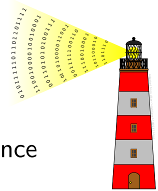
The new NIST reference for Randomness Beacons

Luís Brandão

Joint work with:
John Kelsey, René Peralta, Harold Booth

National Institute of Standards and Technology (Gaithersburg MD, USA)



Presentation at ICMC19
International Cryptographic Module Conference
May 17, 2019 @ Vancouver, Canada

Outline

1. Introduction
2. Pulse format
3. Beacon Protocol
4. Using a Beacon
5. Brief security considerations
6. Conclusion

Outline

1. Introduction

2. Pulse format

3. Beacon Protocol

4. Using a Beacon

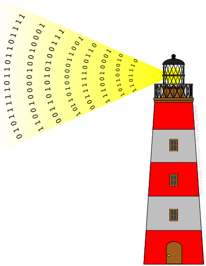
5. Brief security considerations

6. Conclusion

A Randomness Beacon

A Randomness Beacon

A service that produces timed outputs of fresh public randomness.

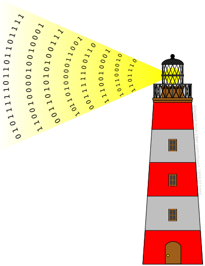


A Randomness Beacon

A service that produces timed outputs of fresh public randomness.

High-level description:

- Periodically *pulsates* randomness (e.g., 1 per min)

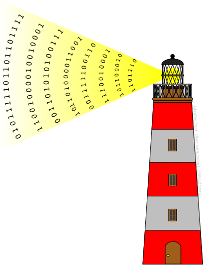


A Randomness Beacon

A service that produces timed outputs of fresh public randomness.

High-level description:

- Periodically *pulsates* randomness (e.g., 1 per min)
- Each pulse has a fresh 512-bit random string

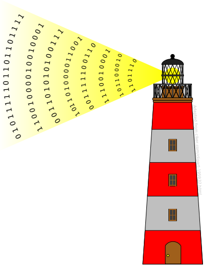


A Randomness Beacon

A service that produces timed outputs of fresh public randomness.

High-level description:

- Periodically *pulsates* randomness (e.g., 1 per min)
- Each pulse has a fresh 512-bit random string
- Each pulse is indexed, time-stamped and signed

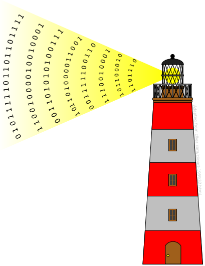


A Randomness Beacon

A service that produces timed outputs of fresh public randomness.

High-level description:

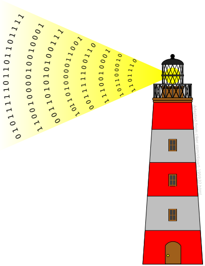
- Periodically *pulsates* randomness (e.g., 1 per min)
- Each pulse has a fresh 512-bit random string
- Each pulse is indexed, time-stamped and signed
- Any past pulse is publicly accessible



A Randomness Beacon

A service that produces timed outputs of fresh public randomness.

High-level description:

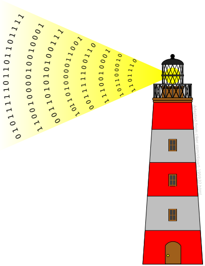


- Periodically *pulsates* randomness (e.g., 1 per min)
- Each pulse has a fresh 512-bit random string
- Each pulse is indexed, time-stamped and signed
- Any past pulse is publicly accessible
- The sequence of pulses forms a hash-chain

A Randomness Beacon

A service that produces timed outputs of fresh public randomness.

High-level description:



- Periodically *pulsates* randomness (e.g., 1 per min)
- Each pulse has a fresh 512-bit random string
- Each pulse is indexed, time-stamped and signed
- Any past pulse is publicly accessible
- The sequence of pulses forms a hash-chain

What can it be useful for?

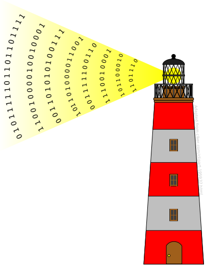
Not good for:

A Randomness Beacon

A service that produces timed outputs of fresh public randomness.

High-level description:

- Periodically *pulsates* randomness (e.g., 1 per min)
- Each pulse has a fresh 512-bit random string
- Each pulse is indexed, time-stamped and signed
- Any past pulse is publicly accessible
- The sequence of pulses forms a hash-chain



What can it be useful for?

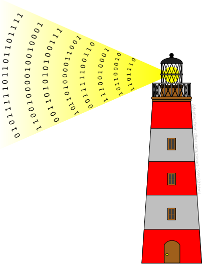
Not good for: selecting your secret keys

A Randomness Beacon

A service that produces timed outputs of fresh public randomness.

High-level description:

- Periodically *pulsates* randomness (e.g., 1 per min)
- Each pulse has a fresh 512-bit random string
- Each pulse is indexed, time-stamped and signed
- Any past pulse is publicly accessible
- The sequence of pulses forms a hash-chain



What can it be useful for?

- public auditability of randomized processes
- coordination between many parties
- prove something happened after a certain time

Not good for: selecting your secret keys

Brief historical note

Brief historical note

Some timeline events:

- 2013-Sep till 2018-Dec: NIST Beacon service version 1.0 online
- 2018-Jul till present: NIST Beacon service version 2.0 online
- 2019-May: “Draft NISTIR 8213” online — specifies the new (draft) *Reference for Randomness Beacons* (version 2)

Brief historical note

Some timeline events:

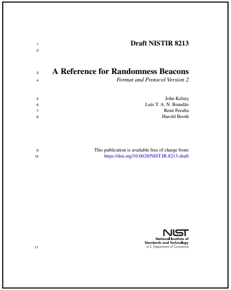
- 2013-Sep till 2018-Dec: NIST Beacon service version 1.0 online
- 2018-Jul till present: NIST Beacon service version 2.0 online
- 2019-May: "Draft NISTIR 8213" online — specifies the new (draft) *Reference for Randomness Beacons* (version 2)

The NIST Beacon will progressively implement all aspects of the Reference.

This talk is about the NISTIR 8213 (draft)

“A **Reference** for Randomness Beacons: Format and Protocol Version 2”

https://doi.org/10.6028/NIST.IR.8213-draft



Draft NISTIR 8213

A Reference for Randomness Beacons

Format and Protocol Version 2

John Kelsey
Luís T. A. N. Brandão
René Peralta
Harold Booth

This publication is available free of charge from:
https://doi.org/10.6028/NIST.IR.8213-draft

NIST
National Institute of
Standards and Technology
U.S. Department of Commerce

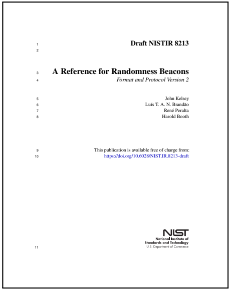
This talk is about the NISTIR 8213 (draft)

"A **Reference** for Randomness Beacons: Format and Protocol Version 2"

https://doi.org/10.6028/NIST.IR.8213-draft

Some topics in the report:

- format for pulses
- protocol for beacon operations
- using Beacon randomness
- security considerations



Draft NISTIR 8213

A Reference for Randomness Beacons

Format and Protocol Version 2

John Kelsey
Luís T. A. N. Brandão
René Peralta
Harold Booth

This publication is available free of charge from:
https://doi.org/10.6028/NIST.IR.8213-draft

NIST
National Institute of
Standards and Technology
U.S. Department of Commerce

This talk is about the NISTIR 8213 (draft)

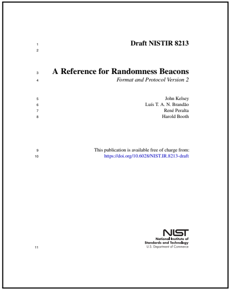
"A **Reference** for Randomness Beacons: Format and Protocol Version 2"

https://doi.org/10.6028/NIST.IR.8213-draft

Some topics in the report:

- format for pulses
- protocol for beacon operations
- using Beacon randomness
- security considerations

Public comments till August 05, 2019.



Draft NISTIR 8213

A Reference for Randomness Beacons

Format and Protocol Version 2

John Kelsey
Luís T. A. N. Brandão
René Peralta
Harold Booth

This publication is available free of charge from:
https://doi.org/10.6028/NIST.IR.8213-draft

NIST
National Institute of
Standards and Technology
U.S. Department of Commerce

This talk is about the NISTIR 8213 (draft)

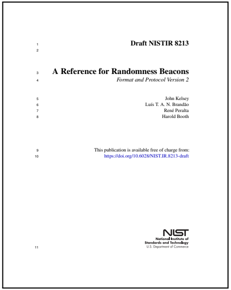
“A **Reference** for Randomness Beacons: Format and Protocol Version 2”

https://doi.org/10.6028/NIST.IR.8213-draft

Some topics in the report:

- format for pulses
- protocol for beacon operations
- using Beacon randomness
- security considerations

Public comments till August 05, 2019.



