Determining an Optimal Threshold on the Online Reserves of a Bitcoin Exchange

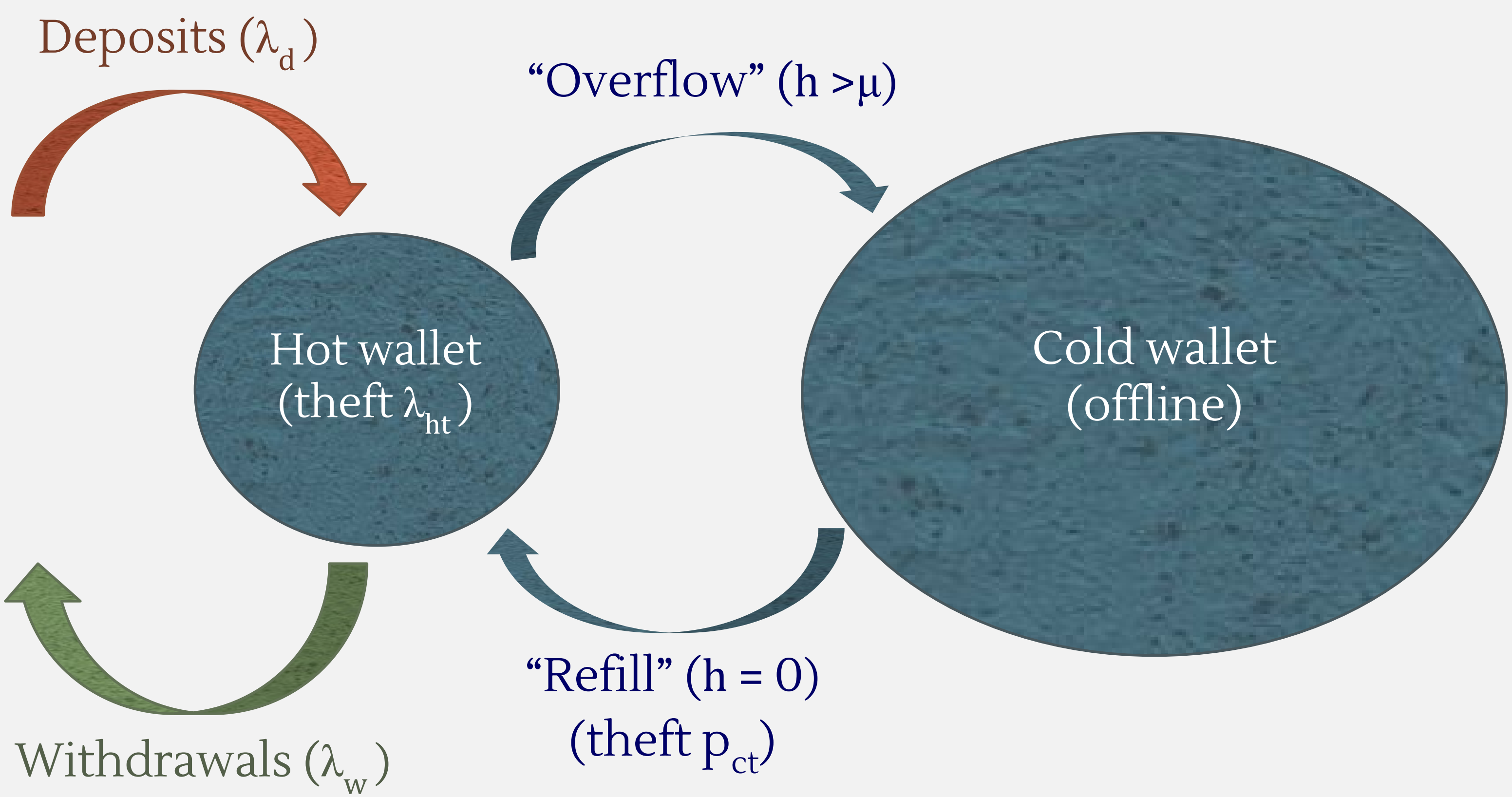
Samvit Jain
Edward Felten
Steven Goldfeder

Princeton University
Department of Computer Science

Overview

- Central question: how to store Bitcoin in a way that reduces impact of theft events
- Key concepts
  - Bitcoin ownership
  - Hot/cold wallet storage
    - Hot – online (e.g. file on computer, smartphone app)
    - Cold – offline (e.g. hard drive locked in safe, paper wallet)

