Research and Development of a Decentralized Random Number Generation Protocol for Blockchain-based Application

Instructors:
Dr. Nguyen An Khuong
Mr. Tran Anh Dung
Dr. Tang Khai Hanh
Dr. Dang Tuan Thuong

Pham Nhat Minh - 1910346, HCMUT.

University of Technology
Ho Chi Minh city

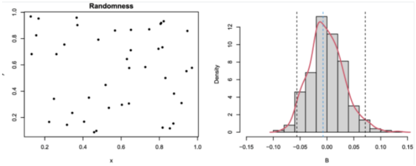
Table of Contents

Table of Contents

Motivation

Randomness is very important:

- Used in many applications in life: Gaming, Simulation, Voting, Cryptography, . . .
- A reliable source of randomness is required for such applications.

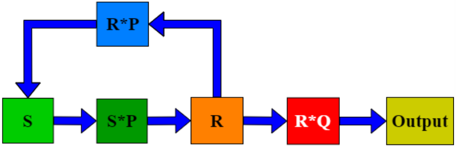


Motivation

Producing randomness is hard.

- True random number generator (TRNG): Produce true random output, but slow and must be stored for testing error.
- Pseudo-random generator (PRNG): Fast, produce pseudo-random outputs. But the owner knows the whole generation process!
- Notorious example of PRNG: The backdoor in Dual EC DRBG of NSA. [1]

PRNG

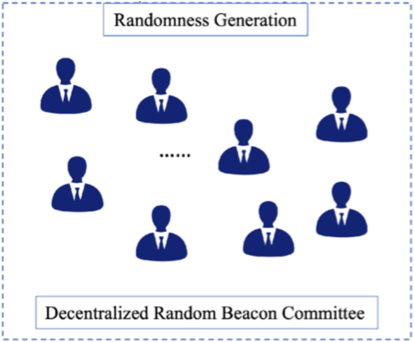


Dual EC DRBG

[1] https://en.wikipedia.org/wiki/Dual_EC_DRBG

Decentralized Random Number Generator

- The role of generating randomness is split among participants.
- Reduce the risk of a single individual having access to the whole randomness-generating process.
- However, older DRNGs have problems in security or efficiency, while newer DRNG constructions are not widely studied.



Objectives

In this thesis, we wish to fulfill the following objectives:

- Formally define the syntax and security properties of DRNGs.

Objectives

In this thesis, we wish to fulfill the following objectives:

- Formally define the syntax and security properties of DRNGs.
- Conduct a systematic literature review and analyze the strengths and weaknesses of existing DRNG constructions.

Objectives

In this thesis, we wish to fulfill the following objectives:

- Formally define the syntax and security properties of DRNGs.
- Conduct a systematic literature review and analyze the strengths and weaknesses of existing DRNG constructions.
- Based on the results of our survey, specify the most suitable DRNG construction tailored for blockchain-based applications.

Objectives

In this thesis, we wish to fulfill the following objectives:

- Formally define the syntax and security properties of DRNGs.
- Conduct a systematic literature review and analyze the strengths and weaknesses of existing DRNG constructions.
- Based on the results of our survey, specify the most suitable DRNG construction tailored for blockchain-based applications.
- Finally, apply the aforementioned protocol to construct a DRNG system for blockchain-based applications.

Table of Contents

Formal Definition

A DRNG consists of two interactive protocols **DRNGSetup, DRNGGen** and an algorithm **DRNGVerify** as follows:

- $\mathbf{DRNGSetup}(1^{\lambda})\langle\{P\}_{P\in\mathcal{P}}\rangle$ **:** Executed by all participants in $\mathcal{P}$. Outputs a set **QUAL**, public parameter **pp**, state $\mathbf{st_0}$ and each qualified P_i holds a secret key $\mathbf{sk}_i$.

