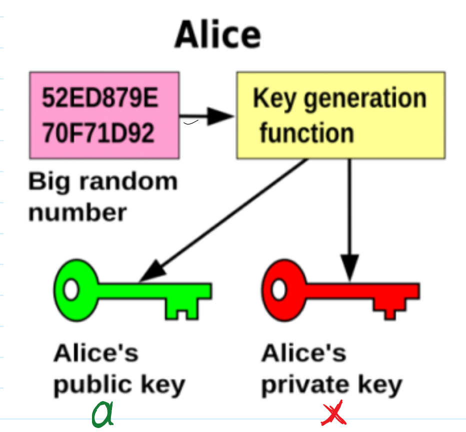
Exam content.
1.**Oral part**. You can prepare formulas in advance without comments.
   1.1.Coin flipping.
   1.2.Bit commitment using RSA.
2.**Computation part**. You should provide a computations and write results in the Google drive.
  The training of this part will be realized in 10-th of December during our class.
   2.1.Proxy signature realization.
   2.2.Additively-multiplicative encryption realization.

Poster Report (**PR**) presentation will be held in 17-th of December during our class.
PR requirements are placed in:
https://docs.google.com/document/d/1raqTudLCNlLm3wLFCDp_V7QnOg_EFH6d/edit?usp=sharing&ouid=111502255533491874828&rtpof=true&sd=true

PR topic are placed in:
https://docs.google.com/document/d/1KjXlhHhRQJnKnbCbK8crbOxoxy-EaSBf/edit?usp=sharing&ouid=111502255533491874828&rtpof=true&sd=true

PKCS - Public Key Crypto System: 1.Key generation $PP = (p, g)$



① $p = 2q + 1$; p, q - are primes; p - strong prime

$2 \mid p-1$ & $q \mid p-1$

p - strong prime

>> p=genstrongprime(l); $l = 28$.

$Z_p^* = \{1, 2, 3, \ldots, p-1\}$ * mod p

② g is generator iff

$g^2 \neq 1 \bmod p$ & $g^q \neq 1 \bmod p$

$Z_{p-1} = \{0, 1, 2, \ldots, p-2\}$ $+, -, *, /$ mod $(p-1)$

Public parameters $= (p, g) = PP$

A: $x \in_R Z_{p-1}$; $PrK_A = (x)$; $a = g^x \bmod p$: $PuK_A = (a)$

$x \leftarrow rand$

$Z_{p-1} = \{0, 1, 2, \ldots, p-2\}$: $+, -, *, /$ mod $(p-1)$

$1 < m < p-1$: message to be encrypted : $m \in Z_p^*$.

$m \in Z_p^*$ $\quad PuK_A = a = g^x \bmod p \quad c = Enc_a(m) = (\varepsilon, \delta)$

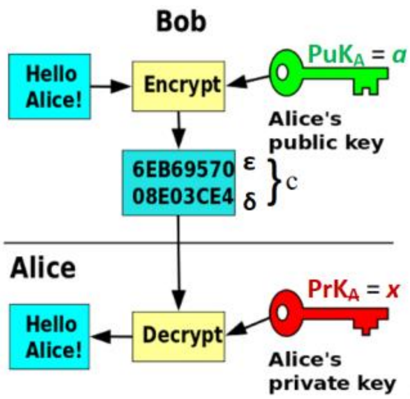
ElGamal Encryption

ElGamal and Schnorr signatures

Asymmetric Encryption - Decryption

c=Enc(PuK_A, m) = (E, D)

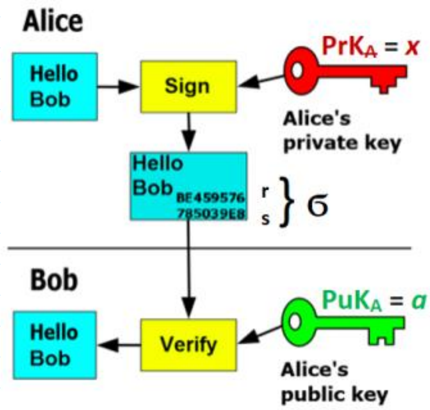
m=Dec(PrK_A, c)



Asymmetric Signing - Verification

Sign(PrK_A, m) = σ = (r, s)

V=Ver(PuK_A, m, σ), V ∈ {True, False} ≡ {1, 0}



$\mathcal{B}$: intends to encrypt message M to $\mathcal{A}$.

