Design & Validation Strategies for Obtaining Assurance in Countermeasures to Power Analysis & Related Attacks

NIST Physical Security Testing Workshop – Honolulu
Monday, Sept. 26, 2005

Paul Kocher
President & Chief Scientist

Cryptography Research, Inc.
www.cryptography.com
575 Market St., 21st Floor, San Francisco, CA 94105

About Cryptography Research

- Founded in 1995:
  - Goal: Help understand and solve important real-world security problems
  - Major applied focus: Products incorporating CRI technology secure over $100 billion in commerce annually
- Main industries served:
  - Financial Services
  - Wireless / Telecommunications
  - Pay Television
  - Internet
  - Entertainment
- Business areas:
  - DPA countermeasure licensing
  - Anti-piracy technology licensing (pay TV, optical disc formats)
  - Other areas include consulting services, DPA workstation, education

The Assurance Problem

- Goal: Obtain confidence in countermeasures to DPA + related attacks
  - Countermeasures are essential for tamper resistant crypto devices
    - Power analysis attacks are practical, well-understood, non-invasive, and easy to repeat
  - Quality of products varies widely
    - Independent validation is needed to verify vendor claims
      - Vendor claims often have little to do with products' quality
      - Some ignorant vendors make incredible claims
      - Some sophisticated vendors may be very modest
  - Validation objective: assess the likelihood that products do (or do not) meet defined security requirements
    - Security testing is an imperfect process (can prove insecurity, but not security)… but is essential for establishing confidence in products
  - Validation framework must address security requirements without imposing excessive burden on vendors or test labs

Assurance needs vary

- Some product types require DPA protection, some don't
  - Not required if device is not expected to be physically tamper resistant
  - Required if keys must be secure from non-invasive attacks
- Among devices that have DPA countermeasures, the strength of the protection & level of the validation vary
  - For a multi-purpose testing framework such as FIPS, different security levels should have different requirements
    - Lower levels = less burden on designers & labs
    - With more effort (or better design) it is possible to obtain higher levels of assurance in security

Approaches

- This talk will explore how higher levels of assurance can be obtained at the lowest cost
  - Will examine validation strategies
    - Testing processes with more information about a product and with more lab resources can give better results
  - Will examine design strategies
    - Products that are designed to be testable can yield much higher assurance than those that are not

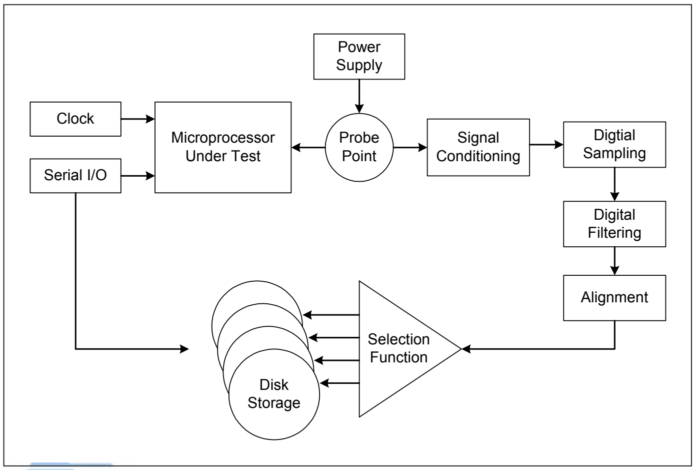
---

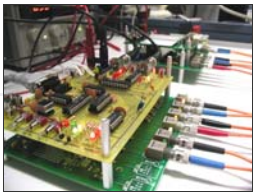
A note on design vs. validation

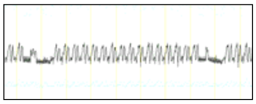
- This talk assumes design & validation are separate:
  - Design goal: Produce a demonstrably secure product
  - Validation goal: Verify the evidence presented by the designer and assess whether the overall risk is acceptable
- Note: Validation role is not to break the product
  - Insecure products should consistently fail validation because the evidence was not conclusive
    - Not necessary to actually demonstrate an attack
  - "Secure" products may fail validation if the evidence is not conclusive
    - Not necessary to actually demonstrate an attack
  - Designer's job is to make a compelling case for security
    - Validator's role is to verify that the case is solid

