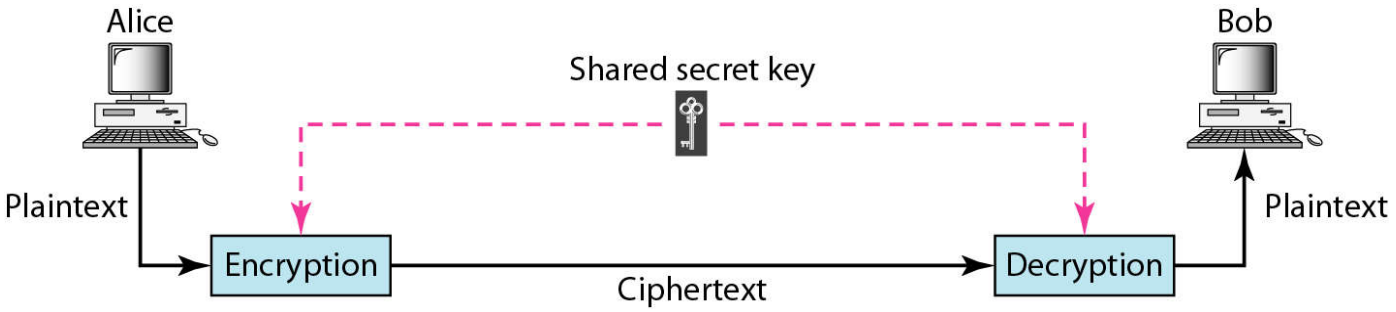
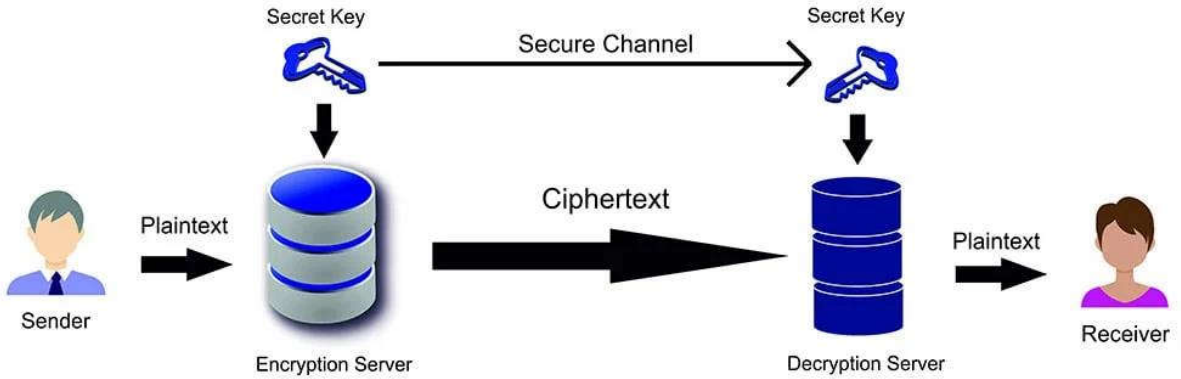
Outline

