CSE 484 / CSE M 584: Computer Security and Privacy

Autumn 2019

Tadayoshi (Yoshi) Kohno
yoshi@cs.Washington.edu

Thanks to Dan Boneh, Dieter Gollmann, Dan Halperin, John Manferdelli, John Mitchell, Franzi Roesner, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials ...

Announcements

- My office hours
  - This week (10/30): Wed 12:30pm, CSE1 403
  - 11/6 (Wed), 1:30pm, CSE1 678 (small room, unfortunately)
  - 11/13 (Wed), 11:30am, CSE1 403
  - 11/20 (Wed), 2:30pm, CSE1 403
  - 11/27 (Wed), None
  - 12/4 (Wed), 12:30pm, CSE1 403
- TA office hours Wed and Fri this week as normal, but different TAs
- HW 2 available

Begin Crypto Review

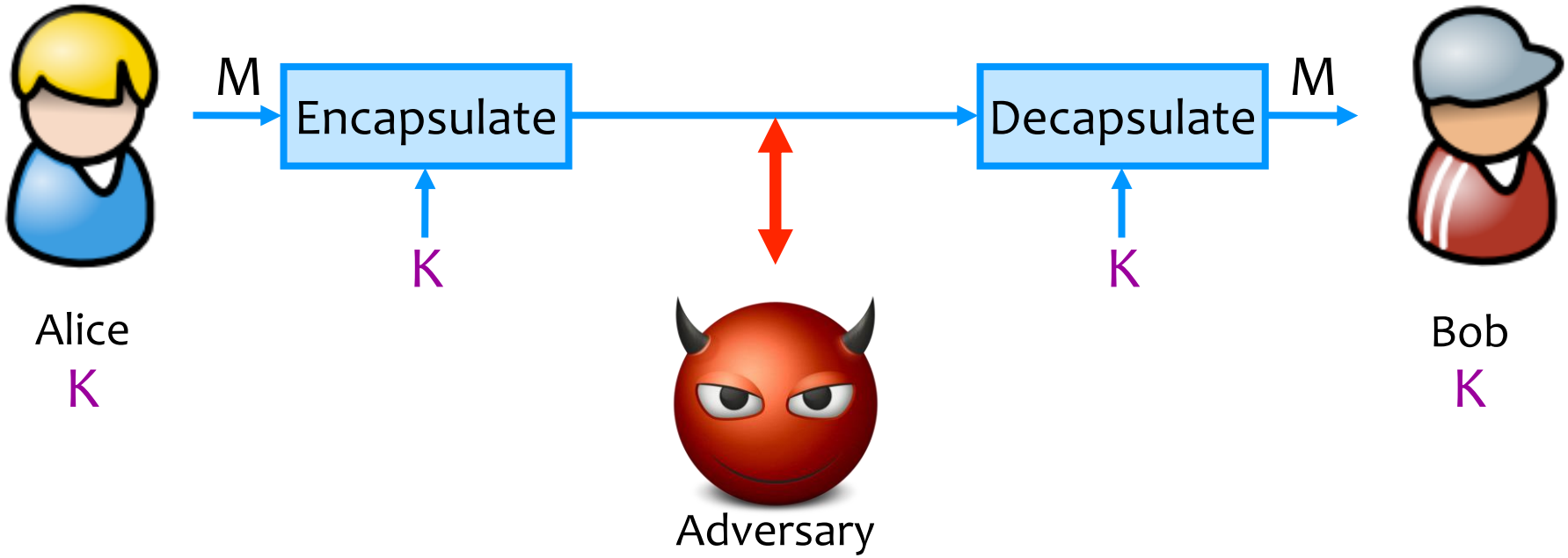
- Also quiz section tomorrow (including RSA)
- Also good anyway, recalling "spiral learning" process

Flavors of Cryptography

- Symmetric cryptography
  - Both communicating parties have access to a shared random string K, called the key.
- Asymmetric cryptography
  - Each party creates a public key pk and a secret key sk.

