cryptography

Article

Contemporary Physical Clone-Resistant Identity for IoTs and Emerging Technologies

Emad Hamadaqa [1,*], Saleh Mulhem [2], Wael Adi [1] and Mladen Berekovic [2]

1 Institute of Computer and Network Engineering, Technical University of Braunschweig, Hans-Sommer Str. 66, D-38106 Braunschweig, Germany; w.adi@tu-bs.de
2 Institute of Computer Engineering, Gebäude 64, University of Lübeck, Ratzeburger Allee 160, D-23562 Lübeck, Germany; mulhem@iti.uni-luebeck.de (S.M.); berekovic@iti.uni-luebeck.de (M.B.)
* Correspondence: e.hamadaqa@tu-bs.de; Tel.: +49-176-3766-5555

Abstract: Internet of things (IoT) technologies have recently gained much interest from numerous industries, where devices, machines, sensors, or simply things are linked with each other over open communication networks. However, such an operation environment brings new security threats and technology challenges in securing and stabilizing such large systems in the IoT world. Device identity in such an environment is an essential security requirement as a secure anchor for most applications towards clone-resistant resilient operational security. This paper analyzes different contemporary authenticated identification techniques and discusses possible future technologies for physically clone-resistant IoT units. Two categories of identification techniques to counteract cloning IoT units are discussed. The first category is inherently cloneable and includes the classical identification mechanisms based on secret and public key cryptography. Such techniques deploy mainly secret keys stored permanently somewhere in the IoT devices as classical means to make units clone-resistant. However, such techniques are inherently cloneable as the manufacturer or device personalizers can clone them by re-using the same secret key (which must be known to somebody) or reveal keys to third parties to create cloned entities. In contrast, the second, more resilient category is inherently unclonable because it deploys unknown and hard to predict born analog modules such as physical unclonable functions (PUFs) or mutated digital modules and so-called secret unknown ciphers (SUCs). Both techniques are DNA-like identities and hard to predict and clone even by the manufacturer itself. Born PUFs were introduced two decades ago; however, PUFs as analog functions failed to serve as practically usable unclonable electronic identities due to being costly, unstable/inconsistent, and non-practical for mass application. To overcome the drawbacks of analog PUFs, SUCs techniques were introduced a decade ago. SUCs, as mutated modules, are highly consistent, being digital modules. However, as self-mutated digital modules, they offer only clone-resistant identities. Therefore, the SUC technique is proposed as a promising clone-resistant technology embedded in emerging IoT units in non-volatile self-reconfiguring devices. The main threats and expected security requirements in the emerging IoT applications are postulated. Finally, the presented techniques are analyzed, classified, and compared considering security, performance, and complexity given future expected IoT security features and requirements.

Keywords: internet of things; IoT security; clone-resistant entities; physical unclonable function PUF; secret unknown cipher SUC; public key cryptography; authentication; identification; secret key identification

Citation: Hamadaqa, E.; Mulhem, S.; Adi, W.; Berekovic, M. Contemporary Physical Clone-Resistant Identity for IoTs and Emerging Technologies. *Cryptography* **2021**, *5*, 32. https://doi.org/10.3390/cryptography5040032

Academic Editor: Jim Plusquellic

Received: 16 September 2021
Accepted: 27 October 2021
Published: 9 November 2021

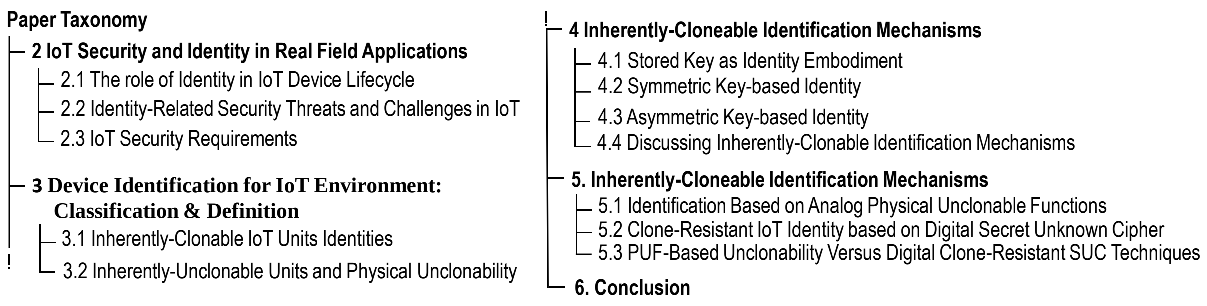
1. Introduction

The Internet of things (IoT) is an essential enabler of the next industrial revolution in the digital world. IoT allows everyday objects (or things) to be connected to the open internet network by equipping such devices with various sensing, networking, and processing capabilities. Such capabilities enable things to communicate with each other and with additional devices or cloud services over the open network environment to perform IoT application tasks. Unfortunately, such IoT objects/items can be replaced, in order to abuse the system. According to the US Department of Homeland Security, physical security solutions are required by regulators to secure and protect IoT devices and their integrity [1]. Therefore, IoT units are required to be physically unique and individually securely identifiable (unclonability is necessary) to prevent abuse of the system. In the context of Industry 4.0, IoT technologies and concepts are also made to be usable for the industrial environment. The "industrial Internet of things" (IIoT) refines the IoT with higher-quality devices, even with reduced functions, and more sensitive sensors that deliver precise and trustworthy information. Efficiency, cost reduction, and fast processes with as much flexibility as possible are the main goals of IIoT. IoT applications require high reliability and integrity for the data reported by the IoT units.

To attain that goal, resilient physical unclonable identifiers of IoT devices are needed to secure the system against dangerous replacement attacks. A single replacement of an abused device can cause an entire system to collapse. According to International Data Corporation (IDC) forecasts [2], currently, around 3.8 billion smartphones are connected to the Internet and operating worldwide, and overall up to 55.7 billion networked devices, vehicles, and machines are expected to be operating on the same open network by 2025, all of which are exposed to remote attacks globally. IoT electronic tends to deploy low-end chips that are relatively small and inexpensive, so they offer limited memory and processing power for sophisticated security functionalities. Hence, most contemporary IoT devices operate insecurely on the open internet and are highly vulnerable to theft, abuse, and hence possibly to a complete system breakdown. Unfortunately, the contemporary security mechanisms do not work as required [3]. Such security gaps need to be fixed before other massive attacks on smart homes, intelligent transportation systems, smart buildings, smart cities, smart grids, etc. are encountered. The traditional secret- and public-key techniques deploying keys stored in volatile or non-volatile memory do not offer real unclonability.

In this paper, two classes of cryptographic mechanisms for physical device identification are investigated and compared. The first is inherently-cloneable and includes the relatively weak and primitive traditional secret- and public-key techniques. The second is inherently-unclonable and includes the analog physical unclonable functions (PUFs) [4], in addition to our proposed clone-resistant digital secret unknown cipher (SUC) technique [5]. The public and secret-key solutions deploy simple memory storage accommodating the IoT device identity as a secret key with some cryptographic protocols, which is inherently clonable as somebody knows the key memory contents. In comparison, PUF and SUC provide hardwired unknown fingerprints without storing any secrets known to a person. Therefore, such techniques are basically clonable. Both PUF and SUC techniques provide a DNA-like resilient identity concept that is hard to model, unique and unclonable, and hard to clone clone-resistant. Regarding future mass production, the SUC technique seems to be a promising technique to make emerging IoT units clone-resistant through highly a resilient digital identity at reasonable cost. Figure 1 shows the full picture of taxonomy in this work.

The remainder of this paper is organized as follows. In Section 2, we discuss the IoT security and identity features and requirements. In Section 3, we introduce and define the IoT device's clonability and clone-resistance concepts. In Section 4, inherently cloneable solutions are presented, and in Section 5, the inherently unclonable solutions are investigated and compared. Finally, Section 6 concludes the paper.



Paper Taxonomy

- 2 IoT Security and Identity in Real Field Applications
  - 2.1 The role of Identity in IoT Device Lifecycle
  - 2.2 Identity-Related Security Threats and Challenges in IoT
  - 2.3 IoT Security Requirements
- 3 Device Identification for IoT Environment: Classification & Definition
  - 3.1 Inherently-Clonable IoT Units Identities
  - 3.2 Inherently-Unclonable Units and Physical Unclonability
- 4 Inherently-Cloneable Identification Mechanisms
  - 4.1 Stored Key as Identity Embodiment
  - 4.2 Symmetric Key-based Identity
  - 4.3 Asymmetric Key-based Identity
  - 4.4 Discussing Inherently-Clonable Identification Mechanisms
- 5. Inherently-Cloneable Identification Mechanisms
  - 5.1 Identification Based on Analog Physical Unclonable Functions
  - 5.2 Clone-Resistant IoT Identity based on Digital Secret Unknown Cipher
  - 5.3 PUF-Based Unclonability Versus Digital Clone-Resistant SUC Techniques
- 6. Conclusion

Figure 1. Organization of the paper.

2. IoT Security and Identity in Real Field Applications

This section will discuss the most relevant security and identity aspects in a real field IoT application. First, we present the role of identity in the IoT device lifecycle, followed by security threats and challenges in IoT environments. Security requirements are investigated showing the necessity for unclonable identification and authentication for IoT devices.

2.1. The Role of Identity in IoT Device Lifecycle

Setting up IoT device identity: Figure 2 shows the lifecycle of a device in IoT, including the bootstrapping, operational, and maintenance phases [6]. The device identity plays an essential role in all phases of the IoT lifecycle. The benefits of such an identity can be perceived as follows:

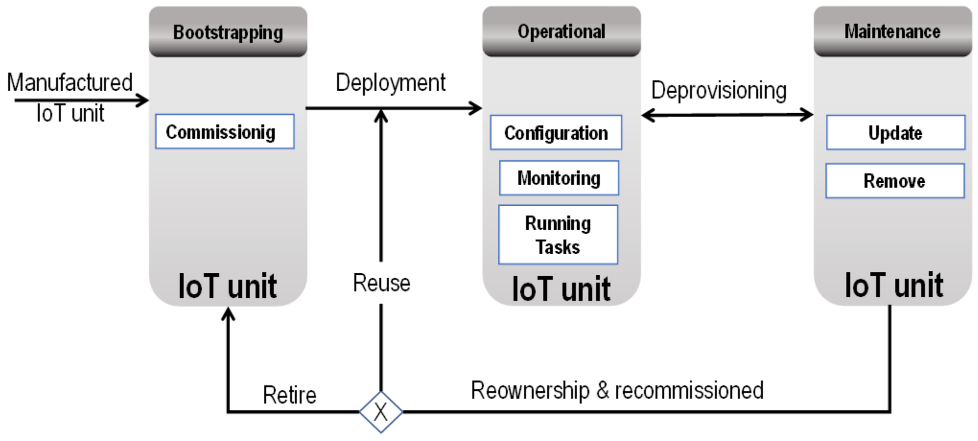


Figure 2. IoT device lifecycle.

Device Bootstrapping: Currently, IoT devices are produced by different manufactures. Various fabrication processes raise challenges in approaching the bootstrapping of the required security footprint in such devices to ensure trusted communication between nodes from the beginning of active operations. One of the first tasks that should take place during the bootstrapping phase is to set up a unique identifier or identity for the IoT device. The secret keys or passwords used during regular operation are made available to the device in this phase. With the device's production, the device is brought into active life during the bootstrapping phase. As a result, the thing is initiated and commissioned by a responsible administrator in a network.

Device Authentication during the Operational Phase: After the device is ready for operation, the device and the system of things are ready to perform the functions of the IoT system, requiring strong, scalable, and cost-effective identification and authentication between IoT entities. The application execution represents the operational phase, in which the IoT device is authentic and functioning as intended.

During the Maintenance Phase: the device's software often needs to be updated, or the device needs to be reconfigured. The device continues through the operation phase and eventual maintenance phase until it is decommissioned at the end of its life cycle. The end of a device's life does not necessarily mean that it is defective, but rather that the IoT system needs to be upgraded to the next generation to provide additional functionality. In some cases, the IoT device (such as a sensor or actuator) can be removed and re-commissioned to be used in another IoT system by restarting the lifecycle. As a result, replacing clone-resistant IoT devices in a secured manner needs careful management to deal with unclonable physical identities throughout the whole life cycle.

2.2. Identity-Related Security Threats and Challenges in IoT

To design and implement complete security solutions for IoT systems, identifying threats and challenges for all IoT networks, IoT devices, and IoT applications is of great importance. We list the identity-related IoT security threats in Table 1, as provided by the Internet Engineering Task Force (IETF) [6].

Table 1. IoT security threats identified by (IETF) [6].

	Security Threats	Description
1	Cloning of things	An untrusted vendor can easily clone the physical characteristics, firmware/software, or security configuration of an IoT device during the manufacturing process. Running devices can also be compromised and their software reverse-engineered to allow cloning or software modifications; therefore, a clone-resistant physical identity is a must.
2	Malicious substitution of things	A fake device, including a fake identity, replaces a genuine device during installation without being detected.
3	Elevation of privilege	An attacker with low privileges can exploit weaknesses in the implemented authentication and authorization mechanisms of an IoT device to gain more privileged access to the device and its data.
4	Eavesdropping attack	Operation of a device on a network may be vulnerable to eavesdropping, typically when operative keying materials, security parameters, or configuration settings are exchanged in clear text over a wireless network or when device authentication is not enabled.
5	Covert channels	Network covert channels are used to hide data in legitimate transmissions in communication networks by deploying different network protocols as carriers and hiding confidential data from network devices. In [7], Lampson introduced covert channels, divided into storage and timing channels. The technique of covert storage channels is a process that writes (directly or indirectly) to a shared resource while another process reads from it. In the context of network steganography, covert storage channels hide data by storing it in the protocol header and or in the protocol data unit. On the other hand, timing channels hide data by deploying event timing, e.g., by sending the same protocol data unit multiple times or by changing the order of packets. Several countermeasures can be performed to prevent covert channel attacks based on identifying and detecting these ad-hoc, or by using a formal method [8]. However, if the legitimate network users are the only uniquely identified users, then the probability of successful detection of convert channels is significant high. Therefore, an inherently-unclonable IoT identification mechanism is required to prevent and mitigate such attacks.

It is notable that unclonable identity is essentially relevant and needed to prevent cloning and malicious substituting of things. It plays the role of trusted anchor for the whole IoT system. This trusted anchor prevents unauthorized access to IoT devices and is considered the first countermeasure against several attacks, such as eavesdropping.

