Kevin Driscoll

Real-Time and Retrofit Cryptography for the Grid

With Supervisory Control and Data Acquisition (SCADA) System Examples

Honeywell

Outline / Overview

- Real-time and/or retrofit cryptography requirements are different from existing cryptography requirements
  - Current crypto algorithms are ill-suited for real-time / retrofit
- Some of our developments
  - A symmetric encryption algorithm (called BeepBeep) that overcomes the problems with using existing cryptography for real-time systems
  - A small cryptography module that is easy to insert into existing communication lines
  - Tamper resistance for embedded software
  - Broadcast / multicast command authentication that uses very little computation and communication resources compared to using a message authentication code (MAC) or public key cryptography

Why is real-time cryptography different?

Can have several orders of magnitude difference in resources/constraints.

	General IT Computing	Embedded Real-Time
Processor	general purpose CPU	microcontroller and/or DSP
Memory **	GB	kB - MB
Power *	100 watts	100 mW (e.g., dongle)
Network bandwidth **	Gbps	kbps
Payload variability **	extremely high	very limited variability in rates and sizes
Payload size *	50 Kbytes ****	20 bytes
Latency and jitter *	seconds	milliseconds***
Integrity importance	less than privacy	greater than privacy
Physical security	human attended	not human attended
Net membership	open / anybody (generalize)	closed (specialize, optimize)

* Difference of about 1,000:1

** Difference of about 1,000,000:1

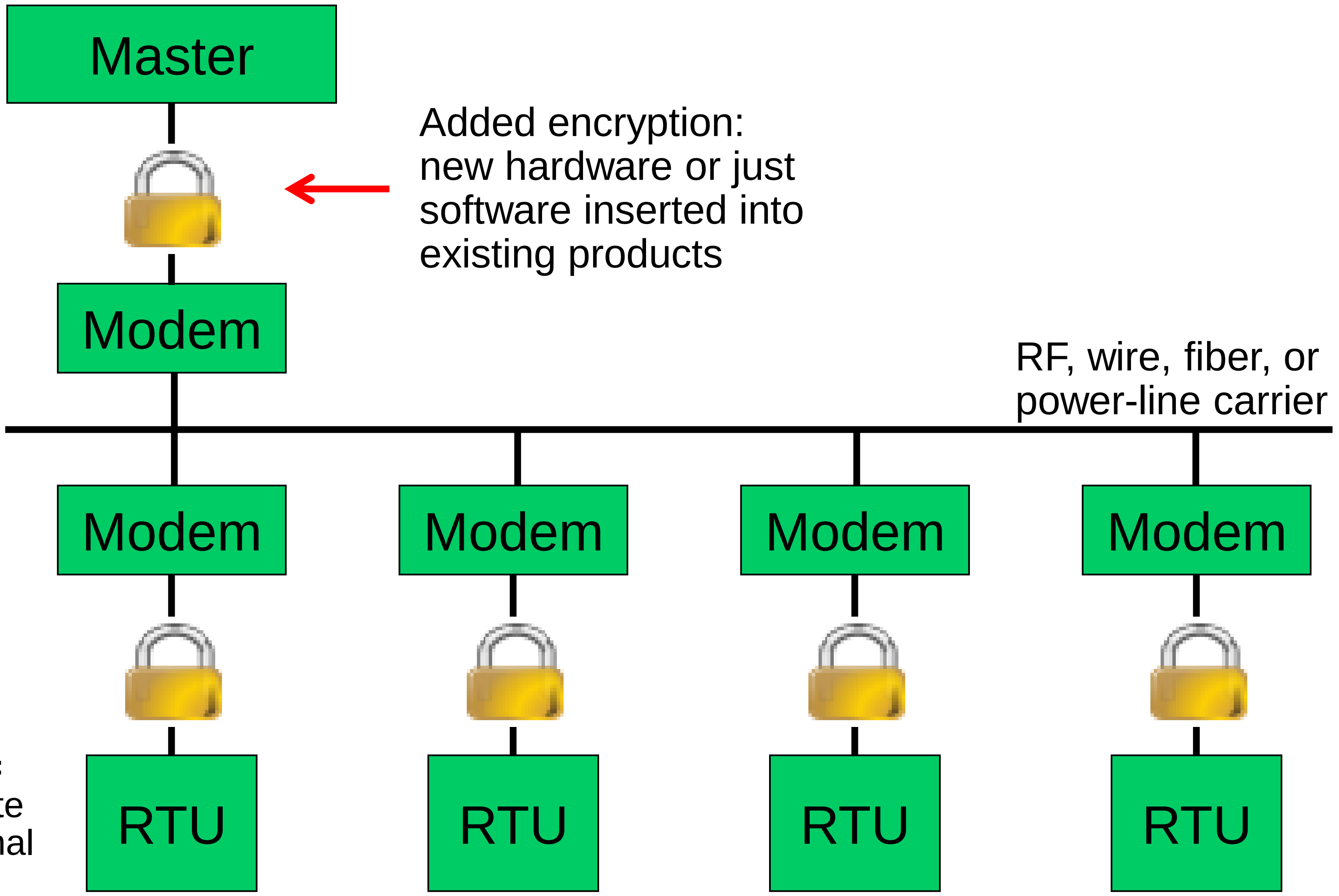
*** Can be due to legacy polled communication constraints (where latency and jitter can accumulate), and/or control loop constraints