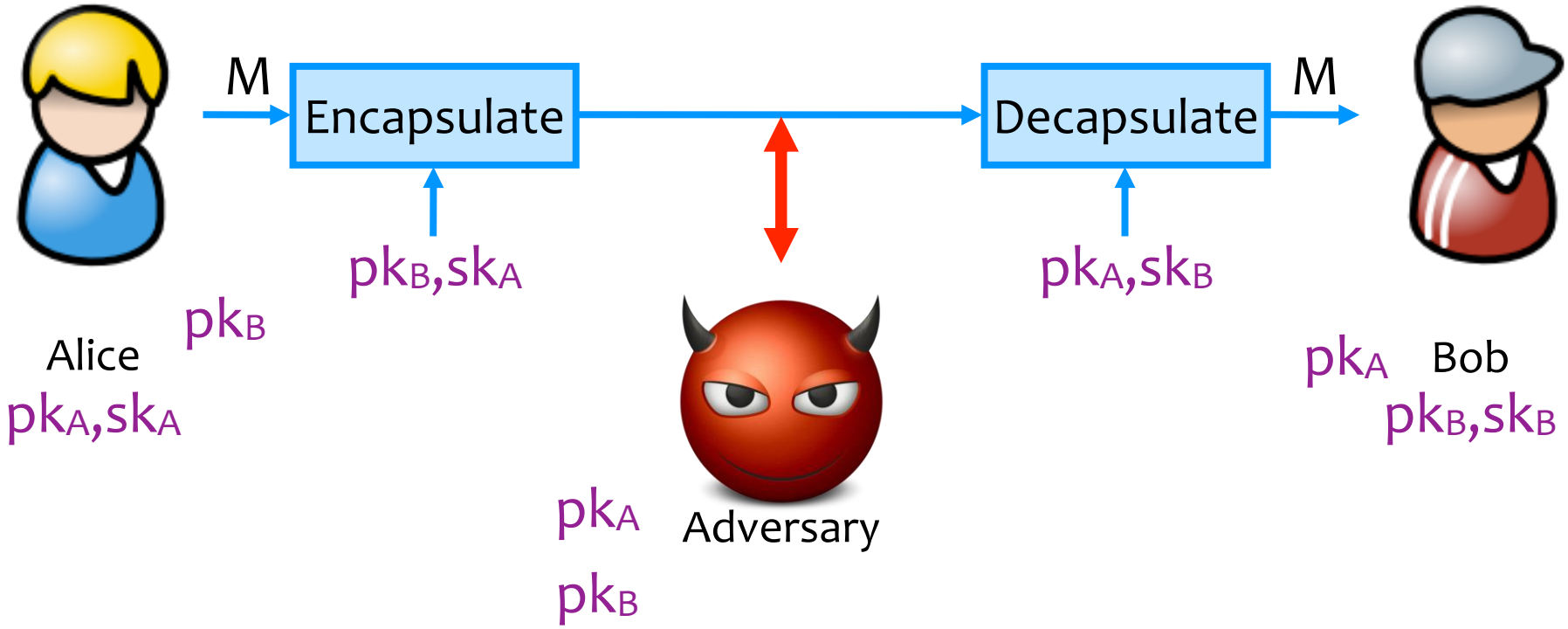
Symmetric Setting

Both communicating parties have access to a shared random string K, called the key.



Asymmetric Setting

Each party creates a public key pk and a secret key sk.



How Cryptosystems are Made

- Primitives first (like block ciphers or RSA)
  - Add today: Hash functions
- Schemes second (like ECB, CTR mode)
  - Add today: MACs
- Protocols third (like SSL/TLS, SSH)

End Review

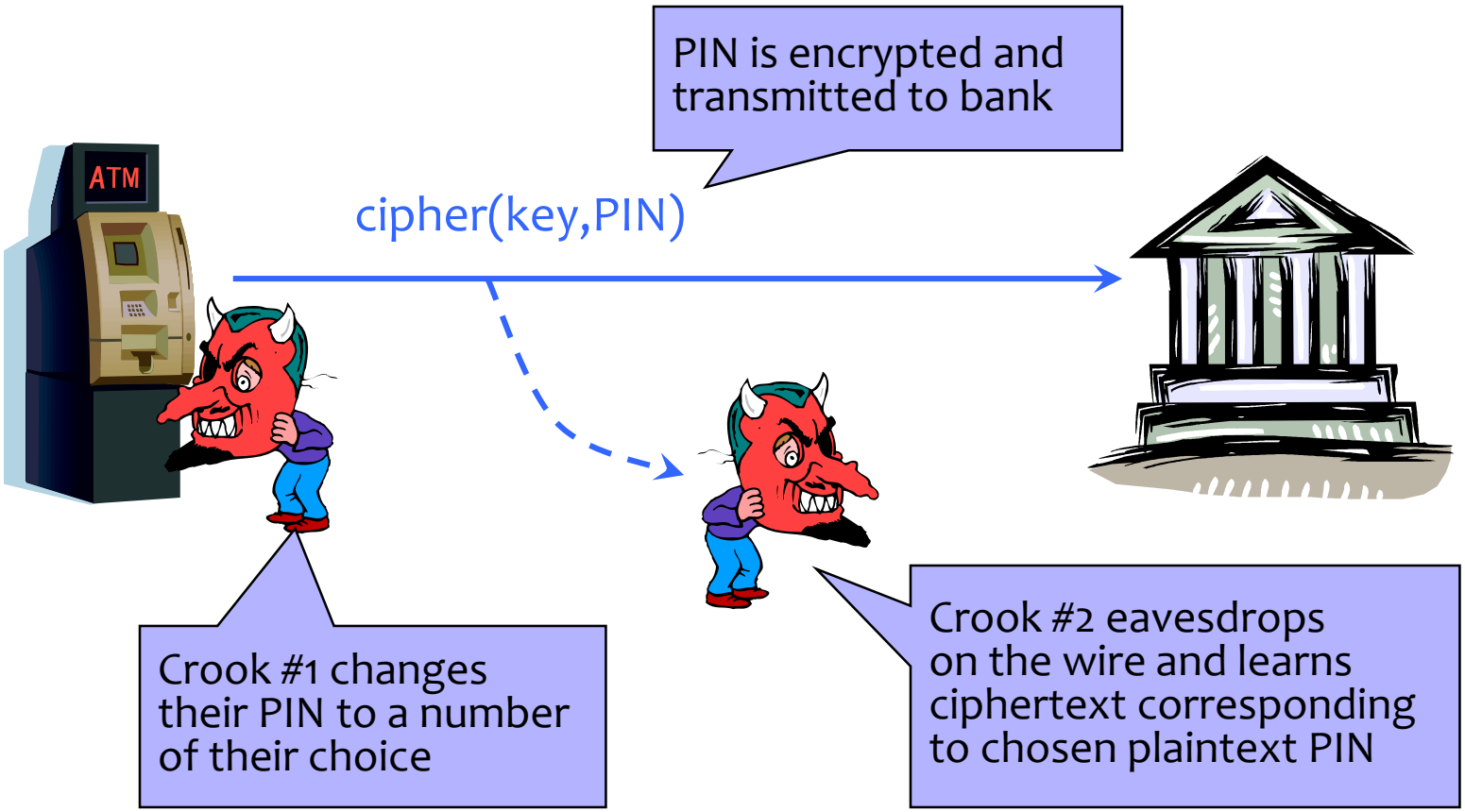
When is an Encryption Scheme "Secure"?

