Attacks and Security Proofs of EAX-Prime

Kazuhiko Minematsu, NEC Corporation
Stefan Lucks, Bauhaus-Universität Weimar
Hiraku Morita, Nagoya University
Tetsu Iwata, Nagoya University

Fast Software Encryption 2013, March 10 -- 13, 2013, Singapore

Authenticated Encryption (AE)

- Authentication + Encryption
- Prevents eavesdropping and forgery
- Widely used in practice
  - Internet (Wifi, SSL/TLS), storage, mobile, satellite, and many more

EAX-Prime (EAX')

- AE based on AES
- Defined at ANSI C12.22
  - Smart grid / Smart meter Protocol
  - also appears at IEEE 1703 and MC1222 (Canada)
  - proposed to NIST in 2011
- Some real products, e.g. smart meters and their management systems

EAX and EAX-Prime

- EAX-Prime is derived from EAX
- EAX
  - developed by Bellare, Rogaway, and Wagner at FSE 2004
  - has a proof of security
- EAX-Prime
  - modified version of EAX
  - some “optimizations” : reducing # of blockcipher calls and the size of memory
  - no formal analysis

Our Results

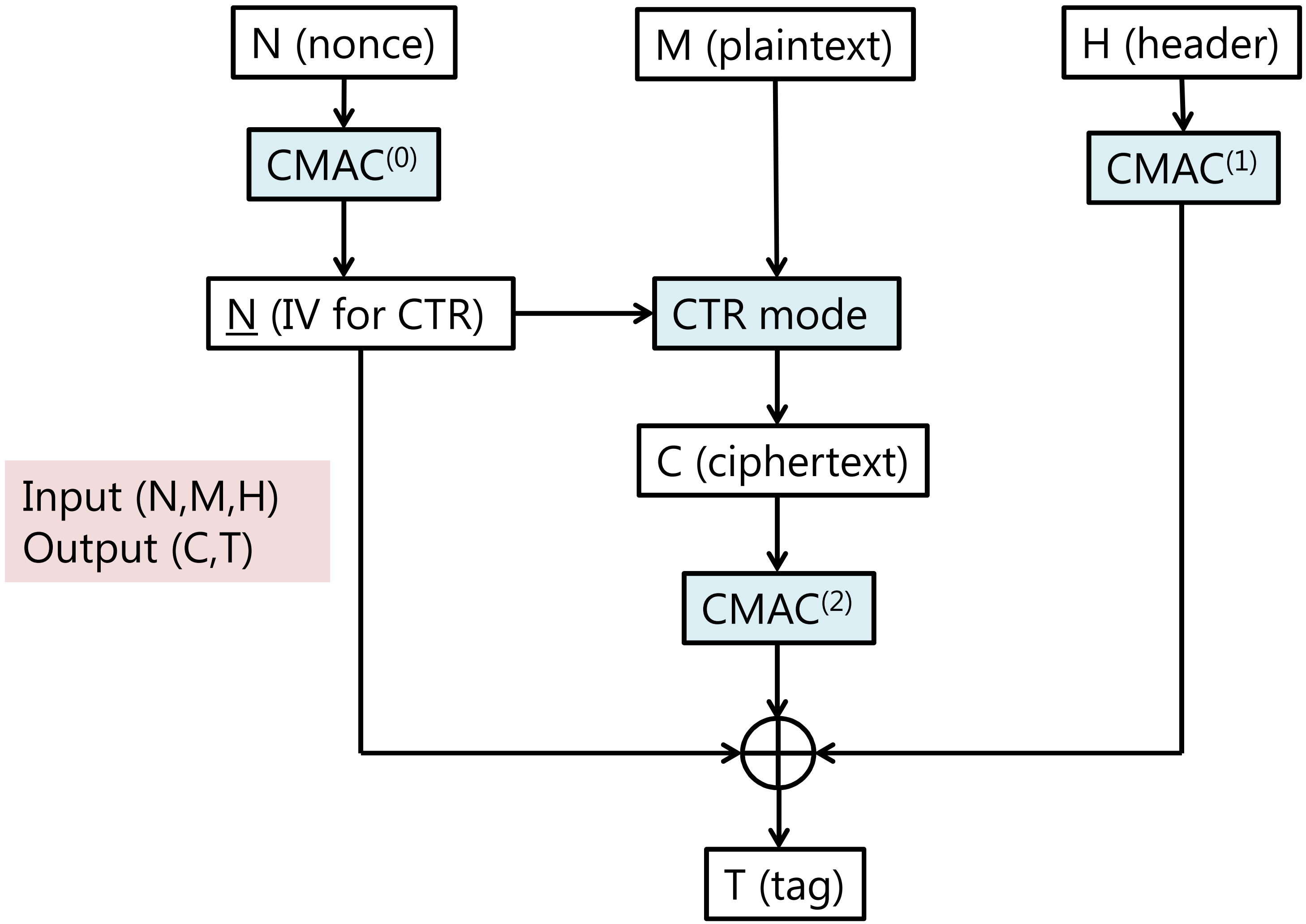
- Security of EAX-Prime is sharply separated w.r.t. *cleartext* (an input variable), as we show ;

1. When cleartext is one-block, effective attacks exist
   - Forgery, distinguisher, and plaintext recovery

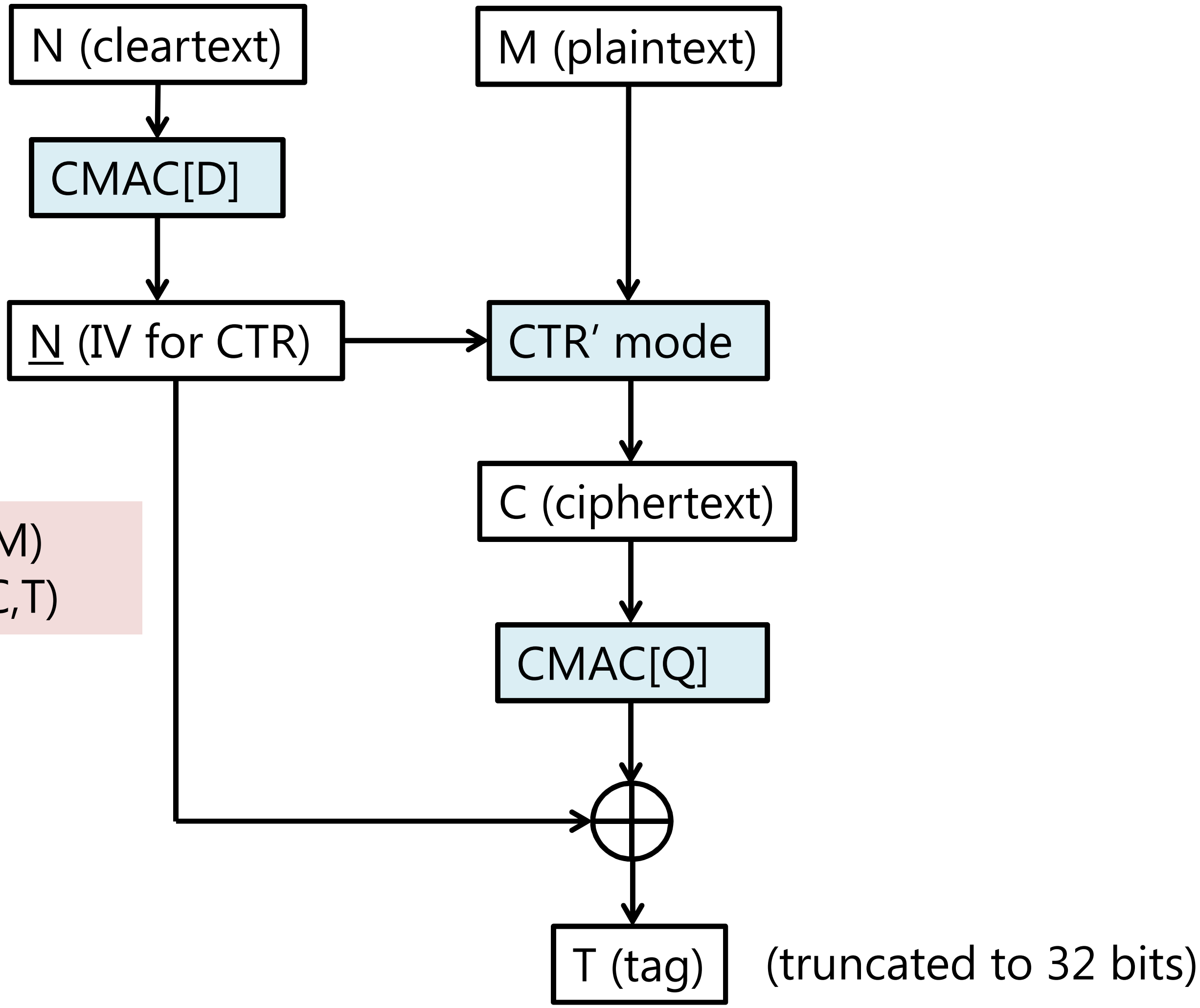
2. When cleartext is more-than-one-block, it has a proof of security based on the standard assumption

(Original) EAX Encryption

- Enc-then-Auth, by CTR and CMAC
- CMAC is tweaked (creating 3 variants)