**** Mean of my email in May

Existing Cryptography Real-Time Problems

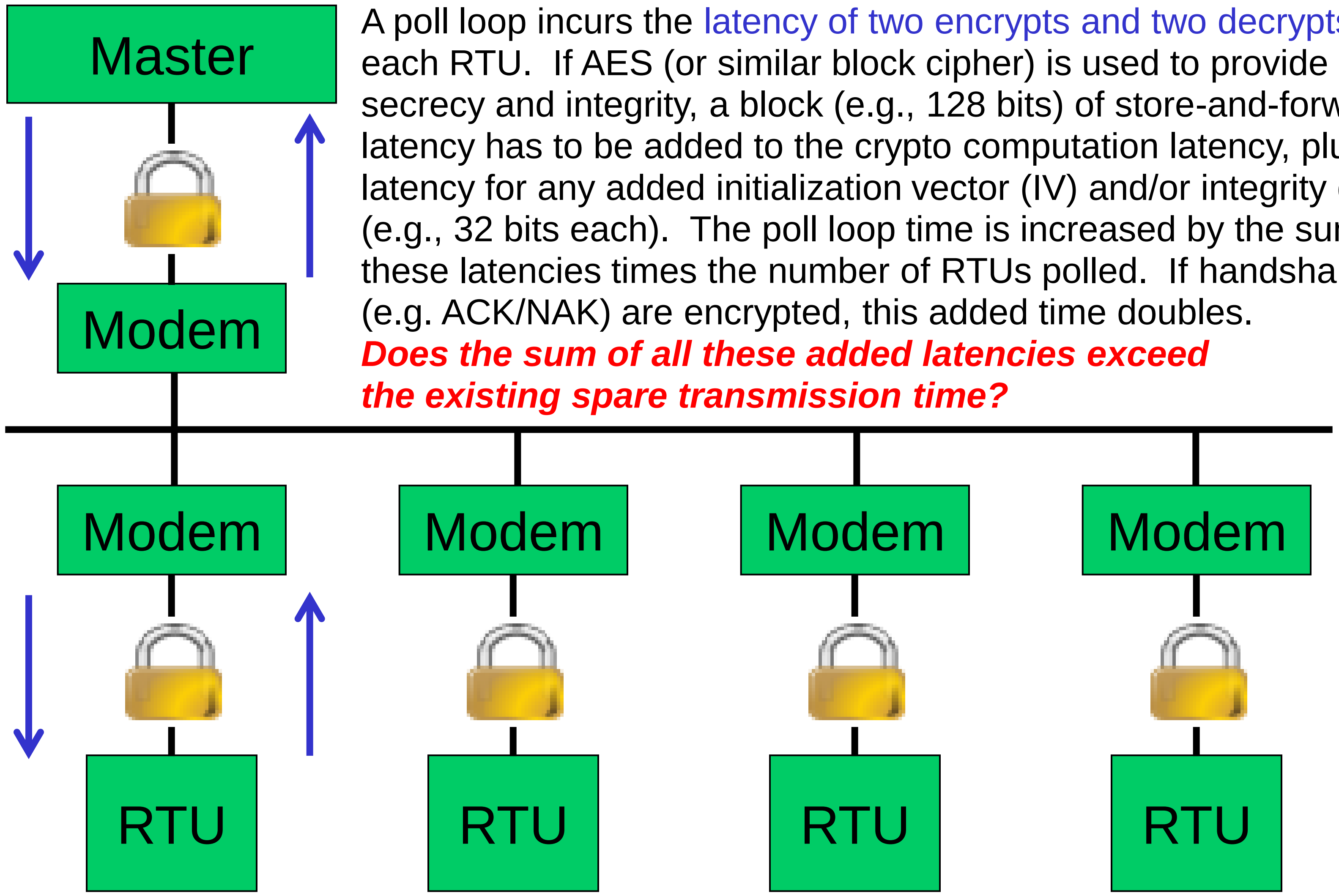
- Slow startup for each key change (due to “key scheduling” being done)
  - Messages (and sessions) are small, less data to amortize startup cost over
  - Latency (delay) and jitter are usually more important than throughput
  - Only worst case timing counts, average is unimportant
    - A missed deadline not helped by finishing early at all other times
    - Real-time control systems often use repeating execution time slots of fixed size
      - If startup slot is longer than average, all time slots must reserve wasted time
  - May need to change key for each message (for unattended remote units)
- Uses too much data memory (cache misses hurt performance)
  - Real time systems are multitasking with many context switches / sec that cause a task’s cache entries to be evicted (replaced with other tasks’ data and instructions)
  - To guarantee timing, one must assume most memory accesses cause cache misses
  - But, crypto performance info (propaganda) assume a “pre-warmed” cache
- Needs more communication bandwidth than may be available
  - Adds: initialization vectors, integrity check data, pads, key management, …
- Uses separate integrity algorithms or integrity mode wrapping
  - Makes execution even slower and uses more power
  - The added latency can preclude “lump in the cable” retrofits
- Many new real-time cryptography installations will be retrofits, which further exacerbates the above problems

Latency Problem for Polled Systems (e.g., SCADA)


Master
Added encryption: new hardware or just software inserted into existing products
Modem
RF, wire, fiber, or power-line carrier
Modem
Modem
Modem
Modem
RTU
RTU
RTU
RTU

RTU = Remote Terminal Unit

Latency Problem (continued)



A poll loop incurs the latency of two encrypts and two decrypts for each RTU. If AES (or similar block cipher) is used to provide secrecy and integrity, a block (e.g., 128 bits) of store-and-forward latency has to be added to the crypto computation latency, plus the latency for any added initialization vector (IV) and/or integrity data (e.g., 32 bits each). The poll loop time is increased by the sum of these latencies times the number of RTUs polled. If handshakes (e.g. ACK/NAK) are encrypted, this added time doubles.

