Distributed Trust and Blockchains	Date:	*22-08-2019*
Instructor: *Sujit Prakash Gujar*	Scribes:	*Padma Dhar, Sarat Sristi*

Lecture 2: Digital Signatures and DigiCash scheme

1 Recap

Hash functions : Hash functions are any functions that map arbitrary size data onto data of fixed size (*i.e* many $\rightarrow$ one function).

Desirable properties of hash functions:

1. It should be *Deterministic*
2. It should be efficient to compute.
3. *Pre-image resistance* : It should be difficult to find a message from the given hash value.

$$H(M) \rightarrow M$$

4. *Second Pre-image resistance* : Given a message x, it is difficult to find another message y such that

$$H(x) = H(y), \text{ given } x \neq y$$

5. *Collision resistance* : It is difficult to find two messages x, y such that their hash function outputs are equal.

Definition 1 (Commitment Scheme)**.** *A (non-interactive) Commitment Scheme (for a message space M) is a triple (Setup, Commit, Open) such that:*

(a) CK $\leftarrow$ Setup(1^k) generates the public commitment key.

(b) for any $m \in M$, $(c, d) \leftarrow Commit_{CK}(m)$ is the commitment/opening pair for m. c = c(m) serves as the commitment value, and d = d(m) as the opening value. We often omit mentioning the public key CK when it is clear from the context.

(c) OpenCK(c, d) $\rightarrow$ m $\in$ M $\cup$ $\perp$, where $\perp$ is returned if c is not a valid commitment to any message. We often omit mentioning the public key CK when it is clear from the context.

(d) Correctness: for any m $\in$ M, OpenCK(CommitCK(m)) = m

RSA
Encryption :

1. Choose tow large primes p, q.
2. Calculate $n = p \times q$.
3. Calculate $\phi = (p - 1) \times (q - 1)$.

4. Choose e and d such that $e.d = 1 \ mod \ \phi(n)$.
5. Calculate cipher text $c = (m^e) \ mod \ n$.

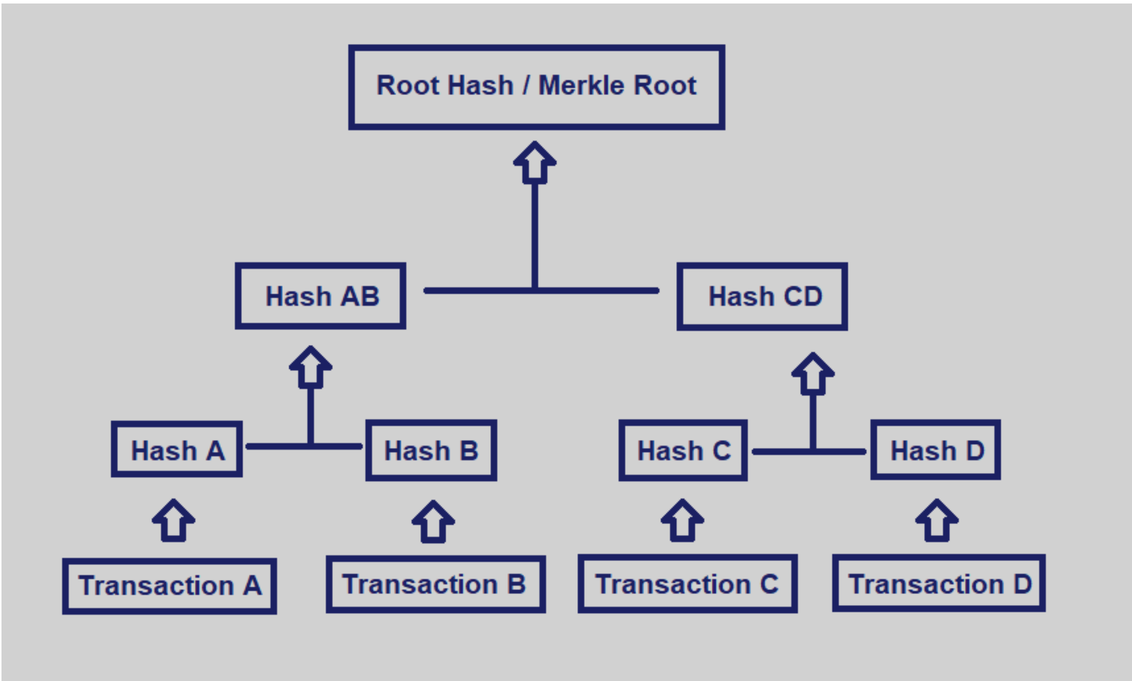
Decryption:

$$d = c^d \ mod \ n \ \ = ((m^e)modn)^d \ mod \ n \ \ = m^{e \times d} \ mod \ n \ \ = m$$

Merkle Tree

Definition 2. *Merkle tree in cryptography is a tree in which every leaf node is labelled with the hash of a data block, and every non-leaf node is labelled with the cryptographic hash of the labels of its child nodes.*

