SoK: Security Models for Pseudo-Random Number Generators

Sylvain Ruhault

March 8^{th}, Tokyo, Fast Software Encryption 2017

OPPIDA
EXPERT EN SÉCURITÉ
DES SYSTÈMES D'INFORMATION

Motivation

Papers about PRNG

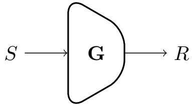
- FSE 96: Jenkins
- FSE 98: Schneier *et al.*
- Usenix 98: Gutman
- EC02: Desai *et al.*
- CT-RSA03: Bellare and Yee
- ACSAC03: Viega
- CHES03: Barak *et al.*
- CCS05: Barak and Halevi
- CCS10: Yu *et al.*
- CCS13: Dodis *et al.*
- C14: Dodis *et al.*
- ...

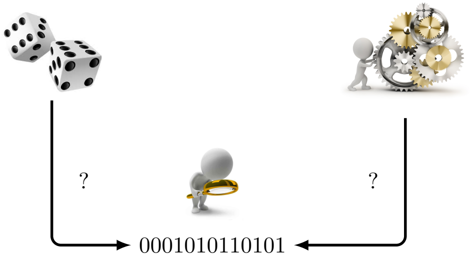
SoK Paper

- Unify security models presentation
- Propose secure constructions based on AES

1. Standard PRNG
2. Stateful PRNG
3. PRNG with input

Standard PRNG





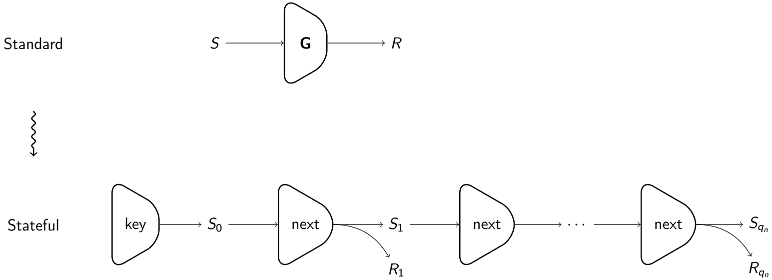
Security of **G**

- Secret S
- $|R| > |S|$
- R is indistinguishable from random.

AES based construction

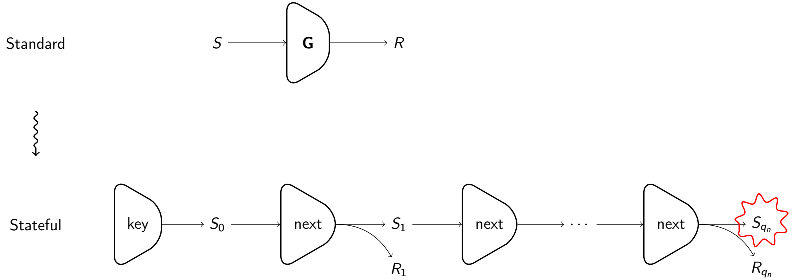
- $S \xleftarrow{\$} \{0,1\}^{128}$
- $R = \mathsf{AES}_S(1)||\mathsf{AES}_S(2)|| \cdots$

Stateful PRNG



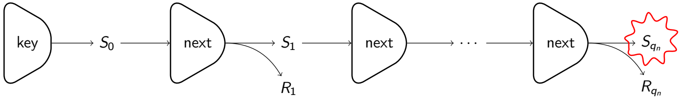
- $R_0, R_1, \cdots$ shall be indistinguishable from random
- S: internal state of the generator

Stateful PRNG



- $R_0, R_1, \cdots$ shall be indistinguishable from random
- S: internal state of the generator

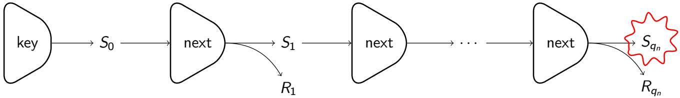
State Compromise



Forward Security

- Past outputs are not compromised
- Can be build upon a secure standard PRNG (BY03)