- Hard to recover the key?
  - What if attacker can learn plaintext without learning the key?
- Hard to recover plaintext from ciphertext?
  - What if attacker learns some bits or some function of bits?

How Can a Cipher Be Attacked?

- Attackers knows ciphertext and encryption algthm
  - What else does the attacker know? Depends on the application in which the cipher is used!
- Ciphertext-only attack
- KPA: Known-plaintext attack (stronger)
  - Knows some plaintext-ciphertext pairs
- CPA: Chosen-plaintext attack (even stronger)
  - Can obtain ciphertext for any plaintext of their choice
- CCA: Chosen-ciphertext attack (very strong)
  - Can decrypt any ciphertext except the target

Chosen Plaintext Attack



… repeat for any PIN value

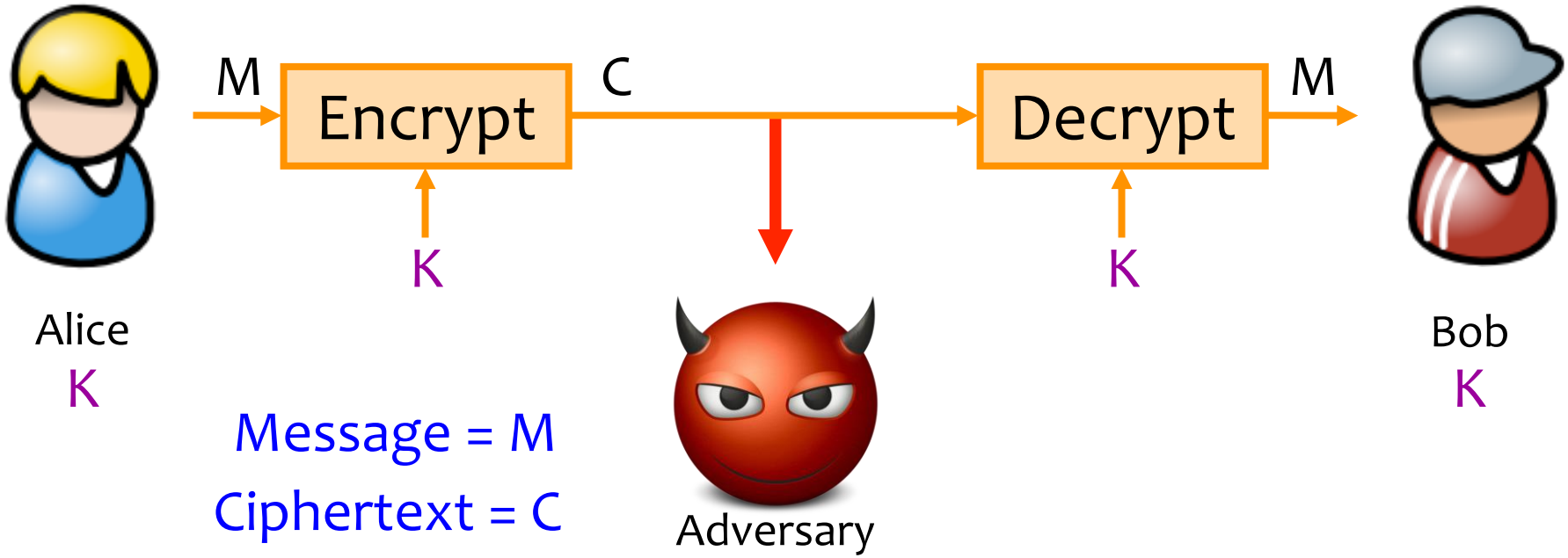
Very Informal Intuition

Minimum security requirement for a modern encryption scheme

- Security against chosen-plaintext attack (CPA)
  - Ciphertext leaks no information about the plaintext
  - Even if the attacker correctly guesses the plaintext, they cannot verify their guess
  - Every ciphertext is unique, encrypting same message twice produces completely different ciphertexts (w.h.p.)
    - Implication: encryption must be randomized or stateful
- Security against chosen-ciphertext attack (CCA)
  - CCA security leverages integrity protection – making it hard to change the plaintext by modifying the ciphertext

