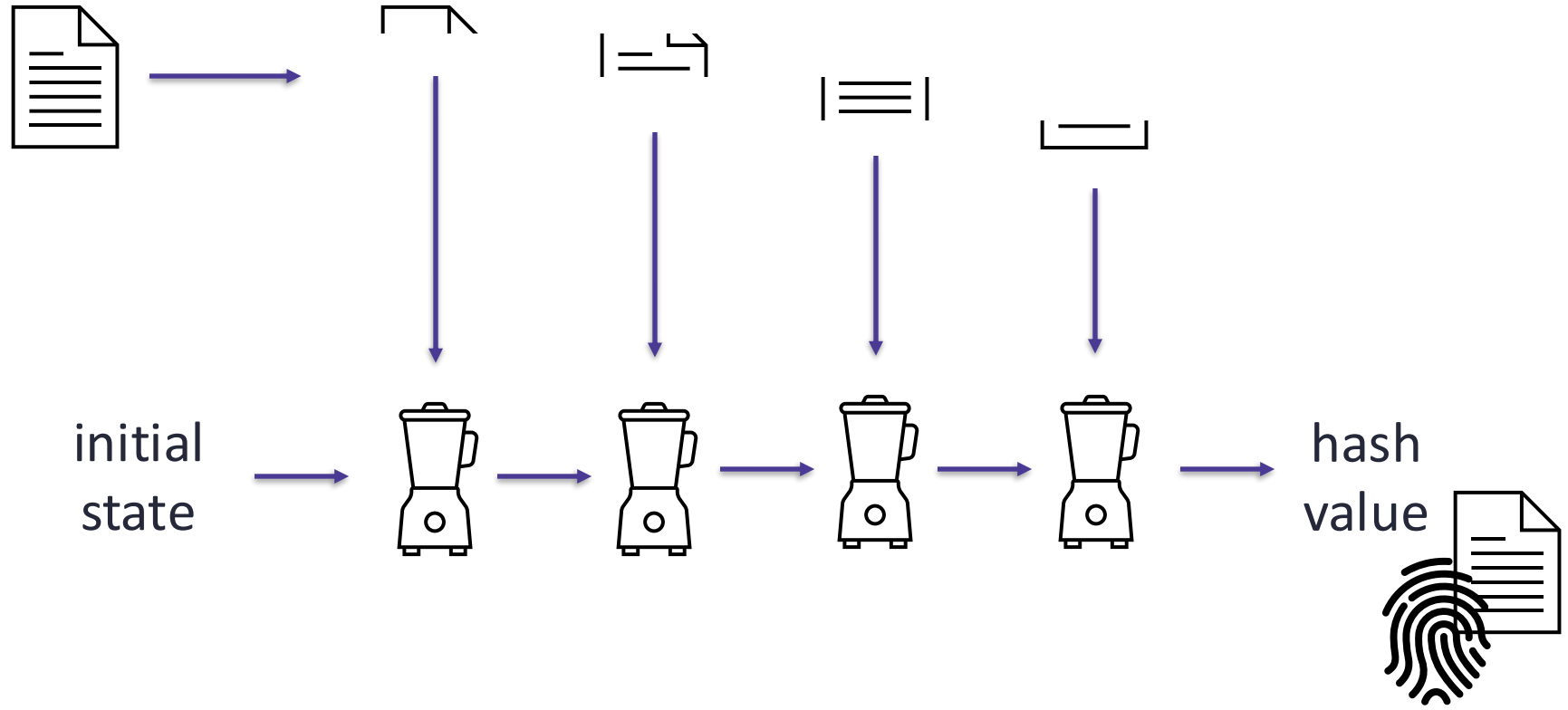
The Road from Academic Research to OpenSSL Contributions

Nicky Mouha

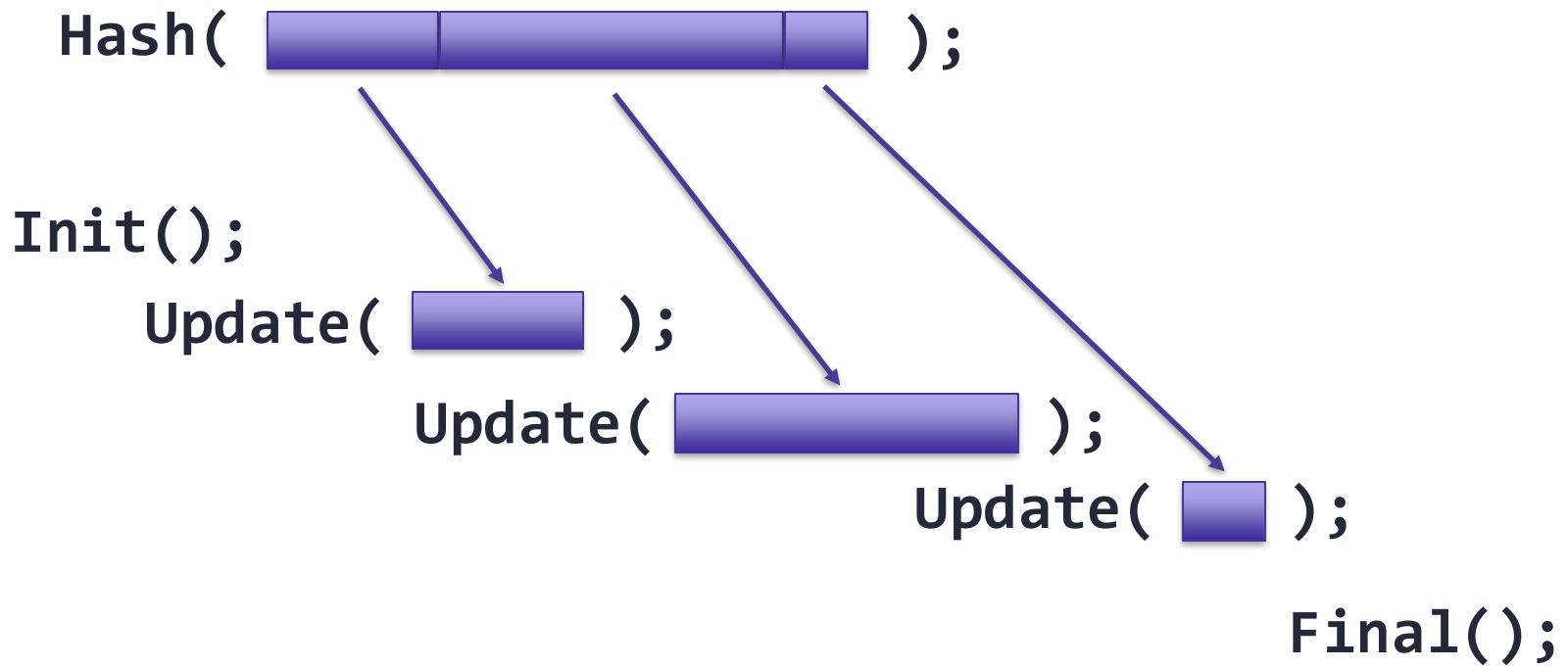
OpenSSL Conference 2025

October 8, 2025

Iterated Hash Functions



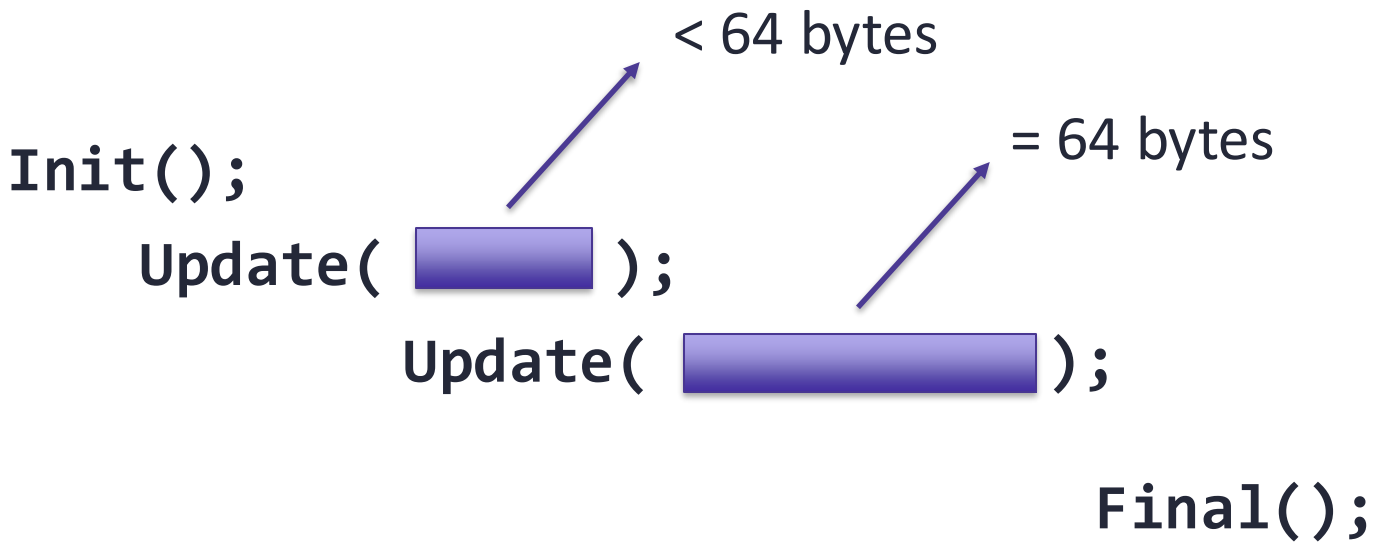
Two Common Hash Function Interfaces



- Q: Where/when are they used?

