cryptography

MDPI

Article

A Robust, Low-Cost and Secure Authentication Scheme for IoT Applications †

Md Jubayer al Mahmod * and Ujjwal Guin

Department of Electrical and Computer Engineering, Auburn University, Auburn, AL 36849, USA; ujjwal.guin@auburn.edu

* Correspondence: jubayer@auburn.edu

† This paper is an extended version of our paper published in 31st International Conference on VLSI Design (VLSID) (Pune, India, 6–10 January 2018).

Received: 20 February 2020; Accepted: 6 March 2020; Published: 8 March 2020

Abstract: The edge devices connected to the Internet of Things (IoT) infrastructures are increasingly susceptible to piracy. These pirated edge devices pose a serious threat to security, as an adversary can get access to the private network through these non-authentic devices. It is necessary to authenticate an edge device over an unsecured channel to safeguard the network from being infiltrated through these fake devices. The implementation of security features demands extensive computational power and a large hardware/software overhead, both of which are difficult to satisfy because of inherent resource limitation in the IoT edge devices. This paper presents a low-cost authentication protocol for IoT edge devices that exploits power-up states of built-in SRAM for device fingerprint generations. Unclonable ID generated from the on-chip SRAM could be unreliable, and to circumvent this issue, we propose a novel ID matching scheme that alleviates the need for enhancing the reliability of the IDs generated from on-chip SRAMs. Security and different attack analysis show that the probability of impersonating an edge device by an adversary is insignificant. The protocol is implemented using a commercial microcontroller, which requires a small code overhead. However, no modification of device hardware is necessary.

Keywords: IoT security; SRAM; edge device; authentication

1. Introduction

The recent growth in Internet of Things (IoT) infrastructure has created an enormous potential for semiconductor design and manufacturing primarily to improve performance and enhance security. Billions of low-cost devices are connected to the Internet to provide seamless integration of computing systems to the physical world. These devices are commonly known as "*things*" or edge devices (*EDs*). The number of these connected devices has grown significantly and will continue growing at an astonishing rate in the near future [1–3]. As these devices are deployed in large geographic areas with limited energy resources and reachability, the power requirement becomes a significant issue for proper operations, which ultimately limits the usage of standard cryptographic schemes for secure operations [4]. Moreover, the cost of these devices requires them to have a low die area, which restricts the use of costly cryptographic primitives in the design. As a result, a majority of the *EDs* do not use computation and resource-heavy cryptographic schemes. HP mentioned in a report that 70% of the tested IoT devices communicate without encryption [5].

The use of cryptographic primitives does not necessarily ensure the authenticity of an *ED*, as the majority of them are manufactured offshore with limited trust and lack of government or other appropriate oversight. Moreover, as these devices travel through many distributors located across the globe, it is very difficult to determine their origin and the complete route in the supply chain. Many untrustworthy third party suppliers and distributors can introduce compromised devices, which look

and function exactly like the authentic ones. It is thus extremely difficult to ensure the authenticity of these resource constraints and low-cost edge devices. Numerous incidents, which include the presence of cloned systems in the US defense supply chain, indicate that inauthentic hardware entered into the electronics supply chain [6–16]. Figure 1 presents the simplified view of an Internet of Things infrastructure, where the *EDs* are connected to the Internet through gateways. In our threat model, we treat the gateways as trusted, as one can implement security measures using traditional cryptographic primitives. However, as the edge devices are limited in resources, it is extremely difficult to ensure their authenticity, as an *ED* can easily be cloned or counterfeited. Note that the hardware attacks can be initiated at the place, where the system is located. For example, an untrustworthy employee in an organization can practically replace an authentic device with its cloned counterpart to gain access to a secure system. In addition, a trusted user can unknowingly add a counterfeit device to the IoT infrastructure, since it is fairly impossible to track their origin if they are acquired from an untrustworthy distributor. Note that various software attacks, such as denial of service (DoS), phishing, and data spoofing, can be performed through untrustworthy hardware [17,18]. As the objective of this paper is to address the hardware attacks, we design our proposed solution to ensure the authenticity of *EDs*.

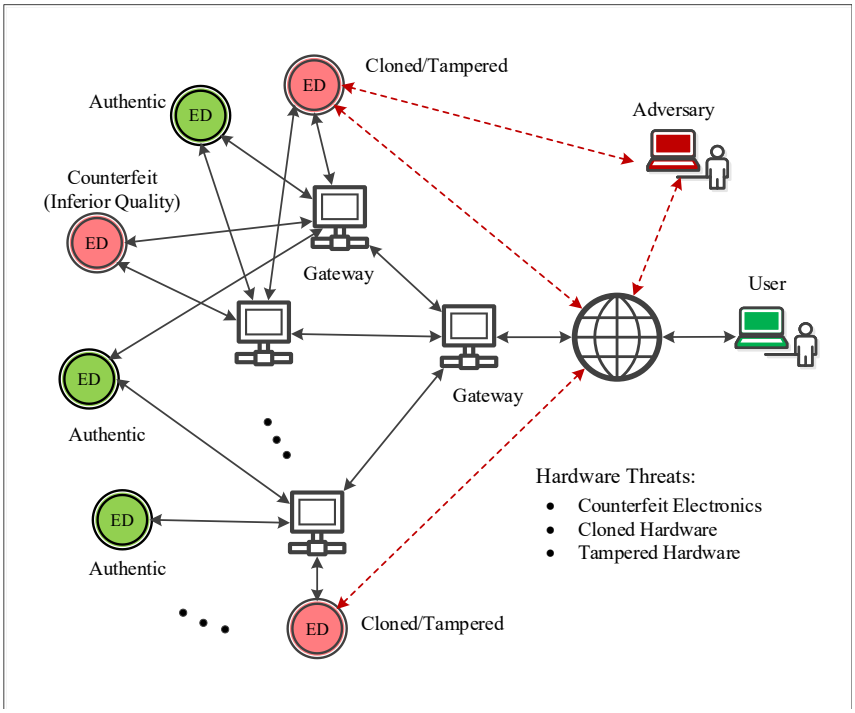

Figure 1. A typical IoT model that illustrates common hardware vulnerabilities.

1.1. Motivation

A recent study suggests that the IoT nodes are severely resource-constrained and they are not well equipped with standard cryptographic protocols [4,5,19]. It is essential for a user to authenticate an *ED* with very high confidence, as it may be cloned or tampered, and may have a secret backdoor, which can be exploited by an adversary. Traditionally, digital signatures [20] are widely used for end-point authentication, which generally uses Rivest–Shamir–Adleman (RSA) [21] or elliptic-curve cryptography (ECC) [22] crypto primitives. Implementing these primitives in *EDs* is too expensive due to severe resource constraints. A typical *ED* consists of an 8MHz microcontroller (MCU), a small (e.g., 128KB) flash memory, and a small (e.g., 10KB) RAM [23]. Software implantation may be preferred in case of hardware limitations; however, these crypto primitives are usually intensive in computation that may not be affordable for *EDs* as well.

The authentication of integrated circuits (ICs) can be performed using the use of physically unclonable functions (PUFs), as they can generate unique and unclonable bits for creating a unique

identifier (ID). Uncontrollable and unpredictable manufacturing process variations are utilized in PUFs to generate random and unclonable bits. Over the years, researchers proposed different PUF architectures and they are the arbiter PUF [24], ring oscillator (RO) PUF [25], SRAM PUF [26], and a few others [27–29]. Every authentication can be unique if a unique challenge-response pair (*CRP*) is used for every communication between the *EDs* and the gateway. To authenticate an IoT device over an unsecured channel using PUF as an ID generation unit, it is essential to have a strong PUF, with a very large number of *CRPs*. However, a large number of *CRPs* management and storage can be another intricate problem to manage for billions of connected devices. In addition, a strong PUF requires hardware modification for *EDs*, which might not be feasible. Instead, we can exploit built-in SRAM and use software support to generate the ID.

Typically *EDs* have on-chip SRAM that can be used to design a PUF, which would offer a cost-effective solution to produce an unclonable device ID. However, there are several challenges to using SRAM-PUF to create device IDs. The bits generated from an SRAM may be unstable and create different IDs for the same device. Error correction codes needs to be used to increase the reliability of the SRAM-PUF response. However, error correction codes usually consume larger memory space, which may not be available in these resource-constraint *EDs*. It is, thus, required to design a low-cost authentication scheme to uniquely identify *EDs* over an unsecured channel.