$F_{Ecode}(M) = m$

$m \in Z_p^*$; $i \xleftarrow{rand} Z_{p-1}$; $Enc(a, i, m) = c = (\varepsilon, \delta)$.

$\varepsilon = m * a^i \mod p$; $\delta = g^i \mod p \Rightarrow c = (\varepsilon, \delta)$

$\mathcal{B}$: $\xrightarrow{c = (\varepsilon, \delta)}$ $\mathcal{A}$: $PrK_A = (x)$; $Dec(x, c) = m$.

1. $\delta^{-x} = (g^i)^{-x} \mod p =$

$= g^{-ix} \mod p$

2. $m = \varepsilon * \delta^{-x} = m * a^i * g^{-ix} =$

$= m * (g^x)^i * g^{-ix} \mod p =$

$= m * g^{xi} * g^{-xi} \mod p =$

$= m \mod p = m$

since $1 < m < p-1$

Additively inverse element -***x*** to element ***x*** modulo ***p***-1.

```
>> mx=mod(-x,p-1)
```

δ

δ^{-x} mod p computation using Fermat theorem:
If p is prime, then for any integer z holds $z^{p-1} = 1 \bmod p$.

$$\delta^{-x} = \delta^{p-1-x} \bmod p$$

$m = \mathbb{Z}_p^* = \{1, 2, 3, \ldots, p-1\}$; $\varepsilon \in \mathbb{Z}_p^*$; $\varepsilon = m * a^i \bmod p = m * (g^x)^i \bmod p$

$i = \mathbb{Z}_{p-1} = \{0, 1, 2, \ldots, p-2\}$; $\delta \in \mathbb{Z}_p^*$; $\delta = g^i \bmod p$

$$Enc_a(i, m) = (\varepsilon, \delta) = c$$

$$Enc_a : \mathbb{Z}_{p-1} \times \mathbb{Z}_p^* \xleftrightarrow{1-to-1} \mathbb{Z}_p^* \times \mathbb{Z}_p^*$$

$|\mathbb{Z}_p^*| = |\mathbb{Z}_{p-1}|$

```
>> mx = mod(-x, p-1)
>> delta_mx = mod_exp(delta, mx, p-1)
```

$PP = (p, g)$

B: $PuK_A = a$;

A: $PrK_A = x$; $a = g^x \bmod p$.

Multiplicatively Homomorphic Encryption

B:

m_1, m_2 - two messages to be encrypted: $1 < \underbrace{m_1, m_2}^{\mathbb{Z}_p^*} < p-1$.

m_1: $i_1 \leftarrow \text{randi}(\mathbb{Z}_{p-1})$

$$\left.\begin{array}{l} \varepsilon_1 = m_1 * a^{i_1} \bmod p \\ \delta_1 = g^{i_1} \bmod p \end{array}\right\} \xrightarrow{c_1 = (\varepsilon_1, \delta_1)}$$

A: $Dec(x, c_1) = m_1$

m_2: $i_2 \leftarrow \text{randi}(\mathbb{Z}_{p-1})$

$$\left.\begin{array}{l} \varepsilon_2 = m_2 * a^{i_2} \bmod p \\ \delta_2 = g^{i_2} \bmod p \end{array}\right\} \xrightarrow{c_2 = (\varepsilon_2, \delta_2)}$$

$Dec(x, c_2) = m_2$

B: $m = m_1 * m_2 \bmod p$

$i = (i_1 + i_2) \bmod (p-1)$

$$m: \left.\begin{array}{l} \varepsilon = m * a^i \bmod p \\ \delta = g^i \bmod p \end{array}\right\} c = (\varepsilon, \delta)$$

A:

