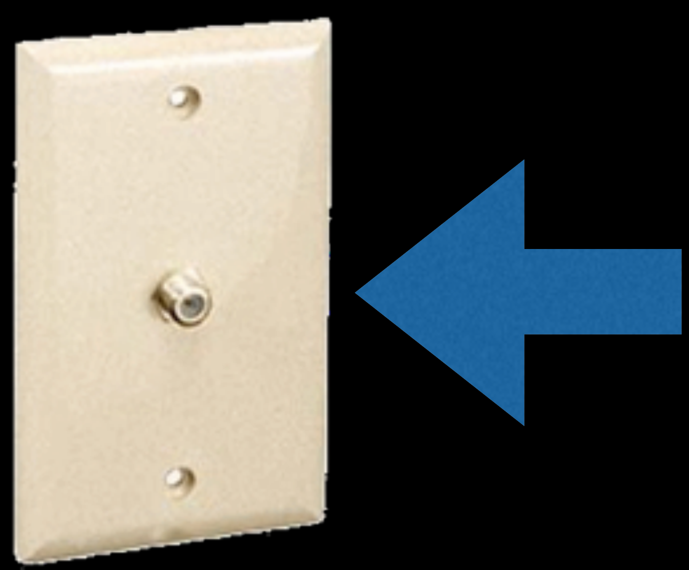
Practical Attacks on DOCSIS

Who am I?

- @drspringfield
- Security researcher at Accuvant
- Work in embedded device security, reverse engineering, exploit dev
- No background in DOCSIS, but I find it interesting

DOCSIS

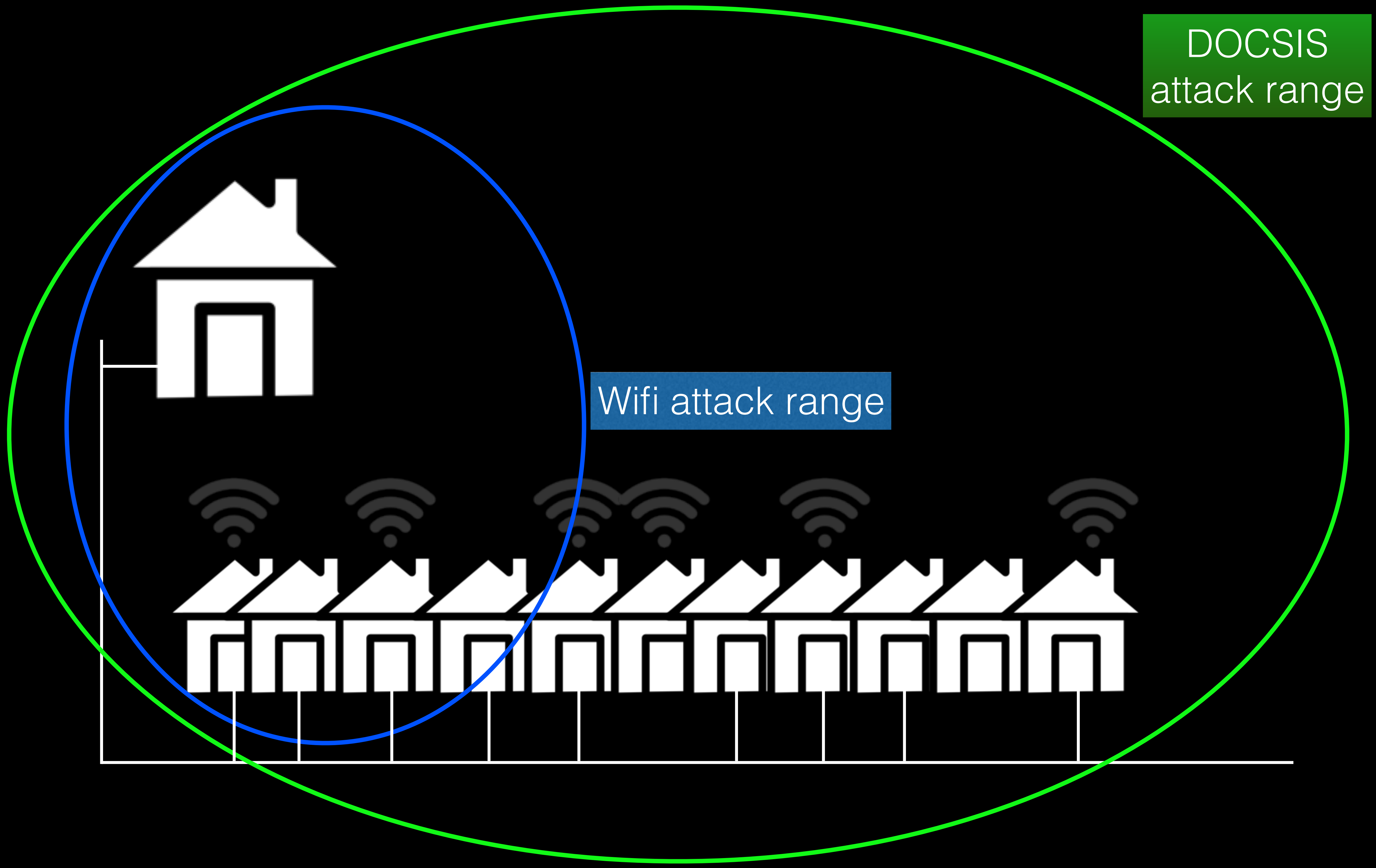
- "Data Over Cable Service Interface Specification"
- RF protocol stack underneath IP



It's how your cable modem accesses the internet

- This presentation focuses on DOCSIS, not cable modems
- Applicable in US, Europe (layer 1 differences), Japan