- Asymmetric Cryptography
- RSA Algorithm
- Digital Signatures

Symmetric-key cryptography



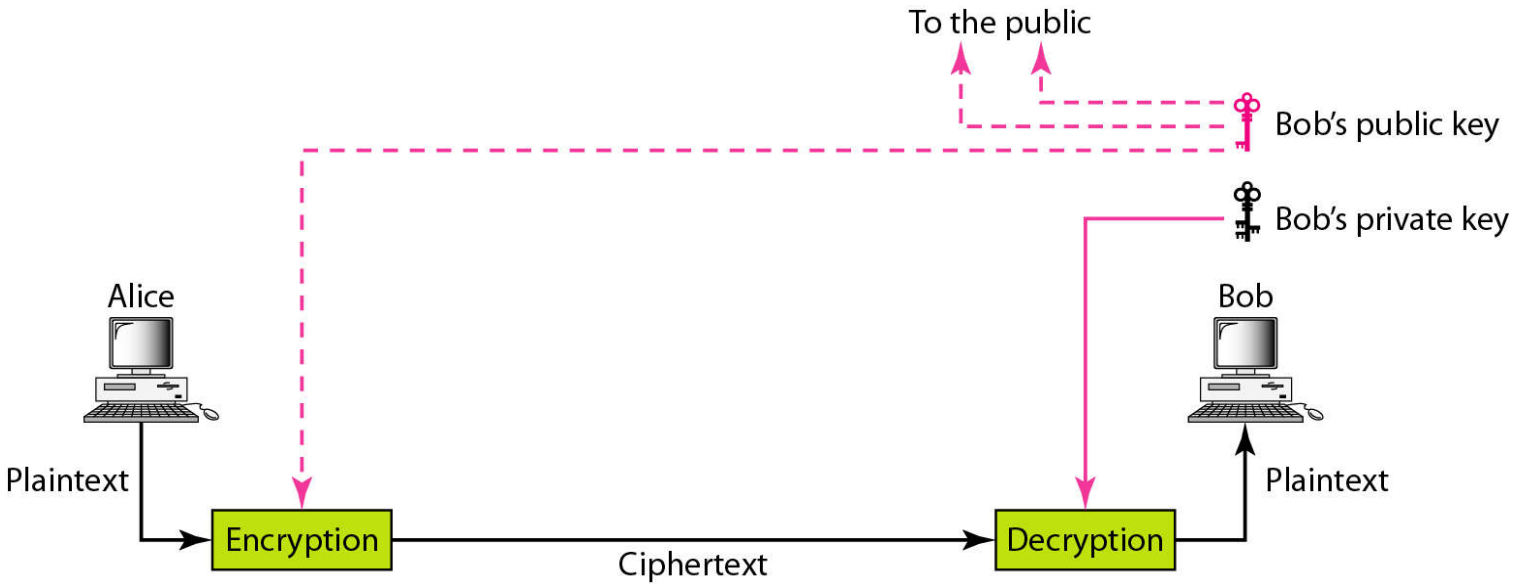


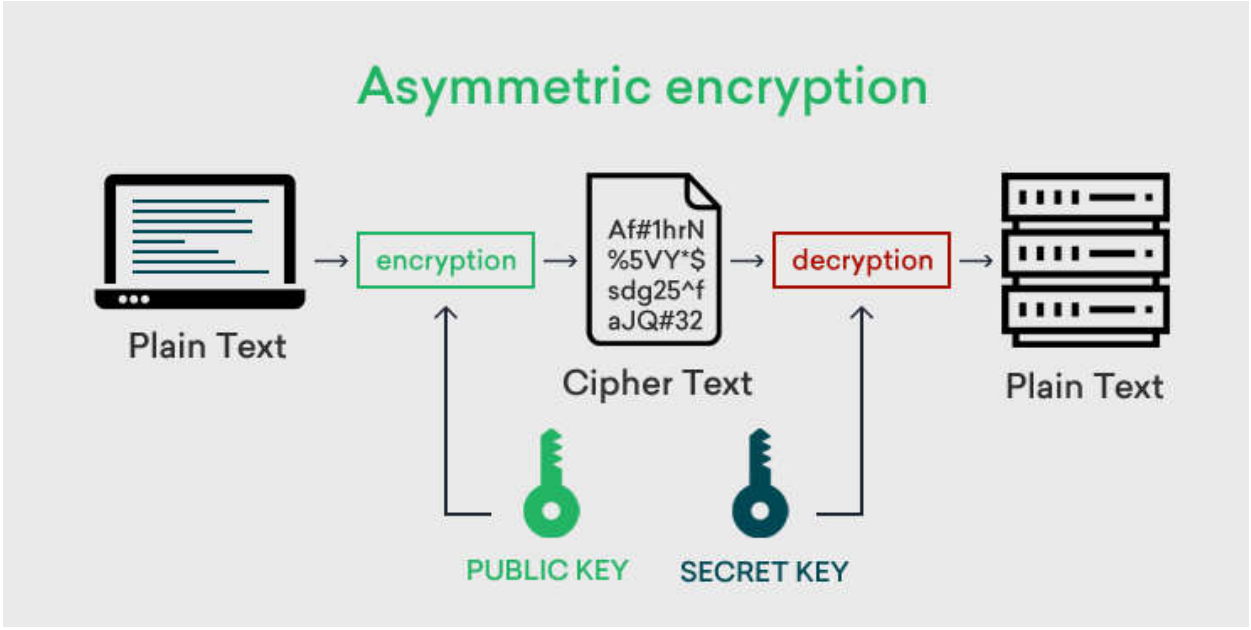
Symmetric Cryptography

Symmetric-key cryptography

- In symmetric-key cryptography, the same key is used by the sender(for encryption) and the receiver (for decryption). The key is shared.
- Algorithm: DES,AES
- **Advantages:**
  - Simple
  - Faster
- **Disadvantages:**
  - Key must be exchanged in a secure way.
  - Easy for hacker to get a key as it is passed in unsecure way.

Asymmetric-key cryptography





Asymmetric-key cryptography

- An asymmetric-key (or public-key) cipher uses two keys: one private (To encrypt data) and one public(To decrypt data).

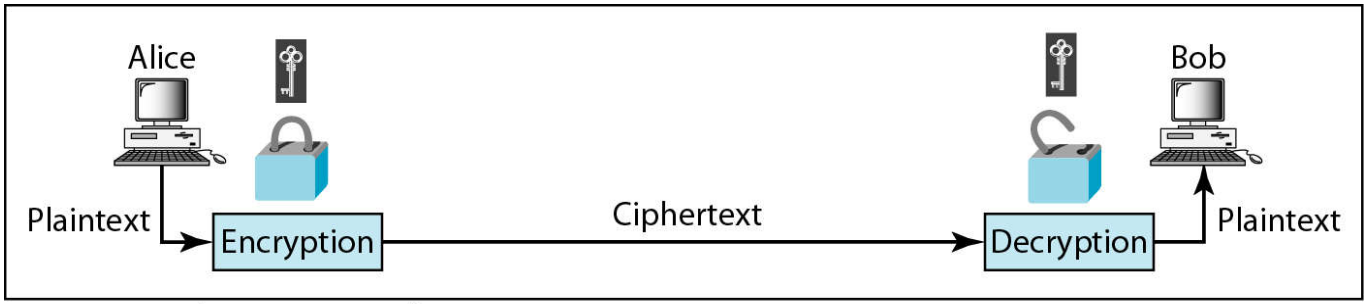
- **Advantages**
  - More Secured
  - Authentication
- **Disadvantages**
  - Relatively Complex
  - Time consuming process for Encryption and Decryption.

