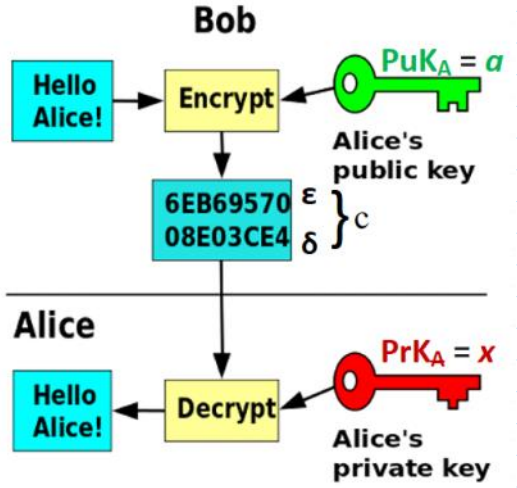
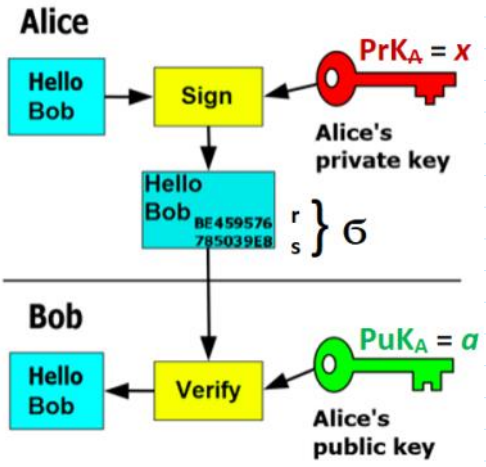
Authenticated Encryption based KAP - AEKAP
Schnorr Siganture
Non-Interactive Zero Knowledge Proof

Asymmetric Encryption - Decryption
c=Enc(PuK_A, m)
m=Dec(PrK_A, c)



Asymmetric Signing - Verification
Sign(PrK_A, m) = σ = (r, s)
V=Ver(PuK_A, m, s), V∈{True, False} ≡ {1, 0}



k=randi(Z_p*)
c_k=Enc(PuK_B,i,k)
M-massage to be encrypted
with symmetric encryption method
e.g. AES128
C_k=E(k,M)=AES128(k,e,M)

c_k →
C_k

k=Dec(PrK_B,c_k)
M=D(k,d,C_k)=AES128(k,d,C_k)

k=randi(Z_p*)
c_k=Enc(PuK_B,i,k)
M-massage to be encrypted
with symmetric encryption method
e.g. AES128
C_k=E(k,M)=AES128(k,e,M)

h=H(C_k)
Sign(PrK_A,h) = **σ = (r, s)**

c_k →
C_k, σ
$Cert_A$

h=H(C_k)
Ver(PuK_A,σ) = True
k=Dec(PrK_B,c_k)
M=D(k,d,C_k)=AES128(k,d,C_k)
Ver($Cert_A$,PuK_{CA})=True
CRL, OCSP

Imagine that W generated her $PrK_W = z$ and $PuK_W = e$,
W send a message to A writing the following message:
„Dear A I am B and I am sending you my $PuK = e$
for our further communications. Trully yours B."

for our further communications. Trully yours B."

PKI - Public Key Infrastructure is created as a Trusted Third Party - TTP to confirm that PuK belongs to the concrete person and to anybody else.

010_002 PKI_TimeStamp

Schnorr Signature Scheme **(S-Sig).**

In general, to create a signature on the message of any finite length ***M*** parties are using cryptographic secure H-function (message digest).
In Octave we use H-function
>> hd28('…') % the input '…' of this function represents a string of symbols between the commas.
% the output of this function is decimal number having at most 28 bits.

Let ***M*** be a message in string format to be signed by **Alice** and sent to **Bob**: >> M='Hello Bob'
For signature creation **Alice** uses public parameters **PP=(*p*, *g*)** and
Alice's key pair is $\mathbf{PrK_A}=x$, $\mathbf{PuK_A}=a=g^x \bmod p$.

Alice chooses at random u, $1<u<p-1$ and computes first component r of his signature:

$$r=g^u \bmod p. \qquad (2.19)$$

Alice computes H-function value h and second component s of her signature:

$$h=H(M\|r), \qquad (2.20)$$

$$s=u+xh \bmod (p-1). \qquad (2.21)$$

Alice's signature on h is $\sigma=(r, s)$. Then **Alice** sends M and σ to **Bob**.

