From Proof to Practice: Real-World Cryptography

CHES 2004

Paul Kocher

President & Chief Scientist,
Cryptography Research, Inc.

www.cryptography.com

575 Market St., 21st Floor, San Francisco, CA 94105

Focus of my work

- Focus on high-risk commercial systems
  - 9 years running CRI, bridging applied work and theory
    - Crypto, risk management, hardware, networking…
    - Many industries (Financial, content, communications…)
  - Most work with big companies with real risks/losses
    - Focus on fraud, piracy, infrastructure…
- Consulting, licensing, and research
  - Consulting: Evaluation, implementation, design
  - Research: Real security problems & responses
  - Licensing: DPA, Tamper-resistance, content security

"Necessity is the mother of invention"

Real-world projects require:

- robust implementations
- actual demonstrations of weaknesses
- confidence there aren't problems

… given very limited information, time, and money

Talk Outline

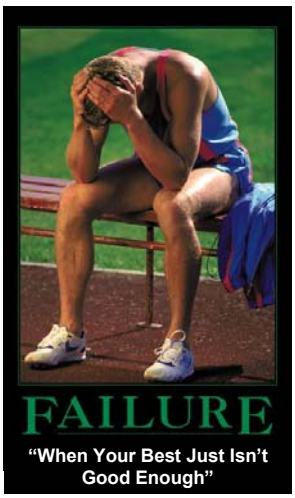
- Pessimism
- Optimism
- Realism

Pessimism





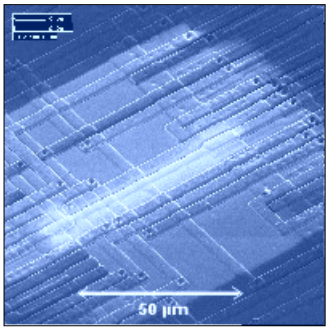
My work: Obsession with Failures



- Evaluating
- Understanding
- Preventing
- Surviving
- Recovering

Applied Cryptanalysis

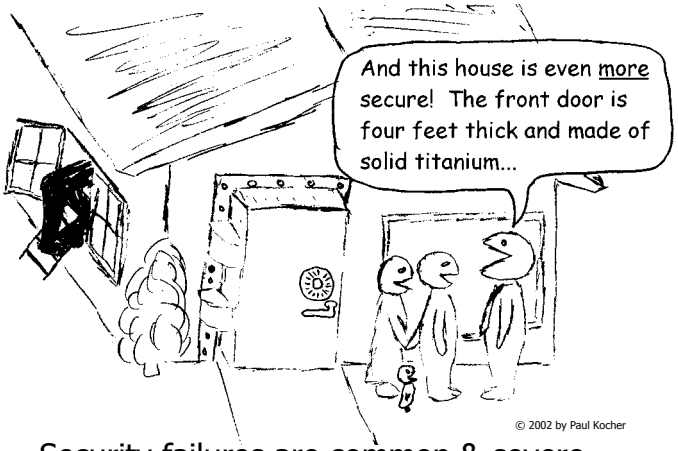
- Problem: Getting a key from a device
- Naïve solutions are great if they work
  - Brute force, Invasive…



Cryptography is advancing fast

- Cryptography is winning over cryptanalysis:
  - Excellent toolbox of algorithms & protocols
  - Solid mathematical tools
  - Vibrant academic research
- Great algorithm strength & key sizes
  - Cryptographic strength is improving with Moore's Law

… but something is very wrong



- Security failures are common & severe
- Attackers bypass our strengths ("how rude!")

Research Mismatch

- Chasm between research and application
  - Few engineers understand crypto
  - Few researchers understand engineering
  - Data about failures is hard to get

Requirements Tested & Test Results (continued)

No.	Requirement	Test Scenario	
1.25	There shall be the use of cryptographic operations during voter authorization.	Various means of "voter identification" should be secure. The data on a voter authorization token should not be discernable.	Voter allow The c not e by in speci preve

Direct Recording Electronic Technical Security Assessment Report, Nov. 2003, Secretary of State, Ohio (http://www.sos.state.oh.us/sos/hava/files/compuware.pdf)

Psychology of crypto & risk:

Our brains are wired poorly

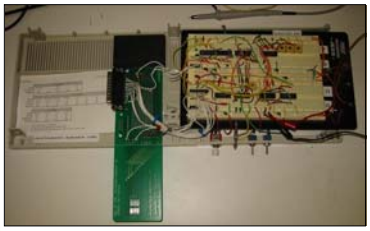
- Hard to think rationally about unexpected, unpleasant, and low probability events
  - Denial about the nasty problems
    - Case study: Company invested in exhaustive security evaluations that "proved security" in a very limited model which did not reflect the attacks that ultimately occurred
  - "Blinded by the light": Tacit assumption that strength in one area will spill over to others
    - Company spent lots to upgrade from 1024 to 2048 bit RSA because of Twinkle… without fixing protocol flaws, poor key management, power analysis vulnerabilities (and added a buffer overflow)

Most security failures result from mismatches between assumptions & reality

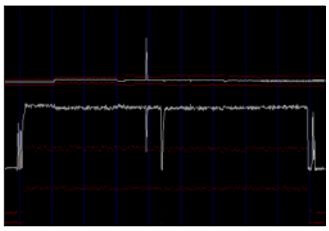
- Reliable (Protocol is what's done)
- Closed (No info beyond protocol)
- Bug-free (Design is correct)

What if the Assumptions are Wrong?

- Reliable (Protocol is what's done)

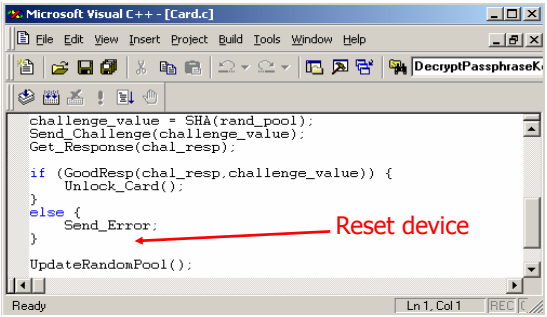


Glitching to violate a reliability assumption



What if the Assumptions are Wrong?

- Reliable (Protocol is what's done)



What if the Assumptions are Wrong?

- Closed (No info beyond protocol)

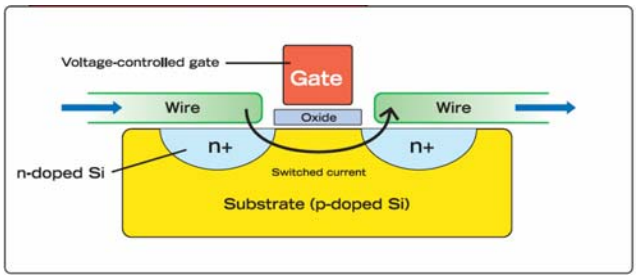
Paul Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems," Crypto 1995.

My starting observations:
① Timing measurements contain info that isn't *clearly safe*
② Crypto is extremely brittle

Discovery:
Timing channels can be a practical way to break crypto

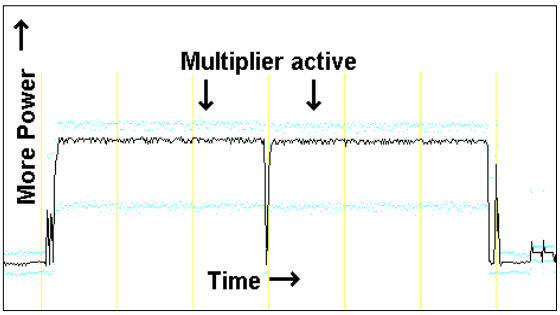
What if the Assumptions are Wrong?

- Closed (No info beyond protocol)



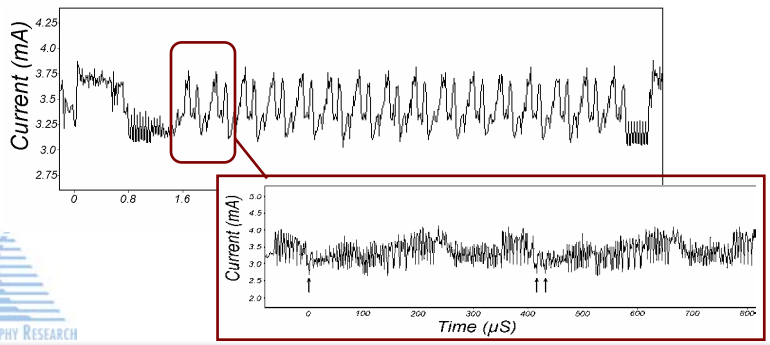
What if the Assumptions are Wrong?

