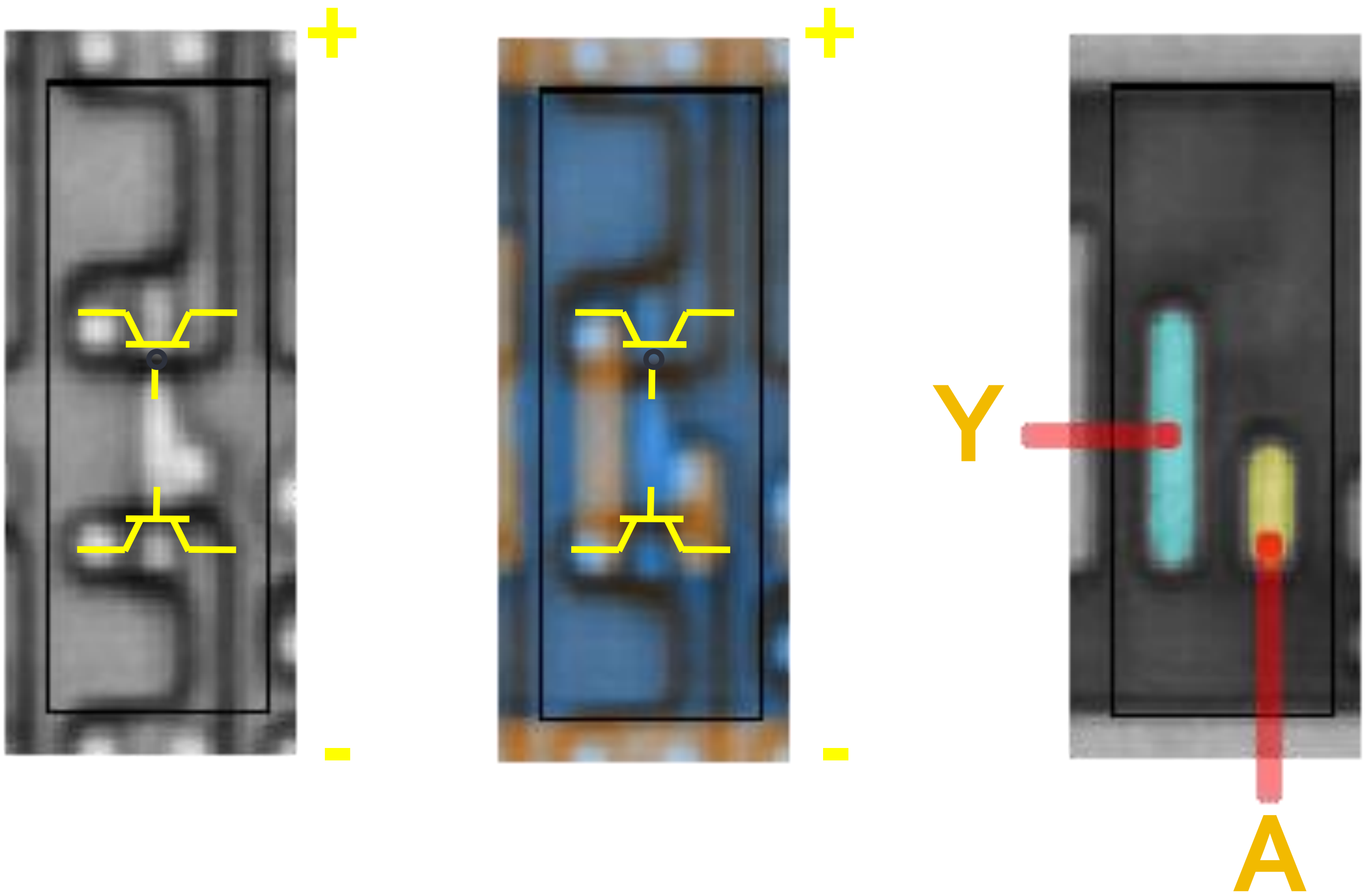
- Logic cells are picked form a library
  - Library contains fewer than 70 gate types
  - Detection automated (template matching using MATLAB)



