Public Key (Asymmetric Key) Encryption

Kehuan Zhang

IERG4130 2022

Learning Goals

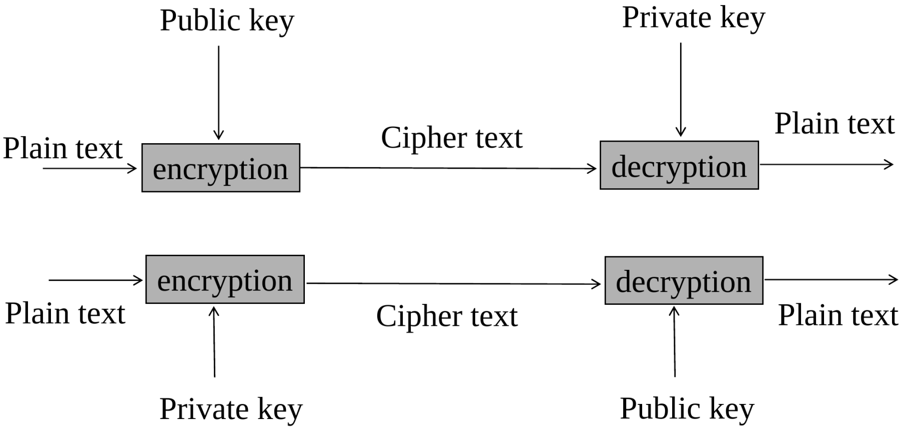
- Why do we need public-key encryption
  - Or limitations of Symmetric Encryption
- Typical use cases of public-key encryption
  - For confidentiality
  - For integrity
- Four algorithms related to public-key encryption
  - RSA, Diffie-Hellman Key Exchange Algorithm, El Gamal, ECC
  - Be able to understand and verify them at high level
    - based on some given math conclusions

Motivation: Why Do We need Public Key Encryption

- Key Management Problem in Symmetric Key Encryption
  - Too many keys: N parties will need
    $$\frac{N(N-1)}{2}$$
    different keys in order to communicate with each other
  - For asymmetric key encryption, the key number is $2N$, with N private and the other N public
- Secure key transmission problem: how to establish the shared secret key
  - Fundamentally, we need to assume there is a secure communication channel for sending symmetric encryption keys
- Symmetric key encryption cannot provide non-repudiation

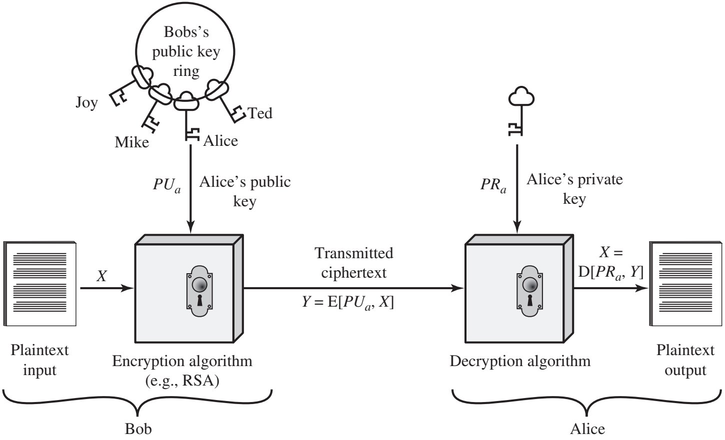
Asymmetric Key Encryptions

- Every participant has a pair of keys: Public Key and Private Key
  - The Public key is published or sent to everyone else in the community openly
  - The Private key is kept secret by its owner



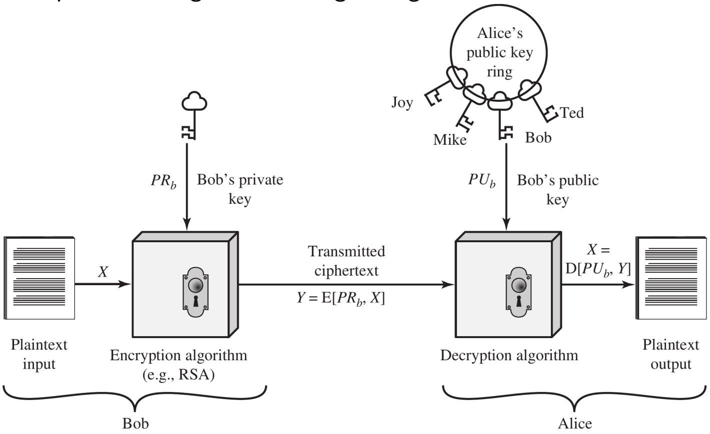
Use Public Key to Achieve Confidentiality