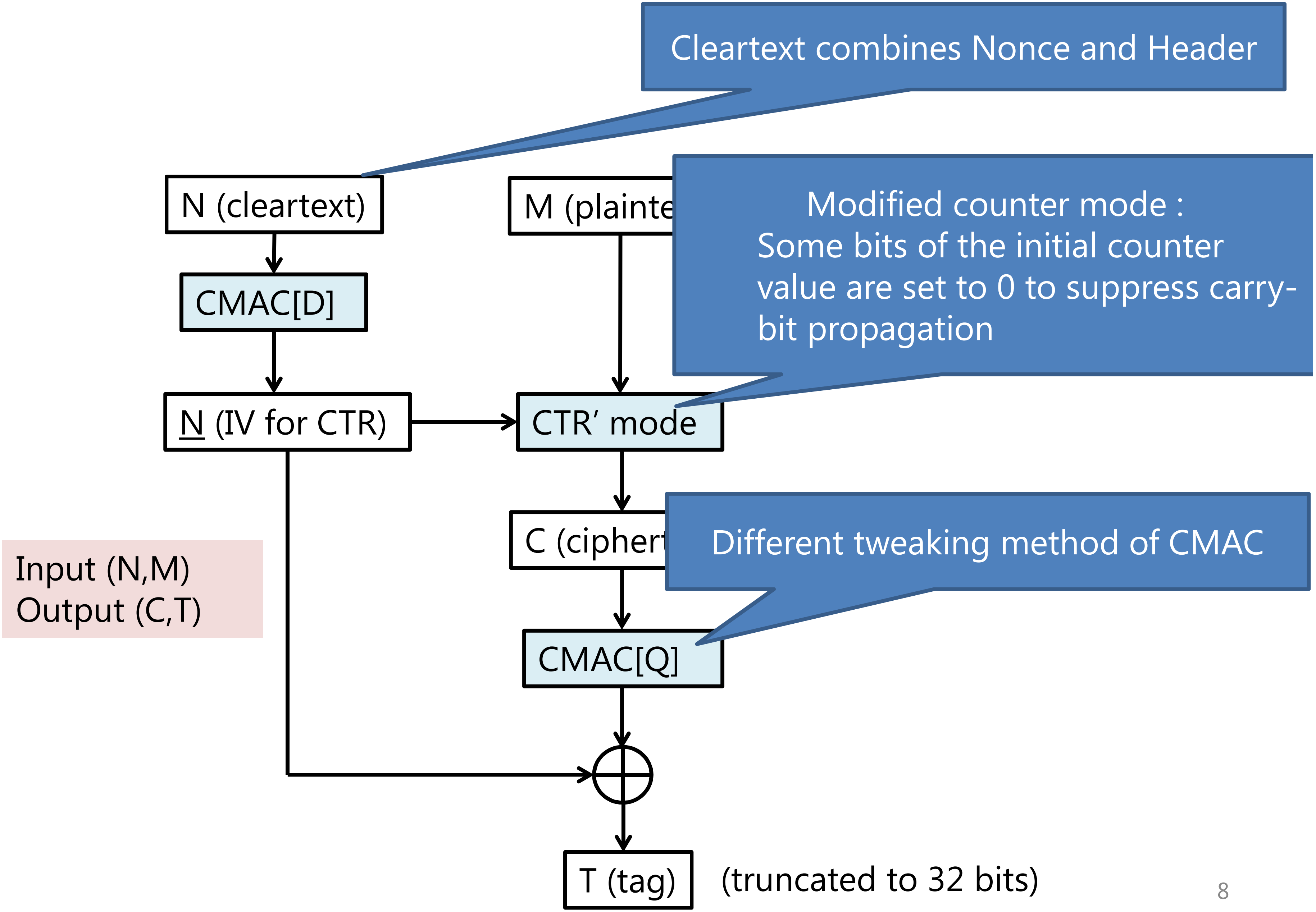


EAX-Prime Encryption



Input (N,M)
Output (C,T)

EAX-Prime Encryption

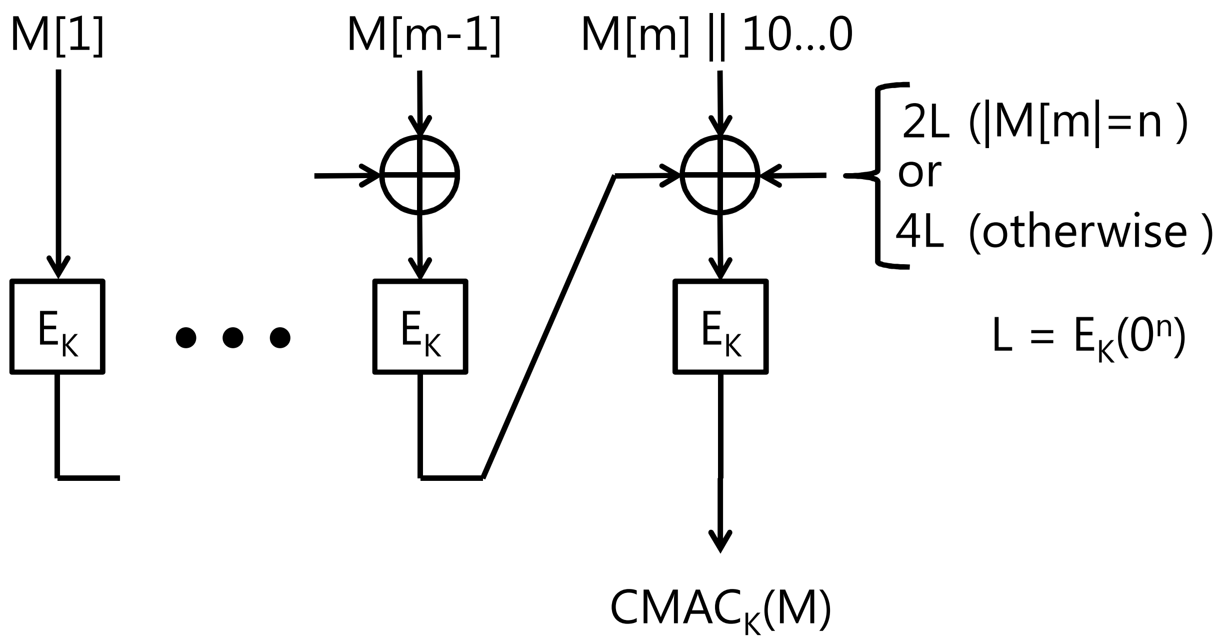


Tweaking Method of CMAC

- CMAC[D] and CMAC[Q]
  - 2 variants
  - Slightly more efficient than the original
  - … and makes our attacks possible

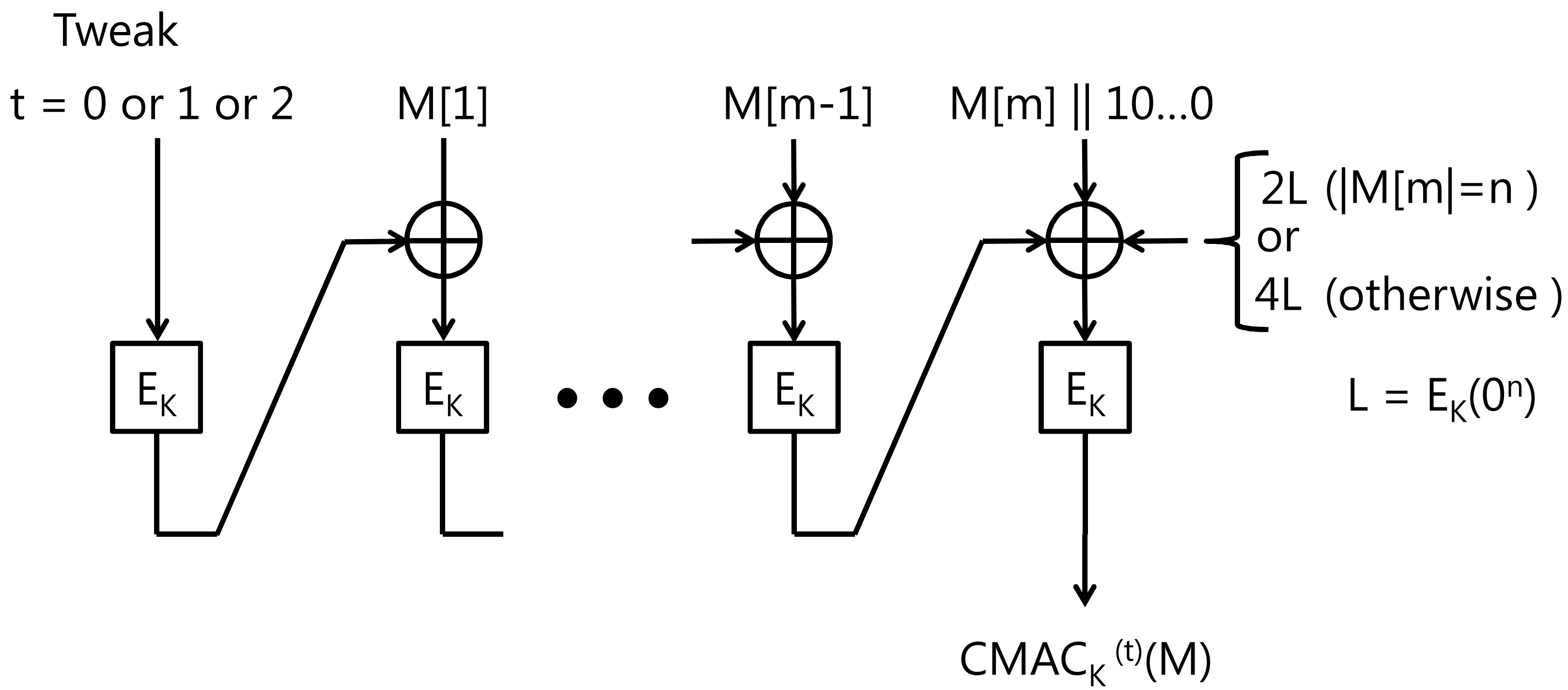
CMAC (NIST SP800-38B)

- CBC-MAC w/ last masking 2L or 4L
- $L = E_K(0^n)$
- 2L : Doubling in $GF(2^n)$, 4L : Twice Doubling



