Diffie-Hellman, Side-channels, RNGs

CS642: Computer Security

Professor Ristenpart

http://www.cs.wisc.edu/~rist/

rist at cs dot wisc dot edu

Diffie-Hellman math

Let p be a large prime number
Fix the group G = $\mathbf{Z}_p^*$ = {1,2,3,…, p-1}

Then G is *cyclic*. This means one can give a member $g \in G$, called the generator, such that

$$G = \{ g^0, g^1, g^2, \ldots, g^{p-1} \}$$

Example: p = 7. Is 2 or 3 a generator for $\mathbf{Z}_7^*$?

x	0	1	2	3	4	5	6
2^x mod 7	1	2	4	1	2	4	1
3^x mod 7	1	3	2	6	4	5	1

Textbook exponentiation

Let G be cyclic group.
How do we compute h^x for any $h \in G$?

```
ModExp(h,x)
X' = h
For i = 2 to x  do
     X' = X'*h
Return X'
```

Requires time O(|G|) in worst case.

```
SqrAndMulExp(h,x)
b_k,…,b_0 = x
f = 1
For i = k down to 0 do
     f = f^2  mod N
     If b_i = 1 then
          f = f*h
Return f
```

Requires time O(k) multiplies and squares in worst case.

```
SqrAndMulExp(h,x)
b_k,…,b_0 = x
f = 1
For i = k down to 0 do
    f = f^2  mod N
    If b_i = 1 then
        f = f*h
Return f
```

$$x = \sum_{b_i \neq 0} 2^i$$

$$h^x = h^{\sum_{b_i \neq 0} 2^i} = \prod_{b_i \neq 0} h^{2^i}$$

$$h^{11} = h^{8+2+1} = h^8 \bullet h^2 \bullet h$$

$b_3 = 1$ $f_3 = 1 \bullet h$

$b_2 = 0$ $f_2 = h^2$

$b_1 = 1$ $f_1 = (h^2)^2 \bullet h$

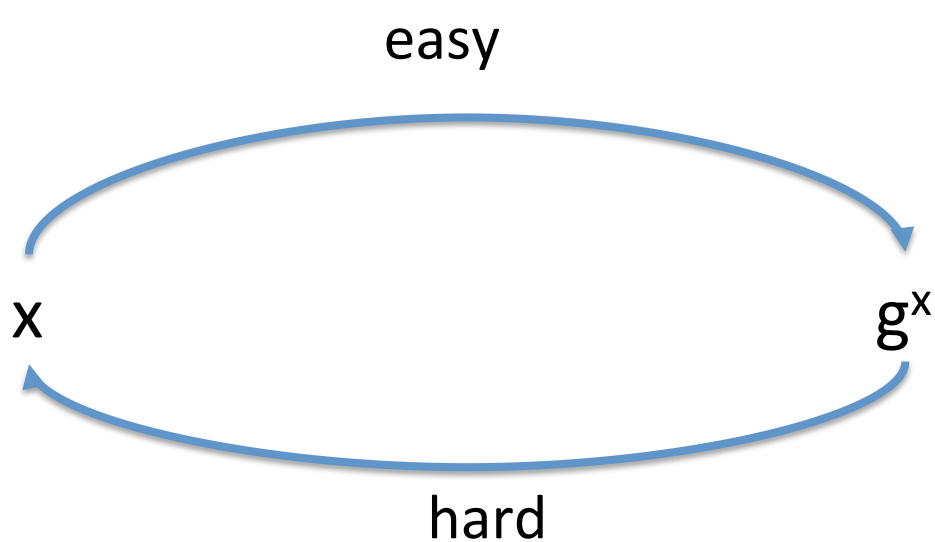
$b_1 = 1$ $f_0 = (h^4 \bullet h)^2 \bullet h = h^8 \bullet h^2 \bullet h$

The discrete log problem

Fix a cyclic group G with generator g

Pick x at random from $\mathbf{Z}_{|G|}$

Give adversary g, $X = g^x$. Adversary's goal is to compute x



The discrete log problem

Fix a cyclic group G with generator g

Pick x at random from $\mathbf{Z}_{|G|}$

Give adversary g, $X = g^x$. Adversary's goal is to compute x

```
𝒜(X):
for i = 2 , … ,  |G|-1 do
     if X  = g^i then
          Return i
```

Very slow for large groups!
$\mathcal{O}(|G|)$

Baby-step giant-step is better:
$\mathcal{O}(|G|^{0.5})$
Nothing faster is known for some groups.

