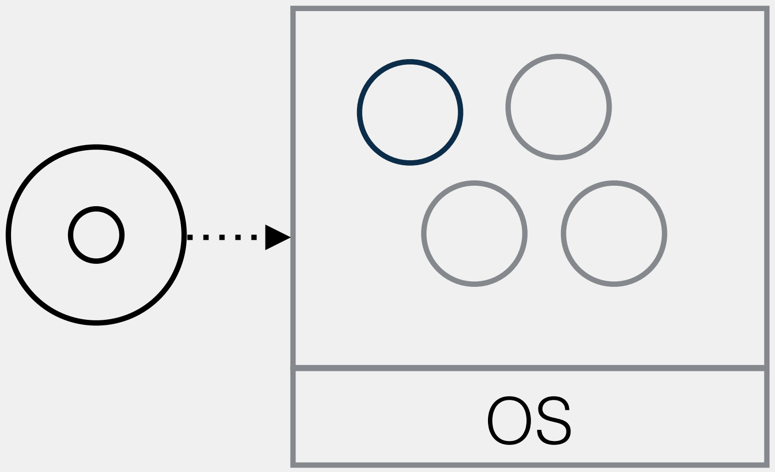
Security of Internet-Scale Services

Thesis Defense — Adam Everspaugh

Committee:
Prof Nigel Boston
Prof Barton Miller
Prof Somesh Jha
Assc Prof Thomas Ristenpart*
Prof Michael Swift*

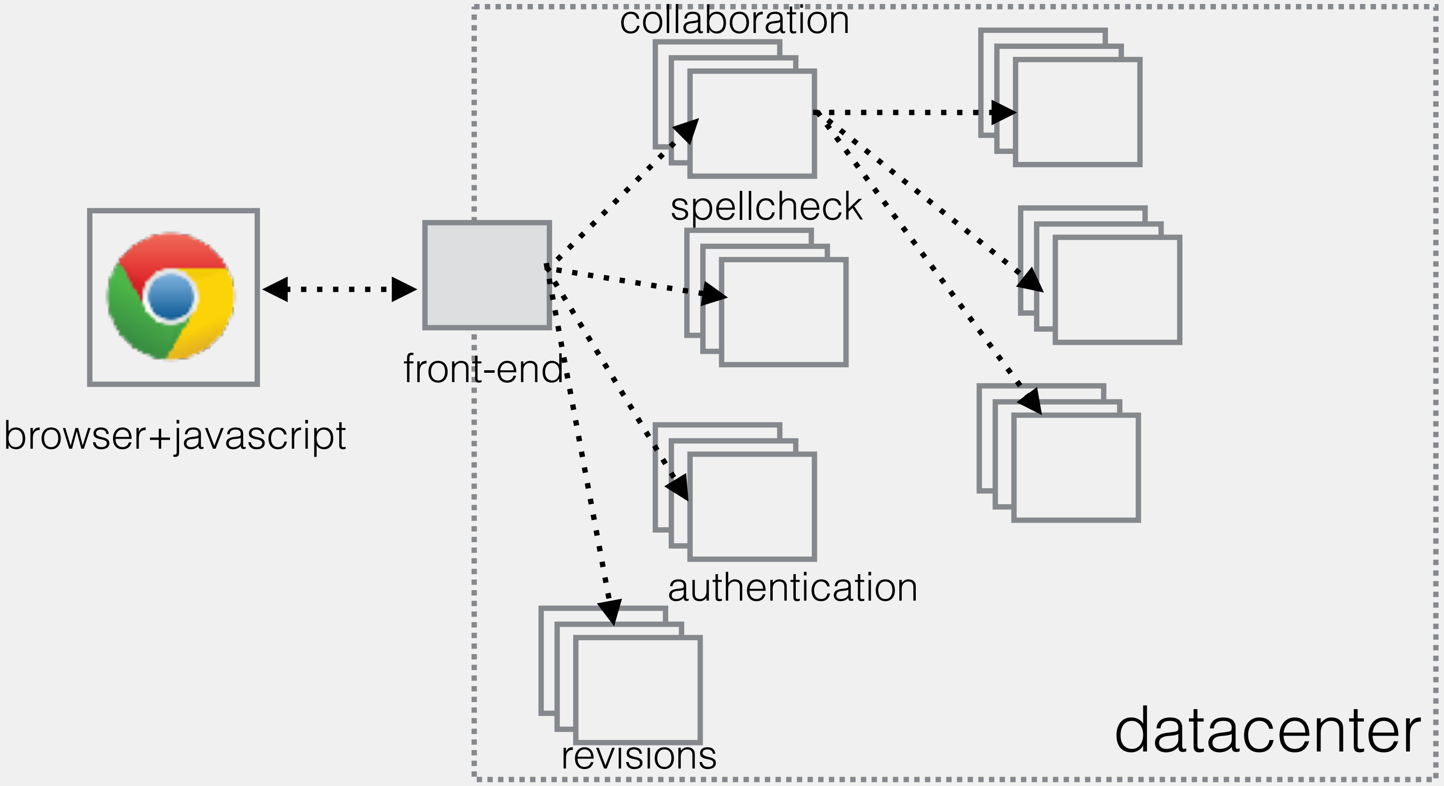
Software Environment Has Changed

Microsoft Word, ~1995



users: 2^0-2^5
machines: 2^0

Google Docs, 2015



users: 2^{20}-2^{30}
machines: 2^{10}

Interesting Properties of Internet-scale Services

- Millions or billions of users
- Geo-replicated applications and storage systems
- Applications built as distributed services: componentized, communication, failures, concurrency
- Highly available: 1.0 - ε
- Security?
  - Carried forward from previous era of application development

Research Question

Can we improve the security of internet-scale services?

- **Not-So-Random Numbers [IEEE S&P '14].** Evaluate RNGs in virtual machine and and cloud compute environments.
- **Pythia PRF Service [Usenix Sec '15].** Design and evaluate a secure password authentication service built around a new cryptographic primitive.
- **Key Rotation for Auth Encryption [Crypto '17].** Examines updatable encryption for cloud storage. Formal analysis of security notions and updatable encryption schemes.

