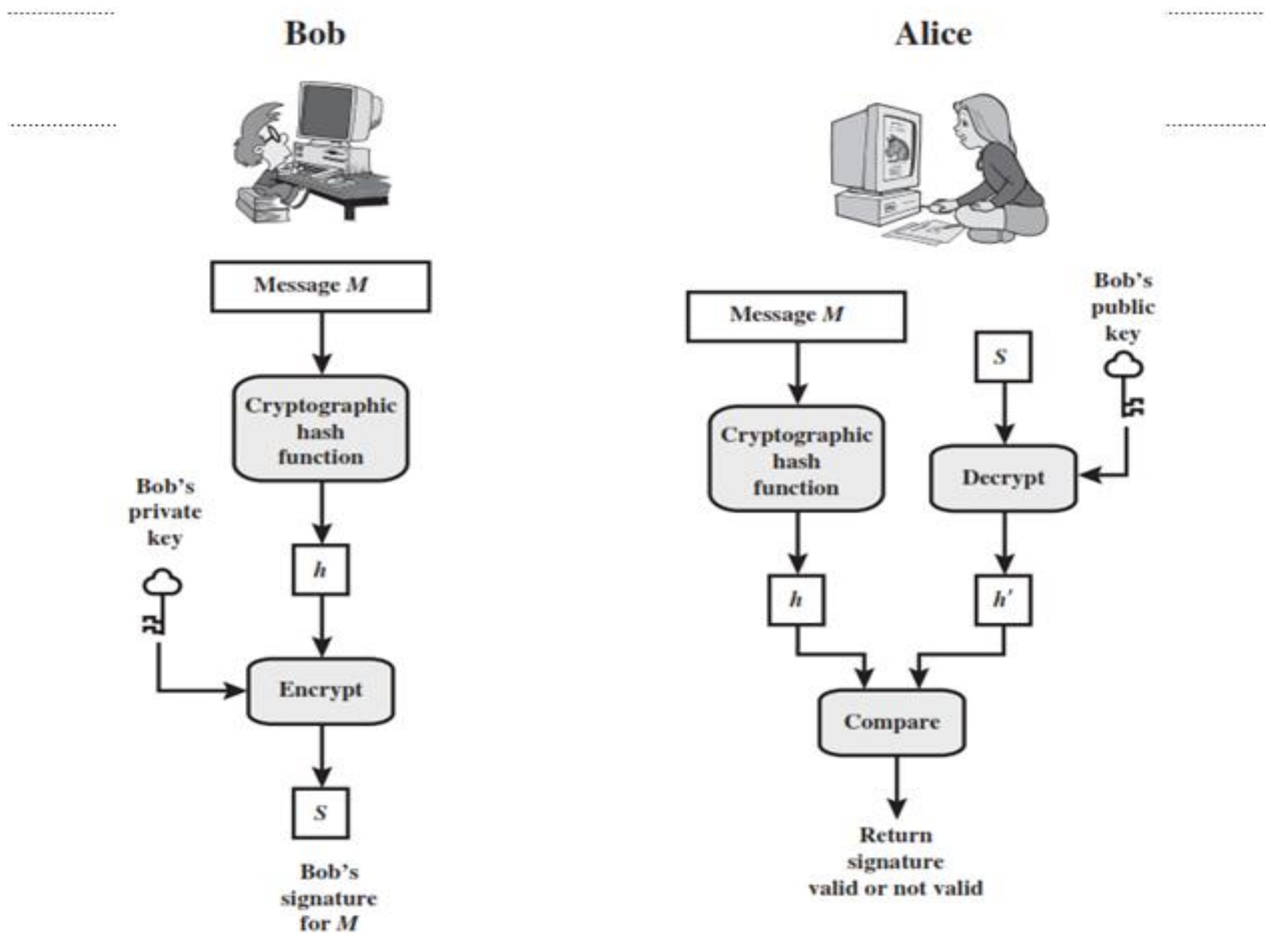
Outline

- Digital Signature
- Digital Signature properties
- Requirements and security
- Various digital signature schemes (Elgamal and Schnorr)
- Digital Signature algorithm / Digital Signature Standard

Digital Signature

- A **digital signature** is an authentication mechanism that enables the creator of a message to attach a code that acts as a **signature**.
- Typically the **signature** is formed by taking the hash of the message and encrypting the message with the creator's private key.
- The **signature** guarantees the source and integrity of the message.
- The **digital signature standard (DSS)** is an NIST standard that uses the secure hash algorithm (SHA).


