國立臺灣大學電機資訊學院電機工程學系

碩士論文

Department of Electrical Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master's Thesis

可信執行環境保護的組合型隨機數產生器於無人機之應用

Combinatorial Random Number Generators Protected by Trusted Execution Environments for Drone Applications

王思翰

Szu-Han Wang

指導教授：吳沛遠 博士

賴怡吉 博士

Advisor: Prof. Pei-Yuan Wu, Ph.D.

Prof. Alexander I-Chi Lai, Ph.D.

中華民國 114 年 6 月

June, 2025

doi:10.6342/NTU202501452

摘要

在現代資安架構中，隨機數產生器（Random Number Generator, RNG）的可靠性對於保護加密運算及提升系統對抗網路威脅的能力具有關鍵性影響。傳統設計普遍依賴單一熵源，當系統部分遭受攻擊時，尤其是在一般執行環境容易被外部攻擊的情境下，隨機數品質易受影響，進而削弱整體系統安全性。

本論文提出一套結合可信任執行環境（Trusted Execution Environment, TEE）與豐富執行環境（Rich Execution Environment, REE） 的組合型隨機數生成框架。此架構於 TEE 與 REE 中分別部署獨立的 RNG，並設計多種混合機制來融合兩個來源之隨機數。藉由跨域熵源混合與冗餘保護設計，即使其中一個熵源遭受破壞，最終隨機數輸出仍可保持高不可預測性，同時透過 TEE 的硬體隔離特性，進一步鞏固整體系統的安全韌性。

本研究使用 NIST SP800-22 隨機性測試套件進行評估，結果顯示所提出之組合型 RNG 在隨機性品質上能維持或優於單一熵源。為驗證其實務應用性，本架構亦整合至無人載具（Unmanned Aerial Vehicle, UAV）系統環境中，展示其於資源受限邊緣設備中提升隨機數安全性的效果。

關鍵字：隨機數產生器、熵源組合、可信任執行環境、無人機

doi:10.6342/NTU202501452

ABSTRACT

Random number generators (RNGs) play a crucial role in cryptographic operations to establish cybersecurity defense. Traditional designs of RNGs, however, typically rely on a single entropy source, causing critical vulnerabilities to the overall system security.

To address the aforementioned challenge, this study proposes a combinatorial RNG scheme protected by a hybrid architecture combining a hardware-protected Trusted Execution Environment (TEE) and the conventional Rich Execution Environment (REE). In this framework, Independent RNGs are separately deployed in the TEE and REE domains, respectively, where their outputs are combined securely in TEE through selected techniques, including XOR operations, SHA-256 hashing, AES encryption, or chaining mechanisms. The hardware isolation enforced by TEE further protects the critical entropy sources as well as the combinatorial operation. By leveraging cross-domain entropy mixture and redundancy, the framework ensures that even if some entropy source is compromised, the final output remains adequately random. A benefit of such a framework is that some entropy sources can be placed outside TEE to save the critical security resources, without compromising the overall security level.

doi:10.6342/NTU202501452

Extensive evaluations using the NIST SP800-22 randomness test suite verified that the proposed combinatorial RNG improves randomness quality compared to single-source RNGs. Moreover, the proposed approach was realized on a companion computer prototype for an unmanned aerial vehicle (UAV) to validate practical applicability, showcasing its potential to enhance randomness security in resource-constrained edge devices.

Keywords: Random Number Generators, Entropy Source Combination, Trusted Execution Environment, Drones / Unmanned Aerial Vehicle (UAV)

doi:10.6342/NTU202501452

Table of Contents

doi:10.6342/NTU202501452

doi:10.6342/NTU202501452

doi:10.6342/NTU202501452

List of Tables

doi:10.6342/NTU202501452

List of Figures

doi:10.6342/NTU202501452

Chapter 1

Introduction

1.1 Background and Motivation

Trusted Execution Environment (TEE) technology [1] plays a critical role in today's embedded systems. It provides a hardware-isolated environment that helps protect sensitive operations from potential attacks coming from the Rich Execution Environment (REE). This separation allows critical security functions to run independently of general-purpose applications, improving the overall resilience of the system.

TEE has already been widely used in securing cryptographic operations, such as key storage, authentication, and secure boot [2]. However, its application in random number generation is still relatively uncommon. Random Number Generators (RNGs) are a foundational component in system security, and their quality directly affects the strength of cryptographic protocols [3]. Traditionally, most embedded platforms and Unmanned Aerial Vehicle (UAV) systems rely on a single entropy source, often located entirely within the normal operating system. This leaves the system vulnerable—if that source is compromised or becomes predictable, the RNG's output can degrade, weakening system security.

If an attacker can predict or influence the REE-based RNG, the generated random numbers may lose their unpredictability. This puts cryptographic opera-

doi:10.6342/NTU202501452

tions, and potentially the entire system as well, at risk. This is the key motivation for designing an RNG architecture with better statistical randomness that spans both the TEE and REE domains, combining their strengths to enhance resilience against such threats.

To demonstrate the practicality of the proposed design, this study also implemented the full application-layer features on a TEE-enabled SoC. These include asymmetric signing, symmetric encryption/decryption.

1.2 Problem Statement

In modern UAVs and IoT systems, RNGs are the foundation of cryptographic security. However, most embedded platforms still rely on a single entropy source—typically located in the REE. According to NIST SP 800-90B (*Recommendation for the Entropy Sources Used for Random Bit Generation*) [4], RNG security depends on the unpredictability of its entropy source. If that source is compromised or behaves abnormally, the output loses its cryptographic strength. To improve resilience, NIST SP 800-90C (*Recommendation for Random Bit Generator (RBG) Constructions*) [3] recommends combining multiple independent entropy sources to ensure unpredictability even if some sources fail.

Meanwhile, ARM TrustZone provides hardware-level isolation via its Trusted Execution Environment (TEE), separating secure operations from potentially untrusted applications. This makes TEE an ideal place to host a trusted entropy source. This combination allows for the construction of a RNG scheme with better statistical randomness.

Given this context, the research focuses on two core questions:

- How to design a lightweight yet secure RNG architecture that mixes entropy

doi:10.6342/NTU202501452

across TEE and REE, suitable for UAVs and other constrained devices?

- How to ensure the mixed entropy remains secure and unpredictable, even if part of the system is compromised?

To answer these questions, the research designed and implemented a dual-domain hybrid RNG system on ARM TrustZone, and evaluated its randomness using the NIST SP 800-22 statistical test suite to verify both feasibility and security.

1.3 Achievements

The key achievements of this study encompass:

- **Identified Better Combination(s) in Randomness on Entropy Sources**
  Multiple entropy combination approaches, including XOR, SHA-256, AES-based, and chaining-based approaches, were evaluated between TEE and REE. According to the results of NIST SP 800-22 statistical tests, the chaining-based mixing method demonstrated superior randomness quality and stability. This finding confirms the effectiveness of entropy source fusion and highlights specific methods that outperform conventional single-source designs in terms of both statistical reliability and structural resilience.

- **Developed a scheme for allocating RNG Combinations to TEE/REE Zones**
  An entropy mixing scheme was developed to allocate entropy sources across the trusted TEE and untrusted REE environments. The scheme ensures that even if the REE entropy source is compromised, the final output remains statistically unpredictable due to secure mixing with the TEE-provided entropy.

doi:10.6342/NTU202501452

The architectural assumptions, security boundaries, and attacker models were clearly defined to verify the feasibility.

- **Designed and implemented the above RNG Combination under TEE/REE using Commercially off-the-shelf (COTS) products**
  The proposed dual-domain RNG system was implemented on a Raspberry Pi 3B+ running OP-TEE, without requiring any custom hardware. Key security functions, including ECDSA digital signatures, AES encryption/decryption, and secure key updates, were realized entirely within the Secure World, the TEE realization of ARM TrustZone. This practical implementation highlights the compatibility of the proposed method with publicly available open-source platforms, demonstrating its deployability in real-world embedded security applications.

1.4 Thesis Organization

The remainder of this thesis is organized as follows. Chapter 2 reviews related works on random number generation, entropy mixing techniques, TEE technologies, secure mechanisms in embedded system and UAV security challenges. Chapter 3 describes the proposed system architecture and threat model, outlining the assumptions, attack scenarios, and design goals. Chapter 4 elaborated and evaluated typical entropy mixing methods in randomness based on the results of NIST SP 800-22 test suite. Chapter 5 details the experiments regarding how to realize and test the proposed solution on a Raspberry Pi 3B+ using OP-TEE, including key security functions like digital signatures, AES encryption, and key updates. Finally, Chapter 6 concludes with a summary of contributions, insights on the system's strengths and limitations, and directions for future work.

doi:10.6342/NTU202501452

Chapter 2

Related Works

2.1 RNG Research and Development

Random number generators (RNGs) play a foundational role in cybersecurity systems, with applications ranging from key generation and encryption to authentication, digital signatures, and secure communication protocols [5]. With the proliferation of embedded systems and Internet of Things (IoT) devices, providing high-quality randomness in resource-constrained environments has become a key design challenge.

2.1.1 Types and Applications of RNGs

RNGs are generally classified into two categories: true random number generators (TRNGs) and pseudorandom number generators (PRNGs). TRNGs rely on physical phenomena, such as thermal noise, oscillator jitter, or SRAM power-up states, to generate high-entropy outputs [6]. While they offer excellent unpredictability, TRNGs are often sensitive to environmental factors and can be costly to implement. On the other hand, PRNGs are based on algorithms and initial seeds to generate large sequences of random numbers. Common implementations include AES-CTR-based deterministic RNGs (CTR-DRBG) and SHA-based constructions (HASH-DRBG) [7]. However, the randomness of PRNGs heavily depends on the quality of the seed, and weak or predictable seeds can compromise system security.

doi:10.6342/NTU202501452

Embedded platforms and IoT devices face additional challenges such as limited access to high-entropy sources, unstable entropy pools during startup, and lack of continuous entropy harvesting [8, 9]. According to NIST SP 800-90B, a secure RNG should include entropy source monitoring, ongoing health checks, and multiple entropy inputs [4]. Without these mechanisms, systems become vulnerable to risks like low-entropy seeds, repeated seeds, and predictable sequences, ultimately weakening encryption and key protection [10].

2.1.2 Development of Combinational RNGs

Combinational RNGs, as a special class of entropy allocation strategies, aim to integrate multiple entropy sources or RNG modules to enhance statistical strength and system resilience. To mitigate issues like entropy shortage or single-point failure, researchers and practitioners have proposed the concept of *combinational RNGs*, which enhance randomness, increase period length, and improve resistance against attacks. Early examples include the Combined Linear Congruential Generator (CLCG) proposed by Wichmann and Hill [11], and Multiple Recursive Generators (MRGs) introduced by LÉcuyer [12], which improve cycle length and distribution properties by combining multiple Linear Congruential Generators (LCGs) or Linear feedback shift register (LFSRs) [13–15].

However, these linear constructions often exhibit predictable *linear artifacts* and struggle to pass sensitive randomness tests [16]. To address this, recent designs incorporate nonlinear combinations using techniques such as XOR mixing, encryption-based post-processing (e.g., AES, SHA), nonlinear feedback shift registers (NFSRs), or chaotic systems. For example, Ramasubramanian et al. [17] designed a chaotic hybrid RNG with nonlinear terms and multilayered bit rotations, achieving significant entropy improvement. Xie et al. [18] proposed a TRNG that

doi:10.6342/NTU202501452

combines SRAM power-up states with NFSRs, demonstrating strong resistance to differential power analysis (DPA) attacks.

From a standards perspective, NIST SP 800-90C explicitly recommends combining multiple entropy sources and applying post-processing (e.g., hashing or XOR) to maintain unbiasedness and decorrelation [8]. Similarly, Dodis et al. [19] introduced the concept of *entropy reconciliation* in fuzzy extractor designs to enhance randomness in noisy entropy inputs. Furthermore, Heninger et al. [20] demonstrated real-world vulnerabilities in embedded systems due to reused or insufficient entropy sources, highlighting the importance of high-quality entropy mixing.

In summary, combinational RNGs have evolved from simple linear operations to complex hybrid strategies involving multiple nonlinear operations, physical entropy sources, and post-processing techniques. Following this trajectory, this research proposes four cross-domain mixing mechanisms (XOR, SHA-256, AES, and chained mixing) within a TEE/REE architecture to enhance both the security and applicability of RNGs in resource-constrained platforms such as UAVs.

2.2 Hardware-Based and Trusted Execution Security Mechanisms

In both embedded systems [21] and general-purpose platforms, the need for security has driven the development of specialized hardware and processor-integrated technologies. These mechanisms create isolated environments to protect sensitive operations from a potentially compromised host system. This section reviews two primary approaches: discrete hardware modules, such as the Trusted Platform Module (TPM) [22] and Secure Element (SE) [23], and processor-integrated architectures known as Trusted Execution Environments (TEEs).

doi:10.6342/NTU202501452

2.2.1 Hardware-Based Security Add-Ons

Hardware-based security modules are physically distinct chips that act as a root of trust, operating independently of the main processor to provide a secure vault for cryptographic operations.

The **Trusted Platform Module (TPM)**, standardized by the Trusted Computing Group (TCG), is a cryptoprocessor that provides hardware-based key storage and platform attestation by measuring system boot components [24]. The **Secure Element (SE)** is another tamper-resistant chip, common in mobile and IoT devices for applications like storing payment credentials. The GlobalPlatform standard is crucial for managing applications within an SE.

However, both TPM and SE require discrete chips, which increases cost and design complexity, making them often impractical for low-cost or highly integrated platforms such as UAVs. This limitation has driven the adoption of processor-integrated alternatives.