- Closed (No info beyond protocol)



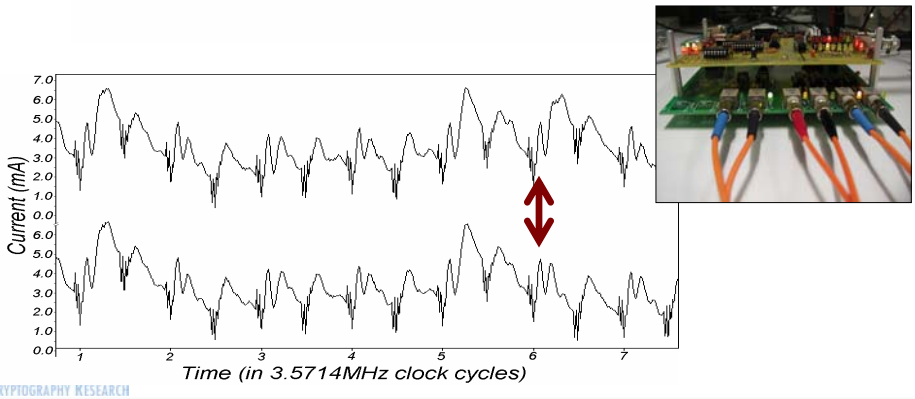
What if the Assumptions are Wrong?

- Closed (No info beyond protocol)



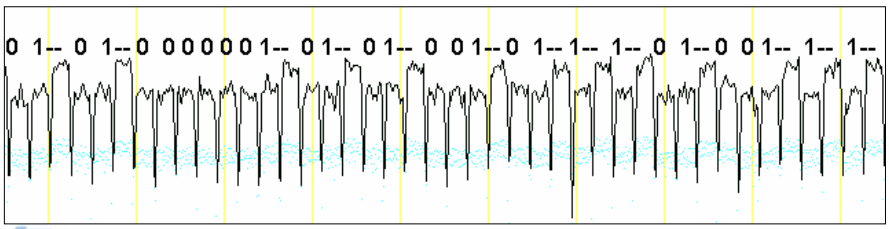
What if the Assumptions are Wrong?

- Closed (No info beyond protocol)

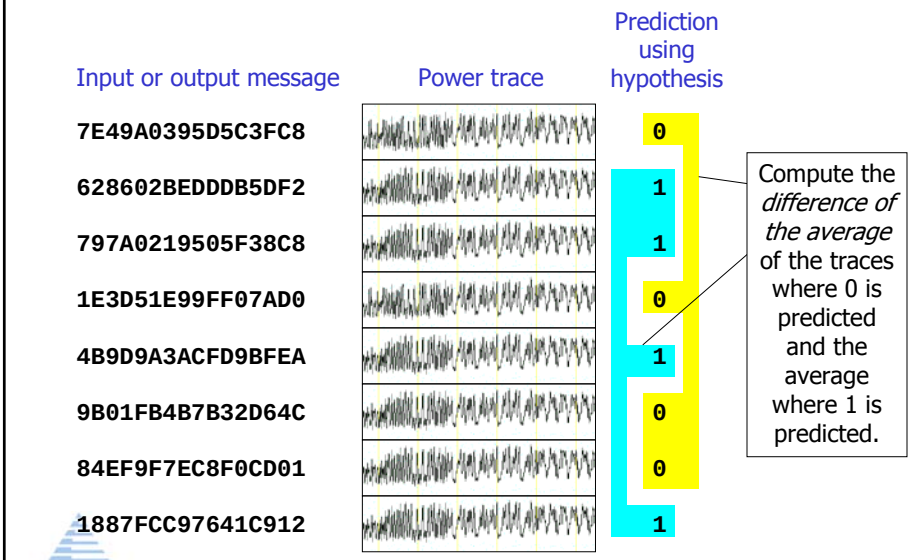


What if the Assumptions are Wrong?