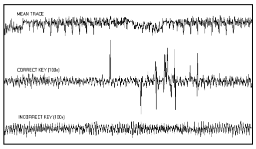
Most effort should be incurred by the product designer. (Test lab has less information and usually much less $)

DPA evaluations: General process



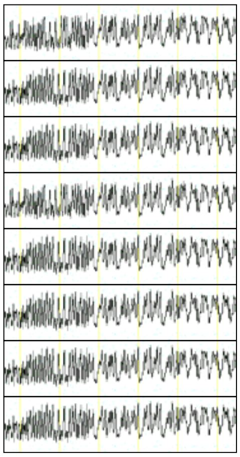






Live AES example: dpa_aes.bat
(see selection function results, several wrong and one right that solves for 8 bits of the key)

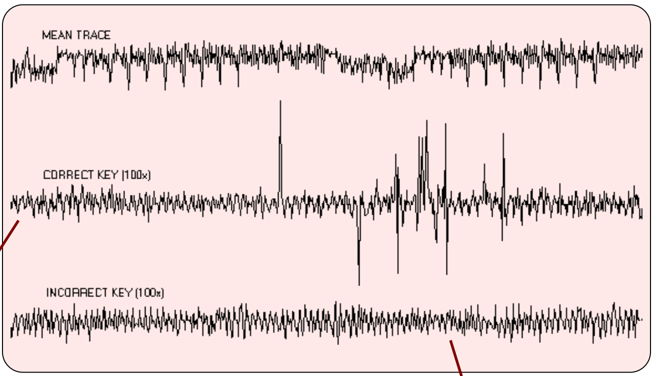
DPA: Hypothesis testing using statistics to exploit tiny leaks buried in noise



Input or output message	Power trace	Prediction using hypothesis
7E49A0395D5C3FC8		0
628602BEDDDB5DF2		1
797A0219505F38C8		1
1E3D51E99FF07AD0		0
4B9D9A3ACFD9BFEA		1
9B01FB4B7B32D64C		0
84EF9F7EC8F0CD01		0
1887FCC97641C912		1
⋮		

Compute the *difference of the average* of the traces where 0 is predicted and the average where 1 is predicted.

A typical DPA result



If the hypothesis is right

Predictions will have some (perhaps tiny) correlation to what the device did, and difference of the averages will approach a nonzero value in these places

If the hypothesis is wrong

Predictions have no correlation to what the device actually did, so the difference of the averages will approach 0 (flat) everywhere

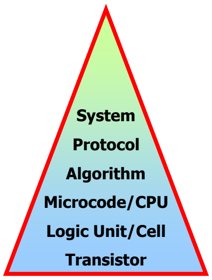
---

“Difference of the averages”

- The statistics automatically pull the key from the noise
  - Enables testing of arbitrary hypotheses
  - Noise, measurement errors, etc. all vanish as the number of measurements increases

What happened?

A characteristic of transistors (the lowest layer) compromised each layer above, ultimately compromising the system & business objectives.



DPA evaluations: General challenges

- DPA involves multiple layers in a design
    - Does not involve just one layer of abstraction
    - Vulnerabilities are not necessarily the result of functional properties that are apparent by looking at source code
        - Example: Analog properties of complex digital circuits
- Multidisciplinary skill set required
    - Cryptanalysis, number theory, transistor physics, digital circuit design, statistics, software development, lab instrumentation, data acquisition, signal processing…
- Difficult to conclusively demonstrate security
    - Hard to show that a key is *not* present in piles of leaked data
    - Similar to hunting for software implementation bugs

DPA evaluations: Black box testing

- Black box evaluations are the common approach today
    - Use power traces to (a) infer information about the design then (b) extract keys.
        - Approach: Form hypotheses then use traces to test them
    - Lack of design information creates testing challenges:
        - Tester must have a deep understanding of the range of possible implementation techniques & countermeasures
        - Effectiveness also depends on lab perseverance & capabilities (Handling multi-gigabyte data sets, advanced data processing/imaging, etc.)