Tweaked CMAC in EAX

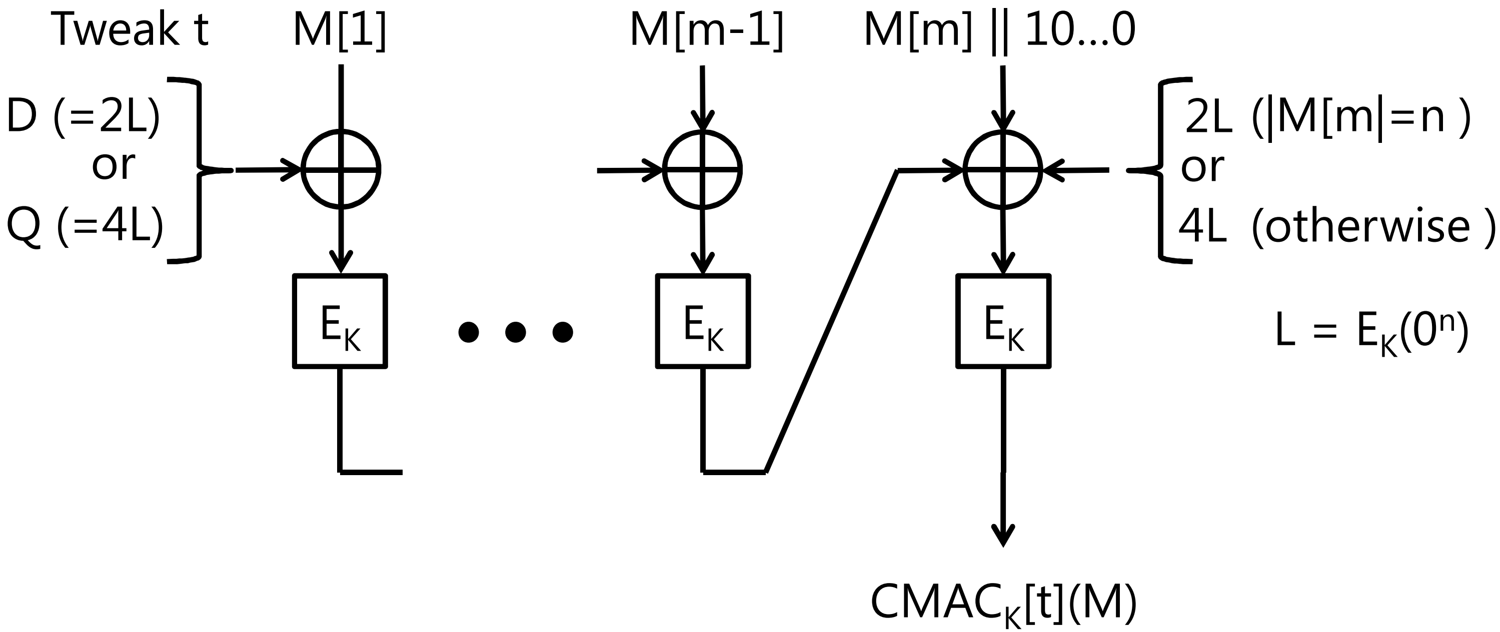
- 3 variants with $\text{CMAC}^{(\text{tweak})} = \text{CMAC}(\text{tweak} \,||\, X)$, tweak = 0,1,2 (in n bits)
  - E_K(tweak) can be cached as initial mask



Tweaked CMAC in EAX-Prime

- 2 variants with CMAC[D] and CMAC[Q]

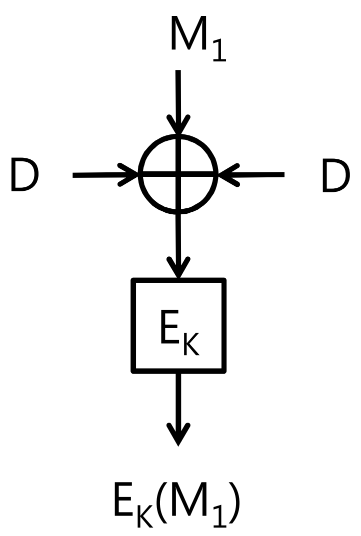
  (tweak = D, Q)
- Use D=2L or Q=4L as initial mask



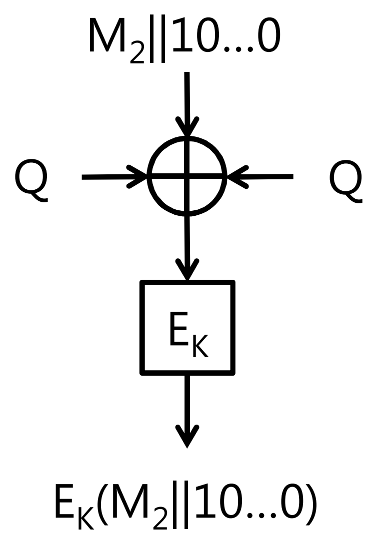
Observation

- CMAC[D] and CMAC[Q] fail to provide (independent) PRFs
- In case $|M| \leq n$;

CMAC[D] when $|M_1|=n$



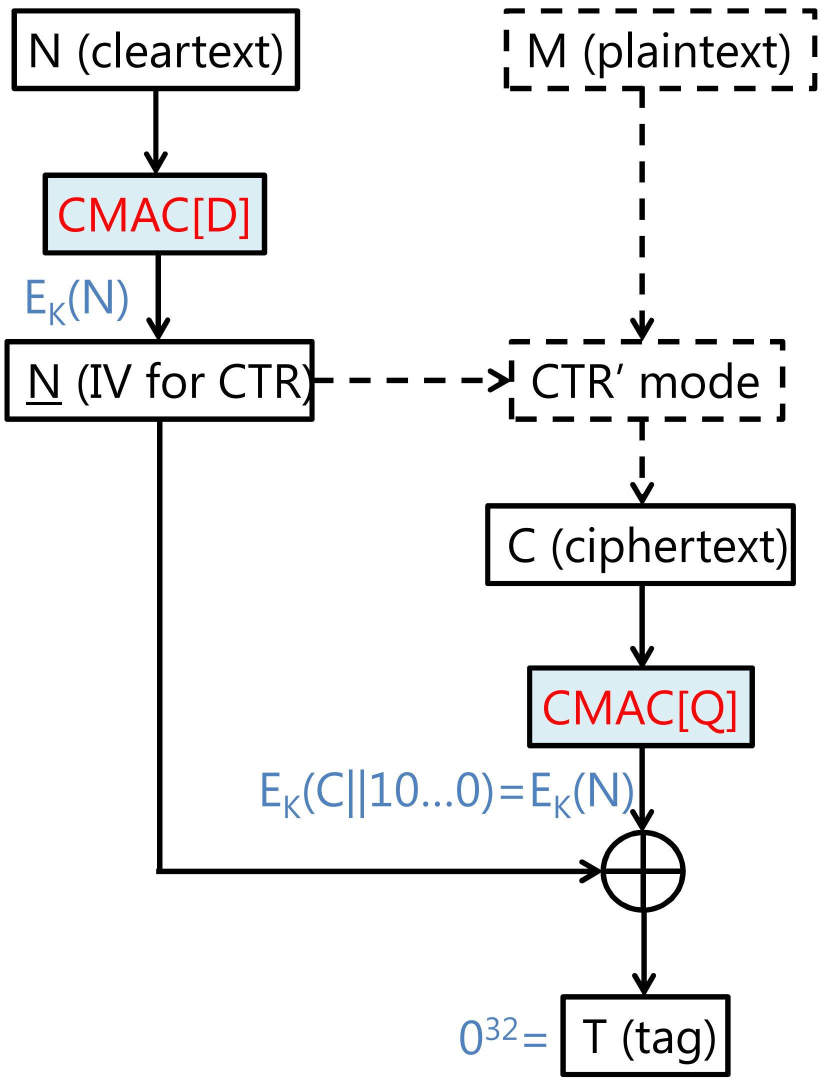
CMAC[Q] when $0 \leq |M_2| < n$



Making $M_1 = M_2||10...0$ yields the same outputs -> unlikely for two independent PRFs

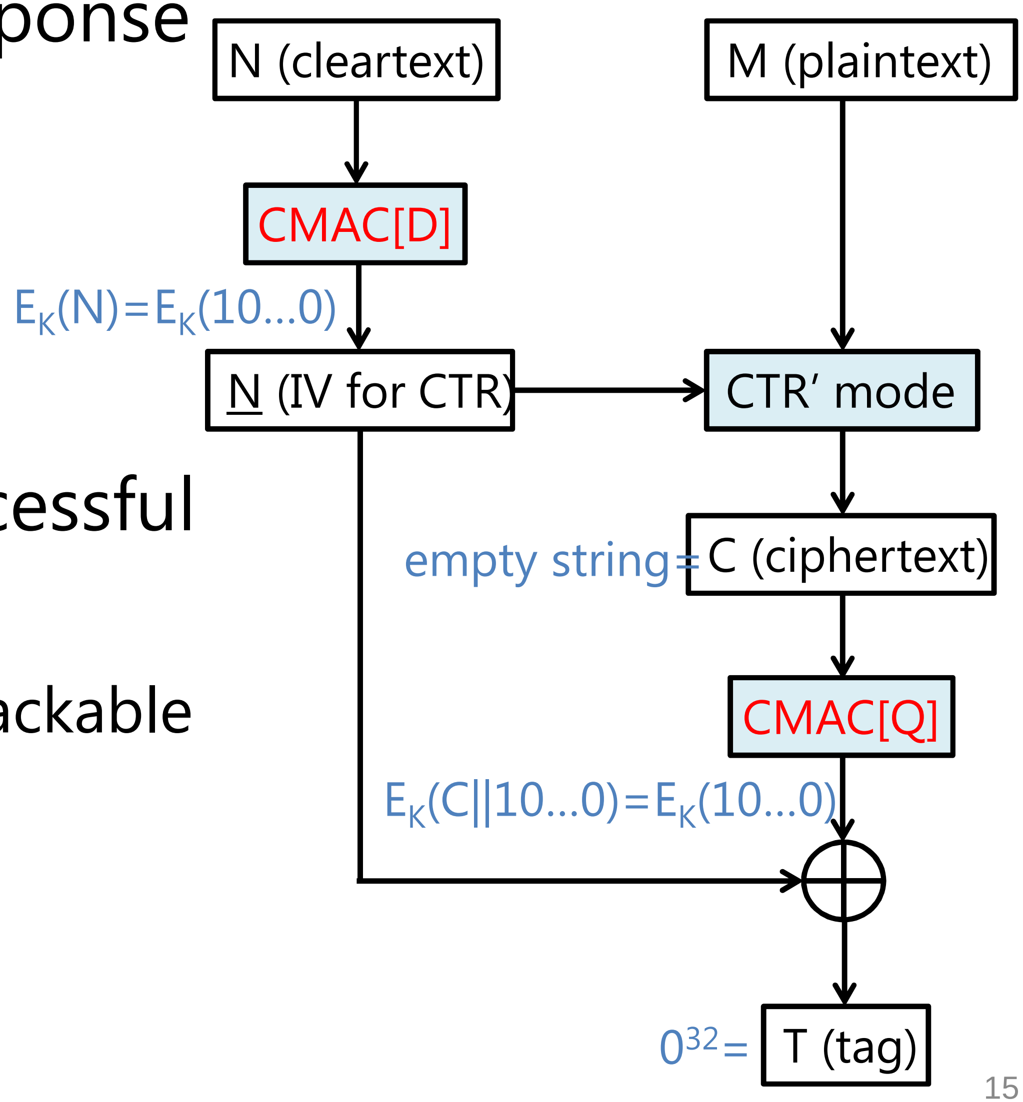
Forgery Attack

- Throw (N,C,T) to the decryption oracle;
  - $|N| = n$, $|C| < n$
  - $C||10..0 = N$
  - $T = 0^{32}$
- always successful
- No enc-query
- Dec-oracle sees random plaintext, giving a great speculation for attack (thanks to Greg Rose)
- Variants
  - $|N|<n$ & $|C|=n$ etc.