Bob
Alice
Message M
Cryptographic hash function
h
Bob's private key
Encrypt
S
Bob's signature for M
Message M
S
Bob's public key
Cryptographic hash function
Decrypt
h
h'
Compare
Return signature valid or not valid

Digital Signature Requirements

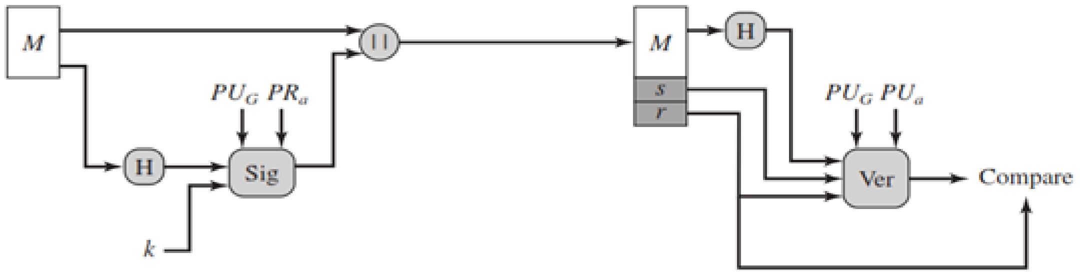
1. The signature must be a **bit pattern** that depends on the message being signed.
2. The signature must use some information **unique** to the sender to prevent both forgery and denial.
3. It must be relatively **easy to produce** the digital signature.
4. It must be relatively **easy to recognize** and **verify** the digital signature.
5. It must be computationally **infeasible to forge** a digital signature, either by constructing a new message for an existing digital signature or by constructing a fraudulent digital signature for a given message.
6. It must be **practical to retain a copy** of the digital signature in storage.

Digital Signature Standard / DSA

- The **DSS** uses an algorithm that is designed to provide only the digital signature function.
- Unlike RSA, it cannot be used for encryption or key exchange.

DSA Approach

- The **hash code** is provided as input to a **signature function** along with a random number ***k*** generated for this particular signature.
- The signature function also depends on the sender's private key **(*PRa*)** and a set of parameters known to a group of communicating principals.
- We can consider this set to constitute a global public key **(*PU*)**
- The result is a signature consisting of two components, labelled ***s*** and ***r***.



(b) DSA approach

DSA Approach

- At the receiving end, the hash code of the incoming message is generated.
- This plus the signature is input to a **verification function**.
- The verification function also depends on the global public key as well as the sender's public key **(*PUa*)**, which is paired with the sender's private key.
- The output of the verification function is a value that is equal to the signature component ***r*** *if the signature is valid.*
- *The signature* function is such that only the sender, with knowledge of the private key, could have produced the valid signature.

Digital Signature Algorithm

Global Public-Key Components

p prime number where $2^{L-1} < p < 2^L$
for $512 \le L \le 1024$ and L a multiple of 64;
i.e., bit length of between 512 and 1024 bits
in increments of 64 bits

q prime divisor of $(p - 1)$, where $2^{N-1} < q < 2^N$
i.e., bit length of N bits

g $= h(p - 1)/q \bmod p$,
where h is any integer with $1 < h < (p - 1)$
such that $h^{(p-1)/q} \bmod p > 1$