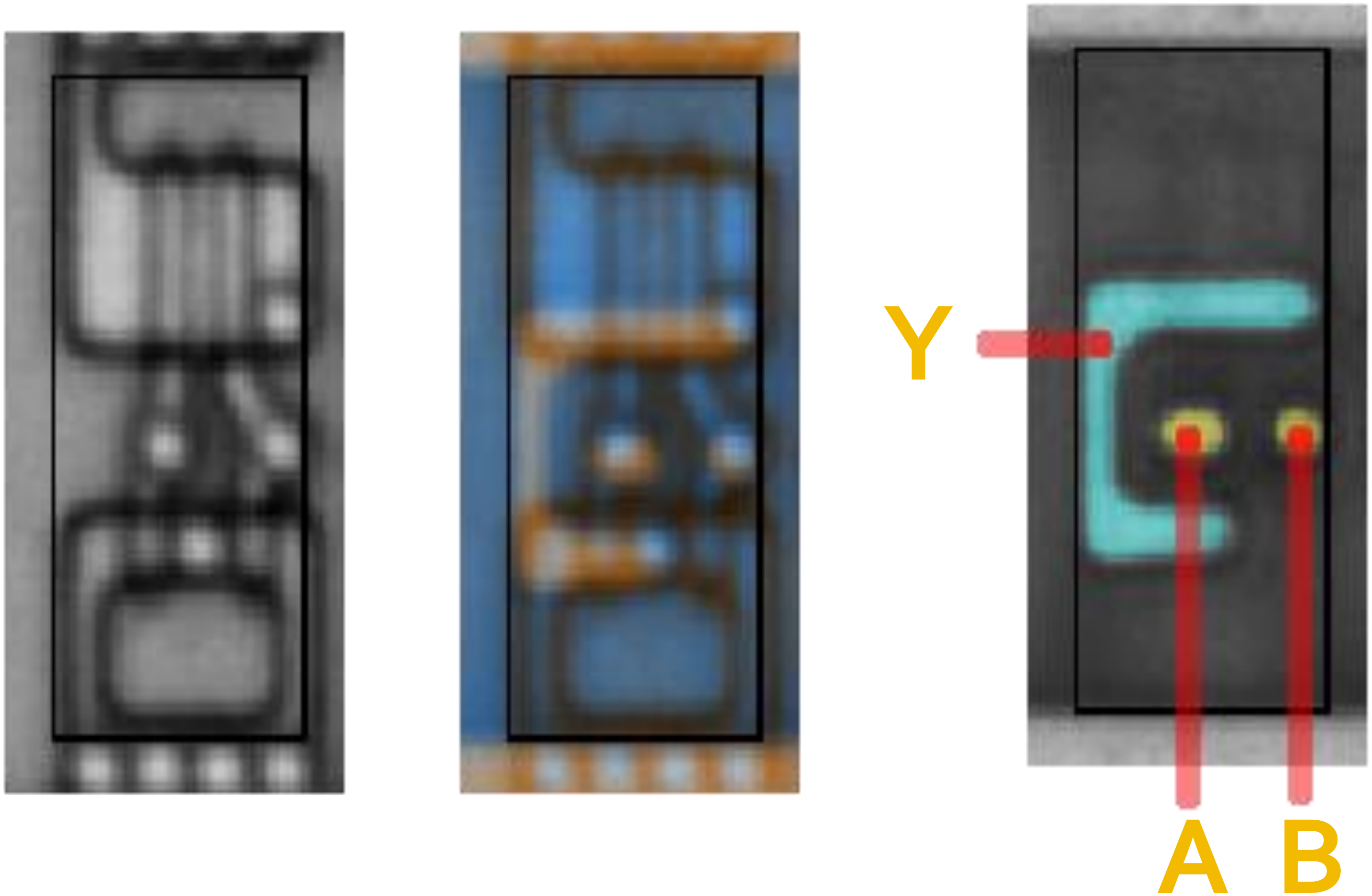
Automated Cell Detection



Logic Gates – Inverter



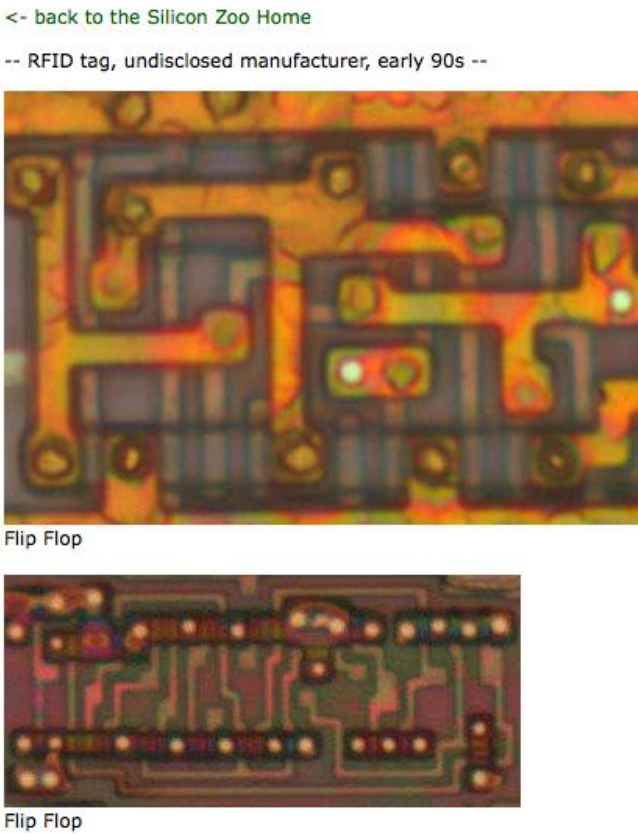
Logic Gates – 2NOR



The Silicon Zoo

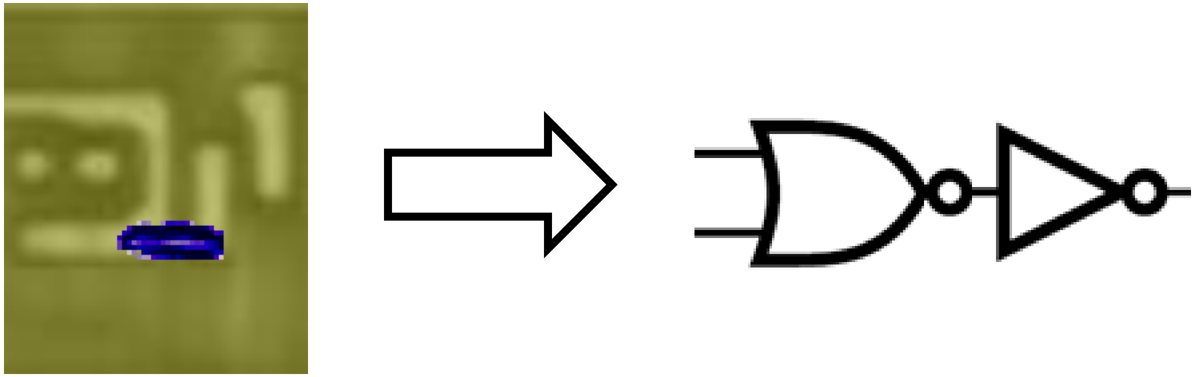
www.siliconzoo.org

- Collection of logic cells
- Free for studying, comparing, and reverse-engineering silicon chips
- Zoo wants to grow—send your chip images!