DPA evaluations: Black box testing

- Advantages of black box approaches
  - Avoids burden for vendors to disclose security info
  - Results tend to be unambiguous (= whether keys extracted)
  - Labs can direct testing resources to strategies that seem the most promising
  - Relatively inexpensive for product designers (no paperwork)
  - Often finds flaws

DPA evaluations: Black box testing

- Disadvantages
  - Results are inconsistent and highly dependent on lab skill
    - Vendors may pick "easy" labs = lab incentives may be backward
  - Relies on skills that are hard for labs to obtain & retain
  - Inefficient use of lab skills
  - Misses many problems
    - Countermeasures that pass "cookbook" testing may fail against adversaries who know the countermeasure design
    - Simply surviving black box testing does not provide positive, verifiable evidence of security
      - Like analyzing a cipher by looking at ciphertext only

DPA evaluations: Black box testing

Black box testing has major limitations, but often finds flaws and is useful for differentiating products with a moderate level of protection from those that are highly vulnerable.

DPA evaluations: Clear box testing

- Clear box = Evaluator has comprehensive information about the product's design
  - Necessary to obtain higher levels of confidence
  - Makes more efficient use of testing resources
    - Avoids trial & error guesswork to infer design
      - Security requirements allow lab to focus on validating, not hunting bugs
    - Places greater burden on designer (must document claims)
      - Lab does not need to understand every possible design strategy or DPA countermeasure, only the ones known to be present
- Conclusiveness of result depends on product's design
  - Product design may be unverifiable
    - Key issue: Design must enable effective validation to get higher levels of confidence…

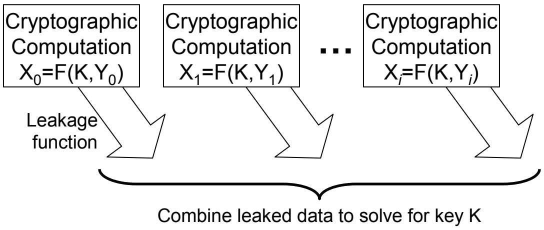
How can designers demonstrate security? How can evaluators validate these claims?

To obtain higher levels of confidence in designs, it must be possible to make verifiable statements that demonstrate the security of a product against DPA & related attacks.

… first some background on leakage …

DPA evaluations: Leakage functions

- Leakage functions
  - When device operates, it leaks some additional information beyond the digital inputs & outputs
    - The actual leaked information depends on the design
    - Significance of leaked info may be obvious (e.g., RSA SPA) or very difficult to interpret (e.g., if advanced statistics required)
    - Attacker observes the *leakage function* of the device state
      - Complex – not a function we are likely to ever know exactly



Cryptographic Computation $X_0=F(K,Y_0)$

Cryptographic Computation $X_1=F(K,Y_1)$

…

Cryptographic Computation $X_i=F(K,Y_i)$

Leakage function

Combine leaked data to solve for key K

DPA evaluations: Leakage rates

- Leakage rates
  - The information content of the leakage function is important
    - L = max info revealed to attacker (units: bits/operation)
    - Not necessarily an integral number of bits
  - The feasibility of obtaining effective security depends on L
    - If the leakage function reveals the whole key in every operation, the device is extremely insecure ($L >$ keysize)
    - If $L = 0$, side channel attacks are not a problem (no information ever is leaked)
  - No amount of testing can guarantee that $L=0$
    - Cannot prove that 10^{-6} bit/operation is not leaking somewhere
    - DPA statistics: Can pull keys from even very tiny leaks

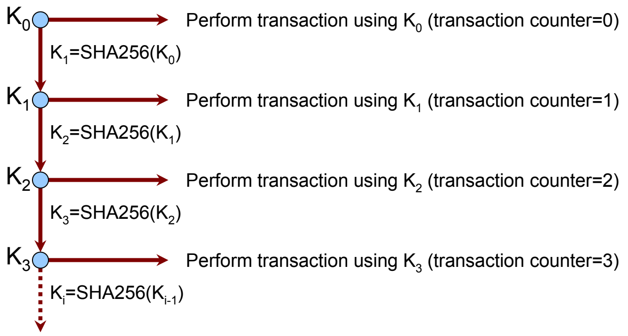
Tolerating leakage

- If labs cannot prove that $L=0$, what can we do?
  - Design crypto with the assumption that $L > 0$.
  - Provides hardware engineers with an achievable goal
    - Make hardware that leaks less information than the crypto assumes, with a suitable safety margin
  - Provides labs with a testable criteria
    - Is leakage rate less than the claimed amount, with a suitable safety margin

