Efficient and Secure Group Key Management in IoT using Multistage Interconnected PUF

ABSTRACT

Secure group-oriented communication is crucial to a wide range of applications in Internet of Things (IoT). Security problems related to group-oriented communications in IoT-based applications placed in a privacy-sensitive environment have become a major concern along with the development of the technology. Unfortunately, many IoT devices are designed to be portable and light-weight; thus, their functionalities, including security modules, are heavily constrained by the limited energy resources (e.g., battery capacity). To address these problems, we propose a group key management scheme based on a novel PUF design: multistage interconnected physically unclonable function (MIPUF) to secure group communications in an energy-constrained environment. Our design is capable of performing key management tasks such as key distribution, key storage and rekeying securely and efficiently. We show that our design is secure against multiple attack methods and our experimental results show that our design saves 47.33% of global energy comparing to state-of-the-art Elliptic-curve cryptography (ECC)-based key management scheme on average.

1. INTRODUCTION

Internet of Things (IoT) has been envisioned to be a revolutionary network that connects physical devices around us to perform intelligent tasks such as monitoring, communication, operation, and optimization. The advancement in IoT technology has enabled a wide spectrum of applications in a variety of environments, including but not limited to homes, factories, hospitals or city streets. While IoT technology has greatly improved the efficiency and quality of our lives and works, various security challenges have become a major concern and doubt for further adoption of the technology. Security improvement in IoT system has become an increasingly popular topic in both academia and industry due to its urgency and profitability. In this paper, we are particularly interested in efficient and secure key management schemes in group communications in an IoT setting.

Group communication through multicast/broadcast enables direct communication with the whole group, which is more efficient when compared to an equivalent unicast-based solution. Securing group communications consists of providing confidentiality, authenticity, and integrity of messages exchanged within the group [1]. Among all security problems in IoT, group key management is one of the fundamentals in securing group communications. A group key essentially is a secret key shared by all members of a group so that all group communication packages are encrypted before they are being transmitted using this group key. An unauthorized user may receive group communication packages due to network error or intentional interception, however, without the right group key, the illegal user cannot decrypt the received packages.

Group key management schemes in IP networks, though have been studied for decades, cannot be directly applied to IoT as IoT devices are heavily constrained by the limited resource and energy capacity. Limited resources impose new challenges regarding storage and computation requirements, meaning each node is incapable of storing a large key database or conduct heavy cryptographic computation. The energy constraint additionally requires key verification and computation procedures to be energy efficient.

For the above two reasons, physically unclonable functions (PUFs), a type of low-power security primitive with unclonable and unpredictable properties, naturally appears as an ideal solution to the problem. In this paper, we propose to apply a novel low power PUF structure called Multistage Interconnected PUF (MIPUF) to the domain of group key management in IoT. We believe the low power and unclonable, unpredictable nature of MIPUF not only improves the security of group key management protocols but also meet the tighter energy requirements on IoT nodes. Our design of interconnection reconfiguration in MIPUF is robust and secure against modeling attacks by changing the challenge-response mapping. The group key is stored and managed by a new set of PUF functions every time we reconfigure the MIPUF in every IoT device, creating an additional layer of security and protection. We also show that our key management scheme including key distribution, key storage and rekeying is resilient against a wide range of attacks. Lastly, we show that our group key management protocol is power and energy efficient. Our simulation results show that we are 47.33% more energy efficient comparing to state-of-the-art ECC-based key management schemes.

2. RELATED WORK

Several efforts have been made in creating efficient group key management protocols for group communications in IoT and wireless sensor networks (WSN) to meet the energy and computation constraints. Notably, Zhu et al. proposed an efficient security mechanism for large-scale distributed sensor networks [2]. Roman et al. analyzed the applicability of public-key cryptography based protocols and link-layer oriented key management systems (KMS) in IoT settings [3]. Abdallah et al. proposed a novel efficient and scalable key management mechanism for wireless sensor networks and proposed to reduce power and energy consumption by using ECC [4]. All work listed above utilizes expensive cryptographic primitives to secure their group key management protocols without investigating the possibility of utilizing some novel low-power hardware security primitives to meet the energy requirements.

