Computation with encrypted data in Data Center.

Existing solution

https://

KAP $\left\{ \begin{array}{l} \xrightarrow{\;t_A\;} \\ \xleftarrow{\;t_B\;} \\ K_{AB} = k = K_{BA} \end{array} \right.$

$\xrightarrow{\;C_Q\;}$

Query: Q
salary for 1 moth
to compute the salary
during the 12 months

$Q : [12, B1, B2]$

$Enc(k, Q) = C_Q$

$Dec(k, C_{sal}) = Sal$

$\xleftarrow{\;C_{sal}\;}$

$Dec(k, Q) = [12, B1, B2]$

$Sal = 12 * (S_{B1} + S_{B2})$

$Enc(k, Sal) = C_{sal}$

Cloud services

$Enc(k, B1) = C_{B1}$

$Enc(k, B2) = C_{B2}$ $\xrightarrow{\;C_{B1},\, C_{B2},\, C_{12}\;}$

$Enc(k, 12) = C_{12}$ $\xleftarrow{\;C_s\;}$ $C_s = C_{12} * (C_{B1} + C_{B2})$

$Dec(k, C_s) = Sal$ Homomorphic encryption

$Sal = 12 * (B1 + B2)$

Omit this part

Let ***G*** and ***H*** be any (algebraic) groups: <***G***, ○>, <***H***, ●> with neutral elements ***e***° and ***e****** respectively.
Definition. The mapping φ is named as homomorphism if for every ***x***, ***y*** ∈ ***G*** there exists ***a***, ***b*** ∈ ***H*** such that

$$\varphi(x \circ y) = \varphi(x) \bullet \varphi(y) = a \bullet b \quad \& \quad \varphi(e^{\circ}) = e^{*} \qquad (I)$$

Definition. The mapping φ is named as homomorphism if for every x, $y \in G$ there exists a, $b \in H$ such that

$$\varphi(x \circ y) = \varphi(x) \bullet \varphi(y) = a \bullet b \quad \& \quad \varphi(e^o) = e^*. \qquad (!)$$

If φ is 1-to-1 mapping then φ is named as isomorphism we denote by ϕ.

Example: $G = Z_{p-1}^+ = \{0, 1, 2, ..., p-2\}$; $<Z_{p-1}^+, + \bmod p-1>$; $e^o = 0 \in Z_{p-1}^+$; $|Z_{p-1}^+| = p-1$.
$H = Z_p^* = \{1, 2, 3, ..., p-1\}$; $<Z_p^*, * \bmod p>$; $e^* = 1 \in Z_p^*$; $|Z_p^*| = p-1$.

We define a function (mapping) ϕ providing an isomorphism ϕ: $Z_{p-1}^+ \rightarrow Z_p^*$.
Modular exponent function for generator g in Z_p^* is defined by equation: $a = g^x \bmod p$
>> mod_exp(g,x,p)

Fermat (little) Theorem. If p is prime, then for any integer z

$$z^{p-1} = 1 \bmod p.$$

Comment. According to Fermat theorem and convention $z^{p-1} = 1 \bmod p$ and $z^0 = 1$.
Then **0** is in some way equivalent to p-1 when we perform a computations in exponent mod p.
This equivqlence we can define in an unique way

$$p-1 = \mathbf{0} \bmod (p-1).$$

Indeed (p-1) mod (p-1) = **0** since the reminder of division of (p-1) by module (p-1) is equal to **0**.

