Are you Feeling Lucky?

Casino Gaming, Java, and IoT

Matt Schuetze
Azul Director of Product Management

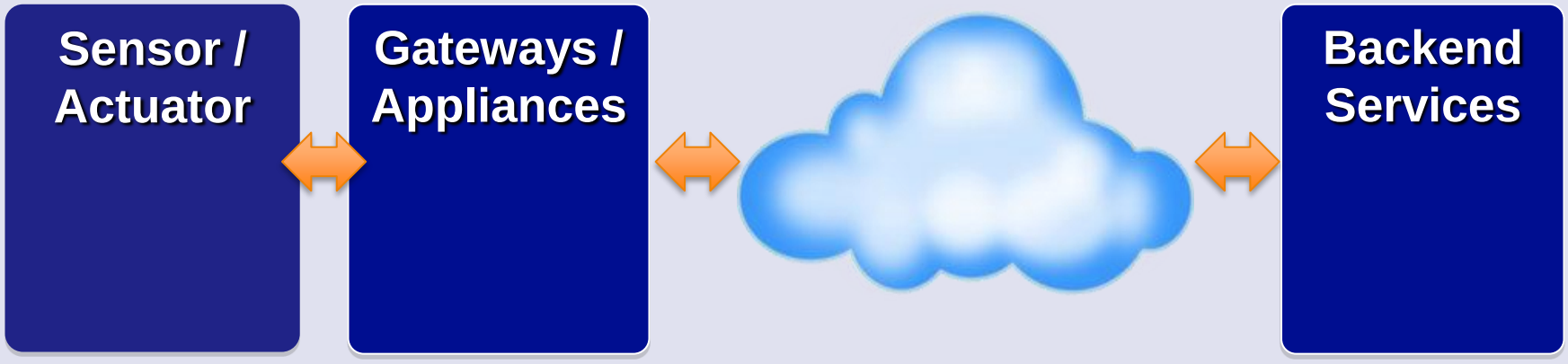
SECR 2017
St. Petersburg, Russia

10/26/2017

About me: Matt Schuetze

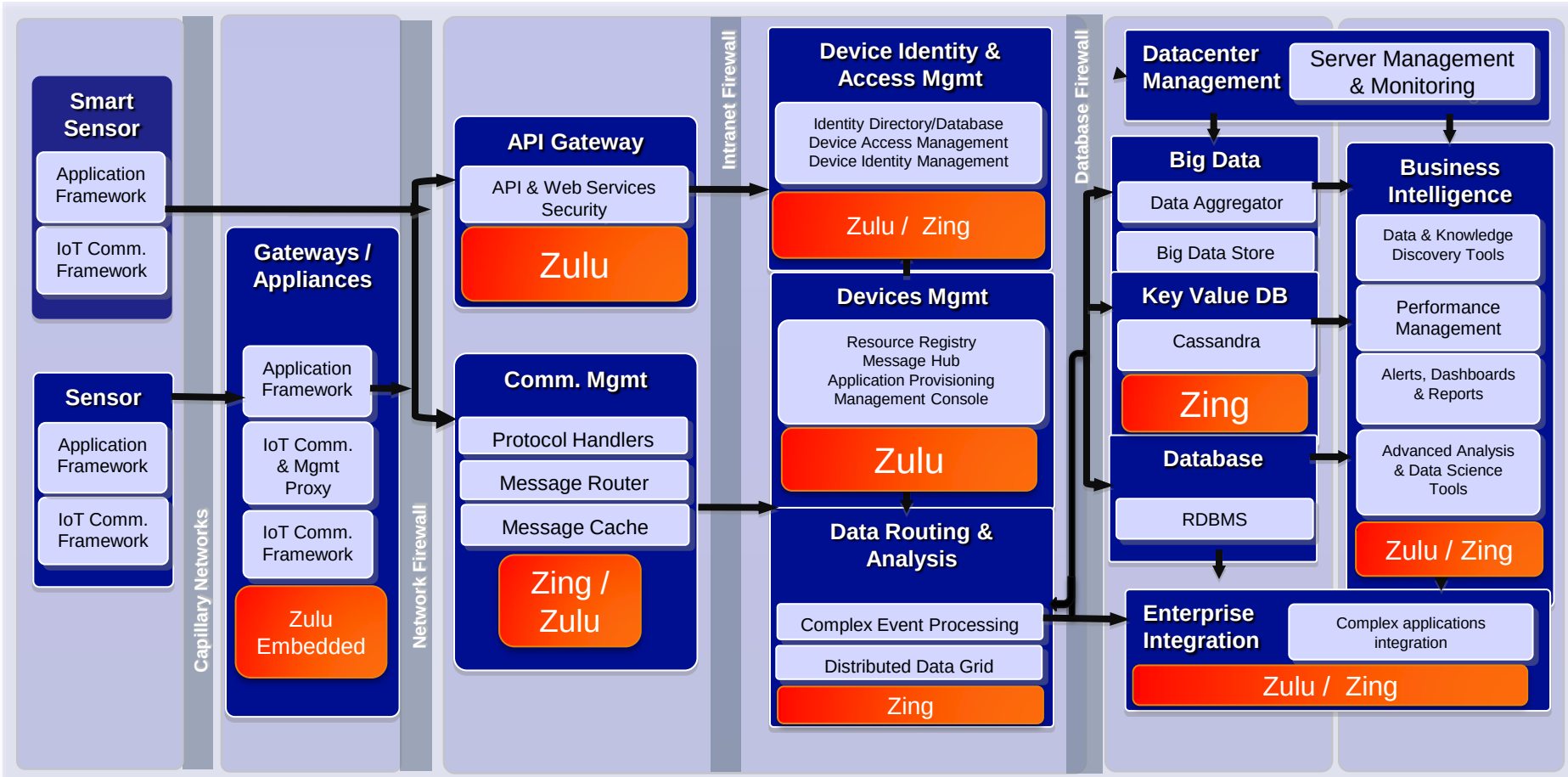
- Product Management Director at Azul Systems
- Wrestle Zing and Zulu requirements
- Push Azul product launches
- Azul alternate on JCP exec committee
- Lead organizer of Detroit JUG
- Heroic Friend of Duke
- Stand on Shoulders of Giants

