Storage Encryption: A Cryptographer's View

Shai Halevi
IBM Research

Motivation

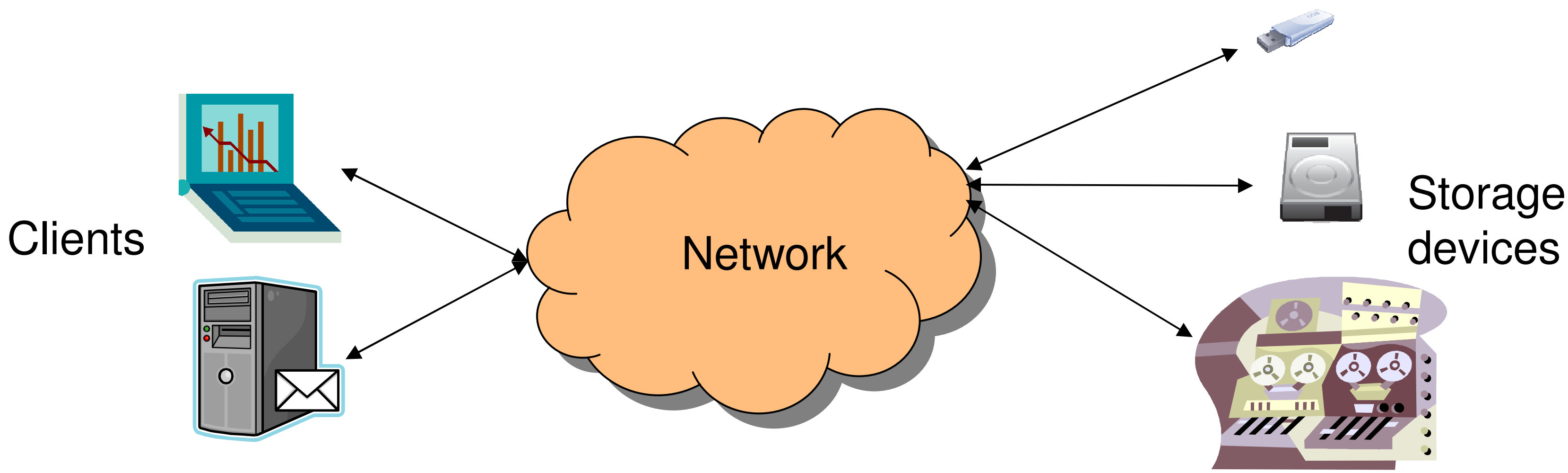
- *"You're working on storage encryption? It must be the most boring thing in the world…"*

  Anonymous
- Encryption is the most basic task in crypto
  - We know what secure encryption means
    - CCA-security, Authenticated encryption, …
  - We have provably-secure schemes
    - Even efficient ones
- What is left to research?

Cryptographically interesting problems with storage encryption

- Choosing the encryption scheme
  - “Transparent” vs. authenticated encryption
- Managing keys and nonces
  - Avoiding nonce re-use, wrapping keys, …
- *Outside the model*
  - Circular encryption

Typical Storage Architecture



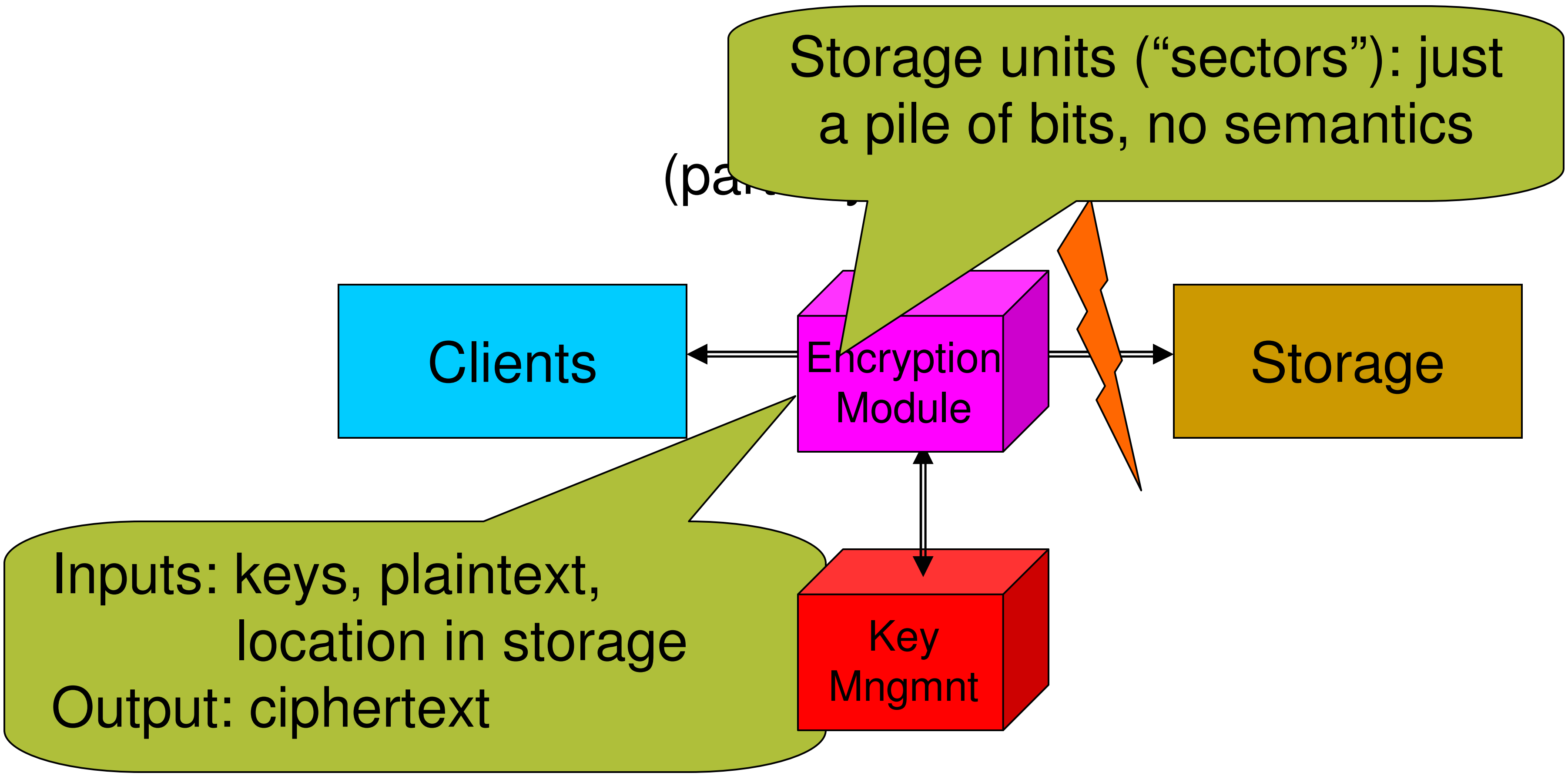
- My focus: encryption at the network/devices
  - Typical th… organiza… zed disclosure / modification
  - E.g., encrypting tapes, lest they fall off the truck

