Hash-CFB

Authenticated Encryption Without a Block Cipher

Christian Forler[1], **Stefan Lucks**[1],
David McGrew[2], **Jakob Wenzel**[1]

[1]Bauhaus-Universität Weimar, Germany, [2]Cisco Systems, USA

DIAC, Stockholm,
July, 2012

Outlook

Goals
From BC-CFB to Hash-CFB
Alternatives
Security Claims
. . . Beyond “Standard” AE
. . . Core Ideas for Proofs
. . . on Side Channels
Final Remarks and Summary

Goals

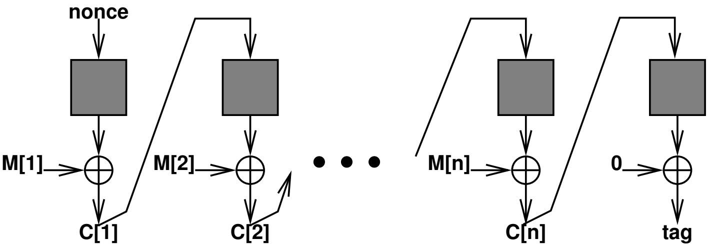
1. security (of course)
2. feasible on constrained devices

one primitive to rule them all,
one primitive to bind them . . .

3. simplicity:
   - easy to describe
   - easy to implement
   - easy to analyze

   based on a “standard” primitive
4. reasonable efficiency

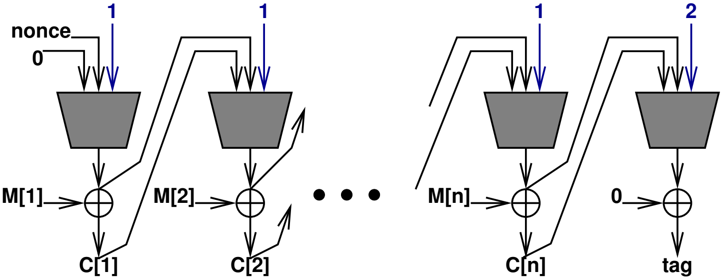
From BC-CFB to Hash-CFB



BC-CFB:

- **privacy:** CFB encryption
- **authenticity:** **trivial attacks!**

From BC-CFB to Hash-CFB

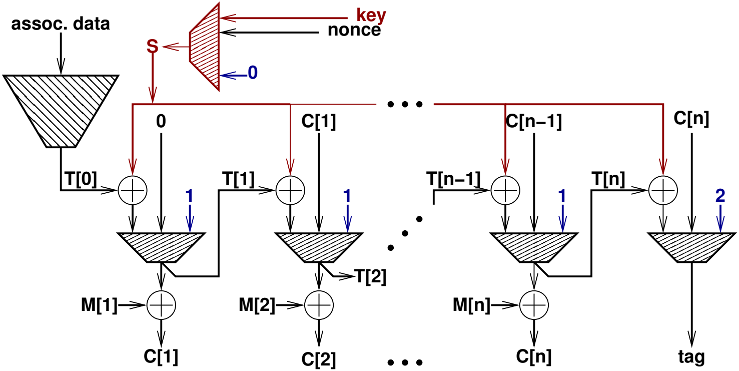


Hash-CFB, using a fixed-input-length (FIL) hash function:

- **privacy:** the same as CFB encryption
- **authenticity:** *secure – see later*