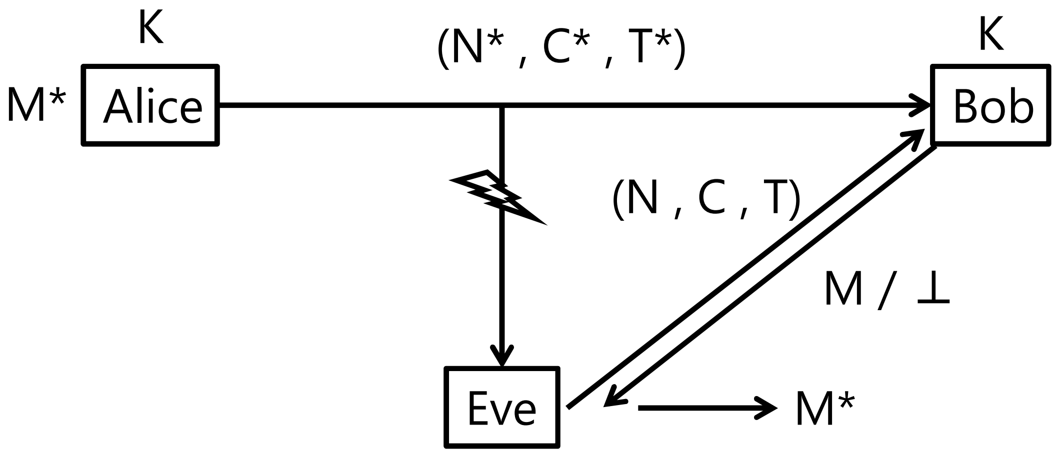
Distinguishing Attack

- One enc-query to distinguish the response from random
  - |N| =n, N = 10..0
  - |M| = 0 (empty)
- See if T = 0^{32}
- almost always successful
- Variants
  - short M is also attackable



(Chosen-Ciphertext) Plaintext Recovery

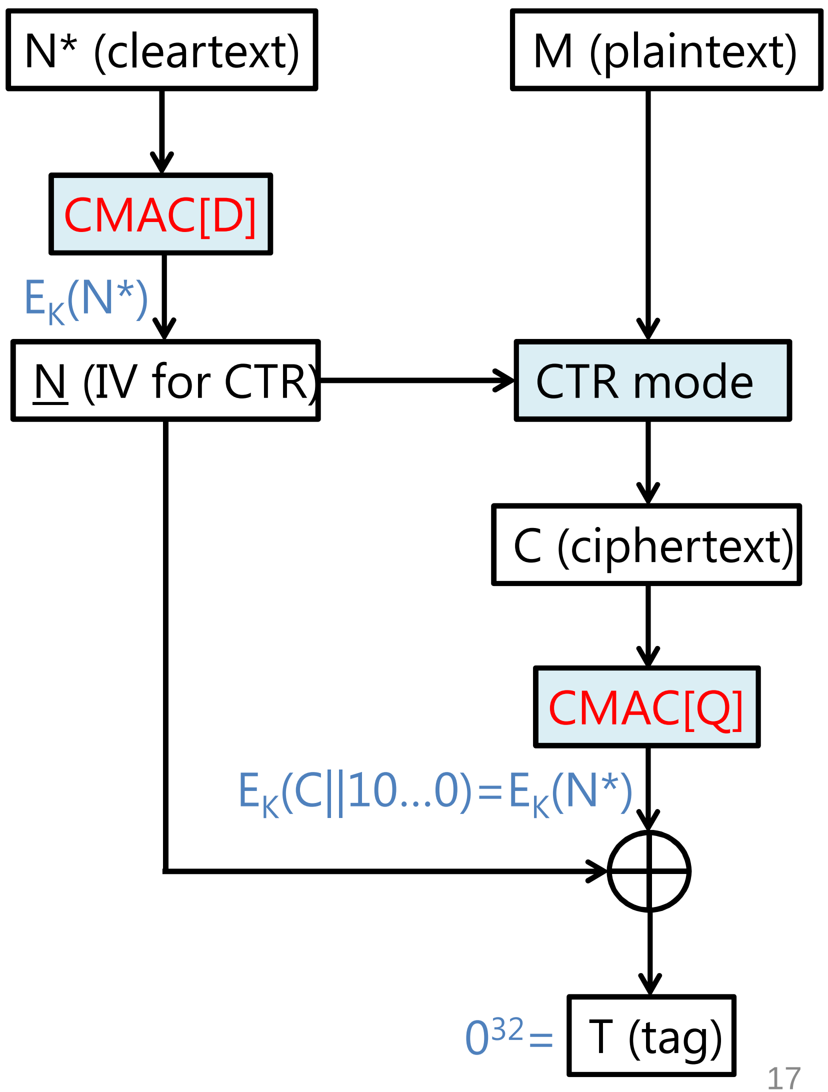
- Scenario
  - Eve eavesdrops (N*, C* , T*)
  - corresponding M* is unknown
- Eve can ask *other* (N , C , T) to Bob (Dec-oracle)
- The goal is to find (a part of) M*



(Chosen-Ciphertext) Plaintext Recovery

1. Suppose (N^*, C^*, T^*) satisfies $|N^*|=n$, $|C^*|<n$
2. Do Forgery attack with $N=N^*$, C s.t. $C||10..0 = N^*$
3. Dec-oracle returns $\tilde{M}$
4. $KS = C \oplus \tilde{M}$ is the keystream for N^*
5. M^* is recoverd as $KS \oplus C^*$

- If $|C^*| \geq n$, it still recovers the first $|C|$ bits of M^*
- Succeeds with probability 1



Applicability to ANSI C12.22

- All attacks require one-block cleartext ($|N| \leq n$)
- Is this possible in C12.22 ?
- We have no clear answer (despite some efforts)
- Cleartext-length check is needed anyway
  - for both encryption and decryption sides

Applicability to ANSI C12.22

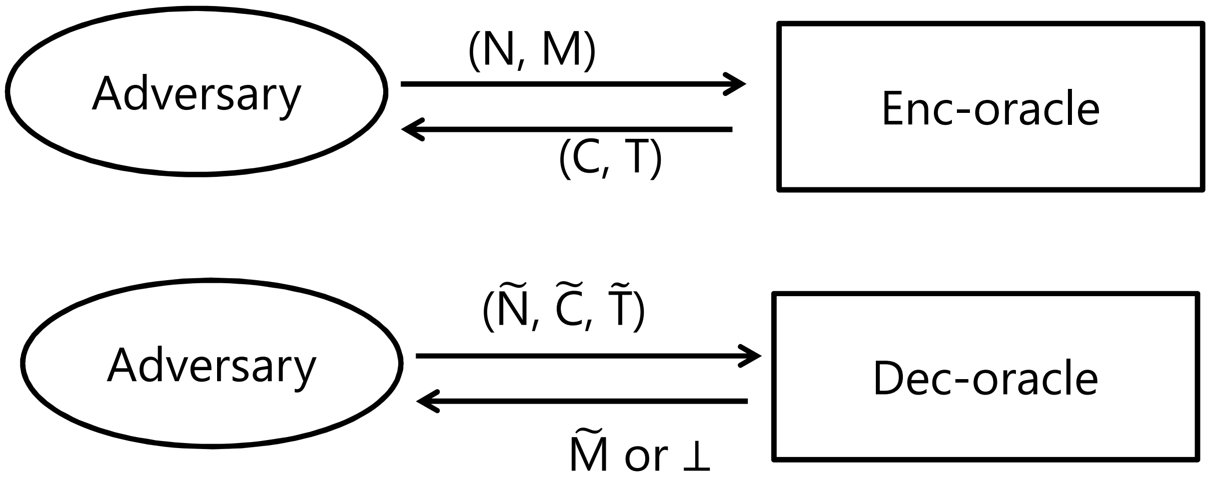
- All attacks require one-block cleartext ($|N| \leq n$)
- Is this possible in C12.22 ?
- We have no clear answer (despite some efforts)
- Cleartext-length check is needed anyway
  - for both encryption and decryption sides

- Is EAX-Prime secure if $|N| > n$ is guaranteed ?

-> Yes, it is provably secure

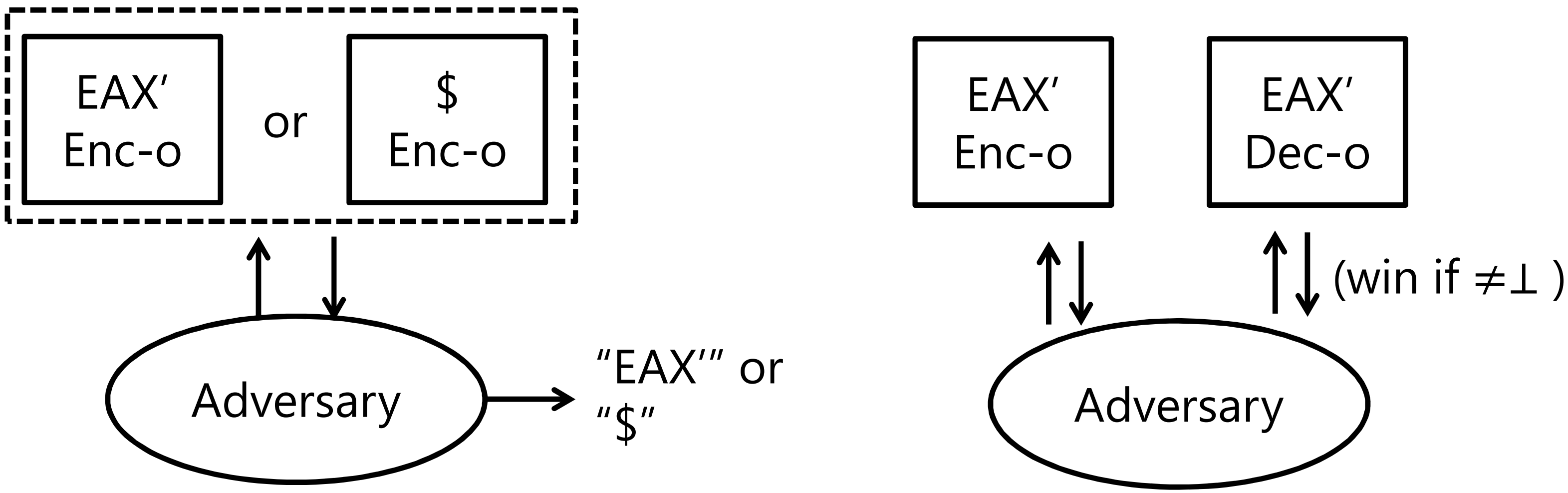
Problem Setting

- Adversary queries to :
  - Enc-oracle : takes (N,M), returns (C,T)
  - Dec-oracle : takes $(\tilde{N}, \tilde{C}, \tilde{T})$, returns $\tilde{M}$ or $\perp$
- Cleartext has at least two blocks ($|N|, |\tilde{N}| > n$)
- Any enc-query (N,M) is allowed provided N is unique (nonce-respecting)
  - dec-query has no such limitation