Several such high-profile incidents in 2005

Two Types of Encryption

- "Transparent" (length-preserving)
  - Used to add encryption to existing data-paths
  - E.g., software hard-disk encryption, or a bump-in-a-wire encryption box
- Authenticated (length-increasing)
  - Used when the "storage medium" allows records of flexible-length
  - E.g., tape encryption, client-side encryption, etc.

Transparent encryption



Inherent limitations

- Random access $\Rightarrow$
  Each "sector" encrypted separately $\Rightarrow$
  Can mix and match
  - $C_1\ C_2 \ldots C_m$ is encryption of $P_1\ P_2 \ldots P_m$
  - $C_1'\ C_2' \ldots C_m'$ is encryption of $P_1'\ P_2' \ldots P_m'$
  - $\Rightarrow$ $C_1\ C_2' \ldots C_m$ is encryption of $P_1\ P_2' \ldots P_m$
- Length preserving $\Rightarrow$ Deterministic $\Rightarrow$
  When re-encrypting a file, we can see what sectors have changed
- Length preserving $\Rightarrow$ No authentication $\Rightarrow$
  Any ciphertext sector is decrypted as "something"

The best we can do: Tweakable Encryption [LRW02]

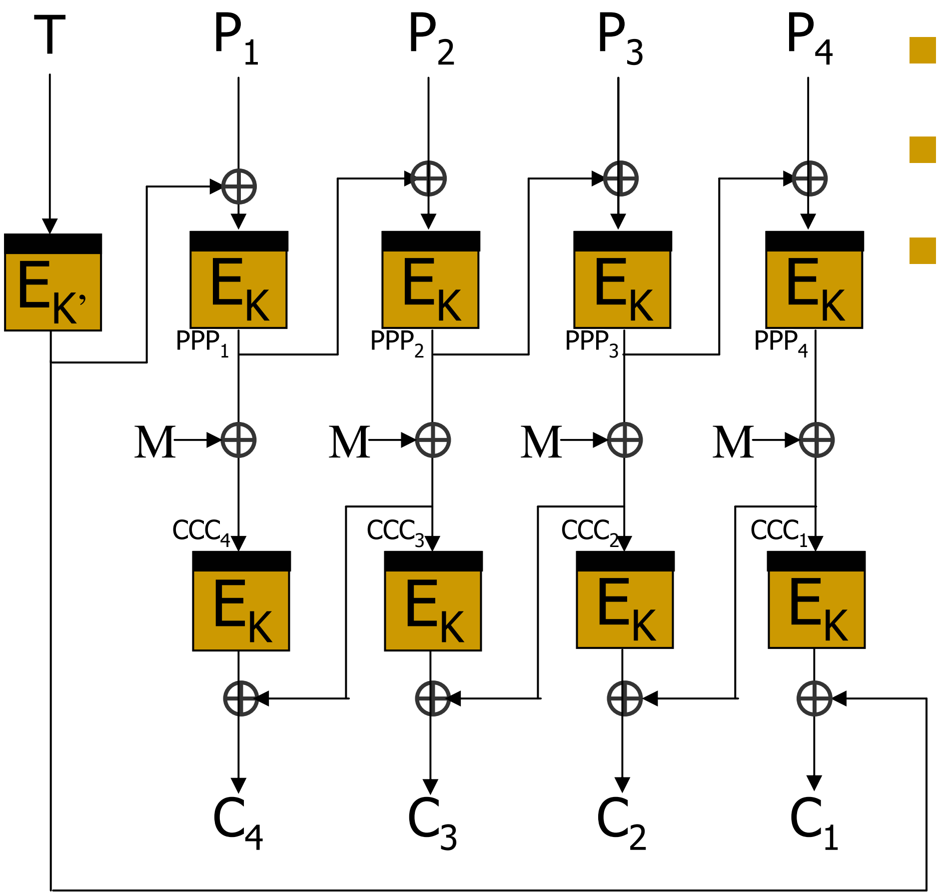
- Enciphering/deciphering routines:
  ciphertext = E(key, tweak, plaintext),
  plaintext = D(key, tweak, ciphertext)
  - ciphertext-length = plaintext-length
  - key is fixed and secret
  - tweak is arbitrary (even adversarially chosen)
- Should look like
  - A block cipher with block-size = plaintext-length
  - Different tweaks look like independent keys

