ECE 547/647: Security Engineering

Meeting 3

Crypto Foundations of Security

Rationale

- Security engineering has mathematical foundations that need to be understood even if working at higher levels.
- This math applies directly to Random number generation, Block encryption, Secure hash functions, and their use in Protocols.

Objectives

- Learn the terminology of random functions
- Learn how to generate random numbers and the various vulnerabilities.
- Learn how random functions apply to Passwords and Protocols

Prior Learning

- Let's Review from Lesson 2: Motivations
  - Security "crash course"
  - Common fallacies of security
  - Enemies of Security: "6 C's"
  - Measuring security: FIPS 140-2 Standards, NIST RMF,…
  - Assignment 1: Apply to SSD paper

Pre-Work

- Do you know the following security-related acronyms?

AES – Advanced Encryption Standard
MAC – Message Authentication Code
RFID – Radio Frequency Identification
SHA – Secure Hash Algorithm
SSL – Secure Socket Layer
PKI – Public Key Infrastructure
PRNG/TRNG – Pseudo/True Random Number Generator

- If you have no crypto background, try this optional reading in textbook, Chap 5.1-5.6, pp. 129-169

Concepts of Modern Cryptography

- **Fixed input-length random function** $R : \{0, 1\}^m \rightarrow \{0, 1\}^n$
  - Assign a random n-bit string to each of the 2^m possible inputs
  - Ex. Flip an unbiased coin $n \bullet 2^m$ times to fill a value table, a random function is just a randomly selected function out of $2^{n \bullet 2^m}$ possible m-bit to n-bit functions

- **Variable input-length random function** $R : \{0, 1\}^* \rightarrow \{0, 1\}^n$
  - Implement by a hypothetical device with memory and random-bit generator
  - Outputs random n-bit words for new inputs
  - Repeats answers for known inputs

Concepts of Modern Cryptography (cont'd)

- **Pseudo-random function**
  - Deterministic and efficiently computable function
  - Cannot be distinguished by any form of practical statistic or cryptoanalytic test from a random function (unless of course its definition is considered)
  - Ex. Changing a single bit of an input should on average change half of all output bits, etc.

- Equivalent definitions apply for **random permutations** and **pseudo-random permutations**

Random Bit Generation

- In order to generate the keys and nonces needed in cryptographic protocols, a source of random bits unpredictable for any adversary is needed. The highly deterministic nature of computing environments makes finding secure seed values for random bit generation a non-trivial and often neglected problems.

- Attack example: Netscape 1.1 Web browser in 1995
  - 2 PhD students broke SSL encryption using weakness in its session key generator
  - It used a random-bit generator that was seeded from only the time of day in microseconds and two process IDs
  - Resulting conditional entropy for an eavesdropper was small enough to enable a successful brute-force search

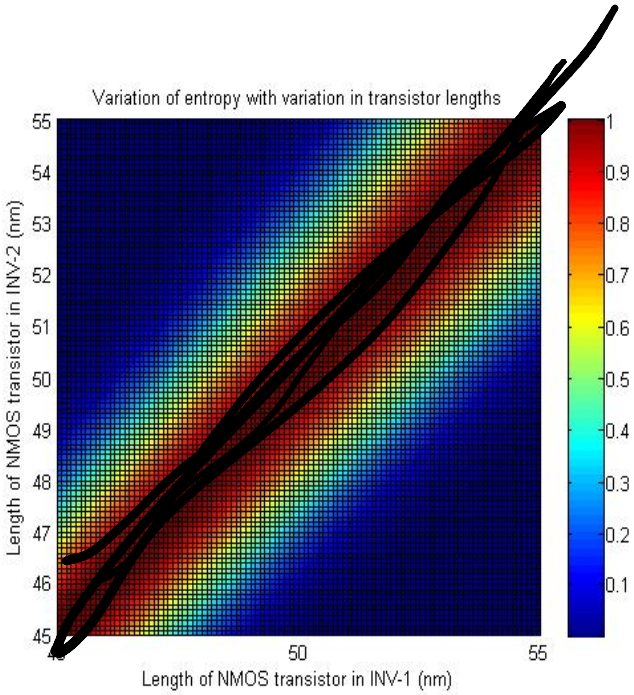
Random Bit Generation (cont'd)

