Inspection Resistant Memory: Architectural Support for Security from Physical Examination

Jonathan Valamehr, Melissa Chase, Seny Kamara, Andrew Putnam, Dan Shumow, Vinod Vaikuntanathan and Timothy Sherwood

Presenter: Jiyong Yu

Table of Content

Motivation: Secret Key Sharing

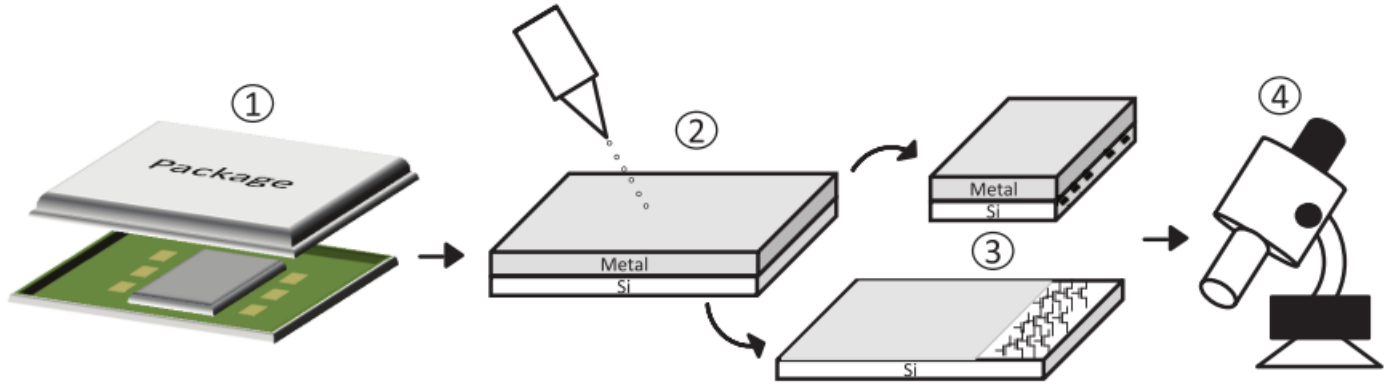
- A real world example:
  - Network-less attestation for console gaming system
  - The same secret key is shared between all controllers

Motivation: Secret Key Sharing

- “Break once, run anywhere”

Background: Physical Inspection Attack

- Type of attacks
  - Passive attacks
  - Intrusive attacks
    - Powered attacks
    - Un-powered attacks (which this paper focuses on)



- Error exists in these attack techniques, but typically not a problem from the attacker's point of view.

Attack Model

- Attacker gets physical access to arbitrary amount of devices, which share an identical secret key set.
  - Number of keys: n
  - Length of a single key: k
- Every bit has probability p to be measurable and leak its true value (p!= 1)
- Whether a bit is measurable is independent of other bits

Attack Model

- The Goal of our solution:
  - Every device does not leak a single bit to the attacker
  - Secret keys are accessed by device without large overhead

Solution: Inspection Resistant Memory

- General idea:
  - Encode every secret key bit with large number of bits (c).
  - Decrease the probability of learning a single bit from p->1 to p^c-> 0

- Psucc
  - Adversary infers anything from one secret key (even a single bit)
  - In this paper it's 1/1 billion