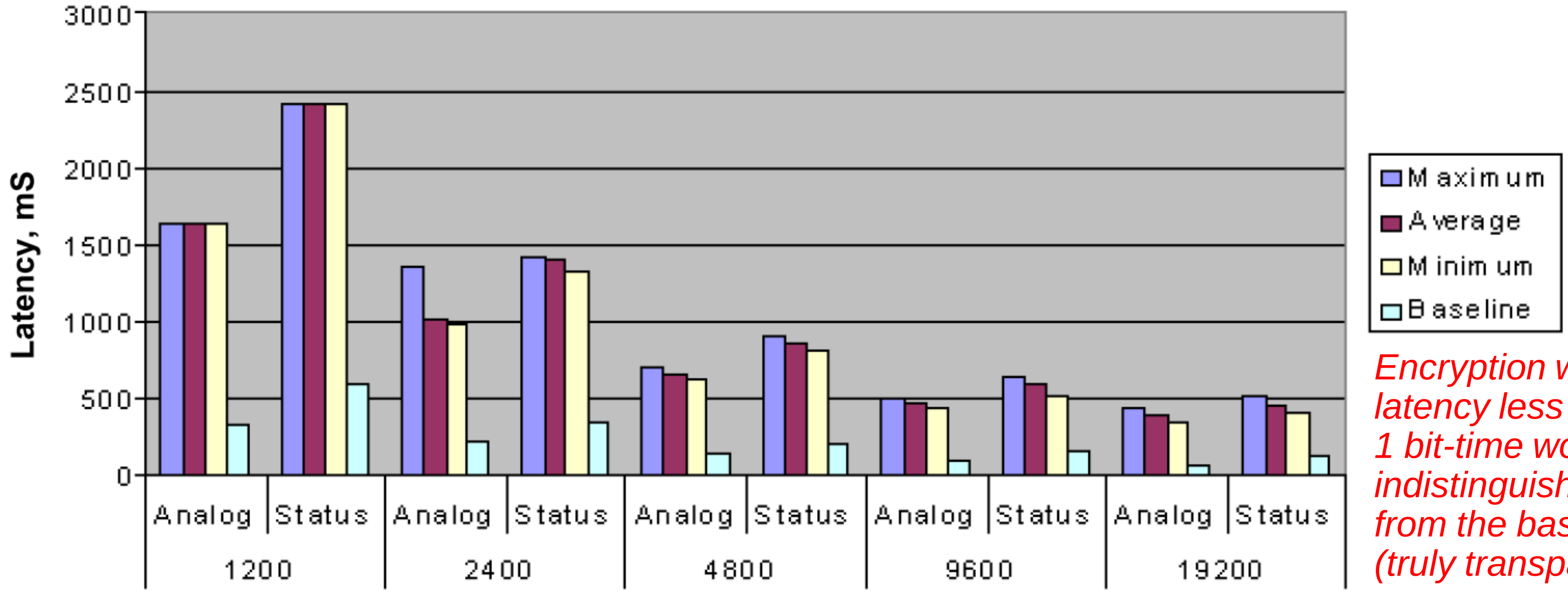
Does the sum of all these added latencies exceed the existing spare transmission time?

Honeywell

→ Honeywell.com

Latency Problem (continued)

- The chart below shows the Max, Ave, and Min per-message latency for two message types at different communication speeds
- The baseline is the latency without any encryption
- At 1200 baud, the time added to the poll loop duration due to the combined SSCP crypto latencies for just the 15 status messages is about 15 * (2.4 – 0.6) = 27 sec (almost half the entire 60 sec budget)
- ***To avoid exceeding the timing budget, this budget would have to have almost 50% spare not allocated to other future growth***



Encryption with latency less than 1 bit-time would be indistinguishable from the baseline (truly transparent)

Crypto Attacks on Real-Time Systems

- Direct cryptanalytic attacks are difficult
  - **Chosen plaintext** (fool system into encrypting messages of your choosing)
    - Requires such invasive physical access (e.g., insider attack) that it would be easier just to read out the key(s) (unless strong anti-tamper measures are used) or to bypass the encryption. Bandwidth too low to generate many chosen-plaintext / ciphertext pairs
    - *However, need a design where loss of one key doesn't compromise the entire network*
  - **Chosen ciphertext** (inject fake encrypted messages and watch response)
    - Communication / crypto integrity mechanism(s) detect and reject most forgeries
    - Existing fault-tolerance mechanisms reject most undetected forgeries
      - e.g. select-before-operate (SBO) protocols
    - Bandwidth is so low that only a small number of messages could be sent before source is detected (by monitoring and alerting service)
  - **Known plaintext** (includes correctly guessed plaintext)
    - Bandwidth too low to get many plaintext / ciphertext pairs
- Security time horizon is usually "tactical" vs "strategic"
  - Secrecy (depends on type of data)
    - Control – up to a few hours
    - Inventory – for a few weeks or months
    - IP (e.g. recipes) – for a few decades (but, very rarely done)
  - Integrity needed only until next key change
- Need only modest strength against cryptanalysis
  - *This doesn't mean cryptographic security is unimportant*

