Trojan-tolerant Hardware
+ Supply Chain Security in Practice

Vasilios Mavroudis
Doctoral Researcher, UCL

Dan Cvrcek
CEO, Enigma Bridge
Who we are

Vasilios Mavroudis
Doctoral Researcher, UCL

Dan Cvrcek
CEO, Enigma Bridge

George Danezis
Professor, UCL

Petr Svenda
Assistant Professor, MUni
CTO, Enigma Bridge
Highlights

- The private life of keys
- Weak links of the supply chain
- Lessons learned from airplanes
- Demo of our crypto hardware
- Protocols, Maths & Magic
- Politics, Distrust & Hardware Security
The Private Life of Keys

1. Someone designs an integrated circuit (IC)
2. IC is fabricated
3. IC is delivered to hardware vendor
4. Vendor loads firmware & assembles device
5. Device is sent to customer
6. Customer generates and stores key on the device
The Private Life of Keys

1. Someone designs an integrated circuit (IC)
2. IC is fabricated
3. IC is delivered to hardware vendor
4. Vendor loads firmware & assembles device
5. Device is sent to customer
6. Customer generates and stores key on the device

*Any attack in these steps can compromise the key!*
Hardware Security Modules

*Physical* computing device that safeguards and manages digital keys for strong authentication and provides cryptoprocessing.

Features:
- Cryptographic key generation, storage, management
- Tamper-evidence, Tamper-resistance, Tamper-response
- Security Validation & Certification

Crypto Operations are carried out in the device
No need to output the private keys!
Hardware Security Modules

Common Applications
- Public Key Infrastructures
- Payment Processing Systems
- SSL Connections
- DNSSEC
- Transparent Data Encryption

Cost
- Hardware (>10k)
- Integration Cost
- Operational/Support
HSM Guarantees

1. Someone designs an integrated circuit (IC)
2. IC is fabricated
3. IC is delivered to hardware vendor
4. Vendor loads firmware & assembles device
5. Device is sent to customer
6. Customer generates and stores key on the device
What could go wrong?

- **Bugs**
  
  CVE-2015-5464
  The HSM allows remote authenticated users to bypass intended key-export restrictions ...

- **Backdoors/HT?**
  
  THIS ‘DEMONICALLY CLEVER’ BACKDOOR HIDES IN A TINY SLICE OF A COMPUTER CHIP

  NSA's Own Hardware Backdoors May Still Be a “Problem from Hell”

  Expert Says NSA Have Backdoors Built Into Intel And AMD Processors

  Snowden: The NSA planted backdoors in Cisco products
Proposed Solutions

- Trusted Foundries
  - Very expensive
  - Prone to errors/bugs

- Split-Manufacturing
  - Still Expensive
  - Again prone to errors/bug

- Post-fabrication Inspection
  - Expensive (+ re-tooling)
  - A huge pain, doesn’t scale
Proposed Solutions

- Trusted Foundries
  - Very expensive
  - Prone to errors/bugs

- Split-Manufacturing
  - Still Expensive
  - Prone to errors/bugs

- Post-fabrication Inspection
  - Expensive (+ re-tooling)
  - A huge pain, doesn’t scale

Arms Race
  - Adversaries always one step forward
  - Can never be 100% certain
A solution from the sky (not the cloud)

**Lockstep systems** are fault-tolerant computer systems that run the same set of operations at the same time in parallel.

- **Dual redundancy** allows error detection and error correction
- **Triple redundancy**
  automatic error correction, via majority vote
  → Triple Redundant 777 Primary Flight Computer
Fault-tolerant systems are built for safety and the computations are simply replicated.

Not enough for security!
Not for Security

Fault-tolerant systems are bad for security:
- The private key is generated/stored in each IC
- Device is as secure as its weakest link
- Increase the attack surface
Our Solution

1. Someone designs an integrated circuit (IC)
2. IC is fabricated
3. IC is delivered to hardware vendor
4. Vendor loads firmware & assembles device
5. Device is sent to customer
6. Customer generates and stores key on the device
Ingredients of the Solution

1. Hardware Components (IC)
   - Independent Fabrication
   - Non-overlapping Supply Chains
   - Programmable
   - Affordable
   - Bonus if COTS

2. Cryptographic Protocols
   - No single trusted party
   - Full Distribution of Secrets
   - Distributed Processing
   - Provably Secure (i.e., Math)
Smart Cards

Many Independent Manufacturers

- Private Fabrication Facilities
- Disjoint Supply Chains (location, factories, design)

Programmable Secure Execution Environment

- NIST FIPS140-2 standard, Level 4
- Common Criteria EAL4+/5+

Off-the-shelf Cost $1-$20
Multiparty Computation Protocols

Distributed Operations
- Random number Generation
- Key Pair Generation
- Decryption
- Signing

Provably Protect against
- All-1 Malicious & Colluding parties
- All Malicious & non-colluding parties
THE

