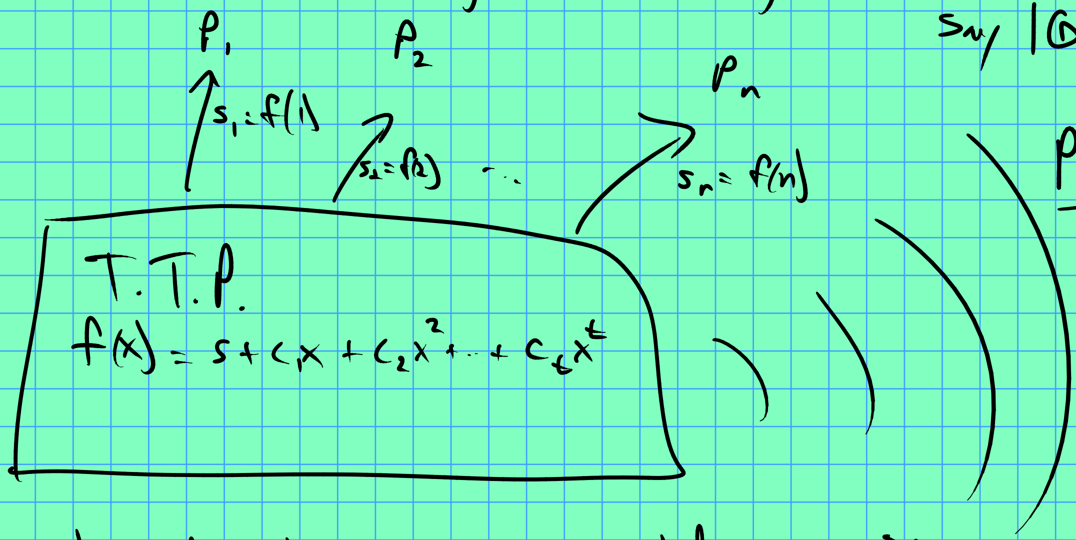
Last time: Threshold decryption for ElGamal.

Recall: $PK = g^s$, $SK = s$, $\langle g \rangle = G < \mathbb{Z}_p^\times$, say $|G| = q$.



$PK = g^s$

$PK_1 = g^{s_1}$

$\vdots$

$PK_n = g^{s_n}$

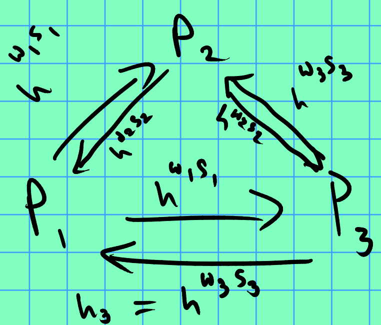
Goal: decrypt messages encrypted via PK, without ever reconstructing s.

$$s = \sum_{i=0}^{t} w_i s_{j_i}.$$

ciphertext: $(\underset{\overset{\shortparallel}{h}}{g^b}, \underset{\overset{\shortparallel}{k}}{PK^b \cdot m})$

want to compute mask $= \underset{\overset{\shortparallel}{g^{bs}}}{h^s} = PK^b = g^{sb}$

$$h^s = h^{\sum_{i=0}^{t} w_i s_{j_i}} = \underset{\overset{\shortparallel}{X}}{\prod_{i=0}^{t} h^{w_i s_{j_i}}}. \qquad m = k X^{-1}$$



What's left to be desired? What if a player violates the protocol? They might learn the decrypted message while convincing all other players the message is something else!

Want Proof that
$$h_3 = (h^{w_3})^{s_3}$$

We'll prove

$(*)$ $\log_{(h^{w_3})} h_3 = \log_g \underset{\overset{\shortparallel}{g^{s_3}}}{PK_3}$

Maybe we can use PK_i to convince other players that the values $h^{w_i s_i}$ are legitimate.

Goal: say $A = g^a$. Want to prove that $A_h = h^a$ for known values g, h.

(Known: $A = g^a$. Given h, A_h want to convince someone that $\underline{\log_g A = \log_h A_h}$.)

<u>Protocol</u> (Pedersen? Schnorr?)

$g, A = g^a, h, A_h$. Convince verifier that $A_h = h^a$, where $a = \log_g A$.