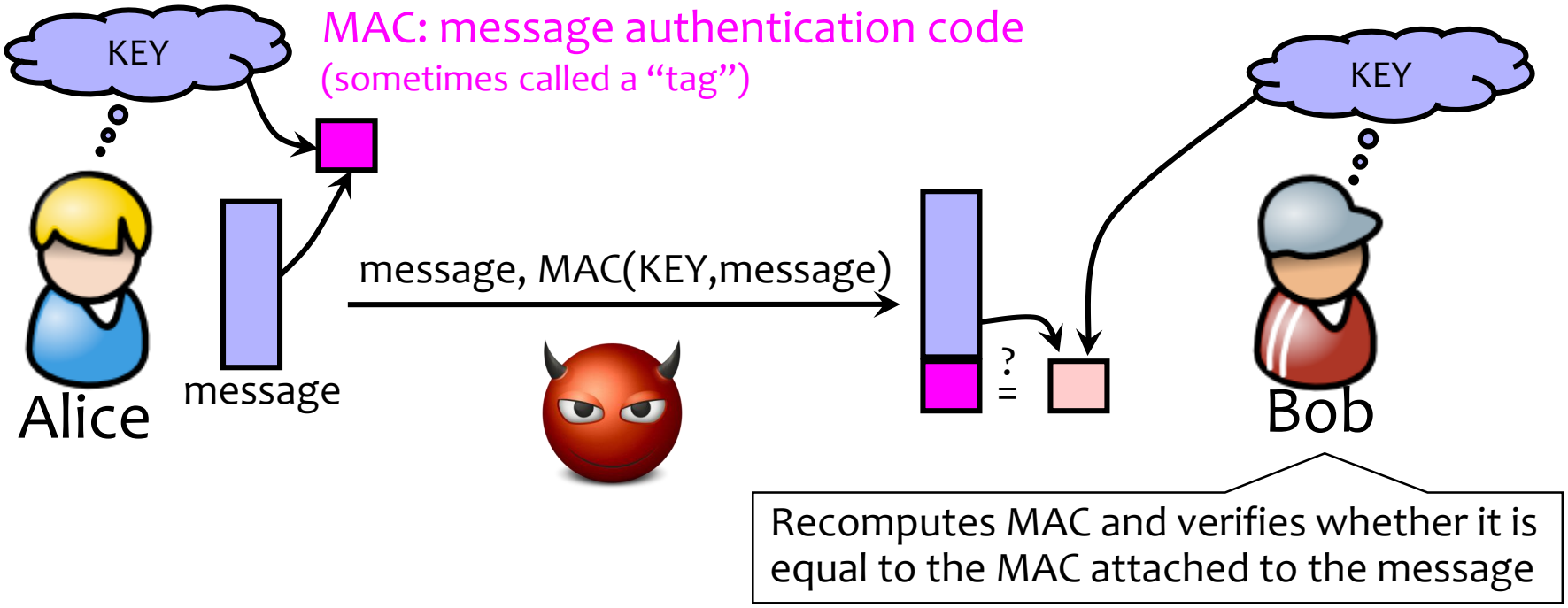
So Far: Achieving Privacy

Encryption schemes: A tool for protecting privacy.
(Figure below is for the symmetric setting)



