Shopshifting

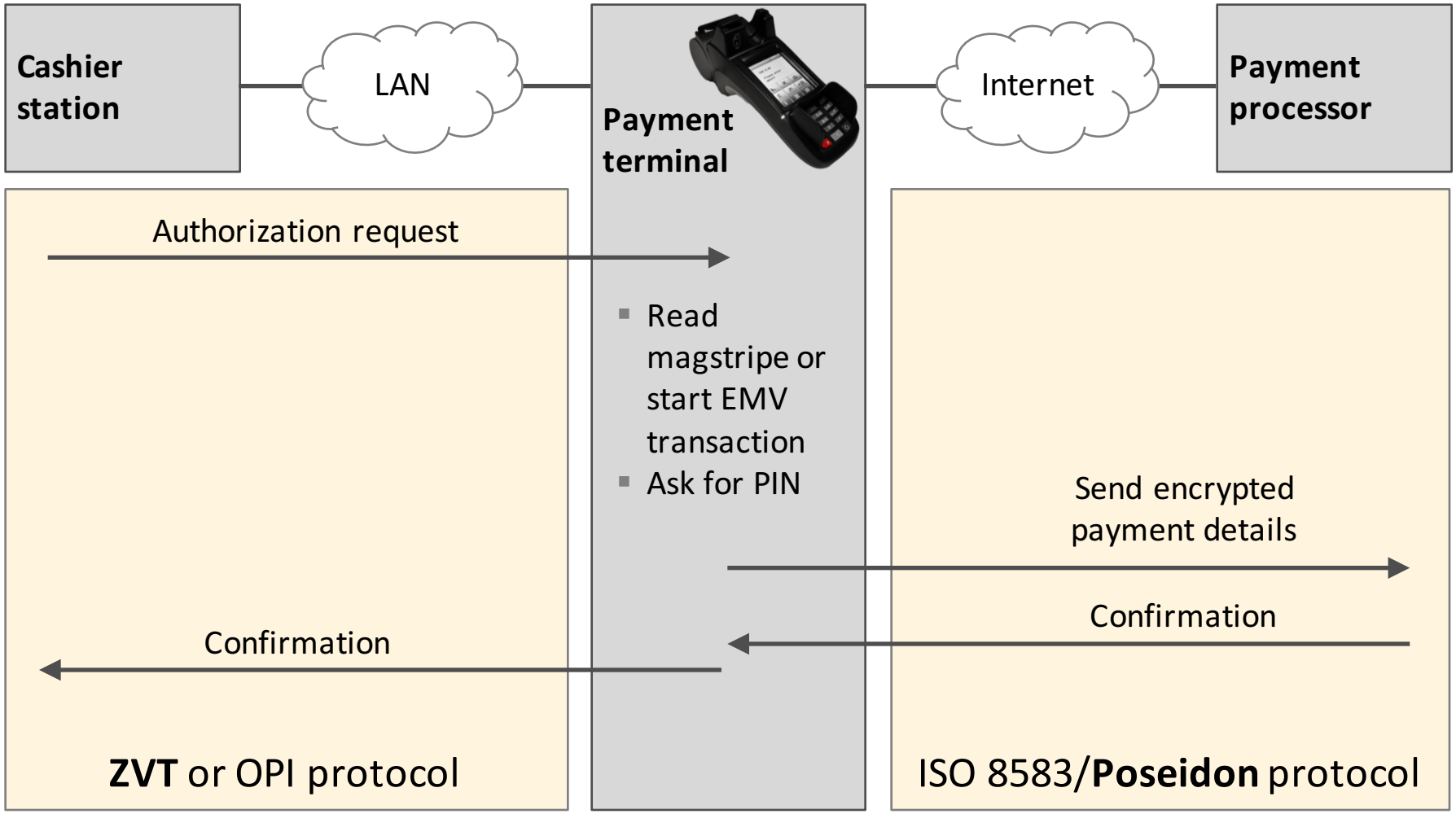
Warning about potential payment system abuse

Fabian Bräunlein <fabian@srlabs.de>
Philipp Maier <dexter@srlabs.de>
Karsten Nohl <nohl@srlabs.de>

SECURITY
RESEARCH
LABS

Card-based payment relies on two protocols

This talk investigates the security of the protocols used to make cashless payment happen