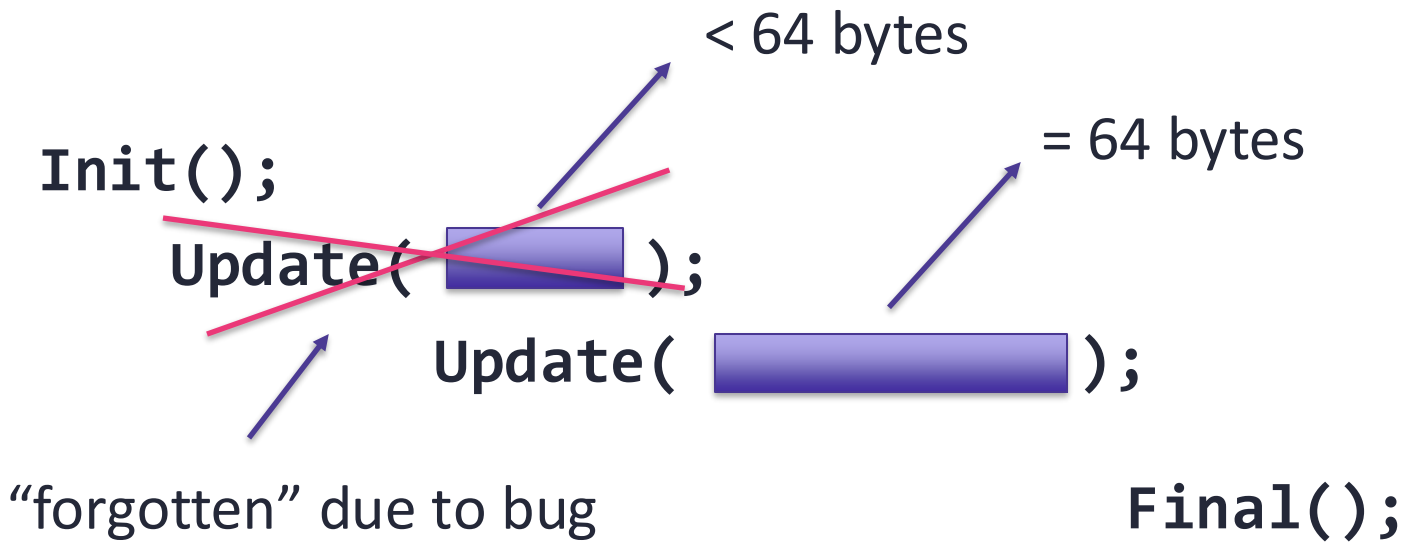
BLAKE Bug

- Appeared in 2008, disclosed in 2015
- BLAKE-256: process message in blocks of 64 bytes



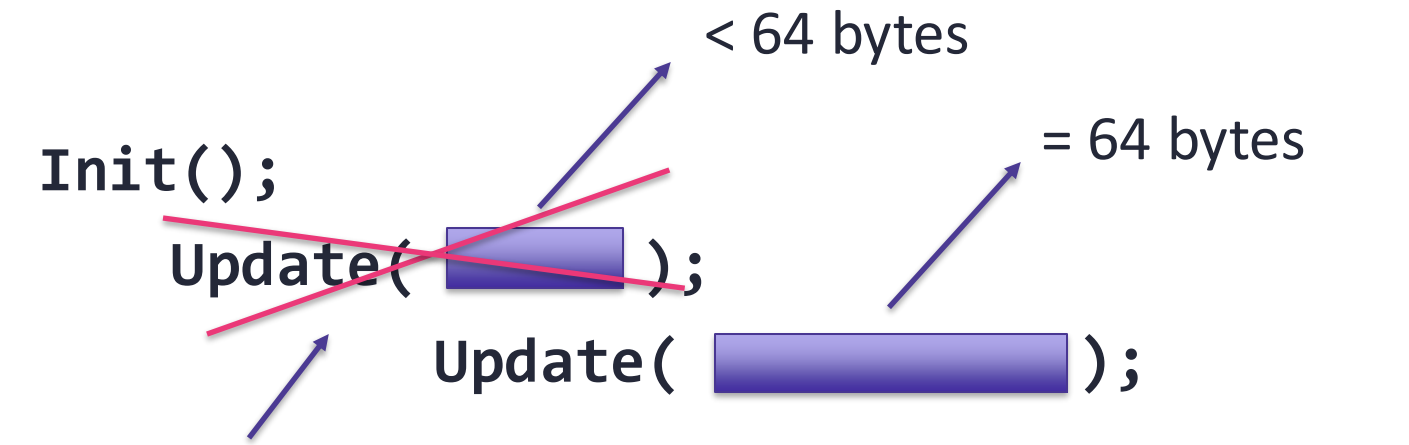
BLAKE Bug

- Appeared in 2008, disclosed in 2015
- BLAKE-256: process message in blocks of 64 bytes



BLAKE Bug

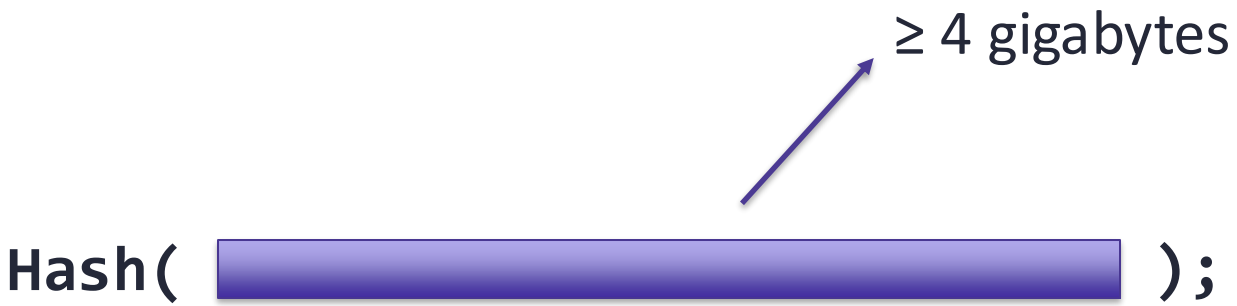
- Appeared in 2008, disclosed in 2015
- BLAKE-256: process message in blocks of 64 bytes



"second preimage"
(modified message, same hash)

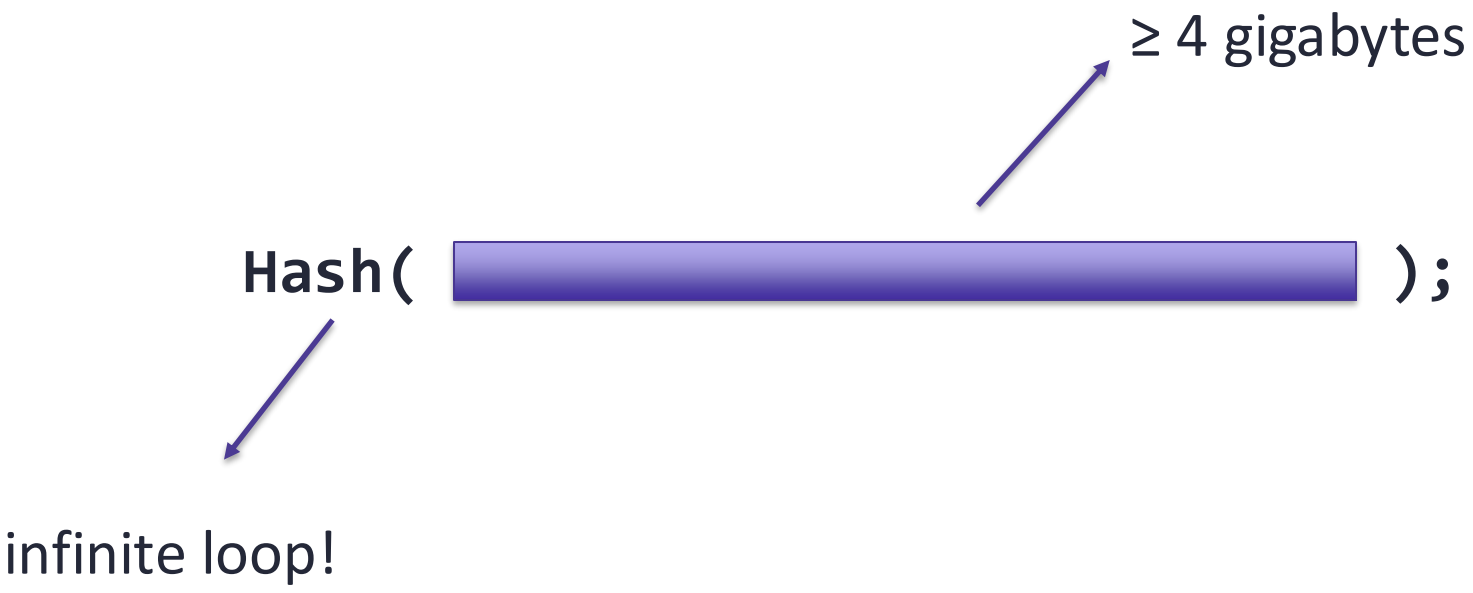
Apple CoreCrypto Bug (CT-RSA 2020)

