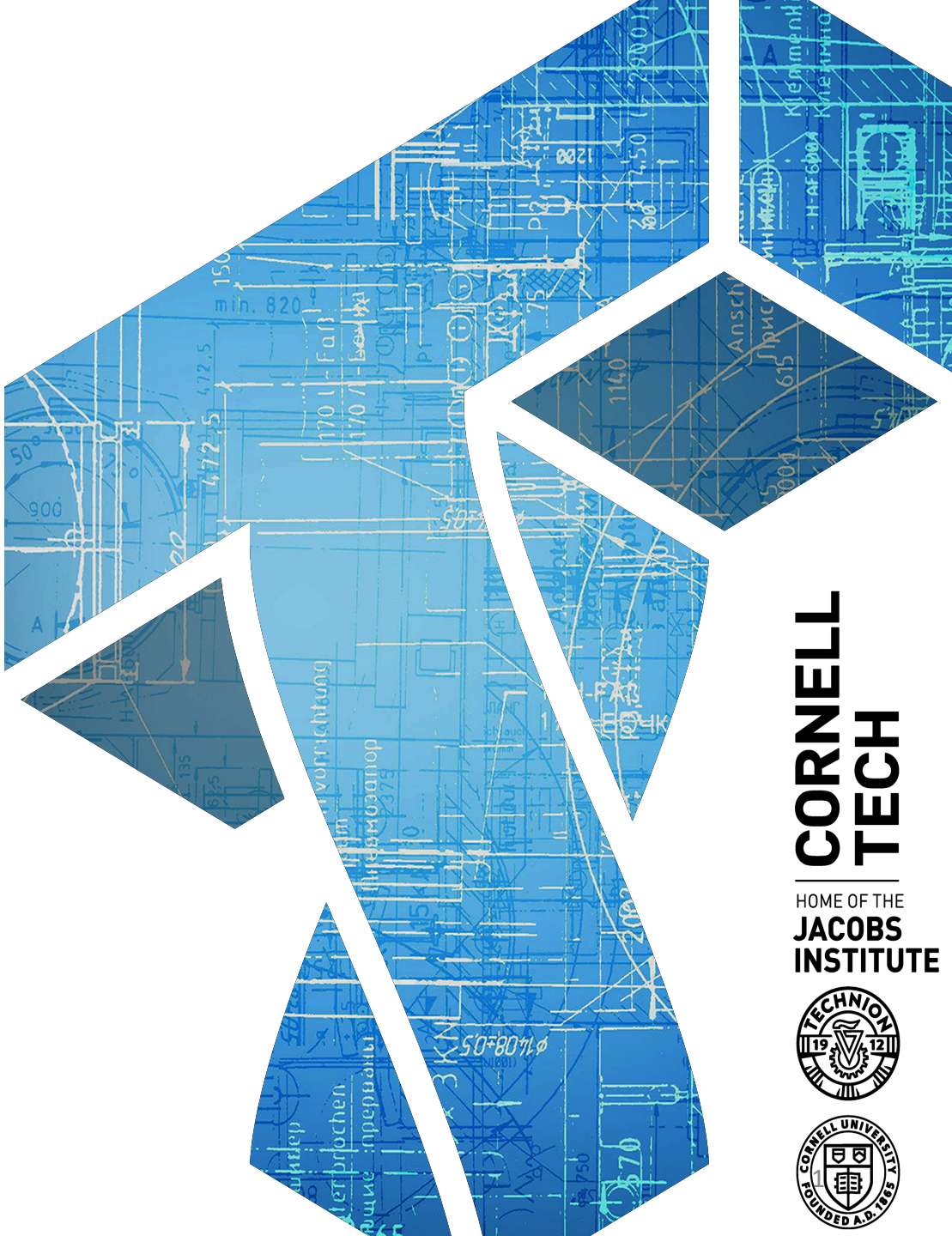
CS 5830
Cryptography

Instructor: Tom Ristenpart
TAs: Yan Ji, Sanketh Menda



CORNELL TECH
HOME OF THE JACOBS INSTITUTE

Recap and where we're at

- What we have so far:
  - Block cipher modes of operation (CBC, CTR)
    - Computationally secure against passive adversaries (IND-CPA)
  - Insecure against active attacks
    - Integrity problems: bit flip "mauling" attacks against CTR, CBC
    - Confidentiality problems: Padding oracle attacks against CBC

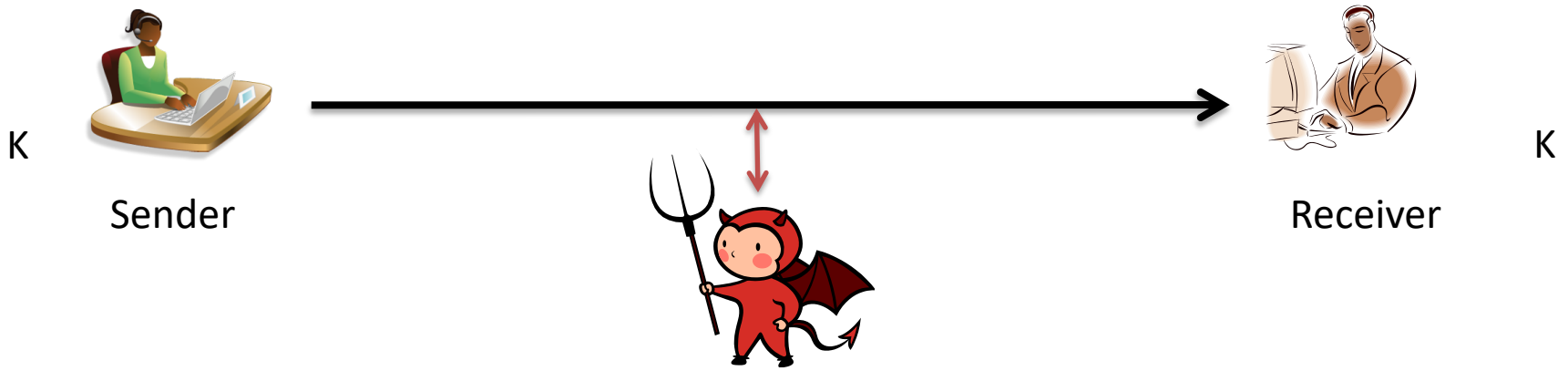
- We need another tool:

  authenticity mechanisms

Roadmap towards authenticated encryption

- Message authentication schemes
  - Do not concern themselves with confidentiality, just authenticity
- Apply message authentication to IND-CPA ciphertexts
  - Yields ***authenticated encryption***
  - Message confidentiality (under both CPAs & CCAs)
  - Message integrity

