Emerging security challenges for ubiquitous devices

Mirosław Kutyłowski

joint work with Piotr Syga and Moti Yung
a chapter in the forthcoming “CRYPTACUS Book”

COST Action CRYPTACUS

Rennes 2018

Design reality and security challenges for IoT

Idealistic situation for designing IoT devices

1. threat analysis, formulating requirements
2. designing a formal model
3. designing a solution
4. creating a formal security proof
5. implementation
6. certification, audits

Reality

1. **threat analysis, formulating requirements –**
   - frequently missing
   - if existing, then frequently not evaluated (Protection Profiles, ...)
   - reversing the order: first a product, then looking for opportunities to sell
   - applying irrelevant PP (insecure products with EAL7 level)
2. designing a formal model
3. designing a solution
4. creating a formal security proof
5. implementation
6. certification, audits

Reality

1. threat analysis, formulating requirements
2. **designing a formal model** – frequently:
   - an ad hoc construction corresponding to the solution and not to the requirements
   - incomplete (ignoring some attack scenarios)
   - isolated (secure component may create vulnerabilities)
3. designing a solution
4. creating a formal security proof
5. implementation
6. certification, audits

Reality

1. threat analysis, formulating requirements
2. designing a formal model
3. **designing a solution** – frequently
   - unrealistic assumptions about available resources on the chip
   - computational/space complexity etc. versus real cost
   - driven by publication policies and not by real impact
   - patents blocking progress
4. creating a formal security proof
5. implementation
6. certification, audits

Reality

1. threat analysis, formulating requirements
2. designing a formal model
3. designing a solution
4. **creating a formal security proof –**
   - if the model is wrong, then the proof has no value
   - a solution is frequently confidential, so there are no publicly available proofs to verify
   - composability of proofs? (usually no)
5. implementation
6. certification, audits

Reality

1. threat analysis, formulating requirements
2. designing a formal model
3. designing a solution
4. creating a formal security proof
5. **implementation**
   - using components (compilers, OS) that got out of control
   - cost reduction – compromises on production regime
   - priority of making the system to run on time over making it to run safely
6. certification, audits

Reality

1. threat analysis, formulating requirements
2. designing a formal model
3. designing a solution
4. creating a formal security proof
5. implementation
6. **certification, audits** – frequently
   - checking only paper work that aims to show compliance of the product with the documentation
   - not covering all aspects
   - expensive
   - you have to trust certification/audit organizations

IoT and Law

General Data Protection Regulation

dramatic change of the legal approach:

- **before:** the manufacturer could deliver insecure products:
  - **r**isk calculation – e.g. keeping financial resources for compensations or buying an insurance instead of improving technology
  - licences **excluding responsibility** (explicit no guarantee for the product)
- **now:** **security-by-design concept**

challenge

security-by-design sounds nicely but is it available or rather falls into category of *science fiction*?

General Data Protection Regulation

GDPR applicable mostly within the EU, but

- many countries traditionally follow EU data protection approach
- hardly possible to design products GDPR-compliant and GDPR-not-compliant (interactions between such devices, cross-border mobility, ...)
- data generated by the devices and their flow must be taken into account as well

General Data Protection Regulation
European Union

The principles of data protection should ***apply to any information concerning*** *an identified or* ***identifiable*** ***natural person****.*

– so most data could be under protection, even if they are irrelevant

GDPR
rules of processing

Personal data shall be:

(f) processed in a manner that ensures appropriate security of the personal data, including **protection against unauthorised or unlawful processing** and against accidental loss, destruction or damage, using **appropriate technical or organizational measures** "integrity and confidentiality"

GDPR
privacy by design

accountability

The controller shall be responsible for, and **be able to demonstrate compliance** with [these rules]

Randomness problem

Role of randomness

unpredictability – enables for instance distance bounding protocols – the responses cannot be computed in advance

deferring replay attacks – enables challenge-response mechanism

particularly inevitable in case when only symmetric algorithms are used

Implementing randomness

Implementation options

- true randomness
- PRNG
- hybrid

Requirements:

- security **by design**
- one has to be able to **demonstrate** that the generator is secure

