BFTRAND: Low-latency Random Number Provider for BFT Smart Contracts

Presenter:Jinghui Liao

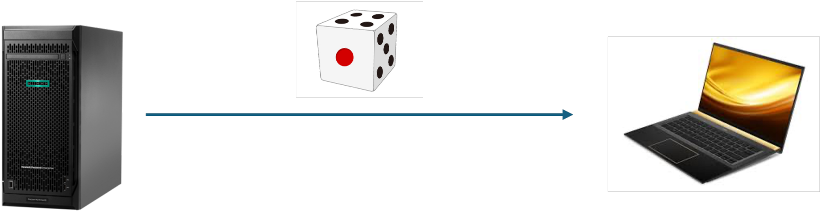
Other Authors: Borui Gong, Wenhai Sun, Fengwei Zhang, Zhenyu Ning, Man Ho Au, and Weisong Shi

Department of Computer Science and Engineering
Southern University of Science and Technology
COMPuter And System Security Lab

COMPASS Lab
COMPuter And System Security Lab

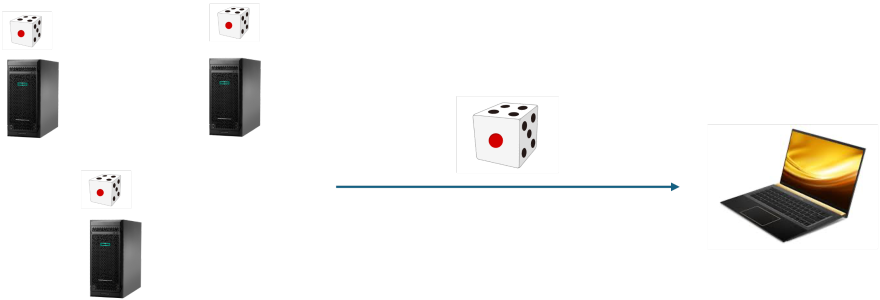
Outline

Server Client



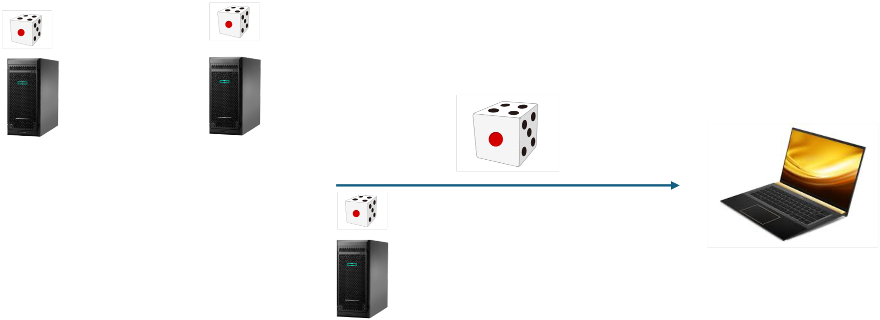
We do not trust single server.

Multi-Server Client



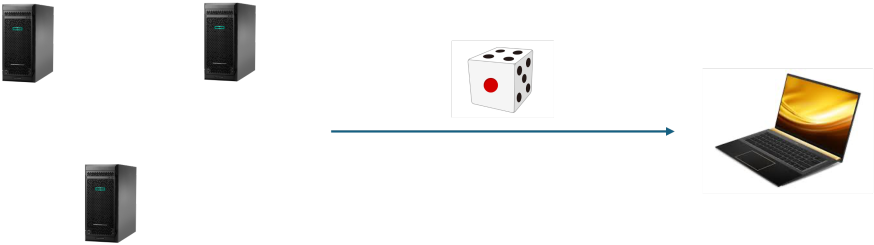
What bad guys can do?

Multi-Server Client: Bias Attack



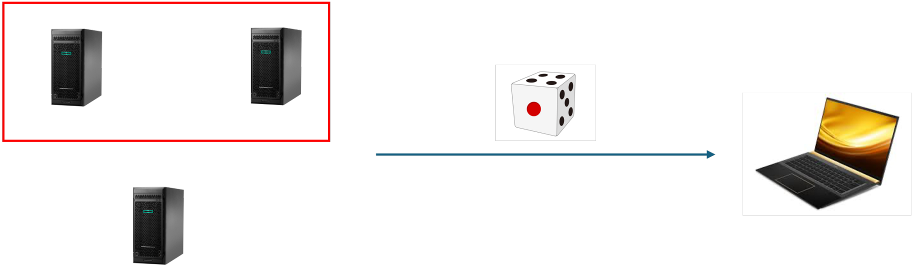
How about aggregation algorithms?

