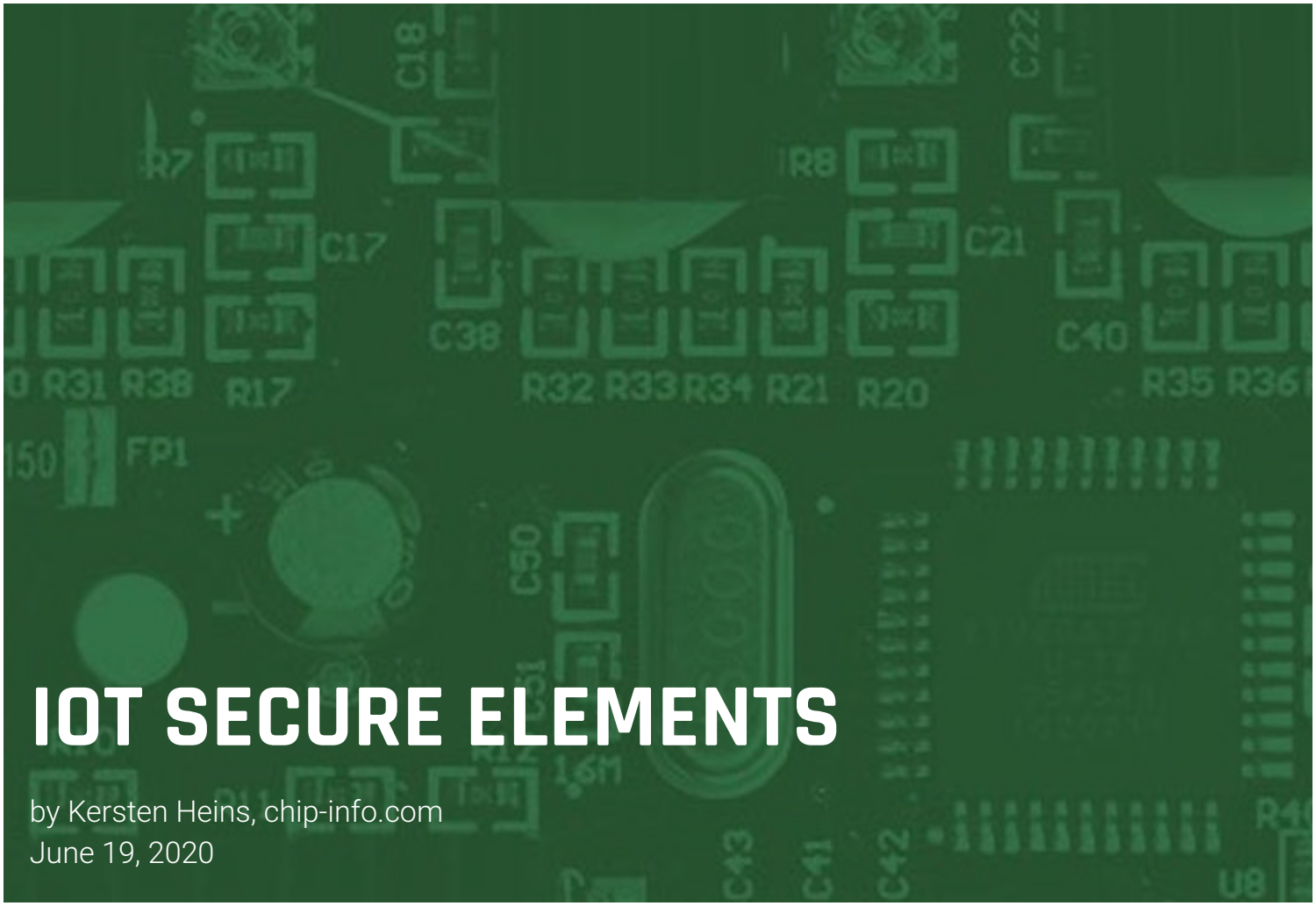

IOT SECURE ELEMENTS

by Kersten Heins, chip-info.com
June 19, 2020

IoT secure elements are specialized microcontrollers offering **bullet-proof control of remote devices**, e.g. for communication with a vending machine or an environmental sensor via Internet. These kind of IoT devices are usually unattended and work autonomously, and system owners will have to protect them against failure or unintentional operation. Objective is to prevent hackers from counterfeiting, cloning, stealing or altering involved objects or data or misusing the IoT application in general.

