0.1 Trivial,XOR,Modular Secret Sharing

Lo scopo del secret sharing è quello di condividere un segreto tra differenti parti senza che i singoli possano risalire al vero segreto a patto che un certo numero di esse forniscano la propria porzione di segreto.Chi distribuisce le porzioni del segreto (**share**),viene detto **dealer** e chi riceve il singolo share viene detto parte.
Inoltre, vogliamo raggiungere l'obiettivo del **perfect secrecy**:un avversario, conoscendo n-1 shares non deve essere ancora in grado di indovinare il segreto e, in altri termini, noti n-1 shares la probabilità di conoscere il segreto deve essere uguale a non conoscerne nessuno.
Un primo approccio che si potrebbe utilizzare per implementare questo schema può essere quello descritto da *Trivial secret sharing*. In particolare, supponiamo, per semplicità, di avere un dealer che deve condividere un segreto e due parti che dovranno ricevere lo share. Questo schema viene detto trivial poiché il dealer prende il segreto, lo divide in parti uguali in base al numero delle parti e condivide il segreto. Questa soluzione non viene adottata poiché non è sicura in quanto, conoscendo una parte la probabilità che un attacker possa indovinare il segreto diminuisce in maniera esponenziale: $\frac{1}{2^{\frac{Nbit}{h}}}$ in cui k è il numero degli share.
Il primo dei due metodi leggermente più raffinati che implementano un One Time Pad (Vernam cipher), ma che non sono implementabili a causa di falle su aspetti di sicurezza è quello ottenuto da *XOR Secret sharing*.
Il dealer prende il segreto, genera delle quantità casuali di cui ne viene fatto lo XOR con il segreto e distribuisce alle parti le quantità casuali e il risultato dello XOR. Al momento della ricostruzione si effettua nuovamente l'operazione di XOR per ottenere il segreto iniziale. In particolare, questa soluzione risulta più sicura dato che la conoscenza di una parte del risultato dello XOR o del solo valore casuale non mette a rischio la password iniziale.
Il secondo metodo che funziona come un One Time Pad (Vernam cipher) è quello dato dal *Modular Secret Sharing* in cui al posto dell'operazione di XOR si utilizza l'operazione del modulo.In particolare, il dealer prende il segreto ne genera, genera un valore casuale modulo N e effettua S-Valore Casuale modulo N;infine, distribuisce come shares il valore S-RAND e RAND.A questo punto per ricostruire il segreto, un numero sufficiente di parti deve condividere il proprio share al fine di effettuare la somma e ricostruisce il segreto.

0.2 Shamir Secret Sharing

L'idea di fondo dello schema di Shamir per la condivisione di un segreto è quella di permettere la ricostruzione del segreto conoscendo un certo numero di shares non strettamente uguale al numero di parti.Questo viene detto schema (t,n) in cui t sono il numero di parti/shares necessari per ricostruire il segreto ed n sono il numero delle parto e quindi degli shares condivisi.
In fase preliminare supponiamo di avere un dealer che deve condividere il segreto con 2 di n parti (schema (t,n)).
Rappresentiamo la coppia dei due shares come coordinate in un piano:se conoscessimo solamente uno shares individueremmo un fascio di rette nel piano poiché soddisfano le coordinate della parte,ma conoscendo un'altro share individueremmo una singola retta nel piano che, riportata lungo l'asse y, ci consentirebbe di ricostruire il segreto dato che la retta in questione viene parametrizzata come $y = a * x + s$ con a numero casuale e s il segreto.
Ora, se vogliamo estendere questa costruzione nel caso in cui gli shares necessari a ricostruire il segreto fossero più di due, dovremmo generare un polinomio di grado t-1 con termine noto s e coefficienti delle x generati casualmente.Il dealer, quindi, genera i relativi shares assegnando una coordinata x e il corrispondente valore della y.Lo share così composto (x,y) viene consegnato alle parti. Ora, per ricostruire il segreto occorre raccogliere t-1 shares per annullare il polinomio e trovare il termine noto.In particolare, per fare ciò occorre interpolare i punti tramite la formula di interpolazione di Lagrange $y = \sum_i \Lambda x_i y_i$ con il termine $\Lambda x_i = \prod_{shares\ x_k \neq x_i} \frac{x - x_k}{x_i - x_k}$ ponendo il valore $x = 0$.
Come è facile vedere, questa costruzione non rispetta il requisito di sicurezza del perfect secrecy poiché se non conoscessimo uno shares di t-1 ci basterebbe ciclare sull'ultimo share per scoprire il segreto.Infatti, nell'applicazione reale di questo schema si utilizzano le operazioni modulari in cui ogni divisione deve essere intesa come il modular inverse effettuato tramite l'algoritmo euclideo e il numero con cui effettuare il modulo deve essere primo e le operazioni del segreto e del polinomio devono essere effettuate nel campo dei numeri primi.
Le prerogative di questo schema è che gli shares devono essere più piccoli del segreto, ma al più grandi come il segreto questo perché ogni shares deve aggiungere un contenuto informativo e quindi diminuire l'entropia del segreto.Lo schema Blackley ammette chiavi più grandi del segreto.