2.3. IoT Security Requirements

The standard IoT security architecture includes three layers: device-, network-, and application layer. The security functions accommodating each of these layers need to be adapted together to secure the whole IoT system. However, each layer's security requirements are different; therefore, we will discuss the layer-specific security requirements with the focus on identity-related IoT security aspects:

2.3.1. Device Physical-Layer Operational Security Requirements

The device level is concerned with people, devices, and geolocations, so physical security should be anchored into the device to secure all processes performed. The device layer's fundamental security requirements include secure booting, firmware updates, software updates, authorization, and authentication. IoT devices shall provide a cryptographic unique and secure identifier to protect the identity from modification and replacement and serve as individual physical device authentication. The physical device identity may play the role of a trusted anchor for the security requirements. All required operational security keys can be derived from the device identity; this ensures the device's trustworthiness. The following application-related essential operations require resilient identity-related security functions to ensure the security stability of the whole system:

(A) Secured Booting

When turning on the IoT device, the installed software's integrity and authenticity should be checked to ensure that only authorized applications can run on the specific device by deploying non-clonable keys coupled with that device. The provided key should be unique and unclonable for each device, otherwise, if one device's secure boot key is compromised, the secure boot of all the other devices using that same key is also compromised. To eliminate this risk, a device-specific secure boot can be deployed. Traditional solutions provide a unique key for every device stored in an NVM; however, the solution is prone to some read-back attacks [9]. A better solution is to deploy a clone-resistant device ID such as a PUF [10–13] in conjunction with the secure boot mechanism. Our proposed solution is introducing a digital SUC-based secured booting as proposed in [14], which is more resilient and practical for mass production than the analog PUF-technology solution seen in Section 5.1.

(B) Secured Updates

IoT devices require software patches and updates due to bug fixing or to roll out new releases to improve functionality. IoT devices should only install signed/authenticated software patches to avoid malicious activities. The signing mechanisms should make use of the device's clone-resistant identity as a unique and robust signature.

(C) Secured Access Control

Access control mechanisms are required to define the access of applications and device components in an IoT environment [15]. The implementation of access control should be isolated such that compromised information can be restricted to specific, compromised network regions. This procedure requires a unique and unclonable device identity as a reference foundation stone to provide the eligible robust access control.

(D) Secured Device Authentication

As soon as new devices are connected to a network, they should identify and authenticate each other via a secure and unique and unclonable/non-replaceable device identifier. There is a need to adopt a physical authentication mechanism for IoT devices so that device spoofing in an IoT environment can be significantly nullified.

2.3.2. Security Requirements on Network Layer

Network layer security mainly refers to the gateway that is used between the IoT devices and the Internet. Gateway devices are mainly subject to physical intrusion or replacement attacks; therefore, the identity protection of the IoT gateway is a primary requirement. To ensure the protection of the IoT gateway, each IoT gateway should be able to identify itself uniquely via the network. The network designer's goals should ensure that a clone-resistant identity-based key agreement protocol is devised to operate on the network layer to protect the gateway against malware by using access control lists and filtering linked to the individual device identity.

2.3.3. Security Requirements on the Application Layer

The service layer in an IoT system deals with the device interactions that occur when data are collected and presented to the user and control commands are sent to these IoT devices. The service layer handles the communication between the device and gateway layer. The interaction should proceed in such a way that the changes made by users and devices cannot be refuted. An inspection track mechanism should be operationally embedded and cover this non-repudiation of the user and device changes. Therefore, identity-based protection mechanisms should be linked to this layer to enable the security of the application layer.

Concluding the need for physical unclonable identity: The above security concerns and requirements show that there is a need for solid unclonable physical identification and authentication for IoT devices. Each device requires a physical, unclonable identity embedded in the device in a secure environment, and which is, as a minimum, not visible from the outside. Therefore, robust physical device identity plays an essential role in protecting both IoT devices and the overall IoT systems.

IoT device unclonability needs to be discussed further in this context. We explicitly distinguish between inherently-clonable and inherent-unclonable identities.

3. Device Identification for IoT Environment: Classification and Definition

Referring to Figure 3, identifying IoTs in a large IoT network can be classified in two categories. The first traditional category is inherently-cloneable allowing production of multiple IoT objects that have the same provable identity. The second category includes all identities which are inherently-unclonable and do not allow the production of multiple objects with the same provable identity. This identity-category is equivalent to biological DNA identity, which inherently does not allow production of two entities with the same identity, due to its natural creation process.

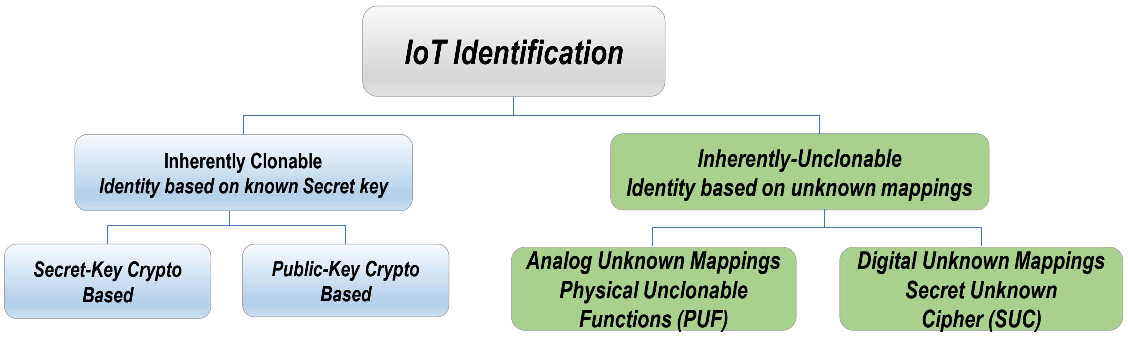

IoT Identification
Inherently Clonable
Identity based on known Secret key
Inherently-Unclonable
Identity based on unknown mappings
Secret-Key Crypto Based
Public-Key Crypto Based
Analog Unknown Mappings Physical Unclonable Functions (PUF)
Digital Unknown Mappings Secret Unknown Cipher (SUC)

Figure 3. The two categories of IoT device identification mechanisms.

3.1. Inherently-Clonable IoT Unit Identities

As shown in Figure 2, all identification mechanisms which basically allow the originator or the trusted authority to produce traceable duplicates result in virtual units. Such units are usable in all applications where producing a duplicate does not violate the basic security requirements. For example, production of a duplicate sim card by a mobile operator is permitted and required. However, producing two physically equal personal identity cards is abusing the system and is unacceptable, and it should be fully prohibited even for the persons personalizing the identity cards. In that case, producing any physical identity is traceable, as the personalizing worker cannot produce two physically equal cards even if he or she wants to. In addition, if it is done, it cannot happen without being traced.

All traditional secret-key and public-key identification mechanisms fall under this category of basically-cloneable identity mechanisms. The reason is that the person who generates the identity knows a secret, enabling him/her to generate a replicate for the same object without being traced.

3.2. Inherently-Unclonable Units and Physical Unclonability

Physical clonability (PU) of IoT units indicates that inherently, no one can configure/produce two devices/units with the same physical and provable identity. The unclonable property prevents even the manufacturer from producing identical IoT devices during the manufacturing process. The unclonable device identity is essentially deployed as a trusted anchor that cannot be physically replaced. It supports several security features such as firmware integrity, software execution protection, the device's security configuration, etc. PU can be seen as a key property of IoT units that cannot be achieved by software or algorithmic solutions. The requirement of physical clonability means that a physical clone is not technologically or financially feasible according to the current state of the art. According to [16], PU was defined mainly in the context of PUFs as practically hard to clone entities. In the same sense, physical unclonability of an IoT Unit (PUU) can be classified in three levels in a similar manner in reference to [16] as:

(A) Basic Unclonability Measure

Definition 1. *A class of IoT units exhibits physical unclonability if it is hard to produce two distinct IoT units with the same identity.*

The qualifier "hard" in the above definition reflects the physical and technical difficulties (or impossibility) of creating such an IOT unit clone. These difficulties must be evaluated in terms of the technical capabilities of the adversary, which are ultimately a function of their expertise and equipment budget.

Let C and R be any challenge and response of specific IoT unit identity functions such as A. Assume that A' is an identification function clone of this IoT unit. The PUU can be mathematically described as follows:

$$P\big[R \leftarrow A(C) = R \leftarrow A'(C)\big] < \varepsilon \text{ where } \varepsilon \text{ is very low} : \textit{For all } C - R \textit{ pairs}$$

If # C-R pairs $\rightarrow \infty$, then a unit can be claimed to be theoretically unclonable.

(B) Mathematical Unclonability (Modeling-Impossible)

In this respect, uniqueness and PU is often not sufficient to ensure security. One also needs unpredictability between responses to a single identification function instance, i.e., unobserved responses remain sufficiently random even after observing responses to other challenges on the same entity. In context, an attacker is restricted to learning a limited number of (possibly random) challenge–response pairs that it can use to train its prediction algorithm. Typically, this is the case in a challenge–response-based protocol. However, a more robust adversarial model must be considered when the attacker has unlimited physical access to an entity. He or she can learn any number of challenge–response pairs and may even make practical observations beyond the identification function.

Definition 2. *A class of IoT units exhibits mathematical unclonability if the response of its identification function is unpredictable.*

Mathematical unclonability is the extension of unpredictability to an attacker with unlimited physical access to an identification function. This implies that mathematical unclonability leads to unpredictability.

(C) True Unclonability

We have defined two different notions of unclonability: physical and mathematical unclonability. Both describe a property with the same objective, i.e., to make it hard to clone an identification function, but from entirely different perspectives. Physical unclonability deals with the actual physical cloning of identification functions, while mathematical unclonability deals only with cloning the challenge–response behavior of *an* identification function. To ensure "true unclonability" for a class of IoT units, both physical and mathematical unclonability measures need to be satisfied.

Definition 3. *A class of IoT units exhibits true unclonability if it is both physically and mathematically unclonable.*

From the above arguments, secured identity becomes an increasingly essential IoT device requirement as an anchor to establish overall solid system security. A comprehensive comparison between traditional inherently-cloneable and inherently-unclonable techniques will be investigated further in the following sections according to Figure 2: in Section 4, devices deploying a key stored in a memory as an "inherently cloneable identity" are presented and in Section 5, devices deploying inherently unclonable identity are investigated. These cover the following two types: first, a born unknown analog function as PUF represents the unit's identity. Second, a new proposed category of "digitally-mutated" unknown digital functions replaces PUFs as embedded/hardwired modules in devices called secret unknown ciphers (SUCs) as clone-resistant identities.

4. Inherently-Cloneable Identification Mechanisms

Traditionally, basic identification and authentication of IoT devices require storing a unique device-key/identity in each device. This key is usually deployed together with a cryptographic primitive to perform identification/authentication protocols [17]. The motivation of using a memory-store for identity can be summarized as follows [18]:

- Memory is available in almost all computing platforms.
- The accessibility and flexibility of the memory is very high as it exhibits a short time-delay.
- Memory is one of few reliable system-components.

Further, two types of memory are proposed as critical secret storage: non-volatile memory (NVM) as permanent key storage and volatile memory as temporary key storage. In order to use the stored key as a permanent device identity, NVM is desirable. Three categories of NVMs have been introduced in several applications as follows [11]:

- Read-only memory (ROM).
- One-time programmable (OTP) NVM.
- Multiple-time programmable (MTP) NVM such as flash memory.

In the following, NVM-based IoT identification mechanisms are investigated carefully. Such a mechanism consists of two phases; first: identity establishment and second, identity verification.

4.1. Stored Key as Identity Embodiment

Many methods are deployed to establish and verify the key/identity among several IoT devices. In reference to Figure 4, two mechanisms are mainly deployed to create and verify the device's stored key/identity. First, symmetric key-based mechanisms, where the key can be established and verified as discussed below. In addition, and secondly, asymmetric key-based mechanisms, where each IoT device should have a public and a private key. The private key is usually generated by using a true random number generator (TRNG).

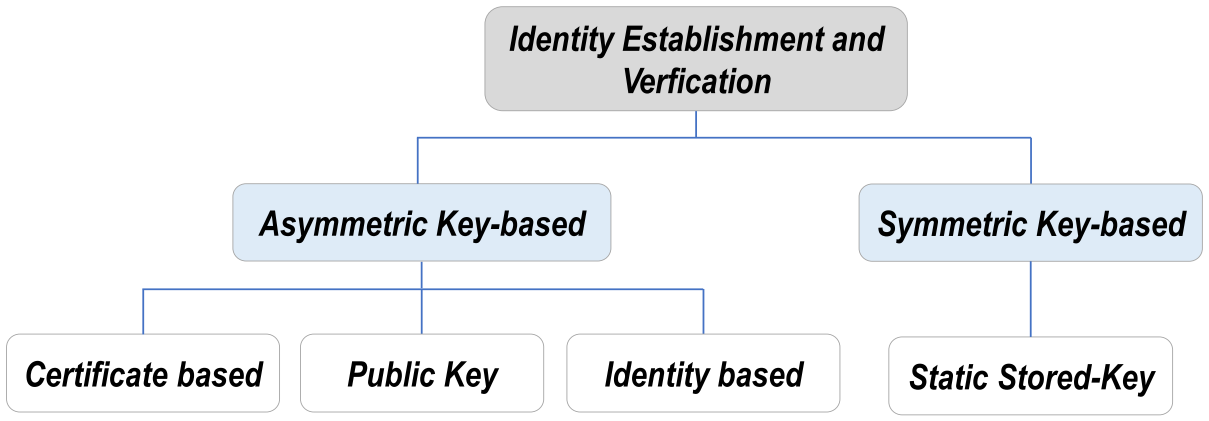


Figure 4. IoT identity by deploying key-storage enrollment and verification mechanisms.

4.2. Symmetric Key-Based Identity

Key Establishment: In the first scenario, one device shares a random, yet unique value (to avoid duplicates) generated by a true random number generator (TRNG) with other devices. All devices store their own random value in NVM and use it as a secret key/identity. In the second scenario, all devices utilize a pseudorandom number generator with the same shared seed to simultaneously generate the same random value. Here, the generated value can be used as a key/identity; however, the shared seed and generated value are required to be stored in NVM.

Key Verification Mechanism: In such schemes, communicating parties share a common secret key within the encrypted or decrypted exchanged messages, where the secret key should be stored permanently in NVM. Such schemes are also known as a shared key, single key, or secret key scheme. First, the involved communicating parties take common credentials, which can be the symmetric key, along with some random bytes previously deployed in the IoT device. The symmetric key is assumed to be used only for communication with the intended devices; such schemes ensure implicit authentication in the communication. These schemes can also use a server as a key distribution center (KDC) to distribute the keys to IoT devices, and symmetric schemes provide a low computational overhead, which is suitable for constrained sensing devices such as IoT.

Figure 5 illustrates how two IoT devices (prover and verifier) perform a basic authentication protocol. The prover and verifier share the same secret key during the phase of key/identity establishment, and they deploy the same agreed-on hash function H. In a similar three-way protocol, the verifier may authenticate the prover by challenging and asking him to respond by encrypting r by a cipher E using the shared secret key. The verifier then checks if the response decrypted using the shared secret key results in r', which should be the same as the challenged random r. Many other refined and extended techniques are derived from the above two primitive three-way challenge–response identification protocols.

