CSE 484 / CSE M 584: Applied Cryptography

Winter 2023

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

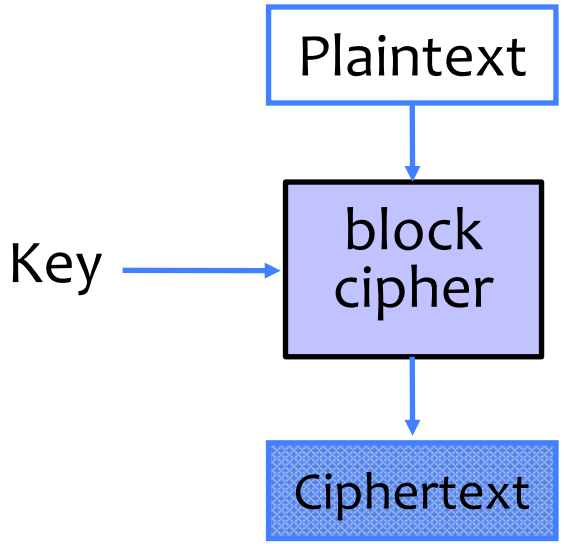
UW Instruction Team: David Kohlbrenner, Yoshi Kohno, Franziska Roesner. Thanks to Dan Boneh, Dieter Gollmann, Dan Halperin, John Manferdelli, John Mitchell, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials ...

Announcements / Plan

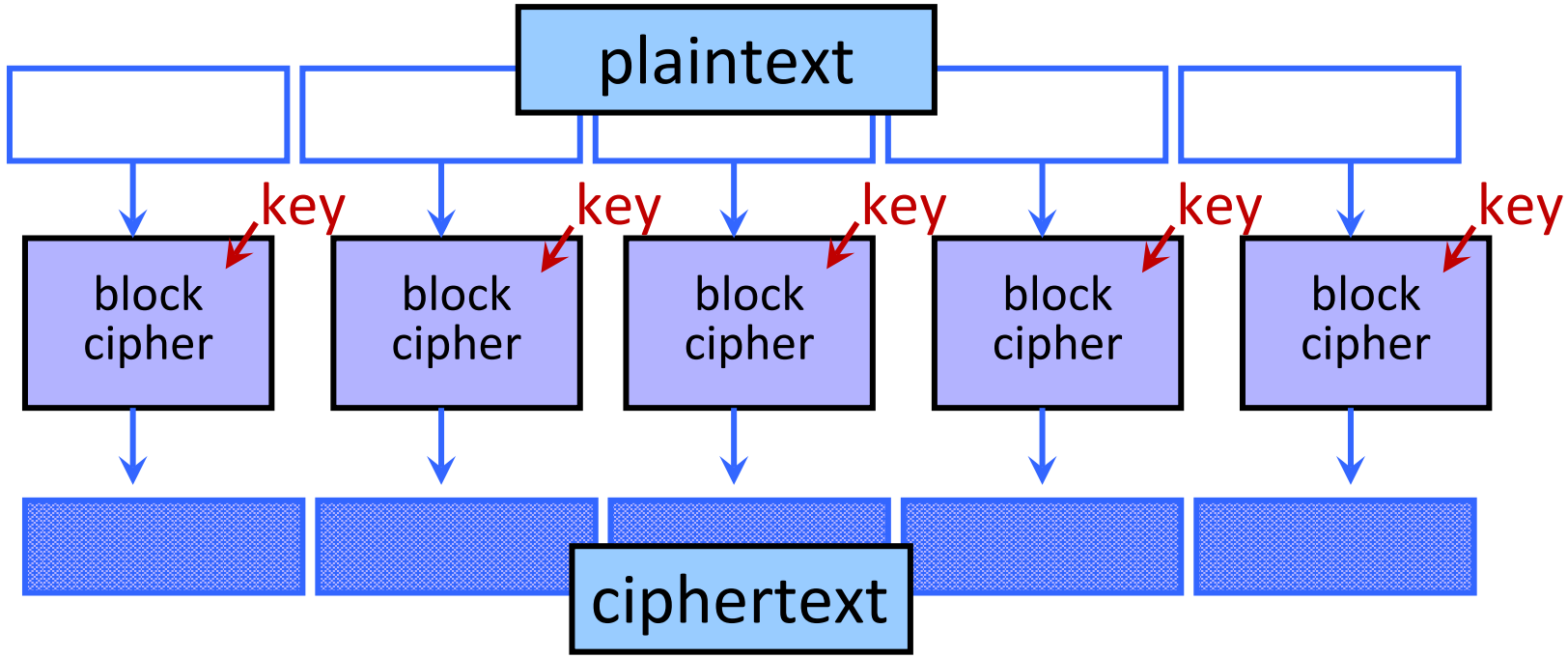
- Monday (1/30): Applied Cryptography
- Wednesday (2/1): Guest Lecture: Prof. Gennie Gebhart (EFF and UW) (On Zoom)
- Friday (2/3) through Wednesday (2/8): Applied Crypto
- Friday (2/10): Guest Lecture: Prof. Elissa Redmiles (MPI)
- Wednesday (2/22): At most Zoom
- Friday (2/24): Guest Lecture: Alex Gantman (Qualcomm) (On Zoom)

Review: Block Ciphers

- Operates on a single chunk ("block") of plaintext
  - For example, 64 bits for DES, 128 bits for AES
  - Each key defines a different permutation
  - Same key is reused for each block (can use short keys)

