CY 2550 Foundations of Cybersecurity

Cryptography Part 5

February 3

Alina Oprea

Associate Professor, Khoury College

Northeastern University

Outline

- Modes of operation for encryption (CTR mode)
- Hash functions
- MACs for integrity
- Digital signatures

- Announcements
  - CIO of Children's Hospital in Boston – Dan Nigrin – will be on campus to give a talk on **Feb 5 from 11:45-12:45 in 655 ISEC**
  - Distinguished Lecture by Laurel Riek, UCSD, **on Feb 7 in ISEC Auditorium. 11:45am-1:00pm**. "Human Robot Teaming in Healthcare "

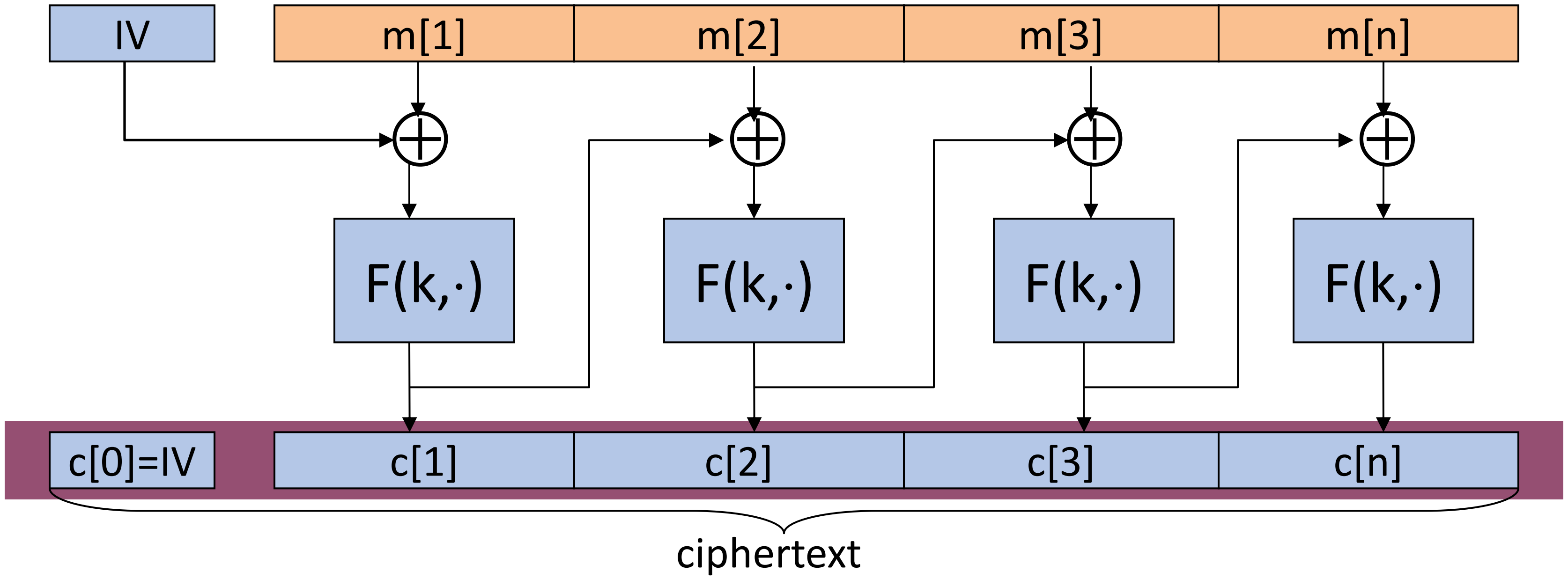
Recap

- Modes of operation for longer messages (CBC encryption)
- Public-key cryptography
- Key exchange
  - Diffie-Hellman protocol based on difficulty of computing discrete logs modulo a large prime
  - Not secure against active attackers
- RSA public-key encryption
  - Use public key to encrypt message
  - Use secret key to decrypt
  - Difficulty of factoring numbers that are products of two large primes

CBC encryption

Let F be a secure block cipher (e.g., ENC-AES)