Message authentication

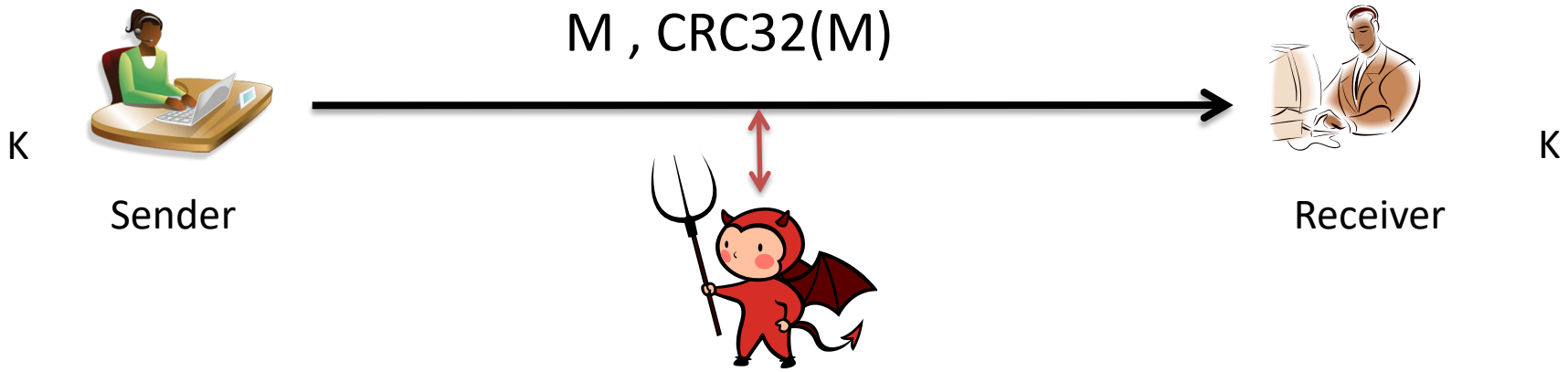


Attacker has read/write access to communications channel

<u>The strategy:</u>

Arrange so that that all bits received can be validated as having come ***only*** from sender (the entity with key K)

Message authentication

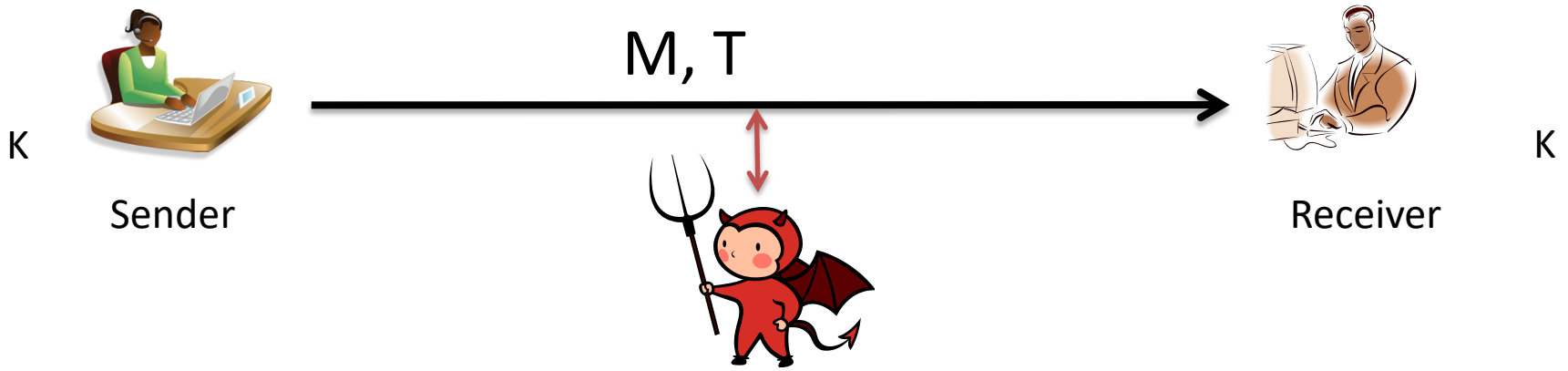


Recall error correction codes (ECCs):

- Don't use a cryptographic key
- Used to help detect & fix **random errors**
- CRC is just one example:
  - Reed-Solomon codes, credit card last digit (Luhn check), etc.

Don't use secret key. Adversary can easily forge an ECC!

Message authentication schemes



Scheme is triple of algorithms MA = (Kg,Tag,Verify):

(1) Kg outputs secret key K

(2) Tag(K,M) outputs a tag T

(3) Verify(K,M,T) outputs 0/1 (invalid / valid)

Message authentication code (MAC):
Tag deterministic +
Verify checks by recomputing Tag

Correctness: Verify(K,M,Tag(K,M)) = 1 always

Security: No computationally efficient attacker can forge tags for a new message even when attacker gets

$$(M_1, T_1), (M_2, T_2), \ldots, (M_q, T_q)$$

for messages of their choosing and reasonably large q.

Unforgeability under chosen-message attack (UF-CMA)

Security: No computationally efficient attacker can forge tags for a new message, even given chosen message attack

UF-CMA(MA, $\mathcal{A}$):
K <-$ Kg
(M*,T*) <-$ $\mathcal{A}^{\text{Tag}}$
If M* ∈ MsgSet then
  Return 0
Return Verify(K,M*,T*)

Tag(M)
MsgSet <- MsgSet U {M}
Return Tag(K,M)

Measure advantage of adversary via probability wins game:

Pr[UF-CMA(MA, $\mathcal{A}$) => 1]

Insecure:
Close to 1 for $\mathcal{A}$ using reasonable time and Tag queries

Secure:
Negligibly far from 0 for any $\mathcal{A}$ using reasonable time and queries

Unforgeability under chosen-message attack (UF-CMA)

Security: No computationally efficient attacker can forge tags for a new message, even given chosen message attack

UF-CMA(MA, $\mathcal{A}$):
K <-$ Kg
(M*,T*) <-$ $\mathcal{A}^{\text{Tag}}$
If M* ∈ MsgSet then
  Return 0
Return Verify(K,M*,T*)

Tag(M)
MsgSet <- MsgSet U {M}
Return Tag(K,M)

Warm-up: tag length is important

Assume M is n-bit message. Let E be a blockcipher on n bits

Trunc_t truncates input string to first t bits

Let's say we set $\text{Tag}(K,M) = \text{Trunc}_t(E_K(M))$

Easy attack strategy?