- Closed (No info beyond protocol)

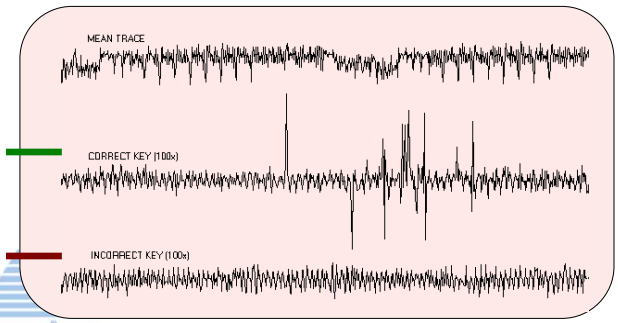


SPA: Reading off an RSA secret exponent



What if the Assumptions are Wrong?

- Closed (No info beyond protocol)



Paul Kocher, Joshua Jaffe, and Benjamin Jun, "Differential Power Analysis", Crypto 1999.

What if the Assumptions are Wrong?

- Bug-free (Design is correct)

  - Architects & engineers are human
    - Humans can't write bug-free code
    - We can't anticipate all attacks
  - Problem: Can we get security despite our human fallibility?

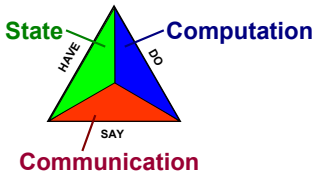
  (More on this in a minute…)

There are many more assumptions…

- Few initial security observations are exploitable attacks
  - "The amount of time consumed depends on the input value."
  - "The first bytes out of RC4 aren't quite random for related keys."
  - "Ethereal core dumps with a particular corrupted capture file."
  - "If the secure link is down, users will switch to an insecure one."
  - "This error message conveys information derived using the key."
  - "Password-protected files can be set to be read-only."
  - "The computer's ID is transmitted twice during the protocol."
  - "The same key is used for both encryption and MACing."
  - "DRAM errors become common above 120ºC."
  - "The new admin has never used e-mail before."
  - "Cosmic rays cause random bit errors in DRAMs."

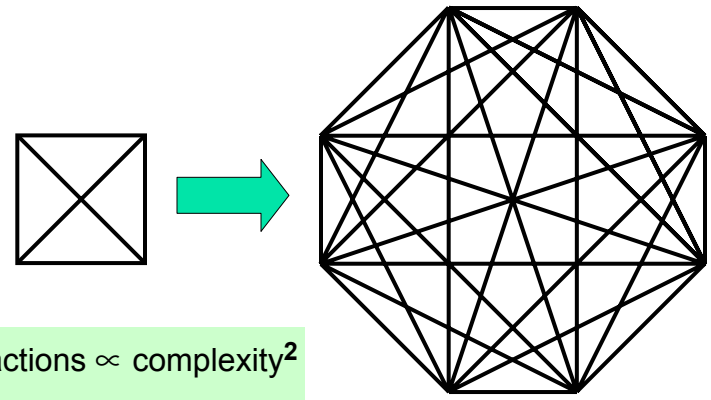
Being Pedantic
(Voiding the Warranty)

- I like to look at the rules & if they make sense
  - What is allowed? What isn't? What info is secret?
- Look for violations (however small)
  - Goal: Find tiny, weird, annoying corner cases – then expand
  - Categories of observations:
    - Clearly OK and anticipated by the design
    - Unexpected but not exploitable
    - Violates the security model
    - ***Unknown (e.g., complex)***



Growth in Interactions

- Bug-free

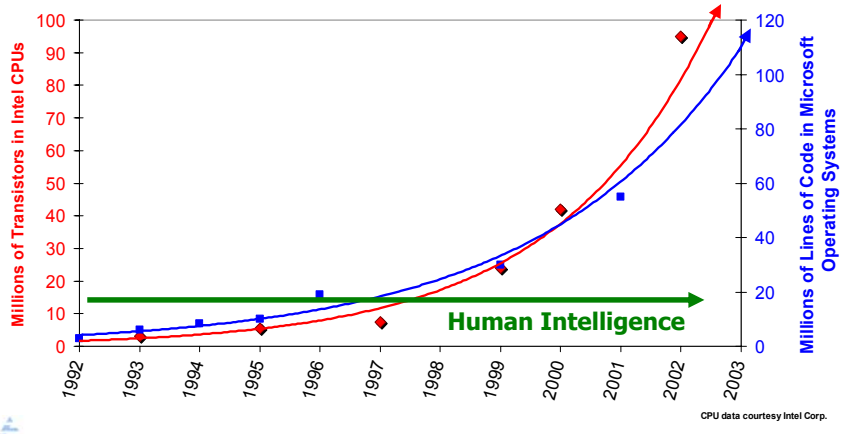


interactions ∝ complexity^2

Observation: # bugs ∝ LOC^2

Growth in Complexity

- Bug-free



More devices with more complex hardware running more software and networked to more computers.
Embedded devices are complex too (merely a few years behind)

