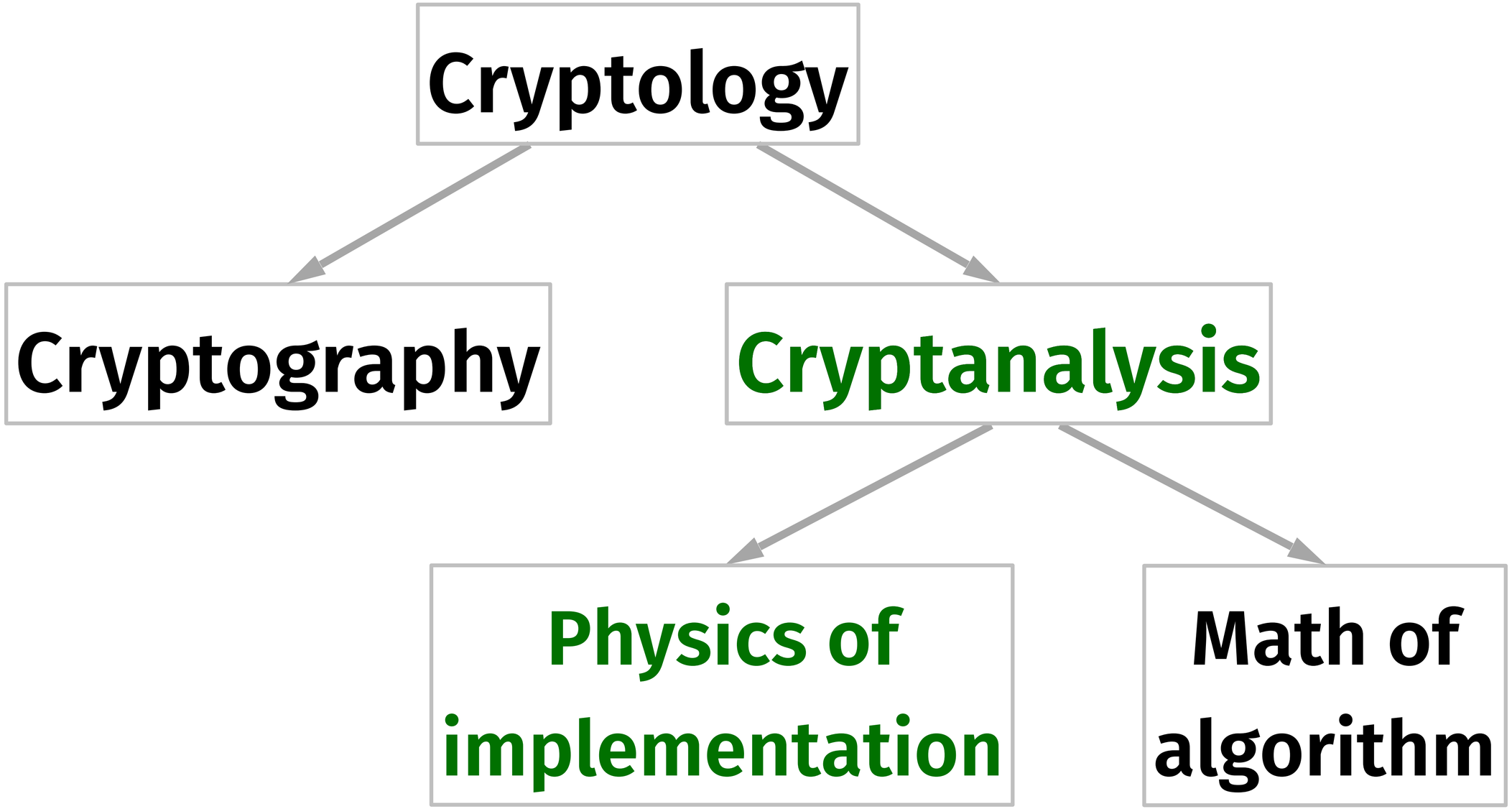
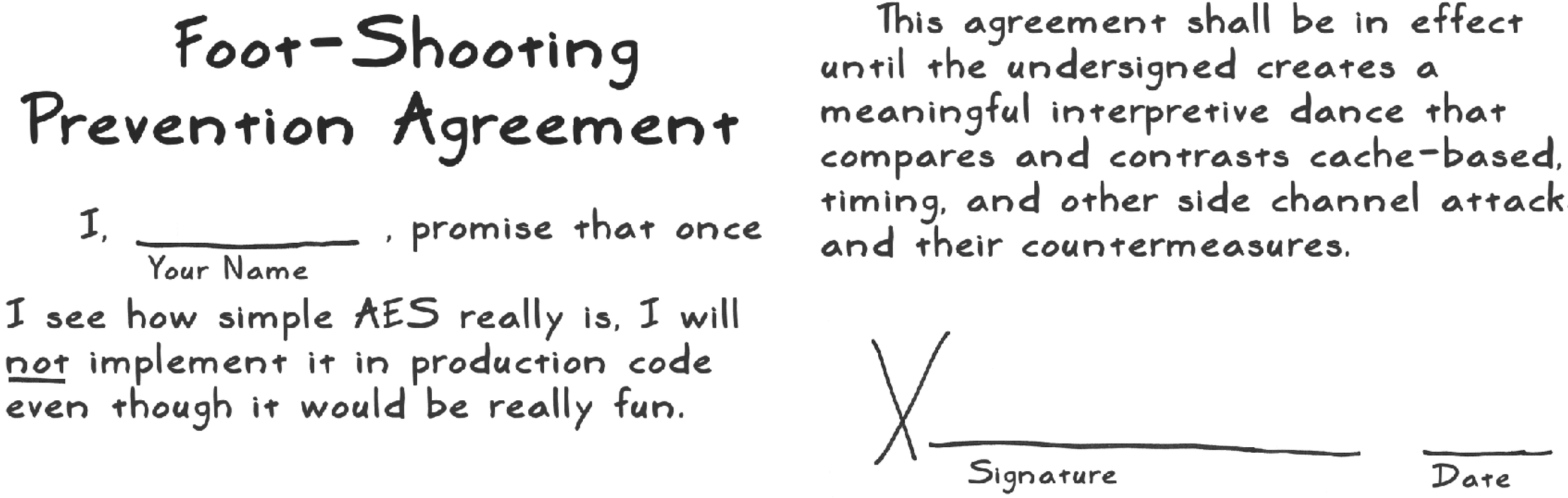
Lecture 12: Authenticated Encryption