Multi-Server Client: DOS Attack



How to prevent bad guys from doing bad things?

Multi-Server Client: Threshold



Distributed Random Beacon (DRB).

Is DRB sufficient for Blockchain?

Is DRB sufficient for Blockchain?

NO

BFTRAND Overview

- A secure runtime random number generator for smart contracts
- Integrates distributed random beacons (DRB) into BFT consensus
- Achieves low latency and on-chain data savings

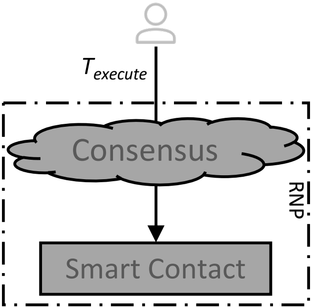


Figure: BFTRand RNP

Motivation

- Importance of randomness in blockchain applications
- Limitations of existing commit-reveal schemes
- Need for a secure and efficient runtime RNG

COMPASS Lab
COMPuter And System Security Lab

Challenges

- Integrating DRB without compromising consensus security
- Mitigating post-reveal undo attacks (PUA)
- Ensuring pseudo-randomness, uniqueness, and availability

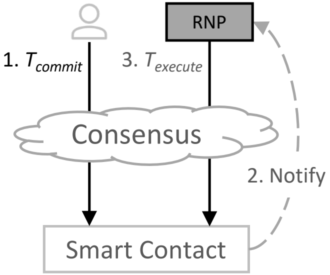
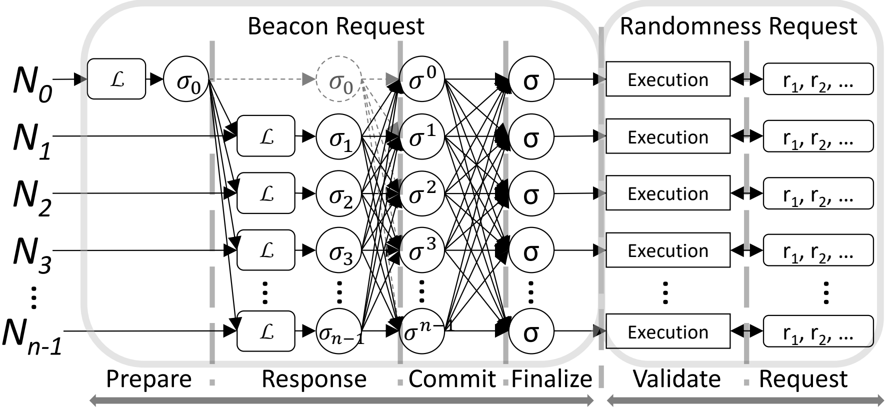


Figure: Commit-execute RNP

BFTRAND Protocol



Beacon Request

- Addressing semantic gaps between DRB and consensus
- Semantic Gap between DRB threshold k and BFT threshold t: $t < k \leq 2t + 1$
- Semantic Gap between DRB Round and Consensus Round: $\textbf{UpdateState}(\mathrm{st}_{b-1}, \perp, \perp, pk) : \mathrm{st} \leftarrow \sigma_{b-1} \parallel b - 1$ when $v = 0$, and $\textbf{UpdateState}(\mathrm{st}_{b,v-1}, \perp, \perp, pk) : \mathrm{st} \leftarrow \mathrm{st}_{b,v-1} \parallel b \parallel v$, otherwise.

COMPASS Lab
COMPuter And System Security Lab

Randomness Request

- Providing random numbers to smart contracts
- Utilizing pseudo-random functions (PRF) for efficiency
- Ensuring unique and unpredictable outputs

Post-reveal Undo Attack (PUA)

- A new attack on runtime RNG schemes
- Exploiting transaction atomicity to revert unfavorable results
- Identified four types: Contract, Fallback, Fee, and Script PUA

COMPASS Lab
COMPuter And System Security Lab

Post-reveal Undo Attack (PUA): Vulnerable BlindBox

```
function MintNFT(from, target, amount)
    CheckWitness(from)
    Require(amount == 1)
❶   Transfer(from, this, amount)
    rarity = GetRandom(1) % 2
    if (rarity == 1){
        // Cost 0.5 GAS
        blindBox = RareBlindBox(target, rarity)
❷   }else{
        // Cost 0.6 GAS
        blindBox = CommonBlindBox(target, rarity)
    }
    Mint(target, blindBox)
❸   Transfer(this, target, 1, {blindBox})
    StoreBlindBox(blindBox, rarity)
    return blindBox
end function

function GetRarity(blindBox)
    return StoreBlindBox[blindBox]
end function
```