Narrow vs. Wide Blocks

- Narrow-blocks
  - Each 16-byte block is encrypted separately (think ECB)
- Wide-block
  - The entire sector is encrypted together
  - Change anywhere effect entire ciphertext
- Quantitative, not qualitative difference
  - They are the same if you use 16-byte sectors

Some Wide-Block Modes

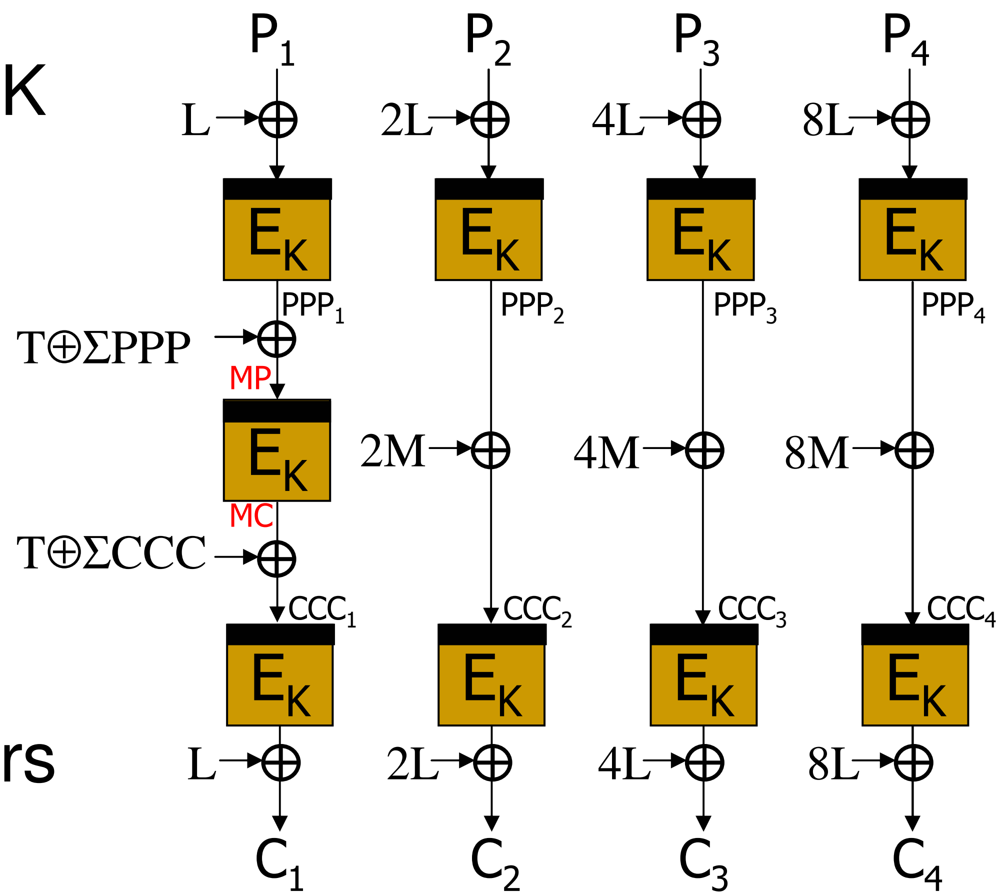
CMC [HR03]



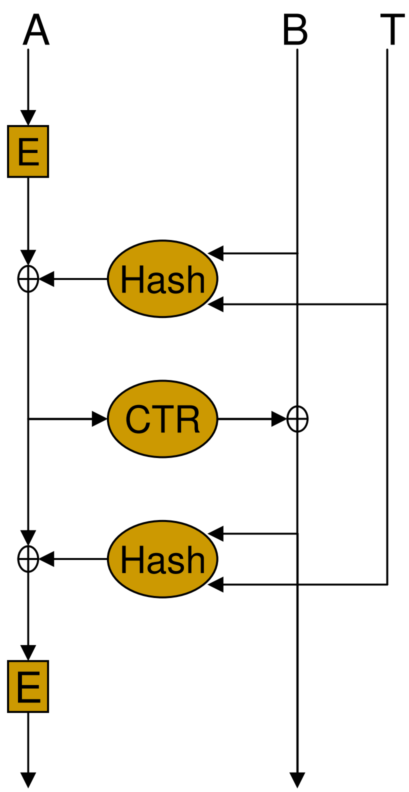
- E_K = AES with key K
- T – tweak
- $M = 2(PPP_1 \oplus PPP_4)$
  $= 2(CCC_1 \oplus CCC_4)$
  - Mult. In $GF(2^{128})$