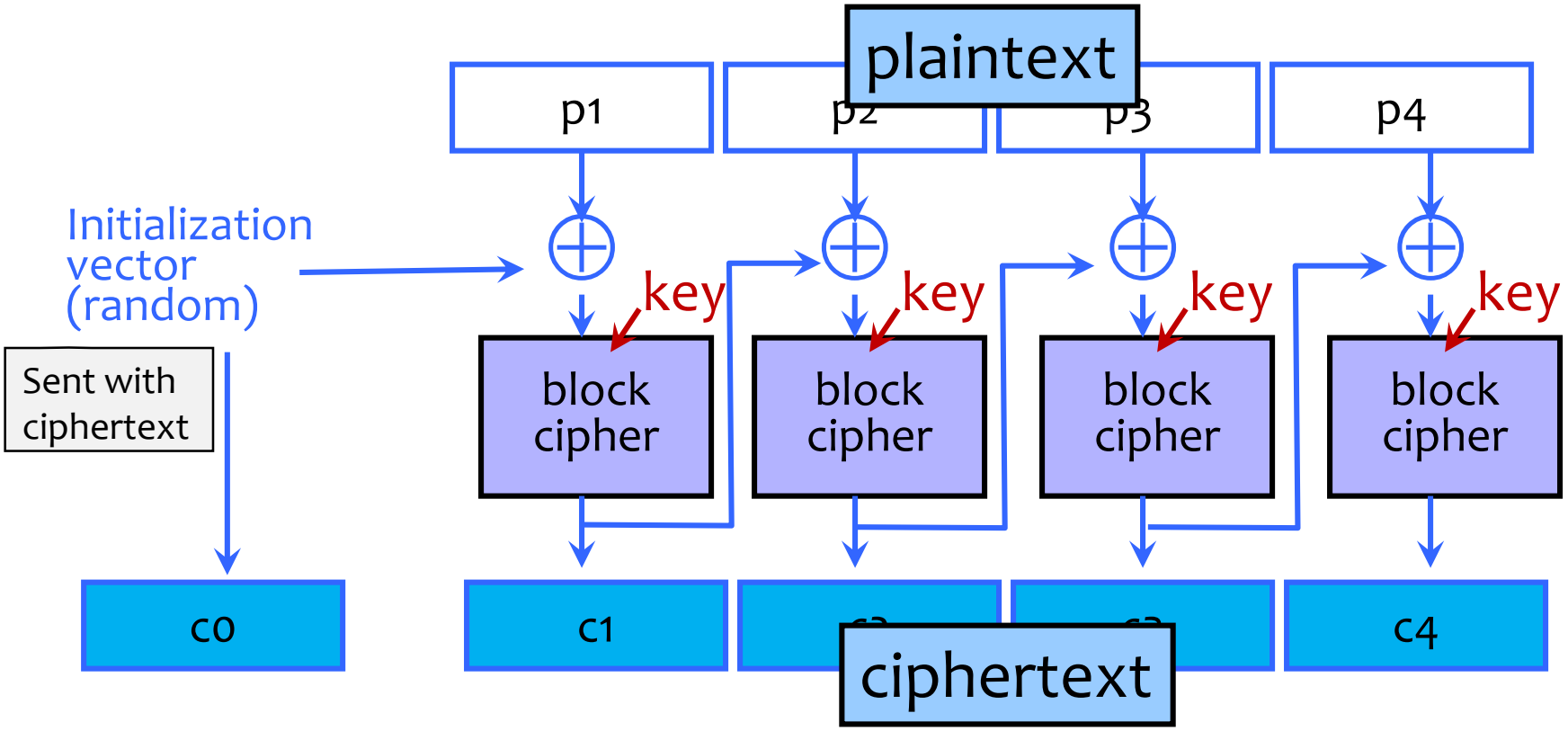
Now: Achieving Integrity

Message authentication schemes: A tool for protecting integrity.



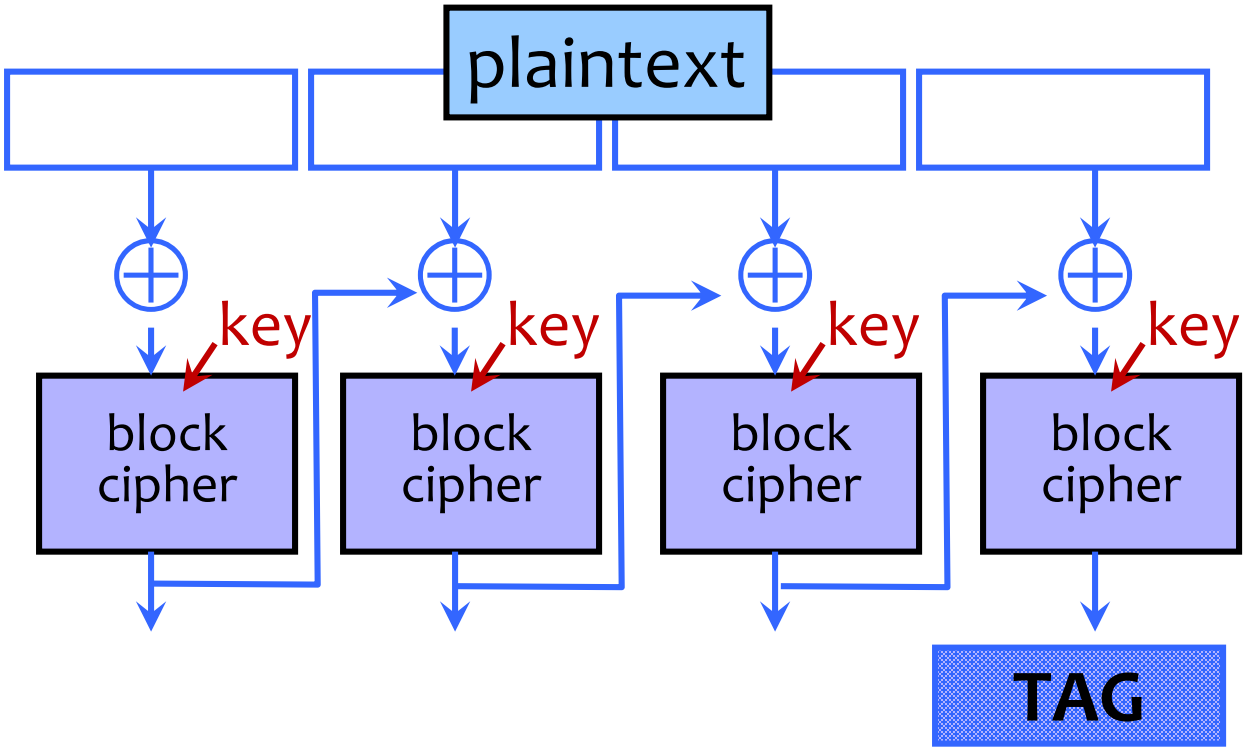
Integrity and authentication: only someone who knows KEY can compute correct MAC for a given message.