Random Number Generators

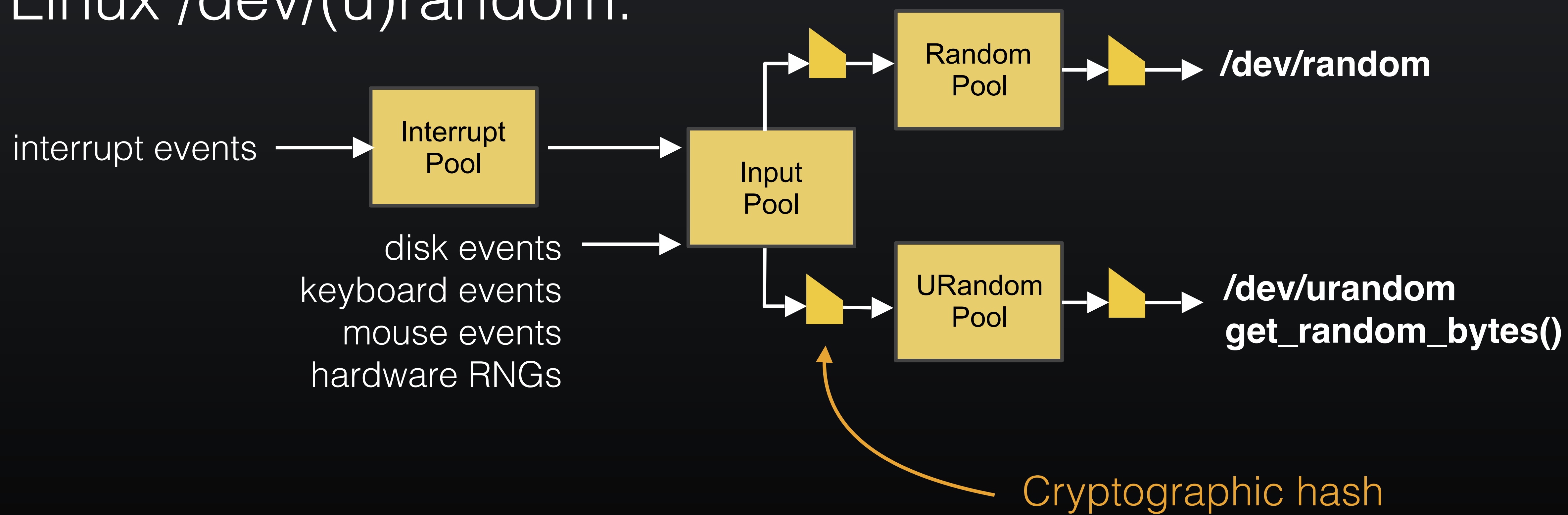
Example uses:

- StackProtector canaries
- TCP/IP sequence numbers
- Cryptographic keys

Random Number Generators

Random Number Generators

Linux /dev/(u)random:



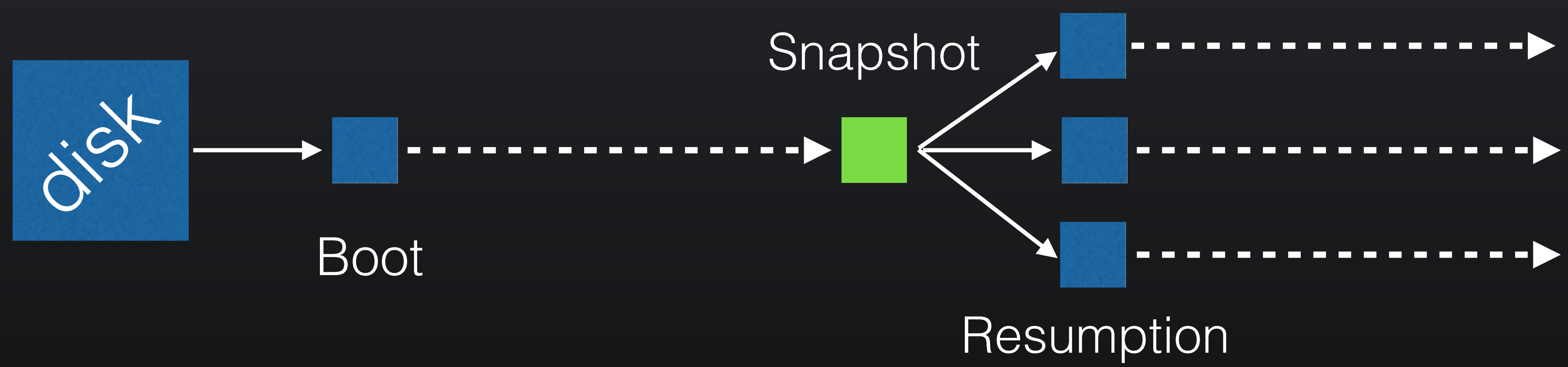
Random Number Generators

Inputs
system events → **RNG** → Outputs
uniformly distributed

Folklore concerns regarding security

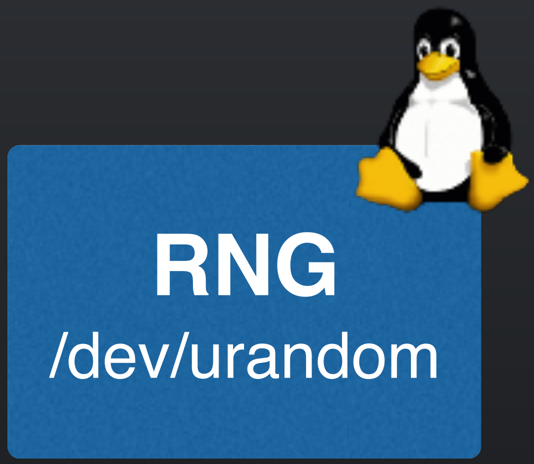
1. Do full-memory snapshots cause problems for system RNGs? [GR05] [RY10]
2. Are input sources entropy-poor inside a virtual machine? [SBW09]

Virtual Machine Snapshot and Resumption



Does the RNG produce distinct outputs with each resumption?