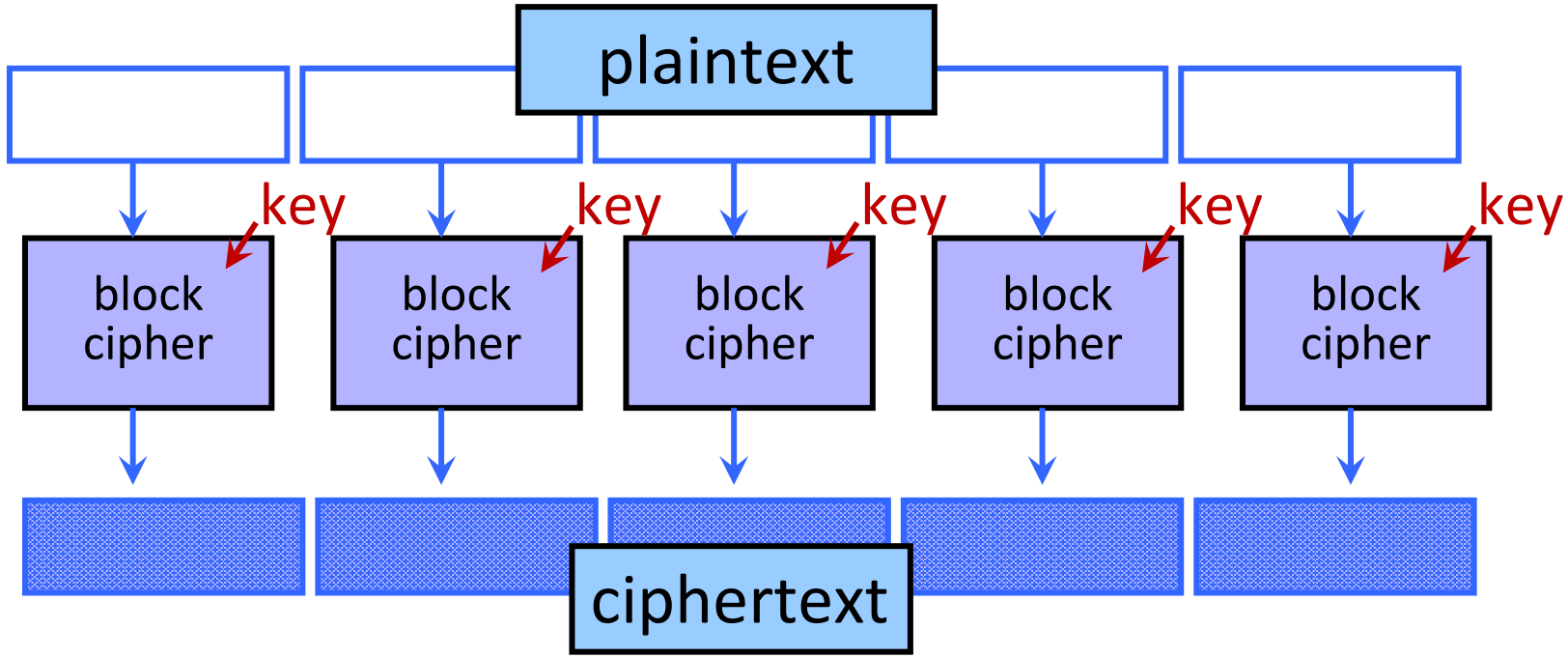


Review: Electronic Code Book (ECB) Mode



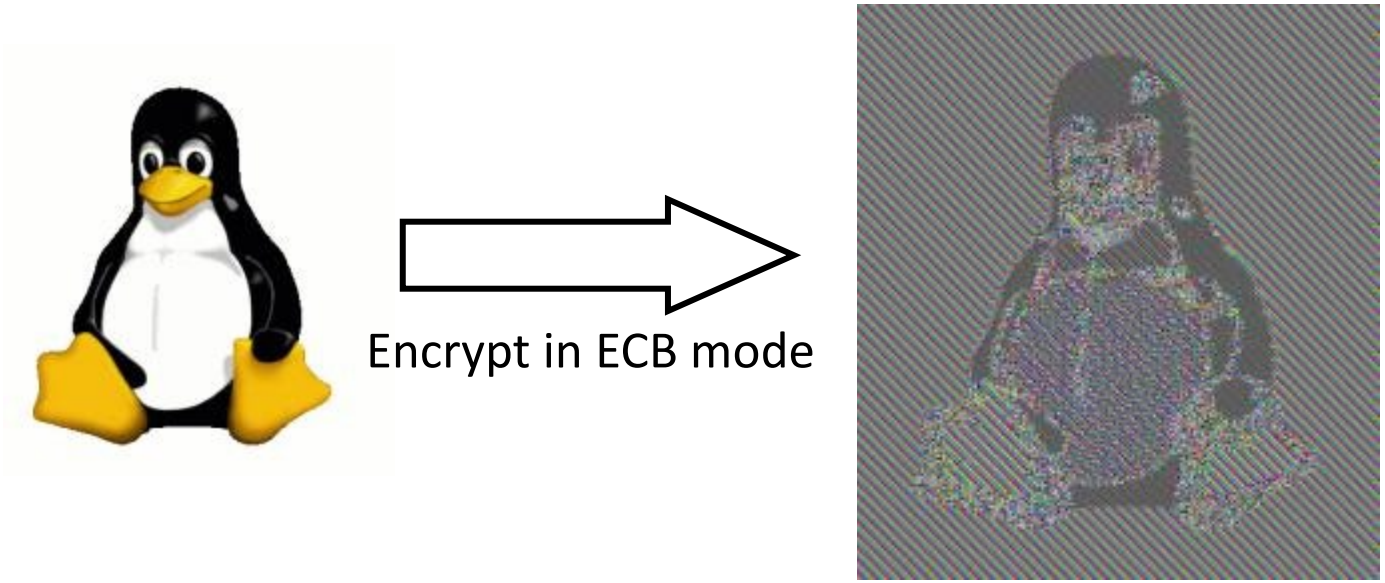
Canvas "quiz" time!

Review: Electronic Code Book (ECB) Mode



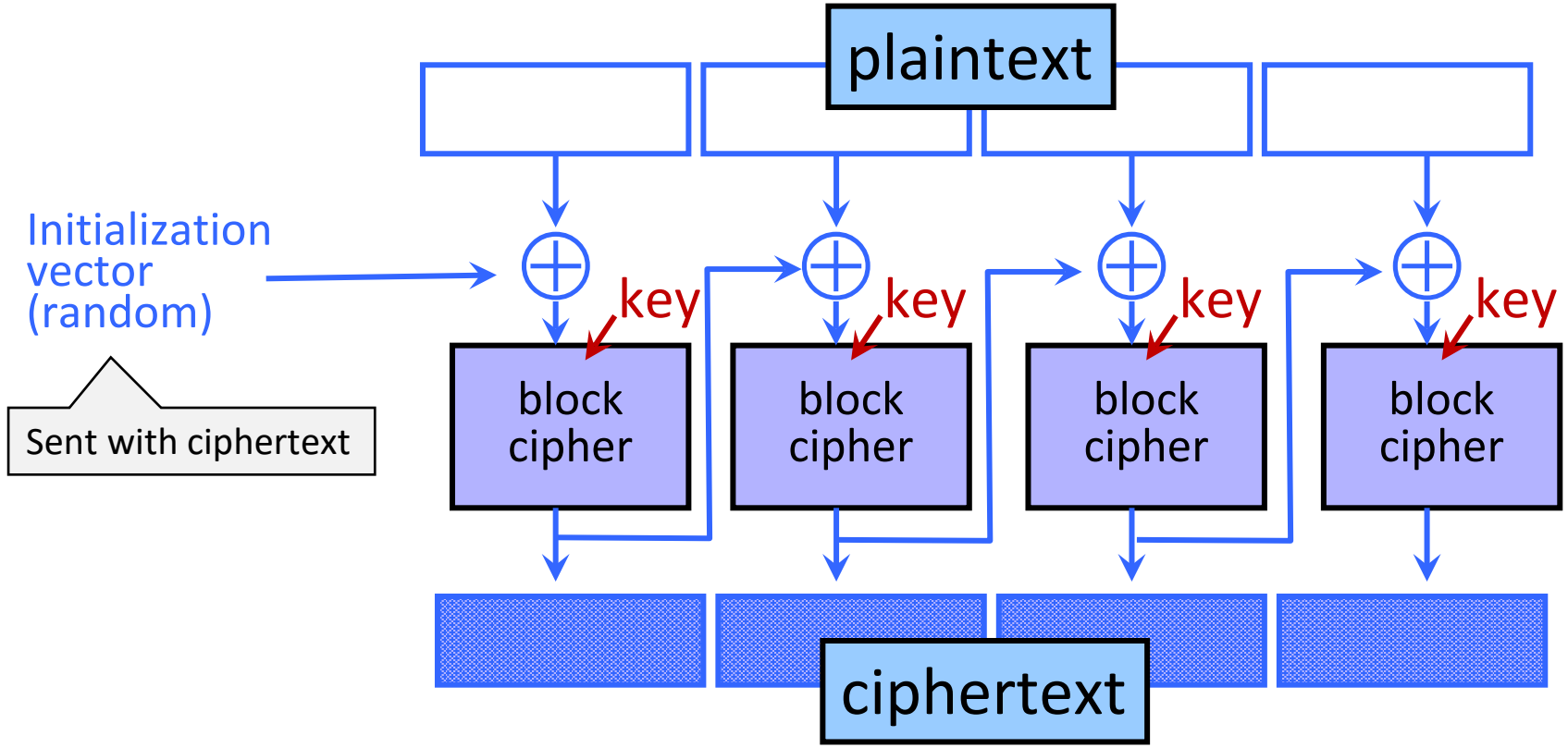
- Identical blocks of plaintext produce identical blocks of ciphertext
- No integrity checks: can mix and match blocks