Asymmetric-key cryptography

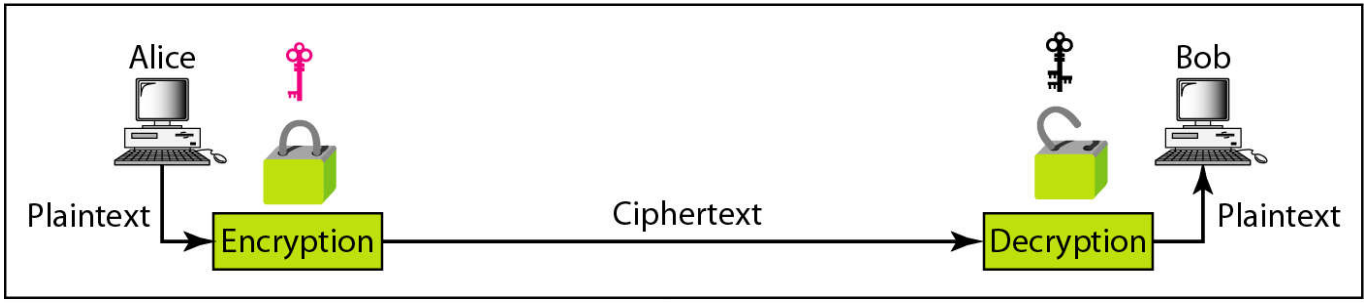
- Two parties don't need to have their private keys already shared in order to communicate using encryption.

- Authentication and Non-Repudiation are possible. (Authentication means that you can encrypt the message with my public key and only I can decrypt it with my private key. Non-repudiation means that you can "sign" the message with your private key and I can verify that it came from you with your public key.)

Comparison between two categories of cryptography



a. Symmetric-key cryptography



b. Asymmetric-key cryptography

Asymmetric-key cryptography

- Asymmetric encryption use two keys, one to encrypt the data, and another key to decrypt the data.
- These keys are generated together.
- . One is named as Public key and is distributed freely.
- The other is named as Private Key and it is kept hidden.
- Both Sender & Recipient has to share their Public Keys for Encryption and has to use their Private Keys for Decryption.

RSA Cryptosystem

- The RSA algorithm (Rivest-Shamir-Adleman), was first publicly described in 1977 by Ron Rivest, Adi Shamir and Leonard Adleman of the Massachusetts Institute of Technology.

- The RSA algorithm is the basis of a cryptosystem -- a suite of cryptographic algorithms that are used for specific security services or purposes -- which enables public key encryption and is widely used to secure sensitive data, particularly when it is being sent over an insecure network such as the internet.

RSA Cryptosystem

- In RSA cryptography, both the public and the private keys can encrypt a message.
- The opposite key from the one used to encrypt a message is used to decrypt it.
- This attribute is one reason why RSA has become the most widely used asymmetric algorithm.
- It provides a method to assure the confidentiality, integrity, authenticity, and non-repudiation of electronic communications and data storage.

RSA Cryptosystem

- Many protocols, including Secure Shell (SSH), OpenPGP, S/MIME, and SSL/TLS, rely on RSA for encryption and digital signature functions.
- It is also used in software programs -- browsers are an obvious example, as they need to establish a secure connection over an insecure network, like the internet, or validate a digital signature.
- RSA **signature verification** is one of the most commonly performed operations in network-connected systems.

RSA Cryptosystem

- A client (for example browser) sends its public key to the server and requests some data.
- The server encrypts the data using the client's public key and sends the encrypted data.
- The client receives this data and decrypts it.

Why is RSA used?

- RSA derives its security from the difficulty of factoring large integers that are the product of two large prime numbers.

- Multiplying these two numbers is easy, but determining the original prime numbers from the total -- or factoring -- is considered infeasible due to the time it would take using even today's supercomputers.

Why is RSA used?

- The public and private key generation algorithm is the most complex part of RSA cryptography.
- Two large prime numbers, x and y, are generated using the Rabin-Miller primality test algorithm.
- A modulus, n, is calculated by multiplying x and y.
- This number is used by both the public and private keys and provides the link between them. Its length, usually expressed in bits, is called the key length.

How RSA works?

1. Generating the keys

1. Select two large prime numbers, x and y. The prime numbers need to be large so that they will be difficult for someone to figure out.
2. Calculate $n = x \text{ x } y$.
3. Calculate the ***totient*** function; $\phi(n) = (x-1)(y-1)$.
4. Select an integer e, such that e is ***co-prime*** to $\phi(n)$ and $1 < e < \phi(n)$. The pair of numbers (n, e) makes up the public key.
5. Calculate d such that $e.d = 1 \; mod \; \phi(n)$.

d can be found using the ***extended euclidean algorithm***. The pair (n, d) makes up the private key.

The Public Key is a value which must match three requirements:

- It must be Prime
- It must be less than the Totient
- It must NOT be a factor of the Totient

How RSA works?

2. Encryption

Given a plaintext P, represented as a number, the ciphertext C is calculated as:

$$C = P^e \bmod n.$$

3. Decryption

Using the private key (n, d), the plaintext can be found using:

$$P = C^d \bmod n.$$

How RSA works? Example

1. Select x=11, y=13
2. Calculate n = x*y = 11x13 = 143
3. Calculate totient = (x-1)(y-1) = 10x12=120
4. Let e=7 such that public key is (143,7)
5. Calculate d = 103 such that private key is (143,103)

How RSA works? Example

1. Encryption:
   Let P = 9
   $9^7 \bmod 143 = 48$
2. Decryption:
   C = 48
   $48^{103} \bmod 143 = 9$

How RSA works?