Logic Gates Interconnect

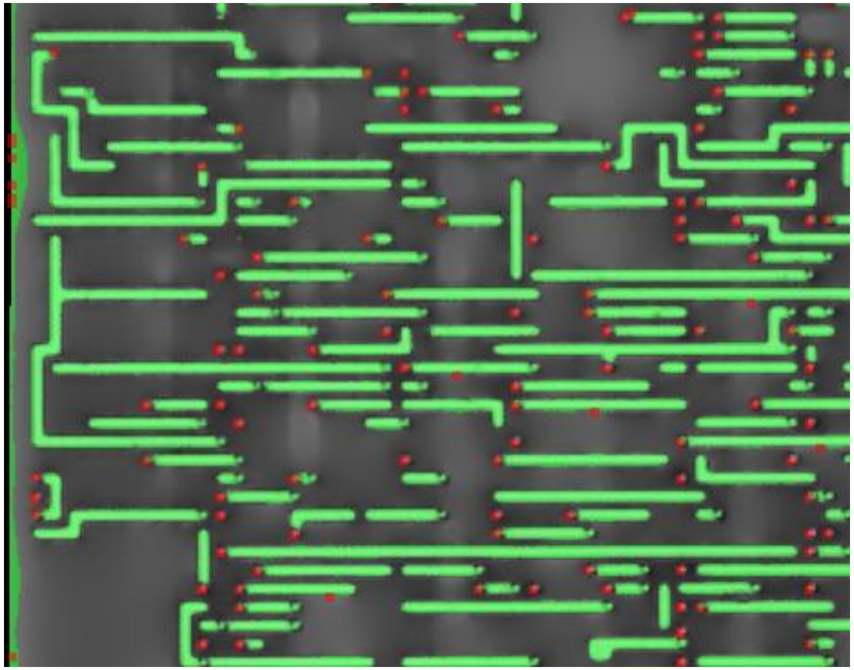
- Connections across all layers

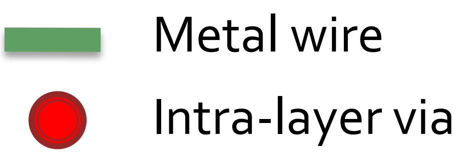


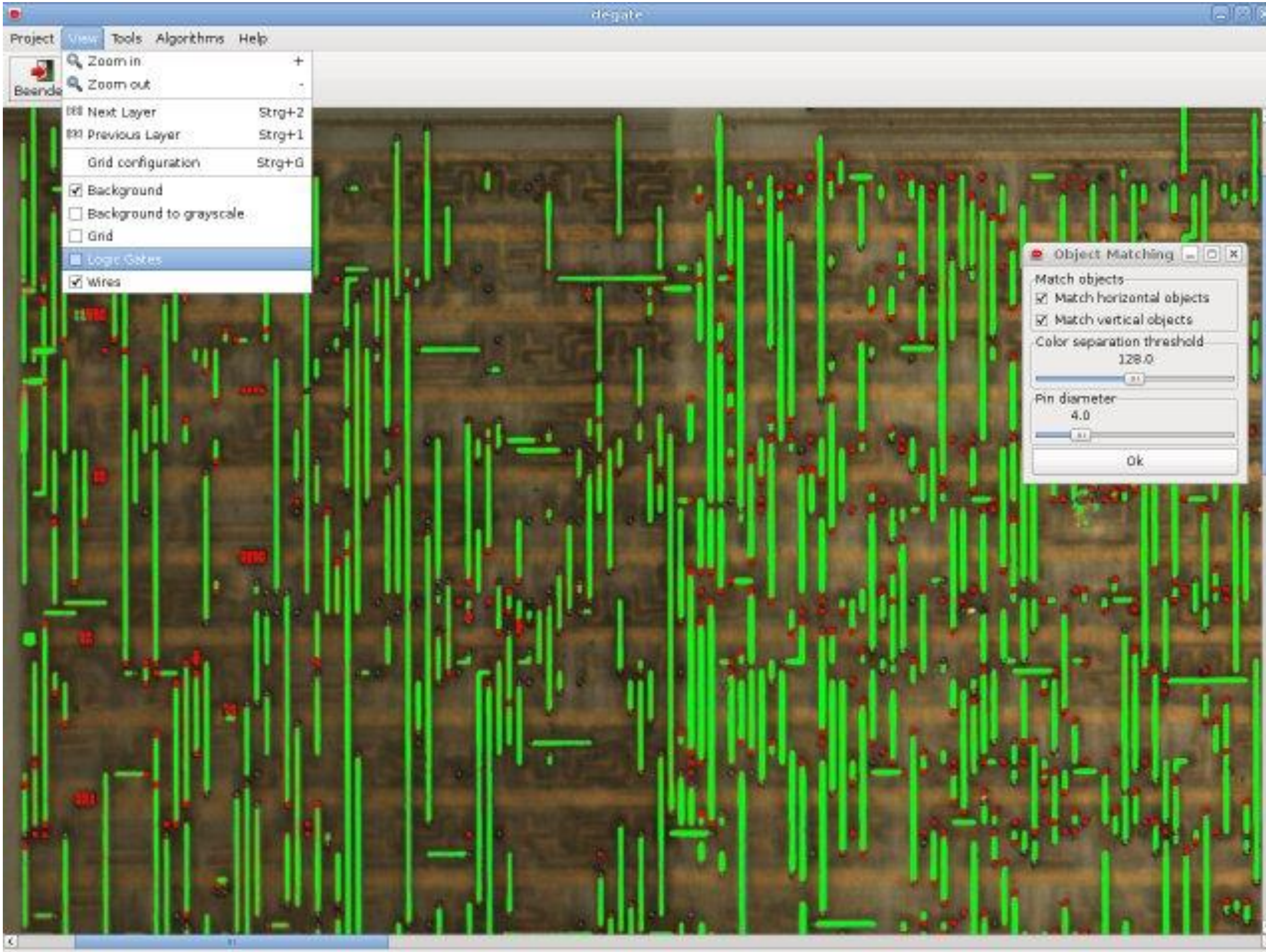
- Traced 1500 (!) connections manually
  - Tedious, time consuming
  - Error-prone (but errors easily spottable)
  - Tracing completely automated by now

Tracing Connections

Automated Tracing

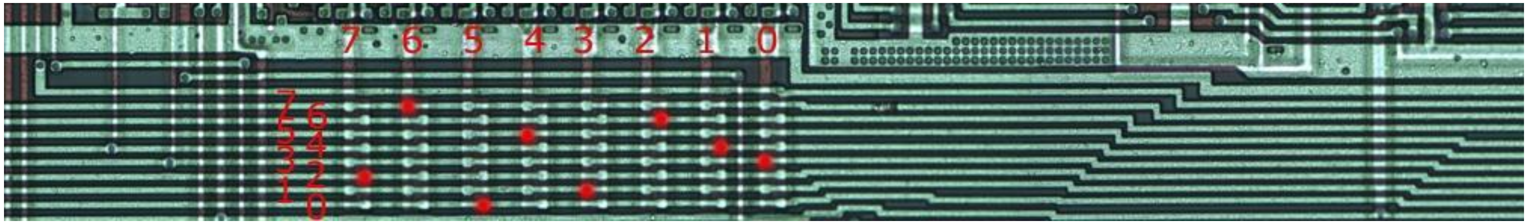





degate
Project
View
Tools
Algorithms
Help
Beende
Zoom in +
Zoom out -
Next Layer Strg+2
Previous Layer Strg+1
Grid configuration Strg+G
Background
Background to grayscale
Grid
Logic Gates
Wires
Object Matching
Match objects
Match horizontal objects
Match vertical objects
Color separation threshold
128.0
Pin diameter
4.0
Ok

Countermeasures

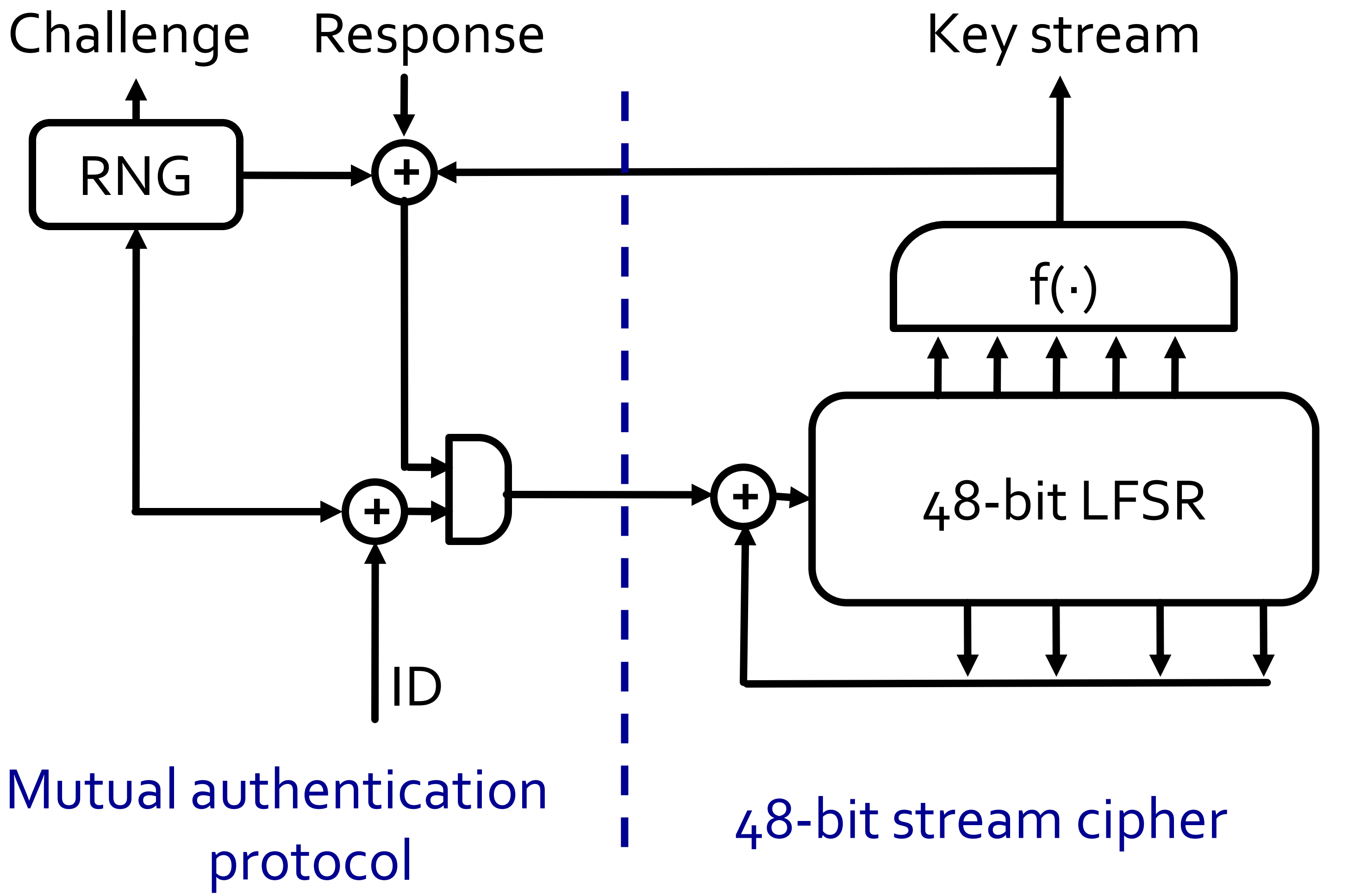
- Obfuscated placing and wiring of logic cells
  - May defeat human inspection, but not automated tools