$\mathcal{A}$:

$$c = c_1 * c_2 \bmod p = (\varepsilon_1, \delta_1) * (\varepsilon_2, \delta_2) = (\varepsilon_1 * \varepsilon_2, \delta_1 * \delta_2) =$$
$$= (m_1 * m_2 * a^{i_1} * a^{i_2}, g^{i_1} * g^{i_2}) =$$
$$= (m * a^{i_1 + i_2}, g^{i_1 + i_2}) = (m * a^{i}, g^{i}) = (\varepsilon, \delta) = c$$

$$Enc_a(i_1 + i_2 \bmod (p-1), m_1 * m_2 \bmod p) = c_1 * c_2 \bmod p = c$$

Multiplicative homomorphic encryption means that encryption of multiplication $m_1 * m_2$ of two messages m_1, m_2 is equal to ciphertext c that is equal to the multiplication of two ciphertexts $c_1 * c_2$.

Homomorphic encryption: cloud computation with encrypted data

Zether: Towards Privacy in a Smart Contract World
Financial Cryptography and Data ..., 2020 - Springer

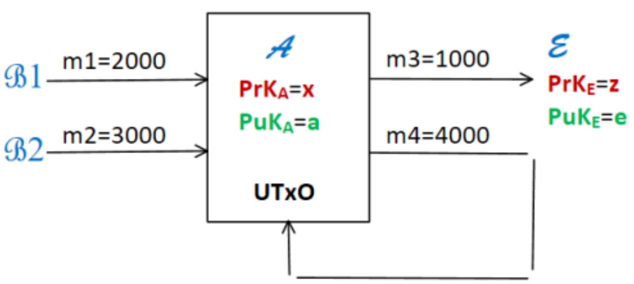
From <https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=Zether%3A+Towards+Privacy+in+a+Smart+Contract+World&btnG=>

Benedikt Bunz1, Shashank Agrawal2, Mahdi Zamani3, and **Dan Boneh**4

1Stanford University, benedikt@cs.stanford.edu
2Visa Research, shaagraw@visa.com
3Visa Research, mzamani@visa.com
4**Stanford University**, dabo@cs.stanford.edu

Ctrl/F --> ElGamal --> Exact mathes **21**

Additively-Multiplicative ElGamal encryption.



How to provide anonymity of transaction amounts and to verify the **balance**: **m1+m2 = m3+m4** ?

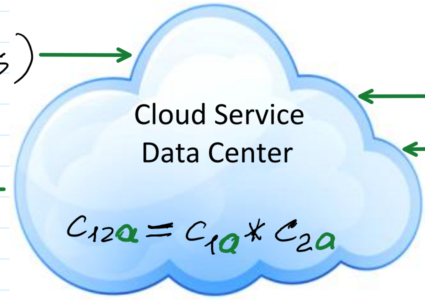
n1= **g**^{m1} mod **p** | **n3**= **g**^{m3} mod **p**
n2= **g**^{m2} mod **p** | **n4**= **g**^{m4} mod **p**

If **m1+m2 = m3+m4**,
Then **n1*n2 = n3*n4**.

$$c_1 \cdot c_2 = c_3 \cdot c_4$$

Query (Total Incomes) →

$c_{12a} = (E_{12a}, D_{12a})$ ←



← c_{1a}

← c_{2a}

$$c_{1a} = (E_{1a}, D_{1a}) = (n_1 * a^{i_1}, g^{i_1})$$
$$c_{2a} = (E_2, D_2) = (n_1 * a^{i_2}, g^{i_2})$$
$$c_{12} = (n_1 * n_2 * a^{i_1+i_2}, g^{i_1+i_2}) = (n_{12} * a^{i}, g^{i})$$
$$i = i_1 + i_2 \mod (p-1)$$

$Dec(x, c_{12a}) = n_{12}$

1. $(D_{12a})^{-x} = (g^{i})^{-x} = g^{-xi} = (g^{x})^{-i} = a^{-i} \mod p$
2. $E_{12a} * (D_{12a})^{-x} \mod p = n_{12} * a^{i} * a^{-i} = n_{12} \mod p$

$$n_{12} = g^{m_1} * g^{m_2} \mod p = g^{m_1+m_2} \mod p.$$

DEF: is one-way function

1) By having p, g and x it is easy to compute $a = g^{x} \mod p$
2) It is infeasible to find x when p, g and a are given!