- Enables realistic assumptions about the hardware

Tolerating leakage: Protocol example #1

- Protocols with the required property can be easy to implement
  - Example: Hash 256-bit key with SHA256 between transactions
  - Hash destroys previously-leaked partial information about K_i



K_0 Perform transaction using K_0 (transaction counter=0)

K_1=SHA256(K_0)

K_1 Perform transaction using K_1 (transaction counter=1)

K_2=SHA256(K_1)

K_2 Perform transaction using K_2 (transaction counter=2)

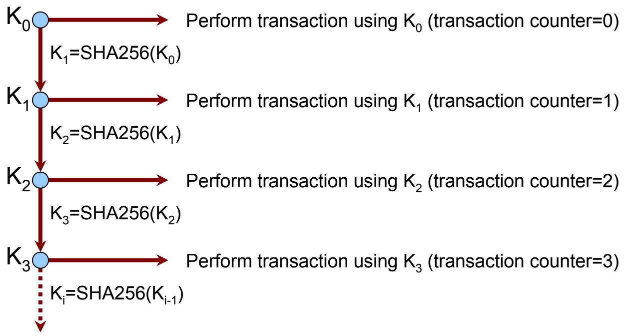
K_3=SHA256(K_2)

K_3 Perform transaction using K_3 (transaction counter=3)

K_i=SHA256(K_{i-1})

Tolerating leakage: Protocol example #1

- Cryptographic strength = $(256 - 2L_0 - L_1)$ bits
  - L_0 = max leakage per SHA256, L_1 = max leakage/transaction
  - L_0 counted twice: each K_i derived AND transformed with hash



K_0 Perform transaction using K_0 (transaction counter=0)

K_1=SHA256(K_0)

K_1 Perform transaction using K_1 (transaction counter=1)

K_2=SHA256(K_1)

K_2 Perform transaction using K_2 (transaction counter=2)

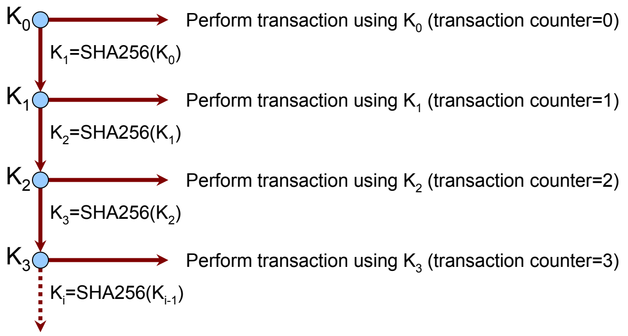
K_3=SHA256(K_2)

K_3 Perform transaction using K_3 (transaction counter=3)

K_i=SHA256(K_{i-1})

Tolerating leakage: Protocol example #1

- Design survives any reasonable leakage function
    - (Only requirement: does not interact with SHA256 update in a way that enables attackers to utilize information leaked before an update in attacking the value after the update.)



K_0 Perform transaction using K_0 (transaction counter=0)

K_1=SHA256(K_0)

K_1 Perform transaction using K_1 (transaction counter=1)

K_2=SHA256(K_1)

K_2 Perform transaction using K_2 (transaction counter=2)

K_3=SHA256(K_2)

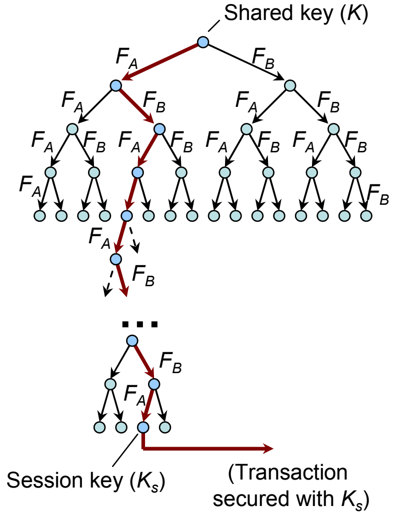
K_3 Perform transaction using K_3 (transaction counter=3)

K_i=SHA256(K_{i-1})