2.2.2 Trusted Execution Environments Within Modern Processors

A **Trusted Execution Environment (TEE)** offers such an alternative by creating a secure, isolated area within the main processor to protect the confidentiality and integrity of sensitive code and data. This approach provides strong, hardware-enforced isolation without extra components. The GlobalPlatform organization provides standard APIs to facilitate the development of portable trusted applications.

A prominent example is **ARM TrustZone** [25], which partitions the system into a Secure World (for a Trusted OS and Trusted Applications) and a Normal World (for a standard OS like Linux). A hardware-enforced Secure Monitor controls

doi:10.6342/NTU202501452

transitions between worlds, preventing the Normal World from accessing secure resources. Open-source frameworks like OP-TEE [26] enable the practical use of TrustZone for secure key storage and cryptographic tasks.

Similarly, **Intel Software Guard Extensions (SGX)** [27] allows applications to create encrypted memory regions called *enclaves.* These enclaves protect code and data even from a compromised OS or hypervisor. A key feature of SGX is remote attestation, which allows an enclave to cryptographically prove its integrity to a remote party.

In summary, TEEs provide a compelling, cost-effective solution for security. This research, therefore, adopts ARM TrustZone with OP-TEE on a Raspberry Pi. This approach offers a practical and scalable path for implementing the proposed secure RNG and cryptographic functions on a resource-constrained UAV platform without requiring additional security chips.

2.3 Security and Regulations in UAV Systems

2.3.1 General Security Challenges in UAVs

As unmanned aerial vehicles (UAVs) are increasingly deployed in logistics, surveillance, and public safety, their cybersecurity vulnerabilities have drawn significant attention. UAVs are often exposed to hostile environments and are susceptible to various threats, including GPS spoofing, signal jamming, and data interception [28]. Additionally, their limited computational and energy resources make it difficult to implement full-scale security mechanisms. Addressing these constraints requires lightweight cryptographic solutions, secure key storage, and reliable system-level protections to ensure the integrity of command and control links, mission data, and onboard firmware. These challenges underscore the need for lightweight, secure

doi:10.6342/NTU202501452

architectures like TEE-based systems that can offer isolated key storage and random number generation capabilities without requiring extra hardware.

2.3.2 Regulatory Requirements and Remote ID Implementations

In the United States, the Federal Aviation Administration (FAA) enforces RID requirements through the *Remote ID Final Rule* [29], which mandates that all commercial drones periodically broadcast identification, location, and altitude data. While the Final Rule serves as the legal basis, the ASTM F3411-22 standard [30] provides the technical foundation, specifying protocol formats, broadcast mechanisms (such as Wi-Fi NAN or Bluetooth), and optional digital signature features for securing RID data.

Japan adopts a more lightweight approach by mandating the use of symmetric-key cryptography. Specifically, UAVs are required to use AES-128-CCM to compute a Message Authentication Code (MAC) for RID message integrity [31]. This design reduces computational burden and is suitable for embedded platforms with limited resources. Taiwan, in the future, is expected to adopt a similar policy, as outlined in the draft amendment to Article 99-10 of the Civil Aviation Act [32], which mandates the integration of RID modules capable of real-time broadcasting and symmetric authentication.

Regardless of whether asymmetric or symmetric cryptography is used, secure key storage remains the cornerstone of RID security. Traditionally, secret keys are stored in hardware-based secure elements such as Trusted Platform Modules (TPMs) or Secure Elements (SEs). However, these solutions require additional hardware integration and cost, limiting their applicability in cost-sensitive or highly integrated platforms like drones or IoT devices. Recent research has thus explored using ARM TrustZone to implement Trusted Execution Environment (TEE)-based

doi:10.6342/NTU202501452

key protection, providing similar isolation guarantees within the System-on-Chip (SoC) and eliminating the need for dedicated secure hardware.

Such an approach aligns with the regulatory trajectory of countries like Japan and Repblic of China, Taiwan, which emphasize low-power, embedded-friendly security designs in future RID deployments. To meet these emerging mandates and scalability challenges, this work proposes a TEE-based dual-domain RNG architecture that not only enhances the security of random number generation but also has the potential to integrate secure key management within SoC platforms—offering practical value for implementing RID in compliance with regional regulations.

doi:10.6342/NTU202501452

Chapter 3

Methodology

This chapter details the methodology for designing and implementing the hybrid TEE-REE combinatorial random number generator. The discussion begins with the core of the framework: four distinct approaches for mixing entropy from the TEE and REE domains. Following the description of these techniques, the chapter establishes the theoretical groundwork for the design, including the rationale for adopting a hybrid model with partial trust. To provide a clear security context, the specific trust assumptions and the adversarial threat model are then formally defined. The chapter concludes by presenting how the dual-domain entropy sources are integrated and securely processed to generate the final random output.

3.1 Combinatorial RNG Design

This section describes four mixing approaches used to construct combinatorial random number generator (RNG): XOR operation, SHA-256 hashing, AES encryption, and Chaining with counter-based enhancement. The main objective of these techniques is to maintain high statistical randomness and ensure the unpredictability of the final output. In addition to the construction, each mixing method is individually analyzed to assess its strengths and potential weakness.

In the design, both RNGs independently generate entropy through either

doi:10.6342/NTU202501452

PRNG or TRNG. The mixing process is assumed to take 256-bit random values from both sides and generates a 256-bit output.

3.1.1 XOR Mixing and Corresponding Properties

This method directly mixes the two entropy sources using bit-wise XOR operator. Let R_A and R_B be two original source, R_OUT be the mixing result and $\oplus$ be the bitwise XOR operation. The mixing method is listed in Equation 3.1.

$$R_\mathrm{OUT} = R_\mathrm{A} \oplus R_\mathrm{B} \tag{3.1}$$

XOR mixing is the most lightweight method among the four. Its key strength lies in preserving output unpredictability even when one of the entropy sources is partially compromised. Specifically, if one input source provides uniformly random bits and is independent from the other, the XOR result remains uniformly distributed.

Morever, observe that as long as one entropy source is uniformly random and independent from that of the other one, XOR effectively neutralizes bias. Consequently, the final output remains statistically unbiased even if the one entropy is skewed or adversarially influenced, as showcased below.

Let $X_n, Y_n \in \{0, 1\}$ be independent random bits from TEE and REE respectively. Suppose X_n is uniformly distributed, i.e., $\mathbb{P}(X_n = 1) = 0.5$, and Y_n is an arbitrary bit with bias $w \in [0, 1]$, i.e., $\mathbb{P}(Y_n = 1) = w$. Define $Z_n = X_n \oplus Y_n$. Then the output Z_n is uniformly distributed:

$$\mathbb{P}(Z_n = 1) = 0.5 \cdot (1 - w) + 0.5 \cdot w = 0.5$$

doi:10.6342/NTU202501452

Notably, since XOR is a linear and reversible operation, knowledge of one of the sources and the mixed result allows the adversary to trivially recover the TEE's RNG output by computing

$$R_{TEE} = R_{OUT} \oplus R_{REE}.$$

3.1.2 SHA-256 Concatenation Mixing

This method concatenates the two sources into a 512-bit input, and then applies SHA-256 to extract and compress the randomness. Let R_{A} and R_{B} be two original source, R_{OUT} be the mixing result, $||$ be the bitwise concatenation operator, and SHA256(X) be the SHA-256 hash function applied to input X, outputs 256 bits. The mixing method is listed in Equation 3.2.

$$R_{\text{OUT}} = \text{SHA256}(R_{\text{A}} \,||\, R_{\text{B}}) \tag{3.2}$$

SHA-256, as one of the most widely used hashing function, is commonly adopted in entropy extractions and post-processing tasks due to its *avalanche effect* [33] and therefore *preimage resistance* [34]. By concatenating R_{TEE} and R_{REE}, the 512-bit input is compressed into a 256-bit output.

Although SHA-256 is widely regarded as a cryptographically secure hash function, using it as a deterministic extractor has known limitations. When one input is adversarially chosen or fixed, and the output is observable, a single SHA-256 operation may not suffice to hide structure or correlations in the input. Prior works [35] have shown that deterministic extractors, such as fixed hash functions, fail to guarantee entropy extraction under such conditions.

doi:10.6342/NTU202501452

3.1.3 AES Encryption-Based Mixing (ECB Mode)

This method uses AES-256 in Electronic Codebook mode (ECB mode), where the 256-bit random number from one source serves as the encryption key. The 256-bit random number from the other one is used directly as the plaintext input. The resulting 256-bit ciphertext is then the mixed output. Let R_{A} and R_{B} be two original sources, R_{OUT} be the mixing result, and $\mathrm{AES256}_K(P)$ be the AES-256 encryption in ECB mode with 256-bit key K on 256-bit plaintext P. The mixing method is listed in Equation 3.3.

$$R_{\mathrm{OUT}} = \mathrm{AES}_{R_{\mathrm{A}}}(R_{\mathrm{B}}) \tag{3.3}$$

The AES-256 mixing method uses one source as the encryption key and the other source as the plaintext. The security of AES under ECB ensures that even small changes in plaintext or key yield entirely different ciphertext outputs. This offers *strong diffusion* [36], with the security grounded in block cipher unpredictability. Furthermore, the AES-based mixing benefits from the *whitening effect* [37] of the block cipher, where input patterns and entropy bias are diffused across the entire ciphertext as well, making the output indistinguishable from uniform random data under standard assumptions of AES.

Notably, even when the encryption key remains confidential and inaccessible, the AES function is inherently deterministic. This implies that if one input to the function is fully controllable and the output is observable, an attacker can infer the mapping between known plaintexts and ciphertexts under a fixed key. Such structural determinism fails to amplify entropy and may result in predictable patterns. Consequently, from an entropy-combination standpoint, AES-based mixing may be

doi:10.6342/NTU202501452

insufficient to obscure correlations or mitigate predictability when one input is weak or adversarially influenced.

3.1.4 Chaining with CTR-based Enhancement

This method applies two rounds of chained SHA-256 operations with counters. Let R_{A} and R_{B} be two original sources, R_{OUT} be the mixing result, $||$ be the bit-wise concatenation operator, $\oplus$ be the bit-wise XOR operator, $C_1, C_2 \in \{0,1\}^{128}$ be 128-bit counter values that increases every operation, and SHA256(X) be the SHA-256 hash function applied to input X. The mixing method is listed in Equation 3.4.

$$\left.\begin{aligned} S_0 &= R_{\text{A}} \oplus R_{\text{B}} \\ H_1 &= \text{SHA256}(S_0 \parallel C_1) \\ S_1 &= H_1 \\ H_2 &= \text{SHA256}(S_1 \parallel C_2) \\ R_{\text{OUT}} &= H_2 \end{aligned}\right\} \tag{3.4}$$

To enhance mixing quality, this method applies a two-stage SHA-256 construction with counter-based domain separation. Specifically, two entropy inputs are first combined via bitwise XOR to produce an intermediate state S_1. A first hash H_1 is then computed using a 128-bit counter C_1 as a domain separator, followed by a second intermediate state S_2, a second 128-bit counter C_2, and thus calculate the hash as the final output.

This two-stage construction provides stronger resilience against structural weaknesses in either input source. By layering the hash function and using evolving domain separators, it prevents collisions, reinforces input decorrelation, and mitigates the risk of output predictability—even when facing adversarially influenced

doi:10.6342/NTU202501452

or low-entropy inputs. While more computationally intensive than single-hash approaches, this design offers a higher assurance of entropy diffusion and output unpredictability under repeated or adaptive attacks. This layered approach follows the principle of *domain separation* [38], a technique widely used in cryptographic constructions to prevent cross-domain interference.

3.1.5 Summary of Mixing Methods

Table 3.1 provides a summary of the four proposed mixing methods and their mathematical formulations,

Table 3.1: Summary of Mixing Methods

Method	**Formula**
XOR	$R_{\text{OUT}} = R_A \oplus R_B$
SHA-256 Concat.	$R_{\text{OUT}} = \text{SHA256}(R_A \parallel R_B)$
AES Encryption	$R_{\text{OUT}} = \text{AES}_{R_A}(R_B)$
Chaining with CTR	$S_0 = R_A \oplus R_B$ $H_1 = \text{SHA256}(S_0 \parallel C_1)$ $S_1 = H_1$ $H_2 = \text{SHA256}(S_1 \parallel C_2)$ $R_{\text{OUT}} = H_2$

where

- $R_{\text{A}} \in \{0,1\}^{256}$ is the 256-bit random number generated from source A.
- $R_{\text{B}} \in \{0,1\}^{256}$ is the 256-bit random number generated from source B.
- $R_{\text{OUT}} \in \{0,1\}^{256}$ is the output of the mixing process.
- $||$ is the bit-wise concatenation operator.
- $\oplus$ is the bit-wise XOR operation.

doi:10.6342/NTU202501452

- SHA256(X) is the SHA-256 hash function applied to input X.
- AES256$_K(P)$ is the AES-ECB-256 encryption with key K on plaintext P.
- $C_1, C_2 \in \{0,1\}^{128}$ are two 128-bit counter values, such as `0x...01` and `0x...02`, with increment after every operation.