$Enc_{CBC}(k,m)$:   choose **random** IV$\in \{0,1\}^n$ and do:



An example CBC analysis

q = # messages encrypted with k

L = length of message (in blocks)

Suppose we want **Pr[Attacker wins CPA game]** $\leq 1/2 + 1/2^{32}$

$q^2 L^2 / 2^n < 1/ 2^{32}$

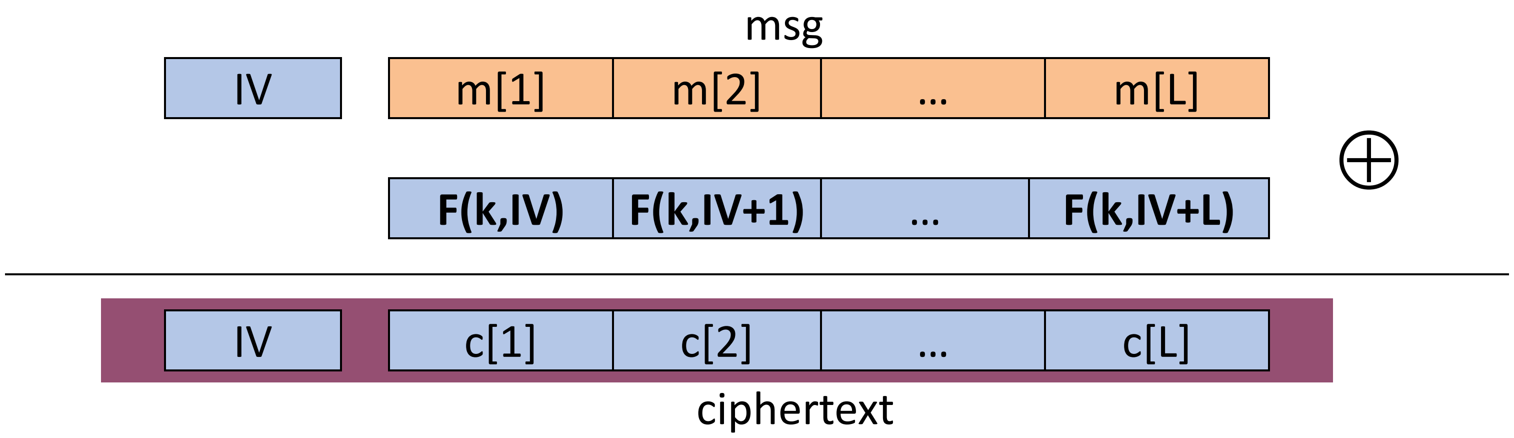
- AES: $2^n = 2^{128} \Rightarrow q L < 2^{48}$

  So, after 2^{48} AES blocks, must change key

CTR-mode encryption

Let F be a secure block cipher (e.g., ENC-AES)

Enc(k,m): choose a random IV and do:



note: parallelizable (unlike CBC)

$$c_i = F_k(IV + i) \oplus m_i$$

Comparison of CBC and CTR Mode

- Both are IND-CPA secure assuming
  - Block cipher itself is secure (pseudorandom permutation)
  - IV is truly random with size of block cipher
  - Use the key for limited number of encryptions (key needs to be changed afterwards)
- CTR mode has better security bounds
- CTR mode is parallelizable, while CBC is sequential

Public-Key Cryptography

- Public-Key Encryption
  - Examples: RSA, ElGamal
- Digital Signatures:
  - Authenticate messages
  - Examples: RSA, DSA
- Key Exchange
  - Protocols to establish a secret key between two parties
  - Examples: Diffie-Helman key exchange
- Intuition for all these
  - Computation in one direction is “easy”, but “hard” in the reverse
  - Hardness assumptions imply that adversary cannot reverse computation

The Diffie-Hellman protocol

Fix a large prime p (e.g. 600 digits)