Linux RNG ***Not*** Reset Secure



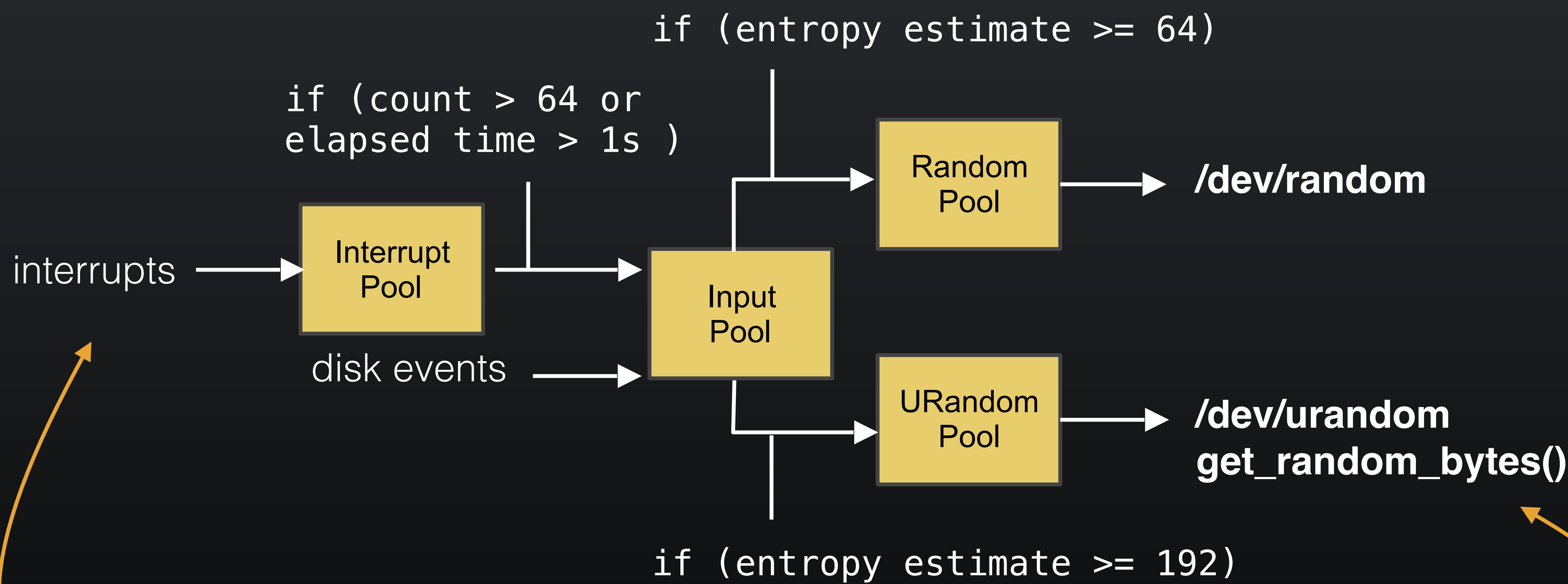
One experiments:

- Boot VM in Xen, idle for 5 minutes
- Start measurement process, capture snapshot
- Resume from snapshot,
  read 512-bits from /dev/urandom every 500 us

Repeat for 8 distinct snapshots
Do 20 resumptions/snapshot

7/8 snapshots produce repeated outputs

Why does this happen?



Buffering and thresholds prevent new inputs from impacting outputs

Linux /dev/(u)random

Reset Vulnerabilities Effect Other Platforms

FreeBSD
/dev/random produces identical output stream
Up to 100 seconds after resumption

Microsoft Windows 7
Produces repeated outputs indefinitely

`rand_s`	(stdlib)
`CryptGenRandom`	(Win32)
`RngCryptoServices`	(.NET)

RNG Summary

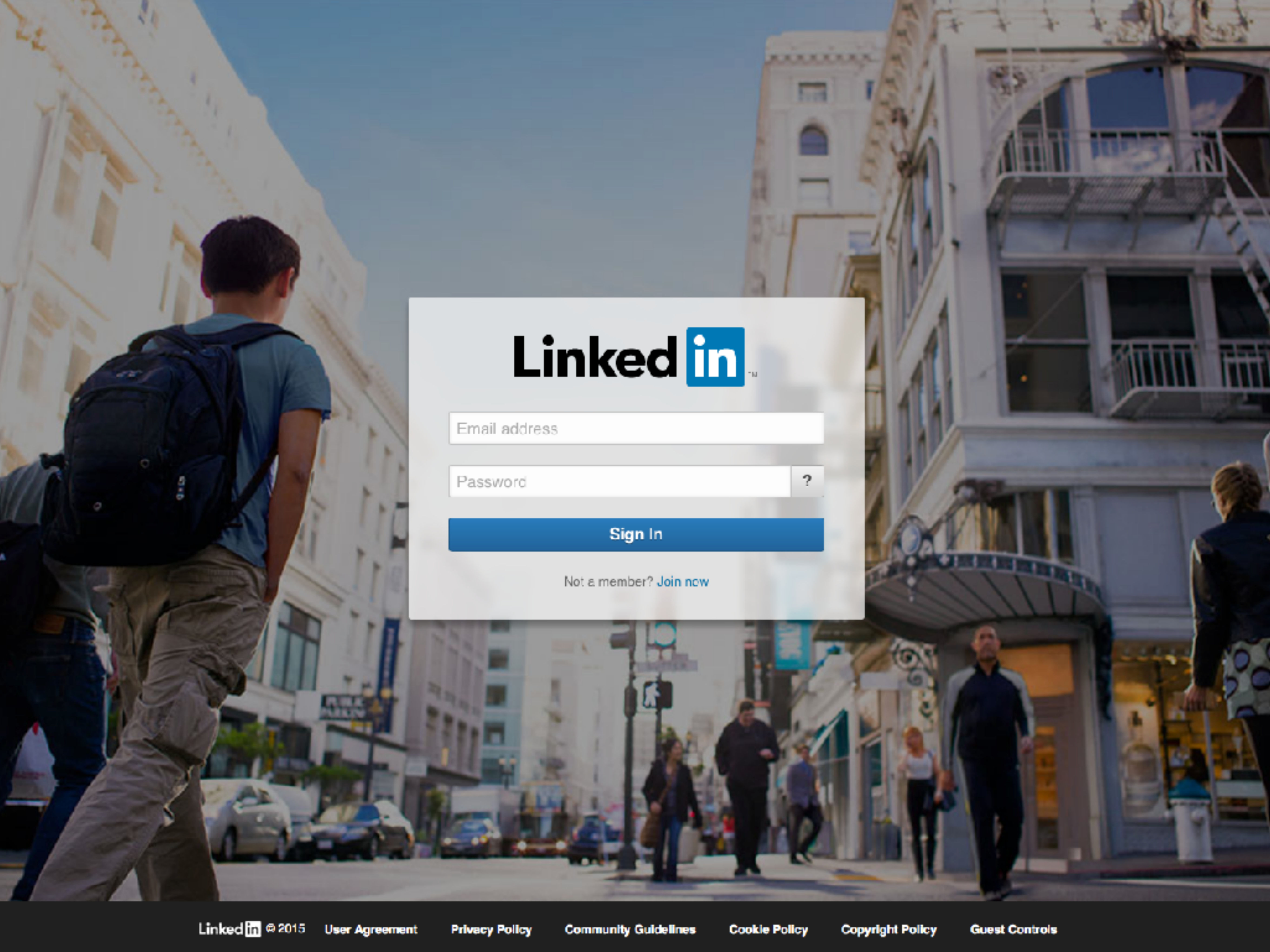
- Snapshots cause problems? → **Yes**
- Entropy-poor inputs? → **No**
- New clean-slate RNG design → **Whirlwind**