Diffie-Hellman Key Exchange

X

Y

Pick random x from $\mathbf{Z}_{|G|}$
$X = g^x$

Pick random y from $\mathbf{Z}_{|G|}$
$Y = g^y$

$K = H(Y^x)$

$K = H(X^y)$

Get the same key. Why? $Y^x = g^{yx} = g^{xy} = X^y$

What type of security does this protocol provide?

Computational Diffie-Hellman Problem

Fix a cyclic group G with generator g

Pick x,y both at random $\mathbf{Z}_{|G|}$

Give adversary g, $X = g^x$, $Y = g^y$.
Adversary must compute g^{xy}

For most groups, best known algorithm finds discrete log of X or Y.
But we have no proof that this is best approach.

Client

TLS handshake for Diffie-Hellman Key Exchange

Server

Pick random Nc — ClientHello, MaxVer, Nc, Ciphers/CompMethods →

Pick random Ns — ← ServerHello, Ver, Ns, SessionID, Cipher/CompMethod

← CERT = (pk_s , signature over it)

Pick random x

$X = g^x$

← p , g , X , σ = Sign(sk_s, p || g || X)

Check CERT using CA public verification key

Check σ

Pick random y

$Y = g^y$

Y →

PMS = g^{xy} (client)

PMS = g^{xy} (server)

ChangeCipherSpec,
{ Finished, PRF(MS, "Client finished" || H(transcript)) } →

ChangeCipherSpec,
{ Finished, PRF(MS, "Server finished" || H(transcript')) } ←

Bracket notation means contents encrypted

MS <- PRF(PMS, "master secret" || Nc || Ns)

Side-channel attacks

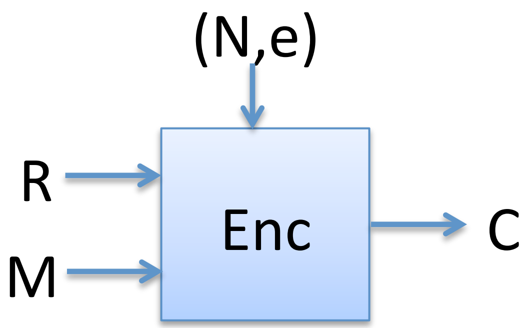
- Implementations might leak information about secret internal state via side-channels:
  - power consumption
  - Electromagnetic emanations (Tempest)
  - timing
  - Shared physical resources (CPU cache)

PKCS #1 RSA encryption

Kg outputs (N,e),(N,d)   where $|N|_8 = n$

Let $B = \{0,1\}^8 / \{00\}$ be set of all bytes except 00

Want to encrypt messages of length $|M|_8 = m$

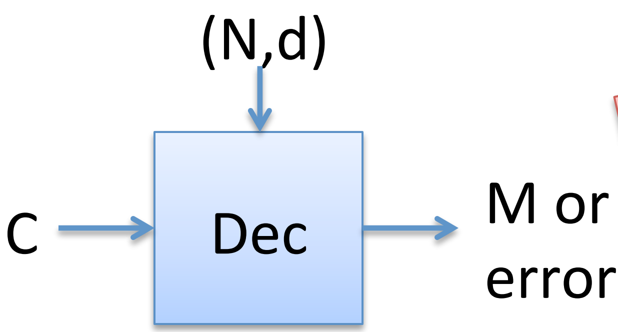


<u>Enc((N,e), M, R)</u>

pad = first n - m - 2 bytes from R that are in B

Y = 00 || 02 || pad || 00 || M

Return Y^e mod N



<u>Dec((N,d), C)</u>

$Y = C^d \bmod N$; aa||bb||w = Y

If (aa ≠ 00) or (bb ≠ 02) or (00 ∉ w)

    Return error

pad || 00 || M = w

Return M

SqrAndMulExp(C,d,N)

$b_k,\ldots,b_0 = d$

f = 1

For i = k down to 0 do

    $f = f^2 \mod N$ S

    If $b_i = 1$ then

        $f = f^*C \mod N$ M

Return f

But:

Squaring and multiplying take different amounts of:

1) power
2) time
3) instruction cache sets

S M S S M S M d = ?

