scientific reports

OPEN

Security primitives for memoryless IoT devices based on Physical Unclonable Functions and True Random Number Generators

Krzysztof Gołofit

The article describes various security primitives for significantly resource-constrained devices, such as sensors or sensor networks, IoT devices, wearables, etc. — i.e., devices without programmable memory. It is dedicated to parts which cannot handle complex algorithms of modern secure cryptography, cannot be equipped with programmable memories, or their circuits or data in permanent memories can be easily reverse-engineered. Instead, all security techniques (e.g., identification, authentication, and encryption) are based on modern hardware cryptography, mainly: physical unclonable functions (PUFs) and true random number generators (TRNGs). The paper addresses numerous issues from untraceable identification to mutual authentication to one-time pad encryption. The communication security is considered to be a trade-off between the device's resources (processing ability, energy consumption, implementation size, response time), preparation complicity (initialization time, size of a server data storage) and the security capabilities and protection levels. Primitives can be included into the communication protocol based on particular needs and available hardware resources.

The inspiration for this article comes from advances in flexible electronics which have recently attracted significant attention as a part of IoT solutions [1,2]. Nowadays, several emerging semiconducting technologies based on various materials, such as amorphous or polycrystalline silicon, oxides, carbon nanotubes, or organics are used [3]. Advantages of this type of electronic circuits lie, especially, in their physical form: they are thin and light as well as they can stretch and bend while operating. However, with the benefits comes a series of limitations, which include: huge and slow one-type transistors, circuits vulnerable to reverse-engineering, lack of multi-time programmable (MTP) memories, and visible one-time programmable (OTP) memories, or read only memories (ROMs).

Naturally, the security primitives presented in this paper are not limited to flexible electronics. Instead, they are applicable to any devices with constrained resources in computation power or energy, which cannot support state-of-the-art data protection methods such as certified encryption or authentication algorithms [4]. Primitives are dedicated to devices in which an MTP memory is either difficult to obtain or it is vulnerable to side-channel attacks (SCAs [5,6]). There is another reason for using memoryless devices — their low cost of implementation in integrated circuit (IC). Usually, MTP memories require additional dedicated layers in technological IC processes, which raise the cost of the whole chips production process. Memories usually use higher voltage, which require charge-pumps and additional electronics, and these are costly in IC layout area. For example, in modern RFID chips (with the chip size around one-tenth of a millimeter) the silicon area of the memory uses more than half of the chip size. Inexpensive data collection systems (based on cheap sensors, cheap nodes and other cheap IoT network devices), where a low device price is one of the main factors, would be cost-effective to produce if they did not require any data storage while powered down.

The general idea of using both TRNG and PUF for securing IoT devices is not new [7,8]. Nevertheless, most of the primitives presented here are novel — they have recently been patented domestically, but they have not been described in either scientific or academic literature to date. It is worth mentioning, that the international PCT procedure WO/2020/240527 [9] was not extended to any country or jurisdiction, therefore the ideas presented here are available free of charge worldwide.

Institute of Electronic Systems, Department of Electronics and Information Technology, Warsaw University of Technology, Nowowiejska 15/19, 00-665 Warsaw, Poland. email: krzysztof.golofit@pw.edu.pl

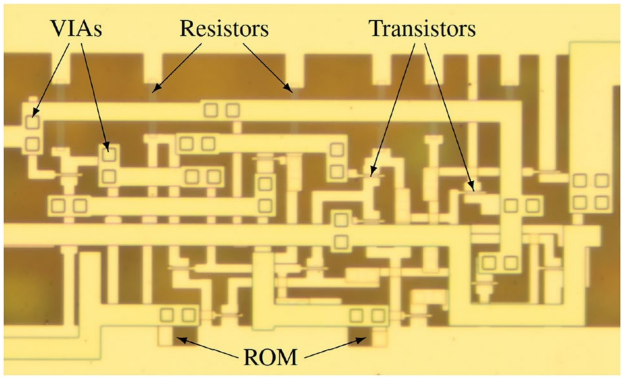

Fig. 1. Flexible electronics layout of a D-flip-flop circuit (with in-design programmable bit of ROM), visible under a regular microscope.

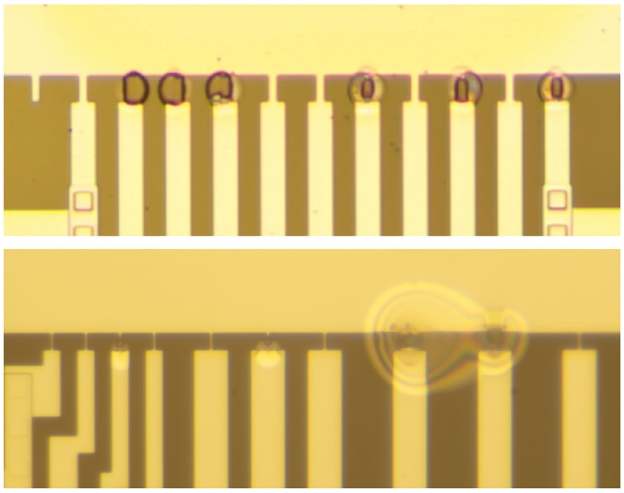

Fig. 2. Laser (top) and current (bottom) programmable fuses of OTP memories used in flexible electronics.

Introduction to resource-constrained technologies

Flexible electronics is a particularly good example of extremely resource-constrained technologies. There are at least a few dozen of such technologies combining several approaches with large diversity of parameters [3]. Commonly, their circuits are of particularly extensive size when compared to modern nanometer technologies, technological stacks only feature several layers, and MTP memory is difficult to develop. Nevertheless, the advantage of circuits being very thin and bendable, and their applicability to various surfaces, opens many doors for modern IoT devices, including: sensors, products tagging, wearable electronics, electronic skin and electronic paper, Internet of Medical Things (IoMT) or Internet of Healthcare Things (IoHT) applications, anti-counterfeiting and authenticity checking, etc. Majority of these technologies are still in a development phase, however, examples of advanced flexible devices are already available: RFID tag [10], ARM microprocessor [11,12], or sensors [13–15].Additional constraints come from the domain of security and refer with regard to side-channel attacks and reverse-engineering. There are many techniques falling into the category of reverse-engineering which are used to obtain information from semiconductor products [16]. Circuit extraction (called *deprocessing*) is a process which is opposite to chip fabrication and which has two main applications: one is to remove the passivation layer, exposing the top metal layer for microprobing attacks; the other — to obtain access to the deep layers and explore the internal structure of the chip [17]. In deprocessing of regular chips, there are three basic methods used: wet chemical etching, plasma or dry etching, and mechanical polishing. Flexible electronics, where a circuit layout is very often easily visible, is often blamed for extreme vulnerability — see Fig. 1. In the same way, the in-design programmable memory can be read — e.g., as placement of a VIA in layout presented in Fig. 1, as missing diffusion layer connection in NOR ROM, or brought through ion implantation in NAND ROM [18].Obtaining the data from OTP memories, which were written using the method of laser programming or by burning fuses is comparably easy — see Fig. 2. Laser programmable fuses need to be visible for programming, as a matter of course, and high current used for burning fuses leaves little irregular deformations in flexible circuits. There is also a lot of literature [17,19,20] on the topic of active side-channel attacks on RAM, Flash, EPROM, EEPROM, including: light, laser, X-rays, temperature, electromagnetic radiation etc.

Introduction to TRNG and PUF

The necessity of random numbers in IoT systems is unquestionable [21,22], especially with an appropriate level of randomness [7]. They are required in many fields of engineering — from cryptography (e.g., secret cipher keys, asymmetric encryption, digital signature algorithms, authenticity algorithms, etc.), to system testing (e.g., communication systems and complex digital circuits), to statistics (e.g., Monte-Carlo methods) [23,24]. Modern circuits of many types are equipped with cryptographic coprocessors that allow access to hardware-generated random streams as software functions. On the other hand, various lightweight systems (like IoT devices) — are expected to ensure an appropriate security level (high-quality randomness and resistance to side-channel attacks) despites the limited hardware resources.