Motivation



Network overview

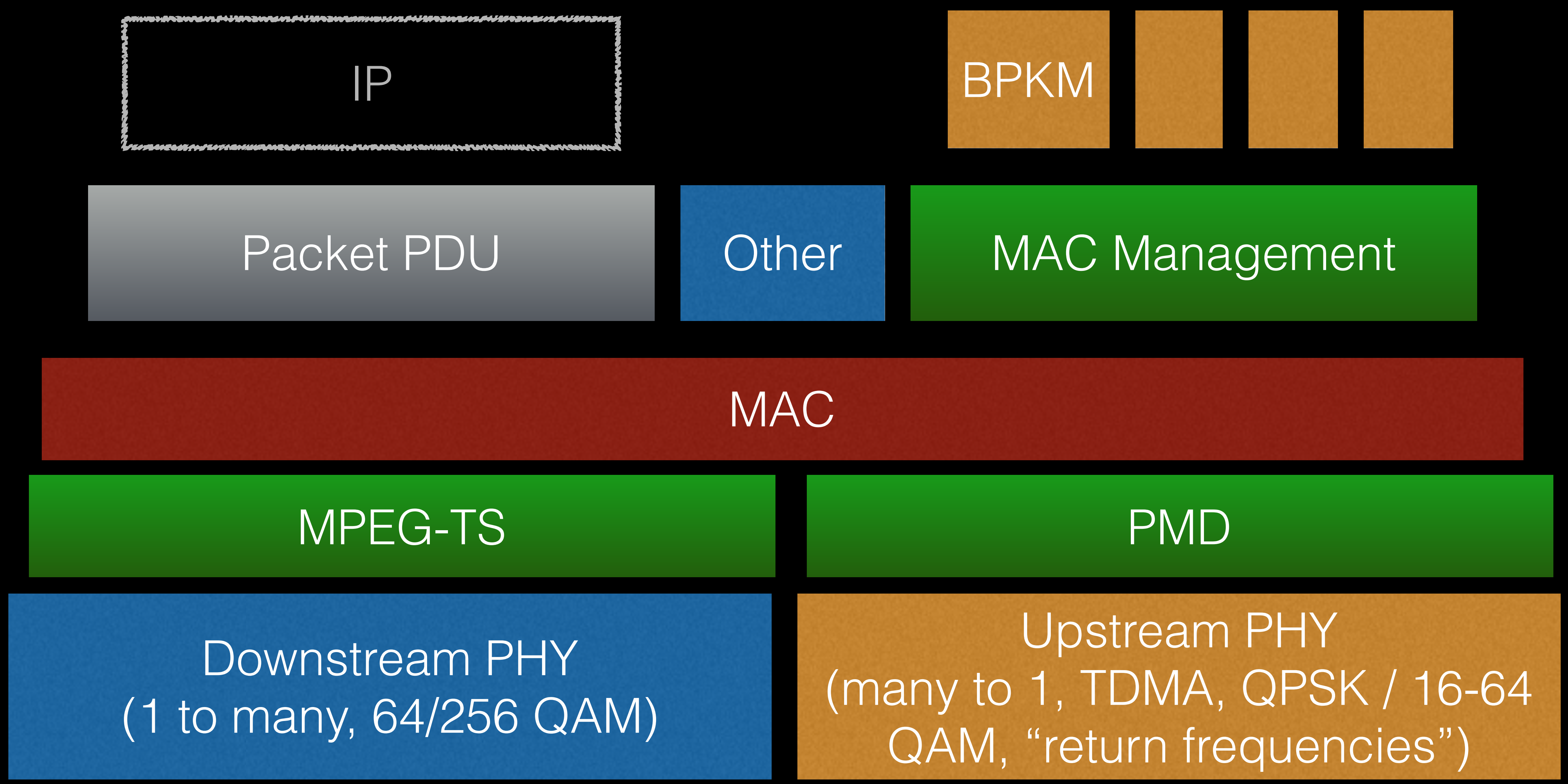
Cable modem termination system = CMTS

Coax/fiber network = HFC

Cable Modem = CM

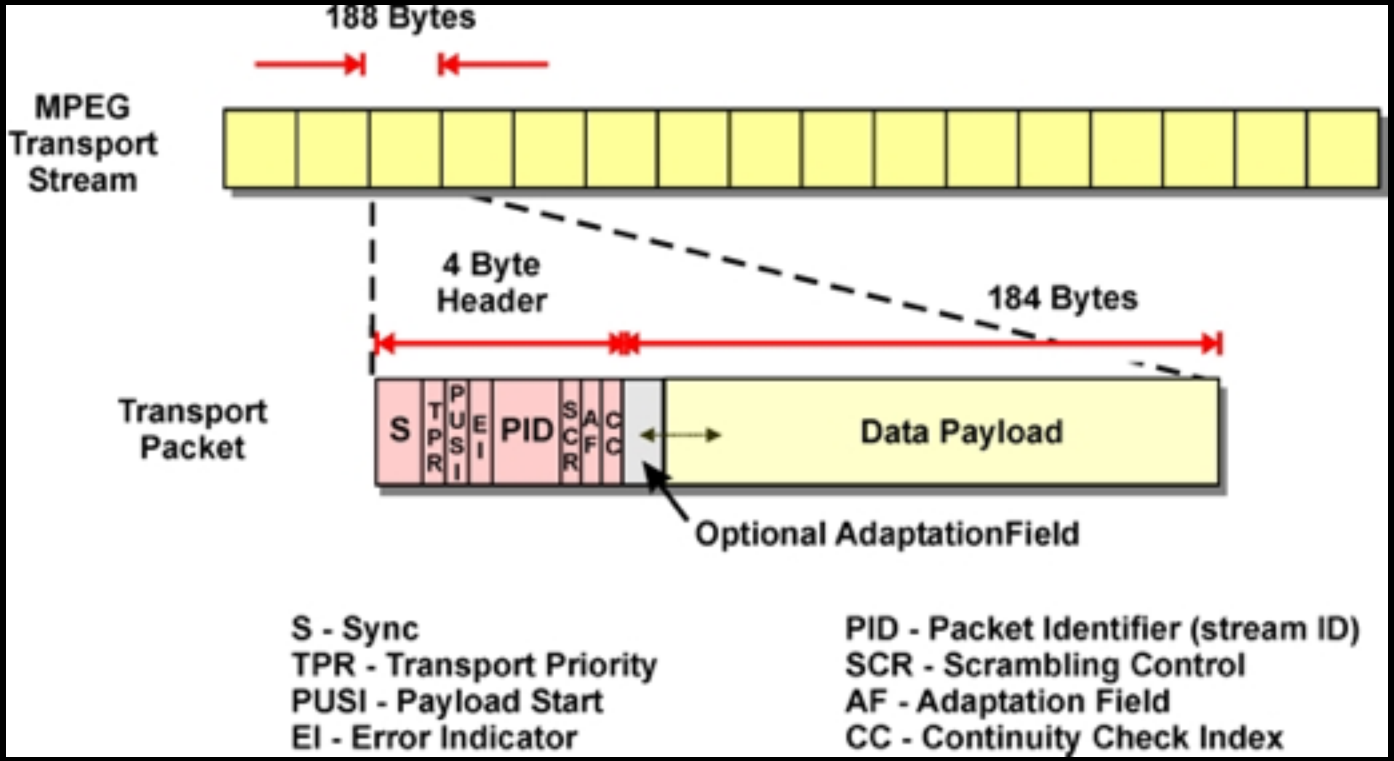
Router = CPE

Protocol overview



Protocol overview

MPEG-TS



- Fixed size packets
- PID is 0x1FFE for DOCSIS
- Desegmentation occurs here

Protocol overview

MAC

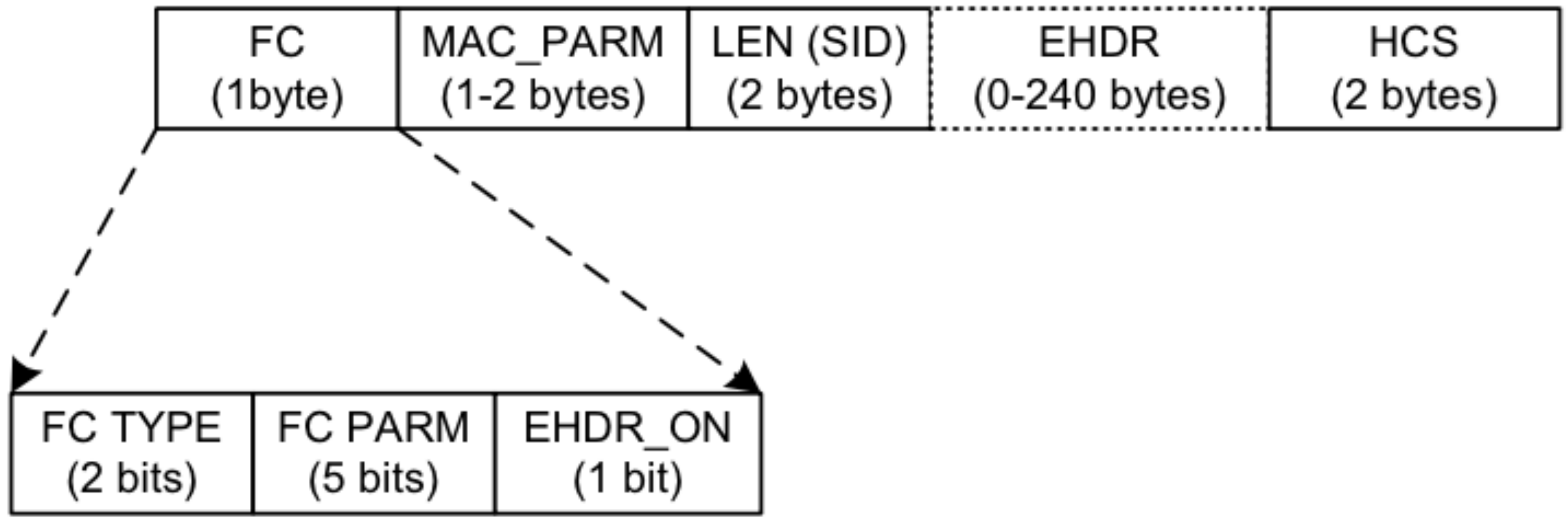


Figure 6–3 - MAC Header Format

- Frame control determines packet type
- Extended header supports TLVs

How to access?

Downstream

1. Hacked cable modem
    - Model-specific, clunky
2. SDR
3. Dedicated hardware
    - Cheap, ready to go

Clear QAM dongle

- MyGica USB QAM HDTV tuner
- $29 on Amazon
- Supported out of the box by Linux DVB API
- Truly plug and play
  - Order one now!

How to access?