Recently, PUFs, as a popular type of low-power security primitive, have been proposed to be used in a number of key management subtasks in IoT settings. Rahman et al. proposed to use PUFs for secure key generation [5]; Mukhopadhyay proposed a novel device authentication method that takes the advantage of the unclonable property of PUFs [6]. Most recently Huang et al. investigated a key distribution protocol assisted using ring oscillator PUFs (ROPUFs)[7] which significantly reduces the storage overhead and latency for securely distributing secret keys. Unfortunately, these works only focus on a specific subtask of key management and fail to provide detailed security or overhead analysis. We differentiate ourselves by design a novel PUF architecture that can be applied to the entire key management lifecycle including key distribution, key storage and rekeying in IoT. We have also performed a security and overhead analysis to prove that our work is both secure and efficient.

3. MULTISTAGE INTERCONNECTED PUF

In this section, we propose a novel PUF structure called Multistage Interconnected PUF (MIPUF). We borrow the idea of multistage interconnection networks (MINs) from computer networks field. MINs allow the processing elements (PEs) to be interconnected using Switching Elements (SEs) such that the interconnection provides high configurability and speed with low cost. We propose to use such structure to interconnect PUFs so that the interconnected PUFs can be configured easily. The interconnected PUFs significantly increase the system complexity as well as break the linearity, resulting in increased difficulty in modeling the system. The configurability also allows the challenge-response pairs (CRPs) of the network to be remapped from time to time, protecting the system from modeling attacks.

3.1 Processing Elements (PEs)

We name the PE in a MIPUF a MIPUF node. A MUPUF node is the most fundamental building block of the network. A MIPUF node is a single or a group of strong PUFs that take an n-bit challenge and generate an m-bit response. A strong PUF is defined as a PUF that supports a large number of CRPs [8], and existing implementation of strong PUFs include but is not limited to PUFs such as optical PUF, arbiter PUF and LRR-DPUF [9]. For the sake of implementation easiness, our implementation of a MIPUF node consists of m n-bit arbiter PUFs running in parallel and sharing the same pulse signal and challenge vector. Even though arbiter PUFs are known to be weak against various modeling attack, our experimental results show that multistage interconnection significantly improves the resilience against them. In this paper, we merely use MIPUF node implemented with arbiter PUFs as an illustrative example and a proof of concept. Security properties of MIPUF implemented using more advanced strong PUFs such as LRR-DPUF [9] are expected to exceed our collected results.

3.2 Switching Elements (SEs)

Similar to the concept of SEs in computer networks, the SE in MIPUF serves as a way to route and switch signals. In our implementation, an SE is a set of multiplexers that switch or not switch n signals based on a configuration bit. In our case, we use SEs to connect the response of a previous MIPUF node to the next node as the new challenge. The SEs between two nodes are controlled by a configuration vector.

3.3 Multistage Interconnection

Multistage interconnection networks find a balance between the cost and configurability. We believe a blocking multistage connection is the most cost-efficient for MIPUF implementation and provides sufficient configurability. A blocking multistage connection cannot realize all possible connections between inputs and outputs since a connection between one free input to another free output is blocked by an existing connection in a network; however, it is much cheaper to implement. We propose to implement a blocking interconnection in a MIPUF in an Omega network style [10] which consists of 2×2 SEs. Each input has a dedicated connection to an output, providing 2^N different switchings and having a complexity of O(N log(N)) for an N×N connection between two MIPUF nodes. An example of such interconnection is shown in Figure 1. To be noted that we do not allow port rearrangement in MIPUF, each input should be routed to a unique output and each output should be directed from a unique input given a specific configuration.

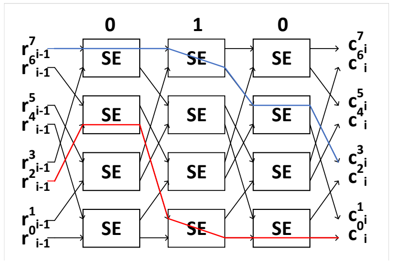


Figure 1: Network structure in Omega network style. r^k_{i-1} indicates the k-th response (output) bit of the $(i-1)$-th MIPUF node, c^k_i indicates the k-th challenge (input) bit of the i-th MIPUF node.

