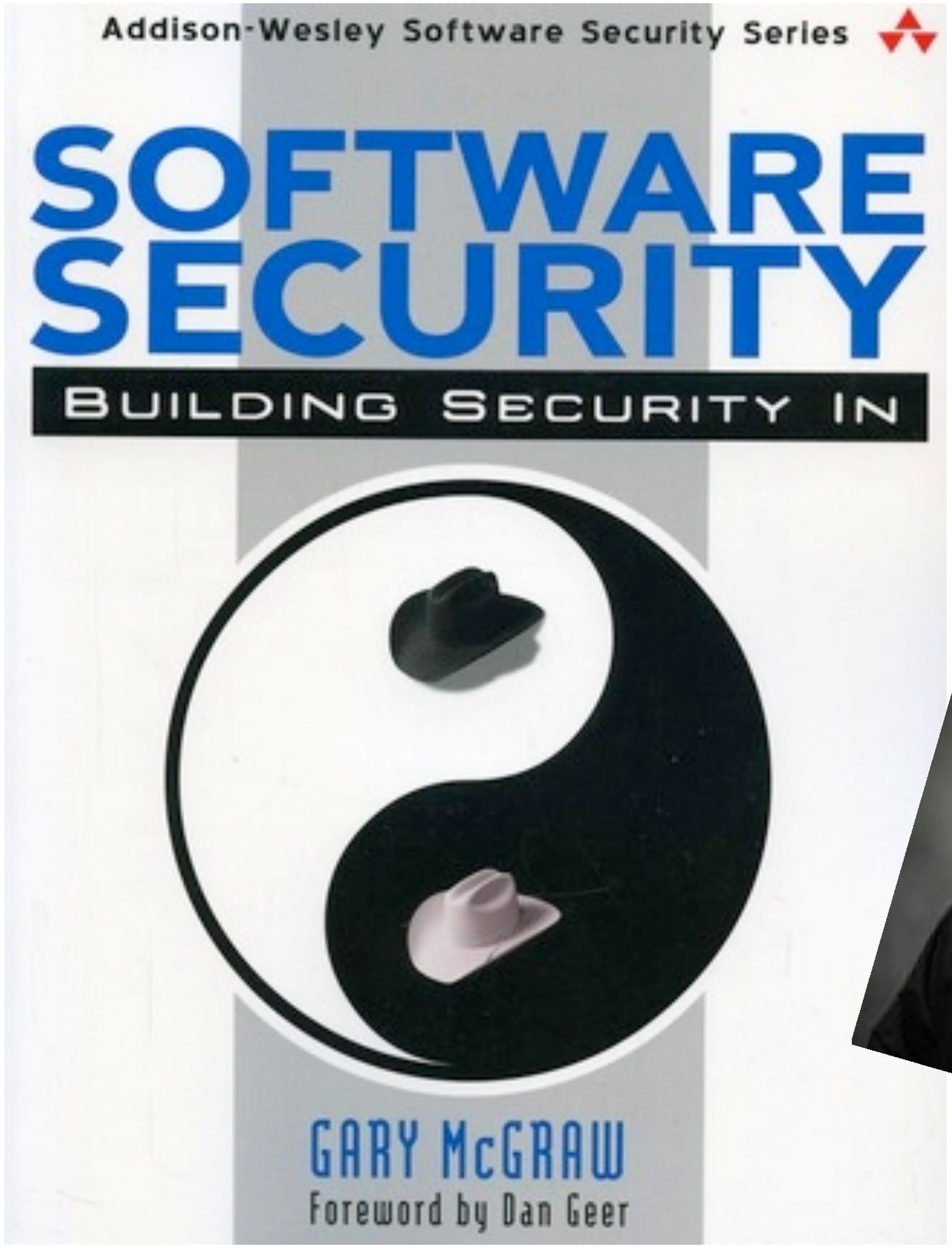
Notes from Prof. Brun

- Project plan due next Tuesday (email him if you have questions)
- Be ready to present project plans on Tuesday (10 minutes per group)