In conclusion, symmetric-key schemes have major disadvantages, such as that the whole security resides with the trusted third party (Trusted Authority TA). In addition, the use of symmetric (or secret) cryptographic keys raises the problem of how to securely preconfigure or transfer such keying material into the device. Moreover, the symmetric key-based identity does not fulfill the IoT unclonability requirements as a must for resilient and solid overall system security.

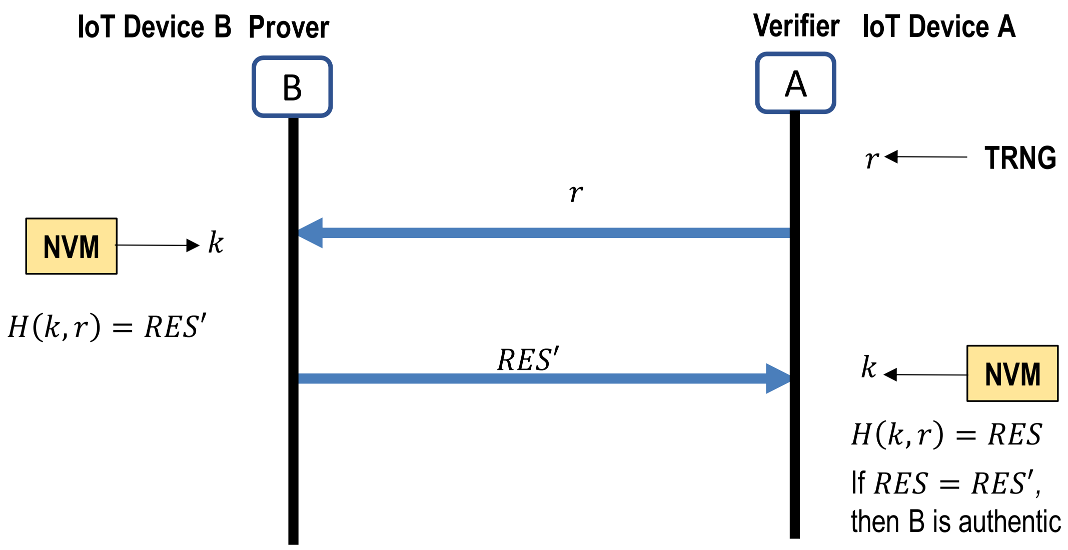


Figure 5. Hash-based challenge–response symmetric-key verification.

4.3. *Asymmetric Key-Based Identity*

Public key cryptography (PKC) enables IoT device manufacturers to embed a cryptographically verifiable identity into each device to ensure secure access from and to the IoT device.

4.3.1. Asymmetric Key-Based Establishment and Verification Mechanism

Asymmetric key identification schemes are based on PKC and use two types of keys: public keys and private keys. As the name implies, the public key is known to all communicating parties, whereas the private key must be kept secret for each communicating device. The two keys are mathematically related; however, deriving a private key from the corresponding public key is mathematically infeasible. The relationship between the two keys involves expensive mathematical operations such as exponentiation and integer factorization. Since such mathematical operations are computationally complex and high energy-consuming, asymmetric algorithms are not suitable for performing on large amounts of data, making such algorithms particularly attractive for IoT devices with constrained devices. RSA and ECC as well-known asymmetric algorithms are widely used on the conventional Internet with a maximum data size of 1024-bit.

PKC-based identifiers are very scalable. However, the current technological challenges in IoT are leading to new design requirements for PKC. For instance, the desired public key cryptosystem should offer additional technological requirements such as low complexity, low power consumption, and less latency. Therefore, the designed cryptosystem needs to consume minor resources without compromising the required level of security. Such design requirements have played a crucial role in introducing and developing a new cryptographic paradigm, the so-called lightweight public-key [19]. The current lightweight PKC still requires a special mechanism to update the public- and private-keys. In recent years, however, much research has focused on optimizing the expensive PKC operations for IoT devices.

Identity-Based Cryptography

Adi Shamir introduced identity-based cryptography (IBC) in 1984 [20]. IBC uses open user identity attributes such as phone numbers or email addresses or device serial numbers to verify signatures. In [21], Boneh and Franklin expanded the concept into a fully functional identity-based encryption scheme. Currently, the IBC is also described in the well-known standard IEEE 1363.3 [22]. This technique requires zero configuration by the receiver party. The IBC relies on a trusted third party known as the private key generator (PKG).

Referring to Figure 6, the PKG generates the private key before the operation and sends the key to the respective entity. However, it should be noted that identity-based systems are vulnerable to key escrow attacks since the PKG knows the private keys of all entities. Another known issue is to ensure a secure connection between the PKG and the IoT device since the private key is passed from the PKG to the IoT device over this connection. This technique offers inherently-clonable identity as somebody knows the identity secrets.

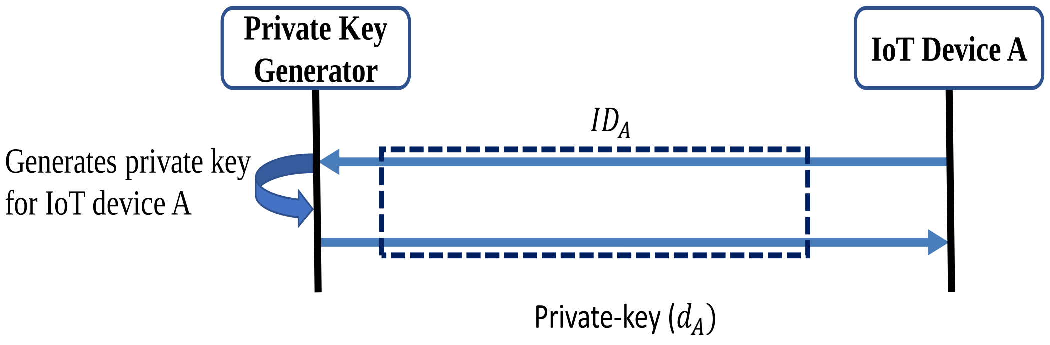


Figure 6. Identity-based Key Generation.

To verify the static stored-key as identity, IBC was implemented by deploying RSA, but ECC and EIGamal public-key locks were also explored for implementation. In IoT communication environments, IBC based on ECC has been extensively investigated, as it offers a more cost-effective alternative to RSA. An excellent approach based on IBC was recently published by Saeed et al. [23]. The authors introduced an identity-based authenticated key agreement between client sensor nodes and cloud servers. The base station hosts the PKG, which supports ID-based schemes that issue private-public key pairs and other system parameters based on their identities to the communicating nodes on the network. On the other hand, the cloud and sensor nodes generate their Diffie–Hellman keys using random integers and ECC-based curve multiplication. These ephemeral keys are signed by the sensor and the cloud application using a one-way hash function and then exchanged with the other party. As shown in Figure 7, after verifying each other's ephemeral key, the communicating nodes conclusively derive the shared key. This technique guarantees perfect forward secrecy, key confidentiality, key control, and scalability. It also relieves the PKG of complete key dependency because even if the adversary compromises the PKG, he or she cannot derive the shared key.

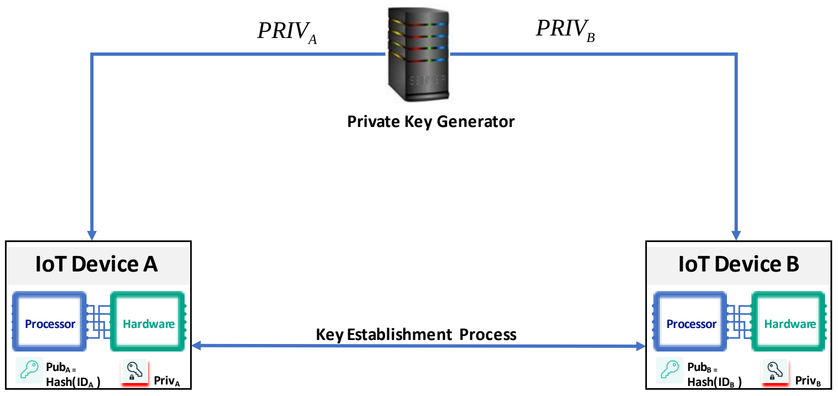


Figure 7. Identity-based cryptographic schemes.

Verifying Public Key Identity

The public key (PK) and the private key are generated and pre-installed on devices, either offline or through an out-of-band mechanism. Therefore, authentication is ensured offline or when an out-of-band mechanism binds the public key to the entity/identity which the key represents. To reduce the certificate burden on resource-constrained devices and to increase efficiency, the use of raw public keys for TLS and DTLS has been standardized by the IETF [24]. In addition, even though these schemes require less messaging than certificates and identities, they can only be used for small network scenarios where each node's public key is known in advance to all other nodes. As an example of using a public key as identity, the authors in [25] proposed a public key-based identification and authentication scheme for heterogeneous IoT networks on software-defined networking (SDN). In Figure 8, the central SDN controller translates the different technology-specific identities from the various IoT domains into a common identity based on virtual IPv6 addresses and authenticates devices and gateways using public and private keys. The authors assumed that the public key of the SDN controller is hardcoded in each device when the device is manufactured; the controller generates the public keys for things using ECC. The gateway also generates its pair of public/private keys using ECC.

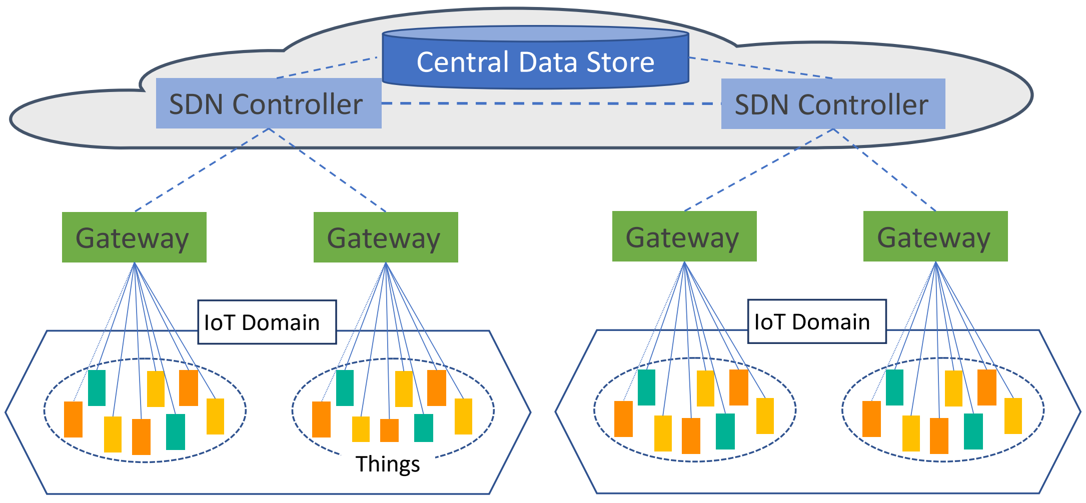


Figure 8. Public key-based authentication scheme for the IoT.

Managing Certificate-Based Identity

From a security standpoint, it is well known that one of the best approaches to authenticating public keys is to have the various entities participate in a public key infrastructure (PKI). A PKI defines, in practice, the set of policies and procedures to manage public key encryption and other services such as the creation, distribution, management, storage, and revocation of digital certificates. A PKI ensures authentication of public keys of users and devices by binding them to their identities. In a PKI, a trusted third party, known as the certificate authority (CA), holds responsibility for registration and issuance of certificates to the various entities. The third component in a PKI repository stores certificates and certificate revocation lists (CRL).

The digital certificates issued by the CA are verified by a chain of trust, and to map the services of a PKI in each IoT environment, the root node can act as the root CA responsible for the registration, issuance, storage, and revocation operations. A certificate has three major constituents: the identification data, a public key, and a digital signature that binds the public key to the identity of the IoT device/user. We also note that certificates may be managed implicitly or explicitly:

- Explicit or conventional certificates are managed and signed by a trusted third party (a CA). Any entity in the network can validate the certificate by verifying the signature of the CA contained in it. This process is illustrated in Figure 9, where Cert is the IoT device certificate including certified signature (Sign) by CA, device identity (ID), and device public key (Pub).
- Implicit certificates are another variant of the public key certificate, where all the certificate components such as identification data, a public key, and digital signatures are superimposed on one another in such a way that the size of the certificate is equal to the size of the public key [26]. Compared to explicit certificates, in the context of which the certificate components are distinct elements, the size of implicit certificates is considerably smaller because digital signatures are superimposed on the public key. The fact that this type of certificate is called implicit is related to the fact that the public key can be extracted and verified from the signature portion of the digital certificate [27]. This makes implicit certificates faster than conventional certificates. In [28], the authors propose that implicit certificates are preconfigured in each device by the network administrator before activating the network. Figure 10 shows the steps in the implicit certificate-based mutual authentication protocol.

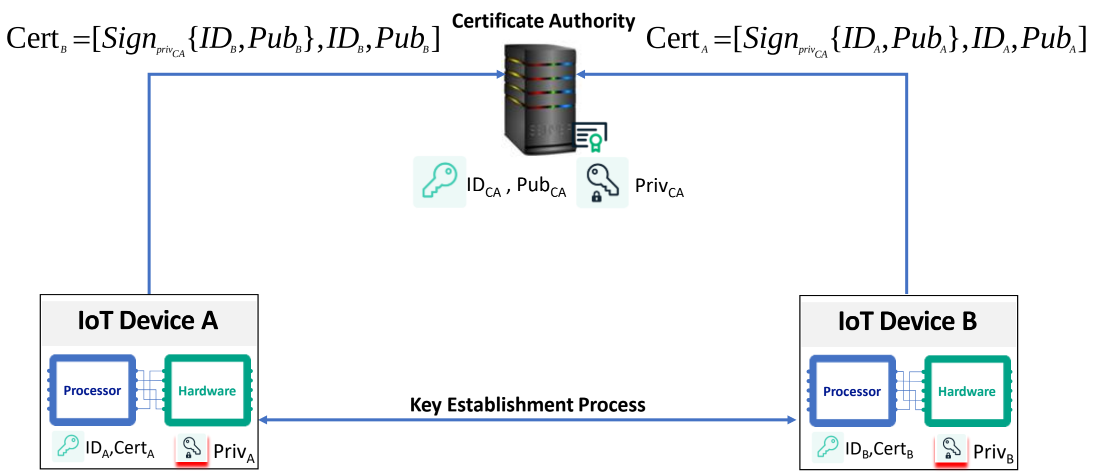


Figure 9. Certificate-based IoT device identification mechanism.