1.2. Contributions

We present a lightweight authentication protocol that verifies the authenticity of a resource constraint edge device. The communication protocol is low-cost, as it uses the resources available in an *ED*, such as a processor and an on-chip SRAM memory. In addition, we have further developed a robust repeated ID matching scheme to authenticate an *ED*, where the ID is created from an on-chip SRAM memory. Our scheme is superior over the simple ID matching scheme, as SRAM PUFs are often found to be unreliable. The contributions of our paper are as follows. We:

- *Developed a low-cost and secure communication protocol:* We propose a novel lightweight communication protocol that utilizes existing hardware resources of an edge device. A secure hash function [30] is used in our proposed protocol and it is implemented using an embedded processor and on-chip memory of an *ED* [31,32]. It is necessary to make sure that the unencrypted device ID does not leave the system. We show that the protocol is at least as secure as a hash function. We provide the security proof of our protocol in Section 5.3. In addition, the heuristic security evaluation shows that our proposed protocol is resistant to various known attacks.
- *Proposed repeated authentication for device identity verification:* We propose a novel repeated ID matching technique (see details in Section 4) to address the reliability issues arisen from an SRAM PUF. Our proposed solution does not require expensive helper data and algorithms for error correction, and thus can be lightweight. If a noisy SRAM PUF is used in the *ED*, an adversary might get lucky to pass the authentication once. However, it is highly unlikely that he/she will pass the authentication a second time and successfully register a fake device using random guesses unless the communication protocol described in Section 3.1 is broken. The unreliable bits from the SRAM PUFs can be identified in the proposed repeated ID matching scheme and will be excluded during the ID matching process. We demonstrate that it is highly unlikely for an adversary to impersonate an *ED* (see the details in Section 1). An adversary can pass the simple ID matching scheme by random trial (see Section 4.1) if the PUF responses are noisy. However, an adversary cannot impersonate an edge device two times with random guesses. Note that one can also implement an authentication scheme that verifies an *ED* more than two times to further increase the difficulty of impersonating an authentic device.
- *Implemented the proposed protocol in low-cost devices:* We have implemented this proposed protocol using Raspberry Pi as a gateway and Arduino as the edge devices. The *ED*s go through a registration process (see Algorithm 1), where the device signature is generated from the power-up states of the on-chip SRAM. We implemented the proposed protocol without any hardware

modifications. The IDs generated from the Arduino *EDs* show good uniqueness properties (see the Hamming distance's analysis in Section 6.4).

The rest of the paper is organized as follows. Section 2 briefly surveys the literature. Section 3 describes our proposed lightweight communication protocol to verify the identity of an *ED*. We present the ID matching scheme by repeated authentication in Section 4. The security evaluation is performed in Section 5. The implementation detail is described in Section 6. Finally, the paper is concluded in Section 7.

2. Prior Work

Over the years, researchers have proposed several PUF-based authentication protocols for resource constraint applications. Most of these protocols consists of a *prover* node, and a *verifier*. The *prover* is a node (e.g., sensors), which responds to the *verifier*'s (e.g., routers) query to confirm that it is an authorized node in the IoT system. To perform a verification, a random challenge is sent to the *prover* by a *verifier*, and the *prover* acknowledges by sending a valid response in return. The authors in [33] combined a delay PUF-based authentication protocol with an HB-based protocol (named after the authors) [34] to remove security vulnerabilities of these individual protocols. Later in their work, they proposed a protocol that reduces power and area overhead by using 2-level noisy PUF instead of using area and power-intensive cryptographic modules, such as hash functions [35]. Katzenbeisser et al. proposed a logically re-configurable PUF [36], which can be used to reduce excessive area requirement [37]. The above protocols use a *CRP* only once in order to prevent the replay attack.

Management and storage of *CRP*s at the *verifier's* end are pressing issues when it comes to millions of devices connected to the Internet. An adversary, listening to the communication among *prover* and *verifier* nodes, can model a PUF mathematically and predict the responses. Rührmair et al. presented a modeling attack for several PUF models including arbiter PUFs and ring oscillator PUFs [38]. To eliminate these issues, the *converse PUF-based* authentication protocol has been proposed in [39], where the *verifier* is considered a resource-constrained device, and the *prover* is considered a resource-rich server. A hardware and software co-verification based lightweight IoT authentication scheme is presented in [40]. The authors applied the hash of firmware along with PUF response to detect software and hardware impersonation attacks. Chatterjee et al. proposed an authentication scheme that combines the concepts of PUF, identity-based encryption (IBE), and keyed hash function to eliminate the need for explicitly storing *CRPs* [41]. Braeken et al. [42] proved that this protocol is vulnerable to the man-in-the-middle (MITM) attack, impersonation, and replay attacks [43].

A preshared key based host identity protocol is proposed in [44] to authenticate an IoT edge node. The primary shortcoming of the method is that adversary might gain access to the network if the shared key gets compromised. Kothmayr et al. proposed a two-way authentication protocol, entitled datagram transport layer security (DTLS) based on the X.509 certificate [45]. Porambage et al. proposed an implicit certificate-based two-phase authentication protocol [46]. Since the certificates are more lightweight than the protocol proposed in [45], distributed IoT applications are supported by this protocol. This implicit certificate-based protocol is suitable for highly resource-constrained devices. However, the protocol is vulnerable to replay attack, DoS attack, and man-in-the-middle (MITM) attack [47]. To mitigate these vulnerabilities, Turkanović et al. presented a four-step authentication model, which is suitable for the scenario where a remote user negotiates a session key with a sensor node without connecting to the gateway [48]. Challa et al. proposed a signature-based user authenticated key agreement scheme using Elgamal and ECC-based signature for IoT device authentication [47]. Although this protocol is advantageous in comparison to [46,48], it is computationally intensive. Hash function-based and chaos-based privacy-preserving schemes for IoT in a smart home system have been proposed in [49]. The authors used symmetric cryptosystem (e.g., Message Authentication Code or MAC) for both schemes. The heuristic suggests that the protocols are secure. Formal security analysis and verification would strengthen the contribution of the proposed method. Wazid et al. proposed three-factor—smart card, password, and personal

biometrics—remote user authentication for a hierarchical IoT network called the *user authenticated key management protocol* [50].

The majority of these protocols use a strong-PUF that requires hardware modification which ultimately imposes a strain on the resource limitation of *EDs*. Mass production and remote deployment are among the primary features of IoT devices. Adding extra hardware could be very expensive in terms of run-time power consumption and production cost. Therefore, a better approach would be to utilize existing hardware to implement security features that would be secure but lightweight.

3. Proposed Authentication Scheme

One of the major constraints in implementing standard authentication protocol is limited resources available in an *ED*. It is essential for an ED to have a low die area, smaller memory, and lower power consumption which, in effect, limit the performance. These constraints ultimately prohibit *EDs* from using standard secure protocols such as TLS and IPsec [4]. Less stress was given on security during the development of IoT edge devices considering that the generated or transmitted data would have little value to the attackers. It has been proven otherwise because this seemingly trivial information can be exploited to break into complex systems [51,52]. Authentication of an *ED* in an IoT network could help avoid the potential threat posed by counterfeit or cloned devices. In this section, we propose an authentication technique for an *ED* by correctly identifying its origin. To prevent various run attacks, the device needs to encrypt its data. However, advanced encryption methods are only feasible for gateways, since they have higher resources. Our proposed scheme does not address authentication of the gateways and treats them as authentic.

3.1. Proposed Communication Protocol

Figure 2 shows our proposed authentication approach. A true random number generator (TRNG) is necessary for the gateway device for random nonce (n) generation. We propose to use a cryptographically secure pseudo-random number generator (CSPRNG) such as [53] or [54] considering area efficiency in its implementation. A one-time-pad (OTP) [55,56] is employed to encrypt the key with n. Note that OTP is area efficient, as it only requires a simple XOR array. To authenticate the l^{th} *ED* in the network, we employed a pair of keys, $\{K_l, ID_l\}$. K_l is stored in an on-chip memory of an *ED*, and it is shared with the gateway. ID_l is a unique device signature that can be generated from an SRAM PUF (see Section 4 for details).

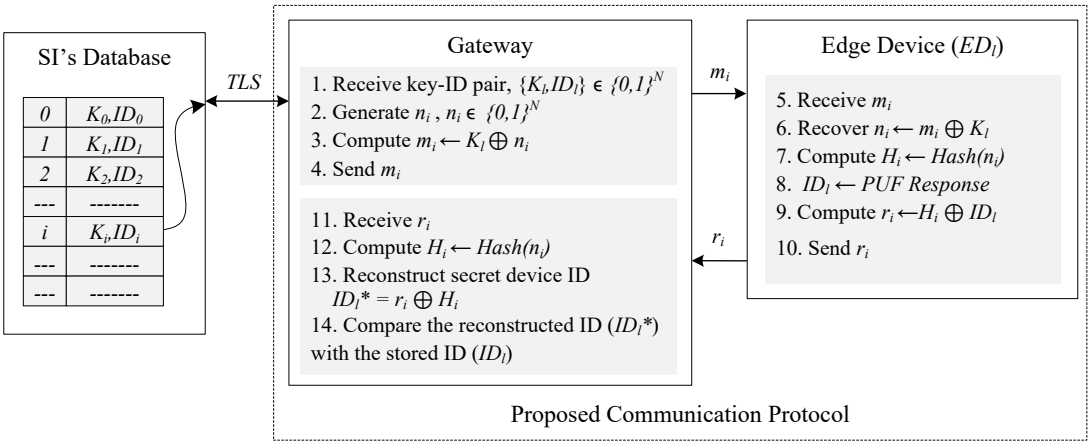

Figure 2. Proposed communication protocol for verifying the identity of an edge device (*ED*) [57].

