Provable Security of Linux-DRBG in the Seedless Robustness Model

ASIACRYPT 2024

Hwigyeom Kim
KAIST, KOREA

2024.12.11

Contents

Deterministic Random Bit Generator

2024.05.17

Deterministic Random Bit Generator

- **DRBG**
  - Generates random number with entropy source
  - Used to generating key, errors, etc.

- **Mathematical Definition of DRBG : the tuples of algirithms**
  - Setup : generate state S from entropy input I
  - Refresh : Using S, I, updates S
  - Next : Using S, generates random bits out

Standards for DRBG

Name	Primitive	Documents
HASH-DRBG	Hash function	NIST SP 800-90A , ISO 18031
HMAC-DRBG	Hash function	NIST SP 800-90A , ISO 18031
CTR-DRBG	Block cipher	NIST SP 800-90A , ISO 18031
OFB-DRBG	Block cipher	ISO 18031
Dual-EC-DRBG	Elliptic curve	NIST SP 800-90A (deleted in 2014)

DRBG in Operating System

- **Operating systems use DRBG**
  - Windows : AES-CTR
  - FreeBSD, iOS, macOS(past) : Yarrow
  - FreeBSD, iOS, macOS(current) : Fortuna
  - NetBSD, OpenBSD : self-designed DRBG(w/ ChaCha20)
  - Linux : self-designed DRBG(w/ Blake2s, ChaCha20)

- **Difficulty of mathematical analysis on OS DRBGs**
  - Uses ARX(add-rotate-xor) cipher or stream cipher
  - Ambiguous state definitions

Robustness of DRBG

2024.05.17

DRBG Provable Security history

- 2005: Robustness By Barak et.al.
- …
- 2013: Improved robustness model by dodis et.al.
- 2014
- 2015: Intel DRBG robustness proof
- …
- 2019: Seedless robustness by dodis et.al.
- 2019: HASH-DRBG, HMAC-DRBG robustness proof
- 2020: CTR-DRBG robustness proof

Distinguishing Game for robustness

Ideal World (S_0)

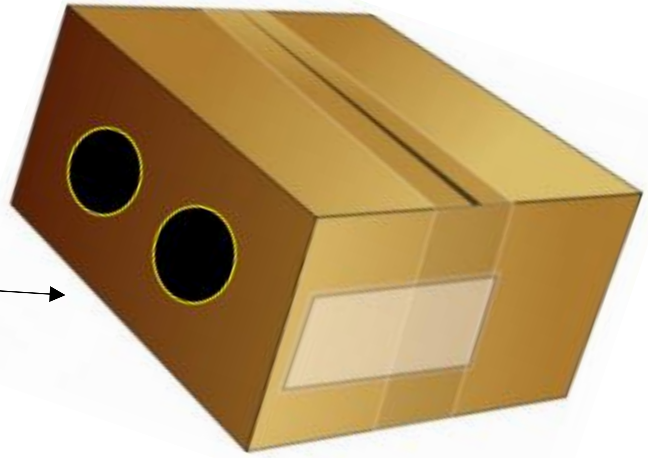
REF, GET SET, E, E^{-1}	ROR_ideal

Real World (S_1)

REF, GET SET, E, E^{-1}	ROR_real

S_1 ? S_0

q query



$Adv \approx 0$ when $q \ll 2^m$: The structure has m-bit security

Contributions

2024.05.17

Contribution 1 : Robustness model on Linux-DRBG

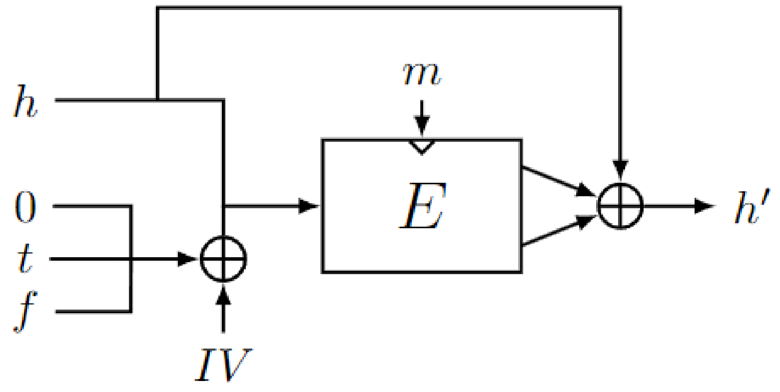
- **Linux**
  - Open source Operating system
  - Current Linux version : 6.4.8
  - No analysis on Linux 6.4.8 DRBG

