Technical Disclosure Commons

Defensive Publications Series

December 03, 2018

Generating cryptographic initialization vectors from SSD wear metrics

Michael William Paddon

Generating cryptographic initialization vectors from SSD wear metrics

ABSTRACT

Data encryption on storage devices is achieved by the application of a suitable cipher mode. Many commonly used cipher modes require an initialization vector (IV). An IV is not secret, yet it must not be reused with the same encryption key in order to preserve confidentiality. A storage device can generate and store a unique IV alongside each encrypted block; however, this capability is not commonly available in mass market implementations. Instead, encrypted storage devices commonly use cipher modes that don't require an IV, e.g., XTS. However, these have well-known vulnerabilities. This disclosure presents techniques that deterministically derive IVs for block encryption such that they are not stored, yet preserve the property of never being reused.

KEYWORDS

- cryptography
- block encryption
- initialization vector
- cipher mode
- XTS
- SSD wear metrics

BACKGROUND

Data encryption on storage devices is achieved by the application of a suitable cipher mode in order to forestall exploits of well-known vulnerabilities. Many commonly used cipher modes require an initialization vector (IV). An IV is not secret, but must not be reused with the same encryption key in order to preserve confidentiality.

A storage device can generate and store a unique IV alongside each encrypted block; however, this capability is not commonly available in mass market implementations. Instead, encrypted storage devices commonly use cipher modes, such as XTS (XEX-based tweaked-codebook with ciphertext stealing), which do not require an IV. These have well-known weaknesses that IV-based cipher modes avoid.

DESCRIPTION

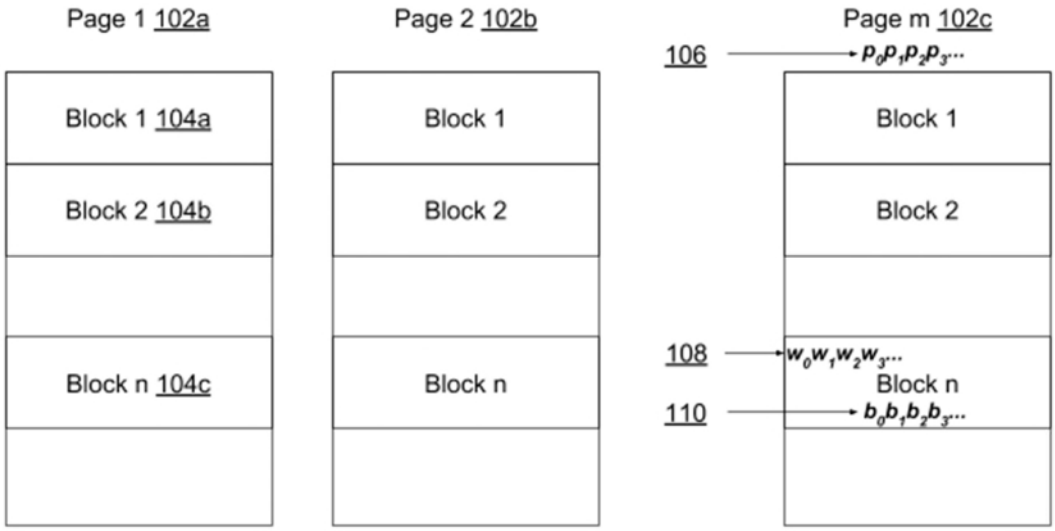

Fig. 1: Organization of a NAND memory

Fig. 1 shows typical organization of NAND-based non-volatile memories, such as those used in solid-state drives (SSD). These memories comprise several pages (102a-c), a page itself comprising several blocks (104a-c). A page is indexed by an address (106) expressed as a binary string $p_0p_1p_2p_3\cdots$. A block within a page is indexed by an address (108) expressed as a binary string $b_0b_1b_2b_3\cdots$.

Data is written in units of blocks and erased in units of pages. A block may be written exactly once after its page is erased, and may not be rewritten until the page is erased again. In

order to maximize device service life, wear metrics are tracked for use by wear-levelling algorithms. One of these metrics is the number of erase cycles for each page. Such a wear metric (108) is tracked on a per-block basis using a binary counter $w_0w_1w_2w_3\cdots$.

Thus, each block has associated with it a page address, a block address, and a counter of page-erase cycles. By combining these three numbers - the page address, the block offset, and the counter of page erase cycles- the techniques of this disclosure generate a unique IV for each block when it is encrypted. For example, the IV for Block *m*, Page *n* of Fig. 1 is

$$p_0p_1p_2p_3\cdots b_0b_1b_2b_3\cdots w_0w_1w_2w_3\cdots.$$

Example: A device has 2^{24} pages, 2^8 blocks per page, and an endurance of one million ($< 2^{20}$) program/erase cycles. Per the techniques, a 52-bit IV is constructed by concatenating the 24-bit page address, the 8-bit block offset and the 20-bit write counter.

The initialization vector, as generated by the techniques herein, is unique to the location and number of erasures of the underlying block, and hence, is never reused. This IV may be used to initialize a cipher mode such as propagating cipher block chaining (PCBC) so that the block may be encrypted prior to writing. The IV need not be stored because it can be trivially reconstructed. If a wear levelling operation rewrites a block to a different page, then the data is decrypted and then re-encrypted with the IV unique to the new location.

In some implementations, a block encryption system is implemented in the firmware of a SSD. In such cases, the firmware has direct access to the physical page addresses, block offsets, and erase cycle counts to construct IVs whenever a block is encrypted and written. In some implementations, the encryption performed by an operating system. In such cases, the device provides access to physical page addresses, block offsets, and erase cycle counts in order for the

operating system to calculate IVs. The device also delegates block rewriting for wear-levelling purposes to the operating system.

SSDs are used both in server computers as well as client devices, e.g., portable devices such as smartphones, tablets, laptops, wearable devices, etc. Security best practices require that the storage device of such devices be encrypted. The described techniques provide more secure cipher modes, offering better protection for encrypted data. The ability to deterministically derive IVs allows the cipher modes to be utilized without an increase in storage costs.

CONCLUSION

This disclosure presents techniques that deterministically and at no extra cost, derive initialization vectors for block encryption such that they are not stored, yet preserve the property of never being reused. The techniques have minimal storage overhead, improve confidentiality, and are usable in portable, cloud-based, and other types of storage devices.