- Affected 11/12 hash functions
- CVE-2019-8741, NVD: 7.5 HIGH



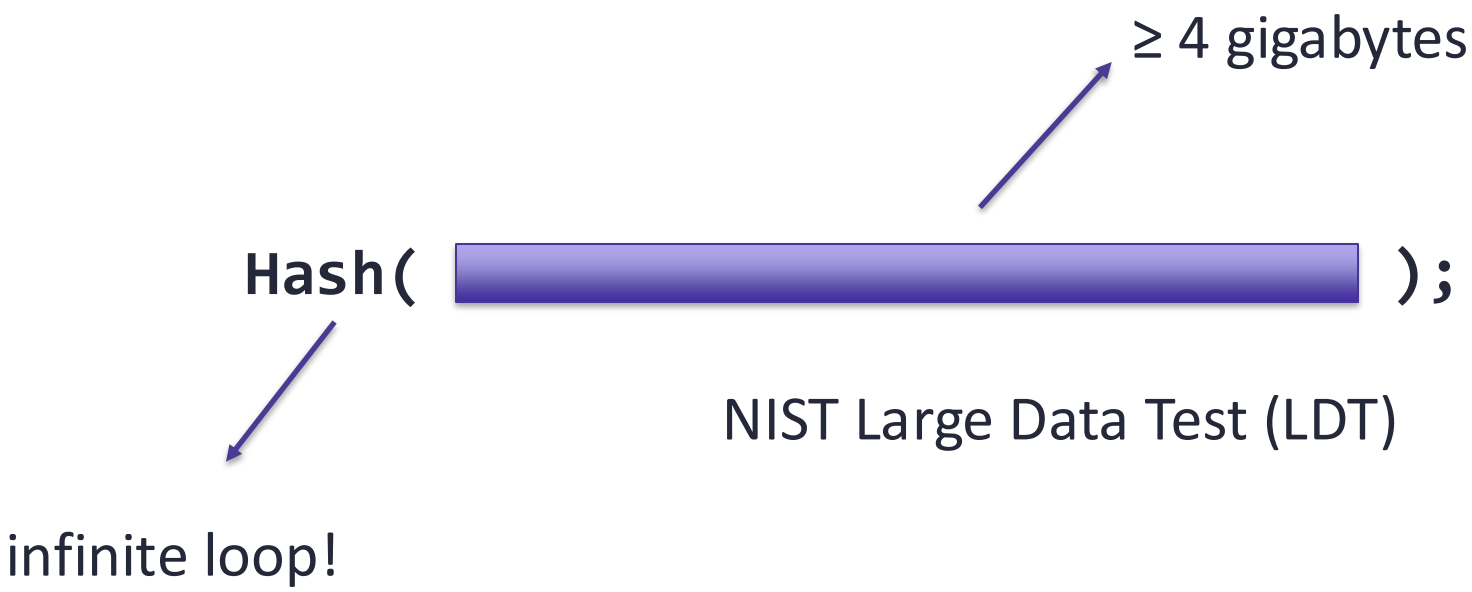
Apple CoreCrypto Bug (CT-RSA 2020)

- Affected 11/12 hash functions
- CVE-2019-8741, NVD: 7.5 HIGH



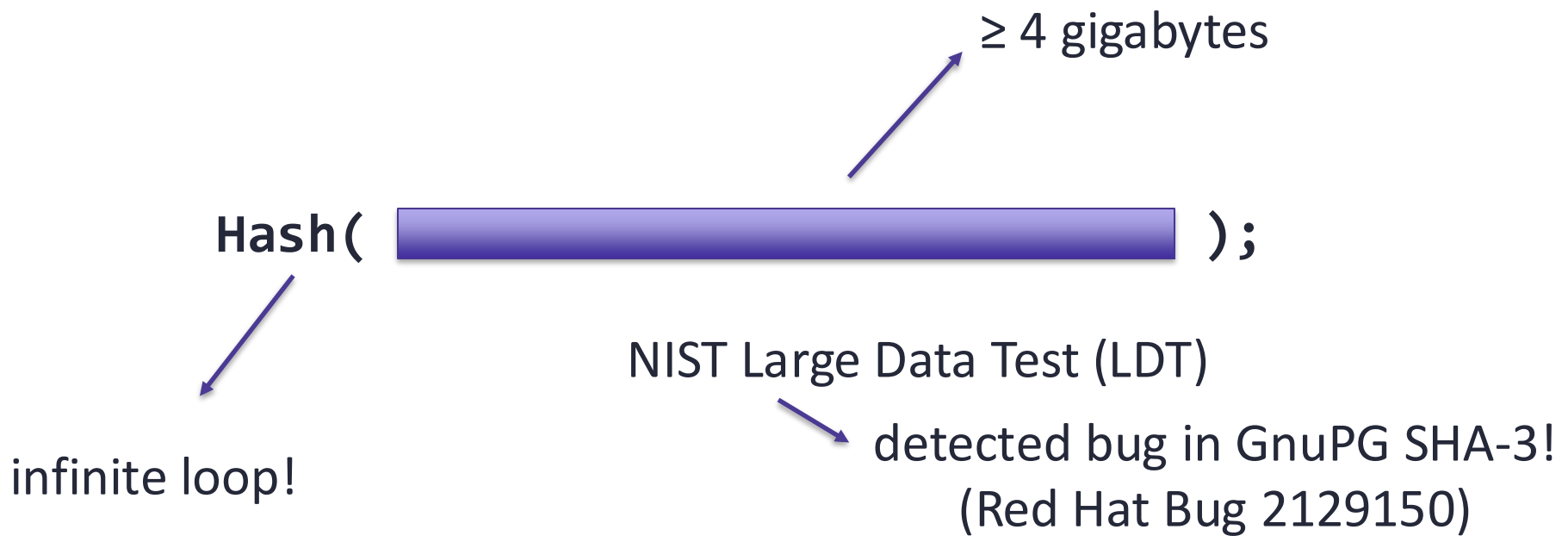
Apple CoreCrypto Bug (CT-RSA 2020)

- Affected 11/12 hash functions
- CVE-2019-8741, NVD: 7.5 HIGH



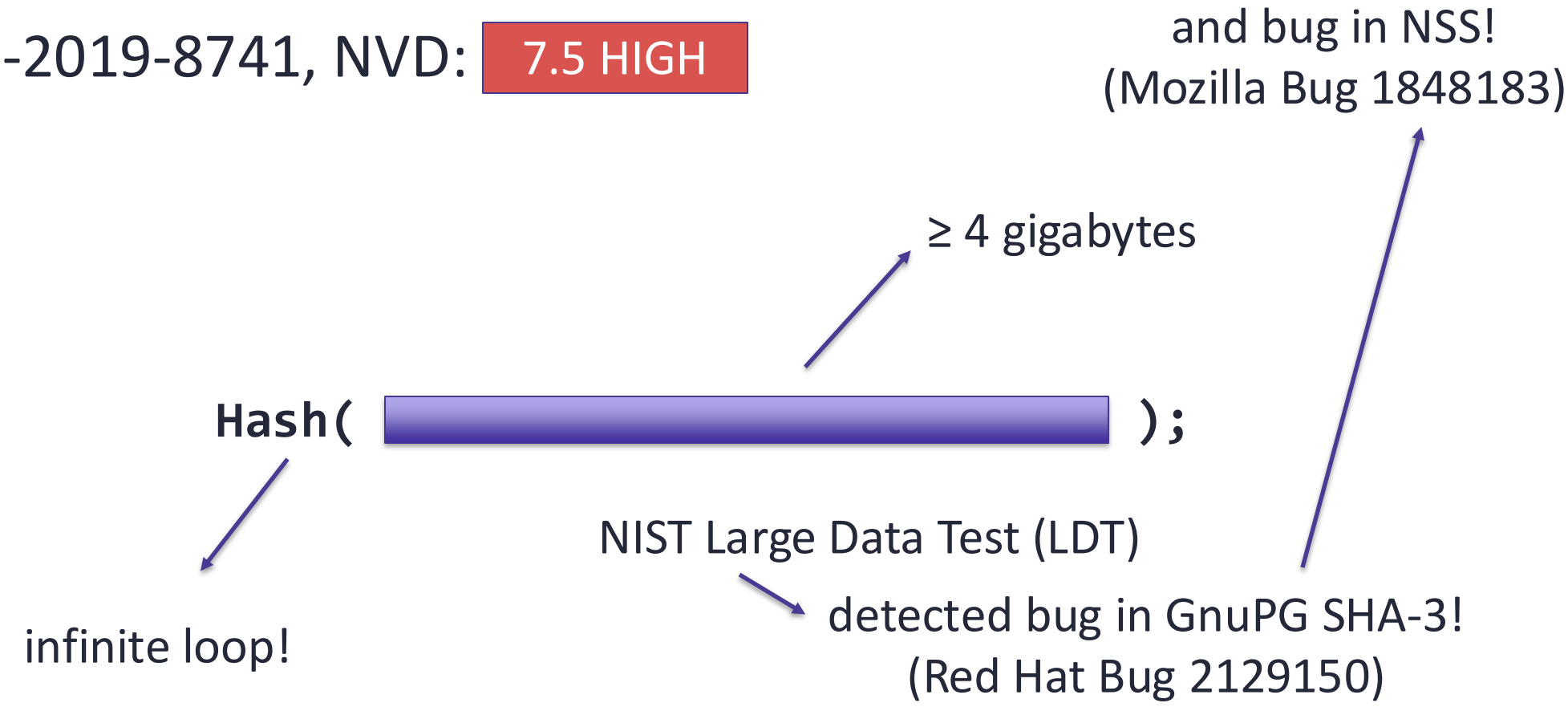
Apple CoreCrypto Bug (CT-RSA 2020)