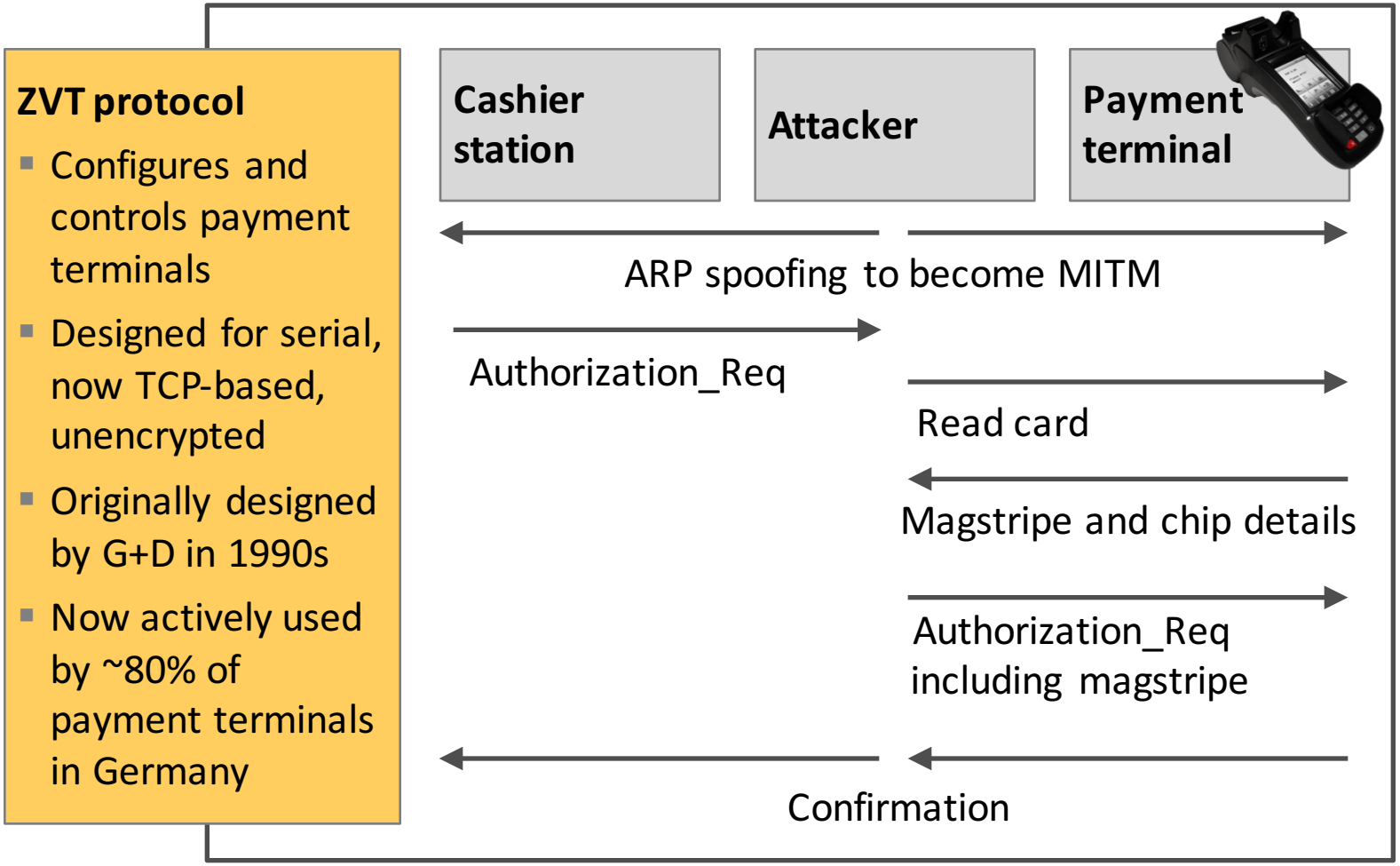
Agenda

- **Local payment abuse**
- Poseidon shopshifting
- Evolution need

ZVT allows unauthenticated access to magstripe data

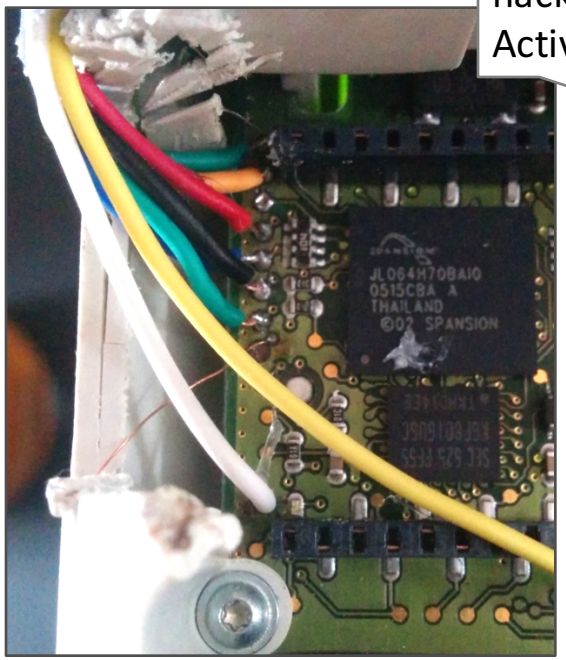
ZVT protocol

- Configures and controls payment terminals
- Designed for serial, now TCP-based, unencrypted
- Originally designed by G+D in 1990s
- Now actively used by ~80% of payment terminals in Germany



Access to PIN requires cryptographic MAC

HSM leaks MAC through timing side channel



Main CPU is easily hackable:
Active JTAG, RCE, …

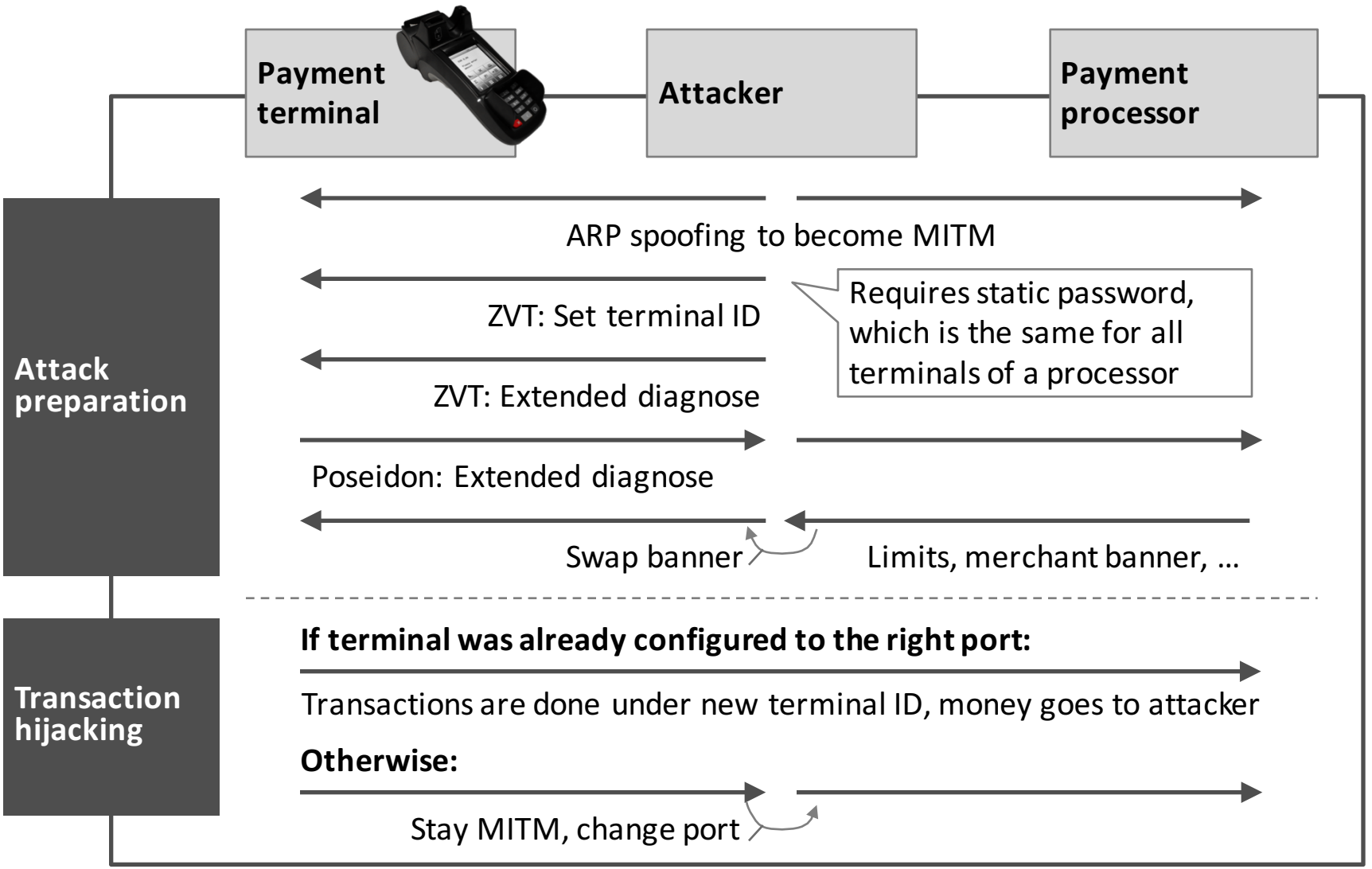
HSM protects secrets and should be much better secured

Main CPU sends MAC	HSM CPU compares MAC	
MAC	**Response**	**Response time**
00 …	Fail	26.000
01 …	Fail	26.000
02 …	Fail	26.000
03 …	Fail	26.005
0301 …	Fail	26.005
0302 …	Fail	26.010
…		
0302AF … 05	Ok	26.040

- MAC comparison is done byte-by-byte
- Response time leaks complete MAC within minutes
- MAC is not terminal-specific: Works across many different terminals