Draft NISTIR 8213

A Reference for Randomness Beacons

Format and Protocol Version 2

John Kelsey
Luís T. A. N. Brandão
René Peralta
Harold Booth

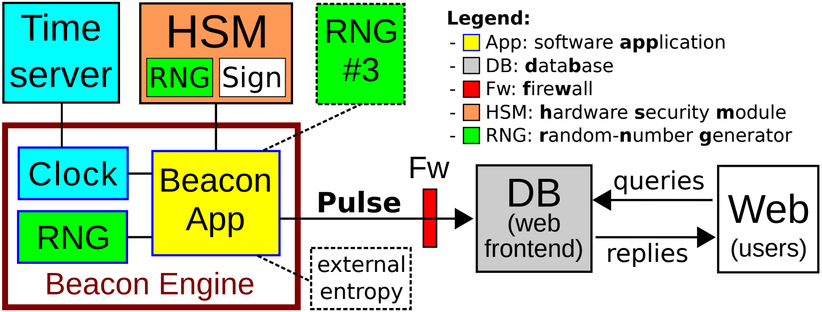
This publication is available free of charge from:
https://doi.org/10.6028/NIST.IR.8213-draft

NIST
National Institute of
Standards and Technology
U.S. Department of Commerce

Two goals in this presentation:

- Provide an overview of the new reference
- Motivate engagement: NISTIR feedback, new beacons and apps

Components of the Beacon service, at a high level



Time server

HSM

RNG

Sign

RNG #3

Clock

RNG

Beacon App

Beacon Engine

Pulse

Fw

external entropy

DB (web frontend)

queries

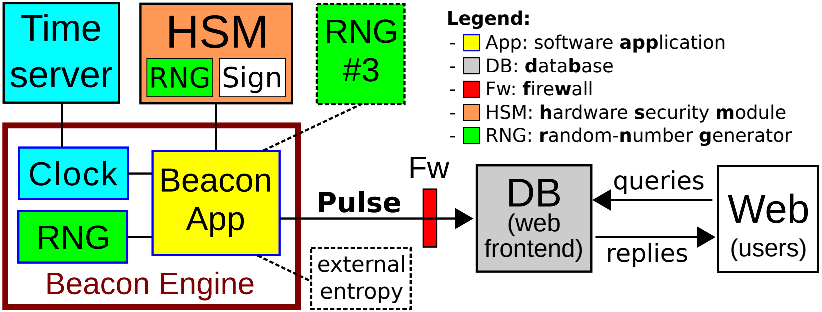
replies

Web (users)

Legend:

- 🟨 App: software **app**lication
- ⬜ DB: **d**ata**b**ase
- 🟥 Fw: **f**ire**w**all
- 🟧 HSM: **h**ardware **s**ecurity **m**odule
- 🟩 RNG: **r**andom-**n**umber **g**enerator

Components of the Beacon service, at a high level



Legend:
- 🟨 App: software **app**lication
- ⬜ DB: **d**ata**b**ase
- 🟥 Fw: **f**ire**w**all
- 🟧 HSM: **h**ardware **s**ecurity **m**odule
- 🟩 RNG: **r**andom-**n**umber **g**enerator

But what exactly is a *pulse*, what is its randomness, ...?

Outline

1. Introduction
2. Pulse format
3. Beacon Protocol
4. Using a Beacon
5. Brief security considerations
6. Conclusion

A pulse (simplified example)

```
uri:str="https://beacon.nist.gov/beacon/2.0/chain/1/pulse/220394"
version:str="2.0"
...
period:dec="60000"
...
chainId:dec="1"
pulseId:dec="220394"
time:str="2018-12-26T16:07:00.000Z"
randLocal:hex="5FF1E0C019C42C77FA72D522...(512 bits total)"
...
out.Prev:hex="BA646CC4E7AE195D2C85E9D3...(512 bits total)"
...
preCom:hex="269908B840E79BE71CEC4EBA...(512 bits total)"
...
sig:hex="17943D886DA8C7C24B9244BE...(4096 bits total)"
randOut:hex="0A8863E03E200F6940A009B0...(512 bits total)"
```

A pulse (simplified example)

```
uri:str="https://beacon.nist.gov/beacon/2.0/chain/1/pulse/220394"
version:str="2.0"
...
period:dec="60000"
...
chainId:dec="1"
pulseId:dec="220394"
time:str="2018-12-26T16:07:00.000Z"
randLocal:hex="5FF1E0C019C42C77FA72D522...(512 bits total)"
...
out.Prev:hex="BA646CC4E7AE195D2C85E9D3...(512 bits total)"
...
preCom:hex="269908B840E79BE71CEC4EBA...(512 bits total)"
...
sig:hex="17943D886DA8C7C24B9244BE...(4096 bits total)"
randOut:hex="0A8863E03E200F6940A009B0...(512 bits total)"
```

- Each pulse is indexed

A pulse (simplified example)

```
uri:str="https://beacon.nist.gov/beacon/2.0/chain/1/pulse/220394"
version:str="2.0"
...
period:dec="60000"
...
chainId:dec="1"
pulseId:dec="220394"
time:str="2018-12-26T16:07:00.000Z"
randLocal:hex="5FF1E0C019C42C77FA72D522...(512 bits total)"
...
out.Prev:hex="BA646CC4E7AE195D2C85E9D3...(512 bits total)"
...
preCom:hex="269908B840E79BE71CEC4EBA...(512 bits total)"
...
sig:hex="17943D886DA8C7C24B9244BE...(4096 bits total)"
randOut:hex="0A8863E03E200F6940A009B0...(512 bits total)"
```

- Each pulse is indexed

A pulse (simplified example)

```
uri:str="https://beacon.nist.gov/beacon/2.0/chain/1/pulse/220394"
version:str="2.0"
...
period:dec="60000"
...
chainId:dec="1"
pulseId:dec="220394"
time:str="2018-12-26T16:07:00.000Z"
randLocal:hex="5FF1E0C019C42C77FA72D522...(512 bits total)"
...
out.Prev:hex="BA646CC4E7AE195D2C85E9D3...(512 bits total)"
...
preCom:hex="269908B840E79BE71CEC4EBA...(512 bits total)"
...
sig:hex="17943D886DA8C7C24B9244BE...(4096 bits total)"
randOut:hex="0A8863E03E200F6940A009B0...(512 bits total)"
```

- Each pulse is indexed

A pulse (simplified example)

```
uri:str="https://beacon.nist.gov/beacon/2.0/chain/1/pulse/220394"
version:str="2.0"
...
period:dec="60000"
...
chainId:dec="1"
pulseId:dec="220394"
time:str="2018-12-26T16:07:00.000Z"
randLocal:hex="5FF1E0C019C42C77FA72D522...(512 bits total)"
...
out.Prev:hex="BA646CC4E7AE195D2C85E9D3...(512 bits total)"
...
preCom:hex="269908B840E79BE71CEC4EBA...(512 bits total)"
...
sig:hex="17943D886DA8C7C24B9244BE...(4096 bits total)"
randOut:hex="0A8863E03E200F6940A009B0...(512 bits total)"
```

- Each pulse is indexed
- Two main random values ("rands"): `randLocal` and `randOut`.

A pulse (simplified example)

```
uri:str="https://beacon.nist.gov/beacon/2.0/chain/1/pulse/220394"
version:str="2.0"
...
period:dec="60000"
...
chainId:dec="1"
pulseId:dec="220394"
time:str="2018-12-26T16:07:00.000Z"
randLocal:hex="5FF1E0C019C42C77FA72D522...(512 bits total)"
...
out.Prev:hex="BA646CC4E7AE195D2C85E9D3...(512 bits total)"
...
preCom:hex="269908B840E79BE71CEC4EBA...(512 bits total)"
...
sig:hex="17943D886DA8C7C24B9244BE...(4096 bits total)"
randOut:hex="0A8863E03E200F6940A009B0...(512 bits total)"
```

- Each pulse is indexed
- Two main random values ("rands"): `randLocal` and `randOut`.
- Other features: signature

A pulse (simplified example)

```
uri:str="https://beacon.nist.gov/beacon/2.0/chain/1/pulse/220394"
version:str="2.0"
...
period:dec="60000"
...
chainId:dec="1"
pulseId:dec="220394"
time:str="2018-12-26T16:07:00.000Z"
randLocal:hex="5FF1E0C019C42C77FA72D522...(512 bits total)"
...
out.Prev:hex="BA646CC4E7AE195D2C85E9D3...(512 bits total)"
...
preCom:hex="269908B840E79BE71CEC4EBA...(512 bits total)"
...
sig:hex="17943D886DA8C7C24B9244BE...(4096 bits total)"
randOut:hex="0A8863E03E200F6940A009B0...(512 bits total)"
```

- Each pulse is indexed
- Two main random values ("rands"): `randLocal` and `randOut`.
- Other features: signature, precommit `randLocal`

A pulse (simplified example)

```
uri:str="https://beacon.nist.gov/beacon/2.0/chain/1/pulse/220394"
version:str="2.0"
...
period:dec="60000"
...
chainId:dec="1"
pulseId:dec="220394"
time:str="2018-12-26T16:07:00.000Z"
randLocal:hex="5FF1E0C019C42C77FA72D522...(512 bits total)"
...
out.Prev:hex="BA646CC4E7AE195D2C85E9D3...(512 bits total)"
...
preCom:hex="269908B840E79BE71CEC4EBA...(512 bits total)"
...
sig:hex="17943D886DA8C7C24B9244BE...(4096 bits total)"
randOut:hex="0A8863E03E200F6940A009B0...(512 bits total)"
```

- Each pulse is indexed
- Two main random values (“rands”): `randLocal` and `randOut`.
- Other features: signature, precommit `randLocal`, chain `randOut`, ...