Fix an integer g in $\{1, \ldots, p\}$

Alice

choose random **a** in $\{1,\ldots,p-1\}$

Bob

choose random **b** in $\{1,\ldots,p-1\}$

$$p, g, A \leftarrow g^a \bmod p \quad \longrightarrow$$

$$\longleftarrow \quad B \leftarrow g^b \bmod p$$

$$B^a \pmod p = (g^b)^a = k_{AB} = g^{ab} \pmod p = (g^a)^b = A^b \pmod p$$

RSA Algorithm

- Security is based on the difficulty of factoring the product of primes
  - Alice chooses two secret primes p and q, $n = pq$, $\phi(n) = (p-1)(q-1)$
  - Choose e such that $1 < e < \phi(n)$, and $gcd(e, \phi(n)) = 1$
  - <n, e> is Alice's public key
  - Private key $d = e^{-1} \bmod \phi(n)$; $d \cdot e = 1 \bmod \phi(n)$
- Encryption and decryption
  - Given a message M, $0 < M < n$
  - Compute ciphertext $C = M^e \bmod n$
  - To decipher, compute $C^d \bmod n = (M^e \bmod n)^d \bmod n = M^{ed} \bmod n = M$
  - Use Euler's theorem: $x^{\phi(n)} = 1 \bmod n$

IND-CPA security for Public-Key Encryption

- Black: IND-EAV; Red: IND-CPA
- In CPA Adv can encrypt messages of its choice

Charlie

Adv

Round 1: Charlie chooses k and encryption algo — Charlie: *pk, sk*; Adv: *pk*

Round 2: Adv can encrypt messages — Query: Encrypt m (Adv → Charlie); Reply: Ciphertext c (Charlie → Adv)

Round 3: Adv chooses two plaintext messages — $m_0, m_1 \in \boldsymbol{M}$ (Adv → Charlie)

Round 4: Charlie chooses a random binary number $b \leftarrow_R \{0, 1\}$

Round 5: Charlie encrypts the corresponding message — $c = Enc_{pk}(m_b)$ (Charlie → Adv)

Round 6: Adv can encrypt messages — Query: Encrypt m (Adv → Charlie); Reply: Ciphertext c (Charlie → Adv)

Round 7: Adv guesses the value of *b* — $b' \in \{0, 1\}$

Adversary wins if $b = b'$

IND-CPA security for Public-Key Encryption

- In public-key encryption, everyone knows the public key
- That means everyone (including the adversary) can encrypt any message
- IND-CPA and IND-EAV are equivalent notions of security!
- Another reason why we demand IND-CPA at a minimum for symmetric-key encryption

Plain RSA Encryption

Plain (textbook) RSA encryption:

- public key: <n, e>
- secret key: < p, q, d >

Encrypt: $c \longleftarrow M^e \bmod n$

Decrypt: $c^d \longrightarrow M \bmod n$

Insecure cryptosystem !! !

- Is not IND-CPA secure and many attacks exist
- Deterministic (public key) encryption is never IND-CPA secure

Attacks Against RSA

- The length of *n=pq* reflects the strength
  - 700-bit *n* factored in 2007
  - 768 bit factored in 2009
- 1024 bit for minimal level of security today
  - Likely to be breakable in near future
  - Recommended use of 2048 or 4096 bits
- RSA encryption/decryption speed is quadratic in key length

Computationally Hard Problems

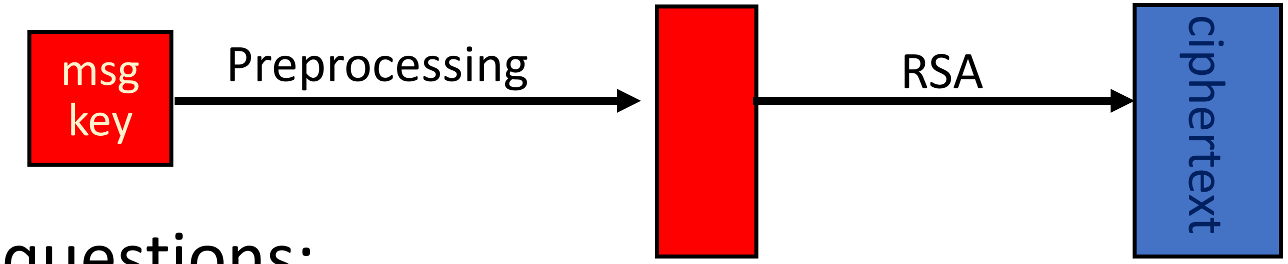
- RSA problem:
  - Given public RSA key, decrypt $m^e \; mod \; n$ for a random message *m*
- RSA assumption:
  - Solving the RSA problem is difficult

