AEAD Ciphers for Highly Constrained Networks

—

René Struik

e-mail: rstruik.ext@gmail.com

Outline

1. Highly Constrained Networks
   – Examples & Use Case Scenarios
   – Constraints
2. Efficient Crypto Constructs
   – AEAD Ciphers
   – Layering Aspects
3. Maintaining State
   – Per-Layer Keys, Nonces, & AEADs
   – “Re-use” Across Layers
4. Implementation Cost
   – Cost of Single Construct
   – Incremental Cost
5. Conclusions & Future Directions

Highly Constrained Networks

- Examples & Use Case Scenarios
- Constraints


Wheeling-Pittsburg Steel Corporation
Photo courtesy Dust Networks

The Promise of Wireless

The Economist, April 28, 2007

Examples of Sensor and Control Networks

- Consumer Electronics
- PC Peripherals, Toys, and Gaming
- Industrial Process Control & Factory Automation
- Smart Metering
- Building Automation & Control (HVAC)
- Supply Chain Management
- Asset Tracking & Localization
- Homeland Security
- Environmental Monitoring
- Healthcare & Remote Patient Monitoring

Catch phrase: “*Internet of Things*”
2008: more “things” connected to Internet than people
2020: est. more than 31B [1] -50B [2] interconnected objects

[1] Intel (September 11, 2011);
[2] Cisco (July 15, 2011);
[3] US DOE Roadmap (2006)

Benefit wireless industrial sensors [3]:
♦ Efficiency gain: 25% ♦ emission reduction: 10% ♦ significant reduction ‘wiring cost’

Wireless Networking Standards

Wireless Local Area Networks (WLANs)
- IEEE 802.11 family (WiFi Alliance)
- Mesh Networking (802.11s)
- Fast Authentication (802.11ai)
- WiFi Alliance

Wireless Personal Area Networks (WPANs)
- 802.15.1 (Bluetooth Alliance)
- 802.15.4 (ZigBee Alliance, Wireless HART, ISA SP100.11a)
- 802.15.6 (“Body Area Networks”)
- Bluetooth ‘Lite’
- Body Area Networks

Networking IETF:
- Routing (RoLL), Applications (CoRE), Home Area Networking (HomeNet)

Other:
- Ubiquitous Computing
- DRM, Networked Gaming
- NFC Forum
- e-Payments

[…]

Constraints (1)

Constraints for Sensor Networks

High throughput is not essential, but rather

- Low energy consumption:
  Lifetime of 1 year with 2 AAA batteries (@750 mAh, 2V) yields 85μA average power consumption, thus forcing 'sleepy' devices (802.15.4 uses 40-60 mW for Tx/Rx)
- Low manufacturing cost:
  Low cost devices force small memory, limited computing capabilities (clock frequency: 4-16 Mhz; 10-32 kbytes ROM, 1-4 kbytes RAM, possibly no flash)

Constraints for Adhoc Networks

- No centralized management:
  No online availability of fixed infrastructure (so, decentralized key management)
- Promiscuous behavior:
  Short-lived communications between devices that may never have met before (so, trust establishment and maintenance difficult)
- Unreliability:
  Devices are cheap consumer-style devices, without physical protection (so, no trusted platform on device)

Constraints (2)

Security Constraints for Adhoc Networks

- Decentralized key management:
  Due to no online availability fixed infrastructure, but also very ‘sleepy’ nodes
- Flexible configuration and trust management:
  Due to promiscuous, adhoc behavior, but also survivability requirements
- Low impact of key compromise:
  Due to unavailability of trusted platform (tamper-proofing, etc.)
- Automatic lifecycle management:
  Due to virtual absence of human factor, after initialization

Security Design Constraints for Sensor Networks

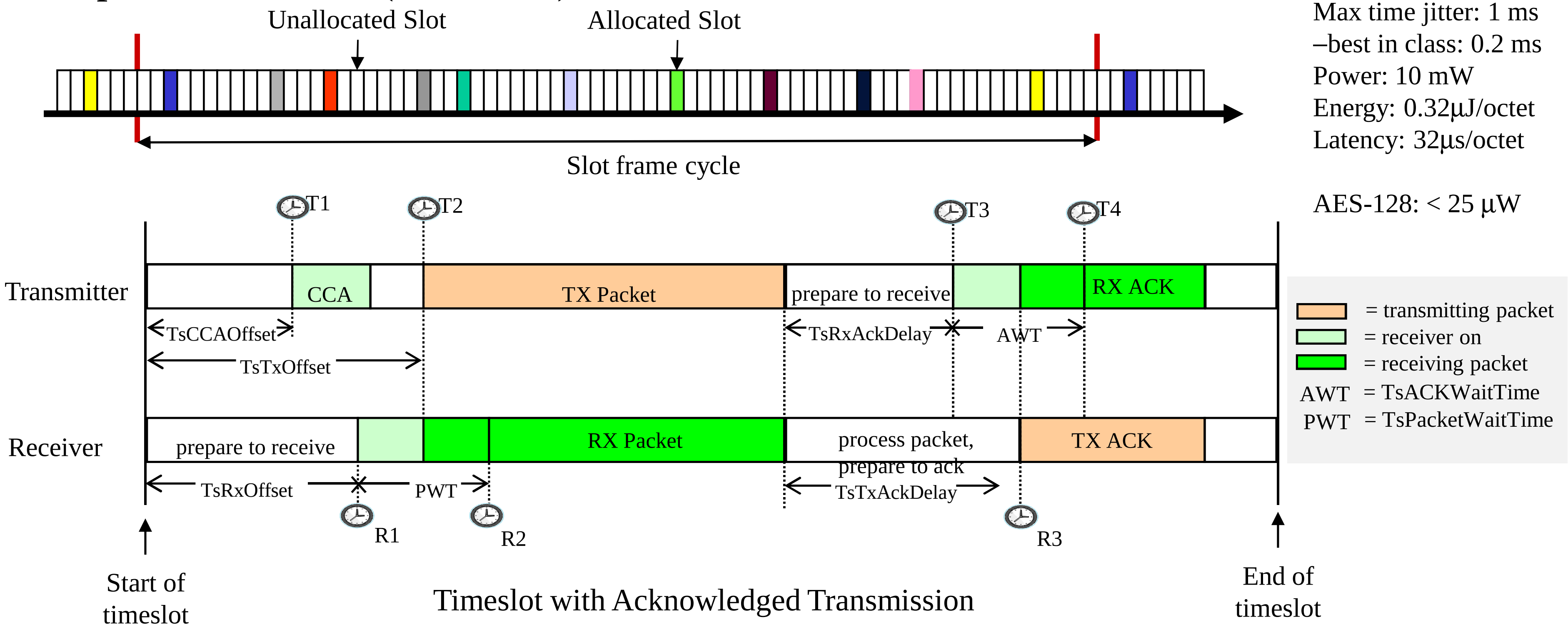
- Implementation efficiency: protocols should use similar cryptographic building blocks
- Parallelism: design protocols have the similar message flows
- Low communication overhead: protocols must avoid message expansion if possible

Efficient Crypto Constructs

- AEAD Ciphers
- Layering Aspects

Communication and Computational Overhead Matters

Example: IEC 62951 (w/HART)



Data rate: 250 kbps
Max time jitter: 1 ms
–best in class: 0.2 ms
Power: 10 mW
Energy: 0.32μJ/octet
Latency: 32μs/octet

AES-128: < 25 μW

Typical frame: 60 octets. Cost: 2,120μs = 200μs (listen) + 1,920μs (60×32μs) = 21.2 μJ
Communication cost savings: 8 octets = 256μs latency=2.56μJ (+14% energy efficiency)
Computational cost (in HW): AES-128 ≈ 0.2μJ

Trade-off: Reduced communication cost ↔ Increased computational cost (& latency)

Light-Weight Crypto Mode of Operation

Are we focusing on the right problem?