- Lab 5 due Friday at 11pm
- No discussion sessions tomorrow
- Nicolas will hold office hours tomorrow at the usual time
- Have a good spring break!

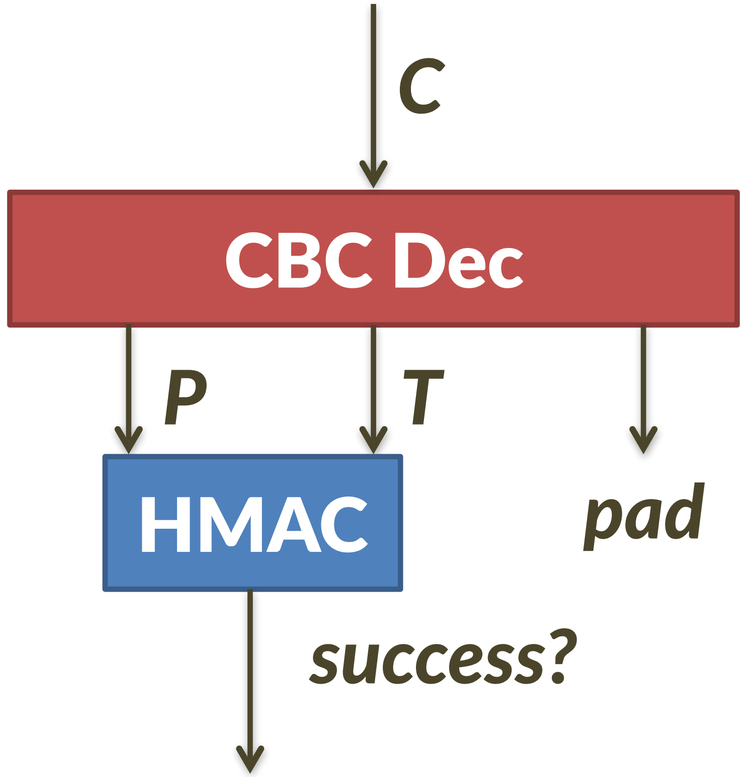

Cryptology
Cryptography
Cryptanalysis
Physics of implementation
Math of algorithm

Side channels ⇒ difficult to implement crypto securely



Source:
moserware.com/2009/09/stick-figure-guide-to-advanced.html

Last time: Padding oracle attack



Outcomes

1. Invalid padding
2. Valid padding, wrong HMAC
3. Valid padding, right HMAC

What to do in cases #1 and #2?

- Typical answer: return error message
- We can use error messages to find *P*!

How can we fix this?

- Remember the three cases

	Required effort
1. Invalid padding	⇒ Read the padding bytes
2. Valid padding, wrong HMAC	⇒ Read padding bytes, compute the HMAC
3. Valid padding, right HMAC	⇒ Read padding bytes, compute the HMAC

- Bob's solution: return the *same* error message in cases #1 and #2
  - Mallory's countermeasure: can still distinguish the two cases by observing the *time* that the MAC-then-Encrypt system takes to execute!
- Bob's new solution: ensure crypto software's runtime is *independent* of input (i.e., perform the HMAC test whether the padding is correct or not)
  - This won't work; Mallory can exploit timing variations within HMAC itself ☹️

Software is hard!

