The FBI created a fake malware-spreading newspaper article to trace a bomb threat

COMMENTS

By Russell Brandom on October 28, 2014 11:22 am Email @russellbrandom

DON'T MISS STORIES FOLLOW THE VERGE

Like Follow Subscribe Follow

SIGN UP EMAIL NEWSLETTE

The best of The Verge, delivered daily – sign up for The Verge Newsletter.

Email address... SIGN UP

THE LATEST HEADLINES

A consumer Oculus Rift is still 'many months' away

Shakespeare's Globe launches an on-demand service with over 50 plays

'Grand Theft Auto 5' will have a first-person mode on Xbox One and PlayStation 4

Share on Facebook (639)

In 2007, the FBI was tra

Documents uncovered by the Electronic Frontier Foundation show that the FBI created a fake web page designed to look like a Seattle Times article, and used the page to spread tracking malware onto the suspect's computer. Creating dummy pages is a common way to spread malware — typically known as spoofing — but it's more common among criminals than law enforcement, and many are already interpreting the fake page as an attack on the press. "We are outraged that the FBI, with the apparent assistance of the U.S. Attorney's Office, misappropriated the name of *The Seattle Times*," a *Times* editor told the paper. "Not only does that cross a line, it erases it." The Associated Press echoed the concern, saying, "this ploy violated AP's name and undermined AP's credibility."

Crypto: Passwords and RNGs

CS 642

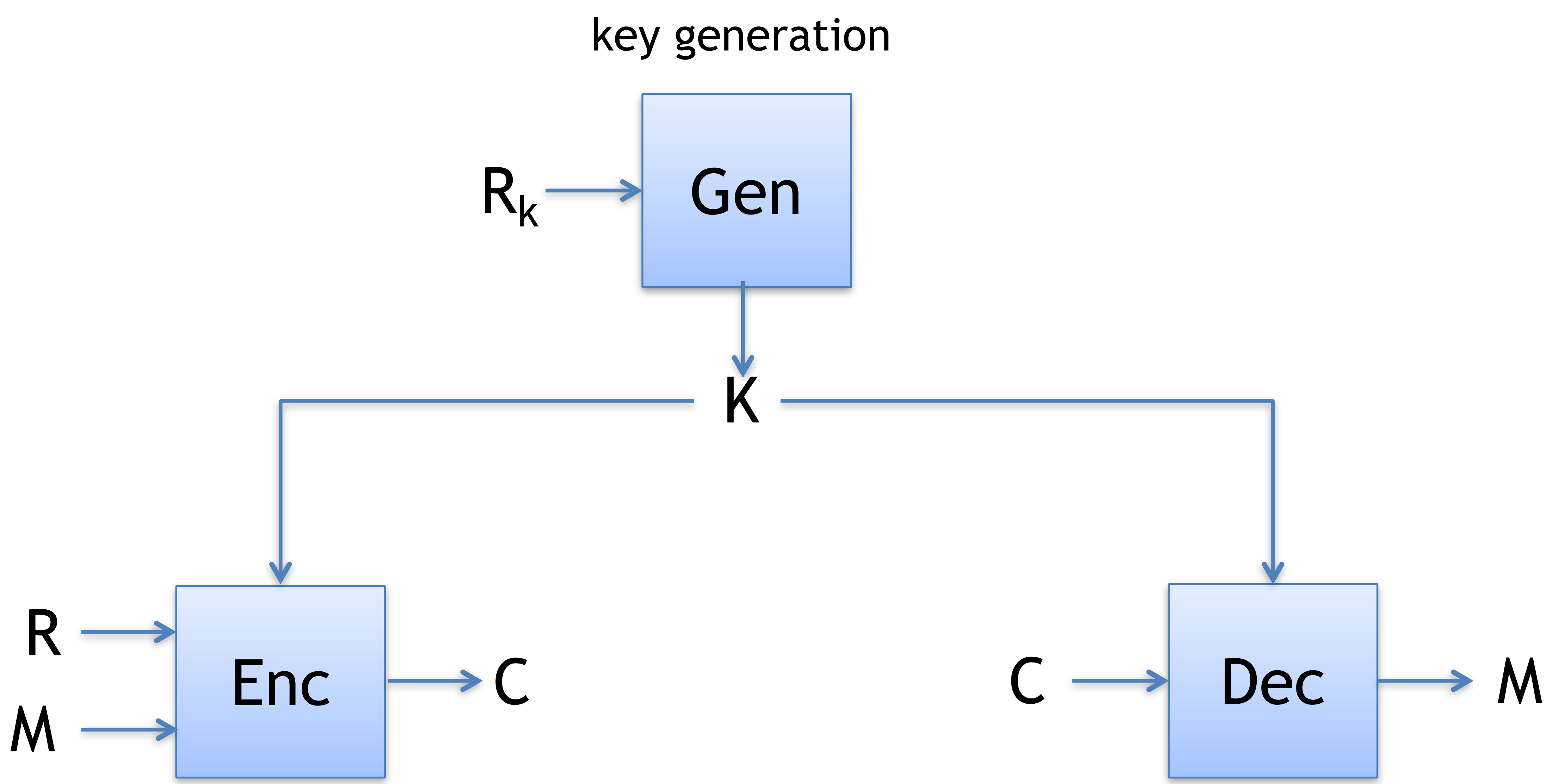
Guest Lecturer: Adam Everspaugh
http://pages.cs.wisc.edu/~ace