Demo 1
Mag stripe and PIN theft via ZVT over LAN

ZVT also allows local terminal hijacking



Demo 2

Redirect merchant transactions to attacker account via ZVT

Small complication: Attackers need their own merchant accounts

Agenda

- Local payment abuse
- **Poseidon shopshifting**
- Evolution need

Poseidon's authentication model is simply wrong

Poseidon protocol

- Dialect of global payment standard *ISO 8583*
- De-facto standard in Germany
- Strong monoculture: Only one backend implementation, used by all processors
- Apparently also used in France, Lux, and Iceland

Payment terminal | **Payment processor**

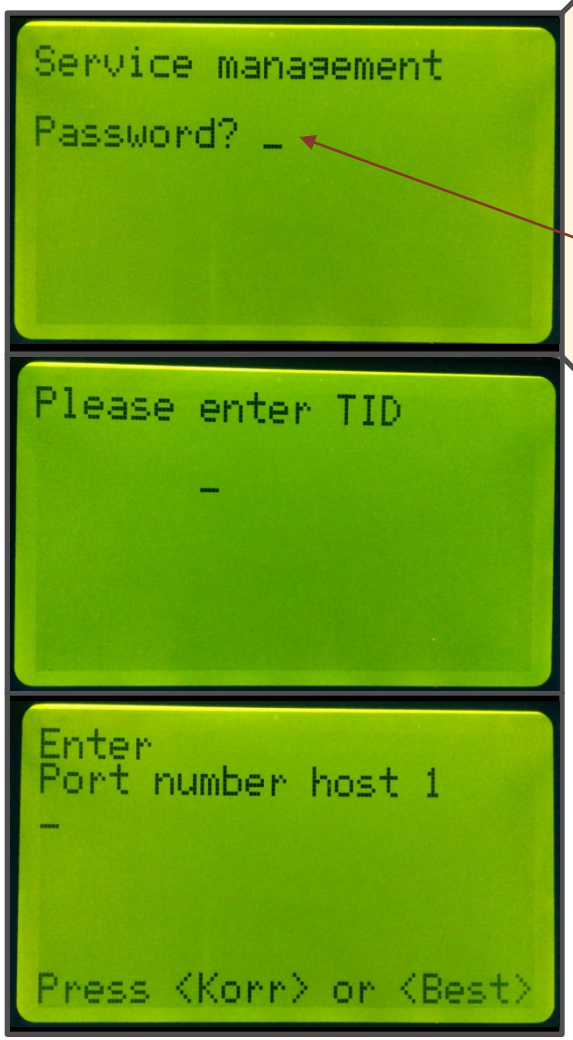
Poseidon initialization:

Terminal ID →

← Terminal configuration, encrypted with terminal key

Poseidon authentication uses pre-shared keys, similar to VPNs. However, the **key is the same for many terminals**!
This cannot be secure.

Few parameters are needed for Poseidon initialization



1. Google

Versionen

Für den Betrieb an der Tankstelle ist zumindest die Version 55.03 der Terminalsoftware nötig

TaskSTAR POS benötigt mindestens V3.05.00 SP2 Hotfix TA 7.0.

Kennwörter

Kassierer: 000000

Servicetechniker: 210888

IP-Adressierung

Sofern durch den DSL-Anschluss oder das vorhandene Hausnetz nichts anderes erzwungen werden feste IP-Adressen mit Subnetzmaske 255.255.255.000 verwendet. Die Term bekommen dann folgende Adressen:

Terminal 1: 192.168.001.101,

Terminal 2: 192.168.001.102,

Terminal 3: 192.168.001.103,

usw.

DHCP: in der Regel NEIN.

Die IP-Adressen für DSL-Zugang **TeleCash-Zahlungshost** lauten 217.073.032.104 217.073.032.105. Die Portnummer ergibt sich bei Hypercom-Geräten aus folgender Formel:

51500 + PU-Nummer.

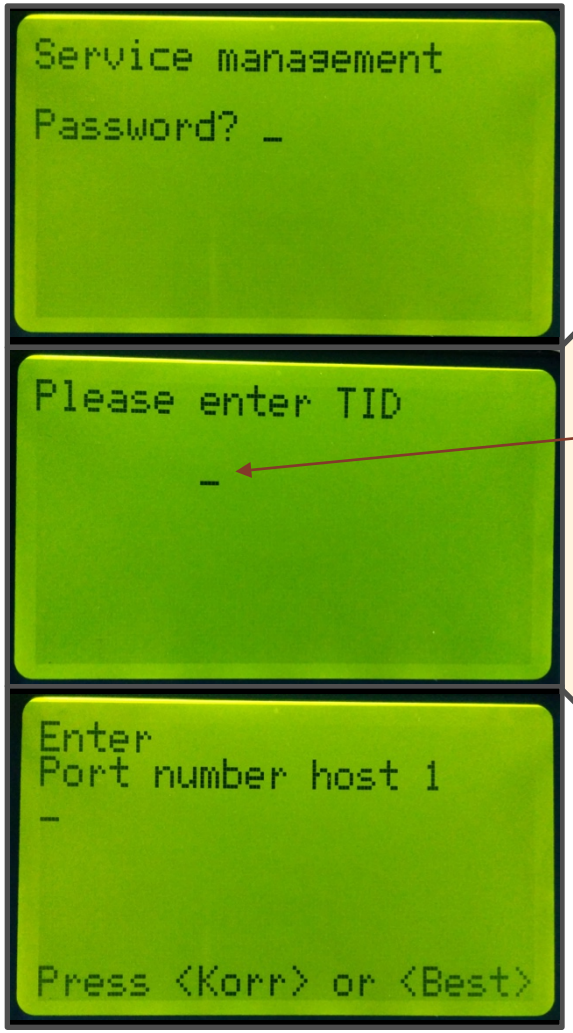
Beispielsweise lautet die Port-Nummer der PU 16 : 51516. Bitte beide Hosts eintragen, beim Ausfall eines der Hosts der andere erreicht werden kann.

Einen DSL-Zugang zum Testhost (PU 99) gibt es Stand 3.11.2009 nicht.

Das Terminal fragt bei der Inbetriebnahme ob ein **IP-Längenbyte** verwendet werden soll

Or brute-force over ZVT, or read through JTAG, …

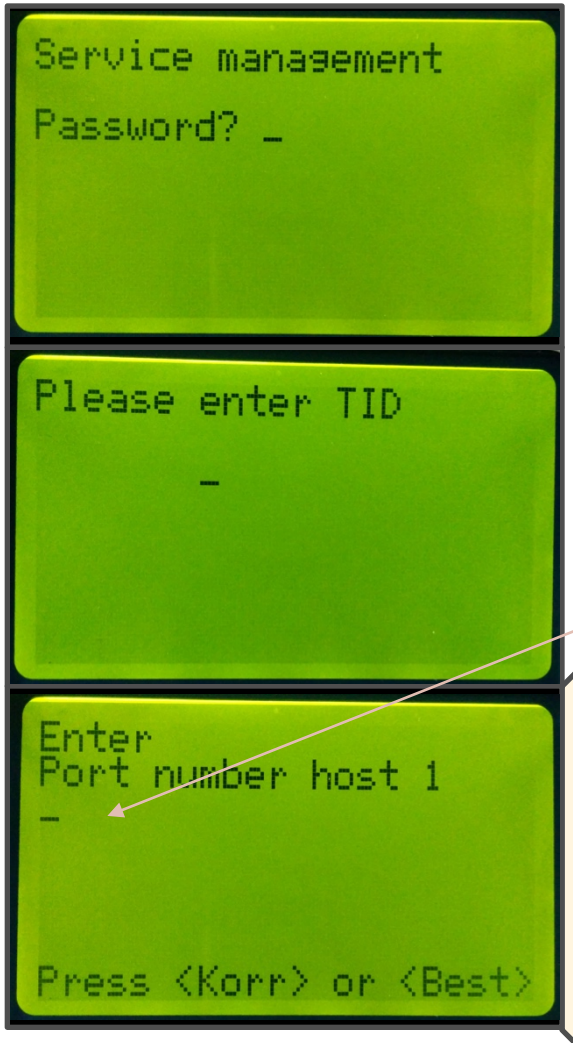
Few parameters are needed for Poseidon initialization



2. Go shopping

Or simply guess TIDs: They are assigned incrementally.