Formal Definition

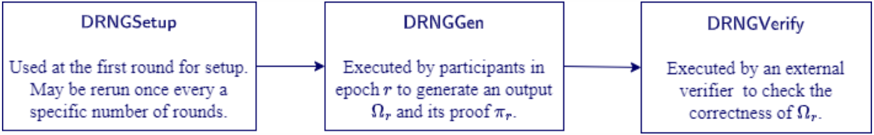
A DRNG consists of two interactive protocols **DRNGSetup, DRNGGen** and an algorithm **DRNGVerify** as follows:

- $\mathbf{DRNGSetup}(1^\lambda)\langle\{P\}_{P\in\mathcal{P}}\rangle$ **:** Executed by all participants in $\mathcal{P}$. Outputs a set **QUAL**, public parameter **pp**, state $\mathbf{st_0}$ and each qualified P_i holds a secret key $\mathbf{sk}_i$.
- $\mathbf{DRNGGen}(\mathbf{st}_r, \mathbf{pp})\langle\{P_i(\mathbf{sk}_i)\}_{i\in\mathbf{QUAL}}\rangle$ **:** Executed by all participants in **QUAL** with common inputs $\mathbf{st}_r$,**pp**. Outputs a value Ω_r, a proof π_r update the state from $\mathbf{st}_r$ to $\mathbf{st}_{r+1}$.

Formal Definition

A DRNG consists of two interactive protocols **DRNGSetup, DRNGGen** and an algorithm **DRNGVerify** as follows:

- $\mathbf{DRNGSetup}(1^\lambda)\langle\{P\}_{P\in\mathcal{P}}\rangle$ **:** Executed by all participants in $\mathcal{P}$. Outputs a set **QUAL**, public parameter **pp**, state $\mathbf{st}_0$ and each qualified P_i holds a secret key $\mathbf{sk}_i$.
- $\mathbf{DRNGGen}(\mathbf{st}_r, \mathbf{pp})\langle\{P_i(\mathbf{sk}_i)\}_{i\in\mathbf{QUAL}}\rangle$ **:** Executed by all participants in **QUAL** with common inputs $\mathbf{st}_r$,**pp**. Outputs a value Ω_r, a proof π_r update the state from $\mathbf{st}_r$ to $\mathbf{st}_{r+1}$.
- $\mathbf{DRNGVerify}(\mathbf{pp}, \Omega_r, \pi_r, \mathbf{st}_r)$ **:** Executed by anyone to verify the correctness of Ω_r. Outputs a bit $b \in \{0, 1\}$.



Security Properties

A secure DRNG protocol satisfies the following properties:[2]

- Pseudo-randomness: For any r, the output Ω_r is pseudo-random. More formally, if $(\Omega_r, \pi_r, \mathsf{st}_{r+1}) \leftarrow \mathsf{DRNGGen}(\mathsf{st}_r, \mathsf{pp})\langle\{P_i(\mathsf{sk}_i)\}_{i\in\mathsf{QUAL}}\rangle$, then for any computational bouned adversaries $\mathcal{A}$, there is a negligible function negl such that

$$|\Pr[\mathcal{A}(\Omega_r, (\Omega_i)_{i=1}^{r-1}) = 1] - \Pr[\mathcal{A}(U, (\Omega_i)_{i=1}^{r-1}) = 1]| \leq \mathsf{negl}(\lambda)$$

  where U is uniformly sampled in $\{0,1\}^\lambda$.

[2] https://eprint.iacr.org/2019/1320.pdf

Security Properties

A secure DRNG protocol satisfies the following properties:[2]

- Pseudo-randomness: For any r, the output Ω_r is pseudo-random. More formally, if $(\Omega_r, \pi_r, \mathsf{st}_{r+1}) \leftarrow \mathsf{DRNGGen}(\mathsf{st}_r, \mathsf{pp})\langle\{P_i(\mathsf{sk}_i)\}_{i\in\mathsf{QUAL}}\rangle$, then for any computational bouned adversaries $\mathcal{A}$, there is a negligible function **negl** such that

$$|\Pr[\mathcal{A}(\Omega_r, (\Omega_i)_{i=1}^{r-1}) = 1] - \Pr[\mathcal{A}(U, (\Omega_i)_{i=1}^{r-1}) = 1]| \leq \mathsf{negl}(\lambda)$$

  where U is uniformly sampled in $\{0,1\}^\lambda$.
- Unbiasability: Adversaries cannot control the distribution of the output for their own goal.

