Cryptography

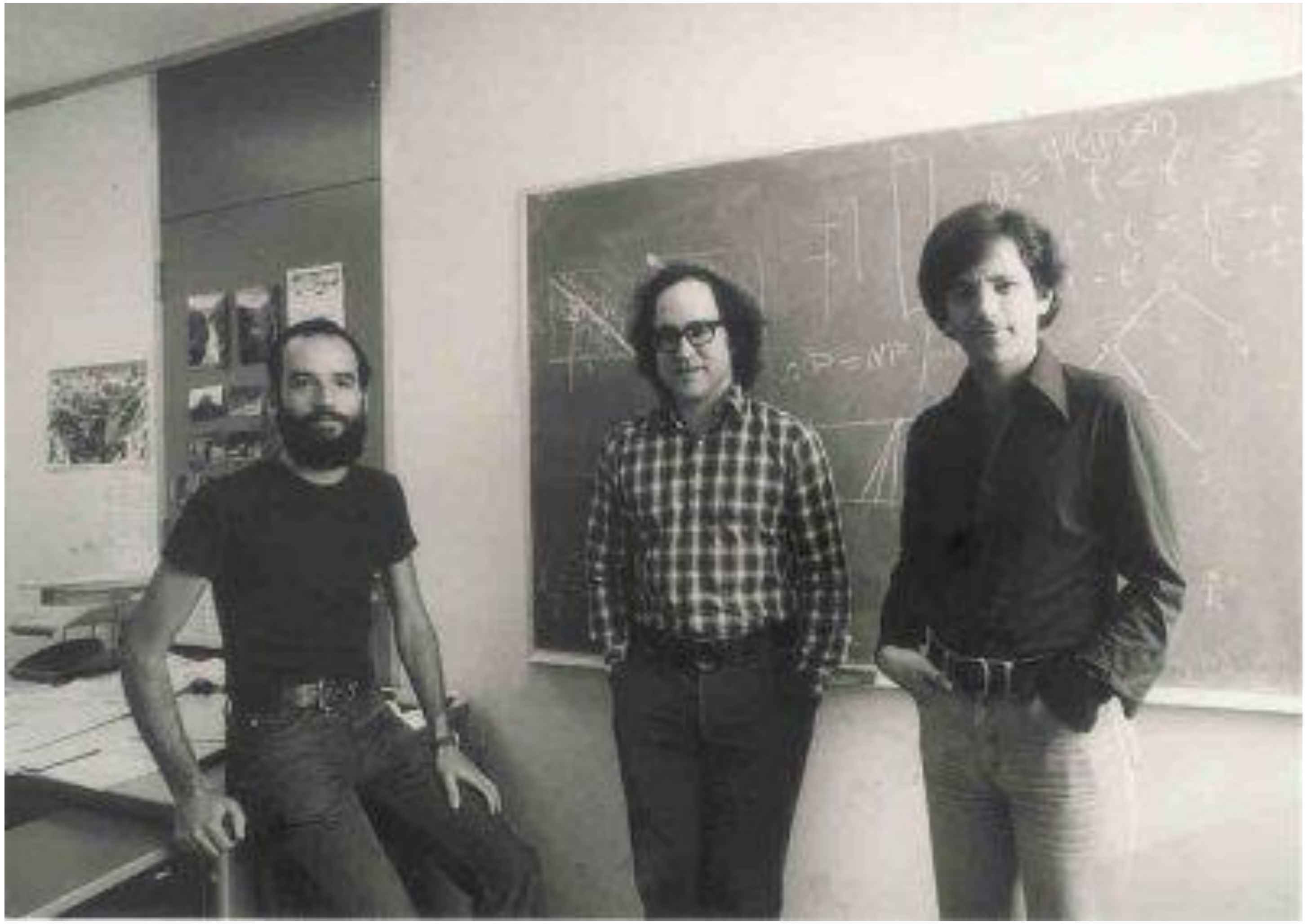
Content

1. Rivest–Shamir–Adleman **(RSA)**

2. Digital Signature Algorithm **(DSA)**

3

1. RSA

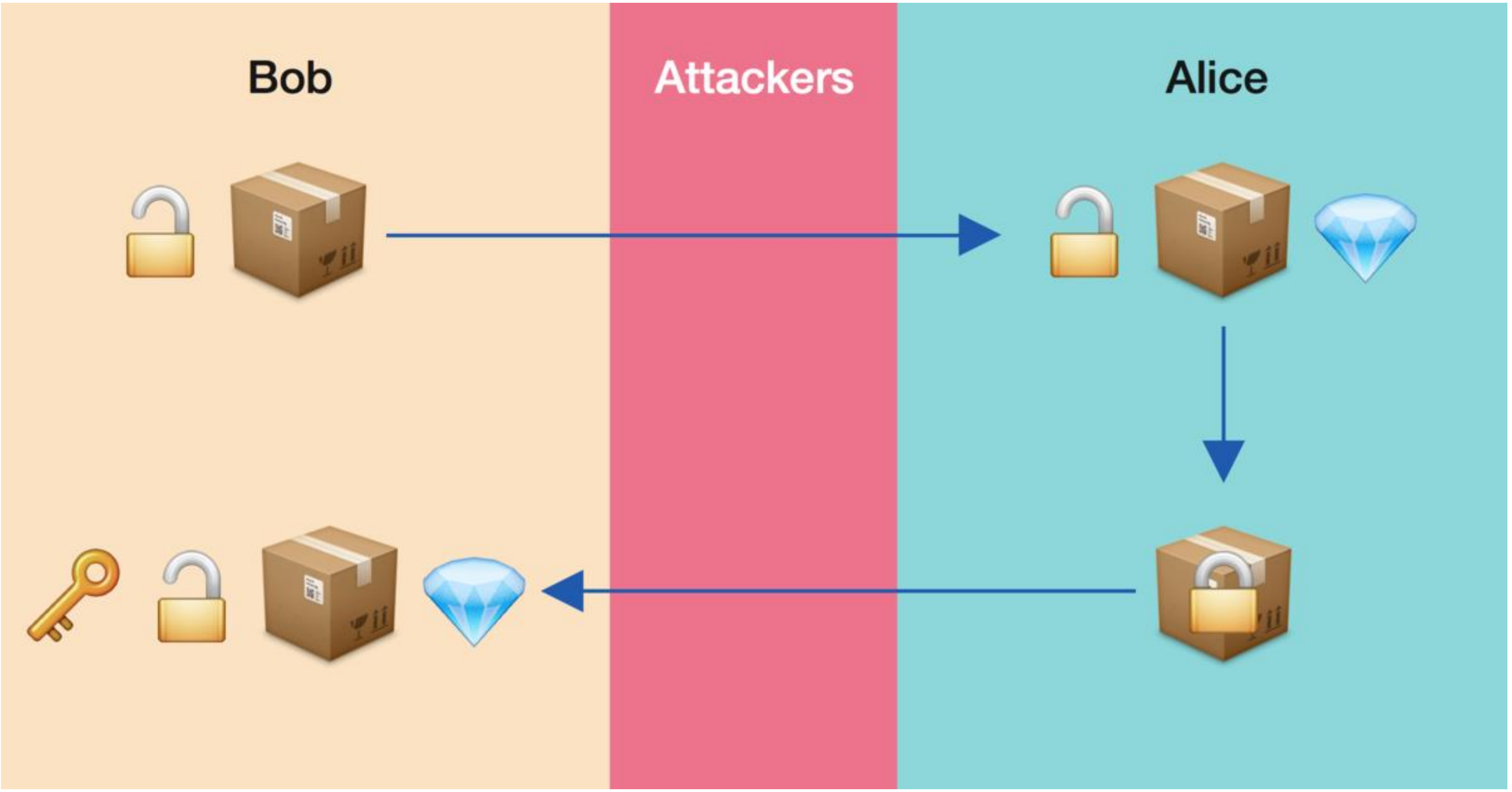


Rivest–Shamir–Adleman

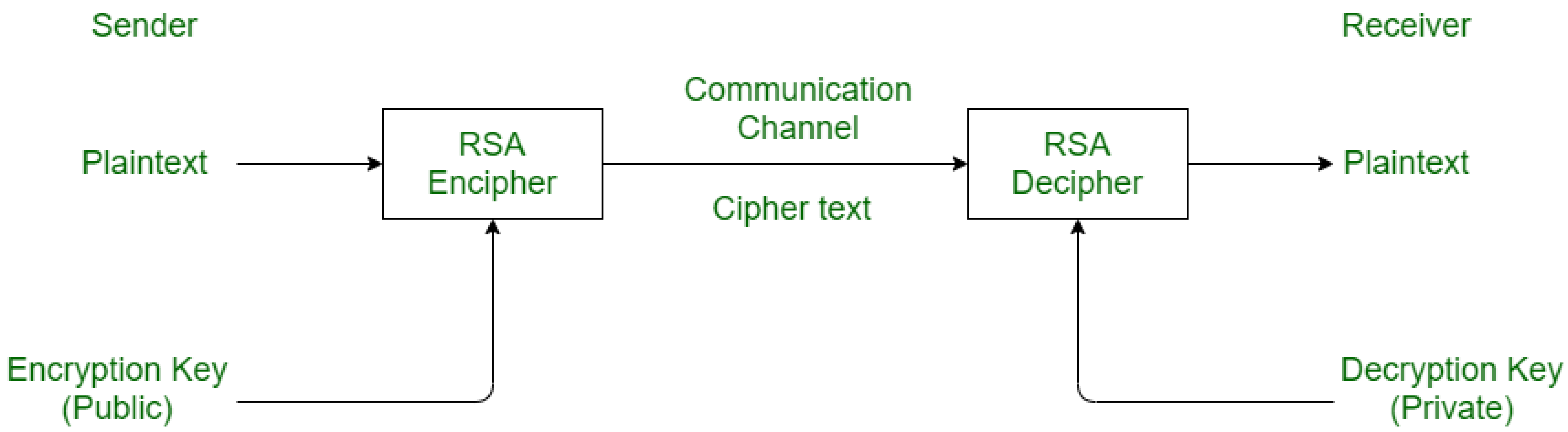
1.1 What is RSA?

- Public key cryptosystem
- First published in 1977
- Asymmetric Cryptography Algorithm

1.1 What is RSA?



1.1 What is RSA?



1.2 Algorithm

- Public Key: (e, n)
- Private Key: (d, n)
- Message: m

- Encryption:

$$c \equiv m^e \bmod n$$

- Decryption:

$$m \equiv c^d \bmod n$$

1.2 Algorithm