1 0 1 1 d = 11

<u>SqrAndMulExp(X,e,N)</u>

$b_k,\ldots,b_0 = e$
$f = 1$
For $i = k$ down to 0 do
    $f = f^2 \mod N$
    If $b_i = 1$ then
        $f = f{*}X \mod N$
Return f

But:
Squaring and multiplying take different amounts of:
1) power
2) time
3) instruction cache sets

From Messerges et al. 1999:

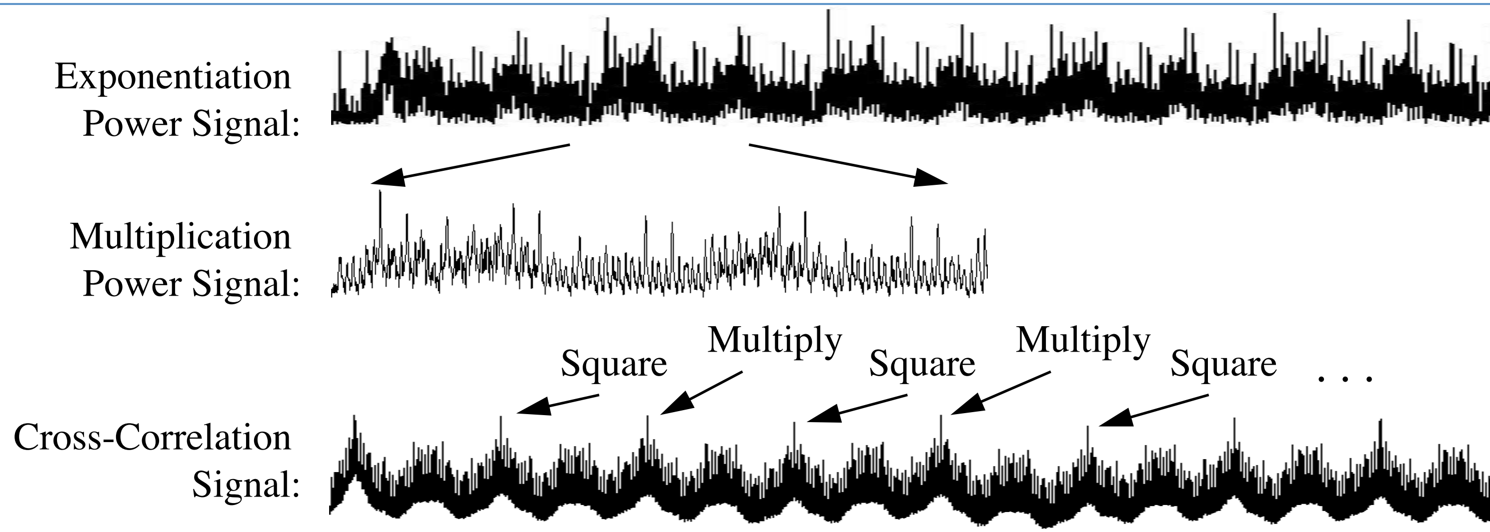


Fig. 2. Cross-Correlation of Multiplication and Exponentiation Power Signals
The above signals were obtained using the power analysis equipment described in Section 4.

```
SqrAndMulExp(X,e,N)
b_k,…,b_0 = e
f = 1
For i = k down to 0 do
    f = f^2  mod N
    If b_i = 1 then
        f = f*X mod N
Return f
```

But:
Squaring and multiplying take different amounts of:

1) power
2) time
3) instruction cache sets

Time:
Remote timing attacks against TLS (Boneh, Brumley 2003)
Chosen ciphertexts + timing = key extraction
~1 million queries (though highly variable)

```
SqrAndMulExp(X,e,N)
b_k,…,b_0 = e
f = 1
For i = k down to 0 do
    f = f^2  mod N
    If b_i = 1 then
        f = f*X mod N
Return f
```

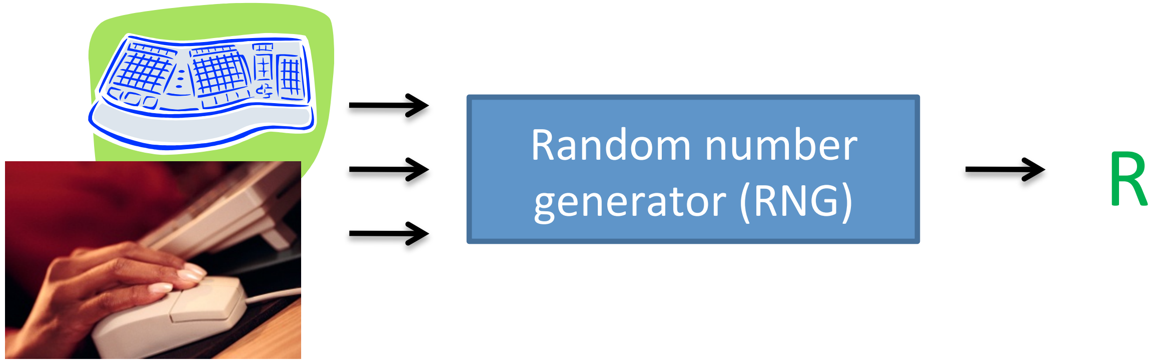
But:

Squaring and multiplying take different amounts of:

1) power
2) time
3) instruction cache sets

Instruction cache sets:

Attacker that shares I-cache with victim can infer which cache sets were used (by timing) and then correlate with squares or multiplies



- Random number generation
- Measure events on system, harvest entropy (unpredictability from them)
  - keyboard presses and timing
  - file/network interrupts
  - mouse movements
- Hash entropy down to “extract” (hopefully) uniform bit strings

→ Random number generator (RNG) → R

```
MD_Update(&m,buf,j);
....
MD_Update(&m,buf,j);   /* purify complains */
```

These lines of code commented out from OpenSSL random number generator code (md_rand.c) to **address complaints by security tools Purify and Valgrind**

Only the PID was used as input to RNG.

It took a ~2 years for the bug to be (publicly) discovered!

Ra

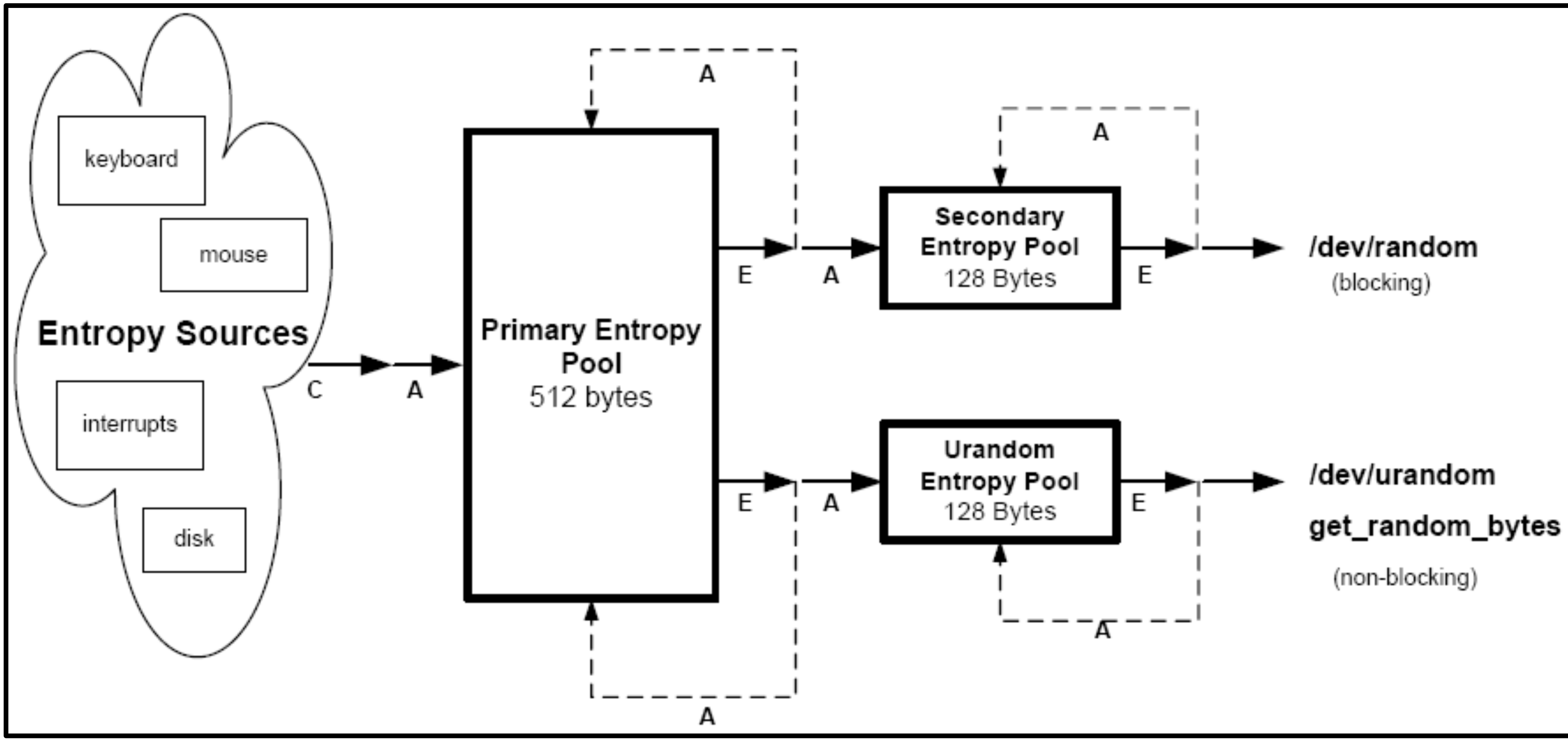
[W
[G
[G
[D
[W
[B
[M
[Abe
[Yilek et al. 2009]