Reminder: CBC Mode Encryption



- Identical blocks of plaintext encrypted differently
- Last cipherblock depends on entire plaintext
  - Still does not guarantee integrity

CBC-MAC

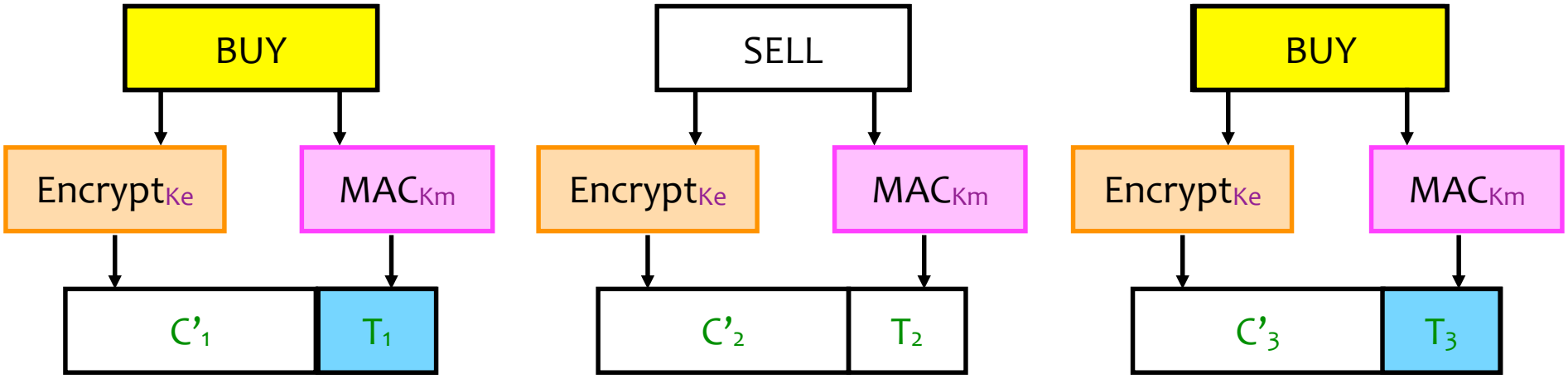


- Not secure when system may MAC messages of different lengths.
- Use different key than when encrypting with CBC mode
- NIST recommends a derivative called CMAC [FYI only]
- Other MACs exist, e.g., GMAC, UMAC, HMAC (next section)

Next Topic

Authenticated Encryption

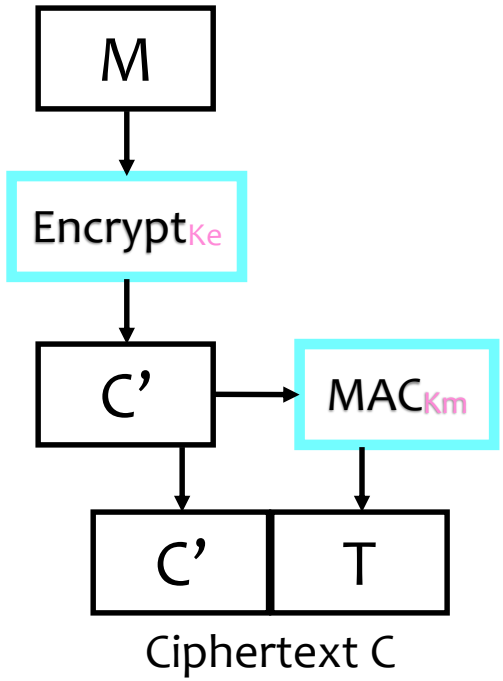
- What if we want both privacy and integrity?
- Natural approach: combine encryption scheme and a MAC.
- But be careful!
  - Obvious approach: Encrypt-and-MAC
  - Problem: MAC is deterministic! same plaintext → same MAC



Authenticated Encryption

- Instead:
  Encrypt *then* MAC.

- (Not as good:
  MAC-then-Encrypt)



Encrypt-then-MAC

Next Topic

Password Storage

- Question on Worksheet: Suppose you are creating a new website, and you expect millions of users. How will you store those user's usernames and passwords, so that users can authenticate later but an adversary who breaks into your computers and steals all your data can't easily figure out everyone's password?