Fast and cheap methods of generating random-like numbers under statistical tests called pseudo-random number generators (PRNGs) are available [21]. Various techniques can provide streams of pseudo-random numbers — for example: linear feedback shift registers [25], exponentiation in a multiplicative group [26], operations on elliptic curves [27], entropy accumulators that collect random data from various sources and use them to reseed the generator [28], and many others. However, a close analysis of a history of generation or analysis of the current state of PRNG can result in successful guessing of incoming numbers [29] or, in some cases, contain a backdoor inserted [30,31]. Therefore, generating truly random numbers usually involves entropy harvesting from some physical process, which is random at the level of physical phenomena.

Currently, we recognize many TRNGs based on various processes from which a few common ones are worth mentioning: amplification of noise processes [32], coherence detection of free-running oscillators [33], metastability, i.e., the uncertainty of either a logical state [34] or its resolve time [35] (alternatively estimated as a metastable oscillation number [36]), quantum random properties [37], chaotic circuits (depending on a method, playing the role of TRNG or PRNG [38,39]) and many others. Different techniques manifest different problems; however, one common drawback is that they require specific and stable initial or operating conditions. In our previous work we joined the advantages of analog chaotic signals and digital circuits by identifying a continuous random variable used in purely digital circuits (FPGA, CPLD) as an analog one [40]. This idea was later extended to a TRNG design in full custom front-to-end flexible integrated circuits technology based on indium-gallium-zinc oxide with thin-film transistors [41].

Physically unclonable functions are attracting immense attention as that they can provide physical obfuscated secrets — “a vault” for cryptographic keys. The advantages of particular PUFs include the following: (i) PUF keys are usually not present in the system (cryptographic keys are not kept in any volatile memory, non-volatile memory, latches nor registers); (ii) a key is temporarily activated (re-generated) when it is required in the system; (iii) keys are unique and different for every instance of a similar device (implemented/programmed in the same way); (iv) keys cannot be copied, cloned nor extracted from the device; as well as (v) they are tamper-proof (any attempt of tampering or measurement should destroy the keys). Furthermore, (vi) so called *strong PUF* (SPUF) maps input *challenge* (C) to output *response* (R) and offers unique challenge-response pairs (CRPs). Therefore, (vii) a key can be activated only by its owner (using owner’s challenge).

A search for methods of PUF harvesting from physical processes includes the following: ring oscillators [42], convergence time of bistable rings [43], sneak paths in the resistive X-point array [44], remanence decay effect [45], occurrence of metastability [46], locally enhanced defectivity [47], combination of multiplexers and arbiters [48], memristors [49], nonlinearities of data converters [50], carbon nanotube network surface [51], mismatch of capacitor ratios [52], customized dynamic two-stage comparator [53], and many others. There are also solutions that can combine both PUF and TRNG in one architecture [38,54].

To evaluate the PUF performance, common metrics are used: uniqueness, reliability, and uniformity [55]. Many strong PUFs are not ideal and suffer from vulnerability to machine learning attacks (MLA) [56], therefore, the underlying mechanism should be wisely chosen [57]. In fact, nowadays many security protocols in IoT use PUFs for authentication or key exchange [8,58].

Primitives for memoryless devices

The quality of random numbers generatable within the system as well as the type of strong PUFs chosen as platform for challenge-response pairs have crucial impact on the security level, implementation costs, and the time and power used for an operation. The TRNG and PUF mechanisms need to be chosen wisely depending on the type and purpose of device. Sometimes, a less secure solution has to be used to save resources. Other times, the choice may be dictated by platform limitations. The discussion thereof remains beyond the scope of this article, nevertheless, a few examples of TRNG and PUF mechanisms were presented in the discussion section of the paper.

Similarly, choosing the appropriate lengths of binary words of challenges and responses of CRPs requires a careful consideration. We do not determine the length of the words (it also depends on the PUF mechanism), however, we assume they were appropriately chosen, and they all are of equal length. We also assume that the data word is also of the same length. Therefore, simple operations: exclusive-or ($\oplus$) or comparison ($\overset{?}{=}$) can be easily performed. Of course, according to the needs, if there is more data or the response word is too short, the words can be combined into bigger blocks. In this paper the examples are simple for clarity of discussion.

Combining a binary message with a key of the same length using the exclusive-or (XOR) operator is an extremely simple encryption method known as Vernam cipher (from Gilbert S. Vernam). However, if a unique one-time key is used, the cipher “ensures a perfect form of confidentiality known as *perfect secrecy*” [59]. Of course, it lacks many properties of advanced ciphers (authentication, integrity etc.) and the key exchange between communication parties must be performed in a safe manner. In practice, the key requires memory and its length

	SERVER			DEVICE	
	data	compute	send	reply	retrieve/compute
(1)	$\{R_f\}$				$\{C_f\}$
(2)			ID_req $\longrightarrow$		
(3)					$R'_f = \text{puf}(C_f)$
(4)			$\longleftarrow$	R'_f	
(5)		$R_f \overset{?}{=} R'_f$			

Table 1. PUF identification based on fixed challenge.

	SERVER			DEVICE	
	data	compute	send	reply	retrieve/compute
(1)	$\{R_{f1},...,R_{fn}\}$				$\{C_{f1},...,C_{fn}\}$
(2)			ID_req $\longrightarrow$		
(3)					$R'_{f1} = \text{puf}(C_{f1})$
					$\vdots$
					$R'_{fn} = \text{puf}(C_{fn})$
(4)					$R'_f = [R'_{f1},...,R'_{fn}]$
(5)					$T = [\text{rng}(),...,\text{rng}()]$
					$[t_1,...,t_m] = T$
(6)					$R'_t = R'_f(T)$
(7)			$\longleftarrow$	T, R'_t	
(8)		$R_f = [R_{f1},...,R_{fn}]$			
(9)		$R_t = R_f(T)$			
(10)		$R_t \overset{?}{=} R'_t$			

Table 2. PUF identification with randomly chosen response.

must be sufficient to cover all future communication. The memory itself must provide secure data storage (e.g., protect against SCAs). Nevertheless, in case of using XOR with hardware-generated one-time keys (e.g., RNGs or PUFs), the security level is determined based on their individual properties.

Let us define the basic prerequisites for TRNG generation as $T = \text{rng}()$ and PUF generation as $R_x = \text{puf}(C_x)$. The function $\text{rng}()$ produces a random word of proper length — every use of the function produces a new result (T), preferably a truly random number. The function $\text{puf}()$ usually is a strong PUF function, preferably immune to various attacks (MLAs, SCAs, or other), which gives a specific response R_x to a specific challenge C_x. Individual responses look random, but they are the same for particular challenges for particular device. Other devices will have different responses to the same challenges and although they look random, the CRPs are stable and unique for every device.

Identification

The purpose of the memoryless device identification process may vary depending on application. It can be extremely simple, as in the case of product tagging, or more complex, as in case of preceding authentication where anonymity or untraceability is required. A unique tagging in a form of public barcode (for example with RFID devices) requires from the identification number (ID) only its uniqueness (maybe along with fixed number structure for producer, product group determination etc., which is not part of the problem). The problem in creating unique numbers for devices (for example consecutive numbers) in IC mass production is that the numbers cannot be different between wafers or even between reticles on the same wafer (since the reticle production masks are fixed). Therefore, a memory is necessary (at least OTP) to differentiate IDs for particular devices.We can avoid the necessity of equipping devices with memory by using PUF structure. The ID (or a part of ID) can be established to be a response R_f to particular challenge C_f, as presented in Tab. 1. The C_f can be (or shall be) set permanently in all the devices to any fixed value, and the responses (random alike) will differentiate the devices. If the number of devices is large and the binary length of the response is low (especially with poor PUF uniqueness), we can use a number of responses to a number of fixed challenges. This is to avoid ID collisions and lower the probability of having the same ID for any two devices, because of the birthday paradox involved in this scenario [60].

	SERVER			DEVICE	
	data	compute	send	reply	retrieve/compute
(1)	$\{C_x, R_x\}$				
(2)			C_x $\longrightarrow$		
(3)					$R'_x = \mathrm{puf}(C_x)$
(4)			$\longleftarrow$	R'_x	
(5)		$R_x \overset{?}{=} R'_x$			

Table 3. Simple PUF authentication.