The proposed communication protocol for authenticating an *ED* is described in the following steps:

1. The gateway receives the secret device key-ID pair $\{K_l, ID_l\}$ from the trusted system integrator (*SI*) using an existing secure communication protocol (e.g., TLS [58]). Here, SI is an entity that produced *ED*. During production, every device needs to be registered in a secure database with a public ID, and a key-ID pair, $\{ID_l \in \{0,1\}^N,\ K_l \in \{0,1\}^N\}$. Here, N is the length of the ID_l and K_l. Depending on the level of security one requires, N could vary. We analyzed the protocol for both $N = 128$ and $N = 256$, and implementation is only demonstrated for a 256 bit ID. The public ID is needed to locate the *ED* in the database. A tamper-proof memory in a gateway can also be used to store this data rather than transferring them from another database. However, since the gateway is always connected to the Internet, standard security measures can be implemented, as it typically does not have any resource limitations. However, it is recommended to receive the key-ID pair from the trusted integrator rather than store it into the gateway itself.
2. The gateway stores $\{K_l, ID_l\}$ in its on-chip (volatile or non-volatile) memory, but no information can be extracted through the input and output of the gateway. This will prevent an adversary from getting access to the $\{K_l, ID_l\}$, which was generated during the registration phase of the *ED*.
3. An on-chip CSPRNG generates a unique nonce (n_i), which is stored in the memory of the gateway. This n_i will be used later for decrypting the secret device ID. A one-time pad (OTP) now encrypts the key (K_l) with this random nonce. The gateway then sends this encrypted key (depicted as (m_i) in the figure) to the l^{th} ED, ED_l, to request for its identification.

$$m_i = n_i \oplus K_l \tag{1}$$

4. The nonce (n_i) is retrieved at the *ED* by XORing the m_i with the shared secret key (K_l). A secure hash (e.g., SHA-2 or SHA-3 [30]) is computed on this nonce (n_i) to produce a 256/512 bit hash output (H_i). Existing hardware resources such as embedded processor and memory [31,32] of the *EDs* are sufficient to compute this hash.

$$H_i = hash(n_i) \tag{2}$$

5. The device ID (ID_l) is created from the on-chip SRAM (extensive desription and analysis of SRAM PUF can be found in [59–63]) of the *ED*. Now, the *ED* encrypts ID_l using N bits of computed H. The encrypted ID $\{r_i\}$ is sent to the gateway for authentication.

$$r_i = ID_l \oplus H_i \tag{3}$$

6. The gateway computes the same hash (SHA-2 or SHA-3) using the n_i after receiving r_i. By this, the secret device ID is reconstructed in the gateway.

$$r_i \oplus H_i = ID_l \oplus H_i \oplus H_i = ID_l \tag{4}$$

7. This reconstructed ID is then verified with the stored ID (see Section 4 for details). Steps 3-6 are repeated for the second stage of the authentication to increase the confidence of an authentic *ED*.

3.2. Security Proof

A protocol preserves the secrecy if an adversary cannot determine secret data, e.g., key, with absolute certainty just by interacting with the protocol. Therefore, the protocol analysis primarily tries to determine a protocol trace that would compromise the secrecy of the system [64]. In this section, we will provide detailed proof for our proposed protocol.

The proposed protocol employs a symmetric key encryption system. First, the gateway sends a nonce encrypted using a shared key to an *ED*, *K*. Then, the *ED* uses the same shared key to recover the

nonce to transfer the ID to the gateway for authentication. Key freshness is ensured, since the nonce is transformed through a secure hash function before it is used again to encrypt the ID.

Let us assume that a probabilistic polynomial time adversary adv performs an authentication experiment $Auth_{adv,\pi}(n, H_{adv}, m_i, r_i, r_j)$ and tries to find a difference between a random data and private information; e.g., key K of the protocol π. The adversary adv has the following information:

1. The hash function H_{adv} with the security parameter N, which is the length of shared key K.
2. He/she can eavesdrop and collect message contents m_i, r_i, and r_j.

To prove that the protocol is secured, we need to show that the probability of success in the $Auth_{adv,\pi}$ experiment is negligible. In other words, there exists a function $negl(.)$ for every probabilistic polynomial time adversary such that:

$$Pr[Auth_{adv,\pi} = 1] \leq negl(N) \tag{5}$$

Definition 1. *For a secure hash function H, it is computationally infeasible to calculate the two input messages x_1 and x_2 ($x_1 \neq x_2$) such that $H(x_1) = H(x_2)$. This property of hash function is commonly denoted as collision resistance [65].*

Definition 2. *Given an N-bit output of a secure hash function H is z, it is computationally infeasible to calculate the input message x such that $H(x) = z$. This property of hash function is commonly denoted as preimage resistance [65].*

Theorem 1. *The protocol π is secured from a polynomial time adversary such that it is not feasible to disintegrate $H(a_i) \oplus H(a_j)$ into $H(a_i)$ and $H(a_j)$. Here, $a_{i,j}$ is any variable of length N and $a_i \neq a_j$.*

Proof. Let us assume an adversary adv can actively monitor and alter $\{m_1, m_2, \cdots, m_i\}$ and $\{r_1, r_2, \cdots, r_i\}$, and performs two corner case experiments. □

First, a *zero* string of length N is injected instead of $n_i \oplus K$ to an authentic ED. Therefore, ED receives $m_i = [00...00]$ and calculates $n_i = m_i \oplus K = K$. Then, the ED computes $r_i = H(K) \oplus ID$ and sends r_i to the adversary. Second, all *one* string of length N is passed to the same ED. Similar to the first attempt, ED receives $m_i = [11...11]$ and calculates $n_i = m_i \oplus K = \overline{K}$. Then, the ED computes $r_i = H(\overline{K}) \oplus ID$ and sends r_i to the adversary. From these two transactions, the adversary can construct $H(K) \oplus H(\overline{K})$ which is a more specific case for any replay attack (See Equation (13)).

Considering more general case, adv can retrieve $H(a_i)$ and $H(a_j)$ if and only if $H(a_i) \oplus H(a_j) = 0$; that is, $H(a_i) = H(a_j)$. This is not computationally feasible because of the collision property (see Definition 1) of a secure hash function. Therefore, the protocol π is secured from a polynomial time adversary because it is not feasible to disintegrate $H(a) \oplus H(\overline{a})$ into $H(a)$ and $H(\overline{a})$.

Now, we consider a more pessimistic case where the adversary identifies $H(K)$ (and therefore violates the Theorem 1), yet we show that the protocol is secured by proving the following theorem.

Theorem 2. *If the preimage resistance definition holds, the proposed authentication protocol π is secured against an eavesdropping adversary.*

Proof. Let us assume that a polynomial time adversary runs an algorithm $\mathcal{A}$ that can compute the inverse hash with a probability $negl(N)$. The adversary constructs an efficient algorithm $\mathcal{A}'$ using the algorithm $\mathcal{A}$ as a subroutine (see Figure 3). The new algorithm is entitled "reduction," and it attempts to solve preimage identification problem. This leads to an adversary; compute n_i as follows:

$$n_i \quad = \quad m_i \oplus K' = K \oplus n_i \oplus K' \tag{6}$$

$$\begin{aligned} H_{adv}(n_i) \oplus r_i &= H_{adv}(n_i) \oplus ID \oplus H(n_i) \\ &= ID \end{aligned} \tag{7}$$

Here, $\mathcal{A}'$ computes a shared key K' using $H(K)$. All future authentications can be impersonated using the Equation (7). Even though the protocol chooses a different nonce ($n_j \neq n_i$) during future authentications, it can be retrieved using shared Key K. Revealing K particularly poses a major threat because any future authentication for that particular ED can be passed. This implies a direct contradiction to the preimage resistance property of a secure hash (see Definition 2) because the algorithm $\mathcal{A}$ needs to find K from its hashed value to execute this attack. The protocol π is secured if the preimage resistance property holds. From the above theorems, we conclude that the probability of success for $Auth_{adv,\pi}(n, H_{adv}, m_i, r_i, r_j)$ is negligible. □

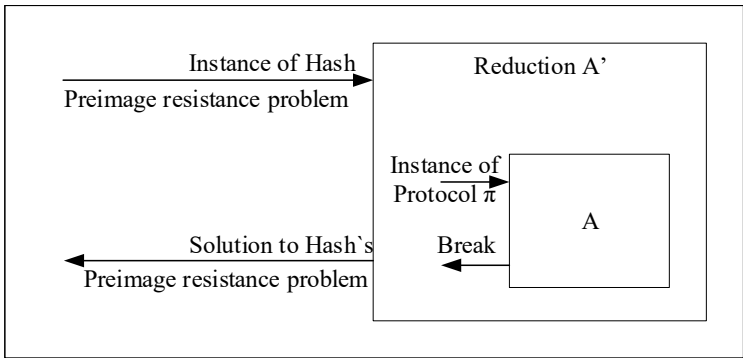

Figure 3. Security proof using reduction method.

4. Device Authentication by ID Matching

Authentication of an ED would be challenging if we consider applying SRAM PUF response as a device signature without any error correction codes. SRAM PUF outputs may vary because of temperature variation, aging degradation, and fluctuations in the supply voltage. Therefore, responses from a PUF could vary significantly if those are taken at different environmental conditions. *How can we use a PUF to verify the identity of a device, if a PUF produces an unreliable ID?* Response collected from a PUF during the registration phase and the response collected during authentication will necessarily match bit-by-bit. To identify a device, however, it is not imperative to match every bit of the stored and new responses. A decision can be reached if the majority of bits (above some predetermined threshold) match among these responses. If the PUF responses are too noisy, an adversary can get lucky to authenticate an illegitimate device. This vulnerability can be addressed by capturing multiple responses from the PUF in a very short duration under similar environmental conditions and perform authentication for once more. To utilize this idea we developed a repeated authentication scheme, where the gateway interrogates an ED more than once in a short duration (see details in Section 4.2). Note that this repeated authentication does not extract more stable bits to identify a device, rather than preventing an adversary to become successful in impersonating an authentic one by random guesses. We show that the probability of impersonating an authentic device twice is negligible.