Topics

Password-based Crypto

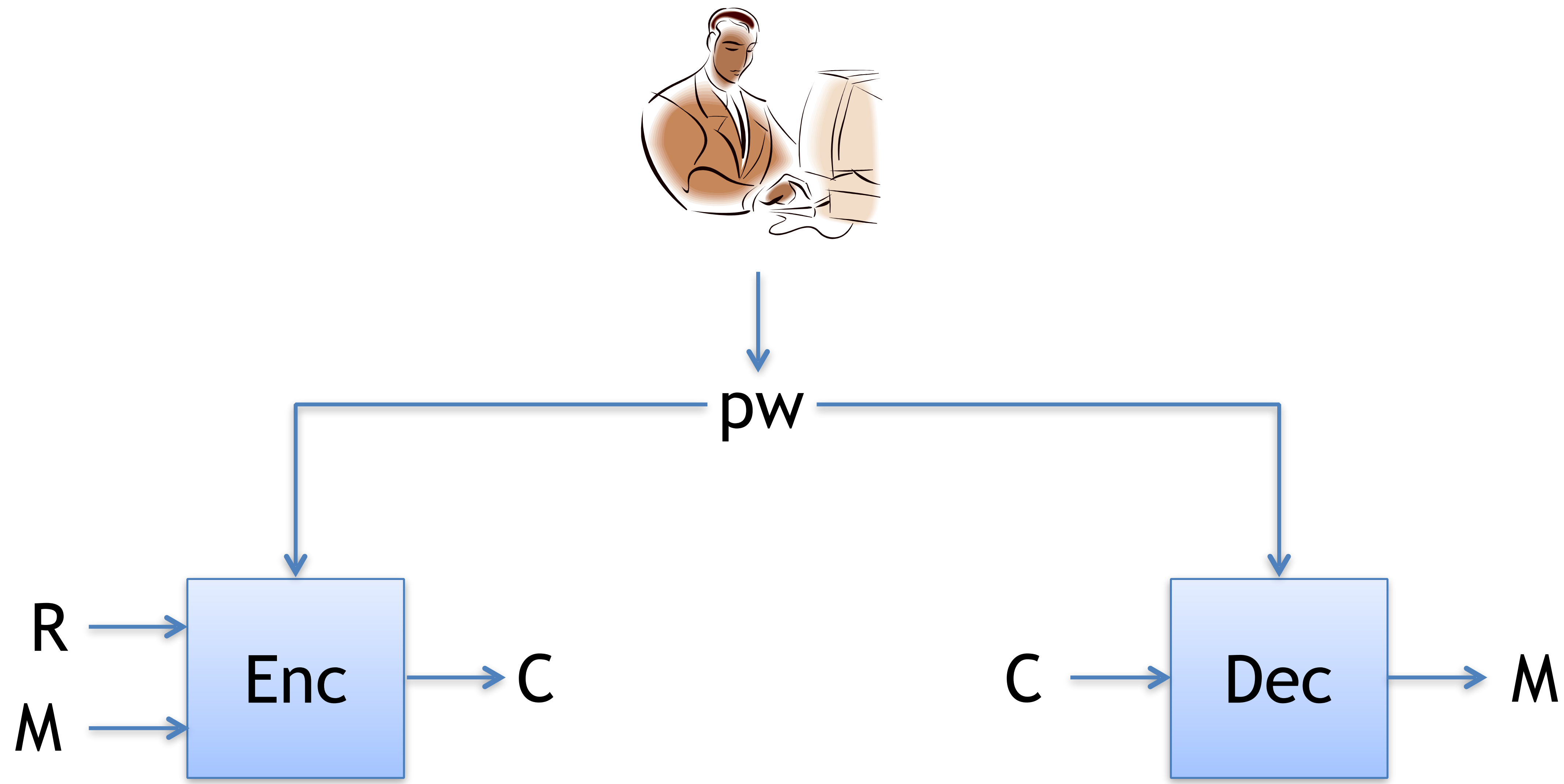
Random Number Generators

Symmetric Key Encryption



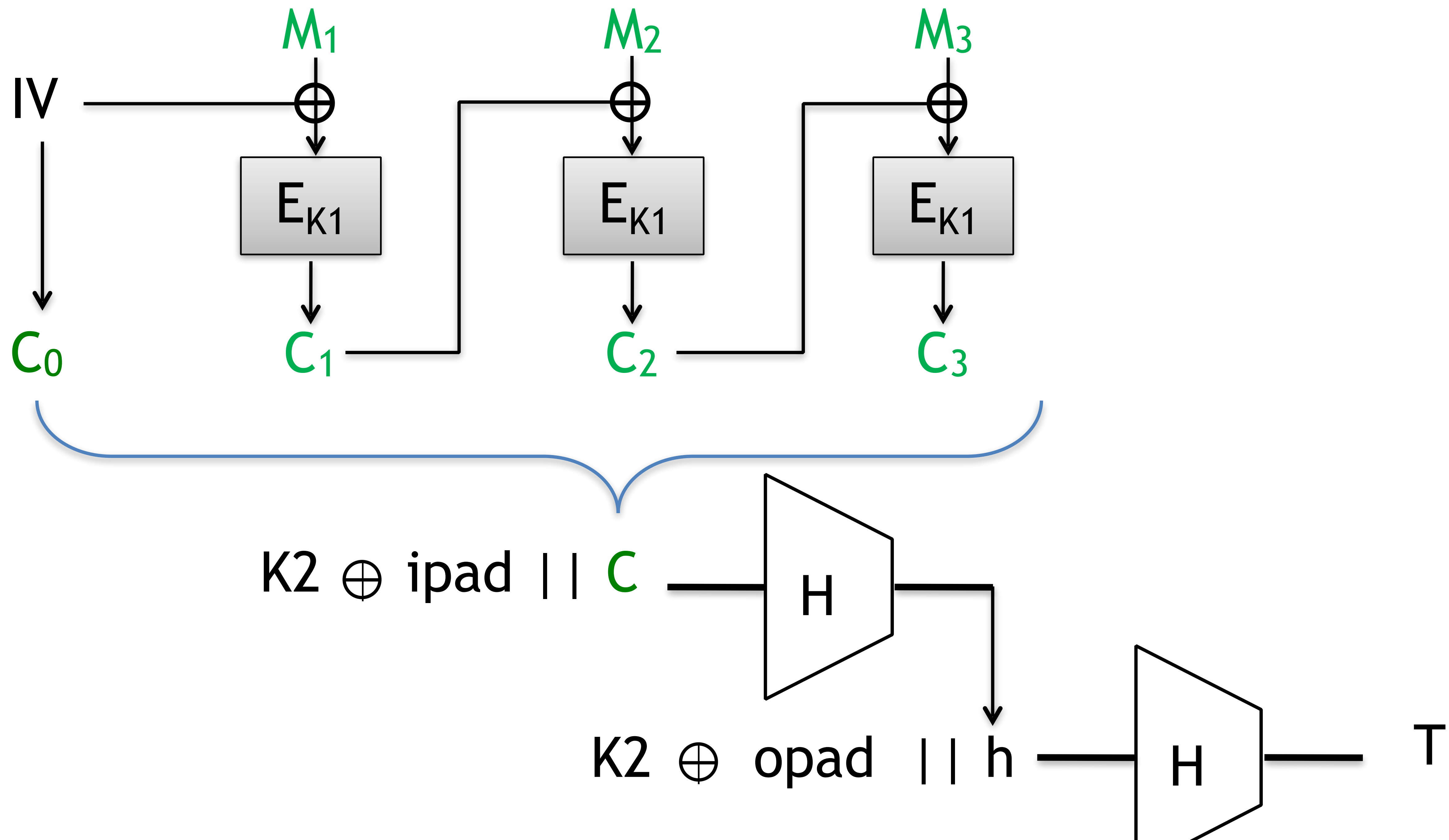
Correctness: $D_k(E_k(M,R)) = M$