Review: Information Leakage in ECB Mode



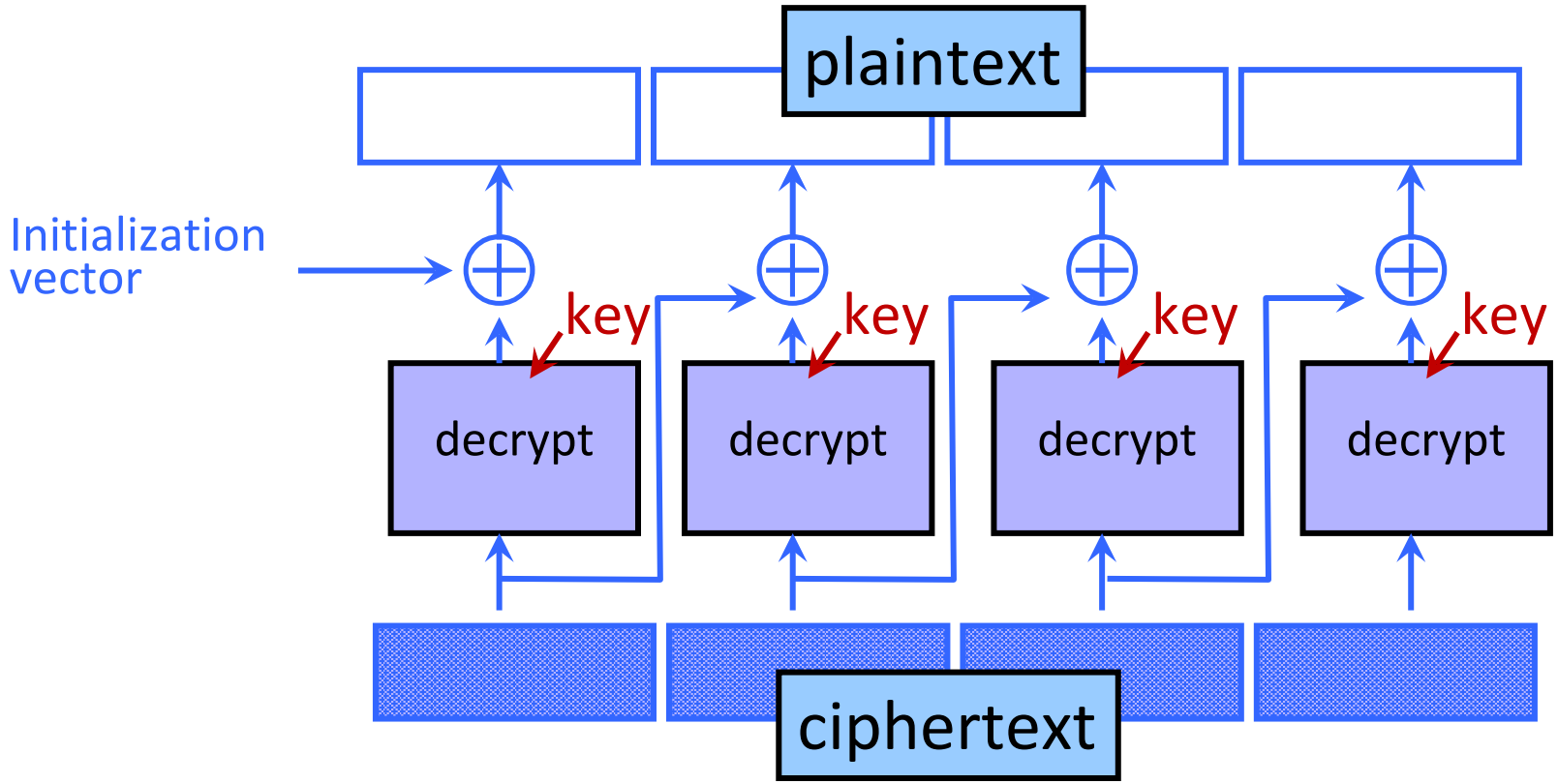
[Wikipedia]

Review: Cipher Block Chaining (CBC) Mode: Encryption

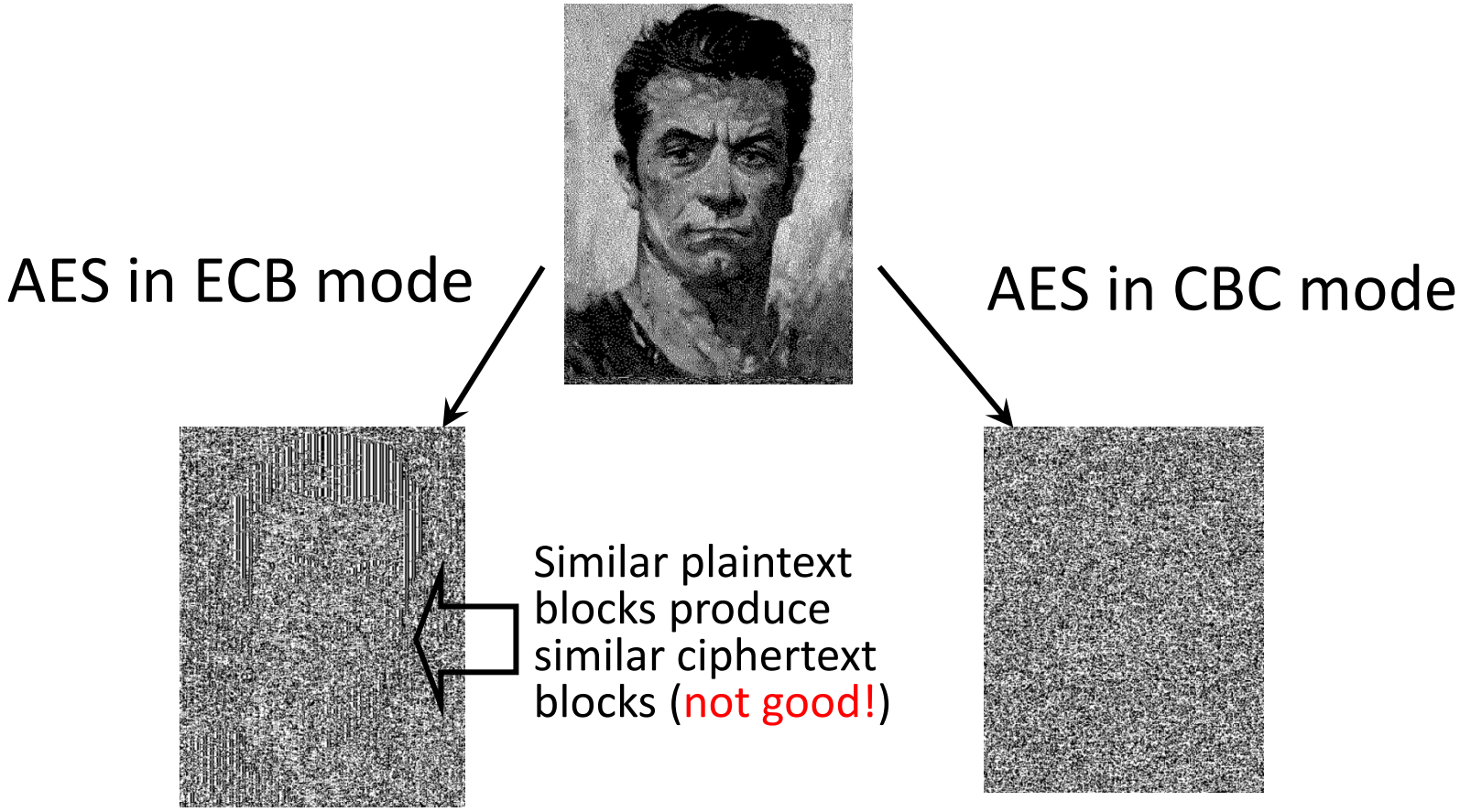


- Identical blocks of plaintext encrypted differently
- Last cipherblock depends on entire plaintext
  - Still does not guarantee integrity