	SERVER			DEVICE	
	data	compute	send	reply	retrieve/compute
(1)	$\{C_x, R_x\}$				D
(2)			C_x $\longrightarrow$		
(3)					$R'_x = \mathrm{puf}(C_x)$
(4)			$\longleftarrow$	$R'_x \oplus D$	
(5)		$D = R_x \oplus (R'_x \oplus D)$			

Table 4. Simple PUF encryption.

The server party does not need to know the C_f at all (or the series of chosen challenges). It just needs to ask for ID for the first time and save it for the future authentication purposes (1). Following the ID request (2), the device calculates the response R'_f with PUF mechanism (3) and sends it back (4). The server needs to check which of the devices corresponds to this response (5).The described way of identifying the device is extremely simple in its implementation. However, it does not offer anonymity or untraceability. If an observer has noticed the communication, they can track the ID in later communication. The risk of traceability can be reduced by responding with randomly chosen bits of a larger response vector (Tab. 2).

A number of n responses to a number of fixed n challenges (3) can be combined to a larger response R'_f vector (4), which should be initially known to the server (1). With a use of TRNG (for this purpose a RNG should be sufficient) the device can randomly choose (5) a smaller number of m bits from the R'_f vector (6) creating R'_t. Of course, the number (m) should be large enough to differentiate identified devices. The device sends back as a response (7) — both the chosen bits R'_t and their places T. The server, based on known responses (1,8), uses the placement T of chosen bits (9) to identify the device among the others (10).

It is obvious that this kind of obfuscation is not a foolproof countermeasure. The more CRPs the higher the untraceability, but also the more time and energy used to construct a response. On the other hand, the number of bits used for identification (m) can be kept at smaller value, with the assumption that: if a birthday problem occurs, the server will ask for yet another package of randomly chosen response bits and use them all to identify the device.

A difficulty appears in the device initialization part, when the server should be initially equipped with all the responses $\{R_{f1}, ..., R_{fn}\}$. Strict identification procedure (in the from presented in Tab. 2) would require a great number of ID requests and answers in order to fill the $\{R_{f1}, ..., R_{fn}\}$ data — and even then some of the bits may be missing (although not all of the bits are required for proper identification, which can be repeated). A different approach is to use a special command that performs initialization. The risk is that at some point the command may be revealed (security by obscurity is rarely a good idea). Another way is to equip the device with a fuse that will irreversibly switch from initialization state to normal work (either automatically after read completion or upon a command). However, technically, a fuse can be considered as a bit of OTP memory, which would mean that the device in question would not be considered "memoryless" any longer (we leave to the reader to decide if a fuse should be recognized as a memory or just as a part of device functionality).

Simple authentication or encryption

Authentication is usually performed after identification. On one hand, since a device is already identified as an instance, there is no birthday problem to occur. On the other hand, authentication needs to meet different security requirements.The simplest memoryless device authentication (presented in Tab. 3) requires at least one CRP stored at the server party (1). The server sends a challenge belonging to the identified device (2), to which a response is generated by the PUF circuit (3) and is sent back to the server party (4). Appropriate authentication of the device is verified by the server party by comparison of the received response with the previously retrieved response (5).The presented authentication can be modified to play the role of simple data encryption from the device to the server — Tab. 4. It can be easily performed by adding an exclusive-or (XOR) operation of calculated response R'_x and the data D (4). Since the server knows the CRP, it can decrypt the data. Such cases may find application in simple sensors, that on request take a measurement D and send it back, but not as a plaintext. The added value of this scheme is that if the decrypted data appear to be valid (5), then the device can be considered as authenticated.

These two procedures are characterized by their simplicity and low engagement of device resources. However, if the communication is vulnerable to eavesdropping, a CRP can be used only once. Of course, during the initialization phase, a number of CRPs can be collected and stored by the server for the purpose of multiple authentications/encryptions. However, the server must keep track on which CRPs have already been used, and there is an obvious limit to the number of authentications/encryptions.

In the ideal case — with the assumption that each CRP is used only once — the security would rely solely on the PUF quality (e.g., vulnerability to MLA). However, that case would require an extensive initialization phase with a number of CRPs that will cover every future authentication and encryption. This may be justified in scenarios where the sensor communicates incidentally, but probably not in situations where frequent transmission is required.

Since such authentication/encryption is extremely efficient in terms of time and energy on part of the device, it is worth discussing a smart approach, which will use resources of a much more powerful server. Depending on the target application, the server may manage the reuse of CPRs — keep some of them unused for the purpose of critical communication, and reuse groups of CRPs depending on various conditions. There can be groups of CRPs exchanged after a certain time of use. There can be groups used depending on the importance of communication. There can be scenarios for the cases when an attack is suspected. In general, the smart management should assure a proper level of trust and certainty when needed.

Mutual authentication

To prevent abuse of use of a device, the server may also be authenticated to the device, before the device performs any additional activity or sends a response. For that purpose, an additional CRP is needed (2) besides the one used for device authentication (1) — Tab. 5. Simple mutual authentication of both parties can be achieved by sending the additional CRP to the device (3), where at first the device uses a challenge (4) and validates the correctness of the CRP (5). If the CRP belongs to the device, it can respond with the other response (7) calculated as in previous procedure (6).

Unfortunately, the server authentication can be performed securely only once in an environment vulnerable to eavesdropping. Without a memory there is no possibility to track which CRPs were used for server authentication, therefore it should be reserved only for special tasks — especially irreversible (e.g. fuse burning). Alternatively, it can be used to distinguish between common and privileged communication (e.g. providing additional data).

Dynamic exchange of challenge-response pairs

The initialization phase, in which the server party saves a number of CRPs for each new device, can be significantly time consuming. An exchange of an old CRP to a new CRP can be achieved by extending the simple authentication (Tab. 3) to a method presented in Tab. 6. A new challenge C_n is randomly generated in the device using TRNG (4), and this challenge is used to generate a new PUF response R_n (5). The new challenge-response pair is then sent from the device to the server party along with the old response (6). If the device authentication is correct (7), the new CRP can replace the one previously used for each authentication (8), hence a chain exchange of CRPs occurs.

Obviously, such exchange has a limited value for secure authentication if communication is vulnerable to eavesdropping. Therefore, to maintain a proper security level of authentication, encryption is required.

	SERVER			**DEVICE**	
	data	**compute**	**send**	**reply**	**retrieve/compute**
(1)	$\{C_x, R_x\}$				
(2)	$\{C_y, R_y\}$				
(3)			$C_x, C_y, R_y \longrightarrow$		
(4)					$R'_y = \text{puf}(C_y)$
(5)					$R'_y \overset{?}{=} R_y$
(6)					$R'_x = \text{puf}(C_x)$
(7)			$\longleftarrow$	R'_x	
(8)		$R_x \overset{?}{=} R'_x$			

Table 5. PUF mutual authentication.

	SERVER			DEVICE	
	data	**compute**	**send**	**reply**	**retrieve/compute**
(1)	$\{C_o, R_o\}$				
(2)			C_o $\longrightarrow$	$\longrightarrow$	
(3)					$R'_o = \text{puf}(C_o)$
(4)					$C_n = \text{rng}()$
(5)					$R_n = \text{puf}(C_n)$
(6)			$\longleftarrow$	R'_o, C_n, R_n $\longleftarrow$	
(7)		$R_o \overset{?}{=} R'_o$			
(8)	$\{C_o, R_o\} \leftarrow \{C_n, R_n\}$				

Table 6. Dynamic exchange of PUF challenge-response pairs.

	SERVER			DEVICE	
	data	**compute**	**send**	**reply**	**retrieve/compute**
(1)	$\{C_x, R_x\}$				
(2)	D_s				
(3)		$K_s = \text{rng}()$			
(4)			C_x		
(5)			$K_s \oplus R_x$		
(6)			$K_s \oplus D_s$ $\longrightarrow$	$\longrightarrow$	
(7)					$R'_x = \text{puf}(C_x)$
(8)					$K_s = R'_x \oplus (K_s \oplus R_x)$
(9)					$D_s = K_s \oplus (K_s \oplus D_s)$

Table 7. One-time random key server-to-device encryption based on TRNG and PUF.