Zether: Towards Privacy in a Smart Contract World
Financial Cryptography and Data ..., 2020 - Springer

The sums $m_1, m_2, \ldots, m_N$ are restricted in such a way that $m_1 + m_2 + \ldots + m_N \mod (p-1) < 2^{32}$

```
>> int64(2^32)
ans = 4 294 967 296
```

To find the sum $m_1 + m_2 = 2000 + 3000 = 5000 \mod (p-1)$

$$Enc(a, i_1+i_2, n_1 \cdot n_2) = Enc(a, i_1, n_1) \cdot Enc(a, i_2, n_2)$$
$$E_{12} = E_1 \cdot E_2 \mod p = n_1 a^{i_1} \mod p \cdot n_2 a^{i_2} \mod p =$$
$$= g^{m_1} a^{i_1} \mod p \cdot g^{m_2} a^{i_2} \mod p =$$
$$= g^{m_1+m_2} \cdot a^{i_1+i_2} \mod p.$$

$$= g^{m_1+m_2} \cdot a^{i_1+i_2} \bmod p.$$

Let $m_1 + m_2 = m_3 + m_4$

$$\begin{array}{cccc} m_1 & m_2 & m_3 & m_4 \\ \downarrow & \downarrow & \downarrow & \downarrow \\ n_1 & n_2 & n_3 & n_4 \\ \downarrow i_1 & \downarrow i_2 & \downarrow i_3 & \downarrow i_4 \\ c_1 & c_2 & c_3 & c_4 \end{array}$$

If $m_1 + m_2 = m_3 + m_4 \bmod (p-1) \Rightarrow c_1 \cdot c_2 \bmod p = c_3 \cdot c_4 \bmod p.$

$A:\ c_1, c_2 \longrightarrow Dec(x, c_1) = n_1 = g^{m_1} \bmod p$

$Dec(x, c_2) = n_2 = g^{m_2} \bmod p$

Sums $m_1, m_2 < 2^{32}$.

However, g^m needs to be brute-forced to compute ***m***.
We argue that this is not an issue.
First, as we will see, the Zether smart contract does not need to do this, only the users would do it.
Second, users will have a good estimate of ZTH in their accounts because, typically, the transfer amount is known to the receiver. Thus, brute-force computation would occur only rarely.
Third, one could represent a large range of values in terms of smaller ranges.
For instance, if we want to allow amounts up to 32bits, we solve the duality problem for amount ***m***.

% Finds discrete logarithm value corresponding to exponent value i
% by total scan of i from start by step until fin
% p - is a strong prime (Public Parameter)
% g - is a generator (Public Parameter)
% def - is a discrete exponent function value computed by mod_exp(g,i,p)
%   where dl=i is a searchable value of exponent

dlog.m

$p = 11;\ g = 2;\quad m_1 = 1;\ g^{m_1} \bmod p \rightarrow c_1 \rightarrow Dec(x, g^{m_1}) = m_1$

$m_2 = 2;\ g^{m_2} \bmod p \rightarrow c_2 \rightarrow Dec(x, g^{m_2}) = m_2$

$$\underset{\bmod 10}{(m_1+m_2) \bmod (p-1)} = \underset{\bmod 10}{(1+2) \bmod (p-1)} = 3$$

$m_1 = 4,\ m_2 = 9 \Big\}\ (m_1+m_2) \bmod 10 = (4+9) \bmod 10 = 13 \bmod 10 = 3$

Prevention: $m_1 + m_2 \leq (p-1)/2$

$\mathbb{Z}_{p-1}$	0	1	2	3	4	5	6	7	8	9
						± 5	-4	-3	-2	-1

$\mathbb{Z}_{p-1}$	0	1	2	3	4	5	6	7	8	9
						± 5	-4	-3	-2	-1

Till this place