ECC Considerations

René Struik
e-mail: rstruik.ext@gmail.com

Outline

1. Cryptography in Highly Constrained Environments:
   - Communication and Computational Overhead Matter
   - Protocol Communication Flows Matter
3. Elliptic Curve Cryptography
   - Cryptographic Security
   - Side Channel Resistance
   - Implementation Cost
   - Curve-Specific Properties
4. ECC and IETF
   - Curve Type
   - Curve Checks
   - Curve-specific Properties
   - Protocols using Curves

Cryptography for Highly Constrained Environments

- Communication/Computation Overhead
- Protocol flows

The Promise of Wireless

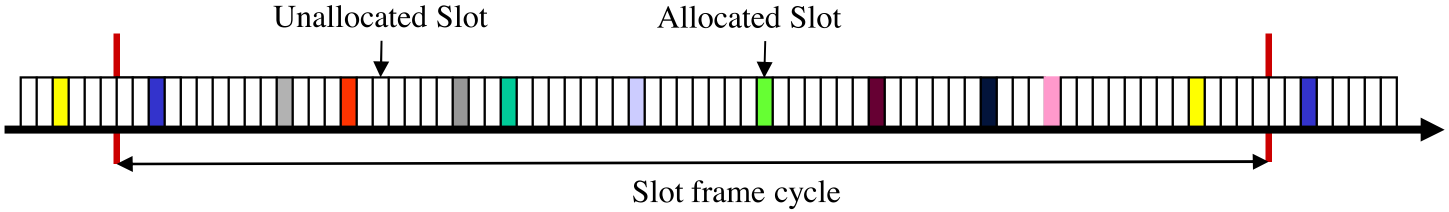
The Economist, April 28, 2007

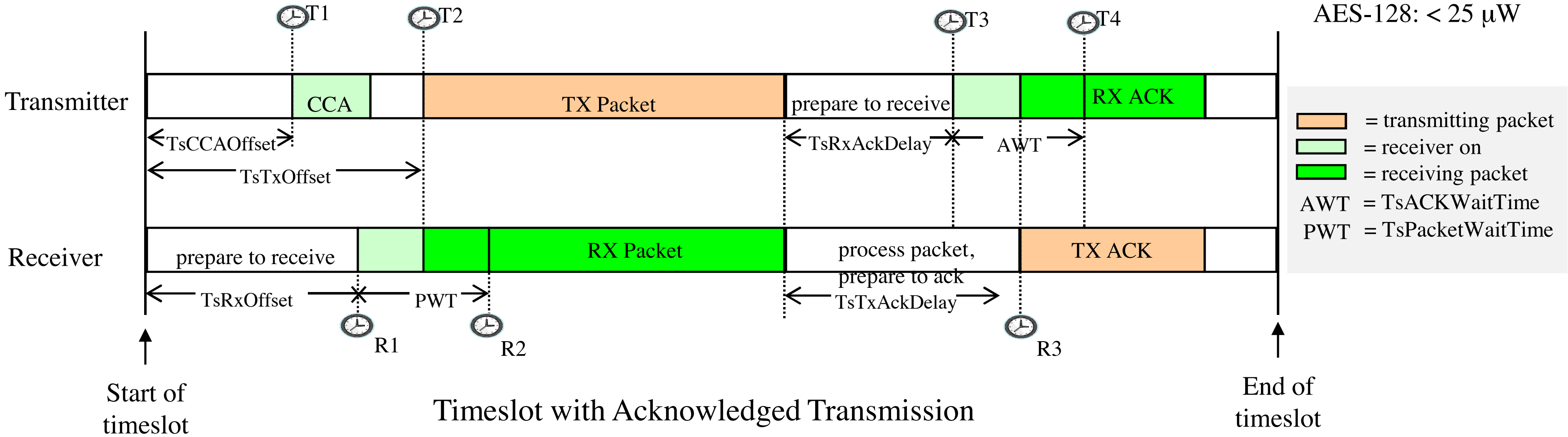
Communication and Computational Overhead Matter

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





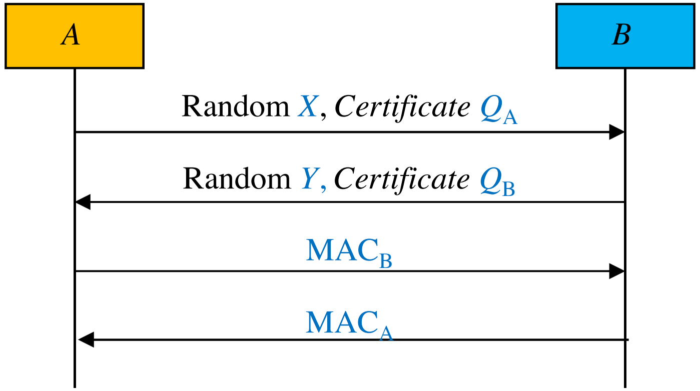
Timeslot with Acknowledged Transmission

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; B-163 scalar multiply ≈ 20μJ-250μJ

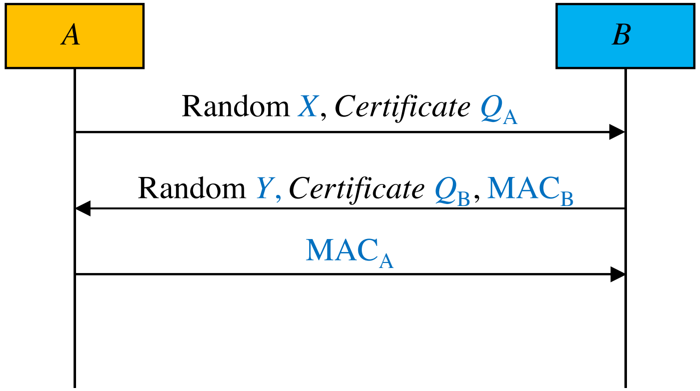
Trade-off: Reduced communication cost ↔ Increased computational cost (& latency)

Communication Flows Matter

Are we using the right communication flows?