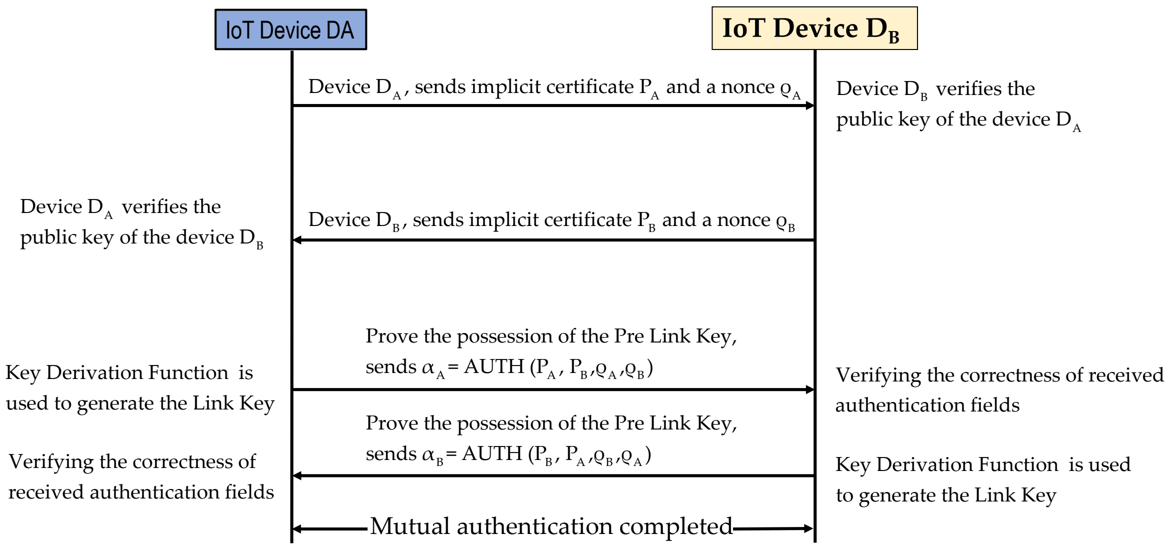

IoT Device DA
IoT Device D_B
Device D_A, sends implicit certificate P_A and a nonce ϱ_A
Device D_B verifies the public key of the device D_A
Device D_A verifies the public key of the device D_B
Device D_B, sends implicit certificate P_B and a nonce ϱ_B
Prove the possession of the Pre Link Key, sends α_A = AUTH (P_A, P_B, ϱ_A, ϱ_B)
Key Derivation Function is used to generate the Link Key
Verifying the correctness of received authentication fields
Prove the possession of the Pre Link Key, sends α_B = AUTH (P_B, P_A, ϱ_B, ϱ_A)
Verifying the correctness of received authentication fields
Key Derivation Function is used to generate the Link Key
Mutual authentication completed

Figure 10. Implicit certificate-based mutual identification protocol.

Furthermore, identity establishment is also defined in IEEE 802.1AR. In this standard, a secure identifier is considered as the most promising approach for a secure device ID in IoT systems. IEEE 802.1AR defines the security credentials to be used for device identification and a device ID module with the interface to use and manage the identifiers. The 802.1AR identifier is based on X.509v3 certificates which, for example, can be used by 802.1X authentication protocols. Manipulation requires the private CA key; depending on the intended security level, binding of the identity to the device can be supported by the hardware; in this case, it is tough to remove or copy the identity and clone the device.

In conclusion, asymmetric-key schemes have similar major disadvantages to secret-key schemes in that the whole system's security resides with the trusted third party (trusted authority—TA). In addition, the use of secret and public keys (or secret) raises the problem of how to preconfigure or transfer such keying material into the device securely and avoid clonability attacks. Moreover, asymmetric key-based identity does not fulfill the IoT unclonability requirements as a must for resilient and solid overall system security.

4.4. Discussing Inherently-Clonable Identification Mechanisms

As shown above, two schemes were investigated targeting identification mechanisms in IoT environment. First: symmetric-key schemes exhibit several drawbacks as discussed in Section 4.1 Second: asymmetric key schemes have low memory requirements, high scalability, and resistance to attacks. They use computationally intensive operations that increase energy consumption and computation costs connected with IoT applications. Table 2 summarizes the results for the inherently-clonable identification mechanisms based on comparison criteria.

Table 2. Comparison criteria of inherently clonable identification mechanisms for IoT security services.

Features	Result
Suitable for IoT constrained devices	Fulfilled
Perfect forward secrecy	Fulfilled
Resilience to physical attack	Not fulfilled
Resilience to modeling attack	Not relevant
Resilience to cloning attack	Not fulfilled
High performance/response time	Fulfilled [11]
NVM usage	YES
Cryptographic primitive usage	YES
Unclonability	Not Fulfilled
Key space	Scalable
Key Entropy	HIGH
Key update needed?	YES
Identification/authentication	Fulfilled
Mutual authentication	Fulfilled

The clear clonability disadvantages of both schemes according to Table 2 motivate the search for alternatives, which should be physically unclonable or clone-resistant techniques to counteract physical attacks in many relevant IoT applications.

5. Inherently-Unclonable Identification Mechanisms

In this section, the inherently-unclonable identification technologies and mechanisms will be investigated: first, devices using physical unclonable functions (PUF) as identity, and then devices using secret unknown cipher (SUC) as identity. Finally, a comparison between PUF and SUC will conclude this section.

5.1. Identification Based on Analog Physical Unclonable Functions

Physically unclonable functions or physical unclonable functions (PUFs) were built based on the physical characteristics (random manufacturing variations) of silicon devices. PUF can be perceived as a DNA-like identity for the device. The random, unpredictable, and uncontrollable nature of PUFs makes them physically very hard to clone, even for the manufacturer. Several PUFs were proposed two decades ago. In the following, three well-known technologies of PUFs are presented as a sample [29]:

- Arbiter PUF

  The arbiter PUF, shown in Figure 11a, is composed of two parallel paths of consecutive multiplexers as symmetric paths and a flip-flop or a latch as arbiter circuit [30]. The arbiter circuit includes N input bits (0 and 1 every stage) and has one-bit output. The core idea of arbiter PUF is to extract the delay of tow paths. The path that reaches the arbiter circuit first decides the output of the arbiter circuit, i.e., either 0 or 1 [29]. Arbiter PUF is classified as a strong PUF as for multiple inputs it responds with multiple corresponding outputs [17].

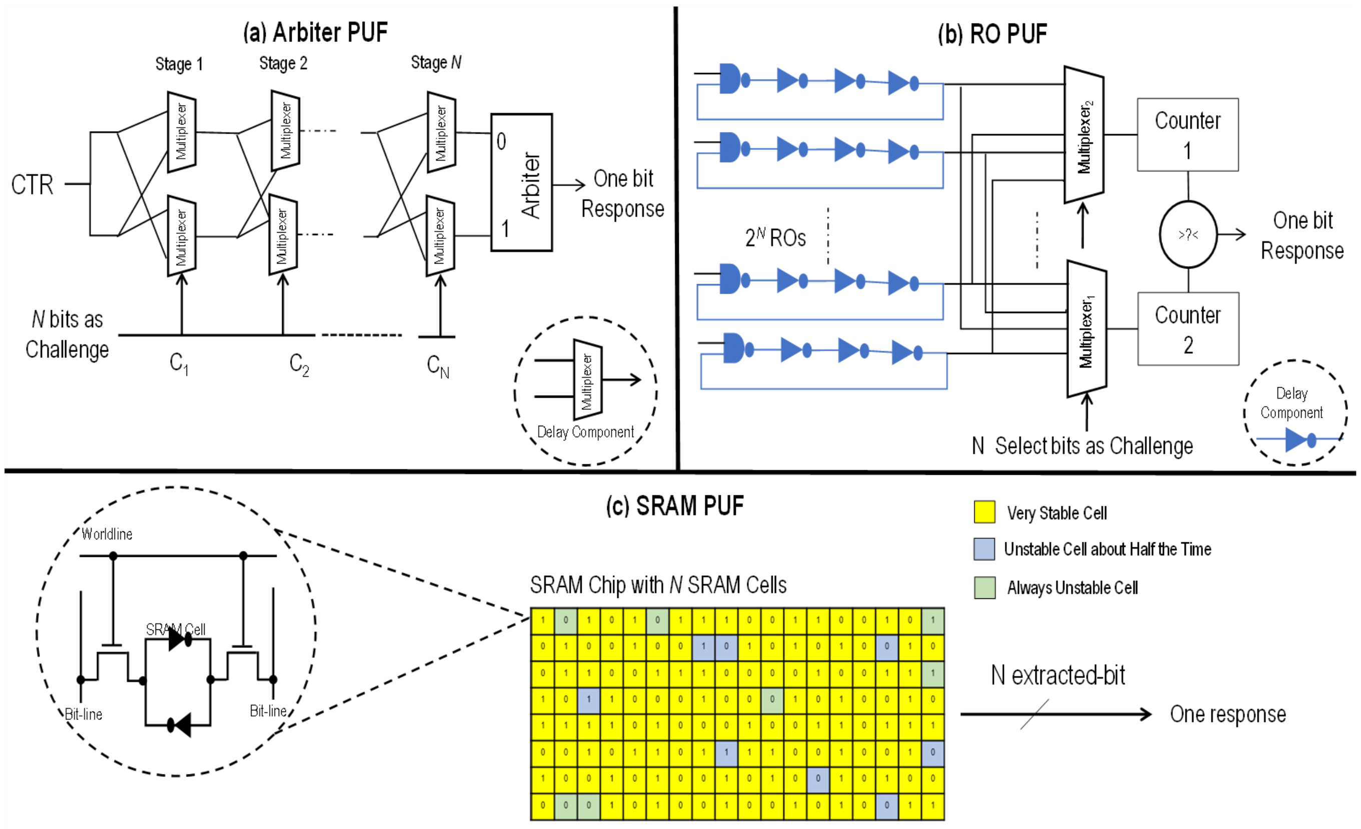


Figure 11. (**a**) Arbiter PUF; (**b**) Ring Oscillator PUF; (**c**) SRAM PUF; PUF proposals adapted from [29,31–33].

- Ring Oscillator PUF

  The ring oscillator PUF (RO PUF) is also defined as a delay-based PUF. Several identical ring oscillators and two counters compare RO frequencies and generate one-bit output as shown in Figure 11b [31]. RO PUF is classified as a weak PUF since it generates a limited number of challenge–response pairs [17].

- SRAM PUF

  A static random-access memory (SRAM) is composed of several cells that can store one bit, either 0 or 1. When the SRAM cell is powered on, the SRAM cell will have either 0 or 1 as an initial value. This random behavior of SRAM memory leads to the construction of SRAM PUF as illustrated in Figure 11c [32]. SRAM PUF is categorized as a memory-based PUF [29]. SRAM PUF is classified as a weak PUF since it generates a limited number of challenge–response pairs [17].

5.1.1. PUF-Based Identification Protocol: Challenges and Drawbacks

PUF can be perceived as unknown, highly non-linear physical mapping. For an input C, PUF responds with an output R. The input–output PUF-pairs or so-called challenge–response pairs (CRPs) are mainly utilized to identify and authenticate the device.

As silicon-based technologies, PUFs are very sensitive to temperature and voltage variants, aging, and circuit noise. Therefore, it is very difficult to ensure that whenever a PUF is fed by a challenge C, the PUF can respond with the same and stable R repetedly. This problem is well-known as a reproducibility problem in PUF-technologies [17]. To ensure the reproducibility of the PUF-response, an additional circuit (a so-called fuzzy extractor) is attached to a PUF to stabilize its PUF-response [34]. The fuzzy extractor can be perceived as an error correction code mechanism where it generates extra data (helper data) allowing correction of the PUF response R for any given challenge C. This solution requires extra hardware resources and decreases the PUF entropy, i.e., it reduces the number of usable CRPs.

Additionally, several side-channel attacks can threaten PUFs, such as invasive [35], semi-invasive [36], and non-invasive attacks [37]. Such attacks aim to create a physical clone of a PUF in the best-case scenario [38] and build a virtual PUF-model in the worst-case scenario.

5.1.2. Primitive PUF-Based Identification Protocol

As shown in Figure 12, a basic PUF-based identification protocol can be carried out after the enrollment phase. In the enrollment phase, PUF is fed by a set of challenges $\{C_i\}_{i=1}^{m}$, and the helper data $\{h_i\}_{i=1}^{m}$ are stored in a server database together with the corresponding responses $\{R_i\}_{i=1}^{m}$. In the identification phase, the server selects one (C_j, R_j, h_j) out of m stored CRPs and sends (C_j, h_j) to the device. The device responds by its PUF and fuzzy extractor with R'_j. The server compares the received R'_j with the stored R_j to verify the PUF.

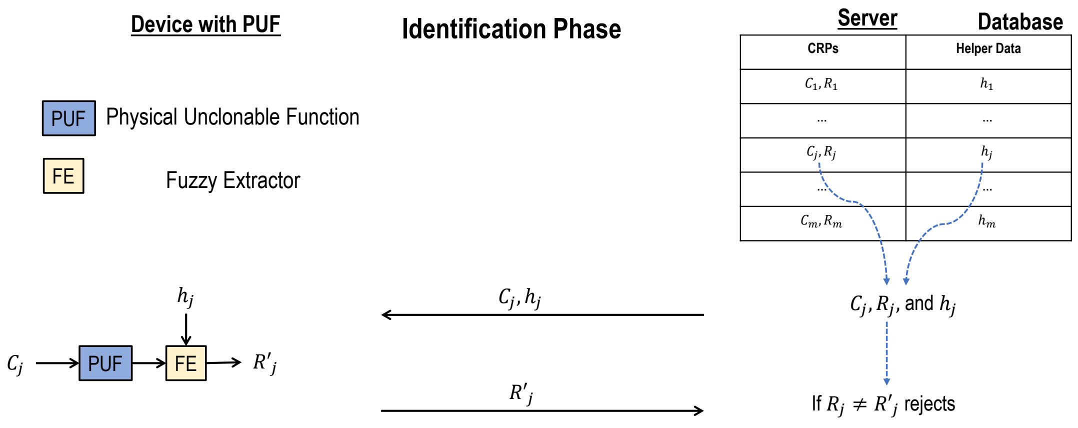


Figure 12. Basic PUF-based identification protocol.