Baseline: No encoding

- Psucc = $1-(1-p)^k$
- Adversary on average learn p*k bits from each secret key

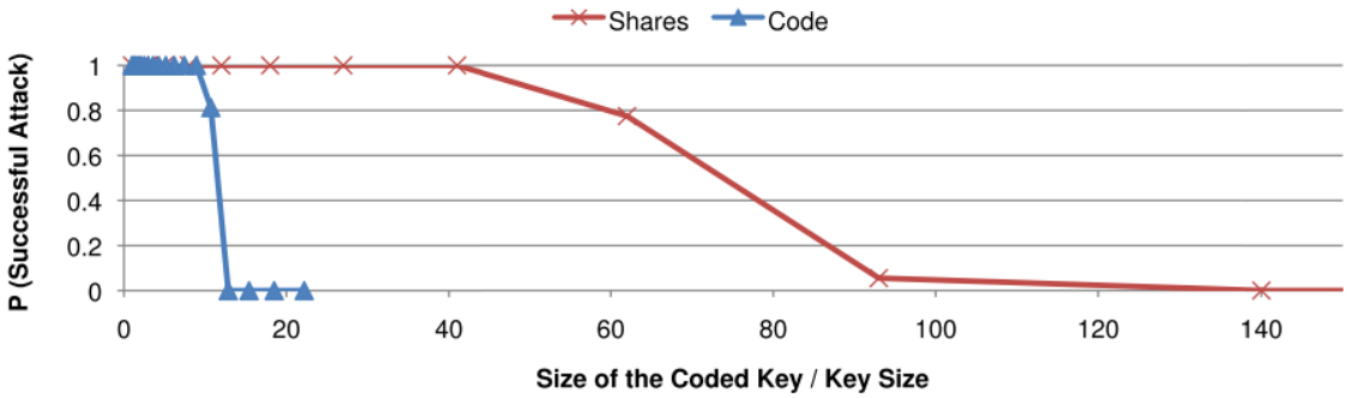
Option 1: Secret-sharing Based Method

- For every secret bit x, generate s random bits (r_1, ..., r_s)
- An extra data bit to recover the value of x by

  $r_{s+1} = x \oplus r_1 \oplus r_2 \oplus \ldots \oplus r_s$
- Every single secret bit x is encoded by [r_1, r_2, …, r_s, r_{s+1}], and computed by

  $x = r_1 \oplus r_2 \oplus \ldots \oplus r_s \oplus r_{s+1}$
- Probability of learning a single bit:

  $Psucc = 1 - (1 - p^{s+1})^k$



Option 2: Code Based Method

- For every secret key $x_1x_2\ldots x_k$, generate

  s random bits (r_1, ..., r_s) and k random sets $T_1, T_2, \ldots, T_k \subseteq \{1, 2, \ldots, s\}$

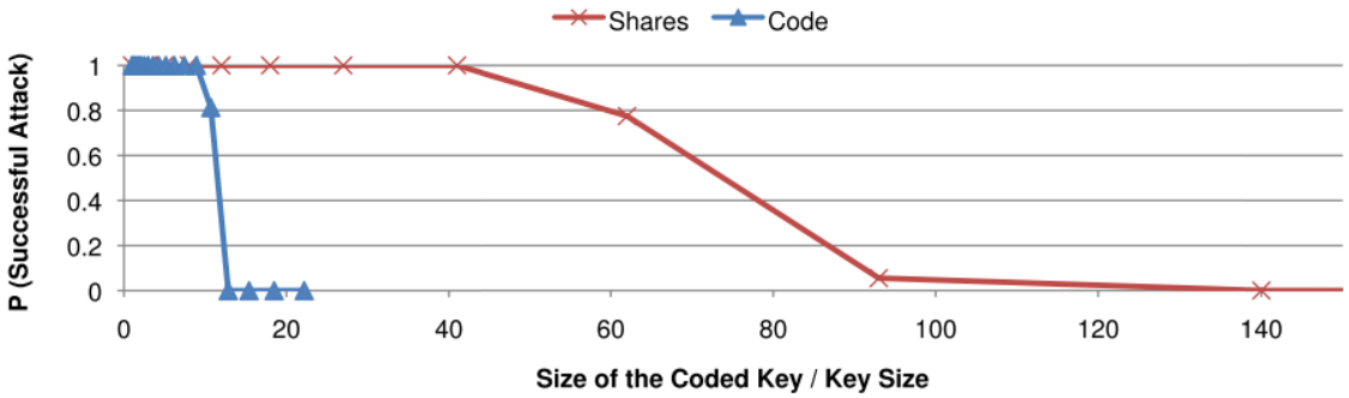
- k extra bits to recover the value of $x_1, x_2, \ldots, x_k$ by

  $r_{s+1} = x_1 \oplus [\oplus_{i \in T_1} r_i]$, ... , $r_{s+k} = x_k \oplus [\oplus_{i \in T_k} r_i]$

- Therefore every key with length k is encoded by s random bits, (s × k) matrix T and k extra bits

