ZKP-Tale
Registration to https://imimsociety.net/en/

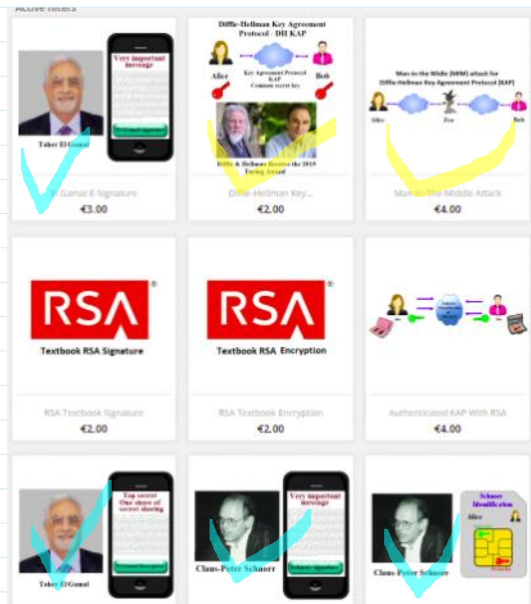
Schnorr Identification

>> p = genstrongprime(24)
>> q = (p-1)/2
>> isprime(q)

What are prime numbers: 3, 4, 5, 6, 7
p=2q+1 is srong prime if q is prime.
What are strong prime numbers: 7, 9, 11, 13

http://crypto.fmf.ktu.lt/telekonf/archyvas/M123%20DataSecurity/S170M123%202021/

$7 = 2 \cdot \overset{q}{3} + 1$
$11 = 2 \cdot 5 + 1$
$13 = 2 \cdot 6 + 1$



$Z_P^* = \{1, 2, 3, \ldots, p\text{-}1\}$ multiplication *mod p

Fact C.23. Say p=2q+1 is **strong prime** (then q is prime),
then g in Z_P^* = {1, 2, 3, ... , p-1} is a generator of Z_P^*
Iff $g^q \neq 1 \bmod p$ and $g^2 \neq 1 \bmod p$.

Public Parameters PP = (p, g): p=15728303; g=5;
p - strong prime; g - generator.

Private key **PrK** and public key **PuK** generation for
Alice and **Ema**.

>> x=randi(p-2)
x = 13426057 % PrK_A=x
>> a=mod_exp(g,x,p)
a = 2045067 % PuK_A=a

$\boxed{a = g^x \bmod p}$

24 bits arithmetics.
>> 2^24
ans = 16777216
>> ansb=dec2bin(ans)
anab = 1 0000 0000 0000 0000 0000 0000

>> p=genstrongprime(24)
p = 15 728 303
>> q=(p-1)/2
q = 7864151
>> isprime(p)
ans = 1
>> isprime(q)
ans = 1
>> pb=dec2bin(p)
pb = 1110 1111 1111 1110 1010 1111

Security: for given a, g, p it is infeasible to find x, e.g. if $p \sim 2^{2048}$.
This computational problem is called Discrete Logarithm Problem – DLP.

pb = 1110 1111 1111 1110 1010 1111
>> ph=dec2hex(p)
ph = EFFEAF

g = 5
>> mod_exp(g,q,p)
ans = 15728302
>> mod_exp(g,2,p)
ans = 25

p=268435019; g=2; Are changed.

Parties: **Alice - A** and **Bank - B**

Registration phase: Bank generates **PrK_A = x** and **PuK_A = a** to **Alice**
And hands over this data in smart card or other crypto chip in **Alice**'s smart phone
Or in software for Smart ID.

A:

$PrK_A = x \leftarrow$ randi(24)

$PuK_A = a = g^x \bmod p$ $\quad 1 < x < p-2$

$\xleftarrow{\ p,\ g,\ x,\ a,\ b\ }$

B:

$PrK_B = y \leftarrow$ randi(24)

$PuK_B = b = g^y \bmod p$

B: $x \leftarrow$ randi(p-2)

$a = g^x \bmod p$

Schnorr Id Scenario: Alice wants to prove Bank that she knows her Private Key - **PrK_A**
which corresponds to her Public Key - **PuK_A** not revealing **PrK_A**.
Protocol execution between **Alice** and **Bank** has time limit.
Alice's computation resources has a limit --> protocol must be computationally effective.

B: Includes Alice name, surname & PuK_A to his data base, i.e. to the clients data base.

Zero knowledge Proof – ZKP

A – is a prover; B – is a verifier

Proof procedure is performed by the conversation between A and B.
Conversation consist of three steps:
1. Commitment t: is computed by A.
2. Challenge h: is computed by B.

3. Response res: is coputed by $\mathcal{A}$.

$\mathcal{A}$: ① Commitment t computation.