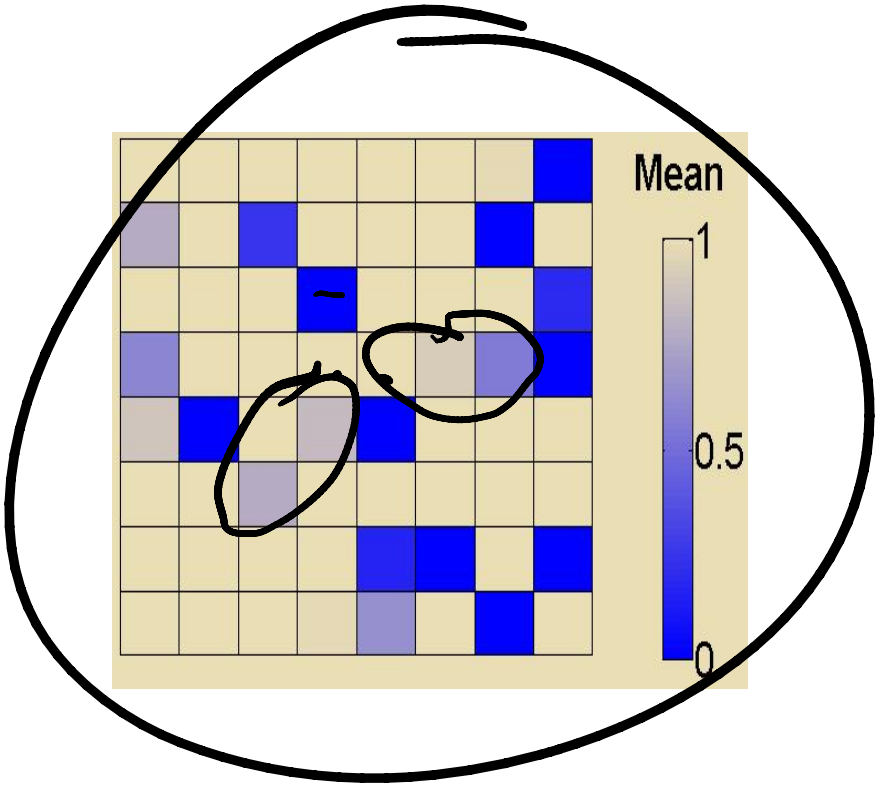
- Examples for sources of randomness:
  - Dedicated hardware random bit generators (amplified thermal noise from reverse-biased diode, unstable oscillators, Geiger counters)
  - High-resolution timing of user behaviors (e.g., key strokes and mouse movement)
  - High-resolution timing of peripheral hardware response times (e.g., disk drives)
  - Noise from analog/digital converters (e.g., sound card, camera)
  - Network packet timing and content
  - High-resolution time
- None of these random sources alone provides high-quality statistically unbiased random bits, but such signal can be fed into a hash function to condense their accumulated entropy into a smaller number of good random bits

Random Bit Generation (cont'd)

- The provision of a secure source of random bits is now commonly recognized to be an essential operating system service.
  - For example, the Linux **`/dev/random`** device driver uses a 4096-bit large **entropy pool** that is continuously hashed with keyboard scan codes, mouse data, inter-interrupt times, and mass storage request completion times in order to form the next entropy pool.
  - Users can provide additional entropy by writing into **`/dev/random`** and can read from this device driver the output of a cryptographical pseudo random bit stream generator seeded from this entropy pool.
  - Operating system boot and shutdown scripts preserve **`/dev/random`** entropy across reboots on the hard disk.

Representations of Randomness





ntropy – Shannon, 1948

- Optional extra reading:

- *https://people.eecs.berkeley.edu/~christos/classics/shannon-report.pdf*

Activity: Random Bit Generation

- Discuss 3 of the following as a source of random numbers. For each, think about 1) the statistics that are generated, and potential attacks in terms of either 2) forcing or 3) predicting the values

  - Coin toss
  - Low-order bits on a real-time clock
  - Low-order bits on a thermal sensor
  - Keyboard activity
  - Network response times
  - Power supply noise
  - Clock jitter
  - Power-up state of CMOS SRAM
  - Memristor delay time…

Secure Hash Functions

- Efficient variable input-length pseudo-random functions that pass trivial statistical tests (e.g., uniform distribution of output values) are called **hash functions** and commonly used for fast table lookup.