State Compromise



Forward Security

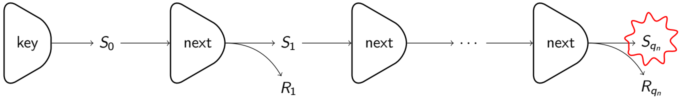
- Past outputs are not compromised
- Can be build upon a secure standard PRNG (BY03)

AES based construction

key
Require: $\varnothing$
Ensure: S
1: $S \xleftarrow{\$} \{0,1\}^{128}$
2: **return** S

next
Require: S
Ensure: S', R
1: $S' = \mathrm{AES}_S(1)$
2: $R = \mathrm{AES}_S(2)$
3: **return** (S', R)

State Compromise



Forward Security

- Past outputs are not compromised
- Can be build upon a secure standard PRNG (BY03)

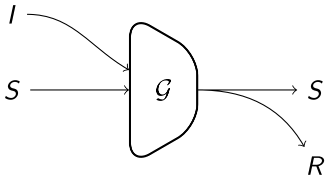
Backward Security ?

- "Next" outputs are not compromised ?

  $\rightsquigarrow$ New input shall be collected

  $\rightsquigarrow$ Recovery mechanism

PRNG with input



PRNG with input

How to Manage Inputs ?

- **Accumulation**: entropy of each input shall be accumulated in the internal state
- **Extraction**: entropy of the collected inputs shall be extracted to generate outputs

$\rightsquigarrow$ these operations are implicit in Fortuna, OpenSSL PRNG, `/dev/random`, NIST CTR_DRBG, ...

PRNG with input

How to Manage Inputs ?

- **Accumulation**: entropy of each input shall be accumulated in the internal state
- **Extraction**: entropy of the collected inputs shall be extracted to generate outputs

$\rightsquigarrow$ these operations are implicit in Fortuna, OpenSSL PRNG, `/dev/random`, NIST CTR_DRBG, ...

Definitions

- **Seeded** extractors, accumulators
- Requires **independence** between public seed and inputs

$\rightsquigarrow$ Potential vulnerability in NIST CTR_DRBG

Barak-Halevi Model (BH05)

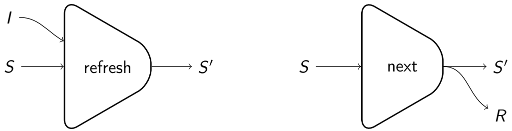
PRNG with input Definition

Two operations

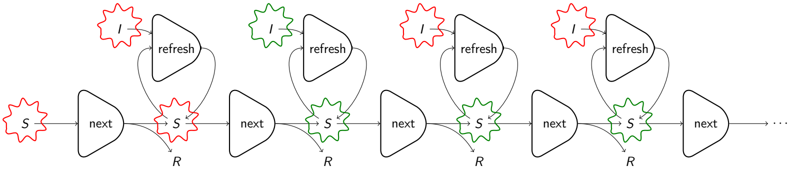
- input collection
- output generation

Where

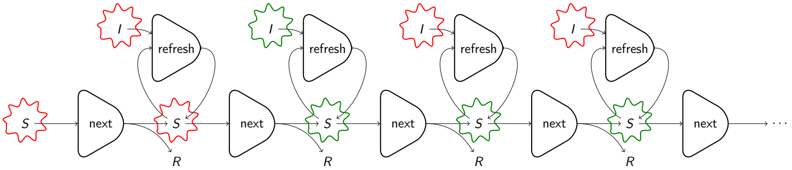
- Operations are not synchronised



Recovery in Barak-Halevi model



Recovery in Barak-Halevi model



AES based construction

setup	refresh	next
Require: r	**Require:** X, I, S	**Require:** S
Ensure: X	**Ensure:** S'	**Ensure:** S', R
1: $X \xleftarrow{\$} \{0,1\}^{512}$	1: $U = [X \cdot I]_{128}$	1: $S' = \mathrm{AES}_S(1)$
2: **return** X	2: $S' = S \oplus U$	2: $R = \mathrm{AES}_S(2)$
	3: **return** S'	3: **return** (S', R)

