(a) Algorithm Description

1. It extracts the SET command from the XDP payload.
2. SET command founded, try to figure the index of the Memcached key in payload.
3. If the key is found, process the key with hash function.
4. If the hash matches the calculated hash, the corresponding cache entry in the map should be invalidated

(b) REX Implementation

```
1  let set_iter = payload                                          // 1
2   .windows(4) // 4 chars as a slice
3   .enumerate()
4   .filter_map(|(i, v)|
5      if v == b"set " { Some(i) } else { None }
6  ); // found the SET command
7  for index in set_iter {                                         // 2
8    ... // set payload index via SET command
9    payload                                                       // 3
10    .iter()
11    .take_while(|&&c| c != b' ')
12    .for_each(|&c| {
13      ... // process the key with hash func
14    });
15  ... // invalidate Memcached cache entry                        // 4
16 }     // if the hash matches
```

(c) Original BMC Implementation

```
1  #pragma clang loop unroll(disable)                              // 1
2  for (unsigned int off = 0; off < BMC_MAX_PACKET_LENGTH &&
4   payload + off + 1 <= data_end; off++) {
5   if (set_found == 0 && payload[off] == 's' &&
6      payload + off + 3 <= data_end &&
7      payload[off + 1] == 'e' && payload[off + 2] == 't') {
9     ... // move offset after the SET command
10    set_found = 1;
11  } else if (key_found == 0 && set_found == 1 &&                  // 2
12              && payload[off] != ' ') {
13    if (payload[off] == '\r') {
14      set_found = 0; key_found = 0;
16    } else {
17      ... // found the start of the key
18      key_found = 1;
19    }
20  } else if (key_found == 1) {                                    // 3
21    if (payload[off] == ' ') {
22      ... // found the end of the key
23      set_found = 0; key_found = 0;
25    } else {
26      if (...) {...} // process the key with hash func
27    }      // invalidate Memcached cache entry                   // 4
28  }        // if the hash matches
29 }
```

- In the original implementation, BMC must write awkward code to bypass the verifier. Specifically, dedicated check conditions for BMC_MAX_PACKET_LENGTH (**L2**) are aiming to minimize the number of jump instructions due to verifier's specific
- restrictions regarding these instructions. However, in the Rex, such checks have become redundant because the inherent feature of slice could help confine data_end(**L4**). Consequently, the four levels of nesting (**L4,20,25,26**) is reduced by converting a `for-loop` (**L2**) with intricate conditions (**L4**) into a clean chain of higher-order functions with closures (**L1, L9**) .