Honeywell

→ Honeywell.com

Outline

- Real-time / retrofit cryptography requirements are different from existing cryptography requirements
  - Current crypto algorithms are ill-suited for real-time / retrofit
- Some of our developments
  - A symmetric encryption algorithm (called BeepBeep) that overcomes the problems with using existing cryptography for real-time systems
  - A small cryptography module that is easy to insert into existing communication lines
  - Tamper resistance for embedded software
  - Broadcast / multicast command authentication that uses very little computation and communication resources compared to using a message authentication code (MAC) or public key cryptography

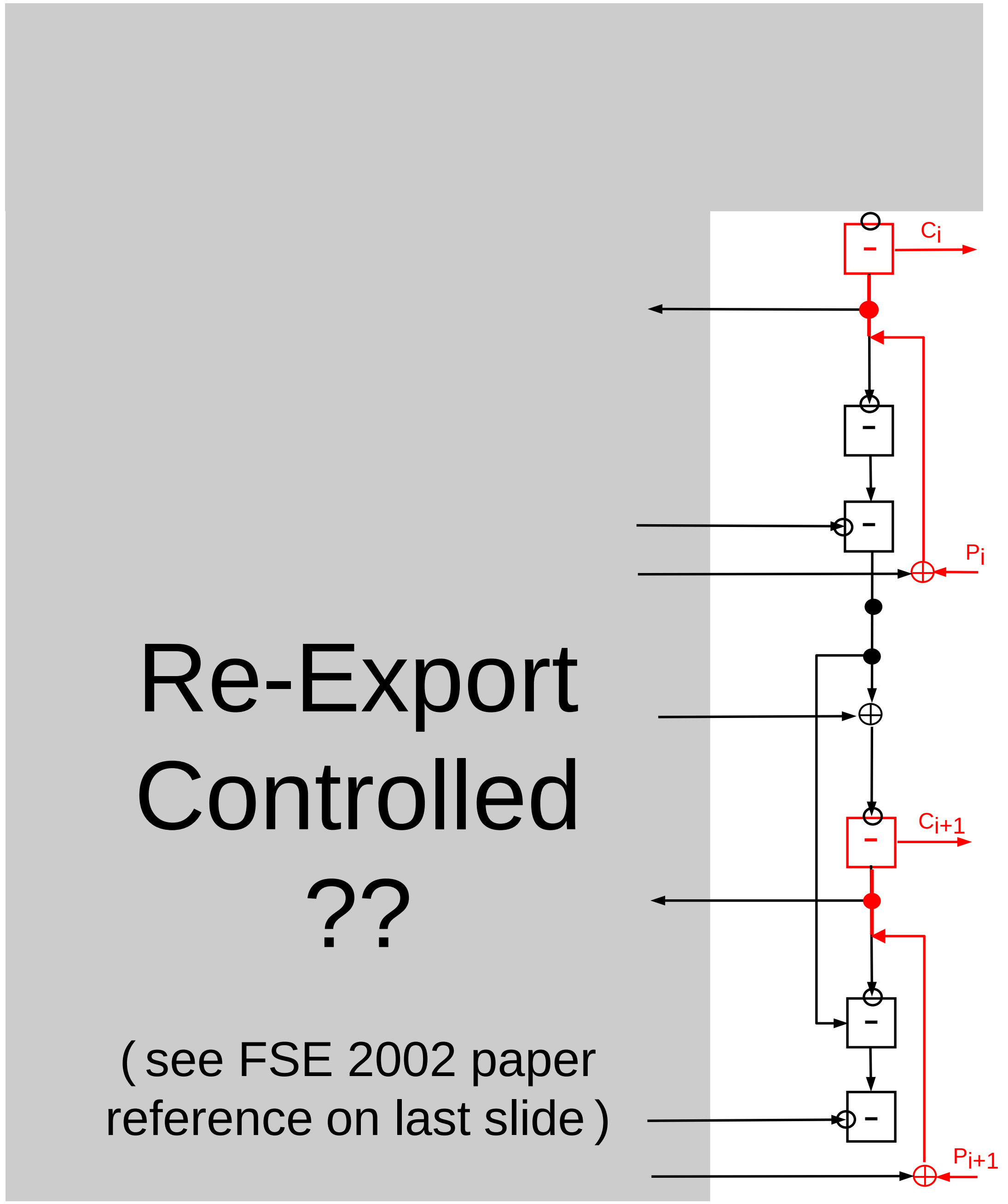
BeepBeep Benefits

- Speed can be better than a bit per CPU clock, memory performance permitting
  - BeepBeep speed is often limited by I/O memory speed
    - ***Hardware-assisted encryption (e.g., for AES) can't be faster***
- Much lower latency and jitter than other algorithms
- Less than half the memory size of other algorithms
  - For software that provides secrecy and integrity
- Includes integrity with secrecy in one pass over the data
  - Allows "lump in the cable" (e.g., dongle) implementations
    - With possible sub-bit-time latency
    - Compare to AES' greater than 128 bit-times of latency
- No significant power spike during key change / message start
  - ***Supports devices that use scavenged power (e.g. dongles)***
- Several thousand times faster and smaller than public key
- Typically has 1:1 byte replacement (to fit existing message sizes)
  - Can exploit existing CRC or checksum for integrity check
    - With sufficient existing bits, no crypto integrity check is needed
  - Can minimize need for an added explicit initialization vector (IV) and anti-replay data
- Optimized for CPUs typically found in embedded, real time, control, and communication systems (exploits hardware multiply instruction)
- ***With a sub-bit-time latency and no message expansion, BeepBeep can be truly transparent to system timing***

