CIS 4360
Secure Computer Systems

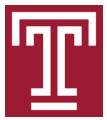
Asymmetric Cryptography

Professor Qiang Zeng

Spring 2017

Previous Class

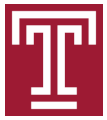
- Symmetric Encryption
  - Block Ciphers
    - DES (don't use it), 3-DES, AES
    - Modes of operation: ECB (don't use it), CBC, CFB
  - Stream Ciphers
    - RC4
- Message Authentication Code
  - HMAC
  - KMAC



Previous class…

When to use Stream Ciphers?

(1) streaming data: stream ciphers can encrypt data whenever bits are generated, while block ciphers have to wait until a whole block of data has been generated
(2) When performance is a main concern
(3) When the length of the data to be encrypted is unknown. With stream ciphers, you can encrypt what is currently known

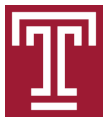


Previous class…

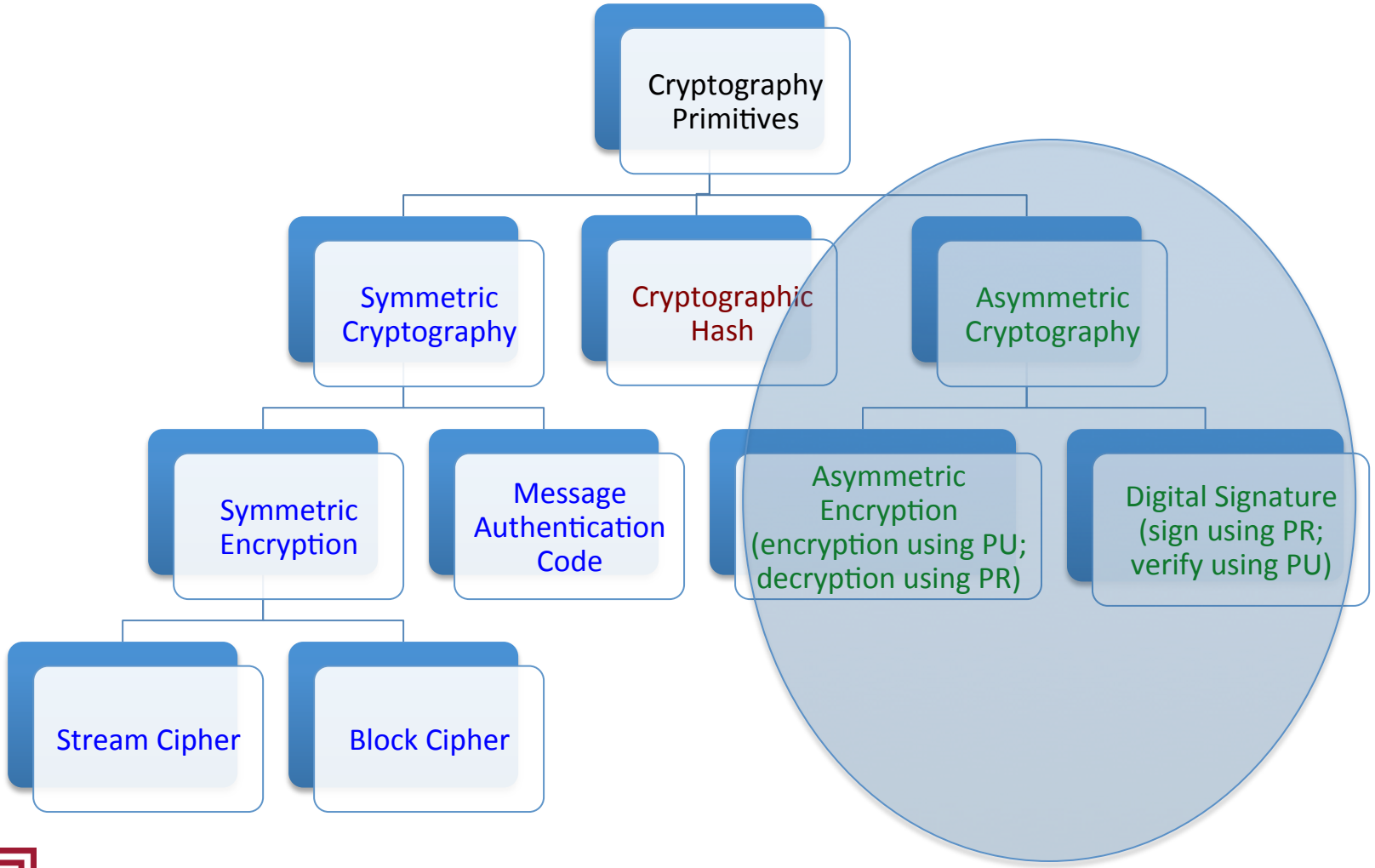
Can MAC be used to achieve non-repudiation?