$a = \log_g A$

$b \in_R \mathbb{Z}_q$

$$\text{Prover} \xrightarrow{B = g^b,\ B_h = h^b} \text{Verifier} \qquad c \in_R \mathbb{Z}_q$$

$$\xleftarrow{\quad c \quad}$$

$$\xrightarrow[(\in \mathbb{Z}_q)]{k = ca + b}$$

Verifier accepts "proof" $\iff$

$$\left(g^k \stackrel{?}{=} \underbrace{A^c}_{= g^a} \underbrace{B}_{= g^b}\right) \wedge \left(h^k \stackrel{?}{=} \underbrace{A_h^c}_{= h^a} \underbrace{B_h}_{= h^b}\right)$$

Why is this convincing to Verifier? Suppose $A_h = h^{\tilde{a}}$ for $\tilde{a} \neq a$. And maybe $B_h = h^{\tilde{b}}$, $\tilde{b} \neq b$.

If proof is accepted by Verifier, then

$$ca + b = k = c\tilde{a} + \tilde{b}$$

But then we can solve for c:

$$(*) \qquad c = \frac{\tilde{b} - b}{a - \tilde{a}}$$

if $a \neq \tilde{a}$, there is only <u>one</u> choice of c that would trick the verifier!

So, if $\underset{\log_h A_h}{\overset{\shortparallel}{\tilde{a}}} \neq a$, $\Pr_{c \in \mathbb{Z}_q}[\text{Prover cheats Verifier}] = 1/q$

So if $q \approx 2^{256}$ ... wow!

Does the protocol "hide" a from verifier?

How to formalize? What does this even mean?

Want to show that Verifier "doesn't learn anything" from interacting w/ Prover.

Key idea: <u>simulation</u>.

If Verifier can "imagine" the whole conversation w/ Prover, then Verifier didn't learn anything by actually having the conversation!

Can we simulate the above conversation??

$$\text{Prover} \quad \xrightarrow{B = g^b,\ B_h = h^b} \quad \text{Verifier}$$

$$\xleftarrow{\quad c \quad}$$

$$\xrightarrow[(\in \mathbb{Z}_q)]{k = ca + b}$$

$b = k - ca$

"Normal" order: (focus on exponents) $(\underset{①}{b}, \underset{②}{c}, \underset{③}{k = ca + b})$

Order for simulation: $(\underset{③}{k - ca}, \underset{①}{c}, \underset{②}{k})$

Same prob. distribution on triples in $\mathbb{Z}_q$!

"Real" transcript: $(\underset{①}{(g^b, h^b)}, \underset{②}{c}, \underset{③}{k = ca + b})$

Simulated: $(\underset{③}{(A^{-c} g^k, A_h^{-c} h^k)}, \underset{①}{c}, \underset{②}{k})$

Yay!

So indeed, verifier only learns $\log_g A = \log_h A_h$.

Note: Verifier is convinced, but can't convince anyone else! (Proof is non-transferable)

Question: is interaction necessary?

Note: this is actually a "proof of knowledge". If given access to Prover (as a "rewindable" black box) one could extract the secret:

$$\xrightarrow{B = g^b,\ B_h = h^b}$$

$$\xleftarrow{\ c\ } \qquad\qquad \xleftarrow{\ c'\ }$$

$$\xrightarrow{k = ca+b} \qquad\qquad \xrightarrow{k' = c'a+b}$$

But then we could solve for a: $\frac{k-k'}{c-c'} = a$

Note: to get rid of the T.T.P. for key generation, think about the following easier task: how to produce new shares of an existing (and already shared) secret?

Say $s = f(0)$, $f(x) = s + \sum_{i=1}^{t} c_i x^i$.

To compute new shares of s, choose a new polynomial w/ no constant term: $f'(x) = \sum_{i=1}^{t} c_i' x^i$.

Then give out shares according to f': P_i gets $s_i' = f'(i)$, $i = 1, \dots, n$.

if original shares were $s_i = f(i)$, new shares $\sigma_i = s_i + s_i' = f(i) + f'(i) = (f+f')(i)$

For distributed key generation, read about Feldman's protocol if you are interested. Also check out "proactive security"

Zero-Knowledge Proofs

Definition (sketch):

Want the following properties:

- Completeness: Honest Prover can <u>always</u> convince verifier of true statements.
- Soundness: Cheating prover has negligible probability of convincing verifier of a false statement.
- Zero Knowledge: Protocol transcript can be <u>simulated</u> by verifier.