Debian OpenSSL bug lead to small set of possible R

Linux /dev/random

Linux random number generator (2500 lines of undocumented code)

Diagram from [Gutterman, Pinkas, Reinman 2006]

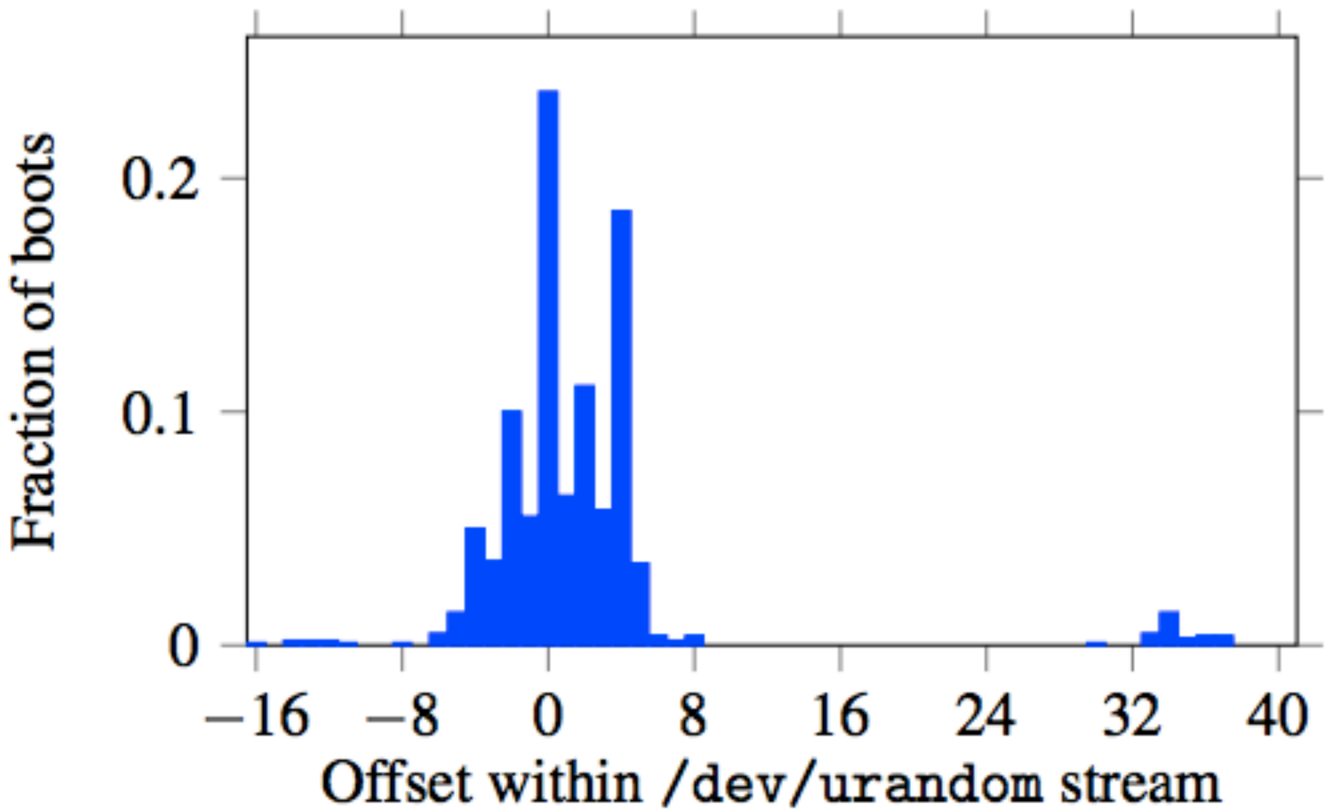


Applications like TLS take randomness from /dev/random

They then maintain an internal pool of random bits

(at least) two points of failure

Embedded systems with few entropy sources



[Heninger et al. 2012]: only entropy obtained by sshd is what offset into stream from /dev/random is used for key generation

Debian Bug Leaves Private SSL/SSH Keys Guessable

Posted by **timothy** on Tuesday May 13 2008, @12:01PM
from the security-is-a-process dept.