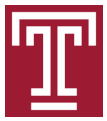
No, even A ever sent a message with the MAC tag to B, A can deny the truth and argue that B has **forged** the message.

Digital Signature is used for data integrity, authentication, and non-repudiation



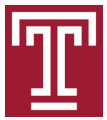
Cryptography Primitives





Symmetric vs. Asymmetric Encryption

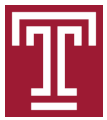
- Symmetric encryption
  - Also called *symmetric-key / secret-key / shared-key encryption*
  - Encryption: C = E(K, P); Decryption: P = D(K, C)
  - Block cipher: e.g., DES, AES
  - Stream cipher: e.g., RC4
- Asymmetric encryption
  - Also called *asymmetric-key / public-key encryption*
  - Encryption: C = E(PU, P); Decryption: P = D(PR, C)
  - E.g., RSA, Elliptic Curve



RSA (Rivest, Shamir, Adelman)

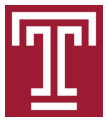
- The most widely used public key algorithm
- Its security is based on the difficulty of integer factorization
- Invented in 1977
- First discovered in 1973 by Clifford Cocks but kept secret until 1997 by Britain

"A Method for Obtaining Digital Signatures and Public Key Cryptosystems", Communications of the ACM, 1978



Difficulty of Integer Factorization

- While it is easy to calculate the product of two primes $n = p \times q$, it is very expensive to determine whether a given prime can divide a large integer
  - You basically rely on trial-and-error
- E.g., n = 91
  - Can 3 divide n -> no!
  - Can 5 divide n -> no!
  - Can 7 divide n -> yes!



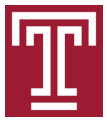
RSA Factoring Challenges

- Challenges put forward by the RSA lab to encourage research into factoring large integers

```
RSA-768 = 123018668453011775513049495838496272077285356959533479219732245215172640050726365751874520219978646938995647494277406384592519255732630345373154826850791702612214291346167042921431160222124047927473779408066535141959745985690214341 3
```

```
RSA-768 = 33478071698956898786044169848212690817704794983713768568912431388982883793878002287614711652531743087737814467999489
        × 36746043666799590428244633799627952632279158164343087642676032283815739666511279233373417143396810270092798736308917
```

The CPU time spent on the factorization is equivalent with almost 2000 years of computing on a single-core 2.2 GHz AMD Opteron-based computer.