The two “rands” in a pulse

The two "rands" in a pulse

randLocal (a.k.a. local random value):

randOut (a.k.a. output value):

The two “rands” in a pulse

randLocal (a.k.a. local random value):

- Hash (SHA512) of randomness output by ≥ 2 RNGs
- Pre-committed 1 minute in advance of release
- Useful for combining beacons

randOut (a.k.a. output value):

The two “rands” in a pulse

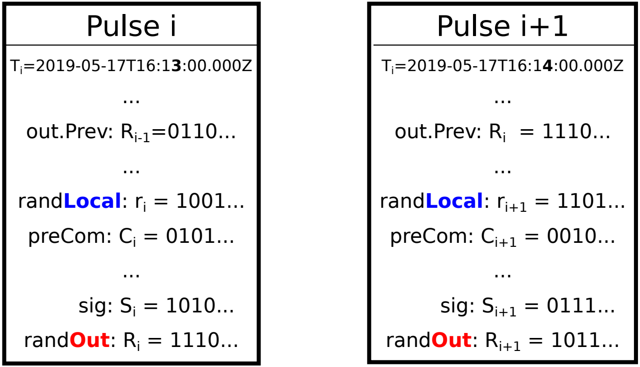
randLocal (a.k.a. local random value):

- Hash (SHA512) of randomness output by ≥ 2 RNGs
- Pre-committed 1 minute in advance of release
- Useful for combining beacons

randOut (a.k.a. output value):

- Hash of all other fields
- **Fresh** at the time of release
- The actual randomness to be used by applications

The two “rands” in a pulse



Pulse i

T_i=2019-05-17T16:1**3**:00.000Z

...

out.Prev: $R_{i\text{-}1}$=0110...

...

rand**Local**: r_i = 1001...

preCom: C_i = 0101...

...

sig: S_i = 1010...

rand**Out**: R_i = 1110...

Pulse i+1

T_i=2019-05-17T16:1**4**:00.000Z

...

out.Prev: R_i = 1110...

...

rand**Local**: r_{i+1} = 1101...

preCom: C_{i+1} = 0010...

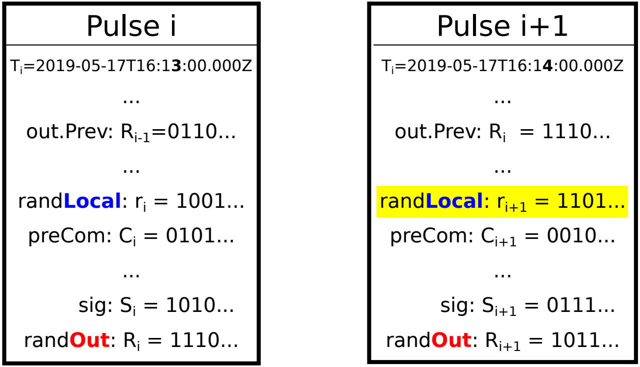
...

sig: S_{i+1} = 0111...

rand**Out**: R_{i+1} = 1011...

The two “rands” in a pulse

`randLocal:` $r_{i+1} = \text{Hash}(\rho_{1,i} \mid\mid \rho_{2,i} \ [\mid\mid \ \rho_{3,i}])$, with random $\rho_{j,i}$ from i^{th} RNG



Pulse i

T_i=2019-05-17T16:1**3**:00.000Z

...

out.Prev: R_{i-1}=0110...

...

rand**Local**: r_i = 1001...

preCom: C_i = 0101...

...

sig: S_i = 1010...

rand**Out**: R_i = 1110...

Pulse i+1

T_i=2019-05-17T16:1**4**:00.000Z

...

out.Prev: R_i = 1110...

...

rand**Local**: r_{i+1} = 1101...

preCom: C_{i+1} = 0010...

...

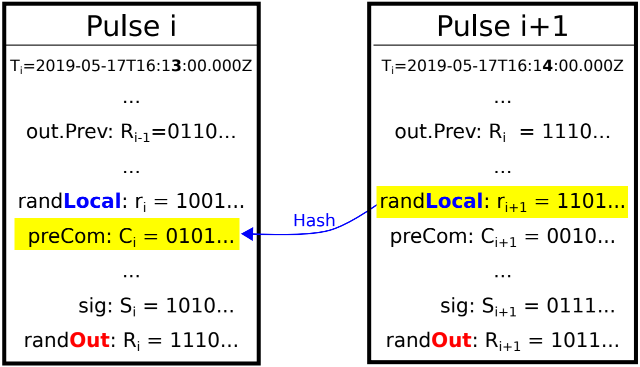
sig: S_{i+1} = 0111...

rand**Out**: R_{i+1} = 1011...

The two "rands" in a pulse

randLocal: $r_{i+1} = \text{Hash}(\rho_{1,i} \mid\mid \rho_{2,i} \; [\mid\mid \rho_{3,i}])$, with random $\rho_{j,i}$ from i^{th} RNG

preCom: $C_i = \text{Hash}(r_{i+1})$, released 1 min before r_{i+1}



Pulse i

T_i=2019-05-17T16:1**3**:00.000Z

...

out.Prev: $R_{i\text{-}1}$=0110...

...

rand**Local**: r_i = 1001...

preCom: C_i = 0101...

...

sig: S_i = 1010...

rand**Out**: R_i = 1110...

Hash

Pulse i+1

T_i=2019-05-17T16:1**4**:00.000Z

...

out.Prev: R_i = 1110...

...

rand**Local**: r_{i+1} = 1101...

preCom: C_{i+1} = 0010...

...

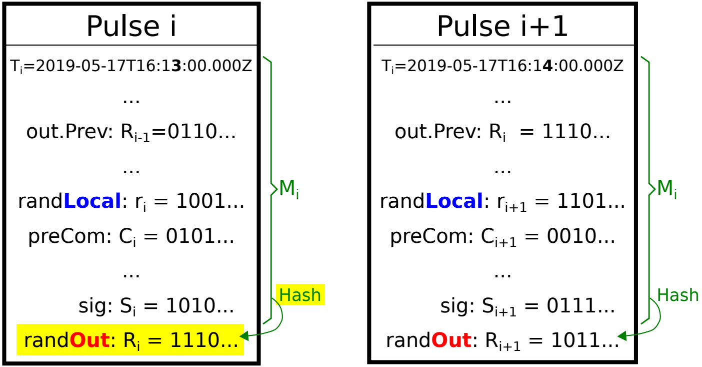
sig: S_{i+1} = 0111...

rand**Out**: R_{i+1} = 1011...

The two "rands" in a pulse

`randLocal`: $r_{i+1} = \text{Hash}(\rho_{1,i} \,||\, \rho_{2,i}\ [||\ \rho_{3,i}])$, with random $\rho_{j,i}$ from i^{th} RNG

`preCom`: $C_i = \text{Hash}(r_{i+1})$, released 1 min before r_{i+1}



Pulse i

T_i=2019-05-17T16:1**3**:00.000Z

...

out.Prev: R_{i-1}=0110...

...

rand**Local**: r_i = 1001...

preCom: C_i = 0101...

...

sig: S_i = 1010...

rand**Out**: R_i = 1110...

Pulse i+1

T_i=2019-05-17T16:1**4**:00.000Z

...

out.Prev: R_i = 1110...

...

rand**Local**: r_{i+1} = 1101...

preCom: C_{i+1} = 0010...

...

sig: S_{i+1} = 0111...