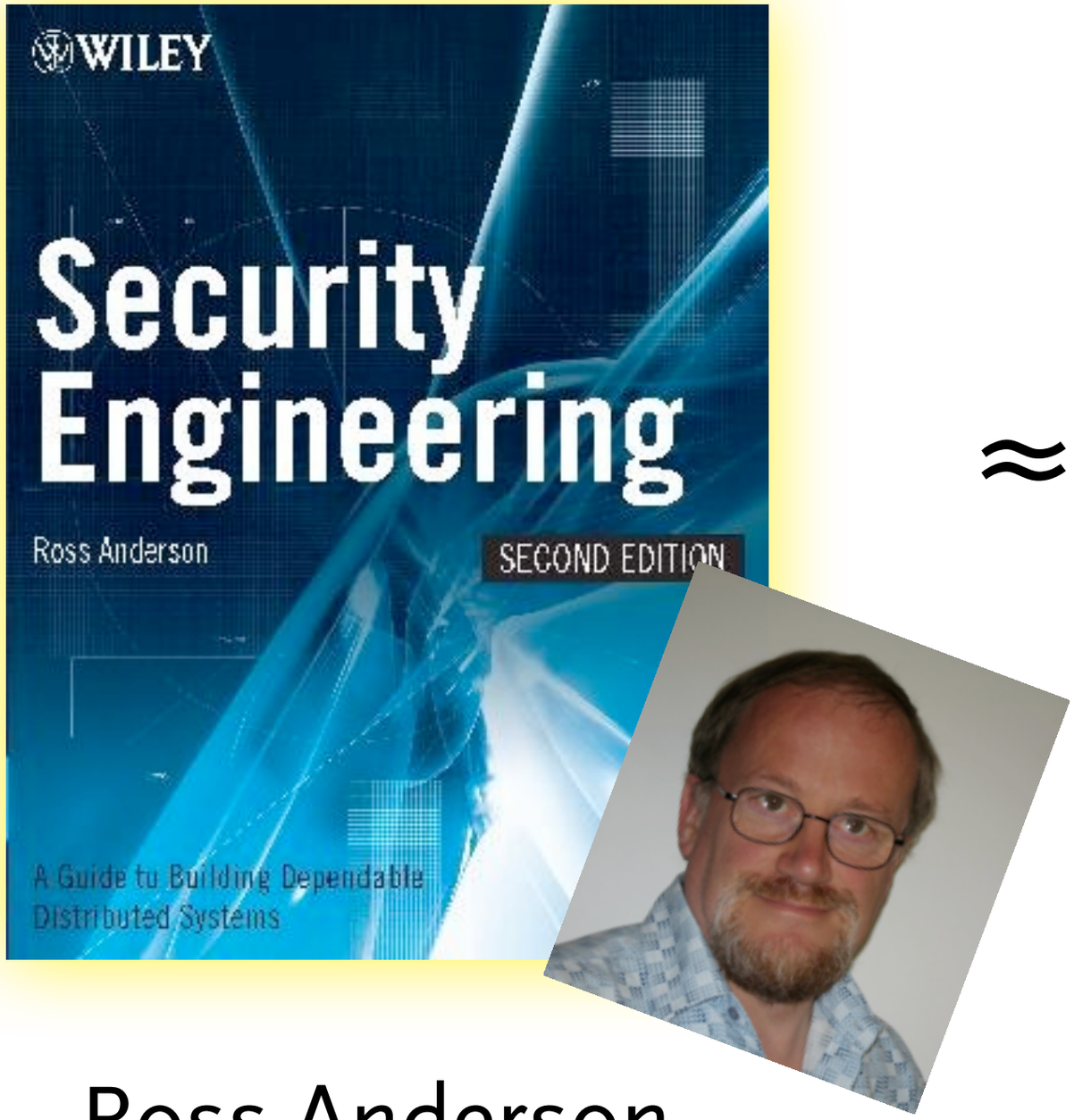
Software Security

Ben Ransford
ransford@cs.umass.edu

CS621 Fall 2012

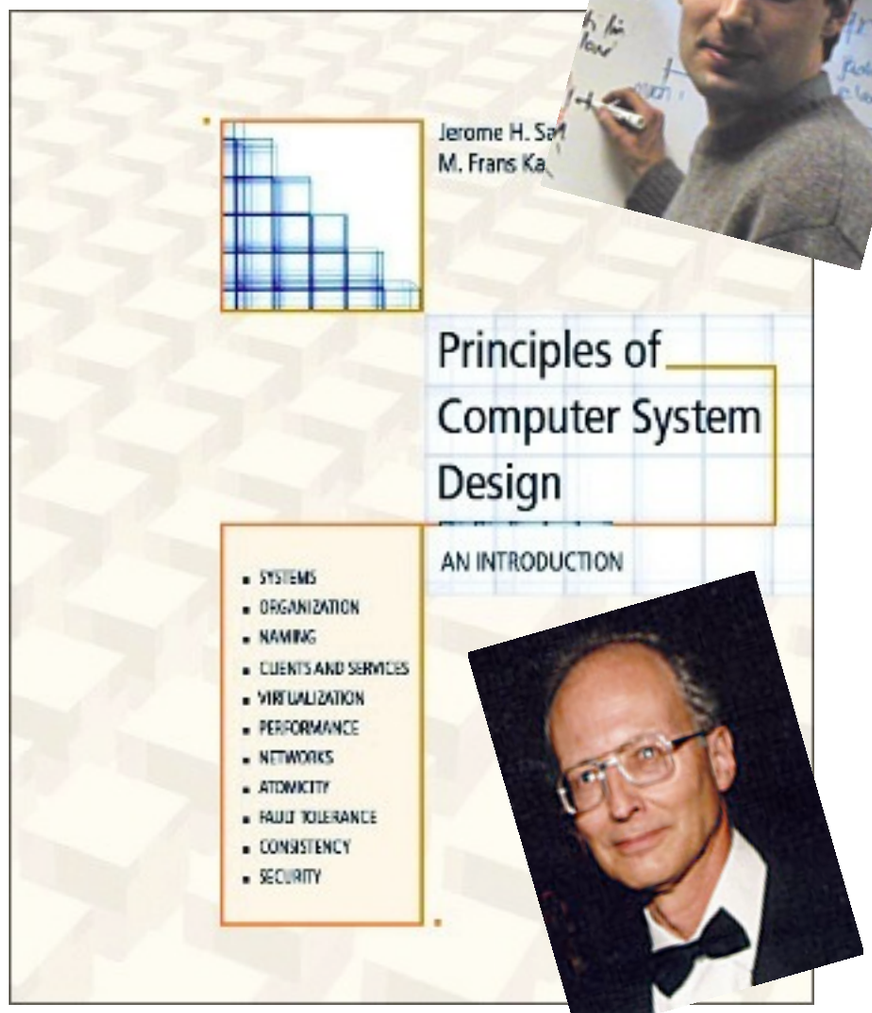

Addison-Wesley Software Security Series
SOFTWARE SECURITY
BUILDING SECURITY IN
GARY McGRAW
Foreword by Dan Geer


Addison-Wesley Software Security Series
SOFTWARE SECURITY
BUILDING SECURITY IN
GARY McGRAW
Foreword by Dan Geer