Security Analysis

AES based construction

setup	refresh	next
Require: r	**Require:** X, I, S	**Require:** S
Ensure: X	**Ensure:** S'	**Ensure:** S', R
1: $X \xleftarrow{\$} \{0,1\}^{512}$	1: $U = [X \cdot I]_{128}$	1: $S' = \mathrm{AES}_S(1)$
2: **return** X	2: $S' = S \oplus U$	2: $R = \mathrm{AES}_S(2)$
	3: **return** S'	3: **return** (S', R)

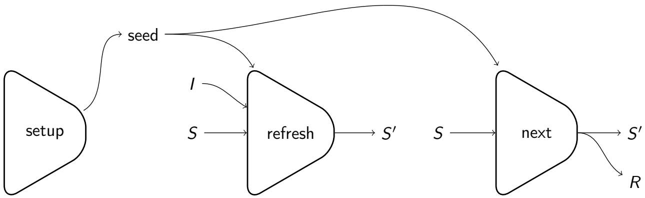
- $|S| = 128$
- Involves a **Seeded Extractor**
- At least one input shall have maximal entropy $\mathbf{H}_\infty(I) = 512$
- Requires a public random seed X of length 512 bits
- Inputs shall be independent from X

Dodis *et al.* Model (DPR+13)

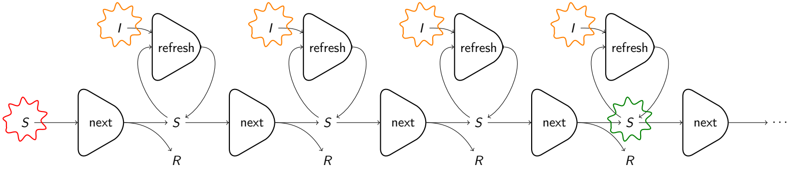
PRNG with input Definition

Triple of algorithms $(\mathsf{setup}, \mathsf{refresh}, \mathsf{next})$:

- setup, seed generation algorithm
- refresh, entropy collecting algorithm, $(S, I) \to S'$
- next, output algorithm, $S \to (R, S')$

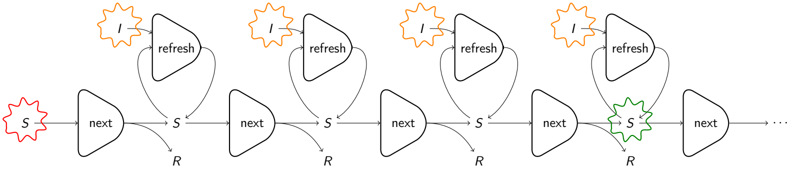


Recovery in Dodis *et al.* Model



- entropy can be accumulated slowly in S
- recovery: after accumulated entropy is OK

Recovery in Dodis *et al.* Model



AES based construction

setup

Require: $\varnothing$

Ensure: X, X'

1: $X \xleftarrow{\$} \{0,1\}^{1024}$

2: $X' \xleftarrow{\$} \{0,1\}^{1024}$

3: **return** X, X'

refresh

Require: X, I, S

Ensure: S'

1: $S' = S \cdot X + I$

2: **return** S'

next

Require: S, X'

Ensure: S', R

1: $U = [X' \cdot S]_{256}$

2: $S' = \mathrm{AES}_U(1)|| \cdots ||\mathrm{AES}_U(8)$

3: $R = \mathrm{AES}_U(9)$

4: **return** (S', R)

Security Analysis

AES based construction

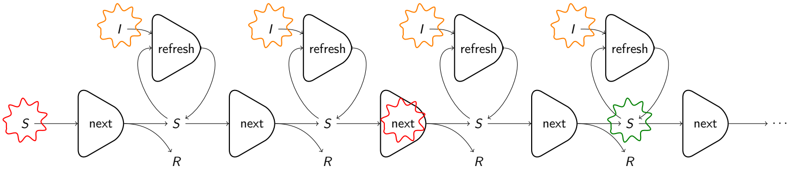
setup
Require: $\emptyset$
Ensure: X, X'
1: $X \xleftarrow{\$} \{0,1\}^{1024}$
2: $X' \xleftarrow{\$} \{0,1\}^{1024}$
3: **return** X, X'

refresh
Require: X, I, S
Ensure: S'
1: $S' = S \cdot X + I$
2: **return** S'

next
Require: S, X'
Ensure: S', R
1: $U = [X' \cdot S]_{256}$
2: $S' = \mathrm{AES}_U(1)||\cdots||\mathrm{AES}_U(8)$
3: $R = \mathrm{AES}_U(9)$
4: **return** (S', R)