Password-based Symmetric Encryption



Correctness: D(pw, E(pw,M,R)) = M

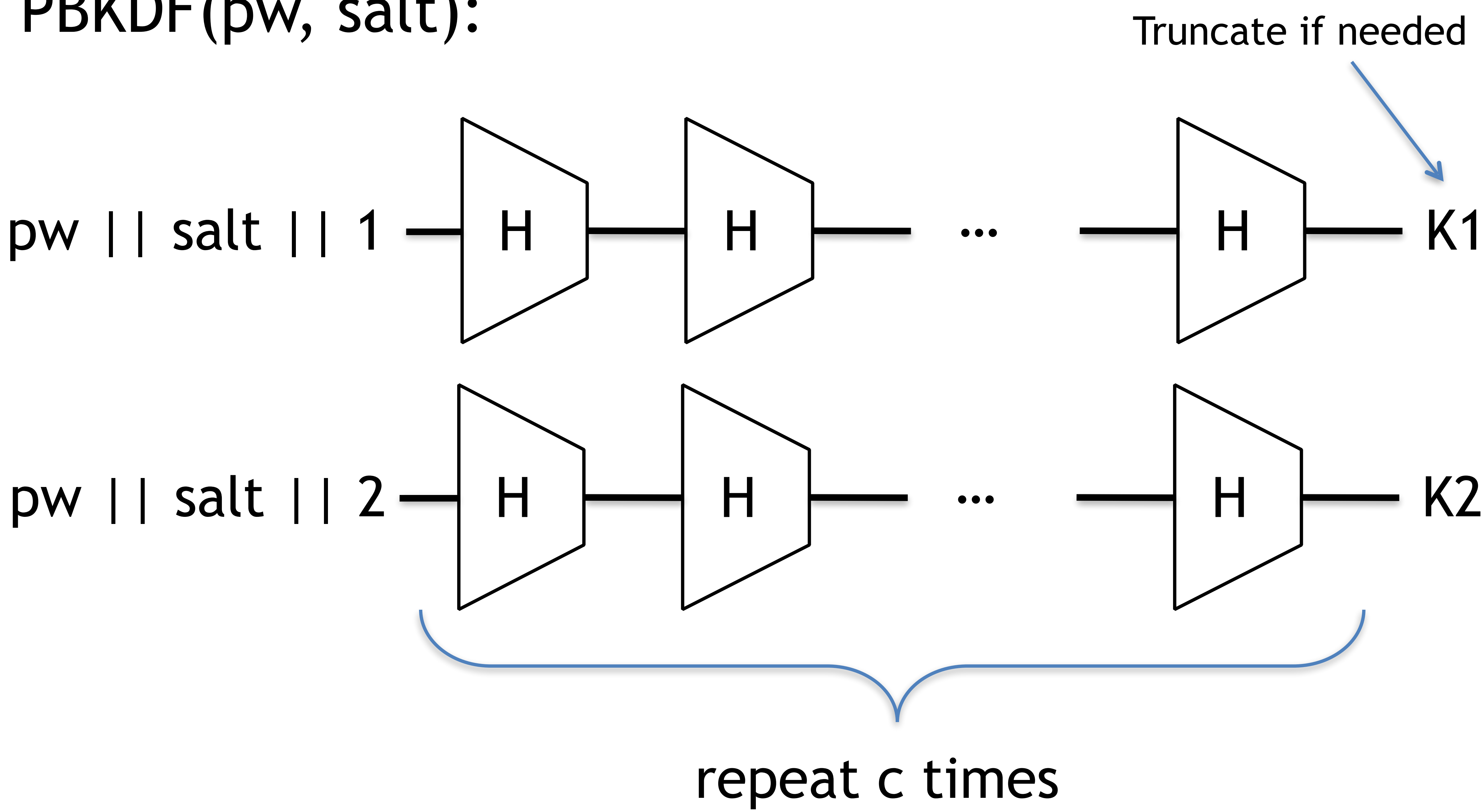
Encrypt-then-MAC with CBC and HMAC



How do we use this with a password?