3.2 Rationale for Partial Trust in RNGs

Having established the combinatorial RNG design, a fundamental architectural question arises: why not place both random number generators within the Trusted Execution Environment (TEE) for maximum security? The decision to distribute the RNGs across both the TEE and the Rich Execution Environment (REE) is a deliberate choice, grounded in established security principles and practical trade-offs, especially for resource-constrained platforms like UAVs. This rationale is based on three core reasons:

- **Avoiding a Single Point of Failure:** The primary motivation for this hybrid architecture is to prevent a single point of failure. By deploying RNGs in two separate domains, the system is designed to withstand a compromise of the REE. Even if an attacker gains full control over the REE-based RNG, the final random output remains unpredictable due to the secure mixing with the hardware-protected entropy source within the TEE, thus safeguarding critical operations.
- **Adherence to TCB Minimization:** A core principle of secure system design is the minimization of the Trusted Computing Base (TCB). The TCB encompasses all components of a system that must be trusted to uphold the security policy. By placing only one RNG and the mixing logic inside the

doi:10.6342/NTU202501452

TEE, we keep the TCB smaller and less complex. This reduces the potential attack surface within the secure world, simplifies formal verification, and conserves scarce secure resources, which is a critical consideration on platforms with limited secure memory.

- **A Flexible Trade-off between Performance and Security:** This dual-domain architecture offers a practical trade-off between performance and security. Non-security-critical applications running in the REE that require random numbers for tasks like statistical sampling can directly and efficiently use the REE's RNG. This avoids the overhead of context switching into the TEE and prevents the TEE's high-quality entropy pool from being unnecessarily depleted by non-critical requests. This allows the TEE's resources to be preserved for operations where cryptographic security is paramount

3.3 Trust Assumptions and Threat Model

To ground the design rationale established in the previous section, this section clarifies the system's security assumptions and threat model. The approach is built on the premise that securing a single entropy source—through confinement within the TEE—can suffice to ensure strong randomness guarantees, even when the other source is potentially compromised.

The following outlines the trust boundaries and adversarial capabilities assumed throughout this work.

- *TEE is trusted and secure*: The secure core of the Trusted Execution Environment (TEE) and its internal Trusted Applications (TAs) are assumed to operate within a hardware-isolated environment, free from known vulnerabilities or backdoors. Initialization procedures, memory isolation, and access control

doi:10.6342/NTU202501452

are enforced by the processor and conform to the security levels specified by GlobalPlatform.

- *SMC mechanism and shared memory channel are access-controlled*: The Secure Monitor Call (SMC) interface and the shared memory channel used to transfer data from REE to TEE are assumed to be properly configured with appropriate permission control, preventing unauthorized access or tampering.
- *The system-on-chip (SoC) hardware platform is trustworthy*: The TRNG hardware module, memory subsystem, and world-switching mechanism are assumed to be correctly implemented.

In contrast, the adversary is assumed to possess the following capabilities:

- *Full control over REE*: The attacker may gain full control of the REE (e.g., Linux user space or kernel) through malware, kernel exploits, or system backdoors, and can manipulate the RNG outputs generated in the REE.
- *No control over TEE*: The attacker cannot compromise the TEE, and has no access to internal program logic, memory, or execution state within the secure world.
- *Passive observation of communication*: The attacker can intercept data transmitted via shared memory between REE and TEE (e.g., intermediate mixed output), but cannot deduce internal state or private keys within the TEE.
- *Attempts to predict final RNG output*: The attacker aims to infer the final random output from partially known or compromised entropy sources, thereby undermining the cryptographic security of operations such as key generation and digital signatures.

Under these assumptions, the proposed system aims to achieve the following security objectives:

- *Resilience against compromised entropy sources*: Even if the REE entropy source is entirely compromised, the final output remains highly unpredictable.
- *Preservation of statistical randomness*: The mixing mechanism must ensure statistically random outputs that pass standard randomness test suites.

Beyond software-based threats, the threat model also considers the risk of Side-Channel Attacks (SCA). An adversary might attempt to infer secrets within the TEE by observing physical properties such as power consumption, electromagnetic emissions, or timing variations. This research mitigates SCA risks through its distributed design; by not concentrating all random number generation logic within the TEE, the secure world becomes a less viable single target for observation, making it harder for an attacker to build a stable predictive model. Furthermore, the choice of SHA-256 for mixing, with its fixed computational structure and strong avalanche effect, helps resist timing and power analysis attacks.

3.4 Protecting RNGs by TEE

This study proposes a dual-source random number generation architecture that isolates entropy sources across different security domains. The design separates the entropy sources between the Rich Execution Environment (REE) and the Trusted Execution Environment (TEE) to enhance both randomness quality and system-level security.

As shown in Figure 3.1, the REE uses either built-in or software-implemented Pseudorandom Number Generators (PRNGs) to generate random data. This data

doi:10.6342/NTU202501452

is temporarily stored in a designated shared memory region and passed to the TEE via a Secure Monitor Call (SMC), which triggers a world switch. The processing is then handled by a Trusted Application (TA) within the TEE.

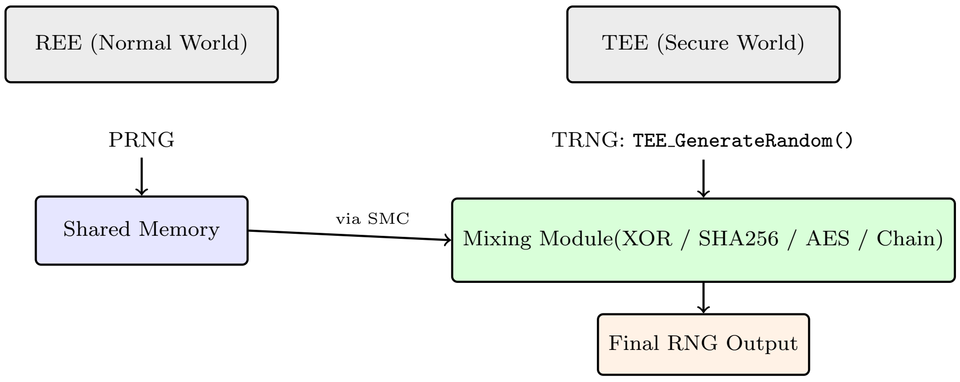


Figure 3.1: Design of the dual-source RNGs.

The combined entropy sources are then processed by a mixing module employing one of four fusion approaches: **bitwise XOR**, **SHA-256 hashing**, **AES encryption**, or a **chaining-based construction**. The resulting output can either be retained within the secure domain for sensitive cryptographic use, or exported to the general environment for broader system use, depending on application needs. Figure 3.2 summarizes the runtime procedure, showing how entropy is retrieved from both domains and processed using a selected mixing function $\mathcal{F}$. The injection interval n is configurable to balance security and speed.

```
   Input: $R_{\text{TEE}}$, $R_{\text{REE}}$, mixing function $\mathcal{F} \in$ {XOR, SHA-256, AES, Chaining}
   Output: $R_{\text{OUT}}$
1: Condition: Entropy injection is triggered every $n$ iterations (configurable)
2: $R_{\text{TEE}} \leftarrow$ 256-bit entropy from TEE
3: $R_{\text{REE}} \leftarrow$ 256-bit entropy from REE
4: $R_{\text{OUT}} \leftarrow \mathcal{F}(R_{\text{TEE}}, R_{\text{REE}})$
5: return $R_{\text{OUT}}$
```

Figure 3.2: Dual-Domain Entropy Mixing Algorithm

In this architecture, the two random sources are explicitly mapped to their respective execution domains: R_{TEE} is generated using the TEE's internal true random number generator via `TEE_GenerateRandom()`, while R_{REE} originates from a software-based PRNG operating in the untrusted REE. These entropy inputs are then combined within the TEE using one of the four mixing approaches described in Section 3.1.

The mappings of the generic variables R_A and R_B used in earlier descriptions now take on concrete roles as R_{TEE} and R_{REE}, respectively. For example, in the AES-based mixing method, R_{TEE} is used as the encryption key, ensuring that the most sensitive input remains confined to the secure world, while R_{REE} serves as the plaintext. In the XOR and hash-based constructions, the mixing occurs entirely within the TEE, with R_{REE} transferred through a controlled shared memory interface.

By enforcing all mixing operations within the TEE, the architecture ensures that the critical entropy path—including the internal TRNG and its post-processing—remains inaccessible to potential REE-based adversaries. The design leverages the asymmetric trust assumption (Section 3.3), where only one entropy source is considered secure. The effectiveness and trade-offs of each method in this context are summarized in Table 3.2, highlighting the balance between computational cost, entropy diffusion, and resistance to input manipulation.

This design not only preserves the security of the trusted entropy input but also increases the overall system quality by allowing entropy mixing to tolerate compromised or low-quality inputs from the untrusted domain.

Leftover Hash Lemma (Formal Version)

doi:10.6342/NTU202501452

Table 3.2: Comparison of Mixing Methods

Method	**Strength**	**Computation Cost**	**Potential Weakness**
XOR	Lightweight; Bias Neutralization	Very low (1 XOR)	Weak for correlated inputs; Invertible under adversarial assumption
SHA-256	Nonlinear; One-way compression; Preimage resistance	1 SHA-256	Vulnerable to adversarially controlled input
AES-256	Nonlinear; Diffusion; Whitening effect	Requires AES encryption	Vulnerable to adversarially controlled input
Chaining with Counter	Nonlinear; Domain separation; Strong against pattern repetition and partial predictability	2x SHA-256 with counter + 1 XOR	Relatively computationally expensive

Let X be a random variable over $\{0,1\}^n$ with min-entropy k and let $m > 0$. Let $h : \mathcal{S}\times\{0,1\}^n \to \{0,1\}^m$ be a 2-universal hash function. If

$$m \leq k - 2\log\left(\frac{1}{\varepsilon}\right),$$

then for S uniform over $\mathcal{S}$ and independent of X, we have:

$$\delta\left((h(S,X),S),(U,S)\right) \leq \varepsilon,$$

where U is uniform over $\{0,1\}^m$ and independent of S.

The *min-entropy* of X is defined as:

$$H_\infty(X) = -\log\max_x \Pr[X = x],$$

doi:10.6342/NTU202501452

The *statistical distance* between distributions X and Y is defined as:

$$0 \leq \delta(X, Y) = \frac{1}{2} \sum_{v} |\Pr[X = v] - \Pr[Y = v]| \leq 1.$$

Leftover Hash Lemma (Formal Version)

Let X be a random variable over $\mathcal{X}$ and let $m > 0$. Let $h : \mathcal{S} \times \mathcal{X} \to \{0,1\}^m$ be a 2-universal hash function. If

$$m \leq H_\infty(X) - 2\log\left(\frac{1}{\varepsilon}\right),$$

then for S uniform over $\mathcal{S}$ and independent of X, we have:

$$\delta\left((h(S, X), S), (U, S)\right) \leq \varepsilon,$$

where U is uniform over $\{0,1\}^m$ and independent of S.

The *min-entropy* of X is defined as:

$$H_\infty(X) = -\log \max_x \Pr[X = x],$$

which measures the difficulty of guessing X (i.e., how unpredictable X is).

The *statistical distance* between distributions X and Y is defined as:

$$0 \leq \delta(X, Y) = \frac{1}{2} \sum_{v} |\Pr[X = v] - \Pr[Y = v]| \leq 1.$$

3.5 Theoretical Foundations and Security Analysis

This section aims to elaborate on the theoretical foundations of the proposed combinatorial RNG design, clarifying the concepts of entropy, mixing, and their relationship with established cryptographic principles.

doi:10.6342/NTU202501452

3.5.1 The Role of Entropy and Mixing

In the field of random number generator, entropy is the fundamental metric for the unpredictability of a noise source. However, it must be emphasized that the mixing methods employed in this study do **not** create new entropy. Instead, mixing is a form of post-processing. Its purpose is to refine and transform the raw data from one or more entropy sources using cryptographic functions, thereby **enhancing their statistical randomness and increasing resilience against known weaknesses or attacks**.

- **XOR**: The primary function of the XOR operation is to *neutralize bias.* As long as one of the input streams is uniformly random and independent, the output will remain statistically unbiased.
- **SHA-256 / AES**: By leveraging their non-linearity, avalanche effect, and diffusion properties, these cryptographic primitives can *decorrelate* potential structural weaknesses present in the input data.
- **CHAIN**: Through multiple rounds of hashing and the use of counters, this method provides the deepest layered protection. Its multi-stage construction offers the strongest resistance against bias and repetitive patterns.

According to the guidelines in NIST SP 800-90B, the direct measurement of entropy is applicable to raw, unconditioned noise sources for health-check purposes. The focus of this research, however, is to evaluate the quality of the output **after combinatorial post-processing**. Therefore, employing the NIST SP 800-22 statistical test suite is the more appropriate standard for measuring the final output's quality.

doi:10.6342/NTU202501452

3.5.2 Relationship with the Leftover Hash Lemma (LHL)

The design philosophy of the CHAIN method is inspired by the **Leftover Hash Lemma (LHL)** [39, 40][1] from cryptography. The LHL provides a theoretical guarantee that a source with sufficient min-entropy, when processed by a hash function randomly chosen from a 2-universal family, will yield an output that is statistically indistinguishable from a uniform random string.

From a functional perspective, the CHAIN method, which uses a construction like $H_c(m) = \text{SHA256}(m \parallel c)$, can be viewed as creating a **family of hash functions** indexed by the counter c. Each new value of the counter effectively selects a new, specific function from this constructed family.

However, it is critical to distinguish this constructed family from the strictly-defined, a-priori random hash family required by the LHL. The CHAIN method is not a strict implementation of the LHL due to two key points:

- The underlying function is always the **fixed and public SHA-256 algorithm**. We are not randomly selecting from a family of different algorithms, but rather deterministically altering the input to a single algorithm.
- Consequently, this constructed family of functions, indexed by the counter, **is not proven to be 2-universal**. The 2-universal property is a specific mathematical condition on collision probabilities that standard cryptographic hashes like SHA-256 are not explicitly designed to satisfy in this manner.