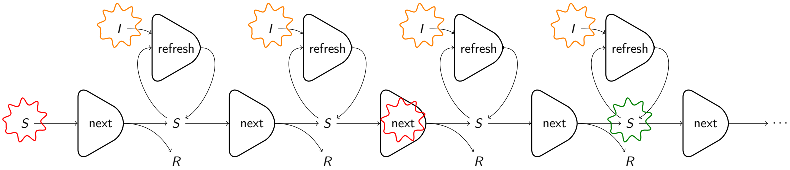
- $|S| = 1024$
- Involves a **Seeded Extractor** and a **Seeded Accumulator**
- Requires a public random (X, X') of length 2048 bits
- Inputs shall be independent from X
- Extensions has been proposed for Leakage Security [CR14, ABPRV15]

Premature Next Attack, Dodis *et al.* (DSSW14)



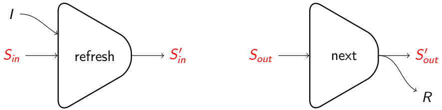
- a next call can be done before recovery

Premature Next Attack, Dodis *et al.* (DSSW14)



- a next call can be done before recovery

Solution: $S = [S_1 \cdots S_{in} \cdots S_{out} \cdots S_p]$, a **scheduler** selects S_{in} and S_{out}



Generalized Fortuna Construction (DSSW14)

$\mathcal{G}_i, i = 1, \cdots, 32$, based on AES

setup$_{\mathcal{G}}$
Require: $\varnothing$
Ensure: X, X'
1: $X \xleftarrow{\$} \{0,1\}^{1024}$
2: $X' \xleftarrow{\$} \{0,1\}^{1024}$
3: **return** X, X'

refresh$_i$
Require: X, I, S
Ensure: S'
1: $S' = S \cdot X + I$
2: **return** S'

next$_i$
Require: S, X'
Ensure: S', R
1: $U = [X' \cdot S]_{256}$
2: $S' = \mathrm{AES}_U(1)||\cdots||\mathrm{AES}_U(8)$
3: $R = \mathrm{AES}_U(9)||\mathrm{AES}_U(10)$
4: **return** (S', R)

AES based scheduler

- Uses AES as a PRF
- $(in, out) \leftarrow SC(skey)$

AES based construction

setup$_{\mathcal{G}}$
Require: $\varnothing$
Ensure: $X, X', skey$
1: $X, X' \leftarrow$ setup$_{\mathcal{G}}$
2: $skey \xleftarrow{\$} \{0,1\}^{128}$
3: **return** $X, X', skey$

refresh
Require: X, key, I, S
Ensure: S'
1: parse S as $(S_\rho, (S_i)_{i=0}^{31})$
2: $(in, out) \leftarrow SC(skey)$
3: $S_{in} \leftarrow \mathrm{refresh}_{in}(X, S_{in}, I)$
4: $(S_{out}, R) \leftarrow \mathrm{next}_{out}(X', S_{out})$
5: $S_\rho \leftarrow S_\rho \oplus R$
6: **return** $S' = (S_\rho, (S_i)_{i=0}^{31})$

next
Require: S
Ensure: S', R
1: parse S as $(S_\rho, (S_i)_{i=0}^{31})$
2: $S_\rho = \mathrm{AES}_{S_\rho}(1)||\mathrm{AES}_{S_\rho}(2)$
3: $R = \mathrm{AES}_{S_\rho}(3)||\mathrm{AES}_{S_\rho}(4)$
4: **return** (S', R)