Security notions

- Two (standard) notions
- Privacy (PRIV) : ciphertexts are pseudorandom
  - Distinguish two Enc-oracles, EAX' and random ($)
- Authenticity (AUTH) : a successful forgery is hard
  - Receiving (non-trivial) $\neq\perp$ response from Dec-oracle



Security Bounds

- Our results (w/ n-bit random perm., τ-bit tag)
- Privacy

$$\mathtt{Adv}^{\mathtt{priv}}_{\mathrm{EAX}'[\mathrm{Perm}(n),\tau]}(\mathcal{A}) \leq \frac{18\sigma^2_{\mathrm{priv}}}{2^n}$$

EAX' specifies $\tau = 32$

σ_{priv} : Total blocks of N and M

- Authenticity

$$\mathtt{Adv}^{\mathtt{auth}}_{\mathrm{EAX}'[\mathrm{Perm}(n),\tau]}(\mathcal{A}) \leq \frac{18\sigma^2_{\mathrm{auth}}}{2^n} + \frac{q_v}{2^\tau}$$

q_v : # of dec. queries

σ_{auth} : Total blocks of N, M, $\widetilde{N}$, and $\widetilde{C}$

Proof Strategy

1. Redefine EAX’ as a mode of “OMAC-e(xtension)”

   * a pair of functions (OMAC-e(0), OMAC-e(1))

2. Prove OMAC-e is a pair of (computationally) independent PRFs

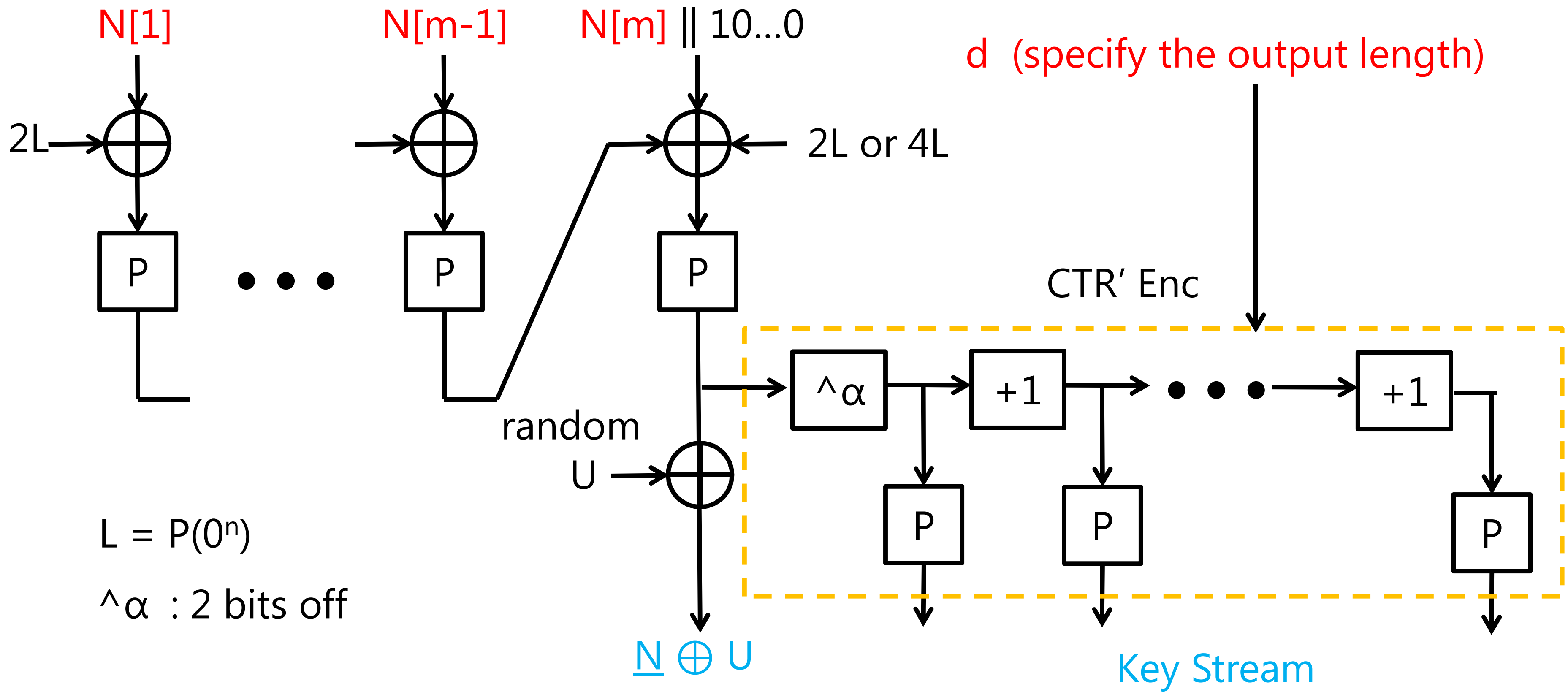
   * Most technical part

3. Prove the security of EAX’ with perfect OMAC-e (pair of random. functions)

   – Following the original EAX proof [BRW04], with some techniques from OMAC proofs [Iwata-Kurosawa 03a, 03b]

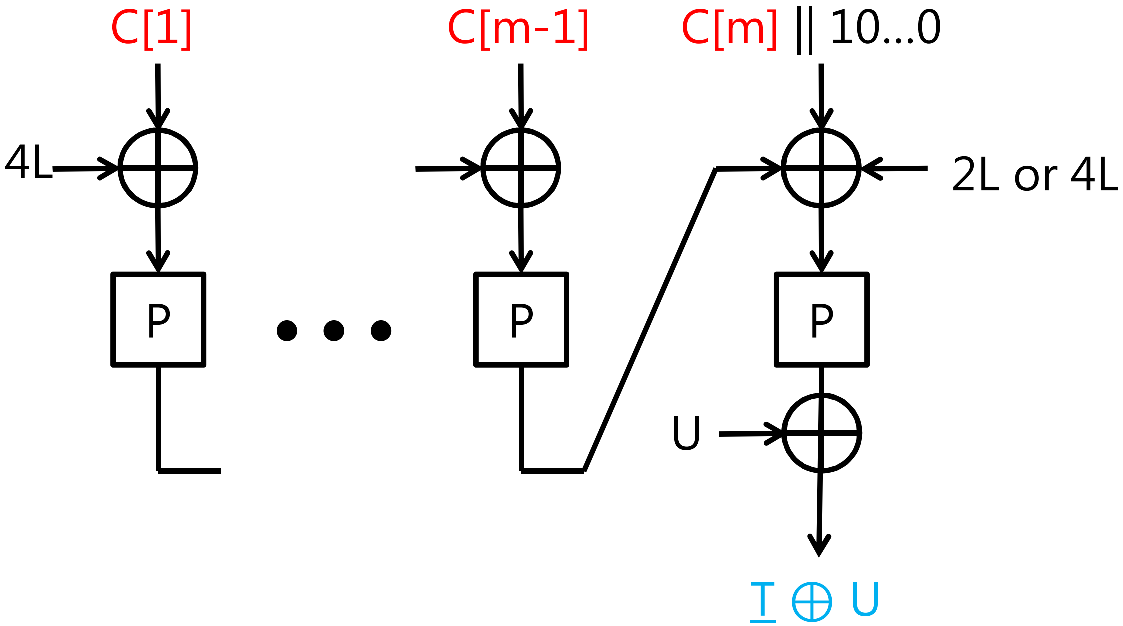
OMAC-e(0)

- Uses an n-bit random permutation P and a random value U
- Computes CMAC[D] and CTR′ (key stream computation, given the output length)
- Input > n bits



OMAC-e(1)