Secure n-bit hash function $h: \{0, 1\}^* \rightarrow \{0, 1\}^n$

Preimage resistance (one-way):	Second preimage resistance (weak collision resistance):	Collision resistance:
For a given value y, it is computationally infeasible to find x with $h(x) = y$.	For a given value x, it is computationally infeasible to find x' with $h(x') = h(x)$	It is computationally infeasible to find a pair $x \neq y$ with $h(x) = h(y)$

Secure Hash Functions (cont'd)

- Commonly used secure hash functions are MD5 ($n = 128$) and SHA-1/SHA-256/SHA-512 ($n = 160/256/512$)
- Fast secure hash functions based on fixed input-length PRF called a **compression function**
- MD5 and SHA-1 for instance use block sizes of n= 512 bits

The input bitstring X is padded in an unambiguous way to a multiple of compression function's block size n

Pad carefully to avoid two strings which differ in number of trailing "zero" bits being indistinguishable. Use a 1 followed by some number of zeros to avoid this.

Padded bitstring X' is split into m n-bit blocks $X_1, \ldots, X_m$, and the hash value $H(X) = H_m$ is calculated via the recursion $H_i = C(H_{i-1}, X_i)$ where H_0 is a predefined start value and C is the compression functions

Birthday Paradox

- With 23 random people in a room, there is a 0.507 chance that two share a birthday. This perhaps surprising observations has important implications for the design of cryptographic systems

- If we randomly throw k balls into n bins, then the probability that no bin contains two balls is

$$\left(1-\frac{1}{n}\right)\bullet\left(1-\frac{2}{n}\right)\cdots\left(1-\frac{k-1}{n}\right)=\prod_{i=1}^{k-1}\left(1-\frac{i}{n}\right)=\frac{n!}{(n-k)!\ \bullet\ n^k}$$

Birthday Paradox (cont'd)

- It can be shown that this probability is less than $\frac{1}{2}$ if k is slightly above $\sqrt{n}$. As $n \rightarrow \infty$, the expected number of balls needed for a collision is $\sqrt{n\pi/2}$.

- One consequence is that is a 2^n search is considered computationally sufficiently infeasible, then the output of a collision-resistant hash function needs to be at least $2n$ bits large

Activity: Secure Hash Functions

- Collect birthdays for classroom and see if there are collisions. Note any interesting collisions with holidays, etc.
- If class is small, ask them to provide their parent's birthdays as well)
- Mention privacy issues associated with sharing birthday. Give the option to provide a false birthday, but must be chosen without knowledge of other birthdays…

Block Ciphers

- Encryption is typically performed today with block cipher algorithms
  - Key-dependent permutations, operating on bit-clock alphabets.
  - Typical alphabet sizes: 2^{64} or 2^{128}

$$E : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$$

- Confusion: Make relationship between key and ciphertext as complex as possible
- Diffusion: Remove statistical links between plaintext and ciphertext

Blockciphers (cont'd)

$$E : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$$

- Prevent adaptive chosen-plaintext attacks, including differential and linear cryptanalysis

- Product cipher: Iterate many rounds of a weak pseudo-random permutation to get a strong one

Feistel Structure

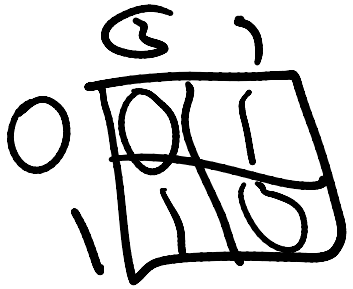
- Technique to build an (invertible) random permutation $E : P \leftrightarrow C$ using a (non-invertible) random function f as a building block.

- Invented by team that designed IBM's proposal for Data Encryption Standard (DES) algorithm in 1970s

- Basic Idea: Split the plaintext block P (e.g. 64 bits) into two equally-sized halved L and R (e.g. 32 bits each):

$$P = L_0 \parallel R_0$$

Feistel Structure (cont'd)



- Then non-invertible function f is applied in each round I alternatingly to one of these halves, and the result is XORed onto the other half respectively:

and for odd

and for even

- After rounds $1, \ldots, n$ have been applied, the two halves are concatenated to form the ciphertext block C:

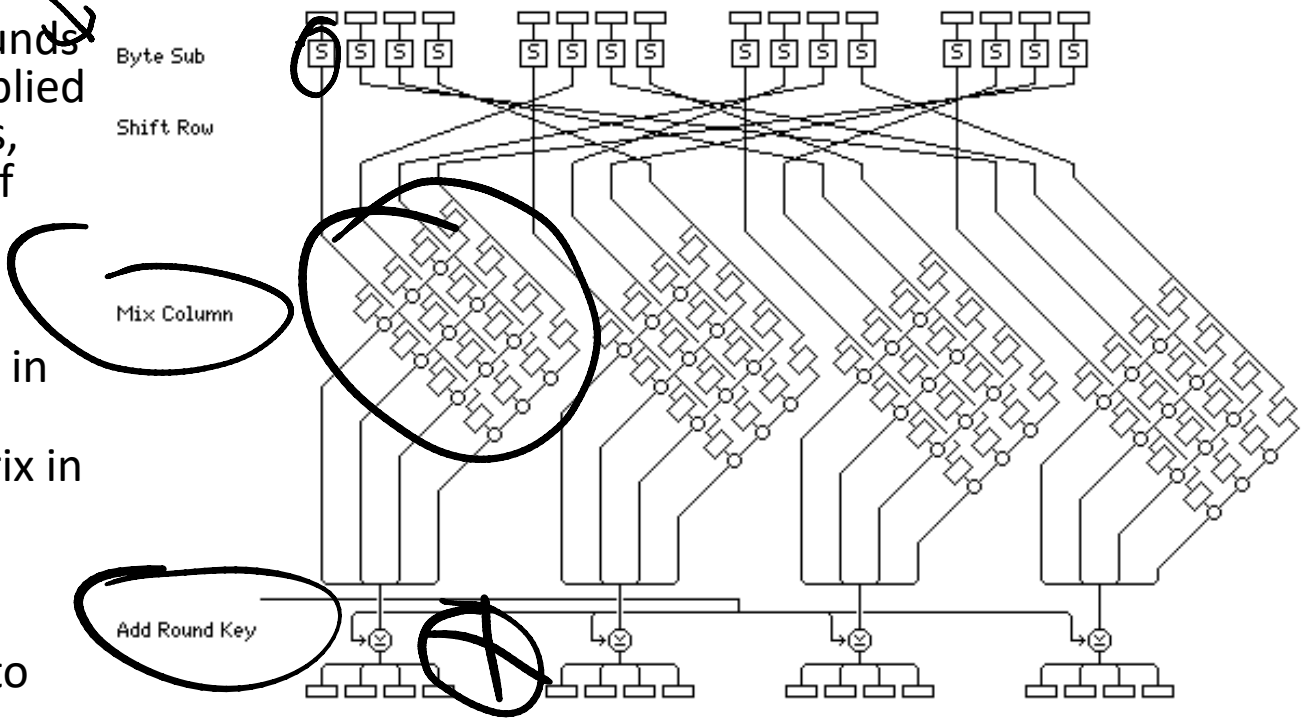
$$C = L_n \| R_n$$

Activity:

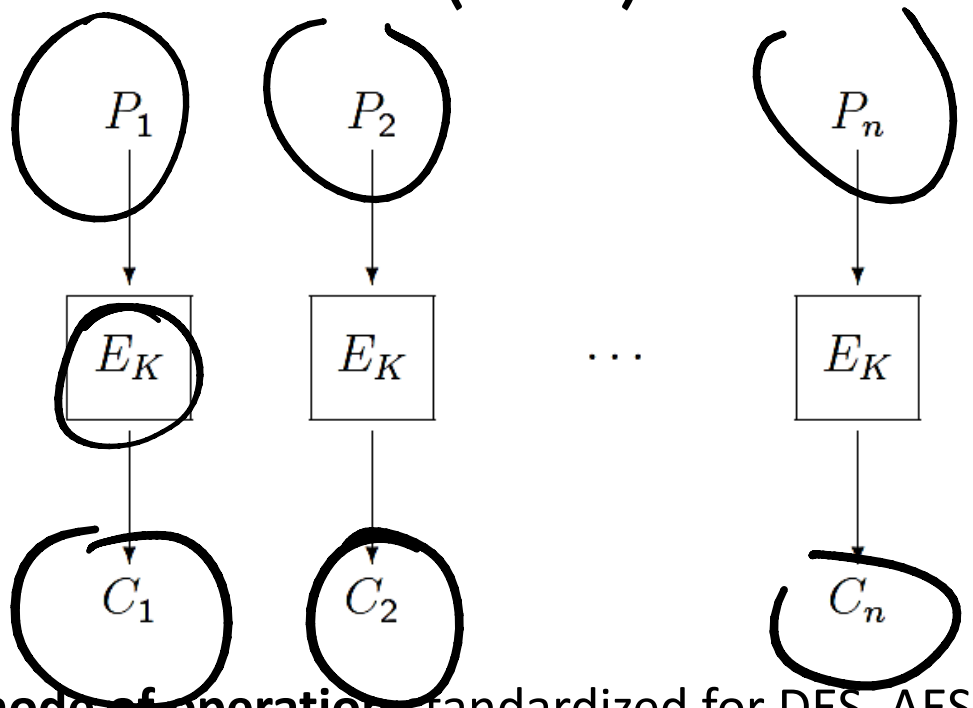
- Why is the XOR function (rather than AND or OR) so ubiquitous in cryptography?