After receiving M' and σ, **Bob** according to (2.20) computes h'

$$h'=H(M'\|r),$$

and verifies if

$$g^s \bmod p = ra^{h'} \bmod p. \qquad (2.22)$$

V1 V2

Symbolically this verification function we denote by

$$Ver(a,\sigma,h')=V\in\{True, False\}\equiv\{1, 0\}. \qquad (2.23)$$

This function yields ***True*** if (2.22) is valid if: $h=h'$ and $\mathbf{PuK_A}=a=F(\mathbf{PrK_A})=g^x \bmod p$.
and: $M=M'$

Non-Interactive Zero Knowledge Proof - NIZKP **PP** = (*p*, *g*).
In blockchain applications

𝒜: NIZKP of knowledge *x*:
$\mathbf{PrK_A}=x=$ **randi(*p*-1)**
$\mathbf{PuK_A}=a=g^x \bmod p$
1. Computes ***r*** for random number ***i***:
i=randi(p-1)
$r=g^i \bmod p$
2. Generates ***h***:
***h*=randi(*p*-1)**
3. Computes:
$s=i+xh \bmod (p-1)$

$\mathbf{PuK_A}=a$
h, (r, s), $Cert_A$

ℬ: $\mathbf{PuK_A}=a$
$Ver(Cert_A, PuK_{CA})$=True
CRL, OCSP
Verifies:
$g^s = ra^h \bmod p$
V1 V2

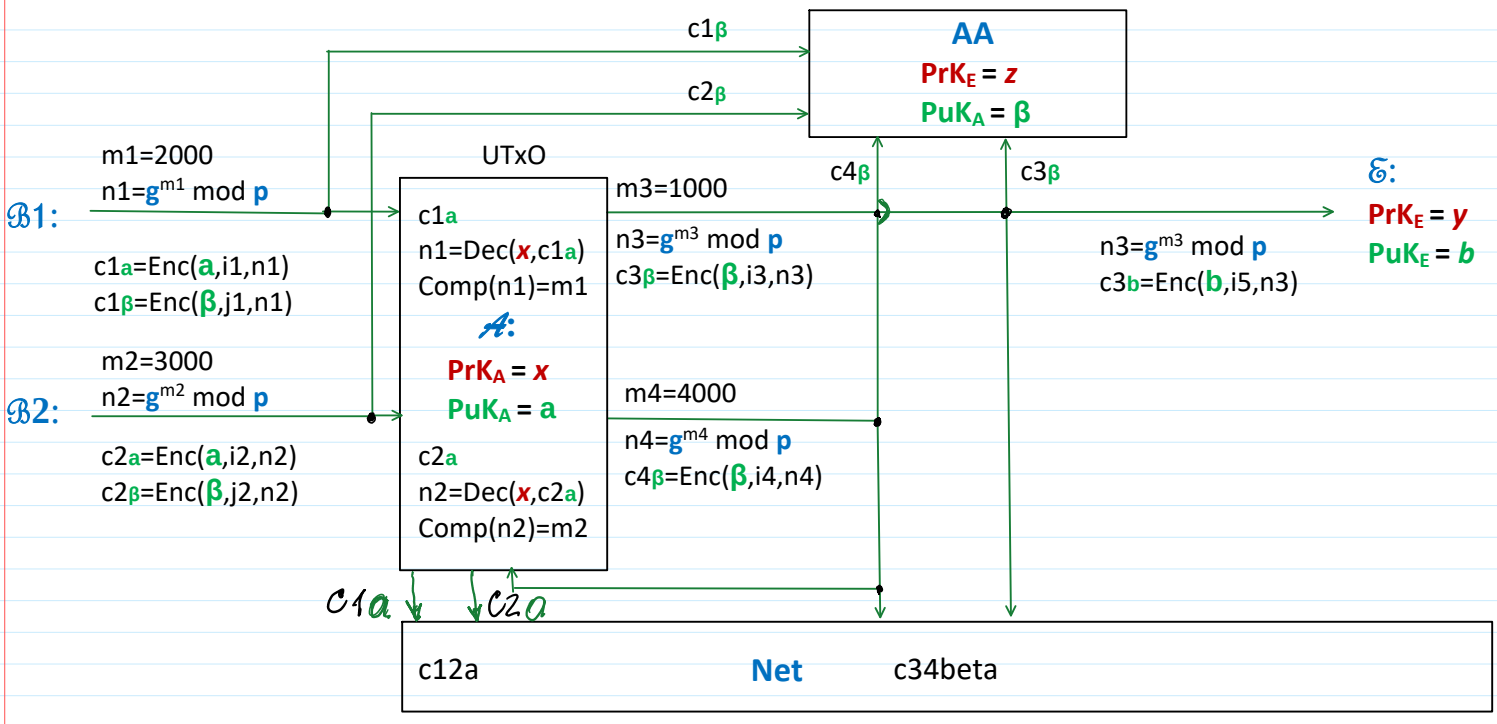
>> p= int64(268435019)
p=268435019
>> g=2;