3.4 Protecting Network Configuration

The signal routing between any two MIPUF nodes is controlled by a configuration vector. We propose to secure the configuration vectors using existing MIPUF nodes in the system so that the real interconnection configuration remains hidden. The configuration vector for the interconnection between node i and $i+1$ depends on the encrypted result of the user provided configuration bits using the nodes from node 1 to node $i-1$, for $i > 1$. To note that we use MIPUF nodes in the previous levels to encrypt SE configurations to reduce correlation between the output of a node and it's immediate SEs. An illustrative example is shown in Figure 2.

The user provided configuration is passed to the SEs between the first two nodes and propagate along the network to configure the remaining SEs connected to the later nodes. An attacker or even the user who provided the configuration, cannot obtain information on the real interconnection between MIPUF nodes without characterizing each node. Besides, we also significantly reduce the number of bits an

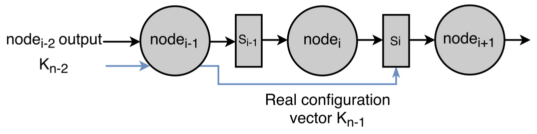


Figure 2: Encrypting MIPUF interconnections using existing MIPUF nodes. K_j is the configuration vector encrypted by a chain of nodes from $node_1$ to $node_j$.

user needs to provide. In a key management protocol, our proposed method could also significantly reduce the communication cost.

3.5 Security Evaluation of MIPUF

3.5.1 Uniqueness and Reliability

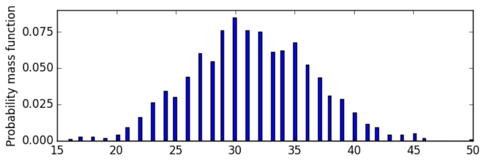
Two most important properties of PUFs are uniqueness and reliability. Uniqueness means that the responses for a specific PUF design implemented on different devices should be significantly different when provided with the same challenge. Reliability indicates the response should be stable enough when repeating the same challenge on the same device. Since our design of MIPUF depends on existing PUF implementations, our focus is that our multistage interconnection does not compromise the security properties of the PUF implementation we depend on. We modify the definition of uniqueness and reliability as follows.

- *Inter-configuration variation (uniqueness).* How many MIPUF output bits are different between two different configurations of the same MIPUF? Ideally, this variation should be 50%.
- *Intra-configuration variation (reliability).* How many MIPUF output bits differs when re-generated again from a MIPUF with a specific configuration? Ideally, this variation should be 0%.

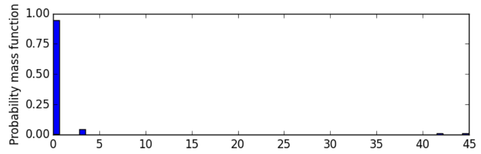
We directly compare these two metrics with intra-chip variation and inter-chip variation metrics in regular PUF evaluations. As a proof of concept, we compare our arbiter PUF based MIPUF implemented using arbiter PUFs with regular FPGA-based arbiter PUFs implemented on five different FPGAs. The results are collected from Xilinx Spartan-6 XC6SLX45 platform using the implementation described in [11].

Figure 3a illustrate the probability distribution of the inter-configuration variation of a MIPUF. The x-axis is the number of output bits that are different between two different interconnection configurations; the y-axis is the probability. The bars show experimental results collected on 1,225 pairs of outputs collected from 50 different configurations. Our experiment results (47.9%) is very close to the ideal case of 50%. Our results even show a slight improvement comparing to the inter-chip variation of arbiter PUFs implemented on FPGAs (47.0%).

We calculated a 35.37% intra-configuration variation when no error correction is applied. Consider the intra-chip variation of 64 128-bit arbiter PUFs implemented on five different FPGAs is only as little as 2.90%, MIPUF is very unstable without error correction. The reason is simple and intuitive, as all MIPUF nodes are connected in such a way that each node takes the output of the previous node as the input, an error in the first node could result in avalanche effect



(a) Inter-configuration variation for MIPUF is 47.9% (Avg = 30.7 bits / 64 bits).



(b) Intra-configuration variation for MIPUF with fuzzy extractor is 2.67% (Avg = 1.71 bits / 64 bits). Environment range from 20 °C, 0.95V to 65 °C, 1.2V.