	SERVER			DEVICE	
	data	**compute**	**send**	**reply**	**retrieve/compute**
(1)	$\{C_y, R_y\}$				
(2)					D_d
(3)			C_y $\longrightarrow$	$\longrightarrow$	
(4)					$R'_y = \text{puf}(C_y)$
(5)					$K_r = \text{rng}()$
(6)				$K_r \oplus R'_y$	
(7)			$\longleftarrow$	$K_r \oplus D_d$ $\longleftarrow$	
(8)		$K_r = R_y \oplus (K_r \oplus R'_y)$			
(9)		$D_d = K_r \oplus (K_r \oplus D_d)$			

Table 8. One-time random key device-to-server encryption based on TRNG and PUF.

One-time random key encryption

One-time pad encryption is referred to in the literature as a technique that cannot be cracked, however, it requires the use of a single-use pre-shared key that is large enough to cover all messages being sent [59]. Such a key would require quite a big and very secure memory, which defies the goal of this study — memoryless IoT devices. However, the idea of using an encryption key only once is very attractive; even more so if the one-time keys are never written in any kind of solid memory — either MTP or OTP.Tables 7 and 8 introduce two-way encryption: for the communication sent to the device and received from the device. Each communication uses different randomly generated keys and can be considered separate; however, the returning communication (Tab. 8) has to be initiated by the server since the device cannot know (it has no memory) which (and how many) CPRs the server holds.

As presented in Tab. 7, the first temporary encryption key K_s is randomly generated by the server party (3). To pass the key to the device, one CRP is necessary (1). A challenge C_x and the encrypted corresponding response $K_s \oplus R_x$ (by simple XOR) are sent to the device (4,5) along with other data encrypted with the same key (6). It is sufficient to retrieve the key in the device (8) by calculating the response R'_x in the device (7). With the K_s key the session data D_s can be decrypted (9).

Similar mechanism can be used in the other direction (see Tab. 8). The second temporary encryption key K_r is randomly generated by the device (5) and sent back as encrypted with the response R'_y (6). The stored response R_y (1) is used by the server to calculate the receiving key K_r (8). Of course, encrypted data D_d can be send along (7) and decrypted (9).

Since the charts present single communication sessions, any other session (with the same server and the same device) will have a new set of random keys: K_s and K_r. An adversary, by observing the communication, fully knowing the procedure, can know C_x and calculate $R_x \oplus D_s$ and $R'_y \oplus D_d$, which are protected by unknown responses. If two-way encryption is performed with only one CRP, one can also calculate $K_s \oplus K_r$, which is random and new for every session. Moreover, they can know $D_s \oplus D_d$ within one session. If one of the data is compromised, the other can be known, therefore using different (or even exchanging) CRPs is preferred. If the server has sent an incorrect CRP or an adversary has had access to the device for testing, the device will still respond, but the return communication in (6 and 7) will be $K_r \oplus R_y{}'$ and $K_r \oplus D_d$ (still secured by random K_r and unknown R'_y). In the case when the adversary controls everything they send, they can receive $K_r \oplus R_y{}'$ in one session and XORs of particular K_r values between various sessions.

Another CRP can be sent to the device encrypted as data to authenticate the server to the device or the device to the server (presented in the next example). Another challenge can be sent encrypted as data to require even deeper encryption with a response known to both parties. Although the communication is encrypted, the device can be traced easily by observation of C_x. Therefore, the server may need to manage the use and reuse of CPRs smartly (as mentioned in the section about authentication), or an exchange of CRPs should be considered (as described in section about CRPs exchange).

Combined primitives

The primitives presented in the previous section can be included or excluded from communication depending on its purpose, requirements, hardware, or computational limitations. The most comprehensive variant integrating the procedures presented in Tables 2, 5, 6, 7, and 8, is shown in Table 9. Techniques may be arbitrarily incorporated into or excluded from the procedure as required or replaced (e.g., identification). The following steps from Table 9 can be classified based on their function:

(a) obfuscated identification of a device: (1)–(10);
(b) server-to-device encryption: (13), (15)–(17), (19)–(21);
(c) device-to-server encryption: (24), (29)–(36), (38)–(41);
(d) server authentication: (12), (16), (17), (22), (23);
(e) device authentication: (11), (14), (18), (29), (37);
(f) CRPs exchange: (25)–(28), (31)–(34), (42), (43).Since the ultimate comprehensive solution should lead to untraceability, the obfuscated (not simple) identification was used (1)–(10). Since it also should provide encrypted communication, a challenge (14) is required for the encryption key delivery, therefore, a constant change of C_x between sessions would be required. If the C_x changes for every session (42), an adversary can record the new encrypted challenge C_{n1} that will become future C_x and will be sent as plaintext. The future C_x could be later used to calculate the current K_r key and all returning communication in this session. Therefore, an additional encryption of C_{n1} is required (31) — either with another CRP, or with K_s (as in this case).

Of course, any additional data can be sent to the device using K_s or from device using K_r. However, in case of data compromising the key (e.g., can be guessed or have partially known structure), another additional CRP should be established and a second random encryption key can be used — using the same procedure as in steps (13)–(15), (18), (19) for server-to-device encryption or (24), (30), (35) for device-to-server encryption.

Discussion

Any security mechanism can be discussed in comparison with the state-of-the-art cryptography. However, some devices are resource-constrained or certain crucial practical aspects (e.g. exceptionally small dimensions or extremely low production price, visible memories, etc.) determine limits for implementation and, therefore, limits for security level that can be achieved. Nevertheless, extreme security may also be unnecessary for some devices that need only to publicly identify themselves with a few-dozen-bit barcode. Other devices may process critical information and they need to be certain about the identity of the other party, encrypt all the communication, be untraceable and must not reveal any secrets even under strong attacks or reverse-engineering. Different levels of vulnerability may also be acceptable for different applications. Whether communication should be immune only to eavesdropping of single sessions, or secure even with all the communication recorded, and whether the physical access to the device is limited, or if an adversary can play weeks of learning attacks on the device — remain valid considerations.

With respect to the presented primitives, crucially, the majority of security levels, implementation costs (e.g., silicon area) and the cost of operation (e.g., consumed energy and time) will be determined by the choice of PUF and TRNG techniques. A small and immune to all known attacks implementation would be ideal, but even when dealing with the top solutions, there is always a trade-off to make. For example, one PUF implementation can be a dozen times bigger than another, the another can be a threescore time more secure. Even within one type

	SERVER			DEVICE	
	data	**compute**	**send**	**reply**	**retrieve/compute**
(1)	$\{R_{f1},\ldots,R_{fn}\}$				$\{C_{f1},\ldots,C_{fn}\}$
(2)			ID_req $\longrightarrow$		
(3)					$R'_{f1}=\mathrm{puf}(C_{f1})$
					$\vdots$
					$R'_{fn}=\mathrm{puf}(C_{fn})$
(4)					$R'_f=[R'_{f1},\ldots,R'_{fn}]$
(5)					$T=[\mathrm{rng}(),\ldots,\mathrm{rng}()]$
					$\{[t_1,\ldots,t_m]=T\}$
(6)					$R'_t=R'_f(T)$
(7)			$\longleftarrow$	T,R'_t	
(8)		$R_f=[R_{f1},\ldots,R_{fn}]$			
(9)		$R_t=R_f(T)$			
(10)		$R_t\overset{?}{=}R'_t$			
(11)	$\{C_x,R_x\}$				
(12)	$\{C_y,R_y\}$				
(13)		$K_s=\mathrm{rng}()$			
(14)			C_x		
(15)			$K_s\oplus R_x$		
(16)			$K_s\oplus C_y$		
(17)			$K_s\oplus R_y$ $\longrightarrow$		
(18)					$R'_x=\mathrm{puf}(C_x)$
(19)					$K_s=R'_x\oplus(K_s\oplus R_x)$
(20)					$R_y=K_s\oplus(K_s\oplus R_y)$
(21)					$C_y=K_s\oplus(K_s\oplus C_y)$
(22)					$R'_y=\mathrm{puf}(C_y)$
(23)					$R'_y\overset{?}{=}R_y$
(24)					$K_r=\mathrm{rng}()$
(25)					$C_{n1}=\mathrm{rng}()$
(26)					$R_{n1}=\mathrm{puf}(C_{n1})$
(27)					$C_{n2}=\mathrm{rng}()$
(28)					$R_{n2}=\mathrm{puf}(C_{n2})$
(29)				$K_r\oplus R'_x$	
(30)				$K_r\oplus R_y$	
(31)				$K_r\oplus C_{n1}\oplus K_s$	
(32)				$K_r\oplus R_{n1}$	
(33)				$K_r\oplus C_{n2}$	
(34)			$\longleftarrow$	$K_r\oplus R_{n2}$	
(35)		$K_r=R_y\oplus(K_r\oplus R_y)$			
(36)		$R'_x=K_r\oplus(K_r\oplus R'_x)$			
(37)		$R_x\overset{?}{=}R'_x$			
(38)		$C_{n1}=K_r\oplus K_s\oplus\ldots$			
		$\ldots\oplus(K_r\oplus C_{n1}\oplus K_s)$			
(39)		$R_{n1}=K_r\oplus(K_r\oplus R_{n1})$			
(40)		$C_{n2}=K_r\oplus(K_r\oplus C_{n2})$			
(41)		$R_{n2}=K_r\oplus(K_r\oplus R_{n2})$			
(42)	$\{C_x,R_x\}\leftarrow\{C_{n1},R_{n1}\}$				
(43)	$\{C_y,R_y\}\leftarrow\{C_{n2},R_{n2}\}$				