- Factoring assumption
  - If *n=pq* with *p* and *q* primes, cannot factor for large *n*
  - If factoring can be done in polynomial time, then RSA problem can be solved in polynomial time

RSA encryption in practice

Never use plain RSA.

RSA in practice

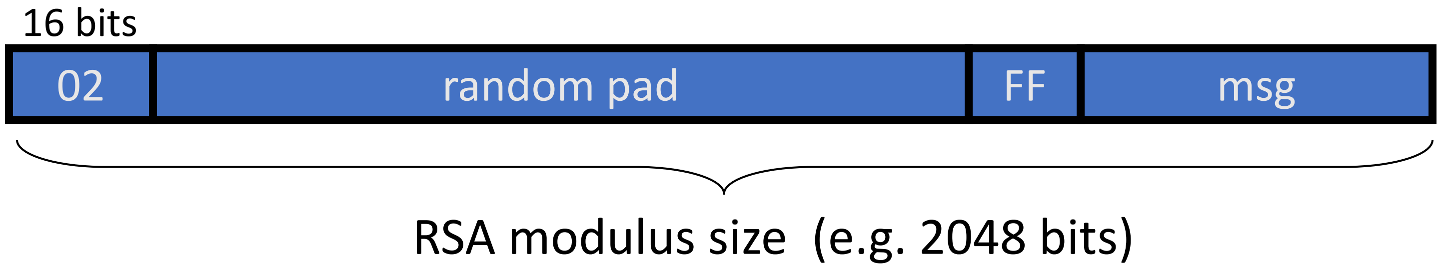


Main questions:

- How should the preprocessing be done?
- Can we argue about security of resulting system?
- How can we randomize it to be IND-CPA secure?