Problem Formulation



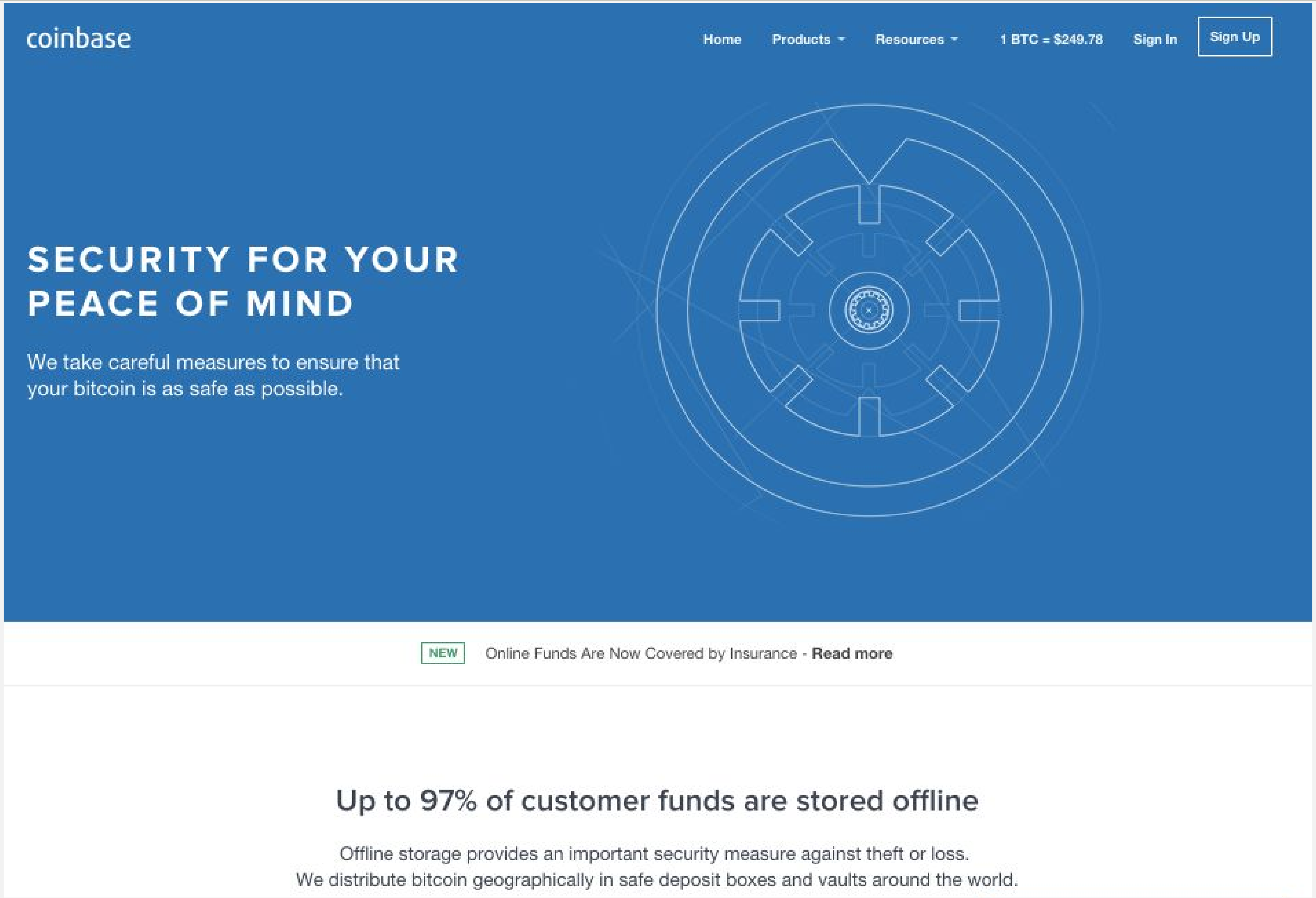
Motivation

- Prevalence of high profile Bitcoin theft
  - 45% of exchanges ever operational shut down (2013)
- Theft nullifies advantages of using Bitcoin
  - Subsidized by higher insurance premiums and exchange fees
- Theft undermines public trust in Bitcoin
  - Influences exchange rate, funding climate, community growth