Password-based Key Derivation (PBKDF)

PBKDF(pw, salt):



PBKDF + Symmetric Encryption yields PW-Based Encryption

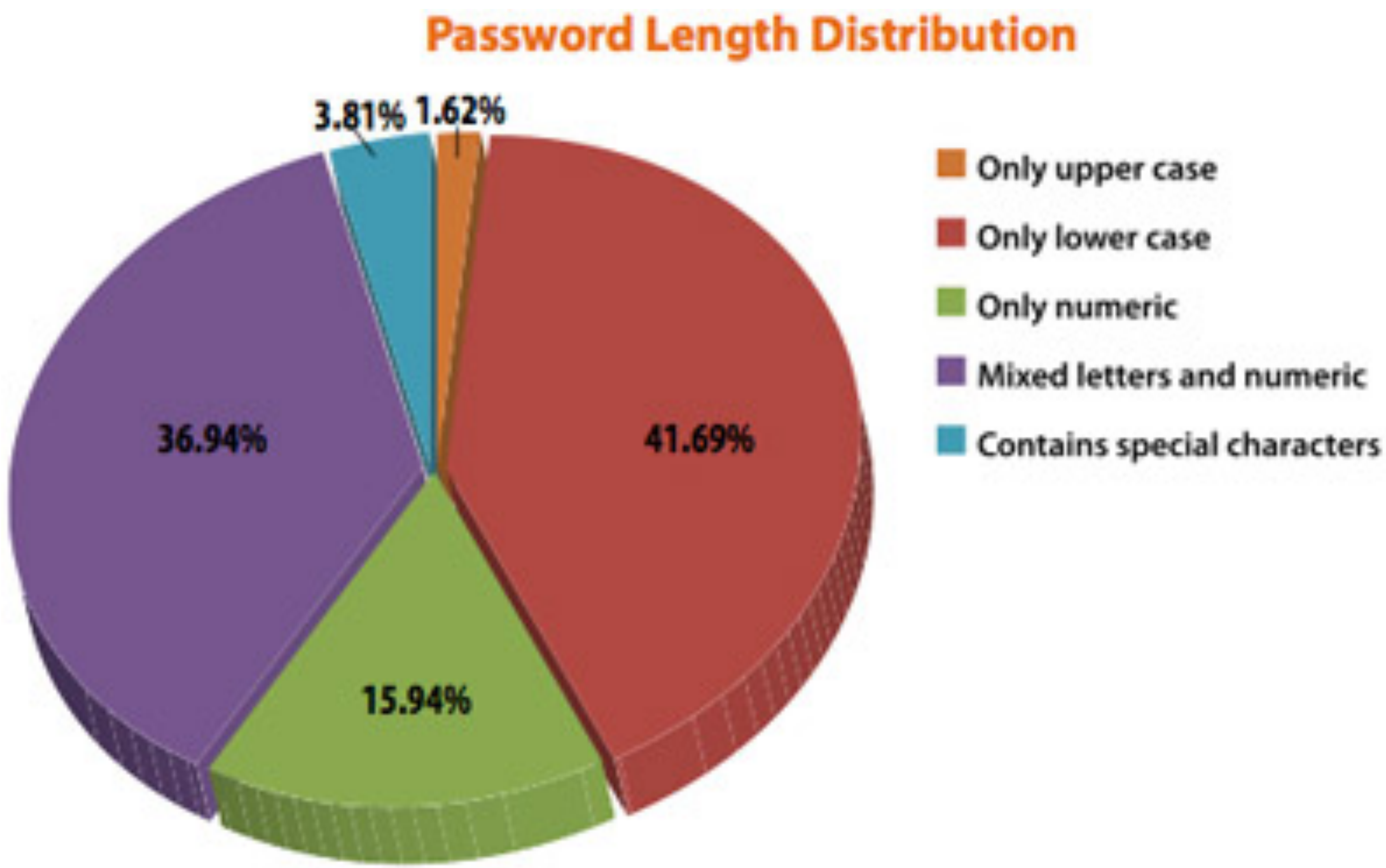
Enc(pw,M,R):
salt || R' = R
K = PBKDF(pw,salt)
C = Enc'(K,M,R')
Return (salt,C)

Here Enc'/Dec' is a typical symmetric encryption scheme (CBC+HMAC)

Dec(pw,C):
salt || C' = C
K = PBKDF(pw,salt)
M = Dec'(K,C')
Return M

Attacks?

Password Distribution

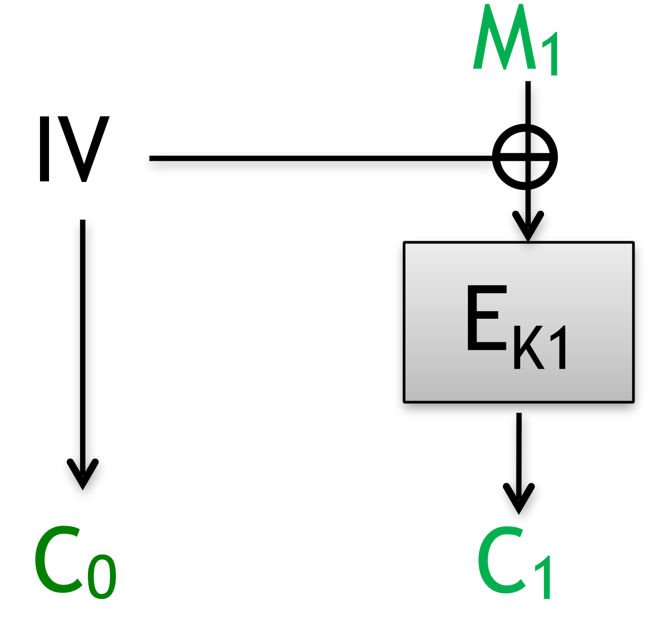


From an Imperva study of released RockMe.com password database (2010)

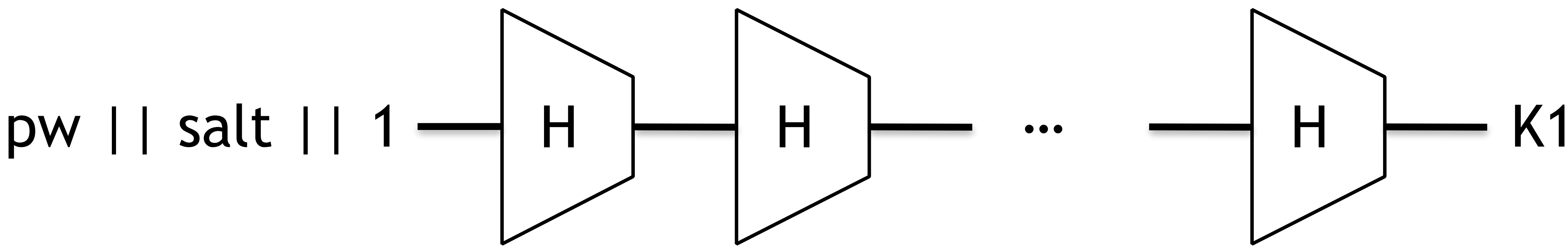
Dictionary Attack

- Given a (message,ciphertext) pair:
- Enumerate a dictionary D of possible passwords, in order of likelihood
- Test each candidate password

```
DictionaryAttack(D,M,C):
R || C' = C
for pw* in D:
   C* = Enc(pw*,M,R)
   if C* == C':
      return pw*
```