Properties of a Solution

- When user enters password, server needs to verify that password is correct

- Could store passwords in plaintext
  - When user enters password, server can compare with stored password
  - But if computer compromised, all passwords leaked
- Could store passwords encrypted
  - When user enters password, server can decrypt stored password and check password
  - But if computer compromised, adversary also gets access to key

- Need to store password in a way that enables verification but not immediate loss of password after server compromise

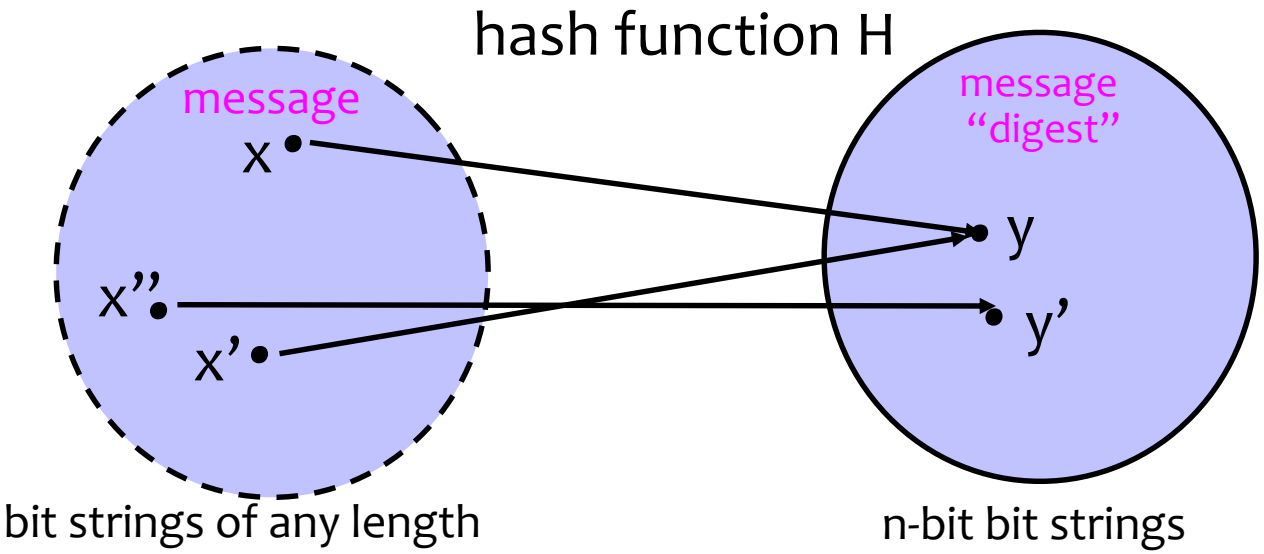
Audit Logs / Transaction Logs

- Suppose you have data about a financial transaction that you would like to be verifiable to someone else who knows that data
  - Verifiable means unmodified by you
  - Verifiable includes not just the data, but the date+time of the transaction

- Solution: write a function of the data + date+time into a public ledger
  - Basis of "blockchain"
  - But what function should one use?

Another Tool: Hash Functions

Hash Functions: Main Idea



- Hash function H is a lossy compression function
  - Collision: h(x)=h(x') for distinct inputs x, x'
- H(x) should look "random"
  - Every bit (almost) equally likely to be 0 or 1
- Cryptographic hash function needs a few properties…

Property 1: One-Way

- Intuition: hash should be hard to invert
  - "Preimage resistance"
  - Let h(x') = y in $\{0,1\}^n$ for a random x'
  - Given y, it should be hard to find any x such that h(x)=y
- How hard?
  - Brute-force: try every possible x, see if h(x)=y
  - SHA-1 (common hash function) has 160-bit output
    - Expect to try 2^{159} inputs before finding one that hashes to y.

Property 2: Collision Resistance

- Should be hard to find $x \neq x'$ such that $h(x)=h(x')$

Birthday Paradox

- Are there two people in the first few rows of this classroom that have the same birthday?
  - 365 days in a year (366 some years)
    - Pick one person. To find another person with same birthday would take on the order of 365/2 = 182.5 people
    - Expect birthday "collision" with a room of only 23 people.
    - For simplicity, approximate when we expect a collision as sqrt(365).
- Why is this important for cryptography?
  - 2^{128} different 128-bit values
    - Pick one value at random. To exhaustively search for this value requires trying on average 2^{127} values.
    - Expect "collision" after selecting approximately 2^{64} random values.
    - 64 bits of security against collision attacks, not 128 bits.