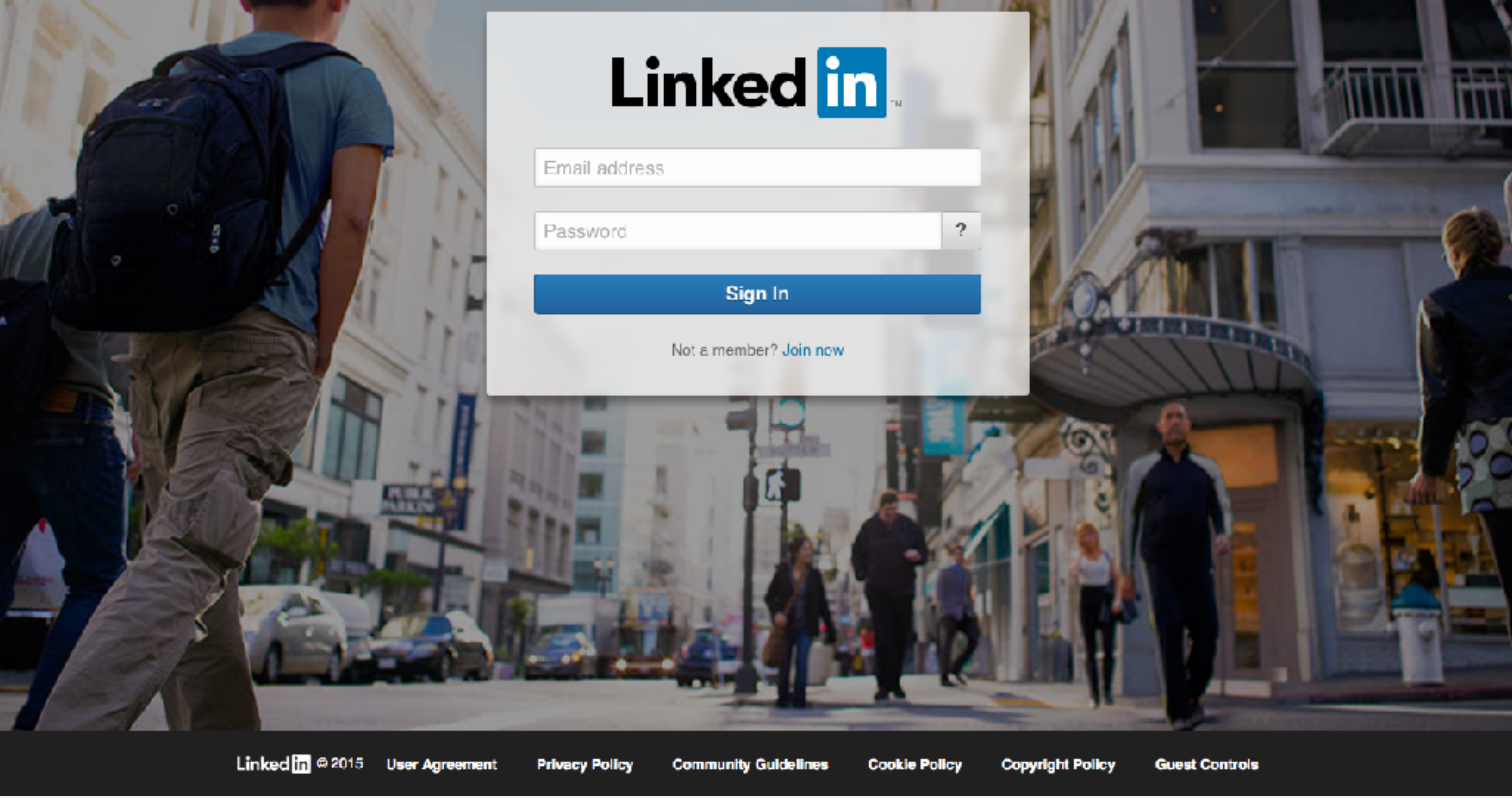
Outline

- **Not-So-Random Numbers [IEEE S&P '14].** Evaluate RNGs in virtual machine and and cloud compute environments.
- **Pythia PRF Service [Usenix Sec '15].** Design and evaluate a secure password authentication service built around a new cryptographic primitive.
- **Key Rotation for Auth Encryption [Crypto '15].** Examines updatable encryption for cloud storage. Formal analysis of security notions and updatable encryption schemes.