Key Generation

n: *modulus*
e: *public exponent* or *encryption exponent*
d: *private exponent* or *decryption exponent*

Procedure

- Pick two primes p and q
- Compute n = pq
- Compute ϕ = (p-1)(q-1)
- Choose *e*, 1 < e < ϕ such that gcd(e, ϕ) = 1
  - greatest common divisor
- Compute d such that de mod ϕ = 1

  Public key: {e, n}

  Private key: {d, n}

Example

- Choose p = 3 and q = 11
- Compute n = pq = 33
- Compute ϕ = 2 * 10 = 20
- Choose e = 7 which satisfies gcd(7, 20) = 1
- Compute d = 3 as (3 * 7) % 20 = 1

  Public key: {7, 33}

  Private key: {3, 33}

RSA's Encryption and Decryption

Procedure

Public key: {e, n}

Private key: {d, n}

- C = Encrypt(PU, P)
  = P^e mod n
- P = Decrypt(PR, C)
  = C^d mod n

Example

Public key: {7, 33}

Private key: {3, 33}

P = 2

- C = 2^7 % 33 = 29
- P = 29^3 % 33 = 2

RSA's Digital Signature

How to sign a message?

- Create a message digest, m, of the information to be signed $(1 < m < n)$
- Use the private key to compute the signature

  $s = Sign(PR, m)$

  $= m^d \bmod n$
- Send the information along with the signature s

How to verify a signature?

- Independently compute the message digest, m_1, of the information received
- Use the sender's public key to recover the message digest from s

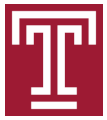
  $m_2 = s^e \bmod n$
- If $m_1 = m_2$, the signature is valid

Question

Can Digital Signature be used to verify data integrity, authentication, and achieve non-repudiation?

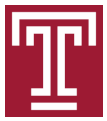
Yes. Data integrity and authentication: the adversary may corrupt or replace the information being sent, but does not have the private key to sign the message digest

Non-repudiation: only the sender can generate the digital signature, since only the sender owns the private key. Thus, the sender cannot deny that the message was signed by her/him



RSA

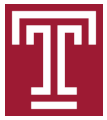
- When we say the key length of RSA, what does it mean on earth?
  - The bit length of the modulus $n = pq$
- What key size should I use for RSA?
  - 1024-bit key is already insecure
  - 2048-bit key is recommended until Year 2030
  - 3072 is needed beyond 2030



Key Size

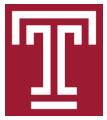
Symmetric key algorithm	Comparable RSA key length	Comparable hash function	Bits of security
2TDEA*	1024	SHA-1	80
3TDEA	2048	SHA-224	112
AES-128	3072	SHA-256	128
AES-192	7680	SHA-384	192
AES-256	15360	SHA-512	256

*2TDEA: 2-key triple Data Encryption Algorithm; i.e., 3DES using two keys
** SHA-224, 256, 384, 512 all belong to SHA-2



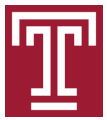
Questions

- How large a message/digest can RSA encrypt or sign?
  - The message/digest, m, to be encrypted or signed should be smaller than the modulus n
  - E.g., with a 2048-bit key, m has to be ≤ 2048 bits
- Asymmetric Encryption (including RSA) is much more expensive than Symmetric Encryption, is it possible to combine the advantages of both?
  - In practice, Asymmetric Encryption is firstly used to establish the key
  - The established key is then used in subsequent communication through inexpensive Symmetric Encryption



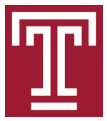
RSA Caveats

- Don’t use the same key for encryption and signing
  - Given that signing and decryption are essentially the same operation, if an attacker can convince a key holder to “sign” an encrypted message, then she gets the original
- Don’t use a common modulus n for different users



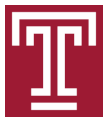
DSA – Digital Signature Algorithm

- Another widely used signature algorithm
  - NIST 1991
  - A variant of the ElGamal Signature Scheme
- DSA vs. RSA
  - Unlike RSA, which works for both encryption and signing, DSA can only sign
  - DSA is faster than RSA when generating signatures; RSA is faster than DAS when verifying signatures
  - DSA's security is based on the difficulty of the discrete logarithm problem, while RSA on integer factorization