Despite this theoretical gap, this approach holds strong practical merit. The counter serves as a powerful **domain separation** technique. By ensuring that each

[1]For a formal statement of the Leftover Hash Lemma, please refer to Appendix A.

doi:10.6342/NTU202501452

invocation operates in a unique "domain" (defined by the counter), it prevents cross-call correlations and effectively simulates the security benefits of using a new random function for each operation. This practical application, combined with the proven strengths of SHA-256 such as its avalanche effect and collision resistance, allows the CHAIN method to achieve a better randomizing diffusion effect, approximating the security goals of the LHL in a real-world setting.

This hybrid entropy model is consistent with resilient RNG frameworks such as Fortuna [41] and NIST SP 800-90B [4], which recommend diversity in entropy sources to withstand partial compromise. Additional support comes from studies on entropy extraction [42] and heterogeneous entropy fusion [43], which show that unpredictability can be preserved even when integrating inputs across different trust boundaries, provided at least one source remains secure. Hence, both cryptographic theory and established RNG architectures suggest that deploying only a single RNG inside the TEE can suffice for secure entropy generation.

doi:10.6342/NTU202501452

Chapter 4

RNG Combination Results and Analysis

This chapter evaluates the proposed TEE/REE combinatorial RNG through statistical randomness tests and performance analysis, covering both individual and mixed designs.

4.1 Testing Tools Overview

To evaluate the statistical quality of random number generators (RNGs), several established test suites have been developed. This section compares broadly used toolsets—NIST SP 800-22, Dieharder, and TestU01—in terms of their statistical rigor, computational requirements, and applicability to the purpose of this work.

NIST SP 800-22. The NIST SP 800-22 test suite [44]comprises 15 statistical tests designed to detect a wide range of non-random patterns in binary sequences. It is widely adopted in cryptographic validation and hardware RNG certifications. The suite is relatively lightweight in computational cost, making it suitable for moderate-scale batch testing, and provides two key evaluation metrics: per-sequence p-values and the uniformity of p-value distributions.

doi:10.6342/NTU202501452

Dieharder. Dieharder [45] is an extension of George Marsaglia's Diehard battery and offers a more diverse set of statistical tests. Although useful for quick validation, the test suite includes some legacy tests with limited sensitivity to certain modern RNG flaws. Furthermore, its documentation and interpretation criteria are less formalized than NIST's.

TestU01. TestU01 [46] provides the most comprehensive collection of tests, especially through its "BigCrush" battery. It is considered the gold standard in theoretical RNG evaluation. However, its runtime cost is prohibitively high—BigCrush can take several hours to complete and is overkill for most application-driven evaluation settings. Moreover, it requires C-language integration, which limits accessibility for certain experimental environments.

Since the goal of this study is to evaluate mixed entropy sources from both TEE and REE domains with 128MB of output per generator, NIST SP 800-22 was chosen as the primary test suite for this study. It offers a balanced tradeoff between computational efficiency and statistical coverage, and is officially recommended for cryptographic validation. [4] In particular, its per-sequence p-value evaluation enables us to assess local randomness consistency within each 1MB block, while the p-value uniformity test across multiple sequences helps assess the statistical consistency of different entropy mixing approaches. These features make NIST SP 800-22 a justifiable and practical choice for the evaluation framework.

4.2 The NIST SP 800-22 Test Suite

The NIST SP 800-22 test suite is a standardized statistical toolkit developed by the U.S. National Institute of Standards and Technology (NIST) for evaluating the quality of random number generators. It acts as a comprehensive "randomness

doi:10.6342/NTU202501452

health check," ensuring that the output from a random number generator (RNG) is statistically indistinguishable from true randomness (as defined in section 4.2.1).

Such statistical indistinguishability is especially critical in domains like cryptography, secure communications, simulations, and safety-critical systems, where predictable patterns can lead to vulnerabilities. The test suite examines the bitstream using a check of 15 statistical tests, each targeting different aspects of randomness such as balance, independence, and unpredictability, etc.

4.2.1 Terminology and Definitions

The following are the terms and symbols used in this context:

- **Bitstream**: A sequence of binary digits (0's and 1's) output by an RNG. Throughout this study, each RNG produces 128 MB of output data, which is divided into 100 blocks of bitstream each of approximately 1.3 MB for using the NIST SP 800-22 test suite.

- **True Randomness**: A bitstream generated by an independent and identically distributed (i.i.d.) Bernoulli process with probability $p = 0.5$. Formally, each bit follows the random variable $X_i \sim \text{Bernoulli}(0.5)$ and thus all bits are unbiased, no correlation, and no deterministic pattern.

- **Test Statistic**: A computed numerical value (e.g., number of 1's, length of longest run of 1's) used to assess randomness.

- **P-value**: The probability of observing a given sequence under the assumption that the generator is truly random, with respect to the null hypothesis: "The tested sequence is truly random".

doi:10.6342/NTU202501452

- **Significance Level** α: A threshold for rejecting the null hypothesis of randomness. The documentation suggests a feasible interval of [0.001, 0.01], and it was set to 0.01, which is considered relatively strict.
- **Pass Ratio**: The percentage of test sequences that yield P-values greater than or equal to α.
- **Overall Pass Ratio (proposed)**: A custom-defined index that aggregates all individual sub-test results to compute the overall success rate across the entire NIST SP 800-22 suite. It provides a concise metric to compare the general randomness quality of different generators.

4.2.2 Pass/Fail Criteria in SP 800-22

For each sub-test applied to m independent sequences (commonly $m = 100$), SP 800-22 defines the following pass criteria:

1. **Proportion of Passing Sequences**: The number of sequences with P-values $\geq \alpha$ should lie within the interval:
$$\hat{p} \in \left[\hat{p}_0 - 3\sqrt{\frac{\hat{p}_0(1-\hat{p}_0)}{m}}, \hat{p}_0 + 3\sqrt{\frac{\hat{p}_0(1-\hat{p}_0)}{m}}\right],$$
where $\hat{p}_0 = 1 - \alpha$ (typically 0.99). For $m = 100$, this implies at least 96 passes are required. In this evaluation, 100 blocks of random data were generated per RNG variant, thus $m = 100$. According to the NIST criteria, the pass ratio must be at least 96 out of 100 to satisfy the minimum threshold for randomness.

2. **Uniformity of P-value Distribution**: The P-values from the m sequences should be uniformly distributed over the interval [0,1]. This is verified using a chi-squared test with 10 bins. The chi-squared value is computed and

doi:10.6342/NTU202501452

converted into a *P-value of P-values*:

$$P_{\text{uniform}} = \text{igamc}\left(\frac{9}{2}, \frac{\chi^2}{2}\right)$$

A P_{uniform} value less than 0.0001 indicates non-uniformity and a failure of the test.

4.2.3 Example: The Frequency (Monobit) Test

Before presenting the results, this section provides a detailed walk-through of the Frequency test (also known as the Monobit test), one of the most basic tests in the NIST SP 800-22 suite. This test serves as an intuitive illustration of how NIST assesses bit-level balance in a random sequence. It evaluates whether the number of ones and zeros are approximately equal in a given bitstream, which is a fundamental expectation of any cryptographically secure RNG.

Procedure:

1. Let n be the length of the bitstream. Convert each bit $x_i \in \{0, 1\}$ to $y_i \in \{-1, +1\}$ via $y_i = 2x_i - 1$.
2. Compute the test statistic:
$$S_n = \sum_{i=1}^{n} y_i$$
3. Normalize the statistic:
$$s = \frac{S_n}{\sqrt{n}}$$
4. Calculate the P-value using the complementary error function:
$$P = \text{erfc}\left(\frac{|s|}{\sqrt{2}}\right)$$
5. Repeat the above for m sequences and evaluate using NIST's two-pass criteria (proportion and uniformity).

doi:10.6342/NTU202501452

An Illustrative Case:

Consider a binary sequence of length $n = 1{,}000{,}000$ with 508,000 ones and 492,000 zeros. Then:

$$S_n = 508{,}000 - 492{,}000 = 16{,}000$$

$$s = \frac{16{,}000}{\sqrt{1{,}000{,}000}} = \frac{16{,}000}{1000} = 16.0$$

$$P = \text{erfc}\left(\frac{16.0}{\sqrt{2}}\right) = \text{erfc}(11.31) \approx 1.6 \times 10^{-58}$$

Since $P < 0.01$, this sequence failed the Frequency Test.

In contrast, if a bitstream contains 500,200 ones and 499,800 zeros, then:

$$S_n = 400, \quad s = \frac{400}{1000} = 0.4$$

$$P = \text{erfc}\left(\frac{0.4}{\sqrt{2}}\right) \approx \text{erfc}(0.2828) \approx 0.752$$

Since $P \geq 0.01$, this sequence passed the Frequency Test. (But not yet the uniformity check)

4.3 Testing Results

Table 4.1 provides a summary of the overall pass ratio for each random number generator (RNG) tested in this study. It aggregates the total number of subtests passed across all categories in the NIST SP 800-22 suite, giving a high-level view of each RNG's statistical reliability. Table 4.2 presents a detailed breakdown of the pass ratios for individual test categories, which shows how each RNG performed on specific statistical tests. Table 4.4 reports the uniformity of p-value distribution for each test category across all RNGs. This evaluates whether the p-values generated by each RNG follow a uniform distribution on the interval $[0, 1)$, as expected under

doi:10.6342/NTU202501452

true randomness. A higher uniformity p-value indicates better conformity to ideal statistical behavior. HELLO

Table 4.1: Overall Pass Ratios(%) from NIST SP 800-22 Statistical Test Suite Across Different RNG Mixing Strategies

RNG	Passed Subtests	Total Subtests	Pass Ratio (%)
CHAIN	**188**	**188**	**100.00**
XOR	188	188	100.00
AES	187	188	99.47
SHA256	186	189	98.41
TEE	187	188	99.47
REE	189	190	99.47

4.3.1 Baseline RNG Performance

To establish a benchmark for evaluating the performance of combinatorial Random Number Generators (RNGs), this section first analyzes the randomness quality of independent RNGs from the Rich Execution Environment (REE) and the Trusted Execution Environment (TEE). All evaluations employ the NIST SP 800-22 statistical test suite.

- **REE RNG**: The random number output generated in the REE performed well, passing 189 out of a total of 190 subtests, achieving an **overall pass ratio of 99.47%**. Its only item that did not meet the standard was the Universal test (proportion of 94/100, below the NIST recommended threshold of approximately 96%). This result indicates that while the entropy source

doi:10.6342/NTU202501452

in the REE environment can generally produce acceptable random sequences, slight weaknesses may exist in specific statistical properties.

- **TEE RNG**: The output generated using the TEE's hardware isolation features and its built-in TEE_GenerateRandom() function successfully passed 187 out of 188 subtests, also yielding an **overall pass ratio of 99.47%**. Its single subtest that did not meet the criteria appeared in one instance of the NonOverlappingTemplate test category (proportion of 95/100, below the approximate 96% threshold). This data confirms that the RNG in TEE can provide high-quality random numbers, with its performance being comparable to the REE RNG's overall pass ratio in this testing round. However, it also shows that even hardware-protected entropy sources might exist some statistical fluctuations in specific subtests.

4.3.2 Performance of Combinatorial RNGs

This section details the randomness performance evaluation of four distinct mixing approaches for combinatorial RNGs. All combinatorial methods fuse 256-bit random numbers from both TEE and REE as input and produce a 256-bit final output. The NIST SP 800-22 test results are summarized as follows:

- **Chaining with CTR (CHAIN)**: This method exhibited the most outstanding performance among all tested combinatorial approaches. Out of a total of 188 subtests, the CHAIN method achieved a **perfect pass ratio of 100% (188/188)**. Whether in basic frequency and runs tests or more stringent analyses like linear complexity and serial entropy tests, the CHAIN method demonstrated impeccable statistical randomness.

doi:10.6342/NTU202501452

- **XOR Mixing (XOR)**: The XOR mixing method also performed excellently in the NIST SP 800-22 statistical tests, likewise achieving a **100% pass ratio (188/188)**, with all subtests successfully passed. This result indicates that, under an ideal statistical model, a simple bitwise XOR operation can effectively combine the randomness of two independent entropy sources. However, it is crucial to emphasize that despite its perfect statistical performance, the security of the XOR mixing method has been identified as inherently risky in Section 3.3.1. Due to the linear and reversible nature of the XOR operation, an attacker with full control over the REE environment and the ability to observe the mixed output could potentially deduce the TEE-side random numbers, thereby compromising the overall system security. Therefore, while its statistical performance is superior, its application in security-critical scenarios requires careful consideration.

- **AES Encryption-Based Mixing (AES)**: The mixing method based on AES-256 (ECB mode) passed 187 out of 188 subtests, resulting in an **overall pass ratio of 99.47%**. Its performance was comparable to that of the single TEE RNG or REE RNG, failing only one subtest within the Universal category (proportion 95/100). This result suggests that the strong diffusion properties of AES encryption contribute to maintaining the quality of random sequences.

- **SHA-256 Concatenation (SHA256)**: This method was relatively weak compared to the other three combinatorial methods. It passed 186 out of 189 subtests, yielding an **overall pass ratio of 98.41%**. Specifically, the SHA-256 mixing failed to meet NIST's passing criteria in one subtest of the Cumulative Sums test (proportion 95/100) and two subtests within the NonOverlappingTemplate category (proportions both 95/100). This might imply that

doi:10.6342/NTU202501452

merely using SHA-256 for concatenation and compression may, in some cases, be insufficient to entirely eliminate or homogenize potential statistical characteristics or weaknesses from the two entropy sources, rendering its reliability slightly inferior to other combinatorial methods in this evaluation.