Light-weight crypto:

- Focus on low-footprint, low-latency ciphers (Present, Hummingbird, etc.)
- From energy consumption perspective, mode of operation more important

Typical frame: 60 octets. Cost: 2,120μs = 200μs (listen) + 1,920μs (60×32μs) = 21.2 μJ
Communication cost savings: 8 octets = 256μs latency=2.56μJ (+14% energy efficiency)
Computational cost (in HW): AES-128 ≈ 0.2μJ

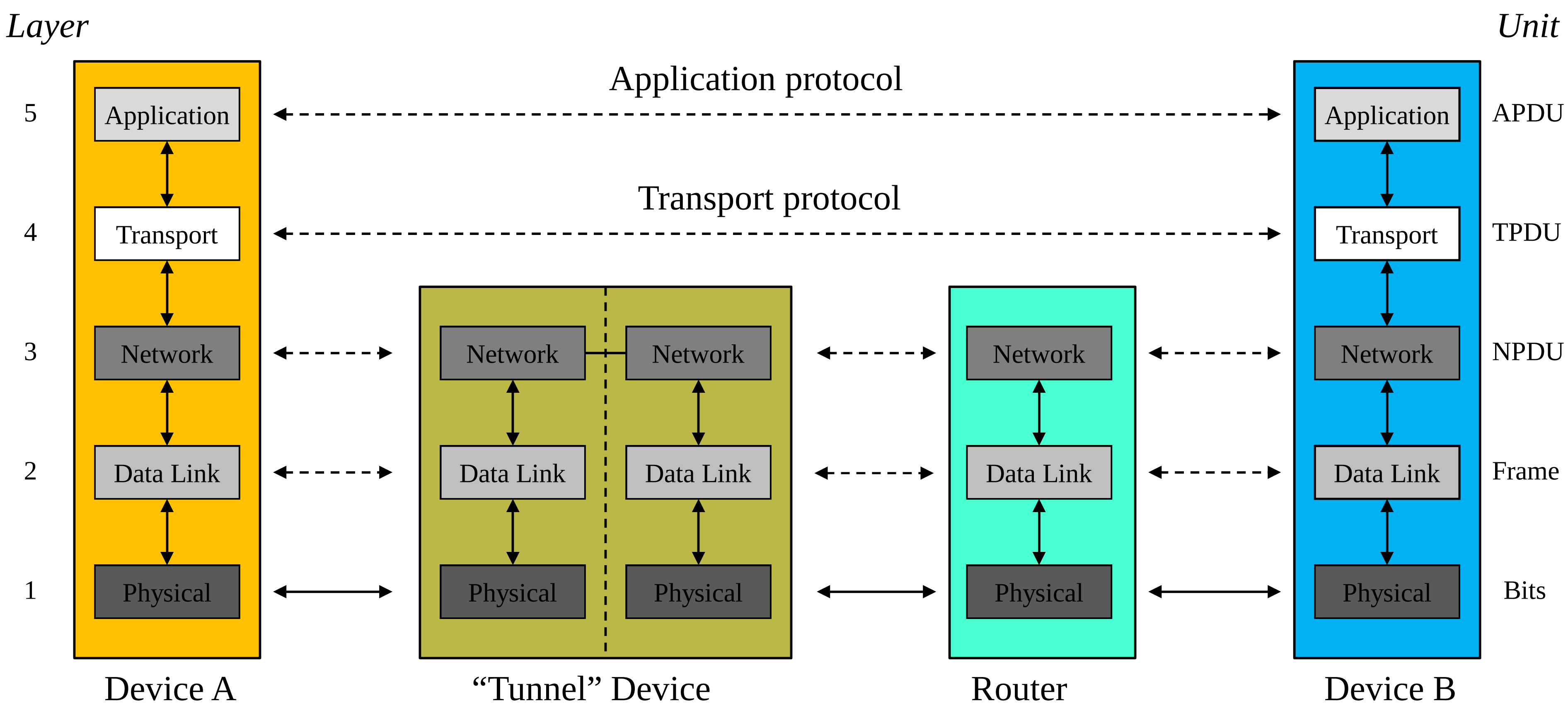
Cost of crypto: 1% of communication cost

Trade-off: Reduced communication cost ↔ Increased computational cost (& latency)