0.3 Proprietà Homomorphic

La proprietà Homomorphic è una proprietà che riguarda in via più generale l'algebra, ma, applicata al secret sharing, è una proprietà che consente di calcolare il segreto (dato dalla somma di altri 2) sia tramite la somma degli shares corrispondente al segreto i-esimo sia tramite la somma degli share di entrambi i segreti.

0.4 Secret Sharing Multiparty Computation

+ Tutti i modelli visti fino ad ora presupponevano che un dealer sia fidato poiché è colui che genera gli shares per le parti. Con SMC o anche detto Secure Multiparty Computation, realizziamo uno schema in cui è possibile trasmettere un valore senza essere a conoscenza dei dati in ingresso.
In particolare, si prendono N parties dette Input peer con valore z_i, inviano il proprio shares z_i e un Privacy Peer calcola il valore delle funzione che prende come argomento gli shares e poi ne pubblica il risultato.Il difetto di questo schema preliminare è che si devono avere come minimo 3 input peers e di 2 privacy peers.Quindi, lo schema risultato è il seguente con l'assunzione di effettuare le operazioni col modulo e che si voglia mettere una soglia su quante parti devono cedere il proprio shares per ottenere il segreto.
Gli input peer i prendono il dato z_i, generano un polinomio $p(x)$ di grado $t-1$ con termine noto il proprio segreto ed invia ad ogni privacy peer gli shares ottenuti come punti all'interno del polinomio scelto.A questo punto, i privacy peer ricevono gli shares da tutti gli input peer, ne fanno la somma e generano il valore RES che potrà essere pubblicato.Per ricostruisce il segreto ti utilizza l'interpolazione di Lagrange, una volta noti un numero sufficiente di shares.

0.5 Verifiable Secret Sharing

Il Verifiable Secret Sharing è tipo di secret sharing il cui obiettivo è quello di determinare qualora una delle parti o il dealer trasmettono uno share/segreto in maniera consistente e veritiera.Ciò allarga la visione di tutti i modelli fino ad ora vista (honest but curious model) in cui l'attaccante/cheater è solamente un agente esterno ad un visione in cui il cheater può trovarsi all'interno della comunicazione e trasmettere dati non corretti.

0.6 Feldman VSS Scheme

Il Feldman Scheme è uno schema di verifiable secret sharing che funziona come segue:

- Si genera un polinomio $p(x)$ di grado $t-1$ e termine noto s;
- Si distribuisce lo share x_i, y_i ad ognuna delle n parti;
- Si pubblicano i Feldman commitment $c_0 = g^s \mod p, c_1 = g^{a_1} \mod p, \ldots, c_{t-1}?g^{a_{t-1}} \mod p$;
- Le parti per verificare se uno share è valido, ricevono x_i, y_i;
- Le parti calcolano, grazie ai commitment pubblici

$$g^{p(x_i)} = c_0^{x_i} c_1^{x_i^2} \cdots c_{t-1}^{x_i^{t-1}} = g^s g^{a_1 x_1} g^{a_2 x_2^2} \cdots g^{a_{t-1} x_i^{t-1}}$$

- La parte calcola inoltre g^{y_i}
- Se i due valori calcolati sono uguali, allora lo share è verificato.