Common Picture of IoT



Front | Back

Azul Work in IoT (all Java)



Are You Feeling Lucky?

- Casinos and Gaming
- Luck vs Chance vs Fairness
- Random Numbers in Java
- Entropy: The Second Law
- Role of Hardware
- Real World Impact on IoT

Casino Games!

Excellent example of fun software projects

Games in Code

- Blackjack
- Craps
- Roulette
- Slots
- Poker

Probabilities

- 1 in 52
- 1 in 38
- 1 in 6
- 1 in 6 x 1 in 6
- Ever present house advantage

Random Selection

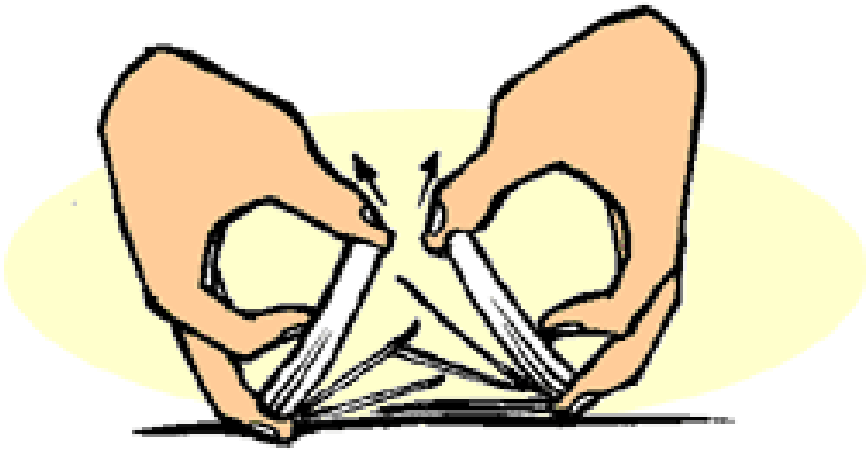
- In card decks it is the shuffle
- In dice it is the roll
- In wheels is it the spin (+ marble)
- All physical sources of randomization (aka entropy)
- Predictable outcomes aren’t “fair”

Java Code Examples

Closer look at shuffles, dice rolls

Card Shuffle

```
public static void shuffle(int card[], int n)
   {

       Random rand = new Random();

       for (int i = 0; i < n; i++)
       {
           // Random for remaining positions.
           int r = i + rand.nextInt(52 - i);

            //swapping the elements
            int temp = card[r];
            card[r] = card[i];
            card[i] = temp;

       }
   }
```