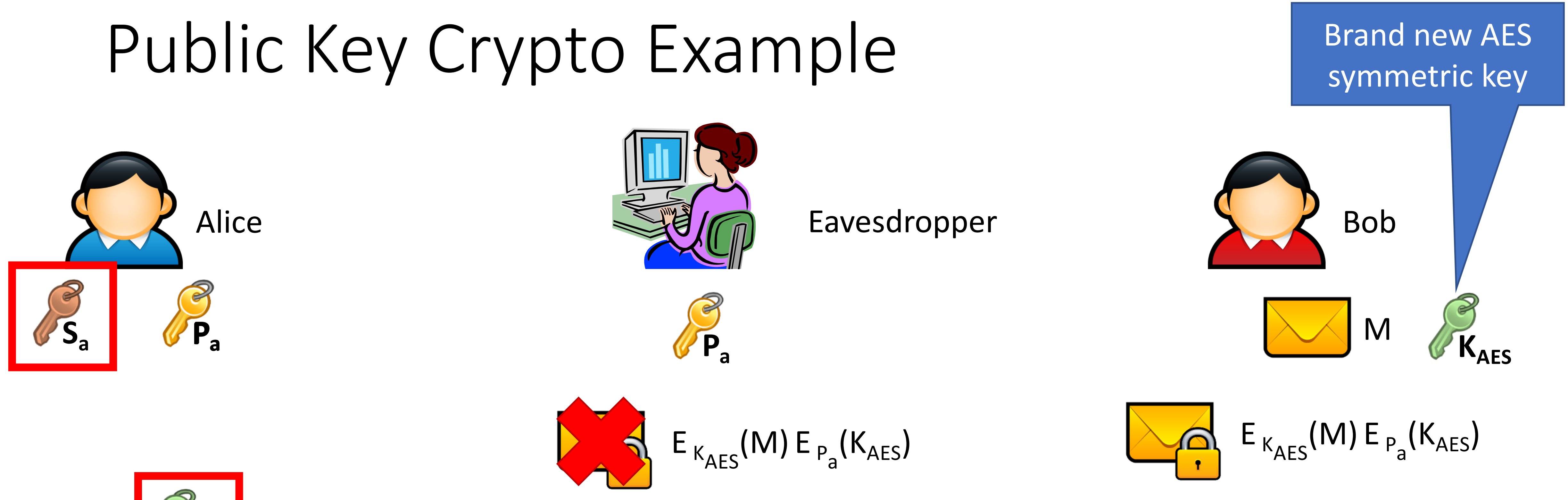
PKCS1 v1.5

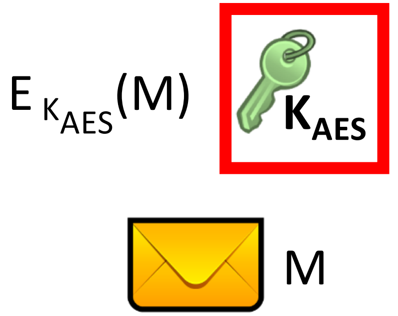
PKCS1 mode 2: (encryption)



- Add random pad before the message
- Resulting value is RSA encrypted
- Widely deployed, e.g. in HTTPS, but it is not IND-CPA secure!
- There are newer versions that are secure (e.g., OAEP)

Public Key Crypto Example





- Why bother with the symmetric key?
  - Why not just encrypt *M* with P_a?
- Performance
  - Asymmetric crypto is slow, symmetric is fast
  - Use asymmetric for K (which is small)
  - Use symmetric for M (which is large)

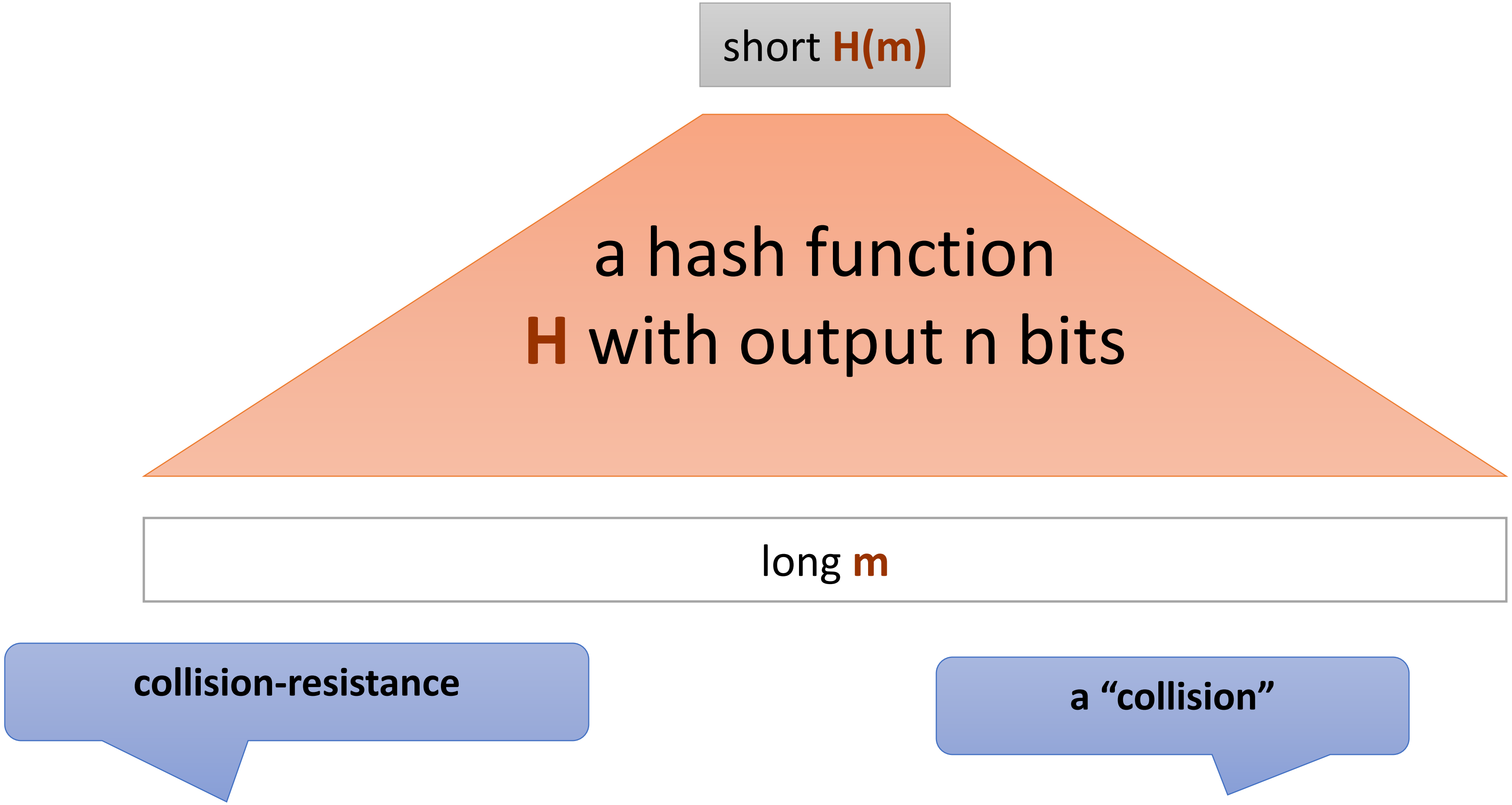
Key sharing can be done with a Key Exchange protocol (e.g., Diffie-Hellman)