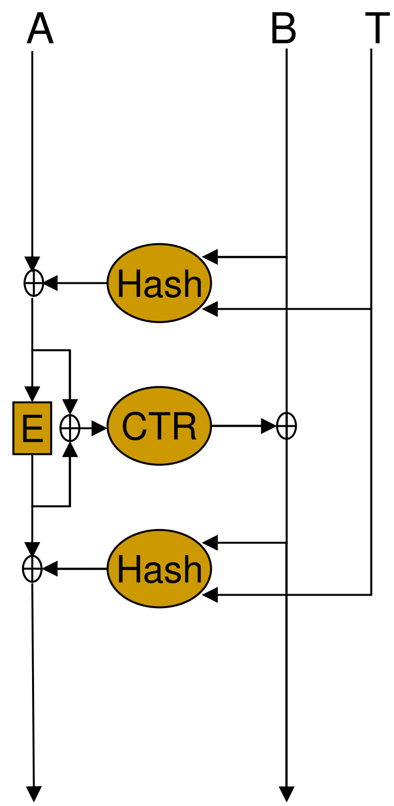
EME [HR04]

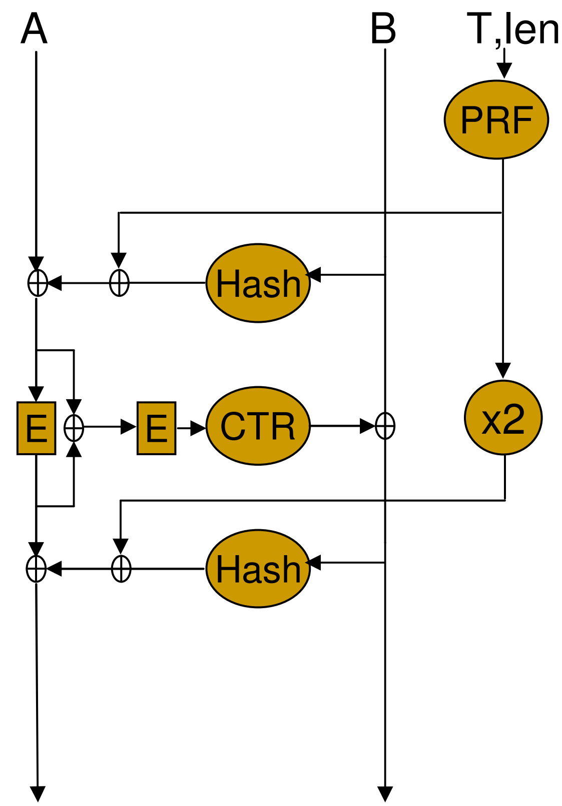
- E_K = AES with key K
- L = another key
- T = tweak
- M = MP ⊕ MC
- EME∗ [H04] is an extension for sectors longer than 2KB



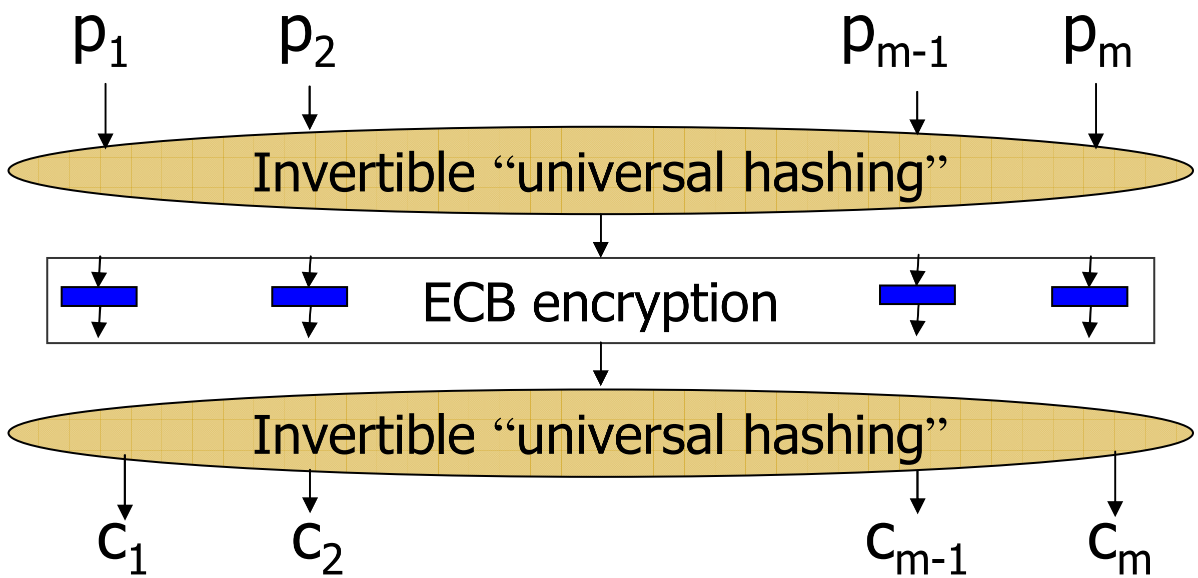
XCB [MF04] HCTR [WFW05] HCH [CS06]