Similar security requirements also apply to payment or identification scenarios where **smartcards** are used by humans as security tokens. Consequently, manufacturers are leveraging smartcard chip technology and shielding techniques. In fact, most secure elements are smartcard IC *derivatives* – offered by very few chip manufacturers with long experience and skills in this particular business.

TARGET APPLICATIONS

IoT secure elements are aiming at remote applications requiring **strong authentication** of connected devices **and secure data exchange** with them. In addition, IoT secure elements can help to ensure **device integrity**, e.g. protect boot process and validate firmware updates of the network device. Related processes are protected by sophisticated hard- and software countermeasures against advanced physical and logical attacks.

IoT means that devices are connected via Internet, but similar authentication techniques (i.e. IoT secure elements reviews here) can also be used for peripherals or consumables (e.g. power-tool batteries) or accessories (e.g. an audio headset) connected to a host device via wired interface – for safety reasons or business support (brand protection, **anti-counterfeiting**). In addition, *secure counters* can enable some business models, e.g. to **limit usage** of a connected device.

HOW PUBLIC-KEY CRYPTOGRAPHY CAN HELP

A fundamental ingredient of secure communication systems is **public–key cryptography** (or asymmetric cryptography) which is an encryption scheme. Other than symmetric cryptography, public-key cryptography uses a **pair of keys** which are different but mathematically linked to each other: i.e. a *public key*, which may be disseminated widely, and a *private key*, which are known only to the "owner". In a one-to-one electronic communication one of the keys (either public **or** private – depending on use case) is used by the sender, the other one is used by the recipient.

Figure 1: Symmetric vs. asymmetric keys (© chip-info.com)

Public-key cryptography has a big advantage when **compared to classic symmetric crypto** schemes where *same* unique secret key is used by sender as well as by recipient: effective security **requires keeping only the private key as a secret**; the public key can be openly distributed without compromising security → key distribution and key storage are much easier to handle. But due to the fact that one key is public, *asymmetric* crypto algorithms are more complex and require longer key lengths compared to *symmetric* cryptography – at the same level of security. For example, security provided with a 1024-bit key using asymmetric RSA is considered approximately equal to an 80-bit key in a symmetric algorithm like **AES**.

In general, public-key cryptography can solve different fundamental security problems related to one-to-one communication scenarios:

1. protect message **privacy**, i.e. nobody else is able read contents of message
2. verify **authenticity** of sender, i.e. sender identity is tamper-proof and unique
3. ensure **integrity** of message, i.e. make sure that nobody is able to modify message contents on its way to the recipient

Problem 1 can be solved by data encryption, i.e. sender uses the *public* key of the recipient → decryption can be done with *recipient's private* key only. Problem 2 and 3 can be solved by use of a **digital signature** – to be applied using the *sender's private key*. So, how does this work?

MESSAGE AUTHENTICITY AND INTEGRITY

In general, the sender's private key is used to sign a message, and all recipients can verify the sender's authenticity with the sender's public key. In order to limit required computational effort to create a digital signature, only the **hash** value of the message is encrypted, not the complete message itself. What does "hash" mean? A hashing algorithm is a mathematical function that condenses data to a fixed size, a hash is a *fingerprint* of the original data. On top of that a **secure hash** is *irreversible* and *unique*. *Irreversible* means "one-way", i.e. from the hash itself you couldn't figure out what the original piece of data was, therefore allowing the original data to remain secure and unknown. *Unique* means that two different pieces of data can never result in the same hash value. Today, for digital signatures SHA-2 algorithms are common, e.g. SHA-256 with a hash length of 256 bit.