SecurityBob writes

> "Debian package maintainers tend to very often modify the source code of the package they are maintaining so that it better fits into the distribution itself. However, most of the time, their changes are not sent back to upstream for validation, which might cause some tension between upstream developers and Debian packagers. Today, a critical security advisory has been released: a Debian packager modified the source code of OpenSSL back in 2006 so as to remove the seeding of OpenSSL random number generator, which in turns makes cryptographic key material generated on a Debian system guessable. The solution? Upgrade OpenSSL and re-generate all your SSH and SSL keys. This problem not only affects Debian, but also all its derivatives, such as Ubuntu."

Reader RichiH also points to Debian's announcement and Ubuntu's announcement.

Virtual machines and secure browsing

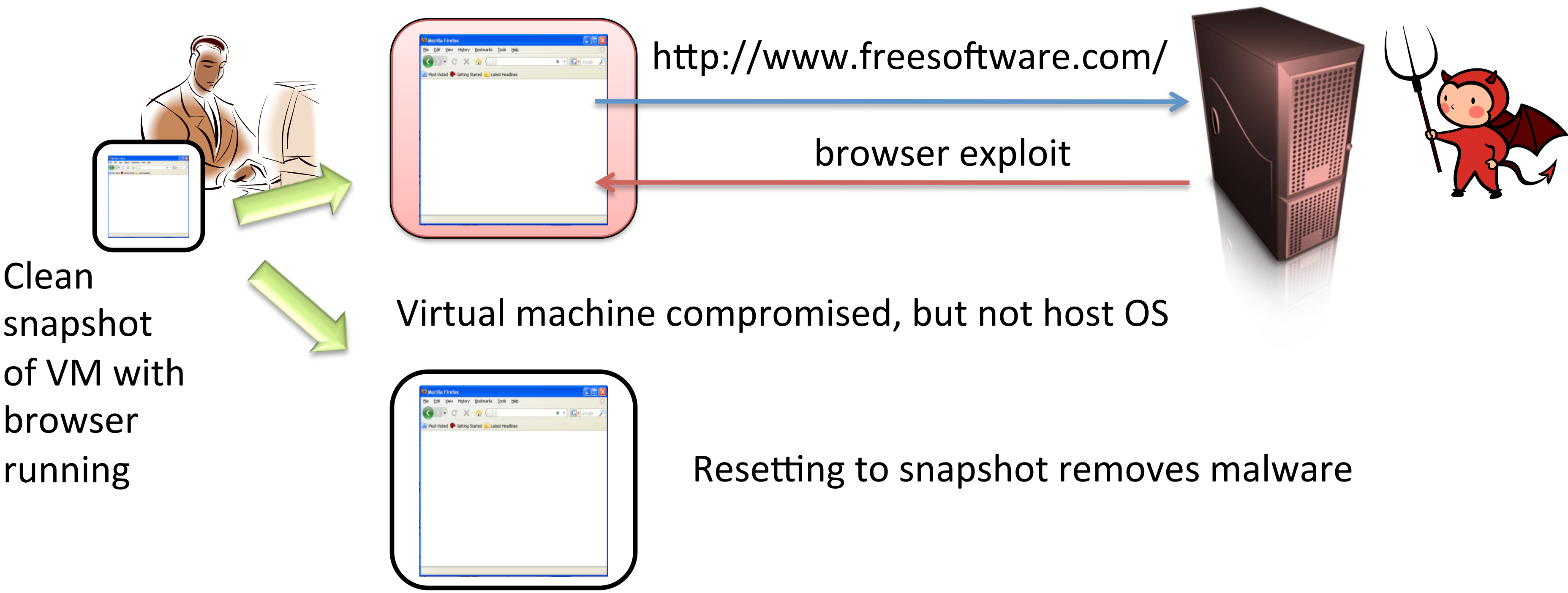
"**Protect Against Adware and Spyware:** Users protect their PCs against adware, spyware and other malware while browsing the Internet with Firefox in a virtual machine."
[http://www.vmware.com/company/news/releases/player.html]

vmware

"Your dad can do his [private] surfing on the virtual machine and can even set it to reset itself whenever the virtual computer is restarted, so there's no need to worry about leaving tracks. … I recommend VMware because you can download a free version of VMware Server for home use. "
[Rescorla, http://www.thestranger.com/seattle/SavageLove?oid=490850]