- Probability of learning a single bit:

$$Psucc \leq 2^{-2\cdot(0.01)^2 ps} + \frac{2^{k+1}}{2^{(0.99-p)\cdot s}}$$



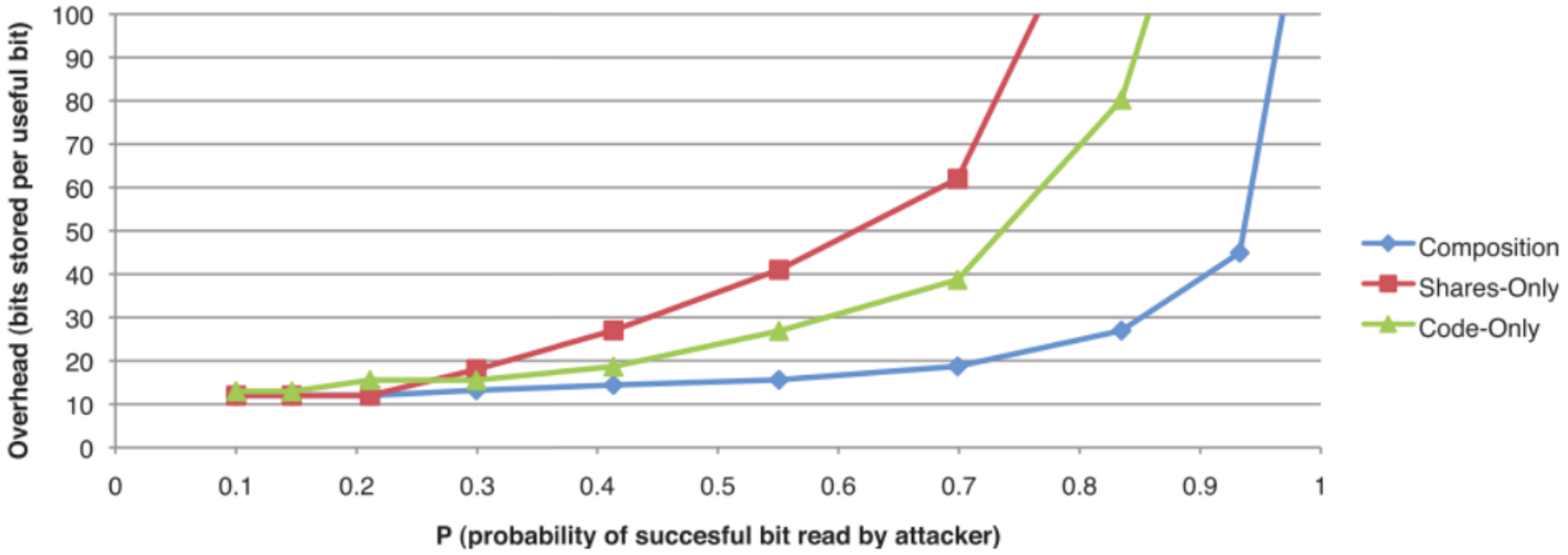
Option 3: Hybrid Method

- Both option 1 and option 2 have large overhead for storing encoded keys
  - Secret-sharing based: (s+1) * k for every key
  - Code based: s + k + s*k/n for every key

Option 3: Hybrid Method

- Step 1: Encode the key with code based scheme with smaller matrix T

- Step 2: Encode the key with secret sharing based scheme



Option 4: Dynamic Matrix Creation

- Replace matrix T (main source of storage overhead) with cryptographic hash function
- T_i = SHA2(seed ◦ i ◦ 1) ◦ SHA2(seed ◦ i ◦ 2) ◦ . . . ◦ SHA2(seed ◦ i ◦ s/256)
- Secrecy is guaranteed by the randomness of SHA-2 hash function
- The encoding and decoding is pipelined to eliminate the latency

Single seed (anti-fuse memory)	0.044 mm^2
Single key (SRAM)	0.084 mm^2
SHA generator	0.46 mm^2
Computational logic	0.08 mm^2
Total (for 1 key)	0.589 mm^2
Total (for 128 keys)	6.18 mm^2

Conclusion

- Physical inspection attack with nonzero measurement error
- Four cryptographic strategies to prevent adversary learning any information from the secret key
- Storage vs. Secrecy tradeoff

Discussion Questions

- No explanation on why hybrid method is better than the first two options
- No apple-to-apple comparison between option 3 and option 4

- Examples besides attacking Xbox?
- Why do you share the same secret key?