[2] https://eprint.iacr.org/2019/1320.pdf

Security Properties

A secure DRNG protocol satisfies the following properties:[2]

- Pseudo-randomness: For any r, the output Ω_r is pseudo-random. More formally, if $(\Omega_r, \pi_r, \mathsf{st}_{r+1}) \leftarrow \mathsf{DRNGGen}(\mathsf{st}_r, \mathsf{pp})\langle\{P_i(\mathsf{sk}_i)\}_{i \in \mathsf{QUAL}}\rangle$, then for any computational bouned adversaries $\mathcal{A}$, there is a negligible function **negl** such that

$$|\Pr[\mathcal{A}(\Omega_r, (\Omega_i)_{i=1}^{r-1}) = 1] - \Pr[\mathcal{A}(U, (\Omega_i)_{i=1}^{r-1}) = 1]| \leq \mathsf{negl}(\lambda)$$

  where U is uniformly sampled in $\{0,1\}^\lambda$.
- Unbiasability: Adversaries cannot control the distribution of the output for their own goal.
- Liveness (Availability): The protocol must produce an output. In other words, $\Omega_r \neq \perp$ for all r.

[2] https://eprint.iacr.org/2019/1320.pdf

Security Properties

A secure DRNG protocol satisfies the following properties:[2]

- Pseudo-randomness: For any r, the output Ω_r is pseudo-random. More formally, if $(\Omega_r, \pi_r, \mathsf{st}_{r+1}) \leftarrow \mathsf{DRNGGen}(\mathsf{st}_r, \mathsf{pp})\langle\{P_i(\mathsf{sk}_i)\}_{i\in\mathsf{QUAL}}\rangle$, then for any computational bouned adversaries $\mathcal{A}$, there is a negligible function **negl** such that

$$|\Pr[\mathcal{A}(\Omega_r, (\Omega_i)_{i=1}^{r-1}) = 1] - \Pr[\mathcal{A}(U, (\Omega_i)_{i=1}^{r-1}) = 1]| \leq \mathsf{negl}(\lambda)$$

  where U is uniformly sampled in $\{0, 1\}^\lambda$.
- Unbiasability: Adversaries cannot control the distribution of the output for their own goal.
- Liveness (Availability): The protocol must produce an output. In other words, $\Omega_r \neq \perp$ for all r.
- Public Verifiability: For any r, an external verifier can check the correctness of Ω_r using **DRNGVerify**.

[2] https://eprint.iacr.org/2019/1320.pdf

DRNG Classification

DRNGs are classified through their cryptographic primitives as follows:

Cryptographic Primitives	DRNG Constructions
Hash	Proof-of-work, RANDAO.
Publicly verifiable secret sharing	Randshare, SCRAPE, ALBATROSS.
Threshold signature	Drand, DFINITY.
Verifiable random function	Algorand, DVRF.
Homomorphic encryption	Nguyen et al., HERB.
Verifiable delay function	Minimal VDF Beacon, Harmony.

Advantages and disadvantages of DRNG constructions:

