Advanced Tools from Modern Cryptography

Lecture 12

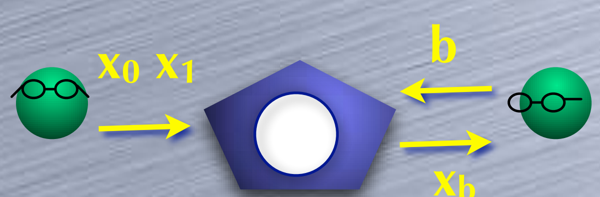
MPC: UC-secure OT

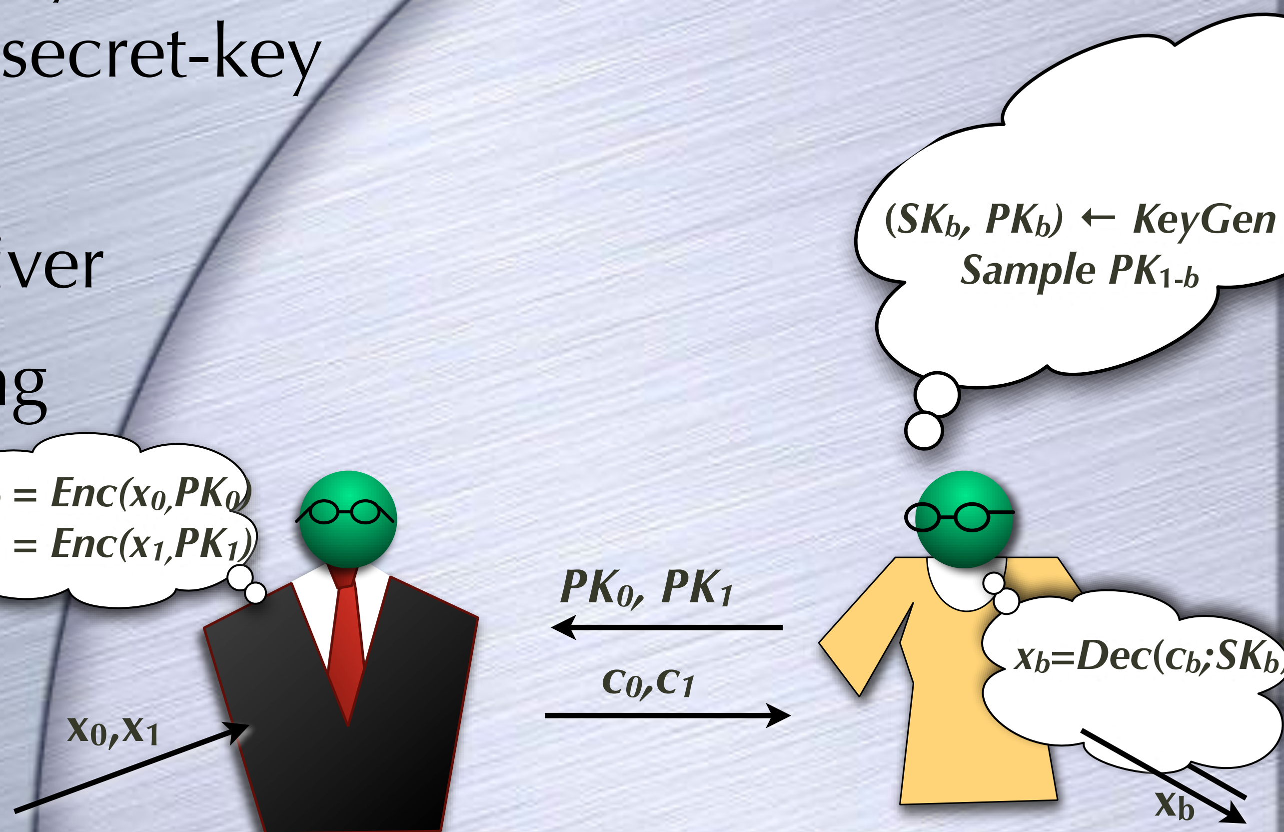
UC-Secure OT

- UC-secure OT is impossible (even against PPT adversaries) in the "plain model" (i.e., without the help of another functionality)
- But possible from simple setups
  - e.g., noisy channel (without computational assumptions)
  - e.g., random coins (needs computational assumptions)
  - Today: from Common random string
    - Like random coins, but reusable across multiple sessions

An OT Protocol (passive corruption)

- Using **(a special) encryption**
  - PKE in which one can sample a public-key without knowing secret-key
- c_{1-b} inscrutable to a passive corrupt receiver
- Sender learns nothing about b





Towards Active Security

- Should not let the receiver pick PK_0 and PK_1 independently!
- (PK_0,PK_1) tied together, in which at most one can be decrypted
  - $(PK_0,PK_1,SK) \leftarrow Gen(b)$ s.t. $check(PK_0,PK_1) = True$
    - (PK_0,PK_1) hides b. SK decrypts $Enc(m;PK_b)$, but not $Enc(m;PK_{1-b})$
    - But a simulator should be able to extract b from (PK_0,PK_1) (if Receiver corrupt) and m from $Enc(m;PK_{1-b})$ (if Sender corrupt)
      - Scheme will use a common random string Q (to be generated by a trusted party)
      - During simulation Simulator can generate (Q,T) where T is a Trapdoor that can be used for extraction