Thinking About Complexity

- Moore's Law: 2X transistors every 18 months
  - Also increasing: Storage, bandwidth, code size, RAM…
- Interactions typically increase as (complexity)^2.
  - 10X LOC = 100X interactions

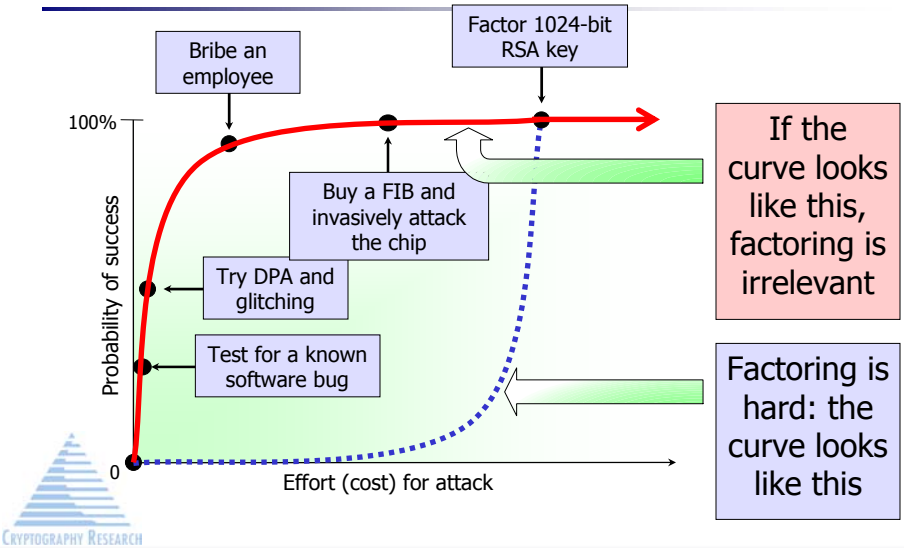
Attacker	**Evaluator**	**Designer**
Complexity is helpful	**Complexity is scary**	**Complexity is awful**
• More avenues • More bugs • More SPFs • More interactions	• More flaws to find • More flaws to miss • More skills required • Reduced confidence • Less time / LOC	• Far more things can go wrong • More skills required • Must get lower bug density to stay even • Failures are worse

Thinking About Abstraction

- Functional engineers use abstraction to deal with complexity...
- But abstraction is a mixed bag when it comes to security

Conventional Engineering:	Security Engineering
▪ Layers of abstraction are good ▪ Can ignore the details of what's inside ▪ Reduces knowledge & skill required for engineers ▪ Decreases design time ▪ Simplifies testing ▪ Increases portability	▪ Still have to know the details ▪ Details of innards can be hard to find (e.g., specs unavail.) ▪ Layers hide problems ▪ Can help, can hurt: ▪ Good: SHA, AES ▪ Scary: RSA_PrivateKey() ▪ Awful: system("gpg –encrypt")

The Low-Hanging Fruit



Pessimism recap:
Real-world security is elusive

- Flaws are very common
  - The vast majority of our product security evaluations find catastrophic flaws
  - We look at better-than-average products
  - Requires creativity & time to find problems: Automating creativity is currently impossible

Job security!