Digital Signature Algorithm

User's Private Key

x random or pseudorandom integer with $0 < x < q$

User's Public Key

y $= g^x \bmod p$

User's Per-Message Secret Number

k random or pseudorandom integer with $0 < k < q$

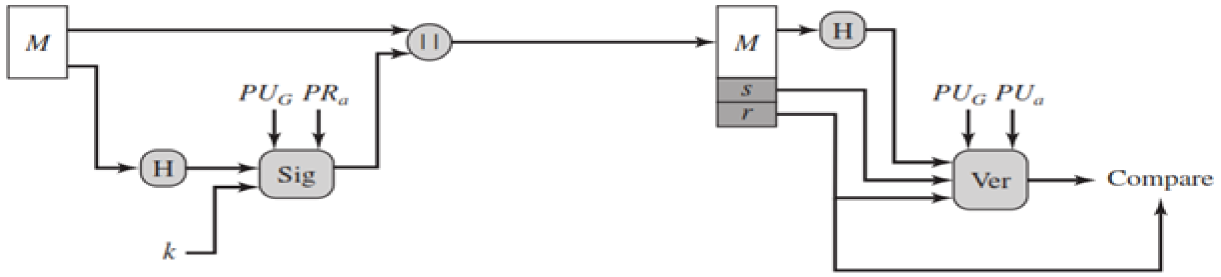
Digital Signature Algorithm

Signing

$r \quad = (g^k \bmod p) \bmod q$

$s \quad = [k^{-1}(H(M) + xr)] \bmod q$

$\text{Signature} = (r, s)$



Digital Signature Algorithm

Verifying

$w = (s')^{-1} \bmod q$

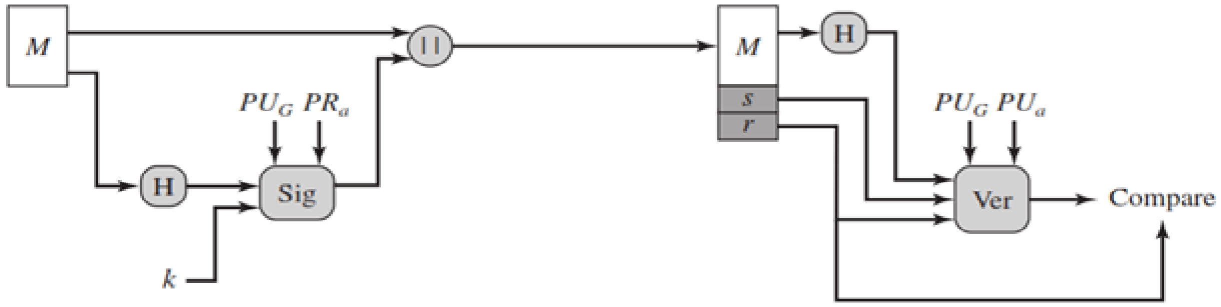
$u_1 = [H(M')w] \bmod q$

$u_2 = (r')w \bmod q$

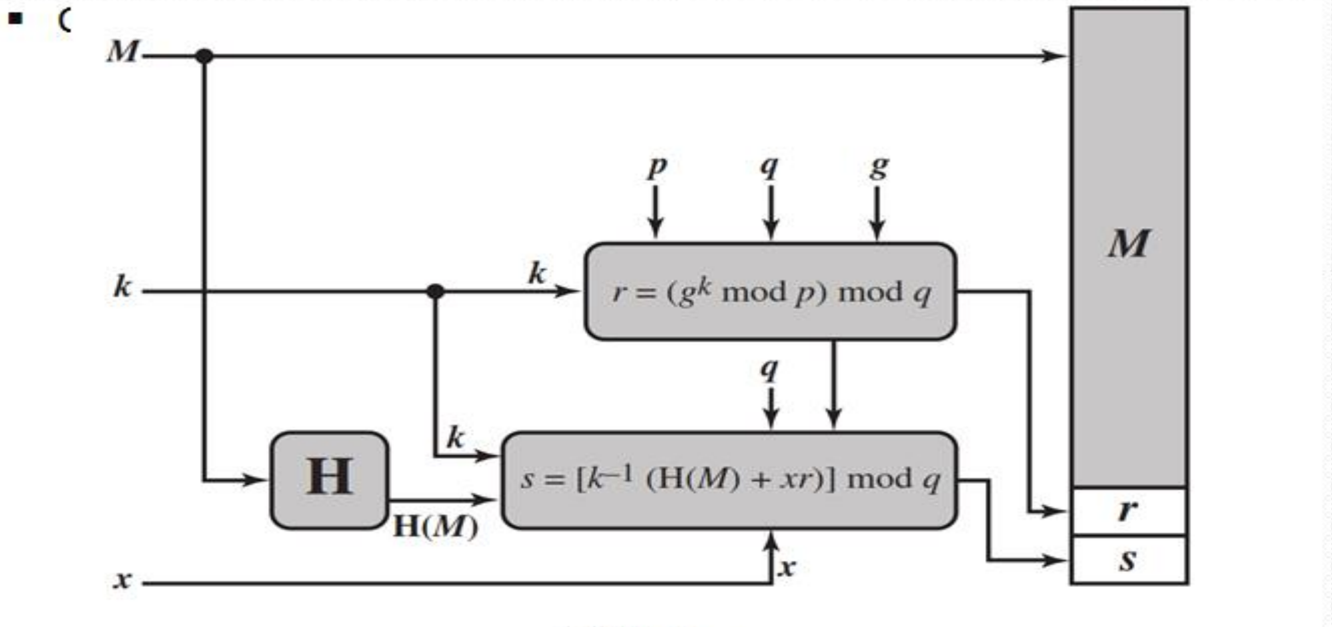
$v = [(g^{u1}\, y^{u2}) \bmod p] \bmod q$