- Use **recipient's public key** to encrypt a message



Use Public Key to Achieve Integrity

- Use **sender's private key** to encrypt a message
  - It is equivalent to generate a digital signature



Common Algorithms Related to Public Key Encryption

- RSA, Diffie-Hellman, El Gamal, ECC
  - All of them are based on the difficulty of **Discrete Logarithm Problem**
  - Let $a = b^k \bmod n$, then given a and k, how to get b ?
  - ECC is based on the discrete logarithm problem on elliptic curve
  - The computation complexity function is an exponential function
- Fundamentally, we need two special functions $D[]$ and $E[]$, so that:
  - D[E[m, k_1]] = m = E[D[m, k_2]]
- Concept of **trap-door function**
  - Trap-door function is a **special** one way function
  - It is easy to compute in one direction, but difficult to reverse
  - However, with some **special information** (it is the *key* in our case), the reverse conputation will become easy

Some Mathematic Background - Group, Ring, Field

- Group $(G, \cdot)$ where $\cdot$ means an operator, G is one element set
  - A set of elements that is **closed** with respect to certain operation
- Group must obey following laws:
  1. *closed*: means the operation result is still in this same set
  2. associative law: $(a \cdot b) \cdot c = a \cdot (b \cdot c)$
  3. has identity 0: $0 \cdot a = a \cdot 0 = a$
  4. Has inverse a^{-1}: $a \cdot a^{-1} = 0$
- If $\cdot$ also obeys commutative law, then it becomes **Abelian Group**
  - $a \cdot b = b \cdot a$
- Example: $(Z_8, +)$ is an Abelian Group if "+" is *modular addition*
  - $Z_8 = \{0,1,2,3,4,5,6,7\}$
  - Modular addition means: the final result is the sum modular 8
    - e.g., $6 + 7 \bmod 8 = 5$
  - Identity element is 0
  - Since $2 + 6 \bmod 8 = 0$, so $2^{-1} = 6$ for $(Z_8, +)$
- Based on contents prepared by Raj Jain at Washington University in Saint Louis.

Cyclic Group

- Exponentiation: repeated application of operator, $a^3 = a \cdot a \cdot a$
- A Group is Cyclic Group if every element is a power of some fixed element
  - $\exists a \in G$, so that $\forall b \in G$, it is an exponentiation of a: $b = a^k$
  - a is called *generator* of the group
  - E.g., $\{1,2,4,8\}$ with *mod 12* multiplication is a cyclic group
    - Generator is 2
    - Can you verify? (hint: $2^4 mod 12 = 4 \in \{1,2,4,8\}$)

Ring $(R, +, \times)$: extend Abelian Group to two operators

- Ring is Abelian Group, so obeys all laws of Abelian Group
- Besides, Ring includes a multiplication operator $\times$ with following laws
  - associative: $(a \times b) \times c = a \times (b \times c)$
  - distrubition law: $a \times (b + c) = a \times b + a \times c$
- Extension: Commutative Ring
  - $\times$ operator also obeys commutative law: $a \times b = b \times a$
- Further extention on Commutative Ring: Integral Domain
  - Multiplication operation has identity and inverse (for non-zero elements)
  - Existence of multiplication identity 1: $1 \times a = a \times 1 = a$

Field

- Field is an integral domain with multiplicative inverse
  - Multiplicative inverse: a^{-1}: $a \times a^{-1} = 1$
  - $(F, +, \times, 0, 1)$
- Finite Fields: a field with finite number of elements, denoted as *GF()*
  - Also called **Galaois Feild** (named in honor of Évariste Galois)
  - GF(p) is the set of integers {0,1,. . . , p-1} with arithmetic operations modulo prime number p (*Prime Field*)
  - Closed for operations of $+, -, \times, \div$ (with some extensions)
  - E.g., GF(2), GF(5), but no GF(9)
- Extention to GF(p^n), where p is a prime number, n is an integer
  - GF(2^8) is one of the widely used Galois Fields,
  - AES works on GF(2^8)
  - Many network encoding algorithms are also working on GF(2^8)
  - But out of scope of this course