Few parameters are needed for Poseidon initialization



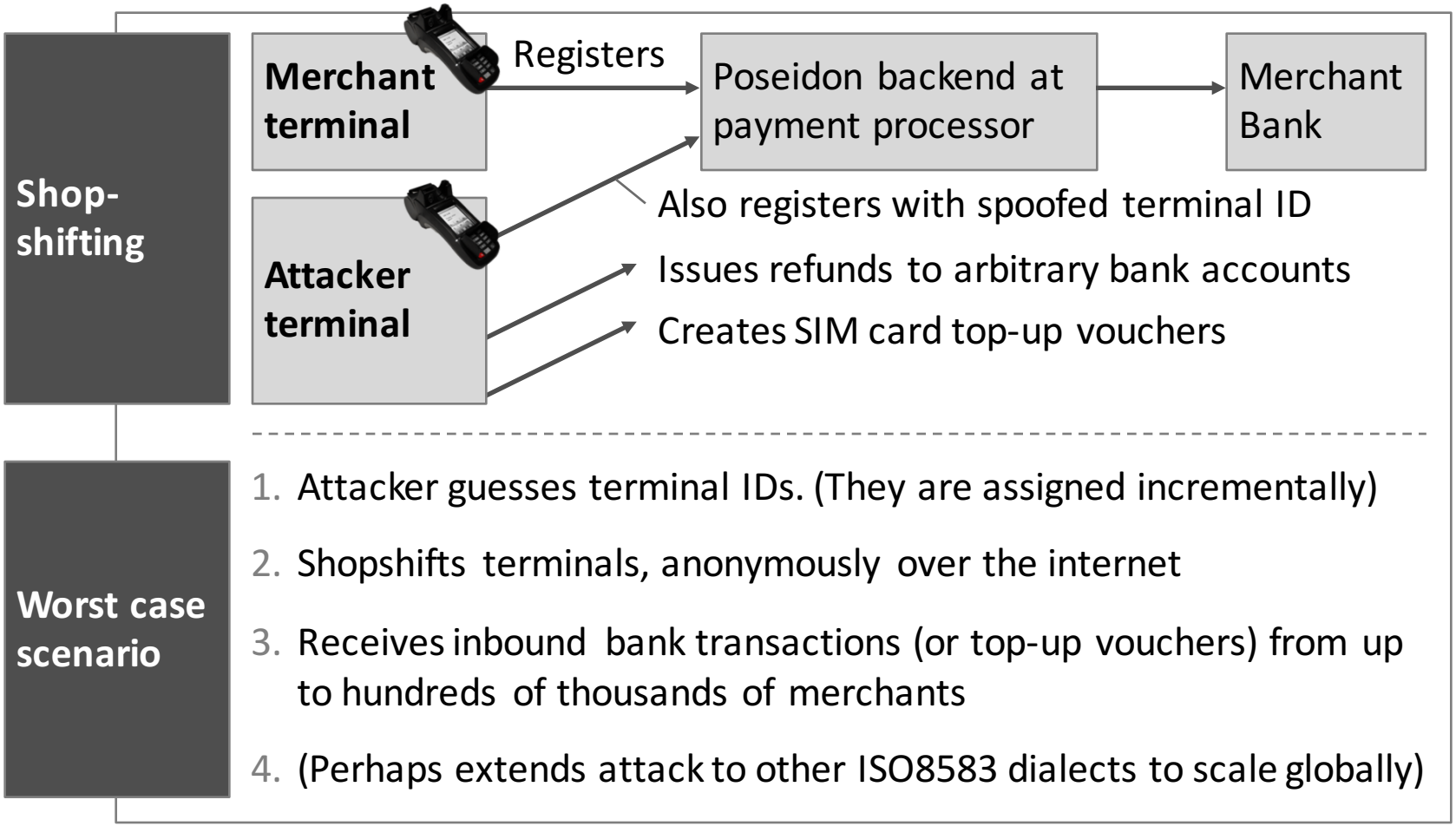
3. Brute force TCP port

```
[+] Received Data for port 55221:
0810203800000280800090782500002614483512188654368149097 8c5ca
[+] Received Data for port 55222:
0810203800000280800090782500002614483512188654368149097 8c5ca
[+] Received Data for port 55223:
0810203800000280800090782500002614483512188654368149097 8c5ca
[+] Received Data for port 55225:
0810203800000280800090782500002614483512188654368149097 8c5ca
[+] Received Data for port 55228:
0810203800000280800090782500002614483512188654368149097 8c5ca
[+] Received Data for port 55229:
0810203800000280801090782500003214483512180054368149097 8f0f8f10a404f6e6e6
0000000002d0000ff50d0
[+] Received Data for port 55232:
0810203800000280800090782500002614483512188654368149097 8c5ca
[+] Received Data for port 55238:
0810203800000280800090782500002614483512188654368149097 8c5ca
[+] Received Data for port 55315:
0810203800000280800090782500002614483512188654368149097 8c5ca
[+] Received Data for port 55316:
0810203800000280800090782500002614483512188654368149097 8c5ca
[+] Received Data for port 55317:
0810203800000280800090782500002614483512188654368149097 8c5ca
[+] Received Data for port 55318:
0810203800000280800090782500002614483512188654368149097 8c5ca
[+] Received Data for port 55321:
0810203800000280800090782500002614483512188654368149097 8c5ca
[+] Received Data for port 55322:
0810203800000280800090782500002614483512188654368149097 8c5ca
```