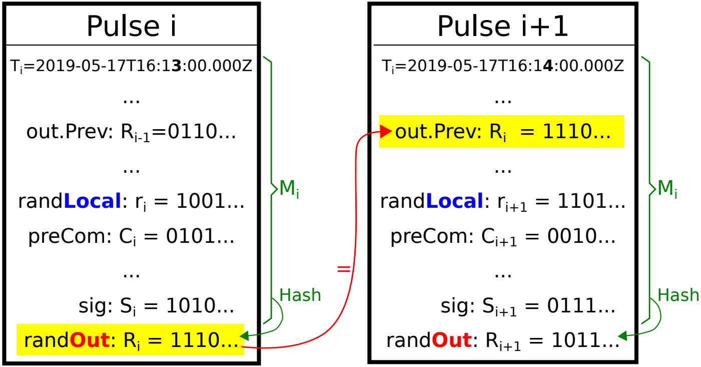
rand**Out**: R_{i+1} = 1011...

`randOut`: R_i $= \text{Hash}(M_i)$, with M_i being the serialization of all previous fields

The two “rands” in a pulse

`randLocal`: $r_{i+1} = \text{Hash}(\rho_{1,i} \mid\mid \rho_{2,i} \; [\mid\mid \; \rho_{3,i}])$, with random $\rho_{j,i}$ from i^{th} RNG

`preCom`: $C_i = \text{Hash}(r_{i+1})$, released 1 min before r_{i+1}



Pulse i

T_i=2019-05-17T16:1**3**:00.000Z

...

out.Prev: R_{i-1}=0110...

...

rand**Local**: r_i = 1001...

preCom: C_i = 0101...

...

sig: S_i = 1010...

rand**Out**: R_i = 1110...

M_i

Hash

=

Pulse i+1

T_i=2019-05-17T16:1**4**:00.000Z

...

out.Prev: R_i = 1110...

...

rand**Local**: r_{i+1} = 1101...

preCom: C_{i+1} = 0010...

...

sig: S_{i+1} = 0111...

rand**Out**: R_{i+1} = 1011...

M_i

Hash

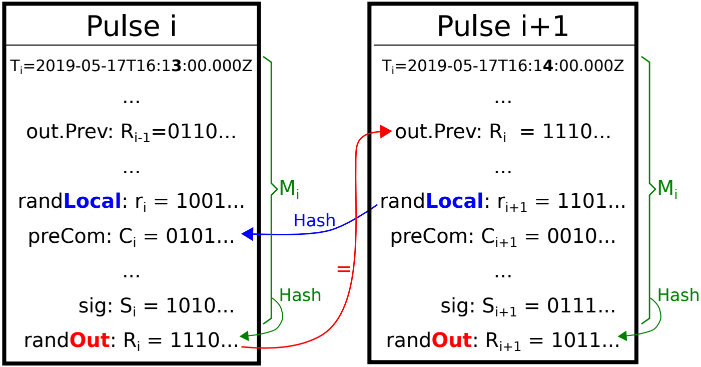
`randOut`: $R_i = \text{Hash}(M_i)$, with M_i being the serialization of all previous fields

`out.Prev` has the output value (R_i) of the previous pulse

The two “rands” in a pulse

`randLocal`: $r_{i+1} = \text{Hash}(\rho_{1,i} \mid\mid \rho_{2,i} \; [\mid\mid \rho_{3,i}])$, with random $\rho_{j,i}$ from i^{th} RNG

`preCom`: $C_i = \text{Hash}(r_{i+1})$, released 1 min before r_{i+1}



Pulse i

T_i=2019-05-17T16:1**3**:00.000Z

...

out.Prev: R_{i-1}=0110...

...

rand**Local**: r_i = 1001...

preCom: C_i = 0101...

...

sig: S_i = 1010...

rand**Out**: R_i = 1110...

Pulse i+1

T_i=2019-05-17T16:1**4**:00.000Z

...

out.Prev: R_i = 1110...

...

rand**Local**: r_{i+1} = 1101...

preCom: C_{i+1} = 0010...

...

sig: S_{i+1} = 0111...

rand**Out**: R_{i+1} = 1011...

`randOut`: $R_i = \text{Hash}(M_i)$, with M_i being the serialization of all previous fields

`out.Prev` has the output value (R_i) of the previous pulse

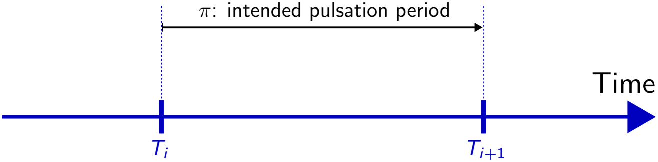
Outline

1. Introduction
2. Pulse format
3. Beacon Protocol
4. Using a Beacon
5. Brief security considerations
6. Conclusion

Beacon proper operation

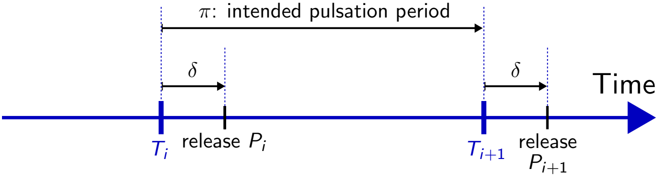
- Timing and entropy requirements
- Beacon interface: getting pulses and skiplists
- Others (not here): external values, status fields, ...

Timing requirements for generation and release



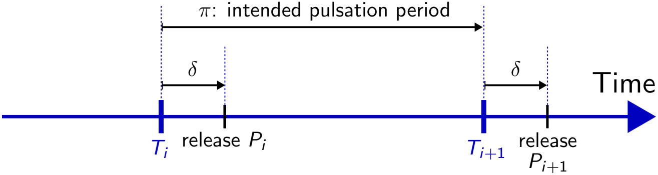
Timing requirements for generation and release

1. No advanced release of pulse ($\delta \geq 0$)



Timing requirements for generation and release

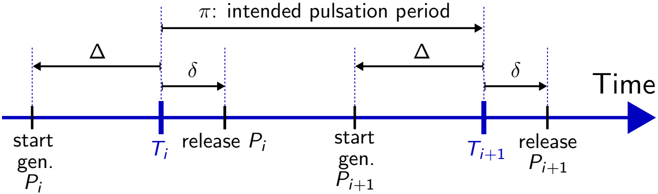
$$\left.\begin{array}{l}\text{1. No advanced release of pulse } (\delta \geq 0) \\ \text{2. Generate with entropy } (\geq 2 \text{ RNGs})\end{array}\right\} \Rightarrow \textbf{Unpredictability}$$



Timing requirements for generation and release

$$\left.\begin{array}{l}\text{1. No advanced release of pulse } (\delta \geq 0) \\ \text{2. Generate with entropy } (\geq 2 \text{ RNGs})\end{array}\right\} \Rightarrow \textbf{Unpredictability}$$

3. Generate not too in advance (small Δ)

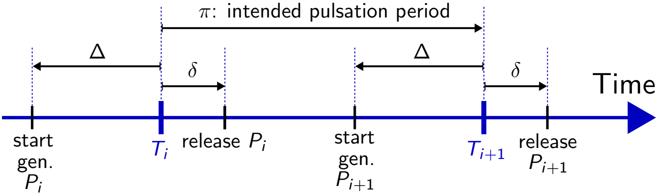


Timing requirements for generation and release

1. No advanced release of pulse ($\delta \geq 0$)
2. Generate with entropy (≥ 2 RNGs)

$\Big\} \Rightarrow$ **Unpredictability**

3. Generate not too in advance (small Δ) $\Rightarrow$ **Freshness**

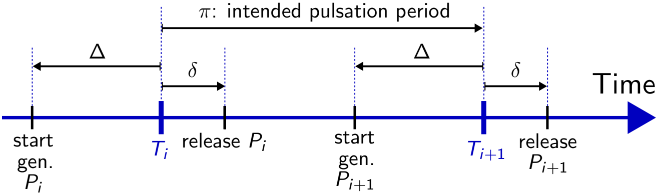


Timing requirements for generation and release

1. No advanced release of pulse ($\delta \geq 0$)
2. Generate with entropy (≥ 2 RNGs)

$\Big\}$ $\Rightarrow$ **Unpredictability**

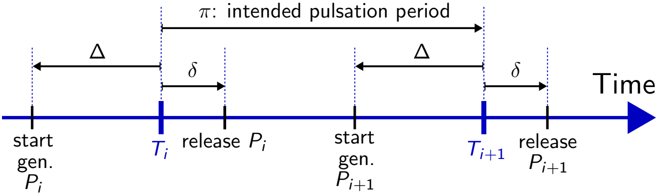
3. Generate not too in advance (small Δ) $\Rightarrow$ **Freshness**
4. Release soon (small δ)



Timing requirements for generation and release

$$\left.\begin{array}{l}\text{1. No advanced release of pulse } (\delta \geq 0) \\ \text{2. Generate with entropy } (\geq 2 \text{ RNGs})\end{array}\right\} \Rightarrow \textbf{Unpredictability}$$