Tolerating leakage: Protocol example #2

- Step #1: Compute a shared nonce (H)
    - Example: Each contributes some data that has hashed
- Step #2: Derive a session key K_S from the nonce and an initial shared key *K*…
    - <u>Cannot</u> just hash with the nonce
        - Any information leaked from this hash would potentially compromise *K*.
    - To do this, we will use two update functions F_A and F_B as shown on the next slide
        - Example of F_A & F_B:
            - F_A = Concatenate with 0 then hash
            - F_B = Concatenate with 1 then hash

Tolerating leakage: Protocol example #2



Apply F_A (bit 0 of H is 0)

Apply F_B (bit 1 of H is 1)

Apply F_A (bit 2 of H is 0)

Apply F_A (bit 3 of H is 0)

Apply F_A (bit 4 of H is 0)

Apply F_B (bit 5 of H is 1)

Apply F_B (bit 126 of H is 1)

Apply F_A (bit 127 of H is 0)

Max leaks:
- K: $2L_0$
- Intermediates: $3L_0$
- Each K_S: L_0+L_1

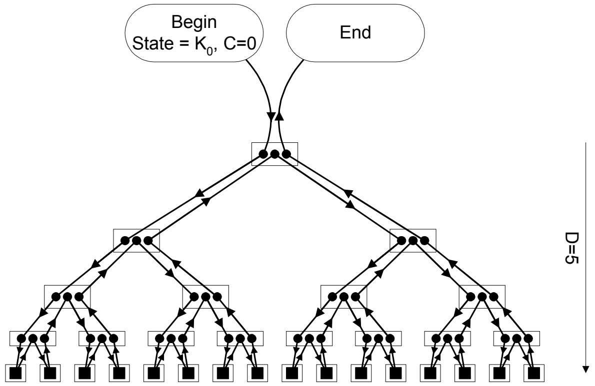
(L_0=leak from F_A,F_B)
(L_1=leak from trans.)

Tolerating leakage: Protocol example #2

- The second example has slightly different properties from the first
  - The first can tolerate virtually any information leaking, up to the threshold amount, per transaction
    - Assumes a trusted server (e.g., not subject to DPA)
  - The second requires a limit on the amount of information leaked per computation
    - Can use for point-to-point protocols (no server involved)
    - Attacker can see the same computation multiple times
    - Requires caution with randomizing countermeasures

Tolerating leakage: Variations (symmetric)

- Other variations possible for symmetric crypto
  - Example: Save RAM with reversible update operations, have O(log(N)) run-time for client/server protocols, etc.



Tolerating leakage: Variations (public key)

- Key updates also possible for public key crypto
  - Typical approach: Compute private key operation in a modified way, but maintain compatibility with public keys
  - Challenge: Update functions tend to be less effective than the example with SHA
    - Evaluator must carefully assess the feasibility that information leaked prior to an update could remain useful for the adversarary
  - Challenge: Computational complexity tends to be higher
    - Public key operations take longer, consume more power, have greater variation – all these tend to lead to higher leakage

Validation strategy

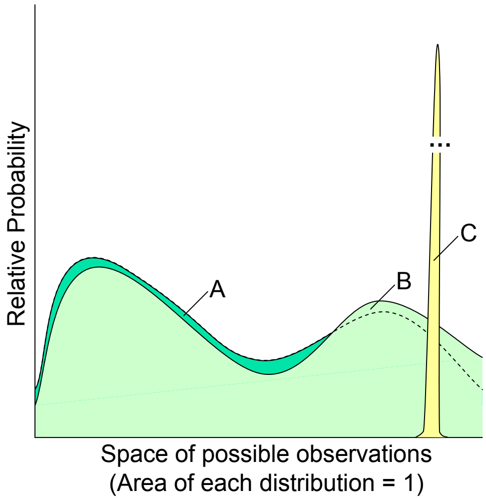
- If a device's protocols can tolerate some leakage, the validation lab has a feasible job:
  - Verify that the protocols have the claimed properties
    - Conventional crypto evaluation
  - Verify that the hardware leaks less than the survivable leakage, with a suitable safety margin
    - Hardware analysis
- Contrast: If protocols require zero leakage, validation is likely to be impossible (if high assurance is required)

Analyzing leakage rates

- Typical process:
  - Characterize device with countermeasures disabled
    - (At least randomizing countermeasures should be off)
  - Characterize countermeasures
  - Estimate overall leakage rate with countermeasures enabled
- Result is usually one of the following:
  - Device leaks massively (fails)
  - No major leaks (none detected or insignificant)
  - Inconclusive (usually due to countermeasures)