Clearly, the **Chaining with CTR (CHAIN) method achieved the overall best performance among the selected methods in terms of randomness quality**. In contrast, although XOR mixing also achieved perfect scores in statistical tests, its known security vulnerabilities make it not the best choice for all scenarios. AES mixing reached fair performance comparable to baseline RNGs, while SHA-256 concatenation showed some potential statistical weaknesses.

4.3.3 Statistical Performance and Uniformity of Mixing Approaches

This section analyzes the four mixing approaches (CHAIN, XOR, AES, and SHA-256) based on their NIST SP 800-22 test suite performance, focusing on overall pass ratios, behavior in specific test categories, and the uniformity of P-value distributions. Data is shown from Table 4.1, 4.2, and 4.4, respectively.

Overall Pass Ratios and Specific Test Performance: As shown in Table 4.1, CHAIN and XOR achieved perfect 100% overall pass ratios. AES mixing followed at 99.47%, while SHA-256 concatenation was the lowest among combinatorial methods at 98.41%. A closer examination of Table 4.2 reveals that both CHAIN and XOR passed all subtests within each of the 15 major NIST test categories. AES mixing failed Universal test (95/100). SHA-256 exhibited weaknesses in Cumulative Sums (one subtest at 95/100) and NonOverlappingTemplate tests (two subtests at 95/100). On the other side, the baseline REE RNG failed the Universal test (94/100) and the TEE RNG failed one NonOverlappingTemplate subtest

doi:10.6342/NTU202501452

(95/100). Both CHAIN and XOR successfully passed these specific tests where baseline or other combinatorial methods faltered, demonstrating their effectiveness in overcoming such statistical flaws.

Uniformity of P-value Distributions: The uniformity of P-value distributions, assessed by a chi-squared test (results in Table 4.4), is crucial for validating the reliability of the test outcomes. For all evaluated RNGs (baseline and combinatorial), the P-values for uniformity in every test category were well above the critical threshold of 0.0001 (e.g., the lowest observed was 0.0009 for CHAIN in RandomExcursionsVariant Test). This confirms that the P-value distributions were uniform, lending confidence to the reported pass/fail proportions.

Based on the comprehensive NIST SP 800-22 evaluation, one can find:

1. The **CHAIN** method consistently demonstrated superior statistical randomness, achieving a perfect pass ratio across all subtests and maintaining good P-value uniformity. This positions it as the best mixing method from a statistical standpoint.

2. The **XOR** method also achieved a perfect statistical pass ratio. However, as discussed extensively in Section 3.3.1, its inherent cryptographic weakness due to linearity and reversibility poses significant security risks in adversarial environments where the REE is compromised. Thus, despite its excellent statistical profile, its practical application must be carefully weighed against these security concerns.

3. The **AES** method achieved relatively good statistical performance, comparable to the baseline TEE RNG, with only a single subtest failure. Its use of a well-established cryptographic primitive offers good diffusion.

doi:10.6342/NTU202501452

4. The **SHA256** method was found to be statistically the worst among the combinatorial methods tested, with several subtest failures. While still largely effective, it may not fully mitigate all statistical weaknesses present in the input entropy sources as effectively as the CHAIN or AES methods.

Table 4.2: Per-Test Pass Ratios (%) for RNG Methods Based on the NIST SP 800-22 Test Suite(*Indicates the pass ratio is below the threshold.)

NIST Test Category	**CHAIN**	**XOR**	**SHA256**	**AES**	**TEE**	**REE**
Frequency	99	99	96	100	99	98
FrequencyWithinBlocks	98	98	97	100	100	100
Runs	99	99	99	99	99	99
LongestRunOfOnes	99	98	98	100	100	99
Rank	100	98	98	99	97	100
DiscreteFourierTransform	98	99	97	98	99	100
NonOverlappingTemplate	98.9	99.0	98.8	98.9	99.0	99.1
OverlappingTemplate	97	97	99	99	97	99
Universal	98	97	98	95*	100	94*
LinearComplexity	99	97	99	97	100	99
Serial	99.5	98.5	97	98	100	98.5
CumulativeSums	99	99	95.5*	99	100	98.5
RandomExcursions	98.8	98.8	97.6	98.9	98.9	98.9
RandomExcursionsVariant	98.8	98.8	97.6	100	98.9	98.9

doi:10.6342/NTU202501452

Table 4.3: P-value Uniformity Test Results for RNG Methods across NIST SP 800-22 Categories

NIST Test Category	**CHAIN**	**XOR**	**SHA256**	**AES**	**TEE**	**REE**
Frequency	0.1373	0.8978	0.4750	0.1453	0.7399	0.6371
FrequencyWithinBlocks	0.5749	0.5749	0.5341	0.5544	0.4560	0.6579
Runs	0.4944	0.8343	0.6371	0.3669	0.7981	0.9558
LongestRunOfOnes	0.3505	0.2133	0.4944	0.7399	0.4750	0.0966
Rank	0.6579	0.2023	0.2368	0.3505	0.7598	0.7981
DiscreteFourierTransform	0.3669	0.1088	0.2493	0.6993	0.1538	0.2368
NonOverlappingTemplate	0.0220	0.0072	0.0013	0.0136	0.0067	0.0179
OverlappingTemplate	0.2493	0.0487	0.3505	0.0062	0.1154	0.0179
Universal	0.6371	0.0156	0.6579	0.8514	0.1538	0.3669
LinearComplexity	0.6579	0.1538	0.8514	0.9357	0.0669	0.6579
Serial	0.7792	0.1816	0.6579	0.2757	0.2368	0.9114
CumulativeSums	0.8677	0.5955	0.1453	0.2133	0.2023	0.4560
RandomExcursions	0.0111	0.0111	0.0078	0.0071	0.0398	0.0262
RandomExcursionsVariant	0.0009	0.0111	0.0021	0.0021	0.1363	0.0055

Table 4.4: P-value Uniformity Test Results for RNG Methods across NIST SP 800-22 Categories

NIST Test Category	**CHAIN**	**XOR**	**SHA256**	**AES**	**TEE**	**REE**
Frequency	**0.1373**	0.8978	0.4750	0.1453	0.7399	0.6371
FrequencyWithinBlocks	**0.5749**	0.5749	0.5341	0.5544	0.4560	0.6579
Runs	**0.4944**	0.8343	0.6371	0.3669	0.7981	0.9558
LongestRunOfOnes	**0.3505**	0.2133	0.4944	0.7399	0.4750	0.0966
Rank	**0.6579**	0.2023	0.2368	0.3505	0.7598	0.7981
DiscreteFourierTransform	**0.3669**	0.1088	0.2493	0.6993	0.1538	0.2368
NonOverlappingTemplate	**0.0220**	0.0072	0.0013	0.0136	0.0067	0.0179
OverlappingTemplate	**0.2493**	0.0487	0.3505	0.0062	0.1154	0.0179
Universal	**0.6371**	0.0156	0.6579	0.8514	0.1538	0.3669
LinearComplexity	**0.6579**	0.1538	0.8514	0.9357	0.0669	0.6579
Serial	**0.7792**	0.1816	0.6579	0.2757	0.2368	0.9114
CumulativeSums	**0.8677**	0.5955	0.1453	0.2133	0.2023	0.4560
RandomExcursions	**0.0111**	0.0111	0.0078	0.0071	0.0398	0.0262
RandomExcursionsVariant	**0.0009**	0.0111	0.0021	0.0021	0.1363	0.0055

doi:10.6342/NTU202501452

4.4 Resource Consumption Analysis

To evaluate the computational cost of the proposed mixing approaches, a straightforward resource usage analysis based on the CPU time required to process 128MB of random data is conducted. Table 4.5 summarizes the number of cryptographic operations per output and the corresponding execution time measured in seconds. On the other hand, memory usage was not separately analyzed, as all mixing methods operate on fixed-length 256-bit inputs and outputs without requiring large buffers or dynamic memory allocation. Since the implementations rely on lightweight operations or hardware-supported primitives, memory footprints are minimal and comparable across all approaches. Other resource metrics such as energy consumption may also be considered and can be further evaluated in the future.

Table 4.5: CPU Time (in seconds) for Mixing approaches with 128MB of Data. Baseline RNG generation costs for REE and TEE are also shown for comparison. All values include random number generation time and the CPU time is calculated in Windows 11 with WSL2 Ubuntu 20.04

Method	**Mixing Ops per Unit**	**Total CPU Time (s)**
XOR Mixing	**1 XOR**	**3.0**
SHA-256 Mixing	1 SHA-256	3.5
AES-256 Mixing	1 AES-ECB	4.8
Chaining Mixing	1 XOR + 2 SHA-256	4.9
REE Baseline (No Mixing)	-	1.2
TEE Baseline (No Mixing)	-	0.9

From Table 4.4, one can find:

XOR Mixing This method uses a simple bitwise XOR operation, resulting in negligible overhead. It is highly efficient and ideal for resource-constrained embedded environments.

doi:10.6342/NTU202501452

SHA-256 Mixing This method performs one SHA-256 hash per output. SHA-256 is widely supported and efficient on modern CPUs, making this method a balanced choice between entropy extraction and performance.

AES-ECB Mixing This method uses one AES-256 encryption blocks. While AES operations benefit from hardware acceleration on many ARM platforms, the required key schedule and block transformations result in increased computational cost.

Chaining Mixing This method combines one XOR with two SHA-256 operations in a chaining pattern. While it has the highest computational cost, it also demonstrated the best performance of statistical randomness in the NIST SP 800-22 tests.

In summary, the choice of an optimal mixing method in a real-world application may need to consider multiple factors such as computational overhead (as discussed in Chapter 3) and the specific threat model, particularly regarding the security implications, to reach a practical trade-off between efficiency and statistical strength. That said, however, the aforementioned findings clearly indicate that the CHAIN method achieves the better statistical randomness than other evaluated approaches.

doi:10.6342/NTU202501452

Chapter 5

TEE-Protection over Security Primitives in UAV

5.1 Application Scenarios Overview

The TEE-protected combinatorial random number generator proposed in this thesis is designed primarily for the application scenario of resource-constrained Unmanned Aerial Vehicle (UAV) platforms, especially to meet the authentication and encryption capabilities required under future Remote ID regulations.

With the increasing adoption of Remote ID (RID) systems in unmanned aerial vehicles (UAVs), ensuring the authenticity, integrity, and availability of broadcasted identity information has become a critical cybersecurity requirement. As illustrated in Figure 5.1, RID frameworks such as ASTM F3411 [30] specify that UAVs periodically transmit identifying information—including unique ID, location, and timestamp—typically through Bluetooth or Wi-Fi broadcast.

Traditionally, cryptographic keys used for signing or encrypting RID data are protected using hardware-based Secure Elements (SEs) or Trusted Platform Modules (TPMs). However, such components yield additional cost, increase hardware complexity, and are difficult to scale across low-cost or compact UAV platforms. As a result, replacing these external secure chips with an equivalent level of protection using a software-isolated Trusted Execution Environment (TEE) has emerged as a promising solution.

doi:10.6342/NTU202501452

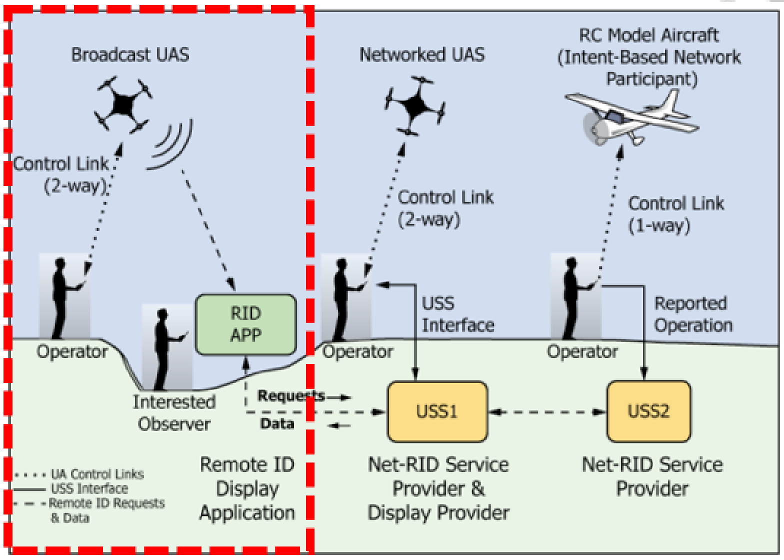


Figure 5.1: The Application Scenario of UAV Remote ID specified in ASTM F3411. The red dot-line box highlights the scope of this work.

To realize this vision, the implementation in this study leverages ARM TrustZone and OP-TEE to serve as a secure foundation for RID cryptographic operations, such as digital signing with ECDSA and authentication with AES-CCM. Moreover, the previously proposed dual-domain entropy mixing design, in which TEE and REE jointly contribute to random number generation is integrated to ensure that the cryptographic keys and nonces used for RID messages are rooted in high-quality randomness. This dual-layer approach enhances both the unpredictability and integrity of RID data under real-time constraints, even in the presence of REE compromise.

This chapter presents the overall system integration design and its application to UAV cryptographic workflows by describing the practical implementation of RNG-based architecture for secure key storage, signature generation, and AES-based authentication, as well as demonstrating how TEE-based solutions can satisfy RID mandates without relying on external hardware security chips.

doi:10.6342/NTU202501452

5.2 Architecture Design

To realize a secure and cost-effective implementation of Remote ID functionality on UAV platforms, the proposed system design integrates TEE-based cryptographic operations and a dual-entropy RNG module into the embedded architecture. This approach aims to replace external secure elements (e.g., TPM, SE) with trusted software-based components without compromising security or compliance. The full-stack software pipeline, illustrated in Figure 5.2, covers key management, message signing/encryption in the TEE, and broadcasting through the REE-managed Bluetooth stack.