3. Generate not too in advance (small Δ) ⇒ **Freshness**
4. Release soon (small δ) ⇒ **Timeliness**

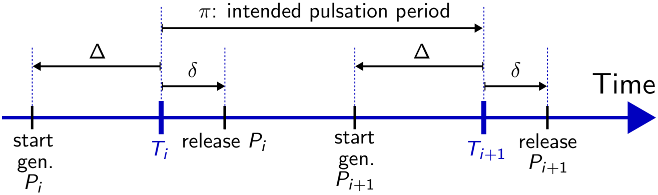


Timing requirements for generation and release

1. No advanced release of pulse ($\delta \geq 0$)
2. Generate with entropy ($\geq$ 2 RNGs)

$\Rightarrow$ **Unpredictability**

3. Generate not too in advance (small Δ) $\Rightarrow$ **Freshness**
4. Release soon (small δ) $\Rightarrow$ **Timeliness**
5. Timestamp (non-repeating) indexation

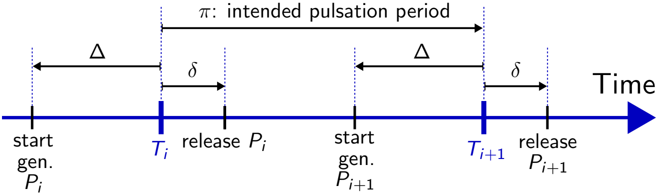


Timing requirements for generation and release

1. No advanced release of pulse ($\delta \geq 0$)
2. Generate with entropy ($\geq$ 2 RNGs)

$\Big\}$ $\Rightarrow$ **Unpredictability**

3. Generate not too in advance (small Δ) $\Rightarrow$ **Freshness**
4. Release soon (small δ) $\Rightarrow$ **Timeliness**
5. Timestamp (non-repeating) indexation $\Rightarrow$ **Unambiguity**

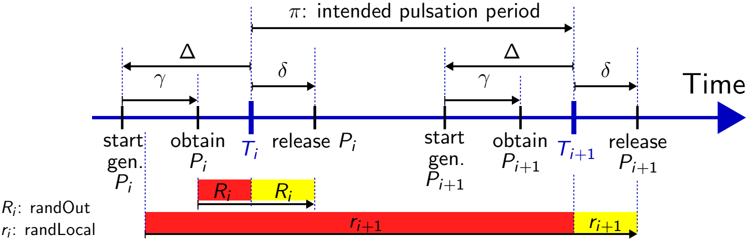


Timing requirements for generation and release

1. No advanced release of pulse ($\delta \geq 0$)
2. Generate with entropy ($\geq$ 2 RNGs)

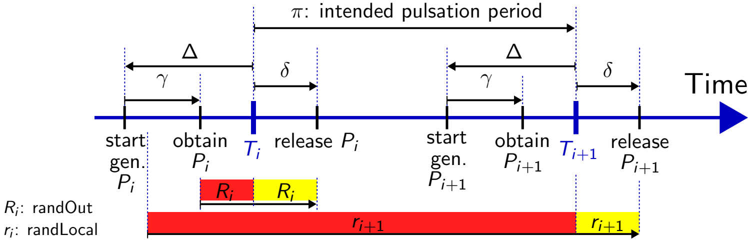
$\Rightarrow$ **Unpredictability**

3. Generate not too in advance (small Δ) $\Rightarrow$ **Freshness**
4. Release soon (small δ) $\Rightarrow$ **Timeliness**
5. Timestamp (non-repeating) indexation $\Rightarrow$ **Unambiguity**



Timing requirements for generation and release

1. No advanced release of pulse ($\delta \geq 0$) } ⇒ **Unpredictability**
2. Generate with entropy ($\geq$ 2 RNGs)
3. Generate not too in advance (small Δ) ⇒ **Freshness**
4. Release soon (small δ) ⇒ **Timeliness**
5. Timestamp (non-repeating) indexation ⇒ **Unambiguity**

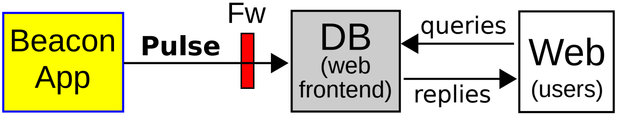


(The reference document specifies allowed intervals for δ and Δ, relative to π)

Fetching pulses

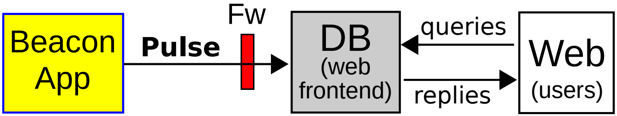
Fetching pulses

Beacon App: a *pulse release* means *sending the pulse to the database*



Fetching pulses

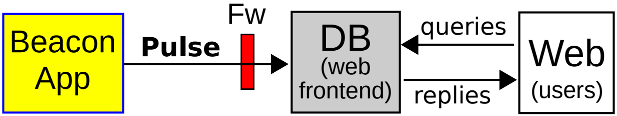
Beacon App: a *pulse release* means *sending the pulse to the database*



How do users request pulses from the database?

Fetching pulses

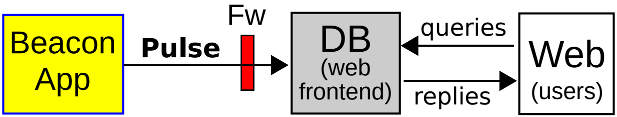
Beacon App: a *pulse release* means *sending the pulse to the database*



How do users request pulses from the database? uri/url

Fetching pulses

Beacon App: a *pulse release* means *sending the pulse to the database*



How do users request pulses from the database? uri/url

https://beacon.nist.gov/beacon/2.0/chain/last/pulse/last
Example: URI for the latest pulse in chain 1 of the NIST randomness Beacon (version 2)

Skiplists — efficient chain verification

Skiplists — efficient chain verification

How to prove that an **old** pulse is consistent with a **recent** pulse?

Skiplists — efficient chain verification

How to prove that an **old** pulse is consistent with a **recent** pulse?

Example: Anchor = **2019-05-17 14:12** → Target = **2016-02-14 17:45**

Skiplists — efficient chain verification

How to prove that an **old** pulse is consistent with a **recent** pulse?

Example: $\textsc{Anchor} = \boxed{\textbf{2019-05-17 14:12}} \rightarrow \textsc{Target} = \boxed{\textbf{2016-02-14 17:45}}$

Solution: check that there is a hash-chain linking them

Skiplists — efficient chain verification

How to prove that an **old** pulse is consistent with a **recent** pulse?

Example: ANCHOR = $\boxed{\textbf{2019-05-17 14:12}}$ → TARGET = $\boxed{\textbf{2016-02-14 17:45}}$

Solution: check that there is a hash-chain linking them

Inefficient: check the hash-chain of (>1M) consecutive pulses

Efficient: check the hash-chaining in a **skiplist** ($<$ 125 pulses).

Skiplists — efficient chain verification

How to prove that an **old** pulse is consistent with a **recent** pulse?

Example: Anchor = **2019-05-17 14:12** → Target = **2016-02-14 17:45**

Solution: check that there is a hash-chain linking them

Inefficient: check the hash-chain of (>1M) consecutive pulses

2019-05-17 14:12 → 2019-05-17 14:11 → (1 per minute) → **2016-02-14 17:45**

Efficient: check the hash-chaining in a **skiplist** (< 125 pulses).

Skiplists — efficient chain verification

How to prove that an **old** pulse is consistent with a **recent** pulse?

Example: $\textsc{Anchor} = \boxed{\textbf{2019-05-17 14:12}} \rightarrow \textsc{Target} = \boxed{\textbf{2016-02-14 17:45}}$

Solution: check that there is a hash-chain linking them

Inefficient: check the hash-chain of (>1M) consecutive pulses

$\boxed{\textbf{2019-05-17 14:12}} \rightarrow \boxed{\text{2019-05-17 14:11}} \rightarrow$ (1 per minute) $\rightarrow \boxed{\textbf{2016-02-14 17:45}}$

Efficient: check the hash-chaining in a **skiplist** ($<$ 125 pulses).

Use the 5 *past output* fields in the pulse format:

- `out.Prev`: the *previous* pulse
- `out.H`, `out.D`, `out.M`, `out.Y`: the first of the **hour**/**day**/**month**/**year**

Skiplists — efficient chain verification

How to prove that an **old** pulse is consistent with a **recent** pulse?

Example: ANCHOR = **2019-05-17 14:12** → TARGET = **2016-02-14 17:45**

Solution: check that there is a hash-chain linking them

Inefficient: check the hash-chain of (>1M) consecutive pulses

2019-05-17 14:12 → 2019-05-17 14:11 → (1 per minute) → **2016-02-14 17:45**

Efficient: check the hash-chaining in a **skiplist** (< 125 pulses).

Use the 5 *past output* fields in the pulse format:

- `out.Prev`: the *previous* pulse
- `out.H`, `out.D`, `out.M`, `out.Y`: the first of the **hour**/**day**/**month**/**year**

2019-05-17 14:12 → **2019-01**-01 00:00 → **2018-01**-01 00:00 → **2017-01**-01 00:00 →

Skiplists — efficient chain verification

How to prove that an **old** pulse is consistent with a **recent** pulse?