Figure 3: Inter-configuration and intra-configuration variation of a MIPUF with four nodes. Each each node is implemented using 64 32-bit arbiter PUFs. The interconnection between nodes is designed in a blocking fashion as shown in Figure 1.

in intra-configuration variation. Thus, we propose to use a lightweight fuzzy extractor between every MIPUF node as an error correction mechanism [12], and the resulting intra-variation is significantly reduced to 2.67%. Since a MIPUF with n nodes requires n clock cycles to generate the result, the fuzzy extractors can be shared for all node outputs.

Figure 3b illustrates the probability distribution of the intra-configuration variation of the same MIPUF. The environments parameters ranging from 20 °C, 0.95V to 65 °C, 1.2V. The bars show experimental results collected on 50,000 different random interconnections. Each configuration is performed on 10,000 challenges and repeated 20 times. Noted that the major contributors to the intra-configuration variation are two extremely rare ($< 0.5\%$) case of Hamming distance greater than 40.

3.5.2 Resilience Against Modeling Attack

Several PUF-based systems are vulnerable to a variety of modeling attacks [13]. We observed that MIPUF significantly boost modeling attack resilience by increasing the system complexity and breaking the system linearity. Table 1 shows the best prediction accuracy on MIPUF vs. a variety of PUFs implemented on FPGA using attack approaches described in [13]. We observe that all prediction accuracies for a single-bit in MIPUF outperforms other designs and are all close to the ideal case of 50%.

Architecture	LR	ES	DL
MIPUF - 4 nodes	**51.33%**	**59.18%**	**50.59%**
256-bit 4-XOR PUF	97.21%	76.02%	78.42%
1024-bit arbiter PUF	96.57%	98.28%	88.98%
1024-bit 64-ff PUF	58.29%	95.68%	87.70%

Table 1: Best single-bit prediction accuracy on different PUF architectures using logistic regression (LR), evolution strategies (ES) and deep learning (DL) attacks out of 100 runs. Each attack uses 100,000 CRPs. Total number of arbiter PUF segments used in all architectures are fixed to 1,024.

In addition to high resilience against modeling attacks, MIPUF also allows cheap and fast reconfiguration. Frequent reconfiguration of MIPUF renders modeling attacks almost impossible. We investigate this topic in more detail in Section 4.3.3.

4. GROUP KEY MANAGEMENT

In this section we show that we can utilize the MIPUF structure to securely establish a group key management protocol with three major components, respectively key distribution, key storage and rekeying. Key distribution is the process to securely deliver the shared secret key to every authorized group member. After the group key has been successfully distributed, the most important task would be to securely store the secret key so that the user could easily access the key when needed, but an adversarial is forbid to peek or tamper with the secret key. Lastly, rekeying allows a group to renew or replace the group key from time to time.

To illustrate our protocol, we first define an IoT model consists of a control unit with higher computational power and multiple IoT device/nodes that are constraint by both computational power and battery life. Each IoT node embeds a MIPUF, a hardware hashing function and a very compact AES implementation.

4.1 Key Distribution

According to the model described above, a well-designed group key distribution protocol is proposed. The protocol is shown in Protocol 1. For each node, a group key can be delivered securely with exchange of two messages.

4.2 Key Storage

After the key distribution, the group key can be extracted from the MIPUF when the correct challenge and interconnection configuration are provided. Unlike other crypto-based key management systems, we do not directly store the group key in the memory. Instead, the group key is extracted on the fly from the group key hint p_i. We believe this approach is secure because an attacker can only obtain the real group key if he has access to both the storage (containing p_i, f_i and $c_i^{\gamma_i}$) and the MIPUF ($\mathbb{F}^{\gamma_i}$, compromising either the storage or the MIPUF does not compromise the security of the whole design. Also, the group key is only used upon receiving or transmitting group messages, thus storing the real key using low-power MIPUF is also highly energy efficient.

4.3 Rekeying

Group keys need to be regenerated, redistributed or updated whenever there is a dynamic change to the group to preserve security. One important motivation to rekey is that groups are not always static. When a member leaves the group, it should not be able to decrypt future group communications (**forward security**); when a new member joins, it should not be able to decrypt past group communications (**backward security**). Group key should also be completely rekeyed when potential leakage is detected for security considerations. Here we discuss all three possible cases.