4.1. Simple ID Matching

The uncertainty in the PUF output is mitigated in the proposed method primarily because the ID matching does not consider the bit-by-bit matching of the IDs during authentication. We propose to apply Hamming distance (HD) to calculate the similarity between stored ID_S and received ID_R. Note that, Hamming distance indicates how many bits are different between two binary numbers.

The authentication can be performed as follows:

$$HD(ID_S, ID_R) \rightarrow \begin{cases} \leq HD_T, \text{ The device is authentic} \\ > HD_T, \text{ The device is counterfeit} \end{cases}$$

where HD_T is the threshold that should be determined once the PUF is characterized. For instance, initially, we can set a threshold of 10 which would mean that as long as stored ID_S and received ID_R are mismatched at most 10 bit, the authentication will pass.

At this point, we determine the probability of impersonating an authentic *ED* by an adversary. Let us assume that the size of the stored ID (PUF response) is of *N*-bits and HD_T is of *k*-bits. Now, the probability of finding one vector with exactly $(N-k)$-bit match or *k*-bit mismatch is as follows:

$$p = \frac{1}{2^N} {}^N C_{(N-k)} = \frac{1}{2^N} \binom{N}{k}$$

Thus, the probability of finding a vector with at most *k*-bit mismatch to pass the ID matching test becomes:

$$p = \frac{1}{2^N} \sum_{i=0}^{k} \binom{N}{i} \tag{8}$$

From Equation (8), we conclude (see Table 1 for details) that the attacker has a low probability of authenticating an illegitimate device.

Table 1. Probability of matching an ID.

	Simple ID Matching		**Repeated ID Matching**									
			$HD_T^{\dagger}$ =1		$HD_T^{\dagger}$=2		$HD_T^{\dagger}$=4		$HD_T^{\dagger}$=8		$HD_T^{\dagger}$=16	
HD_T	ID =128	ID = 256	ID =128	ID = 256	ID =128	ID = 256	ID =128	ID = 256	ID =128	ID = 256	ID =128	ID = 256
1	3.8×10^{-37}	2.2×10^{-75}	2.9×10^{-73}	9.8×10^{-150}	1.8×10^{-71}	1.3×10^{-147}	2.4×10^{-68}	6.7×10^{-144}	3.2×10^{-63}	1.6×10^{-137}	2.1×10^{-55}	3.9×10^{-127}
2	2.4×10^{-35}	2.8×10^{-73}	3.6×10^{-71}	2.5×10^{-147}	2.3×10^{-69}	3.2×10^{-145}	3.0×10^{-66}	1.7×10^{-141}	3.8×10^{-61}	3.9×10^{-135}	2.4×10^{-53}	9.3×10^{-125}
4	3.2×10^{-32}	1.5×10^{-69}	1.9×10^{-67}	5.4×10^{-143}	1.2×10^{-65}	6.8×10^{-141}	1.5×10^{-62}	3.5×10^{-137}	1.8×10^{-57}	7.9×10^{-131}	9.7×10^{-50}	1.8×10^{-120}
8	4.5×10^{-27}	3.7×10^{-63}	4.1×10^{-61}	2.0×10^{-135}	2.5×10^{-59}	2.5×10^{-133}	2.9×10^{-56}	1.3×10^{-129}	3.1×10^{-51}	2.6×10^{-123}	1.2×10^{-43}	5.2×10^{-113}
16	3.2×10^{-19}	9.3×10^{-53}	6.9×10^{-51}	1.3×10^{-122}	3.9×10^{-49}	1.5×10^{-120}	4.0×10^{-46}	7.2×10^{-117}	3.2×10^{-41}	1.3×10^{-110}	7.0×10^{-34}	2.0×10^{-100}
32	6.4×10^{-9}	5.9×10^{-37}	7.9×10^{-36}	4.9×10^{-102}	3.8×10^{-34}	5.5×10^{-100}	2.8×10^{-31}	2.3×10^{-96}	1.2×10^{-26}	3.1×10^{-90}	6.7×10^{-20}	2.6×10^{-80}
64	5.4×10^{-1}	2.4×10^{-16}	1.9×10^{-18}	7.5×10^{-72}	6.0×10^{-17}	7.2×10^{-70}	2.0×10^{-14}	2.2×10^{-66}	1.5×10^{-10}	1.6×10^{-60}	2.1×10^{-5}	3.7×10^{-51}

Although the success probability for adversaries passing authentication is indeed insignificant, this is not sufficient to prevent guessing an ID by random trials when there are many unreliable bits in an ID. This is not unlikely if an *ED* uses an SRAM PUF without any error correction code. Hence, it is essential to further improve the simple matching scheme.

4.2. Repeated ID Matching

It is not likely that an attacker will authenticate a fake device a second time consecutively, unless the communication protocol described in Section 3.1 is broken. This section describes a repeated ID matching scheme, where an *ED* is certified as authentic if it passes two consecutive ID verification tests. In addition, we can extract more stable bits from the PUF response during the second ID matching by discarding unreliable bits. Unreliable bits can be defined as the bits which flip during the first ID matching, are and computed using the following equation:

$$ID_R[i] \neq ID_S[i]$$

where ID_R and ID_S are the received and stored PUF responses, respectively.

The repeated authentication scheme is described in the following steps:

1. The gateway requests l^{th} edge device (ED_l) for its device ID by sending $n_1 \oplus K_l$. The *ED* returns encrypted ID_R ($H(n_1) \oplus ID_R$). The gateway first decrypts the ID (see Equation (4), and then computes the mismatch locations of the received ID ($ID_R[i] \neq ID_S[i]$). A robust ID RID is created by discarding mismatch bits. Additionally, the gateway keeps track of the mismatch locations.

$$RID[k] = ID_R[i], \text{ if } ID_R[i] = ID_S[i];\ 0 < k < i < N \tag{9}$$

2. The gateway again requests ED_l for its ID by sending $n_2 \oplus K_l$, ($n_1 \neq n_2$). Similar to the first authentication, ED_l then returns the encrypted ID_R, ($H(n_2) \oplus ID_R$). This is decrypted in the gateway side using the Equation (4). Then, the gateway computes the new robust ID (RID^*) by using Equation (9).
3. The two robust IDs are compared by using Hamming distance, which is described below:

$$HD(RID, RID^*) \rightarrow \begin{cases} \leq HD_T^\dagger, \text{ The device is authentic} \\ > HD_T^\dagger, \text{ The device is counterfeit} \end{cases}$$

Note that PUF produces a similar response for similar conditions; therefore, $HD_T^\dagger$ is much less than HD_T. It is important to keep in mind that depending on the expected security of the system one can implement an authentication scheme that uses more than two repeated IDs from the same device.

Now, let us calculate the probability of authenticating a device twice by an adversary. We assume that the size of the stored ID is of N-bits, HD_T is of k-bits, and $HD_T^\dagger$ is of r-bits ($r < k$). The probability of passing two repeated authentication becomes:

$$p = \frac{1}{2^N} \sum_{i=0}^{k} \binom{N}{i} \times \frac{1}{2^{(N-k)}} \sum_{i=0}^{r} \binom{N-k}{i} \tag{10}$$

It is clear from the Equation (10) that the probability of successful authentication is significantly reduced if repeated authentications are performed. Successive failed authentication can also be prevented simply by implementing a counter in the gateway to keep track of the failed attempts. If the count crosses some threshold, that can raise a flag.

5. Security Analysis

We provide a detailed security analysis in this section by showing the attack success probabilities and show that the proposed protocol is robust against known attacks, such as denial of service, the replay attack, and different physical attacks.

5.1. Probability Analysis for Proper ID Matching

For a fixed ID length, Hamming distance plays an important role that could reduce the chance of impersonating a device. A smaller Hamming distance makes it harder for an attacker to guess an ID by random trials. However, the reliability of a PUF determines the Hamming distance in this case. Simple ID matching should be enough for a reliable PUF to prevent cloning. However, repeated ID matching can provide the solution for even a very unreliable PUF.

Table 1 lists the probabilities of matching IDs for both a simple ID matching scheme (see Section 4.1) and the proposed repeated ID matching scheme (see Section 4.2). The analysis here considered 128-bit and 256-bit device IDs. The Hamming distances ($HD_T s$) chosen for the analysis are 1, 2, 4, 8, 16, 32, and 64. The PUFs are considered very reliable when HD_T values are 1, 2, and 4. On the other hand, we also consider very noisy PUFs where 32 or 64 bits may be flipped during authentication. As expected, the probabilities of ID matching increase with an increase of HD_T. This is intuitive, as an adversary has a lesser required effort for matching an ID. For a reliable PUF, an attacker has a significantly low probability of matching an authentic ID. For example, the probability of finding a match becomes 4.5×10^{-27} and 3.7×10^{-63} considering an HD_T of 8, when the IDs are 128 bit and 256 bit respectively. However, the probability increases significantly to 5.4×10^{-1} and 2.5×10^{-16} considering HD_T of 64 for an ID of 128 and 256 bits, respectively.