- Timing independence is hard

OpenSSL Fact @OpenSSLFact Jul 24, 2013
/*The aim of right-shifting md_size is so that the compiler doesn't figure out that it can remove div_spoiler...which I hope is beyond it.*/

- So is software in general

OpenSSL Fact @OpenSSLFact Jan 22, 2013
/* EEK! Experimental code starts */

OpenSSL Fact @OpenSSLFact Sep 3, 2012
/* [we should] obviate the ugly and illegal kludge in CRYPTO_mem_leaks_cb. Otherwise the code police will come and get us.*/

OpenSSL Fact @OpenSSLFact Sep 5, 2012
/* BIG UGLY WARNING! This is so damn ugly I wanna puke ... ARGH! ARGH! ARGH! Let's get rid of this macro package. Please? */

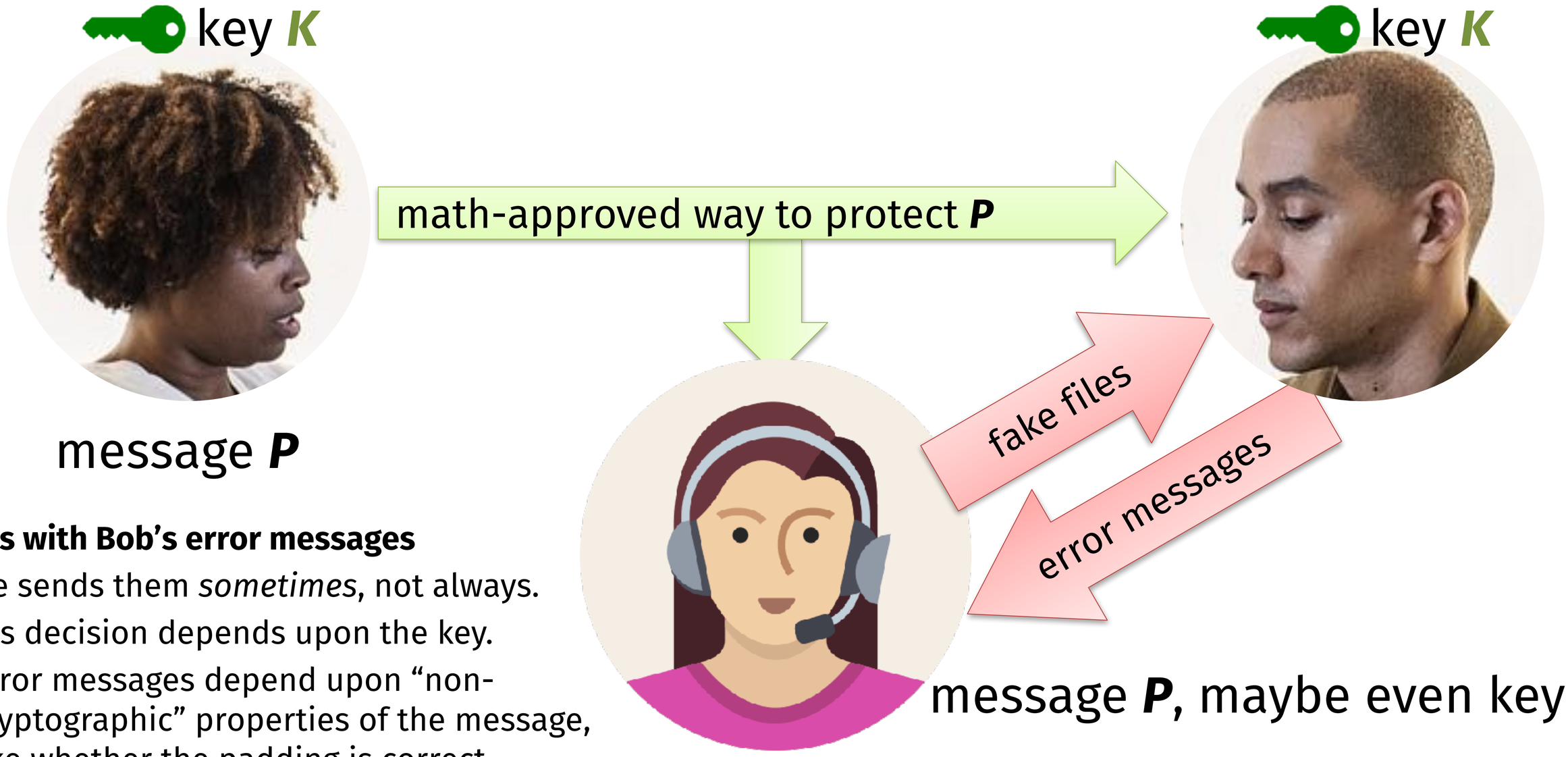
- So are compilers in general

Mudge @dotMudge · Jan 25
Modern compilers make a lot of optimizations and perform advanced heuristics to determine what to emit. The resulting binaries have many (attack-able) components you cannot learn from the source alone.