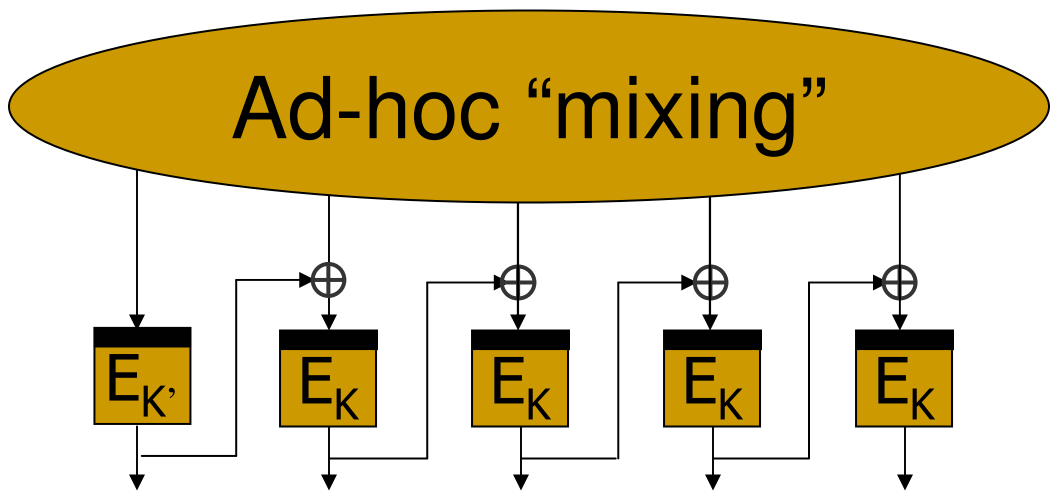
Naor-Reingold Modes: TET [H07], HEH [S07]



- "Universal hashing" ensures no collisions in the input to the ECB layer

Microsoft BitLocker [F06]

- Not quite an AES mode of operation

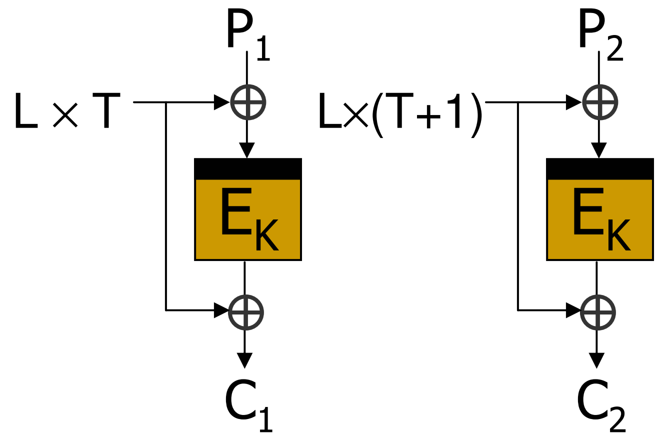


- "Block-cipher-like" mixing
  - Detailed analysis of resistance to attacks, but no reduction to the security of AES

Some Narrow-Block Modes

LRW Mode [LRW02]

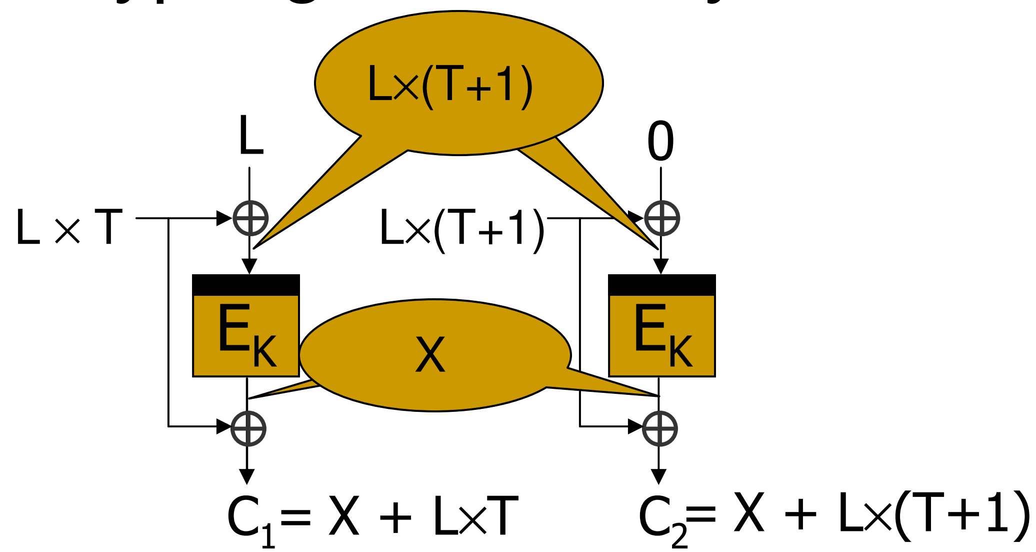
- E_K - AES with key K
- L - another key
- $L \times T$ in $GF(2^n)$



- A handy optimization:
  - Think about using tweaks T, T+1, T+2, …
  - Once $L \times T$ is computed, easy to compute $L \times (T+1)$, $L \times (T+2)$, …
- IEEE 1619 intended to standardize this mode

What's Wrong with LRW?