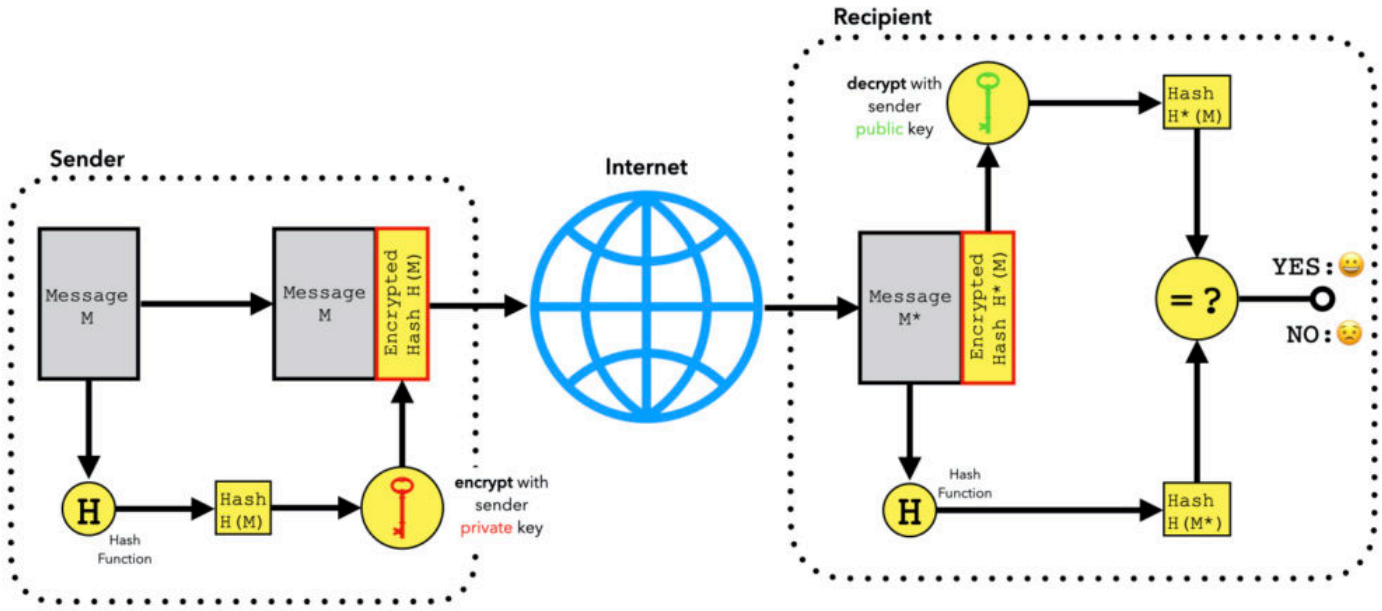

Figure 2: Signature creation and verification process (© chip-info.com)

So, **on sender side** (see Figure 2) the message will be hashed, encrypted with sender's private key (aka "signed") and attached to the message before being transmitted via a public (i.e. unsecured) network. In order to verify sender authenticity and message integrity this signature will be decrypted

with the sender's public key **on receiver side**. This operation creates H*(M). For verification purposes the received message M* will be hashed using the same hash algorithm as on sender side. This auxiliary hash H(M*) must be identical to the received decrypted hash H*(M). If not equal, very obviously something is wrong, either because

1. used public key on sender side does not match → authentication of sender ha failed or
2. received message is not identical with original message → message contents is not trustworthy

Practically, for Internet communication, **mutual TLS** (mTLS), an option of the TLS protocol is used to extend described one-way authentication between IoT client and server. Instead, a two-way authentication of both parties is performed at the same time (using a challenge-response approach) before any data exchange will start.

PUBLIC KEY INFRASTRUCTURE (PKI)

In general, a public key represents an **electronic identity** of a person or an object. With a public key you can verify a digital signature, i.e. authenticity of the sender, but it does not provide any information about the sender itself. This why a **public key infrastructure (PKI)** is required for every application using public-key cryptography.

A PKI consists of a set of policies and a an associated system that manage the creation, distribution, revocation and administration of **electronic identities** incl. corresponding key pairs. PKI is about how two entities can trust each other in order to exchange messages securely. Usually, this is done by means of *delegated trust* using **certificates** issued be a mutually trusted entity, a so-called **certification authority (CA)**. Each certificate links a public key to the corresponding electronic identity, it contains relevant information required for the application to work properly and in a trustworthy and tamper-proof manner.

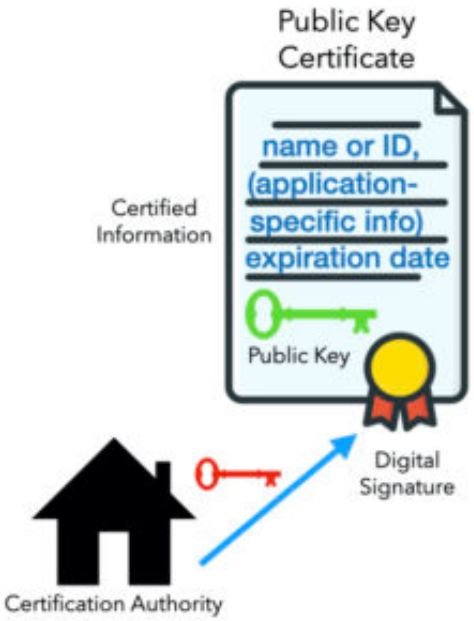

Figure 2: PKI Certificate (© chip-info.com)

A PKI has to specify how new electronic identities are added to the application environment and how to revoke obsolete or expired certificates. An **up-to-date certificate database** must ensure that all valid public keys are available to all participants, i.e. all potential message recipients, e.g. via central online repository.