Source is the intent, the binary is reality.

Steven Bellovin @SteveBellovin · Jan 25
My favorite is how hard it is to zero out a cryptographic key that you're done with--the optimizer says "this variable is never used again", so it deletes the zeroize operation.

Part 2: Breaking crypto via side channels



Issues with Bob's error messages

1. He sends them *sometimes*, not always.
2. His decision depends upon the key.
3. Error messages depend upon "non-cryptographic" properties of the message, like whether the padding is correct.

Our desired countermeasure

key ***K***

message ***P***

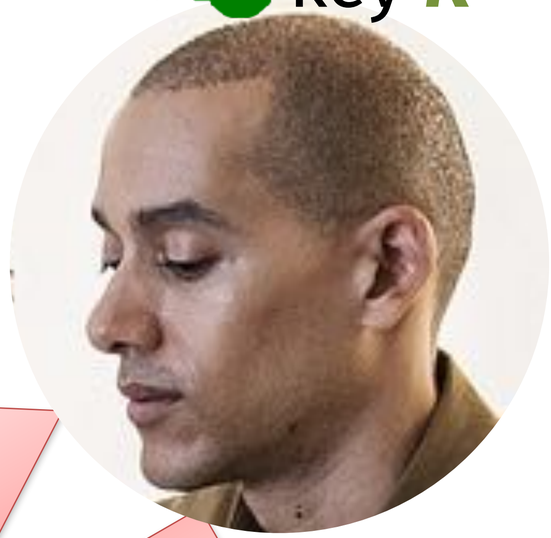
math-approved way to protect ***P***

fake files

I'm not talking to you, Mallory

2. Ideally with minimal use of the key

key ***K***



*1. Bob **always** rejects Mallory's messages*

3. All of Bob's checks are cryptographic

“Confidentiality xor authenticity is not possible. If you don't have both, often you don't have either.”

– *Prof. Matthew Green, Johns Hopkins*

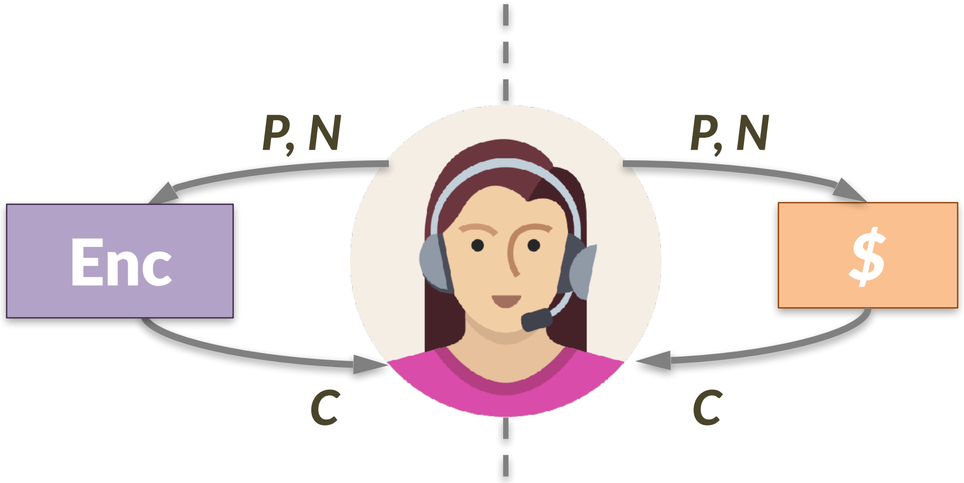
“If you have to perform any cryptographic operation before verifying the MAC on a message you’ve received, it will somehow inevitably lead to doom!”

