

Asymmetric Cryptography

Segurança Informática em Redes e Sistemas
2024/25

David R. Matos, Ricardo Chaves

Ack: Miguel Pardal, Miguel P. Correia, Carlos Ribeiro

Roadmap

- Asymmetric Ciphers
- Digital Signatures

Roadmap

- **Asymmetric Ciphers**
- Digital Signatures

Asymmetric ciphers: overview

- Also known as **public-key cryptography**
- Appeared with the objective of solving a hard problem: **key distribution**
- Use a **pair of keys**
 - One is **private**, personal, non-transmissible
 - One is **public**, can/should be widely known
- Allow for
 - **Confidentiality** with the exchange of secret keys
 - **Authentication** with digital signatures
 - **Integrity** with digital signatures

Asymmetric ciphers: assessment

- **Advantages**

- N communicating parties \Rightarrow N key pairs
- Instead of $N(N-1)/2$ as symmetric ciphers

- **Disadvantages**

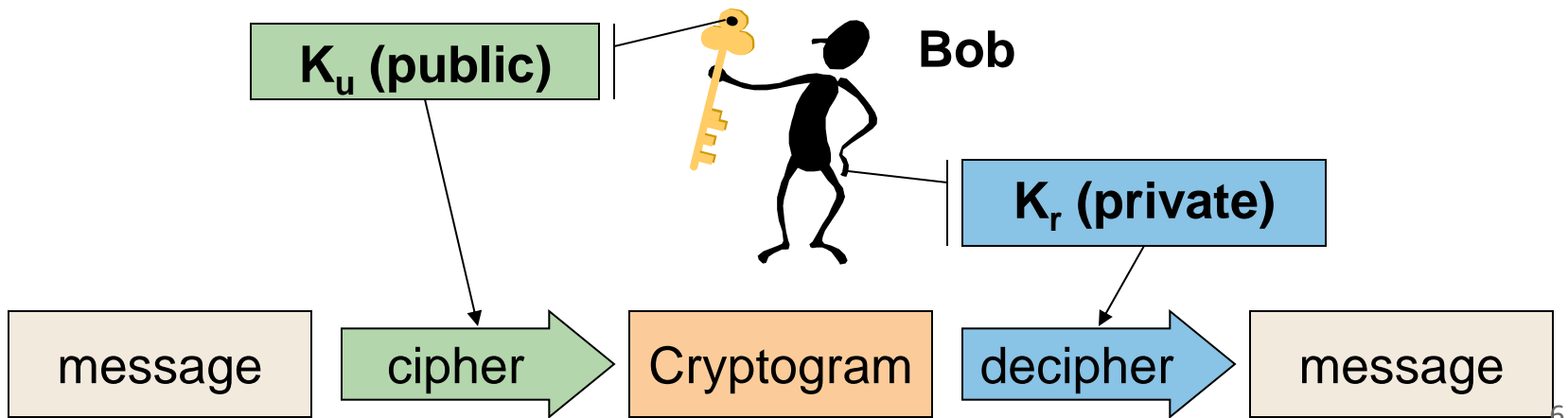
- Performance: typically inefficient in comparison to symmetric ciphers (but signatures are not too slow today)

- **Problems**

- Distribution of public keys
 - Although much simpler than distributing secret keys
- Life-time of the key pairs (revocation)