- Affected 11/12 hash functions
- CVE-2019-8741, NVD: 7.5 HIGH



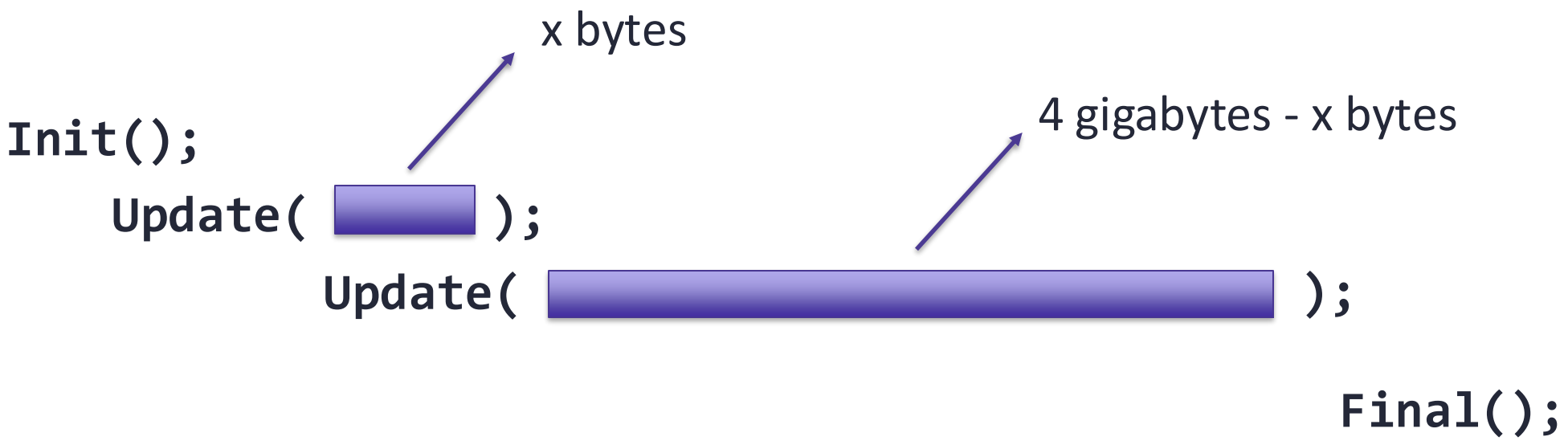
Apple CoreCrypto Bug (CT-RSA 2020)

- Affected 11/12 hash functions
- CVE-2019-8741, NVD: 7.5 HIGH



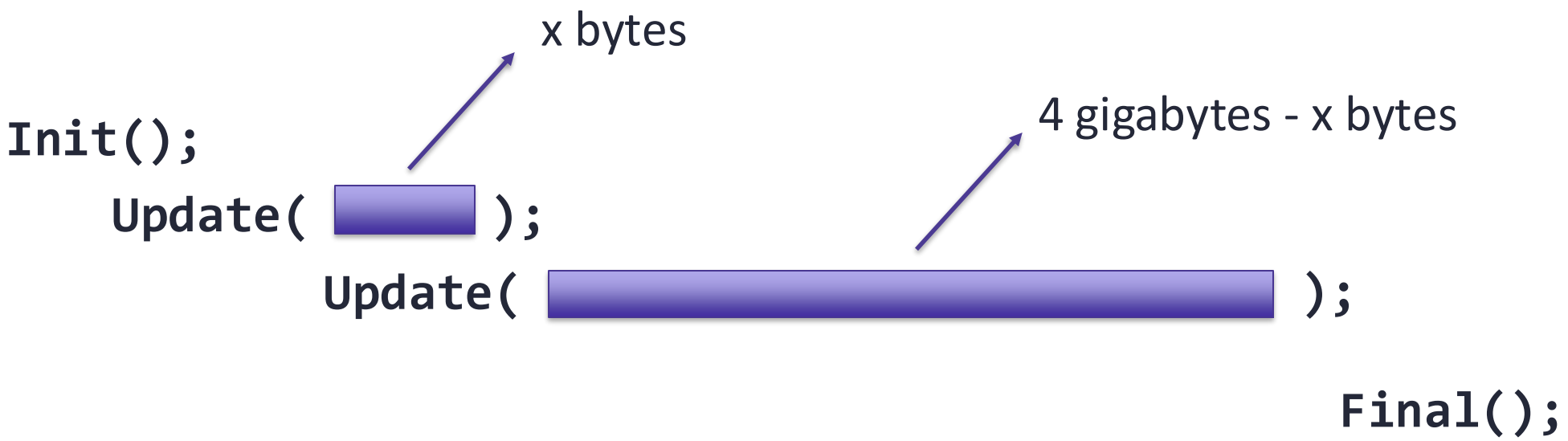
SHA-3 Bug (CT-RSA 2023)

- Appeared in 2011 (final-round Keccak submission)
- CVE-2022-37454, NVD: 9.8 CRITICAL



SHA-3 Bug (CT-RSA 2023)

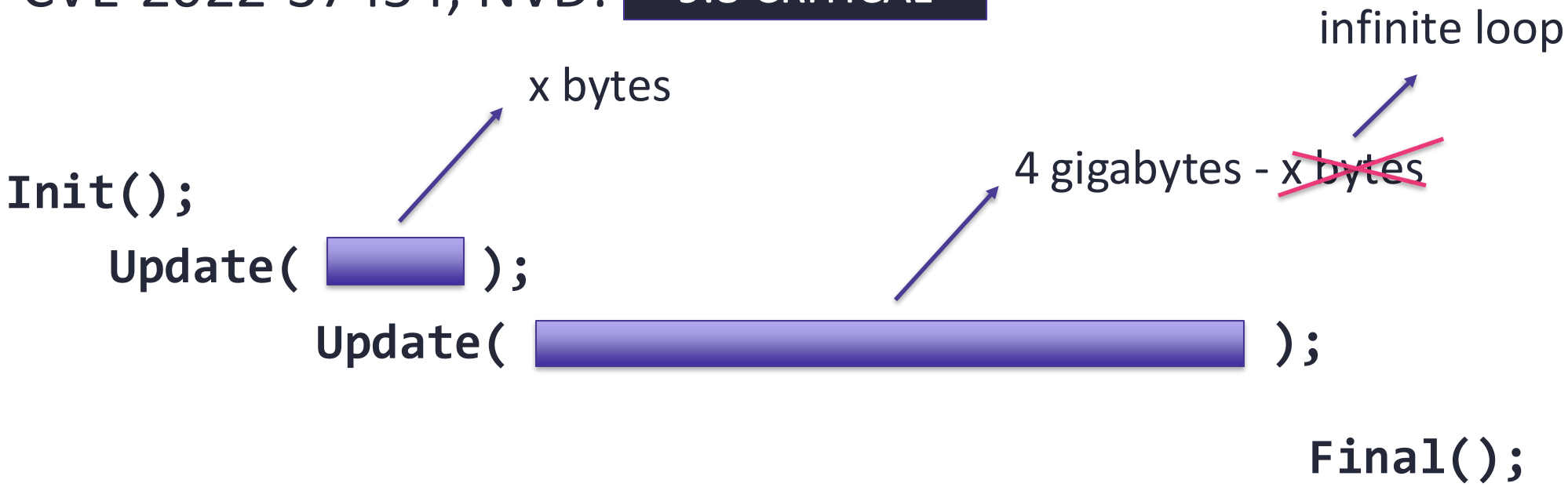
- Appeared in 2011 (final-round Keccak submission)
- CVE-2022-37454, NVD: 9.8 CRITICAL



buffer overflow!
(arbitrary code execution)

SHA-3 Bug (CT-RSA 2023)

- Appeared in 2011 (final-round Keccak submission)
- CVE-2022-37454, NVD: 9.8 CRITICAL



buffer overflow!
(arbitrary code execution)

KeccakSponge.inc

```
partialBlock = (unsigned int) (dataByteLen - i);
if (partialBlock + instance->byteIOIndex > rateInBytes) {
    partialBlock = rateInBytes - instance->byteIOIndex;
}
```

KeccakSponge.inc

```
partialBlock = (unsigned int) (dataByteLen - i);
if (partialBlock > rateInBytes - instance->byteIOIndex) {
    partialBlock = rateInBytes - instance->byteIOIndex;
}
```

KeccakSponge.inc

```
partialBlock = (unsigned int) (dataByteLen - i);
if (dataByteLen - i > rateInBytes - instance->byteIOIndex) {
    partialBlock = rateInBytes - instance->byteIOIndex;
}
```

KeccakSponge.inc

```
if (dataByteLen - i > rateInBytes - instance->byteIOIndex) {
    partialBlock = rateInBytes - instance->byteIOIndex;
} else {
    partialBlock = (unsigned int) (dataByteLen - i);
}
```

CodeQL

“Add query for CVE-2022-37454” (https://github.com/github/codeql/pull/12036)

```
todo = digest_len;
if (done + todo > out_len) {
  todo = out_len - done;
}
OPENSSL_memcpy(out_key + done, previous, todo);
done += todo;
```

CodeQL

“Add query for CVE-2022-37454” (https://github.com/github/codeql/pull/12036)

```
todo = digest_len;
if (todo > out_len - done) {
  todo = out_len - done;
}
OPENSSL_memcpy(out_key + done, previous, todo);
done += todo;
```

OpenSSL HKDF

“Update hkdf.c to avoid potentially vulnerable code pattern”
(https://github.com/openssl/openssl/pull/20990)

```
copy_len = (done_len + dig_len > okm_len)
   okm_len - done_len :
   dig_len;
memcpy(okm + done_len, prev, copy_len);
done += copy_len;
```

OpenSSL HKDF

“Update hkdf.c to avoid potentially vulnerable code pattern”
(https://github.com/openssl/openssl/pull/20990)