For the repeated ID matching scheme, the Hamming distances ($HD_T^\dagger s$) chosen for the second stage of the authentication process were of 1, 2, 4, 8, and 16 bits. As before, the probability of ID matching is higher with an increase of $HD_T^\dagger s$. However, it is much less when we compare it to the

simple ID matching scheme. We now have a significant improvement of not finding an ID, as the probability has decreased significantly. For example, the probability of finding an ID is 1.9×10^{-18} for a PUF that is heavily impacted by aging (64 out of 128 bits are unstable) assuming stable bits remain stable ($HD_T^{\dagger}$ is of 1). For a 128 bit ID, we also have a very low probability of 1.5×10^{-10} and 2.1×10^{-5} with $HD_T^{\dagger}$ of 8 and 16 bits, respectively. For a 256 bit ID, it is fairly impossible for an adversary to pass the repeated ID matching scheme even though PUF responses are unreliable.

5.2. Denial of Service (DoS) Attack

When an adversary intentionally stages a failed authentication for a genuine *ED* so that it cannot be in service, it is known as denial of service attack. For example, a security camera can be disabled by making it fail to register itself to the server. For the protocol presented in this paper, an attacker can eavesdrop the communication between *ED* and gateway *G*. Every authentication starts with a nonce n_i generation in the *G*, and then n_i is sent l^{th} *ED* as $m_i = K_l \oplus n_i$. In response, the *ED* returns $r_i = H(n_i) \oplus ID_l$ to the *G*. At this point, an attacker can intercept r_i and feed a modified version as r_i^*. This would certainly fail the authentication as *G* will fail to reconstruct the ID_l. To prevent this attack, the proposed protocol needs the following modification:

5.2.1. First Authentication

- *G* generates a random nonce n_i, and sends it as $m_i = K_l \oplus n_i$ to ED_l.
- ED_l returns $r_i = H(n_i) \oplus ID_l$ to the gateway as response. Attacker eavesdrops and replaces r_i with $r_i^* (\neq r_i)$.
- *G* retrieves $ID_l^{(1)}$ as the ID of ED_l by computing $ID_l^{(1)} = H(n_i) \oplus r_i^*$. Since the $ID_l \neq ID_l^{(1)}$, the authentication fails.

5.2.2. Second Authentication

- G generates a random number $n_j (\neq n_i)$ and sends $m_j = K_l \oplus n_j$ to ED_l.
- ED_l returns $r_j = H(n_j) \oplus ID_l$ back to the gateway. Similarly to the first authentication phase, an attacker may eavesdrop and send $r_j^* (\neq r_j)$ to the gateway.
- Gateway retrieves $ID_l^{(2)} \neq ID_l$ as the ID of the *ED* is ID_l and authentication fails.
- Finally, gateway verifies the $ID_l^{(1)}$ and $ID_l^{(2)}$ to determine whether an attack has been launched.

According to the Table 1, the probability of matching these two IDs is very low.

5.3. Replay Attack

An attacker attempts to authenticate an *ED* by impersonating the ID using prior communications. We assume that an attacker does not have access to the secret key (K_l), which is stored in the ED_l. Let us assume that the attacker observes two prior communications. First, he/she observes $n_1 \oplus K_l$ from the gateway and $H_{n_1} \oplus ID_l$ from ED_l. From this observation the attacker can compute $K_l \oplus ID_l \oplus n_1 \oplus H_{n_1}$, which is shown below:

$$(n_1 \oplus K_l) \oplus (H_{n_1} \oplus ID_l) \tag{11}$$

From the second communication, the attacker observes $n_2 (\neq n_1) \oplus K_l$ from the gateway and $H_{n_2} \oplus ID_l$ from ED_l, and can compute $K_l \oplus ID_l \oplus n_2 \oplus H_{n_2}$.

Now the attacker can perform the following operations:

$$(n_1 \oplus K_l) \oplus (n_2 \oplus K_l) = n_1 \oplus n_2 \tag{12}$$

$$(H_{n_1} \oplus ID_l) \oplus (H_{n_2} \oplus ID_l) = H_{n_1} \oplus H_{n_2} \tag{13}$$

From Equation (13), it is obvious that an adversary successfully replays a prior communication if it becomes zero, when $H_{n_1} = H_{n_2}$. This would certainly contradict the collision property of a secure hash [30]. Thus, the communication protocol becomes is resistant to replay attack, when it has been implemented with a secure hash function (SHA-2 or SHA-3).

5.4. Physical Attacks

We need to use a tamper-proof memory to store the secret key K_l in the *ED*. As discussed previously K_l is shared between the gateway and *ED*, and so a breach in this information will break the security of the protocol. Reverse engineering and physical attack can be a way to steal this secret information. Nowadays, sophisticated optical microscopes can capture high-resolution 3D images of a microchip. Scanning electron microscopes and transmission electron microscopes can produce images of different inner layers of a microchip. Chipworks (now TechInsights) have performed such experiments legitimately for competitive analysis and patent research. The physical layout of a chip can be constructed from a legitimate chip through destructive physical attacks as well. Infrared backside imaging can reveal stored data in an NVM. All these physical attacks can be used to reveal secret information. Since our proposed protocol uses PUF for ID generation, this ID will be different for different *EDs*. Performing a physical attack will expose the unique key K_l, and thus the PUF generated ID of ED_l only and not any other devices. From the financial point of view, this does not provide a strong motivation for an adversary to launch physical attacks.

6. Implementation Details

In this section, we show that the proposed protocol can be efficiently implemented with low code overhead using commercial resource-constrained devices. Arduino Mega [66] and Raspberry Pi-3 model B (RPi) [67] have been used as the *ED* and gateway, respectively. Note that Arduino Mega is equipped with an ATMEGA2560 microcontroller [68] that contains 8KB SRAM and 256KB flash storage. We exploit the uninitialized power-up states of this SRAM to generate device signature (i.e., ID). Figure 4 shows the implementation setup. The proposed protocol is implemented into two phases, enrollment phase and authentication phase, which are described in the following.

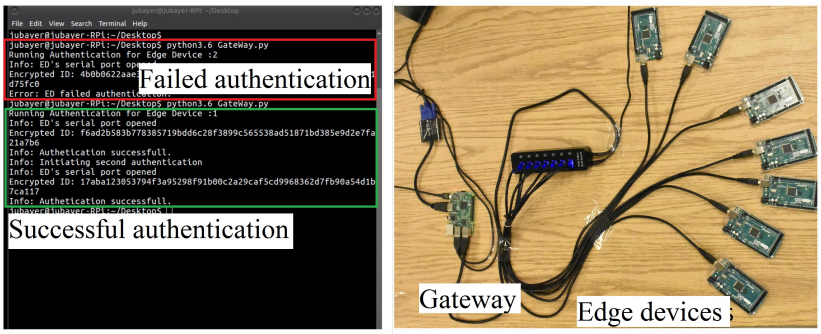

Figure 4. Implementation setup with Arduino Mega 2560 as the ED and Raspberry Pi as the gateway.

6.1. Enrollment Phase

The objective of the enrollment phase is to read the unclonable device ID of an *ED* and store it in a secure database for future authentication. The uninitialized memory space between the stack pointer (*SP*) and the heap pointer (*HP*) is extracted to generate device ID. Figure 5 illustrates the memory space for a typical microcontroller. Note that the memory space between *HP* and *SP* shrinks or expands depending on the workload of a particular firmware. However, we can capture initial *SP* during the program start-up and keep track of the changes to get the SRAM data from the preset address range every time the ID is constructed. We present ID extraction and setup in Algorithm 1. The ID extraction flow and all data processing in the enrollment phase are written in Python, whereas the firmware for

the ID extraction is written in C. All the operations involved in this implementation use 256-bit data. Algorithm 1 lists the steps to extract the signature of a device and is described as follows:

Algorithm 1: ID extraction of an *ED* and enrollment phase.

Input : ID length
Output : Unstable SRAM cell locations and device ID

1 Read initial SRAM state $M \leftarrow fullRAMSpace$;
2 Read available SRAM memory state with authentication and user subroutines $M' \leftarrow useableIDSpace$;
3 $[L, H] \leftarrow longestMatchingString(M, M')$;
4 Compute $IDStart \approx \frac{L+H}{2\times 8}$;
5 **for** *index* $i = 1$; $i \leq N + e$; $i = i + 1$ **do**
6 $\quad id'[i] \leftarrow usableIDSpace(IDStart, IDStart + (e - 1))$
7 **end**
8 $usCells \leftarrow unstableCellLocation()$;
9 $j = 0$;
10 **while** $(i <= N)\&\&(i\,! = usCells[j])$ **do**
11 $\quad ID[j] \leftarrow id'[1 : end]$;
12 $\quad j + +$;
13 **end**
14 Load *usCell*, *IDStart*, *e*, shared key *K* in *ED*;
15 Store $\{K, ID\}$ in database ;

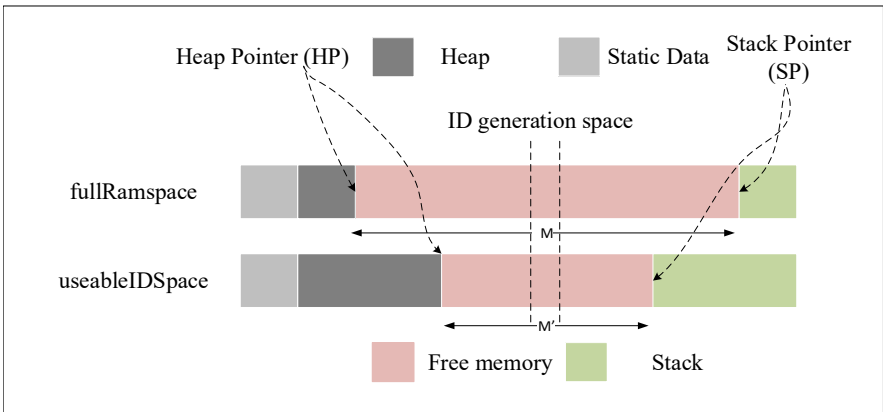

Figure 5. Typical memory map of a microcontroller.