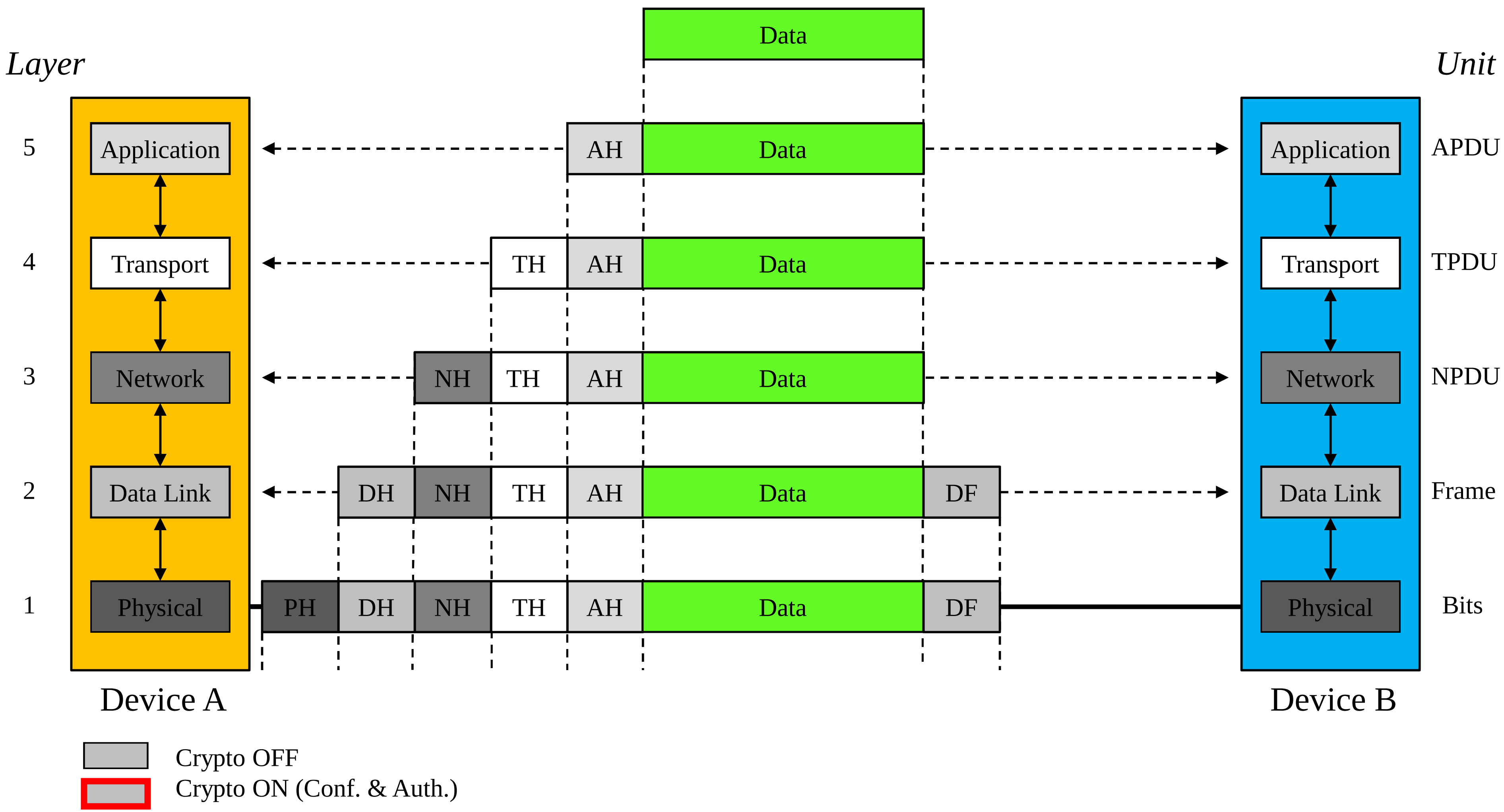
<u>Example:</u>

- Shaving off 8 octets may justify making symmetric-key crypto 10× more expensive