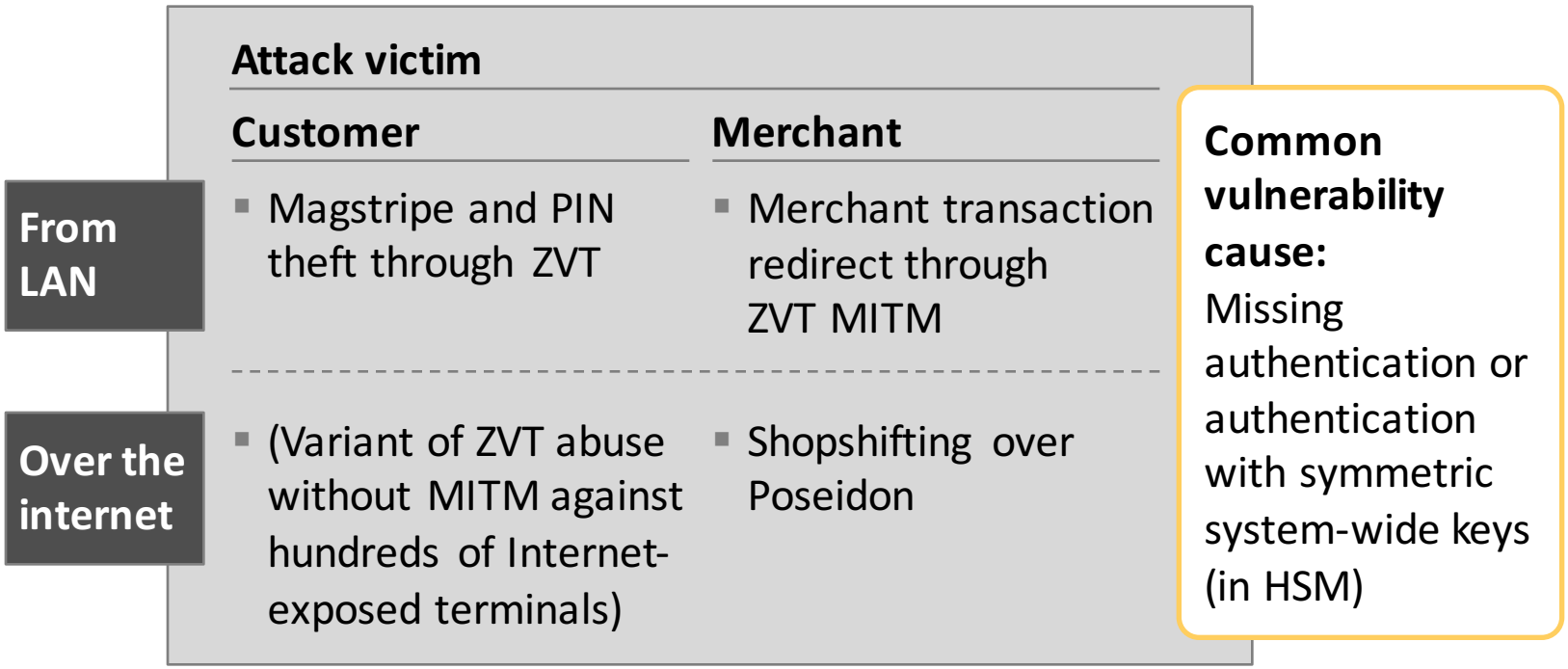
Demo 3

Shopshifting over the Internet: Issuing a refund transaction

Shop shifting attack puts merchants at significant fraud risk

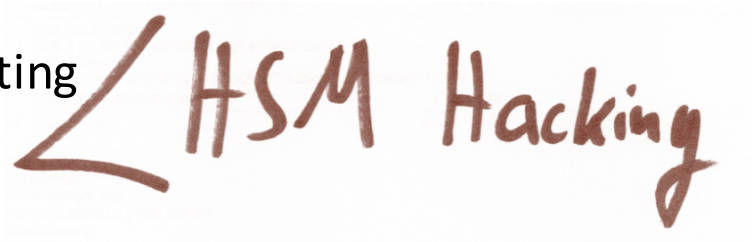


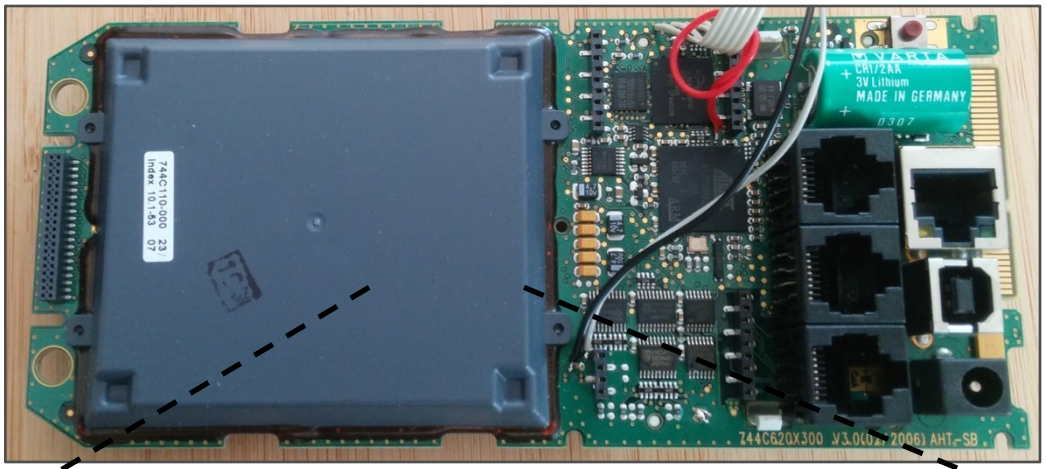
Customers and merchants are vulnerable to various payment abuse scenarios