Review: CBC Mode: Decryption

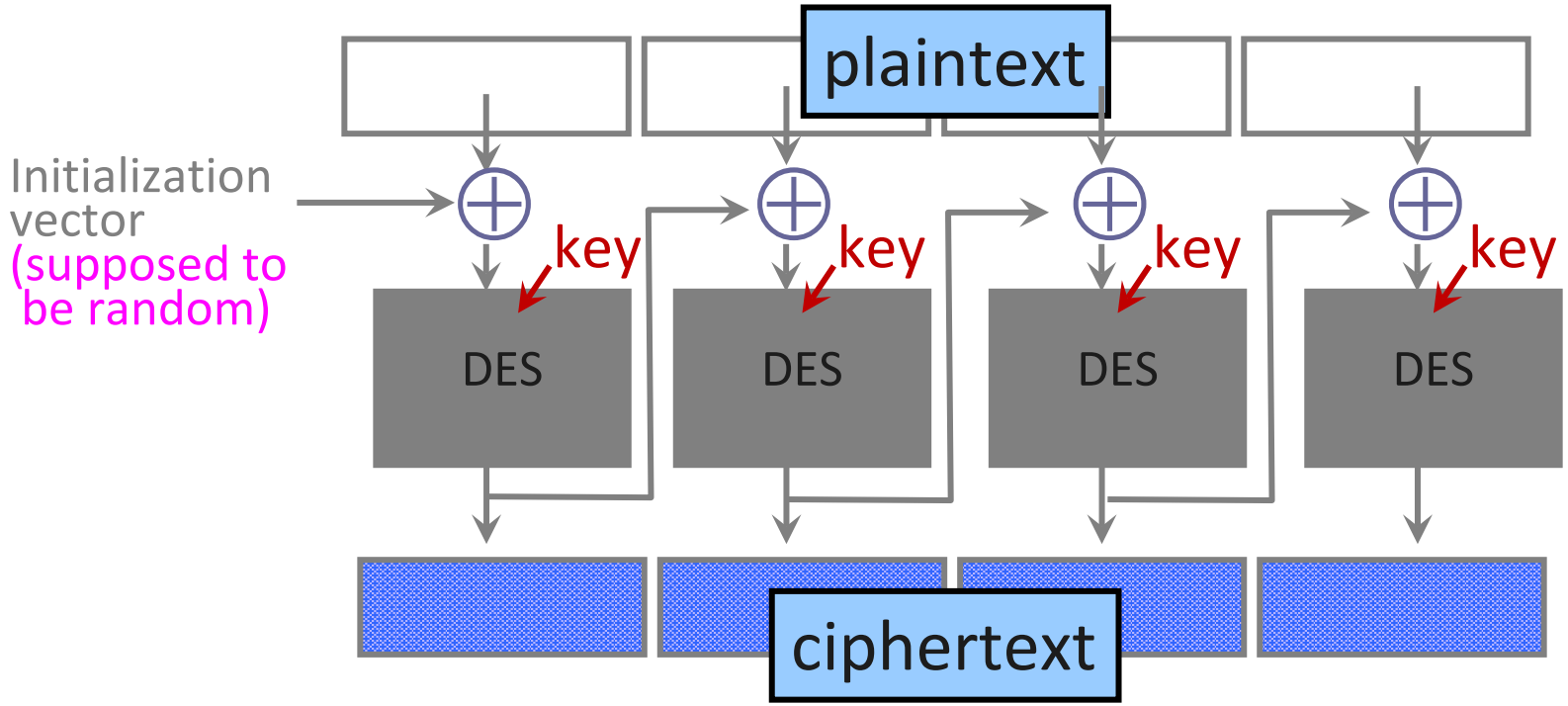


Review: ECB vs. CBC



[Picture due to Bart Preneel]

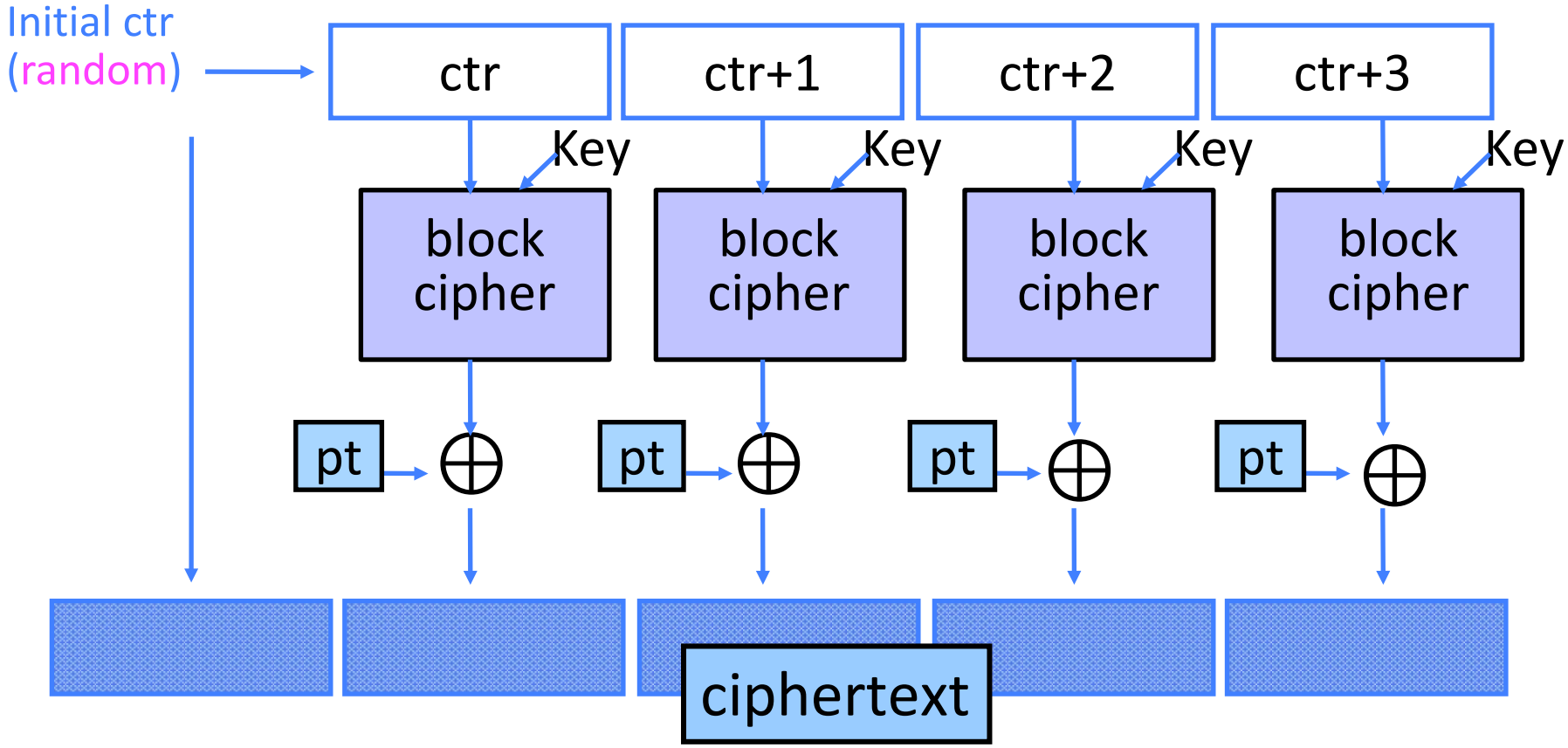
Initialization Vector Dangers



Found in the source code for Diebold voting machines:

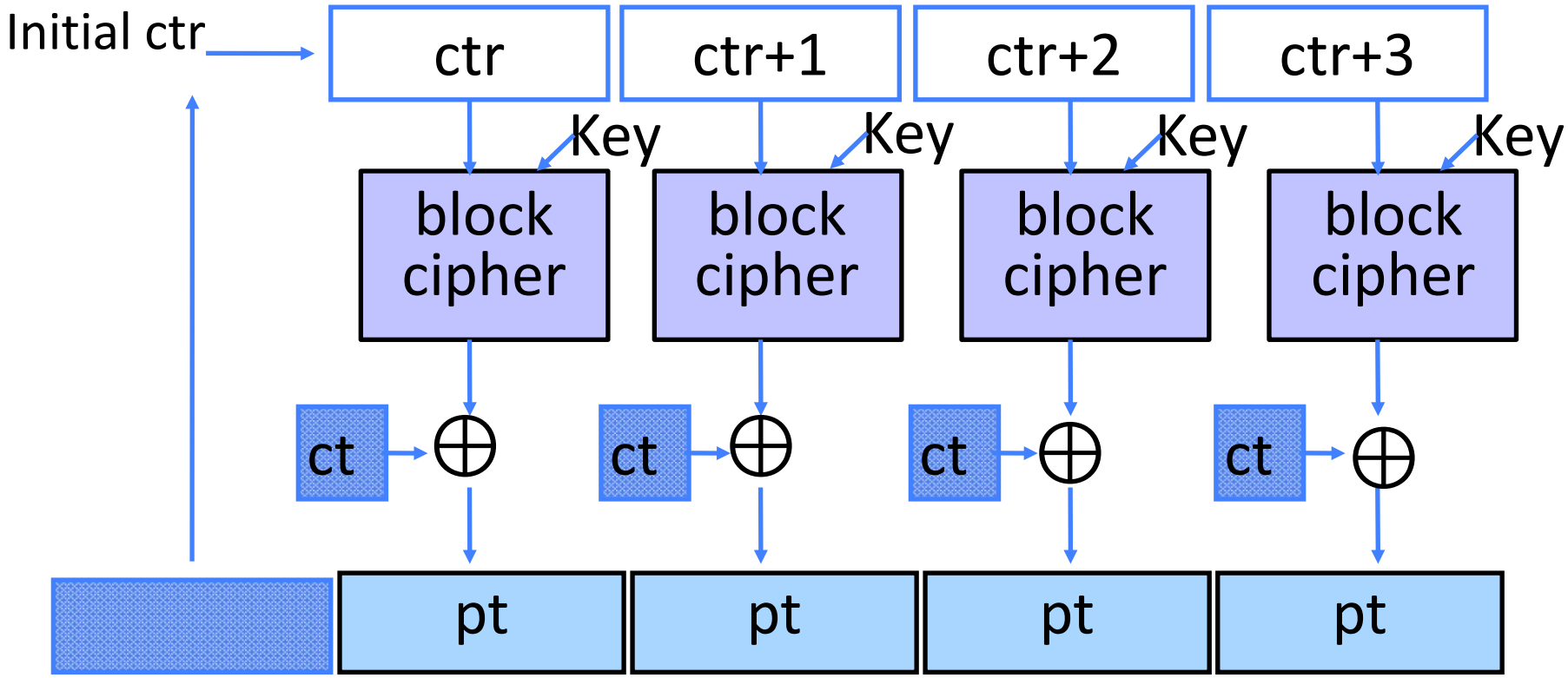
```
DesCBCEncrypt((des_c_block*)tmp, (des_c_block*)record.m_Data,
        totalSize, DESKEY, NULL, DES_ENCRYPT)
```

Counter Mode (CTR): Encryption



- Identical blocks of plaintext encrypted differently
- Still does not guarantee integrity; Fragile if ctr repeats

Counter Mode (CTR): Decryption



Ok, so what mode do I use?