- **Linux-DRBG**
  - Self-designed DRBG
  - Primitives : Hash function Blake2s, stream cipher ChaCha20
  - Do not fit in mathematical DRBG definitions
  - DRBG codes are in *drivers/char/random.c*

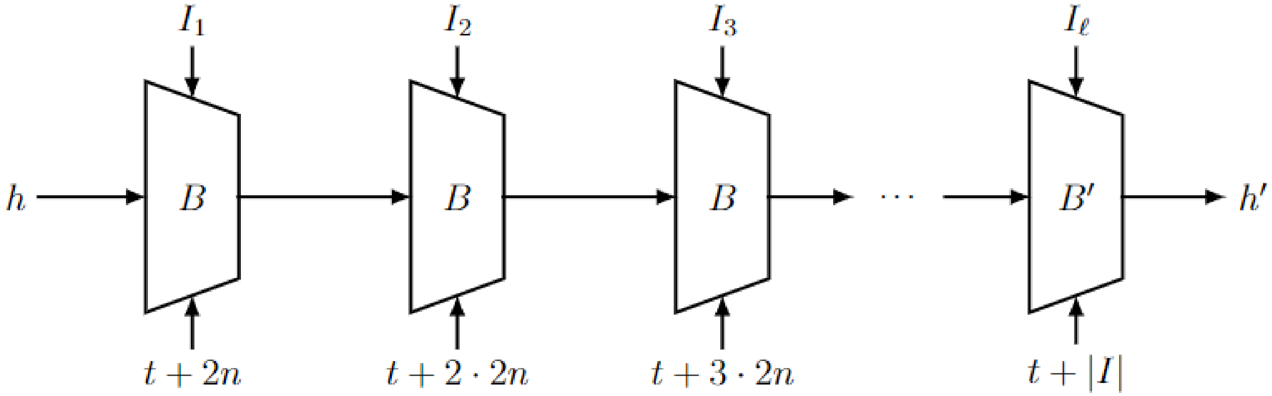
```c
static void _get_random_bytes(void *buf, size_t len)
{
    u32 chacha_state[CHACHA_STATE_WORDS];
    u8 tmp[CHACHA_BLOCK_SIZE];
    size_t first_block_len;

    if (!len)
        return;

    first_block_len = min_t(size_t, 32, len);
    crng_make_state(chacha_state, buf, first_block_len);
    len -= first_block_len;
    buf += first_block_len;

    while (len) {
        if (len < CHACHA_BLOCK_SIZE) {
            chacha20_block(chacha_state, tmp);
            memcpy(buf, tmp, len);
            memzero_explicit(tmp, sizeof(tmp));
            break;
        }

        chacha20_block(chacha_state, buf);
        if (unlikely(chacha_state[12] == 0))
            ++chacha_state[13];
        len -= CHACHA_BLOCK_SIZE;
        buf += CHACHA_BLOCK_SIZE;
    }

    memzero_explicit(chacha_state, sizeof(chacha_state));
}
```

Contribution 1 : Robustness model on Linux-DRBG

- **BLAKE2s hash function**
  - BLAKE : NIST hash function competition final candidate(lost to Keccak = SHA3)
  - BLAKE2s is based on BLAKE
  - 1 round of BLAKE2s can be modelled with (weakly ideal) block cipher