Table 9. Combined primitives for untraceable identification, mutual authentication, and two-way encryption.

	Mechanism	Word length [bits]	Techn. [nm]	Silicon area [μm^2]	Throughput [Mbit/s]	Energy per bit [pJ]
SPUF	subthreshold current array [62]	65	130	44700	0.006	11
	subthreshold voltage divider array [61]	60	65	18700	12.5	0.3
	oscillation collapse in a ring-oscillator [63]	100	40	845	1.6	17.75
	inverter amplifiers [64]	65	28	188	0.8	0.118
TRNG	MOSFETs single-oxide traps noise [65]	1	120	9000	0.2	250
	ring oscillator based [66]	1	65	252	47.8	2.1
	metastability (pre-charged cross-coupled inverters) [67]	1	45	4004	2400	2.9
	jitter in 3-edge ring oscillator [68]	1	28	375	23.2	23
conventional techniques	symmetric encryption — 7 implementations of AES [69]	128–256	65	143000–471000	6400–156000	1.16–1.91
	hash function — SM3 hash[70]	256	65	75000	6540	1.1
	asymmetric algorithm — 3 implementations of RSA [71]	32	65	276000–311000	0.307–0.437	11980–25640
	asymmetric algorithm — ECC core* [72]	256	65	472000	0.173	39300
	2 kbit embedded flash memory (erase / write / read) [73]	16	65	86000	0.016 / 1.6 / 1600	1400 / 14 / 0.0016†

Table 10. Comparison of implementations costs of example strong PUFs and TRNGs as well as standard cryptography functions and memory. *1.48 ms per elliptic curve cryptography (ECC) point multiplication (PM) at 10 μJ per PM (256-bit prime field). †Energy per bit was estimated based on average cell current (2.19 μA) and nominal cell voltage (10/10/1.2 V).

Primitive	Table.	Required functions	Initialization time	Cost of execution	Security level
imple identification	1	PUF	fast (one or a few responses)	very low (1× puf())	no untraceability, birthday paradox vulnerability
complex identification	2	SPUF, TRNG, matrix register	mediocre (multiple responses)	high (several puf(), several rng(), random bits extractions)	adjustable untraceability
simple authentication	3	PUF	fast (one or a few CRPs)	very low (1× puf())	low security (eavesdropping vulnerability)
managed authentication	3	SPUF	mediocre (a number of CRPs)	very low (1× puf())	managed security (high ability to collect data for MLA)
extensive authentication	3	SPUF	very high (a lot of CRPs)	very low (1× puf())	high security (but high ability to collect data for MLA)
simple encryption	4	PUF, XOR	fast (one or a few CRPs)	very low (1× puf(), 1× $\oplus$)	low security (vulnerability to repetition)
managed encryption	4	SPUF, XOR	mediocre (a number of CRPs)	very low (1× puf(), 1× $\oplus$)	managed security (managed use of CRPs)
extensive encryption	4	SPUF, XOR	very high (a lot of CRPs)	very low (1× puf(), 1× $\oplus$)	high security (one-time-use of any CRP)
mutual authentication	5	SPUF, comparison	fast (two or a few CRPs)	very low (2× puf(), 1× $\overset{?}{=}$)	security depends on encryption and CRP management
CRPs exchange	6	SPUF, TRNG	very fast (one CRP)	low (2× puf(), 1× rng())	security depends on encryption of communication
server-to-device encryption	7	SPUF, XOR	fast (one or a few CRPs)	very low (1× puf(), 2× $\oplus$)	good security, but depends on number of CRPs
device-to-server encryption	8	SPUF, TRNG, XOR	fast (one or a few CRPs)	low (1× puf(), 1× rng(), 2× $\oplus$)	good security, but depends on number of CRPs
combined primitives (without identification)	9	SPUF, TRNG, XOR, comparison	fast (two CRPs)	mediocre (4× puf(), 3× rng(), 10× $\oplus$, 1× $\overset{?}{=}$)	high security (immune to MLA)

Table 11. Comparison of device resources required by particular primitives.

of PUF technique, shorter CRP words mean less security, but also a smaller, faster and less energy consuming circuit. An appropriate evaluation and choice of PUF and TRNG is crucial to making the best of a resource-constrained devices. Although the discussion and comparison of various TRNGs and PUFs remain beyond the scope of this paper, a few achievements in these fields were put together in Table 10. These solutions show the difference when compared with resources needed for classical cryptography implementations using standard memory. Of course, a variety of solutions is to be found, however, the purpose of this comparison is to provide a point of reference for standard implementation requirements.

Nevertheless, all the primitives presented here can be very easily assessed considering the type and number of mechanisms and operations required — presented in Table 11. If we look at the simplest identification, authentication and encryption (Tab. 1, 3, 4), they all need only one PUF mechanism and only one use of it per session (encryption — also one XOR). These cases are extremely valuable from the resource-constrained point of view, however, there are also significant disadvantages. Firstly, in order to build a functional CRPs database we need substantial time in a secure environment for the server to collect the data. Secondly, if the number of CRPs cannot cover all communication (it may be sufficient for occasional communication) an smart management of CPR resources would be required on the part of the server party. Thirdly, the traceability with simple identification is obvious, but the authentication and encryption are extremely vulnerable to machine learning, brute force, and other attacks. Therefore, the PUF technique has to be chosen wisely and may be resource expensive (PUF implementations differ significantly as shown in Table 10 or in the literature [61,62]).

More advanced primitives (cf. Tab. 9) may require TRNG as well as other operations, however, if there are no CRPs visible on the open channel (all responses are encrypted with a new set of random keys per session), machine learning attacks are useless. Since the device authenticates the server, it can also respond with useless data in cases when authentications are not valid. Regular changes or CRPs also makes the server part less demanding since it has to keep only two CRPs per device with no management at all. In this case the devices need to be equipped with one PUF and one TRNG mechanism. In the second phase (after identification) they are used multiple times: PUF — 4 times, TRNG — 3 times, XOR — 10 times, and 1 comparison (although XORs and comparisons are too inexpensive to be counted at all).

The implementation cost of identification may differ depending on the target untraceability level. For example, in the case presented in Table 11, having 32-bit TRNG and PUF words, 64-bit identification number, and 512 bits of ID-space (from which the 64-bit ID will be randomly chosen): PUF will be used 16 times in order to generate 512 bits of ID-space, and since we need to address 64 bits within 512-bit space (64 of 9-bit addresses), TRNG will be used 18 times. It shows that untraceability costs are the highest. Of course, there may be additional CRPs and operations depending on the amount of transferred data. Furthermore, the example uses only one server party, but since a device has no memory, it can communicate with anyone who holds its CRPs.