1. Hash trees allow efficient and secure verification of the contents of large data structures. Hash trees are a generalization of hash lists and hash chains.
2. Demonstrating that a leaf node is a part of a given binary hash tree requires computing a number of hashes proportional to the logarithm of the number of leaf nodes of the tree.
3. height of tree = log(nodes of data)
4. In the given diagram, transaction A,B,C,D are stored at the leaf nodes. The hash of each of them is calculated and stored one level up in the tree. Then 2 of these hashes are combined via hash to generate a new hash and so on each level up.



credits: https://hackernoon.com/merkle-trees-181cb4bc30b4

2 Digital signatures

Definition 3. *(Digital signature). A signature scheme is a tuple of three PPT algorithms:* (Gen,Sign,Vrfy) *satisfying the following:*

1. *The key-generation algorithm* Gen *takes as input a security parameter n, and outputs a pair of keys* (pk,sk). pk *is the public key and* sk *is the secret key. Assume both have length* n.

2. *The signing algorithm Sign, takes as input a private key sk and a message* $\in \{0,1\}^*$ *. It outputs a signature* σ*, denoted* $\sigma \leftarrow Sign_{sk}(m)$*.*

3. *The deterministic verification algorithm* Vrfy *takes as input a public key* pk*, and a message* m*, and a signature . It outputs a bit* b=1*, meaning valid, and* b=0 *meaning invalid.We denote this as* b = $Vrfy_{pk}$(m, σ)

2.1 RSA based digital signature :

1. Gen(n): Outputs *(N,e,d)*, where $N = pq$, where p and q are both n bit primes, $ed = 1 \ mod \ N$.
2. Sign: On input a private key sk = *(N,d)*, and a message $m \in Z_n^*$,

$$\sigma = m^d \text{ mod N}$$

3. Vrfy: On input a public key *pk=(N,e)*, and a message $m \in Z$, and a signature scheme Z, output 1 if and only if:

$$m = \sigma^e \ mod \ N$$

2.1.1 Forging a signature on an arbitrary message :

If an adversary wants to output a forgery of any given message m, then he can successfully forge the signature of m by having two signatures of chosen messages.

1. Let m_1 be any random chosen messages and σ_1 be its respective signature.
2. Now adversary calculate $m_2 = m/m_1$ and send this resultant m_2 to signer to get signature σ_2.
3. Now note, any valid sign for m, is

$$\sigma = m^d = (m_1.m_2)^d = \text{m}_1^d \ . \ {m_2}^d = (\sigma_1.\sigma_2)\text{mod N}$$

2.1.2 Solution *(Hashed RSA) :*

1. The basic idea is to modify the textbook RSA by applying some hash function H to the message before signing.
2. The scheme considers a public known function:

$$H : \{0,1\}^* \rightarrow Z_N$$

3. The sign σ is computed from m, as follows:

$$\sigma = [H(m)]^d mod \ N$$

4. If the hash function is collision resistant,it becomes difficult to find two messages $m \neq m_1$, st $H(m) = H(m_1)$.
5. so, lets consider

$$s_1 = [H(m_1)]^d mod \ N$$
$$s_2 = [H(m_2)]^d mod \ N$$
$$s_3 = [H(m_1.m_2)]^d mod \ N$$

If we try to apply forge the signature s_3, we have to find m_1 and m_2 st:

$$[H(m_1.m_2)] = [H(m_1)].[H(m_2)]$$

2.2 ElGamal Scheme

2.2.1 Key aspects :

1. Based on the Discrete Logarithm problem.
2. Randomized encryption scheme.

2.2.2 Key Generation :

Participant A generates the public/private key pair.

1. Generate large prime p and generator g of the multiplicative group $\mathbb{Z}_p^*$.
2. Select a random integer a, $1 \leq a \leq p\text{-}2$, and compute g^a *mod p*.
3. A's public key is $(p,\ g,\ g^a)$; A's private key is a.

2.2.3 Encryption :

Participant B encrypts a message m to A.

1. Obtain A's public key $(p,\ g,\ g^a)$.
2. Represent the message as integers in the range of $\{\ 0,\ 1,\ 2,\ \ldots,\ p\text{-}1\ \}$.
3. Select a random integer k, $1 \leq k \leq p\text{-}2$.
4. Compute $\gamma = g^k$ *mod p* and $\delta = \ m \times (g^a)^k$.
5. Send cipher text $c = (\gamma,\ \delta)$.

2.2.4 Decryption :

Participant A receives encrypted message m from B.

1. Use private key a to compute (γ^{p-1-a}) *mod p*.
   Note : $(\gamma^{p-1-a}) = (\gamma^{-a})$.
2. Recover m by computing $(\gamma^{-a}) \times \delta$ *mod p*.

$$\begin{aligned} &= (\gamma^{-a}) \times \delta \ mod\ p \\ &= (g^a)^{-k}\ mod\ p \times\ m \times (g^a)^k\ mod\ p \\ &= (g^{-ak}) \times (g^{ak}) \times\ m\ mod\ p \\ &= m\ mod\ p \\ &= m \end{aligned}$$