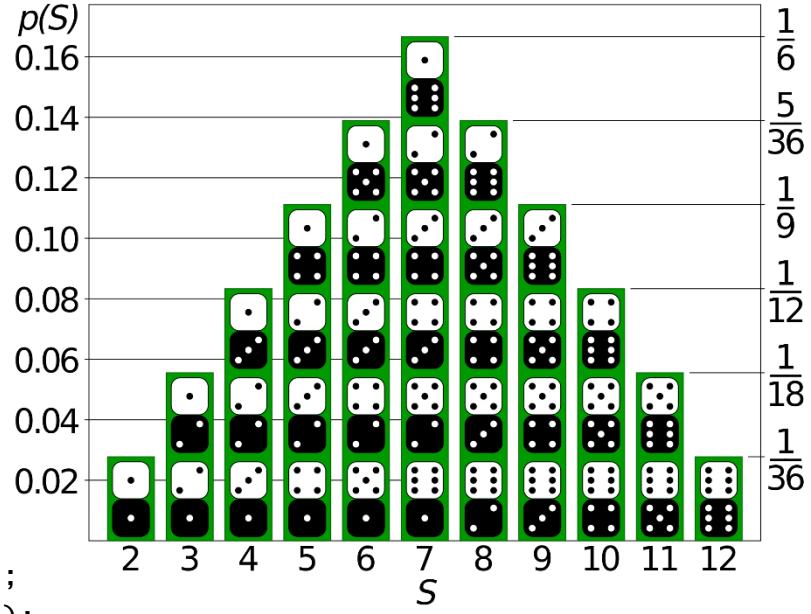


Card Shuffle (via Collections)

```
import java.util.ArrayList;
import java.util.Collections;

public class Test {
    private static final int DECK_SIZE = 52;

    public static void main(String args[]) {
        ArrayList<Integer> deck = new ArrayList<Integer>();

        for (int i = 0; i < DECK_SIZE; ++i) {
            deck.add(i);
        }

        Collections.shuffle(deck);

        System.out.println(deck);
    }
}
```

Dice Roll

```
import java.util.Random;

public class RollTheDice {

public static void main(String[] args) {

        Random diceRoller = new Random();
        int die1;   // The number on the first die.
        int die2;   // The number on the second die.
        int roll;   // The total roll (sum of the two dice).

        die1 = diceRoller.nextInt(6) + 1;
        die2 = diceRoller.nextInt(6) + 1;
        roll = die1 + die2;

        System.out.println("The first die comes up " + die1);
        System.out.println("The second die comes up " + die2);
        System.out.println("Your total roll is " + roll);

    }  // end main()
 }  // end class
```



Quantify Randomness

And why software alone falls short

Random Behavior is Noisy

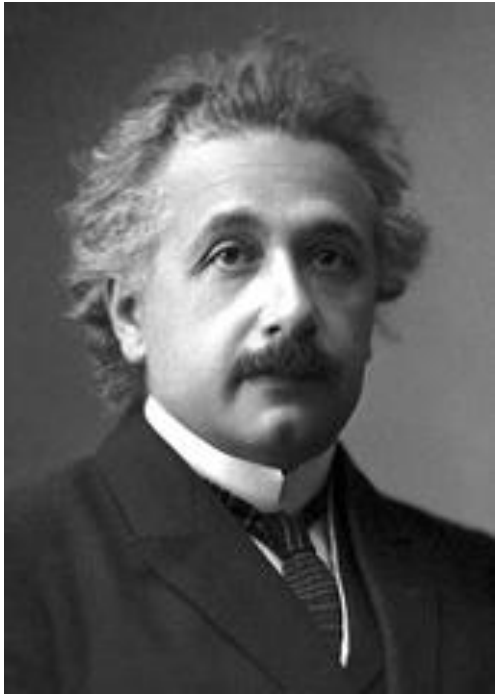
- Highly disordered state == noisy
- Analog circuits always have noise
- Digital circuits always reject noise
- All software rides digital circuits
- Measure of disorder is entropy (randomness) in units of bits

Entropy

Origin Story

On Shoulders of Giants

Albert Einstein



Brownian Motion
Proves Temperature

Sadi Carnot

Proves Losses
from Heat into Work

Ludwig Boltzmann

Loss Proven as
Randomness

Claude Shannon

Information “loss” is
Useful: Uncertainty

Second Law of Thermodynamics

- Entropy is always increasing
- Generally the whole universe tends towards randomness
- Digital circuits (often) don’t have enough randomness
- In software, it takes extra entropy to widen random chances

Not just Games

- High-quality random numbers, through entropy, can be used with scientific, gambling and lottery applications.
- They can improve the performance, security and reliability of servers.
- In Java, RNGs gird the Java Cryptography Architecture, used in all secure communications.
- Secure means: encrypted messages appear to be indistinguishable from random characters.