$u \leftarrow$ randi(p-2)

$t = g^u \bmod p$.

```
>> u=randi(p-2)
u =
>> t=mod_exp(g,u,p)
t =
```

$$\xrightarrow{PuK_A = a,\ t}$$

$\mathcal{B}$: verifies if $PuK_A = a$ is included in his clients data base and verifies to what client it belongs.

② computes challenge h at random, e.g.

$$\xleftarrow{h}$$

$h \leftarrow$ randi(p-2)

$\mathcal{A}$: ③ Response computation

$res = u + x \cdot h \mod (p-1)$

the computations in exponents

```
>> xh=mod(x*h,p-1)
xh =
>> res=mod(u+xh,p)
res =
```

$$\xrightarrow{res}$$

$\mathcal{B}$: verifies if $\mathcal{A}$ knows her $PrK_A = x$ corresponding to her $PuK_A = a$. Verification is performed using conversation data: t, h, res, a.

$$\boxed{g^{(res)} \bmod p = t \cdot a^h \bmod p} \quad (Ver)$$

```
>> g_res=mod_exp(g,res,p)
g_res =
>>
>> a_h=mod_exp(a,h,p)
a_h =
>> ta_h=mod(t*a_h,p)
ta_h =
```

$$g^{(res)} \bmod p =$$

$$= g^{u+xh} \bmod p = g^u \cdot g^{xh} \bmod p =$$

$$= t \cdot (g^x)^h \bmod p = t \cdot a^h \bmod p$$

$\mathcal{E}$: is an eavesdropping adversary she is recording a conversation data between A and B: $t, h, res, a, (p, q)$

$\mathcal{I}o$: does not know $PrK_A = x$

random generated $u \leftarrow randi(p-2)$

$\mathcal{I}o$: 1) it is infeasible to compute $PrK_A = x$ by solving the equation $a = g^x \bmod p$ when p is large prime $p \sim 2^{2048}$.

$\mathcal{I}o$: 2) is trying to impersonate A against B trying to find such a res satisfying (Ver) equation

$$\boxed{g^{(res)} \bmod p = t \cdot a^h \bmod p} \quad \text{(Ver)}$$

when $t, h, a, (p, q)$ and res are given.

Brute force, total scan attack: $\mathcal{I}o$ must find such x that $g^x \bmod p = a \Rightarrow$ It is infeasible due to 1) attack scenario.

A: computation resources are small $\Rightarrow$

$\Rightarrow$ arithm. operations should be effective.

Most expensive operation is $\boxed{t = g^u \bmod p}$ and it is effective even using smart phones or cryptographic chips.

1) Time slot of Id is restricted

2) t is sent before the h is received.

Till this place

H-functions

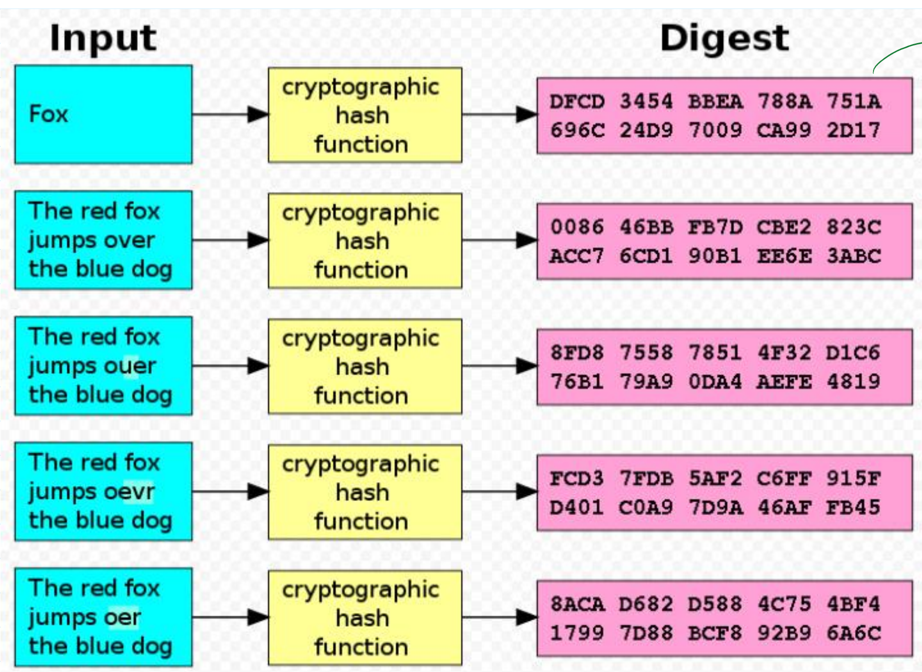
H-Functions are working horses in cryptography [Bruce Schneier].