Observation: The plain-to-cipher paths (shown in red) use just one ADD and one XOR. If data arrives least significant bit (LSB) first, each plain bit can produce its cipher bit before the next plain bit arrives; i.e., latency less than one bit time. (Note that crypto bit order can be defined as LSB first, regardless of communication defined bit order.)

BeepBeep Block Diagram

(Bruce Schneier's version)*



Re-Export Controlled ??

(see FSE 2002 paper reference on last slide)

* Doesn't imply design involvement nor endorsement. In fact, his first impression was: "Something that small and that fast can't possibly be secure."

Minimizing Growth in Message Size

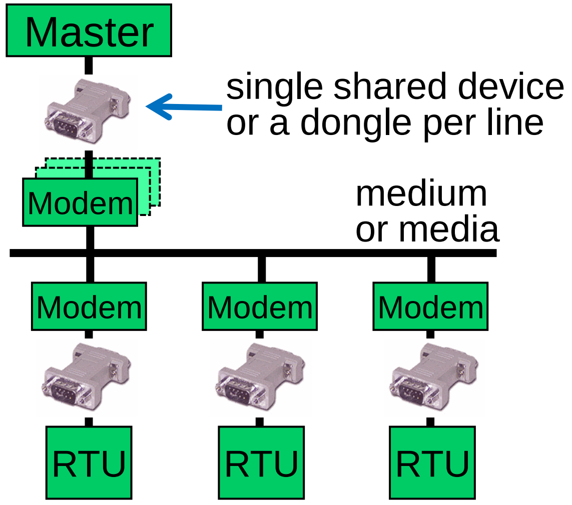
- **Don't use block padding (BeepBeep isn't a block cipher)**
- **Minimize or eliminate Initialization Vector (IV) and anti-replay data**
  - Use existing data (e.g. any unencrypted changing header fields)
  - Use explicit or implicit message identification (e.g., time or sequence numbers)
    - Most real time systems have these (e.g., isochronous protocol implied time)
  - Eliminate IVs and anti-replay data by chaining messages together
    - Auto-keyed crypto-state is carried over between messages
    - Can be used only with reliable message delivery
    - Fast recovery procedure for lost/rejected messages
- **Use plaintext's existing check data for integrity**
  - Check data includes checksums, CRCs, parity, etc.
  - Auto-key carries any alterations through to check data
  - Crypto algorithm doesn't need to do the check
    - Reduces latency by not buffering messages
  - May need to add bits if existing check bits aren't enough
    - Added bits will cause a message delay equal to the number of bits that have to be added
    - Failed integrity can be signaled by inverting the Stop bit or corrupting other end-of-frame or end-of-message delimiter(s)
      - Resulting framing error causes message rejection
      - This is similar to one of our Ethernet fault-tolerance patents

Outline

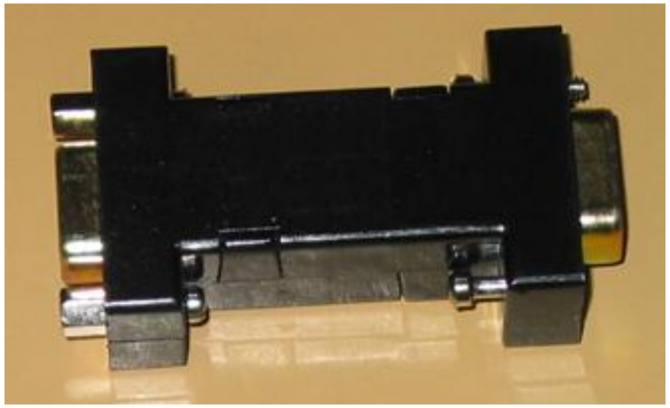
- Real-time / retrofit cryptography requirements are different from existing cryptography requirements
  - Current crypto algorithms are ill-suited for real-time / retrofit
- Some of our developments
  - A symmetric encryption algorithm (called BeepBeep) that overcomes the problems with using existing cryptography for real-time systems
  - A small cryptography module that is easy to insert into existing communication lines
  - Tamper resistance for embedded software
  - Broadcast / multicast command authentication that uses very little computation and communication resources compared to using a message authentication code (MAC) or public key cryptography

Basic Concepts for SCADA Retrofit

- Inline addition of security components between existing SCADA masters and their RTUs
  – Between master serial ports and modem (bank)
  – Between each field modem and its RTU
  – Securing all modem-to-modem communications
- ***Transparent to existing communication***
- Change of existing software is <u>not required</u>
  – Can be a “lump in the cable” hardware dongle
  – But, software retrofit implementations are an option
    - Facilitated by BeepBeep’s: – Small code size and zero working data size
      – Few CPU cycles needed, even during startup
    - It is possible to reduce deployment cost when existing master and/or RTU software can be updated
      – RTUs have to be remotely upgradeable, otherwise hardware dongle is cheaper
      – Need a single RTU type, otherwise hardware dongle is probably cheaper
      – Master and/or RTU vendor(s) need to be amenable to a software change
      – However, retrofitting software can be more expensive if the source code ***and*** knowledgeable programmers are not available
- Installation costs for added security components need to be very low