Secure randomness

Challenges

- **true randomness**
  - how do you know that for instance a circuit processing randomness from a physical source is not faulty/malicious?
  - statistical tests detect only the **extreme faults**
  - acceptable for standard randomized algorithms but not for security mechanisms
  - bypathing the true random source might be built-in and hidden in a black box device
  - circuit inspection usually destroys the device and might be ineffective (Hardware Trojans)

Secure randomness

Challenges

- **PRNG:**
  - how do you know that PRNG is working as described, e.g. there is no trapdoor like:
    `if` *challenge* $= x$ `then output` c
  - who initializes the secret seed?
    - **the manufacturer:** there is no unpredictability when the manufacturer is concerned
      a security disaster if the manufacturer is dishonest or forced to be dishonest
    - **uploaded by the user:**
      – but who creates the secret for the user?
      – the device may be controlled only if a copy of the seed is retained
      –but the copies should be erased for security reasons
    - **the device** – how do you know that the value generated has not been in fact preinstalled?
- **Hybrid:** problems of both approaches

Conclusion

it seems that we have to admit that

possibilities to create random number generators such that their security can be easily demonstrated

are very limited

for cheap IoT devices

nevertheless deployment of IoT will happen anyway.

what to do?

Multiparty protocols

Multiparty approach:

- in a protocol execution a device A is replaced by two devices A_1, A_2 that **jointly mimic** A,
- A_1 and A_2 come from **different sources** (their creator are not collaborating)

Disadvantages:

- increasing the price
- the user remains passive
- independent manufacturers may collude
- some changes might be necessary for the other protocol participants
- no general recipe how to rebuild the protocols

Concept of watchdogs

Watchdog

Watchdog

- a man-in-the-middle unit between the IoT device and the communication unit
- user controlled
- no secrets installed in the watchdog, not tamper-proof, etc

Watchdogs activities

- interact with the IoT device
- forward the protocol messages – but truncate the additional *control output*

Watchdogs mission

1. detect irregular behavior of the IoT
2. destroy potential hidden channels

Watchdogs approaches

monolithic approach

- redesign the whole protocol
- optimize

disadvantages:

- requires redesign on a case-by-case basis
- new security proofs have to be presented

Watchdogs approaches

fine grained approach

- redesign cryptographic primitives used by the protocols
  - the proposed changes in a protocol should be minimal,
  - whenever possible exactly the same protocol should be executed by other participants of the protocol,
- **plug-and-play approach**

advantages:

- the number of protocols is huge, the number of primitives small
- easier to make decisions what to deploy based on the risk assessment

Cryptographic primitives with watchdogs

Commitments

hash based mechanism

1. the device chooses x at random and computes $c := \mathrm{Hash}(x)$
2. c **presented** as a commitment
3. **opening** c: by presenting x

problem: both c and x may leak information

naïve approach:

1. the Device chooses r' at random, computes $c' := \mathrm{Hash}(r')$ and sends c' to the Watchdog,
2. the Watchdog selects ρ and returns to the Device,
3. the device computes the final value $r := r' \mathrm{xor} \rho$ and sends the final commitment $c := \mathrm{Hash}(r)$.

Problem: c may be constructed in a different way (leaking secret) but the session can be interrupted **before** the commitment is opened

Commitments

easy with with Pedersen commitments:

- the Device chooses r and s at random, computes $c' := g^{r'} \cdot h^{s'}$, and presents c' to the Watchdog,
- the Watchdog chooses r'', s'' at random computes $c := c' \cdot g^{r''} \cdot h^{s''}$ and:
  - sends r'', s'' to the Device,
  - sends the commitment c to the Reader,
- the Device computes the committed values: $r := r' \cdot r''$, $s := s' \cdot s''$.

expensive asymmetric crypto involved

Basic commitment from encryption

1. choose plaintext t and key k at random,
2. compute $c := \mathrm{Enc}_k(t)$,
3. present (t, c) as a commitment for k.

The commitment opening **test** is $\mathrm{Enc}_k(t) \stackrel{?}{=} c$.
Opening data: k.