2.2.5 Sign :

1. Select key k randomly. $r{=}(g^k)\times$ *mod p*
$s{=}(k^{-1}) \times (m - xr) \times (mod\ p)$
return $((r,s))$

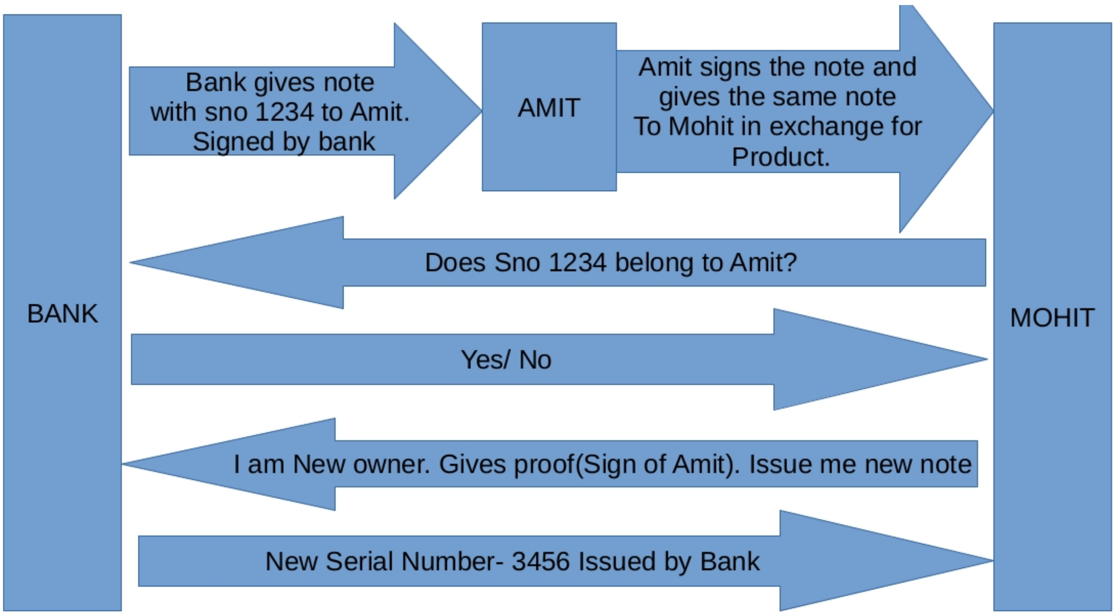
2.2.6 Verify :

We have
$s=(k^{-1})\times(m\text{-}xr)$
$ks=(m\text{-}xr)$
$m=(ks)+(xr)$
$(g^m) = (g^{ks+xr}) = (g^{x^r})\times(g^{k^s})$
$(g^m) = (h^r)\ \times(\mathrm{r}^s)$

$Verify\ if\ (\mathrm{g}^m) == (h^r)\ \times(r^s)$.
If true, return true, else false.

3 Digital Currency

3.1 First attempt at digital currency



3.1.1 Advantages

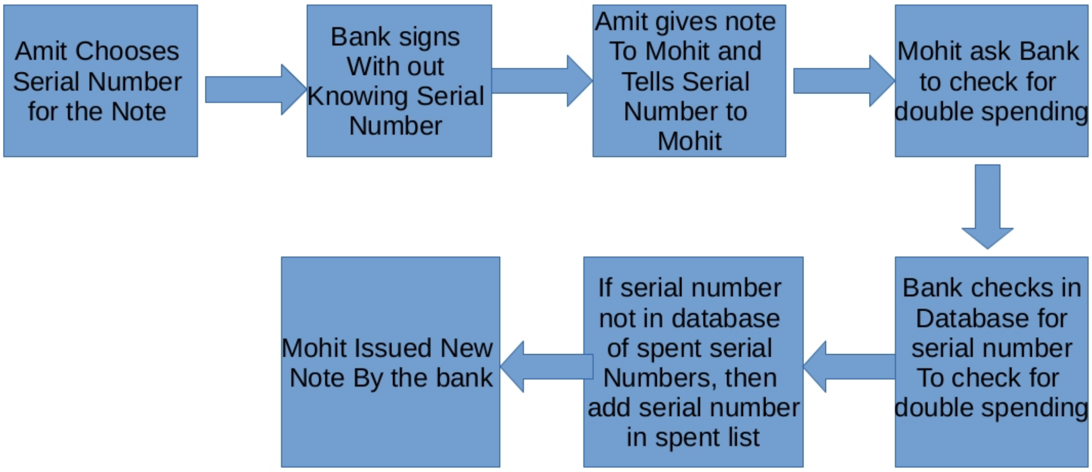
- Proof of Ownership due to digital signature. No fake ownership can be claimed.
- No double spending as Mohit has to check with bank as soon as he receives the note from Amit.
- Transfer of ownership possible.

3.1.2 Challenges

- Bank knows how the user is spending money. There is no anonymity possible.
- Bank has to always be online and never be down.

3.2 Attempt to Introduce anonymity

3.2.1 DigiCash



- Here, we are also providing anonymity to the issuer-Amit as Bank does not know serial Number.
- A doubt might arise that since the bank does not provide serial number, there maybe a possibility of 2 people choosing the same serial number thus causing a collision. However, if the size of the serial number is taken to be very large- 256 bits, possibility of such an occurrence becomes extremely negligible.