Password Database Compromises

Password DB breaches are common


LinkedIn
Email address
Password
?
Sign In
Not a member? Join now
LinkedIn © 2015
User Agreement
Privacy Policy
Community Guidelines
Cookie Policy
Copyright Policy
Guest Controls



Website stores one of:

- pw
- Hash(pw)
- salt, Hash(salt, pw)
- salt, Hash^{4096}(salt, pw)

6.5M hashes leaked

90%

recovered 2 weeks

Facebook’s Password Onion

```
$cur  = ‘password’
$cur  = md5($cur)
$salt = randbytes(20)
$cur  = hmac_sha1($cur, $salt)
$cur  = remote_hmac_sha256($cur, $secret)
$cur  = scrypt($cur, $salt)
$cur  = hmac_sha256($cur, $salt)
```

[Moffet RWC15]

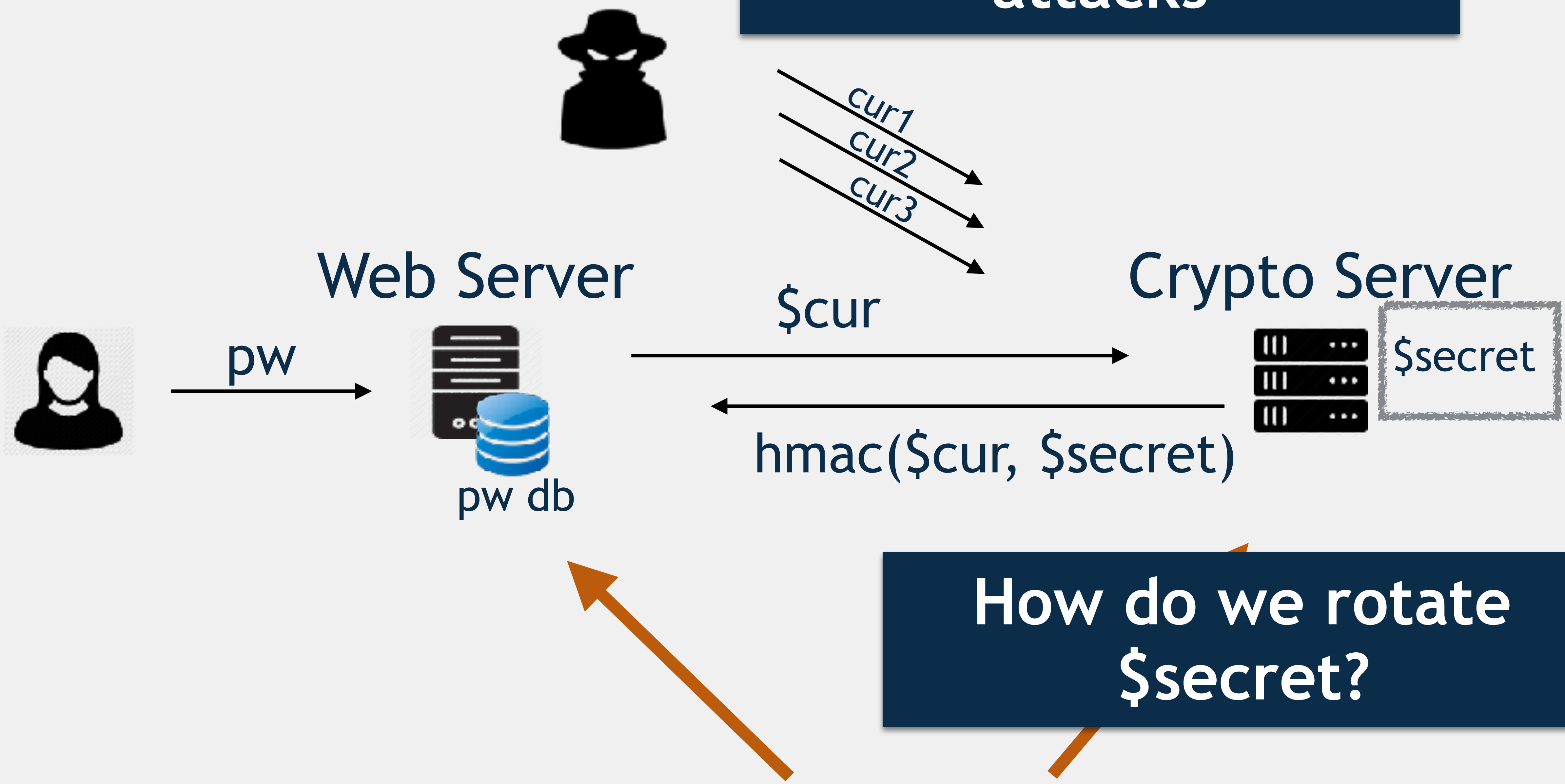
Facebook’s Password Onion

```
$cur  = ‘password’
$cur  = md5($cur)
$salt = randbytes(20)
$cur  = hmac_sha1($cur, $salt)
$cur  = remote_hmac_sha256($cur, $secret)
$cur  = scrypt($cur, $salt)
$cur  = hmac_sha256($cur, $salt)
```

[Moffet RWC15]

