Threshold Signatures

Part 1: Discrete Log based schemes

Bar-Ilan University Winter School on Cryptography

Rosario Gennaro

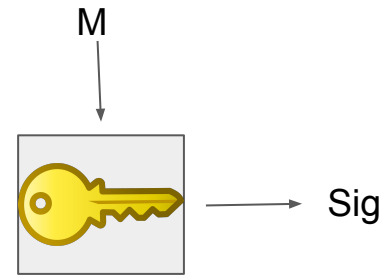
Protocol Labs
Research

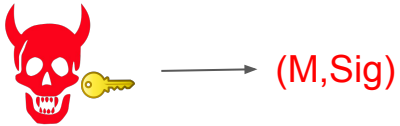
Yvo Desmedt:Threshold cryptography. Eur. Trans. Telecommun. 5(4): 449-458 (1994)

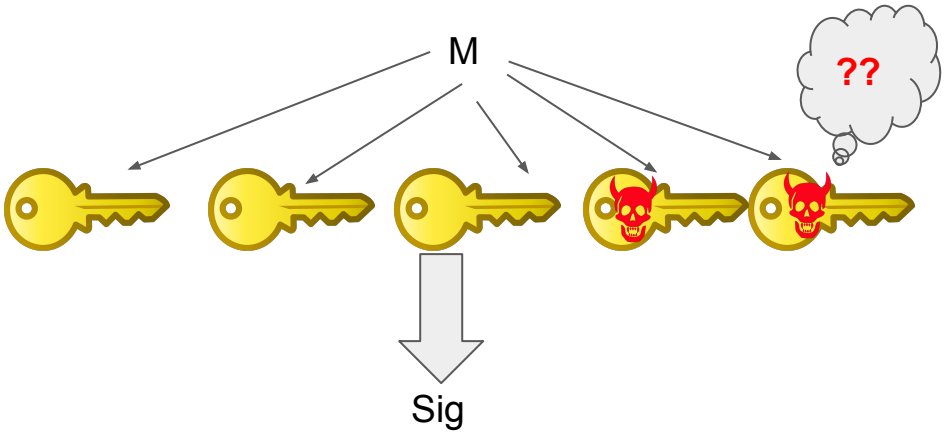
Threshold Cryptography

Cryptographic computations over shared keys

- Cryptographic computation:
  - Decryption
  - ***Signatures***
- Knowledge of the key is the security enabler
  - The key is a single point of failure
- Distribute the key across many devices
  - Assume only a fraction can be compromised
- Introduced by Yvo Desmedt in the early 90s

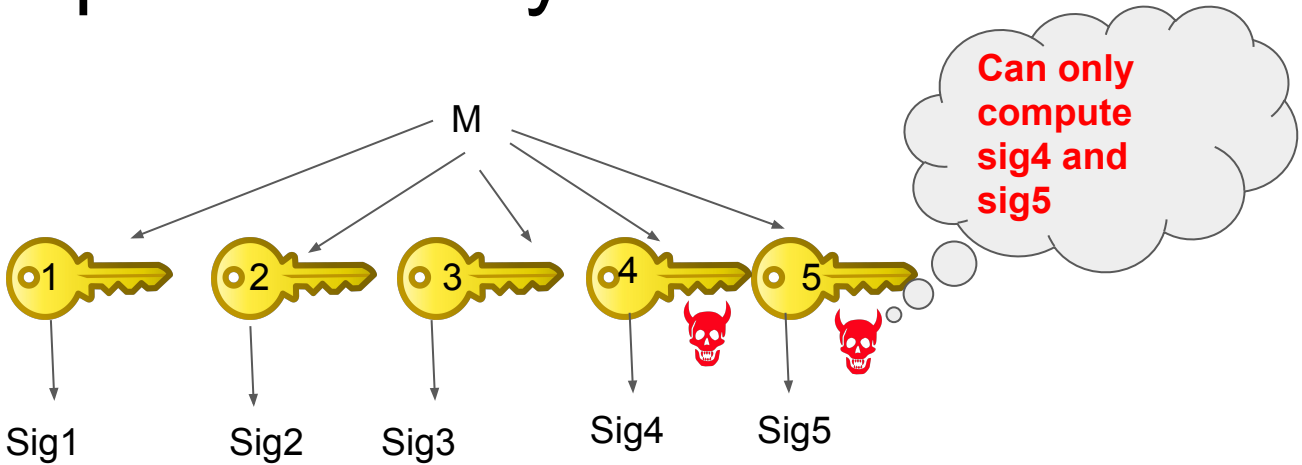






Replication

Many Independent Keys



Issues:

- Inefficient (replication)
- Non-transparent security policy

Multisignatures address some of these issues

What we want:
a signature that looks like it was produced by a single party, yet the key is stored in shared form across many devices

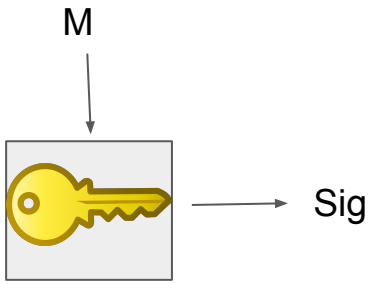
Secret Sharing 1

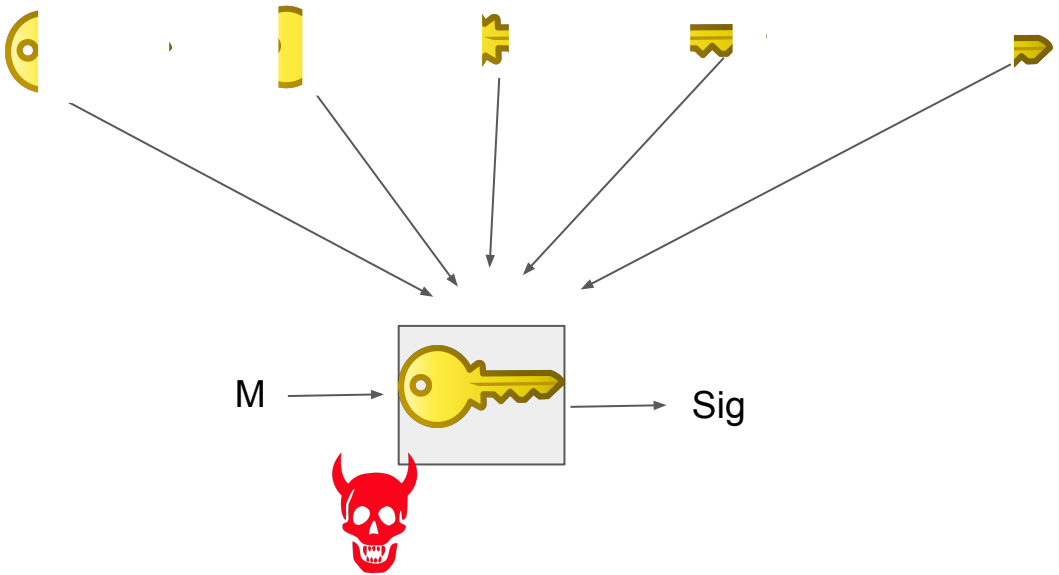
Shamir's classic scheme

- A ***dealer*** holds a secret ***s*** and they want to share it among ***n*** players in a such a way that
  - ***t*** players have no information ***s***
  - ***t+1*** players can recover ***s***
- Let ***q*** be a prime and assume $s \in Z_q$
- Choose a random polynomial $F(x) \in Z_q[X]$ of degree ***t*** such that:
  - $F(0)=s$
- Send to player P_i the share $s_i=F(i) \bmod q$
- ***t+1*** players can recover the secret by polynomial interpolation
- ***t*** players have no information about the secret in a strong information-theoretic sense
  - For any possible secret ***s'*** there is a polynomial ***F'*** which agrees with the secret and the shares held by the adversary
  - Interpolate ***F'*** with $F'(0)=s'$ and $F'(i)=s_i$ for the ***t*** indices ***i*** corresponding to the adversary's shares