5.1.3. Security Evaluation of PUF-Based Identification Protocol

Due to the limited number of CRPs in many weak PUF technologies, an adversary can feed a PUF with all possible challenges and store the corresponding responses in a PUF-codebook which can be perceived as a PUF-substitution. This PUF-codebook can carry out the mentioned basic PUF-based identification protocol similar to the original PUF. Therefore, this attack is equivalent to a/the PUF-cloning attack. On the other hand, if a PUF can generate an exponentially-large number of CRPs, the PUF is called a strong PUF [17]. However, if the generated CRPs are highly correlated, then a set of PUF-CRPs can be fed to a machine learning (ML) algorithm which may build a PUF-predictive model [39]. For a new challenge C_k, a/the PUF-predictive model responds with the same response R_k as the original PUF. Thus, an adversary (man in the middle attack) can successfully perform this attack scenario against the mentioned basic PUF-based identification protocol after collecting a large enough number of CRPs as a training set. Table 3 summarizes several successful molding attacks against different types of PUFs along with their prediction rates. For instance, a successful molding attack on the Arbiter PUF with 64-bit size of the challenge requires approximately 20,000 CRPs to construct a predictive PUF-model within 0.60 s by using logistic regression. As a result, most of the proposed PUFs can be molded by using a few CRPs within a short time. More details about the molding attacks on PUFs can be found in [39].

Table 3. Successful molding attacks against different types of PUFs (adapted from [39]).

PUF-Types	Challenge Size	Response Size	Algorithm/ ML Technique	Required CRPs	Prediction Rate	Time Complexity
Arbiter PUF	64-bit	One bit	Logistic Regression	18,050	99.9%	0.60 s
XOR Arbiter PUFs	64-bit as input of 4 Arbiter PUFs	One bit	Logistic Regression	12,000	99%	03:42 min
Feed Forward Arbiter PUF	64-bit	One bit	Evolution Strategies	50,000	97.72%	07:51 min
RO-PUF	256-Oscillators	One bit	Quick Sort	28,891	99.9%	Not Available

In order to provide a PUF with immunity against modeling attacks, additional algorithm on software level or a cryptographic function should be deployed [17]. Such countermeasures increase both the execution time and the hardware complexity.

5.1.4. PUF for IoT Device Identification and Authentication

Every IoT system aims to achieve a balance between hardware cost, performance, and security. In [40], PUFs were presented as a robust solution for securing IoT devices. The idea was to investigate how PUF-based protocols meet the requirements of secure communication with little resource overheads [40]. As a result of their investigations, a secure PUF-based IoT system should be robust and resilient against side channel attacks and man in the middle attacks. In the following, new criteria for PUF-based IoT systems are addressed and discussed. Such criteria give good indications for the performance and hardware cost of PUF-based IoT systems.

5.1.5. PUF-Based IoT Identification Protocols: Communication Complexity

As mentioned above, a/the basic PUF-based identification protocol cannot be deployed for IoT systems due to the security concerns mentioned above. Therefore, several PUF-based identification protocols have been designed to overcome PUF vulnerabilities by communicating more with the server and/or using a proposed extra cryptographic function [17]. The number of communications between the server and the PUF together with the additional cryptographic function affects and impacts the overall power consumption of the IoT system, not only for the IoT devices but also the required power consumption to perform the protocol. Furthermore, the communication channel size required to complete the protocol between the server and IoT device with PUF provides a precise indication of the bit volume of the exchanged data. Table 4 shows a variety of communication complexities of several PUF-based identification protocols proposed for IoT system, where n is the bit length of PUF-CR. To simplify the calculation, n is also used to indicate the size of any transmitted value via the protocol such as a nonce and digest.

Table 4. Communication complexity of a sample of PUF-based identification protocols.

Proposed PUF-Class	Target of Protocol	Additional Cryptographic Function	# of Server-Device Communications	Max. Required Channel Size
Two Strong PUFs [41]	Authentication	No	3	n-bit
Strong PUF [42]	Identification and Authentication	XoR	4	$2n$-bit
Strong PUF [43]	Identification and Authentication	Hash	5	$2n$-bit
Weak PUF [44]	Authentication	Hash	3	$3n$-bit
Strong PUF [45]	Authentication	Cipher	6	n-bit
Strong PUF [46]	Authentication	XoR	5	n-bit
Strong PUF [47]	Identification and Authentication	HashMAC	4	$3n$-bit
Strong PUF [48]	Authentication	Hash and XoR	3	$4n$-bit
Strong PUF [49]	Authentication	Hash and XoR	3	$3n$-bit
Strong PUF [50]	Identification and Authentication	HashMAC and XoR	4	n-bit

According to Table 4, a minimum number of communications between the server and the IoT device to perform the identification protocol is three, each with n-bit as a communication channel size. As a result, a PUF-based identification and authentication protocol requires multiple communications between the server and IoT device via a channel of at least size n bits.

5.1.6. PUF-Based IoT Identification Protocols: Hardware Complexity

Since a low area is desirable for IoT systems, the proposed PUFs should consume a small portion of hardware resources and have low implementation cost/complexity. Here, the hardware cost indicates the required area that the proposed PUF occupies. Apparently, a PUF requiring less complexity is more desirable for IoT systems. Table 5 shows several PUF-proposals with implementation costs in Field Programmable Gate Array (FPGA). In [51], compact FPGA-implementations of RO-PUF, Arbiter PUF, and RS Latch-PUF (RS-LPUF) were presented. The PUF-implementations were optimized to achieve very competitive area trade-offs. Note that the results presented in Table 5 do not show the required extra-overhead for the fuzzy extractor and/or the error correction code technique.

Table 5. A sample of implemented PUFs on different FPGAs.

Type of PUF	Targeted FPGA	Hardware Resources	
		# LUTs	# DFFs
PolyPUF [52]	-	213	450
Slender PUF [53]	-	652	1400
OB-PUF [54]	-	680	360
RSO-based PUF [55]	Xilinx Artix-7	405	1500
RPUF [56]	-	350	389
Enhanced Arbiter PUF [57]	Xilinx Artix-7	419	264
PUF-FSM [58]	-	960	1500
RO PUF [51], Optimized	Xilinx Spartan-6	82 Slices $\approx$ 328 *	82 Slices $\approx$ 656 *
RS-PUF [51], Optimized	Xilinx Spartan-6	54 Slices $\approx$ 216 *	54 Slices $\approx$ 432 *
Arbiter PUF [51], Optimized	Xilinx Spartan-6	234 Slices $\approx$ 936 *	234 Slices $\approx$ 1872 *

*: The approximation number of LUTs and DFFs are computed based on the technology used in the reference [51].

According to Table 5, a/the PUF circuit without any additional stabilizer circuit consumes an enormous amount of the hardware resources. For instance, an/the optimized Arbiter PUF itself requires almost 936 LUTs and 1872 DFFs [51].

5.1.7. PUF-Based Identification Discussion

PUFs display inconsistent behavior because of their sensibility to environmental and operational conditions and variations such as temperature, voltage, radiation, and aging factors. Additionally, many attacks on PUFs have been recently proposed. These target both weak PUFs and strong PUFs [59]; weak PUFs have fewer challenges, commonly only one challenge per PUF instance. Hence, it is assumed that access to the weak PUF response is restricted. In contrast, a strong PUF produces a high number of CRPs which are unpredictable. The process of cloning a PUF consists of two steps [60]:

Characterization: a process in which the attacker gains knowledge of the challenge/response behavior of a PUF.

Emulation: the process of recreating or modelling the unique response of a PUF, i.e., creating a PUF with identical challenge/response pairs.

Two decades since their introduction, we summarize those PUFs that still have drawbacks and shortcomings: their inconsistency and vulnerability to diverse cloning attacks. Table 6 shows the results for PUF-based identification mechanisms using selected comparison criteria.

Table 6. Selected comparison criteria of PUF-based security schemes.

Features	Result
Suitable for IoT constrained devices	Fulfilled
Perfect forward secrecy	Fulfilled
Resilience to physical attack	Not fulfilled [61]
Resilience to modeling attack	Not fulfilled [39]
Resilience to cloning attack	Not fulfilled
High performance/response time	Fulfilled [11]
Extra NVM usage	YES
Cryptographic primitive usage	YES
Key space	Unscalable
Key entropy	MEDIUM
Key update needed?	NO
Identification/authentication	Fulfilled
Mutual authentication	Fulfilled

5.2. Clone-Resistant IoT Identity Based on Digital Secret Unknown Cipher

Since PUFs have very concerning and severe drawbacks due to their sensitivity to operating conditions such as voltage, temperature, radiation, race (metastability), and other effects [62]. PUFs have therefore become unattractive for many IoT applications. Indeed, most of the problems result from aging effects that cause the material properties to change over time, making reidentification unstable. To counteract all these effects, expensive fuzzy extractors or helper data algorithms [63] have been proposed, resulting in highly complex systems that are also more susceptible to side-channel attacks, even on the fuzzy extractors [64]. In [5], Adi first proposed the SUC concept as a digital alternative to PUF; SUC is a self-created (digitally mutated) internal permanent digital structure that can encrypt and decrypt deploying functions that nobody knows. Since aging effects in digital structures are negligible, such ciphers remain consistent throughout the whole lifetime of digital products. To make the paper self-contained, the concept of creating secret unknown ciphers is reproduced with further details summarizing early publications [65].

Definition 4. *A physical secret unknown cipher (SUC) is a randomly and internally self-created unknown and unpredictable cipher module inside a chip, where both user and manufacturer have no influence on the created ciphering functions/modules. The resulting cipher is permanent, non-removable and tamper-proof. Even the device manufacturer should not be able to backtrace the creation process, nor predict or reveal the resulting cipher.*

SUC as a designed pseudorandom (PR) bijective function exhibits higher entropy than a conventional PUF, which is equivalent to a collision-prone hash function. The SUC's invertibility property as a cipher tends to be much more efficient in its generic identification and authentication protocols compared with PUFs [65].

Figure 13 describes the concept of embedding SUC in system-on-chip (SoC) units within self-reconfiguring, non-volatile SoC FPGAs. The SoC FPGA should be a core part in the hardware unit of each IoT device. As the IoT device provisioning process is conducted in a post-fabrication process by the trusted authority (TA), the creation of the SUCs happens without involving the device manufacturers; this is the most important security measure that can be carried out by the trusted authority or even by the application users.

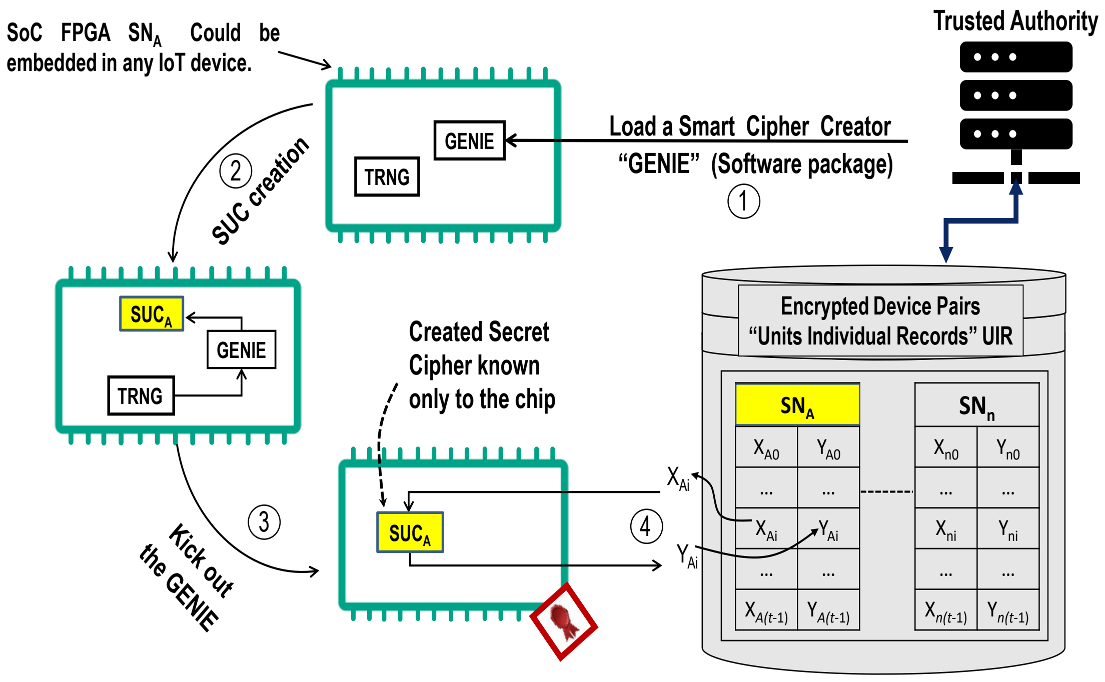

SoC FPGA SN$_A$ Could be embedded in any IoT device.
Trusted Authority
GENIE
TRNG
Load a Smart Cipher Creator "GENIE" (Software package)
①
② SUC creation
SUC$_A$
GENIE
TRNG
Created Secret Cipher known only to the chip
Encrypted Device Pairs "Units Individual Records" UIR
SN$_A$
SN$_n$
X$_{A0}$ Y$_{A0}$
X$_{Ai}$ Y$_{Ai}$
X$_{A(t-1)}$ Y$_{A(t-1)}$
X$_{n0}$ Y$_{n0}$
X$_{ni}$ Y$_{ni}$
X$_{n(t-1)}$ Y$_{n(t-1)}$
③ Kick out the GENIE
SUC$_A$
④
X$_{Ai}$
Y$_{Ai}$

Figure 13. Key idea for embedding SUCs in VLSI devices.

In the following, the steps needed to personalize each SoC FPGA by TA are described:

Step 1: TA uploads a cipher designer called "GENIE" into the SOC FPGA. The cipher designer contains an algorithm for creating internally unpredictable and unknown random secure ciphers. The cipher creation process runs just one time during the device lifecycle on the SoC FPGA.

Step 2: The cipher designer creates a permanent (non-volatile) and unpredictable random cipher by consulting and using an unknown random bit string from an internal and unpredictable true random number generator (TRNG).

Step 3: After the SUC creation is finished, the cipher designer is completely removed and the SoC FPGA gets its unique and unpredictable SUC.

Step 4: The TA challenges the SUC$_A$ using a set of random cleartext challenges $X_A = \{X_{A,0}, X_{A,1}, \ldots, X_{A,t-1}\}$ and receives the corresponding ciphertext responses $Y_A = \{Y_{A,0}, Y_{A,1}, \ldots, Y_{A,t-1}\}$. The TA securely stores the challenge–response (CR) pairs in a secure "unit individual records" (UIR) database for later usage; each pair is associated with the serial number of the IoT device (SN$_A$).

The IoT device with the embedded SUC is now ready to be commissioned in a network. The device can be securely identified and authenticated using the secret CR pairs. Note: the device manufacturer cannot influence the device personalization process and has no information about the cipher designer. This step can always be performed by a trusted network administrator of the IoT system. The devices can be irreversibly locked after Step 4. The cipher created in the device cannot be changed, reversed, or removed. Note that the trusted authority cannot create two SUCs with the same identity. In other words, the TA cannot clone devices without being traced.