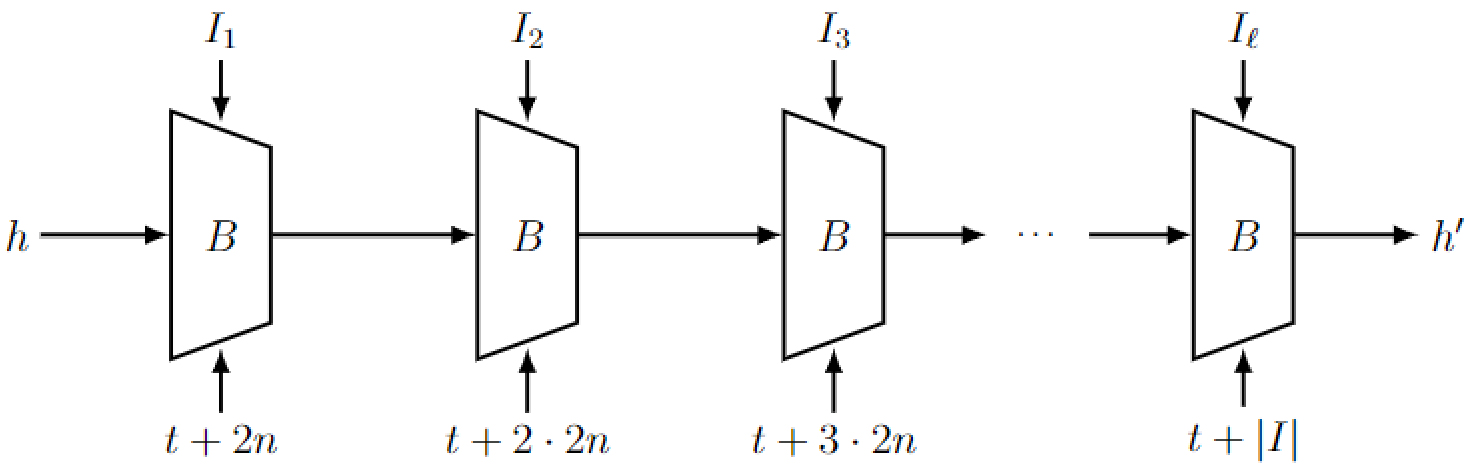
$\boldsymbol{BLAKE2s\ 1\ round\ (B)}$



$\boldsymbol{BLAKE2s\ \ (COMP)}$

Contribution 1 : Robustness model on Linux-DRBG

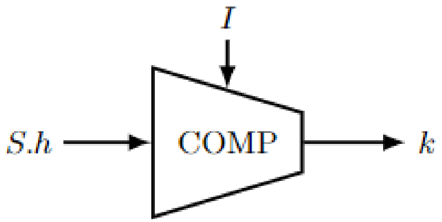
- **2 refresh functions**
  - Because BLAKE2s is a hash function and entropy is collected gradually, finalization is no need in entropy accumulation process
  - $refresh_a$: entropy accumulation without finalization
  - $refresh_f$: finalize and ChaCha20 key updates

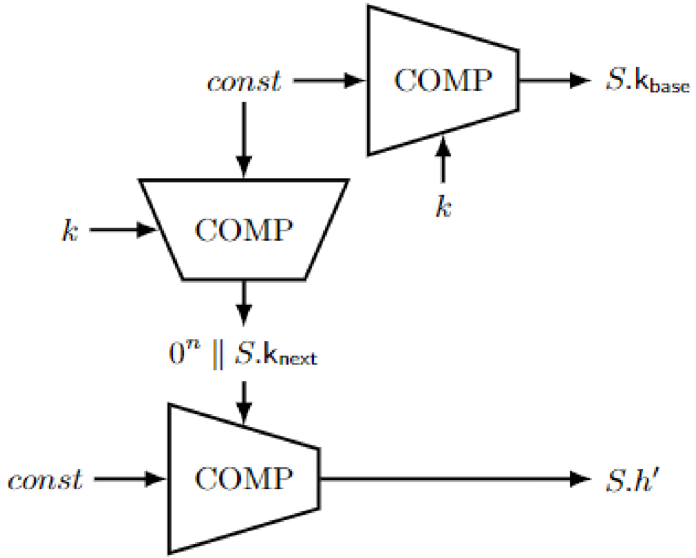


$\boldsymbol{refresh_a}$

Contribution 1 : Robustness model on Linux-DRBG

- **2 refresh functions**
  - Because BLAKE2s is a hash function and entropy is collected gradually, finalization is no need in entropy accumulation process
  - $refresh_a$: entropy accumulation without finalization
  - $refresh_f$: finalize and ChaCha20 key updates