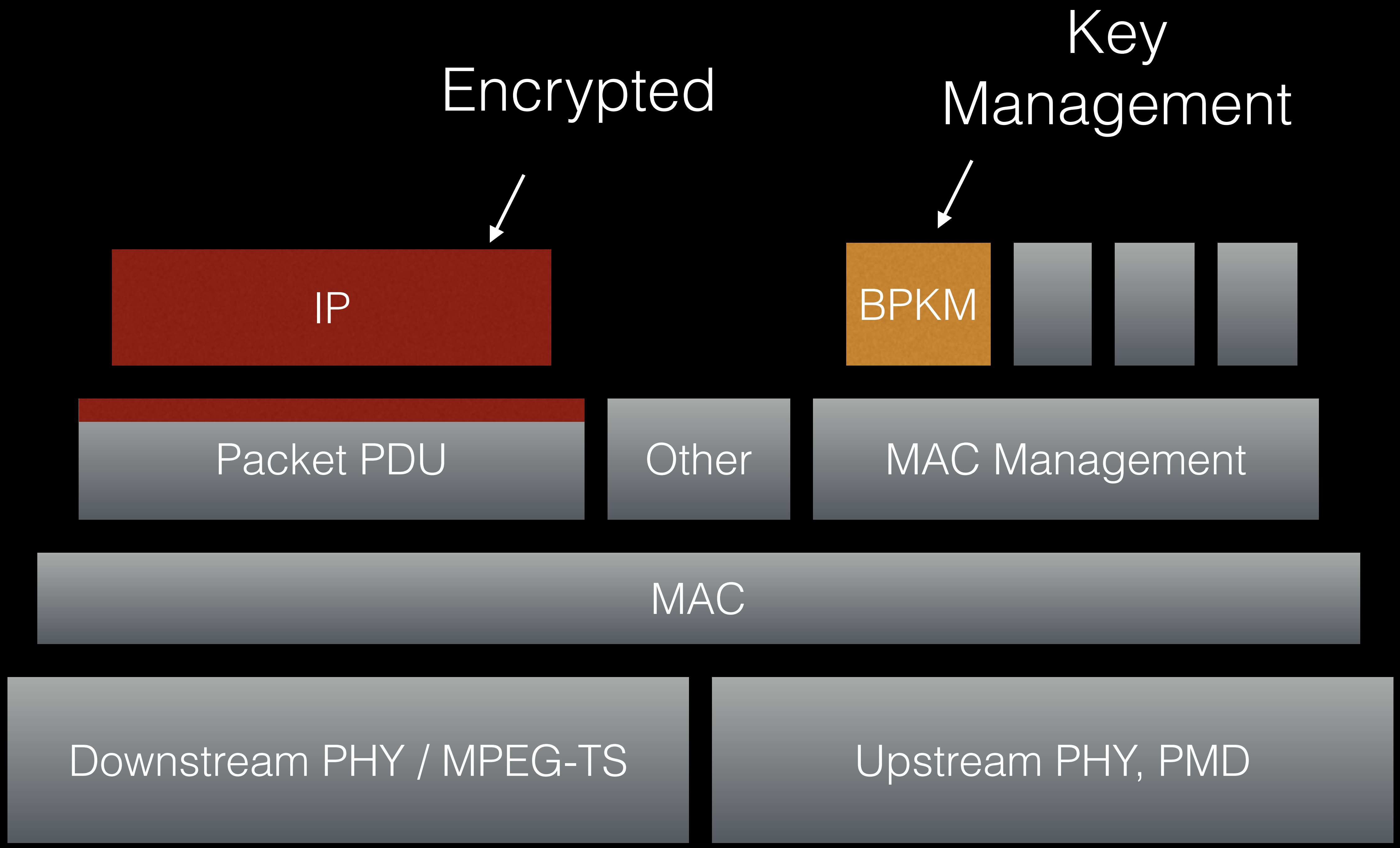
Upstream

1. ~~Hacked cable modem~~
    - Hacked CMTS?
        - Expensive, highly model-specific
2. SDR
    - Best/only option
3. ~~Dedicated hardware~~
    - Doesn’t exist (that I know of)

DOCSIS SEC / BPI+

- You didn't think it was all in the clear did you?
- DOCSIS SEC/BPI+
  - Encryption and authentication protocol in DOCSIS
    - BPI (Baseline Privacy Interface) in DOCSIS 1.0
    - BPI+ in DOCSIS 1.1 and 2.0
    - SEC (Security) in DOCSIS 3.0

DOCSIS SEC / BPI+



BPKM

Baseline Privacy Key Management

- Client/server key synchronization protocol
- Operates on SAIDs (Security Association IDs)
  - 14-bit random integers
- Authorization:
  - Prevent cable theft and device spoofing by cryptographically identifying modems
  - Don’t care much (this talk is not about service theft)
- Traffic Encryption Key (TEK) provisioning
  - Provisioning TEKs that encrypt customer traffic

BPKM Authorization

Auth Request	Auth Response
Identification info serial#, MAC, CM’s public key	Authorization Key (RSA encrypted with CM’s key)
Certificate	Key lifetime
Security Capabilities (supported algorithms)	Key sequence #
SAID (initialized to zero)	Attributes (cryptographic algorithm)

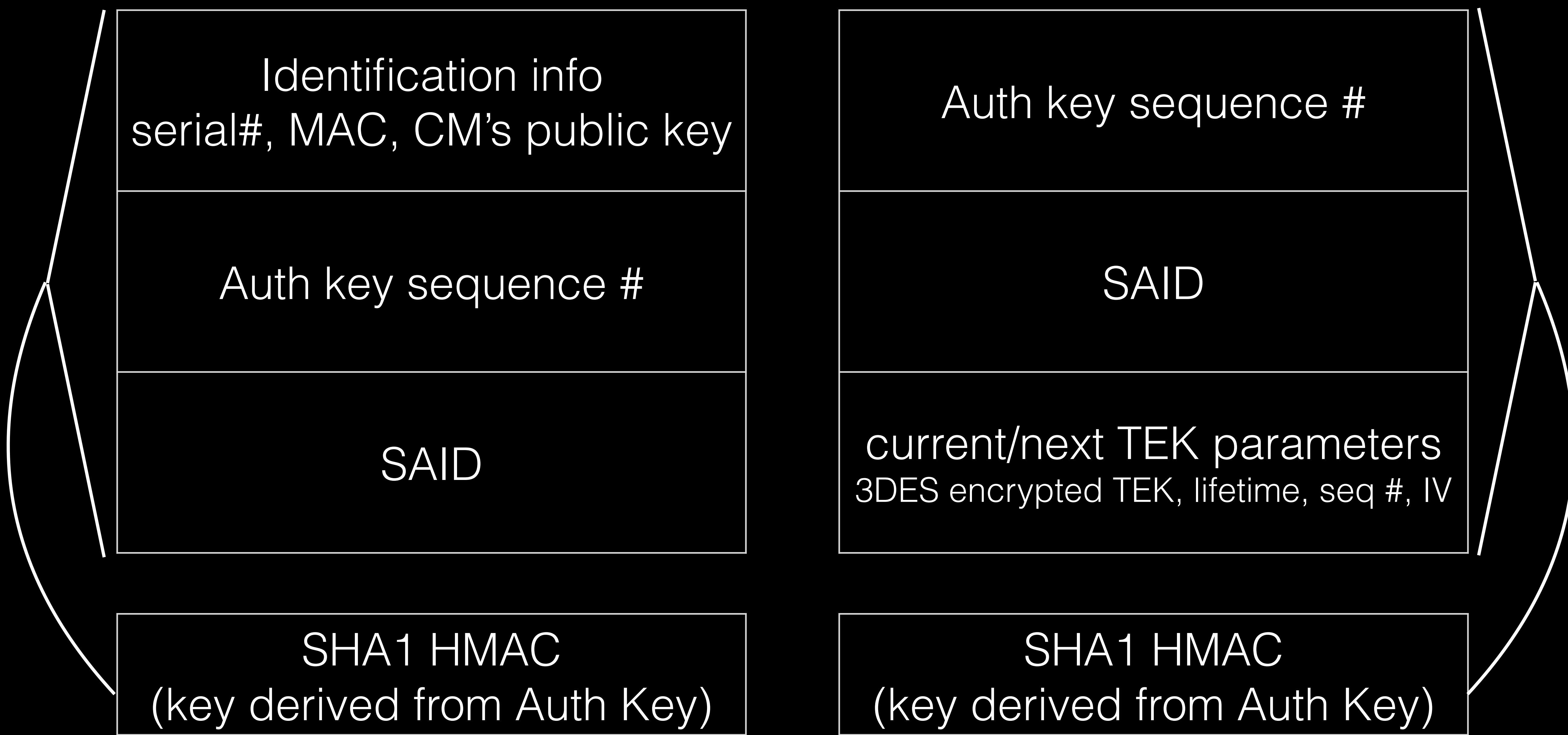
BPKM Authorization

- This request is rare
  - One week lifetime*
  - On CM boot
- This is where supported security capabilities are announced and selected
  - Note that supported capabilities field is unsigned
  - Downgrade attack possible?

Algorithms

- Encryption algorithms
  - ~~40-bit DES~~
  - 56-bit DES
  - 128-bit AES (added in DOCSIS 3.0)
- Data authentication
  - None

BPKM TEK provisioning



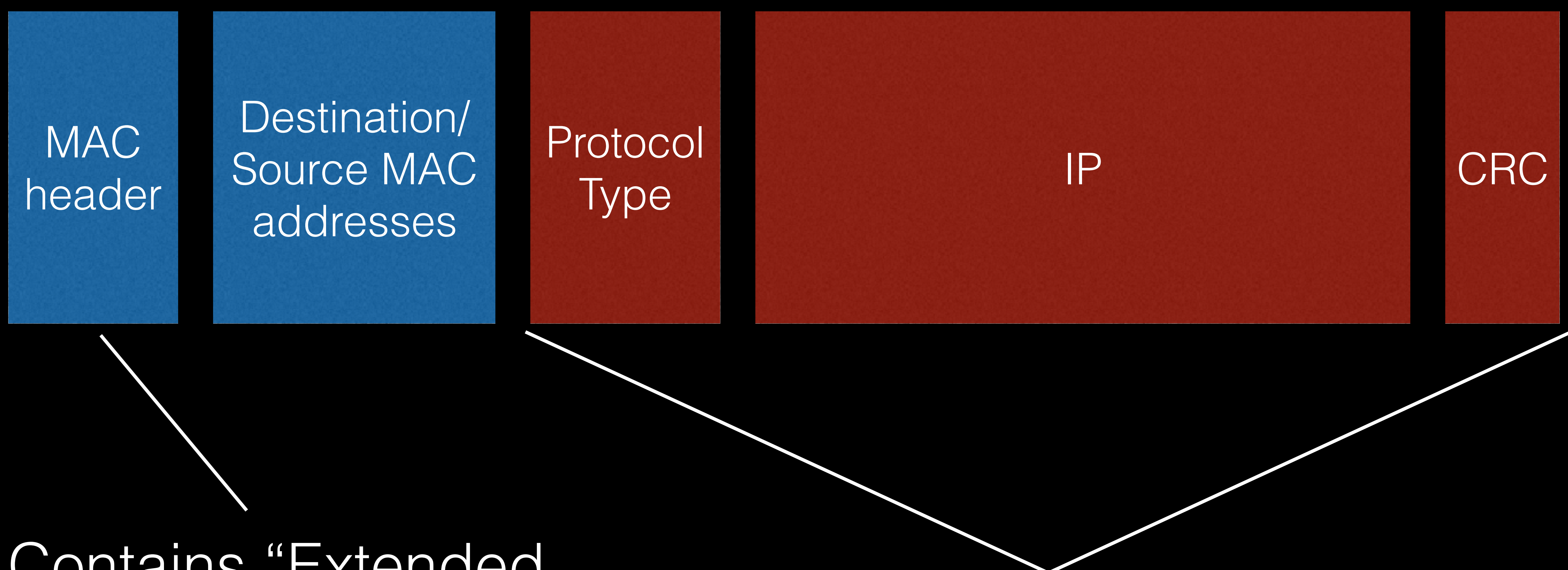
Key Request

Key Response

BPKM TEK provisioning

- More frequent
  - New TEK 6 hours*
- TEK is protected with Auth key-derived KEK
- IV is in the TEK parameters
  - Only 1 IV is used for the lifetime of TEK
  - Chaining is re-initialized with each frame