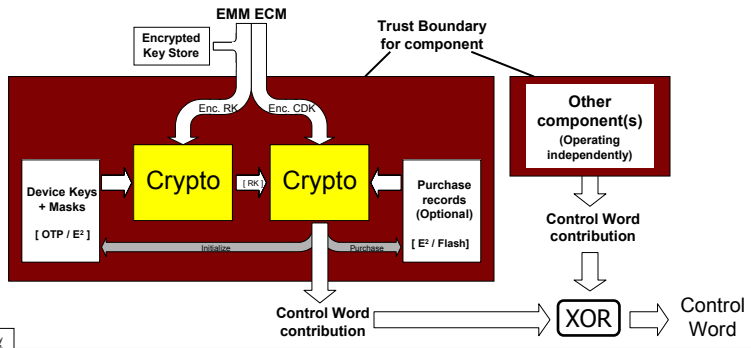
Optimism:
Crypto can help solve these problems

Security Requires Correct Assumptions

- Two approaches for security
  - Make systems that are reliable, closed, and bug-free…
  - Make systems that survive glitching, leaking, bugs…
- Good crypto architectures can help with both approaches

Example: Pay TV security (CryptoFirewall™)

- Traditional model: A complex CAM derives control words (keys)
- Improved model: Multiple components contribute to keys
  - In CAM die/package, but keyed independently
  - Hardened component – only task is security (= managed complexity)
  - Eliminates single points of failure



Note: CRI's CryptoFirewall™ technology is covered by U.S. patents 6,289,455 & 6,640,305. Other U.S. & international patents issued and/or pending.

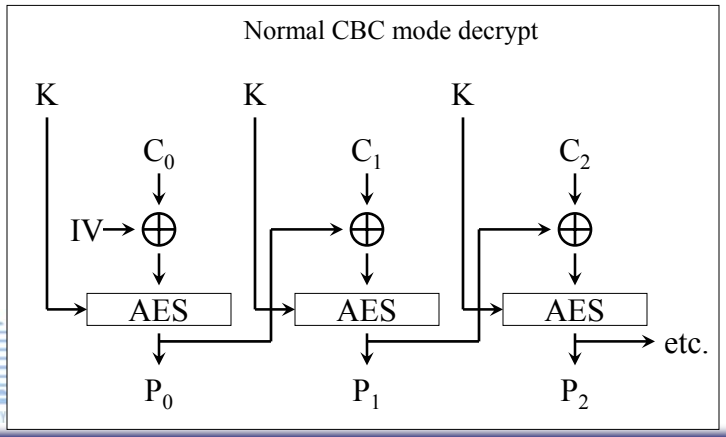
Example: Pay TV security (CryptoFirewall™)

- Made security-critical roles redundant & isolated
- We just used some simple crypto to address an assumption problem:
  - Enabled strong security despite some bugs
    - Security ≥ max component strength
      (Traditional approach: Security ≤ worst component)
  - Can isolate complexity
    - Modern CAMs are quite complex
    - Complexity can be tightly controlled because non-critical tasks are offloaded

Symmetric Leak-Proof™ Crypto

Normal CBC encryption

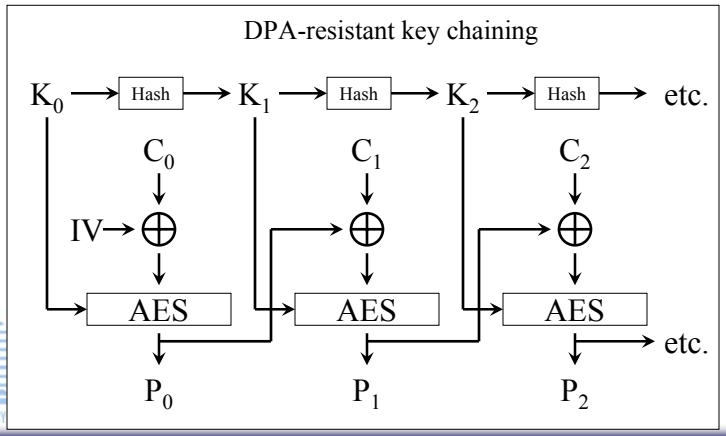
Normal encryption re-uses the same key for every block.



Example:
Symmetric Leak-Proof™ Crypto

DPA-resistant key chaining

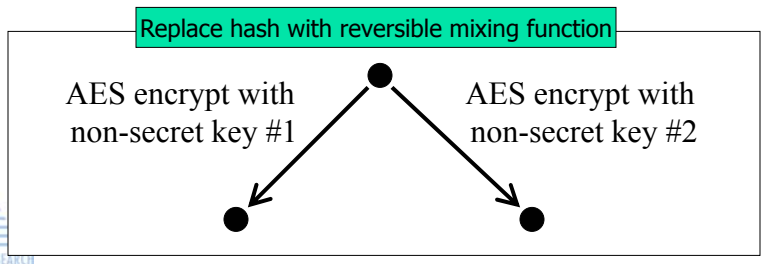
Key updates between blocks cause "healing" of leaked data.