Commitment with a watchdog

1. the Device creates a commitment (t', c') for k' using the **basic mechanism** (i.e. $c' := \mathrm{Enc}_{k'}(t')$ and t' is a single n bit block), and presents (t', c') to the Watchdog,
2. the Watchdog chooses θ and κ at random and presents them to the Device,
3. the Device recomputes the basic commitment:
   - $t := t' \text{ xor } \theta$,
   - $k := k' \text{ xor } \kappa$,
   - $c := \mathrm{Enc}_k(t)$,

   and presents c to the Watchdog,
4. the Watchdog chooses α at random, recomputes t and sets
   - $\zeta := \mathrm{Enc}_c(\alpha)$.
5. the Watchdog returns α to the Device and presents the **final commitment** (t, α, ζ) concerning the element k to the Reader.

Remarks

- the Watchdog has no control over the computation of c,
- the Device can install a subliminal channel in a corrupted c
- ... but c is hidden behind the commitment (α, ζ) created by the Watchdog

Opening the commitment

controlled by the Watchdog:

1. the Device presents k' to the Watchdog,
2. the Watchdog computes $k := k' \text{ xor } \kappa$ and aborts if:
   1. $c' \neq \text{Enc}_{k'}(t')$,
   2. $c \neq \text{Enc}_k(t' \text{ xor } \theta)$,
3. the Watchdog presents k to the Reader,
4. the Reader checks the commitment:
   1. $\bar{c} := \text{Enc}_k(t)$,
   2. $\zeta \stackrel{?}{=} \text{Enc}_{\bar{c}}(\alpha)$.

Encrypted Random Challenge

Protocol primitive

the Device:

- chooses r at random,
- sends $\mathrm{Enc}_k(r)$ to the Reader (k is shared with the Reader)

requirement: except for the Reader, nobody (including the Watchdog) should learn r

So no way to control the Device??

Encrypted Random Challenge

A version with a Watchdog:

1. the Watchdog sends a **commitment** θ to a **random** α (e.g. basic commitment),
2. the Device chooses r' **at random**, $s := \mathrm{Enc}_k(r')$ and sends s to the Watchdog,
3. the Watchdog computes $\sigma := \alpha \otimes s$ and sends
   - σ to the Reader,
   - α and θ to the Device

— the encryption scheme must fulfill some properties

Encrypted answers to challenges

Problem

for inspection the Watchdog requires either

- an advanced encryption scheme – **too heavy computations**
- or the plaintext, but then the Watchdog would need the encryption key – **impossible**, the key must be kept secret,
  the plaintext answer must be kept secret as well

Encrypted answers to challenges

1. the Reader:
   1. create a commitment c to a random key z'
   2. sends c to the Watchdog of the Device,
   3. $c_0 = \mathrm{Enc}_k(z')$ with the key k shared with the Device,
   4. send c_0 to the Device
2. the Device decrypts c_0 and reveals z' to its Watchdog,
3. the Watchdog:
   1. checks correctness of z' against the commitment c
   2. chooses a key z at random,
   3. sends $\mathrm{Enc}_{z'}(z)$ to the Reader,
4. from now on, all messages forwarded by the Watchdog to the Reader are (extra) encrypted with z.

Distance bounding

Problem

during Rapid Bit Exchange there is **no time for interactions with the Watchdog**

Distance bounding

PREPARATION:

- the Watchdog and the Reader: establish a session key z as described above
- a pseudorandom string $B[z]$ computed on both sides

RAPID BIT EXCHANGE:

executed as usual except that:
the Watchdog receiving $A[i]$ from the Device computes

$$A'[i] := A[i] \text{ xor } B(z)[i]$$

and forwards $A'[i]$ to the Reader.
Verification adjusted accordingly.

Conclusion

the situation is **not that hopeless** as one may think at the beginning

technical solutions may replace blind trust assumptions

Privacy

Problem

security assumption each pair of devices shares a secret key

problem before starting a session a device has to learn identity of the interlocutor

$\Rightarrow$ **perfect for tracing users by Big Brother**

with Diffie-Hellman not a problem, but how to do it without asymmetric crypto?

No good news ...

partial solutions

1. random key predistribution for the initialization phase
2. responses to a challenge in a Bloom filter
3. stateful protocols (dynamic pairwise identifiers)

... still looking for an universal and practical solution

Thank you for your attention