Packet PDU encryption (almost* all Packet PDUs)



Contains “Extended Header” identifying Encryption enabled and SAID

Encrypted with TEK using CBC with residual block termination

Problems with DOCSIS SEC

- Use of 56-bit DES
  - DOCSIS 3.0 adds support for AES
  - Never seen AES used*
  - Lack of use likely due to DOCSIS 2.0 support
    - CMTS are not picking most secure cryptographic algorithm supported by CM

Problems with DOCSIS SEC

- Re-use of CBC IV in each frame
  - Required by specification
  - Identical packets will have identical ciphertext

Exploiting these vulnerabilities

- First focused on attacks performable with **passive downstream read access only**
  - Reduced cost and complexity to perform
  - No significant chance of detection
  - Doesn’t even require being a subscriber

DOCSIS DES brute force

1. Identify the victims
2. Obtain tuples for each victim (X, E(X))
3. Brute-force DES key to determine X from E(X)
   - If X static, time/memory tradeoff possible

Identifying the victims

- Packet PDU exposes source & destination MAC addresses in clear
- ARP traffic is in the clear
  - IP registration occurs prior to encryption/ authentication (in normal provisioning flow)
    - Unless EAE enabled (Early Authentication & Encryption)
    - Never seen this enabled*

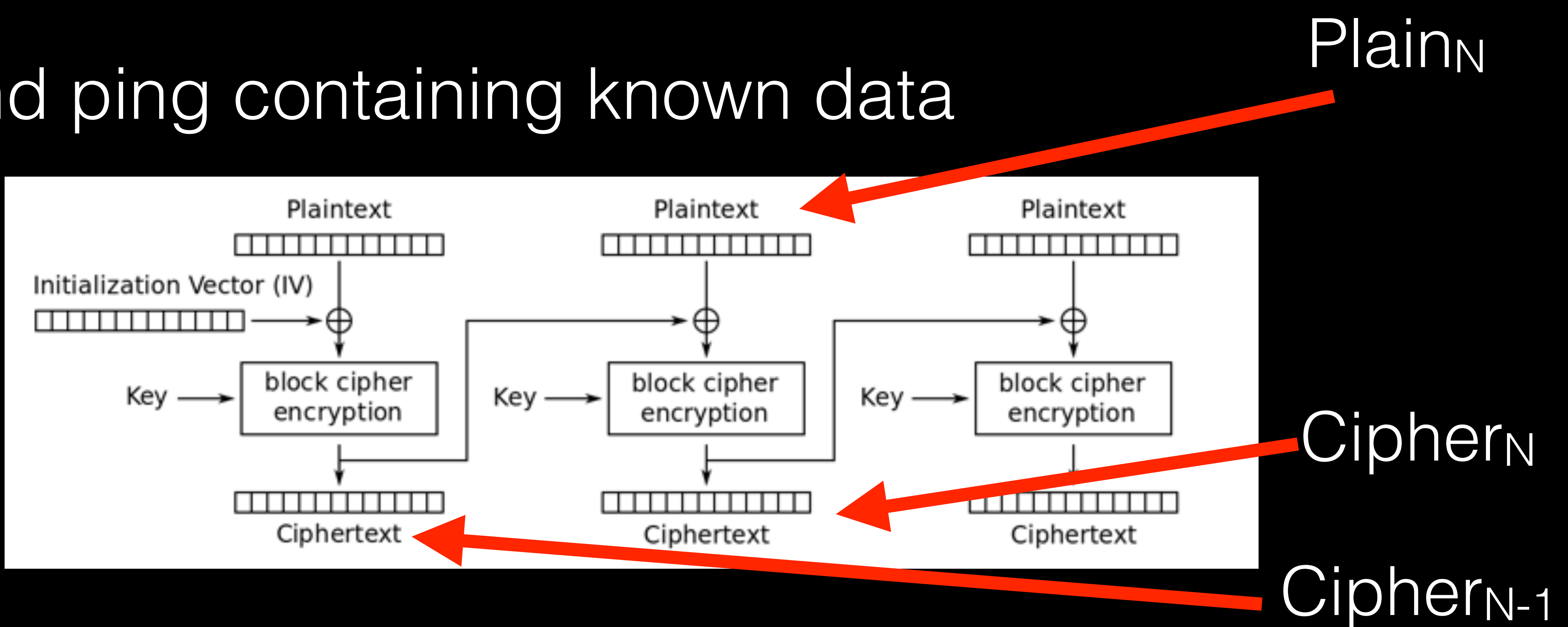
Identifying the victims

- Sniff ARP traffic on downstream and collect subnets
- Send ICMP ping sweeps across subnets with various packets sizes
    - Irrelevant how victim CPE responds
- Perform correlation between encrypted packet sizes and sent ICMP packet length
    - Produce (MAC, IP) tuples

Obtaining known plaintext values

- Send ping containing known data

CBC



- Re-send *identical* packet but change $Plain_N = Cipher_{N-1}$
- Subsequent $Cipher_N = E(Cipher_{N-1} \oplus Cipher_{N-1}) = E(0)$
- Sniffing lossy due to channel bonding

Brute-forcing 56-bit DES

- Attacking 56-bit DES is not new
    - EFF DES Cracker (1998)
    - Moxie Marlinspike (2012) for MS-CHAPv2 using FPGAs
    - Karsten Nohl (2013) for SIM cards using rainbow tables
    - Sergey Gordeychik/Alex Zaitsev (2014) reproducing Karsten's attack using FPGAs

DOCSIS use-case

- DES TEKs are only useful for 6 hours of traffic
    - Ideally, cracking DES TEKs should be cheap, fast (<6hrs), and repeatable
    - Some upfront cost is acceptable

Existing DES attack platforms

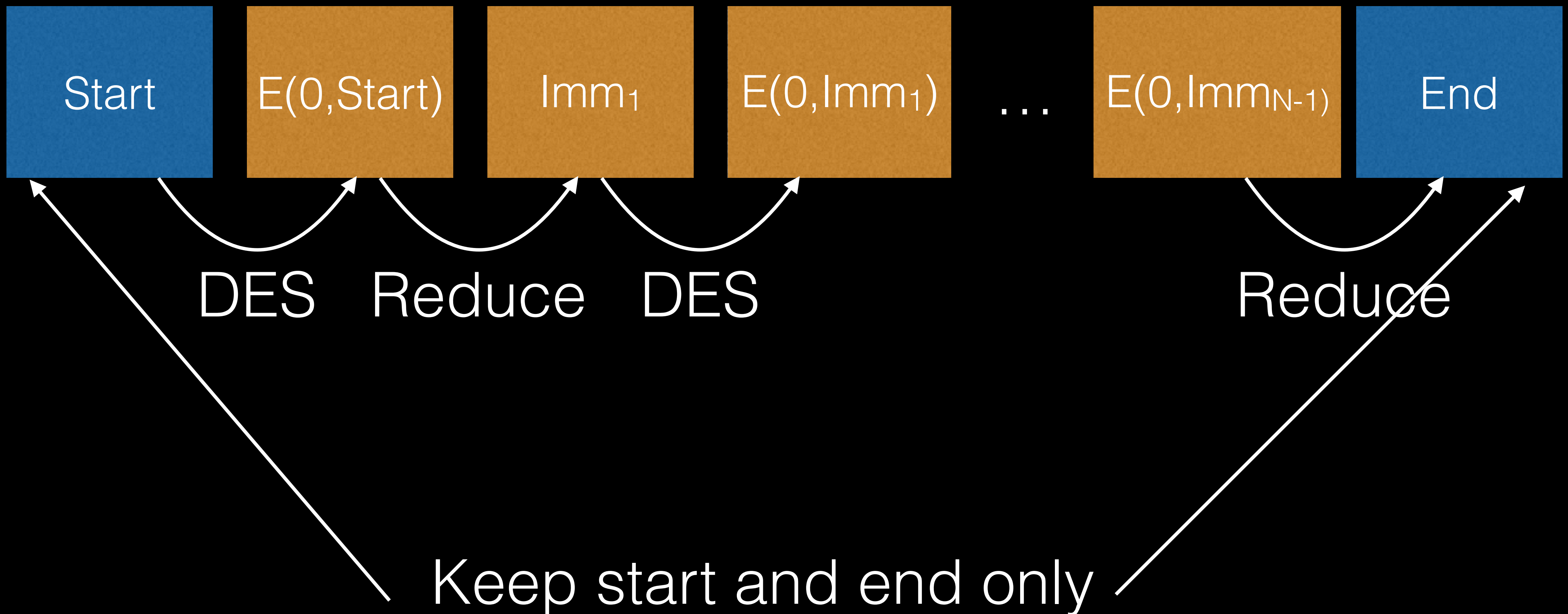
Implementer	Upfront cost	Crack time
Karsten	~ EUR 1500	1 minute
Moxie	?	23 hours
Sergey/Alex	~$1400	3.4 days

