> 患者盲化自己的口令后发送给每一个密钥服务器请求servers-hardening password (spw)
>
> 在向密钥服务器注册时,密钥服务器为患者生成一个spw
>
> 在向云服务器注册时,用spw加密自己的私钥并上传
>
> 在向医院注册时,医院为患者部署一个智能合约

blindPw($pw_{\mathcal{P}}$**)** {

$r \leftarrow^{\$} Z_p^*$;
$pw_{\mathcal{P}}^* \leftarrow r \cdot H_1(pw_{\mathcal{P}})$;
send $(ID_{\mathcal{P}}, pw_{\mathcal{P}}^*)$ to $\mathcal{KS}_i$;

}

genSpw($ID_{\mathcal{P}}, pw^*, t$**)** {

if $ID_{\mathcal{P}}$ not in local storage {
$f_i(x) \leftarrow^{\$} a_{i,0} + a_{i,1}x + \ldots + a_{i,t-1}x^{t-1}$;
send $\{a_{i,\gamma}P, f_i(j)\}$ to $\mathcal{KS}_j$; //$\gamma = 0, 1, \ldots, t-1; j = 1, 2, \ldots, n, j \neq i$
}

if $f_j(i)P = \sum_{\gamma=0}^{t-1} i^{\gamma} \cdot a_{j,\gamma}P$ {
$s_i = \sum_{j=1}^{n} f_j(i)$;
$Q_i = s_i P$;
}
}

getSpw($\sigma_1^*, \sigma_2^*, \ldots, \sigma_t^*, t$**)** {

if $e(\sigma_i^*, P) = e(pw_{\mathcal{P}}^*, Q_i)$ {
if $e(\sigma_{\mathcal{P}}, P) = e\big(H_1(pw_{\mathcal{P}}), Q\big)$ {
$spw_{\mathcal{P}} = \hbar(\sigma_{pw} || pw_{\mathcal{P}})$;
}
}
}

register_P($ID_{\mathcal{P}}, pw_{\mathcal{P}}, ID_{\mathcal{KS}_i}, ID_{\mathcal{CS}}, ID_{\mathcal{H}}$**)** {

$pw_{\mathcal{P}}^*$ =blindPw($pw_{\mathcal{P}}$);
$spw_{\mathcal{P}}$ =getSwp(...);

$au_i = \hbar(ID_{\mathcal{KS}_i} || spw_{\mathcal{P}})$; //$i = 1, 2, \ldots, n$
send $(ID_{\mathcal{P}}, au_i)$ to $\mathcal{KS}_i$; //$i = 1, 2, \ldots, n$

$au_{\mathcal{CS}} = \hbar(ID_{\mathcal{CS}}||spw_{\mathcal{P}})$;
$csk_{\mathcal{P}} = \mathrm{E}(spw_{\mathcal{P}}, sk_{\mathcal{P}})$;
send $\{csk_{\mathcal{P}}, (ID_{\mathcal{P}}, au_{\mathcal{CS}})\}$ to $\mathcal{CS}$;

$au_{\mathcal{H}} = \hbar(ID_{\mathcal{H}}||spw_{\mathcal{P}})$;
send $(ID_{\mathcal{P}}, au_{\mathcal{H}})$ to $\mathcal{H}$;
}

register_KHC(ID, au) {

if ID not in localStorage {
store (ID, au);
}

if $\mathcal{CS}$ {
store $csk_{\mathcal{P}}$;
}
if $\mathcal{H}$ {
$Add_{scp} \leftarrow$ deploy $SC_{\mathcal{P}}$;
store Add_{scp};
}
}

$SC_{\mathcal{P}}$**: smartContract()** {

$i \leftarrow 0$;
function Storage(string memory _data) {
data[i] = _data;
i++;
}

function Audit(i) {
return data[i];
}
}