PBKDF Slows Down Dictionary Attacks



Iterating c times should slow down attacks by factor of c

Salts:

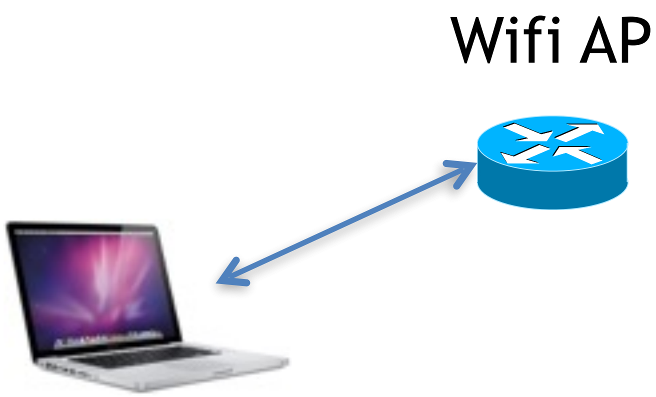
- Different derived keys, even if same password
- Slows down attacks against multiple users
- Prevents precomputation attacks, if salts chosen randomly

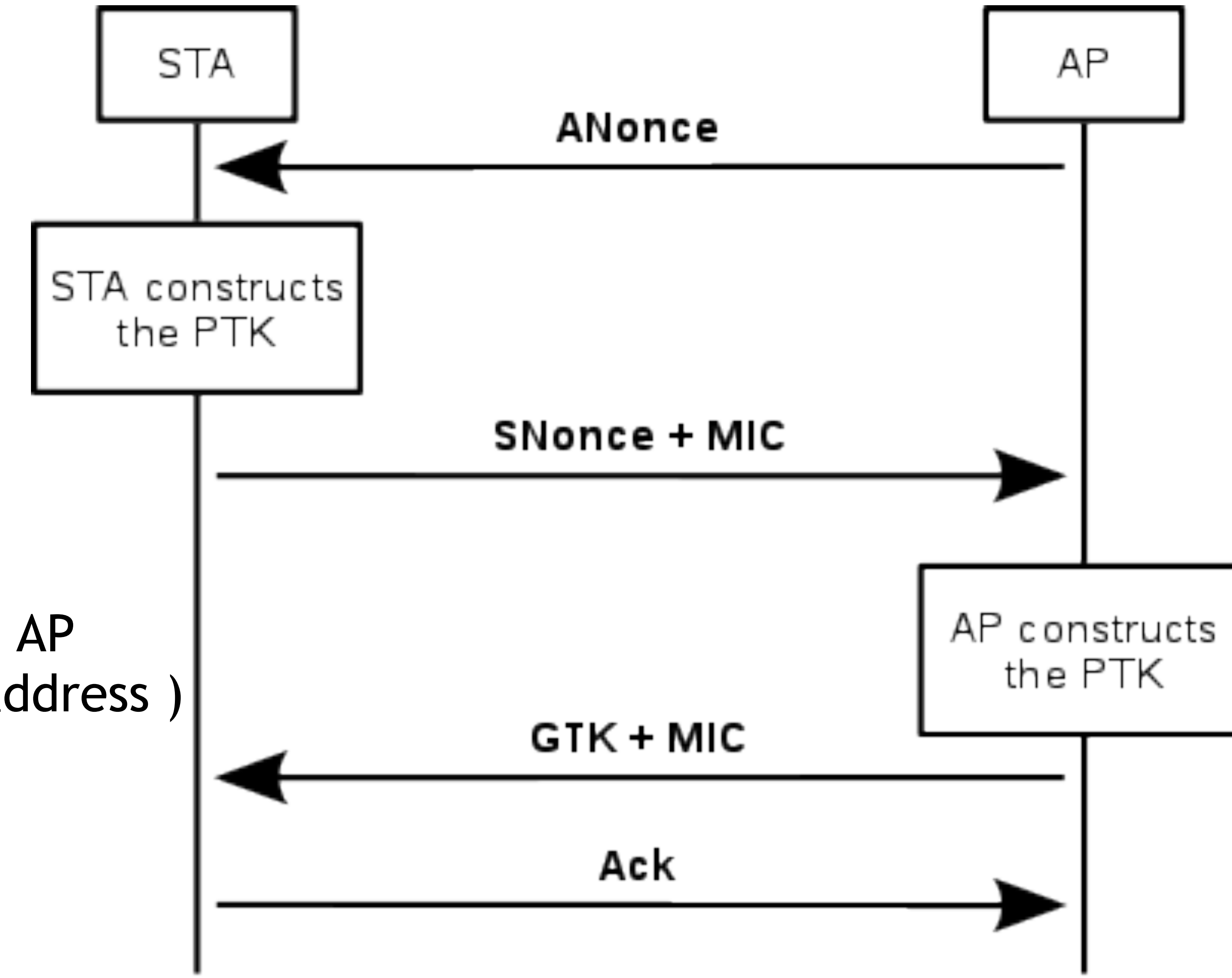
How Fast Are Dictionary Attacks?