Brief Introduction to Modular Arithmetic

- If $a = k_1 \cdot n + x$, then $a \bmod n = x$ (also written as $a \equiv x (\bmod n)$)
- Add: $(a + b) \bmod n = ((a \bmod n) + (b \bmod n)) \bmod n$
  - Proof: Let $a = k_1 \cdot n + x$, $b = k_2 \cdot n + y$, then $(a + b) \bmod n = (k_1 \cdot n + x + k_2 \cdot n + y) \bmod n = ((k_1 + k_2) \cdot n + (x + y)) \bmod n = (x + y) \bmod n = ((a \bmod n) + (b \bmod n)) \bmod n$
- Multiplication: $(a \cdot b) \bmod n = ((a \bmod n) \cdot (b \bmod n)) \bmod n$
  - Proof: $(a \cdot b) \bmod n = ((k_1 \cdot n + x) \cdot (k_2 \cdot n + y)) \bmod n = (k_1 k_2 \cdot n^2 + (k_1 y + k_2 x) \cdot n + x \cdot y) \bmod n = x \cdot y \bmod n = ((a \bmod n) \cdot (b \bmod n)) \bmod n$
  - Extension: $a^b \bmod n = (a \bmod n)^b \bmod n$
  - Further extension: $a^{b+c} \bmod n = ((a^b \bmod n) \cdot (a^c \bmod n)) \bmod n$
  - Another extension: $(a^b \bmod n)^c \bmod n = a^{bc} \bmod n$
- Euler's Theorem: a and n are coprime, then $a^{\phi(n)} \equiv 1 \pmod{n}$
  - $\phi(n)$: Euler's totient function: num of positive coprime integers up to n
    - E.g., $\phi(9) = 6, [1, 2, 4, 5, 7, 8]$. For prime number p, $\phi(p) = (p - 1)$
    - Special case (proof skipped): if $n = p \cdot q$, and p, q are prime, then $\phi(n) = \phi(p \cdot q) = \phi(p)\phi(q) = (p - 1)(q - 1)$ (we will use it in RSA)

RSA Algorithm

- Ron Rivest, Adi Shamir, Len Adleman found the function in following form:
  - Encryption: $c = m^e$ mod n
  - Decryption: $m = c^d$ mod n
- How to generate the keys?
  1. Choose two large prime numbers p, q (e.g., 2048 bits each)
  2. Compute $n = p \cdot q$, $z = \phi(n) = (p-1) \cdot (q-1)$
  3. Choose e (with $e < n$) that has no common factors with z (so e and z are "relatively prime" $>$)
  4. Choose d such that $e \cdot d - 1$ is exactly divisible by z (in other words: $e \cdot d = K \cdot z + 1 = K \cdot \phi(n) + 1$
  5. Public key K_{pub} is (n, e), and private key K_{priv} is (n, d)

RSA Algorithm: Decryption and Encryption

- Given $K_{pub} = (n, e)$, and $K_{priv} = (n, d)$
  - Encryption $c = E(K_{pub}, m) = m^e \bmod n$
  - Decryption $m = D(K_{priv}, c) = c^d \bmod n$
- The magic behind is: (using above useful math conclusions)

$$
\begin{aligned}
& D(K_{priv}, c) = c^d \bmod n \\
& = E(K_{pub}, m)^d \bmod n = (m^e \bmod n)^d \bmod n \quad = m^{e \cdot d} \bmod n \\
& = m^{K \cdot \phi(n) + 1} \bmod n \\
& = (m^{K \cdot \phi(n)} \bmod n) \cdot (m^1 \bmod n) \bmod n \\
& = (m^{\phi(n)} \bmod n)^K \cdot m \bmod n \\
& = (1^K \cdot m) \bmod n \\
& = m \bmod n = m
\end{aligned}
$$

A Running Example for RSA

1. Bob chooses $p = 5, q = 7$
2. Compute $n = p \cdot q = 5 \times 7 = 35$,
   $z = (p-1) \cdot (q-1) = (5-1) \times (7-1) = 24$
3. Choose $e = 5$ (so e, z are relatively prime)
4. Choose $d = 29$ (so $e \cdot d - 1 = 5 \times 29 - 1 = 144$ which is exactly divisible by $z (= 24)$
5. Public key K_{pub} is $(n, e) = (35, 5)$, and private key K_{priv} is $(n, d) = (35, 29)$

- Let plaintext is letter 'D', so m=4
- Encryption: $c = m^e \bmod n = 4^5 \bmod 35 = 1024 \bmod 35 = 9$
- Decryption: $m = c^d \bmod n = 9^{29} \bmod 35 = \cdots =?$