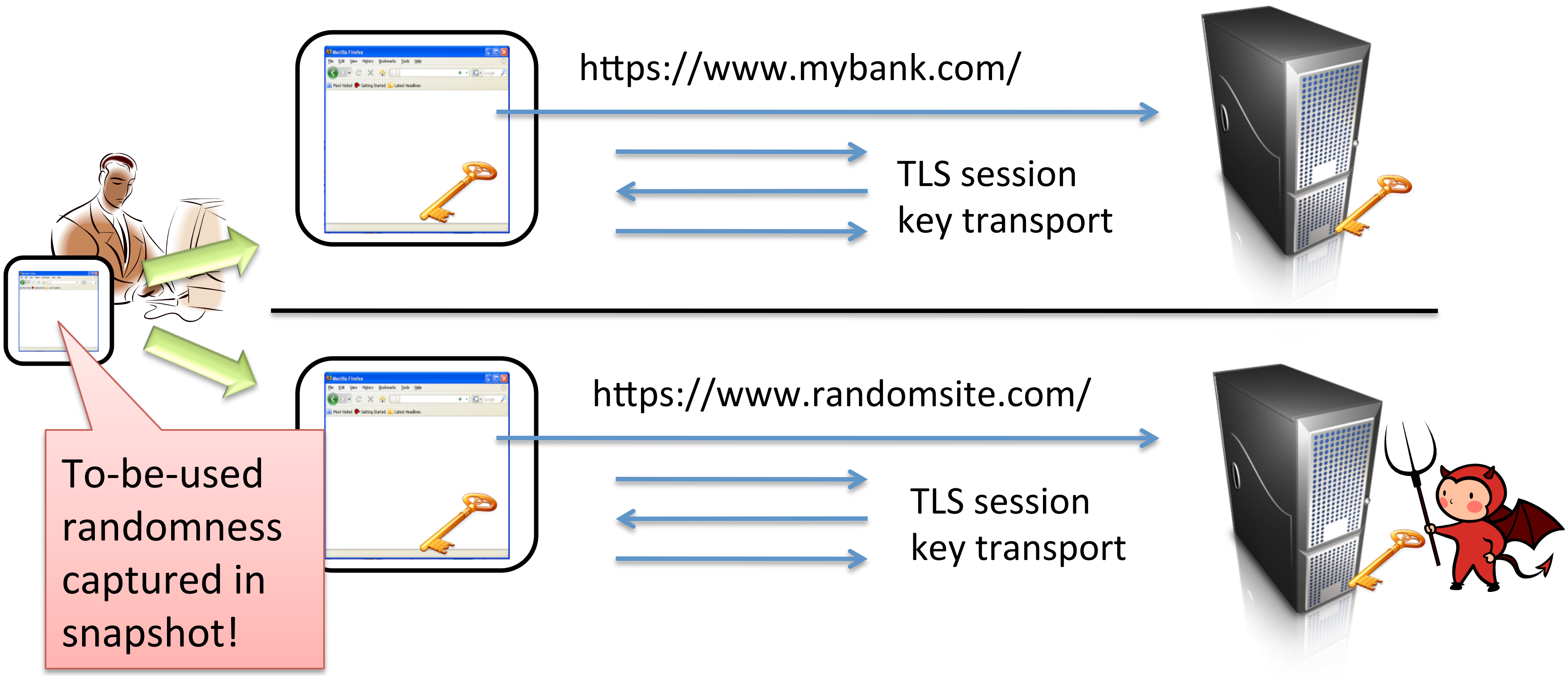
Virtual machines and secure browsing

"**Protect Against Adware and Spyware:** Users protect their PCs against adware, spyware and other malware while browsing the Internet with Firefox in a virtual machine."
[http://www.vmware.com/company/news/releases/player.html]



Virtual machine resets lead to RNG failures

[R., Yilek – NDSS ‘10]