- Fails when "encrypting its own key" (?)

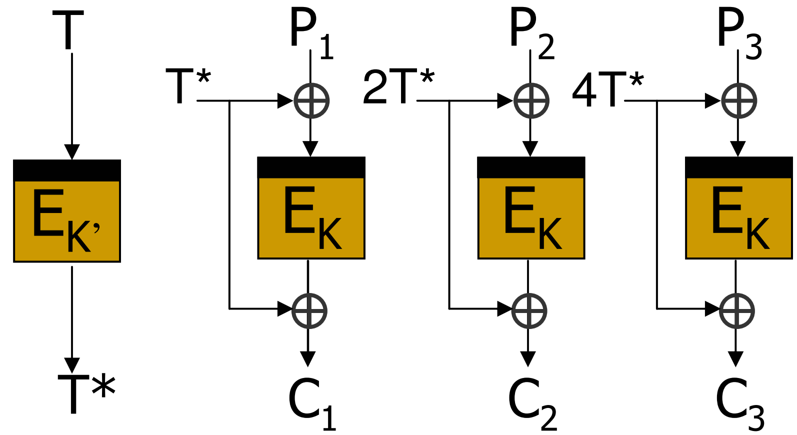


- Extract $L = C_1 - C_2$

Is This a Problem in Practice?

- Lively argument in the 1619 mailing list
  - *"No one in their right mind will ever do that"*
- Turns out that "encrypting own key" can happen, e.g., in Windows Vista™
  - A driver does sector-level encryption
  - On hibernate, driver itself stored to disk
- So a different mode (based on Rogaway's XEX) was chosen for the standard

XTS Mode [Ro04]



- Tweak is (T,i)
  - $T^* = E_{K'}(T)$, $T_i^* = 2^i \times T^*$
  - $C = T_i^* \oplus E_K(P \oplus T_i^*)$
- Similar handy optimization
  - (T,0), (T,1), (T,2), … for sequential blocks
  - About as efficient
- The attack from b
  - How do we know th ere aren't other attacks in this vein?

We'll talk later about circular security

Remaining problems

- Narrow vs. wide-block in practice
  - Wide-block is 2-3 times more expensive
  - Limit attacker to more coarse granularity
    - Traffic-analysis/malleability of whole sectors, rather than each 16-byte block
  - Does this add security in practice?
- Security beyond the birthday bound
  - With big disk-arrays in the petabytes, $q^2/2^{128}$ may get too close for comfort

Authenticated Encryption

- Each record is stored with a nonce (IV), and an authentication tag
  - $Enc_K(P) = \langle IV, C, tag \rangle$
  - $Dec_K(IV, C, tag) = P$ / *fail*
- IVs must be "fresh"
  - Encrypting the same plaintext twice results in a different ciphertexts

Many "standard" Encryption Modes

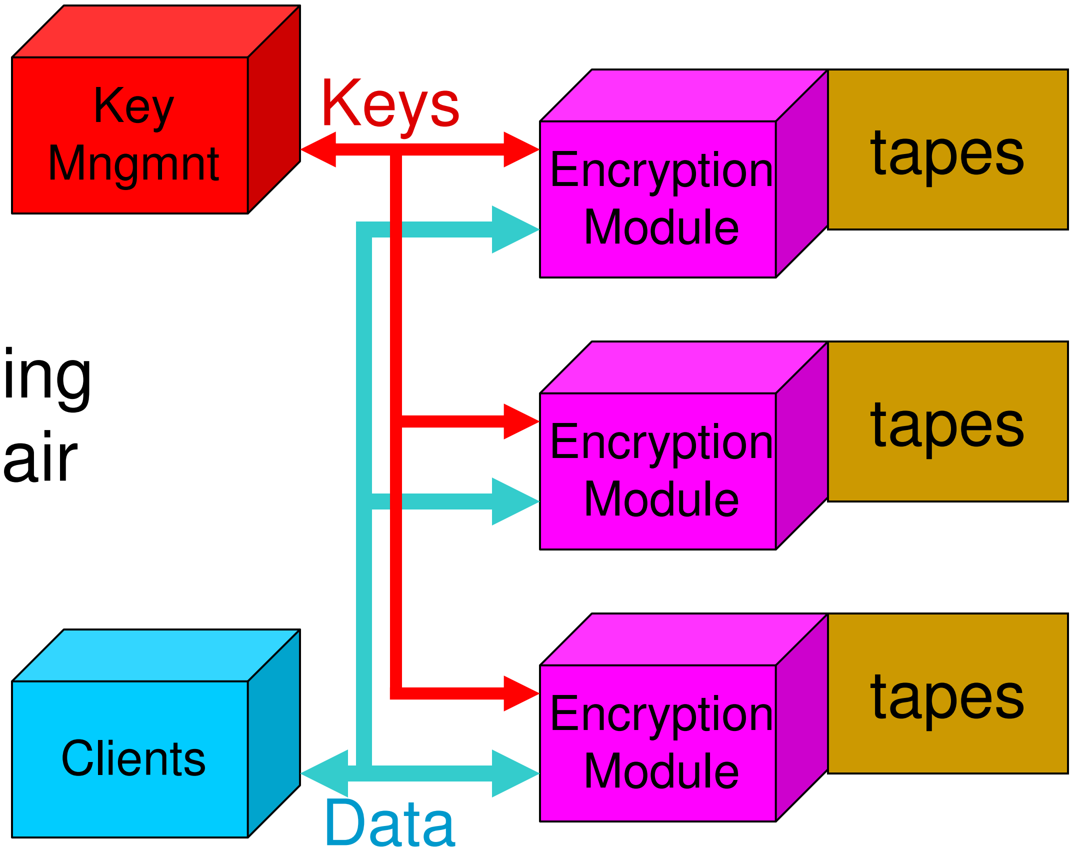
- Two-Pass Modes
  - Encrypt-then-authenticate (e.g., GCM [MV05])
    - Choose IV, $C=E_K(IV, P)$, $tag=MAC_{K'}(IV,C)$
    - E: AES-based encryption, MAC: HMAC or others
  - Authenticate-then-encrypt (e.g., CCM [WHF03])
    - Choose IV, $t=MAC_{K'}(IV,P)$, $C=E_K(IV, P, t)$
- One-Pass Modes (IAPM [J01], OCB [R01],…)
  - Compute CTXT & MAC together, more efficient
  - None is used in practice today ☹
  - Due to patent issues ☹☹