Unforgeability under chosen-message attack (UF-CMA)

Security: No computationally efficient attacker can forge tags for a new message, even given chosen message attack

UF-CMA(MA, $\mathcal{A}$):
K <-$ Kg
(M*,T*) <-$ $\mathcal{A}^{\text{Tag}}$
If M* ∈ MsgSet then
  Return 0
Return Verify(K,M*,T*)

Tag(M)
MsgSet <- MsgSet U {M}
Return Tag(K,M)

Warm-up: tag length is important

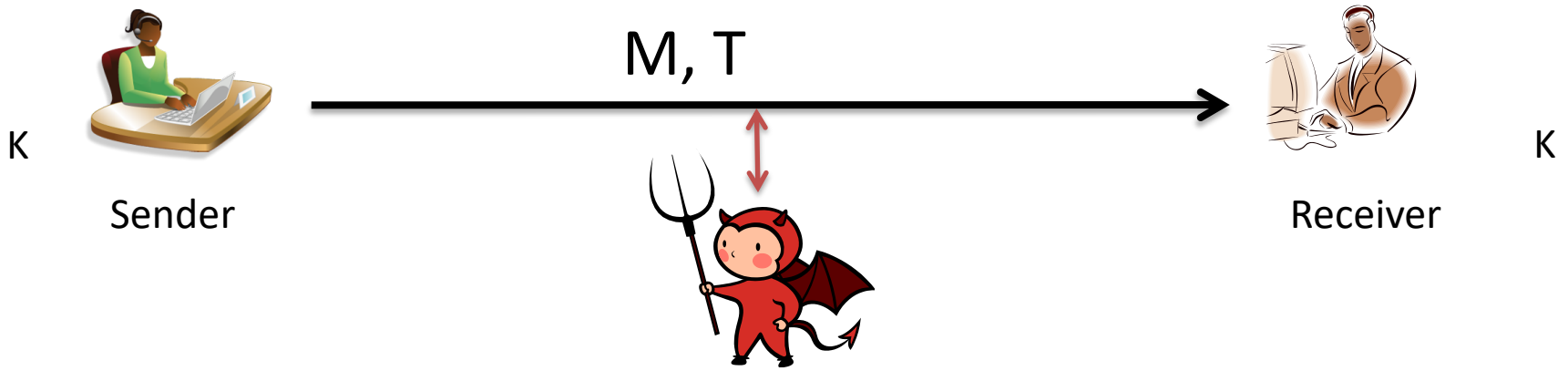
Assume M is n-bit message. Let E be a blockcipher on n bits

Trunc_t truncates input string to first t bits

Let's say we set $\text{Tag}(K,M) = \text{Trunc}_t(E_K(M))$

Any ideas for attack strategy?

Message authentication using pseudorandom functions (PRFs)



Let $F : \{0,1\}^k \times \{0,1\}^* \rightarrow \{0,1\}^t$ be a secure PRF

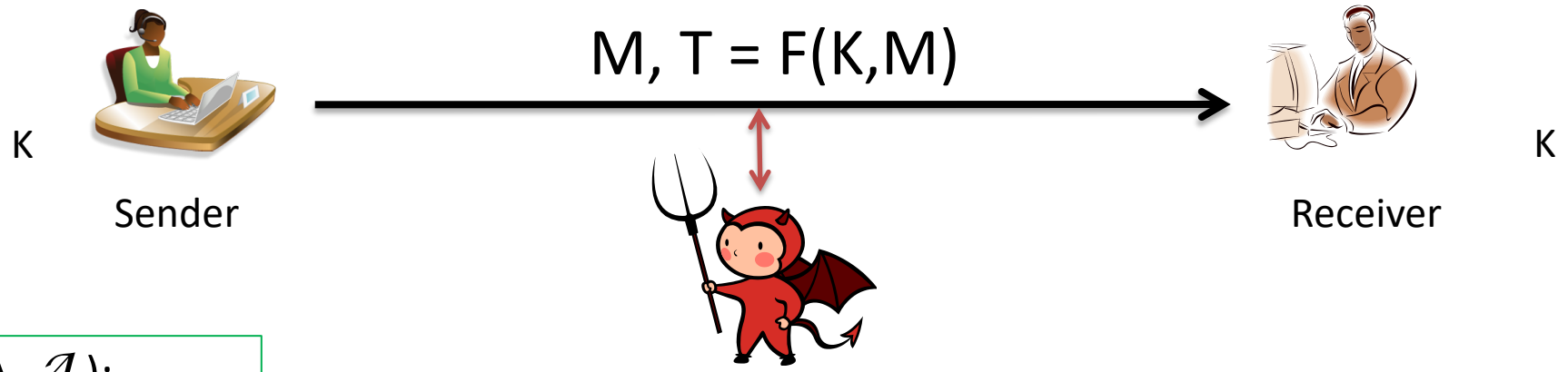
Tag(K,M)
Return F(K,M)

Verify(K,M,T):
If F(K,M) = T then Return 1
Return 0

When is this secure?

Can you sketch an argument for why?

Message authentication using pseudorandom functions (PRFs)



UF-CMA(MA, $\mathcal{A}$):
K <-$ Kg
(M*,T*) <-$ $\mathcal{A}^{\text{Tag}}$
If M* ∈ MsgSet then
  Return 0
Return Verify(K,M*,T*)

Tag(M)
MsgSet <- MsgSet U {M}
Return Tag(K,M)

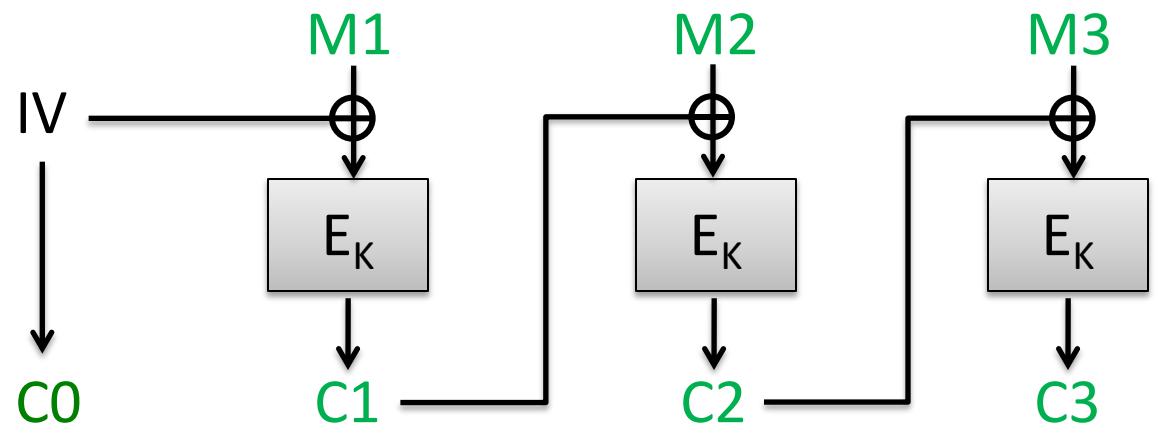
PRFs for variable-length messages

- Blockciphers are good PRFs, but only accept n-bit input
- We want to construct PRFs for longer and variable lengths

- How do we do this? We can again build a PRF from a blockcipher, using it in a mode of operation
  - Goal is different from building encryption
  - $F : \{0,1\}^k \times \{0,1\}^* \rightarrow \{0,1\}^t$