Corollary. For all x, y, $z \in Z_{p-1}^+$ the exponent operations performed in Z_p^* in general must be performed mod (p-1) to avoid a mistakes for more complicated expressions, e.g.

$$g^{z(x+y) \bmod (p-1)} \bmod p = g^{(zx+zy) \bmod (p-1)} \bmod p = (g^{zx \bmod (p-1)} \bmod p * g^{zy \bmod (p-1)}) \bmod p.$$

Let Z be a set of positive integers $Z = \{0, 1, 2, 3, ... \infty\}$. And let p=11.
Integers taken mod p-1 are mapped to the set $Z_{p-1}^+ = \{0, 1, 2, ..., p-2\}$.
If p=11, then p-1=10 and we obtain $Z_{10}^+ = \{0, 1, 2, ..., 9\}$ which is an additive group $<Z_{10}^+, +>$.
Interresting observation: please verify that mapping $\varphi_{\bmod 11}: Z \rightarrow Z_{10}^+$ is a homomorphism, where $Z = \{0, 1, 2, 3, ... \infty\}$ we are interpretting as infinite additive group of integers: $<Z, +>$.
This result can be generalized for any mapping $\varphi_{\bmod n}: Z \rightarrow Z_n^+$, where n is any finite positive integer and Z_n^+ is and additive group with addition operation mod n, i.e. $<Z_n^+, +>$.

Let p is prime and g is a generator in Z_p^*.
Then modular exponent function for generator g in Z_p^* and defined by equation:

$$a = g^x \bmod p. \qquad (!!)$$

is a mapping ϕ: $Z_{p-1}^+ \rightarrow Z_p^*$.

Example. Let p=11 then p-1=10, then $Z_{10}^+ = \{0, 1, 2, ..., 9\}$ and $Z_p^* = \{1, 2, 3, ..., 10\}$. Then the generator in in Z_p^* is g=2. Check it.

Theorem. Function (mapping) ϕ: $Z_{p-1}^+ \rightarrow Z_p^*$ is an isomorphism.
Proof. ⊳ 1. According to Fermat theorem ϕ is 1-to-1 mapping since $|Z_{p-1}^+| = p-1 = |Z_p^*|$ and g is a

Theorem. Function (mapping) $\phi: Z_{p-1}^{+} \to Z_p^{*}$ is an isomorphism.

Proof. ⊳ 1. According to Fermat theorem ϕ is 1-to-1 mapping since $|Z_{p-1}^{+}| = p-1 = |Z_p^{*}|$ and g is a generator of Z_p^{*}, i.e. it generates all the values in Z_p^{*} by powering with integers in Z_{p-1}^{+}. Looking deeper it is a consequence of Lagrange theorem of algebraic groups. Dan Boneh

2. Now we prove equation (!). Taking into account that modular exponent function is defined by the generator g as a parameter we denote it by

$$\phi_g(x) = a = g^x \bmod p. \qquad (!!!)$$

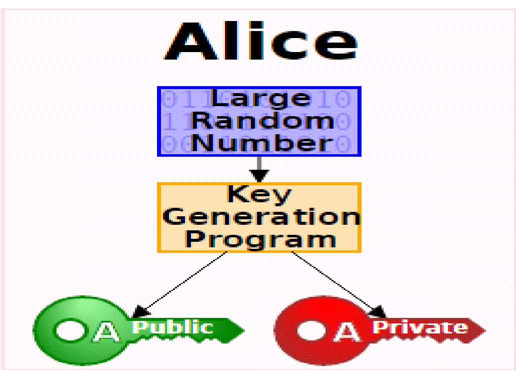
For all x, $y \in G = Z_{p-1}^{+} = \{0, 1, 2, \ldots, p-2\}$ there exists a, $b \in H = Z_p^{*} = \{1, 2, 3, \ldots, p-1\}$ such that $a = g^x \bmod p$ and $b = g^y \bmod p$.

Then the following identities takes place analogous to the identities of ordinary exponent function

$\phi_g(x+y) = g^{x+y} \bmod p = (g^x \bmod p * g^y \bmod p) \bmod p = \phi_g(x) * \phi_g(y) = g^x * g^y \bmod p = a * b \bmod p.$

$$\phi_g(e^{\circ}) = \phi_g(0) = g^0 \bmod p = 1 \bmod p = 1 = e^{*} \in Z_p^{*}$$

The theorem is proved ◄.