Source: flylogic.net

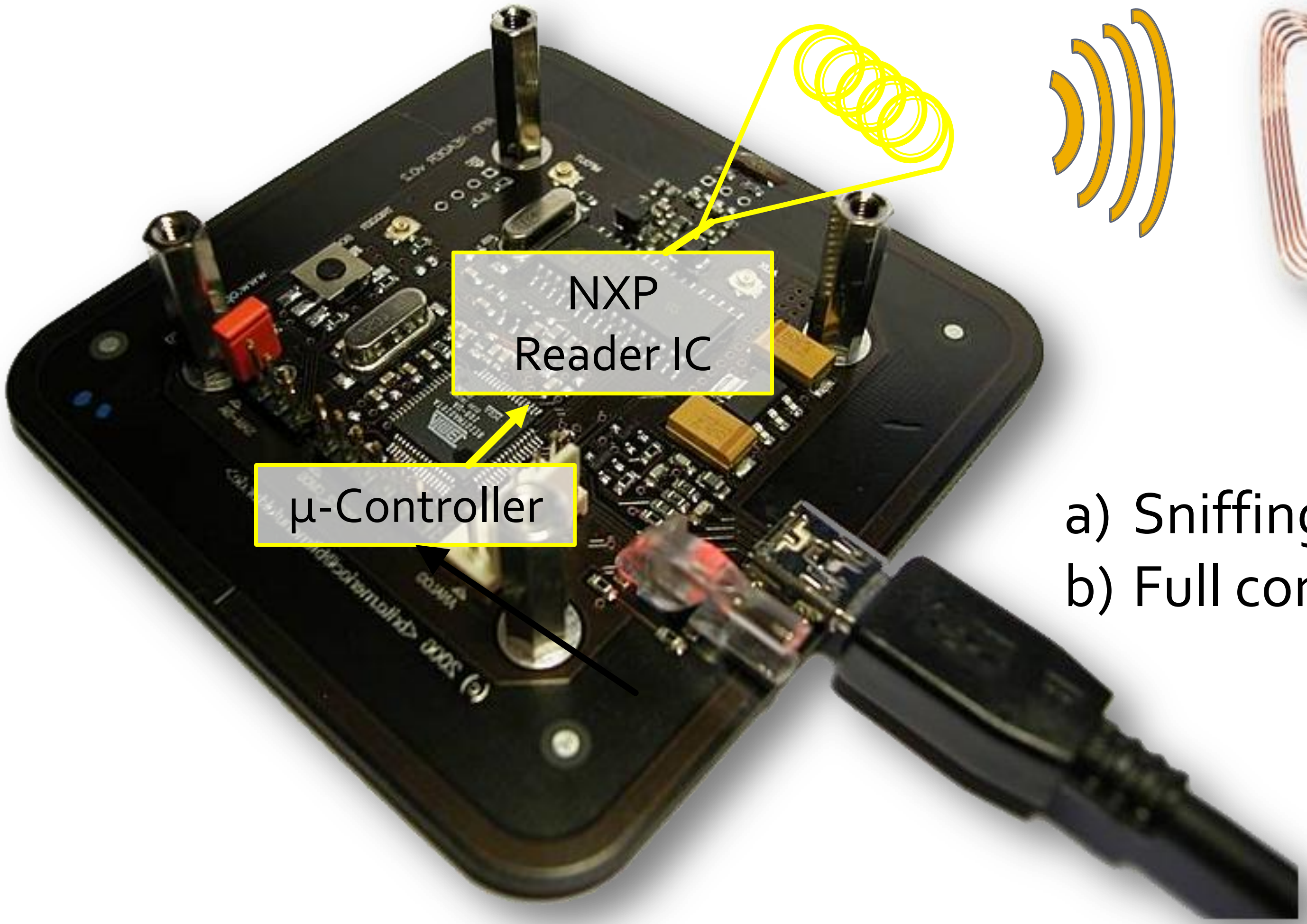
- Dummy cells
  - Makes reversing harder, but not impossible
- Large chips
  - Huge effort, huge rewards?
- Self-destructive chips?
  - May protect secret keys, not secret algorithms

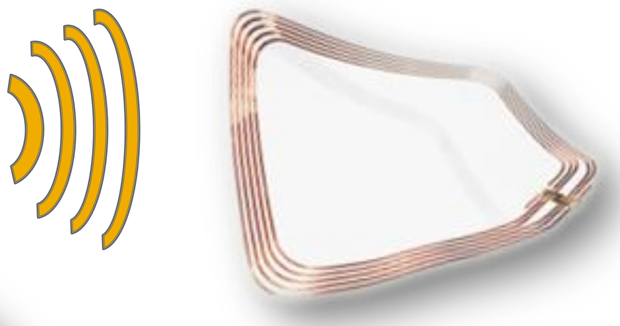
Mifare Crypto-1



Proprietary Encryption: Vulnerabilities

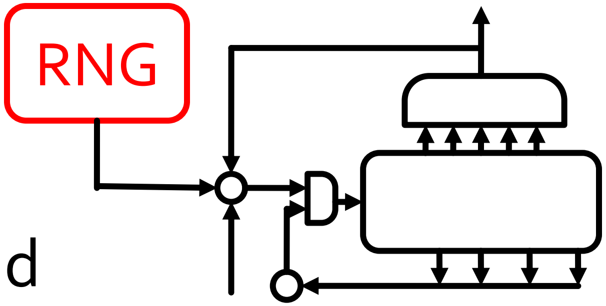
Hardware: OpenPCD (+PICC)





a) Sniffing data
b) Full control over timing!