4.3.1 New Member Joins the Group

Without loss of generality, we assume a new IoT node N_α intend to join a group $\mathcal{G}$, $N_\alpha \notin \mathcal{G}$. For efficiency considerations, redistributing a new key to all group members is expensive and inefficient. Instead, we propose to use the current secret to encrypt the new group key and this process is leakage free. Specifically, the control unit sends out a message $msg^{join} = \{E_{key_g}(key_g')\}$ to $\forall N_i \in \mathcal{G}$. The existing group members calculate and store the new group key hint

Protocol 1 Group Key Distribution Protocol

Inputs. A list of group member in group $\mathcal{G} = \{N_0 \cdots N_n\} \subseteq \mathcal{N}$ n being the total number of IoT nodes in the group. A random group key key_g.

Goal. Securely deliver key_g to all $N_i \in \mathcal{G}$.

1. **Preliminary Phase**
   (a) Before the deployment of an IoT node, the control unit assigns a unique ID (N_i) to it. Initially the node derives the interconnection configuration $\gamma_i = H(N_i)$ from N_i and generates a CRP ($c_i^{\gamma_i}$, $r_i^{\gamma_i}$) using the MIPUF $\mathbb{F}_i$ where $r_i^{\gamma_i} = \mathbb{F}_i^{\gamma_i}(c_i^{\gamma_i})$.
   (b) The control unit securely store the tuple (γ_i, $c_i^{\gamma_i}$, $r_i^{\gamma_i}$) in the database, and node N_i securely stores $c_i^{\gamma_i}$ and γ_i.
2. **Key Delivery Phase**
   (a) When a group $\mathcal{G}$ is formed, the control unit first check if all group members exists based on the unique ID. If not, the protocol is aborted.
   (b) For IoT node $N_i \in \mathcal{G}$, the control unit generates a random new configuration γ_i'.
   (c) For IoT node $N_i \in \mathcal{G}$, a group key hint $p_i = r_i^{\gamma_i} \otimes key_g$ and a new configuration hint $f_i = r_i^{\gamma_i} \otimes \gamma_i'$ are generated.
   (d) An encrypted message msg_i^k containing p_i and f_i is transmitted using unicast to each group member $N_i \in \mathcal{G}$. $msg_i^k = \{E_{r_i^{\gamma_i}}(N_i \| p_i \| f_i) \| H(N_i \| c_i^{\gamma_i})\}$, "$\|$" indicates the concatenation operation, E is the encryption operation using AES and H is a hashing operation.
3. **CRP update Phase**
   (a) Upon receiving msg_i^k, IoT node N_i first decrypts the message using $r_i^{\gamma_i}$: $D_{r_i^{\gamma_i}}(E_{r_i^{\gamma_i}}(N_i \| p_i \| f_i))$, D being the AES decryption operation. N_i verifies the validity of the message by comparing the hash $H(N_i \| c_i^{\gamma_i})$. If there exists a mismatch in the hash, report error to the control unit.
   (b) The group key key_g and the new interconnection γ_i' are derived from p_i and f_i decrypted from the decrypted msg_i^k.
   (c) N_i generates a new CRP ($c_i^{\gamma_i'}$, $r_i^{\gamma_i'}$) where $c_i^{\gamma_i'} = H(c_i^{\gamma_i})$, $r_i^{\gamma_i'} = \mathbb{F}_i^{\gamma_i'}(c_i^{\gamma_i'})$. N_i sends an encrypted message $msg_i^u = E_{r_i^{\gamma_i}}(N_i \| c_i^{\gamma_i'} \| r_i^{\gamma_i'})$ back to the control unit.
   (d) The control unit decrypt msg_i^u using r_i^{γ} and updates the database by replacing the tuple (γ_i, $c_i^{\gamma_i}$, $r_i^{\gamma_i})with(\gamma_i'$, $c_i^{\gamma_i'}$, $r_i^{\gamma_i'}$). If the control unit has not received an update message msg_i^u after some predefined timeout or $c_i^{\gamma_i'} = H(c_i^{\gamma_i})$, an abort is called.

$p_i' = r_i^{\gamma_i} \otimes key_g'$ and deletes key_g' upon receiving and decrypting msg_i^{join}. The new member will have to complete the whole key distribution process described in Section 4.1. Backward security is preserved using this method since the new member has no information about the old group key.