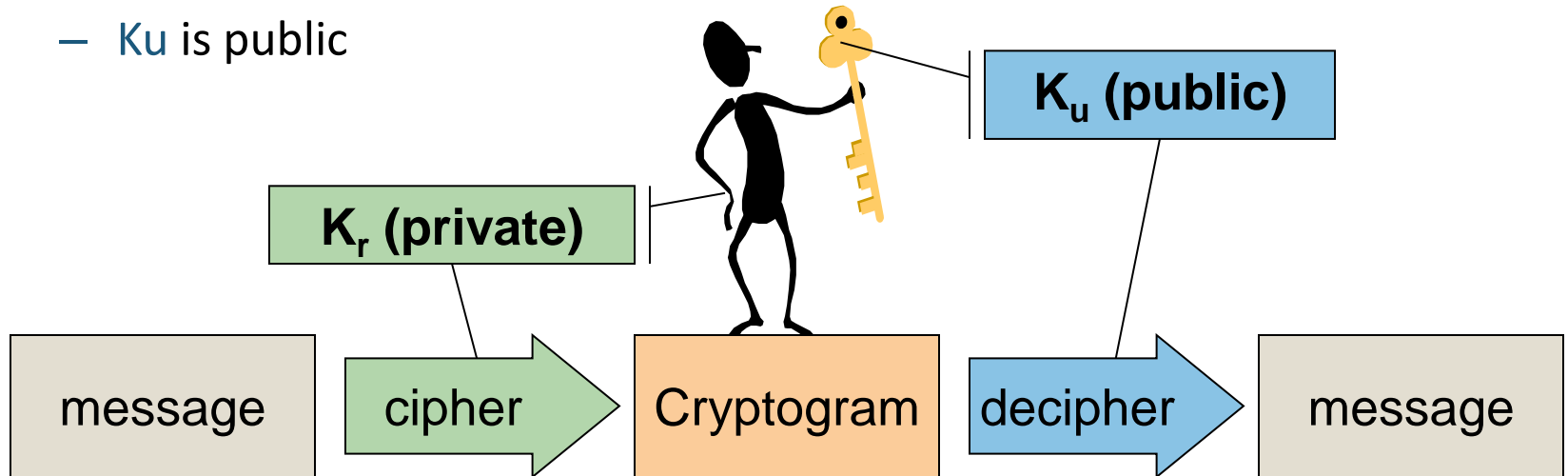
Asymmetric ciphers: Confidentiality

- Cipher / decipher with a pair of keys
 - $C = E(P, K_u)$ $P = D(C, K_r)$
 - To communicate with confidentiality with X we only need to know K_u
 - Very inefficient for large P, so not used like this for encrypting messages/files
- No authentication
 - Bob does not know who generated the cryptogram

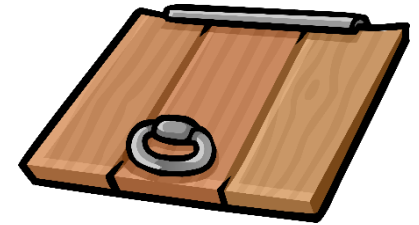


Asymmetric ciphers: Authentication

- The cryptogram cannot be modified
 - $C = E(P, K_r)$ $P = D(C, K_u)$
 - Only X knows the key K_r used to generate the cryptogram
 - Better idea to encrypt $H(P)$: $C = E(H(P), K_r)$
- No confidentiality
 - Whoever knows the key K_u is able to decipher the cryptogram
 - K_u is public



Trapdoor functions



- Functions with the properties:
 - Computing in one direction is easy $A \rightarrow B$
 - Computing the inverse is very hard (unless you know a secret) $A \nleftarrow B$
- Trapdoor functions are the core concept that enables asymmetric cryptography
 - If you find mathematical functions with these properties, you can build a cryptographic scheme around it

Asymmetric Ciphers

- Used mathematical approximations
 - Factorization of large numbers
 - Discrete logarithm problem
 - Knapsack problem
- Most common encryption algorithms
 - RSA (Rivest-Shamir-Adleman) – first and most used asymmetric encryption algorithm
 - ECC (Elliptic Curve Cryptography) – much used today, especially for signatures
- Other public-key techniques (not encryption)
 - Diffie-Hellman (DH), also known as Diffie-Hellman-Merkle – was foundational in the development of asymmetric crypto

Modular arithmetic

- Idea: do arithmetic normally ($+$ $-$ \times \div) but result is the remainder of dividing by a modulus
 - mod, i.e., the result is the remainder of the integer division of the result by the modulus
 - Performs arithmetic operations not on a line but on a circle
 - mod can be applied “as often as you want”
 - Examples:
 - $1537 \times 4248 \pmod{10} = 6529176 \pmod{10} = 6$
 - $1537 \pmod{10} \times 4248 \pmod{10} = 7 \times 8 \pmod{10} = 56 \pmod{10} = 6$

RSA

- Rivest-Shamir-Adleman
- First **asymmetric encryption** algorithm
 - But not the first asymmetric cryptography algorithm
 - that was Diffie-Hellman-Merkle
- A major step forward for cryptography
 - Simplified key distribution

Programming
Techniques

S.L. Graham, R.L. Rivest*
Editors

A Method for Obtaining Digital Signatures and Public-Key Cryptosystems

R. L. Rivest, A. Shamir, and L. Adleman
MIT Laboratory for Computer Science
and Department of Mathematics

An encryption method is presented with the novel property that publicly revealing an encryption key does not thereby reveal the corresponding decryption key. This has two important consequences:

(1) Couriers or other secure means are not needed to transmit keys, since a message can be enciphered using an encryption key publicly revealed by the intended recipient. Only he can decipher the message, since only he knows the corresponding decryption key. (2) A message can be "signed" using a privately held decryption key. Anyone can verify this signature using the corresponding publicly revealed encryption key. Signatures cannot be forged, and a signer cannot later deny the validity of his signature. This has obvious applications in "electronic mail" and "electronic funds transfer" systems. A message is encrypted by representing it as a number M , raising M to a publicly specified power e , and then taking the remainder when the result is divided by the publicly specified product, n , of two large secret prime numbers p and q . Decryption is similar; only a different, secret, power d is used, where $e * d = 1 \pmod{(p-1) * (q-1)}$. The security of the system rests in part on the difficulty of factoring the published divisor, n .

Key Words and Phrases: digital signatures, public-key cryptosystems, privacy, authentication, security, factorization, prime number, electronic mail, message-passing, electronic funds transfer, cryptography.

CR Categories: 2.12, 3.15, 3.50, 3.81, 5.25

General permission to make fair use in teaching or research of all or part of this material is granted to individual readers and to nonprofit libraries acting for them provided that ACM's copyright notice is given and that reference is made to the publication, to its date of issue, and to the fact that reprinting privileges were granted by permission of the Association for Computing Machinery. To otherwise reprint a figure, table, other substantial excerpt, or the entire work requires specific permission as does republication, or systematic or multiple reproduction.

This research was supported by National Science Foundation grant MCS76-14294, and the Office of Naval Research grant number N00014-67-A-0204-0063.

* Note. This paper was submitted prior to the time that Rivest became editor of the department, and editorial consideration was completed under the former editor, G. K. Manacher.

Authors' Address: MIT Laboratory for Computer Science, 545 Technology Square, Cambridge, MA 02139.

© 1978 ACM 0001-0782/78/0200-0120 \$00.75

120

TS LINK

I. Introduction

The era of "electronic mail" [10] may soon be upon us; we must ensure that two important properties of the current "paper mail" system are preserved: (a) messages are *private*, and (b) messages can be *signed*. We demonstrate in this paper how to build these capabilities into an electronic mail system.

At the heart of our proposal is a new encryption method. This method provides an implementation of a "public-key cryptosystem", an elegant concept invented by Diffie and Hellman [1]. Their article motivated our research, since they presented the concept but not any practical implementation of such a system. Readers familiar with [1] may wish to skip directly to Section V for a description of our method.

II. Public-Key Cryptosystems

In a "public-key cryptosystem" each user places in a public file an encryption procedure E . That is, the public file is a directory giving the encryption procedure of each user. The user keeps secret the details of his corresponding decryption procedure D . These procedures have the following four properties:

(a) Deciphering the enciphered form of a message M yields M . Formally,

$$D(E(M)) = M. \quad (1)$$

(b) Both E and D are easy to compute.

(c) By publicly revealing E the user does not reveal an easy way to compute D . This means that in practice only he can decrypt messages encrypted with E , or compute D efficiently.

(d) If a message M is first deciphered and then enciphered, M is the result. Formally,

$$E(D(M)) = M. \quad (2)$$

An encryption (or decryption) procedure typically consists of a *general method* and an *encryption key*. The general method, under control of the key, enciphers a message M to obtain the enciphered form of the message, called the *ciphertext* C . Everyone can use the same general method; the security of a given procedure will rest on the security of the key. Revealing an encryption algorithm then means revealing the key.

When the user reveals E he reveals a *very inefficient* method of computing $D(C)$: testing all possible messages M until one such that $E(M) = C$ is found. If property (c) is satisfied the number of such messages to test will be so large that this approach is impractical.