Post-reveal Undo Attack (PUA): Contract PUA

```
function ContractPUA(user)
❹   blindBox = call NFT.MintNFT with (user, 1)
    rarity = call NFT.GetRarity with (blindBox)
    if rarity == 0 then
❺       call revert();
    end
    return true
end function
```

COMPASS Lab
COMPuter And System Security Lab

Post-reveal Undo Attack (PUA): Fallback PUA

```
function Fallback(from, amount, object)
    rarity = call NFT.GetRarity with (blindBox)
❻   if rarity == 0 then
        call revert();
    end
    return true
end function
```

Post-reveal Undo Attack (PUA): Fee-based PUA

```
function MintNFT(from, target, amount)
    CheckWitness(from)
    Require(amount == 1)
    Transfer(from, this, amount)
    rarity = GetRandom(1) % 2
    if (rarity == 1){
        // Cost 0.5 GAS
        blindBox = RareBlindBox(target, rarity)
    }else{
        // Cost 0.6 GAS
        // revert if fee insufficient
        blindBox = CommonBlindBox(target, rarity)
    }
    Mint(target, blindBox)
    Transfer(this, target, 1, {blindBox})
    StoreBlindBox(blindBox, rarity)
    return blindBox
end function
```

Post-reveal Undo Attack (PUA): Script-based PUA

```
load "Blindbox Contract"
push amount
push target
push from
call "MintNFT" // push blindBox to the stack
call "GetRarity" //pop blindBox, push rarity
jmpz revert() // if rarity == 0, jump to revert() ❾
```

COMPASS Lab
COMPuter And System Security Lab

Input Validation-based Detection (IVD)

- A countermeasure against PUA
- Validates user inputs and detects malicious transactions
- Maintains transaction atomicity and security properties

COMPASS Lab
COMPuter And System Security Lab

IVD

Input Validation-based Detection (IVD) $(C, \mathcal{T}, \sigma)$

For simplicity, we use the name of the contract as the contract address and the abstract invoking function of C as invoking C:

```
if ({C,tx} ≠ ⊥) {
    // Minimum transaction fee
    // and maximum script size from C
    ($G_min, S_max) ⟵ C;

    ($g_max, S) ⟵ tx
    // Fee PUA Detected
    if ($G_min ≤ $g_max) Revert();
    // Script PUA Detected
    if (S_max ≥ |S|) Revert()t;
    // Acquire the entry script
    // from the virtual machine
    C_entry ⟵ EntryScriptFromVM;
    // Contract PUA Detected
    if (C_entry ≠ S) Revert();
} else {
    // Get the script from the blockchain
    S ⟵ GetContract(Addr)
     // Fallback PUA Detected
    if (S ≠ ⊥) Revert();
}
```

Security Analysis - Pseudorandomness

- Leveraging the pseudo-randomness of DRB and PRF
- Ensuring unpredictable and unbiased random numbers
- Resilient against precomputation attacks

Security Analysis - Uniqueness

- Guaranteeing deterministic uniqueness of random numbers
- Mitigating replay attacks and validator collusion
- Utilizing threshold-based DRB and secure PRF

COMPASS Lab
COMPuter And System Security Lab

Security Analysis - Availability

- Ensuring consistent random number generation
- Tolerating Byzantine faults and DoS attacks
- Leveraging the robustness of the underlying BFT consensus

Implementation and Evaluation

- Prototype implementation on Neo blockchain
- Utilizing DBLS scheme for DRB and BLS signature
- Demonstrating efficiency and scalability advantages

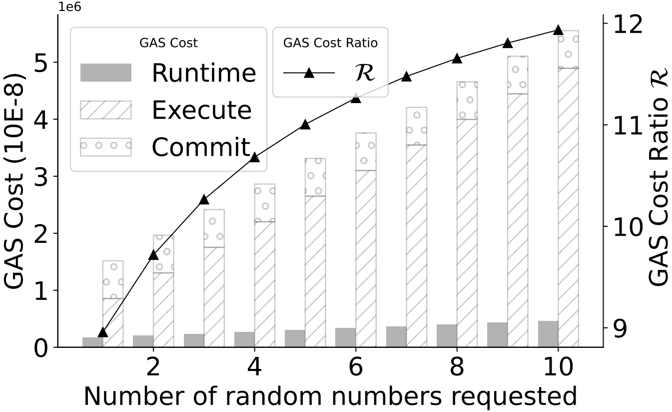
Evaluation - Transaction Cost

Table: Applications Transaction Fee (GAS/$).