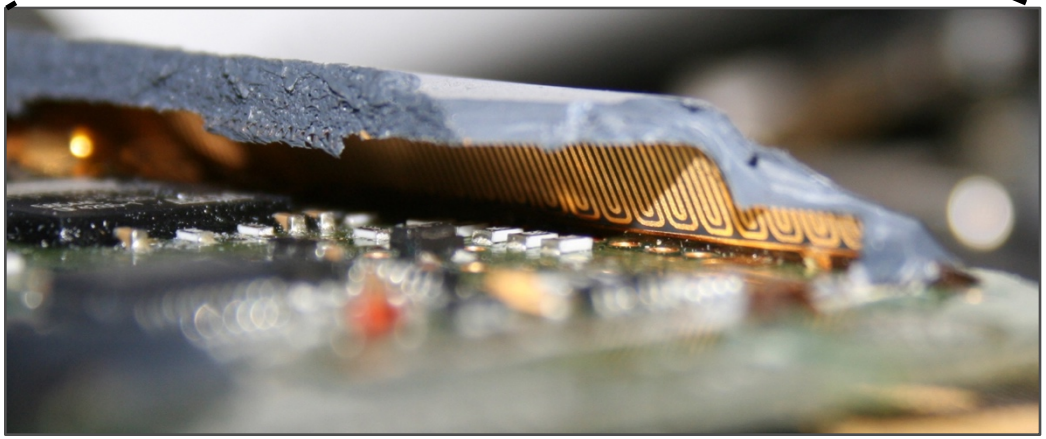


Agenda

- Local payment abuse
- Poseidon shopshifting
- **Evolution need**

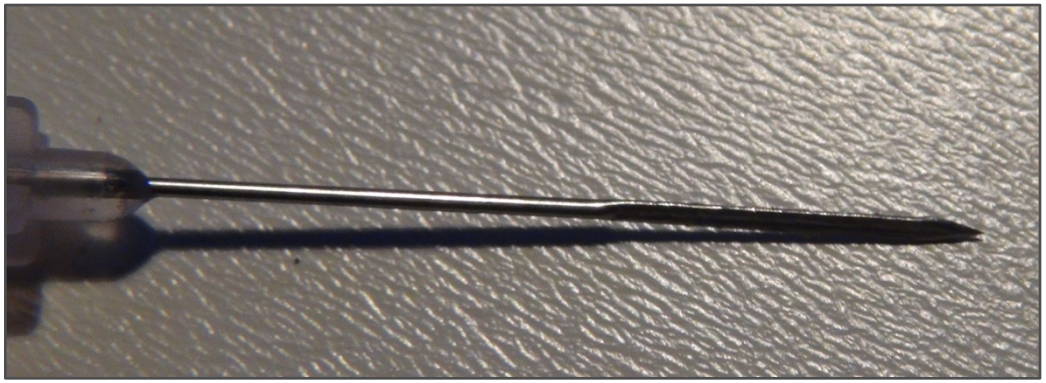






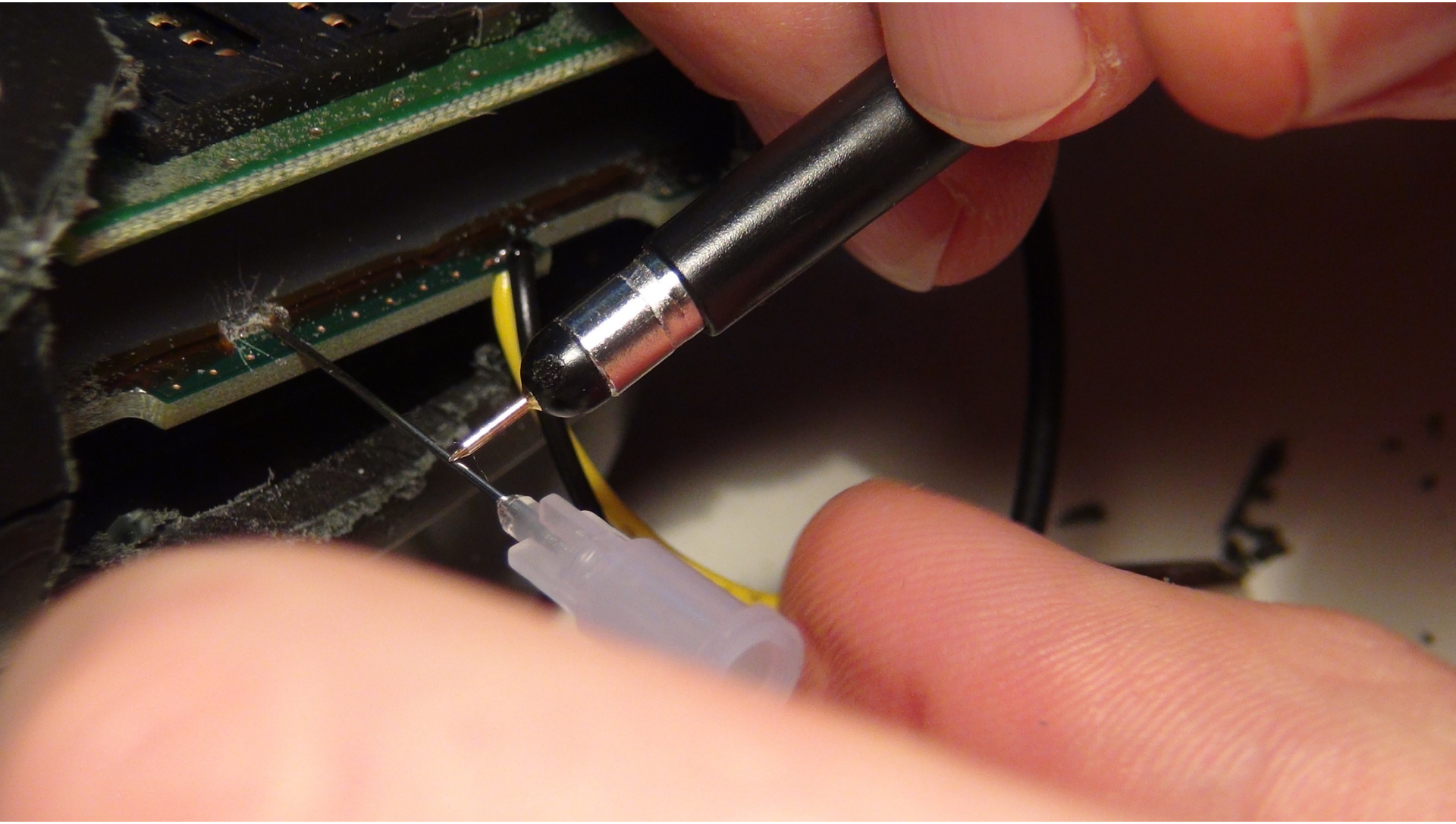
HSM Hacking Challenge – Secrets are stored in a battery-backed RAM under a plastic cover.

When a metal mesh in this plastic cover is breached, the secrets are erased.