Hash Functions and Authentication

Cryptographic Hash Functions

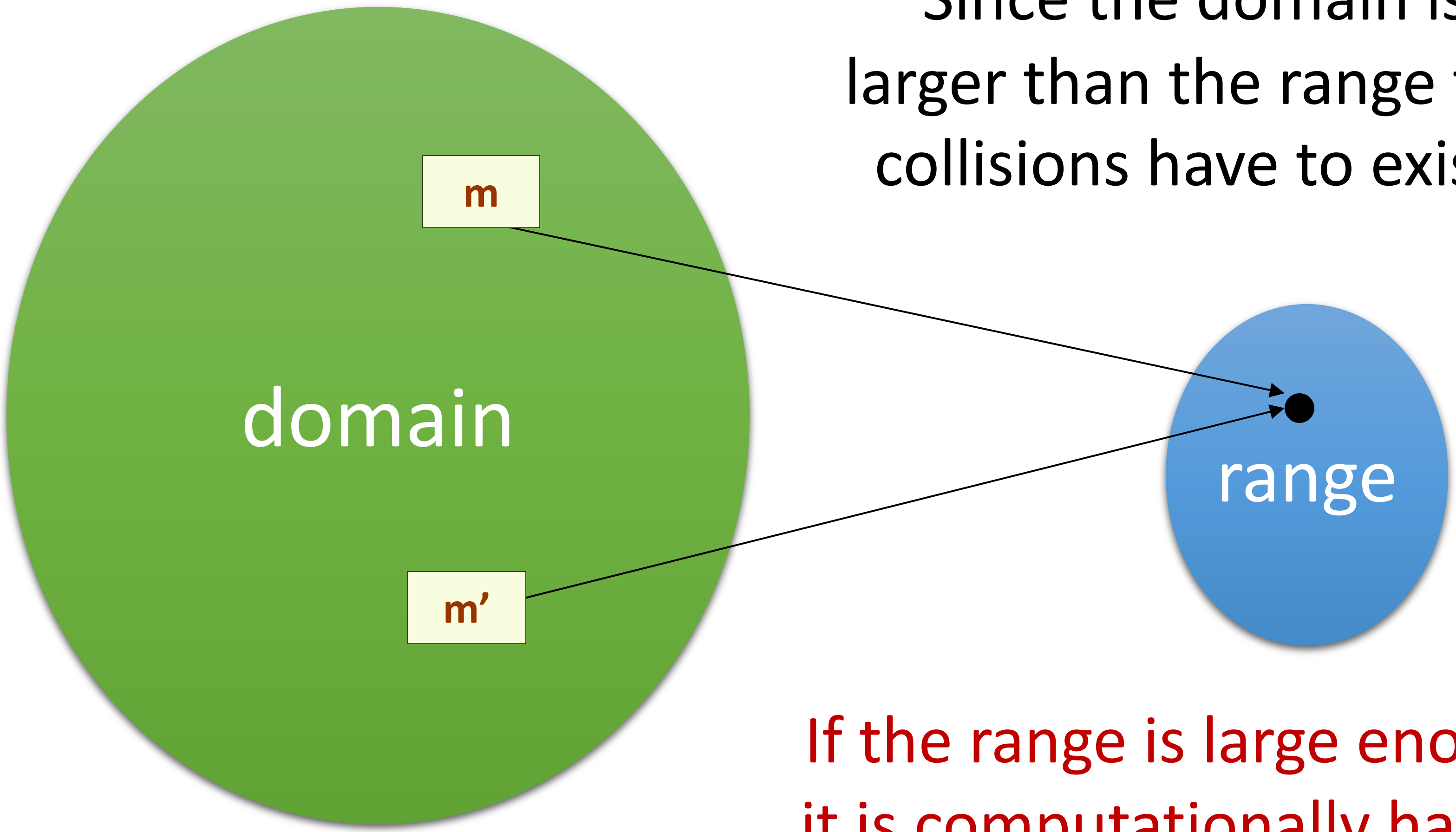
- Cryptographic hash function transform input data into scrambled output data
  - Arbitrary length input → fixed length output
  - Deterministic: H(A) is always the same
  - High entropy:
    - md5('security') = e91e6348157868de9dd8b25c81aebfb9
    - md5('security1') = 8632c375e9eba096df51844a5a43ae93
    - md5('Security') = 2fae32629d4ef4fc6341f1751b405e45
  - Collision resistant
    - Locating A' such that H(A) = H(A') takes a long time
    - Example: 2^{21} tries for md5