Proof of RSA's important property

- $D(K_{priv}, E(K_{pub}, m)) = D(K_{pub}, E(K_{priv}, m)) = m$
  - We have already proved (confidentiality part) that $D(K_{priv}, E(K_{pub}, m)) = m$
  - Now let's prove (integrity part) that $D(K_{pub}, E(K_{priv}, m)) = m$

$$
\begin{aligned}
& D(K_{pub}, E(K_{priv}, m)) \\
& = E(K_{priv}, m)^e \bmod n \\
& = (m^d \bmod n)^e \bmod n \\
& = m^{d \cdot e} \bmod n \\
& = m^{K \cdot \phi(n)+1} \bmod n \\
& = (m^{\phi(n)} \bmod n)^K \cdot m \bmod n \\
& = m \bmod n = m
\end{aligned}
$$

Security of RSA

- It is a discrete logarithm problem to infer plaintext from ciphertext
  - Currently there is no efficient algorithm for solving discrete logarithm problem
- How about brute-force attack to get private key (n, d)?
  - Normally we require p and q to be 2048-bit or more
    - Thus n is 4096, while e and d are similar
    - It is extremely difficult to get d with brute-force attacks
- Can we get d from the known public key (e, n)?
  - d is chonse so that $e \cdot d \bmod z = 1$
  - Since $z = (p - 1) \cdot (q - 1)$, and $n = p \cdot q$, so the problem is converted to:
    - Given n, can we decompose it into two prime numbers p and q
    - Currently, there no efficient algorithm, but we also cannot exclude the existence of such algorithms
- Of course, the randomness of p and q are very important
  - Poor randomness of p and q will make decomposition easier
  - Please click Flaw Found in an Online Encryption Method for one example

Security of RSA (Cont.) - More Attacks

- RSA is vulnerable to Man-in-the-middle attack
  - Because there is no strong binding between public key and user's identity
- Chosen ciphertext attack:
  - Attacker could send you some documents (which actually is ciphertext) for encryption
    - ★ If you return the "encrypted" content, you actually sent out the plaintext
    - ★ Especially when you use the same pair of keys for both encryption and signing
- Cube-root attack for smaller e: if e is too small (say $e = 3$), then maybe $m^3 < n$
  - So $c = m^e \bmod n = m^3 \bmod n = m^3$, so $m = \sqrt[3]{c}$

Performance of RSA

- For hardware implementation, RSA is about 1000 timers slower than DES
  - Generally, longer key means more secure, but is more slower
  - For software implementation, RSA is about 100 timers slower
- RSA is recommended to encrypt small amount of data
  - E.g., use RSA to encrypt a random number (to be used as session key for AES)

Other Public Key Encryption Algorithms: Diffie-Hellman

- Diffie-Hellman is used to establish a shared secret key over insecure channel
- The Diffie-Hellman algorithm
  0. Preparation: choose a prime number q and α (where α is a primitive root of q). Make both of them public
  1. Alice: chooses a random number x in $[2, ..., q-1]$ as her secret, and send Bob $v = \alpha^x \bmod q$
  2. Bob: chooses a random number y in $[2, ..., q-1]$ as his secret, and send Alice $w = \alpha^y \bmod q$
  3. The shared key is $K_{AB} = v^y \bmod q = w^x \bmod q$
- What is **primitive root**?
  - α is a primitive root of q if: for every integer b coprime to q, there is an integer k such that $\alpha^k = b \bmod q$
  - if q itself is a prime, then all numbers in $[1, ..., (q-1)]$ are coprime to q
- Correctness Proof: $v^y \bmod q = w^x \bmod q$ (using math conclusion in page12)
  - $v^y \bmod q = (\alpha^x \bmod q)^y \bmod q = \alpha^{x \cdot y} \bmod q = K_{AB}$
  - $w^x \bmod q = (\alpha^y \bmod q)^x \bmod q = \alpha^{y \cdot x} \bmod q = K_{AB}$

A Running Example for Diffie-Hellman Algorithm

0. Alice and Bob agreed on the prime $q = 353$ and q's primitive root of $\alpha = 3$
1. Choose random secret keys
   - Alice chooses $x = 97$, Bob chooses $y = 233$
2. Computer their own public keys:
   - Alice: $v = \alpha^x \bmod q = 3^{97} \bmod 353 = 40$
   - Bob: $w = \alpha^y \bmod q = 3^{233} \bmod 353 = 248$
3. Alice sends $v = 40$ to Bob, and Bob sends $w = 248$ back to Alice
4. Compute shared secret key:
   - Alice: $K_{AB} = w^x \bmod q = 248^{97} \bmod 353 = 160$
   - Bob: $K_{AB} = v^y \bmod q = 40^{233} \bmod 353 = 160$

Question: How to quickly calculate $40^{233} \bmod 353$?