5.2.1. SUC Identification Protocol

As shown in the personalization process, TA stores in the UIR all secure X/Y pairs identified by a/the serial number for each personalized IoT device. In Figure 14, a generic protocol to identify an IoT device D_A, with SUC_A proceeds as follows:

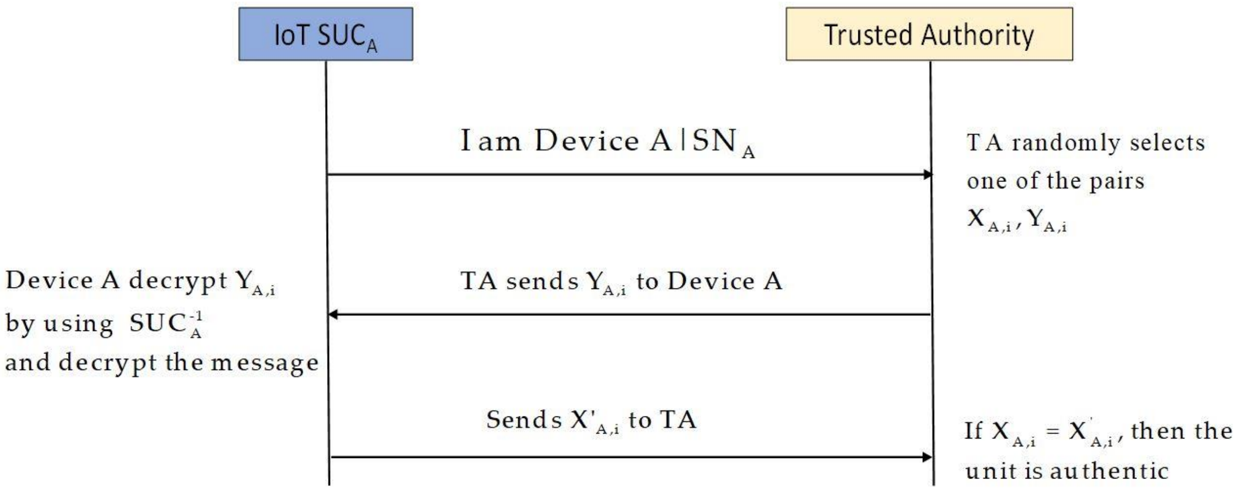

IoT SUC_A
Trusted Authority
I am Device A | SN_A
TA randomly selects one of the pairs $X_{A,i}$, $Y_{A,i}$
Device A decrypt $Y_{A,i}$ by using SUC_A^{-1} and decrypt the message
TA sends $Y_{A,i}$ to Device A
Sends $X'_{A,i}$ to TA
If $X_{A,i} = X'_{A,i}$, then the unit is authentic

Figure 14. Generic identification protocol for a SUC.

Step 1: IoT $Device_A$ with embedded SUC_A sends to TA an identification request with its device serial number.

Step 2: TA selects randomly one of the $X_{A,i}/Y_{A,i}$ pairs of the given serial number from the UIR and sends $Y_{A,i}$ to $Device_A$. If the serial number does not exist in the UIR, TA aborts the communication.

Step 3: $Device_A$ computes $Y_{A,i}$ by using SUC_A^{-1} and sends $X'_{A,i} = SUC_A^{-1}(Y_{A,i})$ to TA. If $X_{A,i} = X'_{A,i}$, then the device is deemed to be authentic and can be accepted. Otherwise, $Device_A$ is not authentic and should be rejected. The pair $X_{A,i}/Y_{A,i}$ is marked as consumed and for highest security performance should not be used later.

Compared to PUFs, SUC has the advantage of being capable of recovering X from Y by using the inverse function SUC^{-1}. This property allows low-complexity and very efficient management of the consumed X/Y-pairs. The property was also used in [66] to build a physical chain of trust for secured authentication in a medical device environment.

5.2.2. SUC Mutual Authentication Protocol

Figure 15 shows a possible protocol to let the IoT device SUC_A and another IoT device with SUC_B communicate securely with the help of TA, which acts as a one-time mediator to allow mutual authentication between the two devices. Both IoT devices are already identified by TA. For encryption, a standard cipher E such as AES is used, the protocol proceeds as follows:

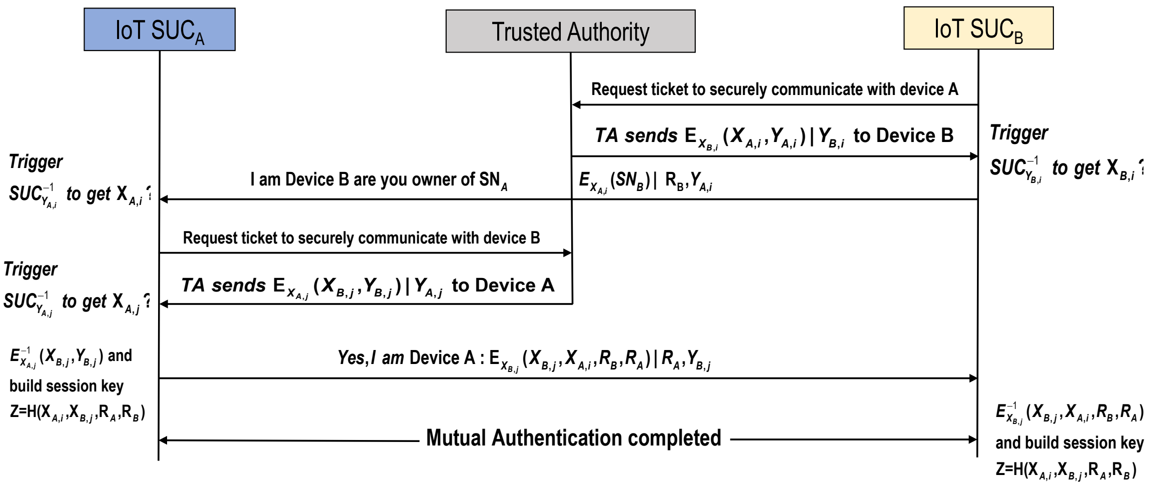

IoT SUC$_A$
Trusted Authority
IoT SUC$_B$
Request ticket to securely communicate with device A
TA sends $E_{X_{B,i}}(X_{A,i}, Y_{A,i}) | Y_{B,i}$ to Device B
Trigger $SUC^{-1}_{Y_{B,i}}$ to get $X_{B,i}$
I am Device B are you owner of SN$_A$
$E_{X_{A,i}}(SN_B) | R_B, Y_{A,i}$
Trigger $SUC^{-1}_{Y_{A,i}}$ to get $X_{A,i}$
Request ticket to securely communicate with device B
TA sends $E_{X_{A,j}}(X_{B,j}, Y_{B,j}) | Y_{A,j}$ to Device A
Trigger $SUC^{-1}_{Y_{A,j}}$ to get $X_{A,j}$
$E^{-1}_{X_{A,j}}(X_{B,j}, Y_{B,j})$ and build session key $Z=H(X_{A,i}, X_{B,j}, R_A, R_B)$
Yes, I am Device A : $E_{X_{B,j}}(X_{B,j}, X_{A,i}, R_B, R_A) | R_A, Y_{B,j}$
Mutual Authentication completed
$E^{-1}_{X_{B,j}}(X_{B,j}, X_{A,i}, R_B, R_A)$ and build session key $Z=H(X_{A,i}, X_{B,j}, R_A, R_B)$

Figure 15. Mutual authentication protocol for two SUCs.

1. $Device_B$ with embedded SUC_B sends TA a request for a ticket token asking to communicate with $Device_A$.
2. TA selects randomly one of the pairs of $Device_A$ $(X_{A,i}/Y_{A,i})$ and sends the pair encrypted by E keyed with $X_{B,i}$ and $Y_{B,i}$ to $Device_B$.
3. $Device_B$ computes $X_{B,i}$ using $SUC^{-1}_{Y_{B,i}}$, then decrypts the incoming message to obtain the access token for $Device_A$.
4. $Device_B$ sends $Device_A$ an authentication request, $Device_B$ encrypts SN_B using $X_{A,i}$ and sends a random R_A and $Y_{A,i}$ to $Device_A$.
5. $Device_A$ with embedded SUC_A sends TA a request for a ticket as access token to communicate with device $Device_B$.
6. TA selects randomly one of the $X_{B,j}/Y_{B,j}$ pairs of $Device_B$ and sends the pair $X_{B,j}/Y_{B,j}$ encrypted by E and keyed with $X_{A,j}$ and $Y_{A,j}$ to $Device_A$.
7. $Device_A$ computes $X_{A,j}$ using $SUC^{-1}_{Y_{A,j}}$, then decrypts the incoming message to obtain the access token for $Device_B$.
8. $Device_A$ sends $Device_B$ encrypted parameters $(X_{A,i}, X_{B,j}, R_A, R_B)$ using $X_{B,j}$ as key and random $R_A, Y_{B,j}$ to build the session key.
9. $Device_B$ computes $X_{B,j}$ using $SUC^{-1}_{Y_{B,j}}$, then decrypts the incoming message to obtain the session key parameters.
10. Both devices share the same secret parameters and can communicate securely later, using $Z = H(X_{A,i}, X_{B,j}, R_A, R_B)$ as a session key, where H is a public hash function.

5.2.3. Security Analysis of SUCs

Various attack scenarios are possible, such as replay and impersonation attacks. However, in an open network an attacker can intercept the message between Device A and Device B, but the attacker cannot decrypt the encrypted messages without having SUC; therefore, the SUC cloning attack is discussed below.

Cloning Complexity of Embedded SUC in IoT Devices

To impersonate an IoT device, an adversary could try to reverse-engineer or clone the targeted IoT device; an intruder aiming to impersonate an IoT device should clone its SUC. SUC cloning attacks can be classified into two categories: mathematical/analytical cloning in the form of cryptographic analysis attacks on the designed SUC cipher classes, and physical cloning in the form of invasive side-channel attacks to extract the SoC bitstream.

- Mathematical/analytical cloning: SUCs are designed as state-of-the-art ciphers to be resistant against known mathematical attacks. Such issues were studied in [67,68]. SUC is considered as practically unclonable by modeling if it is not feasible to store all the challenge–response space as the cipher codebook size (CCBS). For an 80-bits cipher $n = 80$ results in CCBS = 2^{80}. Resistance to attacks with complexity in the order of 2^{80} is considered the minimum adequate level for SUCs to meet today's security requirements. For post-quantum security, complexity in the order of at least 2^{160} is required.

- Physical cloning: An adversary with physical access to IoT devices pins may try to reverse-engineer the embedded SUCs through "bitstream attacks". In [69], the authors show that a clone attack is highly infeasible as applying adequate side-channel attacks requires knowledge of the SUC location, design structure, and related mappings, which are mostly unknown and hard to predict.

We conclude that, practically, such attacks on SUC structures are highly infeasible without very expensive physical invasive attacks on the IoT device. Furthermore, break-one break-all does not work for SUCs as no unit is created with the same randomizing process.

5.2.4. SUC Hardware Complexity

Several SUC designs and architectures were investigated and implemented for the SmartFusion®2SoC FPGA technology. SmartFusion®2SoC FPGA is the only non-volatile FPGA technology with flash-based distributed switching fabrics and programmable cells. SmartFusion®2SoC FPGA provides powerful arithmetic units, the so-called MACC, high-performance communication interfaces, and flash-based FPGA fabric incorporating an integrated ARM Cortex-M3 processor. SUC as a stream cipher was proposed in [68]; it was optimized by deploying the arithmetic units MACC. In [70], SUC as a Feistel-like cipher was proposed by replacing the XOR-operation in the Feistel network with a new involution operation based on multipliers MACC in modern FPGA devices. In [71], a large class of generalized Feistel network was presented. The proposed cipher is implemented based on mini-blocks of 4-bit mappings and bundle permutations, using only 4-input LUT and DFF. The KSG requires 37 LUTs and 223 DFFs in [69]; potentially, this can be considered as zero cost in many real-world use cases. Mars et al. show in [67] that the I-SUC version is a more efficient design (as an involutive cipher), consuming much fewer resources for both encryption and decryption than the NI-SUC version. Table 7 summarizes the hardware complexity of previous SUC proposals for SmartFusion®2 SoC FPGAs.

Table 7. Sample hardware complexities of selected SUC designs on SmartFusion®2 FPGA-Technology (Adapted from [66]).

SUC Proposals		Hardware Required Resources		
		LUTs	**DFFs**	**MACC**
SUC as a Stream Cipher	SUC based on T-Function [68]	81	80	10
	SUC based on combining NFSRs [69]	37	223	0
SUC as a Block Cipher	SUC as a Feistel-Like Cipher [70]	208	113	8
	SUC as Generalized Feistel Network [71]	138	150	0
	SUC as Non-Involutive Cipher [67]	212	72	0
	SUC as Involutive Cipher [67]	226	72	0

5.2.5. SUC-Based IoT Device Identity Discussion

SUC is a clone-resistant digital PUF; in other words, it can be used as a digital replacement for traditional analog PUFs. As digital modules, SUCs have no disadvantages in terms of inconsistency in aging and sensitivity to operating conditions such as temperature and supply voltage. SUCs can be used in various IoT applications that require secure clone-resistant physical identity. In contrary to PUFs, SUCs cannot be seen as unclonable because if it were possible for the device to be invasively attacked and the bitstream obtained, then full consistent cloning would be possible for a single unit. In this sense, SUC has practical security in that the attacker needs to physically invade the device to clone it. However, emerging 3-D VLSI technology is expected to make such attacks infeasible as the bitstream (as the required secret) is destroyed when physically attacked.

Based on the protocol shown in Figure 14, Table 8 shows the result for SUC-based identification mechanisms using the comparison criteria.

Table 8. Comparison criteria of SUC-based security schemes.

Features	**Result**
Suitable for IoT constrained devices	Fulfilled (SoC FPGA)
Perfect forward secrecy	Fulfilled
Resilience to physical attack	Partially Fulfilled
Resilience to modeling attack	Fulfilled
Resilience to cloning attack	Fulfilled
High performance/response time	Fulfilled
Extra NVM Usage	NO
Cryptographic primitive usage	NO
Key space	Scalable
Key Entropy	HIGH (Maximum)
Key update needed?	NO
Identification/authentication	Fulfilled
Mutual authentication	Fulfilled

5.3. PUF-Based Unclonability Versus Digital Clone-Resistant SUC Techniques

This section compares the advantages and disadvantages of analog-based PUFs as unclonable identity and digital-based SUC as clone-resistant physical IoT identities. Table 9 summarizes the comparison criteria. The comparison covers the following criteria:

- Function inconsistency
- Resilience to physical attacks
- Resilience to modeling attacks
- Cryptographic primitive usage
- Key space/entropy
- CR pair management

Table 9. PUF Versus SUC.