Random Number Generator



- 16(!!)-bit random numbers
  - LFSR –based
  - Value determined by time of read

Our Attack:

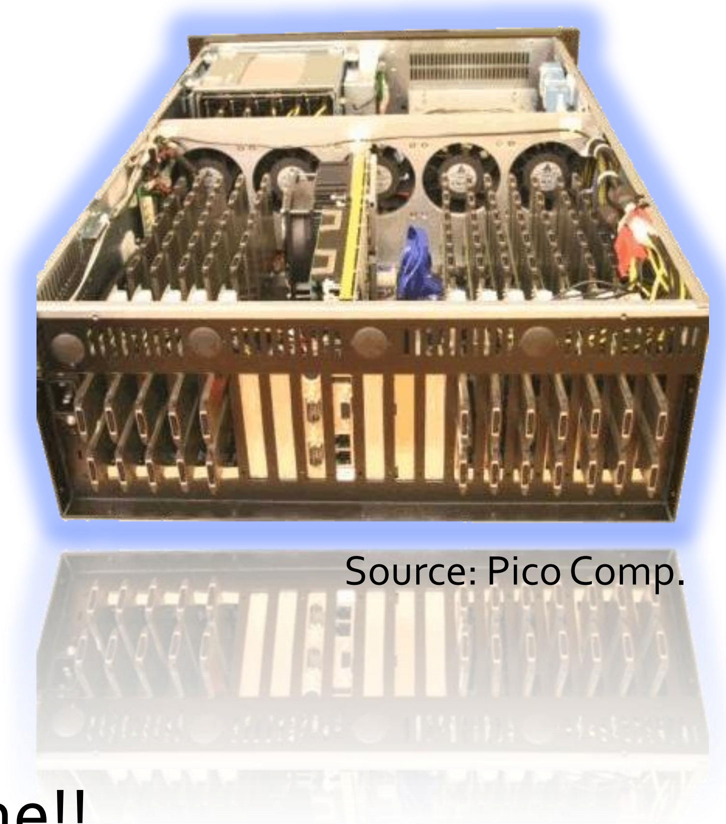
- Control timing (OpenPCD)

  = control random number (works for tag and reader!)

  = break Mifare security :)

For Starters: Brute-Force

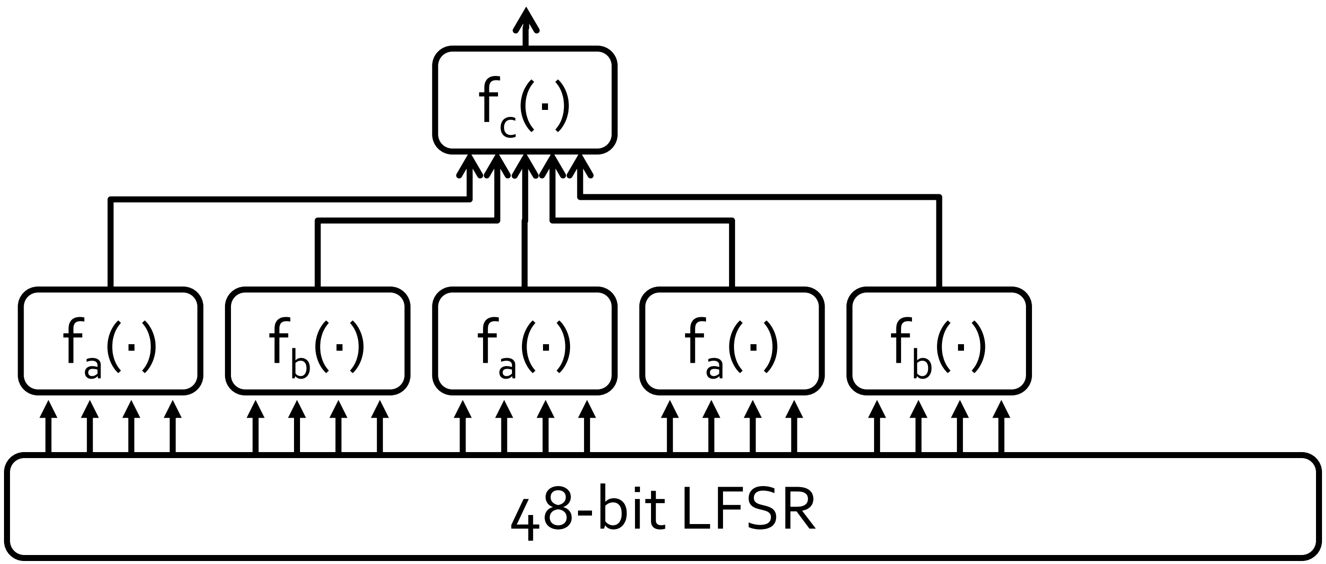
- Cipher complexity low
  - Was probably a primary design goal
  - Very efficient FPGA implementation



Source: Pico Comp.

FPGA cluster finds key
in 50 minutes!
30 sec. when trading space for time!!

Weak Filter + Protocol Flaw



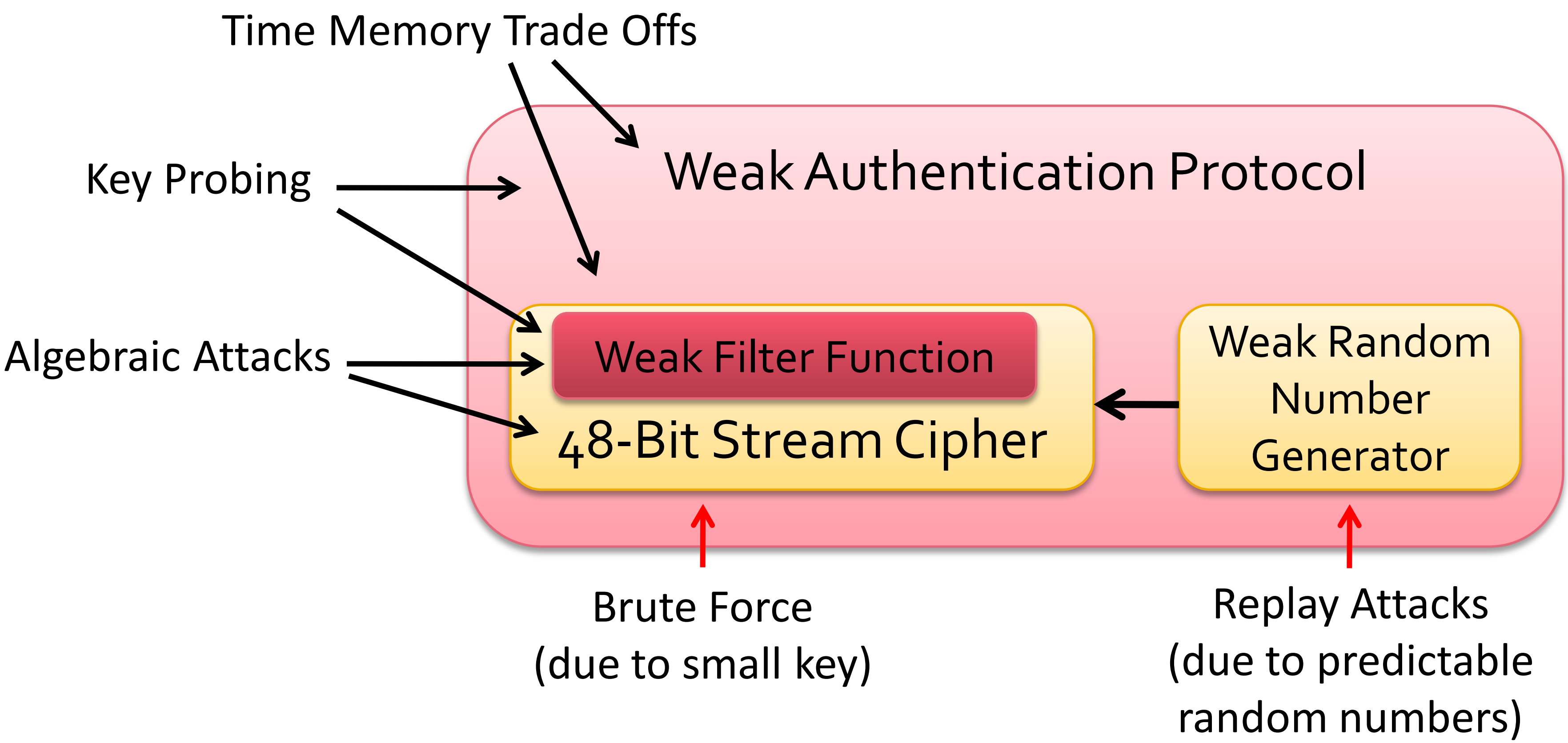
- Filter function is a network of smaller functions that are statistically biased
- Adversary controls inputs, can probe for internal state bits
- Attack takes < 1 minute on laptop

Algebraic Attacks

- Unpublished attacks that exploit simple feedback structure and statistical weaknesses
- Completely passive
- Works for strong random numbers
  - Hence, even against Crypto-1 on Mifare Plus!

