Space-Efficient Block Storage Integrity

Alina Opera, Mickael Reiter, Ke Yang

CMU and Google

NDSS 2005

Lisa Johansen

April 19, 2007

The problem

"… two years ago the IEEE Security In Storage Working Group (SISW) announced a call for algorithms for block level encryption. Among the requirements was that the new encryption algorithm be length preserving, so that the block boundaries do not shift or need to be adjusted as a result of encryption."

- Research has already been presented and considered for standardization
- So why are we here?
- Hint: 3 security buzz words
  - Confidentiality is done with the crypto
    - Lengh-preserving, pseudorandom permutation (tweakable enciphering scheme)
  - Availability is impossible

Integrity

- But this paper gives us a definition(s)
- Integrity is:
  - "if a client returns block *B* in response to a read request for address *a*, then the client previously wrote *B* to *a*."
  - "if the client returns block *B* in response to a read request for address *a*, then *B* is the content most recently written to address *a*."
- Difference?
- Which one is harder?

Solve it

- How is integrity solved conventionally?
  - Why won't that work?
- With the added restriction, how do they do it?
  - Time-space tradeoff
  - In order to get out of this trap, they need external help
    - "Blocks written to disk do not look random in practice"
    - "If an adversary tries to modify ciphertexts encrypted with a tweakable enciphering scheme, the resulting plaintext looks random with very high probability"
  - But, ya got trouble, my friend, right here, I say, trouble right here in River City
    - "do not look", "looks random", "high probability"
  - Prove it

So how did they do it

- They propose 2 methods
  - Method 1: The method that isn't feasible
    - This one solves the harder integrity problem
    - Requires storage
  - Method 2
    - Solves the easier integrity problem
    - Look at the data and determine if it is considered random
    - If so, store the hash on your computer
    - If not, do nothing
    - Upon reading, if the string is random and there is no hash stored, then you know that it has been tampered with

Justification

- How can you reasonably store hash values
  - Can you?
  - If you are only storing a few of them (How few is a few?)

Methodology/Results (Method 2)

- Tested the “randomness” of 100,000 1024-byte random blocks
- 8-bit test results in entropy between 7.73-7.86
- Set the threshold to 7.7 and only 2% of real data are classified as random
  - Thus we are only holding 2% of the hash values for the entire data set
- Performance was decent too
  - 19% overhead compared to 44% overhead for hashing

Mixture of the two

- Protect against replay
- Keep a counter for anything written more than once
- Integrate that value into the “tweak” encryption
- Storage required for this method lies between the other two:
  - 16.262 MB, .022 MB, .351 MB