Network Layering, Protocols, Interfaces

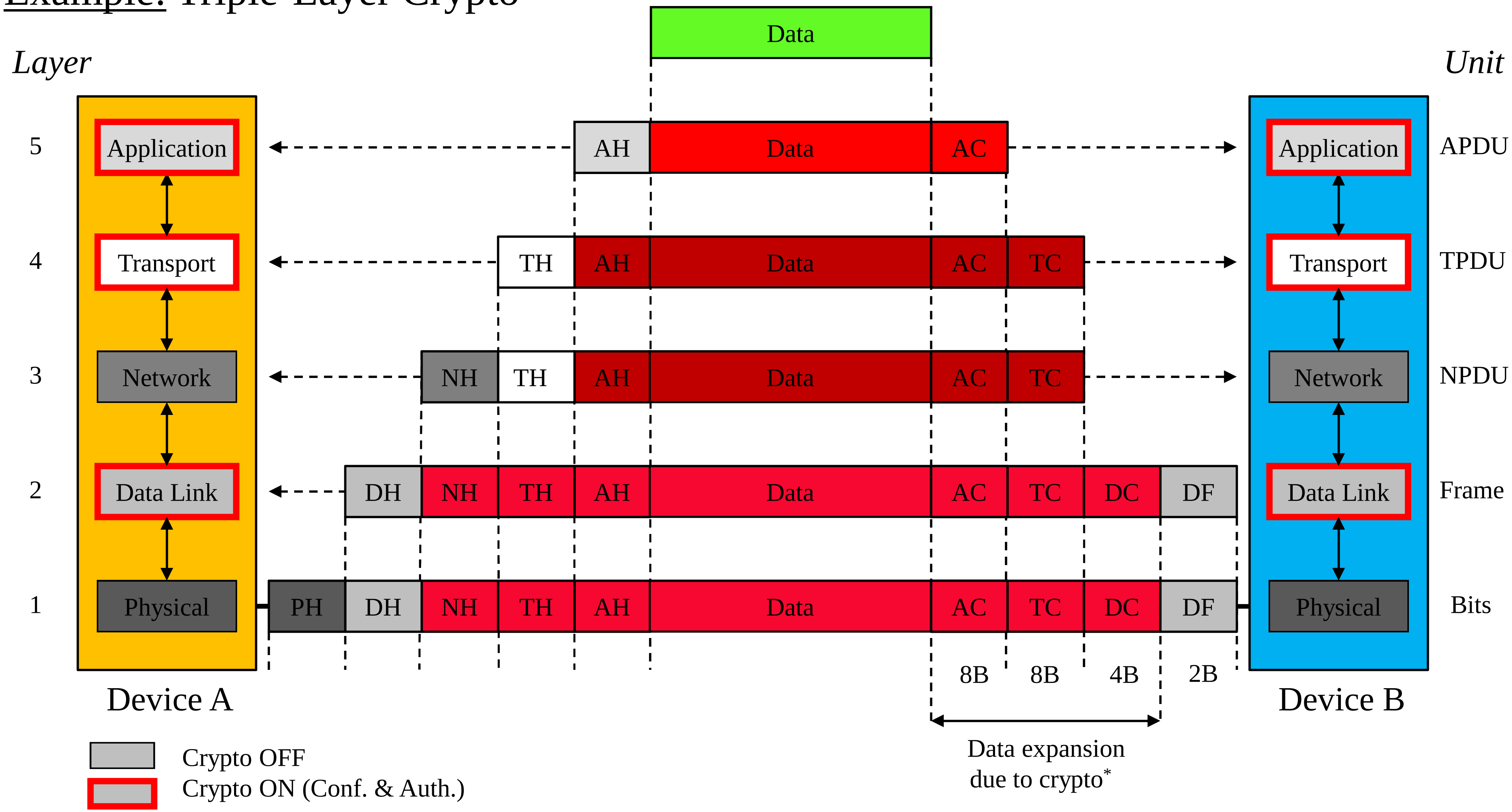


Network Layering, without Crypto



Network Layering, with Traditional Crypto

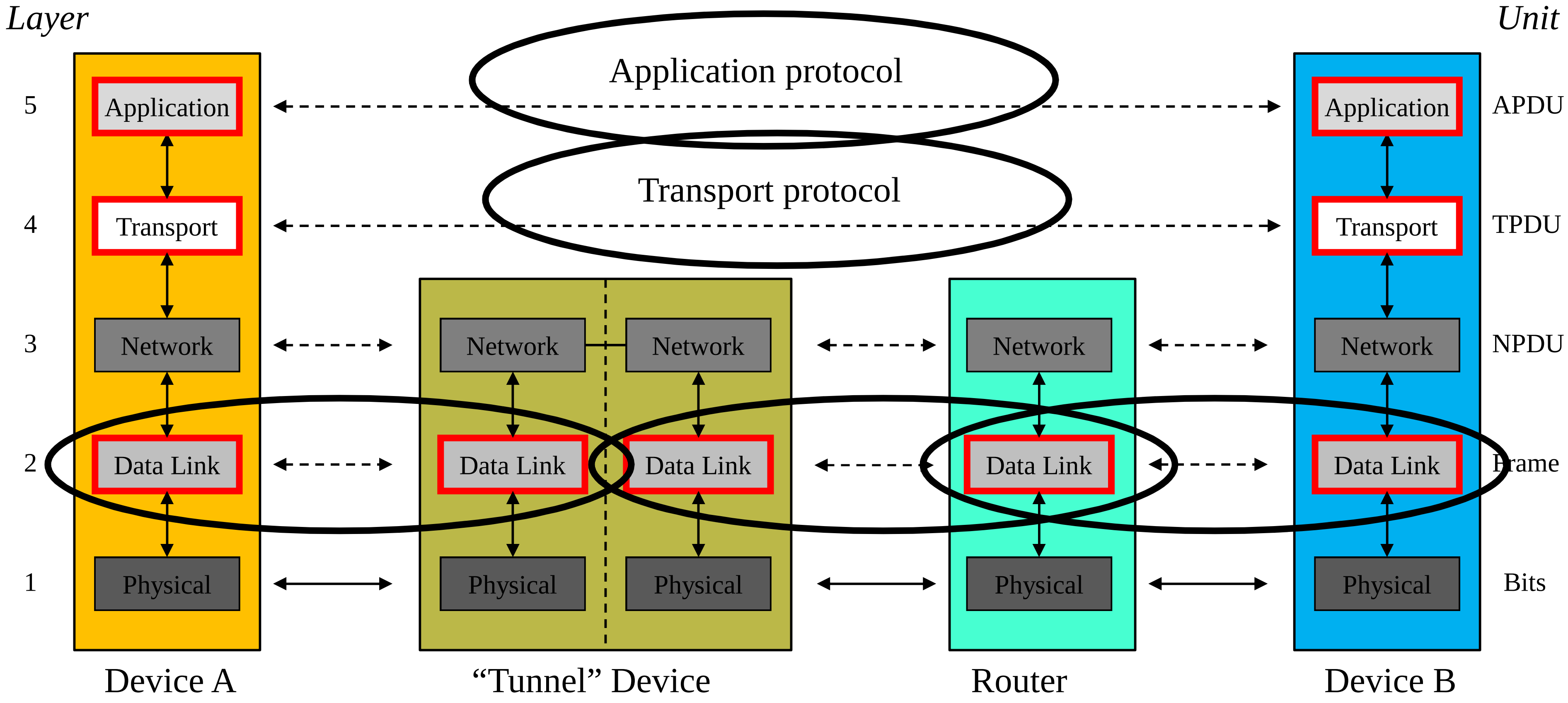
Example: Triple-Layer Crypto



*ignoring security admin in headers

Network Communications, with Traditional Crypto

Example: Triple-Layer Crypto



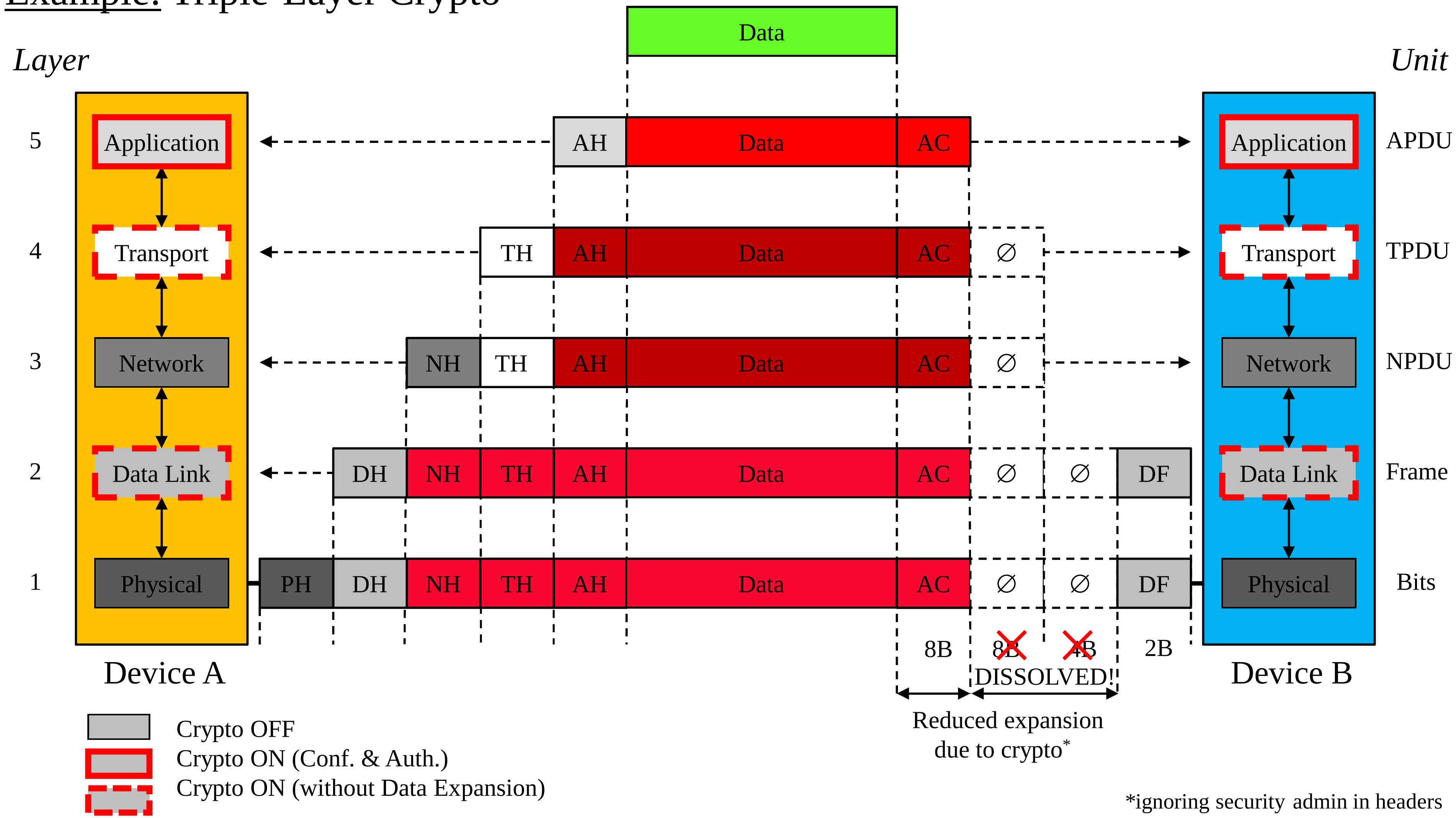
Crypto OFF

Crypto ON (Conf. & Auth.)