- Attack takes 6 seconds on laptop
- Stay tuned for publication …

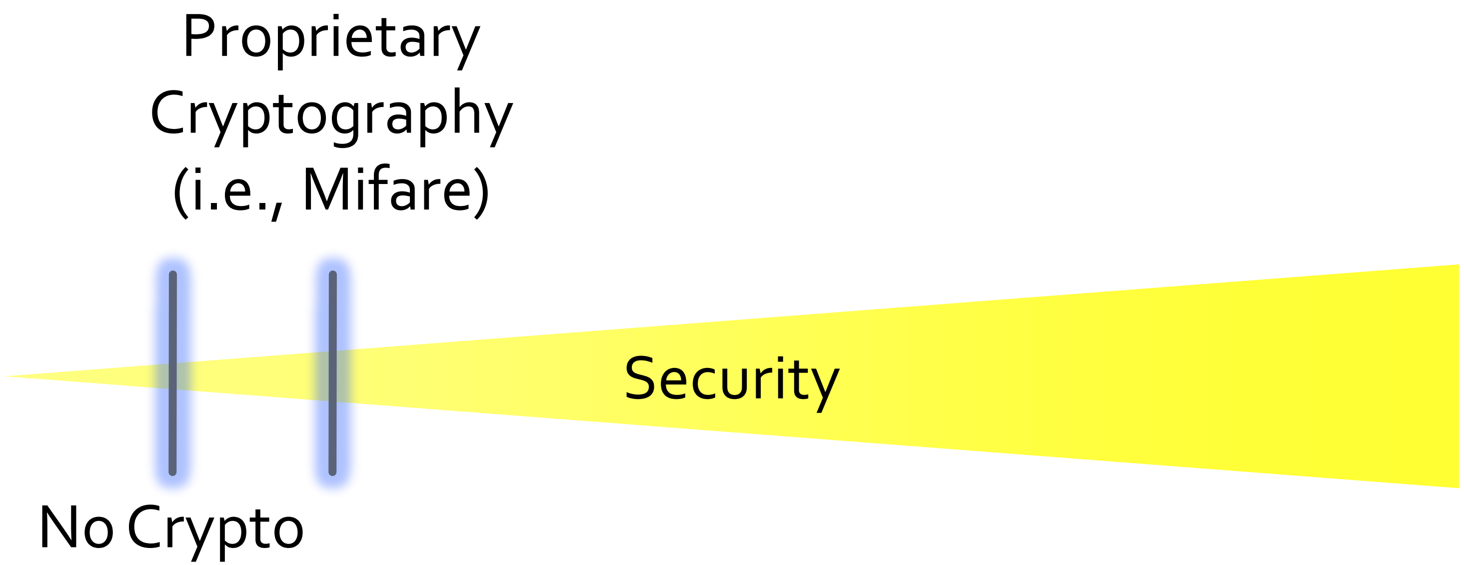
Mifare Classic Weaknesses



Attack Properties

	Runtime on FPGA Cluster ($100,000)	Runtime on PC	Works despite strong random numbers (Mifare Plus)	Hard to Detect
Replay Attacks	(0)	(0)	No	No
Brute Force	50 min	—	**Yes**	**Yes**
Time Memory Trade Offs (TMTO)	30 sec	—	No	Maybe
Key Probing	—	1 min	No	No
Algebraic Attacks	—	**6 sec**	**Yes**	**Yes**

Proprietary Insecurity

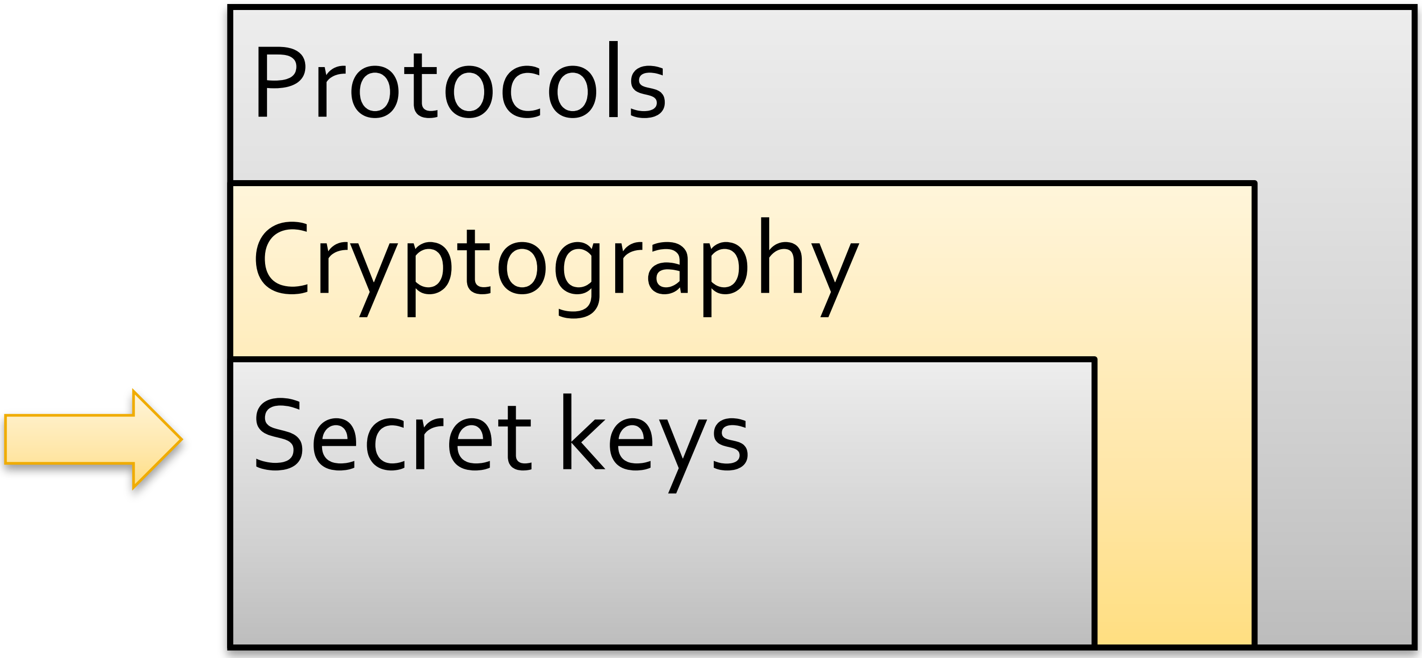


- Protection insufficient for almost all common uses of smart cards:
  - Access control, car theft protection, payment tokens, TV subscriptions, hardware dongles, …