Whence Cometh thy Nonce?

- Re-using the same (key,IV) pair to encrypt different records is a security violation
  - Especially in schemes based on CTR mode
    - Re-using (key,IV) is the same as two-time-pad
  - Especially^2 in GCM mode
    - Re-using (key,IV) may leak the authentication key
- Avoiding nonce re-use may be tricky

Common Tape-Encryption Setting

- Same key can be served to several encryption modules
- They must avoid using the same (key,IV) pair
- Without much coordination



Random Nonces?

- Some modes have 96-bit nonces (GCM)
  - Is this enough?
- How many times can the same key be served? What if you use just one key for all your corporate tapes?

Systematic Nonces?

- E.g., use the module serial # in the nonce
  - Reduces the IV space further
  - Sensitive to mis-configuration
  - Module must remember “the current nonce”
    - Through reset, power-failures, crashes, …
- Using encryption modules from several different manufacturers?
  - An organization may have two drives from IBM, one from HP, one from SUN, etc.

Better: Wrapped Keys

- The served key (from key-management) is only used as a key-encrypting-key (KEK)
  - Module generates a “fresh” data key (DK)
  - Use KEK to encrypt DK, store ciphertext on tape
  - Use DK to encrypt data
- David Wheeler: *All problems in computer science can be solved by another level of indirection…*
  … but that usually will create another problem.

How to Wrap Keys?

- Using standard encryption (symmetric/pkey)
  - Need to worry again about fresh IVs / randomness
- Using "deterministic encryption"
  - E.g., ANS X9.102 draft standard
- [RS06]: Deterministic Authenticated Encryption
  - Essentially "the strongest security possible with deterministic encryption"
    - Similar to strong PRP, but need not be a bijection
  - SIV mode: $IV = PRF_{k1}(DK)$, $C = CTR_{k2}(IV, DK)$

More on Key-Wrapping [GH08]

- Some “secure schemes” are not DAE
  - DAE an overkill for wrapping encryption keys
- Secure key-wrap is just like secure encryption, except the plaintext is random
  - Rather than adversarially chosen
- Hash-then-Encrypt: “SIV-like” constructions
  - IV = Hash(DK), C = ENC(IV, DK)
  - Hash either keyed or not
  - ENC any “standard encryption mode”

Hash-then-Encrypt

Hash Encrypt	XOR	Linear	Universal	2nd preimage
CTR	✘	✘	✔	✔
ECB	✘	~~✔~~	✘	✔*
CBC	✘	~~✔~~	?	✔*
Masked ECB/CBC	~~✔~~	~~✔~~	✔	✔
XEX	✔	✔	✔	✔

Remaining Problems

- Authenticated Encryption does not solve:
  - "Replay attacks:" replace current record on medium with a previous one
  - Re-ordering of records
- No good crypto solutions to either problem
  - Merkel trees work, but they are too expensive
  - Not clear that one can do better [DNRV08]

Back to “Key-Dependent Security”