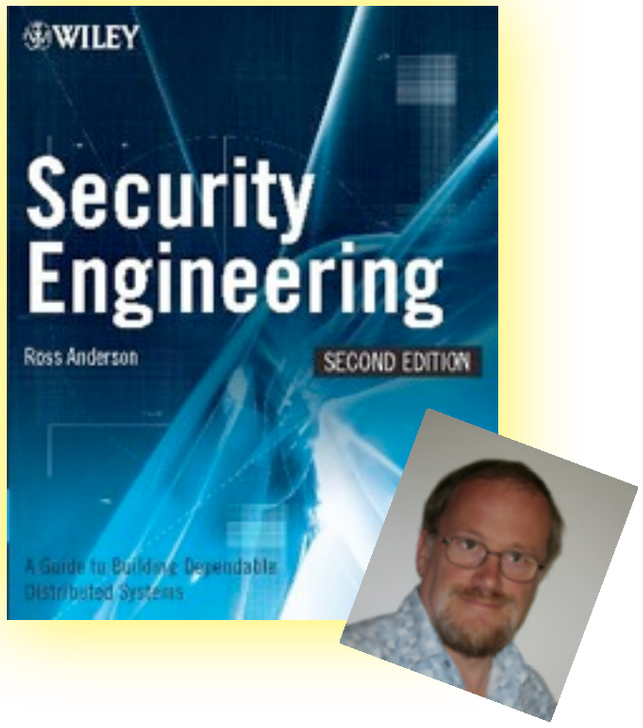


Ross Anderson,
Security Engineering

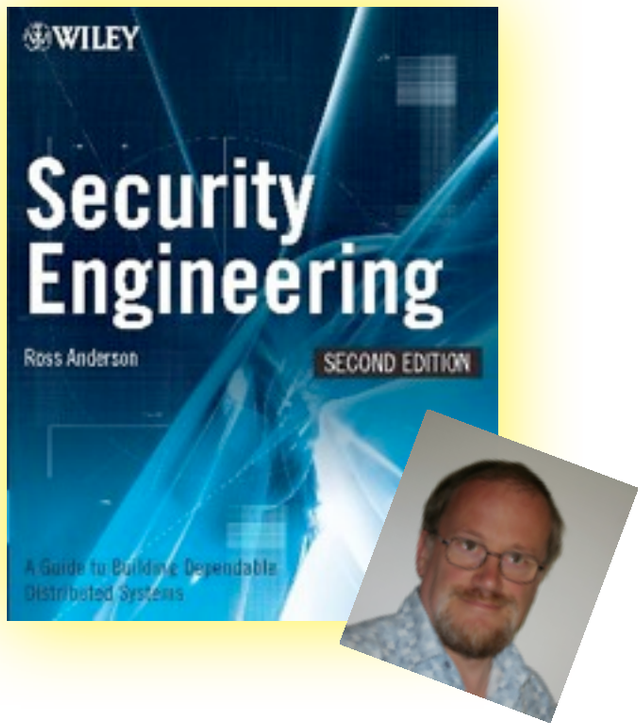
≈



Saltzer & Kaashoek,
P. of C. S. D.

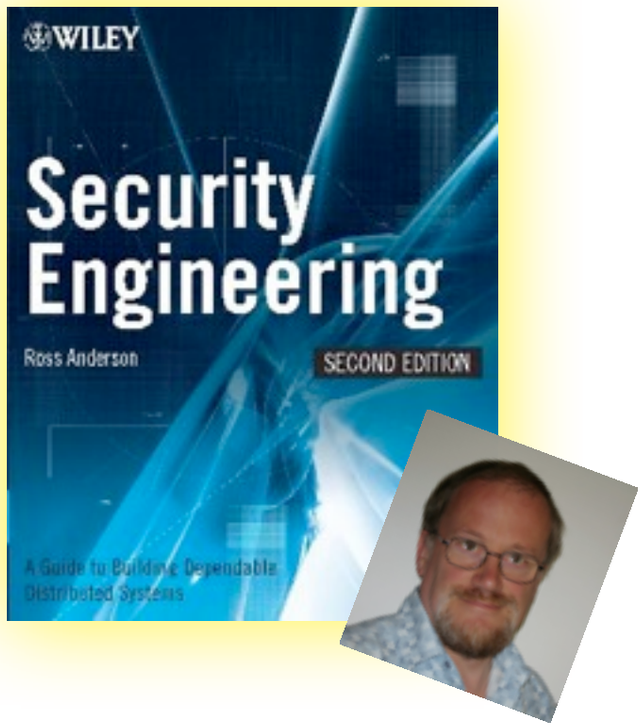


"Security engineering is about building systems to remain dependable in the face of malice, error, or mischance."



Security =

Policy + Mechanism + Assurance + Incentive



Security =

Policy + Mechanism + Assurance + Incentive

Insecurity ≈

How can I break this system?

Threat Modeling

- ... is **your job** in system design
- Think like an attacker
- **Understand** and **prioritize** incentives
- Imagine a **realistic** attacker

Attack Surface

- Which parts of your system interface with other stuff?
  - Network ports, I/O
  - Command-line inputs
  - Dependencies on other systems

Attacker Incentives

- For each element of attack surface:
  - What can a successful attacker gain?
  - What's it worth?


KRYPTONITE
KRYPTOLOK


FAIL

Chronicle / Kim Komenich

(Some) Kinds of Attackers

Value	Example	Attacker
Low	Generic PC	Script kiddie
Medium	Personal bank account	Phisher
High	State nuclear program	Another state

Script Kiddies

- Largely unskilled; main resource = time
- Use pre-packaged exploits
- May wish to sell compromised resources (e.g., sell zombie PCs to botnet)

Midrange "Hackers"

- Somewhat skilled; may have specific targets
- May be willing to use **social engineering**
- Motivations include fame, revenge, vandalism, $$$