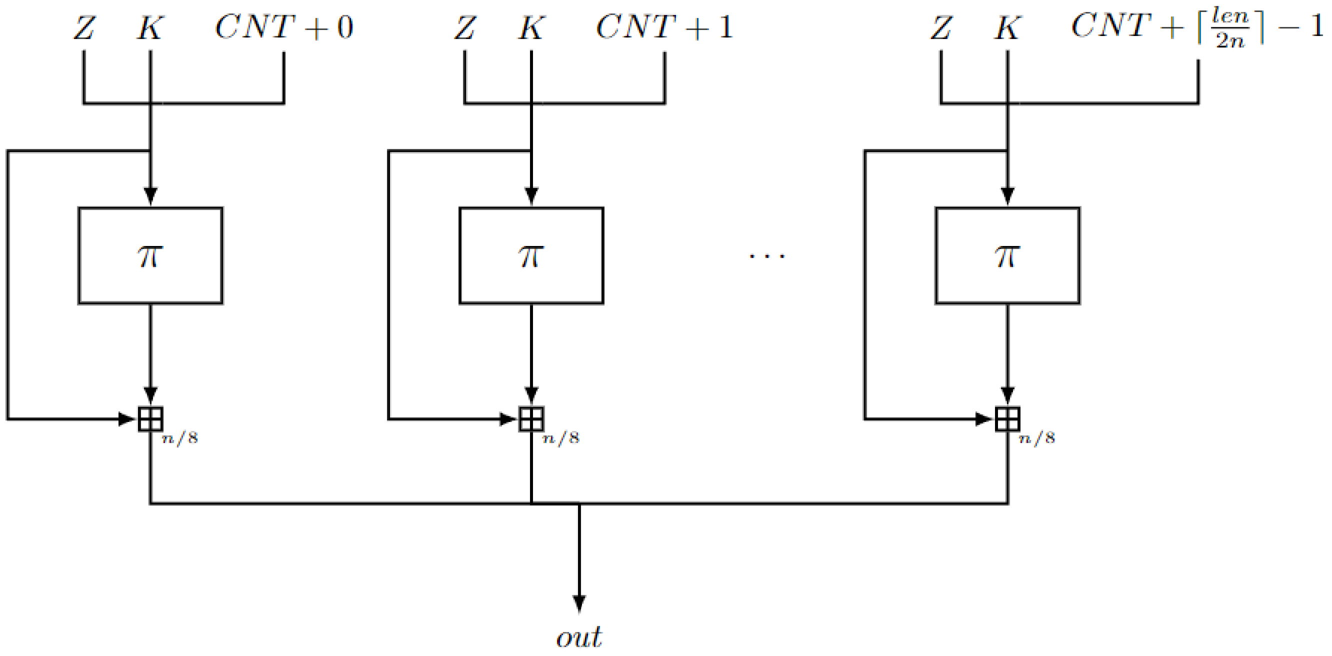




$\boldsymbol{refresh_f}$

Contribution 1 : Robustness model on Linux-DRBG

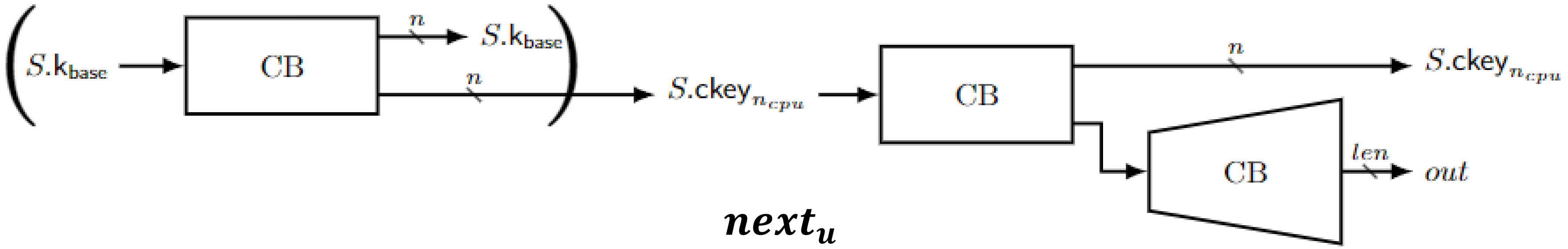
- **ChaCha20 stream cipher**
  - Salsa20 : eSTREAM project software portfolio
  - ChaCha20 is based on Salsa20
  - ChaCha20 can be modelled with ideal permutation



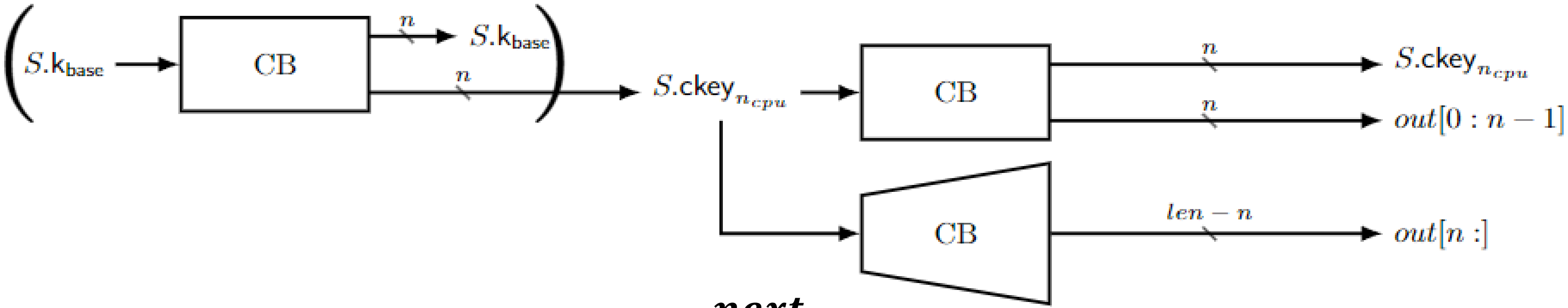
$\boldsymbol{ChaCha20}\ (\boldsymbol{CB})$

Contribution 1 : Robustness model on Linux-DRBG

- 2 next functions
  - Linux-DRBG supply 2 different next functions(seems redundant)
  - Modelled as $next_u$ and $next_k$