Key generation:

- Step 1: Choose two distinct prime numbers **p** and **q**
- Step 2: Compute $\boldsymbol{n} = pq$
- Step 3: Compute
  $$\boldsymbol{\varphi}(n) = (p-1)(q-1)$$
- Step 4: Choose an integer e

$$1 < e < \boldsymbol{\varphi}(n) \text{ and } \gcd\big(e, \boldsymbol{\varphi}(n)\big) = 1$$

- Step 5: Determine d as
  $$d \equiv e^{-1} (mod\ \boldsymbol{\varphi}(n))$$
  Or $de \equiv 1\ (mod\ \boldsymbol{\varphi}(n))$

Example:

- Step 1: $p = 3; q = 11$
- Step 2: $\boldsymbol{n} = 33$
- Step 3: $\boldsymbol{\varphi}(n) = (p-1)(q-1) = 20$
- Step 4: $e = 7$
- Step 5: Using extended Euclidean algorithm to determine d = 3

1.3 Proof

- Encryption: $c \equiv m^e \; mod \; n$
- Decryption: $m \equiv c^d \; mod \; n$

Prove that:

$$(m^e)^d \equiv \mathrm{m} \text{ (mod n)}$$

Or:

$$m^{ed} \equiv \mathrm{m} \text{ (mod n)}$$

1.4 Problem

Key generation

- Finding the large primes: 1024 to 4096 bits
- The primes should not be "too close": p – q is more than $2n^{1/4}$
- Private key *d* be large enough:

2. DSA

2.1 What is Digital Signature?

Public-Key Cryptography is used to **verify ownership** on a blockchain.