High-End Hackers

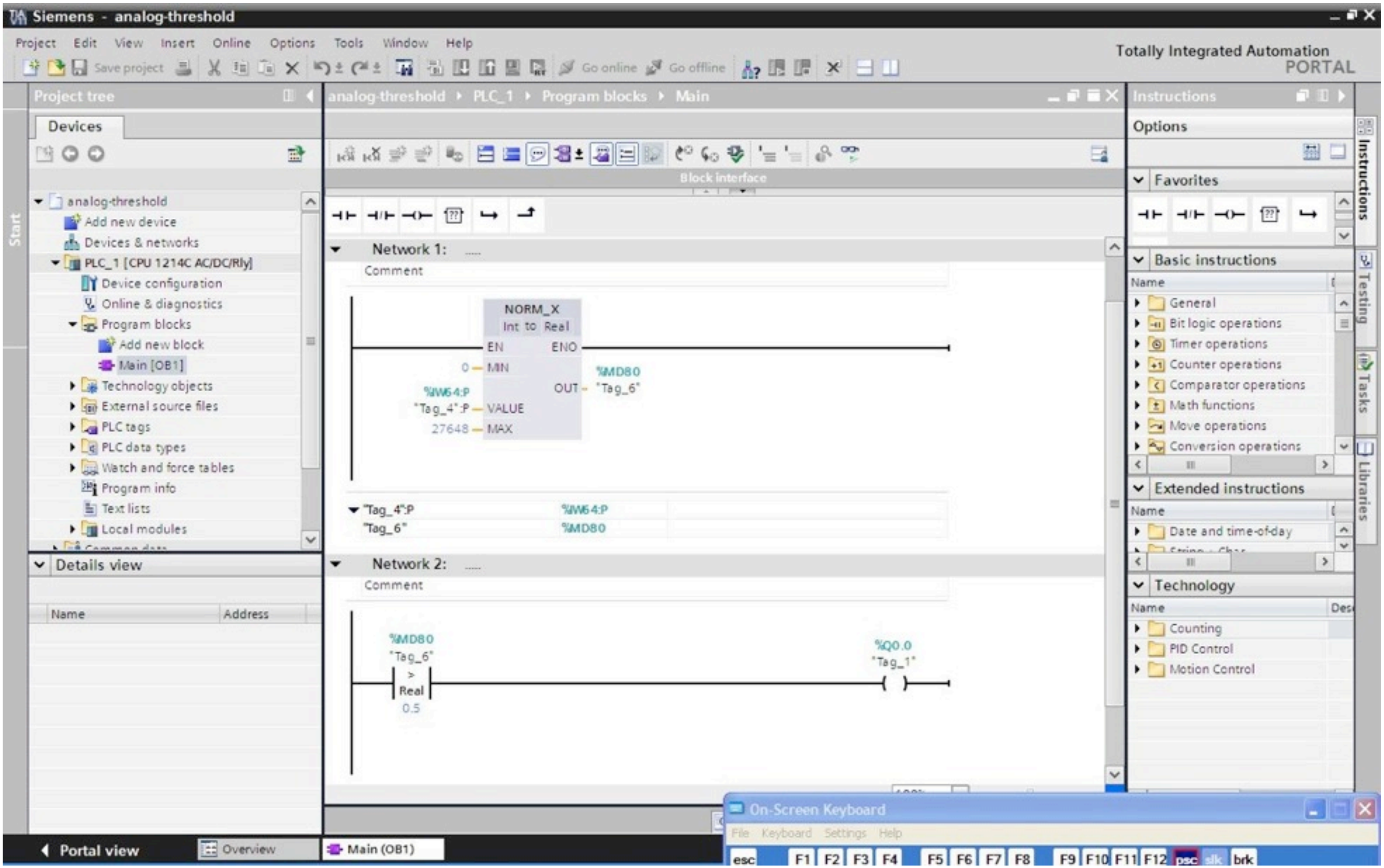
- Deep understanding of target
- Write exploits
- These days, sell exploits for $$$$$

High-End Hackers

ADOBE READER	$5,000–$30,000
MAC OSX	$20,000–$50,000
ANDROID	$30,000–$60,000
FLASH OR JAVA BROWSER PLUG-INS	$40,000–$100,000
MICROSOFT WORD	$50,000–$100,000
WINDOWS	$60,000–$120,000
FIREFOX OR SAFARI	$60,000–$150,000
CHROME OR INTERNET EXPLORER	$80,000–$200,000
IOS	$100,000–$250,000

Even Higher-End Hackers

- E.g., state agencies (NSA, Mossad)
- Specific targets for **espionage** or **sabotage**
- Advanced persistent threats — get into target and stay there


Siemens - analog-threshold
Project Edit View Insert Online Options Tools Window Help
Save project
Go online
Go offline
Totally Integrated Automation
PORTAL
Project tree
Devices
analog-threshold
Add new device
Devices & networks
PLC_1 [CPU 1214C AC/DC/Rly]
Device configuration
Online & diagnostics
Program blocks
Add new block
Main [OB1]
Technology objects
External source files
PLC tags
PLC data types
Watch and force tables
Program info
Text lists
Local modules
Details view
Name
Address
analog-threshold ▸ PLC_1 ▸ Program blocks ▸ Main
Block interface
Network 1:
Comment
NORM_X
Int to Real
EN
ENO
0
MIN
%MD80
OUT
"Tag_6"
%IW64:P
"Tag_4":P
VALUE
27648
MAX
"Tag_4":P
%IW64:P
"Tag_6"
%MD80
Network 2:
Comment
%MD80
"Tag_6"
>
Real
0.5
%Q0.0
"Tag_1"
Instructions
Options
Favorites
Basic instructions
Name
General
Bit logic operations
Timer operations
Counter operations
Comparator operations
Math functions
Move operations
Conversion operations
Extended instructions
Name
Date and time-of-day
Technology
Name
Counting
PID Control
Motion Control
Start
Instructions
Testing
Tasks
Libraries
On-Screen Keyboard
File Keyboard Settings Help
esc F1 F2 F3 F4 F5 F6 F7 F8 F9 F10 F11 F12 psc slk brk
Portal view
Overview
Main (OB1)