Advanced Encryption Standard (AES)

- Each of the 9-13 Rijndael rounds starts with an 8-bit s-box applied to each of the 16 input bytes, followed by a permutation of the byte positions.
- Next follows the column mix in which each 4-byte vector is multiplied with a 4×4 matrix in $CG(2^8)$
- Finally, a subkey is XORed into the result



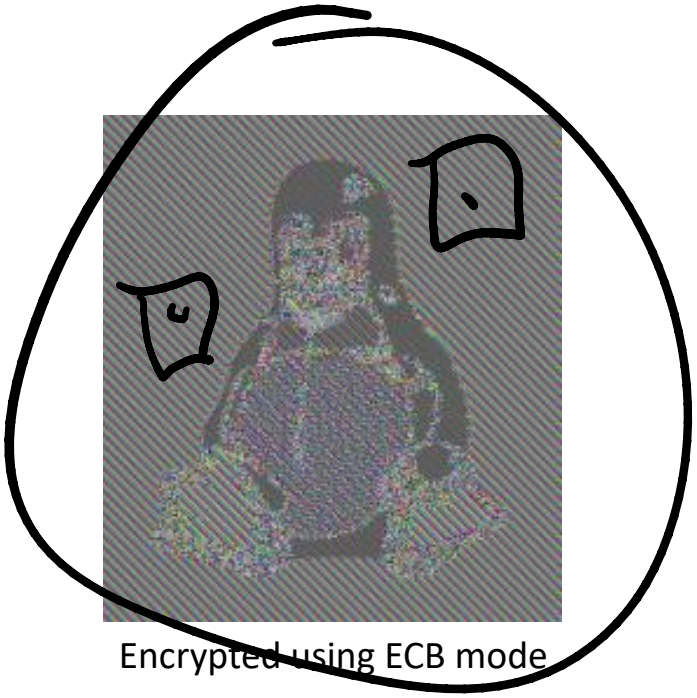
Electronic Code Book (ECB)



- In the simplest **mode of operation** standardized for DES, AES, etc., message is cut into n-bit blocks, where n is the block size of the cipher, and then the cipher function is applied to each block individually

Problems with ECB

Original image



Encrypted using ECB mode

Insecure encryption of an image as a result of electronic codebook mode encoding.

Problems with ECB (cont'd)

- The Electronic Code Book (ECB) mode is generally not recommended for use because:
  - Repeated plaintext messages can be recognized by the eavesdropper as repeated ciphertext.
  - Plaintext block values are often not uniformly distributed