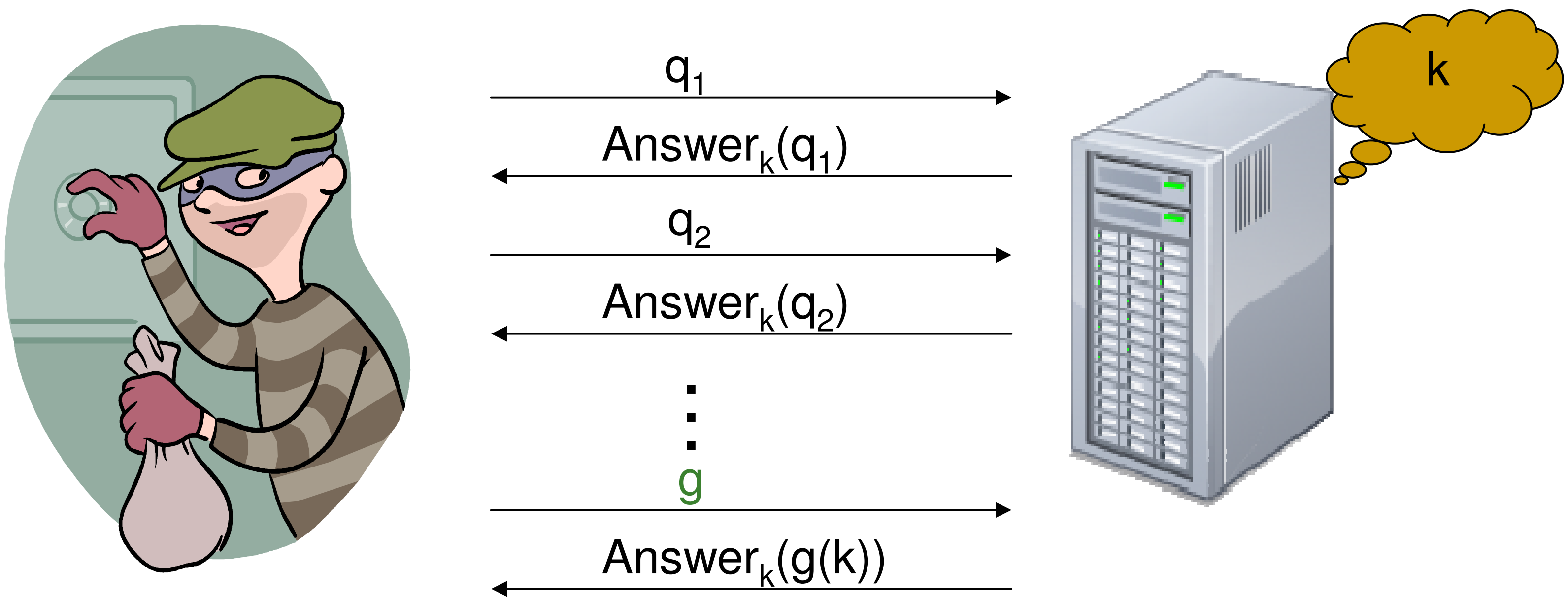
- Adversary sees encryptions of the secret key
  - Maybe even some functions of this key
- How to define security in this case?
- How to achieve it?

Aside:

- The definitional issue was noted already in [GM84], but explicitly scoped out
- [CL01] had a “key-dependent-secure” public-key encryption in the ROM

[BRS01] Definitions

- Start from the “usual notions”



- Let the attacker specify a function of the key

[BRS01] Construction

- Textbook scheme: $Enc_k(m) = \langle r, f_k(r) \oplus m \rangle$
- With $f_k(x) = H(k|x)$ and H a random oracle, this is "key-dependent-secure"
- As usual: in lieu of a true random oracle, we can use, e.g., SHA1
  - $f_k(x)$ = SHA1-Compression(IV=k, M=x)
  - This should be safe…

[HK07] Insecurity in Standard Model

- SHA1 follows the Davis-Meyer approach
  - Roughly Compression(IV,M) = $E_M(IV) \oplus IV$
  - E is a “block cipher” (easily invertible given M)
  - SHA1 actually uses + rather than $\oplus$
    - But we will ignore that fact
- We get $Enc_k(m) = <r, E_r(k) \oplus k \oplus m>$
  - In particular $Enc_k(k) = <r, E_r(k) \oplus k \oplus k>$
  - Given $<r,c>$ recover $k = E_r^{-1}(c)$ ☹

Key-dependent security w/o ROM?

- [HH'08]: Unlikely from "general assumptions"
- [BHHO'08]: But possible from DDH
- Think ElGamal Encryption:
  - $pk=(v,w=v^a)$, $sk=a$, $Enc_{pk}(m)=<v^r, m \times w^r>$
  - So $Enc_{pk}(sk)=<v^r, a \times v^{ar}>$
    - Security unlikely to follow from DDH
- What if we use $sk=u^a$ $(u \neq v)$?
  - We get security from DDH, but cannot decrypt…

Decrypting with "sk in the exponent"?

- Use single bits in the exponent for secret key
  - Can recover b from v^b
- $pk = (v_1\ v_2 \ldots v_m\ w=\Pi\ v_i^{b_i})$
  $sk = (u^{b_1}\ u^{b_2} \ldots u^{b_m})$
  $\mathrm{Enc}_{pk}(m) = (v_1^r\ \ v_2^r \ldots v_m^r\ \ m\times w^r)$
  - So $\mathrm{Enc}_{pk}(u^{b_i}) = (v_1^r\ \ v_2^r \ldots v_m^r\ \ u^{b_i}\times w^r)$

Thm: This is CPA-secure against encryptions of any affine function of the secret key

- [CCS08] build on this to get CCA-security

Morals to take away

- Applying crypto to real-world systems is fun
  - Can even find interesting questions to look at
- 1st law of commercial crypto: *"cryptosystems will be (ab)used beyond their security model"*
- We still do not know everything there is to know about encryption
- Storage encryption is (a little) special
  - Mostly: harder to get synchronization between encryptor and decryptor

Thank you