Cryptography

Do's & don'ts

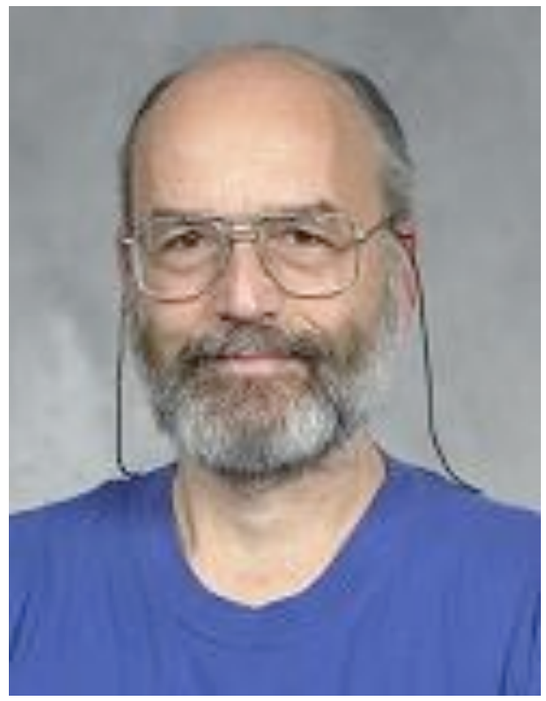
Note: cryptography != security

Rule #1

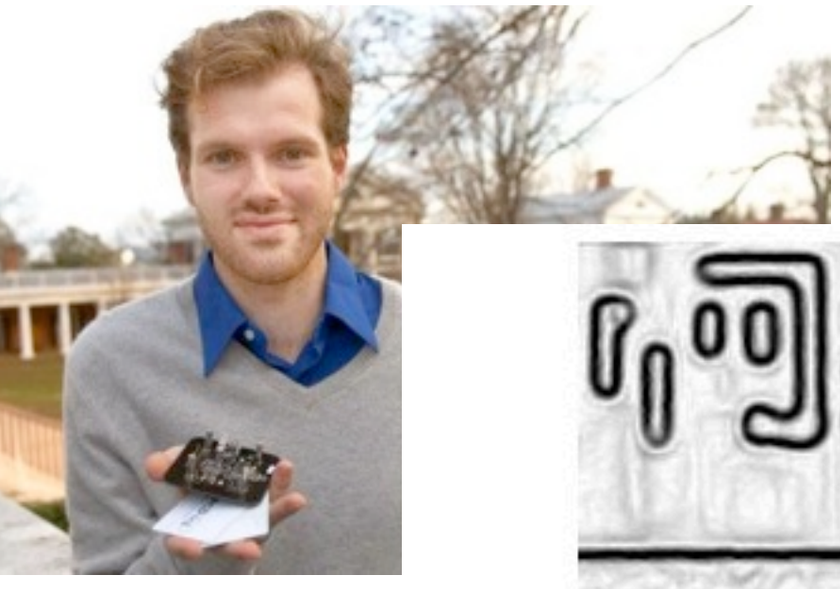
Don’t design your own cipher!

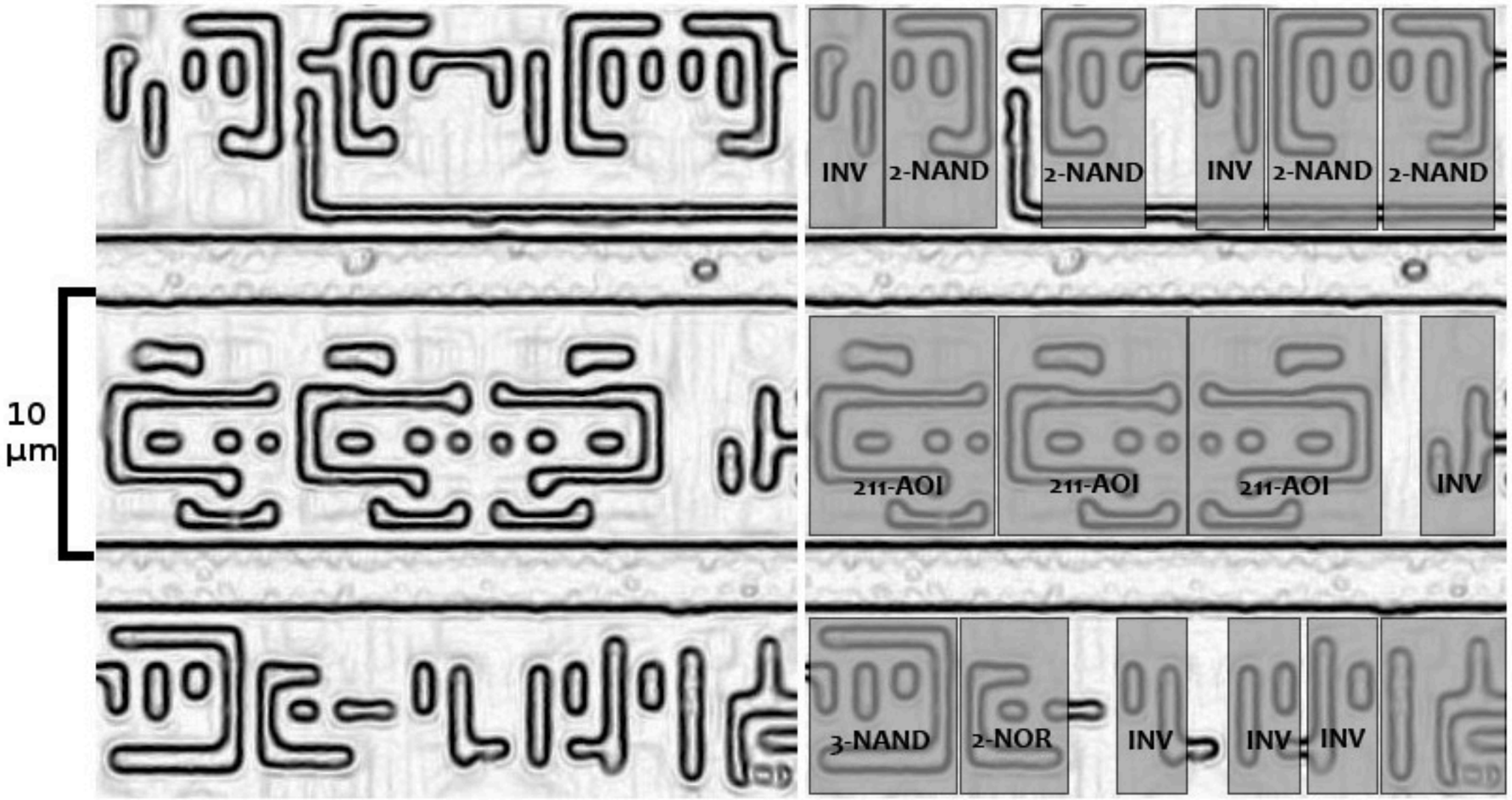
Use an existing one.

