Cache Timing Analysis of HC-256

Erik Zenner

Technical University Denmark (DTU)
Institute for Mathematics
e.zenner@mat.dtu.dk

SASC 2008, Feb. 14, 2008

Cache Timing Attacks

Our Attack Model

3 Attacking HC-256

4 Conclusions

Outline

- Cache Timing Attacks
- Our Attack Model
- Attacking HC-256
- 4 Conclusions

Cache Workings (Simplified)

Motivation: Loading data from cache is much faster than loading data from RAM (by a factor of ≈ 10).

Working principle (simplified): Let n be the cache size. When data from RAM address a is requested by the CPU, proceed as

- Check whether requested data is at cache address $(a \mod n)$.
- If not, load data into cache address (a mod n).
- Load data item directly from cache.

Similarly for writing data to RAM.

Idea: Data that is used now will more likely be used again in the future.

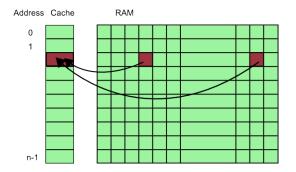
⇒ Keeping copies in cache reduces the average loading time.

follows (simplified):

Cache Eviction (Simplified)

Problem: Cache is much smaller than RAM.

Consequence: Many RAM entries compete for the same place in cache.



Handling: New data overwrites old data (First in, first out).

Sample Attack Setting

Starting point: Reading data is faster if it is in cache (cache hit), and slower if it has to be loaded (cache miss).

Sample attack (prime-then-probe): Imagine two users A and B sharing a CPU. If user A knows that user B is about to encrypt, he can proceed as follows:

- A fills all of the cache with his own data, then he stops working.
- B does his encryption.
- A measures loading times to find out which of his data have been pushed out of the cache.

This way, A learns which cache addresses have been used by B.

Practical Difficulties

For didactical reasons, we worked with a simplified cache model.

Real-world complexities include:

- Cache data is not organised in bytes, but in blocks.
 - \Rightarrow We do not learn the exact index, but only some index bits.
- Other processes (e.g. system processes) use the cache, too.
 - \Rightarrow We can not tell "encryption" cache accesses apart from others.
- Timing noise disturbs the measurement.
 - ⇒ Not all slow timings are due to cache misses.
- Cache hierarchy is more complex.
 - ⇒ Several layers of cache, several cache blocks for each memory block.

Nonetheless, as it turns out, these difficulties can be overcome in practice (Bernstein 2005, Osvik/Shamir/Tromer 2005, Bonneau/Mironov 2006).

Outline

- Cache Timing Attacks
- Our Attack Model
- Attacking HC-256
- 4 Conclusions



Motivation

When setting up the attack model, our motivation was as follows:

- Abstract away technical details of the cache timing attacks.
- Have a model that is suitable for **designing** cryptographic algorithms.
- → Model has to be rather generous w.r.t. the attacker's abilities.

Attack Model

Available Oracles:

The adversary can use any of the following functions / oracles at will:

- Key_setup: Sets up a new cipher instance. Does not return any output.
- Key_Cache_Access: Returns an accurate list of the cache blocks accessed while running Key_setup.
- **IV_setup(N)**: Resets the cipher instance with initialisation vector *N*, as chosen by the attacker. Does not return any output.
- IV_Cache_Access(N): Returns an accurate list of the cache blocks accessed while running IV_setup(N).
- **Keystream(i)**: Returns the keystream block *i*.
- KS_Cache_Access(i): Returns an accurate list of the cache blocks accessed while running Keystream(i).



Discussion

Clarification:

This model is very generous towards the adversary. In the real world, he may not be able to

- observe every encryption operation,
- get a precise list of cache block accesses,
- choose the IV, or
- observe the keystream.

This means that cryptanalytic results obtained in this model are not necessarily attacks in the real world.

But: As with all other design criteria in cryptography, the designer should not rely on things that the adversary *might* not be able to do!

Outline

- Cache Timing Attacks
- Our Attack Model
- 3 Attacking HC-256
- 4 Conclusions



About HC-256

- Stream cipher (FSE 2004), eStream software finalist.
- Key/IV: 256 bit each.
- Inner State: Two tables, 1024 · 32 bit each.
 - \Rightarrow 65, 536 bits of inner state.
- One Round:
 - Update one of the tables.
 - Produce 32 bit of output.
- Performance:
 - Designed for software.
 - Slow key/IV setup (due to table initialisation).
 - Fast keystream generation.