Property 2: Collision Resistance

- Should be hard to find $x \neq x'$ such that $h(x)=h(x')$
- Birthday paradox means that brute-force collision search is only $O(2^{n/2})$, *not* $O(2^n)$
  - For SHA-1, this means $O(2^{80})$ vs. $O(2^{160})$

One-Way vs. Collision Resistance (Primarily FYI Re: Details)

- One-wayness does not imply collision resistance
  - Suppose g is one-way
  - Define h(x) as g(x') where x' is x except the last bit
    - h is one-way (to invert h, must invert g)
    - Collisions for h are easy to find: for any x, h(x0)=h(x1)
- Collision resistance does not imply one-wayness
  - Suppose g is collision-resistant
  - Define y=h(x) to be 0x if x is n-bit long, 1g(x) otherwise
    - Collisions for h are hard to find: if y starts with 0, then there are no collisions, if y starts with 1, then must find collisions in g
    - h is not one way: half of all y's (those whose first bit is 0) are easy to invert (how?); random y is invertible with probab. ½

Property 3: Weak Collision Resistance

- Given randomly chosen x, hard to find x' such that h(x)=h(x')
  - Attacker must find collision for a specific x. By contrast, to break collision resistance it is enough to find any collision.
  - Brute-force attack requires $O(2^n)$ time
- Weak collision resistance does not imply collision resistance.

Hashing vs. Encryption

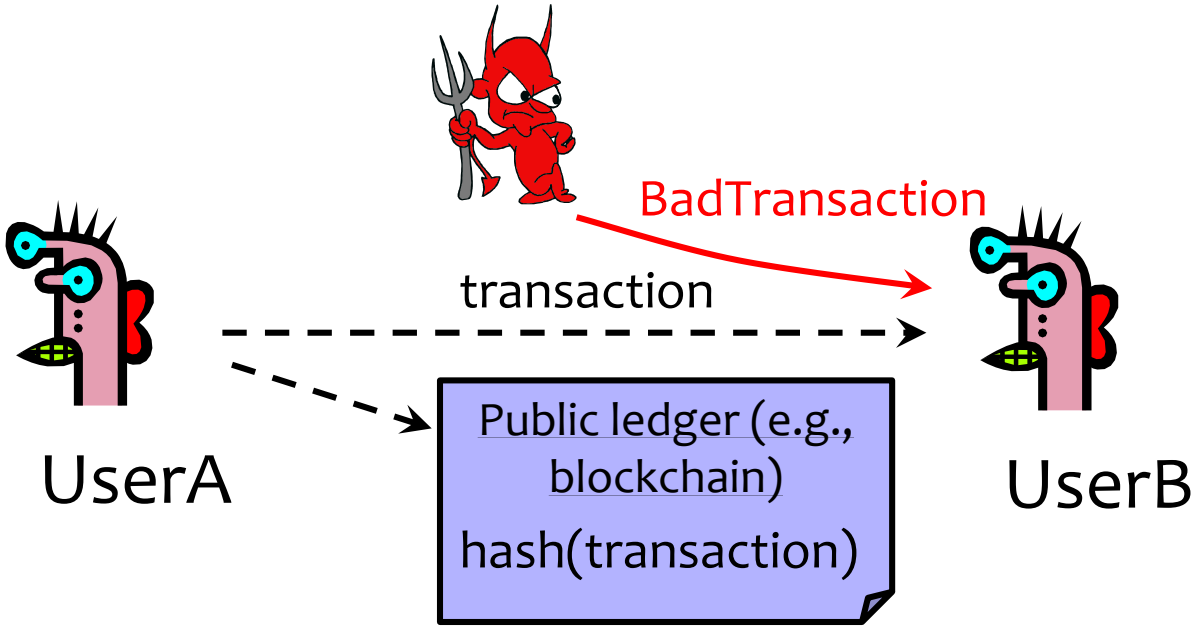
- Hashing is one-way. There is no "un-hashing"
  - A ciphertext can be decrypted with a decryption key… hashes have no equivalent of "decryption"
- Hash(x) looks "random" but can be compared for equality with Hash(x')
  - Hash the same input twice → same hash value
  - Encrypt the same input twice → different ciphertexts
- Crytographic hashes are also known as "cryptographic checksums" or "message digests"

Application: Password Hashing

- Instead of user password, store
  - (random string, hash(random string, password))
- When user enters a password, compute its hash and compare with the entry in the password file
- Why is hashing better than encryption here?

- System does not store actual passwords!
- Cannot go from hash to password! (Unless try all possible passwords)

Application: Financial Transactions



Goal: Strore information about transactions in a way that anyone can verify (if they know details of transaction)

Idea: given transaction and hash(transaction), very hard to find BadTransaction such that hash(transaction)=hash(BadTransaction)

Idea: hard for UserA to find transaction1 and transaction2 such that hash(transaction1) = hash(transaction2)

Which Property Do We Need?

- UNIX passwords stored as hash(password)
  - **One-wayness:** hard to recover the/a valid password
- Financial transactions
  - **Weak collision resistance** (first example)
  - **Collision resistance** (second example)
- Auction bidding
  - Alice wants to bid B, sends H(B), later reveals B
  - **One-wayness:** rival bidders should not recover B (this may mean that she needs to hash some randomness with B too)
  - **Collision resistance:** Alice should not be able to change her mind to bid B' such that H(B)=H(B')