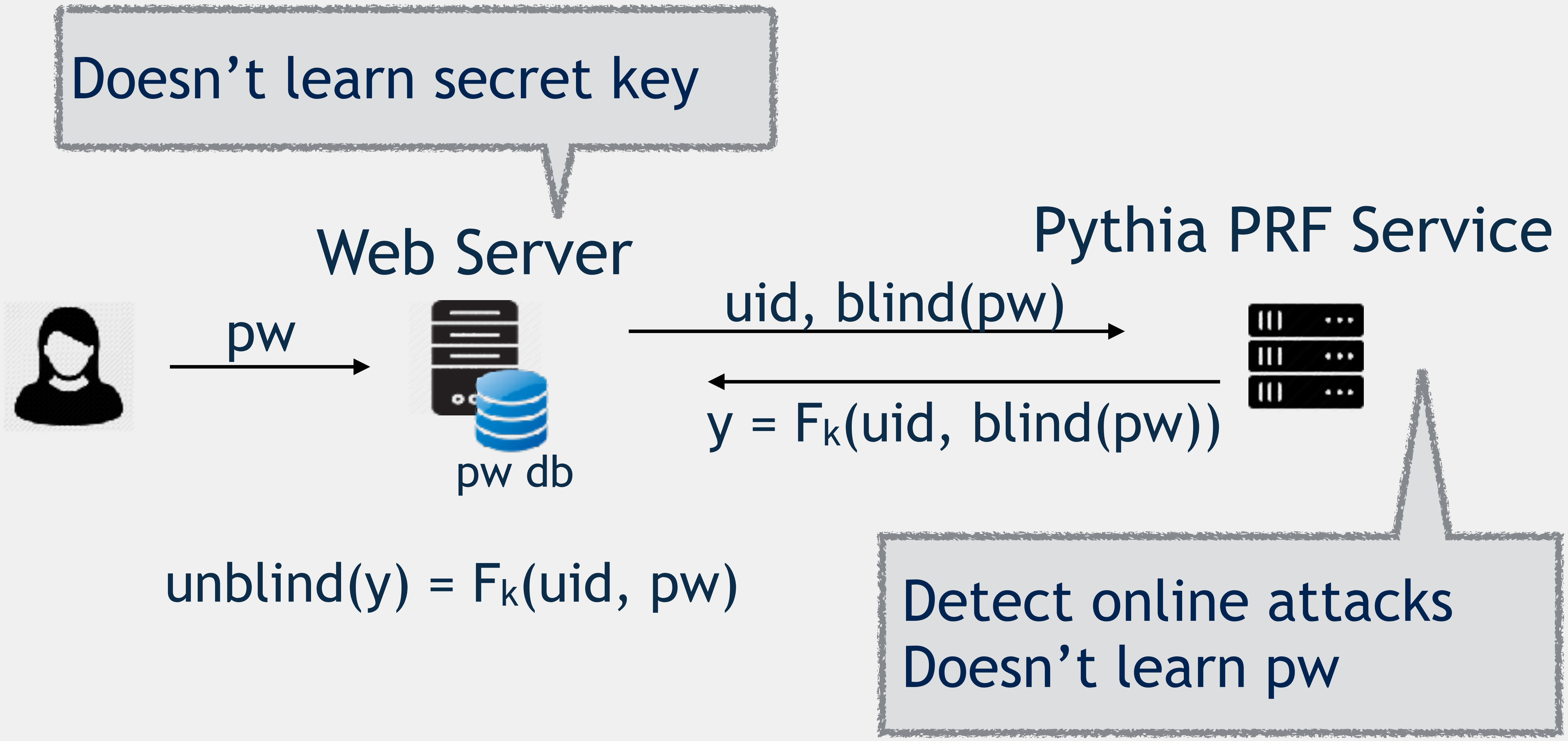
Remote HMAC Distributes Trust

Hard to detect online attacks

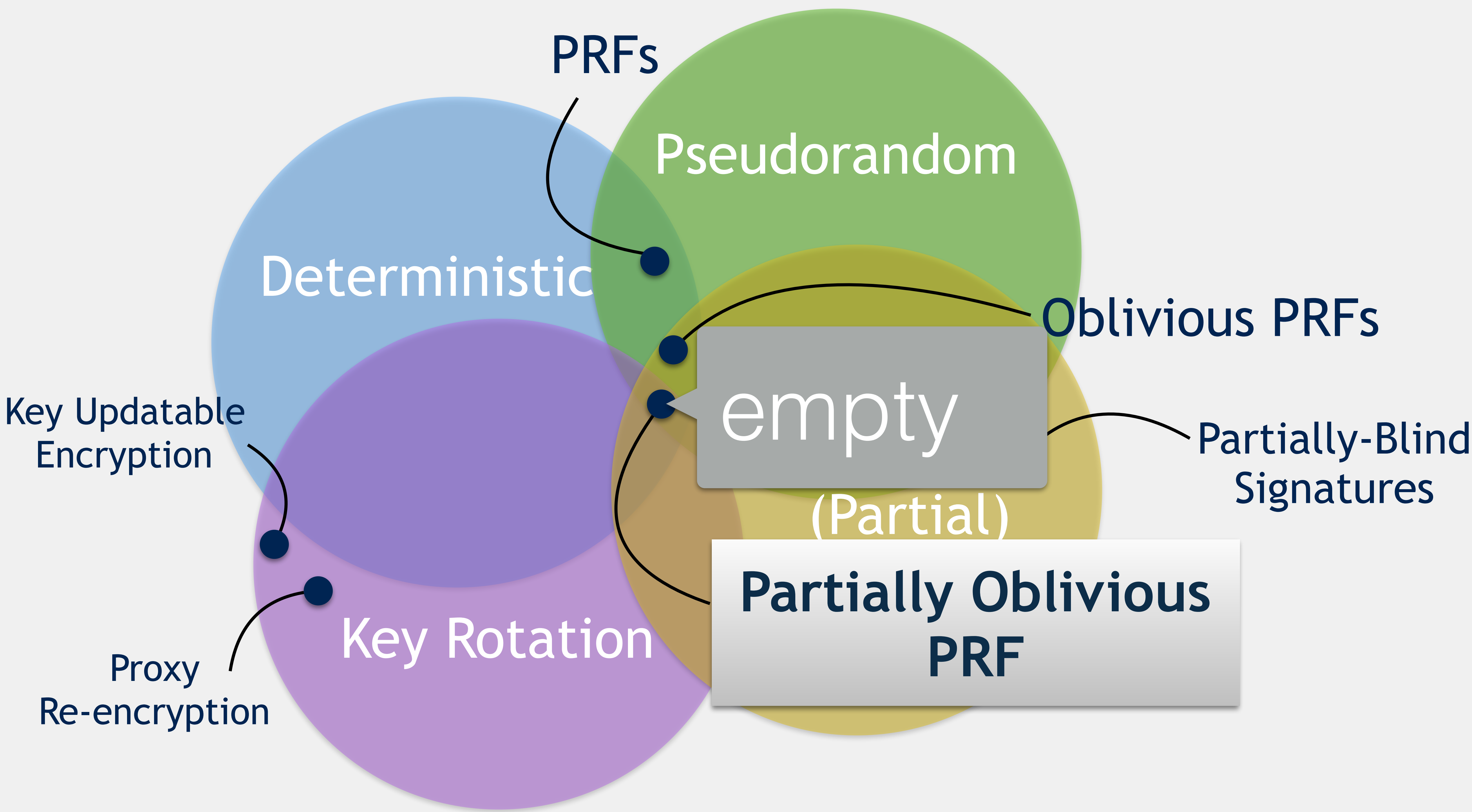


How do we rotate $secret?