A function E satisfying (a)-(c) is a "trap-door one-way function;" if it also satisfies (d) it is a "trap-door one-way permutation." Diffie and Hellman [1] introduced the concept of trap-door one-way functions but

RSA overview

- **Base values and modulus**
 - p, q – **secret**; large **prime** numbers
 - $n = p \times q$ – **public**; large (1K, 2K, 3K bits); used as **modulus**; key length
- **Public key** – exponent e and n ($e < n$)
- **Private key** – exponent d ($d < n$) and primes p and q
- **Key generation**: choose p, q large primes
 - e chosen such that it is coprime with $z = \phi(n) = (p-1)(q-1)$
 - *coprime* means that the only factor that divides both is 1
 - $\phi(n)$ is the Euler's totient function; counts factors up to n that are coprime with n
 - d chosen such that $e \cdot d \equiv 1 \pmod{z} \iff e \cdot d \bmod z = 1$
- **Encryption with public key**: $C = P^e \bmod n$
- **Decryption with private key**: $P = C^d \bmod n$
 $= (P^e \bmod n)^d \bmod n = P^{ed} \bmod n = P^{ed \bmod z} \bmod n = P^1 \bmod n = P$

Public key example

- Taken from a browser

Public Key Info

Algorithm RSA Encryption (1.2.840.113549.1.1.1)

Parameters None

Public Key

= modulus

256 bytes: D0 18 CF 45 D4 8B CD D3 9C E4 40
EF 7E B4 DD 69 21 1B C9 CF 3C 8E 4C 75 B9 0F
31 19 84 3D 9E 3C 29 EF 50 0D 10 93 6F 05 80
80 9F 2A A0 BD 12 4B 02 E1 3D 9F 58 16 24 FE
30 9F 0B 74 77 55 93 1D 4B F7 4D E1 92 82 10 F6
51 AC 0C C3 B2 22 94 0F 34 6B 98 10 49 E7 0B
9D 83 39 DD 20 C6 1C 2D EF D1 18 61 65 E7 23
83 20 A8 23 12 FF D2 24 7F D4 2F E7 44 6A 5B
4D D7 50 66 B0 AF 9E 42 63 05 FB E0 1C C4 63
61 AF 9F 6A 33 FF 62 97 BD 48 D9 D3 7C 14 67
DC 75 DC 2E 69 E8 F8 6D 78 69 D0 B7 10 05 B8
F1 31 C2 3B 24 FD 1A 33 74 F8 23 E0 EC 6B 19 8A
16 C6 E3 CD A4 CD 0B DB B3 A4 59 60 38 88 3B
AD 1D B9 C6 8C A7 53 1B FC BC D9 A4 AB BC DD
3C 61 D7 93 15 98 EE 81 BD 8F E2 64 47 20 40
06 4E D7 AC 97 E8 B9 C0 59 12 A1 49 25 23 E4
ED 70 34 2C A5 B4 63 7C F9 A3 3D 83 D1 CD 6D
24 AC 07

exponent e

Exponent 65537

Key Size 2 048 bits

= size of the modulus

RSA example with small numbers

- Key generation

- Pick primes **p** and **q**: 3 and 11
- Calculate modulus **n** = $p \times q = 3 \times 11 = 33$
- Calculate **z** = $\phi(n) = (p-1) \times (q-1) = 2 \times 10 = 20$
- Choose **e** such that **e** is co-prime with **z** and $e < n$: $e = 7$
- Public key: **e** = 7, **n** = 33
- Calculate **d**
 - $e \cdot d = 1 \pmod{z}$ and $d < n \Leftrightarrow$
 - $7 \cdot d \pmod{20} = 1$ and $d < 33 \Rightarrow$
 - **d** = 3

- Encrypt w/ public key **P** = 14

- $C = P^e \pmod{n}$
- $C = 14^7 \pmod{33}$
- $C = 20$

- Decrypt w/ private key

- $P = C^d \pmod{n}$
- $P = 20^3 \pmod{33}$
- $P = 14$

RSA calculator

- Follow the algorithm steps with an online calculator:
 - <https://www.cs.drexel.edu/~popyack/IntroCS/HW/RSASWorksheet.html>

RSA Calculator

JL Popyack, October 1997

This guide is intended to help with understanding the workings of the RSA Public Key Encryption/Decryption scheme. No provisions are made when dealing with large numbers.

Step 1. Compute N as the product of two prime numbers p and q:

p

q

Enter values for p and q then click this button:

The values of p and q you provided yield a modulus N, and also a number $r=(p-1)(q-1)$, which is very important. You will need to find a number which appears a list of some numbers which equal $1 \bmod r$. You will use this list in Step 2.