PROTOTYPE
Many Smart Cards

Components

- **120** SmartCards
  - **40 Groups** of 3 Cards
  - **1.2Mbps** dedicated inter-IC buses
- FPGA manages the communication bus
  - **1Gbit/s** bandwidth for requests
Custom boards with 120 JCs
JavaCards
- FIPS140-2 Level 3
- CC EAL5+
Gigabit link to untrusted Linux server
Geographically Distributed IC Control
Giving smart-cards an infrastructure

mpc.enigmabridge.com

LAS VEGAS (192.168.42.10)

MPC RESTful server

CAMBRIDGE UK (84.92.209.143)

MacBook-2
Giving smart-cards an infrastructure

mpc.enigmabridge.com

MPC RESTful server

LAS VEGAS (192.168.42.10)

CAMBRIDGE UK (84.92.209.143)

MacBook-2

jCardSim

- ARM
- Intel
- SPARC
Key Generation

Normal Operation
Giving smart-cards an infrastructure

mpc.enigmabridge.com

MPC RESTful server

LAS VEGAS

MacBook-2
Key Generation

Attack Mode
Visualizing Cryptography

mpc.enigmabridge.com

Node-red
- HTTP requests (switch evil)
- MPC key generation
- web-socket servers

MPC RESTful server

ICs with Hardware Trojans

MacBook-2
Performance
Scalability
PROTOCOLS
Key Points

- No single IC is trusted with a secret (e.g., private key)
- Misbehaving ICs can be detected by honest ones
- If one IC is excluded from any protocol, user can tell

Bonus: Minimize interaction between ICs for performance
Sharing a Secret

- Split a secret in *shares*
- The secret can be *reconstructed* later
- Without *sufficient* shares not a single bit is leaked
- Splitting Parameters:
  - How many shares the secret is split into (n)
  - How many shares you need to reconstruct the secret (t)

*In our case: Each 3 ICs hold shares for a secret*
Classic Key Generation

Single IC System
1. Bob asks for **new key pair**
2. Backdoored IC generates compromised key
3. Private Key is “securely” stored
4. **Weak** Public key is returned

Problems
- Malicious IC has full access to the private key
- Bob can’t tell if he got a “bad” key
Distributed Key Generation

1. ICs holding key shares
2. Public Keys
3. +
4. *THE* Public Key
Distributed Key Generation

Key Points

- No single IC is trusted with a secret (e.g., private key) ✔
- Misbehaving ICs can be detected by honest ones ✔
- If one IC is excluded from any protocol, user can tell ✔

Bonus: Minimize interaction between ICs for performance ✗
Classic Decryption

Single IC System

1. Bob asks for ciphertext decryption
2. Backdoored IC decrypts ciphertext
3. Bob retrieves plaintext

The IC needs full access to the private key to be able to decrypt ciphertexts.
Distributed Decryption

Decryption Shares
ICs holding key shares

Help me decrypt this email
Distributed Decryption

Help me decrypt this email

Decryption Shares
ICs holding key shares
Distributed Decryption

Help me decrypt this email

Decryption Shares
ICs holding key shares
Distributed Decryption

ICs holding key shares

Decryption Shares

1 2 3 4
Distributed Decryption

Key Points

- No single IC is trusted with a secret (e.g., private key) ✔
- Misbehaving ICs can be detected by honest ones -
- If one IC is excluded from any protocol, user can tell ✔

Bonus: Minimize interaction between ICs for performance ✔
Classic Signing

Single IC System

1. Bob asks for **document signing**
2. Backdoored IC signs the plaintext
3. Bob retrieves signature

*The IC needs full access to the private key to be able to sign plaintexts.*
Distributed Signing

ICs holding key shares
Distributed Signing

Signature Shares
ICs holding key shares

Help me sign this document
Distributed Signing

ICs holding key shares

Signature Shares

1

2

3

4
Distributed Signing

Key Points

- No single IC is trusted with a secret (e.g., private key) ✔
- Misbehaving ICs can be detected by honest ones ✔
- If one IC is excluded from any protocol, user can tell ✔

Bonus: Minimize interaction between ICs for performance ✔
How we made it scale
How we made it scale
How we made it scale
How we made it scale

Key A
Key B
Key C
Key D
How we made it scale
How we made it scale

But how can all these groups have shares for the same key?
1. Group A generates a public key
2. A1, A2, A3 send their shares to B1, B2, B3
3. Each IC in B receives shares from A1, A2, A3
4. Each IC in B combines the 3 shares and retrieves its private key
Key Replication

1. Group A generates a public key
2. A1, A2, A3 send their shares to B1, B2, B3
3. Each IC in B receives shares from A1, A2, A3
4. Each IC in B combines the 3 shares and retrieves its private key
5. A1, A3 and B2 collude

The adversary retrieves the secret!
1. Group A generates a public key
2. Then each IC in A splits its private key in three shares and sends them to B1, B2, B3
3. Each IC in B receives shares from A1, A2, A3
4. Each IC in B combines the 3 shares and retrieves its private key share

The full public keys of A and B are the same!
GEOPOLITICS
“We can guarantee security if there is at least one honest IC that is not backdoored or faulty.”
"We can guarantee security if there is at least one honest IC that is not backdoored or faulty."