Where to get Entropy?

Hint: Java gets it from the underlying OS.

Random Selection

- `jre/lib/security/java.security`
- `securerandom.source=file:/dev/random`
- `/dev/random`
- `/dev/urandom`

- eg. You can plug in another RNG

Physical Entropy Source 1

Lava Lamp

Entropy via Thermal

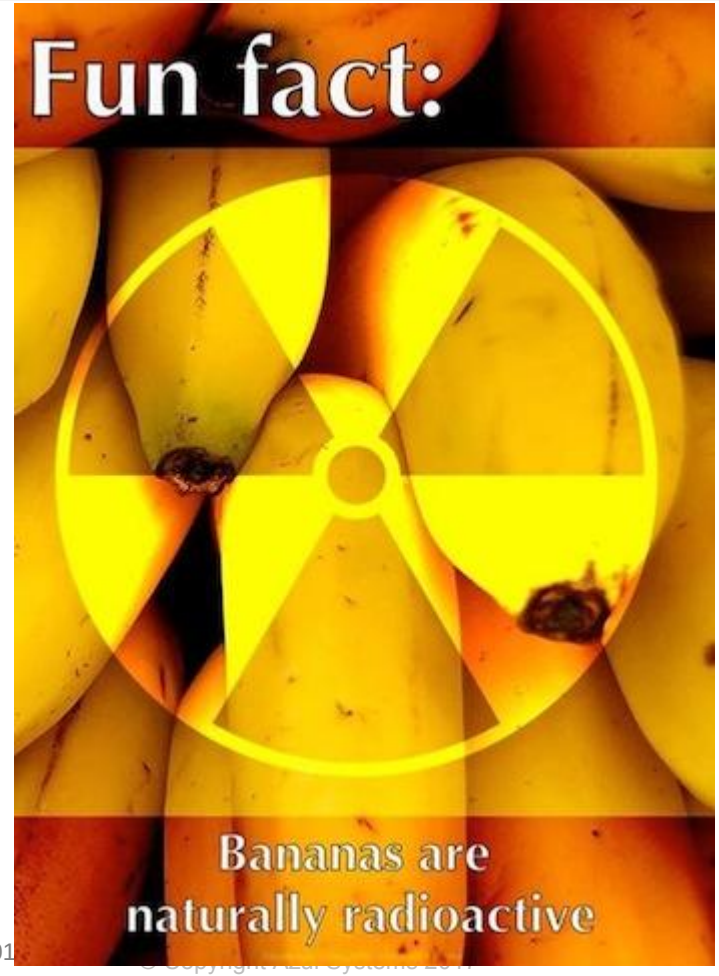
Pros:

- Chaotic non-linear process
- *Groovy!*

Cons:

- Prone to spills
- Hot to touch
- Bulky for IoT

Physical Entropy Source 2



Banana

Entropy by Radioactivity

Pros:

- Potassium K-40
- Average half life is 1.2B years
- Emits beta particles

Cons:

- Need a Geiger counter
- Peels are slippery

Physical Entropy Source 3

Brazil Nut

Entropy by Radioactivity

Pros:

- Radium Ra-226
- Average half life is 1.6k years
- Emits alpha particles

Cons:

- Need a Geiger counter
- Vents Radon gas

Hardware Entropy Sources

Operating principle	Manufacturer
Analog-to-Digital converter noise	Flying Stone Technology
Atmospheric noise	Generic
Avalanche diode	Moonbase Otago
Beam splitter	ID Quantique SA, QuintessenceLabs
Johnson–Nyquist noise	Intel, LETech, WaywardGeek
Mix of Shot noise, Johnson–Nyquist noise, Flicker noise, and some Electromagnetic interference	BitBabbler
Photoelectric effect	Quant-Lab
Registerless Linear Feedback Shift Registers	Kidekin
Reverse biased semiconductor junction	Araneus Information Systems Oy, Altus Metrum, TectroLabs, ubld.it, Simtec Electronics
Shot noise	Comscire, TRNG98
Photon Bunching	Whitewood