ALICE	BOB
Chooses two prime numbers (p and q) NOTE: $p \neq q$	
Calculates modulus (n) $n = p \times q$	
Calculates totient of mod n ($\varphi(n)$) $\varphi(n) = (p-1)x(q-1)$	
Chooses public key (e) NOTE: e MUST BE LESS THAN AND COPRIME WITH $\varphi(n)$	
Calculates private key (d) using Extended Euclidean algorithm NOTE: d IS THE MULTIPLICATIVE INVERSE OF e MOD $\varphi(n)$	
Releases (e, n) as public key, retains (d, n) as private key	
	Bob wants to send Alice a message (M)
	Bob obtains Alice's public key (e, n)
	Bob encrypts M into ciphertext (C): $C = M^e \bmod n$
Alice decrypts M using her private key (d, n) $M = C^d \bmod n$	

How RSA works? Example

For example, suppose the receiver selected the primes $p = 11$ and $q = 17$, along with $e = 3$.

1. The receiver calculates $n = pq = 11 \cdot 17 = 187$, which is half of the public key.
2. The receiver also calculates $\phi(n) = (p-1)(q-1) = 10 \cdot 16 = 160$. $e = 3$ was also chosen.
3. The receiver calculates $d = 107$, since then $de = 321 \equiv 1 \pmod{\phi(n)}$ $(\text{since } \phi(n) = 160)$.
4. The receiver distributes his public key: $n = 187$ and $e = 3$.

Now suppose the sender wanted to send the message "HELLO".

1. 'H' is 72 in ASCII, so the message text is $m = 72$.
2. The sender calculates $m^e = 72^3 \equiv 183 \pmod{187}$, making the ciphertext $c = 183$.
   not have the private key.
3. The receiver calculates $c^d = 183^{107} \equiv 72 \pmod{187}$, thus getting the message of $m = 72$.
4. The receiver translates 72 into 'H'.

Digital Signatures

- A digital signature is a mathematical technique used to validate the authenticity and integrity of a digital document, message or software.
- It's the digital equivalent of a handwritten signature or stamped seal, but it offers far more inherent security.
- A digital signature is intended to solve the problem of tampering and impersonation in digital communications.

Digital Signatures

- Digital signatures can provide evidence of origin, identity and status of electronic documents, transactions or digital messages.

- In many countries, digital signatures are considered legally binding in the same way as traditional handwritten document signatures.

How do digital signatures work?

- Digital signatures are based on public key cryptography, also known as asymmetric cryptography.

- Using a public key algorithm -- such as Rivest-Shamir-Adleman, or RSA -- two keys are generated, creating a mathematically linked pair of keys: one private and one public.

How do digital signatures work?

- For encryption and decryption, the person who creates the digital signature uses a private key to encrypt signature-related data. The only way to decrypt that data is with the signer's public key.

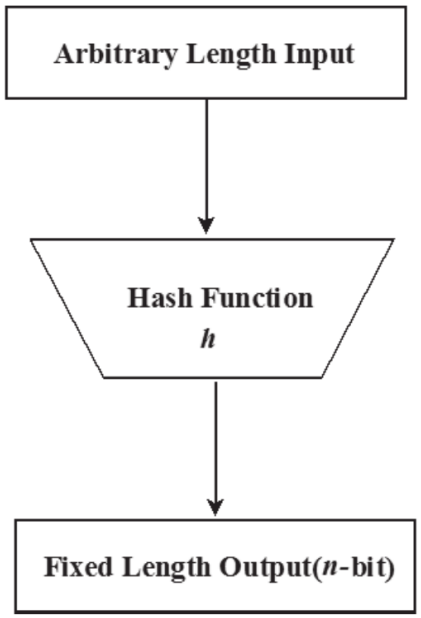
- If the recipient can't open the document with the signer's public key, that indicates there's a problem with the document or the signature. This is how digital signatures are authenticated