All crypto processing yields *explicit verdict* on data authenticity, via Message Authentication Code, but *at a cost* (data expansion)

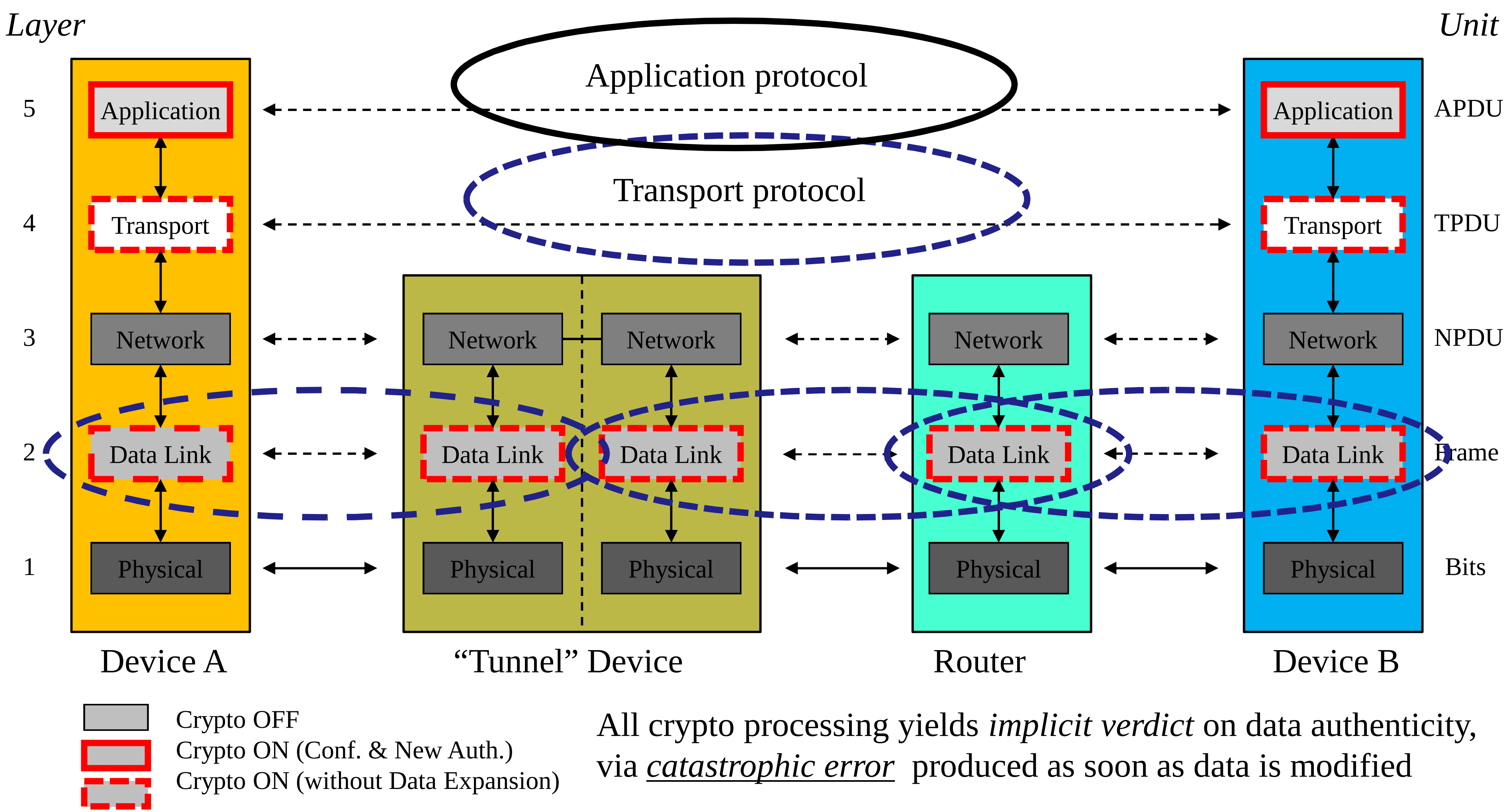
Network Layering, with “NEW” Crypto

Example: Triple-Layer Crypto



Network Communications, with "NEW" Crypto

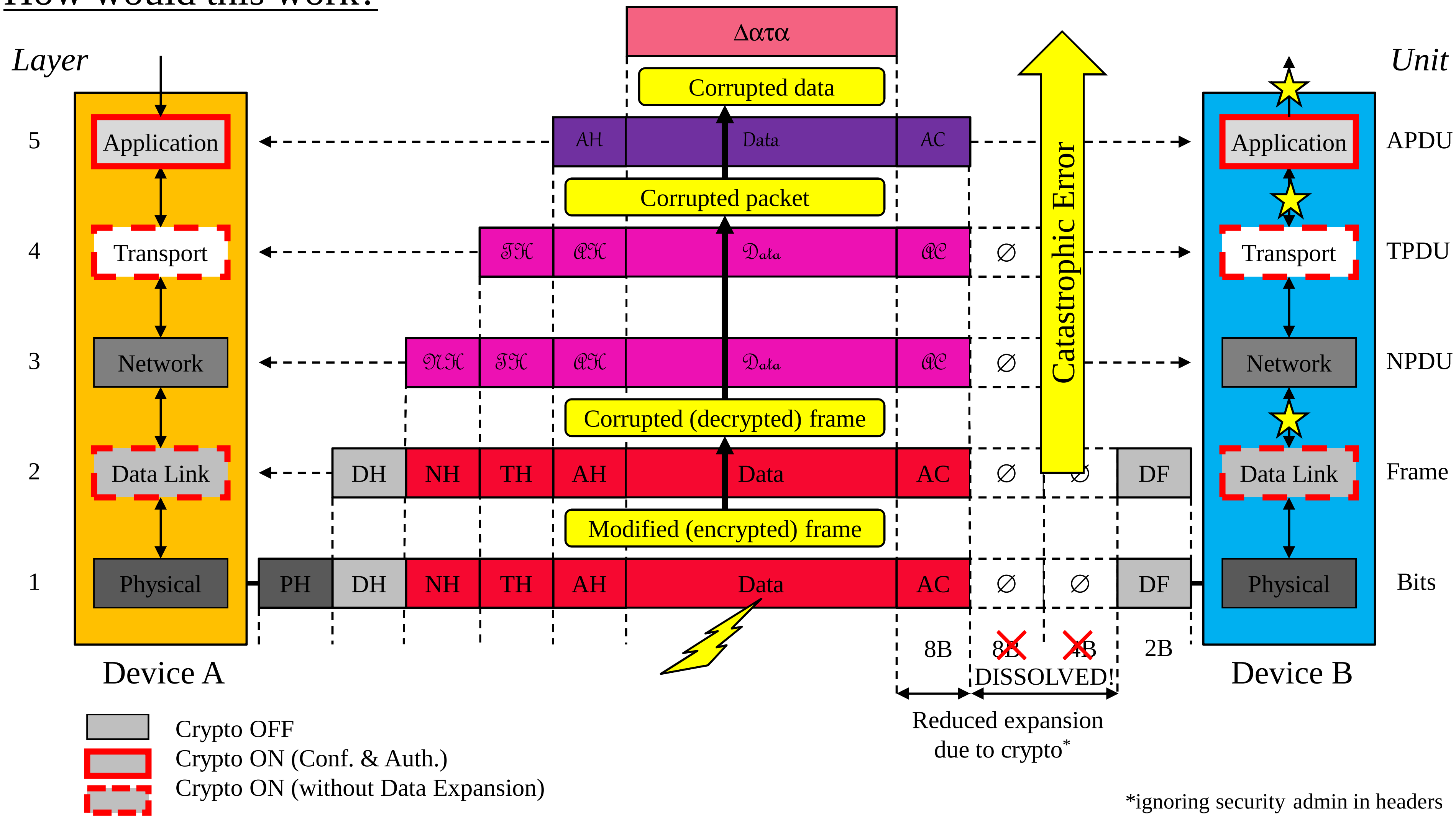
Example: Triple-Layer Crypto



All crypto processing yields *implicit verdict* on data authenticity, via *catastrophic error* produced as soon as data is modified

Incoming Processing, with “NEW” Crypto

How would this work?



*ignoring security admin in headers

"New" Crypto Mode of Operation

Applications to cryptographic protocol layering

- Significant reduction in cryptographic data expansion at lower layers
- No[1] cryptographic rejection of modified packets "in flight"
- Still possible to reject corrupted packets "in flight", if protocol layers have built-in redundancy that can easily be checked (usually the case, due to header info, etc.)