- Don’t choose a mode, use established libraries ☺

- Good modes:
  - GCM - Galois/Counter Mode
  - CTR (sometimes)
  - Even ECB is fine in ‘the right circumstance’

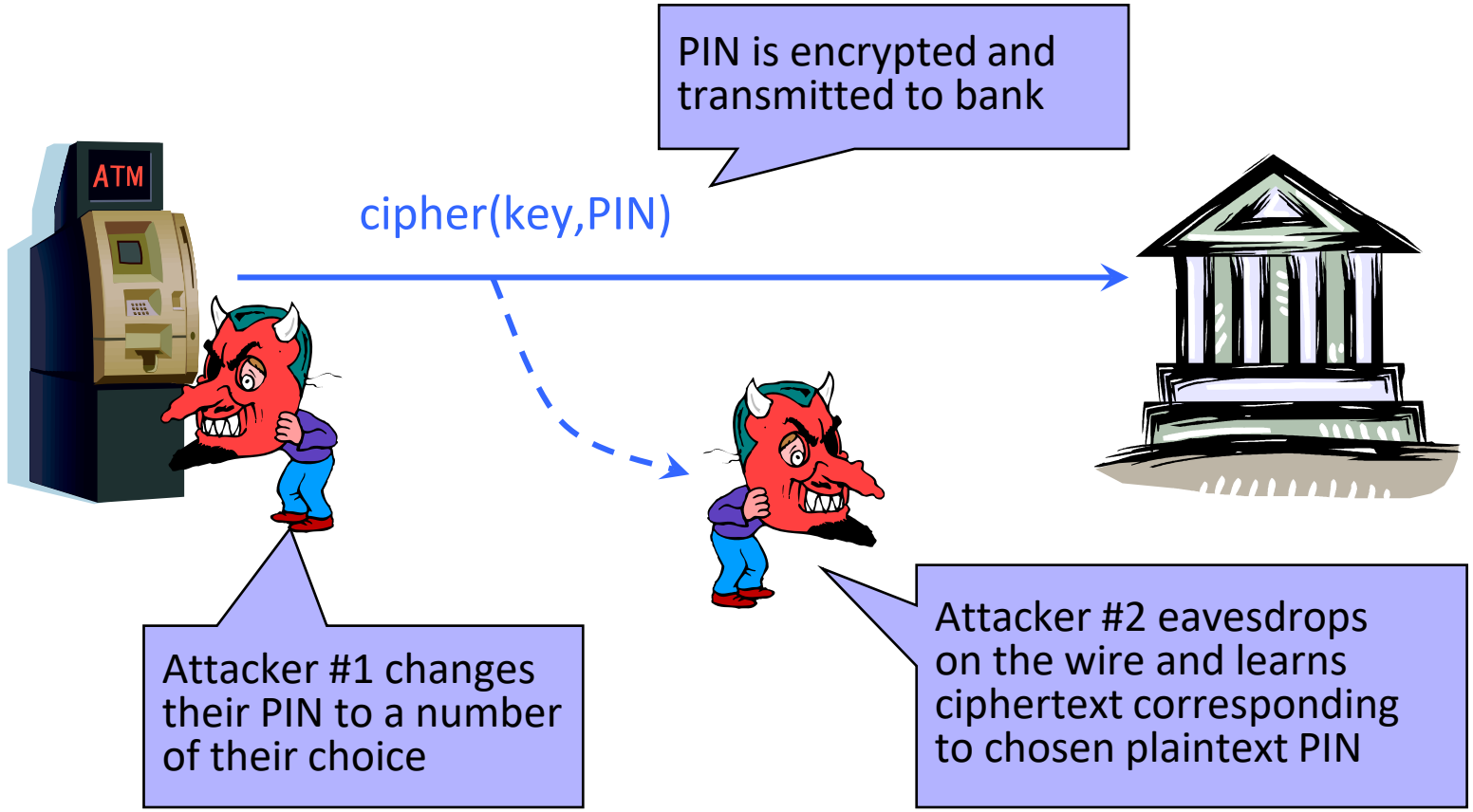
When is an Encryption Scheme "Secure"?

- 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 algorithm
  - 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 his choice
- CCA: Chosen-ciphertext attack (very strong)
  - Can decrypt any ciphertext except the target