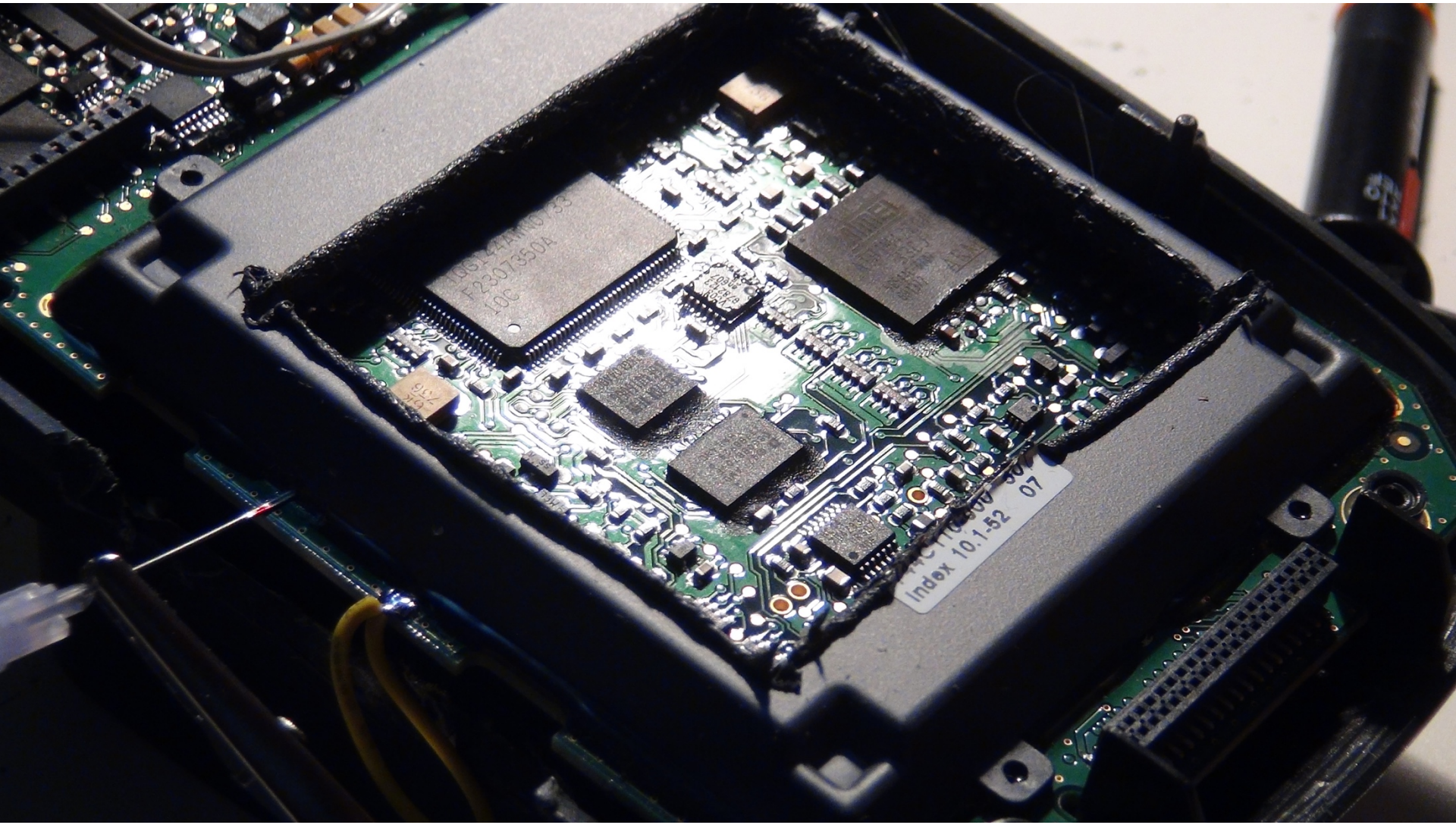


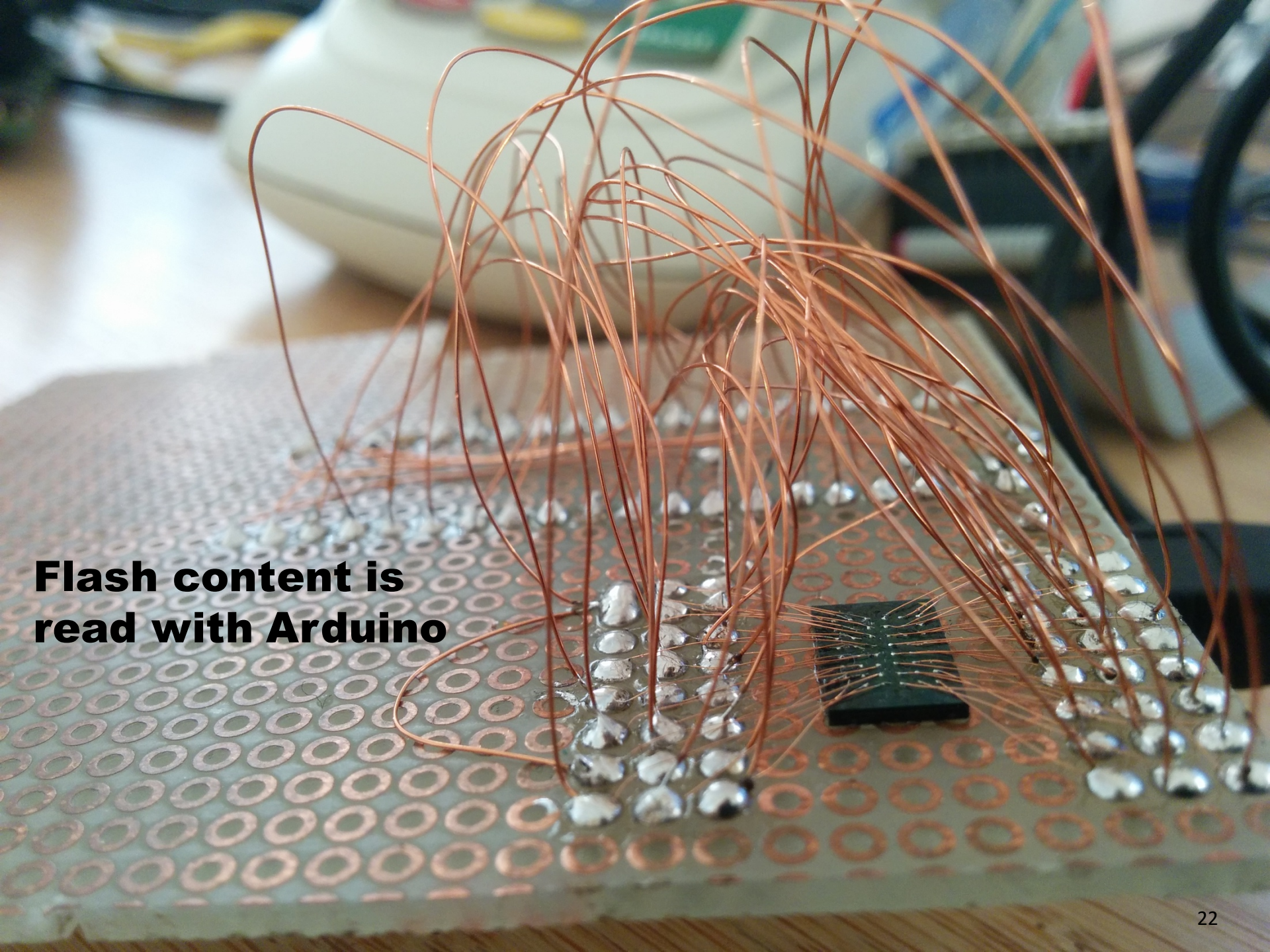
Tool of choice – The Hacking Needle

Needle fits underneath mesh, overwrites mesh check

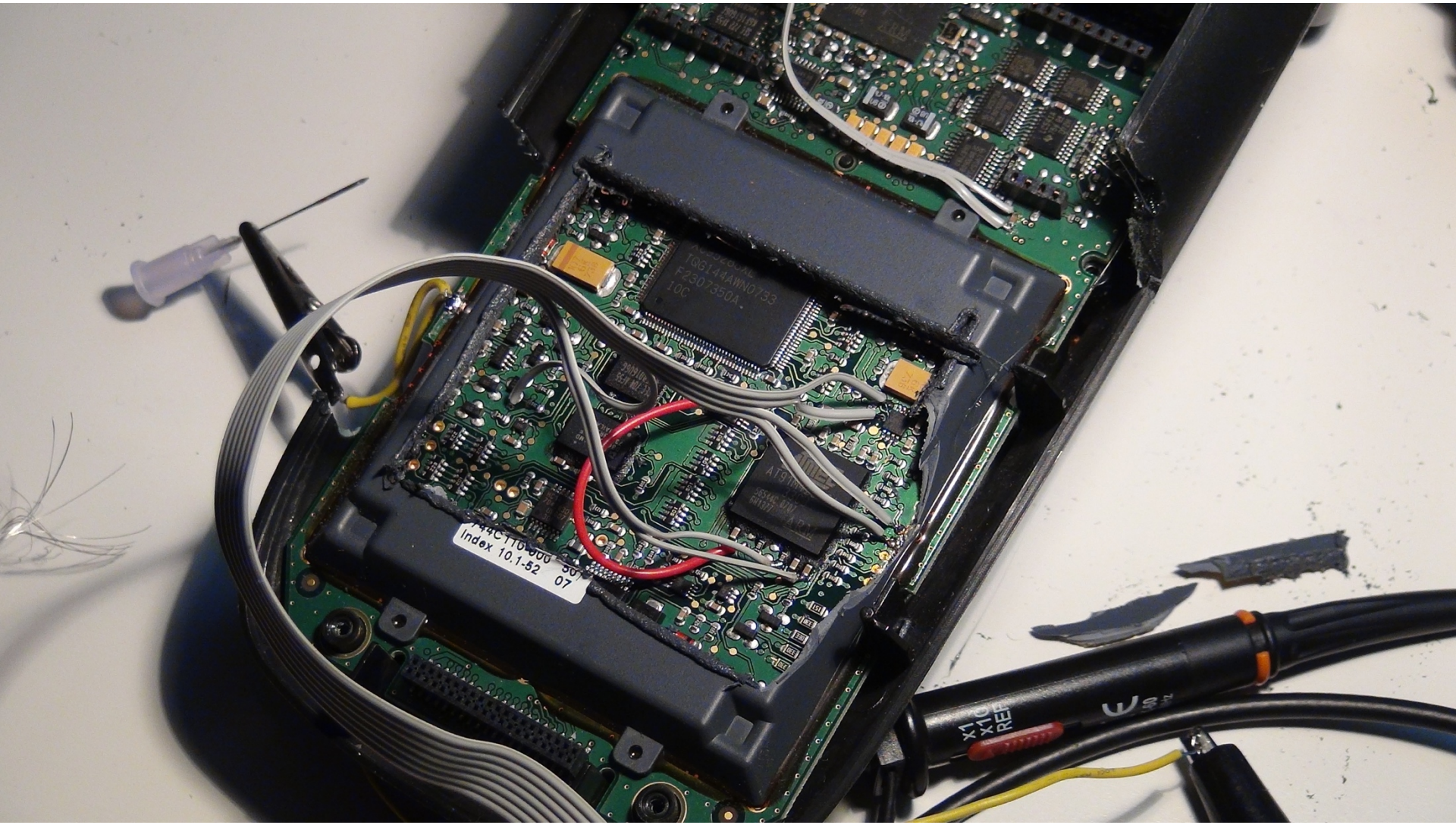


With security check deactivated, RAM inside HSM can be read




Flash content is
read with Arduino

Active JTAG in HSM allows for debugging



HSM compromise affects keys for ZVT, Poseidon, EMV and others

ZVT MAC computation inside HSM:

```
0002e780  01 01 01 21 3f 18 00 01  40 00 84 00 08 16 46 6f  |...!?...@.....Fo|
0002e790  6e 74 36 78 38 20 0d f0  00 08 ff f7 00 00 00 00  |nt6x8 ..........|
0002e7a0  e4 5b 02 01 02 00 08 00  c6 01 41 41 41 41 41 41  |.[........AAAAAA|
0002e7b0  41 41 00 00 0c 0a 48 65  6c 6c 6f 2c 10 57 6f 72  |AA....Hello,.Wor|
0002e7c0  6c 64 21 0a 00 ff ff ff  ff ff ff ff ff ff ff ff  |ld!.............|
[...]
000031f0  14 04 00 00 08 32 00 01  7e 02 aa 4d 04 85 5f 5e  |.....2..~..M.._^|
00003200  44 32 00 01 02 00 00 00  b5 eb f1 a1 11 e1 18 1a  |D2..............|
00003210  cc 33 00 01 3c 05 14 ac  00 42 c2 88 44 32 00 01  |.3..<....B..D2..|
```

Agenda

- Local payment abuse
- Poseidon shopshifting
- **Evolution need**

ZVT and Poseidon are not secure by design

Vulnerability root causes

ZVT

System-wide signature keys

Used with symmetric crypto

Stored in insecure HSMs

Poseidon

System-wide auth keys

Also, but not making matters worse:

Stored in insecure HSMs

Both protocols mix “security through obscurity” (system-wide keys) with “security certification” (HSMs).
Neither implements “security by design”

Heuristic defenses are needed in the short term

Payment system need better protocols and more secure hardware!
While these are being developed, a few stop-gap measures are available:

	ZVT	Poseidon