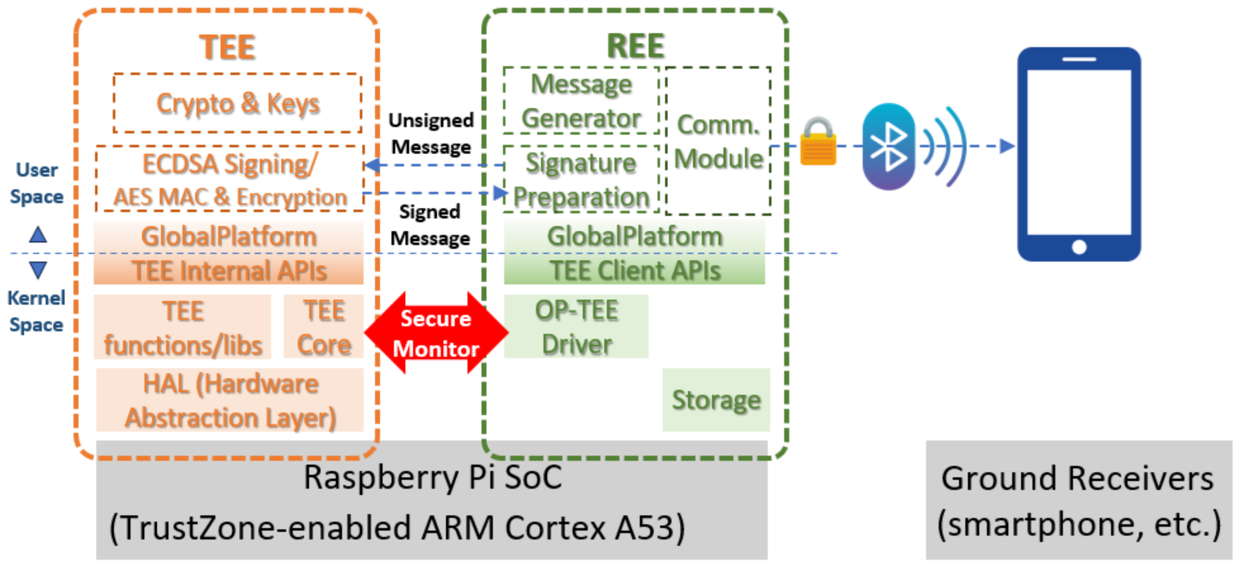


Figure 5.2: The proposed architecture of TEE-Protection over crypto primitives on Raspberry Pi for Remote ID broadcasting.

The integration design is structured around the following modules:

- **RNG Combinations:** Each cryptographic operation involving key generation or nonce creation draws random numbers from the hybrid RNG, which mixes entropy from both TEE (trusted TRNG) and REE (external sources) via secure mixing methods such as XOR, SHA-256, or AES.

- **TEE-Based Key Storage and Operations:** All cryptographic keys (e.g., ECDSA signing key, AES key) are securely stored and used inside the Trusted Execution Environment. No key material is exposed to the Rich Execution Environment (REE), preventing leakage or tampering even if the REE is compromised.
- **Remote ID Broadcast Flow:** The UAV collects location, timestamp, and unique ID data, and prepares the Remote ID message. A Message Authentication Code (MAC) is then generated using AES-CCM or a digital signature using ECDSA, depending on the regional requirements.
- **Secure Output Interface:** The signed or authenticated message is forwarded back to REE for wireless transmission over Bluetooth. Although the REE handles I/O, all sensitive processing is kept within the TEE.

This integration ensures compliance with standards like ASTM F3411 and regional RID regulations, while minimizing hardware cost and maximizing platform compatibility. Figure 5.3 presents a high-level system diagram of the proposed design.

5.3 Implementation of RNG Combinations and additional Crypto-Primitives in TEE

This section outlines the cryptographic implementation integrated into UAV system using the Trusted Execution Environment (TEE), including key generation, digital signature generation using ECDSA, AES-256 authentication and encryption. All operations are executed within OP-TEE on Raspberry Pi 3B+.

doi:10.6342/NTU202501452

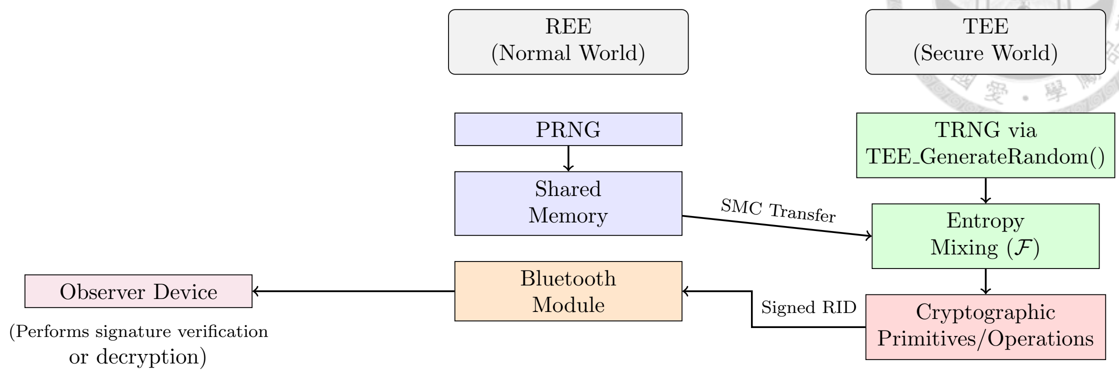

REE
(Normal World)
TEE
(Secure World)
PRNG
Shared Memory
TRNG via TEE_GenerateRandom()
SMC Transfer
Entropy Mixing ($\mathcal{F}$)
Observer Device
Bluetooth Module
Signed RID
Cryptographic Primitives/Operations
(Performs signature verification or decryption)

Figure 5.3: Design of the integrated broadcast system with dual-domain entropy.

5.3.1 Deployment of Dual-Domain RNGs

As established in the proposed architecture, the system utilizes two distinct entropy sources allocated across the two security domains. The primary, **trusted source** is the TEE's internal True Random Number Generator (TRNG), which is accessed securely via the `TEE_GenerateRandom()` function call from within a Trusted Application (TA). The secondary, **untrusted source** is a software-based Pseudo-random Number Generator (PRNG) operating in the Rich Execution Environment (REE).

To combine these sources, the random data from the REE is passed into the secure world via a shared memory buffer, invoked by a Secure Monitor Call (SMC). The critical mixing process is performed exclusively inside the TA, ensuring that any potential manipulation of the REE source does not compromise the final combination operation. Based on the superior statistical randomness demonstrated in Chapter 4, the **CHAIN** method was selected as the mixing algorithm for this practical implementation.

The high-quality random output from this combinatorial process then serves as a secure seed for all subsequent cryptographic operations within the TEE, includ-

doi:10.6342/NTU202501452

ing the key generation described in Section 5.3.2 and the nonce creation for ECDSA and AES-CCM operations.

5.3.2 Key Generation

The key generation and management process is designed to be fully confined within the TEE, adhering to the GlobalPlatform specification for secure operations. The workflow, illustrated in Figure 5.4, ensures that private key material is never exposed to the REE.

The process begins when a client application in the REE requests the creation of a new key. This request is securely passed to a Trusted Application (TA) inside the TEE through a session-based interface mediated by Secure Monitor Calls (SMCs).

Upon receiving the request, the TA utilizes the high-quality random output from the combinatorial RNG (as detailed in Section 5.3.1) to generate the cryptographic key. For persistence, the TA then invokes the Secure Storage API. This function encrypts the newly generated key with a hardware-unique key before exporting the resulting encrypted object to the REE's file system for storage. Access is protected by a unique object identifier, ensuring that only the originating TA can later decrypt and use the key in subsequent operations. This method provides strong guarantees for key confidentiality, even if the non-secure file system is compromised.

5.3.3 Digital Signatures Primitives: ECDSA

To ensure the authenticity of broadcast messages, the Elliptic Curve Digital Signature Algorithm (ECDSA) is implemented using the NIST P-256 curve (secp256r1) and SHA-256 inside the TEE.

When the REE wishes to sign a message, it initiates a Secure Monitor Call (SMC) to invoke a trusted session with the TA inside the TEE. The TA securely

doi:10.6342/NTU202501452

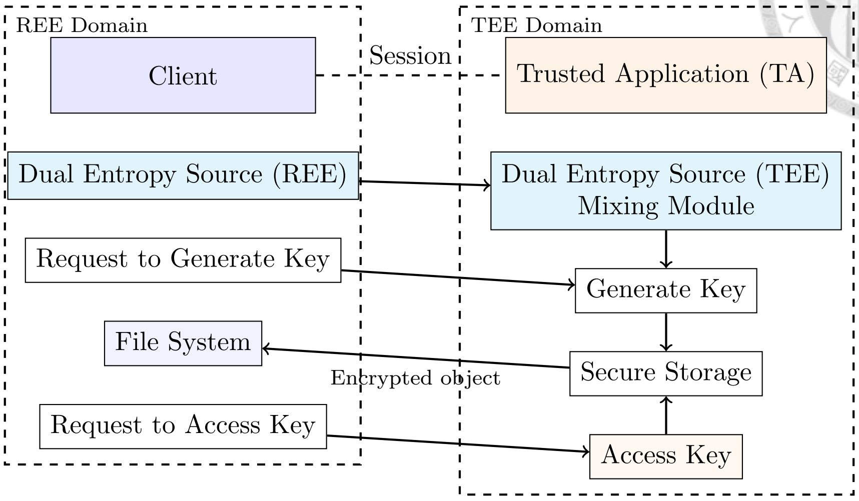


Figure 5.4: Component Structure Diagram of Key Storage

loads the private key from persistent storage, then generates a per-signature nonce k by calling the dual-entropy mixing module. This ensures unpredictability even under partial entropy compromise. Finally, the ECDSA engine computes the signature pair (r, s) and returns the result to the REE for subsequent broadcasting.

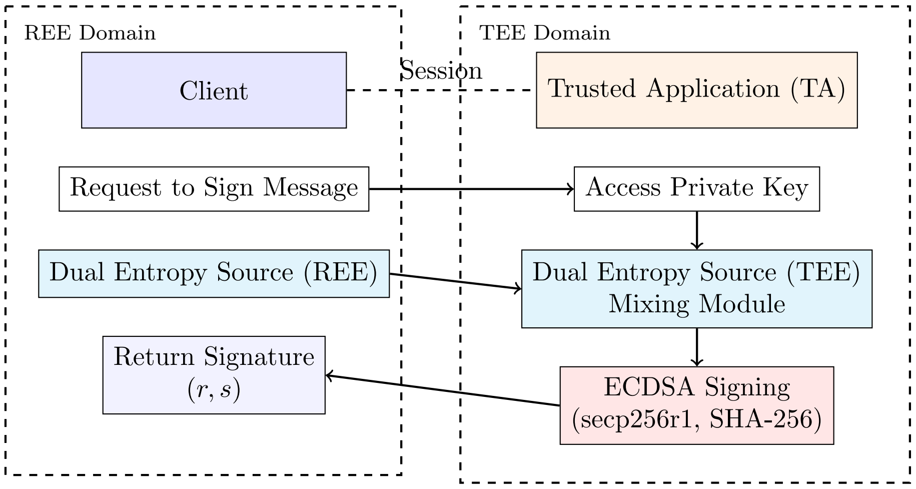


Figure 5.5: Component Structure Diagram of ECDSA Signing

doi:10.6342/NTU202501452

5.3.4 Authentication and Encryption Primitives: AES-CCM-256

To protect message confidentiality and integrity, the system implements AES-CCM with a 256-bit key. The choice of AES-CCM over the also common AES-GCM is a deliberate design decision, tailored for the specific requirements of UAV Remote ID applications and emerging regulatory standards. The primary difference lies in how the authentication tag is generated; AES-CCM uses CBC-MAC on the plaintext, a method that is computationally lightweight and highly suitable for scenarios requiring only message authentication without encryption. This aligns with the approach taken by regulators in countries like Japan, making it a fitting choice for current Remote ID implementations. The architecture remains forward-compatible, allowing for a future upgrade to the more rigorous AES-GCM mode should applications require encrypted data transmission.

During each encryption session, the TEE retrieves the key, constructs a 12-byte nonce, and invokes the AES-CCM engine. The message payload to be encrypted typically includes the UAV ID, timestamp, latitude, and longitude, e.g., “B-AAB1230 2025-05-17 14:22:29:067 25.6666000 124.5555456”. A 16-byte MAC tag is generated to ensure message authenticity and integrity.

The randomness required to generate the nonce is derived from the dual-entropy mixing module introduced in Chapter 3. This ensures freshness and unpredictability even if either entropy source is partially compromised.