Sharing the key

We want the key to never be in one place





Interpolation is a linear function

- Given a set S of $t+1$ values s_i for $i \in S$ we want to find the polynomial $F[X]$ of degree t such that
  - $F(i) = s_i$ for $i \in S$

- Let $\Lambda_{i,S}[X]$ be the ***Lagrangian*** polynomial of degree t defined by
  - $\Lambda_{i,S}[i]=1$ and $\Lambda_{i,S}[j]=0$ for $j \in S, j \neq i$
  - $\Lambda_{i,S}[X] = [\prod_{j \in S, j \neq i} (X-j)] / [\prod_{j \in S, j \neq i} (i-j)]$
  - Then it must be that
    - $F[X] = \sum_{i \in S} \Lambda_{i,S}[X] s_i$
    - Since both sides of the equation are polynomials of degree t agreeing on $t+1$ points

Requires a field

- Remember that in our case we want to find $s=F(0)$ then
  - $s = \sum_{i \in S} \lambda_{i,S} s_i$
  - where $\lambda_{i,S} = \Lambda_{i,S}[0]$ the ***0-Lagrangian*** coefficients associated with S
  - $\lambda_{i,S} = [\prod_{j \in S, j \neq i} j] / [\prod_{j \in S, j \neq i} (j-i)]$

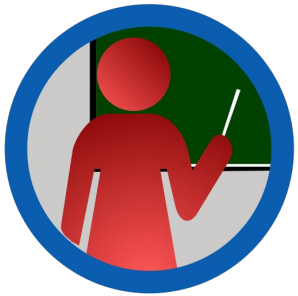
Requires a field

- Actually true for any $s_j = F(j)$
  - $s_j = \sum_{i \in S} \lambda_{j,i,S} s_i$ where $\lambda_{j,i,S} = \Lambda_{i,S}[j]$ the ***j-Lagrangian*** coefficients associated with S

Our first example

BLS signatures

We have a cyclic group G of prime order q
Efficient test T to check if given $y,g,s,m \in G$
there exists $x \in Z_q$ such that $y=g^x$ and $s=m^x$



$PK=y=g^x$

$SK=x$

On input a message M,
we hash it to obtain $m \in G$
and compute the signature
$s=m^x$

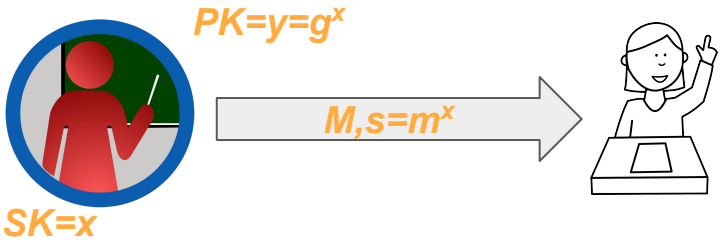
M,s

Computes $m=H(M)$ and uses
test T to check if there exists
$x \in Z_q$ such that $y=g^x$ and $s=m^x$

D.Boneh, B.Lynn, H.Shacham: Short Signatures from the Weil Pairing. J. Cryptol. 17(4): 297-319 (2004)

Our first example

Threshold BLS signatures



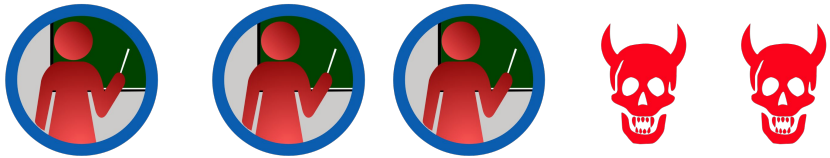
- A ***dealer*** shares the secret key ***x*** among ***n*** parties using Shamir
  - Let $[x_1 \dots x_n]$ be the shares
  - Remember there is a polynomial ***F[X]*** of degree ***t*** such that $F(0)=x$ and $F(i)=x_i$
  - Everything ***mod q***, the order of the group ***G***

- On input ***M*** every player outputs $s_i = m^{x_i}$
  - Given a set ***S*** of ***t+1 partial signatures***
  - Since $x= \sum_{i \in S} \lambda_{i,S} x_i$ and $s=m^x$
  - Then $s= \prod_{i \in S,} s_i^{\lambda i,S}$
  - ***Interpolation in the exponent***

A. Boldyreva. Efficient Threshold Signatures, Multisignatures and Blind Signatures based on the Gap-Diffie-Hellman-Group Signature Scheme.PKC 2003 LNCS 2567, pp. 31-46,

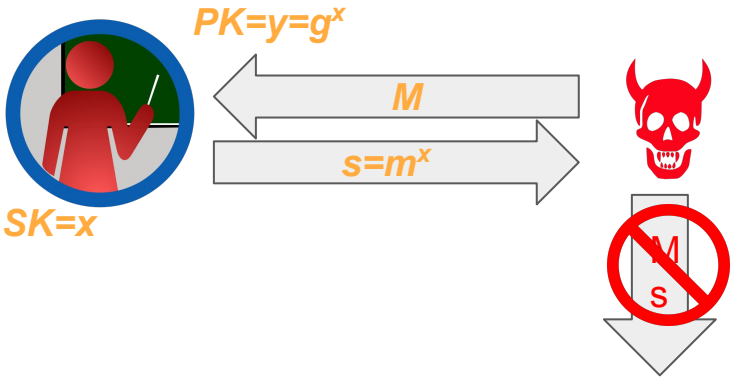
Threshold BLS Signatures

Why is this secure?



The adversary learns nothing more than $s = m^x$

- Given his own ***t partial signatures*** $s_i = m^{xi}$ and $s = m^x$
- They have a set ***S*** of ***t+1*** points and can interpolate in the exponent the other partial signatures
- Since $x_j = \sum_{i \in S} \lambda_{j,i,S} x_i$ then $s_j = \prod_{i \in S,} s_i^{\lambda j,i,S}$



Simulator:
Given a signature it can simulate the entire view of the adversary

This implies that the adversary cannot forge messages in the distributed scheme unless they can forge them in the centralized one.