Features	**PUF**	**SUC**
Suitable for IoT constrained devices	Fulfilled	Fulfilled
Resilience to physical attack	Fulfilled	Partially Fulfilled
Resilience to modeling attack	Not Fulfilled	Fulfilled
Resilience to cloning attack	Not fulfilled	partially fulfilled
High performance/response time	Fulfilled	Fulfilled
Extra NVM Usage	YES	NO
Cryptographic primitive usage	YES	NO
Key space	Unscalable	Scalable
Key entropy	MEDIUM	HIGH
Key update needed?	NO	NO
Identification/authentication	Fulfilled	Fulfilled
Mutual authentication	Fulfilled	Fulfilled

(A) The Impact of the Reliability Problem on Entropy

PUFs exhibit unreliable behavior due to the analog nature of PUF-Modules. Any CMOS-based PUF is very sensitive to aging, temperature variation, etc. This means a failure in the PUF-response can occur within the whole lifetime of an IoT unit with high probability. For instance, the reliability of the RO PUF was investigated and analyzed in [72]. The experimental results show that the long RO PUFs (7-bits challenge size) are unreliable compared to the short arbiter PUFs with a challenge of size three-bits. The unreliable behavior of the PUFs indicates that a PUF responds with the same response to two different challenges. Therefore, there is always a compromise between PUF reliability and PUF entropy. As a result, a PUF with high entropy does not exist due to the reliability problem and due to the fact, that, in the best case, a PUF is equivalent to a collision-prone weak hash function. Several approaches have been published to study and solve this problem [73,74]. It should be noted that the current efficient solutions require either extra hardware resources (as shown in Section 5.1.6) or addition of some error correction algorithms at software level. Both solutions are unsuitable for an IoT environment. Consuming more hardware resources leads to high power consumption, whereas a software solution opens the door to more security threats and bugs. On the other hand, deploying pure digital hardware structures such as SUC is perfectly consistent in the whole lifetime of an IoT unit and the reliability of SUC is equivalent to the reliability of the digital IoT unit itself.

(B) The Impact of SUC Invertibility on Key Space A pure PUF circuit without any additional circuit/component can be perceived as a many-to-one collision-prone function such as a random weak hash/compression function. Therefore, the PUF -response space cannot avoid the inherent collision of its outputs. Such a collision reduces the PUF-key space from 2^n to $2^{n/2}$ on average in the best-case scenario, where n is the bit length of the PUF-response. On the other hand, SUC is a reversible designed one-way pseudorandom function (PRF) which exhibits the following advantages compared to a PUF mapping:

1. The whole input–output space of the cipher is usable as it is an on-to and a one-to-one function. As a result, it is a fully collision-free mapping. The size of SUC- input–output space is 2^n, where n is the SUC-input–output bit length.
2. The entropy of identifiable objects includes the whole cipher space due to the collision-free operation.

(C) CRP Management

As SUC deploys a reversible function, the clear-text space is fully usable in a structured manner without loss of security. This results in a tremendous advantage in managing the used pairs for identification on the side of a low-complexity device. To clarify this point, assume that SUC and PUF have n-bit as output length, so the average number of usable input–output pairs is 2^n in the case of SUC and $2^{n/2}$ in the best-case scenario for the PUF. For n = 128, an example for managing 1024 C-R pairs to check double-usage in a PUF requires storage of 128 $\times$ 1024 = 128 Kbits of memory with an average search mechanism of 1024/2 = 512 cycles [65], whereas memory storage of only 1024 bits with a single search cycle is needed to check C-R double usage in the case of SUC [75].

6. Conclusions

In this comparison study, we defined the role of identity in the IoT device lifecycle and introduced the identity-related security threats and challenges in IoT. Furthermore, we updated the IoT security requirements of the device, network, and application layer, emphasizing the secure device identity; therefore, the required basic definitions and a framework to analyze and investigate the IoT device identities were presented. Finally, we classified the current proposals of secure device identity into two categories: first, inherently-cloneable identification mechanisms and second, inherently unclonable identification mechanisms. The investigation of the first category indicates that all contemporary cryptographic identification technologies are basically clonable as somebody knows the embedded identity, because IoT units need to be unclonable even by the manufacturer, user, and operator in such cases. The investigation of the second category shows that the only perfect identity is one that has unpredictable behavior and can produce infinite unknown C-R pairs. The second category includes PUF and SUC. On the one hand, we presented a solid background on PUF. Each PUF type is usable in a specific hardware environment; however, PUFs exhibit drawbacks and shortcomings generally in terms of their inconsistency and vulnerability to diverse cloning attacks. Therefore, PUFs are relatively very expensive and highly complex for IoT mass products. On the other hand, we introduced the SUC technology as a possible practical replacement for PUFs at low cost and with excellent consistency over an IoT lifetime. However, SUCs are only clone-resistant as digital modules and require self-reconfiguring non-volatile VLSI technology, which is not yet commercially available as required. Figure 16 summarizes the presented identification solutions in this work. Furthermore, a systematic comparison was introduced regarding the following criteria: first, resilience to several classes of attacks; second, offering high performance; third, requiring low hardware complexity, etc. The comparison shows that SUCs and PUFs have a high level of security; however, SUC has lower hardware costs and is highly consistent/reliable but clone-resistant only, while PUF has high hardware costs and is less reliable in terms of consistency but is unclonable.

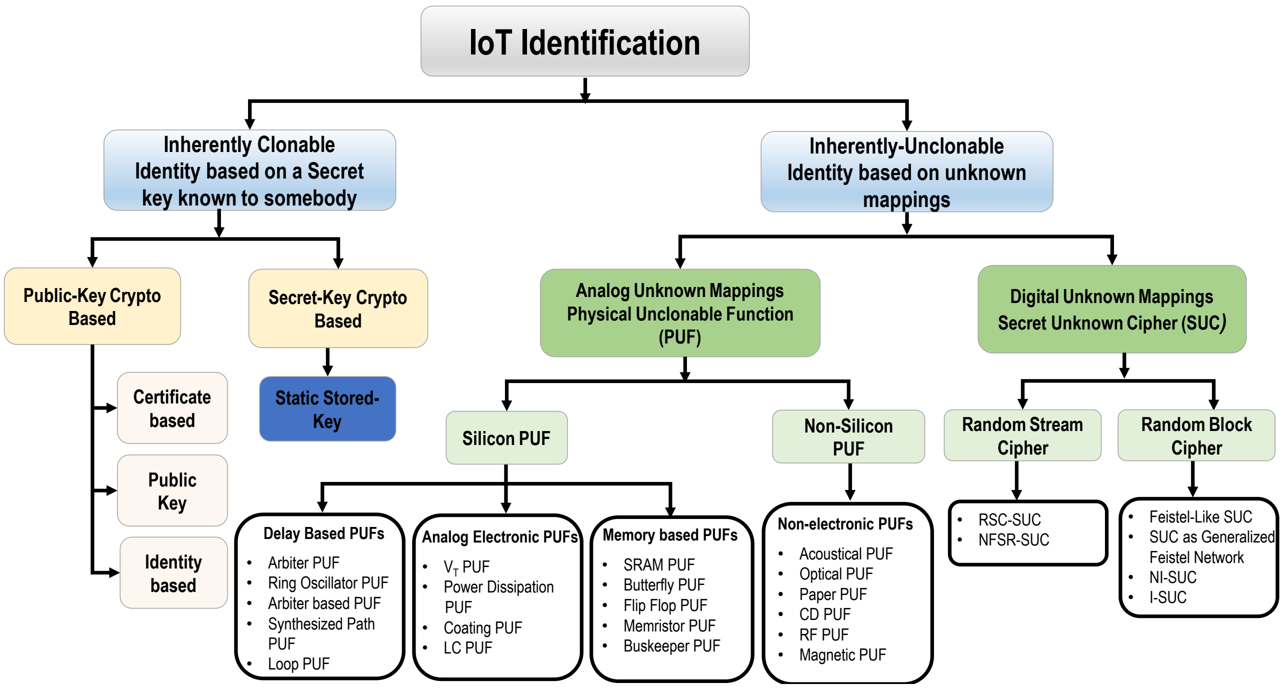

IoT Identification
Inherently Clonable Identity based on a Secret key known to somebody
Inherently-Unclonable Identity based on unknown mappings
Public-Key Crypto Based
Secret-Key Crypto Based
Static Stored-Key
Certificate based
Public Key
Identity based
Analog Unknown Mappings Physical Unclonable Function (PUF)
Digital Unknown Mappings Secret Unknown Cipher (SUC)
Silicon PUF
Non-Silicon PUF
Random Stream Cipher
Random Block Cipher
Delay Based PUFs: Arbiter PUF, Ring Oscillator PUF, Arbiter based PUF, Synthesized Path PUF, Loop PUF
Analog Electronic PUFs: V_T PUF, Power Dissipation PUF, Coating PUF, LC PUF
Memory based PUFs: SRAM PUF, Butterfly PUF, Flip Flop PUF, Memristor PUF, Buskeeper PUF
Non-electronic PUFs: Acoustical PUF, Optical PUF, Paper PUF, CD PUF, RF PUF, Magnetic PUF
RSC-SUC, NFSR-SUC
Feistel-Like SUC, SUC as Generalized Feistel Network, NI-SUC, I-SUC

Figure 16. IoT Identification Mechanisms.

Finally, the authors expect the emerging non-volatile VLSI technology to provide the necessary SUC-usable devices as soon as commercial products. Even the highly secure 3-D technology is expected to fulfill SUC requirements. However, SUC research is still in its infancy and looks very promising for future practical IoT applications. The authors hope to draw attention to and encourage more discussions and research on digitally mutated clone-resistant physical identity technologies.

Author Contributions: Conceptualization, E.H. and S.M.; methodology, E.H., S.M. and W.A.; validation, E.H., S.M. and W.A.; formal analysis, E.H. and S.M.; investigation, E.H. and S.M.; resources, W.A.; writing—original draft preparation, E.H. and S.M.; writing—review and editing, E.H., S.M., W.A. and M.B.; visualization, E.H. and S.M.; supervision, W.A. and M.B.; project administration, W.A.; funding acquisition, W.A. and M.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by Volkswagen AG, Microsemi, a Microchip Company, San Jose USA, and the German Federal Foreign Office funding by DAAD combined scholarship and support program (STIBET), as well as by the German Federal Ministry of Education and Research (BMBF) under the project KI-IoT (FKZ 16ME0091K).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. David, G. Strategic Principles for Securing the Internet of Things (Iot). Introduction and Overview. 2016. Available online: https://www.dhs.gov/sites/default/files/publications/Strategic_Principles_for_Securing_the_Internet_of_Things-2016-1115-FINAL....pdf (accessed on 23 September 2021).
2. IoT Growth Demands Rethink of Long-Term Storage Strategies, Says IDC. Available online: https://www.idc.com/getdoc.jsp?containerId=prAP46737220 (accessed on 29 January 2021).
3. Sadique, K.M.; Rahmani, R.; Johannesson, P. Towards security on internet of things: Applications and challenges in technology. *Procedia Comput. Sci.* **2018**, *141*, 199–206. [CrossRef]
4. Gassend, B.; Clarke, D.; van Dijk, M.; Devadas, S. Silicon physical random functions. In *Proceedings of the 9th ACM Conference on Computer and Communications Security—CCS '02*; Association for Computing Machinery (ACM): New York, NY, USA, 2002; p. 148.
5. Adi, W. Autonomous physical secret functions and clone-resistant identification. In Proceedings of the 2009 International Symposium on Bio-inspired, Learning, and Intelligent Systems for Security, Edinburgh, UK, 20–21 August 2009; pp. 83–88.
6. RFC 8576: Internet of Things (IoT) Security: State of the Art and Challenges. Available online: https://www.rfc-editor.org/rfc/rfc8576.html (accessed on 23 September 2021).
7. Lampson, B.W. A note on the confinement problem. *Commun. ACM* **1973**, *16*, 613–615. [CrossRef]
8. Zander, S.; Armitage, G.; Branch, P. A survey of covert channels and countermeasures in computer network protocols. *IEEE Commun. Surv. Tutor.* **2007**, *9*, 44–57. [CrossRef]
9. Skorobogatov, S.P. Number 630 Semi-Invasive Attacks-A New Approach to Hardware Security Analysis. 2005. Available online: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.228.2204&rep=rep1&type=pdf (accessed on 29 January 2021).
10. Jacob, N.; Wittmann, J.; Heyszl, J.; Hesselbarth, R.; Wilde, F.; Pehl, M.; Sigl, G.; Fischer, K. Securing FPGA SoC configurations independent of their manufacturers. In Proceedings of the 30th International System on Chip Conference, Munich, Germany, 5–8 September 2017; pp. 114–119.
11. Muller, K.U.; Ulrich, R.; Stanitzki, A.; Kokozinski, R. Enabling Secure Boot Functionality by Using Physical Unclonable Functions. In Proceedings of the PRIME 2018—14th Conference on Ph.D. Research in Microelectronics and Electronics, Prague, Czech Republic, 2–5 July 2018; pp. 81–84.
12. Owen, D.; Heeger, D.; Chan, C.; Che, W.; Saqib, F.; Areno, M.; Plusquellic, J. An autonomous, self-authenticating, and self-contained secure boot process for field-programmable gate arrays. *Cryptography* **2018**, *2*, 15. [CrossRef]
13. Haj-Yahya, J.; Wong, M.M.; Pudi, V.; Bhasin, S.; Chattopadhyay, A. Lightweight Secure-Boot Architecture for RISC-V System-on-Chip. In Proceedings of the 20th International Symposium on Quality Electronic Design (ISQED), Santa Clara, CA, USA, 6–7 March 2019; pp. 216–223.
14. Zarrouk, R.; Mulhem, S.; Adi, W.; Berekovic, M. Clone-Resistant Secured Booting Based on Unknown Hashing Created in Self-Reconfigurable Platform. In *International Symposium on Applied Reconfigurable Computing*; Springer: Cham, Swizerland, 2021.
15. Giuliano, R.; Mazzenga, F.; Neri, A.; Vegni, A.M. Security access protocols in IoT capillary networks. *IEEE Internet Things J.* **2017**, *4*, 645–657. [CrossRef]
16. Maes, R. Physically Unclonable Functions: Properties. In *Physically Unclonable Functions*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 49–80.
17. Delvaux, J.; Peeters, R.; Gu, D.; Verbauwhede, I. A Survey on Lightweight Entity Authentication with Strong PUFs. *ACM Comput. Surv.* **2015**, *48*, 1–42. [CrossRef]
18. Gordon, H.; Edmonds, J.; Ghandali, S.; Yan, W.; Karimian, N.; Tehranipoor, F. Flash-Based Security Primitives: Evolution, Challenges and Future Directions. *Cryptography* **2021**, *5*, 7. [CrossRef]
19. Lara-Nino, C.A.; Diaz-Perez, A.; Morales-Sandoval, M. Elliptic Curve Lightweight Cryptography: A Survey. *IEEE Access* **2018**, *6*, 72514–72550. [CrossRef]
20. Shamir, A. Identity-Based Cryptosystems and Signature Schemes. In *Workshop on the Theory and Application of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 1985; Volume 196, pp. 47–53.
21. Boneh, D.; Franklin, M. Identity-based encryption from the weil pairing. In *Annual International Cryptology Conference*; Springer: Berlin/Heidelberg, Germany, 2001; Volume 2139, pp. 213–229.
22. IEEE 1363.3–2013–IEEE Standard for Identity–Based Cryptographic Techniques Using Pairings. Available online: https://standards.ieee.org/standard/1363_3-2013.html (accessed on 23 September 2021).
23. Saeed, M.E.S.; Liu, Q.Y.; Tian, G.Y.; Gao, B.; Li, F. AKAIoTs: Authenticated key agreement for Internet of Things. *Wirel. Netw.* **2019**, *25*, 3081–3101. [CrossRef]
24. Gilmore, J.; Weiler, S.; Kivinen, T. *Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)*; Wouters, P., Tschofenig, H., Eds.; Internet Engineering Task Force (IETF): Fremont, CA, USA, 2014.
25. Salman, O.; Abdallah, S.; Elhajj, I.H.; Chehab, A.; Kayssi, A. Identity-based authentication scheme for the Internet of Things. In Proceedings of the IEEE Symposium on Computers and Communications, Messina, Italy, 27–30 June 2016; pp. 1109–1111.
26. Vanstone, S. Explaining Implicit Certificates.
27. Campagna, M. Standards for Efficient Cryptography. 2013. Available online: https://www.secg.org/sec4-1.0.pdf (accessed on 29 January 2021).

