Cryptography for the Internet of Things

Kenny Paterson
Information Security Group
@kennyog; www.isg.rhul.ac.uk/~kp
The Internet of Things (IoT) is the network of physical objects or "things" embedded with electronics, software, sensors and connectivity to enable it to achieve greater value and service by exchanging data with the manufacturer, operator and/or other connected devices.

Wikipedia

- Billions of “things” ranging in size from micro-sensor to car.
- Connectivity via different network technologies.
- Trillions of dollars.
- Autonomy/control/privacy/security issues abound.
Why Do We Need Cryptography?

Cryptography is one of the best tools we have for:

- Ensuring that information in transit remains confidential.
- Providing assurance about the integrity and (origin) authenticity of information.

Mostly used to provide communications security and achieve access control.

But cryptography is not a kind of magic security dust for IoT:

- Needs careful consideration at system design stage.
- Cryptography is rarely “free at the point of use”.
- Implementation and deployment pitfalls aplenty.
What are Some of the Technical Challenges?

- Handling the full cryptographic key management lifecycle
- Deployment in constrained environments
- Provision of algorithm and protocol agility
- Improving implementation quality
- Dependence on randomness
- Vulnerability to physical attack
Detailed Challenges – Key Management

• Cryptographic algorithms and protocols need keys.

• Where do those keys come from?

• Wireless security for your home router:
  • Manual provisioning.
  • Poor quality keys, infrequently changed.

• Your browser’s SSL/TLS connections:
  • Root public keys are configured in the browser.
  • Public key infrastructure and the CA ecosystem.
  • Complex and automated key exchange protocol reliant on PKI to establish session keys.
Detailed Challenges – Key Management

- “Thing” may not have a user interface for manual configuration of keys.
  - Where’s the other end of the comms path anyway?
  - Need to get the key there too.
  - So we need to understand communications pattern at the outset.
  - Not best left to end users in general!

- “Thing” may not have capability to run a complex key exchange protocol.
  - OK, so let’s use just symmetric key crypto.
  - Now we probably need unique keys per (device, end-point) pair.
  - Having a system-wide key is asking for trouble.
  - Can we ever update the device keys?
  - How is that mechanism itself secured?
Detailed Challenges – Key Management

Even if a more complex key exchange protocol is possible:

• “Thing” may have no mechanism to update root keys or be too dumb to handle standard revocation procedures.

• Manufacturer of “thing” may have little financial incentive to implement robust key lifecycle processes.

• Algorithm agility:
  • Can the base algorithms be modified in response to attacks?

• Protocol agility:
  • Can the communications protocol be updated?
Some of the most “interesting” IoT applications involve widely-deployed networks of very low-cost sensors.

Chip area barely enough to perform core data processing and comms operations.
- Meaningful crypto in <1k gates is ... challenging.

Profiling DTLS is not going to get you there.

But obvious dangers from roll-your-own algorithms and protocols.
- Recent history is littered with examples of ROT13-level security.
- A personal favourite: PLAID...
Detailed Challenges – PLAID Example

• Contactless card/reader authentication protocol.

• Developed by CenterLink, an agency of the Australian government.

• Australian national standard, fast-tracked through ISO processes

“PLAID [...] is cryptographically stronger, faster and more private [...]”
Centrelink PLAID Specification v8.0, 2009

“[...] strong authentication [...] in a fast, highly secure and private fashion without the exposure of [...] identifying information or any other information which is useful to an attacker.”
ISO/IEC 25185-1.2, 2014
Detailed Challenges – PLAID Example

Unauthenticated!

Non-standard PKE

Legacy PKCS#1 v1.5 used!

No forward secrecy!

Non-standard padding

CBC mode with IV = 0^n

Trial decryption under multiple keys!?

No integrity protection

Reuse of session key

Channel secured with $k_{session}$ (session key)
Detailed Challenges – Implementation Quality

SERVER, ARE YOU STILL THERE?
IF SO, REPLY "HAT" (500 LETTERS).

User Meg wants these 500 letters: HAT. Lucas requests the "missed connections" page. Eve (administrator) wants to set server’s master key to "14835038534". Isabel wants pages about snakes but not too long. User Karen wants to...
Detailed Challenges – Randomness

• Cryptographic algorithms and protocols are heavy consumers of randomness.
  • Keys, Nonces, IVs.
  • Key exchange protocols.
  • Inputs to signature schemes and public key encryption algorithms for strong security.

• Robust randomness generation is hard, even with access to traditional entropy sources.
  • IRQ timings, LSBs of clock time, disk seek time,…

• For IoT applications, many of those sources are not available.
  • Leading to, for example, fixed keys, low-entropy keys, repeated primes in RSA,…
Detailed Challenges – Randomness

TOUR OF ACCOUNTING

Over here we have our random number generator.

NINE NINE NINE NINE NINE

Are you sure that's random?

That's the problem with randomness: you can never be sure.

DILBERT
by SCOTT ADAMS
Detailed Challenges – Randomness

“Factoring RSA keys for fun and a profit of $9000”

• We performed an Internet-wide scan for SSL/TLS export-grade RSA keys.
• 9.7% of 22.7 million hosts offered such keys.

• 294 hosts had trivially factorable keys (shared primes with other keys, vulnerable to pairwise gcd attack).
  • Attributable to poor randomness in key generation.

• One RSA key was repeated 28,394 times: cheap router with SSL VPN module.
  • Attributable to lazy vendor or really bad randomness in key generation.

https://martinralbrecht.files.wordpress.com/2015/03/freak-scan1.pdf
In implementations, we can usually trade randomness for long-term state.

Use a counter and a key to generate pseudo-random values.

Counter needs to be persistent across system reboots/battery changes.
  - Use a small amount of non-volatile memory.

Need unique, random key to be provisioned per device.
  - So we have reduced a many-time problem to a one-time problem.
  - The problem is still non-trivial.
Detailed Challenges – Physical Attack

- Low-cost “things” may be badly protected against physical attacks.
  - Aiming to extract software, keys, other sensitive data.

- Side-channel attacks exploiting timing, power consumption, EM radiation, fault injection,...

- Countermeasures include physical shielding, specialized implementation and frequent key changes.
  - Widely deployed in financial services and mobile sectors, where risk and costs are well understood.
Closing Remarks

- Heterogeneity of IoT vision precludes “one size fits all” approaches to Cryptography.

- But security is done best in a homogeneous fashion.
  - Because crypto/protocol design and deployment is hard.

- We might try to identify different use cases and their characteristics, then design algorithms and protocols for those.

- Cryptography can help but is not magic pixie dust.