A **cryptographic hash function** is a special class of hash function that has certain properties which make it suitable for use in cryptography.
It is a mathematical algorithm that maps data of arbitrary finite size to a bit string of a fixed size (a hash function) which is designed to also be a one-way function, that is, a function which is infeasible to invert.
The only way to recreate the input data from an ideal cryptographic hash function's output is to attempt a brute-force search of possible inputs to see if they produce a match.
The input data is often called the ***message***, and the output (the ***hash value*** or ***hash***) is often called the ***message digest*** or simply the ***digest***.

M – message to be signed (big message ~ 1 GB → 8 G bits)

$|p| \sim 2048$ bits

$H(M) = h$; $|h| \sim 256$ bits



$SHA_{160}(Fox) =$

awelange effect

Schnorr Signature CORRECTED COMPUTATIONS

Public Parameters $PP = (p, g)$

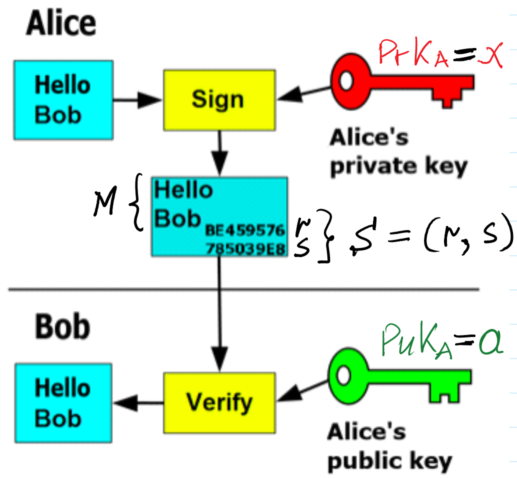
p = 264043379; **g** = 2;

A:

```
>> p = 264043379;
>> g=2;
>> x=randi(p-1)
```

x = 84102568
>> a=mod_exp(g,x,p)
a = 46883346

Asymmetric Signing - Verification
S=Sig(PrK$_A$, h)=(r, s)
V=Ver(PuK$_A$, S, h), V$\in$ {True, False} $\equiv$ {**1**, **0**}



M - be a message to be signed

Signing:

$u \leftarrow randi(p-1)$

$r = g^u \bmod p$; t-"commitment"

$h = H(M||r)$

$s = u + x h \bmod (p-1)$; res-in Schnorr Id protocol.

$S = (r, s)$

Verifying

$g^s \bmod p = g^{u+xh} \bmod p =$

$= g^u \cdot g^{xh} \bmod p = r \cdot (g^x)^h \bmod p =$

$= r \cdot a^h \bmod p.$

hd28

A:

>> M='Hello Bob, I bought you an electrocar costing 111222 Eur'
M = Hello Bob, I bought you an electrocar costing 111222 Eur

>> u=randi(p-1)
u = 168293184
>> r=mod_exp(g,u,p)
r = 12693468

>> h=hd28('Hello Bob, I bought you an electrocar costing 111222 Eur'||'12693468')
h = 126174618
>> hd28('M'||'12693468')
ans = 126174618

>> xh=mod(x*h,p-1)
xh = 172722116
>> s=mod(u+xh,p-1)
s = 76971922

Signature   **S=(r, s)**

$S = (r, s)$

B: Verification

$S = (r, s)$

B: Verification

Verification identity:

$$\boxed{g^s = r \cdot a^h \bmod p}$$

g_s   ra_h

>> g_s=mod_exp(g,s,p)
g_s = 127805170

>> a_h=mod_exp(a,h,p)
a_h = 76904588
>> ra_h=mod(r*a_h,p)
ra_h = 127805170

Computation with errors

```
p =  264043379
>> g
g =  2
>> x
x =  223492566
>> a
a = 159796474
>>
>> m='Hello Bob, I bought you an electrocar costing 111222 Eur'
m = Hello Bob, I bought you an electrocar costing 111222 Eur
>> u=randi(p-1)
u =  148919531
>> M=m
M = Hello Bob, I bought you an electrocar costing 111222 Eur
>>
>> r=mod_exp(g,u,p)
r = 218757680
>> h=hd28('Hello Bob, I bought you an electrocar costing 111222 Eur'||'218757680')
h =  126174618
>> h1=hd28('M'||'218757680')
h1 =  126174618
>>
>> xh=mod(x*h,p-1)
xh =  180685612
>> s=mod(u+xh,p-1)
s =  65561765
>>
>> g_s=mod_exp(g,s,p)
g_s = 148583808
>> a_h=mod_exp(a,h,p)
a_h = 245106544
>> ra_h=mod(r*a_h,p)
ra_h = 66248474
```