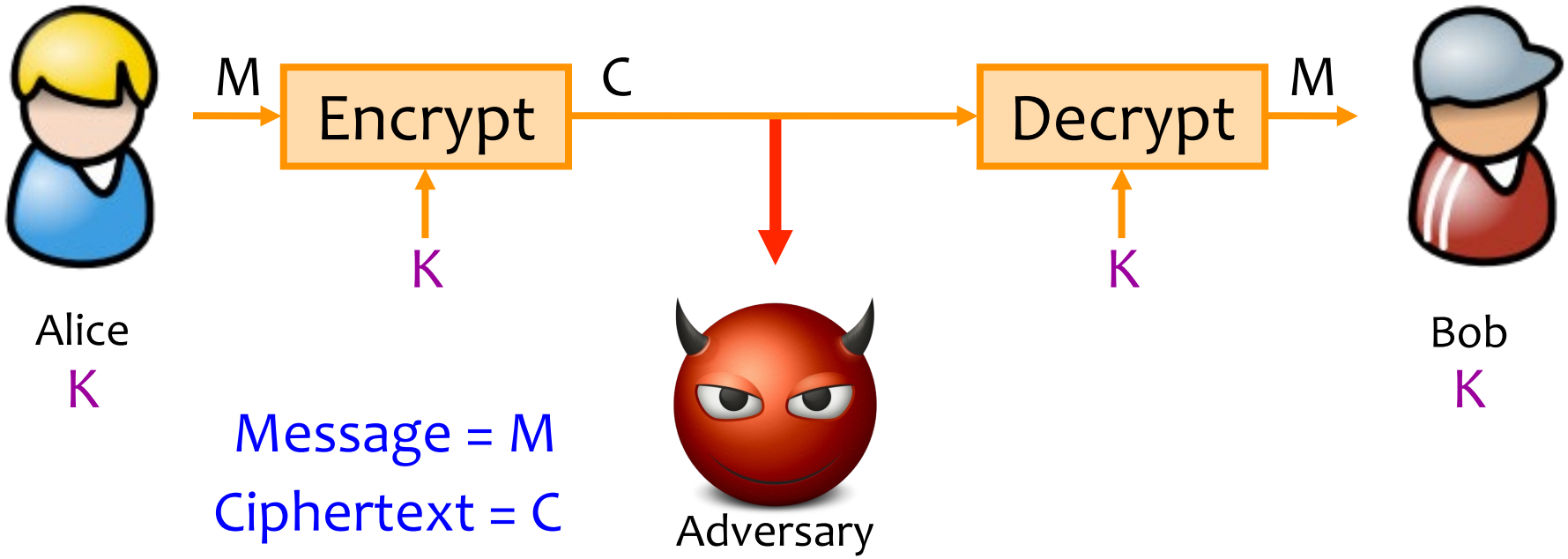
Chosen Plaintext Attack



… repeat for any PIN value

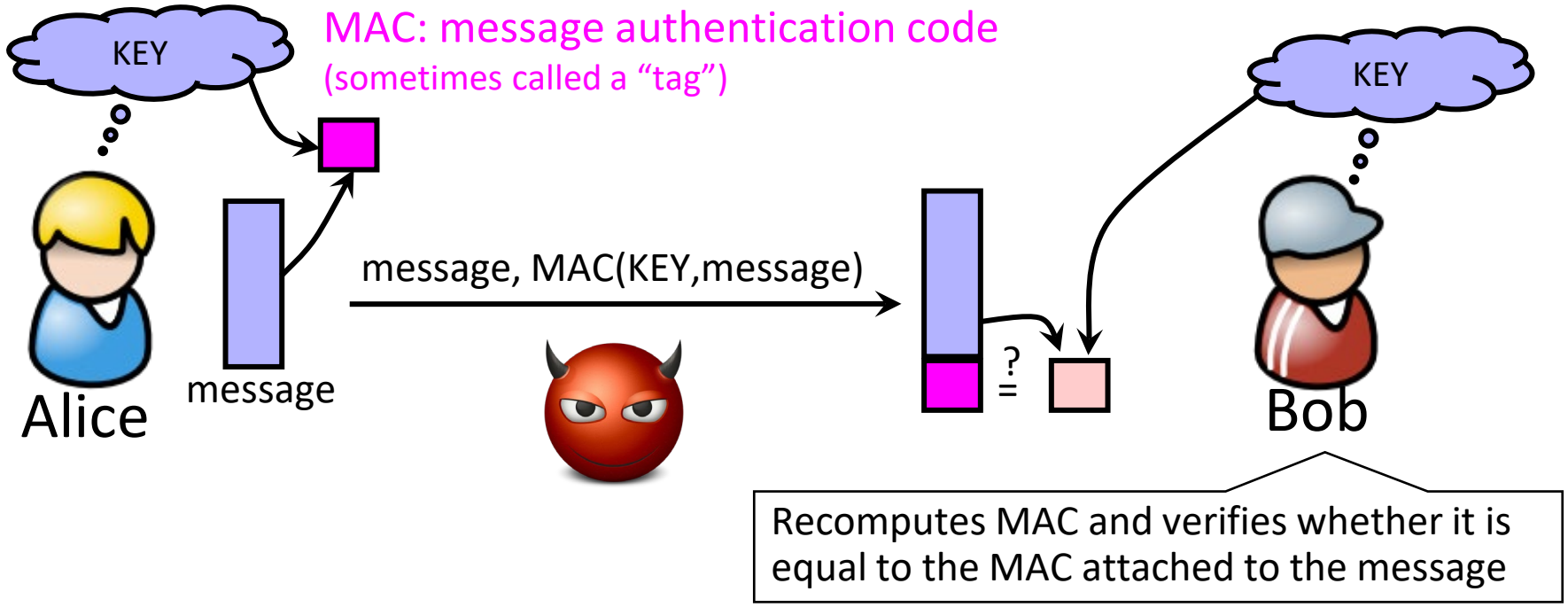
So Far: Achieving Privacy

Encryption schemes: A tool for protecting privacy.



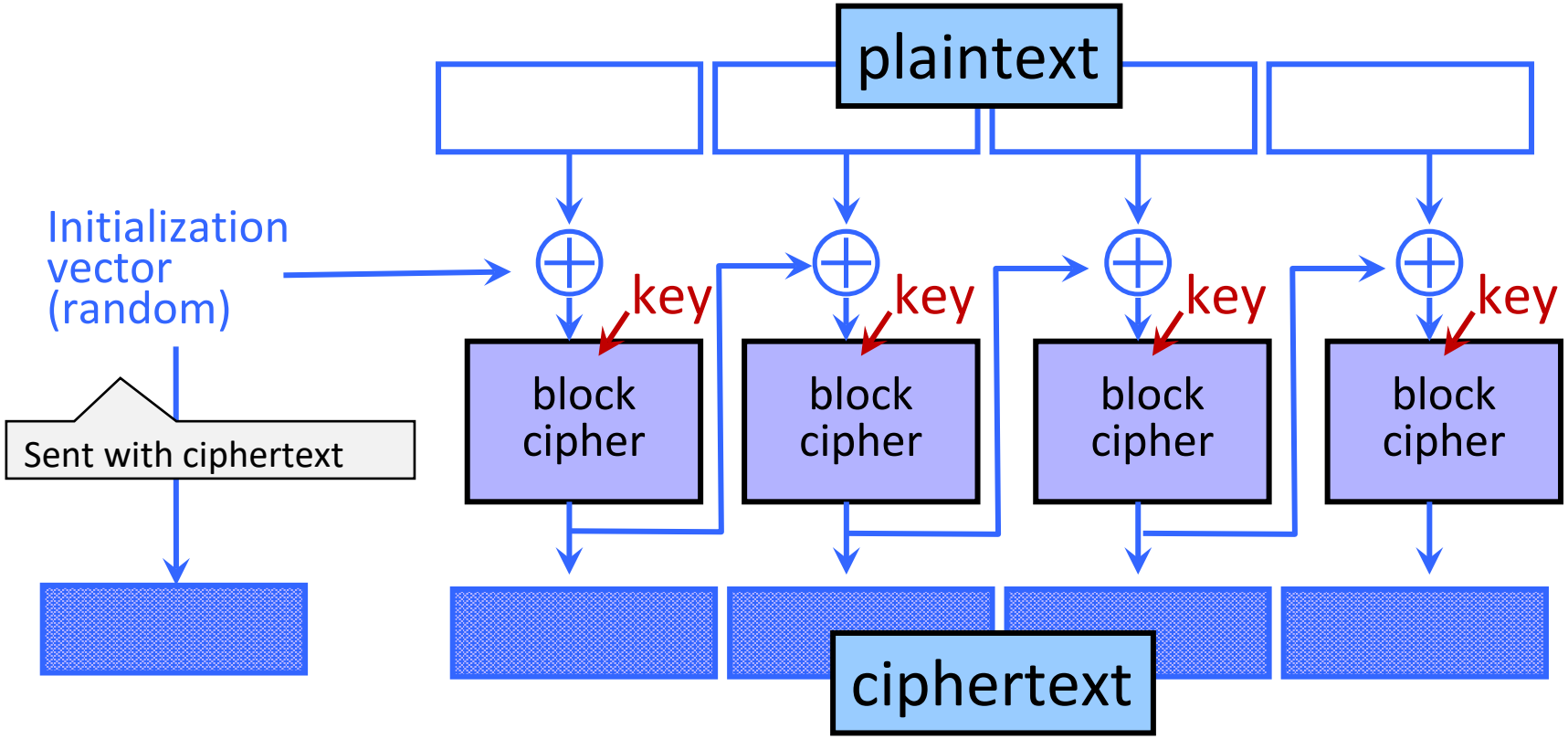
Now: Achieving Integrity

Message authentication schemes: A tool for protecting integrity.



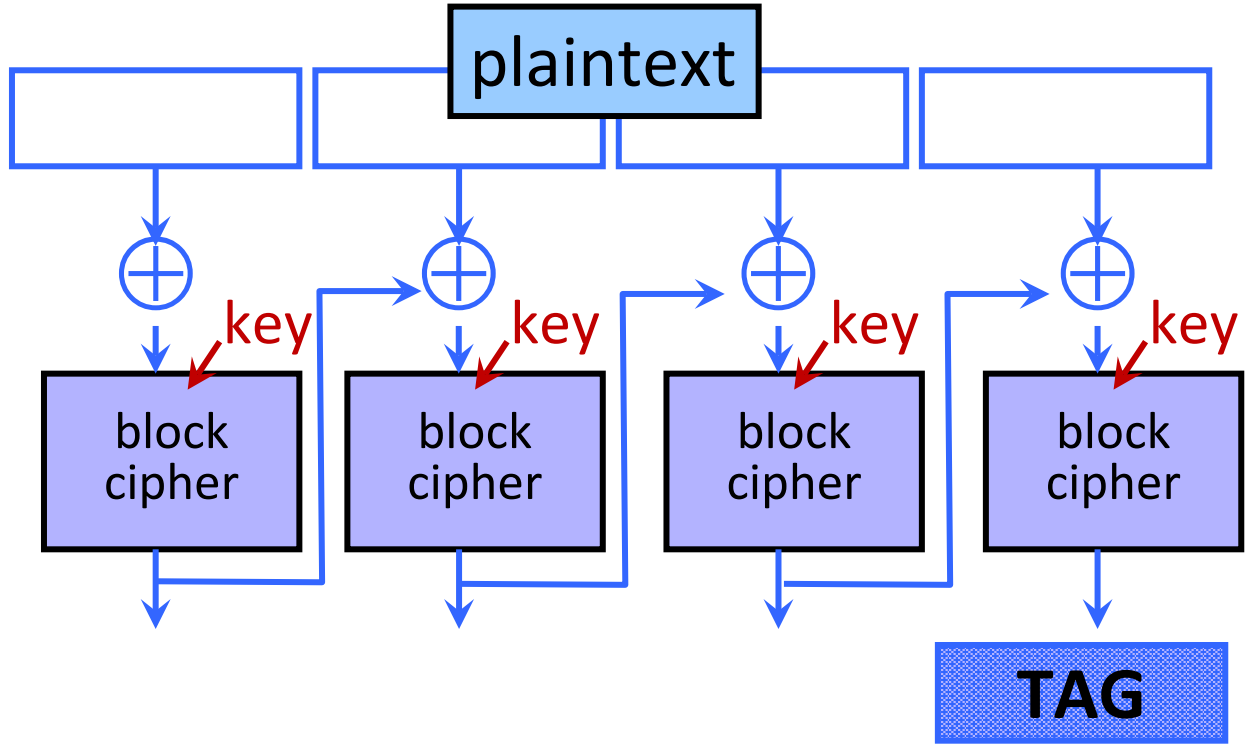
Integrity and authentication: only someone who knows KEY can compute correct MAC for a given message.