The signed and encrypted output is returned to the REE, where it is relayed to the broadcasting module. As shown in Figure 5.6

doi:10.6342/NTU202501452

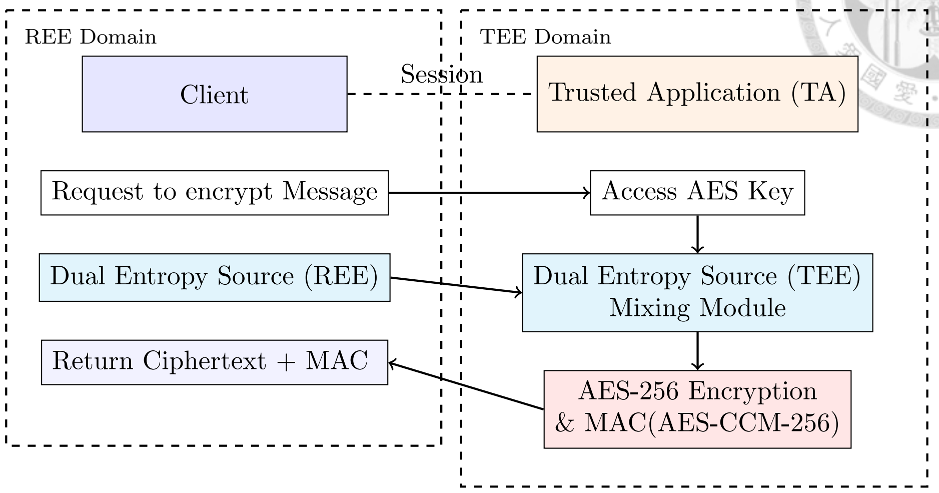


Figure 5.6: Component Structure Diagram of AES Encryption and Authentication

5.4 The Working Prototype

To further verify the proposed architecture, a working prototype comprising a Bluetooth broadcasting module (ESP32), a TEE-based cryptographic node (Raspberry Pi 3B+ running OP-TEE), and a ground receiver (mobile phone with pre-installed public key and OpenDroneID-compatible verification app) was constructed.Table 5.1 summarizes the system specifications of the device under test (DUT), including the software stack and crypto modules. Figure 5.7 shows the physical PoC setup. This configuration emulates the complete Remote ID (RID) cryptographic flow—from data generation, secure signing, encryption, and wireless broadcast, to external reception and verification.

The receiver application running on the above prototype successfully verified both digital signatures and MAC tags from received RID messages. As shown in Figure 5.8, aside from the basic RID information, field (A) indicates that ECDSA signature verification returned `TRUE`, while field (B) confirms that the AES-CCM-256 MAC tag was authenticated. The decrypted message payload, seen in (C), includes a timestamp and location tuple matching the original transmission.

doi:10.6342/NTU202501452

Table 5.1: DUT (Device-Under-Test) Configuration and System Specifications

Config. Item (CI)	**Attribute/Name**	**Spec/Version**
Target Device	Raspberry Pi	Model 3B+
TEE	OP-TEE	4.0.0
OS	Raspbian	2019-06-20 Raspbian buster
Dev. Language	Python	3.11
Receiver App	OpenDroneID OSM	Jan 12, 2024
Receiver Device	SAMSUNG Galaxy A22	Android 13
Digital Signature	ECDSA	secp256r1 (256-bit)
Encrypt / Decrypt	AES-CCM	256-bit
Security Keys	Pre-distributed	–

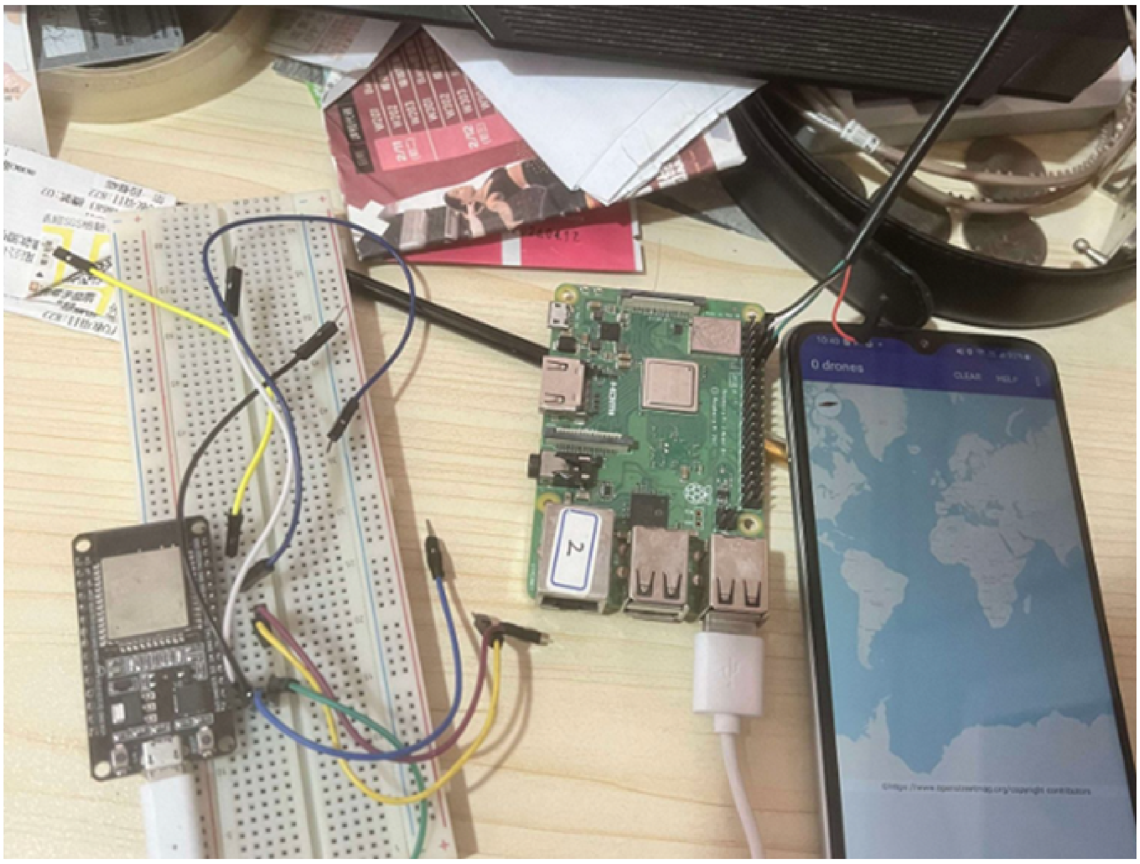
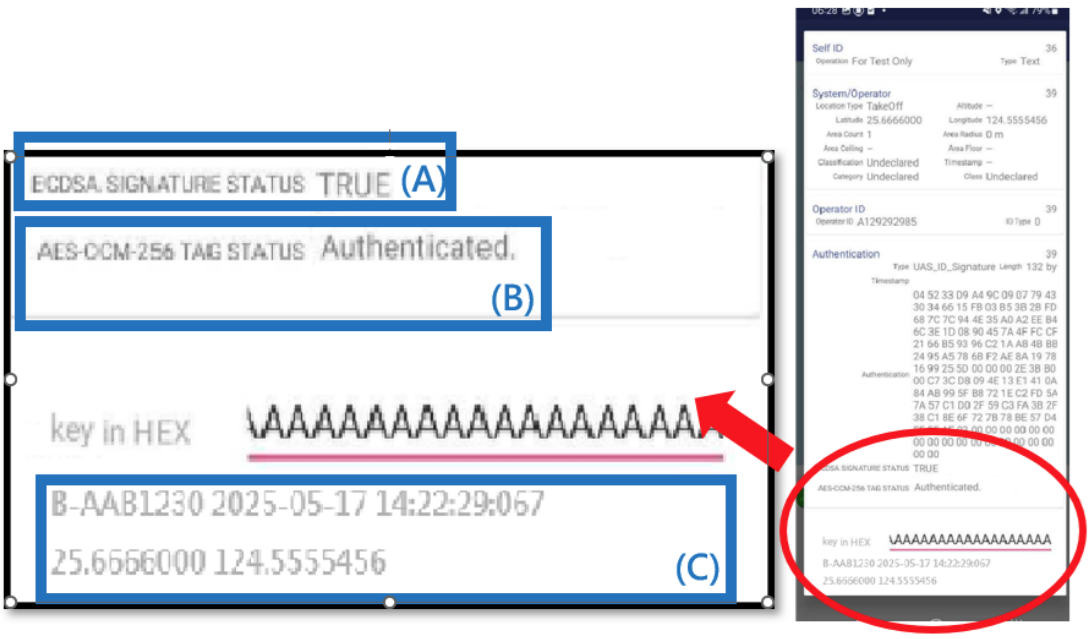


Figure 5.7: Physical testbed: broadcasting module (left), TEE-based secure node (center), and smartphone receiver (right).

doi:10.6342/NTU202501452



```
ECDSA SIGNATURE STATUS TRUE (A)
AES-CCM-256 TAG STATUS Authenticated. (B)
key in HEX \AAAAAAAAAAAAAAAAA
B-AAB1230 2025-05-17 14:22:29:067
25.6666000 124.5555456 (C)
```

```
06:28                                             79%
Self ID                                            36
Operation For Test Only                   Type Text

System/Operator                                    39
Location Type TakeOff          Altitude –
     Latitude 25.6666000      Longitude 124.5555456
   Area Count 1             Area Radius 0 m
 Area Ceiling –              Area Floor –
Classification Undeclared     Timestamp –
     Category Undeclared          Class Undeclared

Operator ID                                        39
Operator ID A129292985                     ID Type 0

Authentication                                     39
               Type UAS_ID_Signature Length 132 by
          Timestamp
                    04 52 33 D9 A4 9C 09 07 79 43
                    30 34 66 15 FB 03 B5 3B 2B FD
                    68 7C 7C 94 4E 35 A0 A2 EE B4
                    6C 3E 1D 08 90 45 7A 4F FC CF
                    21 66 B5 93 96 C2 1A A8 4B BB
                    24 95 A5 78 6B F2 AE 8A 19 78
     Authentication 16 99 25 5D 00 00 00 2E 3B B0
                    00 C7 3C D8 09 4E 13 E1 41 0A
                    84 AB 99 5F B8 72 1E C2 FD 5A
                    7A 57 C1 D0 2F 59 C3 FA 3B 2F
                    38 C1 8E 6F 72 7B 78 BE 57 D4
                    [illegible] 00 00 00 00 00 00
                    00 00 00 00 [illegible] 00 00 00
                    00 00
ECDSA SIGNATURE STATUS TRUE
AES-CCM-256 TAG STATUS Authenticated.

key in HEX \AAAAAAAAAAAAAAAAA
B-AAB1230 2025-05-17 14:22:29:067
25.6666000 124.5555456
```

Figure 5.8: Verification output from ground receiver. (A) Signature verification, (B) AES-CCM MAC validation, (C) plaintext extracted from decrypted ciphertext.

This result verifies that the private key within the TEE correctly signed the message, the AES encryption engine generated a valid MAC, and the public key and AES key pre-distributed to the mobile device were sufficient for verification and decryption.

5.5 Summary

The overall implementation demonstrates that TEE-based cryptographic modules on Raspberry Pi are capable of securely generating, storing, and using keys for signing and encryption, without leaking sensitive material to the REE. The system achieves end-to-end Remote ID data protection that follows the security guideline of ASTM F3411, while ensuring practical compatibility with off-the-shelf mobile receivers. These results verify the feasibility of deploying software-based trusted components as lightweight alternatives to hardware secure elements in UAV systems.

Chapter 6

Conclusions and Future Works

6.1 Conclusion

This thesis successfully addressed the critical need for robust and resilient random number generation in resource-constrained environments by proposing and evaluating a combinatorial RNG protected by a Trusted Execution Environment. After a thorough analysis of multiple mixing methods and a practical implementation on a UAV-centric prototype, the key contributions of this study are summarized as follows:

- **Identified a Superior Mixing Method:** We designed and empirically evaluated four cross-domain entropy mixing approaches. The results from the NIST SP 800-22 statistical test suite conclusively demonstrated that the proposed CHAIN method is the most effective, achieving a perfect pass ratio and exhibiting superior statistical reliability.
- **Developed a Resilient TEE/REE Architecture:** This research proposed a dual-domain architecture that enhances security resilience while adhering to the principle of TCB minimization. We established that by protecting just one entropy source and the mixing operation within the TEE, the system can withstand a full compromise of the REE-based source. Such a counter-

doi:10.6342/NTU202501452

intuitive deployment manner provides a practical and efficient security model for UAVs or similar application platforms with limited resources.

- **Validated Feasibility on a COTS Platform:** The proposed system was successfully implemented on a Raspberry Pi 3B+ with OP-TEE, demonstrating its applicability on commercially off-the-shelf hardware. This working prototype, tailored for a UAV Remote ID broadcast application, verified that a TEE-based combinatorial RNG is a practical and resilient solution for enhancing security in real-world scenarios and supporting compliance with emerging regulatory requirement for Taiwan.

6.2 Future Work

The findings from this study open several promising avenues for future research, focusing on enhancing the core mechanism and broadening its system-level applications, in line with emerging regulatory and technological trends.

- **Strengthening Randomness and Security:** Future work could explore more advanced mixing algorithms and formally analyze the trade-offs between computational cost and security properties. Developing an adaptive framework that dynamically adjusts the mixing strategy based on real-time health checks of the entropy sources is another valuable direction. Furthermore, the security of the TEE-based architecture itself could be enhanced to provide stronger protections within the ARM TrustZone environment.

- **Formal Security Analysis via Relaxed LHL Conditions**: To more rigorously bridge the gap between the practical CHAIN method and the theoretical Leftover Hash Lemma, a valuable direction for future research is to adopt a

doi:10.6342/NTU202501452

relaxed version of the LHL's premises. For instance, the strict 2-universal condition could be extended to a *k-2-universal* property, which only requires that for any distinct inputs x, y, the collision probability $P(f(x) = f(y))$ is at most $k/|B|$ (where $0 < k < 1$). Based on this foundation, one could employ Monte Carlo simulations or empirical collision rate testing to estimate the degree to which the implemented hash function satisfies this k-2-universal property under a given confidence level. This approach would enable the derivation of a quantitative lower bound on entropy retention, serving as a reasonable and formal compromise between the ideal LHL theory and the practical implementation.