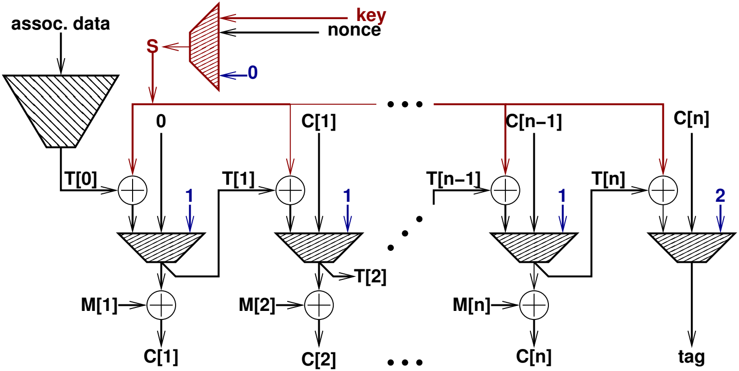
1. make both **T[i] = C[i]** $\oplus$ **M[I]** and **C[i]** inputs for the $(i + 1)$st call
2. differentiate last primitive call from previous calls

From BC-CFB to Hash-CFB



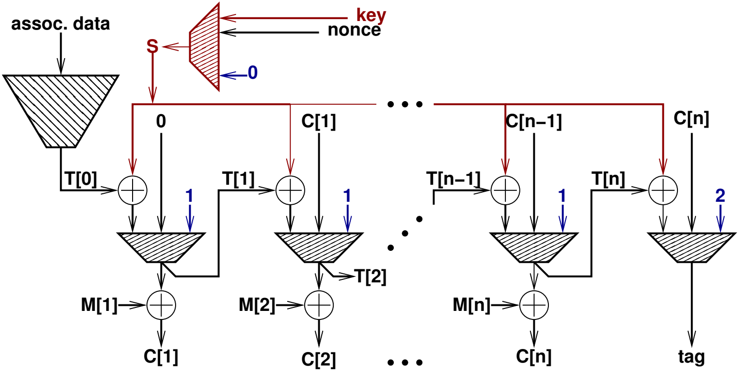
- long-term **key** and **nonce** define message-secret **S**

From BC-CFB to Hash-CFB



- long-term **key** and **nonce** define message-secret **S**
- **S** is xor-ed to the previous hash output
  (recall that a hash function is unkeyed, by nature)

From BC-CFB to Hash-CFB



- long-term **key** and **nonce** define message-secret **S**
- **S** is xor-ed to the previous hash output (recall that a hash function is unkeyed, by nature)
- use a VIL (variable input length) hash of the *associated data*

Alternatives

	primitive	solution
1.	block cipher	block cipher based hash fun.
2.	hash function	generic composition (e.g., counter mode & HMAC)
3.	compression function	(whatever)

Alternatives

	primitive	solution
1.	block cipher	block cipher based hash fun.
2.	hash function	generic composition (e.g., counter mode & HMAC)
3.	compression function	(whatever)

1. standard block cipher: AES, $n = 128$
   need $2n$-bit hash function; plenty of good DBL-hashes in literature, but
   what is the "standard" for DBL hashing?

Alternatives

	primitive	solution
1.	block cipher	block cipher based hash fun.
2.	hash function	generic composition (e.g., counter mode & HMAC)
3.	compression function	(whatever)

1. standard block cipher: AES, $n = 128$
   need $2n$-bit hash function; plenty of good DBL-hashes in literature, but
   what is the “standard” for DBL hashing?
2. **how to deal with additional complexity and storage?** (two independent keys, two states, . . .)

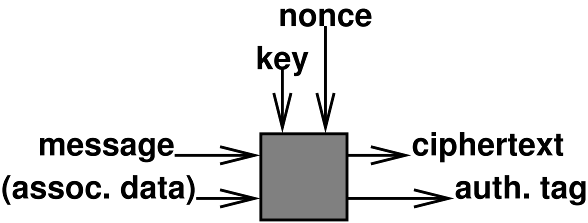
Alternatives

	primitive	solution
1.	block cipher	block cipher based hash fun.
2.	hash function	generic composition (e.g., counter mode & HMAC)
3.	compression function	(whatever)

1. standard block cipher: AES, $n = 128$
   need $2n$-bit hash function; plenty of good DBL-hashes in literature, but
   what is the “standard” for DBL hashing?
2. **how to deal with additional complexity and storage?** (two independent keys, two states, . . .)
3. cryptographers know the “compression functions”, but
   which standards or APIs actually define them?