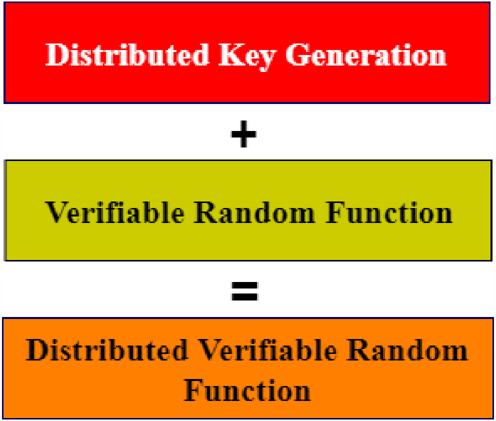
	Advantages	Disadvantages
Hash	Achieve $O(n^2)$ communication cost, $O(n)$ computation and verification cost.	Do not achieve Unbiasability and Liveness.
PVSS	Achieve full security properties.	Suffer $O(n^3)$ to $O(n^2 \log^2 n)$ computation and verification cost.
Thres. Sig.	- Achieve full security properties. - Achieve $O(n^2)$ communication cost.	Suffer $O(n \log^2 n)$ computation cost.
VRF	- Achieve $O(n^2)$ communication cost. - DVRF achieves full security properties. - Algorand enjoys $O(n)$ computation cost.	- DVRF suffers $O(n \log^2 n)$ computation cost. - Algorand does not achieve Unbiasability.
HE	- Achieve $O(n^2)$ communication cost. - HERB achieves full security properties. - Nguyen et al. enjoys $O(n)$ computation cost.	- HERB suffers $O(n \log^2 n)$ computation cost. - Nguyen et al. does not achieve Unbiasability.
VDF	- Achieve full security properties. - Achieve $O(n^2)$ communication cost and $O(n)$ verification cost.	Suffer very high computation cost.

Summary

	Pseudorandom	Unpredictable	Unbiasability	Liveness	Public Ver.	Comm. Cost	Comp. Cost	Verf. Cost	Primitives
Proof-of-work	✗	✓	✗	✗	✓	$\mathcal{O}(n^2)$	very high	$\mathcal{O}(1)$	Hash
RANDAO	✓	✓	✗	✗	✓	$\mathcal{O}(n^2)$	$\mathcal{O}(1)$	$\mathcal{O}(n)$	Hash
RandShare	✓	✓	✓	✓	✓	$\mathcal{O}(n^3)$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^3)$	PVSS
SCRAPE	✓	✓	✓	✓	✓	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2 \log^2 n)$	$\mathcal{O}(n^2 \log^2 n)$	PVSS
ALBATROSS	✓	✓	✓	✓	✓	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2 \log n)$	$\mathcal{O}(1)$	PVSS
Algorand	✗	✓	✗	✗	✓	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$	VRF
DVRF	✓	✓	✓	✓	✓	$\mathcal{O}(n^2)$	$\mathcal{O}(n \log^2 n)$	$\mathcal{O}(n \log^2 n)$	VRF
Nguyen19	✓	✓	✗	✓	✓	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	HE
HERB	✓	✓	✓	✓	✓	$\mathcal{O}(n^2)$	$\mathcal{O}(n \log^2 n)$	$\mathcal{O}(n \log^2 n)$	HE
Harmony	✗	✓	✓	✓	✓	$\mathcal{O}(n^2)$	very high	$\mathcal{O}(n)$	VDF
Minimal VDF	✗	✓	✓	✓	✓	$\mathcal{O}(n^2)$	very high	$\mathcal{O}(n)$	VDF
Drand	✗	✓	✓	✓	✓	$\mathcal{O}(n^2)$	$\mathcal{O}(n \log^2 n)$	$\mathcal{O}(n \log^2 n)$	BLS Signature
DFINITY	✗	✓	✓	✓	✓	$\mathcal{O}(n^2)$	$\mathcal{O}(n \log^2 n)$	$\mathcal{O}(n \log^2 n)$	BLS Signature