My attack platform

- Uses rainbow tables for time/memory tradeoff since E(0)
- Created with Amazon EC2, tables stored in S3
- Total upfront EC2 cost was around $2000 (across ~3 weeks) on GPU spot instances
- Cracking cost is $0.22, takes 23 minutes
  - Assuming spin-up time is amortized

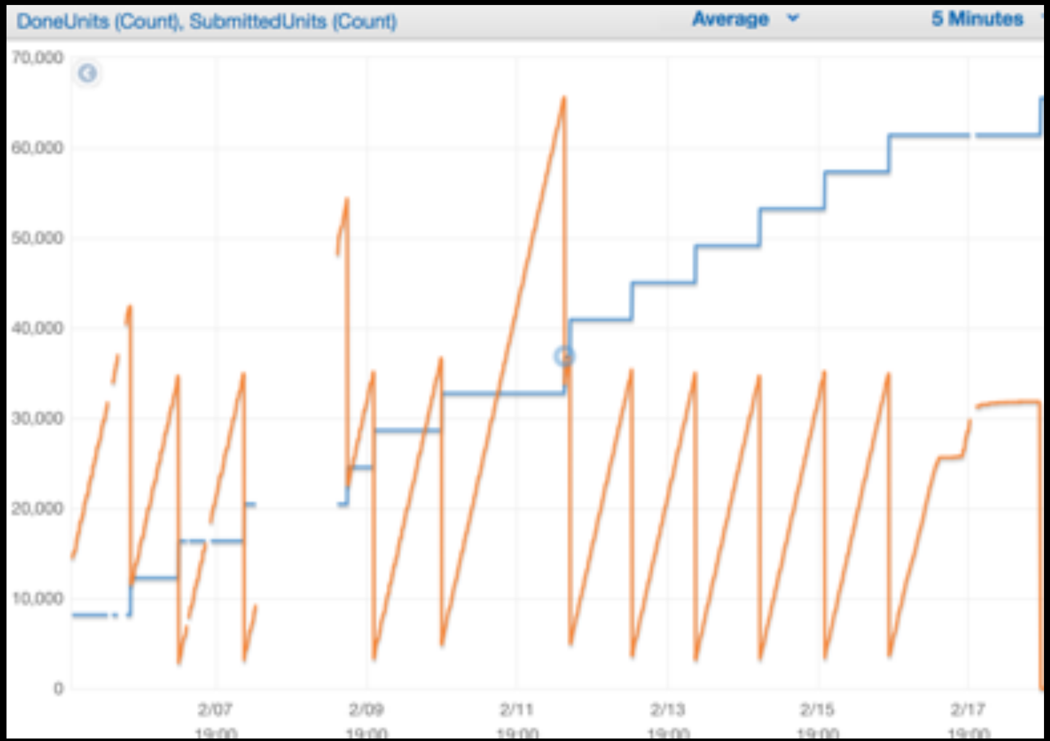
DES rainbow tables

- 16 rainbow tables, 64gb apiece (1 tb total)
- Each chain represents 1048576 DES operations (1 MegaDES)



Attack platform systems: generation

- RTGen
  - GPU+CPU instances creating work units
  - Each work unit is 1 TeraDES
  - Massively parallelized



- Assimilator
  - Memory-optimized instance
  - Sorts and uniques work unit chains (by endpoint) into finished tables
  - Each finished table is 64gb, and represents 4 PetaDES

Attack platform systems: cracking

- Cracker
  - GPU instance (one for each table index)
  - Does “precalculation”: calculate every possible chain endpoint that would contain E(X)
  - Takes 20 minutes
- Table Lookup
  - Memory instance (one for each table index)
  - Looks up all precalculated chain endpoints in final table
  - For each found, walks chain from start point to find cleartext X
  - Takes ~3 minutes

Success probability

- Due to reduction step, cracking success is not guaranteed
- Probability of two ciphertexts reducing to the same key is 255/2**56
  - Unless collision occurs at same position, chains will not merge
  - This is because position is used in reduction function
- In practice, about 4% endpoint collisions in a table part

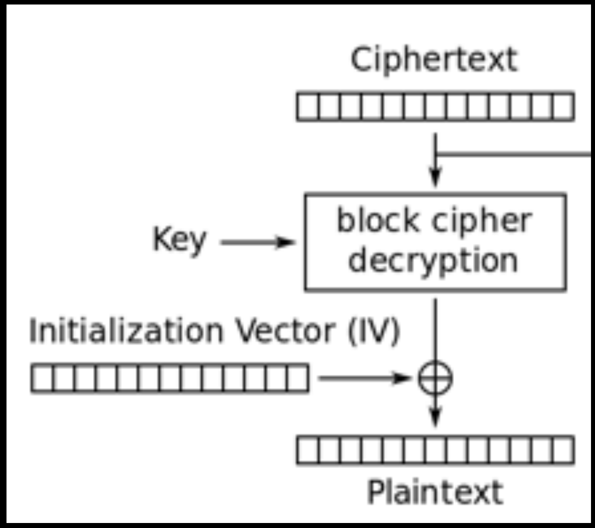
Notes on performance

- Use bit slicing to do N parallel DES operations
  - N depends on GPU (32) vs CPU (128)
  - Each DES bit is represented in a single variable
  - This makes permutation (bit-shuffling) operations free, which are otherwise costly on CPUs
  - Bits in the variable represent DES operations occurring in parallel

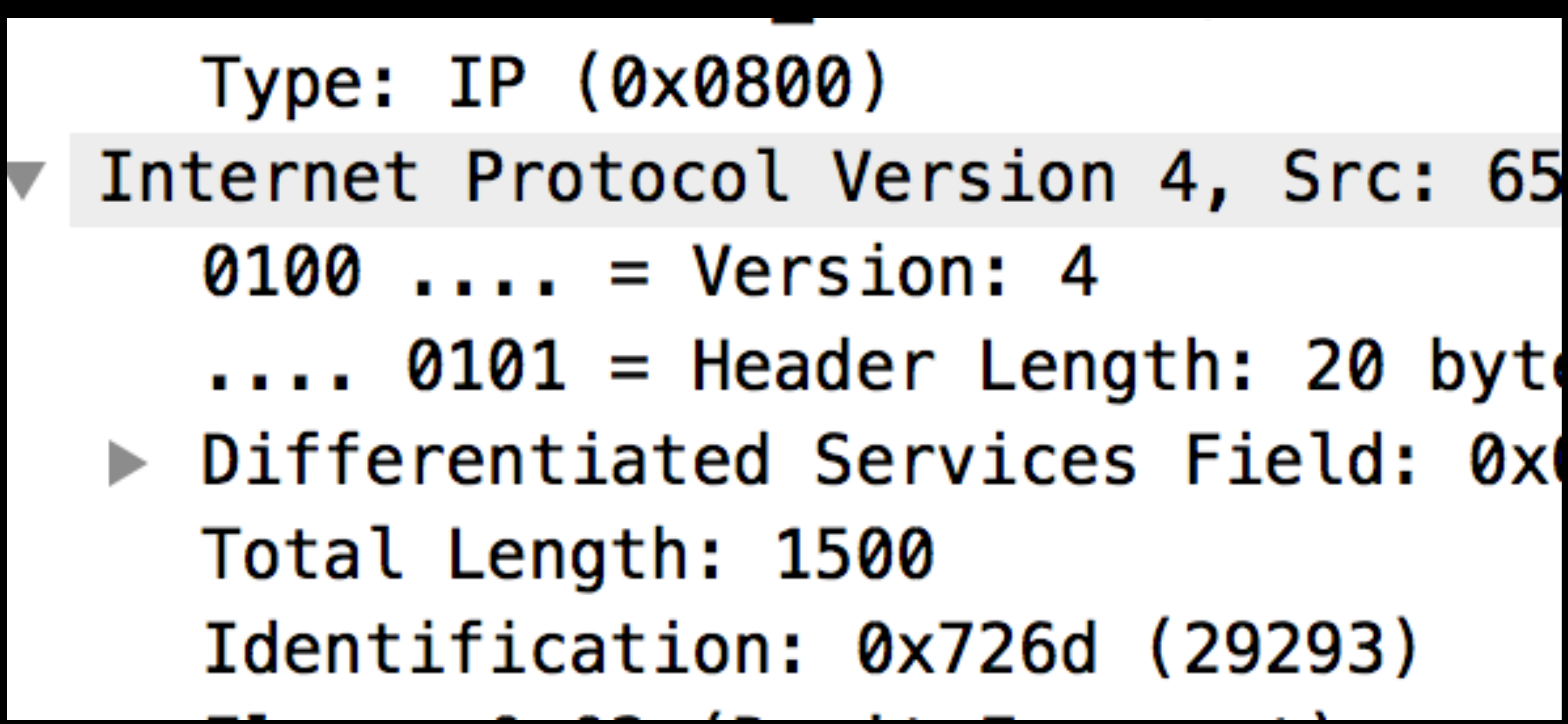
Data Bit X	Key 0	Key 1	Key 2	Key 3	Key 4	…	Key N