La pubblicazione dei commitment non rivela nulla sul segreto se il numero p è prima e i coefficienti del polinomio sono presi nei Large Fields quindi ci basiamo sulla proprietà homomorphic e sul problema del Dlog. Tutto questo conto viene fatto senza sapere quale sia effettivamente il segreto.

0.7 Commitments

Un commitment è una costruzione crittografica che consente di generare un qualcosa che non può essere modificato e quindi una volta pubblicato non si hanno si può modificare. Per essere valido deve possedere due proprietà:

- Hiding: è la proprietà che, una volta ricevuto un commitment, esso non deve rivelare nulla sul segreto (COMMIT PHASE);
- Binding: è la proprietà per cui l'unico modo per risalire al segreto contenuto nel commitment è possedere il valore segreto (REVEAL PHASE);

Inoltre, è impossibile ottenere entrambe le proprietà valide in contemporanea: è impossibile essere perfectly hiding e perfectly binding, ma solamente perfectly hiding/binding e computationally hiding/binding.Un esempio è la funzione hash che è sia hiding che binding, ma in maniera computazionale.Il commitment di Feldman è computationally hiding poiché è difficile capire quale sia x do $c = g^x \mod p$ se p è nei numeri primi e x grande ed è perfectly binding poiché il destinatario non può trovare un x' tale che $g^{x'} = c$

0.8 Pedersen Commitment

Il difetto dello schema di Feldman è che è perfectly binding e computationally binding. Tramite il Pedersen commitment otteniamo uno schema che sia perfectly hiding e computationally binding e che possegga la homomorphic property.Definiamo questo commitment come $c = g^a h^h \mod p$.
Rispetta la proprietà homomorphic poiché:

$$Commit(a+b, r+r') = g^{a+b}h^{r+r'} = g^a h^r \cdot g^b h^{r'} = Commit(a,r) \cdot Commit(b,r')$$

Andiamo ora a verificare che si perfectly hiding e computationally. Nel primo caso dobbiamo verificare che non è possibile trovare un $a' \neq a$ tale che $Commit(a,r) = g^a g^r = g^{a'} h^{r'} = Commit(a',r')$ e data la proprietà del DLog possiamo nascondere un qualsiasi valore di a senza rivelare quale sia effettivamente il segreto.
Inoltre, per mostrare che è computationally binding notiamo che possiamo fornire una trapdoor affinché si possa ottenere il segreto.In particolare, noto a,r,r' costruiamo un valore $h = g^w \quad w = \log_g h$ che funge da trapdoor del commitment nel seguente modo:

$$\begin{aligned} g^a h^r &= g^{a'} h^{r'} \\ g^a g^{wr} &= g^{a'} g^{wr} \\ g^{a+wr} &= g^{a'+wr'} \\ a + wr &= a' + wr' \mod q \\ r' &= w^{-1}(a - a' + wr) = w^{-1}(a-a') + r \end{aligned}$$

Lo schema funziona nel seguente modo:

- Genera due polinomi $f(x), f'(x)$ con rispettivi termini noti s,r;
- Per ogni parte genera la tripletta x_i, y_i, z_i;
- Pubblica i commitment di pedersen $c_0 = g^s h^r, c_{t-1} = g^{a_{t-1}} h^{b_{t-1}}$
- La parte riceve la tripletta x_i, y_i, z_i;
- Per verificare che uno share sia valido, calcola $g^{fx_i}, g^{f'(x_i)'}$ tramite il prodotto dei commitment $c_0 \cdots c_{t-1}$
- verifica che sia uguale a $g^{y_{x_i}} h^{z_{x_i}}$: in caso affermativo lo share è valido.