$\boldsymbol{next_u}$



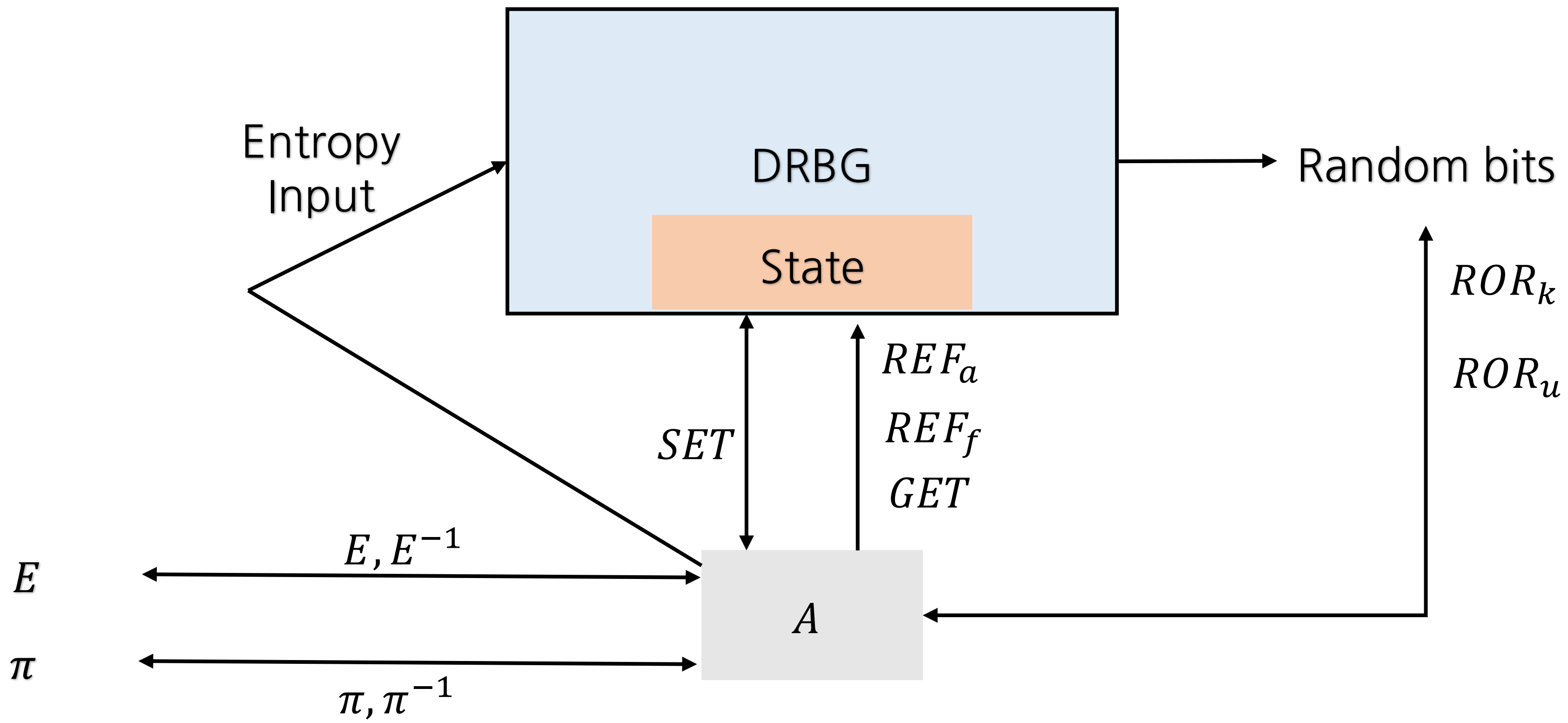
$\boldsymbol{next_k}$

Contribution 1 : Robustness model on Linux-DRBG

- **Other concerns**
  - Existence of ready flag
  - CPU core-dependent bit generating
  - Reseed term is always 60 seconds
  - Buffer to store entropy inputs

- **We re-defined (seedless) robustness game for Linux-DRBG**

Seedless Robustness Flow diagram (Linux)



Contribution 2 : New Reducing technique on Robustness

$$
\begin{aligned}
Adv_{rob}(p, q_1, q_2, \ell_1, \ell_2, \sigma_1, \sigma_2, \lambda) \leq\ & 2Adv_{M-EXT}(p + 3q_1 + \sigma_1, q_1, \sigma_1, \lambda) \\
& +2q_1(Adv_{pREF_a}(p + 3q_1 + \sigma_1, \ell_1) + Adv_{pREF_f}(p + 3q_1 + \sigma_1, \ell_1)) \\
& +2q_2(Adv_{bROR_u}(p + 2q_2 + \sigma_2, \ell_2) + Adv_{bROR_k}(p + 2q_2 + \sigma_2, \ell_2)) \\
& +2q_2(Adv_{cROR_u}(p + 2q_2 + \sigma_2, \ell_2) + Adv_{cROR_k}(p + 2q_2 + \sigma_2, \ell_2))
\end{aligned}
$$

Existing Subgame reduction(Dodis 13)