4.3.2 Existing Member Leaves the Group

Removing an existing member from the group is more complicated than adding a new member. Here we propose to divide group $\mathcal{G}$ into m subgroups $\mathrm{g}_j \subset \mathcal{G} = \{\mathrm{g}_1 \cdots \mathrm{g}_m\}, 1 \leq j \leq m$. All nodes in the same subgroup share the same interconnection configuration $\gamma_{\mathrm{g_j}}$. Again, without losing the generality, we assume an IoT node $N_\beta \in \mathrm{g}_j \subset \mathcal{G}$ intend to leave the subgroup where all members in the subgroup use the same MIPUF interconnection configuration γ_j. The control node first multicast/broadcast $m-1$ messages $msg_i^{leave} = \{E_{\gamma_i}(key_g' \| H(\gamma_i)\}$ containing the new key to all the subgroups encrypted using the configuration $\gamma_i, i \neq j, 1 \leq i \leq m$. Upon receiving the message, each node first decrypts the message using its own configuration γ_i and check if $\mathrm{H}(\gamma_i)$ matches the one in the decrypted msg_i^{leave}. If so then the decrypted new group key key_g' is valid, otherwise, discard the message. No member of g_j including the leaving node have any knowledge of the configurations of other subgroups, thus incapable of decrypting the message correctly. The control unit should then perform unicast communications to all members of g_j by distributing the new group key key_g' and a new configuration γ_j' to replace γ_j.

4.3.3 Complete Rekeying

MIPUF can still be modeled if a significantly large enough set of CRPs is collected. However, MIPUF can be reconfigured to neutralize modeling attacks by completely remap the input-output mapping. We propose to perform a full rekeying once the total number of CRPs generated exceeds a calculated sample complexity lower bound that equals to the sufficient training set size to break the MIPUF. Equation 1 describe a sample size lower bound in terms of the IPN model parameters, where m is the number of nodes in MIPUF and n is the maximum number of PUFs in a MIPUF node. $k = VC(\mathbb{F})$ where VC is the Vapnik-Chervonenkis-dimension and $\mathbb{F}$ is largest single PUF in MIPUF. δ is the failure probability and ϵ is the learning error.

$$\text{Sample complexity} \sim \frac{(m \cdot k + m) \cdot n + ln(\frac{1}{\delta})}{\epsilon} \qquad (1)$$

5. EVALUATION

5.1 Security Analysis

We make the following assumptions for our security analysis. The physical security of the MIPUF is secured; however, an attacker is allowed query the CRPs as much as needed. The wireless channels used for communication are not secured after the initial preliminary phase. The hash function and compact AES on each node are secure. The control unit key database is secure. We summarize our security analysis against several popular attacks as below:

Eavesdropping Attack: During the key distributing and rekeying process, all messages containing γ_i, $c_i^{\gamma_i}$, $r_i^{\gamma_i}$ or key_g are encrypted by AES. Thus eavesdropping attack is invalid.

Man-in-the-middle Attack: Before updating the new CRP in **3c** in Protocol 1, the new challenge is a one-way hash of the previous challenge which is checked by the control unit, thus render the attack useless.

Replay Attack: Neither the IoT node nor the control unit would be able to correctly decrypt a message encrypted using a previous response since old responses are discarded after the update.Thus the hash check in **3a**, **3d** in Protocol 1 would fail during key distribution. Forward security in the rekeying process is designed to protect the system from such attacks.

Impersonation Attack: Based on our assumption, the preliminary phase is secure thus the initial key and MIPUF configuration are secured. Also, the modeling attack resilience and the reconfigurability of MIPUF prevents an attacker to impersonate an IoT node even if we allow him to query the CRPs.

5.2 Overhead Evaluation

In this section we assume that there are N nodes in the group and the group is split into M subgroups. The MIPUF we use in each IoT nodes takes an **a**-bit challenge, a **b**-bit configuration vector and generates a **c**-bit output (assuming $\mathbf{b} \geq \mathbf{c}$). Node ID is a **l**-bit vector. The hash function hashes any input to an **a**-bit string. The group key has length of **c**-bits. The random number generator cost E_R units of energy per operation. The energy consumption of MIPUF, hash function, XOR operation and the very compact AES are E_P, E_H, E_X and E_A.

