How careful should we be when implementing cryptography for software update mechanisms in the IoT?

Alexandre Adomnicai[1,3], Jacques Fournier[2,3], Laurent Masson[1], and Assia Tria[2,3]

[1]Trusted Objects
80 avenue Villevieille, 13790 Rousset, France
{a,l}.lastname@trusted-objects.com

[2]CEA Tech
880 route de Mimet, 13120 Gardanne, France
firstname.lastname@cea.fr

[3]Mines Saint-Étienne
880 route de Mimet, 13120 Gardanne, France
firstname.lastname@emse.fr

The ability to update connected devices in the field could be a useful mechanism to perform upgrades of the devices, to circumvent any security problem or correct any bug. Unfortunately, an attacker could use that update to replace the current firmware by a compromised one and misuse the connected device. For example, a smart door locks firmware could be replaced by one that embeds a Trojan and would give access to an attacker. One way of handling such essential security issues is through the use of cryptographic tools. In this paper, we discuss about the importance of looking at the way cryptographic algorithms are implemented for software update mechanisms in the IoT.

During the software update of an IoT device there are two fundamental security issues: *who* is installing *what*. For *the who*, we would like to make sure that the software to be installed comes from a legitimate issuer. For *the what*, we would like to make sure that the software to be installed is the correct one. In addition, the issuer might want to protect the confidentiality of the software (*e.g.* for intellectual property protection issues) and make sure that it is installed on a legitimate device (*i.e.* not a clone or a fake). In practice, most of these issues can be handled using cryptographic techniques.

For example, to ensure that the software comes from a reliable source, the IoT device might authenticate the issuer. Unfortunately, there are products on the market that do not implement this rather basic feature [1]. One way to answer this is to require the storage of a server public key into the devices in order to verify signed software updates. This mechanism ensures that only officially signed updates are applied to devices but it does not authenticate them to the server. It means that a firmware update could be done on an untrusted device (*e.g.* a clone). One way to handle such an issue could be by covering the confidentiality issue using symmetric encryption. In that case, the untrusted device would then have to know the secret key used to encrypt the software to install it. However, it does not allow an attacker to update a malicious software as long as he does not know the server's signature private key. On the other hand, if a bug is discovered in a given firmware version whose signature is known to the attacker, then he could re-update it after each 'real' update to continue to exploit this known vulnerability. This suggests that perfect forward secrecy (PFS) could be a valuable property for software updates.

Concerning the device authentication, if the network allows uplinks, it could be wise to set up a mutual authentication to ensure that the firmware (even encrypted) is not sent to an untrusted device. The best option would be to set up a mutual authenticated key exchange to achieve the PFS property. Unfortunately, the disadvantage of using asymmetric cryptography for authentication is that such calculations are computationally intensive. Indeed, for low resource devices (*e.g.* battery powered devices) a cryptographic operation with a small footprint and a limited amount of energy consumption is important. Unless the device owns a hardware accelerator for elliptic curve cryptography (ECC),

which is not necessarily the case for low-cost devices, this does not seem realistic because of limited performance. Another alternative may be to use a couponing scheme signature [2]. It is a space-time tradeoff which considerably reduces computations overhead for the signatory. Unfortunately, this kind of algorithm is not easy to implement in practice due to the coupon management complexity. Plus, it only lightens the signature computation, not the verification.
A lightweight mutual authentication protocol based on symmetric cryptography could also be considered but would require shared secrets between the device and the server.

Regarding the issue of software integrity, if the server authenticates itself to the device by signing the encrypted firmware, then it is already taken into consideration. Otherwise, it could be done using an authenticated encryption with associated data (AEAD) or append a Hash of the plaintext before encryption.

In all the above cases, cryptography has to be embedded into the IoT device which shall have to store and handle secret/sensitive cryptographic keys at some point. The security of those keys within the device has to be guaranteed throughout the life cycle of the device, *i.e.* from the manufacturing of the device through the personalization stage up to its end of life. In the meantime, the device will be in the field and since it can be a hostile environment (*i.e.* physically accessible to hackers), physical attacks must be taken in account. We can distinguish three classes of physical attacks.

> **Side channel attacks** (SCA) allow to recover the encryption key by using information leakages (*e.g.* computation time, power consumption, ...) during cryptographic computations [3].
>
> **Fault attacks** aim at introducing a fault (*e.g.* clock glitches, Electromagnetic pulses, laser/light flashes, ...) during cryptographic computations and cause errors which can be exploited to recover the involved secret [4].
>
> **Invasive attacks** refer to attacks where the physical properties of the chip are irreversibly modified (*e.g.* microprobing) [5].

SCA are easy to use and do not require much equipment. Fault attacks are a little more complicated to set up but are very powerful and may allow an attacker to break an unprotected system faster than any other SCA. Finally, invasive attacks are the most powerful attack class, but also the most expensive ones. Although the required equipment sounds expensive, the second-hand market makes all attack equipment affordable making such attack scenarios relevant even for low cost IoT devices.