== Use AES.



Don’t pull a Mifare



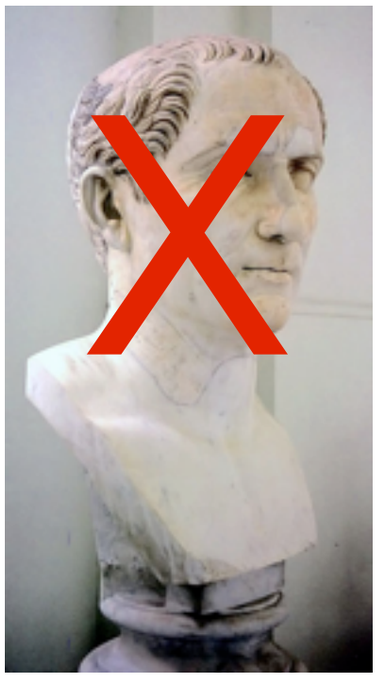


(a) (b)

Figure 1: (a) Source image of layer 2 after edge detection; (b) after automated template detection.

Rule #2

Don't rely on security through obscurity.
Your system's design will become known.



== Assume only the keys are secret.

Rule #3

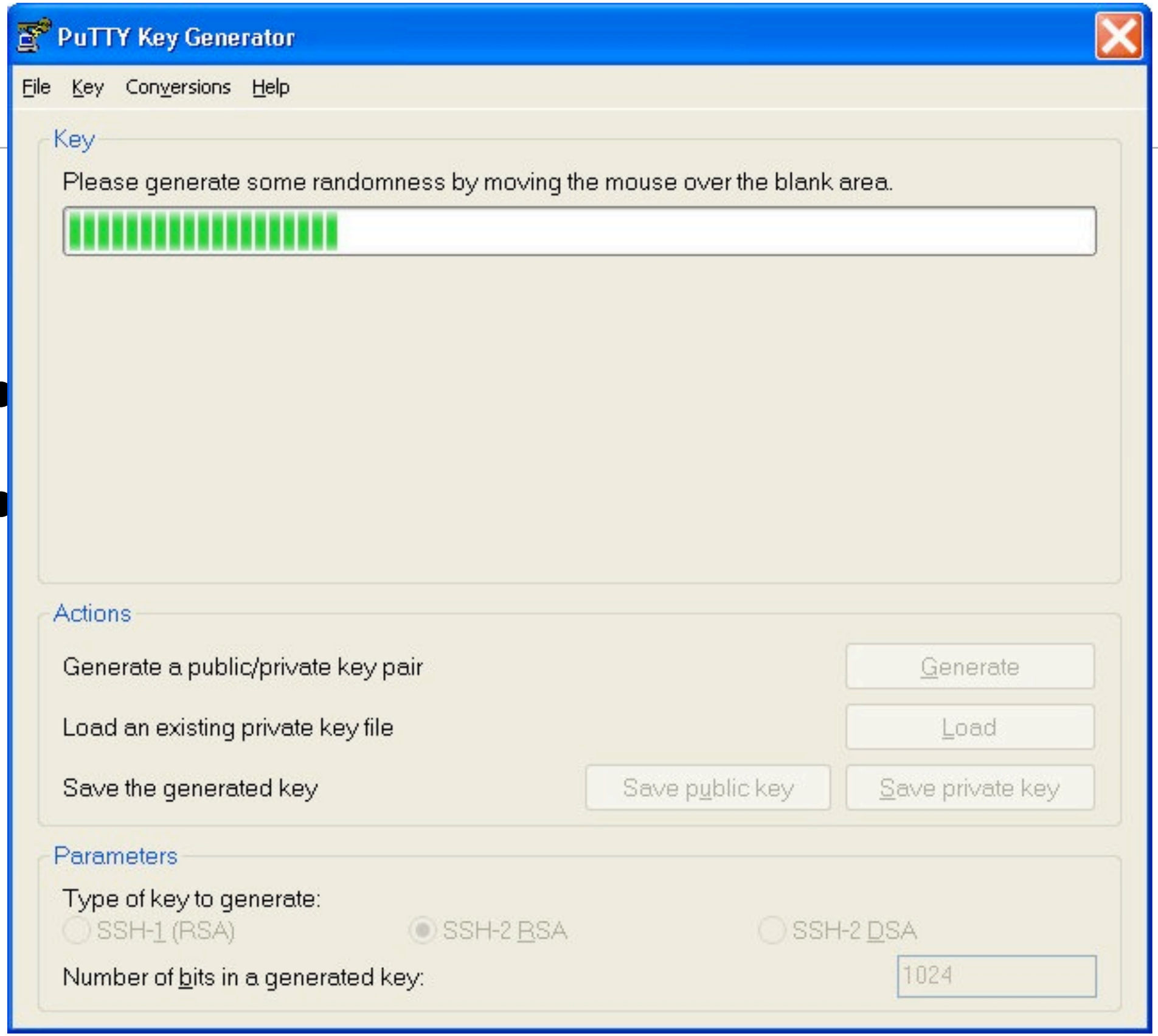
Don't use randomness incorrectly or use predictable "randomess."

Bad randomness makes attacks easy.