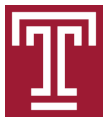
Difficulty of Discrete Logarithm

- $g^n \bmod p = m$
  - Given g, p, n, it is easy to calculate m
  - But given g, p, m, it is very difficult to calculate n
- https://www.khanacademy.org/computing/computer-science/cryptography/modern-crypt/v/discrete-logarithm-problem



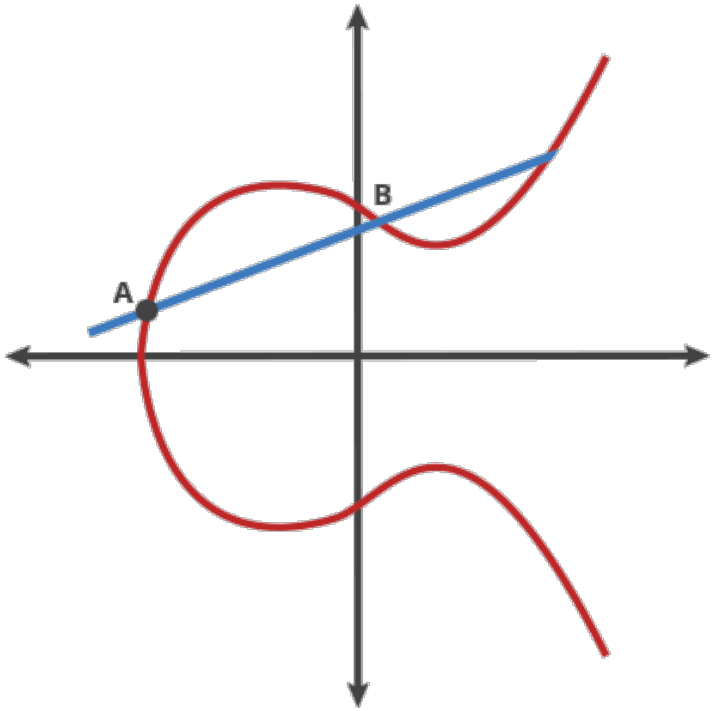
ECC (Elliptic Curve Cryptography)

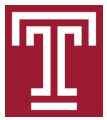
- A new approach to public-key cryptography
  - Proposed independently by Koblitz and Miller 1985
  - Based on algebraic structure of elliptic curves
- Become popular since 2004
- ECC requires smaller keys (e.g., 256bits), thus the generated signatures are smaller, reducing bandwidth and storage consumption
- But ECC is complicated and tricky to implement correctly; ECC has some uncertain patent issues



The Elliptic Curve Discrete Logarithm Problem

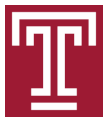
- http://arstechnica.com/security/2013/10/a-relatively-easy-to-understand-primer-on-elliptic-curve-cryptography/





Symmetric vs. Asymmetric Cryptography

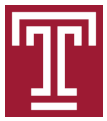
- Symmetric cipher is much faster
- With asymmetric ciphers, you can post your Public Key to the world and then the world can communicate with your secretly without having to meet you first
  - Why?
  - Only you have the private key to decrypt ciphertext
- Non-repudiation can only be achieved through asymmetric cryptography
  - Digital Signature



Summary

- RSA and ECC: encryption and digital signatures
  - Private key is used for signing and decryption
  - Public key is used for verifying and encryption
- DSA: digital signatures only

public-key system	mathematical problem	example
Integer factorization	Given a number n, find its prime factors	RSA, Rabin-Williams
Discrete logarithm	Given a prime n, and numbers g and h, find x such that $h = g^x \bmod n$	ElGamal, Diffie-Hellman, DSA
Elliptic curve discrete logarithm	Given an elliptic curve E and points P and Q on E, find x such that Q=xP	EC-Diffie-Hellman, ECDSA, ECMQV



Writing Assignments

- How do Digital Signatures assure non-repudiation?
- Since Asymmetric Cryptography is so versatile, can it replace Symmetric Cryptography completely?