Recall CBC mode

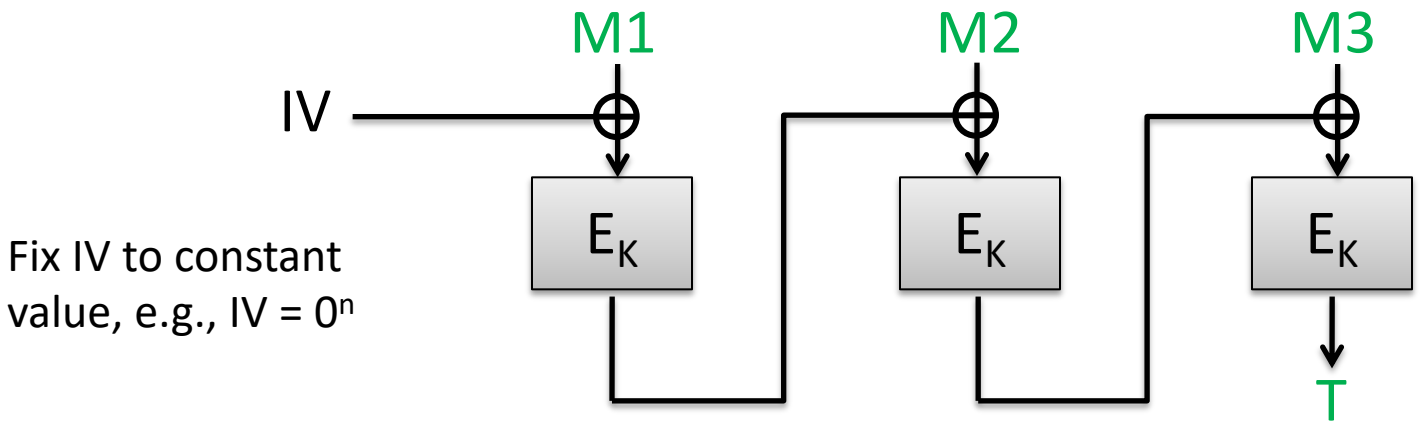
Ciphertext block chaining (CBC)
Pad message M to M1,M2,M3,... where each block Mi is n bits
Choose random n-bit string IV
Then:



Can we convert this into variable-message-length PRF?

CBC-MAC

Message authentication code (MAC)



Fix IV to constant value, e.g., IV = 0^n

Turns out this is (provably) a good PRF

if K used only on same-length messages

Breaking variable-input-length CBC-MAC

Consider CBC-MAC but allow any messages of length a multiple of n.

E.g.: $\text{Tag}(K,M_1) = E_K(M_1)$ $\text{Tag}(K,M_1 \,||\, M_2) = E_K(M_2 \oplus E_K(M_1))$

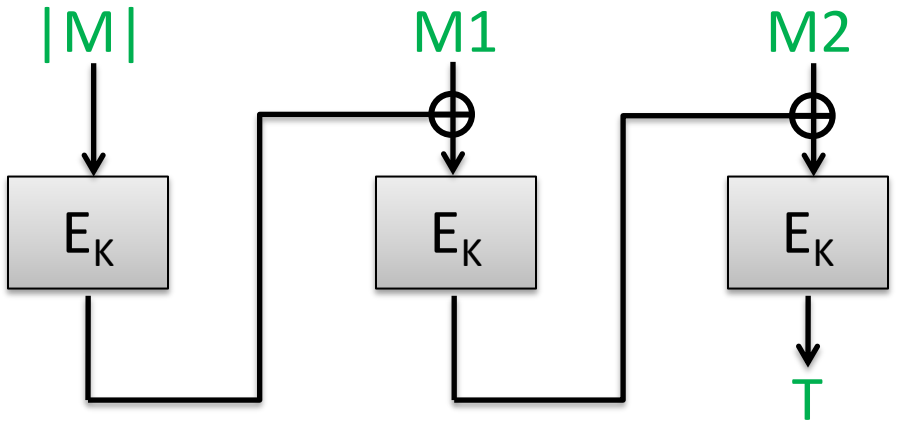
UF-CMA(MA, $\mathcal{A}$):
K <-$ Kg
(M*,T*) <-$ $\mathcal{A}^{\text{Tag}}$
If M* ∈ MsgSet then
  Return 0
Return Verify(K,M*,T*)

Tag(M)
MsgSet <- MsgSet U {M}
Return Tag(K,M)

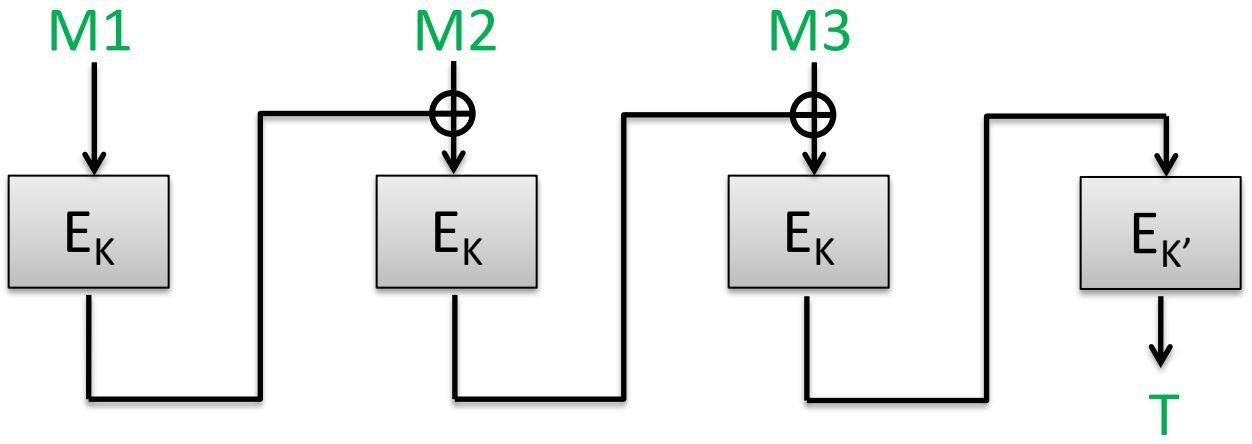
Give UF-CMA adversary $\mathcal{A}$ that forges successfully using few queries