== Use TRNG or a good seeded PRNG

Good PRNG

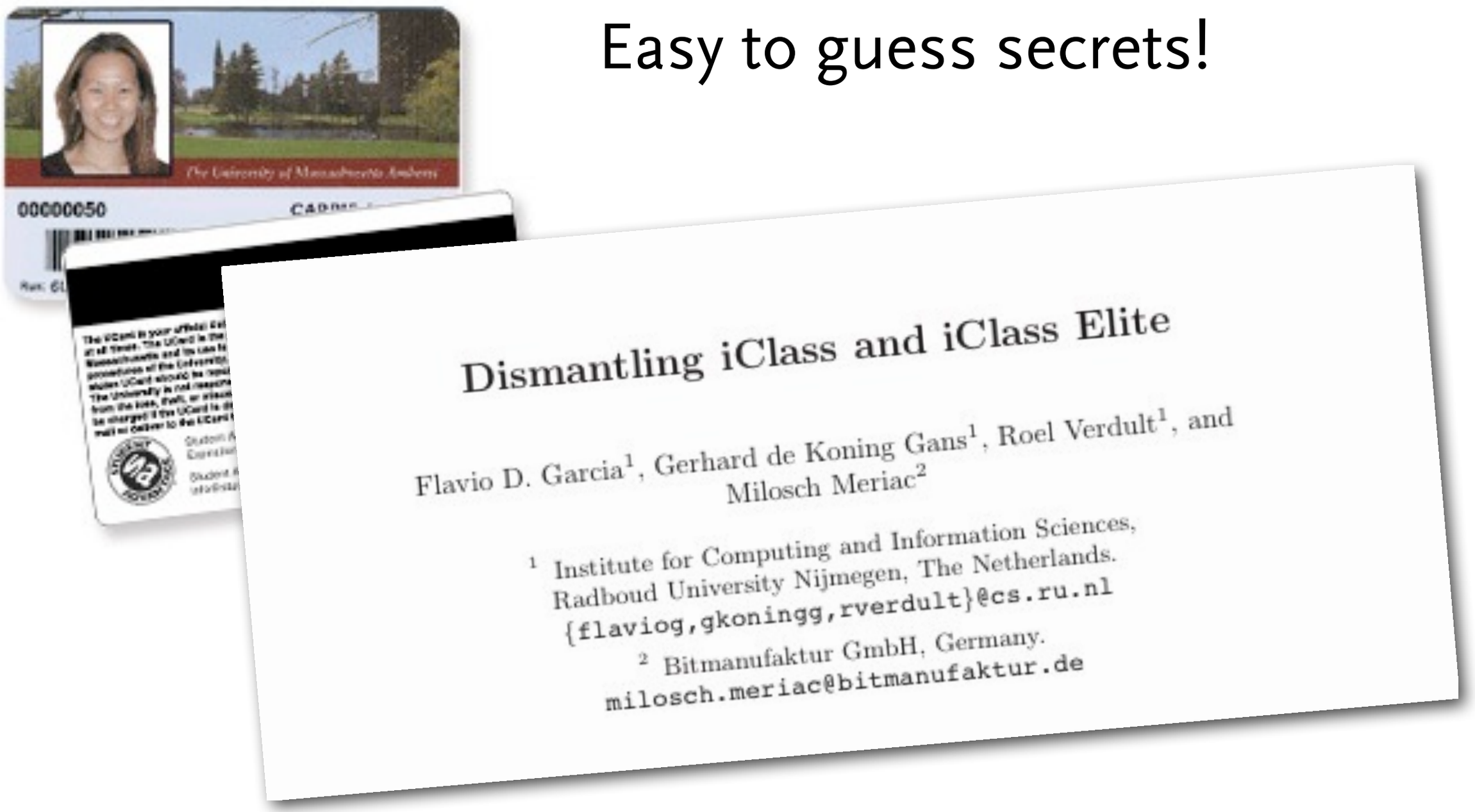
- Doesn't repeat itself (long **period**)
- Does use sources of "random" bits


PuTTY Key Generator
File Key Conversions Help
Key
Please generate some randomness by moving the mouse over the blank area.
Actions
Generate a public/private key pair
Generate
Load an existing private key file
Load
Save the generated key
Save public key
Save private key
Parameters
Type of key to generate:
SSH-1 (RSA)
SSH-2 RSA
SSH-2 DSA
Number of bits in a generated key:
1024

Bad PRNG

Easy to guess secrets!

Bad PRNG



Easy to guess secrets!

Note: Multiple PRNGs

(demo of Linux /dev/urandom vs. /dev/random)

Don’t use urandom when you want random.

Harping on Randomness

Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices

Nadia Heninger[†*] Zakir Durumeric[‡*] Eric Wustrow[‡] J. Alex Halderman[‡]