– *Moxie Marlinspike*

Encryption xor Authentication

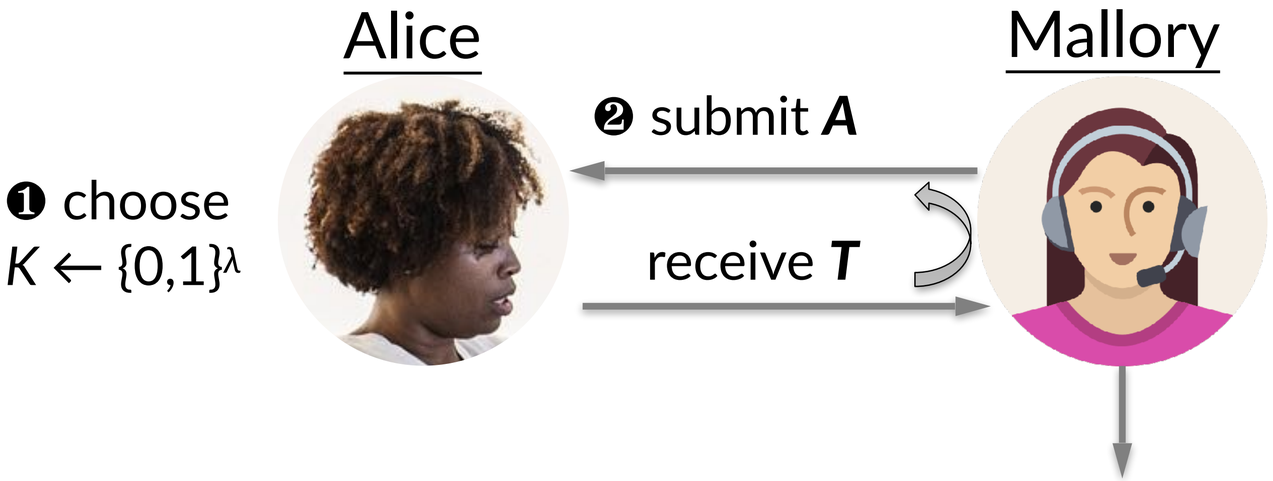
Privacy

IND$-CPA against
nonce-respecting Eve



Authenticity

Even after viewing many (***A***, ***T***) pairs,
Mallory cannot forge a new one

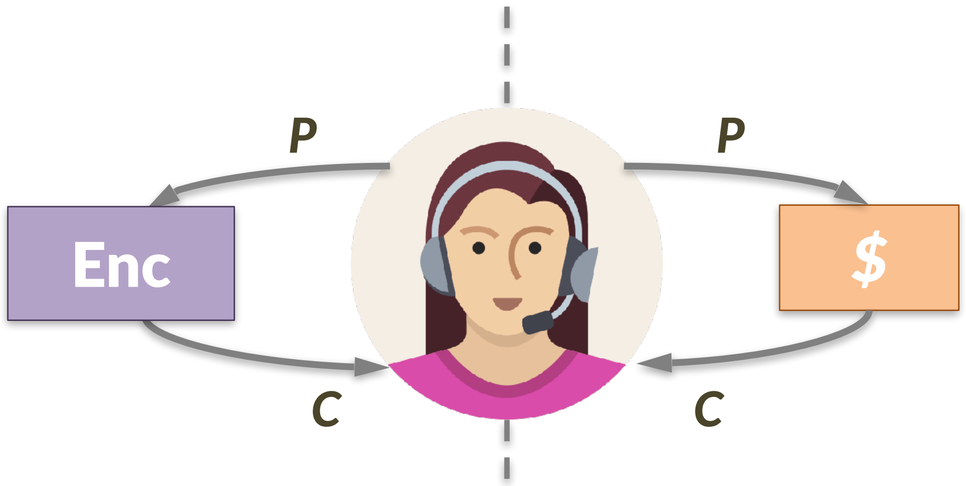


Mallory wins if:

1. It's a valid forgery
2. It's new

First, let's strengthen privacy

IND$-CPA



IND$-CCA

Same thing, but now Mallory has access to encryption *and* decryption oracles

What is the connection to padding oracles?

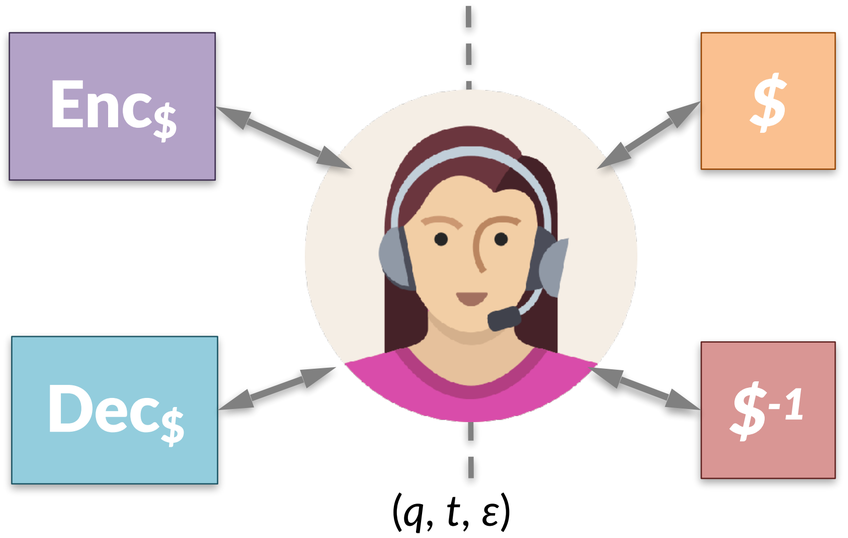
Formalizing IND$-CCA

Comprises 3 algorithms:

- KeyGen(λ) outputs a key $K \leftarrow \{0,1\}^\lambda$
- Encrypt_K(message P, nonce N) → C
- Decrypt_K(ciphertext C, nonce N) → P