Communication Cost: The length of messages: msg_i^k, msg_i^u, msg^{join} and msg_i^{leave} are: $\mathbf{a}+\mathbf{b}+\mathbf{c}+\mathbf{l}$, $\mathbf{a}+\mathbf{c}+\mathbf{l}$, $\mathbf{a}+\mathbf{c}$ and $\mathbf{a}+\mathbf{c}$. Thus the total number of messages need to be sent for key distribution and node join/leave rekeying are: $\mathbf{N}$, $\mathbf{3}$ and $(\frac{\mathbf{2N}}{\mathbf{M}}\textbf{-2}) + (\textbf{M-1})$. For node leave rekeying, minimum cost is achieved when $M = \sqrt{N}$.

Storage Overhead: The control unit stores the Node ID, CRPs and the current configuration of all nodes; thus the storage overhead at the control unit is $N \cdot (\mathbf{a}+\mathbf{b}+\mathbf{2c}+\mathbf{l})$ bits. Each IoT node stores the Node ID, current challenge, current configuration and the group key hint which has a storage overhead of $\mathbf{a}+\mathbf{b}+\mathbf{c}+\mathbf{l}$ bits.

Energy Cost: The control unit spends $2E_A + 2E_H + 2E_R + 2E_X$ units of energy to distribute the group key to one node. Each node spends $2E_A + 2E_H + E_P + 2E_R$ to receive the key and update the CRP. During member join rekeying, the control units spends $E_A + E_R$ units of energy to update the group key to existing members and $2E_A + 2E_H + 2E_R + 2E_X$ to the new member. The new node spends $2E_A + 2E_H + E_P + 2E_X$ and old members spend $E_A + E_X$ units of energy respectively. During member leave rekeying, the control unit spends $(M-1) \cdot (E_A + E_H + E_R)$ units of energy to update the group key to existing members that are not in the same subgroup as the leaving node and $(\frac{N}{M} - 1) \cdot (2E_A + 2E_H + 2E_R + 2E_X)$ to the update all members in the same subgroup. All members that are in and not in the same subgroup as the leaving node spends $(M-1) \cdot (E_A + E_H + E_R)$ and $E_A + E_H$ units of energy.

We compare our global energy consumption to two other key management protocols: Localized Encryption and authentication protocol (LEAP) [2] and Elliptic Curve Public Key Cryptography (ECPKC) [4] in simulation using the parameters described in [4] for a fair comparison. The comparison for simulated results for global energy consumption for three key management schemes can be seen in Figure 4. LEAP uses significantly much more energy than both

ECPKC and our proposed scheme as it grows quadratically. The global energy consumption of both ECPKC and our proposed scheme grows linearly. Since our proposed design uses low-power MIPUF instead of energy-hungry ECC to achieve power efficiency. We observe that our proposed scheme uses about 47.33% less energy for key distribution.

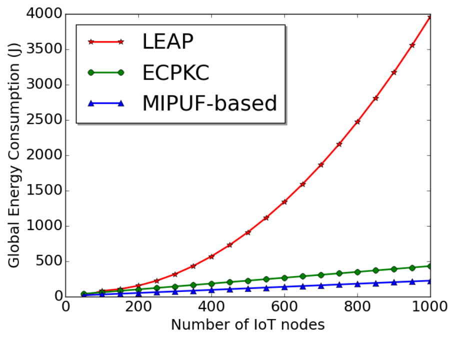


Figure 4: Simulated global energy consumption (J) vs. total number of IoT nodes under the settings introduced in [4].

5.3 Implementation Results

We implemented our key management hardware support for IoT nodes on Xilinx Spartan-6 LX45 FPGAs to measure the area and power. Our implementation consists of a MIPUF with three nodes; each node consists of 128 32-bit Arbiter PUFs. The fuzzy extractor was implemented based on [12], the hash function the AES module are implemented based on [14] and [15]. Table 2 shows the area and power overhead break down of our implementation. Our MIPUF seems to be more expensive due to FPGA-based arbiter PUF implementations are known to be inefficient. Both area and power overhead of MIPUF and our hardware support are expected to be significantly reduced if we implement MIPUF on ASIC. We are also expected to see further improvement if MIPUF is built using more efficient strong PUFs.