Finally, when memoryless primitives are considered, it is hard to compare them with cryptographic algorithms or protocols, because they are based on different assumptions (having OTP and/or MTP memory). No solutions for memoryless devices have been found yet. Nevertheless, many lightweight ideas use TRNGs and PUFs in order to take advantage of good quality random numbers, hardware key vaults or strong PUF mechanisms. A comparison of a few such solutions is presented in Table 12. Unfortunately, when a protocol involves certain functions or functionalities (encryption, authentication, hash function, elliptic cure cryptography etc., or especially non-volatile memory) it requires their implementation, which costs resources (area, power and time). Many of the presented solutions (cf. the literature from Tab. 12) do not specify implementation costs or even the types of mechanisms they use (hash function, encryption, random numbers generation, etc.) and leave the choice to be made based on target application.

Conclusion

The article showcases equipping memoryless IoT devices with basic security mechanisms, as long as they can have one PUF mechanism and one TRNG mechanism implemented (in some cases only the first one). Besides these mechanisms, only basic operations, such as XOR and comparison, are required. The work is dedicated to significantly resource-constrained devices, which cannot handle complex algorithms of modern secure cryptography. The primitives for identification, authentication, and encryption can be included or excluded from a security procedure depending on the requirements. Especially, such problems as untraceable identification, mutual authentication, and temporary key encryption were addressed. As a result, a functionality can be achieved starting with single PUF operation and, in most advanced cases, with several PUF/TRNG operations.

Additional information

The primitives have recently been domestically patented (in Poland), however, the international PCT procedure [9] has not been extended to any other country or jurisdiction, therefore the ideas presented here are available

Protocol purpose/idea	Provided functionality	Required functions	Use of memory
Lightweight protocol for IoT sensors [74]	authentication	SPUF, cipher (RC5), XOR, RNG, timestamp	yes
PLAKE: lightweight primitives for IoT protocol [75]	authentication, key exchange	SPUF, hash, XOR, RNG, congruence modulo n	yes
PUF Based Authentication Protocol for IoT [76]	authentication, key agreement scheme	SPUF(FSM), ECC (160-bit), hash (SHA2 or SHA3), RNG, timestamp	yes
Lightweight and anonymous protocol for edge IoT nodes [77]	mutual authentication, key agreement	hash, XOR, RNG, SPUF, fuzzy extractor, timestamp	yes
Lightweight scheme for IoT fog architecture [78]	mutual authentication	SPUF, fuzzy extractor, encryption, hash, RNG	yes
PLGAKD: lightweight group protocol [79]	authentication, key distribution	SPUF, factorial tree, CRT, encryption, XOR, hash (HMAC), RNG, timestamp	yes
Blockchain-based lightweight protocol for IoMT [80]	authentication	blockchain, SPUF, fuzzy extractor, RNG, hash, XOR, timestamp	yes
Lightweight IoT protocol for resource constrained nodes [81]	continuous authentication, mutual authentication	SPUF, public key encryption/decryption, hash (HMAC), TRNG, XOR	yes
Lightweight IoT protocol without CRPs database [82]	authentication	geometric threshold secret sharing, SPUF, RNG, hash, signature, XOR, modulo prime	yes
This work	**Iidentification, mutual authentication, encryption, untraceability**	**SPUF, TRNG, XOR, comparison**	**no**

Table 12. Comparison of lightweight PUF-based security protocols for IoT.

free of charge worldwide. Patents family can be found in various databases, and the domestic patents directly concerning methods for identification, authentication and encryption are enclosed in three Polish patents: PL238956, PL241997, and PL242117.

Data availibility

The datasets used and/or analysed during the current study available from the corresponding author on reasonable request.

Received: 28 May 2024; Accepted: 4 October 2024
Published online: 14 October 2024

References

1. Takeda, Y. et al. Fabrication of ultra-thin printed organic tft cmos logic circuits optimized for low-voltage wearable sensor applications. *Scientific reports* **6**, 25714. https://doi.org/10.1038/srep25714 (2016).
2. Bhalerao, S. R., Lupo, D. & Berger, P. R. Flexible thin film transistor (TFT) and circuits for internet of things (IoT) based on solution processed indium gallium zinc oxide (IGZO). In *2021 IEEE International Flexible Electronics Technology Conference (IFETC)*, 0023–0025, https://doi.org/10.1109/IFETC49530.2021.9580506 (2021).
3. Mirshojaeian Hosseini, M. J. & Nawrocki, R. A. A review of the progress of thin-film transistors and their technologies for flexible electronics. *Micromachines* **12**, 1–19, https://doi.org/10.3390/mi12060655 (2021).
4. Abutaha, M., Atawneh, B., Hammouri, L. & Kaddoum, G. Secure lightweight cryptosystem for iot and pervasive computing. *Scientific Reports* **12**, 19649. https://doi.org/10.1038/s41598-022-20373-7 (2022).
5. Gong, H. & Ju, T. Distributed power analysis attack on sm4 encryption chip. *Scientific Reports* **14**, 1007. https://doi.org/10.1038/s41598-023-50220-2 (2024).
6. Hu, F., Wang, H. & Wang, J. Cross subkey side channel analysis based on small samples. *Scientific Reports* **12**, 6254. https://doi.org/10.1038/s41598-022-10279-9 (2022).
7. Baturone, I., Román, R. & Corbacho, Á. A unified multibit PUF and TRNG based on ring oscillators for secure IoT devices. *IEEE Internet of Things Journal* **10**, 6182–6192. https://doi.org/10.1109/JIOT.2022.3224298 (2023).
8. Lee, S.-W. et al. Designing secure puf-based authentication protocols for constrained environments. *Scientific Reports* **13**, 21702. https://doi.org/10.1038/s41598-023-48464-z (2023).
9. Gołofit, K. Electronic seal and method of electronic seal verification (2020). (patent family: PL238366, PL238956, PL242116, PL241997, PL242117).
10. Wieczorek, P. Z., Starecki, K., Gołofit, K., Radtke, M. & Pilarz, M. A thin elastic NFC Forum type 1 compatible RFID tag. *IEEE Journal of Solid-State Circuits* 1–12, https://doi.org/10.1109/JSSC.2023.3300256 (2023).
11. Myny, K. et al. A thin-film microprocessor with inkjet print-programmable memory. *Scientific reports* **4**, 7398. https://doi.org/10.1038/srep07398 (2014).
12. Biggs, J. et al. A natively flexible 32-bit arm microprocessor. *Nature* **595**, 532–536. https://doi.org/10.1038/s41586-021-03625-w (2021).
13. Meister, T., Ishida, K., Carta, C., Münzenrieder, N. & Ellinger, F. Flexible electronics for wireless communication: A technology and circuit design review with an application example. *IEEE Microwave Magazine* **23**, 24–44. https://doi.org/10.1109/MMM.2021.3136684 (2022).
14. Ozer, E. et al. Malodour classification with low-cost flexible electronics. *Nature Communications* **14**, 777. https://doi.org/10.1038/s41467-023-36104-z (2023).
15. Liu, P. T. et al. Highly responsive blue light sensor with amorphous indium-zinc-oxide thin-film transistor based architecture. *Scientific reports* **8**, 8153. https://doi.org/10.1038/s41598-018-26580-5 (2018).
16. Torrance, R. & James, D. The state-of-the-art in IC reverse engineering. In *International Workshop on Cryptographic Hardware and Embedded Systems*, 363–381 (Springer, 2009).
17. Skorobogatov, S. P. Semi-invasive attacks — A new approach to hardware security analysis. Tech. Rep. No. 630, UCAM-CL-TR-630, University of Cambridge, Computer Laboratory (2005).
18. Kömmerling, O. & Kuhn, M. G. Design principles for tamper-resistant smartcard processors. *Smartcard* **99**, 9–20 (1999).
19. Schmidt, J.-M., Hutter, M. & Plos, T. Optical fault attacks on aes: A threat in violet. In *2009 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, 13–22 (IEEE, 2009).
20. Standaert, F. X. *et al.* Electromagnetic analysis and fault attacks: State of the art. *ECRYPT, Network of Excelence in Cryptography* (2005).
21. Kietzmann, P., Schmidt, T. C. & Wählisch, M. A guideline on pseudorandom number generation (PRNG) in the IoT. *ACM Computing Surveys (CSUR)* **54**, 1–38 (2021).
22. Ji, Z., Brown, J. & Zhang, J. True random number generator (TRNG) for secure communications in the era of IoT. In *2020 China Semiconductor Technology International Conference (CSTIC)*, 1–5, https://doi.org/10.1109/CSTIC49141.2020.9282535 (2020).
23. Sunar, B. True random number generators for cryptography. *In Cryptographic Engineering, chap.* **4**, 55–73. https://doi.org/10.1007/978-0-387-71817-0 (Springer 2009).
24. L'Ecuyer, P. Random numbers for simulation. *Communications of the ACM* **33**, 85–97. https://doi.org/10.1145/84537.84555 (1990).
25. Panda, A. K., Rajput, P. & Shukla, B. Design of multi bit lfsr pnrg and performance comparison on fpga using vhdl. *International Journal of Advances in Engineering & Technology* **3**, 566–571 (2012).
26. Blum, L., Blum, M. & Shub, M. A simple unpredictable pseudo-random number generator. *SIAM Journal on computing* **15**, 364–383 (1986).
27. AbdElHaleem, S. H., Abd-El-Hafiz, S. K. & Radwan, A. G. A generalized framework for elliptic curves based prng and its utilization in image encryption. *Scientific Reports* **12**, 13278. https://doi.org/10.1038/s41598-022-17045-x (2022).
28. McEvoy, R., Curran, J., Cotter, P. & Murphy, C. Fortuna: cryptographically secure pseudo-random number generation in software and hardware. In *2006 IET Irish Signals and Systems Conference*, 457–462 (IET, 2006).
29. Pasalic, E. On guess and determine cryptanalysis of lfsr-based stream ciphers. *IEEE Transactions on Information Theory* **55**, 3398–3406. https://doi.org/10.1109/TIT.2009.2021316 (2009).
30. Easttom, W. *Random Number Generators*, chap. 12, 257–276 (Springer, 2021).
31. Kostyuk, N. & Landau, S. Dueling over dual_ec_drbg: The consequences of corrupting a cryptographic standardization process. *Harv. Nat'l Sec. J.* **13**, 224 (2022).
32. Jun, B. & Kocher, P. The intel random number generator. *Cryptography Research Inc. white paper* **27**, 1–8 (1999).
33. Robson, S., Leung, B. & Gong, G. Truly random number generator based on a ring oscillator utilizing last passage time. *IEEE Transactions on Circuits and Systems II: Express Briefs* **61**, 937–941. https://doi.org/10.1109/TCSII.2014.2362715 (2014).
34. Hata, H. & Ichikawa, S. Fpga implementation of metastability-based true random number generator. *IEICE TRANSACTIONS on Information and Systems* **95**, 426–436 (2012).