Secure variable-message-length CBC-MAC

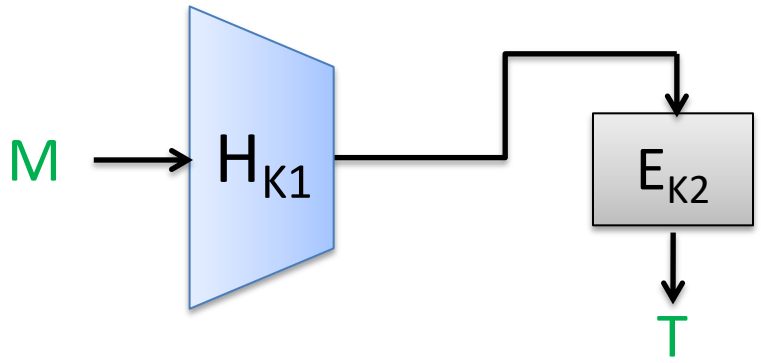
- Prepend message length



- Encrypted CBC-MAC



Universal hash then PRF approach



A keyed function $H : \{0,1\}^k \times \mathcal{M} \rightarrow \{0,1\}^n$ is an ϵ-almost universal (AU) hash function if for all $M \neq M'$

$$\Pr\left[\, H_K(M) = H_K(M') \,\right] \leq \epsilon$$

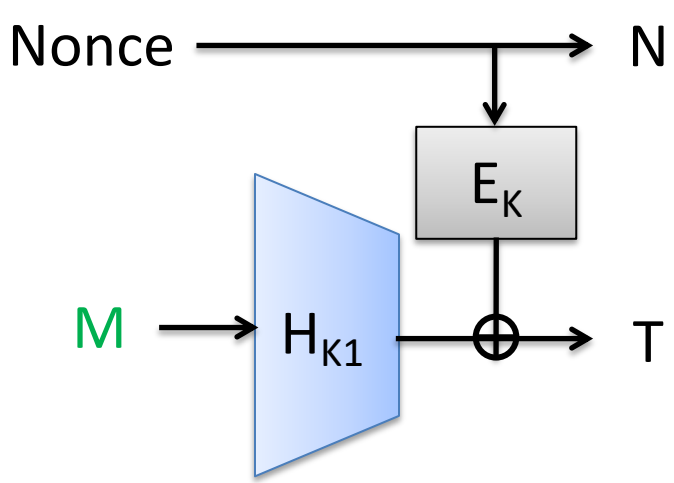
where the probability is over choice of K.

Turns out this is a good construction for any H that is a (computational) ***universal hash function***

Intuition: E_{K2} hides info about K1 unless collision. For q queries, chance of collision at most

$q^2 \epsilon / 2$

Carter-Wegman MA Schemes



Nonce must be fresh for each M
Random value or counter
If repeated, security fails

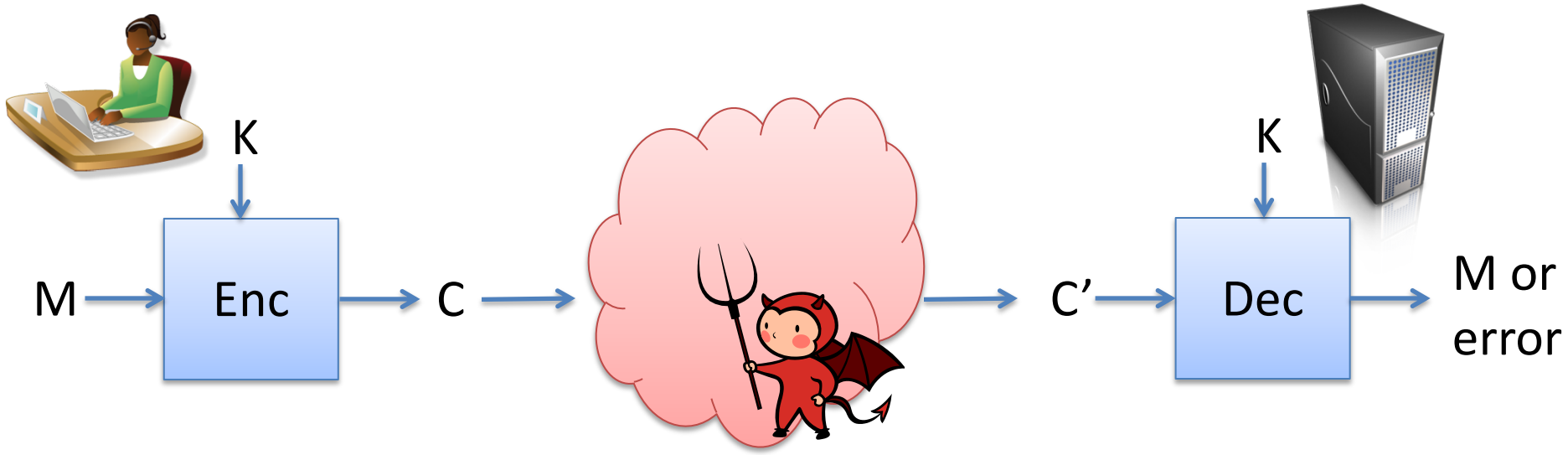
Intuition:
Hide universal hash by encrypting it

A keyed function $H : \{0,1\}^k \times \mathcal{M} \rightarrow \{0,1\}^n$ is an ϵ-almost universal (AU) hash function if for all $M \neq M'$

$$\Pr\left[\, H_K(M) = H_K(M') \,\right] \leq \epsilon$$

where the probability is over choice of K.

Authenticated encryption (AE)



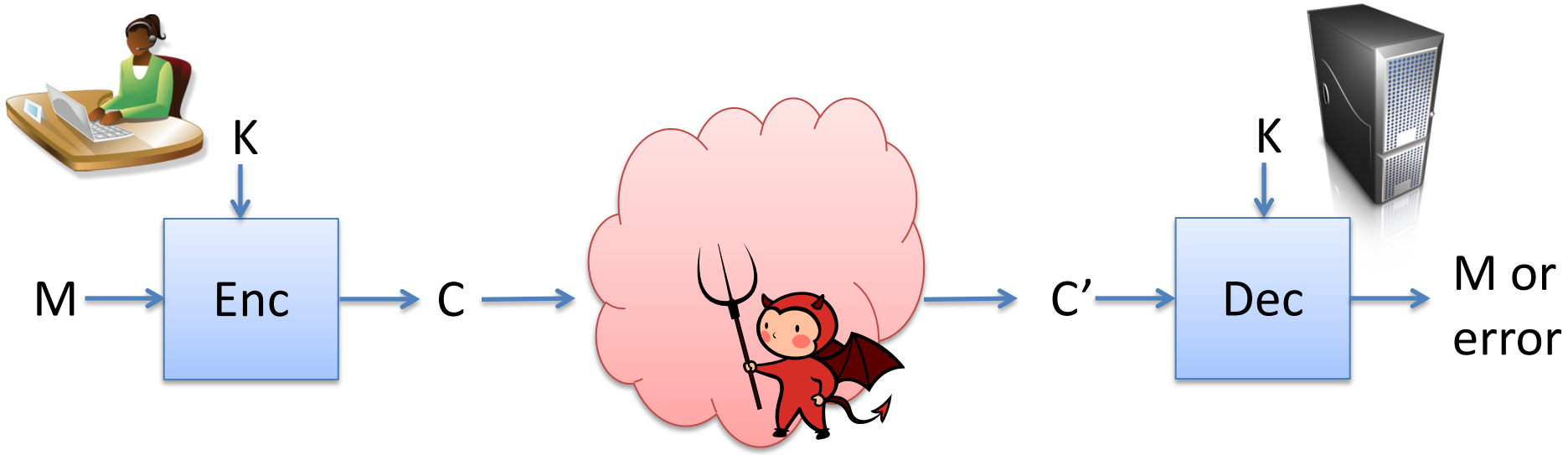
What security properties do we need from symmetric encryption?