For large-scale rollouts of complex applications like a national citizen ID card a suitable PKI will be very complex (see Ref. 3), but for many IoT or industrial applications requirements might be much more simple and easier to implement, e.g. you might not need multiple “registration authorities”. Your PKI might be tailored to your application requirements and should be as simple as possible, but at least you will have to consider, specify and implement all relevant aspects how to **handle and deploy public keys** for all communication endpoints within your application environment .

SYMMETRIC AUTHENTICATION

Last but not least we should take a brief look at the classical cryptography using the **same key** on both sender and receiver side (see Fig. 3.). Achievable security does not fall behind public-key crypto schemes as long as secure

algorithms (e.g. AES) with appropriate key lengths are used (see Ref. 2), but – by nature – key deployment is a challenge. Authentication of a network node can be done like this:

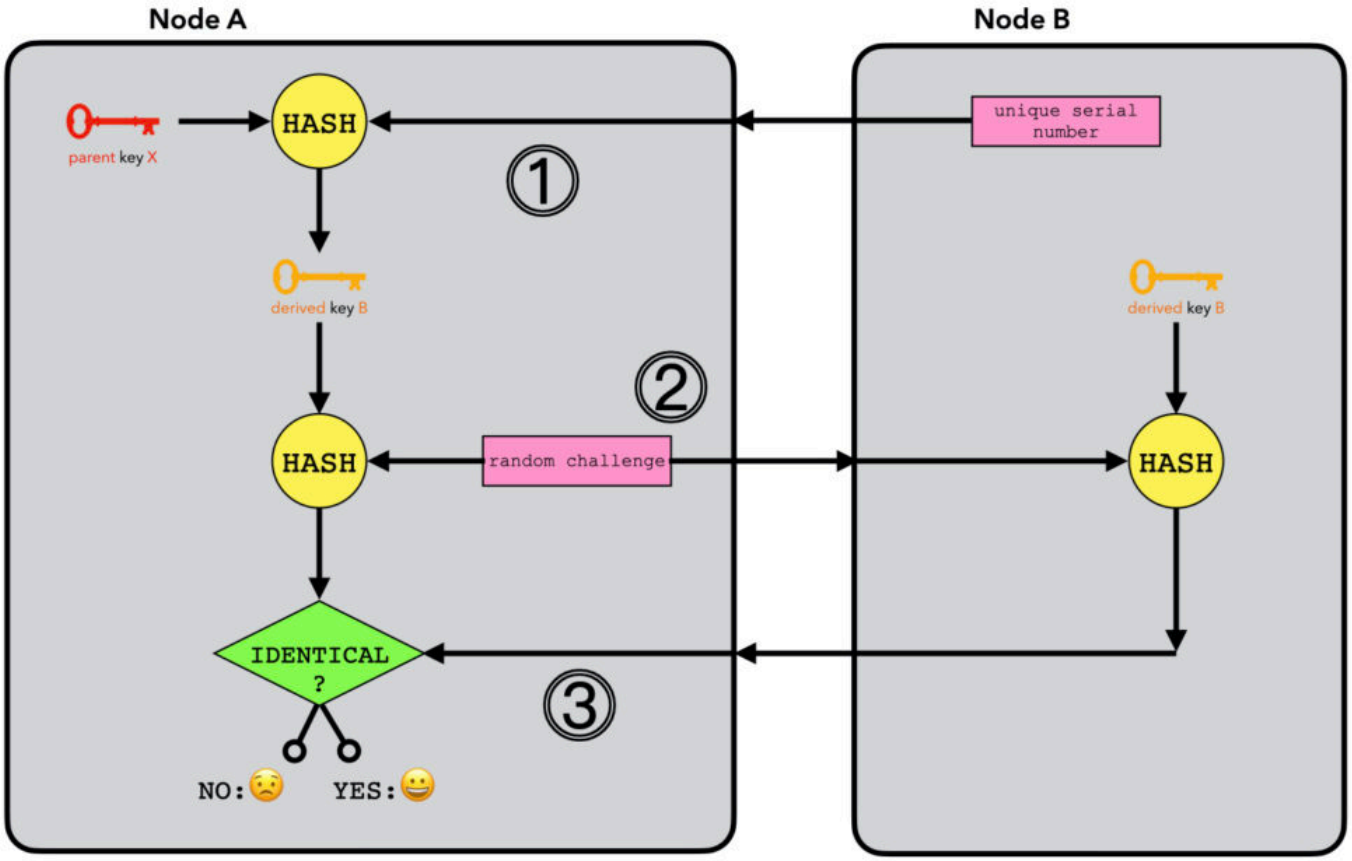

Figure 3: Symmetric Authentication (© chip-info.com)

The idea is to create a shareable key for each IoT node. For this purpose a unique ID and a common secret is needed. Some manufacturers are assigning a **unique serial number** to each die (of a secure element) during production. In addition, all secure elements used in same application X *must have* a pre-installed **common parent key X** to be used as shared secret. Mentioned unique ID of **node B** in combination with the common parent key can be used to calculate a dedicated **derived key B** using a suitable and agreed **hash** algorithm

Now, in the field, to establish a secure connection between **node A** and **node B** the following process can be used:

1. Node A requests node B serial number. Then, node A can calculate derived

key B which does not have to leave the protected environment of a secure element at any time.
2. Then, node A generates a random number ("challenge") which is sent to node B. Node B calculates a hash value from received challenge in combination with derived key B. Result is returned to node A.
3. Node A will perform same operation and compare both hash values. If values are not identical, node B is not authenticated.

This alternate method is simple and an example how symmetric cryptography can be used for some applications. It will work fine and securely as long as mentioned parent key X is present in all deployed IoT devices. No PKI is needed, but you will have to load the patent key securely into all devices and protect related production processes carefully, otherwise application X will at risk. In case mentioned parent key X has been uncovered somehow, you should have a plan how to replace the key and update all devices securely in the field somehow.