Related Work

- Companies (e.g. Coinbase) implement security heuristics
  - Data encryption, safe storage, geographic diversification
  - Practices don't generalize, aren't necessarily optimal
- Significant research on improving Bitcoin wallets
  - Extensions to core protocol, cryptographic innovations
- What's missing: system analysis at a *given level of security*
  - Goal: better high-level designs for storage systems

Related Work



Approach

- Goal: maximize hot/cold wallet balance at arbitrary time T
- Formula for expected total balance
  - By linearity of expectation

$$B(T) = Ex[D - W] - k_1\mu - \frac{k_2}{\mu}$$

  - D – W represents net arrivals (deposits minus withdrawals)
  - $k_1\mu$ represents losses due to hot wallet theft
  - k_2/μ represents losses due to cold wallet theft

Approach

- Determine optimal hot wallet threshold μ

$$B(T) = Ex[D - W] - k_1\mu - \frac{k_2}{\mu}$$

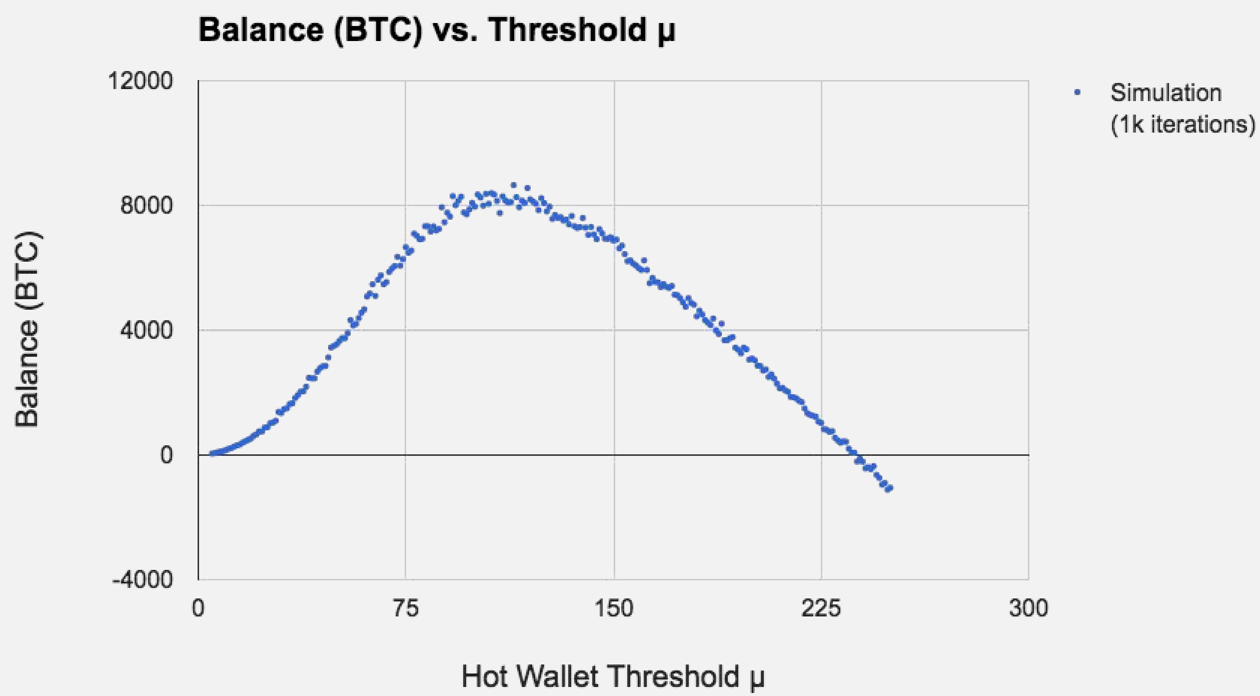
  - Optimize B(T) by setting first derivative to 0 and solving for μ

$$\frac{dB(T)}{d\mu} = -k_1 + \frac{k_2}{\mu^2} = 0$$

$$\mu = \sqrt{\frac{k_2}{k_1}}$$

Approach

- Experimental evidence that optimal μ can be found



Theory

- Series of models, each a larger subsystem of original setup
  - <u>Model 1</u>: Hot wallet only. No thefts.
  - <u>Model 2</u>: Hot wallet only. Hot wallet theft with rate λ_{ht}
  - <u>Model 3</u>: Hot and cold wallet.