- Replace typical reduction algorithm (addition) with bitslice-friendly (xor) to avoid serializing bits between steps

IV recovery



- Cracking only reveals key, what about IV?
  - Only in BPKM Key Reply
  - Ignoring it is OK, since you only lose the first block, but Wireshark won't like your pcaps
- Heuristic recovery



0800

45

00

05dc

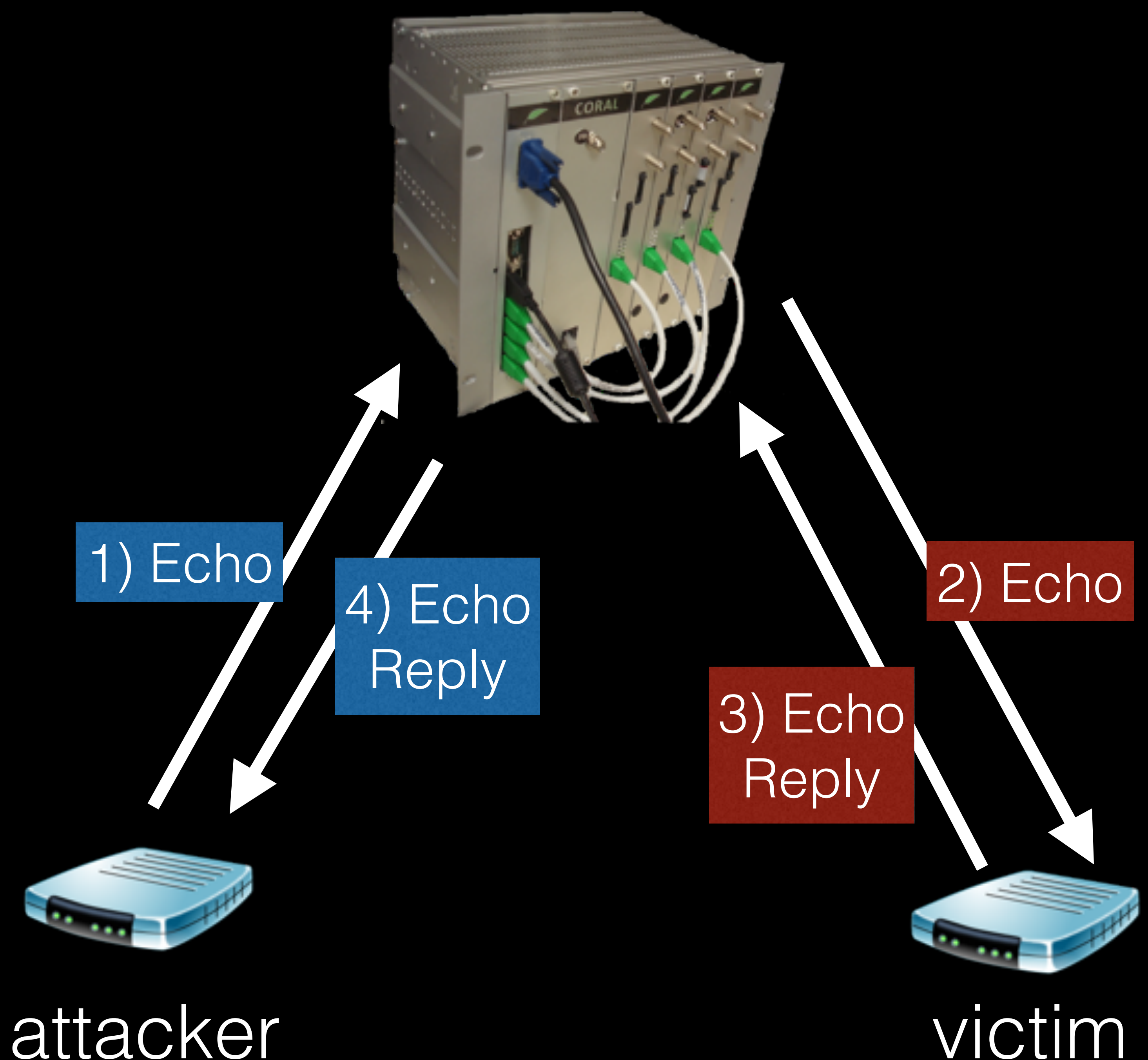
Video Demo

```
braden@cabletables:~/docsis_cracker_1.0$ sudo python docsis_crack_ui.py --seed-channel 657
```

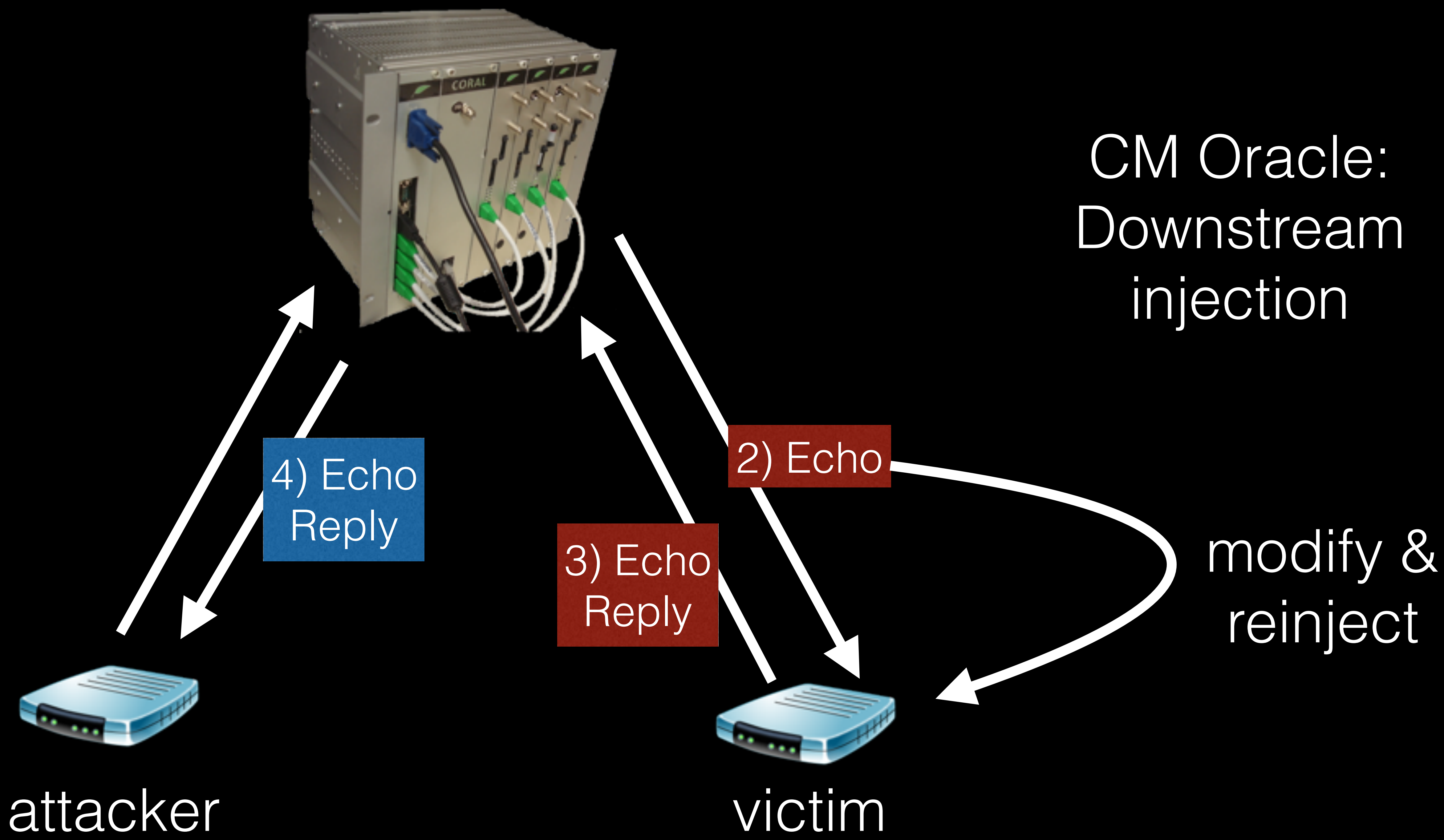
Decryption oracle attacks

- Two theoretical attacks that are *nearly* practical
- Work regardless of encryption algorithm
  - These will be more important after AES used
- Use the CMTS/CM + ICMP as a decryption oracle
  - This is an active attack, and only available for TEK lifetime
  - Requires functional ICMP to/from victim
  - Requires being a subscriber