$PP = (p, g)$

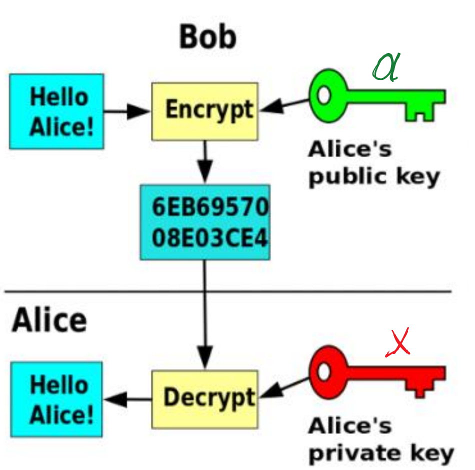
2.Key generation

- Randomly choose a private key x with $1 < x < p - 1$.
- Compute $a = g^x \bmod p$.
- The public key is **PuK** $= a$.
- The private key is **PrK** $= x$.

Asymmetric Encryption - Decryption

c=Enc(PuK$_A$, m)

m=Dec(PrK$_A$, c)



$A:$ $\xrightarrow{PuK_A = a}$ $B:$ is able to encrypt m to A: $m < p$

$B:$ $r \leftarrow randi(\mathbb{Z}_p^*)$

$$\left.\begin{aligned} E &= m \cdot a^r \bmod p \\ D &= g^r \bmod p \end{aligned}\right\} c = (E, D) \longrightarrow$$

A: is able to decrypt $c = (E, D)$ using her $PrK_A = x$.

1. $D^{-x \bmod (p-1)} \bmod p$
2. $E \cdot D^{-x} \bmod p = m$

$(-x) \bmod (p-1) = (0 - x) \bmod (p-1) =$
$= (p - 1 - x) \bmod (p-1)$

D^{-x} mod ***p*** computation using Fermat theorem:
If ***p*** is prime, then for any integer ***a*** holds ***a^{p-1}* = 1 mod *p***.

$D^{p-1} = 1 \bmod p \quad / \cdot D^{-x}$

$\underbrace{D^{p-1} \cdot D^{-x}} = 1 \cdot D^{-x} \bmod p \Rightarrow \underbrace{D^{p-1-x}} = D^{-x} \bmod p$

$$\boxed{D^{-x} \bmod p = D^{p-1-x} \bmod p}$$

Homomorphic property of ElGamal encryption

Let we have 2 messages m_1, m_2 to be encrypted

$r_1 \leftarrow randi(\mathbb{Z}_p^*)$

$Enc_a(r_1, m_1) = (E_1, D_1) = c_1$

$E_1 = m_1 \cdot a^{r_1} \bmod p$

$D_1 = g^{r_1} \bmod p$

$r_2 \leftarrow randi(\mathbb{Z}_p^*)$

$Enc_a(r_2, m_2) = (E_2, D_2) = c_2$

$E_2 = m_2 \cdot a^{r_2} \bmod p$

$D_2 = g^{r_2} \bmod p$

Multiplicative homo rphic encryption:

$Enc_a(r_1 + r_2, m_1 \cdot m_2) = Enc_a(r_1, m_1) \cdot Enc_a(r_2, m_2)$

$c_{12} = c_1 \cdot c_2$

$$c_{12} = c_1 \cdot c_2$$

$$(E_{12}, D_{12}) = (E_1, D_1) \cdot (E_2 \cdot D_2)$$

$$(E_1 \cdot E_2, D_1 \cdot D_2)$$

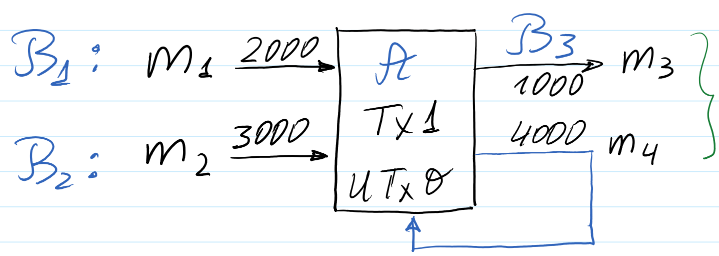
$$(m_1 \cdot m_2 \cdot a^{r_1+r_2} \bmod p, g^{r_1+r_2} \bmod p) = (m_1 a^{r_1} \bmod p, g^{r_1} \bmod p) \cdot (m_2 a^{r_2} \bmod p, g^{r_2} \bmod p)$$