- openssl speed sha1
- Assume: 4 cores @ 2.2M hashes per second

	Size of Dictionary	Computation time c=1	Computation time c=4096
6 digit PIN	10	0.11 seconds	7.8 minutes
6 alphanumerics (lowercase)	36	4.1 minutes	11.7 days
8 alphanumerics (mixed case)	62	287 days	3,222 years

802.11 WPA Authentication





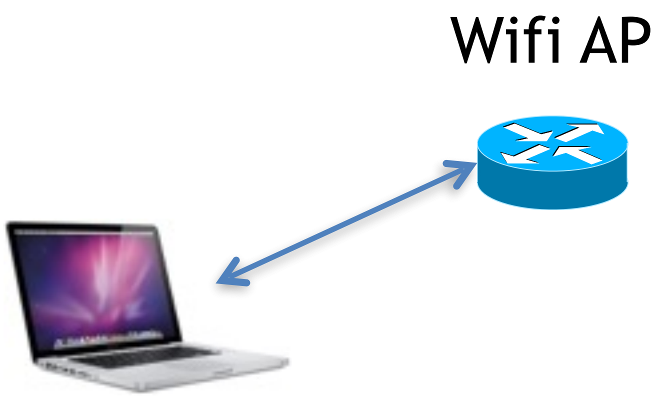
PMK = PBKDF(pw, ssid || ssidlength)
with c = 4096

PTK = H(PMK || ANonce || SNonce || AP MAC address || STA MAC address)

MIC = HMAC-MD5(PTK, M2)

Observe just one handshake by another party, and attacker can mount ***offline*** dictionary attack against the password

Attacking WPA Passwords



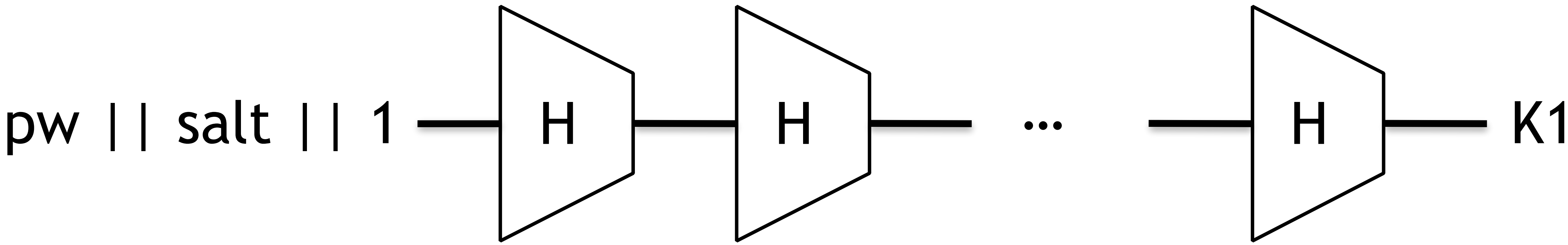
PMK = PBKDF(pw, ssid||ssidlength)
with c = 4096

PTK = H(PMK || ANonce || SNonce || AP MAC address || STA MAC address)

MIC = HMAC-MD5(PTK, M2)

```
DictionaryAttack(D,MIC,ANonce,SNonce,SSID,M2):
for pw* in D:
   PMK* = PBKDF(pw*, ssid||ssidlength)
   PTK* = H(PMK* || ANonce || … )
   MIC* = HMAC-MD5(PTK*, M2)
   If MIC* == MIC:
      return pw*
return None
```

Recap: Password-based Crypto



- Allows use of passwords in existing crypto schemes

- **Gain:**
  - Increases attackers computations
  - Prevents precomputation

- **Cost:**
  - Increased computation

- **Limitation:**
  - Strength of key still limited to strength of password
  - Don’t make it easy for attacker to mount offline dictionary attacks

INTERMISSION

Uses for Secure Random Numbers

Cryptography

- Keys
- Nonces, initial values (IVs), salts

System Security

- TCP Initial Sequence Numbers (ISNs)
- ASLR
- Stack Canaries

Where can we get secure random numbers?

OSX/Linux

- `cat /dev/urandom`
- `xxd -l 1024 -p /dev/urandom`
- `openssl rand 256 -hex`

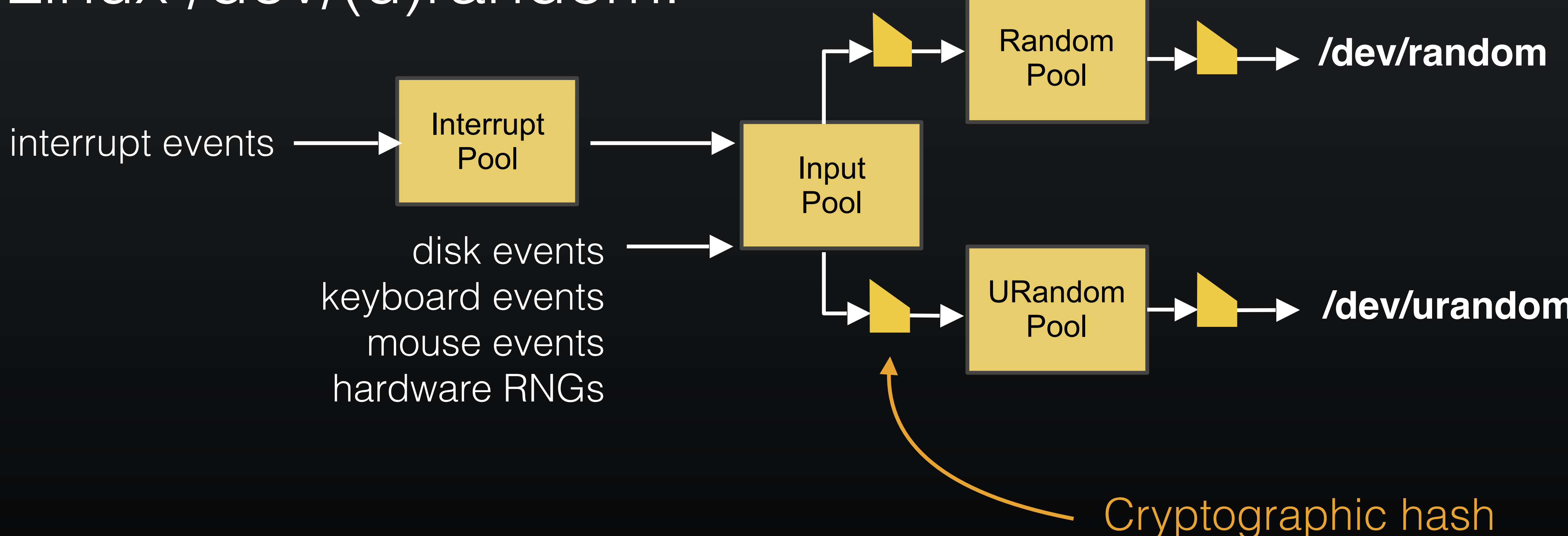
Intel HW RNG

- **OSX:** `sysctl -a | grep RDRAND`
- **Linux:** `cat /proc/cpuinfo | grep rdrand`