Security of Diffie-Hellman Algorithm

- Its security is also built on the difficulty of **discrete logarithm** problem
  - Knowing v (which is result of $\alpha^x \bmod q$), it is still computationally difficult to infer the value of x
- Similar to RSA, it is vulnerable to Man-in-the-Middle attack
  - There is no way to ensure that w is really coming from Bob, since w derived from a random number which cannot be used to verify the real identity of the other end

Other Public Key Encryption Algorithms: El Gamal

- El Gamal can be considered as a generalization of Diffie-Hellman key-exchange algorithm

0. Key generation
   - Let q be a prime number, and α is a primitive root of q
   - Choose a random number x in $[1, \cdots, (q-1)]$
   - Compute $y = \alpha^x \bmod q$
   - Public key is (y, α, q), and Private key is x
1. Encryption of plaintext message M $(M < q)$
   - Select a random k from $[1, \cdots, (q-2)]$
   - $C_1 = \alpha^k \bmod q$
   - $C_2 = (y^k M) \bmod q$
   - Ciphertext $C = (C_1, C_2)$
2. Decryption of ciphertext $C = (C_1, C_2)$
   - $M = [C_2 \cdot (C_1^x)^{-1}] \bmod q$
   - here $b^{-1} \bmod q$ is the "multiplicative inverse" of $b \bmod q$, so that:

$$[b \cdot b^{-1}] \bmod q = 1 \bmod q$$

Proof of El Gamal Algorithm

- Why $[C_2 \cdot (C_1^x)^{-1}] \bmod q = M$?

$[C_2 \cdot (C_1^x)^{-1}] \bmod q = [((y^k M) \bmod q) \cdot (C_1^x)^{-1}] \bmod q$
$= [\alpha^{xk} M \cdot (C_1^x)^{-1}] \bmod q$
$= [(\alpha^k)^x M \cdot (C_1^x)^{-1}] \bmod q$
$= [C_1^x M \cdot (C_1^x)^{-1}] \bmod q$
$= [M \cdot 1] \bmod q$
$= M$

- How to Calculate multiplicative inverse during decryption?
  - Use Euler's Theorem: (on Page 12)
    - ★ $a^{\phi(q)} = 1 \bmod q$ if a and q are coprime
  - Since $a \cdot a^{-1} \bmod q = 1 = a^{\phi(q)} \bmod q = (a \cdot a^{\phi(q)-1}) \bmod q$
  - So $a^{-1} \bmod q = a^{\phi(q)-1} \bmod q$, or written as: $a^{-1} = a^{\phi(q)-1} \bmod q$
  - If q is a prime, then $\phi(q) = q - 1$
  - then: $a^{-1} \bmod q = a^{\phi(q)-1} \bmod q = a^{q-2} \bmod q$

A Running Example of El Gamal Algorithm

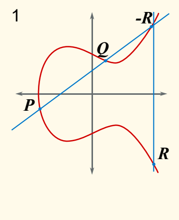
0. Let prime $q = 11, \alpha = 2$. Choose secret private key $x = 3$
   - Compute $y = \alpha^x \bmod q = 2^3 \bmod 11 = 8$
1. Encryption of plaintext message $M = 7 (< q)$:
   - Select $k = 4$ from $[1, \cdots, (q-2)]$
   - $C_1 = \alpha^k \bmod q = 2^4 \bmod 11 = 16 \bmod 11 = 5$
   - $C_2 = (y^k M) \bmod q = (8^4 \times 7) \bmod 11 = 6$
   - Ciphertext $C = (C_1, C_2) = (5, 6)$
2. Decryption of ciphertext $C = (C_1, C_2) = (5, 6)$
   - First calculate $(C_1^x)^{-1} \bmod q = (5^3)^{-1} \bmod 11 = 125^{(11-2)} \bmod 11 = 3$
   - Then calculate $M = [C_2 \cdot (C_1^x)^{-1}] \bmod q = [6 \times (5^3)^{-1}] \bmod 11 = (6 \times 3) \bmod 11 = 7$