Our design	MIPUF	SHA-1	AES	Overall
Flip-flops	1,626	1,151	598	3,401
LUTs	7,028	1,590	501	9,219
Slices	3,717	544	222	4,553
Block RAMs	0	0	3	3
Power(mW)	123.7	30.4	16.2	175.7

Table 2: FPGA resource and power characteristics of the hardware support of our proposed key management scheme.

6. CONCLUSIONS

In this paper, we first proposed a novel PUF structure: MIPUF that is both secure and reconfigurable. We showcased the uniqueness, reliability, modeling attack resilience and reconfigurability of MIPUF. We then proposed a group key management scheme in IoT consists of key distribution, key storage and rekeying based on MIPUF. Security and overhead analysis on the scheme show that our design is not only secure against multiple attack methods but also low power. Our simulation result indicates that our proposed scheme spends 47.33% less energy compared to the state-of-the-art crypto-based scheme ECPKC [4] since we use low-power and energy efficient MIPUF instead of power-hungry ECC.

7. REFERENCES

[1] L. Veltri, S. Cirani, S. Busanelli, and G. Ferrari, "A novel batch-based group key management protocol applied to the internet of things," *Ad Hoc Networks*, vol. 11, no. 8, pp. 2724–2737, 2013.

[2] S. Zhu, S. Setia, and S. Jajodia, "Leap+: Efficient security mechanisms for large-scale distributed sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 2, no. 4, pp. 500–528, 2006.

[3] R. Roman, C. Alcaraz, J. Lopez, and N. Sklavos, "Key management systems for sensor networks in the context of the Internet of Things," *Computers & Electrical Engineering*, vol. 37, no. 2, pp. 147–159, 2011.

[4] W. Abdallah, N. Boudriga, D. Kim, and S. An, "An efficient and scalable key management mechanism for wireless sensor networks," in *ICACT*, pp. 480–493, IEEE, 2015.

[5] M. T. Rahman, F. Rahman, D. Forte, and M. Tehranipoor, "An aging-resistant RO-PUF for reliable key generation," *IEEE Transactions on Emerging Topics in Computing*, vol. 4, no. 3, pp. 335–348, 2016.

[6] D. Mukhopadhyay, "PUFs as promising tools for security in Internet of things," *IEEE Design & Test*, vol. 33, no. 3, pp. 103–115, 2016.

[7] M. Huang, B. Yu, and S. Li, "PUF-assisted Group Key Distribution Scheme for Software-Defined Wireless Sensor Networks," *IEEE Communications Letters*, 2017.

[8] C. Herder, M.-D. Yu, F. Koushanfar, and S. Devadas, "Physical unclonable functions and applications: A tutorial," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1126–1141, 2014.

[9] J. Miao, M. Li, S. Roy, and B. Yu, "LRR-DPUF: Learning resilient and reliable digital physical unclonable function," in *ICCAD*, pp. 1–8, IEEE, 2016.

[10] D. H. Lawrie, "Access and alignment of data in an array processor," *IEEE Transactions on Computers*, vol. 100, no. 12, pp. 1145–1155, 1975.

[11] J. X. Zheng, D. Li, and M. Potkonjak, "A secure and unclonable embedded system using instruction-level PUF authentication," in *FPL*, pp. 1–4, IEEE, 2014.

[12] C. Herder, L. Ren, M. van Dijk, M.-D. Yu, and S. Devadas, "Trapdoor computational fuzzy extractors and stateless cryptographically-secure physical unclonable functions," *IEEE Transactions on Dependable and Secure Computing*, vol. 14, no. 1, pp. 65–82, 2017.

[13] U. Rührmair, J. Sölter, F. Sehnke, X. Xu, A. Mahmoud, V. Stoyanova, G. Dror, J. Schmidhuber, W. Burleson, and S. Devadas, "PUF modeling attacks on simulated and silicon data," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 11, pp. 1876–1891, 2013.

[14] I. Kawazome, "Secure Hash." https://github.com/ikwzm/SECURE_HASH, 2013.

[15] P. Chodowiec and K. Gaj, "Very compact FPGA implementation of the AES algorithm," in *International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 319–333, Springer, 2003.