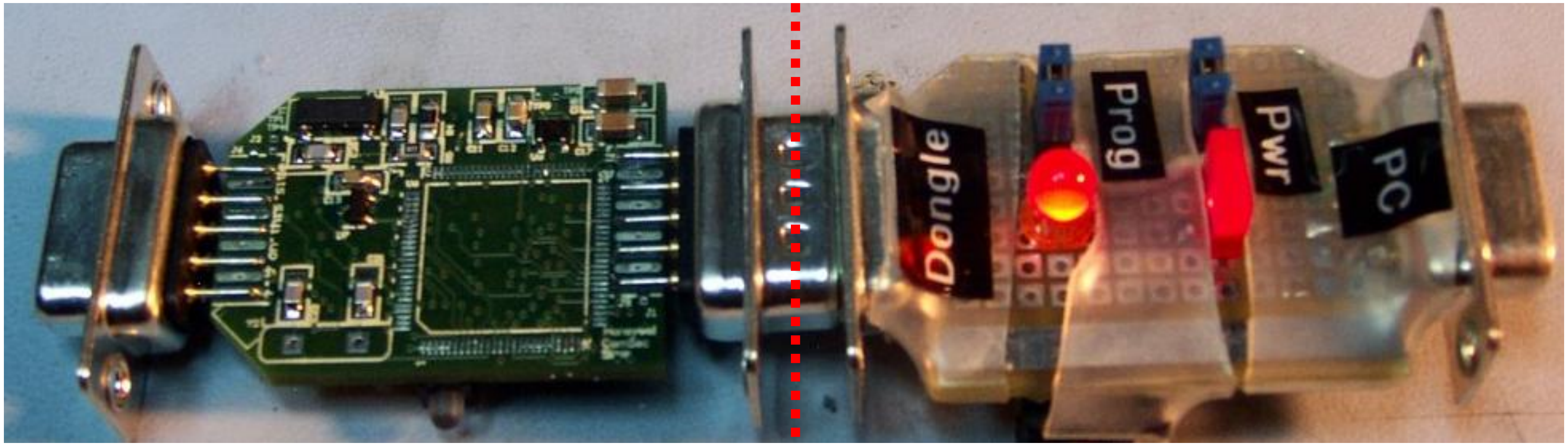
"Crypto Dongle" Implementation



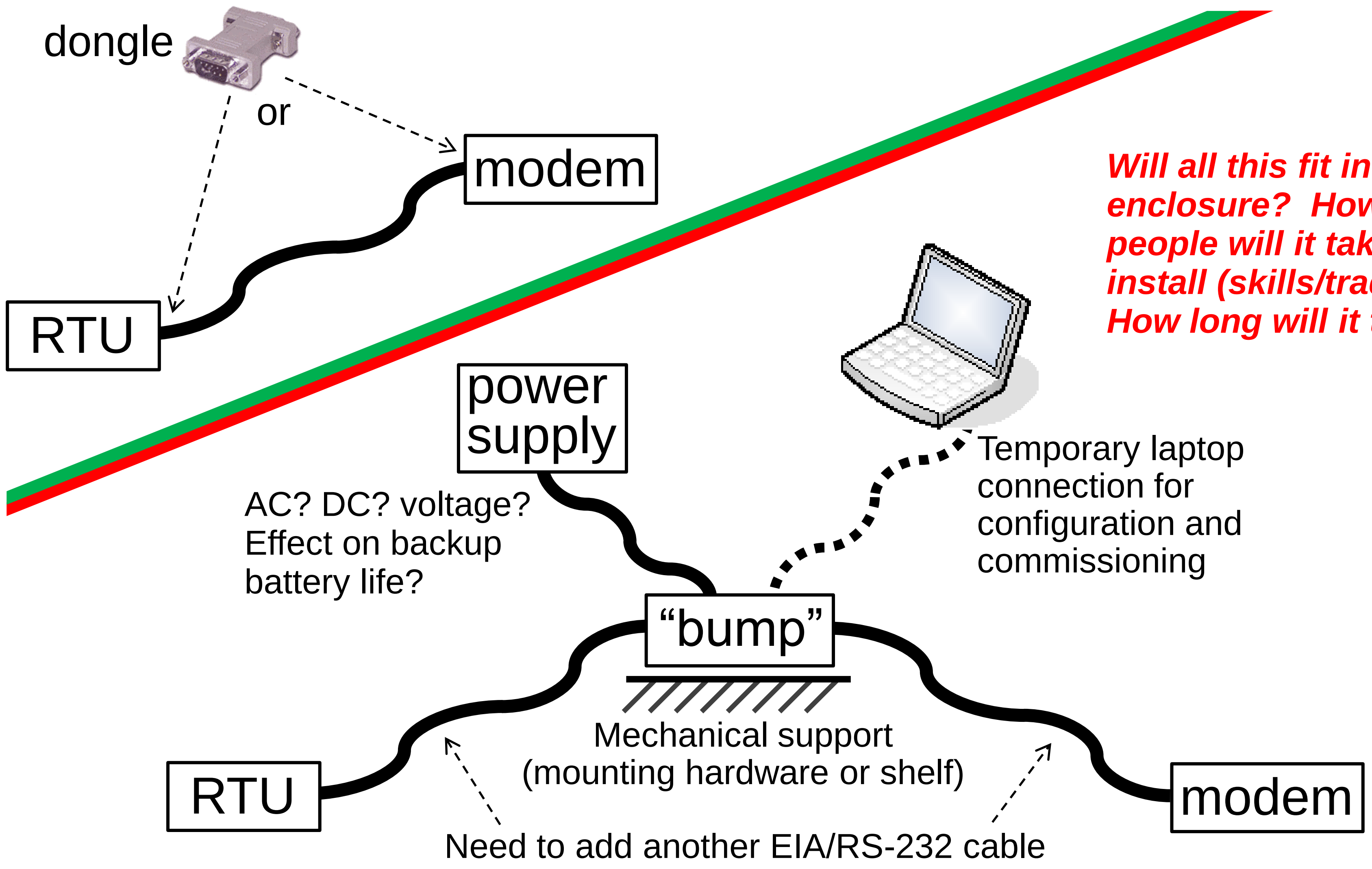
- RS-232 DB-9 connectors
- Transparent functionality
  - "Free" scavenged power
  - Low latency operation
- Simple, rapid field retrofit
  - need only a screwdriver
  - no special skills