Example: ANCHOR = **2019-05-17 14:12** → TARGET = **2016-02-14 17:45**

Solution: check that there is a hash-chain linking them

Inefficient: check the hash-chain of (>1M) consecutive pulses

2019-05-17 14:12 → 2019-05-17 14:11 → (1 per minute) → **2016-02-14 17:45**

Efficient: check the hash-chaining in a **skiplist** ($<$ 125 pulses).

Use the 5 *past output* fields in the pulse format:

- `out.Prev`: the *previous* pulse
- `out.H`, `out.D`, `out.M`, `out.Y`: the first of the **hour**/**day**/**month**/**year**

2019-05-17 14:12 → **2019-01**-01 00:00 → **2018-01**-01 00:00 → **2017-01**-01 00:00 →
2016-**12-01** 00:00 → (1 per month) → 2016-**03-01** 00:00

Skiplists — efficient chain verification

How to prove that an **old** pulse is consistent with a **recent** pulse?

Example: ANCHOR = **2019-05-17 14:12** → TARGET = **2016-02-14 17:45**

Solution: check that there is a hash-chain linking them

Inefficient: check the hash-chain of (>1M) consecutive pulses

2019-05-17 14:12 → 2019-05-17 14:11 → (1 per minute) → **2016-02-14 17:45**

Efficient: check the hash-chaining in a **skiplist** (< 125 pulses).

Use the 5 *past output* fields in the pulse format:

- out.Prev: the *previous* pulse
- out.H, out.D, out.M, out.Y: the first of the **hour**/**day**/**month**/**year**

2019-05-17 14:12 → **2019-01**-01 00:00 → **2018-01**-01 00:00 → **2017-01**-01 00:00 →
2016-**12-01** 00:00 → (1 per month) → 2016-**03-01** 00:00 → 2016-02-**29 00**:00 →
(1 per day) → 2016-02-**15 00**:00

Skiplists — efficient chain verification

How to prove that an **old** pulse is consistent with a **recent** pulse?

Example: Anchor = **2019-05-17 14:12** → Target = **2016-02-14 17:45**

Solution: check that there is a hash-chain linking them

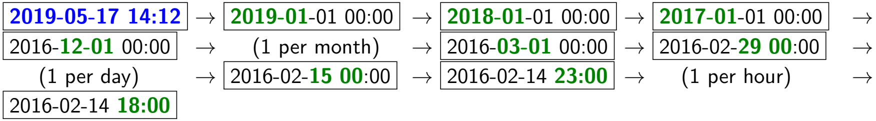
Inefficient: check the hash-chain of (>1M) consecutive pulses

2019-05-17 14:12 → 2019-05-17 14:11 → (1 per minute) → **2016-02-14 17:45**

Efficient: check the hash-chaining in a **skiplist** (< 125 pulses).

Use the 5 *past output* fields in the pulse format:

- `out.Prev`: the *previous* pulse
- `out.H`, `out.D`, `out.M`, `out.Y`: the first of the **hour**/**day**/**month**/**year**



2019-05-17 14:12 → **2019-01**-01 00:00 → **2018-01**-01 00:00 → **2017-01**-01 00:00 →

2016-**12-01** 00:00 → (1 per month) → 2016-**03-01** 00:00 → 2016-02-**29 00**:00 →

(1 per day) → 2016-02-**15 00**:00 → 2016-02-14 **23:00** → (1 per hour) →

2016-02-14 **18:00**

Skiplists — efficient chain verification

How to prove that an **old** pulse is consistent with a **recent** pulse?

Example: ANCHOR = **2019-05-17 14:12** → TARGET = **2016-02-14 17:45**

Solution: check that there is a hash-chain linking them

Inefficient: check the hash-chain of ($>$1M) consecutive pulses

2019-05-17 14:12 → 2019-05-17 14:11 → (1 per minute) → **2016-02-14 17:45**

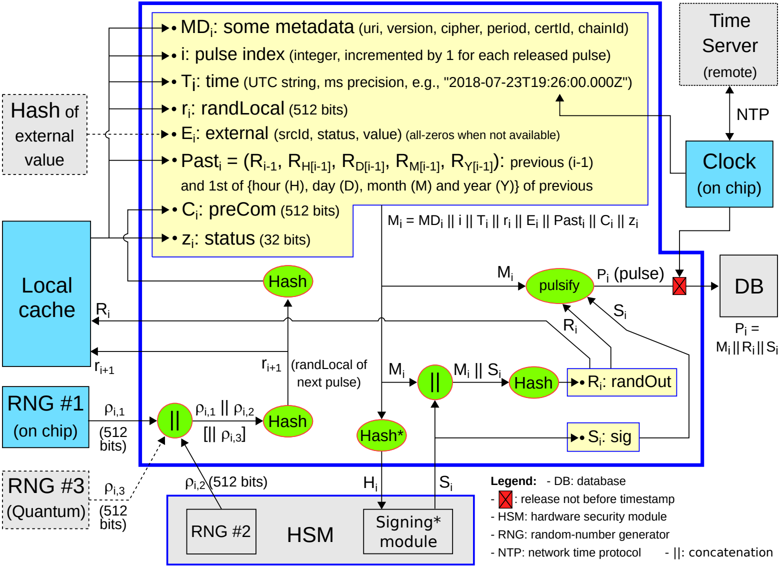
Efficient: check the hash-chaining in a **skiplist** ($<$ 125 pulses).

Use the 5 *past output* fields in the pulse format:

- `out.Prev`: the *previous* pulse
- `out.H`, `out.D`, `out.M`, `out.Y`: the first of the **hour**/**day**/**month**/**year**

2019-05-17 14:12 → **2019-01**-01 00:00 → **2018-01**-01 00:00 → **2017-01**-01 00:00 →
2016-**12-01** 00:00 → (1 per month) → 2016-**03-01** 00:00 → 2016-02-**29 00**:00 →
(1 per day) → 2016-02-**15 00**:00 → 2016-02-14 **23:00** → (1 per hour) →
2016-02-14 **18:00** → 2016-02-14 17:**59** → (1 per minute) → **2016-02-14 17:45**

A possible diagram of pulse generation



- MD_i: some metadata (uri, version, cipher, period, certId, chainId)
- i: pulse index (integer, incremented by 1 for each released pulse)
- T_i: time (UTC string, ms precision, e.g., "2018-07-23T19:26:00.000Z")
- r_i: randLocal (512 bits)
- E_i: external (srcId, status, value) (all-zeros when not available)
- $Past_i = (R_{i-1}, R_{H[i-1]}, R_{D[i-1]}, R_{M[i-1]}, R_{Y[i-1]})$: previous (i-1) and 1st of {hour (H), day (D), month (M) and year (Y)} of previous
- C_i: preCom (512 bits)
- z_i: status (32 bits)

$M_i = MD_i \,||\, i \,||\, T_i \,||\, r_i \,||\, E_i \,||\, Past_i \,||\, C_i \,||\, z_i$

$P_i = M_i \,||\, R_i \,||\, S_i$

Legend:
- DB: database
- ☒: release not before timestamp
- HSM: hardware security module
- RNG: random-number generator
- NTP: network time protocol
- ||: concatenation

For simplicity, the diagram omits serialization details (e.g., field lengths and padding) and some metadata fields.

Outline

1. Introduction

2. Pulse format

3. Beacon Protocol

4. Using a Beacon

5. Brief security considerations

6. Conclusion

Using Beacon randomness (if I trust the beacon)

(some simplifications for presentation purpose)

Using Beacon randomness (if I trust the beacon)

(some simplifications for presentation purpose)

Simply getting a practically uniform number in $[0, N-1]$:

Using Beacon randomness (if I trust the beacon)

(some simplifications for presentation purpose)

Simply getting a practically uniform number in $[0, N-1]$:

- Just calculate `randOut` $\pmod{N}$, if $N < 2^{384}$

Using Beacon randomness (if I trust the beacon)

(some simplifications for presentation purpose)

Simply getting a practically uniform number in $[0, N-1]$:

- Just calculate `randOut` $\pmod{N}$, if $N < 2^{384}$

If I want future auditability of a randomized operation:

Using Beacon randomness (if I trust the beacon)

(some simplifications for presentation purpose)

Simply getting a practically uniform number in $[0, N-1]$:

- Just calculate `randOut` $\pmod{N}$, if $N < 2^{384}$

If I want future auditability of a randomized operation:

1. **Commit upfront:**

2. **Derive a seed:**

3. **Perform the operation:**

Using Beacon randomness (if I trust the beacon)

(some simplifications for presentation purpose)

Simply getting a practically uniform number in $[0, N-1]$:

- Just calculate `randOut` $\pmod{N}$, if $N < 2^{384}$

If I want future auditability of a randomized operation:

1. **Commit upfront:** publish a statement S that explains my deterministic operation that will use the Beacon randomness (the output value `randOut`) from future time t;
2. **Derive a seed:**
3. **Perform the operation:**

Using Beacon randomness (if I trust the beacon)

(some simplifications for presentation purpose)

Simply getting a practically uniform number in $[0, N-1]$:

- Just calculate `randOut` (mod N), if $N < 2^{384}$

If I want future auditability of a randomized operation:

1. **Commit upfront:** publish a statement S that explains my deterministic operation that will use the Beacon randomness (the output value `randOut`) from future time t;
2. **Derive a seed:** Get $R =$ `randOut`$[t]$ (from the pulse with `timestamp` t), and set the seed as $Z = \text{SHA512}(S||R)$
3. **Perform the operation:**

Using Beacon randomness (if I trust the beacon)

(some simplifications for presentation purpose)

Simply getting a practically uniform number in $[0, N-1]$:

- Just calculate `randOut` $\pmod{N}$, if $N < 2^{384}$

If I want future auditability of a randomized operation:

1. **Commit upfront:** publish a statement S that explains my deterministic operation that will use the Beacon randomness (the output value `randOut`) from future time t;
2. **Derive a seed:** Get $R =$ `randOut`$[t]$ (from the pulse with `timestamp` t), and set the seed as $Z = \text{SHA512}(S||R)$
3. **Perform the operation:** Do what the statement S promised, using Z as the seed for all needed pseudo-randomness.

Using Beacon randomness (if I trust the beacon)

(some simplifications for presentation purpose)

Simply getting a practically uniform number in $[0, N-1]$:

- Just calculate `randOut` (mod N), if $N < 2^{384}$

If I want future auditability of a randomized operation:

1. **Commit upfront:** publish a statement S that explains my deterministic operation that will use the Beacon randomness (the output value `randOut`) from future time t;
2. **Derive a seed:** Get $R = \texttt{randOut}[t]$ (from the pulse with `timestamp` t), and set the seed as $Z = \text{SHA512}(S||R)$
3. **Perform the operation:** Do what the statement S promised, using Z as the seed for all needed pseudo-randomness.

We defer reference guidance to complementary future documentation

Combining Beacons

Combining Beacons

What Beacon randomness $R[t]$ to use if I do not trust any Beacon in isolation?

Combining Beacons

What Beacon randomness $R[t]$ to use if I do not trust any Beacon in isolation? ... but trust that two Beacons (A and B) will **not** collude?

Combining Beacons

What Beacon randomness $R[t]$ to use if I do not trust any Beacon in isolation? ... but trust that two Beacons (A and B) will **not** collude?

Desired properties:

- A single Beacon **cannot** bias the output;
- Even two colluding beacons **cannot** fully control the output.

Combining Beacons

What Beacon randomness $R[t]$ to use if I do not trust any Beacon in isolation? ... but trust that two Beacons (A and B) will **not** collude?

Desired properties:

- A single Beacon **cannot** bias the output;
- Even two colluding beacons **cannot** fully control the output.

Not good:

- $R[t] = \text{Hash}(A[t].\texttt{randOut}||B[t].\texttt{randOut})$

Combining Beacons

What Beacon randomness $R[t]$ to use if I do not trust any Beacon in isolation? ... but trust that two Beacons (A and B) will **not** collude?

Desired properties:

- A single Beacon **cannot** bias the output;
- Even two colluding beacons **cannot** fully control the output.

Not good:

- $R[t] = \mathrm{Hash}(A[t].\texttt{randOut}||B[t].\texttt{randOut})$ (A could wait to know B's value)

Combining Beacons

What Beacon randomness $R[t]$ to use if I do not trust any Beacon in isolation? ... but trust that two Beacons (A and B) will **not** collude?

Desired properties:

- A single Beacon **cannot** bias the output;
- Even two colluding beacons **cannot** fully control the output.

Not good:

- $R[t] = \text{Hash}(A[t].\texttt{randOut} || B[t].\texttt{randOut})$ (A could wait to know B's value)
- $R[t] = \text{Hash}(A[t].\texttt{randLocal} || B[t].\texttt{randLocal})$

Combining Beacons

What Beacon randomness $R[t]$ to use if I do not trust any Beacon in isolation? ... but trust that two Beacons (A and B) will **not** collude?

Desired properties:

- A single Beacon **cannot** bias the output;
- Even two colluding beacons **cannot** fully control the output.

Not good:

- $R[t] = \text{Hash}(A[t].\texttt{randOut}||B[t].\texttt{randOut})$ (A could wait to know B's value)
- $R[t] = \text{Hash}(A[t].\texttt{randLocal}||B[t].\texttt{randLocal})$
  (A & B could force repeating $R[t]$)

Combining Beacons

What Beacon randomness $R[t]$ to use if I do not trust any Beacon in isolation? ... but trust that two Beacons (A and B) will **not** collude?

Desired properties:

- A single Beacon **cannot** bias the output;
- Even two colluding beacons **cannot** fully control the output.

Not good:

- $R[t] = \text{Hash}(A[t].\texttt{randOut}||B[t].\texttt{randOut})$ (A could wait to know B's value)
- $R[t] = \text{Hash}(A[t].\texttt{randLocal}||B[t].\texttt{randLocal})$
  (A & B could force repeating $R[t]$)

Solution: get $R[t]$ from two $\texttt{randLocal}[t]$ and two $\texttt{randOut}[t-\pi]$ values:

Combining Beacons

What Beacon randomness $R[t]$ to use if I do not trust any Beacon in isolation? ... but trust that two Beacons (A and B) will **not** collude?

Desired properties:

- A single Beacon **cannot** bias the output;
- Even two colluding beacons **cannot** fully control the output.

Not good:

- $R[t] = \text{Hash}(A[t].\texttt{randOut}||B[t].\texttt{randOut})$ (A could wait to know B's value)
- $R[t] = \text{Hash}(A[t].\texttt{randLocal}||B[t].\texttt{randLocal})$ (A & B could force repeating $R[t]$)

Solution: get $R[t]$ from two $\texttt{randLocal}[t]$ and two $\texttt{randOut}[t-\pi]$ values:

$$R[t] = \text{Hash}(A[t-\pi].\texttt{randOut}||B[t-\pi].\texttt{randOut}||A[t].\texttt{randLocal}||B[t].\texttt{randLocal})$$

Combining Beacons

What Beacon randomness $R[t]$ to use if I do not trust any Beacon in isolation? ... but trust that two Beacons (A and B) will **not** collude?

Desired properties:

- A single Beacon **cannot** bias the output;
- Even two colluding beacons **cannot** fully control the output.

Not good:

- $R[t] = \text{Hash}(A[t].\texttt{randOut}||B[t].\texttt{randOut})$ (A could wait to know B's value)
- $R[t] = \text{Hash}(A[t].\texttt{randLocal}||B[t].\texttt{randLocal})$ (A & B could force repeating $R[t]$)

Solution: get $R[t]$ from two $\texttt{randLocal}[t]$ and two $\texttt{randOut}[t-\pi]$ values:

$$R[t] = \text{Hash}(A[t-\pi].\texttt{randOut}||B[t-\pi].\texttt{randOut}||A[t].\texttt{randLocal}||B[t].\texttt{randLocal})$$

Also need to check:

- reception of $A[t-\pi].\mathit{randOut}$ and $B[t-\pi].\mathit{randOut}$ before time T
- correctness of standalone pulses: $A[t-\pi], B[t-\pi], A[t], B[t]$
- hash-chaining (e.g., $A[t].\texttt{out.Prev} = A[t-\pi].\texttt{randOut}$)
- pre-commitments (e.g., $\text{Hash}(\text{A}[\text{t}].\texttt{randLocal}) = A[t-\pi].\texttt{preCom}$)

Some Beacons in development

Three countries are developing Beacons to match the current reference:

United States
Brazil
Chile

- (United States) NIST Randomness Beacon
  https://beacon.nist.gov/home
- (Chile) CLCERT Randomness Beacon
  https://beacon.clcert.cl/
- (Brazil) Brazilian Randomness Beacon
  https://beacon.inmetro.gov.br/

Some Beacons in development

Three countries are developing Beacons to match the current reference:

- (United States) NIST Randomness Beacon
  https://beacon.nist.gov/home
- (Chile) CLCERT Randomness Beacon
  https://beacon.clcert.cl/
- (Brazil) Brazilian Randomness Beacon
  https://beacon.inmetro.gov.br/

We would like others to join

Some conceivable applications

"*You have been randomly selected for additional screening*"

Some conceivable applications

"*You have been randomly selected for additional screening*"

Example applications:

- Select test and control groups for clinical trials
- Select random government officials for financial audits
- Assign court cases to random judges
- Sample random lots for quality-measuring procedures
- Provide entropy to digital lotteries

Some conceivable applications

"*You have been randomly selected for additional screening*"

Example applications:

- Select test and control groups for clinical trials
- Select random government officials for financial audits
- Assign court cases to random judges
- Sample random lots for quality-measuring procedures
- Provide entropy to digital lotteries

Some generic goals:

- Prevent auditors from biasing selections (or being accused of it)
- Prevent auditees from addressing only the items to-be sampled
- Enable public verifiability of correct sampling

Outline

1. Introduction

2. Pulse format

3. Beacon Protocol

4. Using a Beacon

5. Brief security considerations

6. Conclusion

Security against intrusions

Security is “easy” in uncompromised scenario!