Example: ZigBee *per-packet* Security Overhead Reduction

Total security expansion ZigBee: 34 octets = 22 (NWK layer) + 12 (APL layer)

- Reduction of per-packet crypto/security overhead, to *at most* 8 octets in total only
- Potential for significant other header overhead reduction (non-security-related)

Much more payload data left for application data (≈50% more, without fragmentation)

Caveat: Cannot be realized with existing CCM* mode of operation implementation

Other applications: "storage encryption", "key wrap"

Cryptographic property: Encryption with Authenticity from Redundancy in Plaintext

Requirements: (a) Works also with tiny plaintext; (b) Respects existing hardware

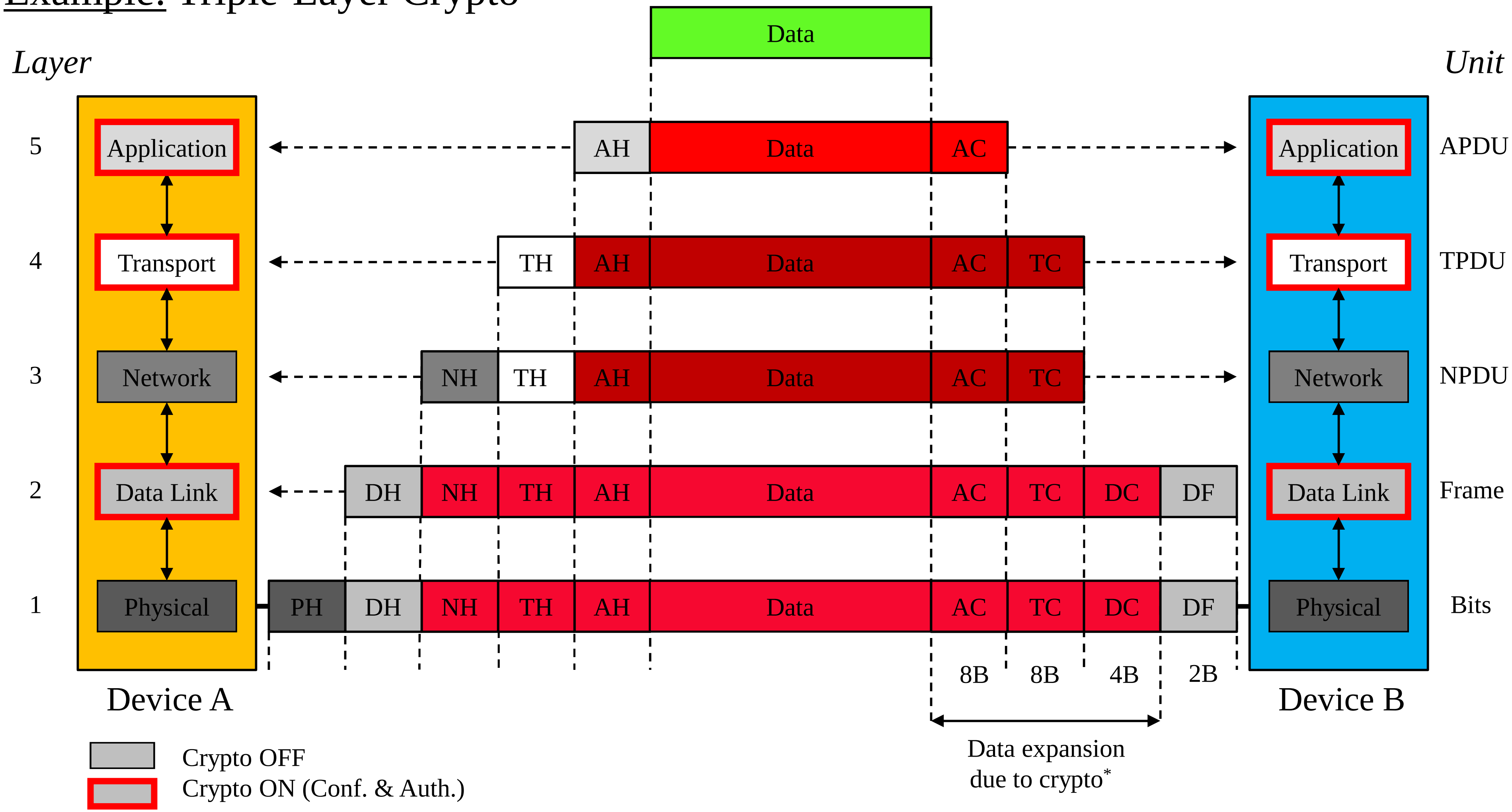
[1] Some cryptographic rejection possible, if some redundancy sprinkled-in (e.g., by padding with fixed 16-bit string)