- Easy key management
- Secure key escrow option
- Export approved (renew)
- Could use standards (e.g. AES, FIPS 140)
- Tamper resistant
- Affordable security

Dongle on left. Programming adapter on right.



Dongle vs "Bump" Installation Comparison



Will all this fit into the enclosure? How many people will it take to install (skills/trades)? How long will it take?

AC? DC? voltage? Effect on backup battery life?

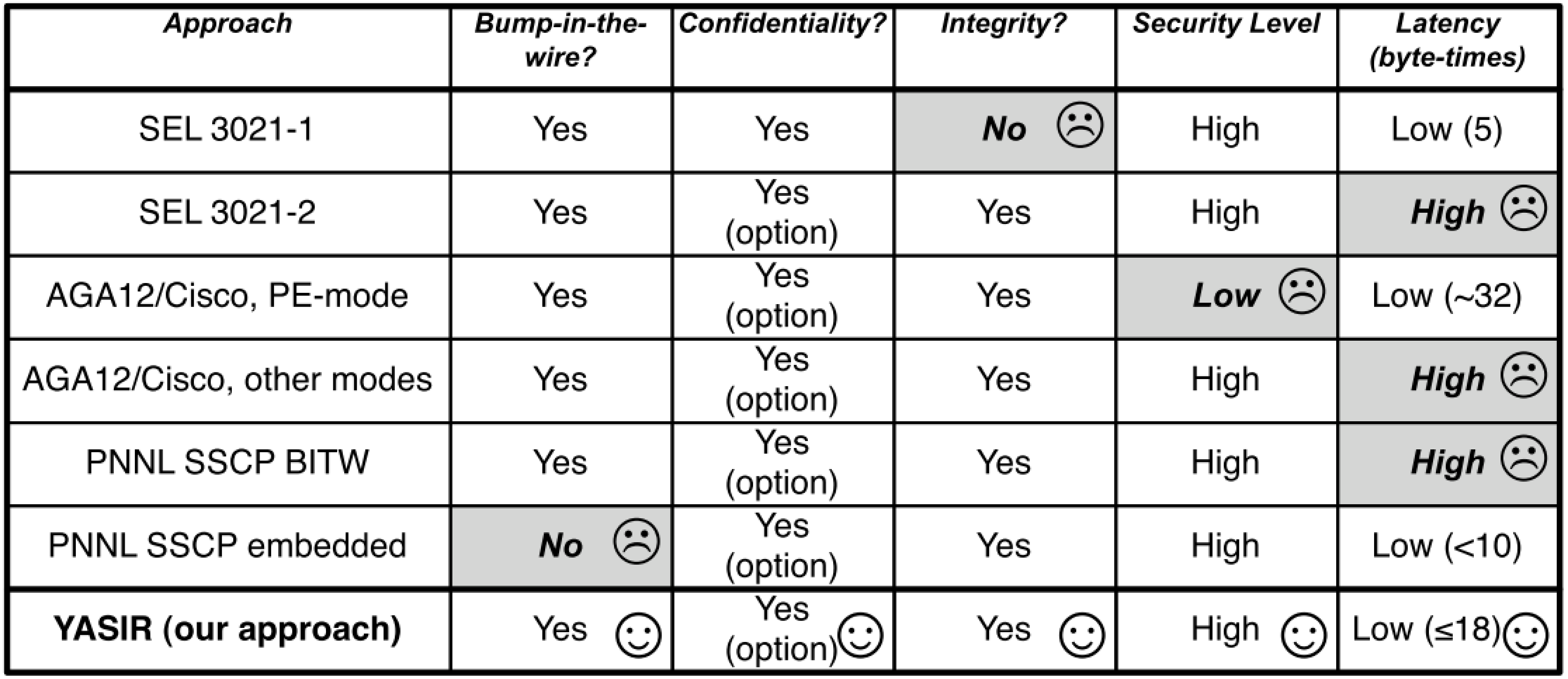
Temporary laptop connection for configuration and commissioning

Mechanical support (mounting hardware or shelf)

Need to add another EIA/RS-232 cable

Feature Comparison

- The table below was taken from “YASIR: A Low-Latency, High-Integrity Security Retrofit for Legacy SCADA Systems” by Tsang, P.P. and Smith, S.W., 2008, in International Federation for Information Processing, Volume 278; *Proceedings of the IFIP TC 11 23rd International Information Security Conference;* Sushil Jajodia, Pierangela Samarati, Stelvio Cimato; (Boston: Springer), p. 449.
- The last row was added for BeepBeep implemented in a hardware dongle.