Additively multiplicative encryption:

Let n_1, n_2 are messages to be encrypted

$$\left.\begin{array}{l} Enc_a(r_1, n_1) = c_1 \\ Enc_a(r_2, n_2) = c_2 \end{array}\right\} \quad c_1 \cdot c_2 = c_{12}^{\oplus} = Enc_a(r_1 + r_2, n_1 + n_2)$$

1. App.: for confid & verifiable transactions



$m_1 + m_2 = m_3 + m_4 = 5000$

$$Enc(m_1 + m_2) = c_{12} = c_{34} = Enc(m_3 + m_4)$$

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

ElGamal-Enc : $PP = (p, g)$ A: $PrK = x$; $PuK = a = g^x \bmod p$

B_1: $n_1 = g^{m_1} \bmod p \longrightarrow Enc_a(r_1, n_1) = c_1 = (E_1, D_1) = (n_1 a^{r_1} \bmod p,$

B_2: $n_2 = g^{m_2} \bmod p \longrightarrow Enc_a(r_2, n_2) = c_2 = (E_2, D_2) = (n_2 a^{r_2} \bmod p,$

Net: $c_1 \cdot c_2 = c_{12} = (E_{12}, D_{12}) = (E_1 \cdot E_2, D_1 \cdot D_2)$

$E_{12} = E_1 \cdot E_2 = n_1 a^{r_1} \cdot n_2 a^{r_2} \bmod p = n_1 \cdot n_2 \cdot a^{r_1+r_2} \bmod p =$

$$= g^{m_1} \cdot g^{m_2} a^{r_1 + r_2} \bmod p = g^{m_1+m_2} \cdot a^{r_1+r_2} \bmod p$$

$$c_{12} = g^{m_1+m_2} \cdot a^{r_1+r_2} \bmod p$$

B_1: $\left.\begin{matrix} c_1 \\ Enc_a(r_{11}, r_1') = c_{r1} \end{matrix}\right\} \longrightarrow$ A: $Dec_x(c_1) = n_1 = g^{m_1} \bmod p$

Verifies if expected sum m_1 corresponds to the value $n_1 = g^{m_1} \bmod p$.

B_2: $\left.\begin{matrix} c_1 \\ Enc_a(r_{22}, r_2') = c_{r2} \end{matrix}\right\} \longrightarrow$ A: dos the same

A: computes $E_{12} = E_1 \cdot E_2 = g^{m_1+m_2} \bmod p \xrightarrow{E_{12}}$ Net

Encrypts value n_3

$r_3 \leftarrow$ rand for enc. value n_3

$$n_3 = g^{m_3} \bmod p$$

$$Enc_a(r_3, n_3) = c_3 = (E_3, D_3) = (n_3 a^{r_3} \bmod p, g^{r_3} \bmod p)$$

computes $r_4 = r_1 + r_2 - r_3 \bmod (p-1) \Rightarrow r_1 + r_2 = r_3 + r_4 \bmod (p-1)$

Encrypts value n_4

$$n_4 = g^{m_4} \bmod p$$

$$Enc_a(r_4, n_4) = c_4 = (E_4, D_4) = (n_4 a^{r_4} \bmod p, g^{r_4} \bmod p)$$

Declares c_3, c_4 to the Net

Net verifies if $\boxed{c_1 \cdot c_2 = c_3 \cdot c_4}$

Till this place

Homomorphic encryption: cloud computation with encrypted data.

Paillier encryption scheme is additively-multiplicative homomorphic and has a potentially nice applications in blockchain, public procurement, auctions, gamblings and etc.

$Enc(Puk, m_1+m_2) = c_1 \bullet c_2$.