1) Confidentiality: should not learn any information about M

2) Authenticity: should not be able to forge ciphertexts

Often referred to as Authenticated Encryption security

Authenticated encryption (AE)



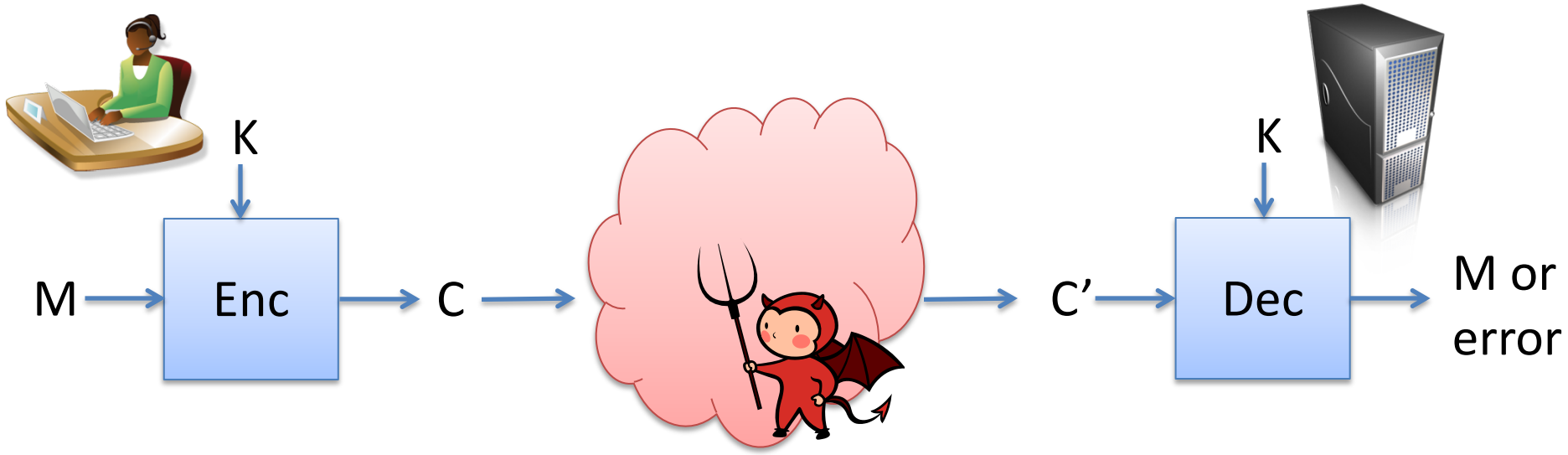
What security properties do we need from symmetric encryption?

1) Confidentiality: computational indistinguishability

2) Authenticity: ???

Often referred to as Authenticated Encryption security

Authenticated encryption (AE)



Ciphertext unforgeability: Let K be honestly generated secret key. No computationally efficient attacker can construct ciphertext C* that decrypts correctly under K, even when given

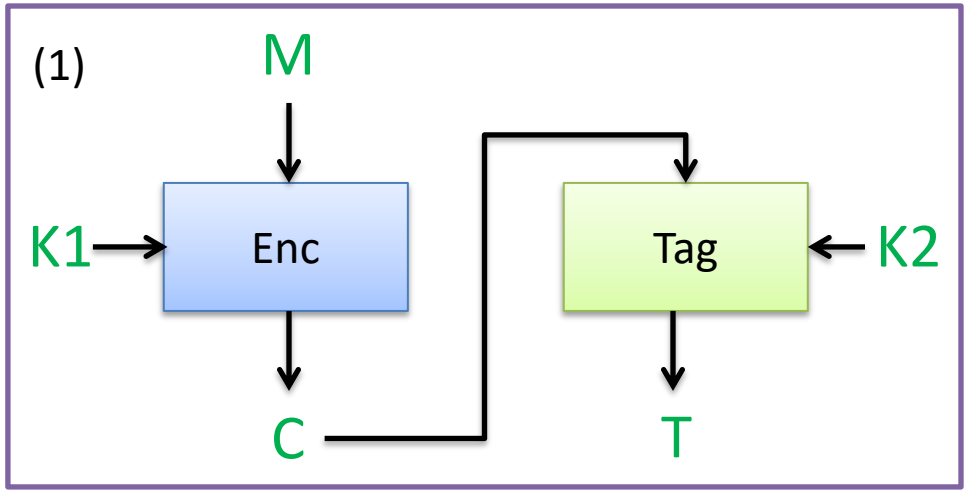
$$(M_1 , C_1) , (M_2 , C_2), \dots , (M_q , C_q)$$

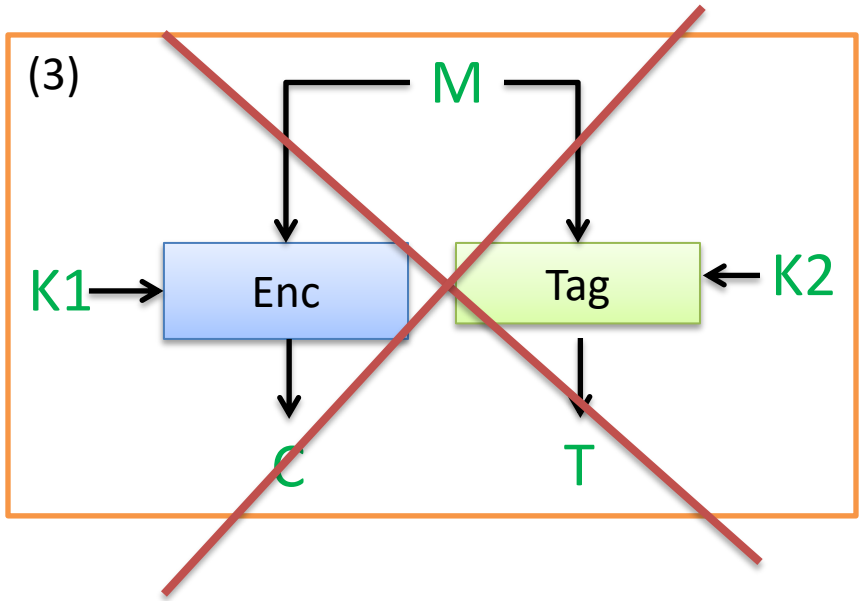
for messages of his choosing and ciphertexts generated under K. It must be that $C^* \neq C_i$ for $1 \leq i \leq q$