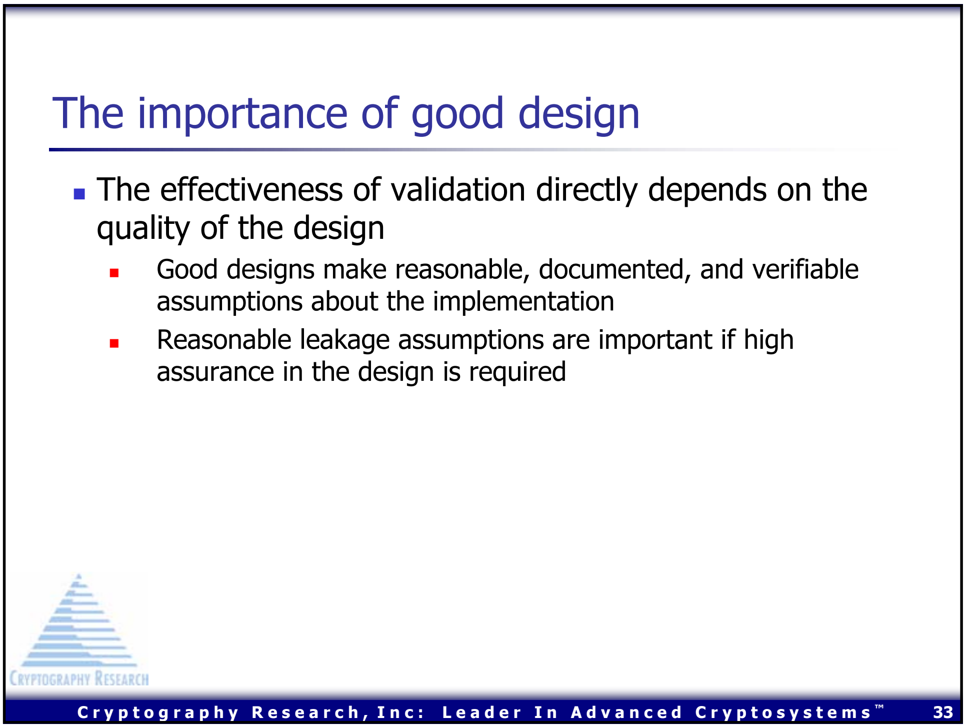
Leakage assessment

- The leakage rate reflects the probability distribution curves associating observed leakage with keys/data
    - Logarithm of overlap