Digital signatures is mechanism to **determine authenticity** of a document file.

2.2 Hash function

2.2.1 Hash function

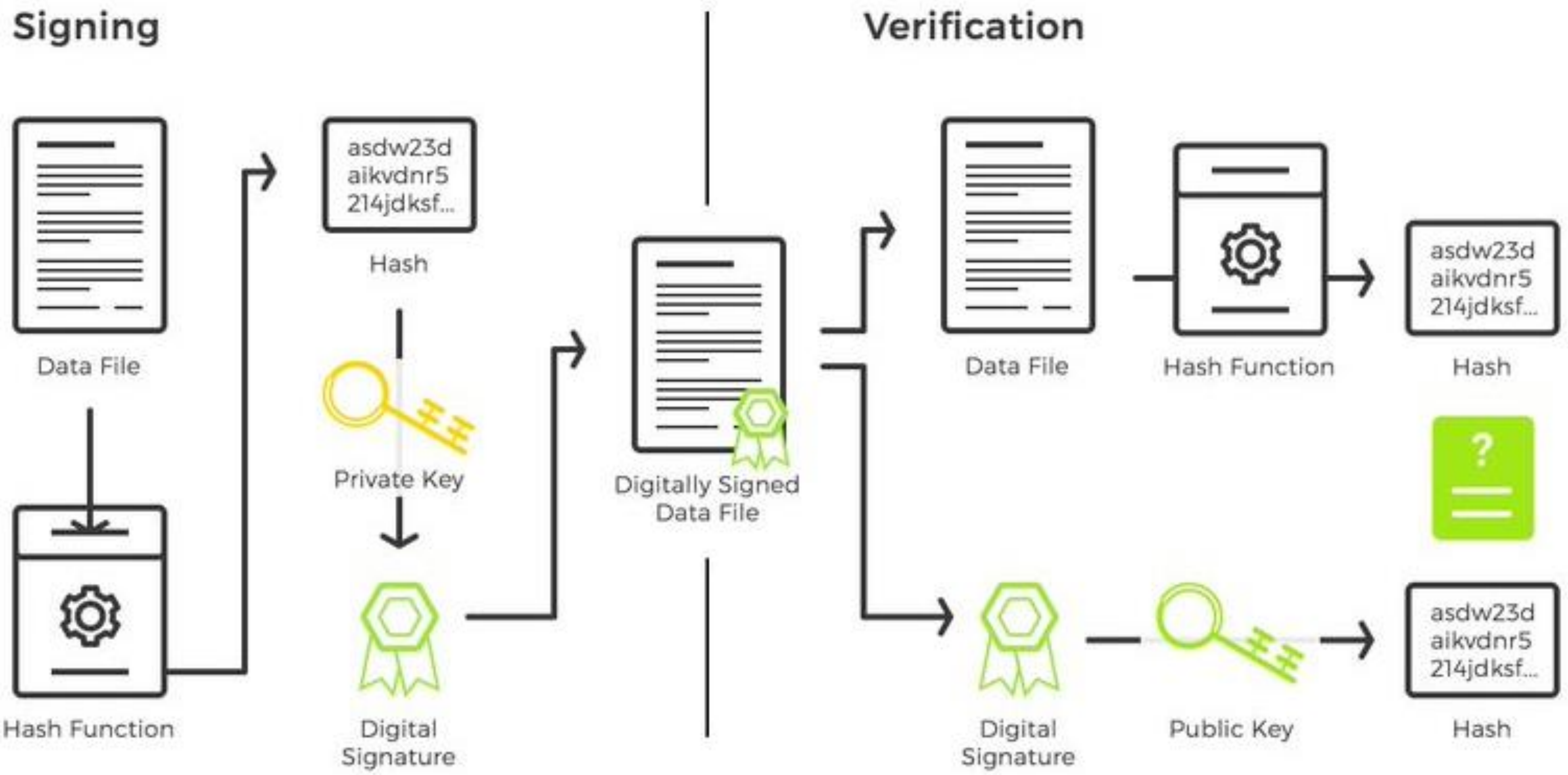
- Non-reversibility, or one-way function
- Determinism
- Diffusion, or avalanche effect
- Collision resistance
- Non-predictable