1. A host computer starts executing Algorithm 1. *fullRAMSpace* firmware is loaded in the microcontroller, and it reads the available uninitialized memory between *HP* and *SP*. The uninitialized memory of the SRAM is shifted out through the serial port and captured on the host computer. These data are stored in a variable *M* (Line 1). Note that *fullRAMSpace* firmware contains only subroutines that extract SRAM data and communicates with the host computer.
2. Authentication subroutine and user application subroutines, data acquisition, monitoring, etc., are combined in *usableIDSpace* firmware and loaded into the microcontroller. Similarly to Step 1, the available uninitialized SRAM data are shifted out and stored in a variable M' in the host computer (Line 2).
3. *longestMatchingString(M,M')* is a string matching algorithm implemented in Python (running in the host computer) that returns the longest common string and its indices. For example, *fullRAMSpace* firmware retrieves 7176 bits from *M*, and *usableIDSpace* firmware retrieves 6312 bits from M'. The string matching algorithm returns the lowest index, *L*, and highest index, *H* of the longest matched string (Line 3). For this specific example, a matched string starts from $M'[4144]$ to $M'[5819]$ (total 209 bytes). Since SRAM has inherently unstable bits, for 256-bit (32 bytes) ID we take $(32 + e)$ bytes from an approximately equal distance away from $M'[L]$ and $M'[H]$.

Here, e must be at least as large as the number of possible unstable bits in the address range selected for ID generation. Memory space for ID generation has been selected approximately in the middle of the matched string so that the processor does not read initialized values from the stack and static data segment during ID extraction. This is important because, during the execution of the user program, *HP* and *SP* grow toward each other and may interfere with the ID generation space. *IDStart* parameter changes depending on the relative position of *HP* and *SP* for each device. Line 4 calculates the starting index of the ID as $\frac{L+H}{2\times 8}$. Here, 8 in the denominator indicates the bit width of the memory. Address range for ID generation will be $M[IDStart]$ to $M[IDStart + (e-1)]$.

4. Once the range $M[IDStart]$ to $M[IDStart + (e-1)]$ is known with an initial value of e, $unstableCellLocation()$ function is loaded in the *ED*, and unstable bits are identified with hardware support (e.g., power-up circuitry). $unstableCellLocation()$ function returns the location of unstable bits and stable bits of the ID' (Line 5–7). These unstable bit locations are stored in the MCU as an array *usCells* (Line 8). For this experiment, we selected $e = 10$ which initially assumes 76.2% stable uninitialized bits of the SRAM [69]. This parameter will vary based on the implementation platform, which can be selected with a few trials of power-ups or data reminiscence approach.
5. The ID is extracted discarding the unstable bits in Lines 10–12. Then, *usCell*, *IDStart*, e, and shared key K are loaded into the MCU. Finally, the device ID is stored in the database for future authentication.

The approach described above makes each *ED* unique because of the inherent randomness in the power-up state of SRAM, inter-device variability of *IDStart*, and *usCell* array.

6.2. Authentication Phase

Authentication phase executes the protocol (Figure 2) in the gateway and edge devices connected to it.

1. In this phase, gateway G starts authentication by sending a token to the *ED*. Since the token is specific (and public) to an *ED*, only a particular *ED* will respond to the authentication request from the G. Along with the token G also sends a nonce n_i encrypted with shared key K.
2. At *ED*, n_i is retrieved and converted into a hash using the SHA-2 algorithm. Then, *ED* runs the ID extraction subroutine that constructs a 256-bit stable ID from SRAM using address range preset during the enrollment phase. ID and $SHA-2(n_i)$ strings are processed to get XOR'ed value r_i.
3. The encrypted ID retrieved from ED (r_i) is decrypted in the G as $ID' = SHA-2(n_i)) \oplus r_i$ and compared with a stored ID. A successful first authentication would trigger another authentication attempt as described in Section 4.2.

6.3. Overhead Analysis

As it is important to have low area and timing overhead in an IoT device, we present analysis in this subsection. The overall code overhead due to the authentication function is 3.65%. The overhead primarily comes from the hash algorithm and string processing. The hardware overhead is only from the memory requirement for storing the symmetric key (K), which is 256-bit in this implementation. Note that no additional hardware is needed to implement the PUF. Considering the importance of security in the IoT infrastructure, this overhead is quite insignificant. Note that no hardware modifications are necessary for *EDs*, which makes our solution feasible for adopting various low-cost applications. Note that the timing overhead will not throttle the performance of the IoT devices, as the proposed protocol is only used for authentication of an *ED* after the power-up of a device or at a large regular time interval. The implementation presented in the paper uses approximately a million cycles in *ATMEGA2560* running at 16MHz for each authentication, which is practically within a few tens of milliseconds and can be ignored.

6.4. Reliability Analysis

Since ID generation would require randomness in the power-up states of SRAMs among different devices, we need to analyze the reliability of our proposed ID generation scheme. We analyzed the power-up states of built-in SRAM to show the probability of the authentication of a random device as genuine is rather negligible. The uniqueness and randomness of an SRAM-PUF are assured by the normal distribution of fractional inter-class Hamming distance (*HD*) [69] with a mean at 40.2%. We choose the minimum value of *HD* (HD_{min}) to calculate the collision probability of *IDs* between two different devices. For normal distribution, confidence interval (*CI*) for mean μ is of the form $l \leq \mu \leq u$, where u and l are upper and lower confidence limits respectively. The limits are calculated as $\bar{x} \pm z_{\alpha/2} \times \frac{\sigma}{\sqrt{s}}$, where $1 - \alpha$ is the confidence coefficient of a standard normal distribution [70]. Considering lowest limit of the mean μ, the probability of matching two IDs becomes $\frac{2^{N*(1-l)}-1}{2^N-1}$. For example, Figure 6 shows the distribution of HD_{min} with sample mean $\bar{x} = 0.3436$ and standard deviation $\sigma = 0.024$ with $s = 500$ samples. We can estimate that 99% CI is $0.3409 \leq \mu \leq 0.3464$ for 256-bit ID, and the probability of generating two devices with a same ID is $\frac{2^{256(1-0.3464)}-1}{2^{256}-1} \approx 6.46 \times 10^{-27}$, which is insignificant.

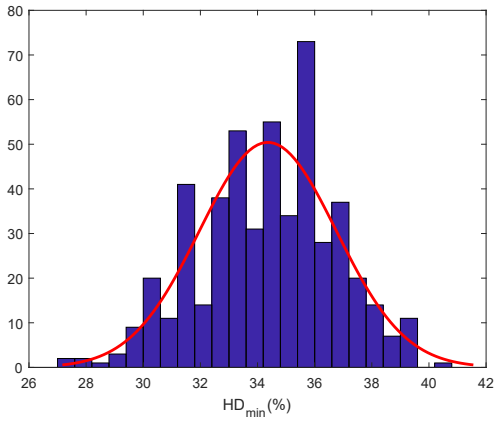

Figure 6. Distribution of fractional Hamming (minimum) distance for 500 IDs.

7. Conclusions

In this paper, a novel, low-cost authentication protocol has been proposed. Production cost reduction without sacrificing security for IoT applications is an extremely difficult task. In addition, a standard cryptographic scheme's implementation in *ED* is prohibitively expensive because of the limited resources (e.g., energy) available during operation. We presented a low-cost protocol that uses a secure hash function to authenticate an edge device in an IoT network. The protocol is designed to utilize already available on-chip resources, which fits it in the domain of low-cost applications, such as IoT. We exploited built-in SRAM to generate unclonable "digital fingerprint" of an IoT edge device and used firmware to extract the fingerprint. Aging degradation and temperature variation make it challenging to implement reliable SRAM-PUF. Our proposed repeated ID matching scheme that utilizes Hamming distance eliminates this inherent PUF unreliability. Thorough security analysis shows that the probability of ID impersonation by an attacker is extremely low. The protocol is implemented in a commercial microcontroller, and the overhead of implementation is only a small part of the memory for the firmware.