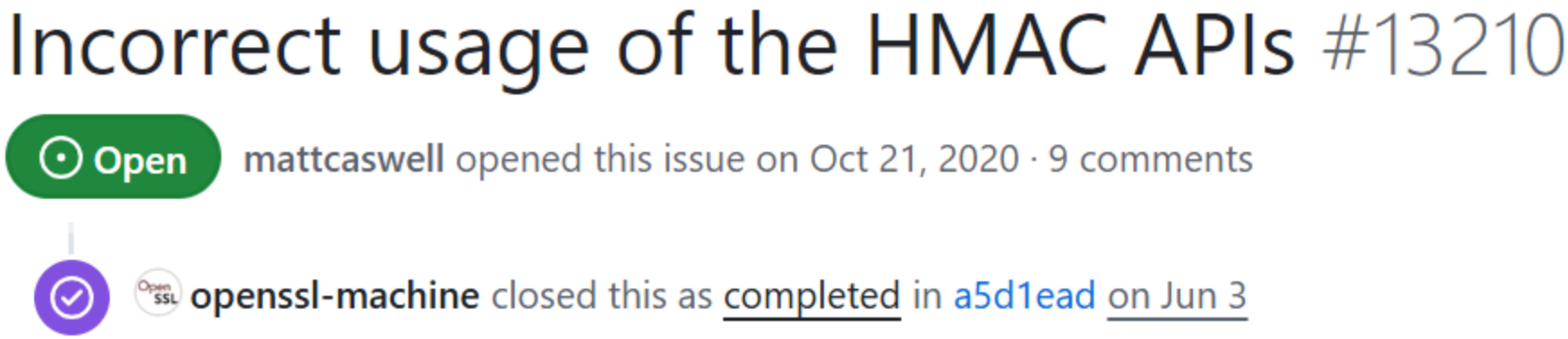
```
copy_len = (dig_len > okm_len - done_len)
   okm_len - done_len :
   dig_len;
memcpy(okm + done_len, prev, copy_len);
done += copy_len;
```

HMAC

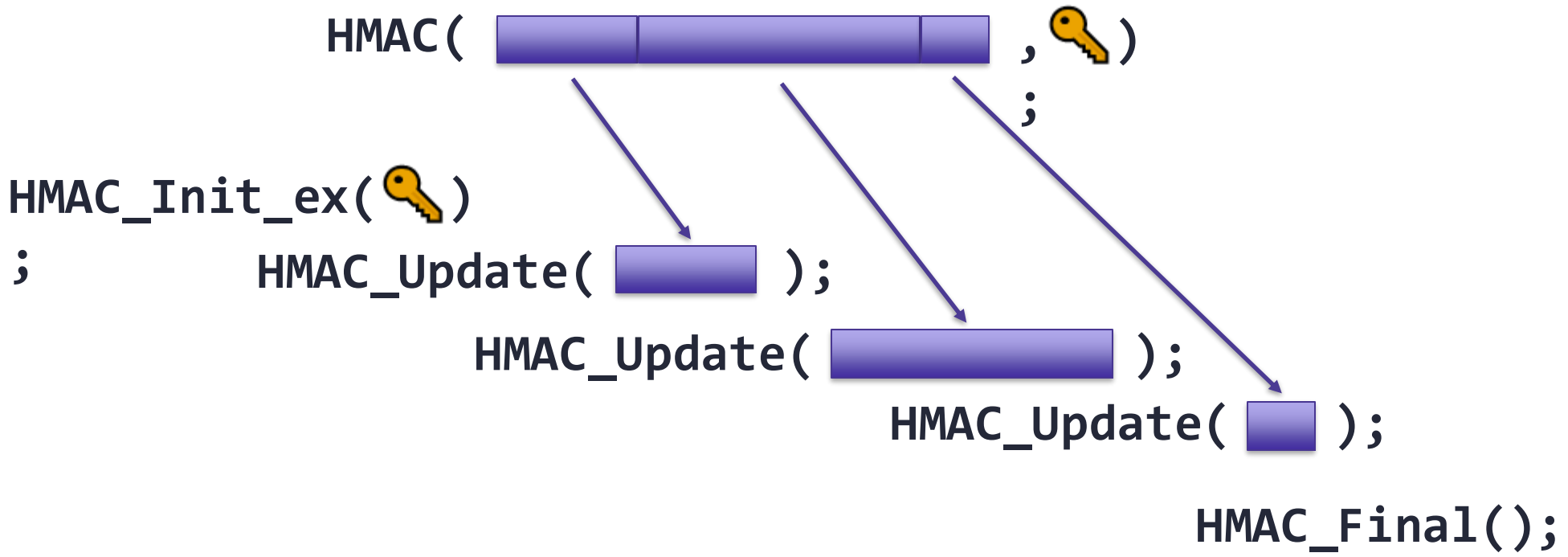
- OpenSSL HMAC API (Benmocha, et al., SAC 2021)

HMAC

- OpenSSL HMAC API (Benmocha, et al., SAC 2021)



Two Common HMAC Function Interfaces



Unintended Feature of HMAC API

```
HMAC_Init_ex();
HMAC_Update();
HMAC_Final();
```

Unintended Feature of HMAC API

```
HMAC_Init_ex();
HMAC_Update();
HMAC_Final();
HMAC_Update();
HMAC_Final();
```

Unintended Feature of HMAC API

```
HMAC_Init_ex();
HMAC_Update();
HMAC_Final();
HMAC_Update();
HMAC_Final();
```

- Q: Where/when is this used?

Unintended Feature of HMAC API

```
HMAC_Init_ex();
HMAC_Update();
HMAC_Final();
HMAC_Update();
HMAC_Final();
```

- Q: Where/when is this used?
- OpenSSL: instantly find collisions and multi-collisions under any key!

Ho, Fromherz, Protzenko (arXiv:2102.01644)

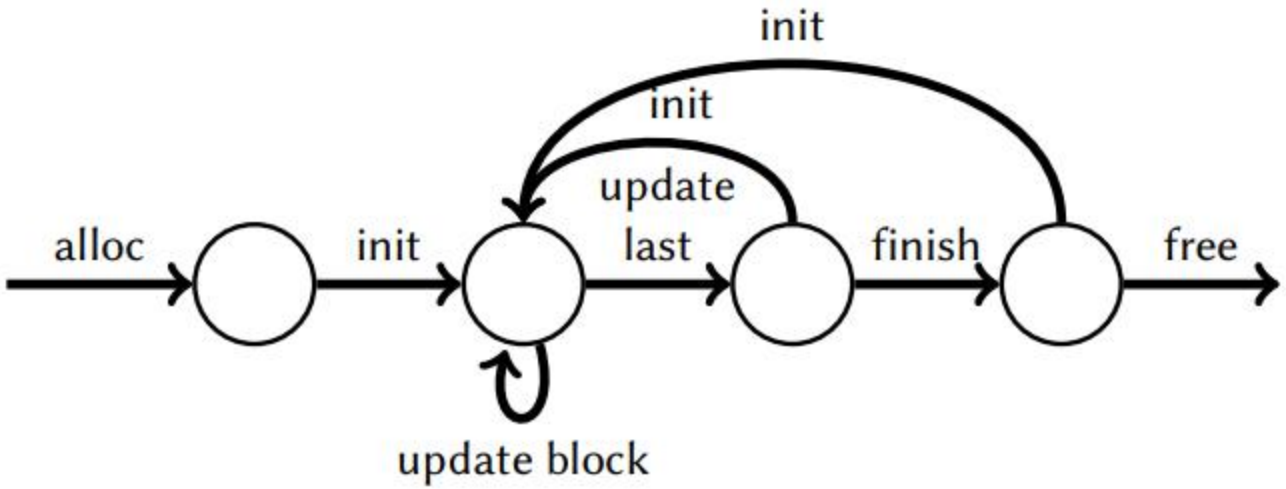


Fig. 1. State machine of an error-prone block-based API

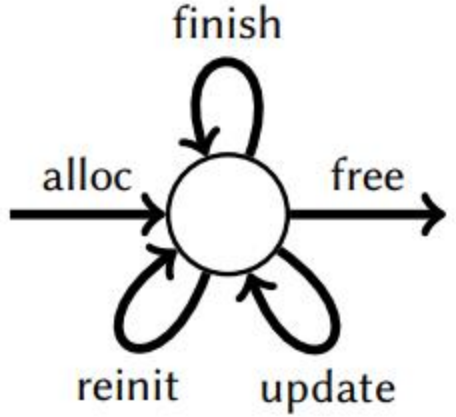


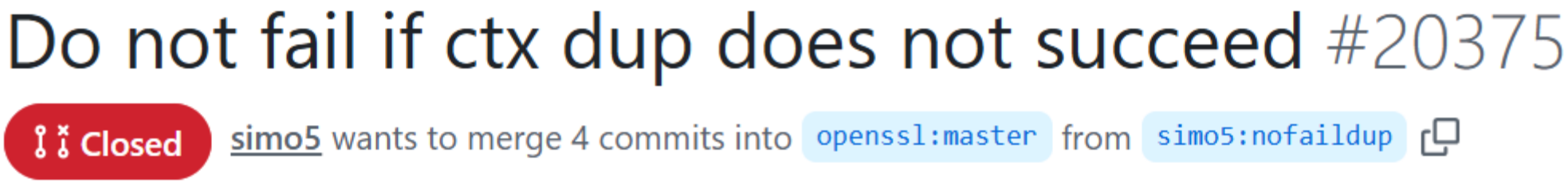
Fig. 2. State machine of a safe, streaming API

Fix?

- Reproduce the unintended feature
  - Since OpenSSL 3.2.0: **not reproducible!**

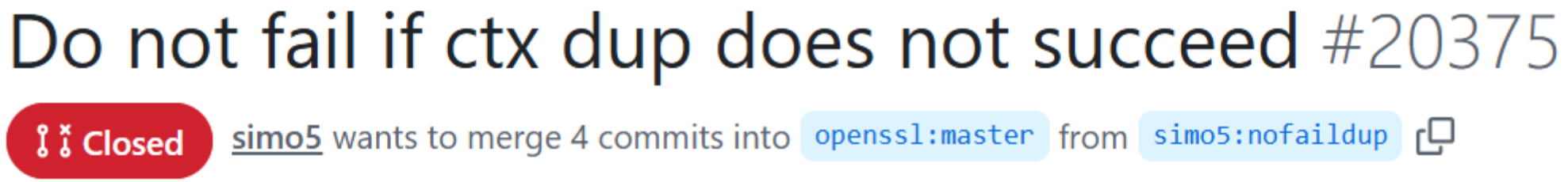
Fix?