0.9 Distributed Key Generation

Il distributed key generation è uno schema che ci consente di avere una chiave pubblica K e una chiave privata Sk senza che le parti la conoscono.Questo schema ha senso in tutti i casi in cui vogliamo distribuire una chiave.IN particolare ognuna delle parti genera il suo segreto e calcola il polinomio corrispondente, calcola gli shares da dare alle parti e le invia.Ora, ogni parte si calcola y data come la somma di tutti gli share ricevuti dalle altre parti.Inoltre, ogni parte pubblica il corrispondente commitment e lo invia a tutte le parti.Infine, per calcolare la chiave pubblica si effettua il prodotto di tutti i commitment ricevuti cosi da ottenere g^s, cioè la chiave pubblica.

0.10 Multiplicative Group $\mod p$

Un gruppo $(G, \circ)$ è una struttura composta da un insieme di elementi G e una operazione $\circ$ che gode delle seguenti proprietà:identità,inversa,associatività e chiusura. Inoltre, un gruppo viene detto Abeliano se gode anche della proprietà commutativa.Un esempio di gruppo Abeliano è il gruppo moltiplicativo $\mathbf{Z}_p$ composto da $p-1$ elementi e la cui operazione è $\mod p$.Di questo gruppo appartiene l'operazione exponentiation che consiste nella ripetizione di k volte della stessa operazione.
Il generatore di un gruppo è un valore g tale che $g^0, g^1, \ldots, g^{p-1}$ appartengono al gruppo.Se m è un numero primo allora qualsiasi membro del gruppo è un generatore ad eccezione dell'identità.
Definiamo strong prime un numero primo p tale che p=2q+1 con q numero primo.Da cui ne consegue che ogni strong prime genera un sottogruppo di ordine q.
Un numero x è un quadratic residue subgroup se ammette la radice quadrata in Zp: $x \equiv a^2 \mod p$.
Un modo facile per verificare che un numero appartiene a quadratic residue subgroup è tramite la formula di Legendre:$a \in QR \; if \; a^{\frac{p-1}{2}} \mod p = 1$ e si forma un sotto gruppo di ordine $\frac{p-1}{2}$.

0.11 Threshold Encryption

la threshold cryptography è tipo di crittografia in cui l'encryption o una signature può essere decriptata se e solo se vi sono un certo numero di partecipanti. Lo schema più famoso è quello di El-Gamal sulla crittografia asimmetrica: si cifra il messaggio con un qualsiasi cipher a chiave simmetrica, si genera k ed infine si invia il messaggio cifrato con la chiave k cifrata secondo lo schema di cifratura asimmetrica di El-Gamal.

0.12 ECIES:Hybrid Encryption

ECIES sta per Elliptic Curve Integrated Encryption Scheme e venne per la prima volta utilizzato nell'IMSI per autenticare la SIM dell'utente.Quindi, presupponiamo che la SIM abbia pre-installato la chiave pubblica del Home Network g^{HN} e per inviare i dati cifrati eseguiamo i seguenti passaggi dello schema Encrypt-then-MAC:

- Generiamo il valore casuale x e calcoliamo g^x;
- utilizziamo un HKDF per generare la chiave $K = HKDF(g^{HNx})$ e cifriamo il dato con un qualsiasi cipher a chiave simmetrica;
- Aggiungiamo l'integrità utilizzando HMAC;
- Inviamo $(g^x, HMAC(AES_k MSG))$;

Per decifrare, l'home network riceverà $(g^x, HMAC(AES_k MSG))$:

- Tramite la sua chiave privata HN calcola $(g^x)^{HN}$;
- Deriva la chiave tramite HKDF $K = HKDF(g^{HNx})$;
- Decifra il messaggio in ingresso.

0.13 Threshold El-Gamal

L'approccio di El-Gamal nella crittografia con soglia consiste nel modificare il protocollo di Diffie-Hellman per distribuire la chiave privata necessaria per decifrare il messaggio e ricostruire il segreto solamente quando vi sono un tot di partecipanti che condividono il segreto. Ricordiamo che per cifrare occorre effettuare (g^r, mh^r) e per decifrare, posto $h^r = (g^s)^r$, la parte si calcola il segreto s ed effettua $m = \frac{c}{g^{rs}} = \frac{mg^{rs}}{g^{sr}}$.In particolare ogni parte ottiene (x_i, y_i), prendono g^r dal messaggio che è stato ricevuto, si calcolano i coefficienti di Lagrange ed ottengono g^{rs} necessaria per decifrare.