Deactivate unnecessary functions	▪ Remote manageability with static password – Should require a confirmation on terminal instead	▪ Refund (activated by default!) ▪ SIM card top-up (deactivated by default)
Detect suspicious behavior	▪ Magstripe transaction from EMV- capable card (must be checked online since card data cannot be trusted)	▪ Terminal IDs connecting to wrong port (already implemented in some places) ▪ Serial number changes for a terminal ID (not effective when HSM is hacked) ▪ Refunds that do not correspond to transaction in cash register (double-entry accounting)

Other payment standards appear equally vulnerable

Main ZVT alternative: OPI

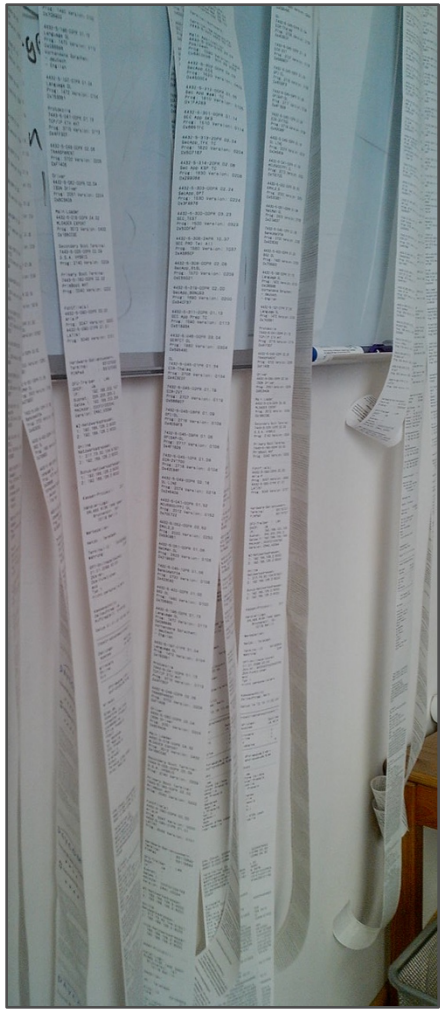
- Open Payment Initiative protocol is more modern than ZVT: XML-based, 2003
- Still lacks authentication and encryption
- Misses some of the functionality that can be abused in ZVT (good!)
- Vendors use proprietary extensions to bring back such vulnerable functionality in OPI (bad!), including remote maintenance

Poseidon's family: ISO 8583

- Poseidon is one of many ISO 8583 dialects
- System-wide symmetric keys, Poseidon's Achilles heel, are not mandatory in ISO 8583
- It does not appear that current terminals go through key exchanges as part of their initialization, suggesting that other ISO 8583 dialects also suffer from Poseidon's security issues
- International security research community: Your help is needed

Take aways

- **Payment systems allow for magstripe/PIN theft and remote attacks on merchants**
- **Payment protocols need actual authentication using individual keys**
- **Victims of card abuse should fight their banks, researchers should help**



Questions?

Fabian Bräunlein <fabian@srlabs.de>
Philipp Maier <dexter@srlabs.de>
Karsten Nohl <nohl@srlabs.de>