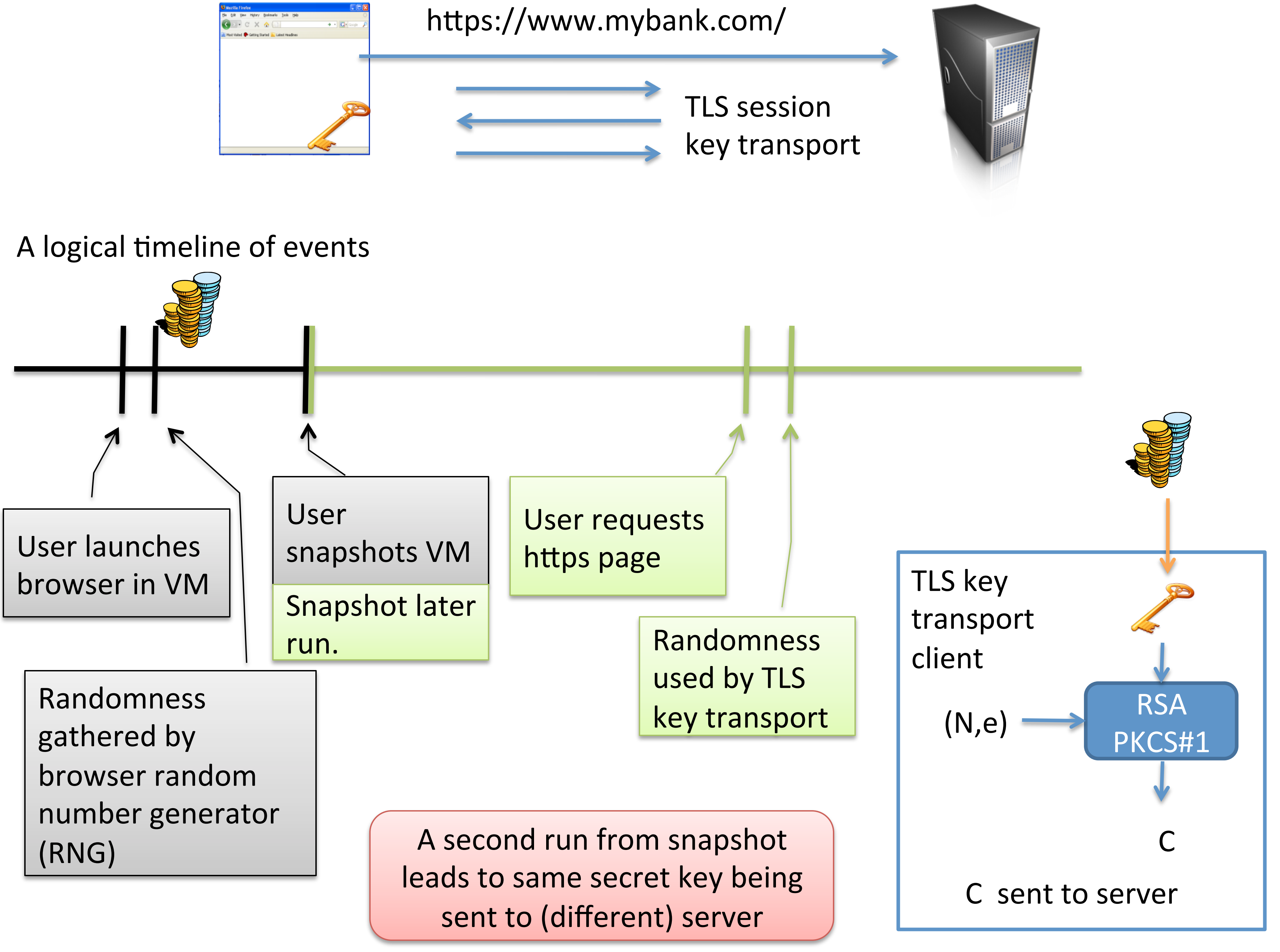


Recent versions of Firefox, Chrome allow session compromise attacks

Apache mod_ssl TLS server: server’s secret DSA key can be stolen!

[Everspaugh et al. 2014] similar problems face /dev/urandom usage

New Linux RNG design that fixes the problem



https://www.mybank.com/
TLS session
key transport
A logical timeline of events
User launches browser in VM
Randomness gathered by browser random number generator (RNG)
User snapshots VM
Snapshot later run.
User requests https page
Randomness used by TLS key transport
TLS key transport client
(N,e)
RSA PKCS#1
C
C sent to server
A second run from snapshot leads to same secret key being sent to (different) server

RNG recap

- Randomness is often a weak link in crypto implementations
- Building a good RNG is not easy
  - Do not roll your own
  - Must use cryptographically-strong RNG
- Intel RNG instructions in next generation chips

Lots of other implementation pitfalls (non-exhaustive list)

- Rolling your own cryptographic algorithms
  - KeeLoq attacks
- Using a same key with different algorithms
  - CBC and CTR mode with same key
  - RSA encryption and signing with same key pair
- Not erasing keys or private randomness from memory
- Traffic-analysis attacks
  - Lengths of CTR mode encryption of “Yes” vs. “No”