Security Analysis

AES based construction

setup
Require: $\varnothing$
Ensure: $X, X', skey$
1: $X, X' \leftarrow \text{setup}_{\mathcal{G}}$
2: $skey \xleftarrow{\$} \{0,1\}^{128}$
3: **return** $X, X', skey$

refresh
Require: X, key, I, S
Ensure: S'
1: parse S as $(S_\rho, (S_i)_{i=0}^{31})$
2: $(in, out) \leftarrow SC(skey)$
3: $S_{in} \leftarrow \text{refresh}_{in}(X, S_{in}, I)$
4: $(S_{out}, R) \leftarrow \text{next}_{out}(X', S_{out})$
5: $S_\rho \leftarrow S_\rho \oplus R$
6: **return** $S' = (S_\rho, (S_i)_{i=0}^{31})$

next
Require: S
Ensure: S', R
1: parse S as $(S_\rho, (S_i)_{i=0}^{31})$
2: $S_\rho = \text{AES}_{S_\rho}(1) || \text{AES}_{S_\rho}(2)$
3: $R = \text{AES}_{S_\rho}(3) || \text{AES}_{S_\rho}(4)$
4: **return** (S', R)

- $S = (S_\rho, (S_i)_{i=0}^{31}), |S| = 33024$
- Involves a **Seeded Extractor** a **Seeded Accumulator** and a **Scheduler**
- Requires a public random (X, X') of length 2048 bits
- Inputs shall be independent from X
- Leakage Security shall be studied: SPOF: $S_\rho, |S_\rho| = 256$

Model and constructions analysis

Ref.	Definition	Property	Attacker Capabilities	Construction				Operations	
				AES	Ext	Acc	SC	refresh	next
BY03	$1: S \leftarrow \mathsf{key}$ $2: (S', R) \leftarrow \mathsf{next}(S)$	FWD	next-ror, get-state	✖					AES (2)
GMOPST14	$1: S \leftarrow \mathsf{key}$ $2: (S', R) \leftarrow \mathsf{next}(S)$	LPR(f)	next-ror, leaknext	✖					AES (3)
DHY02	$1: (K, S) \leftarrow \mathsf{key}$ $2: (S', R) \leftarrow \mathsf{next}(S, K, I)$	CIA	getinput, get-state, setinput						
		CSA	getinput, get-state, set-state	✖		✖			$+$ (3), $\times$ (2), AES (2)
		KKA	getinput, get-key						
BST03	$1: \mathsf{seed} \leftarrow \mathsf{setup}$ $2: R \leftarrow \mathsf{next}(\mathsf{seed}, I)$	RES($\mathcal{F}$)	next-ror		✖				$\times$ (1), [ ] (1)
BH05	$1: S' \leftarrow \mathsf{refresh}(S, I)$ $2: (S', R) \leftarrow \mathsf{next}(S)$	ROB($\mathcal{F}$)	good-refresh, bad-refresh, get-state, next-ror	✖	✖			$\times$ (1), [ ] (1), $\oplus$ (1)	AES (2)
DPRVW13	$1: \mathsf{seed} \leftarrow \mathsf{setup}$ $2: S' \leftarrow \mathsf{refresh}(\mathsf{seed}, S, I)$ $3: (S', R) \leftarrow \mathsf{next}(\mathsf{seed}, S)$	ROB(γ^*)	$\mathcal{D}$-refresh, set-state, get-state next-ror	✖	✖	✖		$\times$ (1), $+$ (1)	$\times$ (1), [ ] (1), AES (9)
DSSW14	$1: \mathsf{seed} \leftarrow \mathsf{setup}$ $2: S' \leftarrow \mathsf{refresh}(\mathsf{seed}, S, I)$ $3: (S', R) \leftarrow \mathsf{next}(\mathsf{seed}, S)$	NROB(γ^*, β)	$\mathcal{D}$-refresh, set-state, get-state next-ror	✖	✖	✖	✖	$+$ (1), $\times$ (2), $\oplus$ (1), [ ] (2), AES (11)	AES (4)

Conclusion

Contribution

- Revisited the notions of Extractors and Accumulators
- Unified the presentation of PRNG models
- Proposed AES based constructions
- Identified a potential vulnerability in NIST CTR_DRBG

Perpectives

- Independence requirement ?
- Leakage security of [DSSW14] construction ?
- Lightweight PRNG ?