(a) Parallelized Computation-Friendly



(a) Flow Number-Friendly

Protocol flow optimization options

- Optimized for computational cost
  This allows online key computation to be executed in parallel
- Optimized for number of message flows

Elliptic Curves

- Standardized Curves
- Cryptographic Security
- Implementation Security
- Implementation Cost
- Curve-Specific Properties

Elliptic Curves

Standardized Curves

NIST:

- Prime curves: P-192, P-224, P-256, P-384, P-521
- Random binary curves: B-163, B-233, B-283, B-409, B-571
- Binary Koblitz curves: K-163, K-233, K-283, K-409, K-571

Brainpool :

- Prime curves: BP-160, BP-192, BP-224, BP-256, BP-320, BP-384, BP-512
- Binary curves: *not defined*

Questions:

- Which ones to pick?
- Do these fit all deployment environments?

Elliptic Curves

Cryptographic Security

This relates to difficulty of solving DLP problem (and, sometimes, DHP problem).

No practical differences with curve choice

- Speed-up of Pollard's rho method by factor up to $\sqrt{(2m)}$, where m is bit-size (for binary curves)
- Parallelization of Pollard's rho method with linear speed-up (for all curves)

Recent work on index calculus attack beating Pollard's rho method only of theoretical interest

- works with $m \rightarrow \infty$, $p \rightarrow \infty$, m composite, etc.
- post-Eurocrypt 2012 results (e.g., [7]) apply *heuristically* for $m \geq 2000$ only {and only considers time complexity, *not* space complexity}

Elliptic Curves

Implementation Security

This relates to resistance against side-channel analysis and fault attacks.

Note: This is still very much a nascent area, with need for more solid footing

Side Channel Resistance

Modular integer arithmetic leaks far more than binary field arithmetic:

- Prime fields: $x \rightarrow r \bullet x \pmod{n}$, where r is random, leaks x (carry-forward attack)
- Binary fields: $x \rightarrow r \oplus x$, where r is random, leaks on $wt_H(x)$ (for CMOS-circuits)
- Modular reduction, with n not of special form, may leak, due to variance execution path then (this applies more to Brainpool than to NIST-p curves)

Fault Resistance

Binary curves seem less susceptible to side channels (or easier to thwart):

- Goubin's attack does apply to prime curves (e.g., P-256), but not to Koblitz curves
- Sign change attack mostly applies to prime curves
- Recent fault attacks yielding points of low order less applicable to binary curves

Elliptic Curves

Implementation Cost

This relates to the foot-print, RAM requirements, etc.

Lack of data on prime curves; binary curves with very low implementation footprint

Data points in hardware [3] (for bit-size m=192):

- Prime curves vs. binary curves
  cycles 3×, energy consumption 4×, power consumption 1.3×
- Energy cost: 14 μJ (binary) vs. 54 μJ

Data points in software:

- No energy cost figure available (to my knowledge), but would be order(s) of magnitude higher

Elliptic Curves

Curve-specific Properties
More esoteric properties…

Hashing into curve:
Binary curves always allow efficient [6] deterministic hashing $x \rightarrow Q(x)$,
prime curves sometimes do (but not for P-256 curve)

Note: non-deterministic mappings possible, but may be susceptible to side channel Attacks (e.g., with password-based key agreement)

Elliptic Curves

Are we using the right curves?

- FIPS 140-2 evaluation suggests almost everyone focusing on *prime curves*
- Technical literature suggests that *binary curves* are better fit

Implementation cost:

Lack of data on prime curves; binary curves with very low implementation footprint

- B-163 scalar multiply ≈ 20μJ-250μJ (in HW)

Computational complexity:

New instruction sets (e.g., Intel's) make binary field arithmetic very efficient

Side channel resistance:

Binary curves seem less susceptible to side channels (or easier to thwart):

- Goubin's attack does apply to prime curves (e.g., P-256), but not to Koblitz curves
- Sign change attack mostly applies to prime curves
- Fault attacks yielding points of low order less applicable to binary curves

Hashing into curve:

Binary curves allow efficient deterministic hashing, prime curves *not* necessarily

Note: Radio engineers familiar with polynomial circuitry (such as CRC-16)

ECC and IETF

- Curve Type
- Curve Checks
- Curve-specific Properties
- Protocols using Curves

ECC and IETF

Discussion Points

1. Curve Type
   IETF mostly goes with prime curves, which seem less suitable for constrained devices and may be far more susceptible to implementation attacks
2. Curve Checks
   IETF mostly keeps silent on curve checks, despite fault attack risk
3. Curve-specific Properties
   Deterministic hashing would be cool property to exploit for IETF
4. Protocols using Curves
   IETF protocol implementation do *not* favor parallel key computation
   IETF protocols are not all role-symmetric (client-server…)

(Other topic all-together [since not ECC-specific]:
proposed use of *raw* public keys (HIP, CoRE folks, etc.))