Approach	***Bump-in-the-wire?***	***Confidentiality?***	***Integrity?***	***Security Level***	***Latency (byte-times)***
SEL 3021-1	Yes	Yes	***No*** ☹	High	Low (5)
SEL 3021-2	Yes	Yes (option)	Yes	High	***High*** ☹
AGA12/Cisco, PE-mode	Yes	Yes (option)	Yes	***Low*** ☹	Low (~32)
AGA12/Cisco, other modes	Yes	Yes (option)	Yes	High	***High*** ☹
PNNL SSCP BITW	Yes	Yes (option)	Yes	High	***High*** ☹
PNNL SSCP embedded	***No*** ☹	Yes (option)	Yes	High	Low (<10)
YASIR (our approach)	Yes ☺	Yes (option) ☺	Yes ☺	High ☺	Low (≤18) ☺
BeepBeep / Dongle	Lump in the cable ☺	Yes (default) ☺	Yes ☺	High ☺	Transparent (~ 0) ☺

Outline

- Real-time / retrofit cryptography requirements are different from existing cryptography requirements
  - Current crypto algorithms are ill-suited for real-time / retrofit
- Some of our developments
  - A symmetric encryption algorithm (called BeepBeep) that overcomes the problems with using existing cryptography for real-time systems
  - A small cryptography module that is easy to insert into existing communication lines
  - Tamper resistance for embedded software
  - Broadcast / multicast command authentication that uses very little computation and communication resources compared to using a message authentication code (MAC) or public key cryptography

Tamper Resistance Ideas for Embedded Software

Without someone guarding it, software in remote devices could be compromised and malware inserted. Some protection ideas:

- If the processor chip has cache or scratch-pad RAM of sufficient size (e.g., approx. 0.5 kB) and at least 12 (preferably 28) bytes of available non-volatile memory on chip
  - Store a secret key in the non-volatile memory
  - Use BeepBeep to encrypt the rest of the memory
    - Fast and small enough to do decryption on the fly
- Keep all software in the CPU chip and lock it
- Use other hardware tamper protection techniques
- Authentication with detection of software tampering
  - A secure site (e.g., the master) sends a challenge to a suspect site
  - Suspect site hashes the challenge and the contents of the memory to be protected and uses the result as the key for a message authentication code (MAC)
  - If the software for the hash and MAC also needs to be protected
    - Fill any unused memory with incompressible random data
    - Hash the entire memory

Outline

- Real-time / retrofit cryptography requirements are different from existing cryptography requirements
  - Current crypto algorithms are ill-suited for real-time / retrofit
- Some of our developments
  - A symmetric encryption algorithm (called BeepBeep) that overcomes the problems with using existing cryptography for real-time systems
  - A small cryptography module that is easy to insert into existing communication lines
  - Tamper resistance for embedded software
  - Broadcast / multicast command authentication that uses very little computation and communication resources compared to using a message authentication code (MAC) or public key cryptography

Broad/Multi-Cast Command Authentication Problems

- Typical scenario: A master wants to broadcast simple commands to many remote nodes through some unsecure broadcast media (e.g., RF)
  - Possible transmissions to large subsets of all nodes (e.g., for load leveling / shedding / cycling)
  - How does a remote node know that a command is authentic and not a spoof?
- Also applicable to SCADA broadcast message problem
- Some costly or very limited “solutions”
  - Public key signatures
  - Message authentication codes
  - A (too) simple adaptation of S/Key
  - Timed Efficient Stream Loss-tolerant Authentication (Tesla)
- Our solution = a better adaptation of S/Key

References

- An example BeepBeep application is described in:
  - Pete Bergstrom, Kevin Driscoll, John Kimball
    Making Home Automation Communications Secure
    Computer, v. 34 n.10, p. 50-56. October, 2001.
- Technical details of BeepBeep published in:
  - Kevin Driscoll.
    BeepBeep: Embedded Real-Time Encryption.
    Fast Software Encryption 2002 / Lecture Notes in Computer Science
    Vol. 2365, p. 164-178. February / May, 2002.
- Some applicable patents (US numbers, most have foreign equivalents):
  - 6,804,354 Cryptographic Isolator Using Multiplication
  - 6,763,363 Computer Efficient Linear Feedback Shift Register
  - 6,760,440 One's [sic] Complement Cryptographic Combiner
  - 7,277,543 Cryptographic Combiner Using Two Sequential Non-Associative Operations
  - 8,051,296 System And Method For Initializing Secure Communications With Lightweight Devices
  - 8,407,479 Tamper Authentication and Tamper Detection
  - 8,549,296 Simple Authentication of Messages