Maintaining State

- Per-Layer Keys, Nonces, & AEADs
- “Reuse” Across Layers

Network Layering, with Crypto Modes of Operation

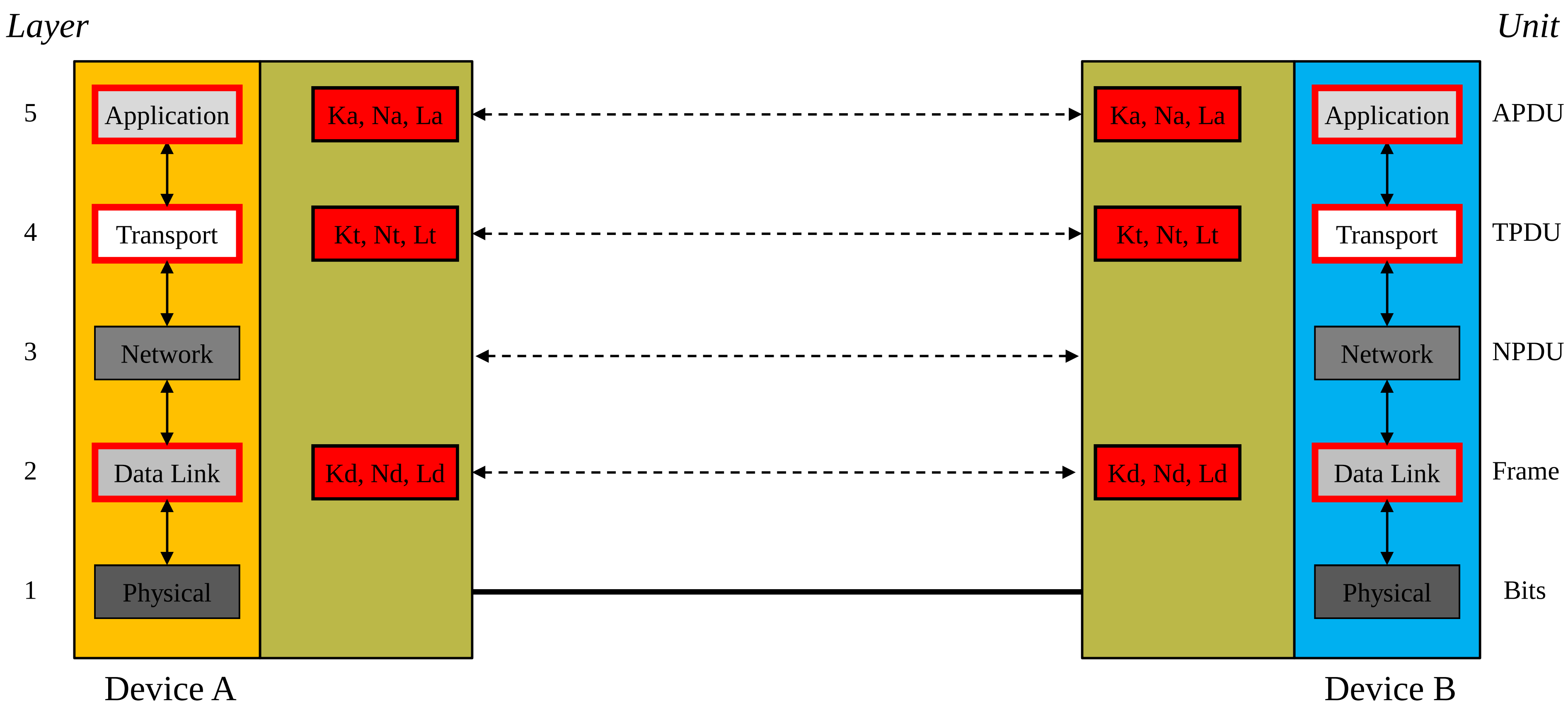
Example: Triple-Layer Crypto



*ignoring security admin in headers

Network Layering, with Traditional Layering of Keying Material

<u>Example:</u> Triple-Layer Crypto



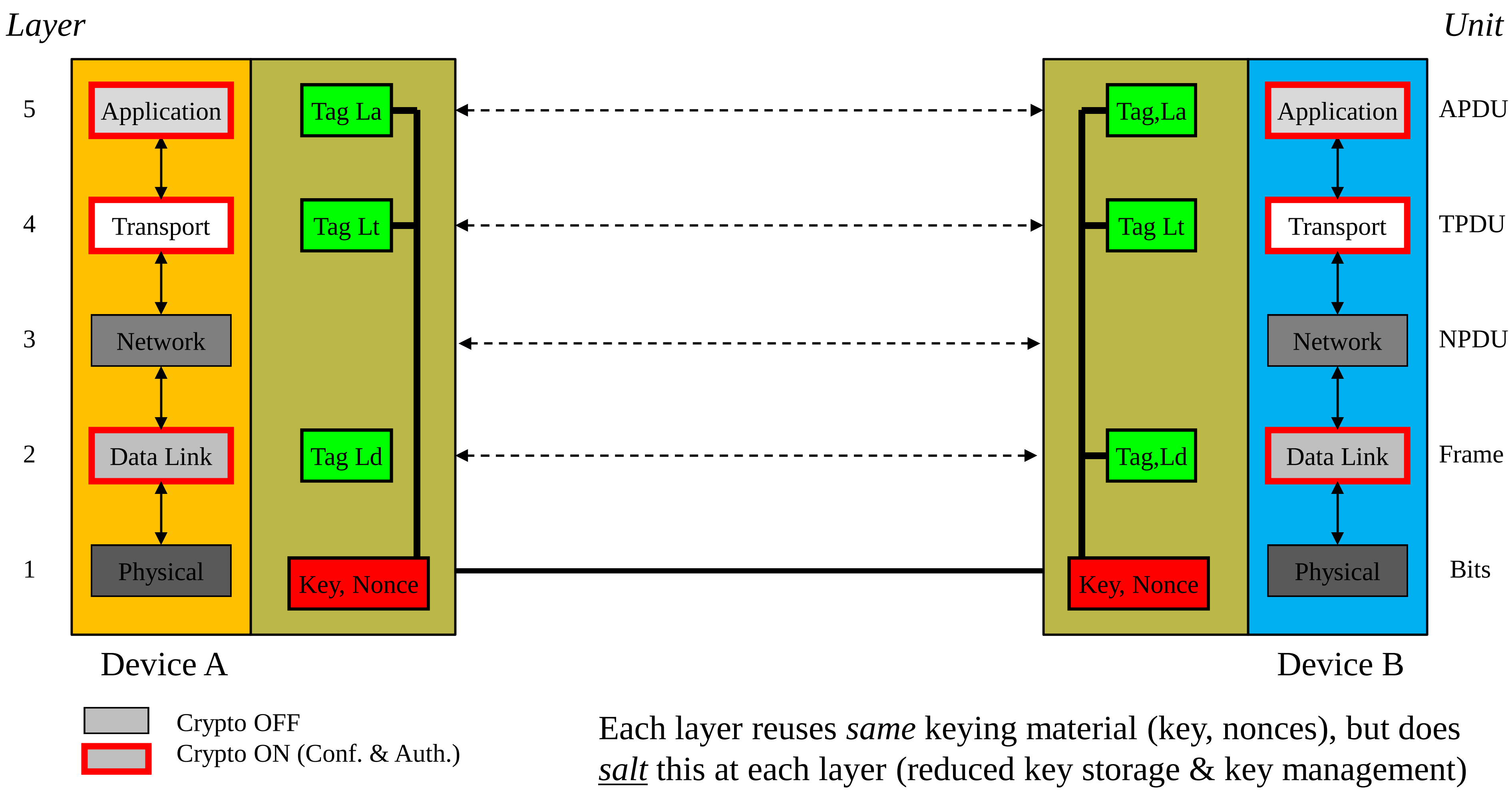
Crypto OFF
Crypto ON (Conf. & Auth.)