35. Wieczorek, P. Z. & Gołofit, K. Dual-metastability time-competitive true random number generator. *IEEE Transactions on Circuits and Systems I: Regular Papers* **61**, 134–145. https://doi.org/10.1109/TCSI.2013.2265952 (2014).
36. Varchola, M. & Drutarovsky, M. New high entropy element for fpga based true random number generators. In *Cryptographic Hardware and Embedded Systems, CHES 2010, 351–365 (Springer* (eds Mangard, S. & Standaert, F.-X.) (Berlin Heidelberg, Berlin, Heidelberg, 2010).
37. Jacak, M. M., Jóźwiak, P., Niemczuk, J. & Jacak, J. E. Quantum generators of random numbers. *Scientific Reports* **11**, 16108. https://doi.org/10.1038/s41598-021-95388-7 (2021).
38. Gołofit, K. & Wieczorek, P. Z. Chaos-based physical unclonable functions. *Applied Sciences* **9**, 1–17. https://doi.org/10.3390/app9050991 (2019).
39. Güler, Ü. & Ergün, S. A high speed, fully digital ic random number generator. *AEU - International Journal of Electronics and Communications* **66**, 143–149. https://doi.org/10.1016/j.aeue.2011.06.001 (2012).
40. Wieczorek, P. Z. & Gołofit, K. True random number generator based on flip-flop resolve time instability boosted by random chaotic source. *IEEE Transactions on Circuits and Systems I: Regular Papers* **65**, 1279–1292. https://doi.org/10.1109/TCSI.2017.2751144 (2018).
41. Gołofit, K., Wieczorek, P. Z. & Pilarz, M. A chaos-metastability TRNG for natively flexible IGZO circuits. *AEU — International Journal of Electronics and Communications* **170**, 154835, https://doi.org/10.1016/j.aeue.2023.154835 (2023).
42. Barbareschi, M., Natale, G. D., Torres, L. & Mazzeo, A. A ring oscillator-based identification mechanism immune to aging and external working conditions. *IEEE Transactions on Circuits and Systems I: Regular Papers* **65**, 700–711. https://doi.org/10.1109/TCSI.2017.2727546 (2018).
43. Tanaka, Y., Bian, S., Hiromoto, M. & Sato, T. Coin flipping PUF: A novel PUF with improved resistance against machine learning attacks. *IEEE Transactions on Circuits and Systems II: Express Briefs* **(Early Access)**, 1–1 (2018). DOI: https://doi.org/10.1109/TCSII.2018.2821267.
44. Liu, R., Chen, P. Y., Peng, X. & Yu, S. X-point PUF: Exploiting sneak paths for a strong physical unclonable function design. *IEEE Transactions on Circuits and Systems I: Regular Papers* 1–10, https://doi.org/10.1109/TCSI.2018.2811643 (2018).
45. Kama, A. et al. Juliet-PUF: Enhancing the security of IoT-based SRAM-PUFs using the remanence decay effect. *IEEE Internet of Things Journal* **10**, 12715–12727. https://doi.org/10.1109/JIOT.2023.3253258 (2023).
46. Wieczorek, P. Z. & Gołofit, K. Metastability occurrence based physical unclonable functions for FPGAs. *Electronics Letters* **50**, 281–283. https://doi.org/10.1049/el.2014.0143 (2014).
47. Wang, W. C., Yona, Y., Diggavi, S. N. & Gupta, P. Design and analysis of stability-guaranteed PUFs. *IEEE Transactions on Information Forensics and Security* **13**, 978–992. https://doi.org/10.1109/TIFS.2017.2774761 (2018).
48. Sahoo, D. P., Mukhopadhyay, D., Chakraborty, R. S. & Nguyen, P. H. A multiplexer-based arbiter PUF composition with enhanced reliability and security. *IEEE Transactions on Computers* **67**, 403–417. https://doi.org/10.1109/TC.2017.2749226 (2018).
49. Ibrahim, H. M., Abunahla, H., Mohammad, B. & AlKhzaimi, H. Memristor-based puf for lightweight cryptographic randomness. *Scientific reports* **12**, 8633. https://doi.org/10.1038/s41598-022-11240-6 (2022).
50. Herkle, A., Becker, J. & Ortmanns, M. Exploiting weak PUFs from data converter nonlinearity—e.g., a multibit CT $\Delta\Sigma$ modulator. *IEEE Transactions on Circuits and Systems I: Regular Papers* **63**, 994–1004, https://doi.org/10.1109/TCSI.2016.2555238 (2016).
51. Jeong, J.-S., Lee, G. S., Park, T.-E., Lee, K.-Y. & Ju, H. Bio-inspired electronic fingerprint puf device with single-walled carbon nanotube network surface mediated by m13 bacteriophage template. *Scientific reports* **12**, 20096. https://doi.org/10.1038/s41598-022-24658-9 (2022).
52. Wan, M., He, Z., Han, S., Dai, K. & Zou, X. An invasive-attack-resistant PUF based on switched-capacitor circuit. *IEEE Transactions on Circuits and Systems I: Regular Papers* **62**, 2024–2034. https://doi.org/10.1109/TCSI.2015.2440739 (2015).
53. Tao, S. & Dubrova, E. Ultra-energy-efficient temperature-stable physical unclonable function in 65 nm CMOS. *Electronics Letters* **52**, 805–806. https://doi.org/10.1049/el.2016.0292 (2016).
54. Rai, V. K., Tripathy, S. & Mathew, J. TRGP: A low-cost re-configurable TRNG-PUF architecture for IoT. In *2021 22nd International Symposium on Quality Electronic Design (ISQED)*, 420–425, https://doi.org/10.1109/ISQED51717.2021.9424347 (IEEE, 2021).
55. Hori, Y., Yoshida, T., Katashita, T. & Satoh, A. Quantitative and statistical performance evaluation of arbiter physical unclonable functions on fpgas. In *2010 International Conference on Reconfigurable Computing and FPGAs*, 298–303, https://doi.org/10.1109/ReConFig.2010.24 (2010).
56. Santiago, L. *et al.* Realizing strong PUF from weak PUF via neural computing. In *2017 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, 1–6, https://doi.org/10.1109/DFT.2017.8244433 (2017).
57. Zhang, J., Shen, C., Guo, Z., Wu, Q. & Chang, W. Ct puf: Configurable tristate puf against machine learning attacks for iot security. *IEEE Internet of Things Journal* **9**, 14452–14462. https://doi.org/10.1109/JIOT.2021.3090475 (2022).
58. Nikolopoulos, G. M. & Diamanti, E. Continuous-variable quantum authentication of physical unclonable keys. *Scientific reports* **7**, 46047. https://doi.org/10.1038/srep46047 (2017).
59. Lugrin, T. *Trends in Data Protection and Encryption Technologies – Chapter 1: One-Time Pad, 3–6* (Springer Nature Switzerland, Cham, 2023).
60. Suzuki, K., Tonien, D., Kurosawa, K. & Toyota, K. Birthday paradox for multi-collisions. In *Information Security and Cryptology - ICISC 2006, 29–40 (Springer* (eds Rhee, M. S. & Lee, B.) (Berlin Heidelberg, Berlin, Heidelberg, 2006).
61. Venkatesh, A., Venkatasubramaniyan, A. B., Xi, X. & Sanyal, A. 0.3 pj/bit machine learning resistant strong PUF using subthreshold voltage divider array. *IEEE Transactions on Circuits and Systems II: Express Briefs* **67**, 1394–1398, https://doi.org/10.1109/TCSII.2019.2943121 (2020).
62. Zhuang, H., Xi, X., Sun, N. & Orshansky, M. A strong subthreshold current array PUF resilient to machine learning attacks. *IEEE Transactions on Circuits and Systems I: Regular Papers* **67**, 135–144. https://doi.org/10.1109/TCSI.2019.2945247 (2020).
63. Yang, K., Dong, Q., Blaauw, D. & Sylvester, D. A physically unclonable function with BER<10-8 for robust chip authentication using oscillator collapse in 40nm cmos. In *2015 IEEE International Solid-State Circuits Conference - (ISSCC) Digest of Technical Papers*, 1–3, https://doi.org/10.1109/ISSCC.2015.7063022 (2015).
64. Lai, Y.-C., Yao, C.-Y., Yang, S.-H., Wu, Y.-W. & Liu, T.-T. A robust area-efficient physically unclonable function with high machine learning attack resilience in 28-nm CMOS. *IEEE Transactions on Circuits and Systems I: Regular Papers* **69**, 347–355. https://doi.org/10.1109/TCSI.2021.3098018 (2022).
65. Brederlow, R., Prakash, R., Paulus, C. & Thewes, R. A low-power true random number generator using random telegraph noise of single oxide-traps. In *2006 IEEE International Solid State Circuits Conference - Digest of Technical Papers*, 1666–1675, https://doi.org/10.1109/ISSCC.2006.1696222 (2006).
66. Klein, N., Harel, E. & Levi, I. The cost of a true random bit–on the electronic cost gain of ASIC time-domain-based TRNGs. *Cryptography* **5**, 25. https://doi.org/10.3390/cryptography5030025 (2021).
67. Mathew, S. K. *et al.* 2.4 gbps, 7 mw all-digital PVT-variation tolerant true random number generator for 45 nm CMOS high-performance microprocessors. *IEEE Journal of Solid-State Circuits* **47**, 2807–2821, https://doi.org/10.1109/JSSC.2012.2217631 (2012).
68. Yang, K. *et al.* A 23mb/s 23pj/b fully synthesized true-random-number generator in 28nm and 65nm CMOS. In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, 280–281, https://doi.org/10.1109/ISSCC.2014.6757434 (2014).
69. LaPiana, A. *A comparative analysis of different AES implementations for 65nm technologies*. Master's thesis, Oklahoma State University (2015).