Threshold BLS Signatures

Unforgeability by Simulation

Simulator gives random x_i to the adversary and plays the role of the honest players

s_1 s_2 s_3

x_4 x_5

M

s

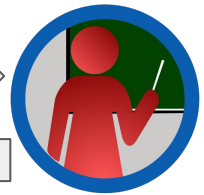
Assume the adversary can forge controlling only t players

M

$PK=y=g^x$

M

$s=m^x$



SIMULATOR
(forging centralized scheme)

M

s

Simulator computes the adversary t ***partial signatures*** $s_i = m^{x_i}$ and knows $s = m^x$

- They can interpolate in the exponent the partial signatures of the honest players
- Since $x_j = \sum_{i \in S} \lambda_{j,i,S} x_i$ then $s_j = \prod_{i \in S,} s_i^{\lambda_{j,i,S}}$

Unforgeability vs Robustness

- $PK=y=g^x$ $SK=x$
  - $[x_1 \dots x_n]$ shares of x (a polynomial $F[X]$ of degree t such that $F(0)=x$ and $F(i)=x_i$)
  - On input M every player outputs the **partial signature** $s_i = m^{xi}$
  - Given a set S of $t+1$ **correct** **partial signatures** s_i then $s= \prod_{i \in S,} s_i^{\lambda i,S}$

- Unforgeability proof holds for any $t<n$
  - Assumes **semi-honest** adversary (gathers information but follows protocol instructions)

- What about a **malicious** adversary (deviates arbitrarily from the protocol)
- Can we guarantee **robustness**
  - The protocol always completes successfully with a valid signature (no **denial of service**)
- First of all we need $n>2t$
  - That's because t corrupted players can always refuse their partial signature
  - But what about corrupted players giving **incorrect** partial signatures?

Small Detour

Error correction?

If $[x_1 \dots x_n]$ are n points on a polynomial $F[X]$ of degree t (e.g. $F(i)=x_i$)

- We know that if $n>3t$ then we can interpolate F[X] even if given the vector $[y_1 \dots y_n]$
  - $y_i=x_i$ for at least $n-t$ indices
  - Reed-Solomon codes

- But we are interpolating in the exponent
  - Given $n>3t$ ***partial signatures*** s_i
  - $n-t$ of the form $m^{F(i)}$ and t arbitrary
    - Can we find $s= m^{F(0)}$

- [Peikert05] shows that this is a problem as hard as CDH :(
  - So how can we deal with ***incorrect*** partial signatures?
  - Try all possible subsets of $t+1$ partial signatures and only accept the one that yields a valid signature s
    - $O(n^t)$ solution so OK only for small n,t

Check partial signatures

- When the ***dealer*** shares the secret key ***x*** among ***n*** parties using Shamir
  - Let $[x_1 \ldots x_n]$ be the shares ($F[X]$ of degree t such that $F(0)=x$ and $F(i)=x_i$)
  - Also publishes $PK_i=y_i=g^{xi}$

- When a player outputs s_i (which should be m^{xi})
  - For BLS signature use the efficient test T to check $DLog_m\ s_i = DLog_g\ y_i$
  - For groups without such a test there are efficient ZK proofs for the statement $DLog_m\ s_i = DLog_g\ y_i$

Wait a minute

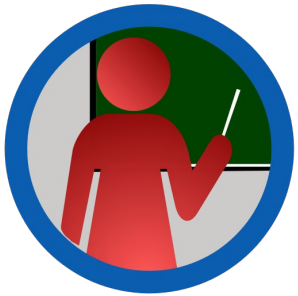
DEALER?

- We have assumed a ***dealer*** who shares the secret key ***x***
- Isn't this a single point of failure?
  - ***YES***
- I thought we didn't want single points of failure?
  - This is already an improvement
  - Sharing is a one-time event, the dealer can destroy all information about ***x*** once the sharing is done
- Can we do without a dealer
  - ***YES***
  - But you have to wait :)
  - ***Distributed Key Generation*** coming up later in the course.

Our second example

Schnorr's signatures

We have a cyclic group G of prime order q



$PK=y=g^x$

$SK=x$

On input a message M

- Choose $k \in Z_q$ at random and compute $R=g^k$
- Compute $m=H(M,y,R) \in Z_q$
- Set $s=k+mx \bmod q$
- Output (R,s)

M,R,s

Computes $m=H(M,y,a)$ and checks

$Ry^m ?= g^s$

Our second example

Threshold Schnorr signatures

- $ k∈Z_q : R=g^k$
- $s=k+mx \bmod q$

- A ***dealer*** shares the secret key x among n parties using Shamir
  - Let $[x_1 \dots x_n]$ be the shares (polynomial $F[X]$ of degree t such that $F(0)=x$ and $F(i)=x_i$)
- A ***dealer*** shares the secret nonce k among n parties using Shamir
  - Let $[k_1 \dots k_n]$ be the shares (polynomial $K[X]$ of degree t such that $K(0)=k$ and $K(i)=k_i$)
- On input M every player outputs $R_i = g^{ki}$
  - Given a set S of ***t+1 partial nonces*** R_i we have that $R= \prod_{i∈S,} R_i^{\lambda i,S}$
  - The players can now compute m and set $s_i = k_i + m\, x_i \bmod q$
  - $s= \sum_{i∈S} \lambda_{i,S}\, s_i$
- Again this only works for semi-honest adversaries

Our second example

Robust Threshold Schnorr signatures

- $ k∈Z_q : R=g^k$
- $s=k+mx \bmod q$

- A ***dealer*** shares the secret key x among n parties using Shamir
  - Let $[x_1 \dots x_n]$ be the shares (polynomial $F[X]$ of degree t such that $F(0)=x$ and $F(i)=x_i$)
  - The dealer also publishes $PK_i=y_i=g^{xi}$

- A ***dealer*** shares the secret nonce k among n parties using Shamir
  - Let $[k_1 \dots k_n]$ be the shares (polynomial $K[X]$ of degree t such that $K(0)=k$ and $K(i)=k_i$)
  - The dealer also publishes $R=g^k$ and $R_i = g^{ki}$

- On input M every player outputs $s_i = k_i + m\, x_i \bmod q$
  - A partial signature is correct if $R_i\, y_i^{\,m} = g^{si}$
  - Given a set S of ***t+1 correct partial signatures***
  - $s= \sum_{i \in S} \lambda_{i,S}\, s_i$

Wait a minute

DEALER AGAIN?

- A ***dealer*** who shares the secret key ***x*** is a single point of failure limited in time
  - Sharing is a one-time event, the dealer can destroy all information about ***x*** once the sharing is done

- A ***dealer*** who shares the secret nonce ***k*** for each signature is a single point of failure ***all the time***
  - Knowledge of the secret nonce ***k*** is equivalent to knowledge of ***x*** once a signature is issued

- Can we do without a dealer
  - ***YES***
  - ***Distributed Key Generation*** can be used to generate the nonce as well.