Collision-resistant hash functions



Requirement: it should be hard to find a pair **(m,m’)** such that **H(m) =H(m’)**

Collisions always exist



Since the domain is larger than the range the collisions have to exist.

If the range is large enough, it is computationally hard to find collisions.

Examples

Are these hash functions collision resistant?

- $H:\{0,1\}^{2n} \to \{0,1\}^n$
  - $H(x||y) = x \text{ XOR } y$
- $H:\{0,1\}^{2n} \to \{0,1\}^n$
  - Let p be an n-bit prime
  - $H(x||y) = x + y \bmod p$
- $H: N \to \{0,1\}^n$
  - Let p be an n-bit prime
  - $H(x) = ax + b \bmod p$, p prime

History of hash functions

H is a **collision-resistant hash function** if it is “*practically impossible to find collisions in **H***”.

- **1991**: MD5
- **1995**: SHA1
- **2001**: SHA2 -- SHA-256 and SHA-512
- **2004**: Team of Chinese researchers found collisions in MD5
- **2007**: NIST competition for new SHA3 standard
- **2012**: Winner of SHA3 is Keccak

Well Known Hash Functions

- MD5
  - Outputs 128 bits
  - Collision resistance totally broken in 2004
- SHA1
  - Outputs 160 bits
  - Partially broken: method exists to find collisions in 2^{80} tries
  - Deprecated
- SHA2 family (SHA-224, SHA-256, SHA-384, SHA-512)
  - SHA-224 matches the 112 bit key length of 3DES
  - SHA-256, SHA-384, SHA-512 match the key lengths of AES (128, 192, 256 bits)
  - Considered safe

The Future: SHA3

- 2007: NIST opens competition for new hash functions
- 2008: Submission deadline, 64 entries, 51 make the cut
- 2009: 14 candidates move to round 2
- 2010: 5 candidates move to round 3
- 2011: final round of public comments
- 2012: NIST selects *keccak* (pronounced “catch-ack”) as SHA3
  - Created by Guido Bertoni, Joan Daemen, Gilles Van Assche, Michaël Peeters