$\text{TEST: } v = r'$

M	= message to be signed
$H(M)$	= hash of M using SHA-1
M', r', s'	= received versions of M, r, s

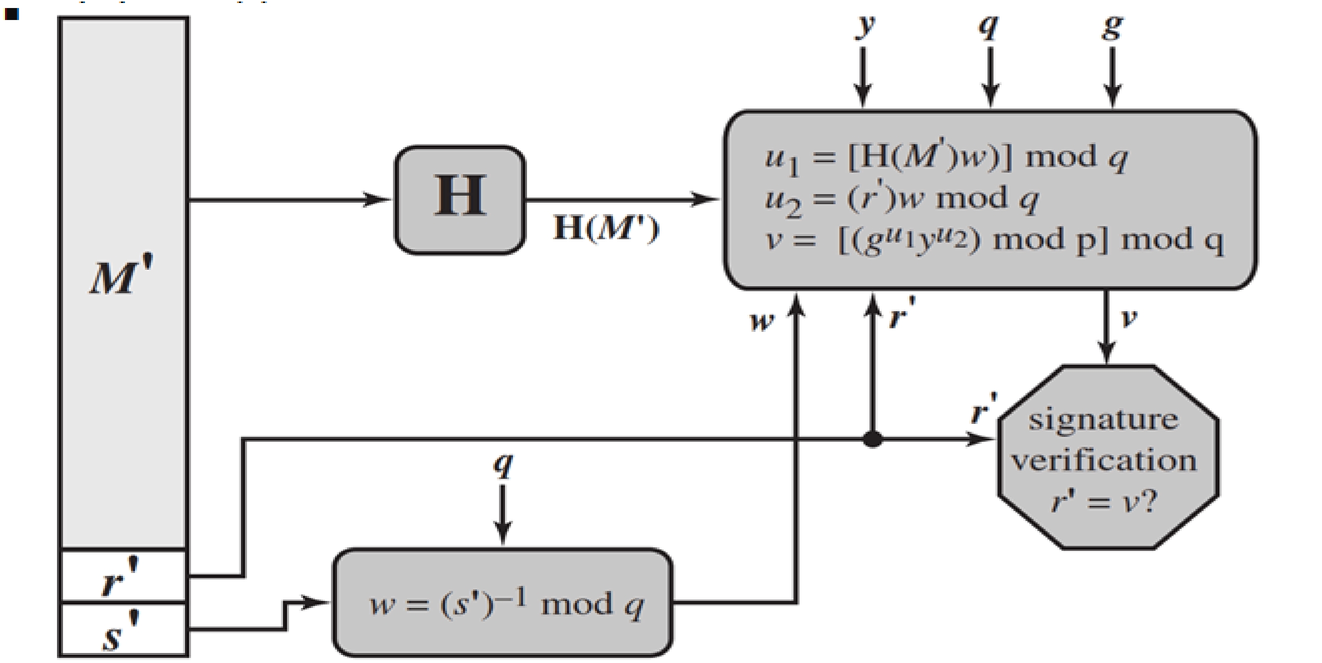


DSA Signing



(a) Signing

DSA Verifying



Schnorr Digital Signature

In cryptography, a **Schnorr signature** is a digital signature produced by the Schnorr signature algorithm that was described by Claus Schnorr. It is a digital signature scheme known for its simplicity, is efficient and generates short signatures. The Verifier should be convinced that they are communicating with the Prover without knowing the private key.

Schnorr Digital Signature to implement Zero Knowledge Proof **:**
Let's take an example of two friends Sachin and Sanchita. Sanchita has announced to the world that she has a public key and can accept and receive information through it. Sachin thinks that Sanchita is lying. Sanchita wants to prove her honesty without showing her private keys. Here is where Schnorr's protocol will help us.

```
Consider the following parameters:

p, q, a, s, v, r, x, y


where,

"p" is any prime number

"q" is factor of p-1

“a” such that a^q = 1 mod p
```

The above three variables are global and public which means anyone can see these three variables at a given scenario.