Operating System Random Number Generators

Linux RNG

Linux /dev/(u)random:

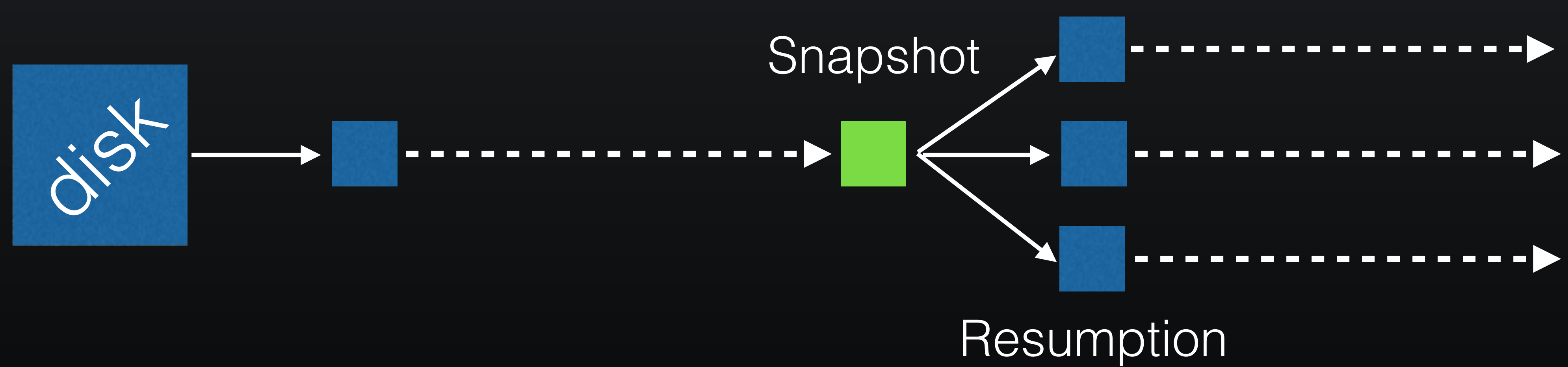


RNG Failures

RNG Failures
Predictable Output
Repeated Output
Outputs from a small range (not-statistically uniform)

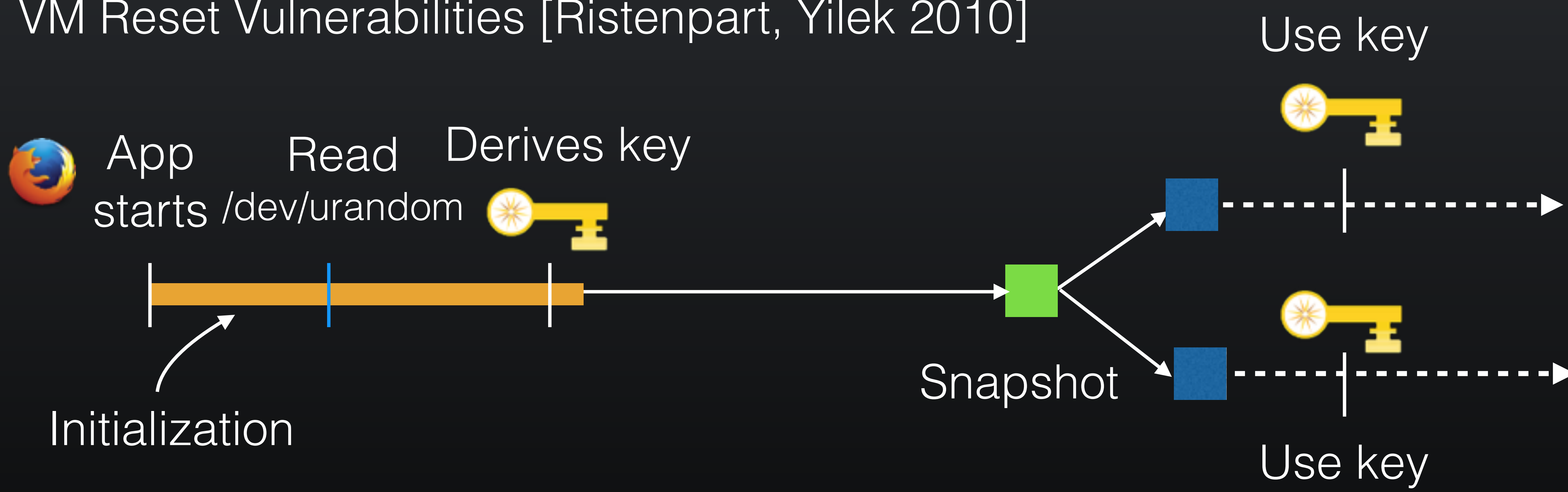
Broken Windows RNG: [DGP 2007]
Broken Linux RNG: [GPR 2008], [LRSV 2012], [DPRVW 2013], [EZJSR 2014]
Factorable RSA Keys: [HDWH 2012]
Taiwan National IDs: [BCCHLS 2013]