Advantages of Partially Oblivious PRF



Existing Crypto Primitives are Insufficient



Fast, Scalable PRF Service

Pythia Query

5.2 ms

Iterated Hashing

8.9 ms

($SHA256^{10k}$)

Throughput: 1350 queries/sec (8-core EC2 instance)

Within factor of 2 of HTTP GET over TLS

Storage: O(1) per web server

Supports arbitrary number of users for each web server

100M Web Server: 18.6 GB (keytable)

nginx

python

django

MongoDB

Relic toolkit

Outline

- **Not-So-Random Numbers [IEEE S&P '14].** Evaluate RNGs in virtual machine and and cloud compute environments.
- **Pythia PRF Service [Usenix Sec '15].** Design and evaluate a secure password authentication service built around a new cryptographic primitive.
- **Key Rotation for Auth Encryption [Crypto '17].** Examines updatable encryption for cloud storage. Formal analysis of security notions and updatable encryption schemes.

Encryption for Cloud Storage

k_1 — secret key

file_0 $\{\text{file}_0\}_{k1}$

$\{\text{file}_5\}_{k1}$

$\{\text{file}_1\}_{k1}$ $\{\text{file}_4\}_{k1}$

$\{\text{file}_2\}_{k1}$ $\{\text{file}_3\}_{k1}$

How do we rotate k_1?