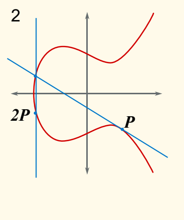
Security of El Gamal Algorithm

- Just like Diffie-Hellman algorithm, security of El Gamal is also built on the difficulty of discrete logarithm
  - Secret is the private key x
  - Then $y = \alpha^x \bmod q$
  - And public key is (y, α, q)
  - It is computationally difficult to infer x from public key, especially from y
- Note that: key is discarded after encryption and is not required for decryption
  - Select a random k from $[1, \cdots, (q-2)]$
  - $C_1 = \alpha^k \bmod q$
  - $C_2 = (y^k M) \bmod q$
  - Ciphertext $C = (C_1, C_2)$
  - For decryption, plaintext $M = [C_2 \cdot (C_1^x)^{-1}] \bmod q$
- Limitation
  - The ciphertext is about twice as big as the plaintext

Elliptic Curve Cryptosystems (ECC)

- A public key encryption scheme proposed by Koblitz and Miller in 1985
- Built on the **Elliptic Curve Discrete Logarithm Problem** (ECDLP)





$$P + Q = (x_1, y_1) + (x_2, y_2) = R = (x_3, y_3)$$

$$\lambda = (y_2 - y_1)/(x_2 - x_1), x_3 = \lambda^2 - x_2 - x_1, y_3 = \lambda(x_1 - x_3) - y_1$$

- On an elliptic curve EP(a,b), Given $K = k \cdot P = P + P + \cdots + P$, can we get get k from (K,P)? - It is difficult.

Elliptic Curve Cryptosystems (ECC) (Cont.)

0. Preparation: Alice need to:
   - Choose Elliptic Curve parameter a and b, and a base point G on curve
   - Choose a random number k as private key
   - Compute public key $K = kG$, and send [EP(a,b), K, G] to Bob
1. Encryption (done by Bob)
   - Encode data to a point M on curve EP(a,b)
   - Choose a random number r
   - Compute $C_1 = M + r \cdot K$; $C_2 = r \cdot G$, send $C[C_1, C_2]$ to Alice
2. Decryption (done by Alice)
   - $M = C_1 - k \cdot C_2$
   - Why?
     $C_1 - k \cdot C_2 = M + r \cdot K - k \cdot (r \cdot G) = M + r \cdot (k \cdot G) - k \cdot (r \cdot G) = M$

- The first true alternative for RSA
  - Shorter Keys (224-bit in ECC vs. 2048-bit in RSA)
  - Fast and compact implementations, especially in hardware
    - Thus it is good in areas of embedded or mobile systems where performance, bandwidth, and storage are limited

NIST Recommended Key Size

Date	Security Strength	Symmetric Algorithms	Factoring Modulus	Discrete Logarithm Key	Discrete Logarithm Group	Elliptic Curve	Hash (A)	Hash (B)
Legacy (1)	80	2TDEA	1024	160	1024	160	SHA-1 (2)	
2019 - 2030	112	(3TDEA) (3) AES-128	2048	224	2048	224	SHA-224 SHA-512/224 SHA3-224	
2019 - 2030 & beyond	128	AES-128	3072	256	3072	256	SHA-256 SHA-512/256 SHA3-256	SHA-1 KMAC128
2019 - 2030 & beyond	192	AES-192	7680	384	7680	384	SHA-384 SHA3-384	SHA-224 SHA-512/224 SHA3-224
2019 - 2030 & beyond	256	AES-256	15360	512	15360	512	SHA-512 SHA3-512	SHA-256 SHA-512/256 SHA-384 SHA-512 SHA3-256 SHA3-384 SHA3-512 KMAC256

- Short-Term: AES-128, RSA-2048, ECC-224

Reference: `https://www.keylength.com/en/4/`

Some Predictions by the Adi Shamir (S. of RSA)

- AES will remain secure for the forseeable future
- Some public key schemes and key sizes will be successfully attacked in the next few years
- Crypto will be invisibly everywhere
- Vulnerabilities will be visibly everywhere
- Crypto research will remain vigorous, but only its simplest ideas will become practically useful
- Non-crypto security will remain a mess

Conclusion

- In this lecture, we have learned
  - The needs for asymmetric key (public-key) encryption
  - Different operating mode of asymmetric key encryption
  - Some math about the the Group/Ring/Group and modular operations
  - Four different public-key encryption algorithms
- Next time:
  - Message Authentication Code and Hash Function
  - To achieve the security goal of Integrity