(simulation could be <u>identical</u>, statistically close, or even computationally indistinguishable)

Closer look at our example (proof that $\log_g A = \log_h A_h$):

Can only simulate if V is <u>honest</u>!

Possible fix (w/ the bonus of making the protocol non interactive):

⊗ Let a hash function replace V. (<u>Fiat-Shamir heuristic</u>)

$$P \quad \xrightarrow{\left(B = g^b,\ B_h = h^b,\ H(g \| h \| A \| A_h \| B \| B_h) = c,\ k = ca + b\right)} \quad V$$

(Secure in "Random Oracle" model.)

Could use this approach to make a signature scheme!

(Public verification key; secret signing key;
VK SK
given message m, $\sigma = \text{sign}(m, SK)$
is hard to predict / forge, and can
be verified using VK)

$\approx$ Schnorr signatures:

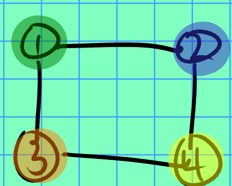
$$\left(B = g^b,\ B_h = h^b,\ H(g \| h \| A \| A_h \| B \| B_h \| M) = c,\ k = ca + b\right)$$

$(\approx \text{ElGamal} \approx \text{DSS/DSA})$

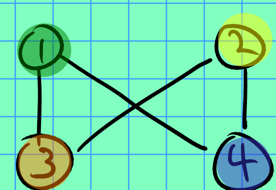
More ZK examples: Graph Isomorphism.

Setup: G_0, G_1, Prover knows $\underline{\varphi : G_0 \xrightarrow{\simeq} G_1}$

G_0:



G_1:



$$G_0 \quad \begin{array}{ccc} 1 & \overset{\varphi}{\longmapsto} & 1 \\ 2 & \longmapsto & 4 \\ 3 & \longmapsto & 3 \\ 4 & \longmapsto & 2 \end{array} \quad G_1$$

Then (i,j) is an edge of G_0
$\iff$ $(\varphi(i), \varphi(j))$ an edge of G_1

Edges of G_0: $(1,2)$, $(1,3)$, $(2,4)$, $(3,4)$

Edges of G_1: $(1,3)$, $(1,4)$, $(2,3)$, $(2,4)$

$(\varphi(1), \varphi(2)) = (1,4)$

"isomorphic"

How can prover convince (in ZK) verifier $G_0 \simeq G_1$?

Idea! Prover makes a random isomorphic copy of say G_1:

$\psi : G_1 \xrightarrow{\cong} H$. H is sent to Verifier.

V then sends P a bit $b \in \{0,1\}$ and

P shows an isomorphism of $G_b \xrightarrow{\cong} H$.

$$\begin{tikzcd} G_0 \arrow[rr, "\varphi"] \arrow[dr, "\psi \circ \varphi"'] & & G_1 \arrow[dl, "\psi"] \\ & H & \end{tikzcd}$$

$$\begin{array}{ccc} P & \xrightarrow{\quad H \quad} & V \qquad b \in_R \{0,1\} \\ & \xleftarrow{\quad b \quad} & \\ & \xrightarrow[\psi \text{ if } b=1]{\psi\circ\varphi \text{ if } b=0} & \end{array}$$

if $G_0 \not\cong G_1$ what is prob. that V is convinced?

Completeness? ✓ (exercise...)

Soundness? $\Pr[P \text{ cheats } V] = 1/2$

So... execute the above n times.

$\Rightarrow \Pr[P \text{ cheats } V] = 2^{-n}$

Zero Knowledge?? Can we simulate transcripts?

Sure! Choose $b \in_R \{0,1\}$ first, and then select random $G_b \xrightarrow{\psi} H$.

(for this, pick a random permutation ψ of vertices of G_b. Then, $\forall$ edges (i,j) in G_b, add $(\psi(i), \psi(j))$ to H...)

For next time: Read about ZK for any NP language in Abhi & Rafael's book. (Section 4.7)