Each layer has its own keying material (key, nonces), but this comes *<u>at a cost</u>* (replication of key storage, key management)

*ignoring security admin in headers

Network Layering, with Light-Weight Layering of Keying Material

Example: Triple-Layer Crypto



Each layer reuses *same* keying material (key, nonces), but does *salt* this at each layer (reduced key storage & key management)

Light-Weight Layering of Keying Material

Applications to cryptographic protocol layering

- Keying material (keys, nonces) stored on per-device level, _*not*_ on per-layer level
- Re-use of _*same*_ keying material and *same* AEAD across layers, with per-layer "salting" of AEAD instantiation

<u>Example:</u> OCB mode with variable-size authentication tags:

– OCB w/ 128-bit tag: Nonce128 = (tag128 || Nonce)

– OCB w/ 64-bit tag: Nonce64 = (tag64 || Nonce)

Note: See IETF CFRG draft draft-cfrg-ocb-03 (with cautionary language…)

Cryptographic property: Instantiation of "salted" AEAD modes has same effect *as if* logically distinct keying material and AEAD parameters used at each layer

<u>Requirements:</u> (a) Small-size "Salt"; (b) "Salting" cheap (compared to, e.g., hashing)

Implementation Cost

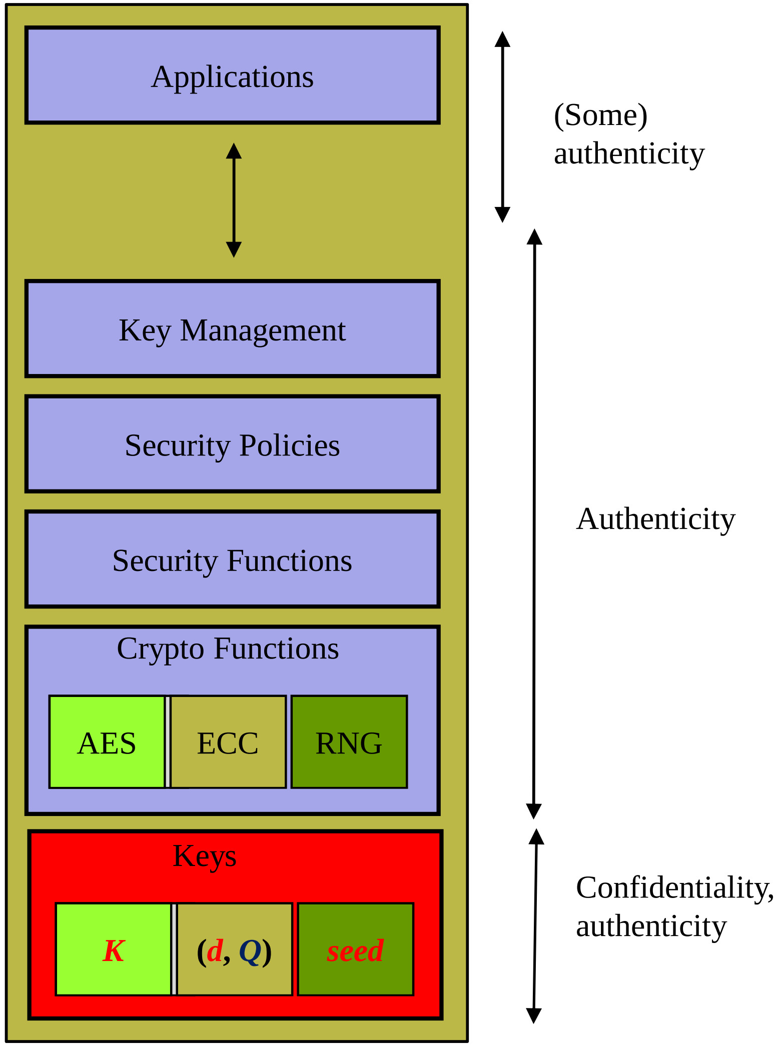
– Cost of Single Construct
– Incremental Cost

Putting Trust in Devices

Conventional Approach

- Trusted implementation of crypto, including side channel resistance
- Trusted security policy routines
- Secure and authentic key storage
- Secure RNG (or RNG seed)

1. Borrow/steal across layers:
   – "Reuse" crypto primitives
   – "Reuse" keying material
2. Borrow/steal functionality other constructs:
   – Intel PCLMULQDQ Instruction
   – Non-crypto support on module
3. Exploit trade-offs:
   – Energy cost computation, communication



Device

Objective: Best *overall* 'fit', *not* per construct

Conclusions & Future Directions

Conclusions & Future Directions

Light-Weight Crypto:

- Performance Crypto Mode of Operation is right metric, *not* Crypto Cipher
- Energy cost *very* important (e.g., in energy harvesting applications)
- Crypto cost should *not* ignore cost of data expansion (in small packet deployments)
  - Authentication tags may be “evil” (authenticity is *not*)

Constrained Devices:

- Focus on performance individual construct (e.g., “*need for speed*”) less important in constrained networks; holistic/system-wide performance is right metric
- Reuse, reuse, reuse… amongst crypto constructs, keying material, stack layers, …

Be aware of eco-system that is under development (IETF 6lowpan, roll, core, dice)

Collaboration? Happy to!

I have worked on ciphers for constrained networks, but still *lots of work remaining*

- Better efficiency, simple proofs, algorithmic tricks, real implementations
- Both inside/outside CAESAR competition

Further Reading

Cryptographic Modes of Operation:

1. P. Rogaway, M. Bellare, “Encode-then-Encipher Encryption: How to Exploit Nonces or Redundancy in Plaintexts for Efficient Cryptography,” in *AsiaCrypt’00*, T. Okamoto, Ed., Lecture Notes in Computer Science, Vol. 1976, Springer, 2000.
2. J.H. An, M. Bellare, “Does Encryption with Redundancy Provide Authenticity?,” in *EUROCRYPT’01*, B. Pfitzmann, Ed., Lecture Notes in Computer Science, Vol. 2045, pp. 512-528, Springer, 2001.

Finite Field Arithmetic:

3. S. Gueron, M.E. Kounavis, “Carry-Less Multiplication and Its Usage for Computing The GCM Mode,” softwarecommunity.intel.com, No. 3787, April 11, 2008.
4. J. Taverne, A. Faz-Hernández, D.F. Aranha, F. Rodríguez-Henríquez, D. Hankerson, J. López, “Software Implementation of Binary Elliptic Curves: Impact of the Carry-less Multiplier on Scalar Multiplication,” IACR ePrint 2011-170.