Security against intrusions

Security is "easy" in uncomprised scenario!

But how to withstand compromised system components?

- **Semi-honest (SH)**, aka honest-but-curious or passive: can exfiltrate internal state, but does not deviate from protocol
- **Malicious (Mal)**, aka byzantine or active: arbitrary behavior

Security against intrusions

Security is "easy" in uncompromised scenario!

But how to withstand compromised system components?

- **Semi-honest (SH)**, aka honest-but-curious or passive: can exfiltrate internal state, but does not deviate from protocol
- **Malicious (Mal)**, aka byzantine or active: arbitrary behavior

Why considering intrusions?

1. We want trust to be leveled with trustworthiness — a security analysis enables reflecting on meaningful security claims.

Security against intrusions

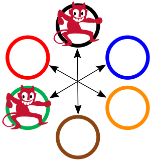
Security is "easy" in uncompromised scenario!

But how to withstand compromised system components?

- **Semi-honest (SH)**, aka honest-but-curious or passive: can exfiltrate internal state, but does not deviate from protocol
- **Malicious (Mal)**, aka byzantine or active: arbitrary behavior

Why considering intrusions?

1. We want trust to be leveled with trustworthiness — a security analysis enables reflecting on meaningful security claims.
2. *Even if operators believe in uncompromised components at launch day, we want security in the long run, against conceivable adversarial threats (goals and capabilities).*



Types of security properties (informal)

- **Relational**: correct hash chain, signatures, timestamps, consistent record (immutable past), ...
- **Availability:** *timely* pulse releases; *accessible* past pulses; *automatic* operation (reduced human operator intervention); ...
- **“Rands” quality**: unpredictable; unbiaseable; fresh and independent;

Types of security properties (informal)

- **Relational**: correct hash chain, signatures, timestamps, consistent record (immutable past), ...
- **Availability:** *timely* pulse releases; *accessible* past pulses; *automatic* operation (reduced human operator intervention); ...
- **"Rands" quality**: unpredictable; unbiaseable; fresh and independent;

Attack consequences:

- breaking *relational* or *availability* properties typically leads to detectable errors, e.g., incorrect signatures or hash-chaining, delayed releases, ...
- next slides mention a few examples of attacks to the *"rands" quality*

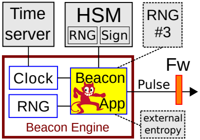
Intrusion scenarios

NISTIR 8213 considers several scenarios with intruded components

Intrusion scenarios

NISTIR 8213 considers several scenarios with intruded components:

- I1. Mal Beacon App → randLocal control attack



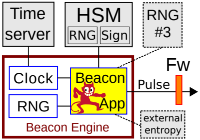
Legend: Mal=malicious

The red dancing devil clipart is from clker.com/clipart-13643.html

Intrusion scenarios

NISTIR 8213 considers several scenarios with intruded components:

- I1. Mal Beacon App → randLocal control attack
- I2. Mal Beacon App → randOut bias attack



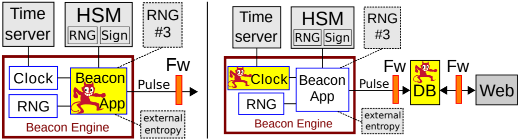
Legend: Mal=malicious

The red dancing devil clipart is from clker.com/clipart-13643.html

Intrusion scenarios

NISTIR 8213 considers several scenarios with intruded components:

- I1. Mal Beacon App → randLocal control attack
- I2. Mal Beacon App → randOut bias attack
- I3. Mal local-clock + SH DB → “rands” predict attack



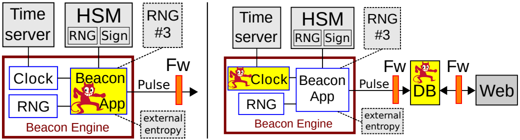
Legend: Mal=malicious; SH=semi-honest; DB=database

The red dancing devil clipart is from clker.com/clipart-13643.html

Intrusion scenarios

NISTIR 8213 considers several scenarios with intruded components:

- I1. Mal Beacon App → randLocal control attack
- I2. Mal Beacon App → randOut bias attack
- I3. Mal local-clock + SH DB → "rands" predict attack
- I4. SH Beacon App → "rands" prediction attack



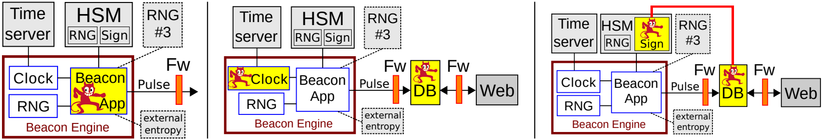
Legend: Mal=malicious; SH=semi-honest; DB=database

The red dancing devil clipart is from clker.com/clipart-13643.html

Intrusion scenarios

NISTIR 8213 considers several scenarios with intruded components:

- I1. Mal Beacon App → randLocal control attack
- I2. Mal Beacon App → randOut bias attack
- I3. Mal local-clock + SH DB → “rands” predict attack
- I4. SH Beacon App → “rands” prediction attack
- I5. Mal DB with HSM sign key → change-history attack



Legend: Mal=malicious; SH=semi-honest; DB=database

The red dancing devil clipart is from clker.com/clipart-13643.html

Conceivable mitigations

The NISTIR mentions some mitigations

(either possible now or conceivable for the future)

Conceivable mitigations

The NISTIR mentions some mitigations

(either possible now or conceivable for the future)

For example, some could be based on the use of:

- publicly-verifiable external entropy (to reduce pre-computation window)
- verifiable delay functions
- secure time synchronization
- a different `randLocal` computation, with non controllable value
- different signature (e.g., $>$ bit-strength, post-quantum, or/and threshold)
- a forward-chaining mechanism

Outline

1. Introduction

2. Pulse format

3. Beacon Protocol

4. Using a Beacon

5. Brief security considerations

6. Conclusion

Final Remarks

Final Remarks

- Randomness Beacons have a **great potential to serve as a public utility**, e.g., to promote public auditability of randomized processes

Final Remarks

- Randomness Beacons have a **great potential to serve as a public utility**, e.g., to promote public auditability of randomized processes

- The *reference* (NISTIR 8213) version 2 introduces new features for better **interoperability, security and efficiency**

Final Remarks

- Randomness Beacons have a **great potential to serve as a public utility**, e.g., to promote public auditability of randomized processes
- The *reference* (NISTIR 8213) version 2 introduces new features for better **interoperability, security and efficiency**
- **Possible developments to be made:**
  - Complementary analysis and guidance
  - Improvements based on feedback

Final Remarks

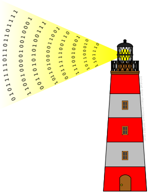
- Randomness Beacons have a **great potential to serve as a public utility**, e.g., to promote public auditability of randomized processes

- The *reference* (NISTIR 8213) version 2 introduces new features for better **interoperability, security and efficiency**

- **Possible developments to be made:**
  - Complementary analysis and guidance
  - Improvements based on feedback

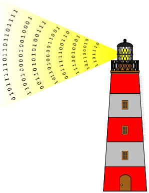
- **We would like to have your collaboration:**
  - public feedback on the NISTIR 8213
  - more deployed beacons
  - external apps using Beacon randomness

- Draft NISTIR 8213: https://doi.org/10.6028/NIST.IR.8213-draft
- Email for feedback on the NISTIR 8213: beacon-nistir@nist.gov
- Beacon project: https://www.nist.gov/programs-projects/nist-randomness-beacon



Thank you for your attention

- Draft NISTIR 8213: https://doi.org/10.6028/NIST.IR.8213-draft
- Email for feedback on the NISTIR 8213: beacon-nistir@nist.gov
- Beacon project: https://www.nist.gov/programs-projects/nist-randomness-beacon



Presentation at the International Cryptographic Module Conference

May 17, 2019 @ Vancouver, Canada

luis.brandao@nist.gov

Disclaimer. Opinions expressed in this presentation are from the author(s) and are not to be construed as official or as views of the U.S. Department of Commerce. The identification of any commercial product or trade names in this presentation does not imply endorsement of recommendation by NIST, nor is it intended to imply that the material or equipment identified are necessarily the best available for the purpose.

Disclaimer. Some external-source images and cliparts were included/adapted in this presentation with the expectation of such use constituting licensed and/or fair use.