$SETUP \longrightarrow Refresh \longrightarrow Next \longrightarrow Next \longrightarrow Refresh \longrightarrow Next \longrightarrow \cdots$

Recovering Game 1 Preserving Game 1 Preserving Game 2

$\cdots \longrightarrow GET/SET \longrightarrow Refresh \longrightarrow Next \longrightarrow GET/SET \longrightarrow Refresh \longrightarrow Next$

Recovering Game $\boldsymbol{i}$ Recovering Game $\boldsymbol{i+1}$

$$Adv_{rob} \le q_2 \cdot \left(Adv_{rec} + Adv_{pres}\right) \le q(Adv_{ext} + Adv_{nxt}) + q(Adv_{mtn} + Adv_{nxt})$$

New Subgame reduction

$SETUP \longrightarrow REF_f \longrightarrow ROR_k \longrightarrow ROR_u \longrightarrow REF_a \longrightarrow REF_f \longrightarrow \cdots$

$\boldsymbol{pROR_k}$ Game 1 $\boldsymbol{pROR_u}$ Game 1 $\boldsymbol{pREF_a}$ Game 1 $\boldsymbol{pREF_f}$ Game 1

$\cdots \longrightarrow GET/SET \longrightarrow REF_f \longrightarrow ROR_k \longrightarrow GET/SET \longrightarrow REF_f \longrightarrow ROR_u$