Threshold Schnorr Signatures

Unforgeability by Simulation

Simulator with y:

- gives random x_i to the adversary and
- Interpolates in the exponents the $y_i = g^{xi}$ of the honest players

s_1 s_2 s_3

Assume the adversary can forge controlling only t players

x_4 x_5

M
R
s

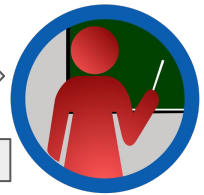
r_4 r_5

M

$PK=y=g^x$

M

R,s



SIMULATOR
(forging centralized scheme)

M
R
s

Simulator with R:

- gives random k_i to the adversary and
- Interpolates in the exponents the $R_i = g^{ki}$ of the honest players
- Now SIM knows s_i of the corrupted players
- With s they can interpolate the s_i of the honest players

What properties do we need

- The n players should jointly generate a sharing of secret key x
  - Let $[x_1 \dots x_n]$ be the private shares
  - The public key $PK=y=g^x$
  - The partial public keys $PK_i=y_i=g^{xi}$
- This protocol is repeated for each signature to generate the nonce k
  - Let $[k_1 \dots k_n]$ be the private shares, the public nonce $R=g^k$ and the partial public keys $R_i=g^{ki}$
- We should have a simulator that on input y
  - Produces an indistinguishable view for the adversary ***on an execution that outputs*** y

Verifiable Secret Sharing (VSS)

Feldman's VSS

- In ***VSS*** the players have a guarantee that there is a unique secret shared and that their shares interpolate to the correct secret

- The dealer on input the secret x
  - Chooses a polynomial $F[X]$ of degree t such that $F(0)=x$
    - Let $[f_o \dots f_t]$ be the coefficients of $F[X]$ ($f_o=x$)
  - Broadcasts $F_j = g^{fj}$
  - Sends to player i the share $x_i = F(i)$

- Player i checks that their share x_i lies on the polynomial defined by $[F_o \dots F_t]$
  - ***Evaluation in the exponent***
  - If it does not they lodge a complaint

- If more than t complaints the dealer is bad and is disqualified
  - Otherwise complaints are resolved by broadcasting the correct share

Pedersen's DKG

- Player i perform a Feldman's VSS of z_i
  - The value $Z_i = g^{z_i}$ is public from the Feldman VSS
  - Each player j receives share z_{ij} from player i
  - The value $Z_{ij} = g^{z_{ij}}$ is also public from the Feldman VSS

- Let Q be the set of players who are not disqualified
  - The key x is defined as $x = \sum_{i \in Q} z_i$
  - $y = g^x = \prod_{j \in Q} Z_j$
  - Player i share is defined as $x_i = \sum_{j \in Q} z_{ji}$
  - $y_i = g^{x_i} = \prod_{j \in Q} Z_{ji}$

T.P. Pedersen: A Threshold Cryptosystem without a Trusted Party EUROCRYPT 1991: 522-526

There's an issue …

G, S.Jarecki, H.Krawczyk, T. Rabin: Secure Distributed Key Generation for Discrete-Log Based Cryptosystems. J. Cryptol. 20(1)

(Non)-Simulation of Pedersen's DKG





$Z_1=g^{z1}$ $Z_2=g^{z2}$ $Z_3=g^{z3}$?? ??

The adversary controls only *t* players

$PK=y=g^x$

SIMULATOR

Simulator with *y*:

- Performs Feldman's VSS for good players ***without knowing the contribution of the adversary***
- There is no way SIM can hit the right distribution (the target value *y*)
- SIM needs to see the contribution of the adversary before committing to the contribution of the honest players

G, S.Goldfeder:Fast Multiparty Threshold ECDSA with Fast Trustless Setup. CCS 2018: 1179-1194

Committed Pedersen's DKG

- Player i commits to $Z_i = g^{zi}$ with a ***non-malleable trapdoor commitment***
- Player i perform a Feldman's VSS of z_i
  - The value $Z_i = g^{zi}$ is public from the Feldman VSS
    - And is checked against the commitment
  - Each player j receives share z_{ij} from player i
  - The value $Z_{ij} = g^{zij}$ is also public from the Feldman VSS

- Let Q be the set of players who are not disqualified
  - The key x is defined as $x = \sum_{i \in Q} z_i$
  - $y = g^x = \prod_{j \in Q} Z_j$
  - Player i share is defined as $x_i = \sum_{j \in Q} z_{ji}$
  - $y_i = g^{xi} = \prod_{j \in Q} Z_{ji}$

Fixing the issue

Simulating Committed Pedersen's DKG





C_1 C_2 C_3 C_4 C_5

Z_1 Z_2 Z_3 Z_4 Z_5

$PK=y=g^x$

SIMULATOR

Rewind

Z_1 Z_2 Z_3 Z_4 Z_5

Such that

$Y = Z_1Z_2Z_3Z_4Z_5$

Can't be changed due to non-malleability

Adversary can always abort

- In the Committed Pedersen's DKG the adversary can always refuse to decommit
  - Simulation gets stuck again
- The guarantee is that ***conditioned to the protocol successfully completing*** we can hit the right distribution of public keys
  - So the adversary can create a denial of service attack
  - But cannot forge
    - Since if the protocol completes we can turn a forgery in the distributed system into one in the centralized one

Prevent the adversary from aborting

- We need a “recoverable commitment”
  - If the adversary refuses to open the honest parties can recover it
- That’s exactly what VSS is!
  - But remember that we need a “non-malleability” condition
  - Preventing the adversary from committing to something related to the honest players
- We are going to use an information–theoretically private VSS to commit
  - The adversary has no information at all about the good players secrets
- Then we use Feldman’s VSS to compute the public key
  - Enforcing that Feldman’s VSS is consistent with the information-theoretic VSS used to commit

Information-Theoretically Private Verifiable Secret Sharing

Pedersen's VSS

The dealer on input the secret x

- Chooses a random polynomial $F[X]$ of degree t such that $F(0)=x$
  - Let $[f_o \ldots f_t]$ be the coefficients of $F[X]$ ($f_o=x$)
- Chooses another random polynomial $R[X]$ of degree t
  - Let $[r_o \ldots r_t]$ be the coefficients of $R[X]$
- Broadcasts $F_j = g^{fj} h^{tj}$
- Sends to player i the share $x_i = F(i),\ y_i = R(i)$

- Player i checks that their shares $x_i\ y_i$ lies on the polynomial defined by $[F_o \ldots F_t]$
  - ***Evaluation in the exponent***
  - If it does not they lodge a complaint

- If more than t complaints the dealer is bad and is disqualified
  - Otherwise complaints are resolved by broadcasting the correct share

Joint-Pedersen's DKG

- Player ***i*** perform a ***Pedersen's*** VSS of z_i
- Player ***i*** perform a Feldman's VSS of z_i
  - Only the public commitment part
  - Uses the same polynomial ***F*** used to share z_i
    - Players already have the shares
- As before if ***Q*** is the set of players who are not disqualified
  - The key ***x*** is defined as $x = \sum_{i \in Q} z_i$ and $y = g^x = \prod_{j \in Q} Z_j$
  - Player ***i*** share is defined as $x_i = \sum_{j \in Q} z_{ji}$ and $y_i = g^{xi} = \prod_{j \in Q} Z_{ji}$