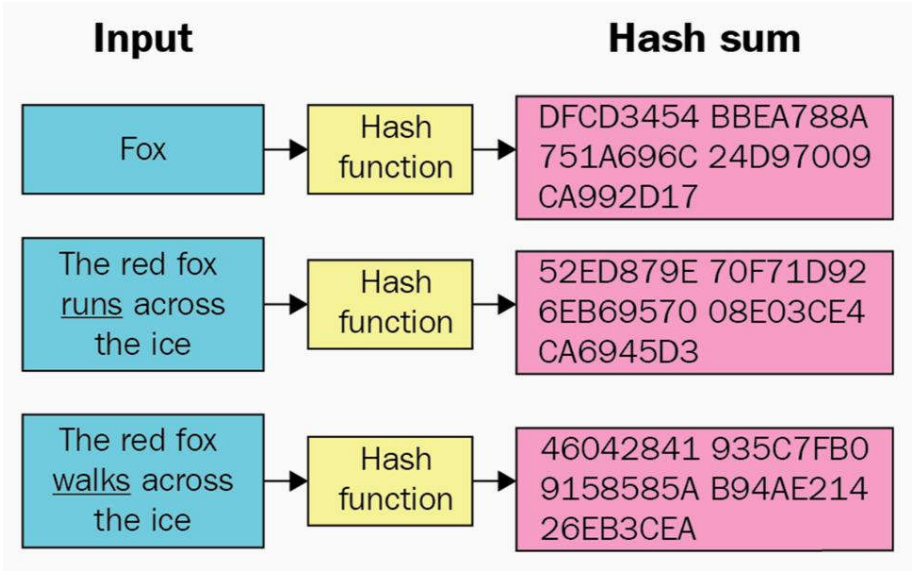
Digital Certificates

- Digital certificates, also called public key certificates, are used to verify that the public key belongs to the issuer.
- Digital certificates contain the public key, information about its owner, expiration dates and the digital signature of the certificate's issuer.
- Digital certificates are issued by trusted third-party certificate authorities (CAs), such as DocuSign or GlobalSign.
- The party sending the document and the person signing it must agree to use a given CA.

Hash Function

- A hash function is a mathematical function or algorithm that simply takes a **variable number of characters** (called a ”message”) and converts it into a string with a **fixed number of characters** (called a hash value or simply, a hash).
- The values returned by a hash function are called hash values, hash codes, digests, or simply hashes.





Modular Hash Function

- $h(k) = k \bmod m$
- m is generally a prime number.

Cryptographic Hash Function

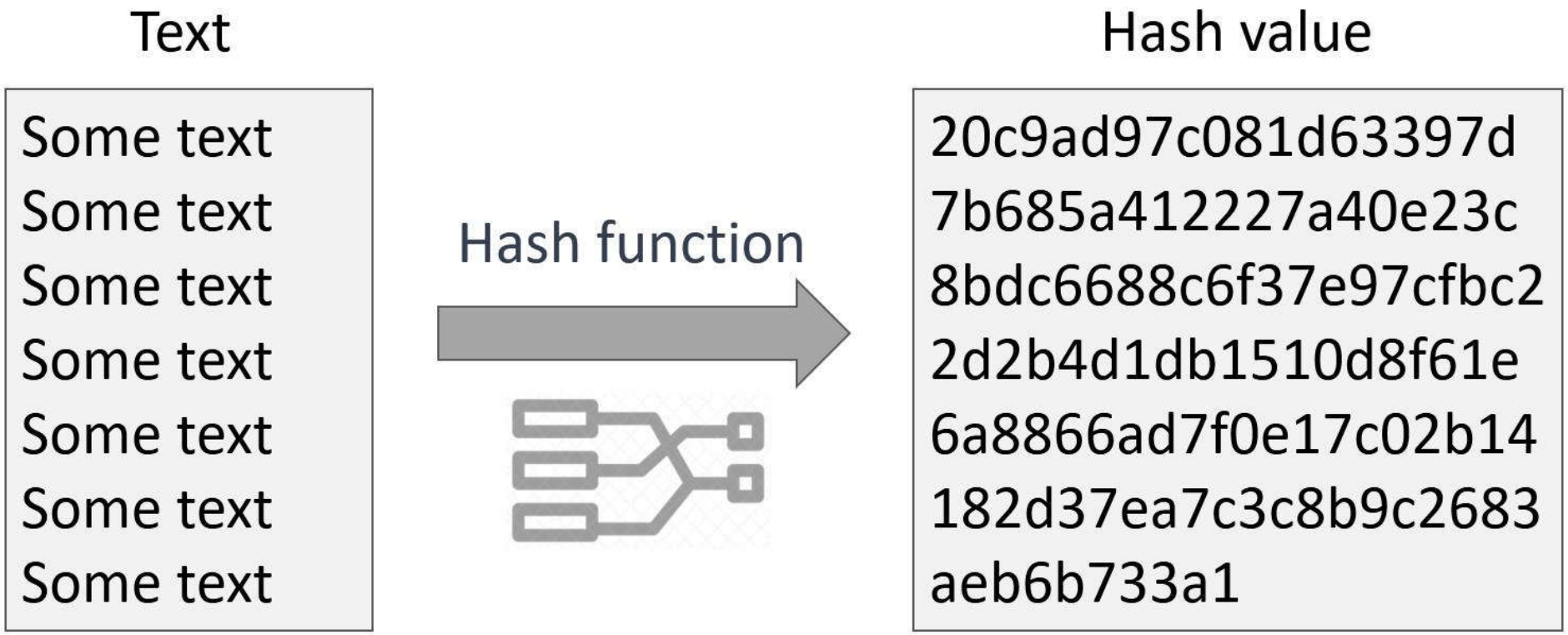
- Hash functions (hashing algorithms) used in computer cryptography are known as "cryptographic hash functions.

- In cryptography, hash functions transform input data of arbitrary size (e.g., a text message) to a result of fixed size (e.g., 256 bits), which is called hash value (or hash code, message **digest**, or simply hash).