70. Du, X. & Li, S. The asic implementation of sm3 hash algorithm for high throughput. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* **99**, 1481–1487 (2016).
71. Moreira, M., Oliveira, B., Moraes, F. & Calazans, N. Impact of C-elements in asynchronous circuits. In *Thirteenth International Symposium on Quality Electronic Design (ISQED)*, 437–343, https://doi.org/10.1109/ISQED.2012.6187530 (2012).
72. Marzouqi, H., Al-Qutayri, M., Salah, K. & Saleh, H. A 65nm asic based 256 nist prime field ecc processor. In *2016 IEEE 59th International Midwest Symposium on Circuits and Systems (MWSCAS)*, 1–4, https://doi.org/10.1109/MWSCAS.2016.7870035 (2016).
73. Song, S.-H., Chun, K. C. & Kim, C. H. A logic-compatible embedded flash memory for zero-standby power system-on-chips featuring a multi-story high voltage switch and a selective refresh scheme. *IEEE Journal of Solid-State Circuits* **48**, 1302–1314. https://doi.org/10.1109/JSSC.2013.2247691 (2013).
74. Yilmaz, Y., Gunn, S. R. & Halak, B. Lightweight PUF-based authentication protocol for IoT devices. In *2018 IEEE 3rd International Verification and Security Workshop (IVSW)*, 38–43, https://doi.org/10.1109/IVSW.2018.8494884 (2018).
75. Roy, S. et al. PLAKE: PUF-based secure lightweight authentication and key exchange protocol for iot. *IEEE Internet of Things Journal* **10**, 8547–8559. https://doi.org/10.1109/JIOT.2022.3202265 (2023).
76. Braeken, A. PUF based authentication protocol for IoT. *Symmetry* **10**, 352. https://doi.org/10.3390/sym10080352 (2018).
77. Wang, H., Meng, J., Du, X., Cao, T. & Xie, Y. Lightweight and anonymous mutual authentication protocol for edge IoT nodes with physical unclonable function. *Security and Communication Networks* **2022**, 1203691. https://doi.org/10.1155/2022/1203691 (2022).
78. De Smet, R., Vandervelden, T., Steenhaut, K. & Braeken, A. Lightweight PUF based authentication scheme for fog architecture. *Wireless Networks* **27**, 947–959. https://doi.org/10.1007/s11276-020-02491-0 (2021).
79. Yıldız, H., Cenk, M. & Onur, E. PLGAKD: A PUF-based lightweight group authentication and key distribution protocol. *IEEE Internet of Things Journal* **8**, 5682–5696. https://doi.org/10.1109/JIOT.2020.3032757 (2021).
80. Wang, W. et al. Blockchain and PUF-based lightweight authentication protocol for wireless medical sensor networks. *IEEE Internet of Things Journal* **9**, 8883–8891. https://doi.org/10.1109/JIOT.2021.3117762 (2022).
81. Goutsos, K. & Bystrov, A. Lightweight puf-based continuous authentication protocol. In *2019 International Conference on Computing, Electronics & Communications Engineering (iCCECE)*, 229–234, https://doi.org/10.1109/iCCECE46942.2019.8941608 (2019).
82. Nimmy, K., Sankaran, S. & Achuthan, K. A novel lightweight PUF based authentication protocol for IoT without explicit CRPs in verifier database. *Journal of Ambient Intelligence and Humanized Computing* **14**, 6227–6242. https://doi.org/10.1007/s12652-021-03421-4 (2023).

Author contributions
All work has been contributed by the author.

Declarations

Competing interests
The author declares no competing interests.

Additional information
Correspondence and requests for materials should be addressed to K.G.