- Reproduce the unintended feature
  - Since OpenSSL 3.2.0: **not reproducible!**



Fix?

- Reproduce the unintended feature
  - Since OpenSSL 3.2.0: **not reproducible!**



- Some APIs explicitly support `Update()` after `Final()`
  - But then `Final()` copies state, which may fail…

Document New API Behavior!

Regression test for incorrect HMAC API usage #27692

Closed **nmouha** wants to merge 1 commit into `openssl:master` from `nmouha:patch-1`

```
@@ -435,6 +458,7 @@ int setup_tests(void)
435  458      ADD_TEST(test_hmac_single_shot);
436  459      ADD_TEST(test_hmac_bad);
437  460      ADD_TEST(test_hmac_run);
     461  +   ADD_TEST(test_hmac_final_update_fail);
438  462      ADD_TEST(test_hmac_copy);
439  463      ADD_TEST(test_hmac_copy_uninited);
440  464      ADD_ALL_TESTS(test_hmac_chunks,
```

Conclusion

- OpenSSL HKDF
  - One line of code, not a bug
  - Avoid vulnerable code pattern

- OpenSSL HMAC
  - No new code, new test vector
  - Open issue closed after five years

Conclusion

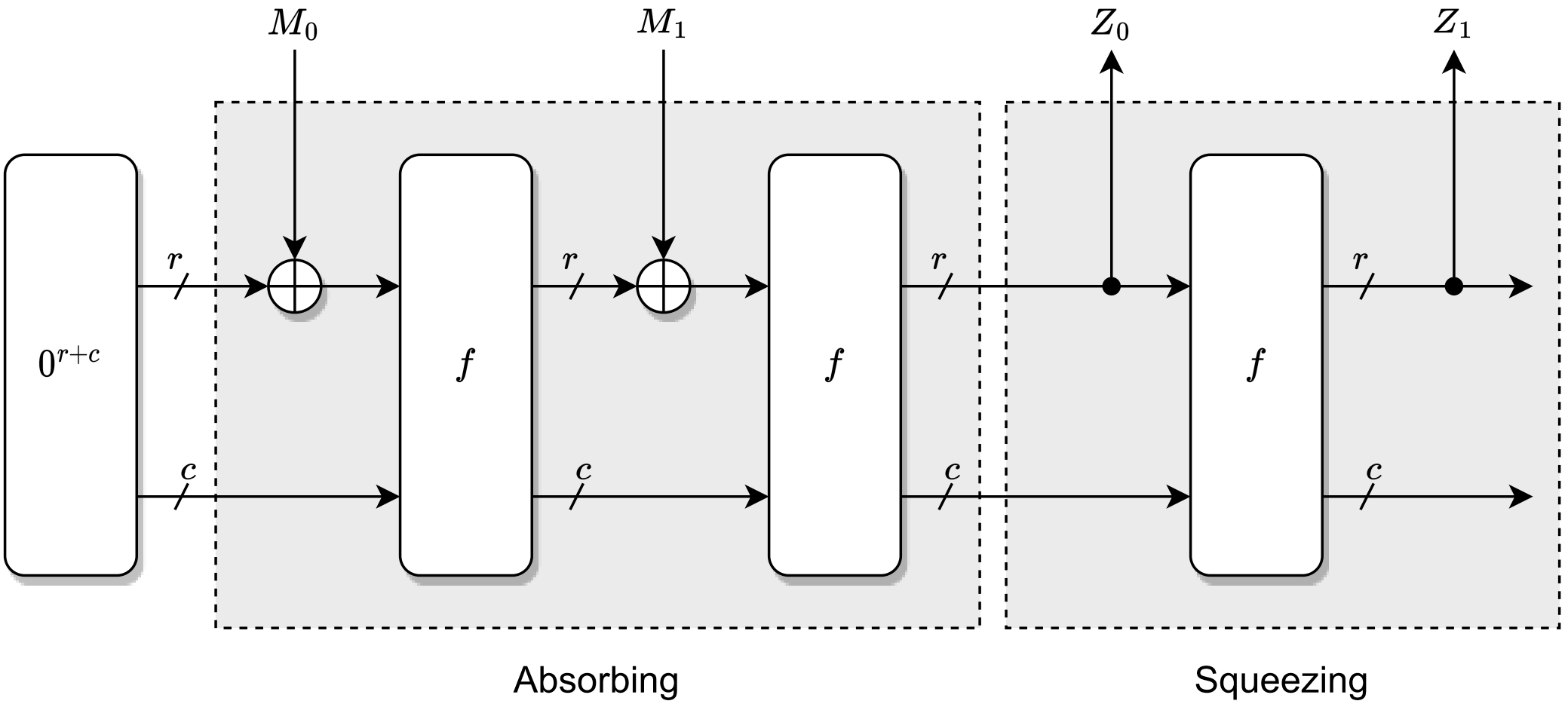
- OpenSSL HKDF
  - One line of code, not a bug
  - Avoid vulnerable code pattern

- OpenSSL HMAC
  - No new code, new test vector
  - Open issue closed after five years

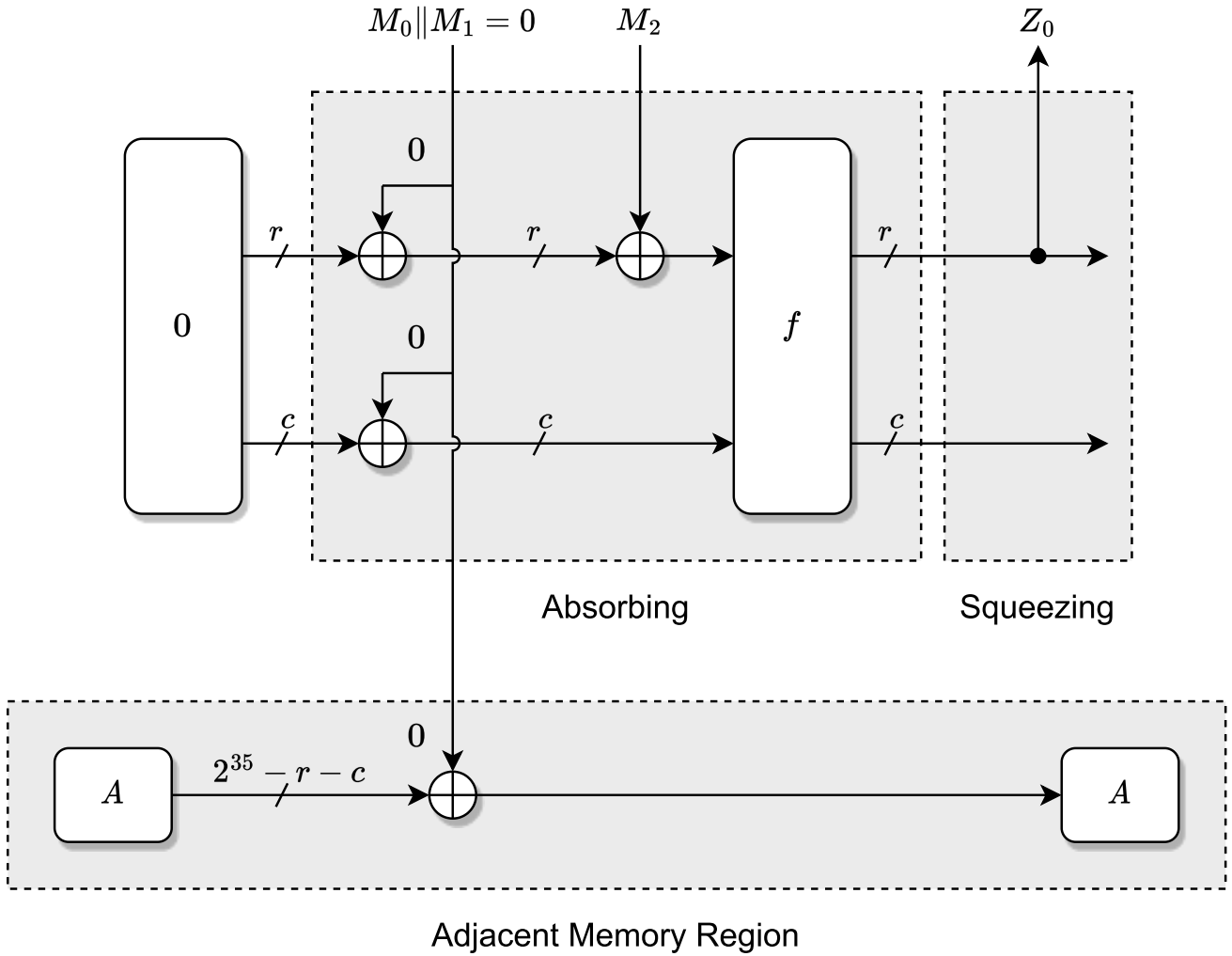
Questions?