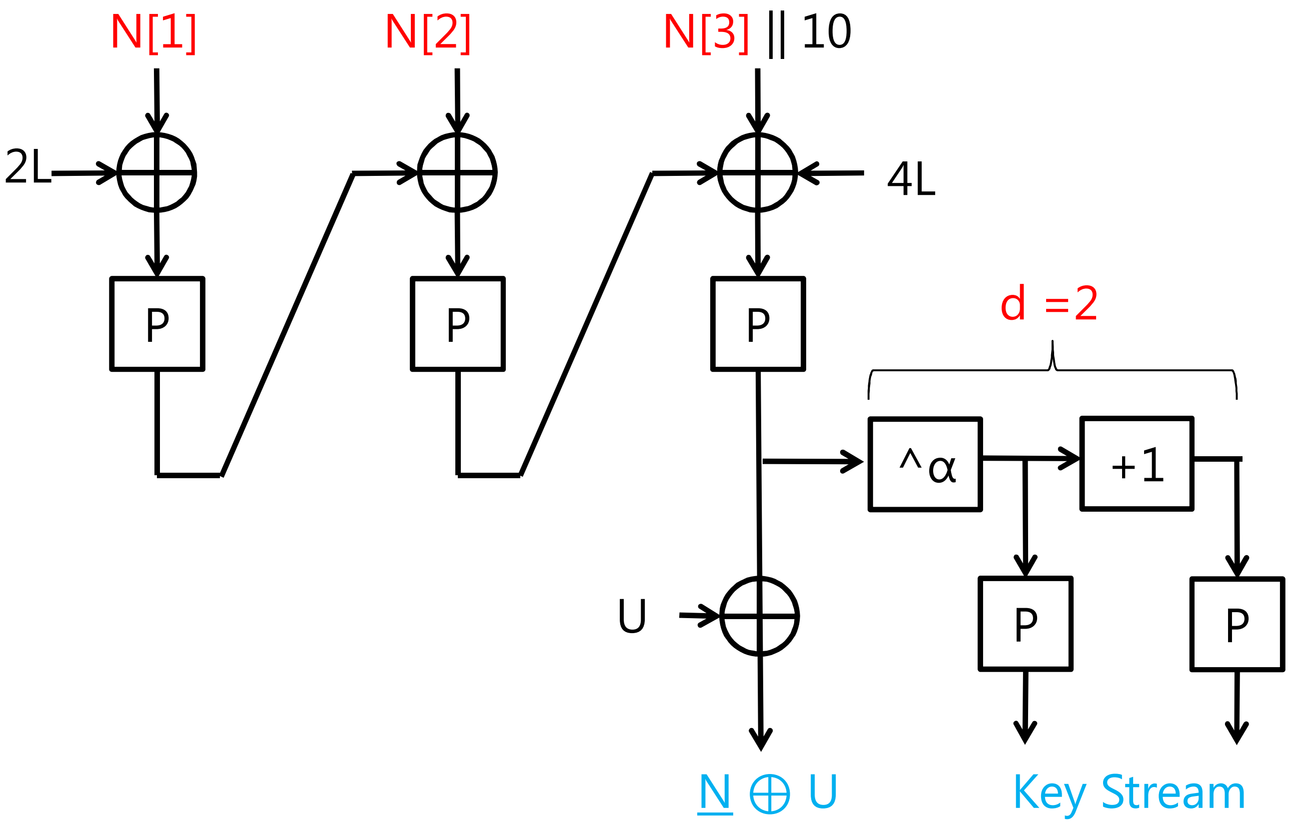
- Computes CMAC[Q]
- Use the same U as in OMAC-e(0)



- OMAC-e can simulate EAX-Prime (U is canceled out)
- Disclaimer : the use of U is missing in the pre-proceeding (thus buggy).
  Proceeding version (and a forthcoming full version) will fix this

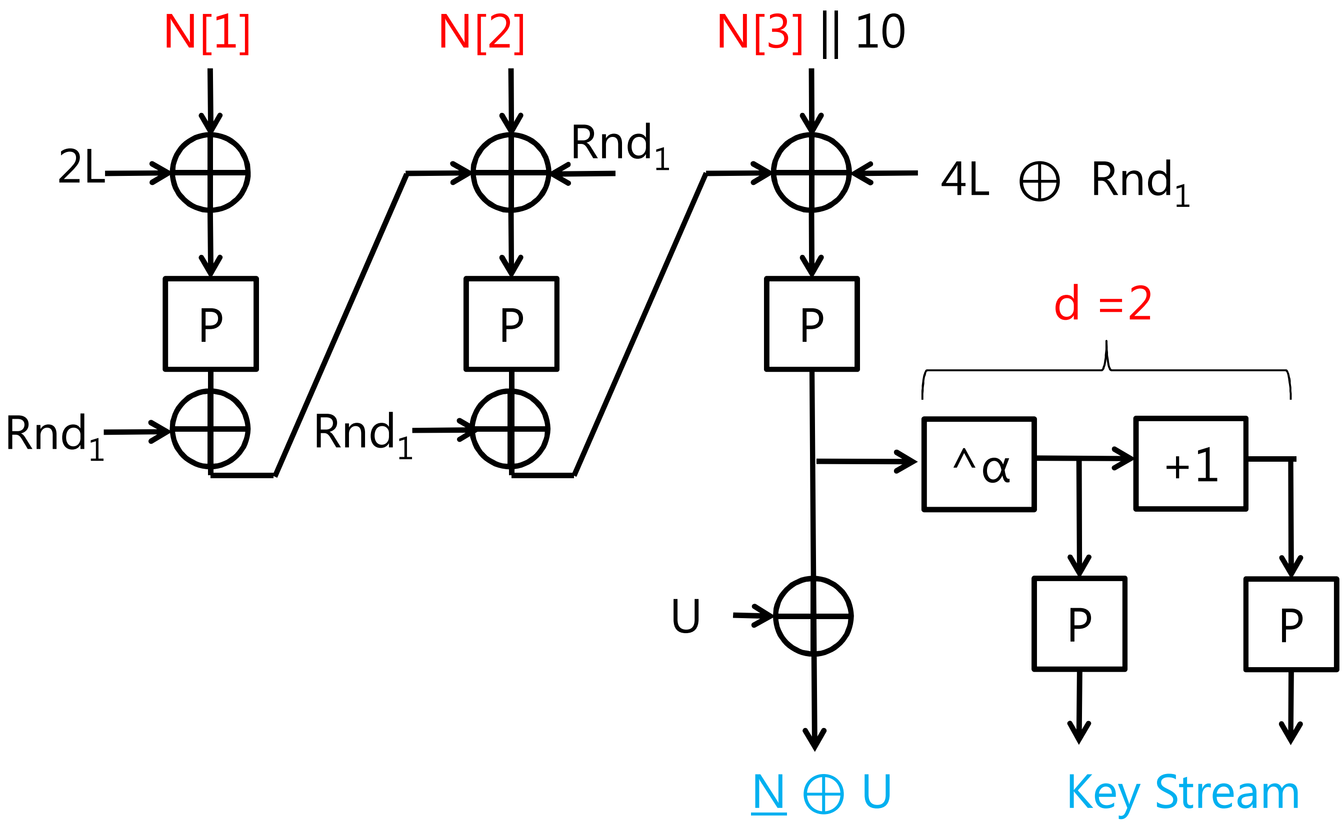
Decomposition of OMAC-e

- We need to prove "OMAC-e is a pair of random functions"



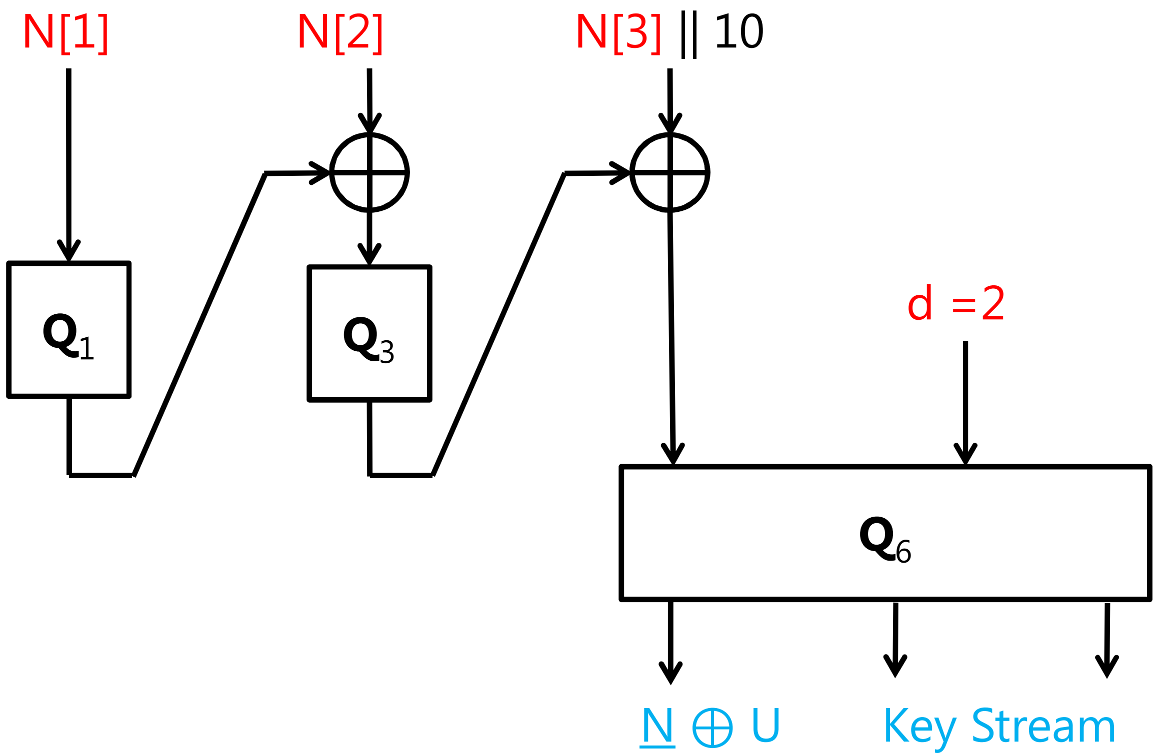
Decomposition of OMAC-e

- We need to prove "OMAC-e is a pair of random functions"
- For this we introduce helper random variables