- Examples of such functions are SHA-256 and SHA3-256, which transform arbitrary input to 256-bit output

SHA-256("hello") = "2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824"

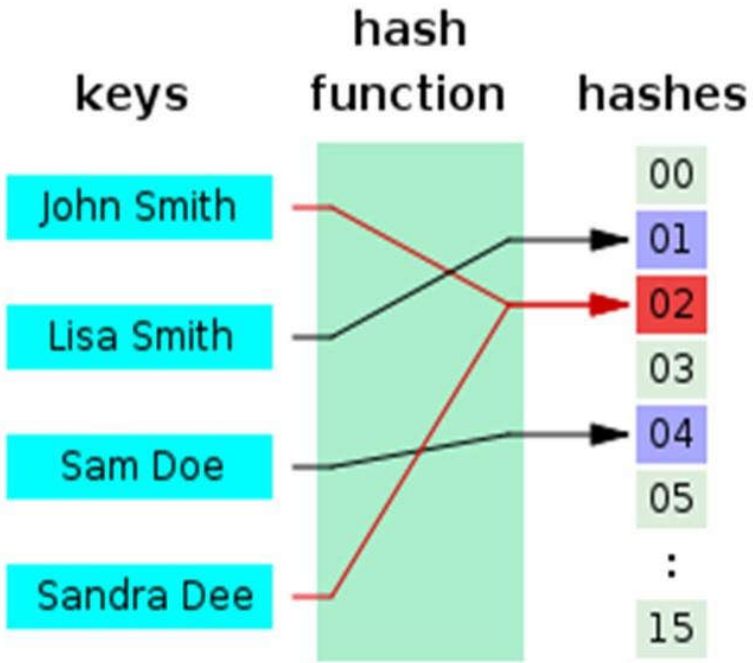
Cryptographic Hash Function



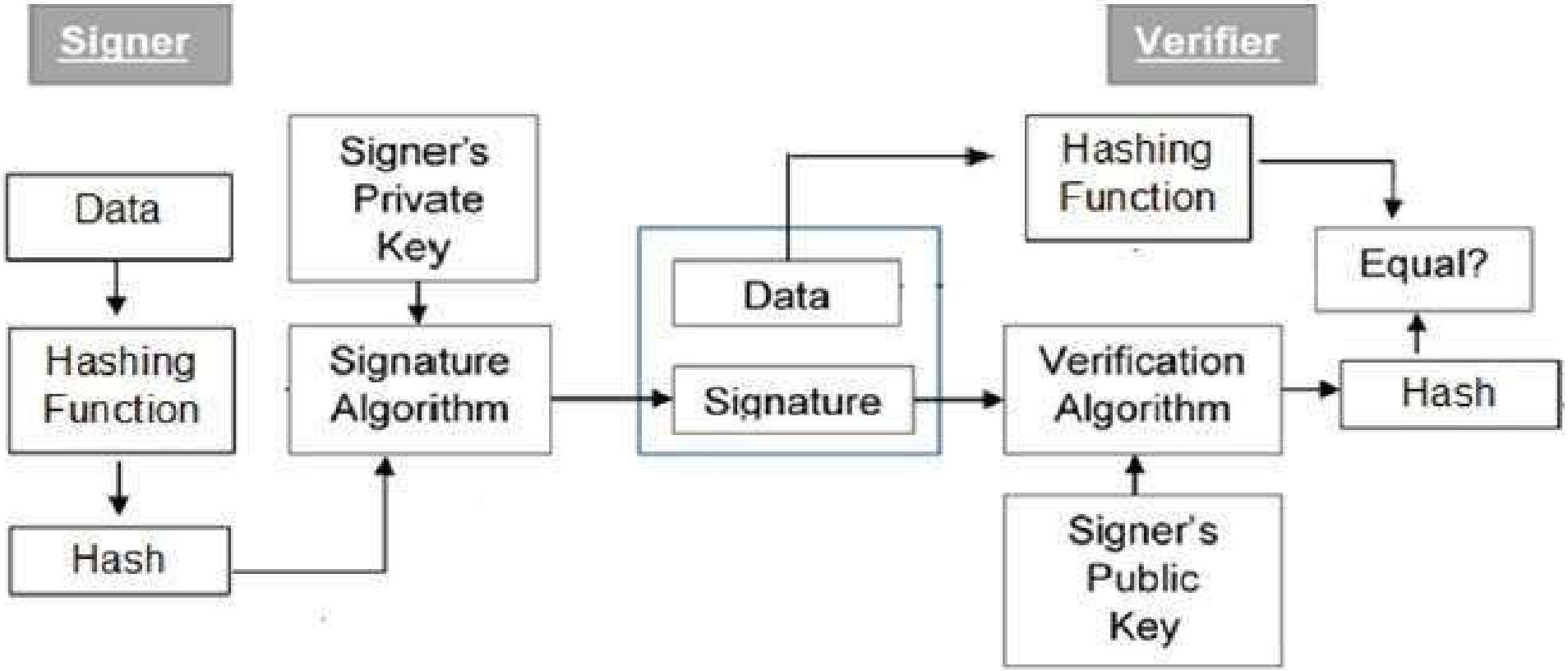
Cryptographic Hash Function

- Hash functions are known to be **collision-resistant** and **irreversible**.

- Hash functions are irreversible by design, which means that there is no fast algorithm to restore the input message from its hash value.



Model of Digital Signature



Model of Digital Signature

- Each person adopting this scheme has a public-private key pair.
- Generally, the key pairs used for encryption/decryption and signing/verifying are different. The private key used for signing is referred to as the signature key and the public key as the verification key.
- Signer feeds data to the hash function and generates hash of data.
- Hash value and signature key are then fed to the signature algorithm which produces the digital signature on given hash. Signature is appended to the data and then both are sent to the verifier.