PROVEN SMARTCARD SECURITY

A well-known IoT use case is remote access to power meters („Smart Meter") where transmitted consumption data is automatically converted into an energy bill which is addressed to the registered customer – without any human interaction or control. Needless to say that energy providers are interested in an efficient infrastructure, but incorrect data delivery would cause financial loss resp. legal/liability problems. Consequently, these kind of applications require strong protection.

But what does "strong protection" mean? On the market you will find a lot of microcontrollers claiming to offer a "trusted zone" incl. „protected" storage and a „secure" operating system. In an effort to differentiate their products, manufacturers of secure elements are offering further *evidence* – provided by independent crypto experts, e.g. a Common Criteria "CC" certificate (see Ref. 4). In fact, successful security evaluation is mandatory for suppliers of national citizen ID cards or passports, for example.

In many cases, IoT applications will not require a CC certificate, but manufacturers would provide evidence that implemented protection measures offer a a high level of quality. But, of course, **protection measures** and related security evaluations **are not available for free**, i.e. CC-certified products will be more expensive (cost per square mm of chip size). In the end, a secure element will cost 0.50-1.00 EUR (see product overview table below) and related design decisions will have to follow prior **risk analysis** resp. good knowledge of potential attack scenarios and goals. So, is your application at risk because people can fake IoT electronic identities or tamper messages? How much money would you loose in this case?

Once you come to the conclusion that extra cost for an IoT secure element is justified, you should take a closer look at each security certificate provided. Does it cover hardware shielding only, i.e. attacks by physical intrusion? Does it also offer protection against fault attacks or side-channel attacks, e.g. power analysis? See Ref. 5.

As a conclusion, IoT secure elements are offering ultimate protection for a private key, they can generate keys and execute crypto operations extremely fast due to dedicated hardware accelerators. In fact, an IoT secure elements offer perfect end-to-end security for IoT communication channels and other applications requiring strong tamper-protection (see section Target Applications).

PRODUCT OVER

As mentioned before erivatives of a smartcard IC – addre ly a few chip manufacturers. For t ved candidates:

- STMicroelectronic
- NXP A1006
- NXP A71CH
- Microchip ATECC608A