Birthday paradox

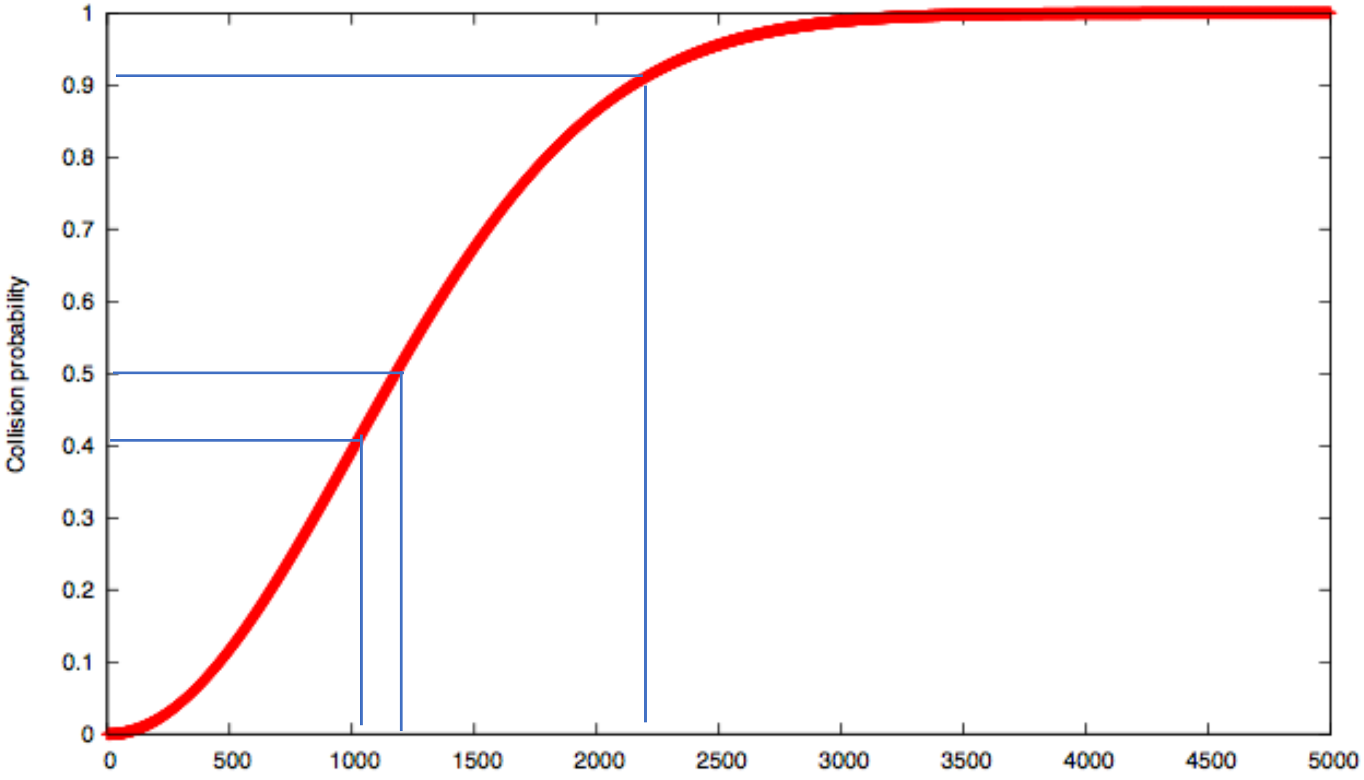
- If we choose q elements $y_1, \ldots y_q$ at random from {1,…,N}, what is the probability that there exists i and j such that $y_i = y_j$?

N=365
possible days

- What is the probability that two people have the same birthday?
- When is this probability higher than 0.5?

Collision probability

$N = 10^6$



samples n

- If $q = \Theta(\sqrt{N})$ items, then probability of collision is approx. ½
- Birthday paradox
  - N = 365, q = 23
- Hash functions
  - $N = 2^{256}, q = 2^{128}$
- Implies n/2 level of security for n-bit hash function in best case