>> x=int64(randi(p-1))
x = 89089011
>> a=mod_exp(g,x,p)
a = 221828624

>> i=int64(randi(p-1))
i = 228451192
>> r=mod_exp(g,i,p)
r = 33418907
h=int64(randi(p-1))
h = 49498768
> s=mod((i+x*h),p-1)
s = 263459276

>> g_s=mod_exp(g,s,p)
g_s = 222564698
V1=g_s;
>> a_h=mod_exp(a,h,p)
a_h = 143512214
>> V2=mod(r*a_h,p)
V2 = 222564698

β a b β a b



A : $m_1 + m_2 = 2000 + 3000 = 5000 = 1000 + 4000 = m_3 + m_4$

To prove that Tx is honest Alice is using additively-homomorphic encryption:

1. Enc(a,m1+m2)=c1a*c2a=c12a
   Enc(beta,m3+m4)=c3beta*c4beta=c34beta
1. Alice must prove that ciphertexts c12a and c34beta are equivalence.
   It means that ciphertexts c12a and c34beta encrypts the same plaintext, namely 5000.

Till this place

Behaviour of the Rivest, Shamir, Tauman ring signature scheme
In the original paper, Rivest, Shamir, and Tauman described ring signatures as a way to leak a secret. For instance, a ring signature could be used to provide an anonymous signature from "a high-ranking White House official", without revealing which official signed the message. Ring signatures are right for this application because the anonymity of a ring signature cannot be revoked, and because the group for a ring signature can be improvised.
Another application, also described in the original paper, is for deniable signatures. Here the sender and the recipient of a message form a group for the ring signature, then the signature is valid to the recipient, but anyone else will be unsure whether the recipient or the sender was the actual signer. Thus, such a signature is convincing, but cannot be transferred beyond its intended recipient.
There were various works, introducing new features and based on different assumptions:

Threshold ring signatures
Unlike standard "*t*-out-of-*n*" threshold signature, where *t* of *n* users should collaborate to sign a message, this variant of a ring signature requires *t* users to cooperate in the ring signing protocol. Namely, *t* parties $S_1, ..., S_t \in \{P_1, ..., P_n\}$ can compute a (t, n)-ring signature, σ, on a message, *m*, on input $(m, S_1, ..., S_t, P_1, ..., P_n)$.

Linkable ring signatures
The property of linkability allows one to determine whether any two signatures have been produced by the same member (under the same private key). The identity of the signer is nevertheless preserved. One of the possible applications can be an offline e-cash system.

Traceable ring signature
In addition to the previous scheme the public key of the signer is revealed (if they issue more than one signatures under the same private key). An e-voting system can be implemented using this protocol.

Efficiency

Most of the proposed algorithms have asymptotic output size ; i.e., the size of the resulting signature increases linearly with the size of input (number of public keys). That means that such schemes are impracticable for real use cases with sufficiently large (for example, an e-voting with millions of participants). But for some application with relatively small median input size such estimate may be acceptable.
CryptoNote implements ring signature scheme by Fujisaki and Suzuki in p2p payments to achieve sender's untraceability. More efficient algorithms have appeared recently. There are schemes with the sublinear size of the signature, as well as with constant size.

The original paper describes an RSA based ring signature scheme, as well as one based on Rabin signatures. They define a keyed "combining function" which takes a key, an initialization value, and a list of arbitrary values.
is defined as, where
 is a trap-door function (i.e. an RSA public key in the case of RSA based ring signatures).
The function
 is called the ring equation, and is defined below. The equation is based on a symmetric encryption function:
It outputs a single value which is forced to be equal to. The equation can be solved as long as at least one , and by extension, can be freely chosen. Under the assumptions of RSA, this implies knowledge of at least one of the inverses of the trap door functions (i.e. a private key), since.

Signature generation

Generating a ring signature involves six steps. The plaintext is signified by, the ring's public keys by.

1. Calculate the key
, using a cryptographic hash function. This step assumes a random oracle for, since will be used as key for.
2. Pick a random glue value.
3. Pick random for all ring members but yourself (will be calculated using the **s**igner's private key), and calculate corresponding.
4. Solve the ring equation for
5. Calculate
 using the signer's private key:
6. The ring signature now is the -tuple

Signature verification

Signature verification involves three steps.

1. Apply the public key trap door on all:
.

2. Calculate the symmetric key
   .
3. Verify that the ring equation holds
   .