Virtual Machine Snapshots



Security Problems with VM Resets

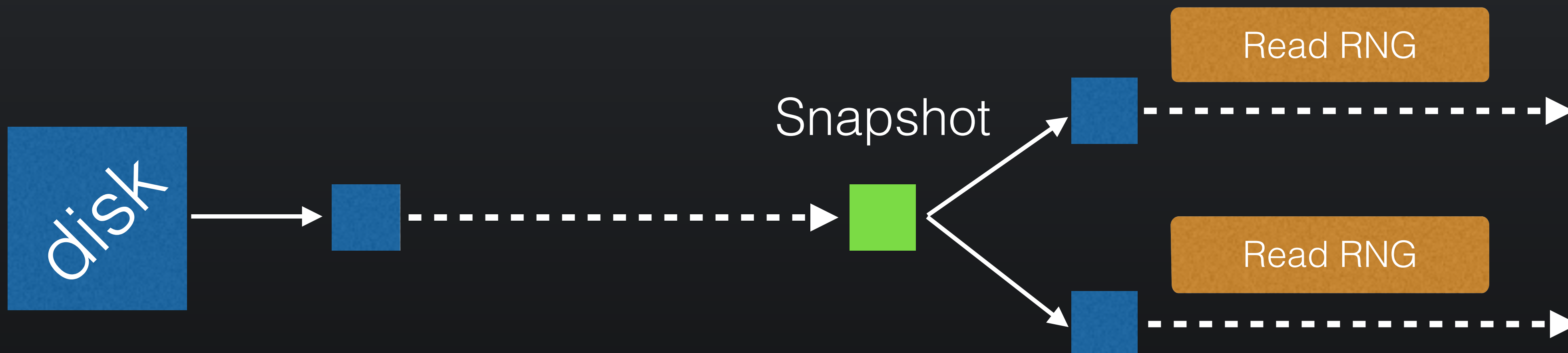
VM Reset Vulnerabilities [Ristenpart, Yilek 2010]



Firefox and Apache reused random values for TLS
Attacker can read previous TLS sessions, recover private keys from Apache

Linux RNG after VM Reset

Not-So-Random Numbers in Virtualized Linux
[Everspaugh, et al, 2014]



Experiment:

- Boot VM in Xen or VMware
- Capture snapshot
- Resume from snapshot, read from /dev/urandom

Repeat: 8 distinct snapshots
20 resumptions/snapshot

/dev/urandom outputs after resumption

Linux RNG is ***not*** reset secure:
7/8 snapshots produce mostly identical outputs

1E6DD331	1E6DD331	1E6DD331
8CC97112	8CC97112	8CC97112
2A2FA7DB	2A2FA7DB	2A2FA7DB
DBBF058C	DBBF058C	DBBF058C
26C334E7	26C334E7	26C334E7
F17D2D20	F17D2D20	45C78AE0
CC10232E	CC10232E	E678DBB2
...	...	...
Reset 1	Reset 2	Reset 3

Reset insecurity and applications

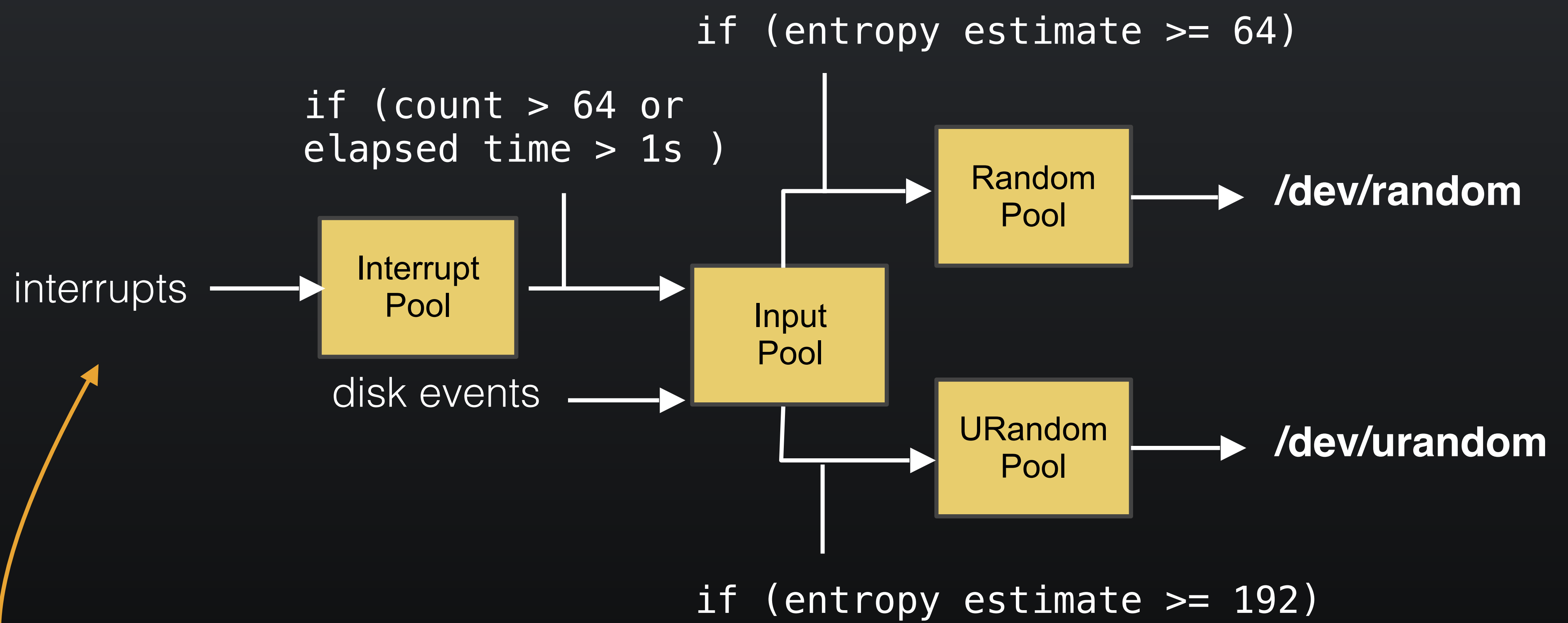
Generate RSA key on resumption:

```
openssl genrsa
```

30 snapshots; 2 resets/snapshot (ASLR Off)

- 27 trials produced identical private keys
- 3 trials produced unique private keys

Why does this happen?



Buffering and thresholds prevent new inputs from impacting outputs

Linux /dev/(u)random

What about 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 Recap

- **RNGs are critical for security**
  - Keys, nonces, etc

- **Building good RNGs is hard**
- **OS provides a strong RNG**
  - e.g.: /dev/urandom
- **Intel CPUs provide an RNG**
  - RDRAND instructions