Model of Digital Signature

- Verifier feeds the digital signature and the verification key into the verification algorithm. The verification algorithm gives some value as output.
- Verifier also runs same hash function on received data to generate hash value.
- For verification, this hash value and output of verification algorithm are compared. Based on the comparison result, verifier decides whether the digital signature is valid.
- Since digital signature is created by ‘private’ key of signer and no one else can have this key; the signer cannot repudiate signing the data in future.

Digital Signature Algorithm (DSA)

- The Digital Signature Algorithm (DSA) is a widely-used asymmetric cryptographic algorithm for digital signatures.

- DSA is a part of the Digital Signature Standard (DSS) established by the U.S. National Institute of Standards and Technology (NIST) to provide secure digital signatures for electronic documents and data.

DSA Steps

1. Parameter Generation
2. Key Pair Generation
3. Signature Generation
4. Signature Verification

DSA: Parameter Generation

- Find a prime number p such that 2^L-1 < p < 2^L, where L is an integer between 512 and 1024 i.e., 512 <= L <= 1024.

- Find another number q which is a prime divisor of (p-1).

- Compute g = h^(p-1)/q mod p, where h is an integer between 1 < h < p-1 and g should be greater than 1 or h^(p-1)/q mod p>1.

$$1 < g < p, g^{**}q \bmod p = 1 \text{ and } g = h^{**}((p-1)/q) \bmod p$$

DSA: Key Generation

- The private key, **a** is any random number such that $0 < a < q$.

- The public key, $A = g^a \bmod p$

DSA: Signature Generation

Algorithm: DSA Signature Generation

INPUT: Domain parameters (p, q, g); signer's private key a; message-to-be-signed, M; a secure hash function $\mathrm{Hash}()$ with output of length $|q|$.
OUTPUT: Signature (r, s).

1. Choose a random k in the range $[1, q-1]$.
2. Compute $X = g^k \bmod p$ and $r = X \bmod q$. If $r = 0$ (unlikely) then go to step 1.
3. Compute $k^{-1} \bmod q$.
4. Compute $h = \mathrm{Hash}(M)$ interpreted as an integer in the range $0 \leq h < q$.
5. Compute $s = k^{-1}(h + ar) \bmod q$. If $s = 0$ (unlikely) then go to step 1.
6. Return (r, s).

DSA: Signature Verification

Algorithm: DSA Signature Verification

INPUT: Domain parameters (p, q, g); signer's public key A; signed-message, M; a secure hash function $\mathrm{Hash}()$ with output of length $|q|$; signature (r, s) to be verified.
OUTPUT: "Accept" or "Reject".

1. Verify that r and s are in the range $[1, q-1]$. If not then return "Reject" and stop.
2. Compute $w = s^{-1} \bmod q$.
3. Compute $h = \mathrm{Hash}(M)$ interpreted as an integer in the range $0 \leq h < q$.
4. Compute $u_1 = hw \bmod q$ and $u_2 = rw \bmod q$.
5. Compute $X = g^{u_1} A^{u_2} \bmod p$ and $v = X \bmod q$.
6. If $v = r$ then return "Accept" otherwise return "Reject".

DSA Example: Signature Generation

INPUT: Domain parameters $(p = 283, q = 47, g = 60)$
INPUT: Alice's private key, $a = 24$
INPUT: Message M with message digest $h = \mathrm{Hash}(M) = 41$.

1. Alice chooses a random $k = 15$ in the range $[1, q - 1]$
2. Alice computes $X = g^k \bmod p = 60^{15} \bmod 283 = 207$ and
   $r = X \bmod q = 207 \bmod 47 = 19$.
   $r \neq 0$ so continue.
3. Alice computes $k^{-1} \bmod q = 15^{-1} \bmod 47 = 22$.
4. Alice computes $h = \mathrm{Hash}(M) = 41$.
5. Alice computes $s = k^{-1}(h + ar) \bmod q = 22(41 + 24 \cdot 19) \bmod 47 = 30$.
   $s \neq 0$ so continue.
6. Alice issues the message M and signature $(r, s) = (19, 30)$.