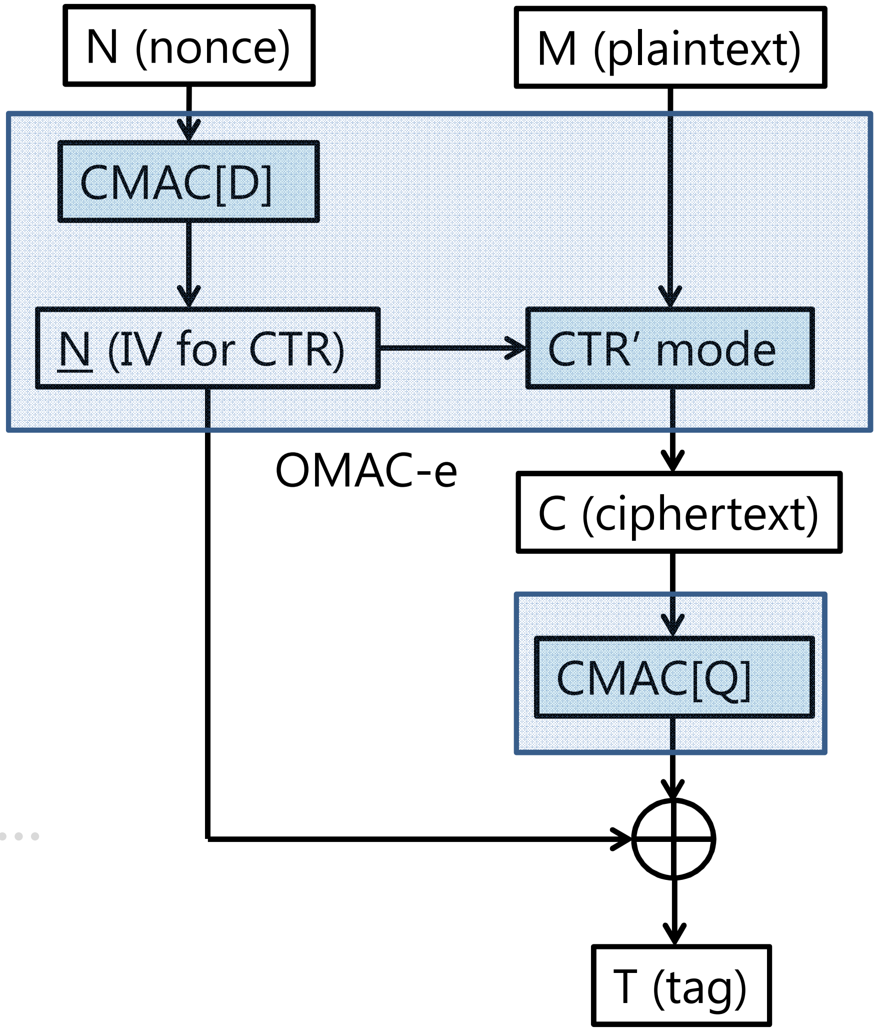
Decomposition of OMAC-e

- and decompose it into a set of ten functions, **Q** = {$\mathbf{Q}_1$, … , $\mathbf{Q}_{10}$}, including the helper variables
- Proving "**Q** = set of rand. functions" is rather easy



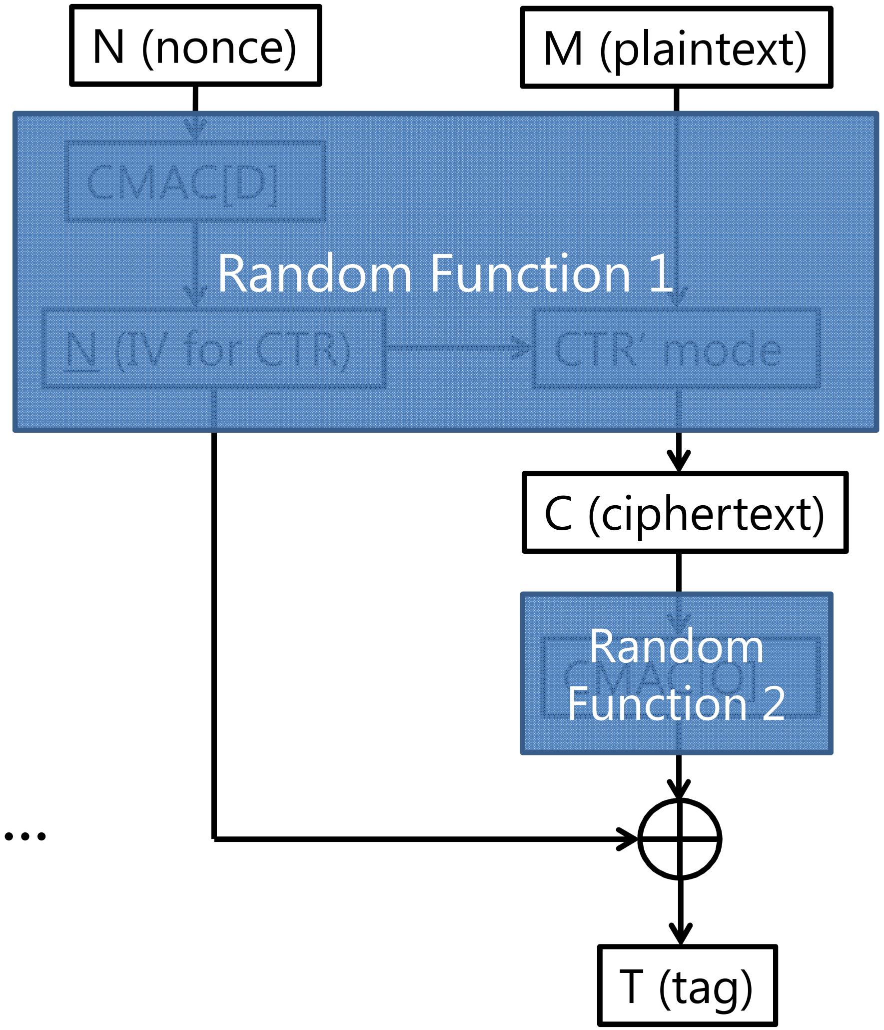
Finalization

- OMAC-e is simulatable by **Q**
- **Q** is indistinguishable from **R** (set of rand. functions)
- OMAC-e simulated by **R** is indistinguishable from a pair of rand. functions
- AE by a pair of rand. functions behaves ideally, the proof goes…



Finalization

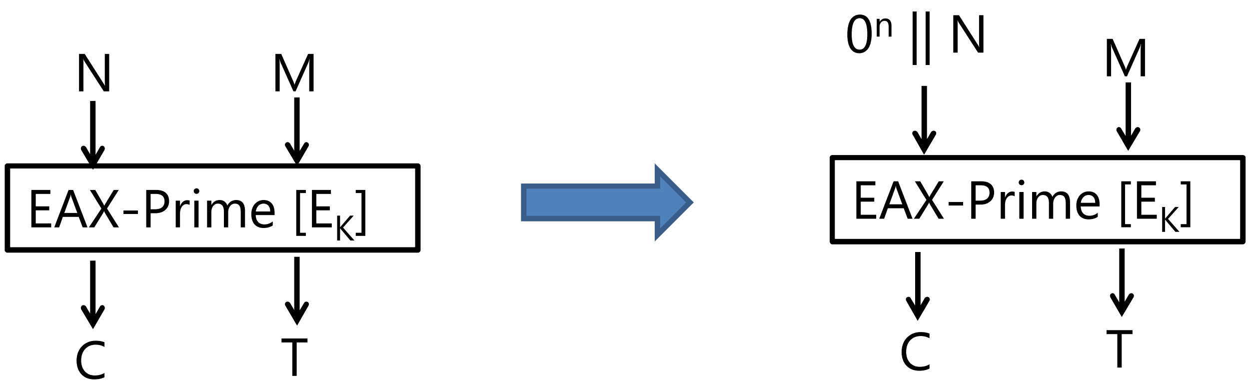
- OMAC-e is simulatable by **Q**
- **Q** is indistinguishable from **R** (set of rand. functions)
- OMAC-e simulated by **R** is indistinguishable from a pair of rand. functions
- AE by a pair of rand. functions behaves ideally, the proof goes…



How to safely use |N| ≤ n ?

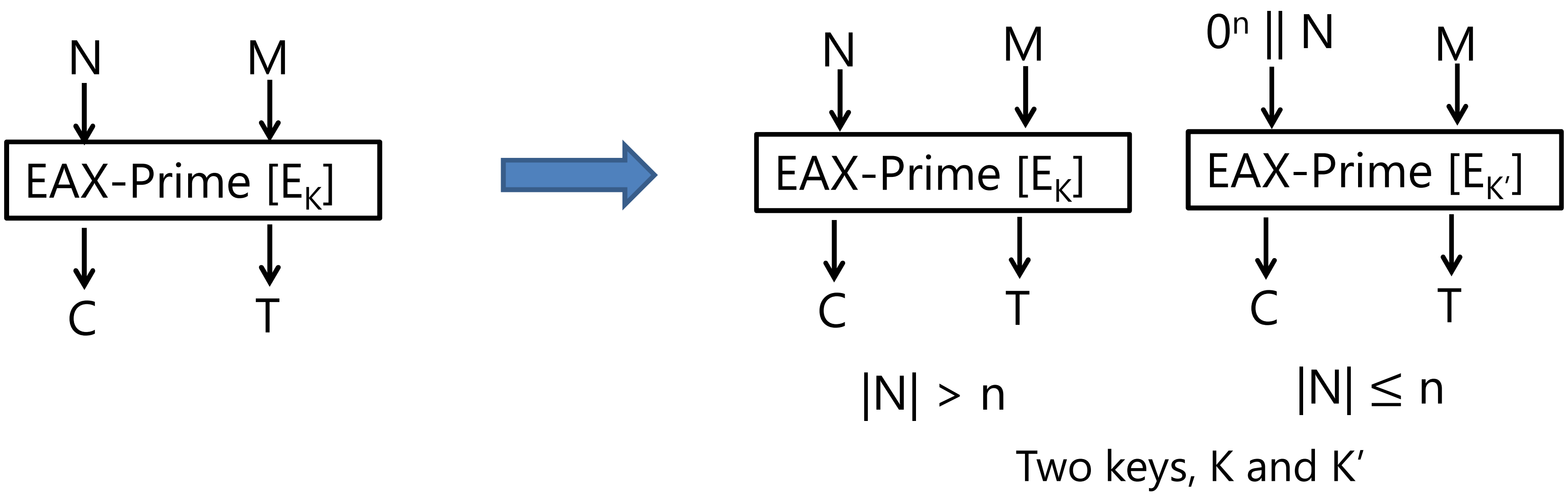
- Suppose we do not want to change the algorithm of EAX-Prime