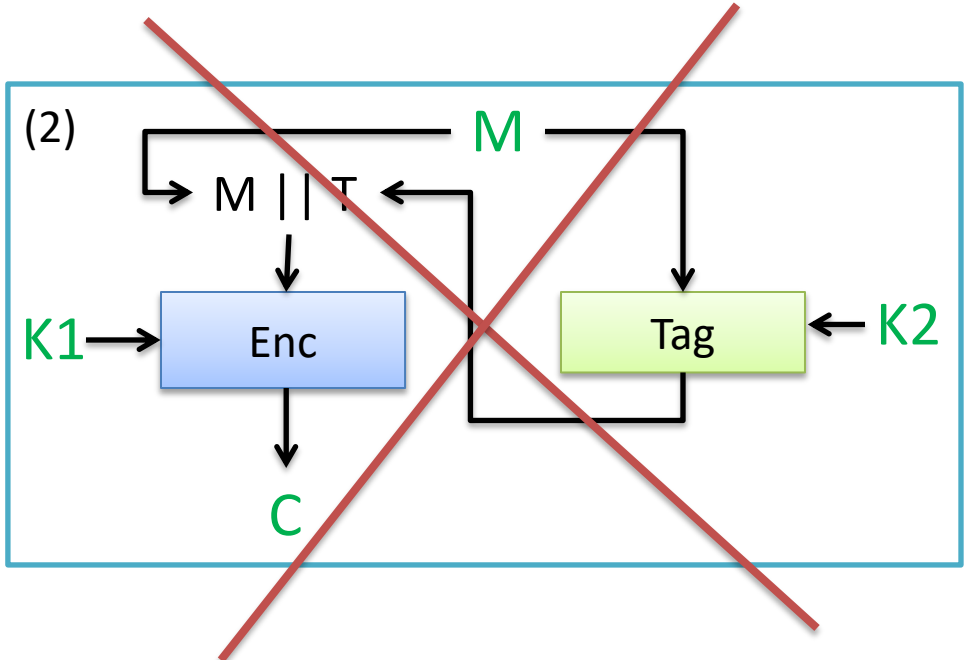
How do we do it?

Build a new scheme from Enc mode (CBC, CTR) and MAC
Kg outputs Enc key K1 and MAC key K2

Several ways to combine:
(1) encrypt-then-mac
(2) mac-then-encrypt
(3) encrypt-and-mac

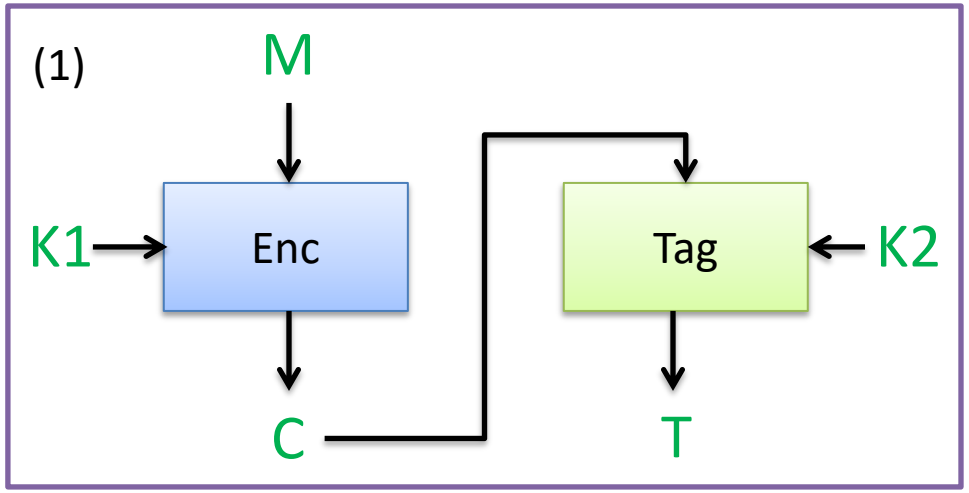






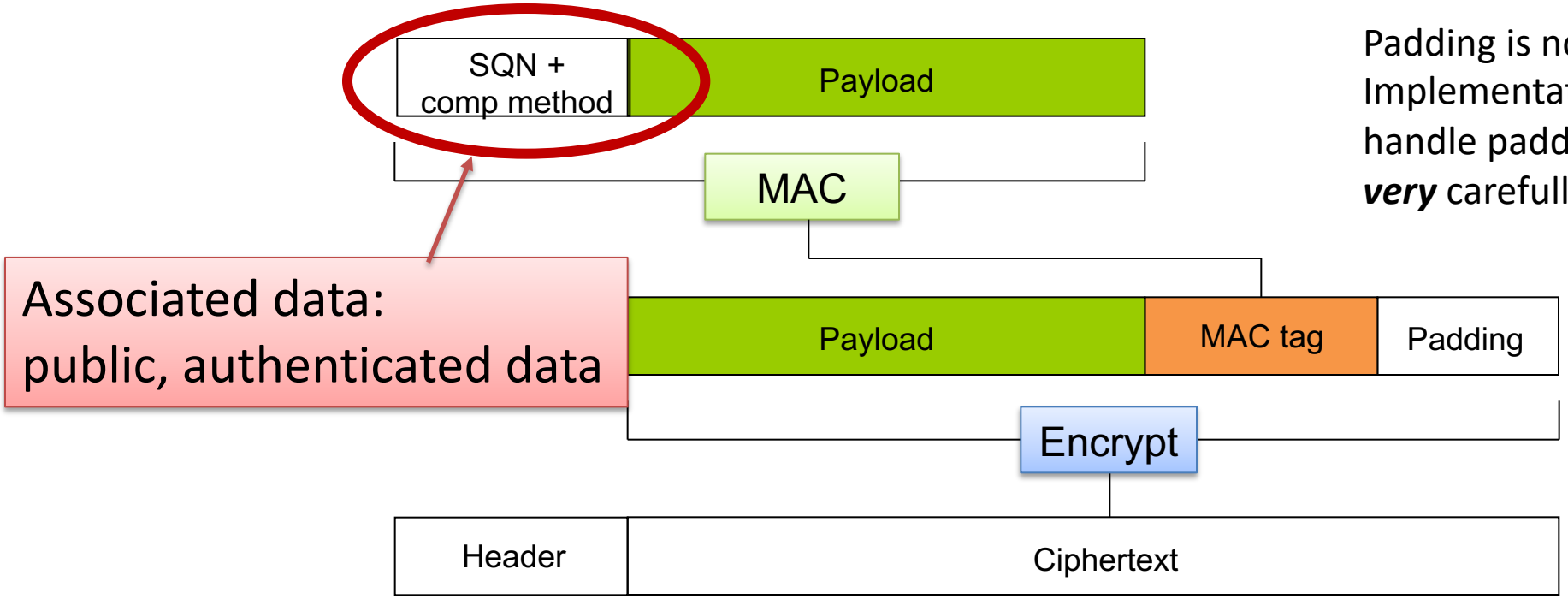
Build a new scheme from Enc mode (CBC, CTR) and MAC
Kg outputs Enc key K1 and MAC key K2