28. Sciancalepore, S.; Capossele, A.; Piro, G.; Boggia, G.; Bianchi, G. Key Management Protocol with Implicit Certificates for IoT systems. In Proceedings of the 2015 Workshop on IoT challenges in Mobile and Industrial Systems, Florence, Italy, 18 May 2015. [CrossRef]
29. Maes, R.; Verbauwhede, I. Physically Unclonable Functions: A Study on the State of the Art and Future Research Directions. In *Towards Hardware-Intrinsic Security*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 3–37.
30. Lee, J.W.; Lim, D.; Gassend, B.; Suh, G.E.; Van Dijk, M.; Devadas, S. A technique to build a secret key in integrated circuits for identification and authentication applications. In Proceedings of the IEEE Symposium on VLSI Circuits, Digest of Technical Papers, Honolulu, HI, USA, 17–19 June 2004; pp. 176–179.
31. Suh, G.E. Srinivas Devadas Physical Unclonable Functions for Device Authentication and Secret Key Generation. In Proceedings of the 44th ACM/IEEE Design Automation Conference, San Diego, CA, USA, 25 June 2007; pp. 9–14.
32. Guajardo, J.; Kumar, S.S.; Schrijen, G.J.; Tuyls, P. FPGA intrinsic PUFs and their use for IP protection. In *International Workshop on Cryptographic Hardware and Embedded Systems*; Springer: Berlin/Heidelberg, Germany, 2007; Volume 4727, pp. 63–80.
33. Securing Billions of IoT Devices with Reliable HW-based Keys that are Never Stored. The Reliability of SRAM PUF. Available online: https://www.intrinsic-id.com/wp-content/uploads/2017/08/White-Paper-The-reliability-of-SRAM-PUF.pdf (accessed on 23 September 2021).
34. Dodis, Y.; Reyzin, L.; Smith, A. Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data. In *International Conference on the Theory and Applications of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 2004; pp. 523–540.
35. Nedospasov, D.; Seifert, J.P.; Helfmeier, C.; Boit, C. Invasive PUF analysis. In Proceedings of the 10th Workshop on Fault Diagnosis and Tolerance in Cryptography, Los Alamitos, CA, USA, 20 August 2013; pp. 30–38.
36. Tajik, S.; Dietz, E.; Frohmann, S.; Seifert, J.P.; Nedospasov, D.; Helfmeier, C.; Boit, C.; Dittrich, H. Physical characterization of arbiter pufs. In *International Workshop on Cryptographic Hardware and Embedded Systems*; Springer: Berlin/Heidelberg, Germany, 2014; Volume 8731, pp. 493–509. [CrossRef]
37. Mahmoud, A.; Rührmair, U.; Majzoobi, M.; Koushanfar, F. Combined Modeling and Side Channel Attacks on Strong PUFs. *IACR Cryptol. ePrint Arch.* **2013**, 632.
38. Ruhrmair, U.; Holcomb, D.E. PUFs at a glance. In Proceedings of the Automation & Test in Europe Conference & Exhibition (DATE), Dresden, Germany, 24–28 March 2014; pp. 1–6.
39. Rührmair, U.; Sehnke, F.; Sölter, J.; Dror, G.; Devadas, S.; Schmidhuber, J. Modeling attacks on physical unclonable functions. In Proceedings of the 17th ACM Conference on Computer and Communications Security, Chicago, IL, USA, 4–8 October 2010; pp. 237–249.
40. Babaei, A.; Schiele, G. Physical Unclonable Functions in the Internet of Things: State of the Art and Open Challenges. *Sensors* **2019**, *19*, 3208. [CrossRef]
41. Hammouri, G.; Öztürk, E.; Sunar, B. A tamper-proof and lightweight authentication scheme. *Pervasive Mob. Comput.* **2008**, *4*, 807–818. [CrossRef]
42. Kulseng, L.; Yu, Z.; Wei, Y.; Guan, Y. Lightweight mutual authentication and ownership transfer for RFID systems. In Proceedings of the 2010 Proceedings IEEE INFOCOM, San Diego, CA, USA, 14–19 March 2010.
43. Van Herrewege, A.; Katzenbeisser, S.; Maes, R.; Peeters, R.; Sadeghi, A.R.; Verbauwhede, I.; Wachsmann, C. Reverse fuzzy extractors: Enabling lightweight mutual authentication for PUF-enabled RFIDs. In *International Conference on Financial Cryptography and Data Security*; Springer: Berlin/Heidelberg, Germany, 2012; Volume 7397, pp. 374–389.
44. Maes, R. *Physically Unclonable Functions: Constructions, Properties and Applications*; Springer: Berlin/Heidelberg, Germany, 2012.
45. Lee, Y.S.; Kim, T.Y.; Lee, H.J. Mutual authentication protocol for enhanced RFID security and anti-counterfeiting. In Proceedings of the 26th IEEE International Conference on Advanced Information Networking and Applications Workshops, Fukuoka, Japan, 26–29 March 2012; pp. 558–563.
46. Xu, Y.; He, Z. Design of a security protocol for low–cost RFID. In Proceedings of the 2012 International Conference on Wireless Communications, Networking and Mobile Computing, Shanghai, China, 21–23 September 2012.
47. Jung, S.W.; Jung, S. HRP: A HMAC-based RFID mutual authentication protocol using PUF. In Proceedings of the International Conference on Information Networking, Bangkok, Thailand, 28–30 January 2013; pp. 578–582.
48. Mahalat, M.H.; Saha, S.; Mondal, A.; Sen, B. A PUF based Light Weight Protocol for Secure WiFi Authentication of IoT devices. In Proceedings of the 2018 8th International Symposium on Embedded Computing and System Design, Cochin, India, 13–15 December 2018; pp. 183–187.
49. Melki, R.; Noura, H.N.; Chehab, A. Lightweight multi-factor mutual authentication protocol for IoT devices. *Int. J. Inf. Secur.* **2020**, *19*, 679–694. [CrossRef]
50. Yilmaz, Y.; Gunn, S.R.; Halak, B. Lightweight PUF-based authentication protocol for IoT devices. In Proceedings of the 2018 IEEE 3rd International Verification and Security Workshop, Costa Brava, Spain, 2–4 July 2018; pp. 38–43.
51. Anandakumar, N.N.; Hashmi, M.S.; Sanadhya, S.K. Compact Implementations of FPGA-based PUFs with Enhanced Performance. In Proceedings of the 2017 30th International Conference on VLSI Design and 2017 16th International Conference on Embedded Systems, Hyderabad, India, 7–11 January 2017; pp. 161–166.
52. Konigsmark, S.T.C.; Chen, D.; Wong, M.D.F. PolyPUF: Physically Secure Self-Divergence. *IEEE Trans. Comput. Des. Integr. Circuits Syst.* **2016**, *35*, 1053–1066. [CrossRef]

53. Rostami, M.; Majzoobi, M.; Koushanfar, F.; Wallach, D.S.; Devadas, S. Robust and reverse-engineering resilient PUF authentication and key-exchange by substring matching. *IEEE Trans. Emerg. Top. Comput.* **2014**, *2*, 37–49. [CrossRef]
54. Gao, Y.; Li, G.; Ma, H.; Al-Sarawi, S.F.; Kavehei, O.; Abbott, D.; Ranasinghe, D.C. Obfuscated challenge-response: A secure lightweight authentication mechanism for PUF-based pervasive devices. In Proceedings of the 2016 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops), Sydney, Australia, 14–18 March 2016.
55. Zhang, J.; Shen, C. Set-Based Obfuscation for Strong PUFs Against Machine Learning Attacks. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2020**. [CrossRef]
56. Ye, J.; Hu, Y.; Li, X. RPUF: Physical unclonable function with randomized challenge to resist modeling attack. In Proceedings of the 2016 IEEE Asian Hardware Oriented Security and Trust, Yilan, Taiwan, 19–20 December 2016.
57. Zalivaka, S.S.; Ivaniuk, A.A.; Chang, C.H. Reliable and modeling attack resistant authentication of arbiter PUF in FPGA implementation with trinary quadruple response. *IEEE Trans. Inf. Forensics Secur.* **2019**, *14*, 1109–1123. [CrossRef]
58. Gao, Y.; Ma, H.; Al-Sarawi, S.F.; Abbott, D.; Ranasinghe, D.C. PUF-FSM: A Controlled Strong PUF. *IEEE Trans. Comput. Des. Integr. Circuits Syst.* **2018**, *37*, 1104–1108. [CrossRef]
59. Rührmair, U.; Devadas, S.; Koushanfar, F. Security based on physical unclonability and disorder. In *Introduction to Hardware Security and Trust*; Springer: New York, NY, USA, 2012; pp. 65–102. ISBN 9781441980809. [CrossRef]
60. Helfmeier, C.; Boit, C.; Nedospasov, D.; Seifert, J.P. Cloning physically unclonable functions. In Proceedings of the 2013 IEEE International Symposium on Hardware-Oriented Security and Trust, Austin, TX, USA, 2–3 June 2013; pp. 1–6.
61. Helfmeier, C.; Boit, C.; Nedospasov, D.; Tajik, S.; Seifert, J.-P. Physical vulnerabilities of Physically Unclonable Functions. In Proceedings of the 2014 Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden, Germany, 24–28 March 2014; pp. 1–4.
62. Katzenbeisser, S.; Kocabaş, Ü.; Rožić, V.; Sadeghi, A.R.; Verbauwhede, I.; Wachsmann, C. PUFs: Myth, fact or busted? A security evaluation of Physically Unclonable Functions (PUFs) cast in silicon. In *Cryptographic Hardware and Embedded Systems—CHES 2012*; Springer: Berlin/Heidelberg, Germany, 2012; Volume 7428, pp. 283–301.
63. Bösch, C.; Guajardo, J.; Sadeghi, A.R.; Shokrollahi, J.; Tuyls, P. Efficient helper data key extractor on FPGAs. In *International Workshop on Cryptographic Hardware and Embedded Systems*; Springer: Berlin/Heidelberg, Germany, 2008; Volume 5154, pp. 181–197.
64. Merli, D.; Schuster, D.; Stumpf, F.; Sigl, G. Side-channel analysis of PUFs and fuzzy extractors. In *International Conference on Trust and Trustworthy Computing*; Springer: Berlin/Heidelberg, Germany, 2011; Volume 6740, pp. 33–47.
65. Adi, W.; Mars, A.; Mulhem, S. Generic identification protocols by deploying Secret Unknown Ciphers (SUCs). In Proceedings of the 2017 IEEE International Conference on Consumer Electronics—Taiwan (ICCE-TW), Taipei, Taiwan, 12–14 June 2017; pp. 255–256.
66. Hamadaqa, E.; Adi, W. Clone-resistant authentication for medical operating environment. In Proceedings of the World Conference on Smart Trends in Systems, Security and Sustainability (WS4 2020), London, UK, 27–28 July 2020; pp. 757–762.
67. Mars, A.; Adi, W. Digitally Mutating NV-FPGAs into Physically Clone-Resistant Units. *arXiv* **2019**, arXiv:1908.03898.
68. Mars, A.; Adi, W.; Mulhem, S.; Hamadaqa, E. Random stream cipher as a PUF-like identity in FPGA environment. In Proceedings of the 2017 Seventh International Conference on Emerging Security Technologies (EST), Canterbury, UK, 6–8 September 2017; pp. 209–214.
69. Mars, A.; Adi, W. New Family of Stream Ciphers as Physically Clone-Resistant VLSI-Structures. *Cryptography* **2019**, *3*, 11. [CrossRef]
70. Mulhem, S.; Mohammad, M.; Adi, W. A New Low-Complexity Cipher Class for Clone–Resistant Identities. In Proceedings of the 2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, Croatia, 20–24 May 2019; pp. 971–976.
71. Mulhem, S.; Ayache, M.; Adi, W. Mini-Block-Based Cipher Class for Physical Clone—Resistant Devices. In Proceedings of the EST—Eighth IEEE International Conference on Emerging Security Technologies, Colchester, UK, 22–24 July 2019.
72. Mustapa, M.; Security, M.N.-S. Undefined Temperature, Voltage, and Aging Effects in Ring Oscillator Physical Unclonable Function. 2015. Available online: Ieeexplore.ieee.org (accessed on 23 September 2021).
73. Kaur, M.; Rashidzadeh, R.; Muscedere, R. Reliability of physical unclonable function under temperature and supply voltage variations. In Proceedings of the Midwest Symposium on Circuits and Systems, Windsor, ON, Canada, 5–8 August 2018; Volume 2018, pp. 1008–1011.
74. Deng, D.; Hou, S.; Wang, Z.; Guo, Y. Configurable Ring Oscillator PUF Using Hybrid Logic Gates. *IEEE Access* **2020**, *8*, 161427–161437. [CrossRef]
75. Mulhem, S.; Zarrouk, R.; Adi, W. Security and Complexity Bounds of SUC-Based Physical Identity. In Proceedings of the 2018 NASA/ESA Conference on Adaptive Hardware and Systems (AHS), Edinburgh, UK, 6–9 August 2018; pp. 317–322.