- Method 1. Prepend to N, e.g. $0^n||N$ instead of N



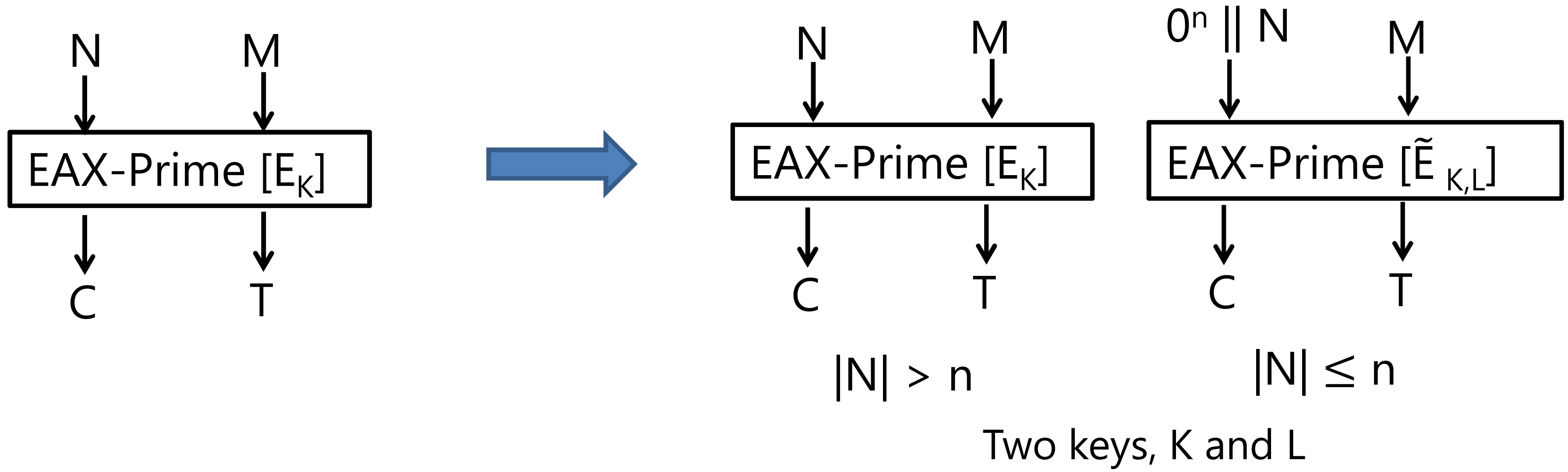
How to safely use |N| ≤ n ?

- Method 2. Use two blockcipher keys, K and K'
  - $E_K(X)$ for |N| > n, otherwise $E_{K'}(X)$ w/ prepending to N
    - Independent keys (safer, but expensive)
    - K' generated from K ⊕ const (e.g., const = $1^{|K|}$)
      - the choice of constant needs cares
      - very limited form of RK-security is required



How to safely use $|N| \le n$?

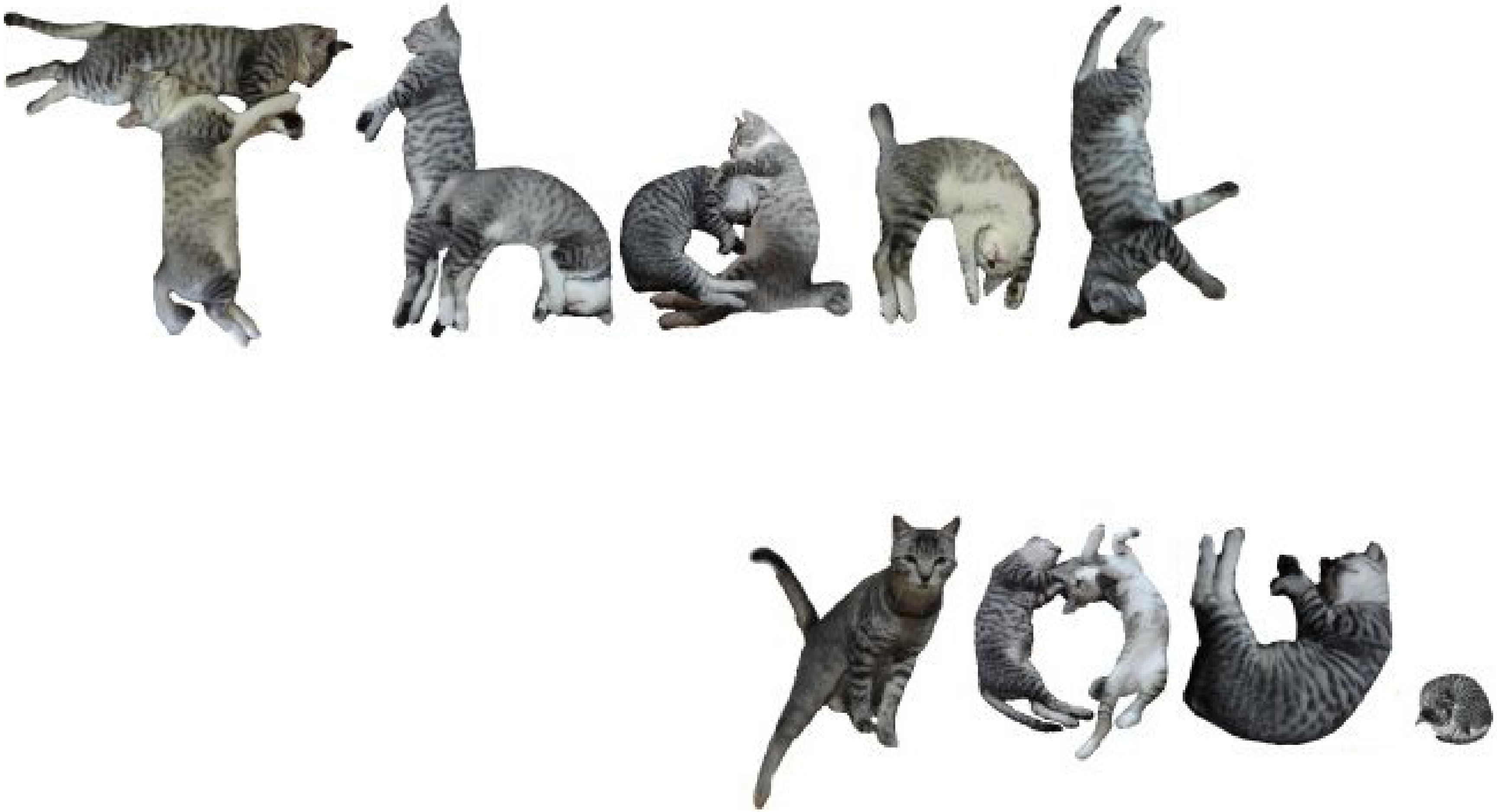
- Method 3. Use tweakable blockcipher with additional independent n-bit key, L
  - $E_K(X)$ for $|N| > n$, otherwise $\tilde{E}_{K,L}(X) = E_K(X \oplus L)$ w/ prepending to N



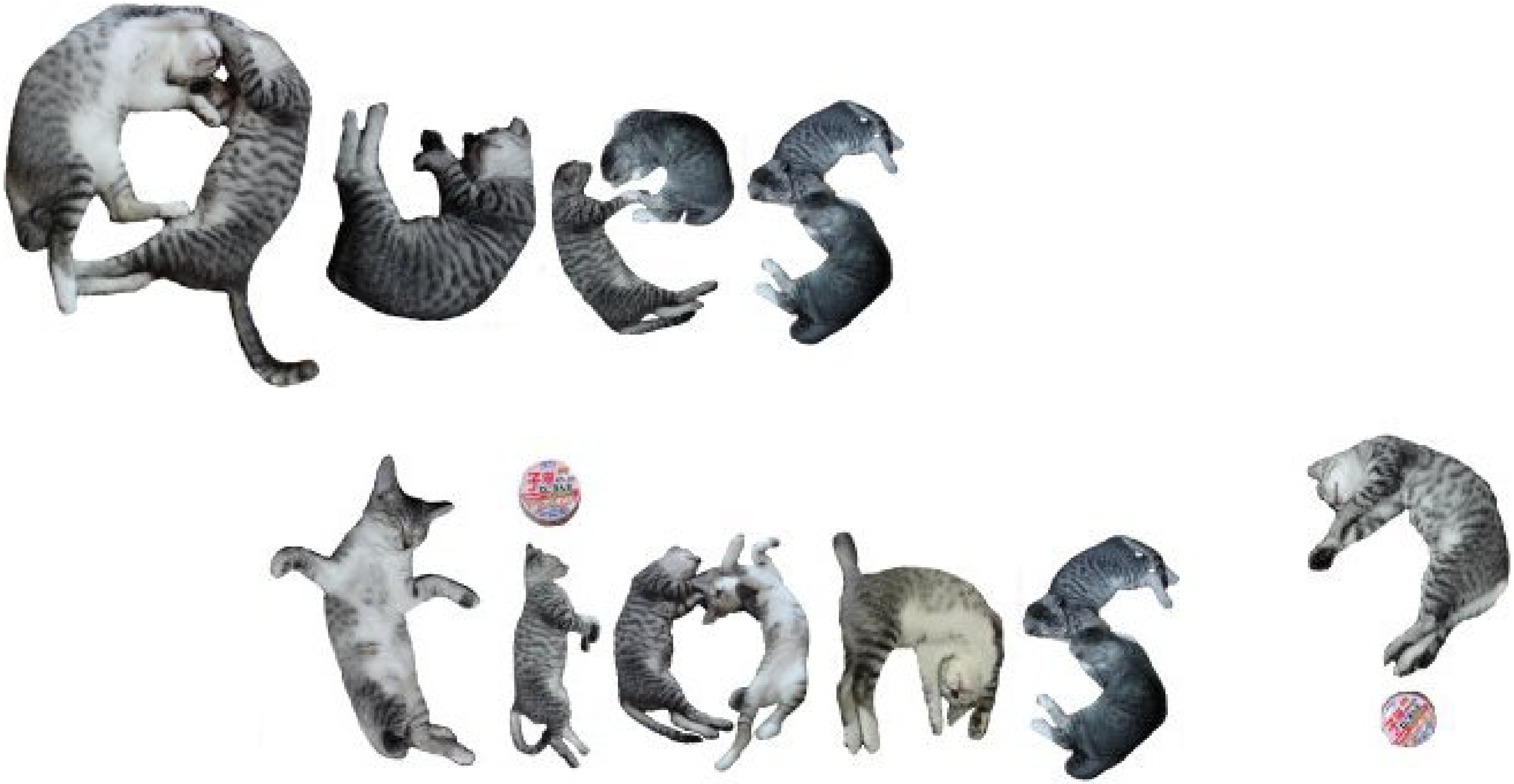
- Each method has good and bad points

Lessons learned

- A seemingly small change can result in fatal consequences
  - A repeated problem in real-world crypto…
- CMAC is *one* PRF : generating multiple PRFs needs cares
  - EAX employs a simple and secure method
- The importance of security proofs
  - Our proof shows that cleartext length check is sufficient for secure (though cumbersome) use of EAX-Prime



via http://nekofont.upat.jp/



via http://nekofont.upat.jp/