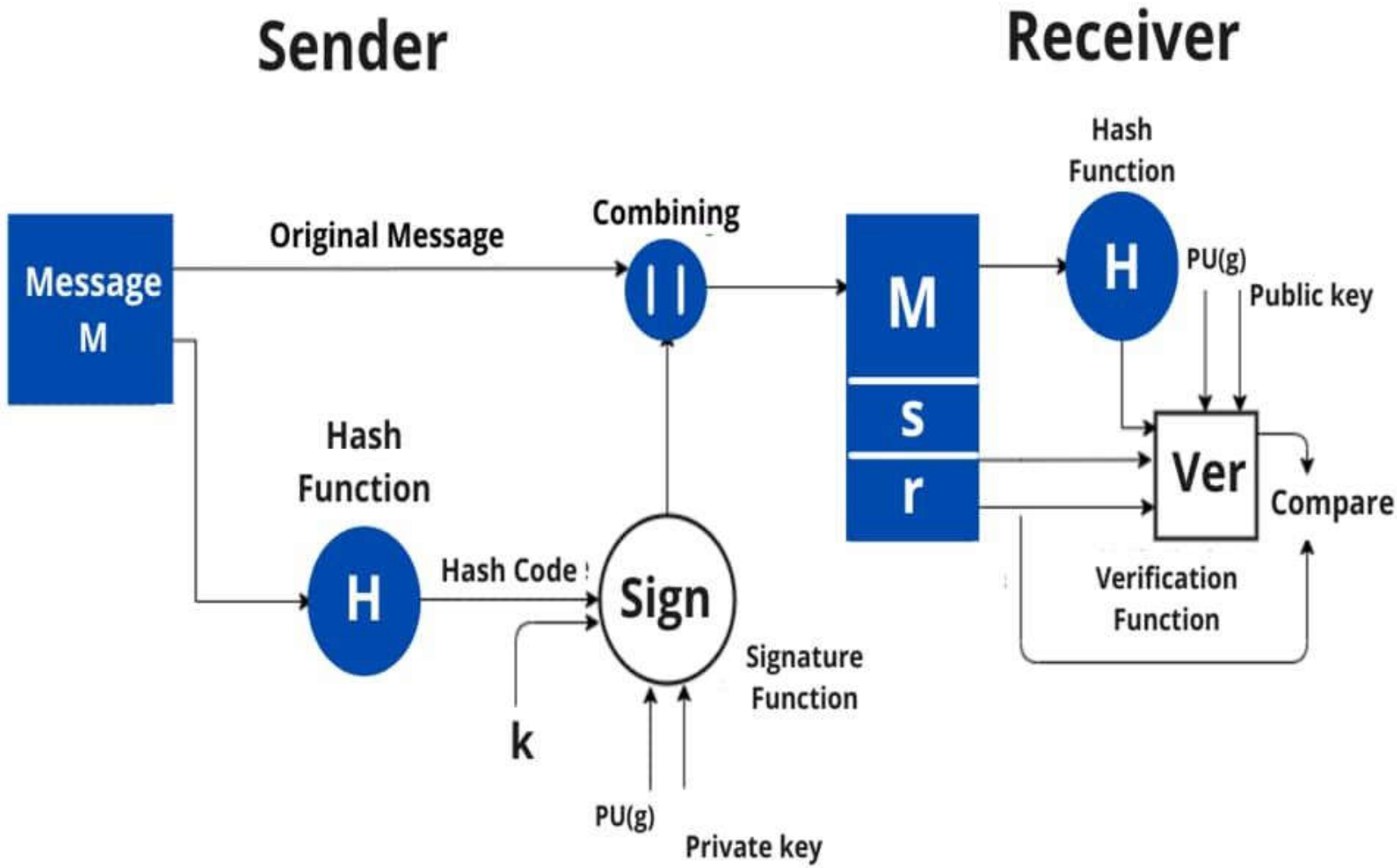
DSA Example: Signature Verification

INPUT: Domain parameters $(p = 283, q = 47, g = 60)$
INPUT: Alice's public key, $A = 158$
INPUT: Message M with message digest $h = \mathrm{Hash}(M) = 41$.
INPUT: Signature $(r, s) = (19, 30)$.

1. Bob verifies that $0 < r = 19 < 47$ and $0 < s = 30 < 47 \Rightarrow$ OK, so continue.
2. Bob computes $w = s^{-1} \bmod q = 30^{-1} \bmod 47 = 11$.
3. Bob computes $h = \mathrm{Hash}(M) = 41$.
4. Bob computes $u_1 = hw \bmod q = 41 \cdot 11 \bmod 47 = 28$ and
   $u_2 = rw \bmod q = 19 \cdot 11 \bmod 47 = 21$.
5. Bob computes $X = g^{u_1} A^{u_2} \bmod p = 60^{28} \cdot 158^{21} \bmod 283 = 106 \cdot 42 \bmod 283 = 207$ and
   $v = X \bmod q = 207 \bmod 47 = 19$.
6. Bob checks that $v = 19 = r$, so he accepts the signature.

Source: https://www.di-mgt.com.au/public-key-crypto-discrete-logs-4-dsa.html

DSA model


Sender
Receiver
Message
M
Original Message
Combining
||
Hash
Function
H
Hash Code
Sign
k
PU(g)
Private key
Signature
Function
M
s
r
Hash
Function
H
PU(g)
Public key
Ver
Compare
Verification
Function

RSA vs. DSA

- **RSA:** Primarily used for encryption, digital signatures, and key exchange. RSA's versatility makes it suitable for various cryptographic applications.

- **DSA:** Specifically designed for digital signatures. It's not intended for encryption or key exchange, but it excels in providing data integrity and authentication through signatures.

RSA vs. DSA

- **RSA:** In RSA, you generate a key pair consisting of a public key and a private key. The security of RSA relies on the difficulty of factoring the product of two large prime numbers.

- **DSA:** DSA key pairs involve a prime number 'p', a subprime 'q', a generator 'g', and a private key 'x' that needs to be generated. The public key 'y' is calculated based on 'x' and the other parameters.

RSA vs. DSA

- **RSA:** Security in RSA depends on the size of the modulus (key length), with longer keys offering higher security. Common key lengths range from 1024 to 4096 bits.

- **DSA:** DSA uses specific prime numbers 'p' and 'q' along with the choice of 'g'. The security depends on the proper selection of these domain parameters.

RSA vs. DSA

- **RSA:** Security in RSA depends on the size of the modulus (key length), with longer keys offering higher security. Common key lengths range from 1024 to 4096 bits.

- **DSA:** DSA uses specific prime numbers 'p' and 'q' along with the choice of 'g'. The security depends on the proper selection of these domain parameters.

Summary

- Asymmetric Cryptography
- RSA