$k_1, k_2 \rightarrow \boxed{} \rightarrow \Delta_{1\to 2}$

Rekey token

$\xrightarrow{\Delta_{1\to 2}}$

$\{\text{file}_1\}_{k2}$ $\{\text{file}_2\}_{k2}$

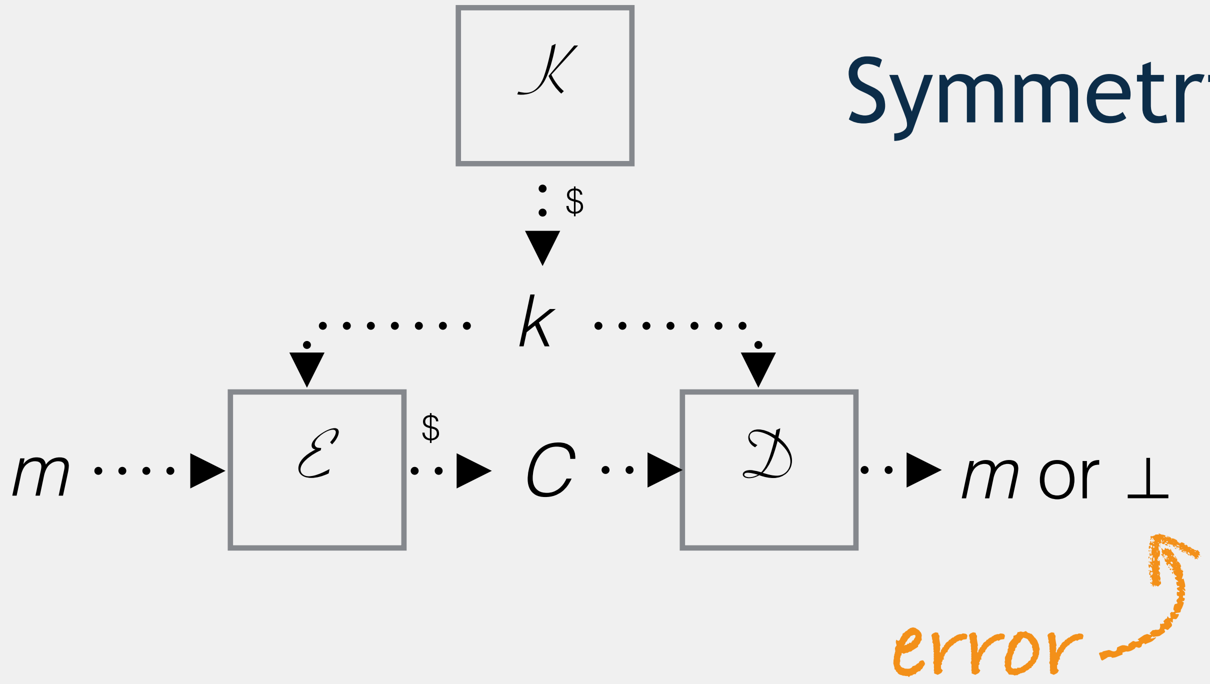
$\{\text{file}_3\}_{k2}$

$\{\text{file}_4\}_{k2}$

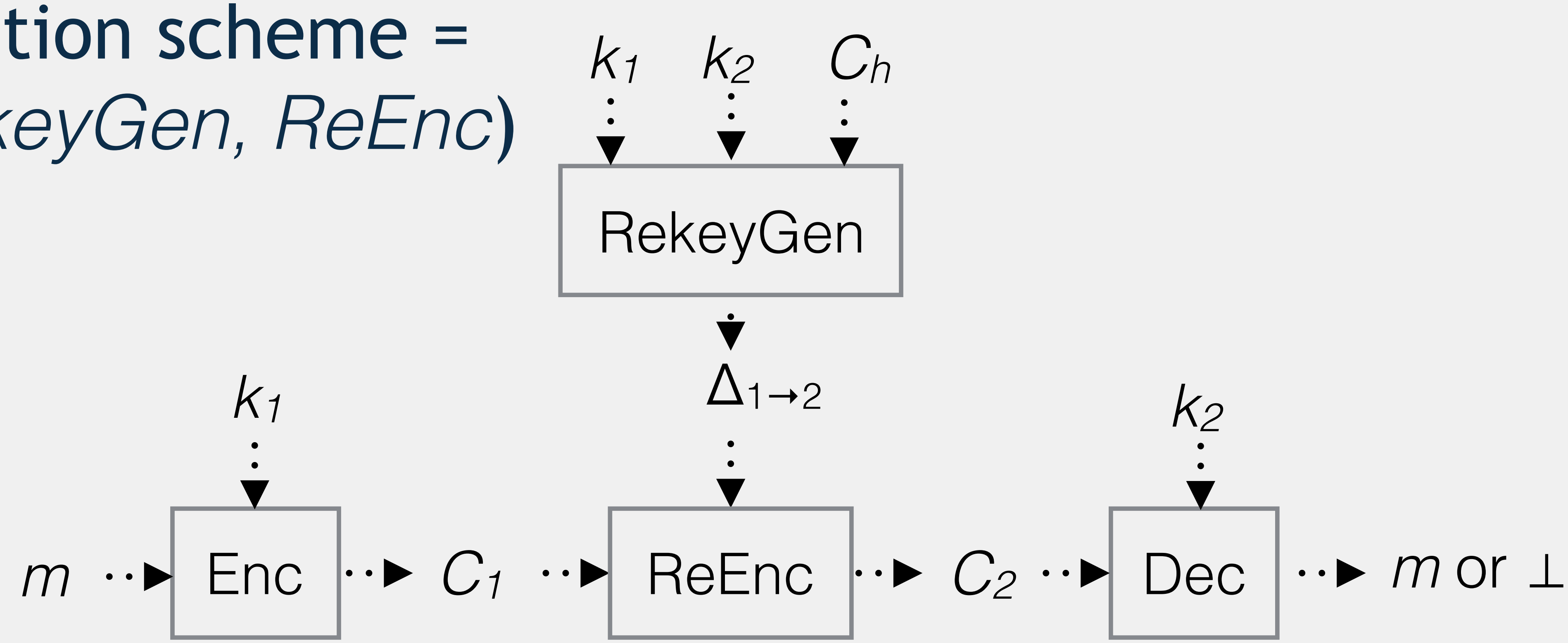
$\{\text{file}_5\}_{k2}$

Updatable Encryption

Symmetric Encryption scheme = $(\mathcal{K}, \mathcal{E}, \mathcal{D})$



Updatable Encryption scheme =
(*Kg*, *Enc*, *Dec*, *RekeyGen*, *ReEnc*)



Security Notions

Symmetric Encryption scheme	Updatable Encryption scheme
Confidentiality: **Ind-Cpa** (indistinguishable to chosen-plaintext attack)	Confidentiality: **Up-Ind**
Integrity: **Int-Ctxt** (integrity of ciphertext)	Integrity: **Up-Int**
Authenticated Encryption: **AE $\Rightarrow$ Ind-Cpa $\wedge$ Int-Ctxt**	Indist. ReEncryption: **Up-ReEnc**

Security of Updatable Schemes

	Confidentiality (Up-Ind)	Integrity (Up-Int)	Indist. ReEncryption (Up-ReEnc)
AE-hybrid	X	X	X
KSS*	✔	✔	X
[BLMR13]	X	X	X
ReCrypt*	✔	✔	✔

* introduced in this work

AE-hybrid is Not Secure

Updatable encryption built with symmetric authenticated encryption (AES-GCM)

$$\mathrm{Enc}_{k1}(m): \quad \underbrace{\{x\}_{k1}}_{\text{header}} \quad \underbrace{\{m\}_x}_{\text{body}} \quad = C_1$$

$$\mathrm{ReEnc}(\Delta, C_1): \quad \{x\}_{k2} \quad \{m\}_x \quad = C_2$$

Give the attacker:
k_1, all headers, C_2

Confidentiality (Up-Ind)

Integrity (Up-Int)

AE-hybrid in production use: amazon web services Google Cloud Platform

AE-hybrid Fixed: KSS

	AE-Hybrid	KEM/DEM with Secret Sharing (KSS)
$\mathrm{Enc}_{k1}(m)$	$\{x\}_{k1}\ \{m\}_x$	$\{x \oplus y, h(m)\}_{k1}\quad y, \{m\}_x$

Key-share hides x in header

Hash gives integrity — binds header/body

✔ Confidentiality (Up-Ind)

✔ Integrity (Up-Int)

✘ Indist. ReEnc (Up-ReEnc)

Strongest Security: ReCrypt

Key Homomorphic Encryption

$E_b(E_a(m)) = E_{a\odot b}(m)$ $D_{a\odot b}(E_{a\odot b}(m)) = m$

d=h(m); gives integrity

Enc: $\underbrace{\{x+y,\ E_x(d)\}_{k1}}_{\text{header}}$ $\underbrace{y, E_x(m)}_{\text{body}}$

ReEnc: $\{x'+y'+x+y,\ E_{x'}(E_x(d))\}_{k2}$ $y'+y,\ E_{x'}(E_x(m))$

- ✔ Confidentiality (Up-Ind)
- ✔ Integrity (Up-Int)
- ✔ Indist. ReEnc (Up-ReEnc)

Strongest Security Impacts Performance

ReCrypt	1 KB	1 GB
Encrypt	10.0 ms	2.6 hrs
ReEnc	8.8 ms	2.4 hrs
Decrypt	9.1 ms	2.4 hrs

ReCrypt operations are 1000x slower than KSS

- Good fit for: small, high-value plaintexts
- E.g. credit card numbers, personally-identifying information, financial information

Conclusions

There are significant opportunities for improving the security of internet-scale services.

- **Not-So-Random Numbers** [IEEE S&P '14]
  Environment is fine — entropy rich inputs.
  New designs fix VM reset vulnerabilities; easier to analyze.
- **Pythia PRF Service** [Usenix Sec '15]
  State-of-the-art is broken — new cryptography in service-oriented setting is a great direction.
- **Key Rotation for Auth Encryption** [Crypto '17]
  Customers need updatable encryption — proper balance of security strength and performance is still an open question.