Summary: Proprietary Encryption

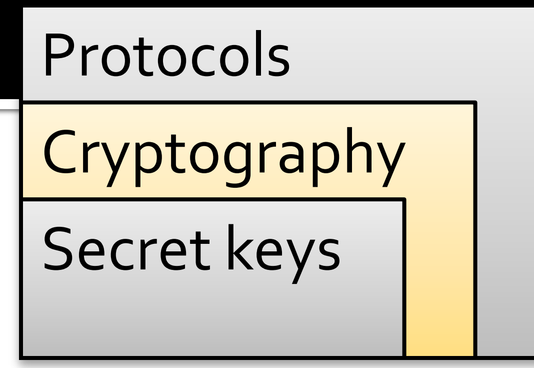
- Reverse-Engineering is possible (and fun)
  - You should try! (we will help)
  - Easy targets: small chips with proprietary crypto
  - Bigger rewards: bugs, backdoors, debug functions
- Obscurity adds security only in the short-run
  - Lack of peer-review hurts later
- Countermeasures mostly ineffective

Secure Key Storage



- More ubiquitous systems typically have more copies of the secret keys in accessible places
  - Will become weakest link in mobile applications

Key storage

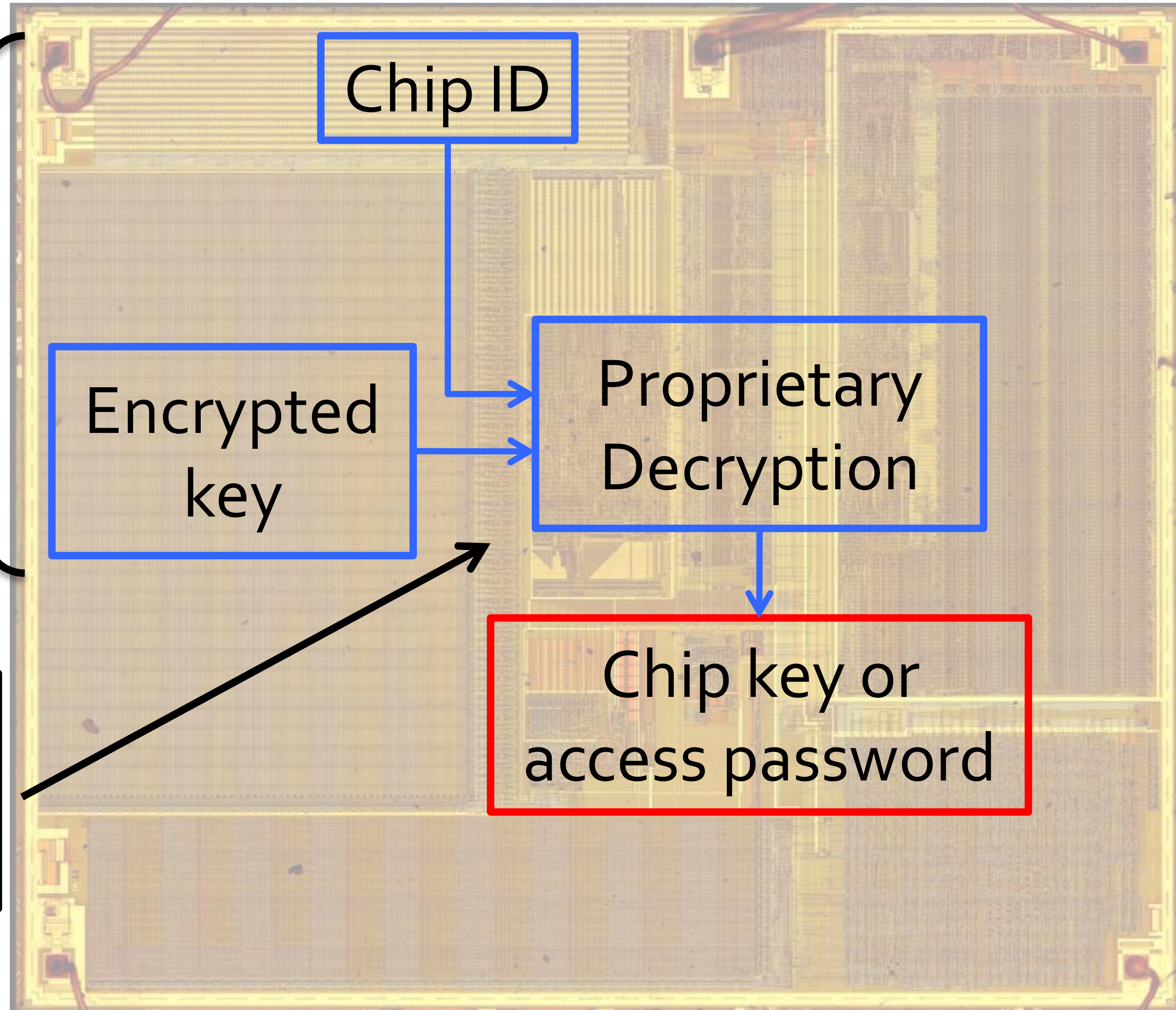


- Secret keys can be stored:
  - Online:
    - Keys only stored on central server
    - Expensive setup, long response times
  - Semi-online:
    - Devices receive keys at boot time
    - Keys often stored in DRAM at runtime; bad idea!
  - Offline:
    - Devices “securely” store key copy

Key Vault

Everything needed to disclose key is found on chip

Secret algorithms can be found; might be costly (>$100,000)



Key Diversity

- Secret keys should be
  - Different for every user
    - Requires many different keys
  - Immediately accessible
    - Requires small number of keys