Backup Slides

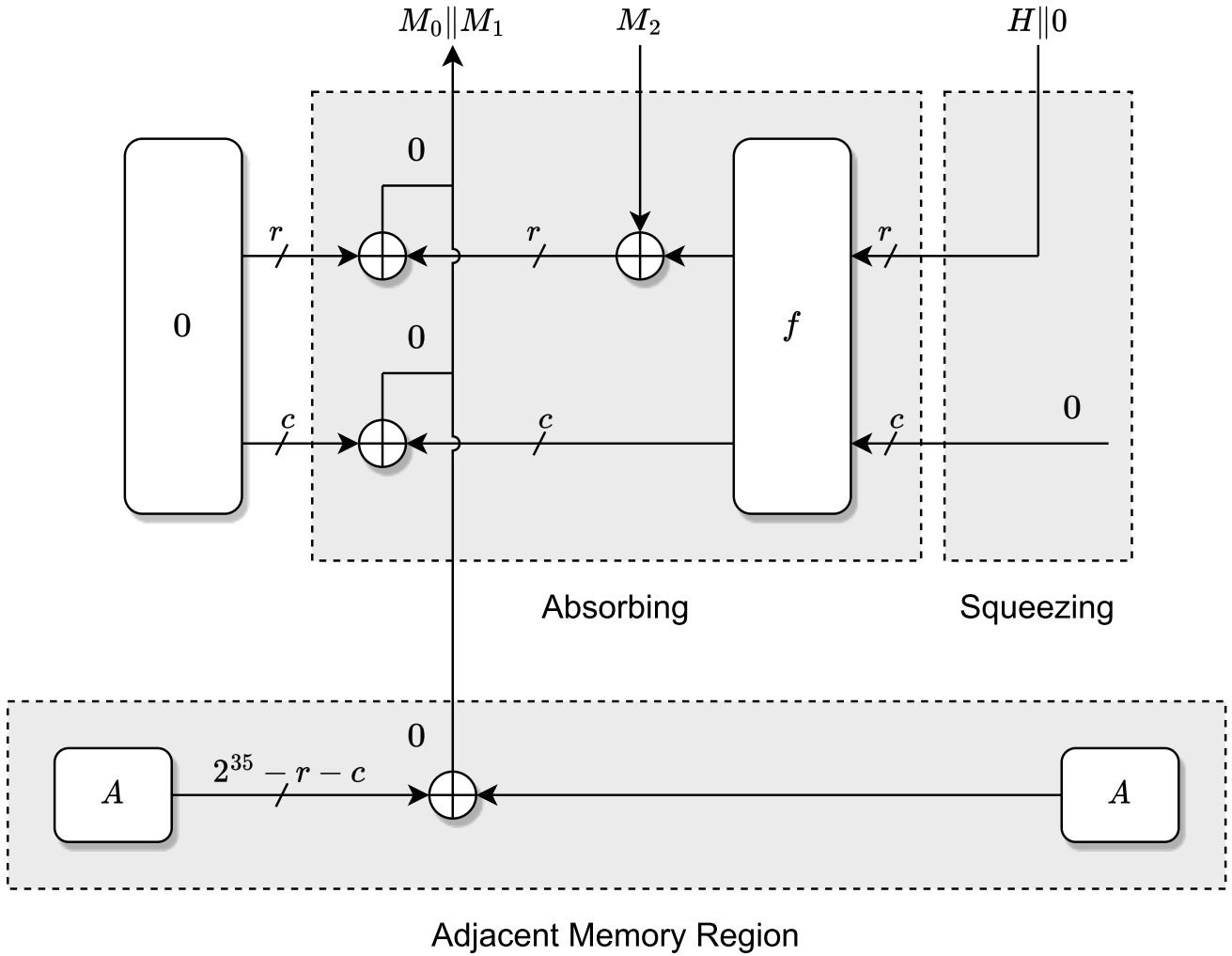
SHA-3: Sponge Function



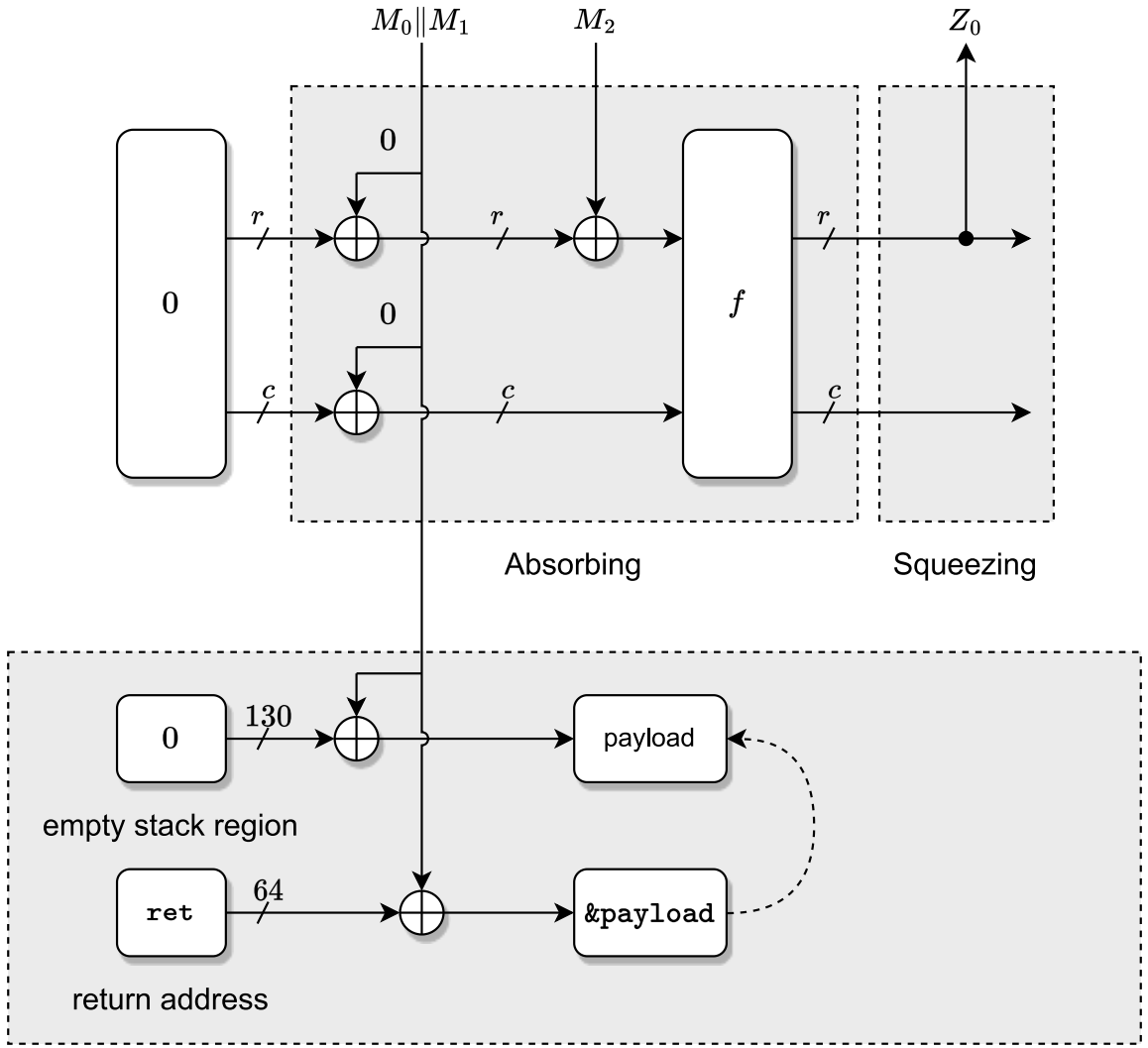
SHA-3: Second Preimage



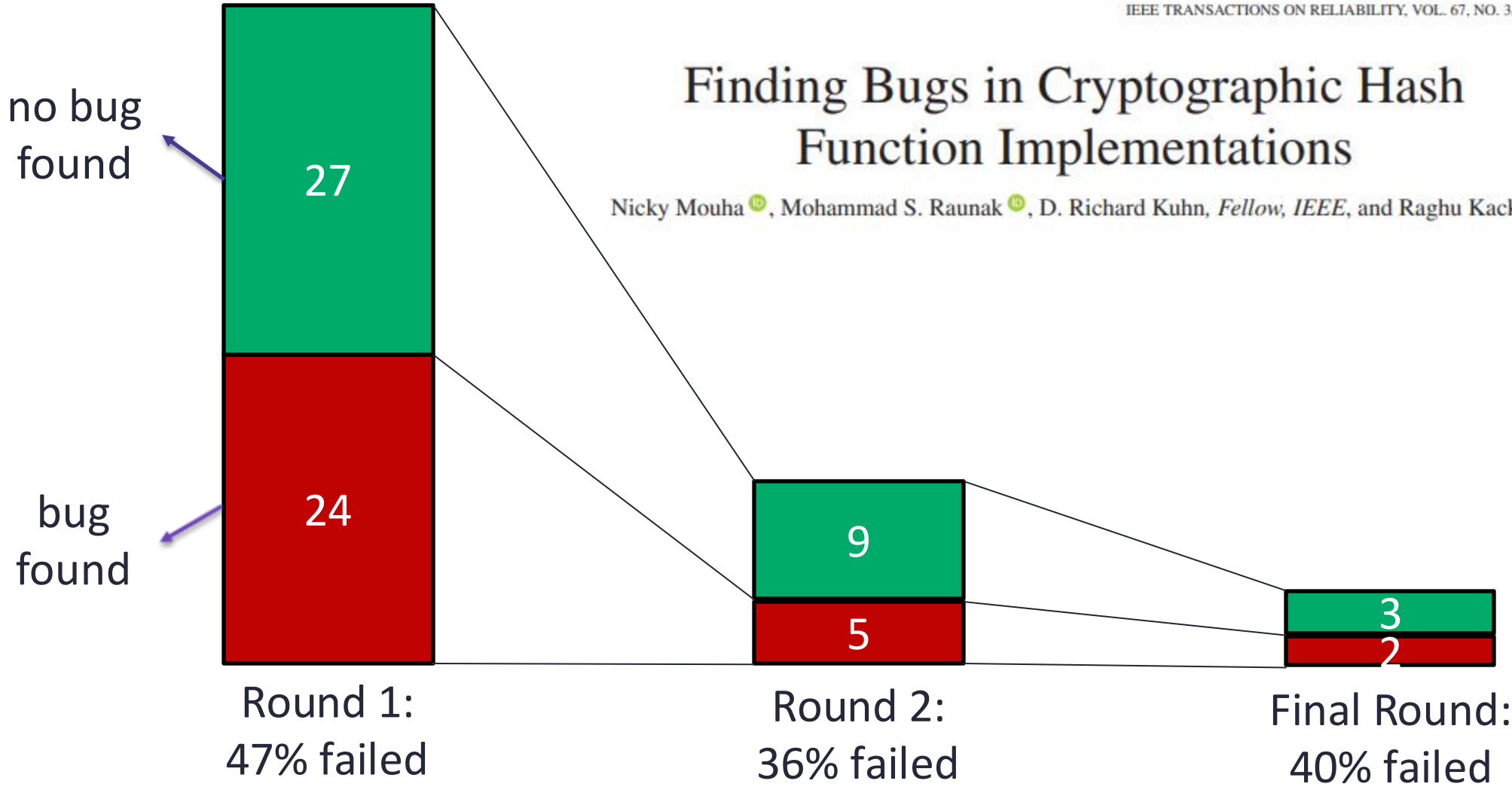
SHA-3: Preimage of Any Hash Value



SHA-3: Exploit



SHA-3 Competition Bugs



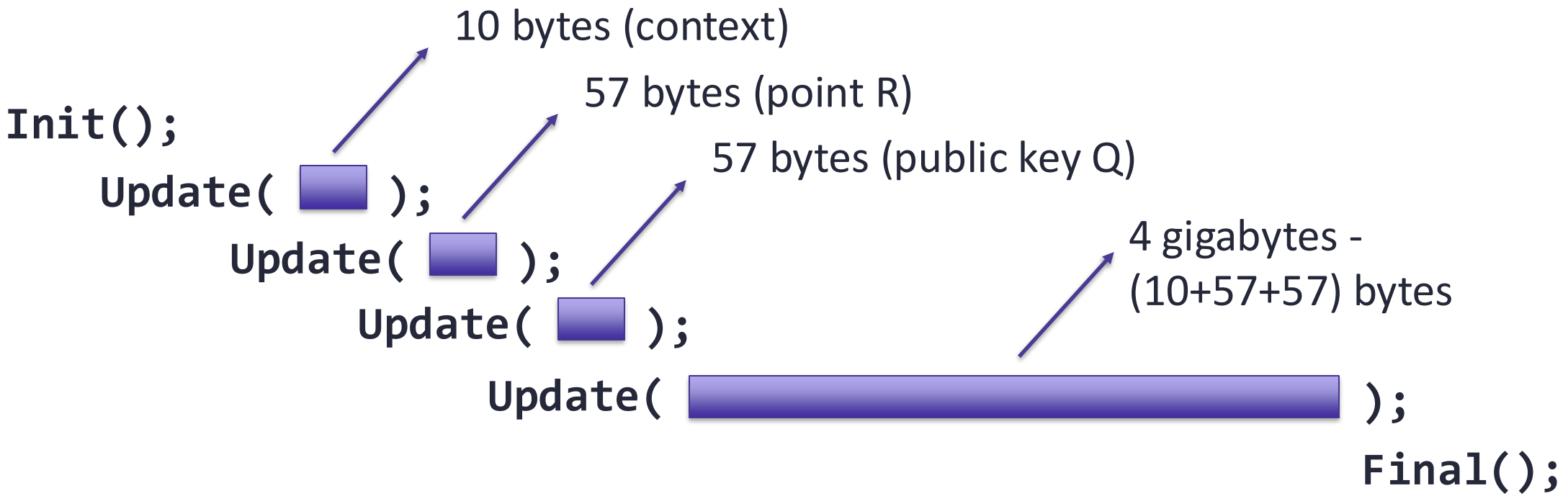
IEEE TRANSACTIONS ON RELIABILITY, VOL. 67, NO. 3, SEPTEMBER 2018

Finding Bugs in Cryptographic Hash Function Implementations