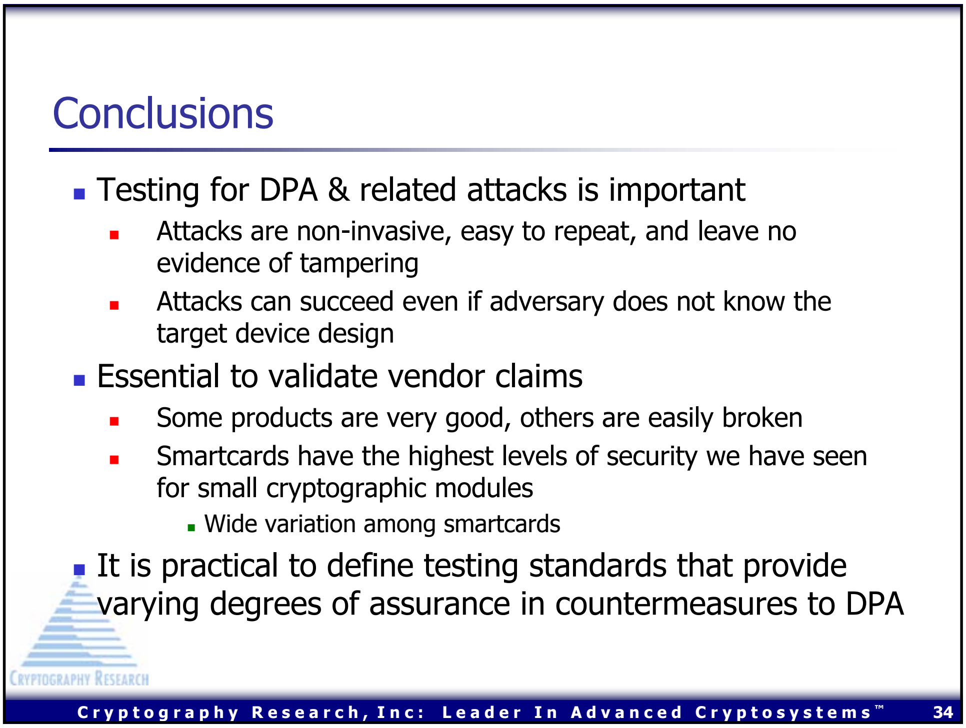
Recap: High-assurance design strategy

- Engineering approach:
    - Build crypto to tolerate some leakage
    - Get signal/noise ratio small
        - Filtering, balancing, randomization…
- Criteria for implementation success:
    - Actual leakage rate << tolerable leakage rate
    - Produce compelling documentation demonstrating secuirty



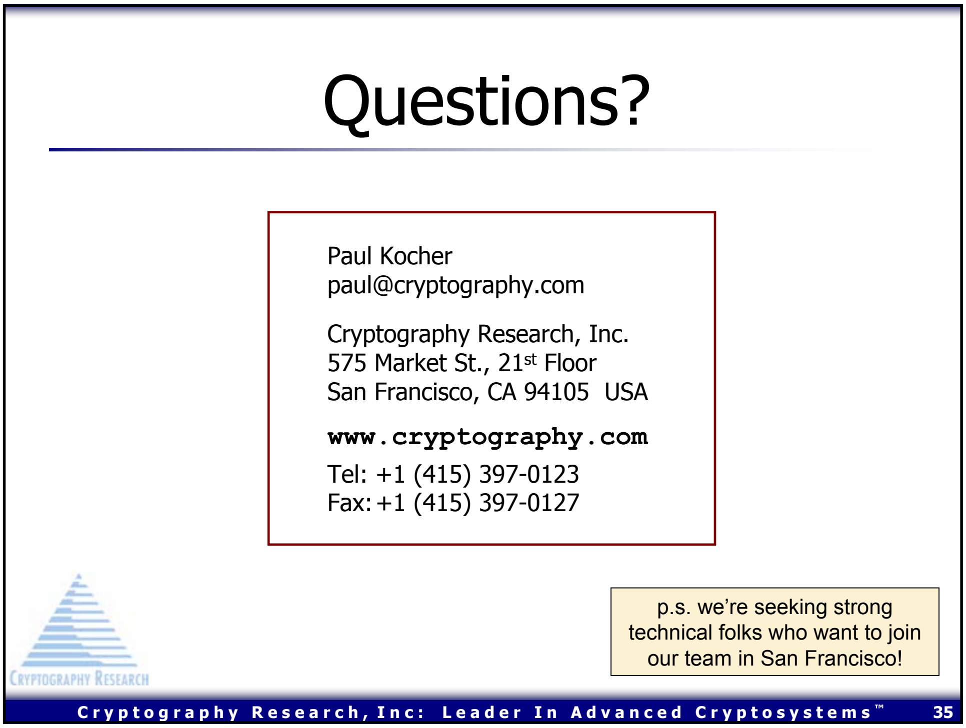
The importance of good design

- The effectiveness of validation directly depends on the quality of the design
  - Good designs make reasonable, documented, and verifiable assumptions about the implementation
  - Reasonable leakage assumptions are important if high assurance in the design is required



Conclusions

- Testing for DPA & related attacks is important
  - Attacks are non-invasive, easy to repeat, and leave no evidence of tampering
  - Attacks can succeed even if adversary does not know the target device design
- Essential to validate vendor claims
  - Some products are very good, others are easily broken
  - Smartcards have the highest levels of security we have seen for small cryptographic modules
    - Wide variation among smartcards
- It is practical to define testing standards that provide varying degrees of assurance in countermeasures to DPA



Questions?

Paul Kocher
paul@cryptography.com

Cryptography Research, Inc.
575 Market St., 21st Floor
San Francisco, CA 94105 USA

www.cryptography.com

Tel: +1 (415) 397-0123
Fax: +1 (415) 397-0127

p.s. we're seeking strong technical folks who want to join our team in San Francisco!