G, S.Jarecki, H.Krawczyk, T. Rabin: Secure Distributed Key Generation for Discrete-Log Based Cryptosystems. J. Cryptol. 20(1)

Solution with Robustness

Simulating Joint Pedersen's DKG

Ped-VSS *Ped-VSS*

$PK=y=g^x$

SIMULATOR

Assuming honest majority SIM knows the values of the adversary
By interpolating the shares

Z_1 Z_2 Z_3 Z_4 Z_5

Such that

$Y = Z_1Z_2Z_3Z_4Z_5$

If the adversary does not reveal them, the honest parties can recover them via the Ped-VSS

Let’s stop for a second

Summary slide so far

- With a simulatable DKG we can construct Threshold Signatures for discrete-log based schemes such as ***BLS*** and ***Schnorr***
  - Honest Majority with robustness
    - ***Joint-Pedersen DKG***
  - Dishonest Majority with abort
    - ***Committed Pedersen DKG***

- Proof follows a simulation argument
  - If you can forge in the threshold setting you can forge in the centralized setting

DSA: The Digital Signature Standard

We have a cyclic group G of prime order q

$PK=y=g^x$

$SK=x$

On input a message M

- Choose $k \in Z_q$ at random and compute $R=g^{inv(k)}$
- Set $s=k(m+xr) \bmod q$
  - $r=H(R)$, $m=H(M) \in Z_q$
- Output (R,s)

M,R,s

Computes $r=H(R)$, $m=H(M) \in Z_q$ and checks

$g^m y^r ?= R^s$

What about DSA

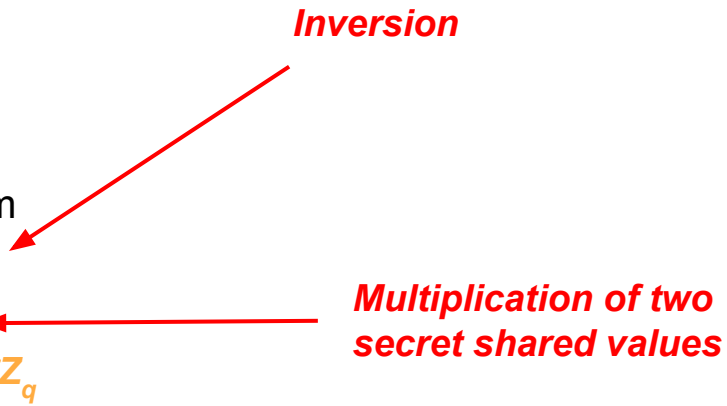
DSA vs Schnorr

On input a message M

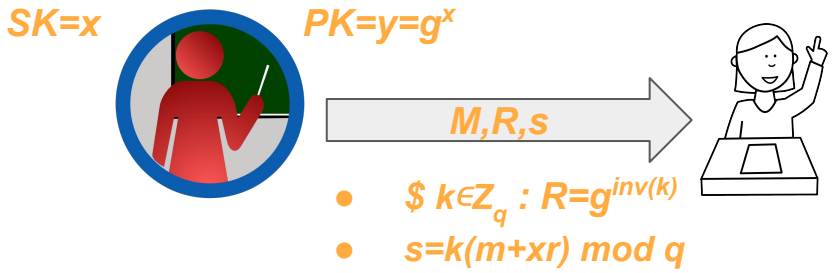
- Choose $k \in Z_q$ at random and compute $R=g^k$
- Compute $m=H(M,y,R) \in Z_q$
- Set $s=k+mx \bmod q$
- Output (R,s)

On input a message M

- Choose $k \in Z_q$ at random and compute $R=g^{inv(k)}$
- Set $s=k(m+xr) \bmod q$
  - $r=H(R)$, $m=H(M) \in Z_q$
- Output (R,s)



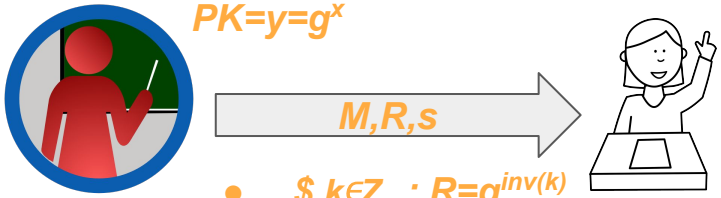
Threshold DSA DKG



- Joint-Pedersen DKG

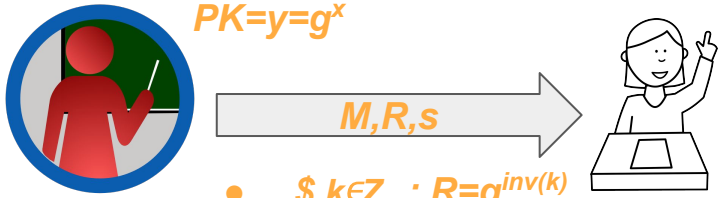
G, S.Jarecki, H.Krawczyk, T. Rabin: Robust Threshold DSS Signatures. Inf.Comput. 164(1)

Threshold DSA nonce



- Players perform ***two*** Joint-Pedersen DKG
  - Let ***k,a*** be the random values generated
  - Only for ***a*** the Feldman phase is performed, so the value $A = g^a$ is public
- Players reconstruct the value ***b=ka***
  - By broadcasting the product shares
    - Requires randomization with a ***0***-polynomial of degree ***2t***
- The players ***c=inv(b) mod q*** and compute $R = g^{inv(k)} = A^c$
  - The players already have shares of ***k***
  - Bar-Ilan & Beaver'91

Threshold DSA *s*-value



- Players have shares of ***k*** and ***x***
  - Each party broadcasts $s_i = mk_i + rk_ix_i$
  - Which interpolates to ***s***
    - Requires randomization with a ***0***-polynomial of degree ***2t***

- In both reconstructions how to weed out bad shares?
  - With error correction codes (requires ***n>4t+1***)
  - Or with ZK-proofs of correctness with respect to the public values generated by the VSSs (requires ***n>3t+1***)

A little prehistoric detour

Multiplication of secrets shared additively

- Assume n players have additive shares of secrets a,b
  - $a=a_1 + \ldots + a_n$ and $b=b_1 + \ldots + b_n$
  - Player i holds a_i and b_i

- The parties want to compute an additive sharing of $c=ab$
  - Note that c= $\sum_{i,j} a_i b_j$
  - If Parties i and j could turn $a_i b_j$ into two values d_{ij} and e_{ij} such that
    - $d_{ij} + e_{ij} = a_i b_j$
  - Then Player i could set c_i to
    - $a_i b_i + \sum_i d_{ij} + \sum_i e_{ji}$