Nicky Mouha, Mohammad S. Raunak, D. Richard Kuhn, *Fellow, IEEE*, and Raghu Kacker

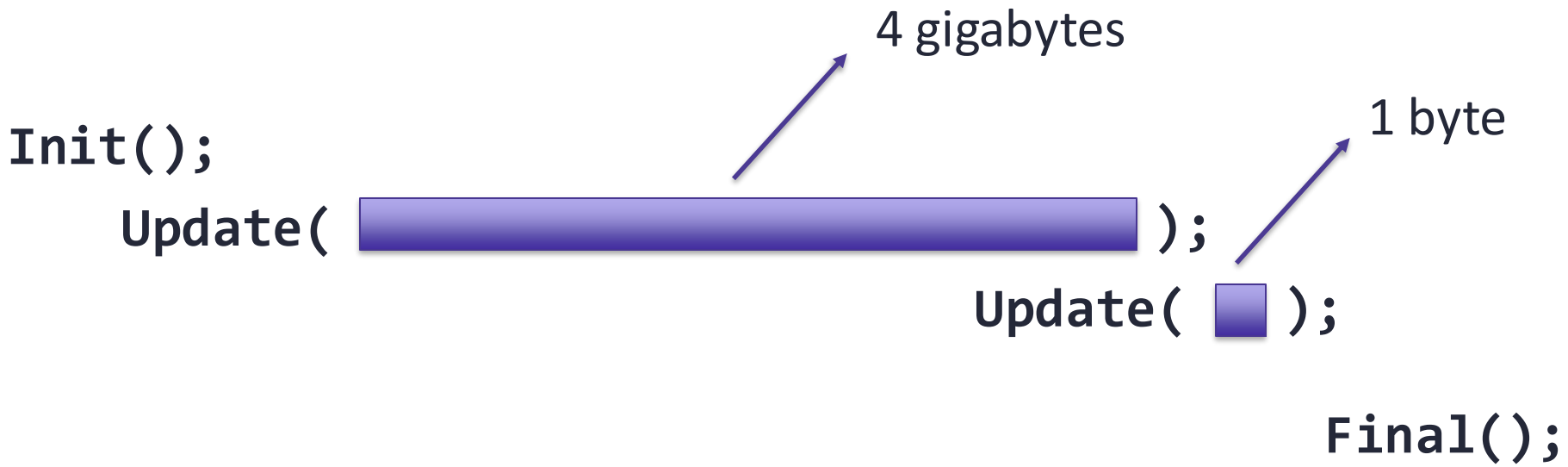
EdDSA (FIPS 186-5)

- Ed448: requires SHAKE256 (block size: 136 bytes)
- Attack **verification**!



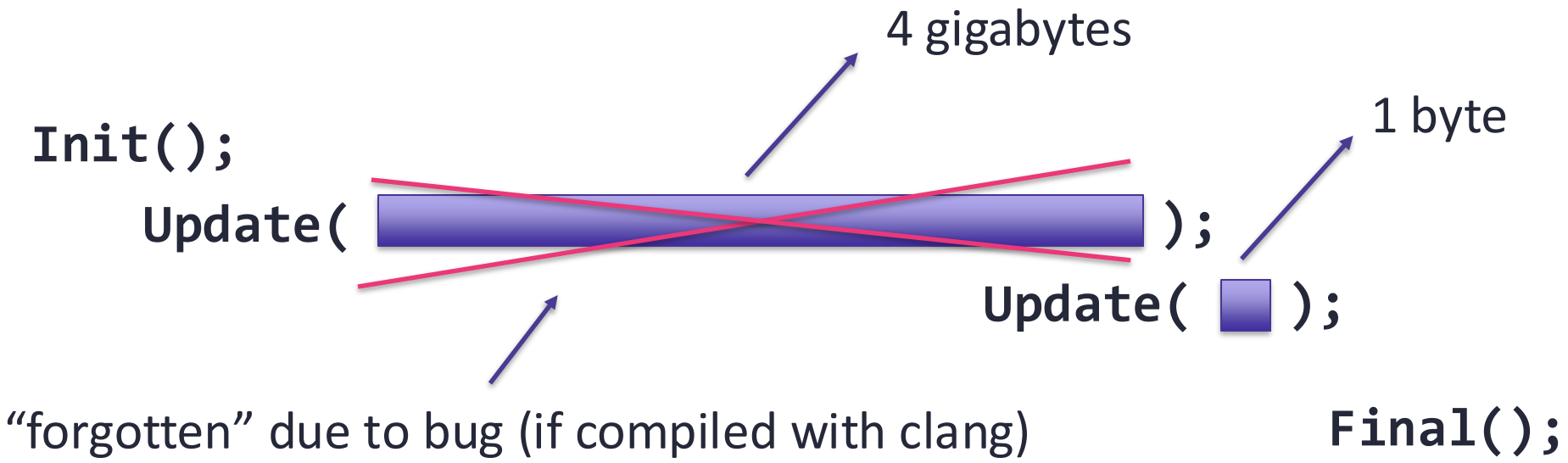
Grøstl Bug (ACISP 2023)

- Appeared in 2008
- First disclosed at ACISP 2023!



Grøstl Bug (ACISP 2023)

- Appeared in 2008
- First disclosed at ACISP 2023!



Grøstl Bug (ACISP 2023)

- Appeared in 2008
- First disclosed at ACISP 2023!