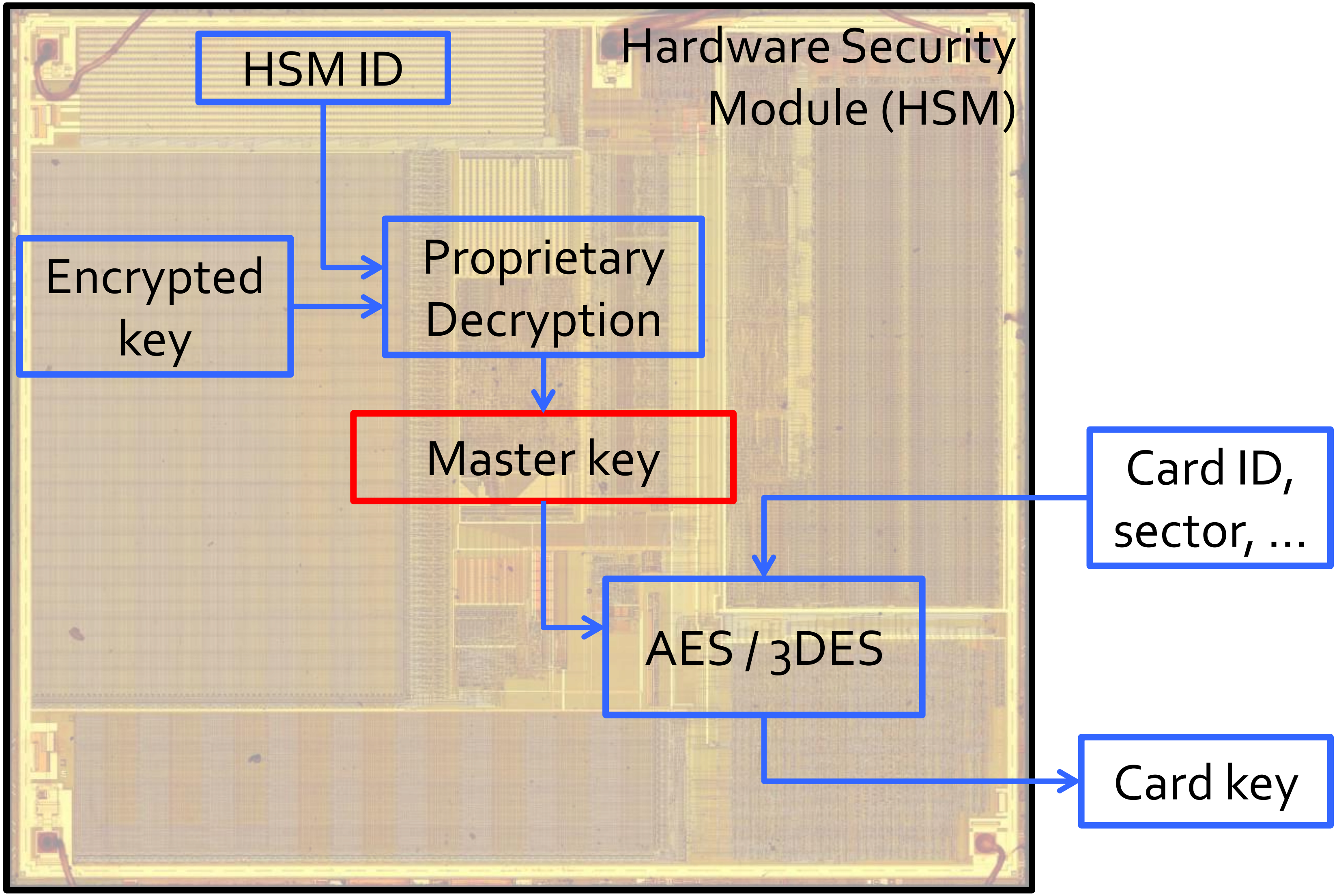
- Best practice: derive user keys from master key; store master key in „key vault“

Secret Key Storage

- Hardware Security Modules (HSM)
  - Used in ATMs (cash machine) and some smart card readers
  - Use proprietary encryption
  - Hence, can be broken
    - Probably high effort

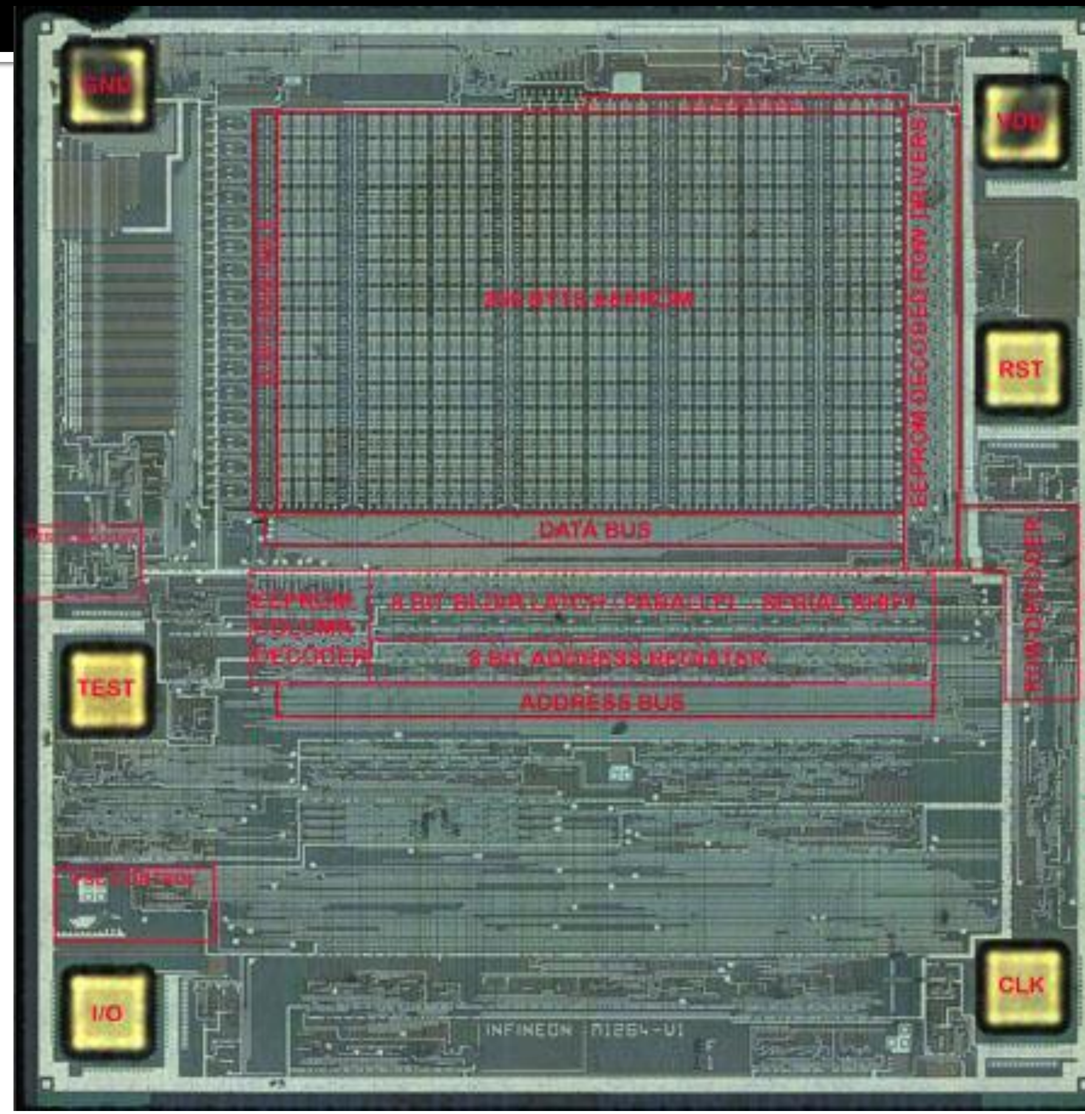
- Secure Access Modules (SAM) are much easier to break
  - Credit card / smart card readers

Key Diversification



SAM chips

- „Secure“ Access Modules are usually standard micro-processors
  - Low effort to extract master keys
  - SIMs/SAMs are becoming cheaper and less secure!
  - (cell phones are not any better)



Source: *Flylogic*

Summary: Key Storage

- Secret algorithms for key storage and diversification can be found
  - Same techniques that find proprietary encryption
  - Easy for most SAMs, high effort for HSMs
- Secret keys can be extracted from "secure" key storage
  - Online systems secure keys, but may necessitate their own access control
  - Best practice: always be prepared for key roll-over

Demonstration

Download Tools

- For the hands-on exercises after the demo, please download and install:

- Or, if you already have Matlab and Image Processing Library installed, get:

Hands-On Lab

Take Away

- Hardware security is hard
  - Cost constraints, legacy support
    - Leads to weak ciphers
  - Hardness of distributed trust
    - Online systems too slow and expensive
    - Offline systems may not protect secret key
- Prepare for failure
  - Layer security measures
  - Support key roll-over
  - Never rely on single manufacturer