- **Broadening Applications and Regulatory Compliance:** The proposed framework provides a strong foundation for wider system-level applications. It could be extended to support secure Key Update and distribution mechanisms, which are critical for both civilian and military applications like Identify Friend or Foe (IFF). Additionally, integrating this secure RNG into advanced services like Unmanned Aircraft System Traffic Management (UTM) and ensuring compliance with evolving Remote ID regulations are important future steps.

- **Key Management and Verification:** The management and verification of cryptographic keys represent a significant area for future development. Expanding the framework to include more sophisticated key lifecycle management within the TEE would further enhance its value and security posture.

doi:10.6342/NTU202501452

Bibliography

[1] M. Sabt, M. Achemlal, and A. Bouabdallah, "Trusted execution environment: What it is, and what it is not," in *2015 IEEE Trustcom/BigDataSE/ISPA*, 2015, pp. 57–64.

[2] P. Jauernig, A. R. Sadeghi, and E. Stapf, "Trusted execution environments: Properties, applications, and challenges," *IEEE Security & Privacy*, vol. 18, no. 2, pp. 56–60, March-April 2020.

[3] E. Barker and J. Kelsey, "Recommendation for random bit generator (rbg) constructions (nist sp 800-90c)," NIST, Tech. Rep. SP 800-90C, 2024.

[4] NIST, "Recommendation for the entropy sources used for random bit generation (sp 800-90b)," NIST, Tech. Rep. SP 800-90B, 2018.

[5] N. Ferguson, B. Schneier, and T. Kohno, *Cryptography Engineering: Design Principles and Practical Applications.* Wiley, 2010.

[6] B. Jun and P. Kocher, "The intel random number generator," Cryptography Research Inc., Tech. Rep., 1999.

[7] NIST, "Recommendation for random number generation using deterministic random bit generators (sp 800-90a rev.1)," NIST, Tech. Rep. SP 800-90A Rev.1, 2015.

doi:10.6342/NTU202501452

[8] P. Kietzmann *et al.*, "A guideline on pseudorandom number generation in the iot," *ACM Computing Surveys*, vol. 54, no. 6, pp. 1–36, 2021.

[9] D. Petro and A. Cecil, "You're doing IoT security RNG," Bishop Fox (Industry survey), 2021.

[10] N. Heninger *et al.*, "Mining your ps and qs: Detection of widespread weak keys in network devices," in *USENIX Security Symposium*, 2012.

[11] B. A. Wichmann and I. D. Hill, "Algorithm as 183: An efficient and portable pseudo-random number generator," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 31, no. 2, pp. 188–190, 1982.

[12] P. L'Ecuyer, "Good parameters for combined mrgs," *Operations Research*, vol. 47, no. 1, pp. 159–164, 1999.

[13] M. Wang, H. Qu, F. Guo, and S. Li, "Combined random number generators: A review," in *2011 IEEE 3rd International Conference on Communication Software and Networks (ICCSN)*, 2011, pp. 443–447.

[14] H. Haramoto *et al.*, "Efficient jump ahead for f-linear rngs," *INFORMS Journal on Computing*, vol. 20, no. 3, pp. 385–390, 2008.

[15] P. L'Ecuyer and F. Panneton, "Fast rngs based on linear recurrences modulo 2," in *Proceedings of the Winter Simulation Conference (WSC)*, 2005, p. 10 pp.

[16] D. Blackman and S. Vigna, "Scrambled linear pseudorandom number generators," *ACM Trans. Math. Softw.*, vol. 47, no. 4, pp. 36:1–36:32, 2021.

[17] K. Ramasubramanian and K. Suresh, "Design of a hybrid rng based on chaotic systems," https://www.researchgate.net/publication/389939574, 2024, preprint.

doi:10.6342/NTU202501452

[18] L. Xie, H. Liu, W. Zhang, and Y. Zhang, "TRNG based on SRAM and NFSR," *Adv. Electronic Materials*, vol. 6, no. 3, p. 1901117, 2020.

[19] Y. Dodis *et al.*, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," *SIAM J. Computing*, vol. 38, no. 1, pp. 97–139, 2008.

[20] N. Heninger, Z. Durumeric, E. Wustrow, and J. A. Halderman, "Mining your ps and qs: Detection of weak keys in network devices," in *USENIX Security Symposium*, 2012.

[21] A. A. Fröhlich, "A comprehensive approach to power management in embedded systems," *International Journal of Distributed Sensor Networks*, vol. 2011, pp. 1–14, 2011.

[22] TCG, "TPM library specification, family "2.0", level 00, revision 01.59," Trusted Computing Group, Tech. Rep., November 2019. [Online]. Available: https://trustedcomputinggroup.org/resource/tpm-library-specification/

[23] H.-S. Yang and S.-J. Yoo, "A study on secure element for smartwork," in *2014 International Conference on IT Convergence and Security (ICITCS)*, Beijing, China, 2014, pp. 1–3.

[24] C. Ryu, J.-H. Lee, D.-H. Kim, H.-S. Lee, Y.-S. Kim, J. nyeo Kim, and J.-H. Han, "A comprehensive survey of tpm for defense systems," *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 18, no. 7, pp. 1953–1967, 2024.

[25] ARM Limited, "ARM Security Technology - Building a Secure System using TrustZone Technology," ARM Limited, White Paper, 2009.

doi:10.6342/NTU202501452

[26] OP-TEE Project, "OP-TEE OS Documentation," https://optee.readthedocs.io/, Accessed on 2024-05-27.

[27] Intel Corporation, "Intel® software guard extensions (intel® sgx)," https://www.intel.com/content/www/us/en/products/docs/accelerator-engines/software-guard-extensions.html, 2024, accessed: 2025-06-07.

[28] L. Gupta, R. Jain, and G. Vaszkun, "Survey of important issues in uav communication networks," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1123–1152, 2015.

[29] Federal Aviation Administration, "Remote Identification of Unmanned Aircraft Final Rule," https://www.ecfr.gov/current/title-14/chapter-I/subchapter-F/part-89, 2021, accessed: 2025-05-31.

[30] ASTM International, "Standard Specification for Remote ID and Tracking," https://www.astm.org/f3411-22.html, 2022, aSTM F3411-22.

[31] Ministry of Land, Infrastructure, Transport and Tourism, "Technical Standards for Remote ID Devices in Japan," https://www.mlit.go.jp/en/koku/uas.html, 2022, requirement for AES-128-CCM in RID authentication.

[32] Civil Aeronautics Administration, Taiwan, "Article 99-10 of the Civil Aviation Act Amendment," https://www.caa.gov.tw/Article.aspx?a=2425&lang=1, 2023, draft implementation of Remote ID requirements.

[33] D. Upadhyay, N. Gaikwad, M. Zaman, and S. Sampalli, "Investigating the avalanche effect of various cryptographically secure hash functions and hash-based applications," *IEEE Access*, vol. 10, pp. 112 472–112 486, 2022.

doi:10.6342/NTU202501452

[34] P. Rogaway and T. Shrimpton, "Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance," in *Fast Software Encryption, FSE 2004*, ser. Lecture Notes in Computer Science, B. Roy and W. Meier, Eds., vol. 3017. Springer, Berlin, Heidelberg, 2004, pp. 371–388. [Online]. Available: https://doi.org/10.1007/978-3-540-25937-4_24

[35] Y. Dodis and A. Smith, "Entropic security and the encryption of high entropy messages," in *Theory of Cryptography Conference (TCC)*, ser. LNCS, vol. 3378. Springer, 2005, pp. 556–577.

[36] N. T. Thu Nga, H. D. Tho, and L. M. Tu, "On the improving diffusion layer and performance of aes algorithm," in *2017 International Conference on Information and Communications (ICIC)*, 2017, pp. 288–292.

[37] I. Corporation, "Advanced encryption standard (aes) instructions set - white paper," Intel Corporation, Tech. Rep., 2010, white Paper. [Online]. Available: https://www.intel.com/content/dam/doc/white-paper/advanced-encryption-standard-new-instructions-set-paper.pdf

[38] M. Bellare, H. Davis, and F. Günther, "Separate your domains: NIST PQC KEMs, oracle cloning and read-only indifferentiability," Cryptology ePrint Archive, Paper 2020/241, 2020. [Online]. Available: https://eprint.iacr.org/2020/241

[39] B. Barak, Y. Dodis, H. Krawczyk, O. Pereira, K. Pietrzak, F.-X. Standaert, and Y. Yu, "Leftover hash lemma, revisited," Cryptology ePrint Archive, Report 2021/1146, 2021, online version: https://eprint.iacr.org/2021/1146. [Online]. Available: https://eprint.iacr.org/2021/1146

doi:10.6342/NTU202501452

[40] R. Impagliazzo and M. Luby, "One-way functions are essential for complexity based cryptography," in *30th Annual Symposium on Foundations of Computer Science*, 1989, pp. 230–235.

[41] R. McEvoy, Robert, J. T. Curran, J. T., Cotter, Paul, C. Murphy, and Colin, "Fortuna: Cryptographically secure pseudo-random number generation in software and hardware," 07 2006.

[42] Y. Dodis, K. Pietrzak, and D. Wichs, "Key derivation without entropy waste," Cryptology ePrint Archive, Paper 2013/708, 2013. [Online]. Available: https://eprint.iacr.org/2013/708

[43] J. Zhang and M. Wu, "Random number generation based on heterogeneous entropy sources fusion in multi-sensor networks," *Sensors*, vol. 23, no. 20, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/20/8497

[44] National Institute of Standards and Technology, "Random bit generation project," https://csrc.nist.gov/projects/random-bit-generation/documentation-and-software, 2025, accessed: 2025-05-31.

[45] R. G. Brown, "Dieharder: A random number test suite," https://webhome.phy.duke.edu/~rgb/General/dieharder.php, 2025, accessed: June 2025.

[46] P. L'Ecuyer and R. Simard, "Testu01: A c library for empirical testing of random number generators," vol. 33, no. 4, Aug. 2007. [Online]. Available: https://doi.org/10.1145/1268776.1268777

doi:10.6342/NTU202501452

Appendix

doi:10.6342/NTU202501452

Appendix A

Formal Definition of the Leftover Hash Lemma

The Leftover Hash Lemma (LHL) is a fundamental result in the field of cryptography and complexity theory. It provides a formal method for extracting a short, high-quality (i.e., statistically close to uniform) random string from a longer, weaker random source that possesses sufficient entropy but is not perfectly uniform. It serves as the theoretical foundation for many cryptographic constructions, including randomness extractors and key derivation functions.

To understand the lemma, some key concepts are established.

A.1 Preliminary Definitions

A.1.1 Min-Entropy

Min-entropy is a measure of a random variable's unpredictability. For a random variable X over a set $\mathcal{X}$, its min-entropy, denoted as $H_\infty(X)$, is defined as the negative logarithm of the probability of its most likely outcome. It quantifies the amount of randomness in a "worst-case" scenario.

$$H_\infty(X) = -\log_2 \left(\max_{x \in \mathcal{X}} P(X = x) \right)$$

A source with at least k bits of min-entropy means that no single output has a probability greater than 2^{-k}.

doi:10.6342/NTU202501452

A.1.2 Statistical Distance

Statistical distance is a measure of how distinguishable two probability distributions are. For two distributions A and B over the same set $\mathcal{Z}$, their statistical distance is defined as:

$$\mathrm{SD}(A,B) = \frac{1}{2}\sum_{z\in\mathcal{Z}} |P(A=z) - P(B=z)|$$

If $\mathrm{SD}(A,B) \le \epsilon$, the distributions A and B are said to be ϵ-close, meaning that no algorithm can distinguish between them with a probability of success that is more than ϵ better than random guessing.

A.1.3 2-Universal Hash Family

A family of hash functions $\mathcal{H} = \{h : \mathcal{U} \to \mathcal{V}\}$ is called a 2-universal hash family if, for any two distinct inputs $x_1, x_2 \in \mathcal{U}$, the probability of a collision is no greater than the probability of a collision for a truly random function. Formally, for a function h chosen uniformly at random from $\mathcal{H}$:

$$P_{h\in\mathcal{H}}[h(x_1) = h(x_2)] \le \frac{1}{|\mathcal{V}|}$$

A.2 The Leftover Hash Lemma

With the above definitions, the Leftover Hash Lemma can thus be described as follows.

Theorem (Leftover Hash Lemma). Let $\mathcal{H} = \{h : \{0,1\}^n \to \{0,1\}^m\}$ be a 2-universal family of hash functions. Let X be a random variable over $\{0,1\}^n$ with min-entropy $H_\infty(X) \ge k$. Let H be a random variable representing a function chosen uniformly from $\mathcal{H}$, independent of X.

doi:10.6342/NTU202501452

The distribution of the output $(H(X), H)$ is ϵ-close to the uniform distribution (U_m, H), where U_m is the uniform distribution over $\{0,1\}^m$. The statistical distance is bounded by:

$$\mathrm{SD}((H(X), H), (U_m, H)) \leq \epsilon$$

where

$$\epsilon = \frac{1}{2}\sqrt{2^{m-k}}$$

In essence, the lemma guarantees that from a source with k bits of min-entropy, an m-bit string that is nearly perfectly random can be extracted using a 2-universal hash function, provided that m is sufficiently smaller than k. The "leftover" entropy, $k - m$, determines how statistically close the output is to a true uniform distribution.

doi:10.6342/NTU202501452