Author Contributions: Conceptualization, methodology, writing—original draft preparation, writing—review and editing, J.M. and U.G.; supervision, project administration, funding acquisition, U.G. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Science Foundation under grant number CNS-1755733 and Intramural Grants Program (IGP) from Auburn University. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

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

Abbreviations

The manuscript uses the following abbreviations:

ED	Edge device
G	Gateway
ECC	Elliptic-curve cryptography
RSA	Rivest–Shamir–Adleman
IoT	Internet of Things
MCU	Microcontroller
PUF	Physically unclonable function
SRAM	Static random-access memory
NVM	Non-volatile memory
CRP	Challenge response pair
HD	Hamming distance
MITM	Man-in-the-middle
IBE	Identity-based encryption
CSPRNG	Cryptographically secure pseudo-random number generator
MAC	Message authentication code
OTP	One-time-pad
DoS	Denial of service

References

1. Fredrik Dahlqvist, Mark Patel, A.R.; Shulman, J. Growing opportunities in the Internet of Things, 2019. Available online: https://www.mckinsey.com/industries/private-equity-and-principal-investors/our-insights/growing-opportunities-in-the-internet-of-things (accessed on 13 October 2019).
2. Evans, D. The Internet of Things: How the Next Evolution of the Internet Is Changing Everything, 2011. Available online: https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf (accessed on 13 October 2019).
3. Research, G. Gartner Says 6.4 Billion Connected "Things" Will Be in Use in 2016, Up 30 Percent From 2015. 2015. Available online: https://www.gartner.com/en/newsroom/press-releases/2015-11-10-gartner-says-6-billion-connected-things-will-be-in-use-in-2016-up-30-percent-from-2015 (accessed on 13 October 2019).
4. Trappe, W.; Howard, R.; Moore, R.S. Low-energy security: Limits and opportunities in the internet of things. *IEEE Secur. Priv.* **2015**, *13*, 14–21. [CrossRef]
5. Rawlinson, K. HP Study Reveals 70 Percent of Internet of Things Devices Vulnerable to Attack. 2015. Available online: https://www8.hp.com/us/en/hp-news/press-release.html?id=1744676 (accessed on 13 October 2019).
6. Robertson, J.; Riley, M. The Big Hack: How China Used a Tiny Chip to Infiltrate U.S. Companies. 2018. Available online: https://iot.eetimes.com/copycats-pose-a-serious-security-threat-to-the-iot/ (accessed on 13 October 2019).
7. Guin, U.; Asadizanjani, N.; Tehranipoor, M. Standards for Hardware Security. *GetMobile: Mobile Comp. Comm.* **2019**, *23*, 5–9. [CrossRef]
8. Valerio, P. BorderHawk found counterfeit IoT devices installed. 2018. Available online: https://iot.eetimes.com/copycats-pose-a-serious-security-threat-to-the-iot/ (accessed on 13 October 2019).
9. Tehranipoor, M.M.; Guin, U.; Bhunia, S. Invasion of the hardware snatchers. *IEEE Spectr.* **2017**, *54*, 36–41. [CrossRef]
10. Cui, P.; Guin, U. Countering Botnet of Things using Blockchain-Based Authenticity Framework. In Proceedings of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI), Miami, FL, USA, 15–17 July 2019.

11. Cyr, B.; Mahmod, J.; Guin, U. Low-Cost and Secure Firmware Obfuscation Method for Protecting Electronic Systems From Cloning. *IEEE Internet Things J.* **2019**, *6*, 3700–3711. [CrossRef]
12. Guin, U.; Cui, P.; Skjellum, A. Ensuring Proof-of-Authenticity of IoT Edge Devices using Blockchain Technology. In Proceedings of the IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Halifax, NS, Canada, 30 July–3 August 2018.
13. Tehranipoor, M.M.; Guin, U.; Forte, D. *Counterfeit Integrated Circuits: Detection and Avoidance*; Springer: Berlin, Germany, 2015.
14. The Federal Bureau of Investigation. Departments of Justice and Homeland Security Announce 30 Convictions, More Than $143 Million in Seizures from Initiative Targeting Traffickers in Counterfeit Network Hardware. 2010. Available online: https://www.justice.gov/opa/pr/departments-justice-and-homeland-security-announce-30-convictions-more-143-million-seizures (accessed on 13 October 2019).
15. Guin, U.; Bhunia, S.; Forte, D.; Tehranipoor, M. SMA: A System-Level Mutual Authentication for Protecting Electronic Hardware and Firmware. *IEEE Trans. Dependable Secure Comput.* **2016**. [CrossRef]
16. Guin, U.; Shi, Q.; Forte, D.; Tehranipoor, M. FORTIS: A Comprehensive Solution for Establishing Forward Trust for Protecting IPs and ICs. *ACM Trans. Des. Autom. Electron. Syst.* **2016**, *21(4)*, 1–20. [CrossRef]
17. Borgohain, T.; Kumar, U.; Sanyal, S. Survey of Security and Privacy Issues of Internet of Things. *arXiv* **2015**, arXiv:1501.02211. Available online: https://arxiv.org/ftp/arxiv/papers/1501/1501.02211.pdf (accessed on 13 October 2019).
18. He, H.; Maple, C.; Watson, T.; Tiwari, A.; Mehnen, J.; Jin, Y.; Gabrys, B. The security challenges in the IoT enabled cyber-physical systems and opportunities for evolutionary computing other computational intelligence. In Proceedings of the 2016 IEEE Congress on Evolutionary Computation (CEC), Vancouver, BC, Canada, 24–29 July 2016; pp. 1015–1021.
19. Bryzek, J. Roadmap for the trillion sensor universe. 2013. Available online: https://myukk.org/SM2017/sm_pdf/SM1534.pdf (accessed on 13 October 2019).
20. Locke, G.; Gallagher, P. Fips pub 186-3: Digital signature standard (dss). 2009. Available online: https://csrc.nist.gov/csrc/media/publications/fips/186/3/archive/2009-06-25/documents/fips_186-3.pdf (accessed on 13 October 2019).
21. Schmitt, C.; Kothmayr, T.; Hu, W.; Stiller, B. Two-way authentication for the internet-of-things. In *Internet of Things: Novel Advances and Envisioned Applications*; Acharjya, D., Geetha, M., Eds.; Springer: Cham, Switzerland, 2017; Volume 25, pp. 27–56.
22. Wallrabenstein, J.R. Practical and secure iot device authentication using physical unclonable functions. In Proceedings of the 4th International Conference on Future Internet of Things and Cloud (FiCloud), Vienna, Austria, 22–24 August 2016.
23. Roman, R.; Alcaraz, C.; Lopez, J. A survey of cryptographic primitives and implementations for hardware-constrained sensor network nodes. *Mobile Network. Appl.* **2007**, *12(4)*, pp. 231–244. [CrossRef]
24. Gassend, B.; Clarke, D.; Van Dijk, M.; Devadas, S. Silicon physical random functions. In Proceedings of the ACM Conference on Computer and Communications Security (CCS), Washington, DC, USA, 18–22 November 2002; pp. 148–160.
25. Suh, G.; Devadas, S. Physical Unclonable Functions for device authentication and secret key generation. In Proceedings of the 44th ACM/IEEE Design Automation Conference, San Diego, CA, USA, 4–8 June 2007; pp. 9–14.
26. Guajardo, J.; Kumar, S.S.; Schrijen, G.J.; Tuyls, P. FPGA intrinsic PUFs and their use for IP protection. In *Cryptographic Hardware and Embedded Systems-CHES 2007;* Paillier, P., Verbauwhede, I., Eds.; Springer: Berlin, Germany, 2007; pp. 63–80.
27. Kumar, S.S.; Guajardo, J.; Maes, R.; Schrijen, G.J.; Tuyls, P. The butterfly PUF protecting IP on every FPGA. In Proceedings of the IEEE International Workshop on Hardware-Oriented Security and Trust, Anaheim, CA, USA, 9 June 2008; pp. 67–70.