Several ways to combine:
(1) encrypt-then-mac
(2) mac-then-encrypt
(3) encrypt-and-mac



Thm. If encryption scheme provides confidentiality against passive attackers and MAC provides unforgeability, then Encrypt-then-MAC provides secure authenticated encryption

TLS 1.2 record protocol: MAC-Encode-Encrypt (MEE)

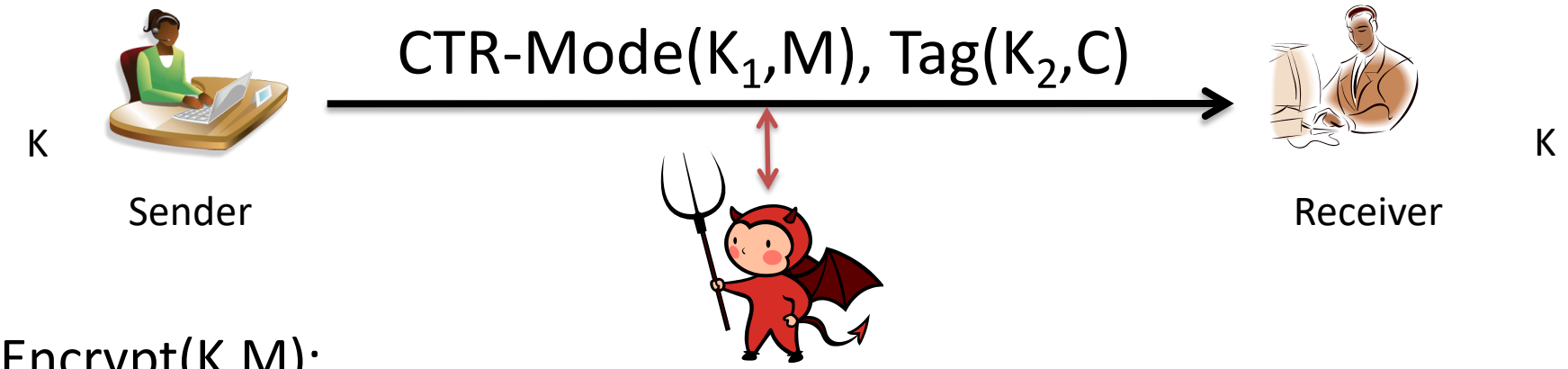


Padding is not MAC'd. Implementations must handle padding checks ***very*** carefully.

MAC	HMAC-MD5, HMAC-SHA1, HMAC-SHA256
Encrypt	CBC-AES128, CBC-AES256, CBC-3DES, RC4-128

Now TLS uses better authenticated-encryption schemes, this is deprecated

Composing encryption and authentication



Encrypt(K,M):

Use secret keys K_1 and K_2 . These can be derived from K if needed

K_1 <- AES(K,0^n) K_2 <- AES(K,1^n)

C <-$ CTR-Mode(K_1, M)

T <- Tag(K_2,C)

Output C||T

Decrypt(K,C||T)

If Verify(K_2,C,T) ≠ 1 then Return error

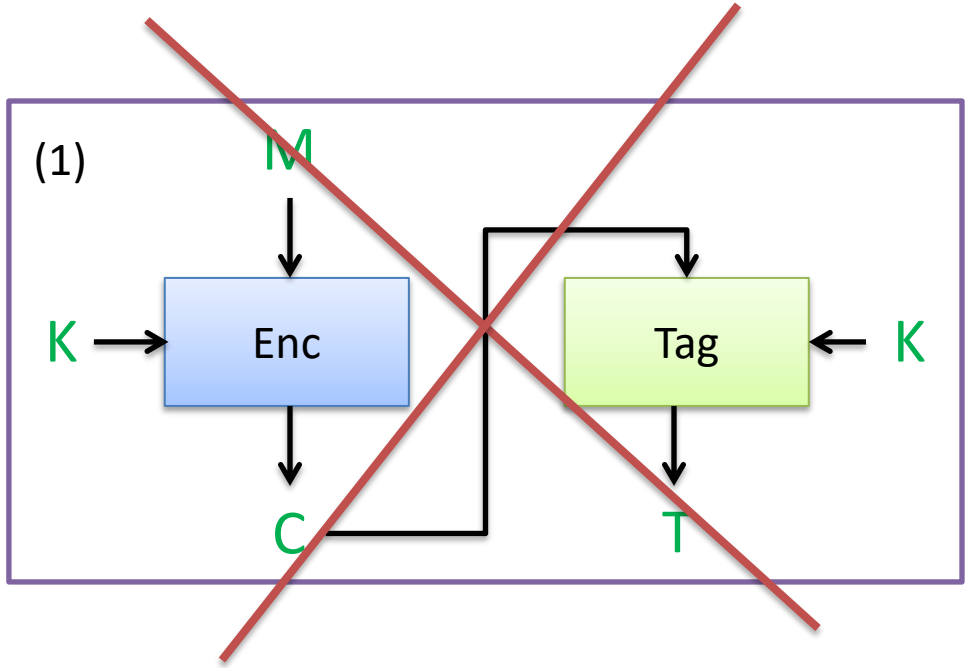
Return CTR-Mode-Decrypt(K_1,C)

Key derivation function (KDF):
Way to use one key to generate several

HKDF is widely used general-purpose KDF

Key separation is essential

If one uses same key for both encryption and MAC, attacks can arise



Consider CBC-MAC plus CBC-mode encryption

General rule: different crypto primitives or different applications of same primitive, need independent keys

Dedicated authenticated encryption schemes

Not a generic composition of Enc, MAC.
Directly construct from blockcipher, stream cipher, etc.

Attack	Inventors	Notes
OCB (Offset Codebook)	Rogaway	One-pass (one blockcipher call per block of message)
GCM (Galois Counter Mode)	McGrew, Viega	CTR mode plus specialized MAC
Salsa20/Poly1305	Bernstein	Stream cipher plus Carter-Wegman MAC
CWC	Kohno, Viega, Whiting	CTR mode plus Carter-Wegman MAC
CCM	Housley, Ferguson, Whiting	CTR mode plus CBC-MAC
EAX	Wagner, Bellare, Rogaway	CTR mode plus OMAC (variant of CBC-MAC)

Symmetric Encryption Advice

Never use CTR mode or CBC mode by themselves

Passive security is almost never good enough!!

Encrypt-then-MAC best way to do generic composition

Dedicated modes that have been analyzed thoroughly are way to go in practice