Theory

- The result

Mean rate of net arrivals

Expected time to cold wallet theft

Average hot wallet balance

Expected number of hot wallet thefts

$$Ex[B] = (\lambda_d - \lambda_w)\frac{X_\mu}{p_{\mathrm{t_c}}} - (\gamma\mu)\left(\lambda_{\mathrm{t_h}}\frac{X_\mu}{p_{\mathrm{t_c}}}\right)$$

  - Gives hot/cold wallet balance after long time T
  - Must only look at events *since last cold wallet theft*
    - First term - expected net arrivals
    - Second term - losses due to hot wallet theft

Expected time to empty hot wallet (C→H refill)

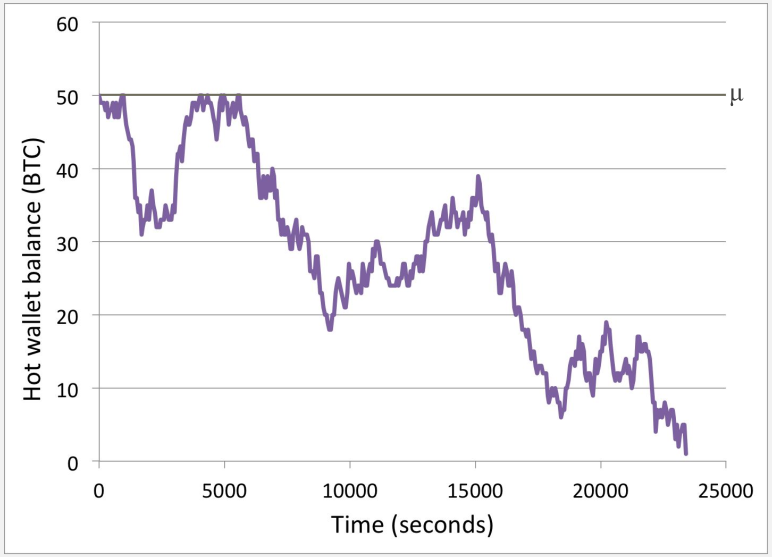
$$\frac{X_\mu}{p_{\mathrm{t_c}}}$$

Probability of cold wallet theft

Last cold wallet theft in [0, T]

Theory

- How to find X_μ (time to empty hot wallet)
  - Model hot wallet balance as *continuous time random walk*



Event	State Transition	Probability
Deposit	$X_k \rightarrow X_{k+1}$	$\boldsymbol{\lambda}_d t$
Withdrawal	$X_k \rightarrow X_{k-1}$	$\boldsymbol{\lambda}_w t$
Hot Wallet Theft	$X_k \rightarrow X_0$	$\boldsymbol{\lambda}_{ht} t$
No Event	$X_k \rightarrow X_k$	$1 - (\boldsymbol{\lambda}_d + \boldsymbol{\lambda}_w + \boldsymbol{\lambda}_{ht})t$

Theory

- How to find X_μ (time to empty hot wallet)
  - Can write recurrence relation

$$X_k = t + (\lambda_d t)X_{k+1} + (\lambda_w t)X_{k-1} + (\lambda_{\mathrm{t_h}} t)X_0 + (1 - (\lambda_d + \lambda_w + \lambda_{t_h})t)X_k$$

    - LHS — expected time to reach 0 BTC from k BTC
    - RHS — t + expected time to reach 0 BTC after passage of small time t
  - Solve recurrence
    - Write/solve characteristic polynomial
    - Impose boundary conditions (X_0 and X_μ)

Theory

- How to find X_μ (time to empty hot wallet)
  - Solution

$$X_\mu = \frac{1}{\lambda_{t_h}} + \frac{1}{\lambda_{t_h}}\left(\frac{\lambda_w(x_2 - x_1)(x_1x_2)^{\mu-1}}{[\lambda_w(x_1-1)+\lambda_{t_h}x_1]{x_1}^{\mu-1} - [\lambda_w(x_2-1)+\lambda_{t_h}x_2]{x_2}^{\mu-1}}\right)$$

Experiment

- Event-driven simulation
  - Set values for $\boldsymbol{\lambda}_d$, $\boldsymbol{\lambda}_w$, $\boldsymbol{\lambda}_{ht}$, $\mathbf{p}_{ct}$ (e.g. 80/hour, 78/hour, 0.01/hour, 0.01/access)
  - Set simulation parameters — threshold μ, timespan T, iterations i