Satisfies 3 constraints

- Performance: all 3 algorithms are efficiently computable
- Correctness: $\text{Dec}_K^{-1}(\text{Enc}_K(P, N)) = P$ for all $K \in \{0,1\}^\lambda$, $N \in \{0,1\}^\mu$, and $P \in \{0,1\}^*$



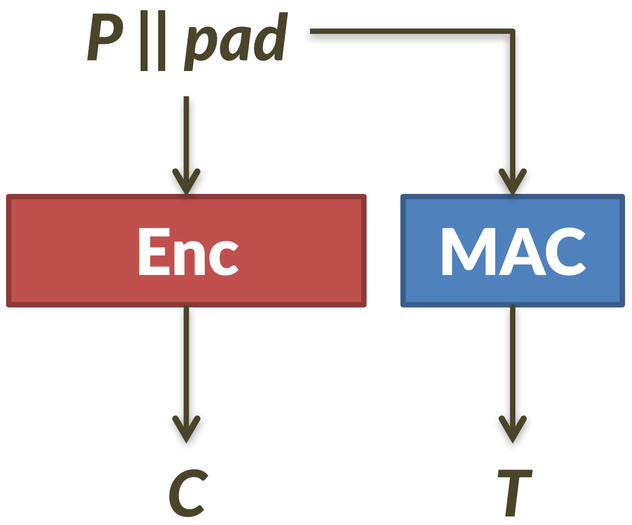
- **(q, t, ε)-IND$-CCA**: for every *nonce-respecting* adversary ***A*** who makes ≤ q queries and runs in time ≤ t,

$$A^{\text{Enc}_K, \text{Dec}_K} \approx_{q,t,\epsilon} A^{\$,\$^{-1}}$$

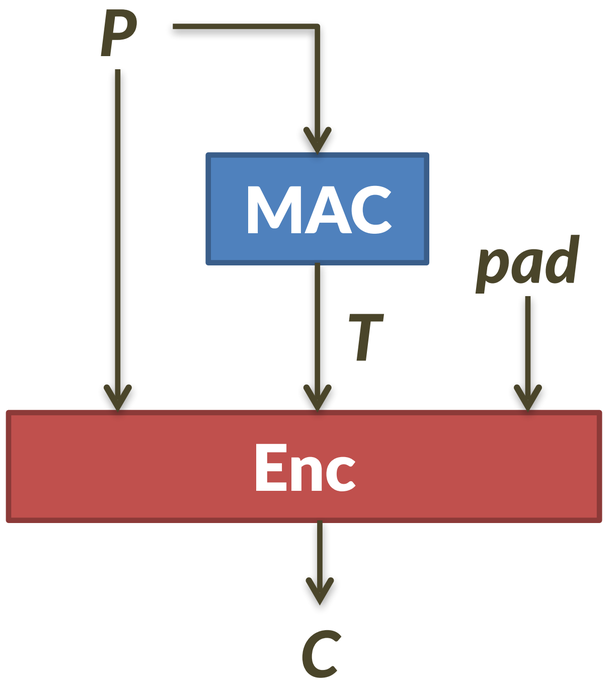
where $ responds randomly and so does $^-1 subject to consistency with $