2.2 Hash function

Some popular hash functions:

- Secure Hashing Algorithm (SHA-2 and SHA-3)
- RACE Integrity Primitives Evaluation Message Digest (RIPEMD)
- Message Digest Algorithm 5 (MD5)
- BLAKE2

Digital Signature



2.3 DSA Implementation

2.3.1 **Key Generation**

Choose **an approved** cryptographic hash function H with output length |H| **bits**

2.3 DSA Implementation

2.3.1 **Key Generation**

Choose L and N tuples (in which, L is key length, N is modulus length)

FIPS 186-4 specifies L and **N** to have one of the values: (1024, 160), (2048, 224), (2048, 256), or (3072, 256)

2.3 DSA Implementation

2.3.1 **Key Generation**

Choose an **N-bit** prime **q**

Choose an **L-bit** prime p such that **p-1** is a multiple of q

Choose an integer **h** randomly from **{**2 **..** p-2**}**

2.3 DSA Implementation

2.3.1 **Key Generation**

Compute $g := h^{(p-1)/q} \mod p$

In the rare case that g**=1** try again with a different h**.** Commonly h=2 is used.

=> The algorithm parameters are **(**p**,** q**, g)**

2.3 DSA Implementation

Per-user keys [edit]

Given a set of parameters, the second phase computes the key pair for a single user:

- Choose an integer x randomly from $\{1 \ldots q-1\}$.
- Compute $\boxed{y := g^x \mod p.}$

- ***Private key*** can be packaged as: {p, q, g, **x**}
- ***Public key*** can be packaged as: {p, q, g, **y**}

2.3 DSA Implementation

2.3.2 **Key Distribution**

- The signer should publish the public key **y** (through receiver via a reliable organization).
- The signer should keep the private key **x** secret.

2.3 DSA Implementation

2.3.3 Signing

A message m is signed as follows:

- Choose an integer k randomly from $\{1 \ldots q-1\}$
- Compute $\boxed{r := \left(g^k \bmod p\right) \bmod q.}$ In the unlikely case that $r = 0$, start again with a different random k.
- Compute $\boxed{s := \left(k^{-1}\left(H(m) + xr\right)\right) \bmod q.}$ In the unlikely case that $s = 0$, start again with a different random k.

The signature is (r, s)

2.3 DSA Implementation

2.3.4 Signature verification

One can verify that a signature (r, s) is a valid signature for a message m as follows:

- Verify that $0 < r < q$ and $0 < s < q$.
- Compute $w := s^{-1} \mod q$.
- Compute $u_1 := H(m) \cdot w \mod q$.
- Compute $u_2 := r \cdot w \mod q$.
- Compute $v := (g^{u_1} y^{u_2} \mod p) \mod q$.
- The signature is valid if and only if $v = r$.

2.4 Correctness of the algorithm

- Fermat's little theorem
- Multiplicative order
- Modulo operation
- Source: [Digital_Signature_Algorithm]

2.4 Correctness of the algorithm

First, since $g = h^{(p-1)/q} \bmod p$, it follows that $g^q \equiv h^{p-1} \equiv 1 \mod p$ by Fermat's little theorem. Since $g > 0$ and q is prime, g must have order q.

The signer computes

$$s = k^{-1}(H(m) + xr) \bmod q$$

Thus

$$\begin{aligned} k &\equiv H(m)s^{-1} + xrs^{-1} \\ &\equiv H(m)w + xrw \pmod q \end{aligned}$$

Since g has order q we have

$$\begin{aligned} g^k &\equiv g^{H(m)w} g^{xrw} \\ &\equiv g^{H(m)w} y^{rw} \\ &\equiv g^{u_1} y^{u_2} \pmod p \end{aligned}$$

Finally, the correctness of DSA follows from

$$\begin{aligned} r &= (g^k \bmod p) \bmod q \\ &= (g^{u_1} y^{u_2} \bmod p) \bmod q \\ &= v \end{aligned}$$

Thanks for your attention