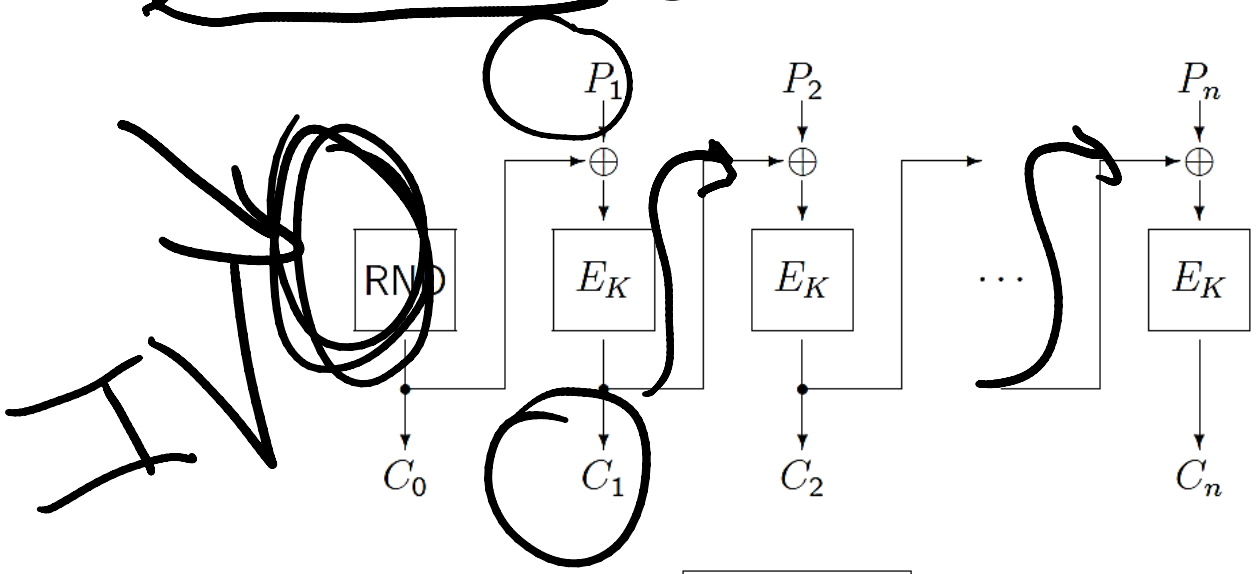
- Both problems can be solved by using other modes of operation ECB

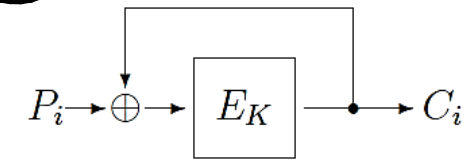
Problems with ECB (cont'd)

- Using a pseudo-random value as the input of the block cipher will:
  - Use its entire alphabet uniformly
  - Independent of the plaintext content, a repetition of cipher input has to be expected only after $\sqrt{2^n} = 2^{\frac{n}{2}}$ blocks have been encrypting with the same key ($\rightarrow$ birthday paradox)

- The Cipher Block Chaining mode XORs the previous ciphertext into the plaintext to achieve this
- Entire ciphertext is randomized by prefixing it with a random initial vector

Cipher Block Chaining (CBC)

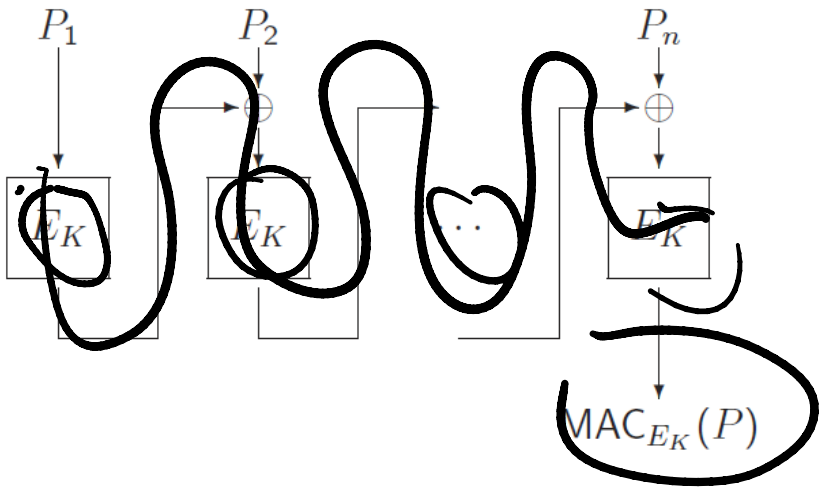




$$C_i = E_K(P_i \oplus C_{i-1})$$

Message Authentication Code (MAC)

- A modification of CBC provides integrity protection for data
  - Initial vector set to fixed value (e.g., 0), and C_n of the CBC calculation is attached to transmitted plaintext
  - Anyone who shares the secret key K with sender can recalculate the MAC over the received message and compare the result
- A MAC is the cryptographically secure equivalent of a ~~checksum~~, which only those who know K can generate and verify



Over the weekend

- Assignment 1 (on Moodle)
- Visit www.autosec.org and read:
- Comprehensive Experimental Analyses of Automotive Attack Surfaces

  Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, Stefan Savage, Karl Koscher, Alexei Czeskis, Franziska Roesner, Tadayoshi Kohno.
  USENIX Security, August 10–12, 2011.

  The full paper is available in PDF form here: PDF. (FAQ)