In case of software updates, the firmware may be sent encrypted so authentic devices would have to perform cryptographic computations to decrypt it. If the key can be recovered by performing physical attacks, then an attacker could get the plain firmware and try to reverse-engineer it to find new security failures. Furthermore, if the same encryption key is shared between the server and several devices for "easy" key management [6], then an old vulnerable software version update could be made on several clone/fake devices (*e.g.* to create botnets for DDoS). To mitigate such a risk, either the encryption key is diversified per device (meaning that the server must manage and store as many encryption keys as there are devices in the field) or the implementation of the cryptograhy used must embed countermeasures against physical attacks.

Barriers to guard against these kinds of attacks are the classical IoT requirements of high performance, low power consumption and low foot print. Implementing an AES on IoT devices is conceivable and fast software implementations are available for 8-bit [7] and 16-bit [8] microcontrollers. Although these implementations are resistant against first-order SCA, it is still not enough against some sophisticated attacks. Existing countermeasures generally used to protect from high-order differential attacks are not feasible for IoT devices due to the overhead in terms of energy and memory consumption, time execution and code size. When combined with cost and budget constraints, these limitations make the design of security measures for the IoT quite challenging. These remarks do not only concern software implementations and are also valid for hardware ones. For example, the design of an efficient and

secure AES hardware implementation that incorporates many countermeasures against side-channel attacks would have a high area requirement, which results in high costs [11]. Thus, some cheap solutions sometimes make use of weak homemade cryptographic algorithms (*e.g.* tweaked XOR cipher) [1].

One answer to such a problem could be the use of lightweight cryptography (LWC). Several algorithms have been proposed in order to set new encryption standards for the IoT [9]. Some of them (*e.g.* PRESENT [10]) have been well studied about their security and are ready to use in practice. Although LWC algorithms are quite appropriated to constraint devices, there is still the issue of the countermeasures' overhead associated to physical attacks. Our work aims at building new lightweight ciphers intrinsically resistant to physical attacks, in such a way that there would be no need of countermeasures implementations because its internal structure would protect against such attacks. We believe that it could promote security integration for the IoT due to better performance and lower area requirement for the same security level.
The second aim would be to propose a secure life cycle management (including software updates) for IoT devices implementing LWC.

References

[1] M. Jordon. *Hacking Canon Pixma Printers - Doomed Encryption.* `http://www.contextis.com/resources/blog/hacking-canon-pixma-printers-doomed-encryption/`, 2014.

[2] D. M'Rahi and D. Naccache. *Couponing Scheme Reduces Computational Power Requirements for DSS Signatures.* 1994.

[3] Y. Zhou and D. Fend. *Side-Channel Attacks: Ten Years After Its Publication and the Impacts on Cryptographic Module Security Testing.* Cryptology ePrint Archive, Report 2005/388.

[4] C. Giraud and H. Thiebeauld. *A Survey on Fault Attacks.* International Federation for Information Processing Digital Library; Smart Card Research and Advanced Applications VI, 2004.

[5] A. Tria and H. Choukri. *Invasive attacks.* Encyclopedia of Cryptography and Security (2nd Ed.) 2011: 623-629.

[6] A. Chapman. *Hacking into Internet Connected Light Bulbs.* `http://www.contextis.com/resources/blog/hacking-internet-connected-light-bulbs/`, 2014.

[7] D. A. Osvik, J. W. Bos, D. Stefan, and D. Canright. *Fast Software AES Encryption.* In Fast Software Encryption. FSE 2010, volume 6147 of LNCS, pages 75-93. Springer, 2010.

[8] A. Moradi, A. Poschmann, S. Ling, C. Paar, and H. Wang. *Pushing the Limits: A Very Compact and a Threshold Implementation of AES.* Advances in Cryptology - EUROCRYPT 2011, volume 6632 of LNCS, page 69. Springer, 2011.

[9] D. Diu, Y. Le Corre, D. Khovratovich, L. Perrin, J. Großschädl and A. Biryukov. *Triathlon of Lightweight Block Ciphers for the Internet of Things.* Cryptology ePrint Archive, Report 2015/209.

[10] A. Bordanov, L. R. Knudsen, Leander, G., C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin and C. Vikkelsoe. *PRESENT: An Ultra-Lightweight Block Cipher.* CHES 2007, pages 450–456. Springer, 2007.

[11] M. Doulcier-Verdier, J.-M. Dutertre, J. Fournier, J.-B. Rigaud, B. Robisson, and A. Tria. *A side-channel and fault-attack resistant* AES *circuit working on duplicated complemented values.* In Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2011 IEEE International, feb. 2011, p. 274 –276.