[†] *University of California, San Diego*
nadiah@cs.ucsd.edu

[‡] *The University of Michigan*
{zakir, ewust, jhalderm}@umich.edu

Harping on Randomness

Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices

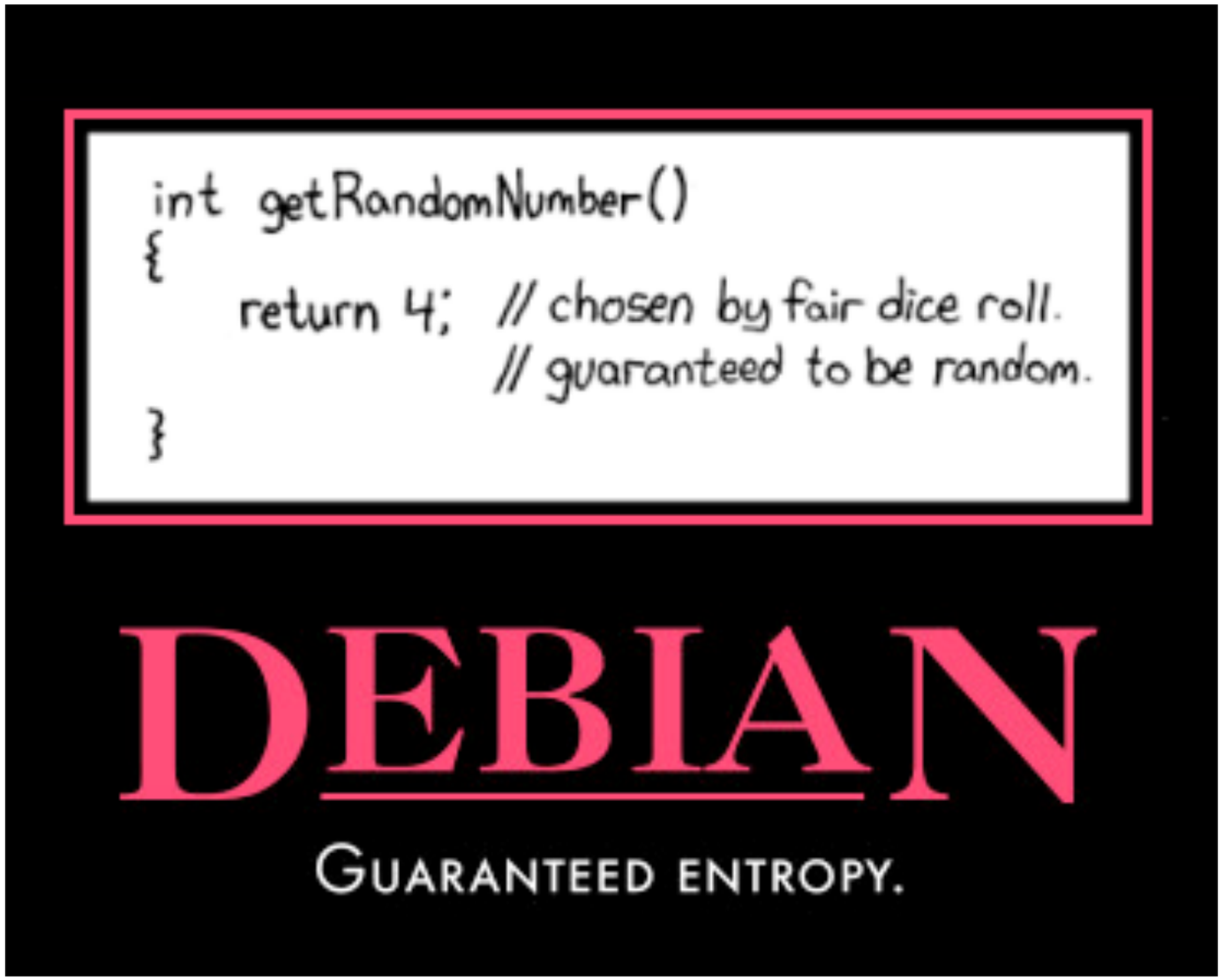
Nadia Heninger[†*] Zakir Durumeric[‡*] Eric Wustrow[‡] J. Alex Halderman[‡]

[†] *University of California, San Diego*
nadiah@cs.ucsd.edu

[‡] *The University of Michigan*
{zakir, ewust, jhalderm}@umich.edu

"We found that 5.57% of TLS hosts and 9.60% of SSH hosts share public keys in an apparently vulnerable manner, due to either insufficient randomness during key generation or device default keys" (source: factorable.net)

Debian OpenSSL disaster



(Don’t trust your tools blindly!)

Greatest Hits

(and how not to get hit)

please put on your C/C++ hats

Buffer overflows (super common)

strcpy(dest, user_supplied_input);

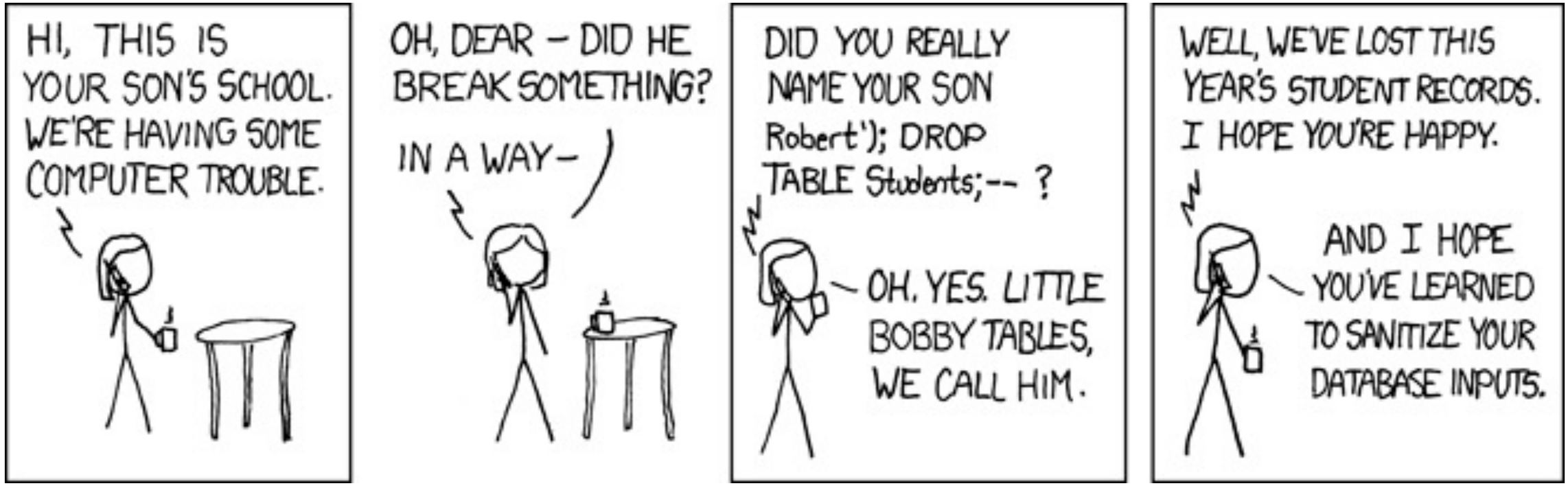
Use-after-free (somewhat common)

```
void f (p_t *p) { ...; free(p); }

f(my_pointer);
*my_pointer = 0x1234;
```

Double free (not all that common)

```
void f (p_t *p) { ...; free(p); }

f(my_pointer);
free(my_pointer);
```

Input validation



Cross-site scripting (super super super common)

Hello my name is <script>stealStuff();</script>