We will have two keys.

```
"s" is the secret key or the private key (0<s<q).

"v" is the public key = a^-s mod q.
```

The public key “v” will be global and public knowledge along with p, q and a. However only Sanchita will have the knowledge of the private key "s".

Now Sanchita signs wants to sends an encrypted message "M". She will follow the following steps to use Schnorr's signature:-

1. She will first choose a random number “r” such that 0<r<q.
2. She will now compute a value X such that: X= a^r mod p.
3. Now that she has computed the value of X, she is going concatenate this with the original message (same as string concatenation).
   So, she is going to concatenate M and X to get M||X. and she is going to store the hash of this value in e.
   `e = H(M||X) where H() is the hash function`
4. She is going to get a value “y” such that:
   `y = (r + s*e) mod q`

Now that all the computations are over, she is going to send the following to Sachin.

1. The message “M”.
2. The signatures e and y.

Along with this, Sachin has the following public piece of information:-

1. Sanchita's public key “v”.
2. The prime number that Sanchita chose “p”.
3. “q” which is the factor of “p-1” which Sanchita chose.
4. “a” such that a^q = 1 mod p, chosen by Sanchita.

Now, Sachin will have to compute X’ such that:

```
X’ = a^y * v^e mod p
```

We know that v = a^-s, let’s substitute that in the equation above and we get:

```
X’ = a^y * a^-se = a ^ (y-s*e)
```

Now we also know that,

```
y = r + s*e


Which means:

r = y-s*e
```

Let’s substitute this value in the equation above:

```
We get: X’ = a^r
```

As we have already seen above:

```
X= a^r
```

So technically:

```
X = X’
```

But Sachin doesn’t know the value of “X” because he never received that value. All that he received are the following: The message M, the signatures (e and y) and the host of public variables (public key “v”, p, q, and a).

So he is going to solve for e by doing the following:

```
e = H ( M||X’)
```

Note that earlier we solved for e by doing:

```
H(M||X))
```

So, by that logic, if the two values of e come up to be the same then that means

```
X = X’
```

This follows all three Properties of Zero Knowledge Proof :

1. **Completeness –**
   Sachin was convinced of Sanchita’s honesty because at the end X = X’.
2. **Soundness –**
   The plan was sound because Sanchita only had one way to prove her honesty and that was through her private key.
3. **Zero Knowledge –**
   Sachin never got to know about Sanchita’s private key.

El-Gamal Digital Signature Scheme

El-gamal digital signature scheme:

- This scheme used the same keys but a different algorithm
- The algorithm creates two digital signatures, these two signatures, are used in the verification phase.
- The key generation process is the same as that of El-gamal algorithms
- The public key remains (e_1, e_2, p) and the private key continues to be d

Signature

This process works as follows

1. The sender selects a random number r
2. The sender computes the first signature s_1 using $s_1 = e_1^r mod p$
3. The sender computes the second signature s_2 using the equation

s_2 =$(M - dXs_1)Xr^{-1} mod(p-1)$

Where P= large prime number

M= original message that needs to be signed

1. The sender sends M, s_1, s_2 to the receiver.

For eg. Let e_1 = 10, e_2 = 4, p=19, M=14, d=16 & r=5

Then, $s_1 = e_1^R mod p$= $10^5 mod 19$= 3

$s_2 = (M - dXs_1)Xr^{-1}mod(p-1) = (14 - 16X3)X5^{-1}mod(18)$= 4

Then these signatures $s_1 and s_2$ are sent to the receiver.

Verification:

This process works as follows.

1.The receiver performs the 1st part of verification called v_1 using the equation

$v_1 = e_1^M mod P$

2.The receiver performs the 2nd part of verification called as v_2 using the equation

$v_2 = e_2^{s_1} s_1^{(s_2)} mod p$

Eg

$v_1 = e_1^M mod p$ = $10^{14} mod 19$ = 16

and

$v_2 = e_2^{s_1} s_1^{(s_2)} mod p = 4^3 X 3^4 mod p$ = 5184 mod 19 = 16

Since v_1=v_2, the signature is valid.