28. Iyengar, A.; Ramclam, K.; Ghosh, S. DWM-PUF: A low-overhead, memory-based security primitive. In Proceedings of the IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), Arlington, VA, USA, 6–7 May 2014; pp. 154–159.
29. Sutar, S.; Raha, A.; Raghunathan, V. D-PUF: An intrinsically reconfigurable DRAM PUF for device authentication in embedded systems. In Proceedings of the 2016 International Conference on Compliers, Architectures, and Sythesis of Embedded Systems (CASES), Pittsburgh, PA, USA, 2–7 October 2016; pp. 1–10.
30. NIST. FIPS PUB 180-4: Secure Hash Standard (SHS). 2015. Available online: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=910977 (accessed on 13 October 2019).
31. Eisenbarth, T.; Heyse, S.; von Maurich, I.; Poeppelmann, T.; Rave, J.; Reuber, C.; Wild, A. Evaluation of SHA-3 Candidates for 8-bit Embedded Processors. 2010. Available online: http://math.fau.edu/~eisenbarth/pdf/SHA3.pdf (accessed on 7 June 2019).
32. Atmel AVR232: Authentication Using SHA-256. 2012. Available online: http://ww1.microchip.com/downloads/en/AppNotes/doc8184.pdf (accessed on 6 December 2019).
33. Hammouri, G.; Sunar, B. PUF-HB: A tamper-resilient HB based authentication protocol. In *Applied Cryptography and Network Security*; Bellovin, S.M., Gennaro, R., Keromytis, A., Yung, M., Eds.; Springer: Berlin, Germany, 2008; Volume 5037, pp. 346–365.
34. Hopper, N.J.; Blum, M. Secure human identification protocols. In *Advances in Cryptology–ASIACRYPT 2001;* Boyd, C., Eds.; Springer: Berlin, Germany, 2001; Volume 2248, pp. 52–66.
35. Hammouri, G.; Öztürk, E.; Sunar, B. A tamper-proof and lightweight authentication scheme. *Pervasive Mob. Comput.* **2008**, *4*. 807–818. [CrossRef]
36. Katzenbeisser, S.; Kocabaş, Ü.; Van Der Leest, V.; Sadeghi, A.R.; Schrijen, G.J.; Wachsmann, C. Recyclable pufs: Logically reconfigurable pufs. *J. Cryptogr. Eng.* **2011**, *1*, 177. [CrossRef]
37. Rührmair, U.; van Dijk, M. PUFs in security protocols: Attack models and security evaluations. In Proceedings of the 2013 IEEE Symposium on Security and Privacy, Berkeley, CA, USA, 19–22 May 2013; pp. 286–300.
38. 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.
39. Kocabaş, Ü.; Peter, A.; Katzenbeisser, S.; Sadeghi, A.R. Converse PUF-based authentication. In *Trust and Trustworthy Computing;* Katzenbeisser, S., Weippl, E., Camp, L.J., Volkamer, M., Reiter, M., Zhang, X., Eds.; Springer: Berlin, Germany, 2012; Volume 7344, pp. 142–158.
40. Hossain, M.; Noor, S.; Hasan, R. HSC-IoT: A Hardware and Software Co-Verification based Authentication Scheme for Internet of Things. In Proceedings of the 5th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud), San Francisco, CA, USA, 6–8 April 2017; pp. 109–116.
41. Chatterjee, U.; Govindan, V.; Sadhukhan, R.; Mukhopadhyay, D.; Chakraborty, R.S.; Mahata, D.; Prabhu, M.M. Building PUF based Authentication and Key Exchange Protocol for IoT without Explicit CRPs in Verifier Database. *IEEE Trans. Dependable Secure Comput.* **2018**, *16*, 424–437. [CrossRef]
42. Braeken, A. PUF Based Authentication Protocol for IoT. *Symmetry* **2018**, *10*, 352. [CrossRef]
43. Dolev, D.; Yao, A. On the security of public key protocols. *IEEE Trans. Inform. Theor.* **1983**, *29*, 198–208. [CrossRef]
44. Garcia-Morchon, O.; Keoh, S.L.; Kumar, S.; Moreno-Sanchez, P.; Vidal-Meca, F.; Ziegeldorf, J.H. Securing the IP-based internet of things with HIP and DTLS. In Proceedings of the sixth ACM conference on Security and privacy in wireless and mobile networks, New York, NY, USA, 17 April 2013.
45. Kothmayr, T.; Schmitt, C.; Hu, W.; Brünig, M.; Carle, G. DTLS based security and two-way authentication for the Internet of Things. *Ad Hoc Netw.* **2013**, *11*, 2710–2723. [CrossRef]
46. Porambage, P.; Schmitt, C.; Kumar, P.; Gurtov, A.; Ylianttila, M. Two-phase authentication protocol for wireless sensor networks in distributed IoT applications. In Proceedings of the Wireless Communications and Networking Conference (WCNC), Istanbul, Turkey, 6–9 April 2014; pp. 2728–2733.
47. Challa, S.; Wazid, M.; Das, A.K.; Kumar, N.; Reddy, A.G.; Yoon, E.J.; Yoo, K.Y. Secure signature-based authenticated key establishment scheme for future iot applications. *IEEE Access* **2017**, *5*, 3028–3043. [CrossRef]

48. Turkanović, M.; Brumen, B.; Hölbl, M. A novel user authentication and key agreement scheme for heterogeneous ad hoc wireless sensor networks, based on the Internet of Things notion. *Ad Hoc Netw.* **2014**, *20*, 96–112. [CrossRef]
49. Song, T.; Li, R.; Mei, B.; Yu, J.; Xing, X.; Cheng, X. A privacy preserving communication protocol for IoT applications in smart homes. *IEEE Internet Things J.* **2017**, *4*, 1844–1852. [CrossRef]
50. Wazid, M.; Das, A.K.; Odelu, V.; Kumar, N.; Conti, M.; Jo, M. Design of Secure User Authenticated Key Management Protocol for Generic IoT Networks. *IEEE Internet Things J.* **2018**, *5*, pp. 269–282. [CrossRef]
51. Ishtiaq Roufa, R.M.; Mustafaa, H.; Travis Taylora, S.O.; Xua, W.; Gruteserb, M.; Trappeb, W.; Seskarb, I. Security and privacy vulnerabilities of in-car wireless networks: A tire pressure monitoring system case study. In SENIX Security Symposium; Washington DC, 2010, pp. 11–13.
52. Barcena, M.B.; Wueest, C. Insecurity in the Internet of Things. Available online: https://www.researchgate.net/profile/Hadeel_Saleh_Haj_Aliwi/post/What_are_the_best_papers_in_IoT_Security/attachment/59dda4b44cde260ad3cea425/AS:548138643853312@1507697844002/download/paper1.pdf (accessed on 5 June 2019).
53. Holcomb, D.E.; Burleson, W.P.; Fu, K. Initial SRAM state as a fingerprint and source of true random numbers for RFID tags. 2007.
54. Sunar, B.; Martin, W.; Stinson, D. A Provably Secure True Random Number Generator with Built-In Tolerance to Active Attacks. *IEEE Trans. Comput.* **2007**, *56*, 109–119. [CrossRef]
55. Vernam, G.S. Secret signaling system. 1919. Available online: https://patents.google.com/patent/US1325574A/en (accessed on 12 July 2018).
56. Katz, J.; Lindell, Y. *Introduction to Modern Cryptography*; CRC Press: Boca Raton, FL, USA, 2014.
57. Guin, U.; Singh, A.; Alam, M.; Canedo, J.; Skjellum, A. A Secure Low-Cost Edge Device Authentication Scheme for the Internet of Things. In Proceedings of the 31st International Conference on VLSI Design and 17th International Conference on Embedded Systems (VLSID), Pune, India, 6–10 January 2018.
58. Dierks, T. The transport layer security (TLS) protocol version 1.2. 2008. Available online: https://tools.ietf.org/pdf/rfc8446.pdf (accessed on 10 September 2019).
59. Holcomb, D.E.; Burleson, W.P.; Fu, K. Power-up SRAM state as an identifying fingerprint and source of true random numbers. *IEEE Trans. Comput.* **2008**, *58*, 1198–1210. [CrossRef]
60. Katzenbeisser, S.; Kocabaş, Ü.; Rožić, V.; Sadeghi, A.; 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*; Prouff, E., Schaumont, P. Eds.; Springer: Berlin, Germany, 2012; pp. 283–301.
61. Barbareschi, M.; Battista, E.; Mazzeo, A.; Mazzocca, N. Testing 90 nm microcontroller SRAM PUF quality. In Proceedings of the 10th International Conference on Design Technology of Integrated Systems in Nanoscale Era (DTIS), Naples, Italy, 21–23 April 2015; pp. 1–6.
62. Maes, R.; Rozic, V.; Verbauwhede, I.; Koeberl, P.; van der Sluis, E.; van der Leest, V. Experimental evaluation of Physically Unclonable Functions in 65 nm CMOS. In Proceedings of the ESSCIRC (ESSCIRC), Bordeaux, France, 17–21 September 2012; pp. 486–489.
63. Hosey, A.; Rahman, M.T.; Xiao, K.; Forte, D.; Tehranipoor, M. Advanced Analysis of Cell Stability for Reliable SRAM PUFs. In Proceedings of the 23rd Asian Test Symposium, Hangzhou, China, 16–19 November 2014; pp. 348–353.
64. Comon, H.; Shmatikov, V. Is it possible to decide whether a cryptographic protocol is secure or not? *J. Inf. Syst. Telecommun.* **2002**, *4*, 5–15.
65. Paar, C.; Pelzl, J. *Understanding Cryptography: A Textbook for Students and Practitioners*; Springer Science & Business Media: Berlin, Germany, 2009.
66. ARDUINO MEGA 2560 REV3. Bloomberg. 2019. Available online: https://store.arduino.cc/usa/mega-2560-r3 (accessed on 10 December 2019).
67. Raspberry PI 3 Model B, 2019. Available online: https://www.terraelectronica.ru/pdf/show?pdf_file=%252Fds%252Fpdf%252FT%252FTechicRP3.pdf (accessed on 15 December 2019).
68. Atmel Corporation. 8-bit Atmel Microcontroller with 16/32/64KB In-System Programmable Flash. 2014. Available online: https://media.digikey.com/pdf/Data%20Sheets/Atmel%20PDFs/ATmega164,324,644P,V%20Rev2010.pdf (accessed on 1 January 2020).

69. Platonov, M.; Hlavác, J.; Lórencz, R. Using Power-Up SRAM State of Atmel ATmega1284P Microcontrollers as Physical Unclonable Function for Key Generation and Chip Identification. *Inf. Secur. J. A Global Perspective* **2013**, *22*, 244–250. [CrossRef]
70. Montgomery, D.C.; Runger, G.C. *Applied Statistics and Probability for Engineers*; John Wiley & Sons: New York, NY, USA, 2010.

© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).