Reminder: CBC Mode Encryption



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

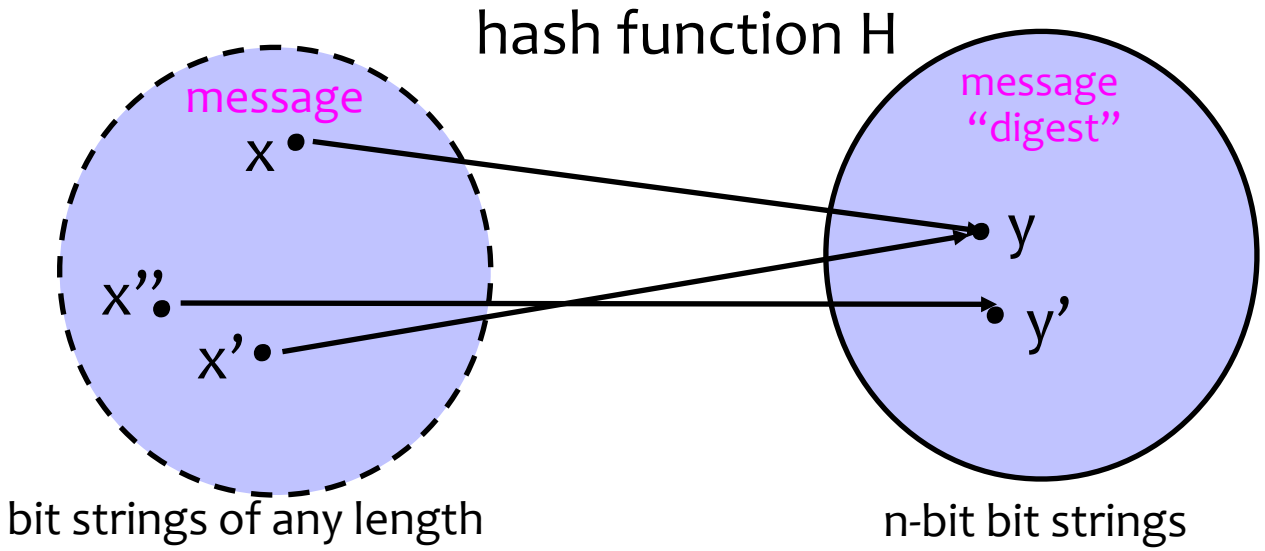
CBC-MAC



- Not secure when system may MAC messages of different lengths *(more in section!)*.
- Use a different key – not encryption key
- NIST recommends a derivative called CMAC [FYI only]

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 1/8 of this class 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

One-wayness does **not** imply collision resistance.

Collision resistance does **not** imply one-wayness.

One can prove this by constructing a function that has one property but not the other.

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 hash(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?

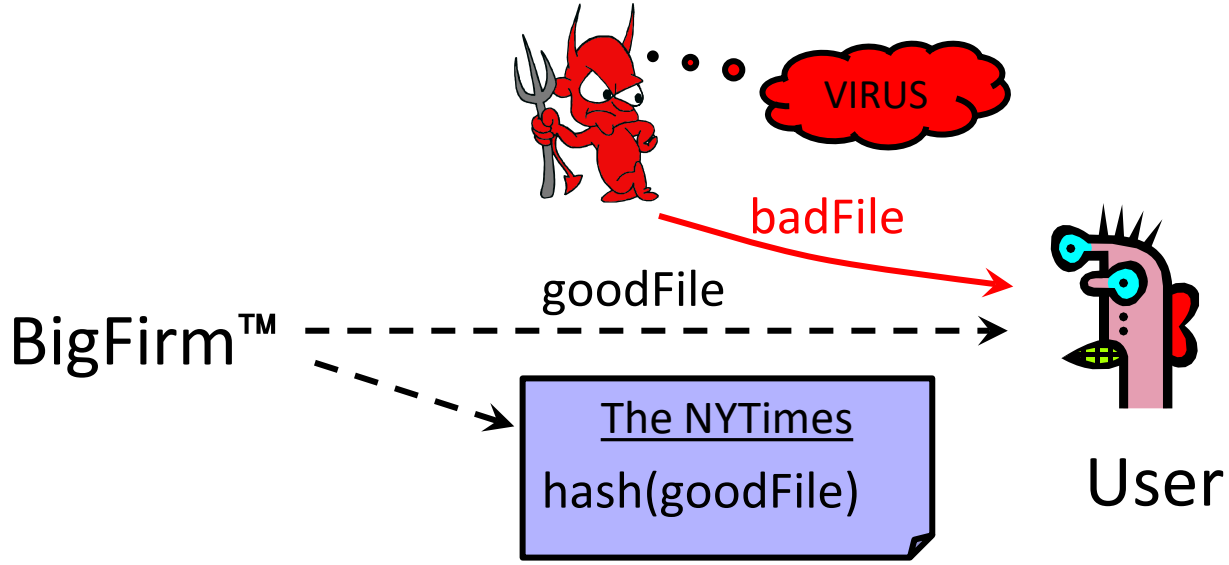
- System does not store actual passwords
- Don't need to worry about where to store the key
- Cannot go from hash to password

Application: Password Hashing

- Which property do we need?
  - One-wayness?
  - (At least weak) Collision resistance?
  - Both?

- This is not the whole story on password storage; we'll return to this later in the course.

Application: Software Integrity



Goal: Software manufacturer wants to ensure file is received by users without modification.

Idea: given goodFile and hash(goodFile), very hard to find badFile such that hash(goodFile)=hash(badFile)

Application: Software Integrity

- Which property do we need?
  - One-wayness?
  - (At least weak) Collision resistance?
  - Both?

Which Property Do We Need?

One-wayness, Collision Resistance, Weak CR?

- UNIX passwords stored as hash(password)
  - **One-wayness:** hard to recover the/a valid password
- Integrity of software distribution
  - **Weak collision resistance**
  - But software images are not really random… may need **full collision resistance** if considering malicious developers

Common Hash Functions

- **SHA-2: SHA-256, SHA-512, SHA-224, SHA-384**
- **SHA-3: standard released by NIST in August 2015**
- MD5 – Don't use for security!
  - 128-bit output
  - Designed by Ron Rivest, used very widely
  - Collision-resistance broken (summer of 2004)
- SHA-1 (Secure Hash Algorithm) – Don't use for security!
  - 160-bit output
  - US government (NIST) standard as of 1993-95
  - Theoretically broken 2005; practical attack 2017!