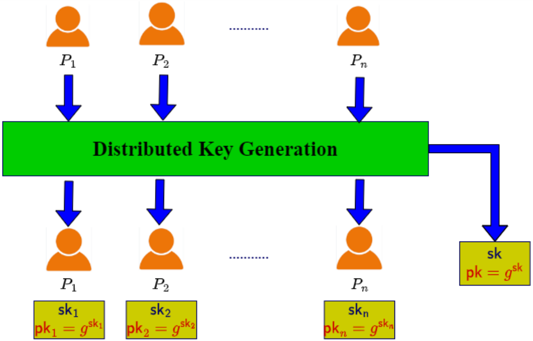
Table of Contents

Components

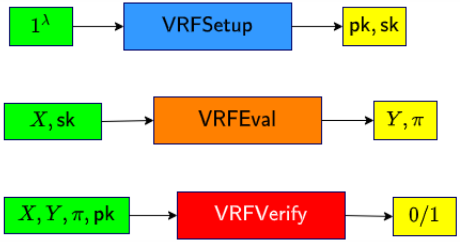

Distributed Key Generation
+
Verifiable Random Function
=
Distributed Verifiable Random Function

Component: Distributed Key Generation

- Executed jointly by n participants.
- Generate a public-secret key pair $(\mathsf{pk} = g^{\mathsf{sk}}, \mathsf{sk})$ and a partial public-secret key pair $(\mathsf{pk}_i = g^{\mathsf{sk}_i}, \mathsf{sk}_i)$ for each participant P_i.
- At least $t+1$ partial secret keys are required to restore sk.



Component: VRF Based on Elliptic Curves



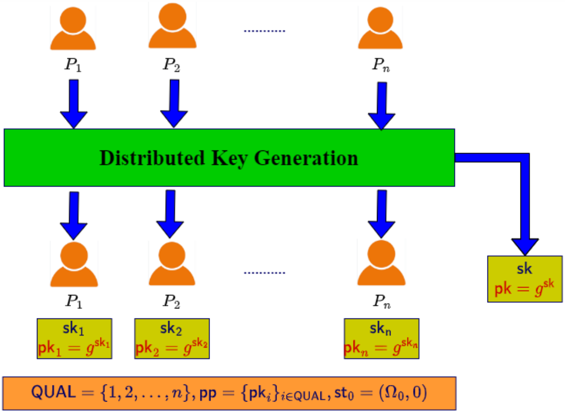
ECVRF construction: [3]

- The public key **pk** is $\mathbf{pk} = g^{\mathbf{sk}}$.
- Let $h = \mathrm{H}(X)$ and compute $Y = h^{\mathbf{sk}}$.
- Compute the proof π to prove the knowledge of **sk** such that $\mathbf{pk} = g^{\mathbf{sk}}$ and $Y = h^{\mathbf{sk}}$, then output (Y, π).

[3] https://eprint.iacr.org/2017/099.pdf

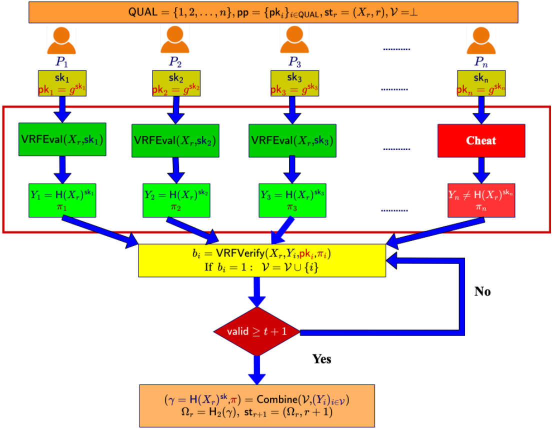
Construction: Setup

DRNGSetup(1^λ)$\langle\{P\}_{P\in\mathcal{P}}\rangle$: For simplicity, see figure below.



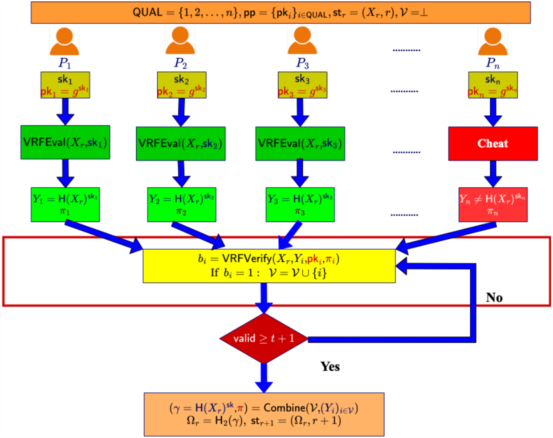
Construction: Generation

DRNGGen(st_r, pp)$\langle\{P_i(\mathsf{sk}_i)\}_{i\in\mathsf{QUAL}}\rangle$: For simplicity, see figure below.