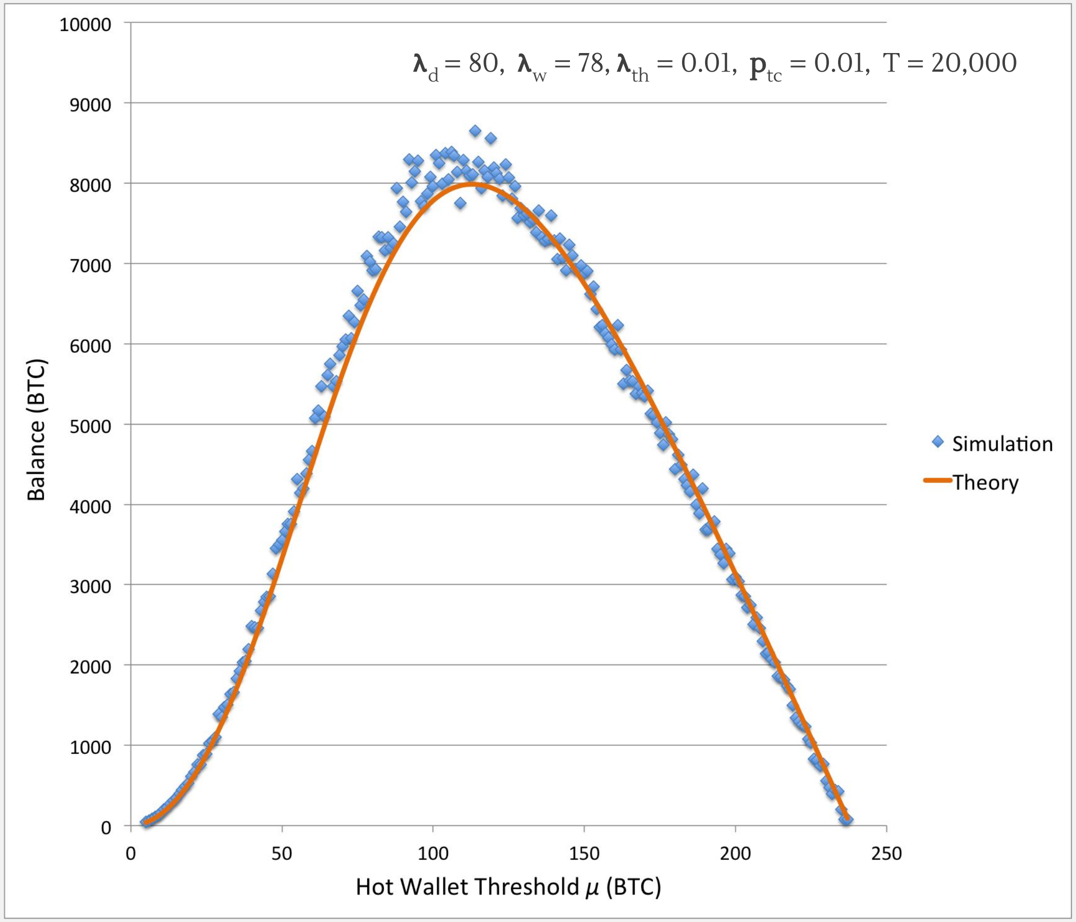
```
while (time < T) {

     Event nextEvent = drawEvent(λd, λw, λht)
     switch (nextEvent.Type) {
          case (Event.DEPOSIT):          deposit()
          case (Event.WITHDRAWAL):       withdraw()
          case (Event.HOT_THEFT):        emptyHotWallet()
     }
     if (hotBalance == 0)                refillHotWallet(pct)

     time += nextEvent.Time
}

print(mu, hotBalance + coldBalance)
```

Results



Theory

$$Ex[B] = (\lambda_d - \lambda_w)\frac{X_\mu}{p_{\mathrm{t_c}}} - (\gamma\mu)\left(\lambda_{\mathrm{t_h}}\frac{X_\mu}{p_{\mathrm{t_c}}}\right)$$