Sketch of the Attack

The attacker uses the following oracles:

- 6148 calls to KS_Cache_Access(i).
- 2048 calls to Keystream(i).

Then he uses three layers of guess-and-verify to determine the inner state:

- Determine the block access ordering.
- Quess-and-eliminate step.
- Guess-and-determine step.

Step 1: Block Access Ordering

Adversary makes 6148 calls to **KS_Cache_Access(i)** and maps the resulting observations to inner state bits.

Problem: Mapping of cache accesses to state variables.

- Each oracle call: 5 cache accesses, e.g.:
 001011xxxx, 011100xxxx, 010011xxxx, 101101xxxx, 111110xxxx
- How to assign them to internal state variables? E.g.: $(00||P_{13}^{(7..0)}),\,(01||P_{13}^{(15..8)}),\,(10||P_{13}^{(23..16)}),\,(11||P_{13}^{(31..24)}),\,(P_{22}\oplus P_{-998})^{(9..0)}$

Solution: Simple internal consistency test works with high probability!

End of step 1:

For almost all inner state words, we know all upper half-bytes.

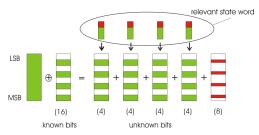
 \Rightarrow 2¹⁶ candidates for each inner state word.



Step 2: Guess-and-Eliminate Step

Adversary makes 2048 calls to **Keystream(i)** and uses an internal equation to further reduce the number of candidates.

Problem: Carry bits complicate the equation.



Solution: Guess the carry bits, too.

End of step 2:

28 remaining candidates for each inner state word.

 \Rightarrow Store in a table (size \approx 3 MByte).

Step 3: Guess-and-Determine Step

Adversary uses guess-and-determine strategy with a different equation to determine the rest of the inner state.

Problem: Many bits have to be guessed before verification becomes possible.

- Guess 2⁴⁸ assignments, obtain 32 verification bits.
 ⇒ 2¹⁶ assignments remain.
- Solution: Guesses start to overlap.
 - Search tree grows less fast than in the beginning.
 - After some steps, search tree starts to shrink.
 - Maximum tree width: 2⁶⁴ guesses.

End of step 3:

Full inner state for one point in time has been recovered.



The Attack in a Nutshell

Requirements:

- 6148 precise cache timing measurements.
- 2¹⁶ known plaintext bits.
- Computational effort corresponding to testing $\approx 2^{55}$ keys.
- $\bullet \approx 3$ MByte of memory.

Result:

- Reconstruction of full inner state.
- Allows to create arbitrary output bits.
- Also allows to reconstruct the key.

Outline

- Cache Timing Attacks
- Our Attack Model
- Attacking HC-256
- 4 Conclusions



Practical Relevance

Question: So is HC-256 broken?

Answer: Not unless you already stopped using AES for security reasons.

- Attack uses very strong assumptions.
- AES would be completely broken under the same assumptions.

But: Relevance of cache timing attacks is currently an open issue.

- A distinguisher using 2⁶⁰ known plaintexts is sufficient to discard a cipher.
- How about a key recovery attack using $\approx 6,000$ precise cache timings?

Other eStream Software Finalists (1)

Cipher	Tables	Relevant
CryptMT	none	-
Dragon	Two 8 × 32-bit S-Boxes	†
HC-128	Two 9×32 -bit tables	
HC-256	Two $10 imes 32$ -bit tables	†
LEX-128	One 8×8 -bit S-Box (ref. code)	
	Eight 8×32 -bit S-Boxes (opt. code)	†
NLS	One 8 × 32-bit S-Box	†
Rabbit	none	-
Salsa-20	none	-
Sosemanuk	One 8×32 -bit table,	
	eight 4×4 -bit S-Boxes (ref. code)	
	One 8×32 -bit table (opt. code)	†

(† = potentially vulnerable)



Other eStream Software Finalists (2)

- **Expectation:** When starting analysis in the above (generous) model, we expected most eStream candidates to break down completely.
- **Surprise:** Most candidates seem to withstand analysis even in the generous model surprisingly well, even though they were not designed to that purpose (exception: Salsa).
- Work on cryptanalysis is still in progress.
 - No one-size-fits-all attack
 - Different ciphers pose different problems
 - Individual analysis required
- **Guess:** Attacks are possible, but require some thought (exception: LEX).

And finally...

Questions? Comments?