What if all ICs are malicious?
Government-level adversaries

- Deep access to fabrication facilities
- Very sophisticated techniques
- Very hard to detect their Backdoors/Trojans
- Very secretive; highly classified
- Won’t share their backdoor details
Government-level adversaries

- Deep access to fabrication facilities
- Very sophisticated techniques
- Very hard to detect their Backdoors/Trojans
- Very secretive; highly classified
- Won’t share their backdoor details
- Unlikely to collude with anyone
“We can guarantee security even when all ICs are malicious, if at least one does not collude.”
Conclusions & Future

**New crypto hardware architecture**

- For the first time, tolerates faulty & malicious hw
- Decent Performance
- Scales nicely; just keep adding ICs
- Suitable for commercial-off-the-shelf components
- Existing malicious insertion countermeasures are very welcome!
**DIY**

**Poor man’s HSM**

1. Buy a USB hub
2. 3-4 card readers (or more)
3. Buy cards from various manufacturers
4. Download our MPC applet
5. Review the code
6. Install the applet into your cards
7. Enjoy your homemade HSM!
Q & A
Trojan-tolerant Hardware

+ Supply Chain Security in Practice

Vasilios Mavroudis
Doctoral Researcher, UCL

Dan Cvrcek
CEO, Enigma Bridge
Trojan-tolerant Hardware + Supply Chain Security in Practice

Vasilios Mavroudis
Doctoral Researcher, UCL

Dan Cvrcek
CEO, Enigma Bridge
Smart Cards

- 8-32 bit processor @ 30MHz+
- Persistent memory 32-500kB (EEPROM)
- Volatile fast RAM, usually <10kB
- True Random Number Generator (FIPS140-2)
- Cryptographic Coprocessor (3DES,ECC,AES,RSA-2048,...)
- Limited attack surface
  - Clear API
  - small trusted computing base
Plugging it into a cloud service

- Registration proxy
- Wrapper (e.g. PKCS11)
- Language binding
- Monitoring dashboards

- External API (JSON)
- Security domain Manager (virtualisation)

- Controller
  - FIPS140-2 L3 hardware
- Controller
  - FIPS140-2 L3 hardware
- Controller
  - FIPS140-2 L3 hardware
The Birth of a Distributed Key

1. User asks for **new key pair**
2. ICs generate their key pairs
3. ICs exchange hashes of their shares
4. ICs reveal their shares
5. ICs **verify** each others’ shares
6. ICs compute the **common public key**
7. ICs return the common public keys
8. Bob **verifies** that all the keys are **same**
1. Bob asks for ciphertext decryption
2. His authorization is verified
3. ICs compute their decryption shares
4. Bob receives the decryption shares
5. Bob combines them to decrypt
Distributed Decryption

Properties

- No single authority gains access to the full private key
- ICs check on each other
- If one IC abstains, decryption fails
Distributed Signing I

Caching
1. Bob sends a caching request
2. The ICs verify Bob’s authorization
3. Generate a random group element based on j
4. Bob sums the random elements

Properties
- Caching for thousands of rounds (j)
- Bob stores $R_j$
Distributed Signing II

**Signing**
1. Bob asks for document signing & sends $R_j, \sigma,$, and the hash of $m$
2. ICs verify his authorization
3. ICs check if $j$ has been used again
4. ICs compute their signature share
5. Bob sums all signature shares

**Properties**
- All ICs must participate
- Significant speed up with caching
That same basic scenario is cropping up more frequently lately, and not just in the Middle East, where conspiracy theories abound. According to a U.S. defense contractor who spoke on condition of anonymity, a “European chip maker” recently built into its microprocessors a kill switch that could be accessed remotely. French defense contractors have used the chips in military equipment, the contractor told IEEE Spectrum. If in the future the equipment fell into hostile hands, “the French wanted a way to disable that circuit,” he said. Spectrum could not confirm this account independently, but spirited discussion about it among researchers and another defense contractor last summer at a military research conference reveals a lot about the fever dreams plaguing the U.S. Department of Defense (DOD).
The Pentagon is worried that “backdoors” in computer processors might leave the American military vulnerable to an instant electronic shut-down. Those fears only grew, after an Israeli strike on an alleged nuclear facility in Syria. Many speculated that Syrian air defenses had been sabotaged by chips with a built-in ‘kill switch’ – commercial off-the-shelf microprocessors in the Syrian radar might have been purposely fabricated with a hidden “backdoor” inside. By sending a preprogrammed code to those chips, an unknown antagonist had disrupted the chips’ function and temporarily blocked the radar.”
Redundancy & Availability

A2

B1

A1

B2

A3

B3