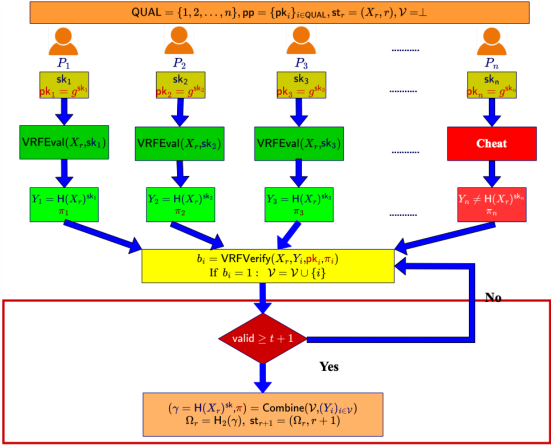
Construction: Generation

DRNGGen(st_r, pp)$\langle\{P_i(\mathsf{sk}_i)\}_{i\in \mathsf{QUAL}}\rangle$: For simplicity, see figure below.



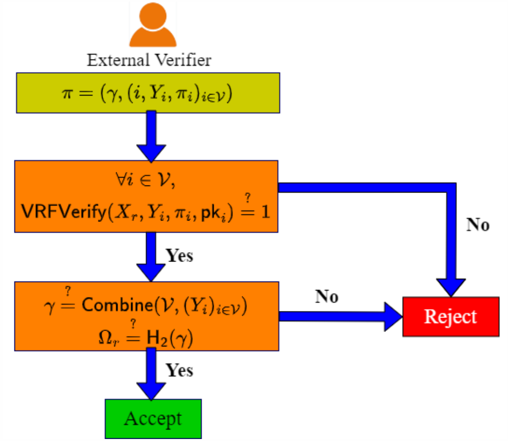
Construction: Generation

DRNGGen(st_r, pp)$\langle\{P_i(\mathsf{sk}_i)\}_{i\in\mathsf{QUAL}}\rangle$: For simplicity, see figure below.



Construction: Verification

DRNGVerify(pp, Ω_r, π_r, st$_r$) : For simplicity, see figure below.



Complexity Analysis

The communication, computation and verification complexity of the protocol depend on the number of participants, denoted by n as follows:

- Communication Complexity: $O(n^2)$.
- Computation Complexity per Participant: $O(n \log^2 n)$.
- Verification Complexity per Verifier: $O(n \log^2 n)$.

Table of Contents

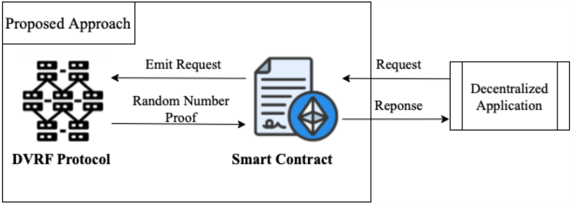
Randomness for Blockchain

- Various blockchain-based games (e.g., Axie Infinity) require randomness for distributing prizes, items.
- Existing blockchain systems employ older DRNGs such as RANDAO → Do not satisfy required security properties.
- We propose to use DRVF protocol above to generate randomness for such applications.