N = p*q

r = (p-1)*(q-1)

```
6865 13729 20593 27457 34321 41185 48049 54913 61777
68641 75505 82369 89233 96097 102961 109825 116689
123553 130417 137281 144145 151009 157873 164737 171601
```

Why is RSA difficult to break?

- RSA is challenging to break due to the computational difficulty of integer factorization
- It is easy to multiply large prime numbers to obtain a modulus n
 - But it is extremely hard to perform the inverse operation, i.e., to deduce the original prime factors from n alone
- The encryption process in RSA is straightforward, utilizing the public key
 - But the decryption (inverse RSA) requires knowledge of these prime factors, which constitute the private key
- The complexity of factorizing large numbers into primes ensures that knowing the public key alone is insufficient for decryption, as it does not reveal the prime factors
- This fundamental asymmetry between the ease of multiplication and the difficulty of factorization underpins RSA's security.
- Future crisis: **Shor's quantum algorithm** threatens RSA by efficiently factorizing integers
 - Current quantum technology can only factorize small numbers
 - ≥ 2048 -bit numbers are safe (for now...)
 - **Post-quantum** crisis; need for **quantum resistance**

RSA signatures

- RSA is often used to produce **signatures**
 - Public key – e, n
 - Private key – d, p, q
- **Signing the message M :**
 - Obtain S such that: $S = (H(M))^d \bmod n$
 - Only the private key owner can generate the signature
- **Verifying the signature:**
 - Check if: $H(M) == S^e \bmod n$
 - Anyone with the public key can verify the signature

Elliptic Curve Cryptography (ECC)

- RSA and DH use integer and polynomial arithmetic with very large numbers/polynomials
 - Impose a significant load in storing and processing keys and messages
- **Elliptic curves** are an alternative
 - Offer same security with smaller bit sizes
 - 224-bit ECC ~ 2048-bit RSA
 - 500-bit ECC ~ 15k-bit RSA
 - Newer, not as well analyzed, but commonly used
 - Many curves exist, each with their tradeoffs
 - Not necessarily faster than RSA for the same equivalent security
 - Normally used for **key agreement or signing**, not ciphering

Elliptic Curve Cryptography (ECC)

- Analogy to RSA:
 - ECC addition is analogous to modulus multiplication
 - ECC repeated addition is analogous to modulus exponentiation
- The “hard” problem is the **elliptic curve logarithm problem**
 - $Q=k \cdot P$, where Q, P belong to an elliptic curve
e.g. $y^2 = x^3 - 3x + b \text{ modulo } p$
 - Easy to compute Q given k, P
 - Hard to find k given Q, P
- Public Key operations $Q(x,y)=kP(x,y)$:
 - Q = public key
 - k = private key
 - P = base point (a parameter of the curve)

ECC Encryption/Decryption

- There are several alternatives, but we consider the simplest: to encode the message **M** as a point **Q_M** on the elliptic curve
- Key generation
 - Select suitable curve and base point P
 - Bob chooses a **private key**: $k_B < n$
 - Bob computes the **public key**: $Q_B = k_B \cdot P$
- Alice **encrypts** **Q_M** using Bob's **public key** **Q_B**:
$$C_M = \{r \cdot P, Q_M + r \cdot Q_B\}$$
 – r is a random number; C_M is a point in the curve
- Bob **decrypts** **C_M** using its **private key** **k_B**:

$$\underbrace{Q_M + r \cdot Q_B - k_B r \cdot P}_{C_M} = Q_M + r(k_B \cdot P) - k_B(r \cdot P) = Q_M$$

Porque $Q_B = k_B P$

ECC Diffie-Hellman

- ECC Diffie-Hellman:
 - Key exchange is possible (analogous to Diffie-Hellman)
 - users select a suitable curve $E_p(a,b)$ – public information
 - select base point $P=(x_1,y_1)$ – public information
 - with large order n ($nP=O$, and n smallest with this property).
 - A & B select private keys:
 - $k_A < n$ & $k_B < n$
 - compute public keys:
 - $Q_A = k_A P$ & $Q_B = k_B P$
 - compute shared key:
 - $K = k_A Q_B$, or $K = k_B Q_A$
 - same since:
 - $K = k_A k_B P$

Roadmap

- Asymmetric Ciphers
- **Digital Signatures**

Digital signatures – objectives

- Authenticate the content of a document
- Authenticate its signer
- Being able to assure authentication towards a third party
- Signer cannot repudiate the signature



Legislação Consolidada

Decreto-Lei n.º 290-D/99 - Diário da República n.º 178/1999, 1º Suplemento, Série I-A de 1999-08-02

Aprova o regime jurídico dos documentos electrónicos e da assinatura digital

Assinatura Digital

A assinatura digital qualificada permite ao titular de um Cartão de Cidadão, por vontade própria, assinar com a chave pessoal existente no seu Cartão de Cidadão.