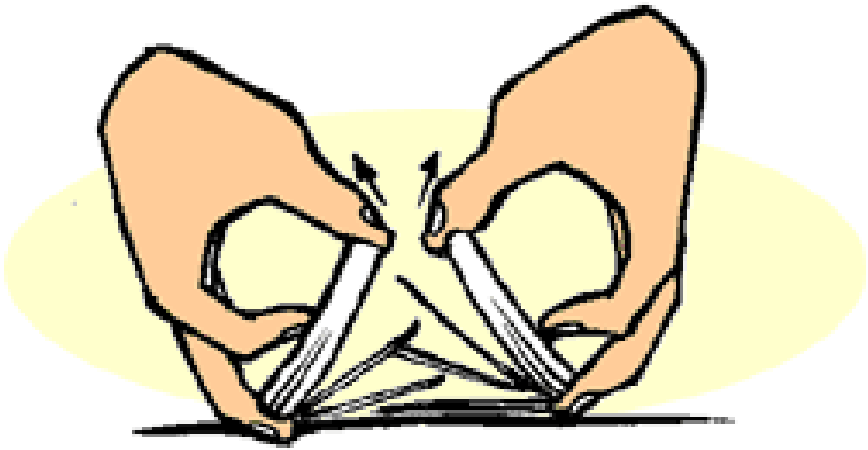
Eg. device uses typically a thermal- or quantum- realm phenomenon, often housed in a portable USB stick. Perfect for IoT!

Java Code Refactored

Improving shuffles, dice rolls

Card Shuffle

```
public static void shuffle(int card[], int n)
   {

       Random rand = new SecureRandom();

       for (int i = 0; i < n; i++)
       {
           // Random for remaining positions.
           int r = i + rand.nextInt(52 - i);

            //swapping the elements
            int temp = card[r];
            card[r] = card[i];
            card[i] = temp;

       }
   }
```



Card Shuffle (via Collections)

```
import java.util.ArrayList;
import java.util.Collections;

public class Test {
    private static final int DECK_SIZE = 52;

    public static void main(String args[]) {
        ArrayList<Integer> deck = new ArrayList<Integer>();

        for (int i = 0; i < DECK_SIZE; ++i) {
            deck.add(i);
        }

        Collections.shuffle(deck, new SecureRandom() );

        System.out.println(deck);
    }
}
```

Dice Roll

```
import java.util.Random;

public class RollTheDice {

public static void main(String[] args) {

        Random diceRoller = new SecureRandom();
        int die1;   // The number on the first die.
        int die2;   // The number on the second die.
        int roll;   // The total roll (sum of the two dice).

        die1 = diceRoller.nextInt(6) + 1;
        die2 = diceRoller.nextInt(6) + 1;
        roll = die1 + die2;

        System.out.println("The first die comes up " + die1);
        System.out.println("The second die comes up " + die2);
        System.out.println("Your total roll is " + roll);

    }  // end main()
 }  // end class
```

Probabilities cast as Entropy

- 1 in 6 — 2.58 bits entropy
- 1 in 38 — 5.25 bits entropy
- 1 in 52 — 5.70 bits entropy

- You want more bits entropy (~2x) than bits in your password cipher

Conclusion and Q&A

Parting thoughts.

Why I care: Fairness

- **Are Online Casino Games Rigged?**
- As much as this question bothers numerous new and experienced players, online casinos ***are not rigged, or fixed***. A casino being rigged means its operations are outside the laws of probability.
- It's actually near-impossible to rig online casino games because of the ***integrity of the software*** used. Reputable casinos use software integrated with ***Random Number Generator (RNG)*** technology and they are audited regularly.

Why I care: Casinos

- I want their business! **$51B** in 2018
- Casino floor machines running Zulu Embedded a superior IoT use case.
- Online action and realtime gaming using Zing removes game lag.
- BestOnlineCasinos.com lists live Java-based sites plus the benefits of Java in gaming
- CasinoTopsOnline lists leading online gaming software developers: Microgaming, Playtech, NetEnt and Realtime Gaming.

Conclusion

- IoT, cryptography, and game play all rely on entropy to achieve security, unpredictability, and fairness.
- JCA cryptography providers use the underlying OS to gather entropy and provide stream of random numbers.
- `SecureRandom()` self-seeds computation of random distributions.
- You must add hardware random numbers generators for inexhaustible entropy pools.

Get Lucky with *Java!*

Further Review

- RFC on Random Number Sources: **tools.ietf.org/html/rfc4086**
- Basics of Entropy: **blogs.cisco.com/security/on_information_entropy**
- Azul: **azul.com**
-  @schuetzematt