  - $c=c_1 + \ldots + c_n$

O.Goldreich, S.Micali, A.Wigderson: How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority. STOC 1987: 218-229

N.Gilboa:Two Party RSA Key Generation. CRYPTO 1999: 116-129

Multiplicative to additive shares

- An ***MtA*** protocol allow two players Alice and Bob
  - Who hold secrets $a,b \in Z_q$ respectively
  - To turn them into secret $d,e \in Z_q$ respectively such that
  - $d+e=ab \bmod q$

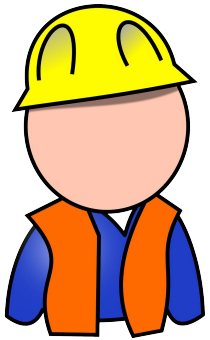
- Let ***E*** be an additively homomorphic encryption scheme
  - With message space and homomorphism over Z_q

d=D[B]

Knows D

A=E[a]

B=E[ab-e]



$ e

Uses homomorphism to compute B

P.MacKenzie, M.K.Reiter: Two-party generation of DSA signatures. Int. J. Inf. Sec. 2(3-4): 218-239 (2004)

Can we use Paillier?

In our case ***q*** will be determined by the ***DSA*** parameters

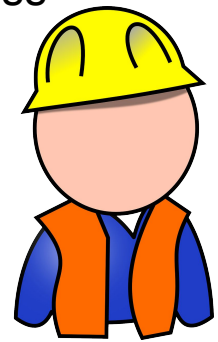
- ***E*** with message space and homomorphism over Z_q exists under assumption over ***class groups***
- What about Paillier?
  - Homomorphism is over Z_N where ***N*** is an RSA modulus.
  - Parties need to add a range ZK-proof that their values are “small”
    - Prevent reduction ***mod N***
    - Important for both privacy and correctness

d=D[B]

Knows D

A=E[a] +ZKP[a is “small”

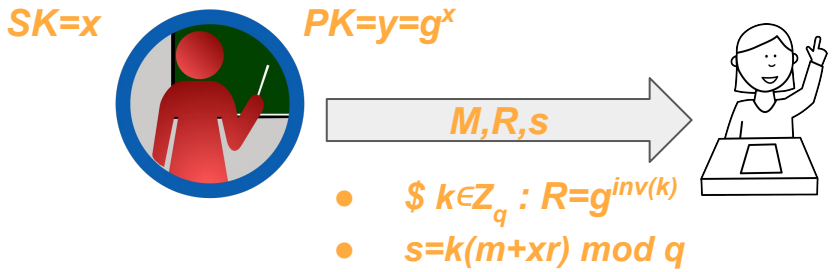
B=E[ab-e]+ZKP[b,e are small]



$ e

Uses homomorphism to compute B

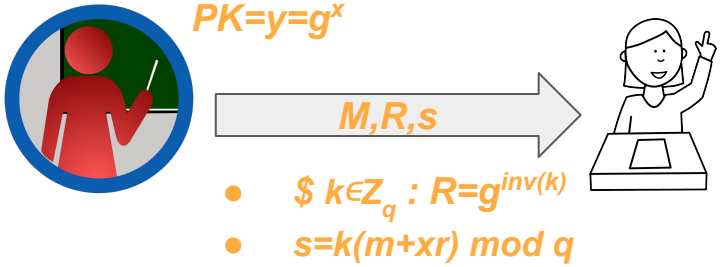
Threshold DSA DKG



- Committed Pedersen DKG
  - Each player has a share of ***x*** in a ***(t,n)*** Shamir scheme
- When ***t+1*** players want to sign we think of their shares as additive shares of ***x***
  - Scaling them with the appropriate Lagrangian coefficient

R.Canetti, G, S.Goldfeder, N.Makriyannis, U.Peled: UC Non-Interactive, Proactive, Threshold ECDSA with Identifiable Aborts. CCS 2020: 1769-1787

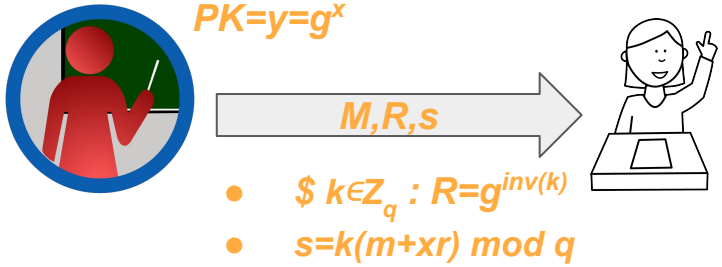
Threshold DSA



Simplified Version

- ***t+1*** players perform ***two*** additive sharings of random values ***k,a***
  - $a=a_1 + \dots + a_n$ and $k=k_1 + \dots + k_n$
  - The values $A_i = g^{ai}$ are committed with a non-malleable commitment
- Players performs ***2 MtA*** protocols to get additive shares of ***b=ka*** and ***z=kx***
  - Each player decommits A_i and players compute $A=g^a=\prod_i A_i$
  - Players reconstruct ***b***, compute ***c=inv(b) mod q*** and $R = g^{inv(k)} = A^c$
- Players broadcast $s_i=k_i m+rz_i$ to interpolate ***s***
  - Protocol aborts if the signature is not correct

Aborting should not reveal info



We need to make sure that if the protocol aborts no information about the secrets of the honest parties is revealed

- Reductions ***mod N*** during the ***MtA*** protocols
  - Avoided by enforcing range proofs
- Adversary using inconsistent values between the ***MtA*** protocols and the reconstructions of ***R,s***
  - Before outputting ***s*** the players check that $R^k=g$ and $R^z=y$
    - Via interpolation in the exponent
    - And use ZK proof to enforce that those values are the same as the ones used in the ***MtA*** protocols and in the reconstruction of ***s***

Additional References

More references on Threshold DSA with abort

Dishonest Majority:

- Similar techniques to the ones described above
  - Y.Lindell, A.Nof: Fast Secure Multiparty ECDSA. CCS 2018
- Using an oblivious-transfer based ***MtA*** protocol (avoids introducing additional assumptions)
  - J.Doerner, Y.Kondi, E.Lee, a.shelat: Secure Two-party Threshold ECDSA from ECDSA Assumptions. IEEE Symposium on Security and Privacy 2018
- Using class groups in the ***MtA*** protocol (no range proofs)
  - G.Castagnos, D.Catalano, F.Laguillaumie, F.Savasta, I.Tucker:Bandwidth-efficient threshold EC-DSA revisited: Online/Offline Extensions, Identifiable Aborts, Proactivity and Adaptive Security. IACR ePrint 2021/291
- Two-party case
  - P.MacKenzie, M.K.Reiter: Two-party generation of DSA signatures. Int. J. Inf. Sec. 2(3-4): 218-239 (2004)
  - Y.Lindell: Fast Secure Two-Party ECDSA Signing. CRYPTO (2) 2017: 613-644
- Using MPC techniques
  - The protocols we discussed are just traditional MPC protocols tailored to the computation of the DSA function: here are other ways using e.g. precomputation of Beaver's triplets
    - D.Abram, A.Nof, C.Orlandi, P.Scholl, O.Shlomovits: Low-Bandwidth Threshold ECDSA via Pseudorandom Correlation Generators. IACR ePrint 2021/1587
    - A.P. K.Dalskov, C.Orlandi, M.Keller, K.Shrishak, H. Shulman: Securing DNSSEC Keys via Threshold ECDSA from Generic MPC. ESORICS (2) 2020: 654-673