Combining Enc and MAC *generically*

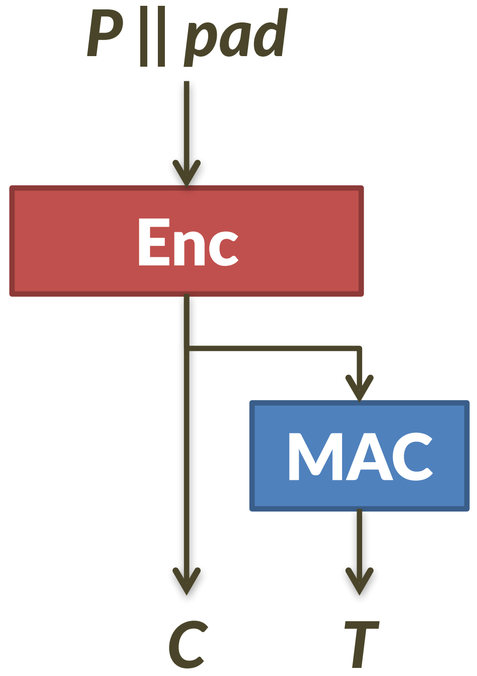
Enc and MAC



MAC then Enc



Enc then MAC

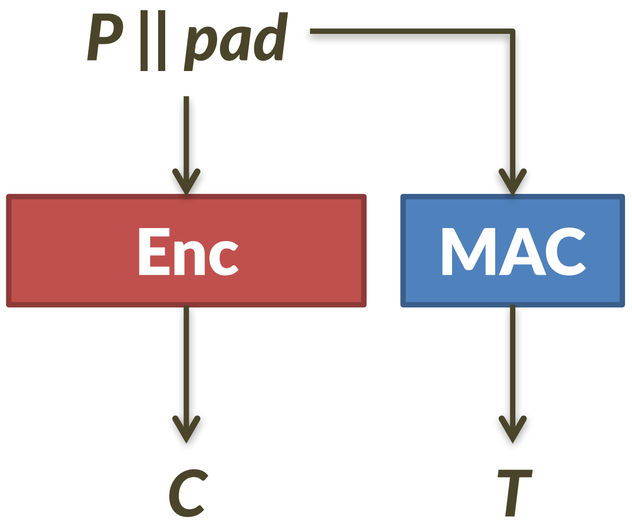


Intuitive concerns with MAC then Enc

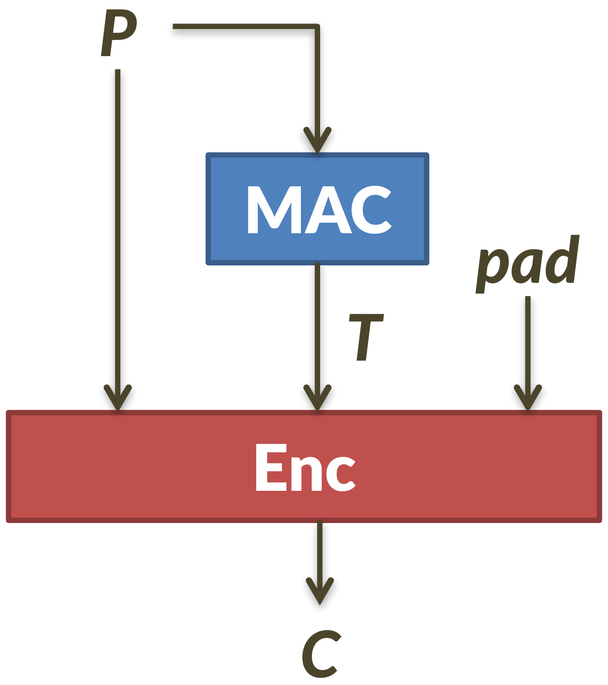
- The private data *P* is authenticated, but *C* is not!
- Recipient must perform decryption before knowing whether the message is authentic

Combining Enc and MAC *generically*

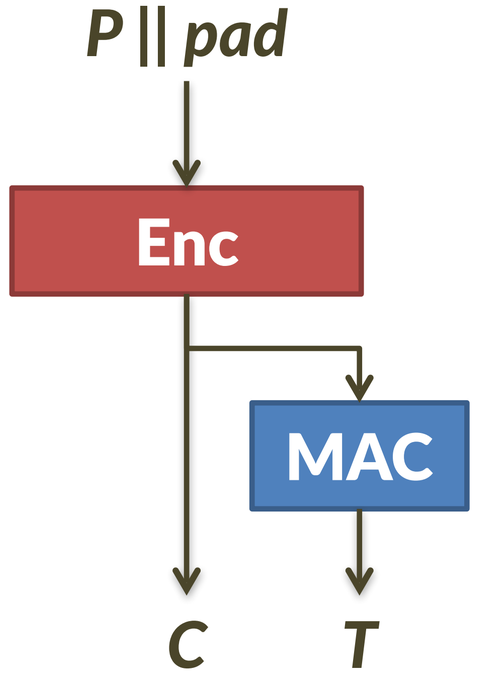
Enc and MAC



MAC then Enc



Enc then MAC



Confidentiality	None	CPA	CCA!
Integrity	*Plaintext integrity*: Cannot make CT that decrypts to message that sender never encrypted		*Ciphertext integrity*: Cannot make new valid CTs, only know sender-made ones