Simulation

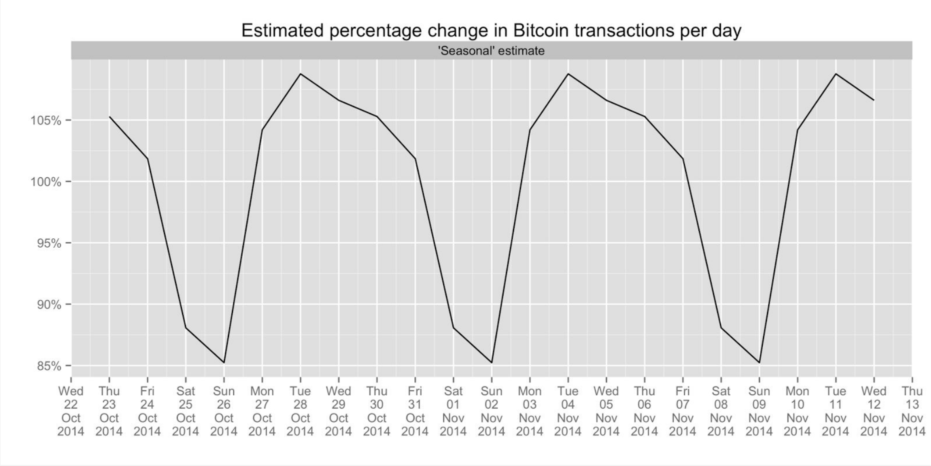
Average over 1000 iterations

Optimal Threshold

Theory	$\mu = 112.88$
Empirical (abs. maximum)	$\mu = 114$
Empirical (poly. interpolation)	$\mu = 111.05$

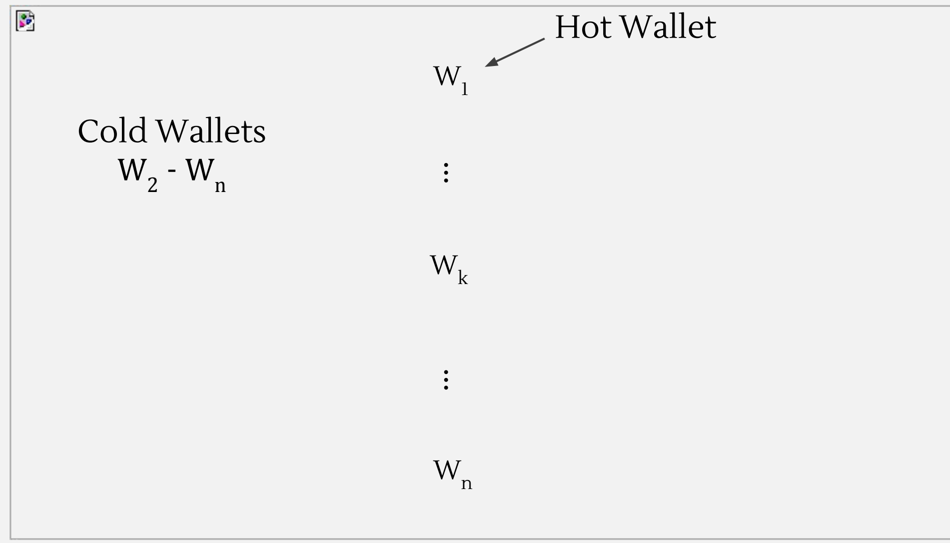
Applications

- Calibrated threshold
  - Exploits periodicities in transaction frequency
  - Organization maintains *history block*
    - Record of last k hours of deposits, withdrawals, thefts, C→H transfers
    - Capacity of hot wallet (i.e. μ) computed and updated dynamically



Further Work

- Multiple wallet systems
  - Goal: separate servicing from storage
  - Pyramid model
    - Layers of security
    - Wallet W_k overflows into W_{k+1} and refills W_{k-1}
    - Lower layers hold majority of reserves



  - Question: optimal threshold for each level?

References

Special thanks to our reviewers and to WEIS for hosting this event

Papers

- T. Moore and N. Christin, "Beware the Middleman: Empirical Analysis of Bitcoin-Exchange Risk," in FC 2013, Springer, 2013, pp. 25-33.

Books

- A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder, *Bitcoin and Cryptocurrency Technologies*, Princeton University, 2015.

Images

- https://www.coinbase.com/security
- http://organofcorti.blogspot.com/2014/11/daily-and-weekly-bitcoin-transaction.html
- https://pixabay.com/en/pyramids-layers-blue-3d-305074/

See https://github.com/SamvitJ/WEIS2016-Programs for simulation code

Appendix

- Expectation
  - Sum rule (linearity) — Ex[A + B] = Ex[A] + Ex[B]
  - Product rule — Ex[AB] = Ex[A] · Ex[B]
- Poisson processes
  - Linear rate scaling — **λ** expected arrivals in time 1
T**λ** expected arrivals in time T
  - Memorylessness — Time to next arrival *not* dependent on time waited