Additional References

More references on Threshold DSA with abort

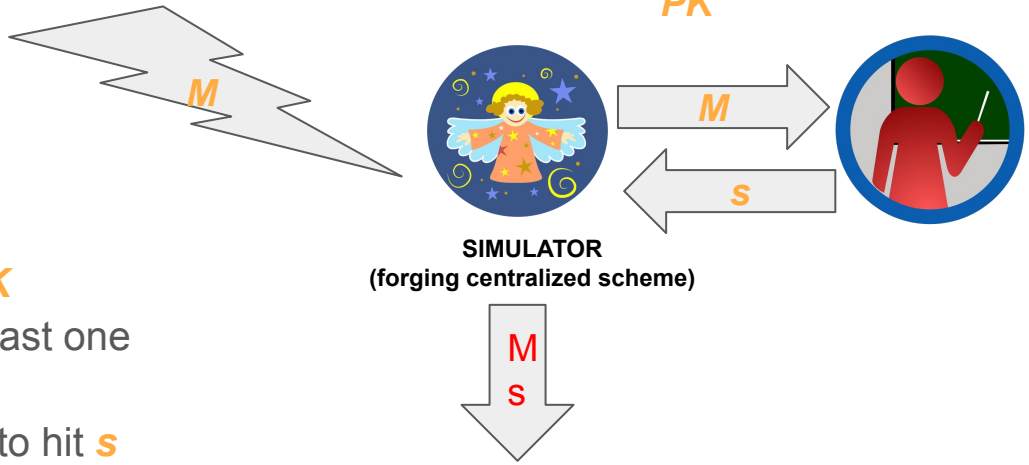
Honest Majority:

- Assumes n>2t+1 but also aborts
- Trade-off is better efficiency and round complexity
- Also no need for a reliable broadcast channel
  - Required by a robust and fair protocol
    - I.Damgård, T.P.Jakobsen, J.B.Nielsen, J.I.Pagter, M.B.Østergård: Fast Threshold ECDSA with Honest Majority. SCN 2020: 382-400

Let’s revisit simulation

Unforgeability by Simulation

Assume the adversary can forge controlling only *t* players

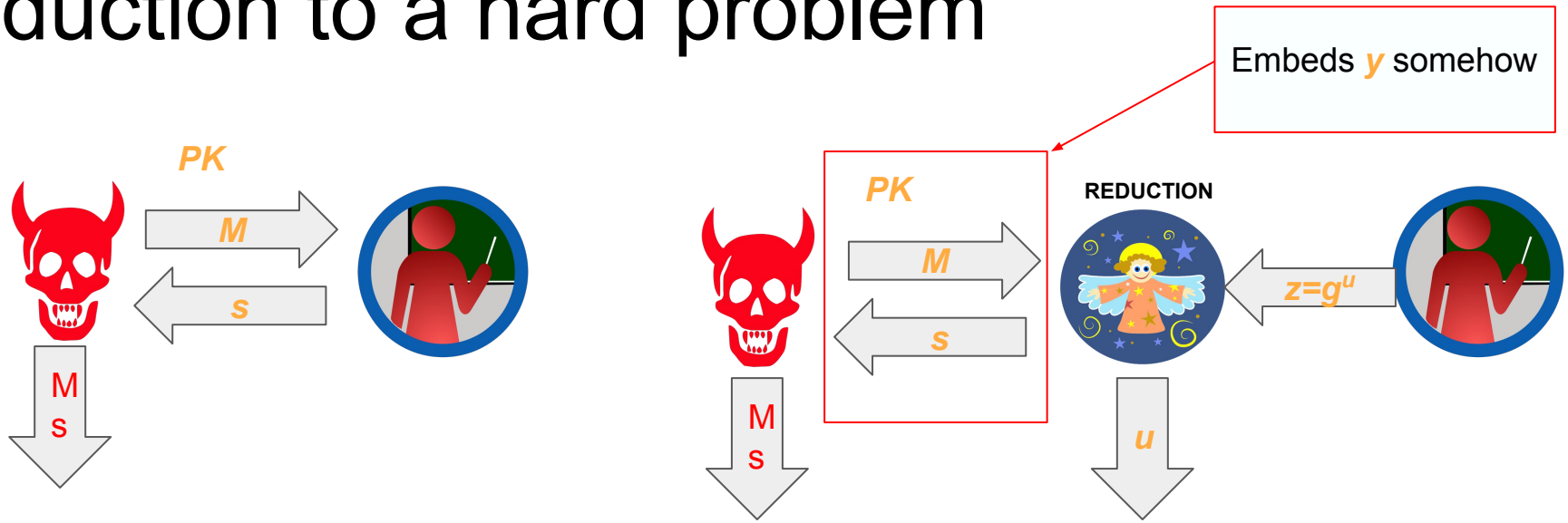


Simulator:

- On input ***PK*** simulates the ***DKG*** to hit ***PK***
  - Does not know secret keys for at least one honest party
- On input ***M*** simulates signature protocol to hit ***s***
- Threshold scheme as secure as centralized one

But there is another strategy

Reduction to a hard problem



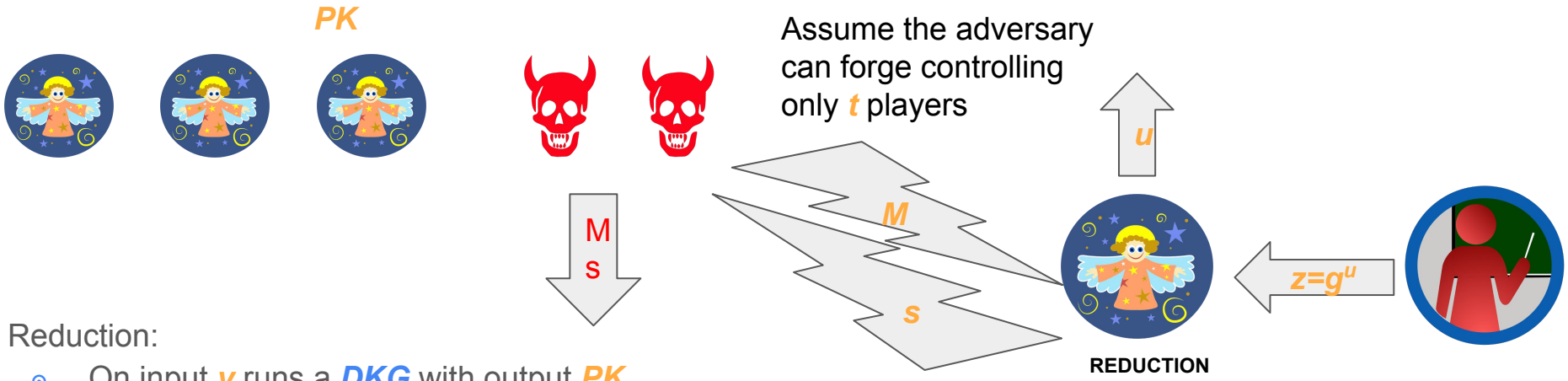
Consider Schnorr's signature scheme:

- We know that forgery can be reduced to the computation of discrete logs over the group *G*
  - D.Pointcheval, J.Stern:Security Arguments for Digital Signatures and Blind Signatures. J. Cryptol. 13(3): 361-396 (2000)

Note that no such reduction is known for ***DSA***

Another strategy

Unforgeability by Reduction



Reduction:

- On input *y* runs a ***DKG*** with output ***PK***
- On input ***M*** runs a signature protocol that outputs a valid signature ***s***
- The challenge ***z*** is embedded in the above transcripts
- When adversary forges it uses the forgery to solve the challenge.

Note that the reduction does not have to hit any specific value. Just be able to embed the challenge.

Let’s go back to Pedersen’s DKG

- A joint parallel execution of Feldman’s VSS of random values
- It may be sufficient for a proof by reduction
  - Unfortunately not for ***DSA***
- If the reduction is able to embed a specific challenge into the parameters of the distributed scheme
  - And extract the solution from a forgery
- Then we can prove that the scheme is secure
  - In terms of unforgeability

- Note that this may not be sufficient for universal-composability
  - Simulation guarantees that no information at all is leaked
  - Allowing for arbitrary compositions

G, S.Jarecki, H.Krawczyk, T. Rabin: Secure Distributed Key Generation for Discrete-Log Based Cryptosystems. J. Cryptol. 20(1)

Threshold Schnorr with Pedersen's DKG

We use Pedersen's DKG both for the key and nonce generation

- At the end of the ***DKG*** the secret key is $x=x_H+x_C$
  - Where x_H is the contribution of the honest players
  - and x_C is the contribution of the corrupted ones
  - And x_H and x_C may be related
    - This is why the adversary can bias the resulting key

- Assuming honest majority
  - x_C Is known to the reduction
    - From the VSS since it controls the majority of players
  - This allows the reduction to embed the challenge in the contribution of one of the honest players

Proving the simplified Schnorr

How the reduction works

The reduction runs on input $z=g^u$ and needs to compute u

- During the ***DKG*** one honest player uses $z=g^u$ as their contribution for their Feldman's VSS
  - The reduction needs to simulate Feldman's VSS for this player
    - Remember we ***can*** simulate Feldman's VSS to hit a particular z
- The resulting secret key is $x=u+x_H+x_C$
  - Where x_H is the contribution of the ***other*** honest players
    - Known to the reduction since it choose it for them
  - and x_C is the contribution of the corrupted ones
    - Known to the reduction from the VSS and honest majority assumption
- Now adapt the standard reduction for the centralized Schnorr to the distributed case
  - When the adversary queries a message M
    - Use the regular Schnorr reduction to produce a valid partial signature for the player who used z
      - Since we don't know u the discrete log
      - Uses programmability of the random oracle
  - When the adversary produces the forgery
    - Use the regular Schnorr reduction to obtain x
    - Which in turns yields u (since the reduction knows x_H and x_C)

The state of the art

C.Komlo, I.Goldberg: FROST: Flexible Round-Optimized Schnorr Threshold Signatures. SAC 2020: 34-65

FROST

Two major improvements over the previous Schnorr threshold scheme

- Assume dishonest majority
  - At the end of the ***DKG*** the secret key is $x=x_H+x_C$
  - We can't assume that x_C is known to the reduction anymore
  - Modifies Pedersen's DKG: each party provides a proof of knowledge of their contribution
    - Using Schnorr's :)
  - Which makes x_C available to the reduction

- Uses a different idea to generate nonces
  - Each party i generates e_i and d_i and publishes $E_i = g^{ei}$ and $D_i = g^{di}$
  - Let M be the message and S the set of $t+1$ players signing
    - Set $k_i = e_i + r_i d_i$ as your additive share of the nonce k
    - Where $r_i = H(M,i,S)$
      - This binds the nonces to the message and the set of players signing making them effectively random across all executions

Let’s talk DKG

The challenge of a good DKG

The ***DKG*** protocols we discussed so far have the following drawbacks

- Requires synchronous networks
- Have quadratic communication
- Require several rounds to resolve complaints

Open problem: Design a truly scalable ***DKG***!

Publicly Verifiable DKG

Reducing Rounds via public verification

In Feldman's VSS parties check that their share match the public commitments

- If they don't we require communication rounds to lodge and resolve complaints
  - Or there is an immediate abort

Using Verifiable Encryption the correctness of shares vs. public commitment can be verified directly

- Dealer performs Feldman's VSS but does not send the shares privately to the parties
- Instead it encrypts them under their public keys and proves in ZK that they are correct values
  - With respect to the public Feldman's VSS commitments

An efficient implementation of this can be achieved with Paillier's encryption

Pierre-Alain Fouque, Jacques Stern: One Round Threshold Discrete-Log Key Generation without Private Channels. Public Key Cryptography 2001: 300-316

Reducing Communication via aggregation

If the VSS is ***publicly verifiable*** (as in the previous slide) then it is not necessary that each party verifies everybody else's VSS

- Instead of broadcasting its VSS to everybody a party ***gossips*** the VSS to a small group
- If the VSS is ***aggregatable*** each party aggregates all the VSS's it receives into a single one and gossips it again
- Eventually the ***DKG*** is the aggregation of all the VSSs with a large reduction in communication

A recent work constructs such a ***DKG*** where however the secret key is a group element, not a field

- Not usable for a "standard" signature scheme
- But they build a new ***Threshold Verifiable Unpredictable Function***

Kobi Gurkan, Philipp Jovanovic, Mary Maller, Sarah Meiklejohn, Gilad Stern, Alin Tomescu:Aggregatable Distributed Key Generation. EUROCRYPT (1) 2021: 147-176

Reducing Communication via trusted setup

Feldman's VSS is an example of a polynomial commitment:

- The dealer commits to a polynomial ***F[X]*** with coefficients $[f_o \dots f_t]$ by publishing $F_j = g^{f_j}$
- The value $x_i = F(i)$ can be checked by ***Evaluation in the exponent***

Is there a way to commit to a polynomial with a short (i.e. ***o(t)***) string?

- Yes! We know polynomial commitments where the public information and proof of correctness are constant
- However they require a trusted setup
  - **A.Kate, G.M.Zaverucha, I.Goldberg: Constant-Size Commitments to Polynomials and Their Applications. ASIACRYPT 2010: 177-194**

There are other polynomial commitments with sub-linear (non constant) parameters without trusted setup

- Very important in SNARKs
- Have not seen them used in DKGs