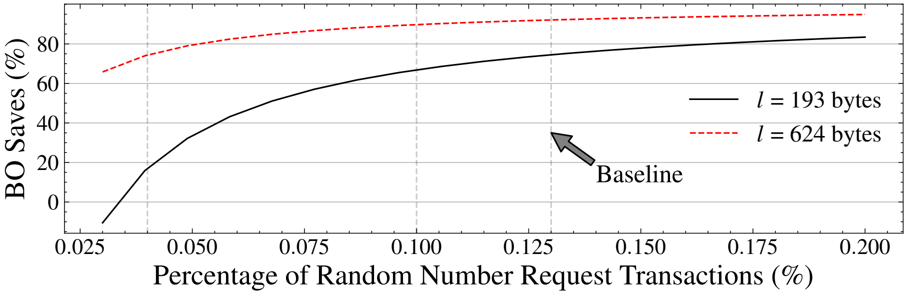
Method	**Network Fee**	**System Fee**
Loot::tokenURI	0.00593250/0.013	0.20694257/0.459
Neoverse:: UnBoxing	0.00119552/0.002	0.07313472/0.162
Neoverse:: BulkUnBoxing	0.00125752/0.002	0.36183988/0.803
RPS:: Play	0.00616260/0.013	0.06588677/0.146

Evaluation - GAS Cost

- The left y-axis is the total GAS consumption, while the right y-axis is the GAS cost ratio R = (Commit + Execute)/Runtime.



Evaluation - Blockchain Overhead



Comparison with Existing RNG Solutions

- Superior efficiency and scalability
- Lower on-chain storage and computational overhead
- Secure against various random number attacks

Protocol	Platform Consensus	Method(s)	Resistance (t)	# random values (r)	Latency (Consensus round)
Drand [31]	PABFT	Threshold SecretBLS	$t < n/2$	$\mathcal{O}(\sigma)$	≥ 2
HERB [36]	∅	Threshold ElGamal	$t < n/3$	$\mathcal{O}(\sigma)$	≥ 2
RandChain [56]	Sequential PoW	PoW	$t < n/3$	$\mathcal{O}(\sigma)$	≥ 2
RandHerd [93]	BFT	Threshold Schnorr	$t < n/3$	$\mathcal{O}(\sigma)$	≥ 2
RandHound [93]	BFT	Client based, PVSS	$t < n/3$	$\mathcal{O}(\sigma)$	≥ 2
BRandRiper [19]	BFT	VSS, q-SDH	$t < n/2$	$\mathcal{O}(\sigma)$	≥ 2
Dfinity [2]	BFT	Threshold BLS	$t < n/2$	∞	≥ 2
Secret [24]	DPoS	Scrt-RNG,TEE	$t < n/2$	∞	≥ 2
Elrond [21]	Secure PoS	BLS,onchain data	$t < n/3$	∞	≥ 2
Klaytn [23]	Istanbul BFT	VRF	$t < n/3$	$\mathcal{O}(\sigma)$	≥ 2
Harmoney [22]	Fast BFT	VRF,VDF	$t < n/3$	$\mathcal{O}(\sigma)$	≥ 2
⋆Chainlink VRF [35]	∅	VRF, TEE	$t < n/2$	$\mathcal{O}(\sigma)$	≥ 2
⋆Automata [76]	∅	VRF, TEE	$t < n/2$	∞	1
BFTRAND$_{commit-execute}$	BFT	∅	$t < n/3$	$\mathcal{O}(\sigma)$	≥ 2
BFTRAND	BFT	Threshold BLS	$t < n/3$	∞	1§

In the table, n denotes the number of consensus nodes, t is the maximum number of Byzantine nodes allowed in the system, and σ denotes the beacon. **Resistance** refers to the tolerance of the system for Byzantine faults. ⋆ is the off-chain third-party Oracle RNP. ∞ means the number of random numbers is upper-bounded by consensus. §BFTRAND is the first smart contract solution in runtime RNP on a BFT-based blockchain.

COMPASS Lab
COMPuter And System Security Lab

Thank You!

Questions and Comments are Welcome.