Qualquer entidade pode verificar a assinatura digital recorrendo ao uso do certificado digital pessoal do cidadão e a meios de verificação da validade deste certificado.

CMD Assinatura

A assinatura eletrónica qualificada através da Chave Móvel Digital (CMD) permite ao cidadão, português ou estrangeiro, por vontade própria, assinar com a palavra-chave por si escolhida e respetivo código de segurança.

Digital signatures

- Used approximations: asymmetric cipher + hash function
- **Basic algorithm** (K_x^r - private key K_x^u - public key)
 - Signature: $S_x(\text{doc}) = E(K_x^r, \text{hash}(\text{doc}))$
 - Validation: check if $D(K_x^u, S_x(\text{doc})) = \text{hash}(\text{doc})$
- Why cannot an adversary provide a fake signature?
- Another option:
 - Signature: $S_x(\text{doc}) = \text{info} + E(K_x^r, \text{hash}(\text{doc} + \text{info}))$
 - Validation: $D(K_x^u, S_x(\text{doc})) = \text{hash}(\text{doc} + \text{info})$
 - Example **info**: timestamp, used algorithm, ...
- Today often **ECDSA** is adopted (with elliptic curve encryption)

Surreptitious forwarding

- A sends B : $\{\{ \text{"I love you"} \}_a\}_B$
- B deciphers $\rightarrow \{ \text{"I love you"} \}_a$
- B sends C : $\{\{ \text{"I love you"} \}_a\}_C$

a – private key
B, C – public keys

Message stealing

- A tries to send | B : $\{\{\text{"my idea"}\} B\}_a$
- C intercepts message
- C sends B : $\{\{\text{"my idea"}\} B\}_c$

a,c – private keys
B – public key

Solutions

- Sign the name of the receiver
 - A ---> B : {{Bob, msg}a}B
- Cipher the name of the sender
 - A ---> B : {{Alice, msg}B, #msg}a
- Incorporate both names in the message
 - A ---> B : {{``A ---> B", msg}B, #msg}a
 - A ---> B : {{``A ---> B", msg}a}B
- Sign + Cipher + Sign*
- Cipher + Sign + Cipher*
- * Not practical
- [Davis2001] Defective Sign & Encrypt in S/MIME, PKCS#7, MOSS, PEM, PGP, and XML

Transitioning the Use of Cryptographic Algorithms and Key Lengths



Elaine Barker
Allen Roginsky

This publication is available free of charge from:
<https://doi.org/10.6028/NIST.SP.800-131Ar2>

C O M P U T E R S E C U R I T Y



**Table 2: Approval Status of Algorithms Used for Digital Signature
Generation and Verification**

Digital Signature Process	Domain Parameters	Status
Digital Signature Generation	< 112 bits of security strength: DSA: $(L, N) \neq (2048, 224), (2048, 256)$ or $(3072, 256)$ ECDSA: $\text{len}(n) < 224$ RSA: $\text{len}(n) < 2048$	Disallowed
	<u>≥ 112 bits of security strength:</u> DSA: $(L, N) = (2048, 224), (2048, 256)$ or $(3072, 256)$ ECDSA or EdDSA: $\text{len}(n) \geq 224$ RSA: $\text{len}(n) \geq 2048$	Acceptable 
Digital Signature Verification	< 112 bits of security strength: DSA ³² : $((512 \leq L < 2048) \text{ or } (160 \leq N < 224))$ ECDSA: $160 \leq \text{len}(n) < 224$ RSA: $1024 \leq \text{len}(n) < 2048$	Legacy use
	<u>≥ 112 bits of security strength:</u> DSA: $(L, N) = (2048, 224), (2048, 256)$ or $(3072, 256)$ ECDSA and EdDSA: $\text{len}(n) \geq 224$ RSA: $\text{len}(n) \geq 2048$	Acceptable 

Summary

- Asymmetric Ciphers
- Digital Signatures