Figure: Axie Infinity [4]

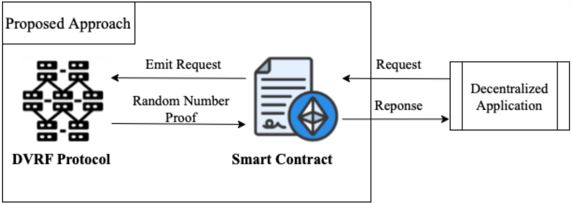
[4] https://axieinfinity.com

Proposed System Architecture



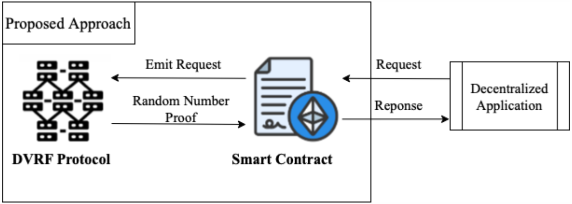
- Dapp: An application wishes to use a random number.

Proposed System Architecture



- Dapp: An application wishes to use a random number.
- Smart Contract: Forward random number request from Dapp to DVRF and verify the result from DVRF.

Proposed System Architecture



- Dapp: An application wishes to use a random number.
- Smart Contract: Forward random number request from Dapp to DVRF and verify the result from DVRF.
- DVRF: Used for generating randomness in a distributed manner.

Proposed System Workflow

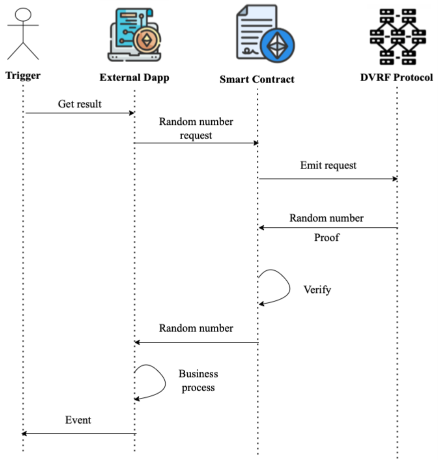


Table of Contents

NIST Test Suite

- Developed and maintained by the U.S. government.

NIST Test Suite

- Developed and maintained by the U.S. government.
- Widely used in the industry standard.

NIST Test Suite

- Developed and maintained by the U.S. government.
- Widely used in the industry standard.
- Consists of different tests to test the randomness of binary sequences produced by hardware or software-based cryptographic random or pseudo-random number generators following different statistical tests.

NIST Test Suite

- Developed and maintained by the U.S. government.
- Widely used in the industry standard.
- Consists of different tests to test the randomness of binary sequences produced by hardware or software-based cryptographic random or pseudo-random number generators following different statistical tests.
- We use NIST Test Suite to test our implementation with **20** numbers; each number is **100000** bits long.

Result

Test name	Pass rate
Frequency	20/20
Frequency in a Block	20/20
Run	20/20
Longest Run of Ones in a Block	19/20
Binary Matrix Rank	20/20
Discrete Fourier Transform	20/20
Non-Overlapping Template Matching	20/20
Overlapping Template Matching	20/20
Universal Statistical	20/20
Linear Complexity	19/20
Serial	20/20
Approximate Entropy	19/20
Cumulative Sums	20/20

In the test suite [5], for **20** numbers, the acceptable pass rate is equal to $\mathbf{0.99 \pm 3\sqrt{(0.99(1-0.99)/20} = [0.923, 1]} \rightarrow$ all test passed.

[5] `https://csrc.nist.gov/pubs/sp/800/22/r1/upd1/final`

Table of Contents

Conclusion and Future Work

Conclusion.

- A summary of the thesis has been accepted at IUKM 2023. [6]
- Formally defined the syntax and security properties of DRNGs.
- Conducted a systematic literature review of existing DRNGs.
- Specified a DRNG construction tailored for blockchain-based applications.
- Applied the aforementioned protocol to construct a DRNG system for blockchain-based applications.
- Assessed the security of the system using the NIST test suite.

Future Work.

- Construct DRNGs that are secure against quantum threats.
- Define DRNGs using the Universal Composable Security framework.

[6] https://www.jaist.ac.jp/IUKM/IUKM2023/

Thank You for Listening!