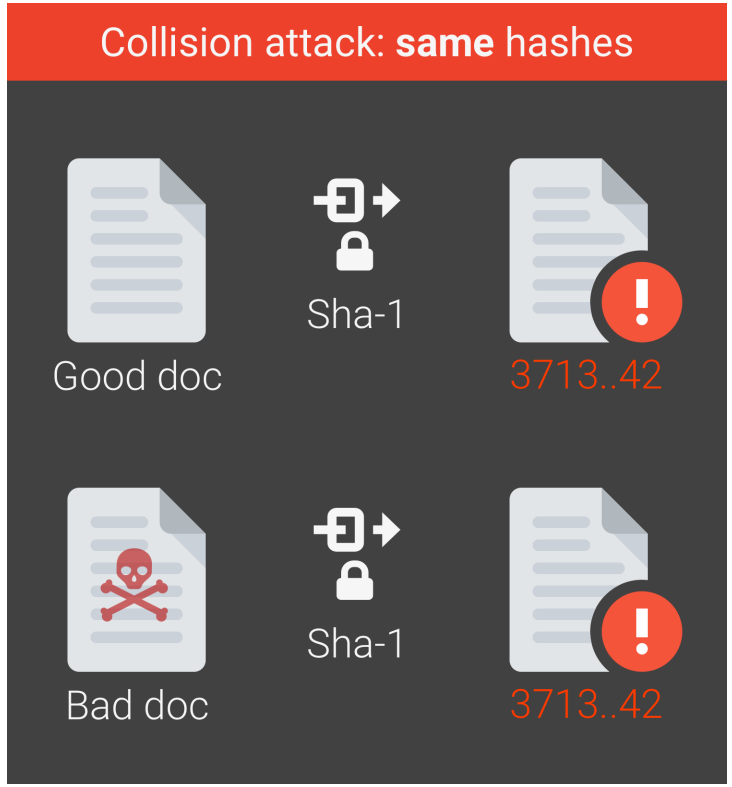
SHA-1 Broken in Practice (2017)

Google just cracked one of the building blocks of web encryption (but don't worry)

It's all over for SHA-1

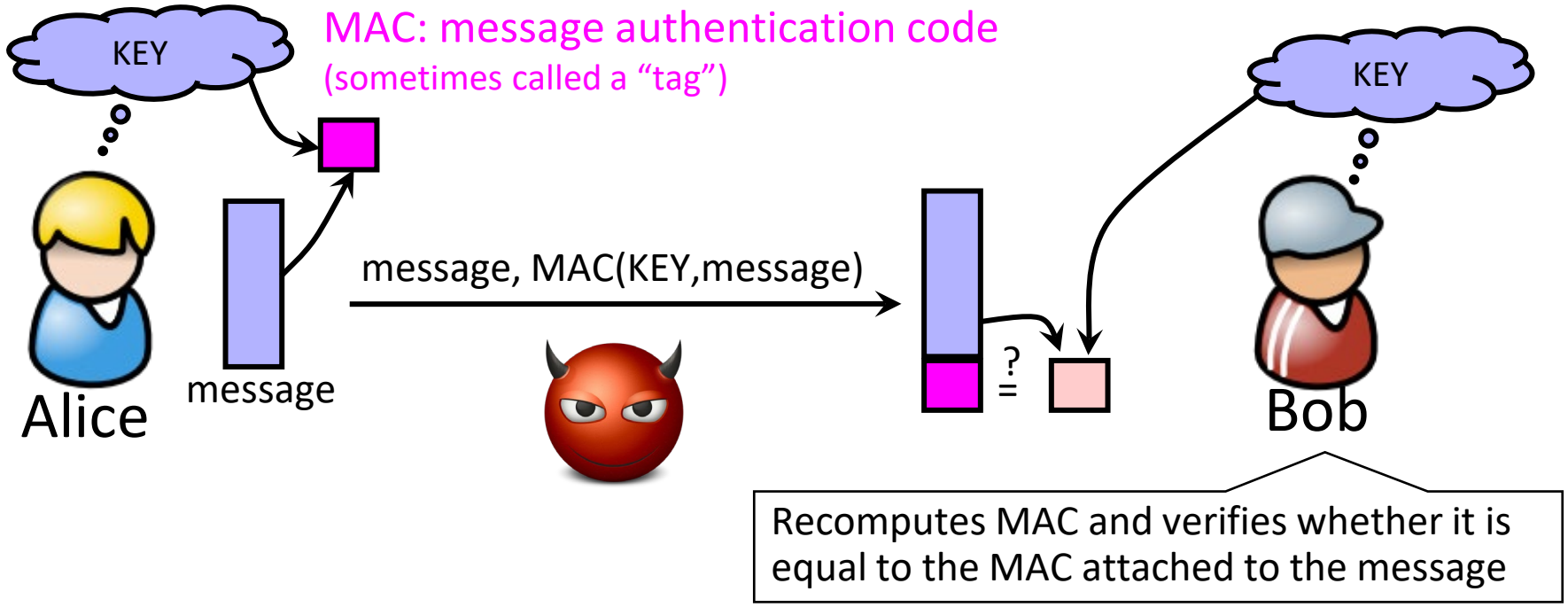
by Russell Brandom | @russellbrandom | Feb 23, 2017, 11:49am EST

https://shattered.io



Recall: 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.

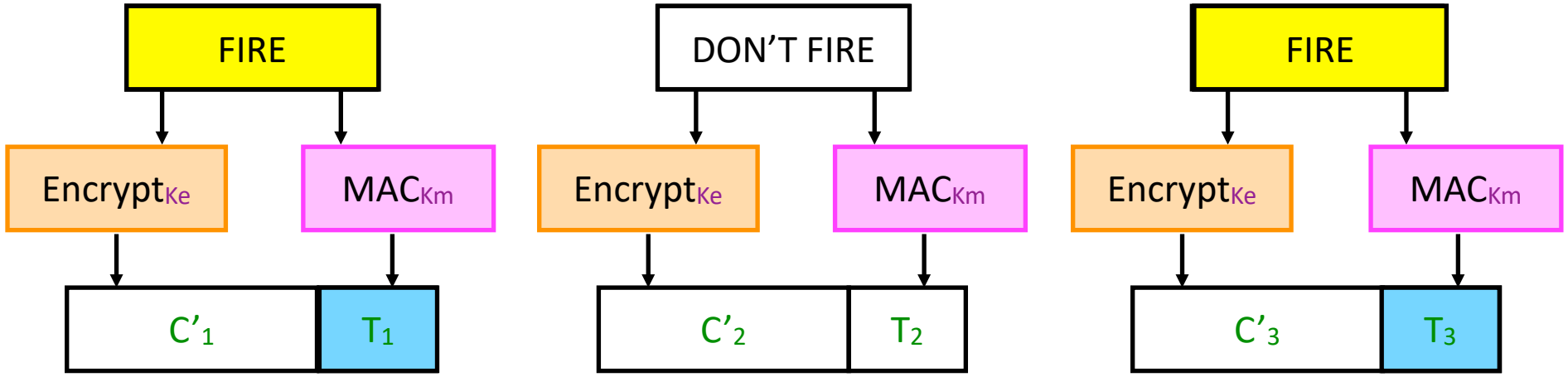
MAC with SHA3

- SHA3(Key || Message)
- Nice and simple ☺
- Previous hash functions couldn't quite be used in this way (see: length extension attack)
  - HMAC construction (FYI)

- Why not encryption? (Historical reasons)
  - Hashing is faster than block ciphers in software
  - Can easily replace one hash function with another
  - There used to be US export restrictions on encryption

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

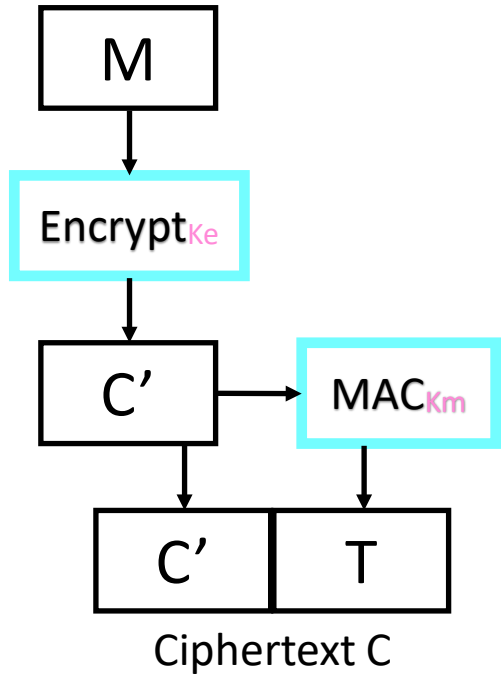


Authenticated Encryption

- Instead:

  Encrypt *then* MAC.

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



Encrypt-then-MAC