Example:
Symmetric Leak-Proof™ Crypto

Hierarchical approach

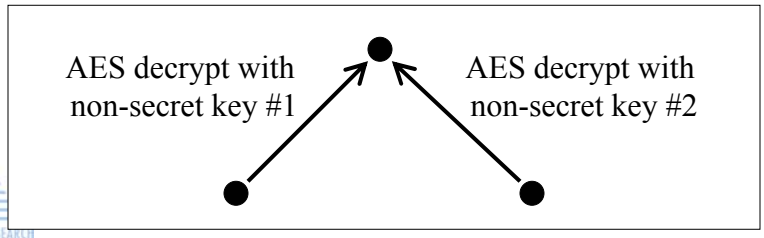
- Eliminates verifier-side complexity of iterated hashing
- Preserves security after many leaky operations
  - Assumes strong cipher & that leakage function does not include key update function



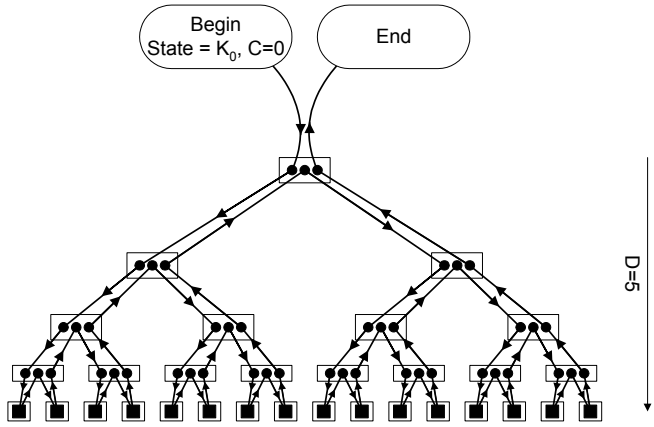
Example:
Symmetric Leak-Proof™ Crypto

Hierarchical approach

- Eliminates verifier-side complexity of iterated hashing
- Preserves security after many leaky operations
  - Assumes strong cipher & that leakage function does not include key update function



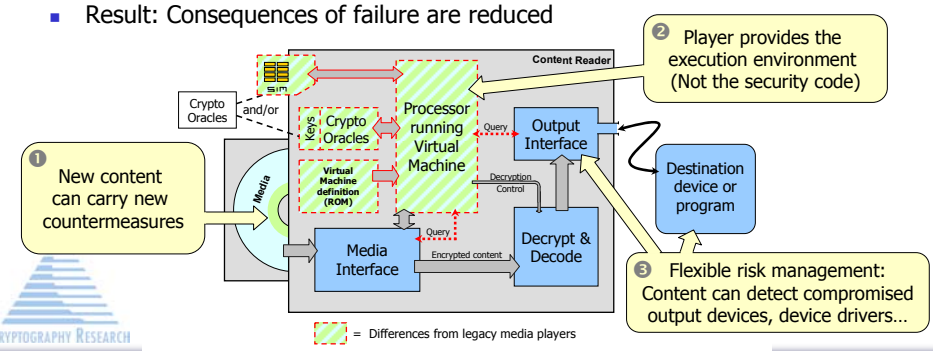
Example:
Symmetric Leak-Proof™ Crypto



Client time = 1 step
Server time = max $\log_2(N)$ steps

Example:
Security for Optical Media (HD-DVD)

- What if bugs are inevitable?
  - Some next-generation DVD player models will get hacked
  - Can't revoke legitimate users' players (or player keys)
  - Must augment crypto with something renewable
- Approach: Put security code on media & run on player
  - Very simple virtual machine (~100 lines of code)
  - Analogies: Anti-virus software, game copy protection, software activation…
- Result: Consequences of failure are reduced



What these examples illustrate

- The careful use of crypto can help mitigate the most likely failure modes
  - Building crypto that acknowledges and addresses the possibility of bugs, leakage…
  - Architectures can reduce overall risk