Decryption oracle



Decryption oracle



CM Oracle: Downstream injection

CM Oracle attack

1. Identify the victims, collect downstream encrypted payloads

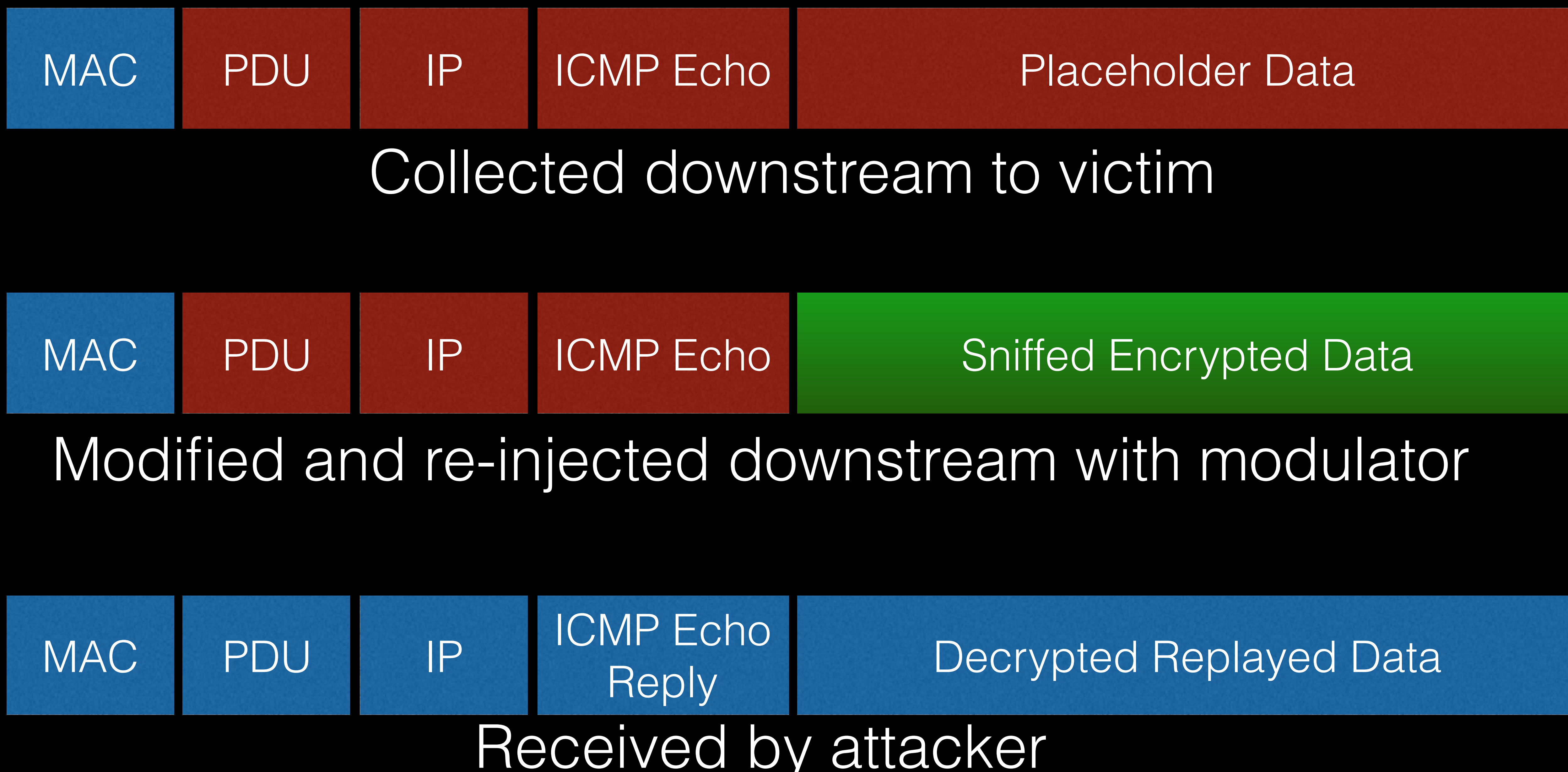
2. Send ICMP echo to victim, collect encrypted ping request

3. **Inject** ICMP echo, but splice in desired payload

   - *Requires QAM modulator*

4. Wait for echo response containing cleartext

CM Oracle attack

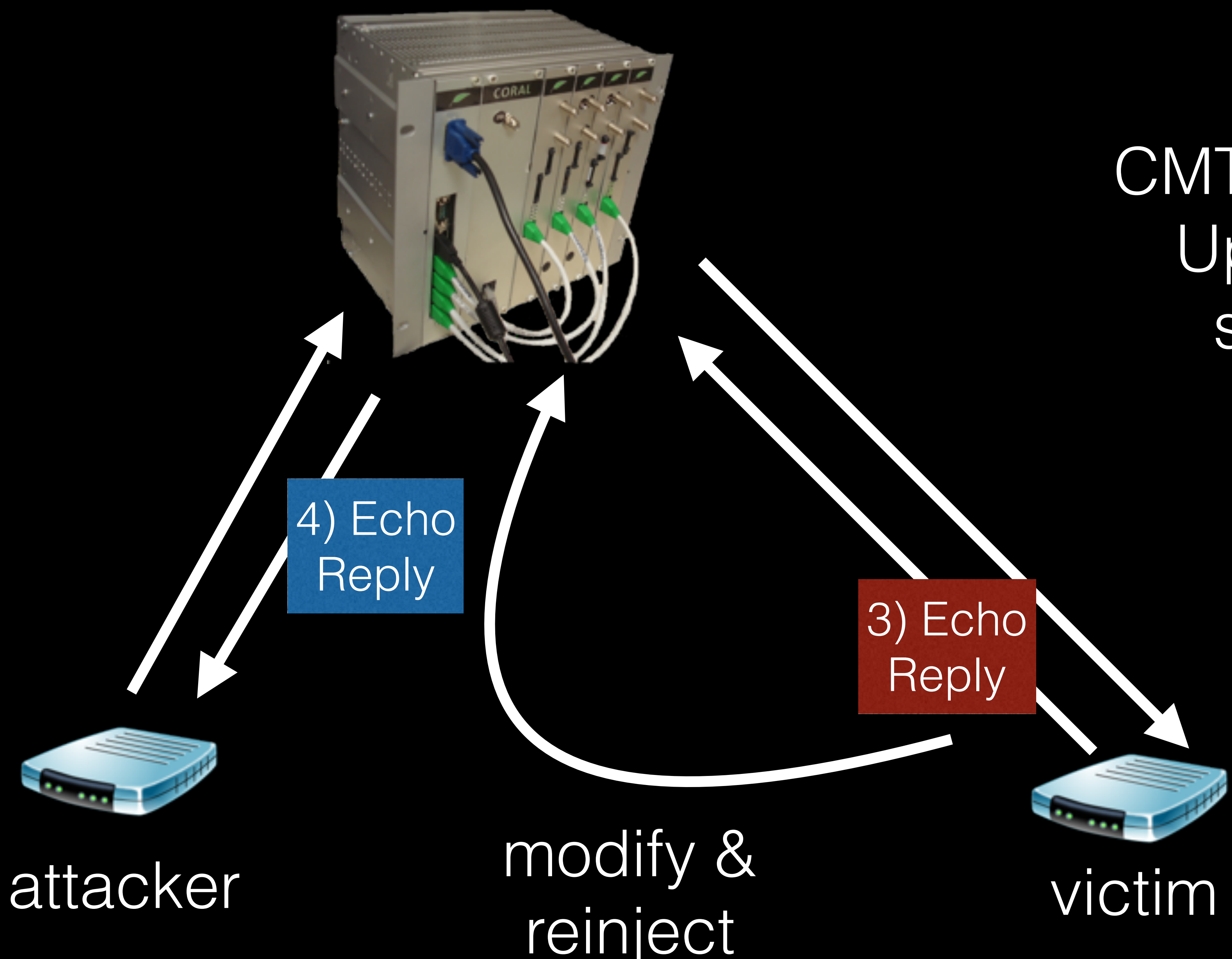


Injection feasibility

- There doesn’t appear to be anything preventing this at the physical layer
- Requires specific hardware
  - QAM modulator
    - Lots of these available on eBay for all kinds of price points, starting at reasonable levels
    - Likely require some level of hacking
  - SDR