Security Claims

Standard AE Claims

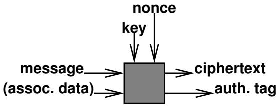


- assume the hash function behaves like a good PRF
- restrict the adversary to be *nonce-respecting*

- **privacy:** chosen plaintext attack (CPA) resistant
- **authenticity:** integrity of ciphertexts (Int-CTXT)
- more **privacy:** CPA and Int-CTXT ⇒ CCA

Security Claims

… Beyond “Standard” AE

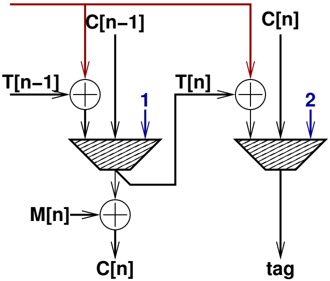


- **nonce misuse:** the adversary is not always *nonce respecting* (e.g., due to implementation errors)
  - **privacy:** still holds when using a new nonce
  - **authenticity:** not affected (!)
- **weak assumptions:**
  - **privacy:** requires the FIL HF to be a good PRF
  - **authenticity:** only requires “forgery resistance” of the FIL HF
- **side-channel resistance:** (see below)

Security Claims

. . . Core Ideas for Proofs

- **privacy:** similar to block cipher based CFB
- **authenticity:** for queries, the final hash input to compute **tag** is always different:



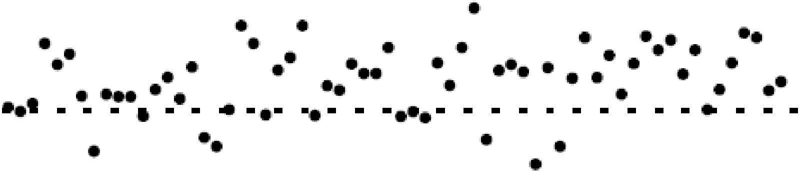
- **T[n]** is a (keyed) hash of the message (⇒ no collisions), and
- the postfix **2** is only used for final hash function calls

so a forger would have to **predict** the output of the final FIL hash function call – even if the same nonce had been used repeatedly

Security Claims

. . . on Side Channels

typical side-channel attacks:

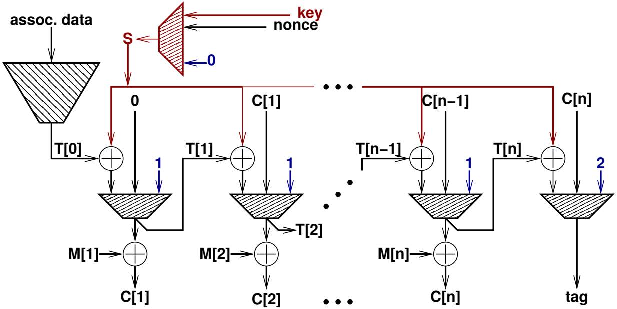


- many measurements of a primitive operations under the same key
- *X* messages, each of length *L* blocks:

  XL measurements for the same key

Security Claims

… on Side Channels



side-channel attacks against hash-CFB:

- *X* messages, each of length *L*, nonce-respecting: *X* measurements for **key** and *L* for each of **S**
- even when not nonce-respecting: adversary may find some **S** but only use it to to compromise messages using that single **nonce**

Final Remarks and Summary

- in the paper
  - SHA-224-based instantiation of HASH-CFB:
  - one FIL hash $\Leftrightarrow$ one compression function call
- our goals:
  - secure, feasable on constrained devices, simple, efficient (in that order)
  - using a hash function seems to be a good approach
- security requirements (beyond “standard”):
  - authenticity even under nonce reuse
  - authenticity needs weaker assumption than privacy
  - some defense against side-channel attacks
- for discussion at DIAC:
  - **Should such security requirements become a standard for new generation AE schemes?**