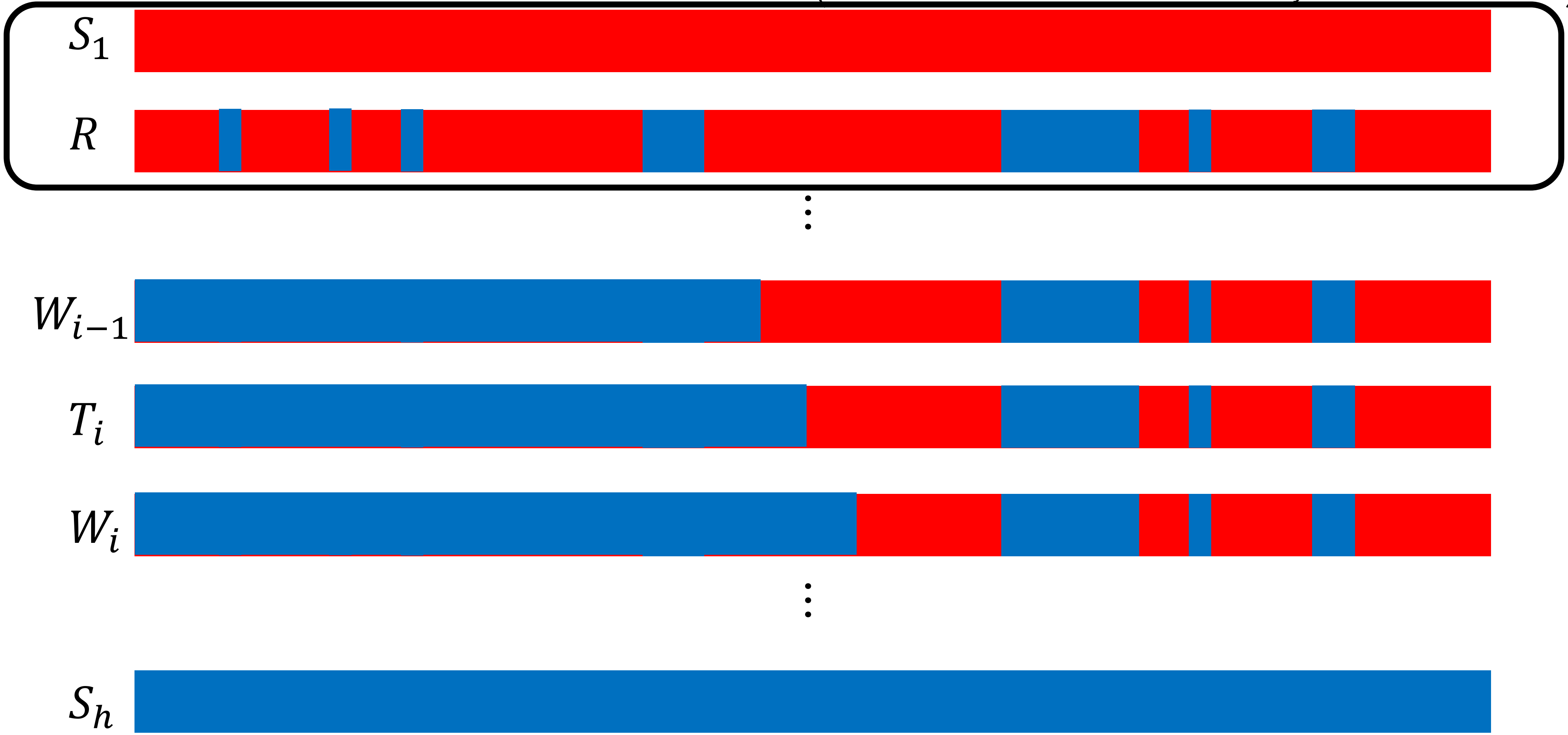
M-Ext Game

$\boldsymbol{pROR_k}$ Game $\mathrm{i} + 1$

$\boldsymbol{pROR_u}$ Game $\mathrm{i} + 2$

How to reduce robustness

$$Adv_{rob}^{MD}(A) = Adv_{S_1,S_0}(A) \leq 2Adv_{S_1,S_h}(A) \leq 2\left(Adv_{S_1,R} + \sum_{i=1}^{q_1} Adv_{W_{i-1},T_i}(A) + \sum_{j=1}^{q_2} Adv_{H_j^C,H_{j+1}^C}(A) \right)$$



Contribution 3 : Tight 128-bit security proof on robustness of Linux-DRBG

$$
\begin{aligned}
&Adv_{rob}(p, q_1, q_2, \ell_1, \ell_2, \sigma_1, \sigma_2, \lambda) \\
&\le \frac{14q_1}{2^{0.5n}} + \frac{8q_2\ell_2(p + 2q_2 + \ell_2 + \sigma_2)}{2^{2n}} + \frac{8q_1(p + 3q_1 + \sigma_1)}{2^{\lambda}} \\
&+ \frac{2p(p + 8q_2 + 27q_1 + 2\sigma_1) + 2q_1(72q_1 + 2\ell_1 + 31\sigma_1 + 2) + 4q_2(8q_2 + 4\sigma_2 + 1) + 4\sigma_1^2}{2^n}
\end{aligned}
$$

- $\boldsymbol{p}$: number of primitive queries
- $\boldsymbol{q_1}$: number of $\boldsymbol{REF_a}$ or $\boldsymbol{REF_f}$ queries
- $\boldsymbol{q_2}$: number of $\boldsymbol{ROR_u}$ or $\boldsymbol{ROR_k}$ queries
- $\boldsymbol{\ell_1}$: maximum block length per entropy input
- $\boldsymbol{\ell_2}$: maximum block length per random bit output
- $\boldsymbol{\sigma_1}$: total block length of entropy input
- $\boldsymbol{\sigma_2}$: total block length of random bit output
- $\boldsymbol{\lambda}$: Entropy threshold
- $\boldsymbol{n}$: primitive block size

For Linux-DRBG, $n = \lambda = 256$. Therefore, Linux DRBG has 128-bit security

Summary

- **Deterministic random bit generator**
  - Generates random number with entropy source
  - Standards : NIST 800-90A, ISO 18031
  - Operating systems implements DRBGs

- **Robustness of DRBG**
  - Mathematical analysis on the security of DRBG
  - Recently, seedless robustness model has been suggested

- **Contributions**
  - Contribution 1 : Robustness model on Linux-DRBG
  - Contribution 2 : New Reducing technique on robustness
  - Contribution 3 : Tight 128-bit security proof on robustness of Linux-DRBG

Thank you!