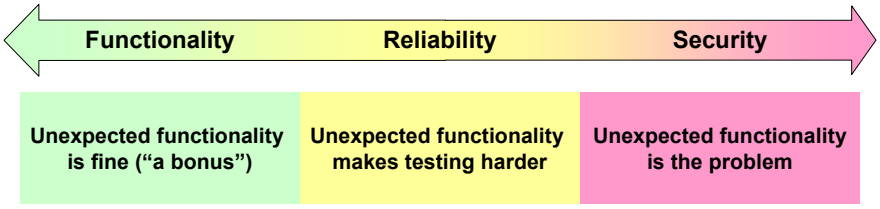
Realism

Range of Objectives

Functionality	Reliability	Security
Criteria for success: It works in normal situations	Criteria for success: It works in the face of the unexpected	Criteria for success: It works despite people trying to make it fail

Crypto / security requirements are fundamentally different from traditional engineering

Insecurity as Bad Functionality



- Conventional engineering is about enabling functionality
  - Example: Programming = hooking APIs together
- Typical security goal: Prevent bad things from happening
  - The ability to do bad things is undesirable functionality
- Problem: Vastly easier to specify, control, and verify what a device *does* than what it *cannot do*

Learning Engineering ≠ Learning Security

- Normal engineering skills can be harmful for security

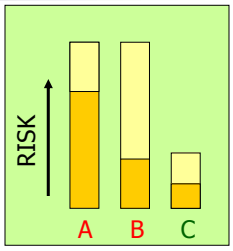
Conventional Engineering	Security Engineering
Iterate: Try, fail, fix • Rewards taking risks	Prevention • Goal is to avoid risks
Creation of "black boxes" • Results are what matters • If it works, it's good	Trust requires transparency • Focus: Process, details • Functionality isn't the problem
Challenge: Performance, cost	Challenge: Assurance
Process is intuitive	Process is often not intuitive

Security = Minimizing Risk

- Security is the ***absence of risk***
- What is risk?

$$\sum_{\text{All risks}} \left(\begin{array}{c}\text{Probability}\\ \text{of failure}\end{array}\right) \times \left(\begin{array}{c}\text{Consequences}\\ \text{of failure}\end{array}\right)$$

(Note: Assumes independence)



- Security is not binary: No (useful) system "is secure"
  - There are only varying degrees of risk
  - Every aspect of a system has a nonzero probability of failing
    - Some probabilities are low (practical cryptanalysis of AES)
    - Some probabilities are high (protocol flaws, software bugs)
  - Contrast: Functionality is binary (feature checkbox)

The Real-World Security Goal

Focus on those who take the risk
Usually, but not always, who pays for security

Provide **relying parties** with **rational confidence** that certain **undesirable outcomes** are unlikely.

Not luck or faith
Goal: Burden of proof defaults to insecure

Must match the security need
- Business need: Broad scope
- Research problem: Narrow scope

"Rational confidence"

- Why are airplanes trustworthy?
  - Conservative engineering
  - Thorough documentation
  - Redundancy (physical & human)
  - Relatively good history of safety
  - Liability for failures
  - Safety standards & regulations
- Not:
  - Functionality (e.g., ability to fly)
  - Incomprehensible creativity
- Trust is based on verifiable evidence that one's interests are protected

Cost / Benefit Perspective

- Crypto has many "costs" that offset strengths
  - Time to market
  - Cost per device
  - Risk of implementation failure
  - Engineering resources
  - Administrative cost
  - User/operator burden
  - Additional failure modes
  - Liability risks
- Research that improves any of these areas will result in better security

Conclusion

- Pessimism
  - Security is a frighteningly hard, subtle, and complex problem
  - Ordinary engineering doesn't work for security – and today's approaches are costly, failure-prone…
- Optimism
  - Crypto is a pillar of strength amid chaos and insecurity
- Realism
  - Many open problems: How can we affordably get strength, renewability & assurance?

Contact Information

Paul Kocher

575 Market St., 21st Floor
San Francisco, CA 94105

paul@cryptography.com
Tel: 415.397.0123

www.cryptography.com

Note: New address

Interested in joining our team? Please let me know or send your resume to jobs@cryptography.com