0.14 Threshold Signature

La threshold signature serve ogni qual volta si vuole garantire la validità di un dato se è stato verificato da un certo numero di parti e,in particolare, se non si raggiunge il numero necessario di partecipanti non si può forgiare nessuna firma.

0.15 RSA Signature

Lo schema classico di funzionamento di RSA è il seguente:

- Genera due numeri primi grandi p,q;
- Calcola il modulo di RSA come N=pq;
- Prendi $\Phi(N) = (p-1)(q-1)$;
- Calcola $1 < e < \Phi N$;
- Genera la chiave privata d tale che de=1 modulo N;
- per cifrare il messaggio $C = (M)^e \mod N$;
- Pre decifrare il messaggio $M = (C)^d \mod N$;

Per adattarlo in un contesto di firma con threshold, manteniamo tutta la generazione del segreto d, ma con la differenza che per decifrare dobbiamo avere almeno t parti.Quindi, generiamo un polinomio di grado t-1 e con termine noto il segreto, calcoliamo gli shares e li trasmettiamo e per ricostruire la firma,prendiamo il messaggio e lo eleviamo all'interpolazione di lagrange con le y.Questo schema ha una problematica dovuta al calcolo del denominatore del coefficiente di Lagrange dato che non è sempre possibile calcolare l'inversa o non è proprio possibile calcolarlo.Per risolvere questo problema si ricorre allo schema di Shoup che si basa sull'osservazione che il denominatore del coefficiente di lagrange è fattorizzazione di $i!(i-L)!$, quindi definiamo il nuovo coefficiente di lagrange come $L!\Lambda(x)$ che, nel momento della ricostruzione restituirà sicuramente un numero intero.Infine, per ricostruire la firma effettuiamo $H(M)^{d \cdot L!} \mod N$.

0.16 RSA Common Modulus Attack

L'idea dell'RSA Common Modulus Attack è quella di sfruttare la debolezza nell'aver inviato un messaggio con due chiavi pubbliche differenti.Note le chiavi pubbliche che sono coprimi, tramite l'algoritmo di Euclide ci possiamo calcolare $e_a x + e_b y = 1$ e elevare i corrispondenti messaggi per i valori x e y ed ottenere $m^{e_a x} \cdot m^{e_b y} = m^{e_a x + e_b y} = m$.
Nella versione shared utilizziamo la stessa idea ma in questo caso abbiamo:

- $L! = \Delta$;
- $H(m)^{dL!} = H(m)^{d\Delta}$
- Ponendo $H(m)^d = y \to y^{\Delta}$;
- Sapendo $H(m)^d = (H(m)^e)^d = y^e$;
- Effettuiamo l'attacco su Δ, e;

0.17 MacKenzie+Reiter

Il Mackenzie-Reiter è uno schema che consente di assicurarsi che il dispositivo che si sta utilizzando sia effettivamente utilizzato dal suo proprietario.Questa problematica viene definita mobile device resilient to capture e si compone di 3 attori:l'utente con la password P,il dispositivo con la sua chiave pubblica e privata e il server con chiave pubblica e privata. La soluzione che si dovrà proporre deve essere robusta e quindi garantire sicurezza almeno se una delle 3 parti vengano crack e nel caso in cui parliamo del server esso è untrusted.
Nella versione base del protocollo abbiamo che nel momento della registrazione del dispositivo si effettuano delle operazioni per cui alcuni dati tra cui la chiave pubblica del server, la privata del dispositivo,la password utente,l'hash della password e il segreto cifrato, vengono cancellati appena calcolati per poi generare un ticket che contiene un valore casuale, l'hash della password e il segreto cifrato (ottenuto da XOR tra chiave segreta del dispositivo e un PRF con un nonce v e la password). Nel momento in cui si riceve il ticket e si vuole recuperare la password, il server prende il nonce dal ticket, verifica l'integrità del messaggio controllando il mac, controlla la password originale sia uguale a quella ricevuta.Così facendo ci siamo assicurati che sia l'utente che il device sono reali e il server ritorna lo xor fra p e s con cui il dispositivo recupera la sua chiave privata.Con questo modello ci siamo protetti da attacchi di tipo:

- Server cracked e la password nota poiché la chiave del dispositivo è ancora segreta e non ottenibile;
- Dispositivo cracked o perso, l'attacher può creare una hash ed effettuare un dictionary attack,facilmente verificabile tramite il MAC;
- Dispositivo e server crackato in cui manca solamente la password e si può effettuare un dictionary attack online facilmente riconoscibile.

Quello che manca ora è il proteggersi da un device stolen e password nota, ma si richiede un protocollo simile a quello appena proposto, ma con qualche accorgimento.

0.18 Linear Secret Sharing & Access Control Matrix

Sono esercizi Vari

0.19 Elliptic Curve Crypto

Dato che le operazioni in modulo classico del DLOG, nei gruppi possono essere risolti in maniera polinomiale, si introducono le curve ellittiche che, nella sua forma generale di Weierstrass, sono curve cubiche nella forma $y^2 = x^3 + ax + b \quad where\ 4a^3 + 27b^2 \neq 0$.
Dato questo insieme di punti dobbiamo definire l'operazione che si effettua in questo gruppo candidato.L'operazione che contraddistingue questo gruppo e che dovrà soddisfare la proprietà di chiusura,identità,inversa e associatività è l'addizione.Per effettuare questa operazione ipotizziamo di dover effettuare la somma fra due punti nella curva, tracciamo la retta passante fra i due punti.Essa intersecherà necessariamente la curva in un altro punto -R se così non fosse, cioè nel caso in cui si somma il punto per se stesso si prende la tangente alla curva, oppure quando i punti sono simmetrici lungo l'asse x allora si prende un punto all'infinito detto O=-O (denominato come zero poiché P+O=-P).Una volta ottenuto il punto -R si traccia una retta verticale da quel punto che, intersecando la curva in un altro punto detto R, ci restituisce il risultato di tale operazione.
La presenza del punto O=-O ci permette di verificare la chiusura dell'insieme per quanto riguarda l'inversa la prendiamo come il punto -P poiché P-P=O e viceversa.La notazione corretta è quella con il meno e non alla meno 1 poiché nel secondo caso staremmo indicando l'inversa di un gruppo moltiplicativo.Le restanti proprietà di associatività, identità e commutatività sono banalmente verificate effettuando le operazioni con punti generici. Proprietà importanti è che se P,Q;R si trovano sulla stessa linea, allora la loro somma è zero.
Per calcolare le coordinate del risultato si può procedere per via analitica nel seguente modo:

$$
\begin{aligned}
P &= (x_1, y_1) \\
Q &= (x_2, y_2) \\
R &= P + Q = (x_3, y_3) \\
x_3 &= \lambda^2 - x_1 - x_2 \\
y_3 &= \lambda(x_1 - x_3) - y_1 \\
\lambda &= \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} & P \neq Q \\ \frac{3x_1^2 + a}{2y_1} & P = Q \end{cases}
\end{aligned}
$$

0.20 Elliptic Curve Modular Integer

Se aggiungiamo l'operazione di modulo nelle curve ellittiche otteniamo oltre al gruppo Abeliano, caratteristiche delle curve ellittiche, anche la ciclicità di tale gruppo e questo campo è finito;quindi possiede un numero finito di elementi.Per definire una curva ellittica nei gruppo in modulo p si utilizza la notazione $E(x, y)$ con x,y interi e si deve soddisfare la condizione che $y^2 \mod p = x^3 + ax + b \mod p \quad 4a^3 + 27b^2 \neq 0$.
Per verificare la ciclicità si potrebbe sommare lo stesso punto della curva ellittica finché non si ritorna al punto iniziale o, tramite un algoritmo analogo allo square multiply con la differenza che si somma e si raddoppia.