Formalizing ciphertext integrity

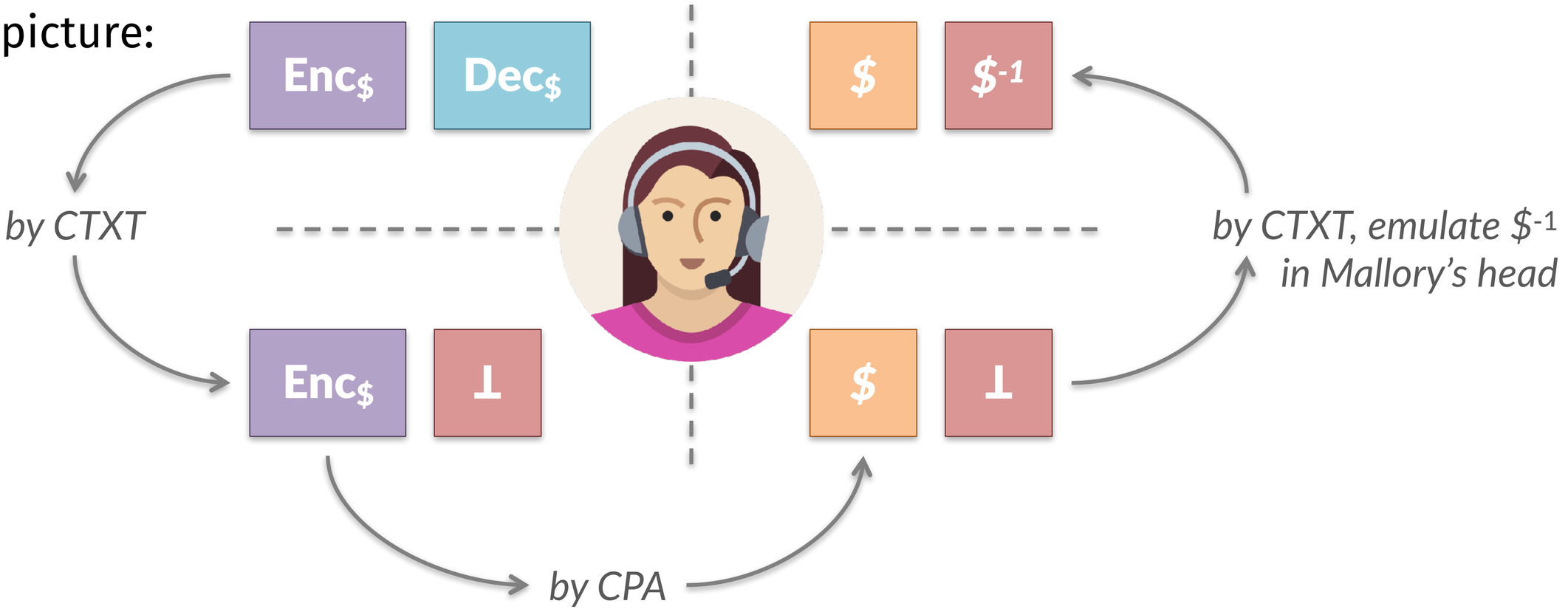
- Goal: Mallory cannot make a valid CT that wasn't previously made by sender
- Imagine that Mallory is trying to perform a padding oracle attack
- If she spams Bob with malformed CTs, now he simply rejects them all!

Operation: This box returns a single "integrity failure" error message no matter what Mallory submits!

Restriction: Mallory cannot attempt to decrypt ciphertexts that are the result of prior encryptions.

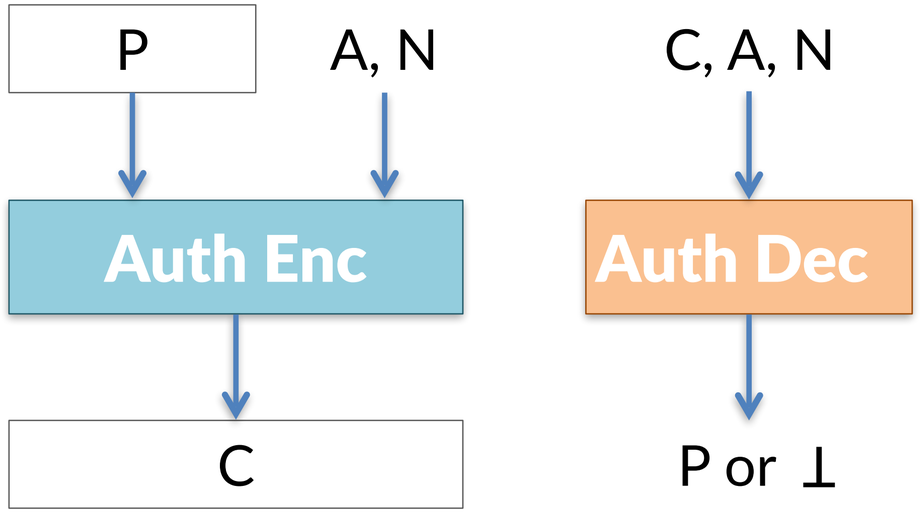
Relating integrity and confidentiality

- **Thm.** Suppose that an encryption scheme provides (q, t, ε_1)-CPA privacy and (q, t, ε_2)-ciphertext integrity. Then, it also provides $(q, t, \varepsilon_1+2\varepsilon_2)$-CCA privacy.
- Intuition: If Mallory can't forge new messages, then Dec oracle useless to her
- Proof by picture:



Def. Authenticated Encryption with Associated Data (AEAD)

- KeyGen: randomly chooses key, as always
- Enc(authenticated data A,
  private + auth data P, nonce N) →
  ciphertext C of length |C| > |P|
- Dec(C, A, N) → P or ⊥



Why combine authentication and encryption?

- Better security: resist some of these physical side channel attacks
- Simplicity: developers have fewer decisions (i.e., opportunities for mistakes)
- Performance: save in time + space costs, also often only need 1 key

AEAD as a picture