Decryption oracle

CMTS Oracle:
Upstream sniffing



CMTS Oracle attack

1. Identify the victims, collect downstream encrypted payloads

2. Send ICMP echo to victim, collect **upstream** ping reply from victim

   - *Requires upstream sniffing capability*

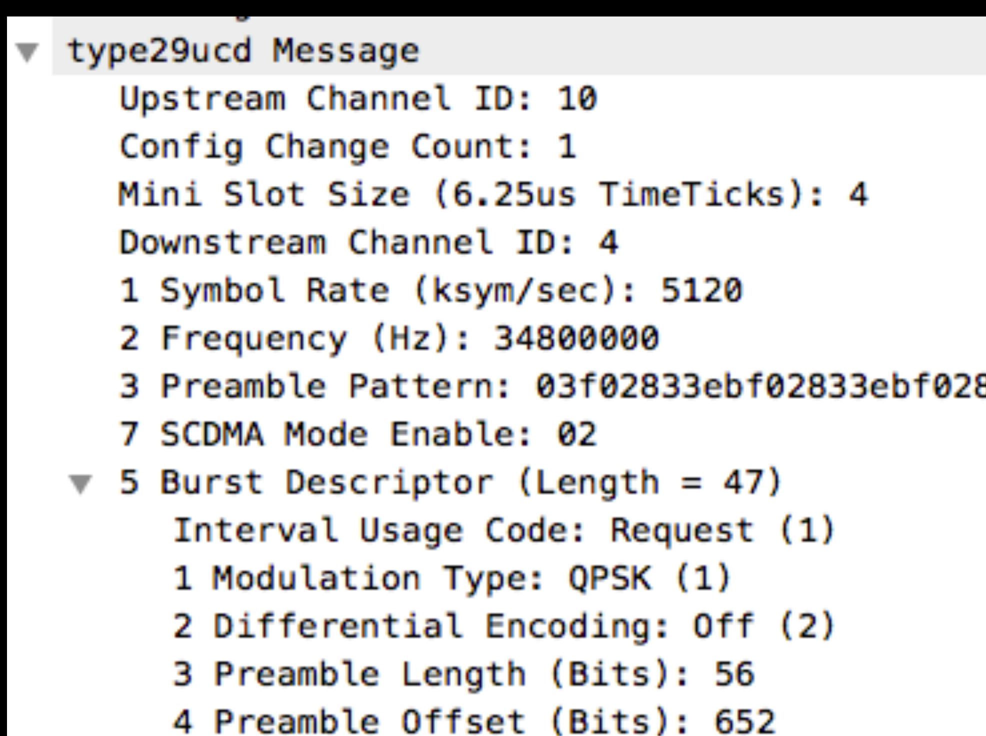
3. Spoofed upstream packet, but splice in desired payload

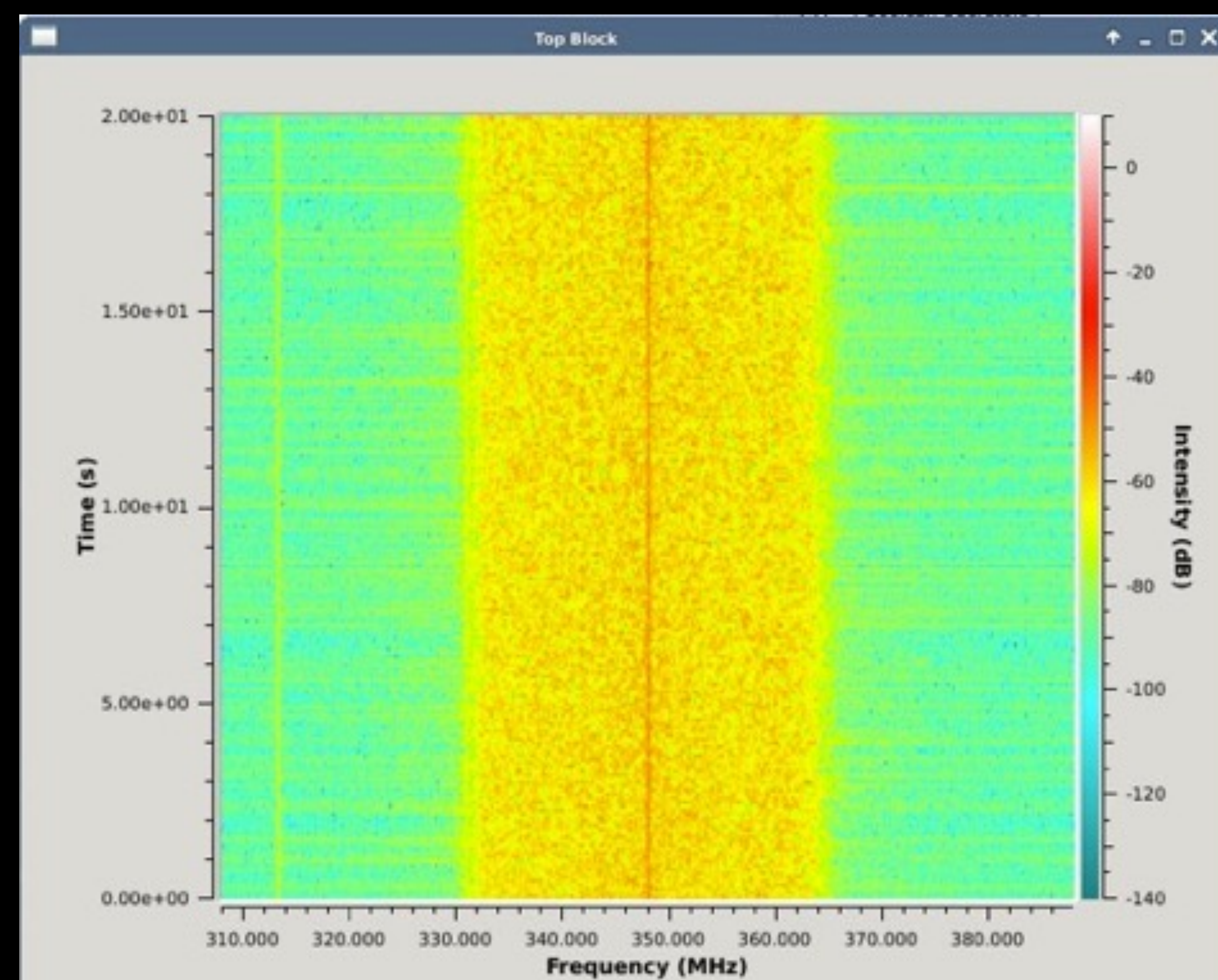
   - *Requires hacked cable modem*

4. Wait for echo response containing cleartext

Upstream sniffing feasibility

- Requires sniffing upstream data
  - Requires sniffing with SDR



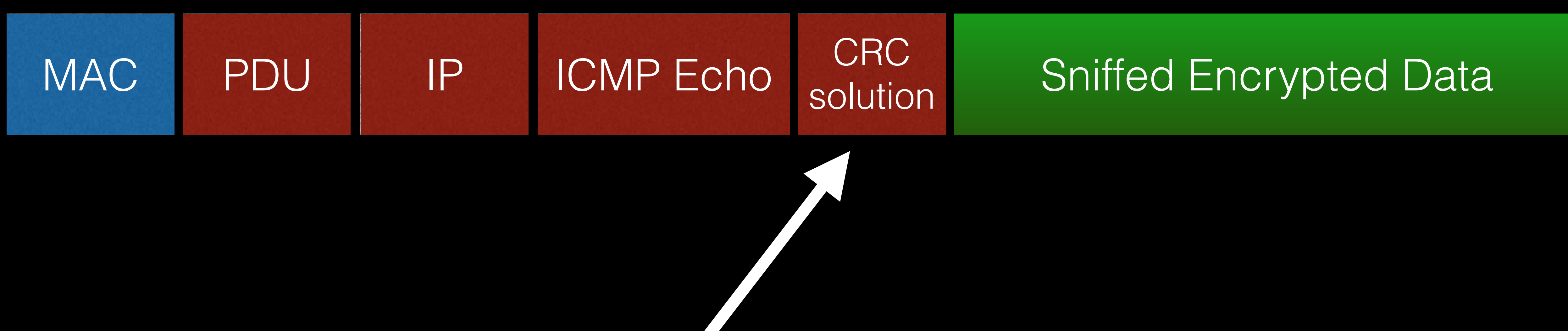


Upstream spoofing on Ubee

- Telnet on 64623 as user/user for root access
- */proc/net/dbrctl/maxcpe*
  - number of CPE devices the modem will forward (default 1)
- */proc/net/dbrctl/addcpe*
  - write another MAC address here to start modem forwarding
- Read current TEK/IV from */dev/mem*
  - decrypt encrypted data with my TEK, expect modem to encrypt it back to original (?)
  - SAID is not in upstream packet, but key version# is

CRC validation

- What about the encrypted CRC in the spliced packet PDU?
  - We don’t know what the correct CRC value is
  - Brute force sucks



4b, “resets” CRC to what it would be after processing MAC header only, so spliced CRC still correct

Conclusions

- Your DOCSIS network is less safe than your wifi network
  - Downstream sniffing is easy and decryption is possible
  - Upstream sniffing is close
  - Active attacks are plausible

Solutions

- Support AES immediately for 3.0 modems
- Support EAE
- Drop ICMP at the CMTS?
  - Consider other traffic types that may be used for oracle
- Add data authentication!

Software Releases

- CableTables software
  - Uses MyGica dongle (or equivalent) to perform DES cracking attack
  - Supports local or cloud-based rainbow tables
  - http://tiny.cc/cabletables
- Cloud DES Rainbow table generation software
  - What I used to generate my DES rainbow tables in EC2
  - What I use for cloud-based cracking
  - http://tiny.cc/cabletables_cloud

Obtaining the tables

- They are available to all AWS users
- Requestor Pays: you just pay for data transfer
  - $0 to US East AWS region
  - 1tb: ~$20 to other AWS region
  - 1tb: ~$90 over the internet to your computer

Q&A