Towards Active Security

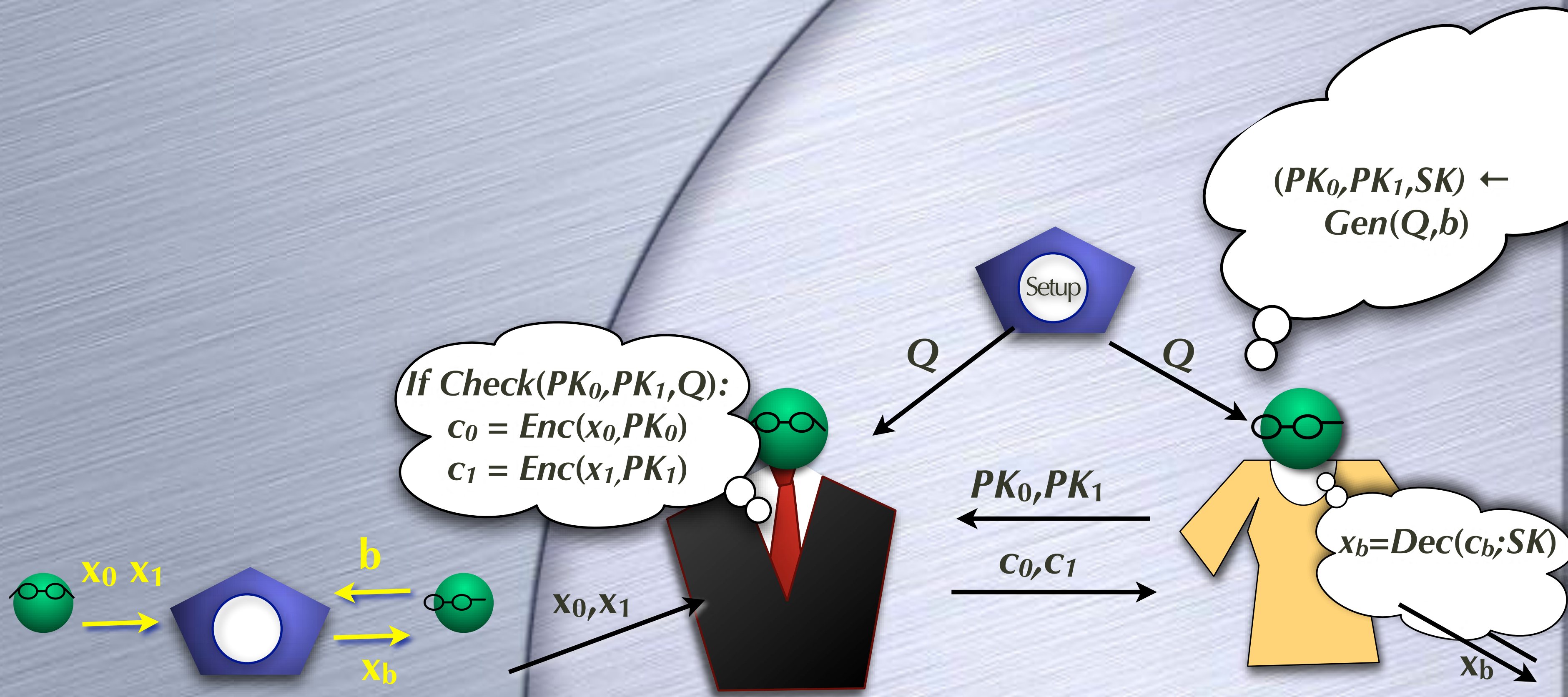
- Need: $(PK_0,PK_1,SK) \leftarrow Gen(Q,b)$ s.t. $check(PK_0,PK_1,Q) = True$.
  - (PK_0,PK_1) hides b. $Enc(m;PK_c)$ hides m for some c (even if (PK_0,PK_1) maliciously generated). Simulator should have trapdoors.
  - Suppose two different types of setups possible such that:
    Type 1 setup: For honest (PK_0,PK_1), b statistically hidden.
    Trapdoor decrypts both $Enc(m;PK_0)$ and $Enc(m;PK_1)$.
    Type 2 setup: Honest $Enc(m;PK_c)$ statistically hides m for some c.
    Trapdoor extracts a "lossy" c from any (PK_0,PK_1).
    Type 1 setup ≈ Type 2 setup (computationally)
  - (PK_0,PK_1) computationally hides b in Type 2 setup too.
    $Enc(m;PK_c)$ hides m for some c in Type 1 setup too.
  - Simulation when Sender corrupt: Use Type 1 setup
  - Simulation when Receiver corrupt: Use Type 2 setup

Dual-Mode Encryption (DME)

- Algorithms: $Setup_{Dec}$, $Setup_{Ext}$, Gen, Check, Enc, Dec
  - Q from $Setup_{Dec}$ and $Setup_{Ext}$ indistinguishable
  - If $(PK_0,PK_1,SK) \leftarrow Gen(Q,b)$, then $Check(PK_0,PK_1,Q)$=True, and $Dec(Enc(x,PK_b), SK) = x$
  - If PK lossy, then Enc(x,PK) statistically hides x
- Two more algorithms required to exist by security property: FindLossy and TrapKeyGen
  - Given trapdoor from $Setup_{Ext}$, and a pair PK_0, PK_1 which passes the Check, FindLossy can find a lossy PK out of the two
  - Given trapdoor from $Setup_{Dec}$, TrapKeyGen can generate PK_0, PK_1 which will pass the Check, along with decryption keys SK_0, SK_1

OT from DME

- Protocol could use either $Setup_{Dec}$ or $Setup_{Ext}$



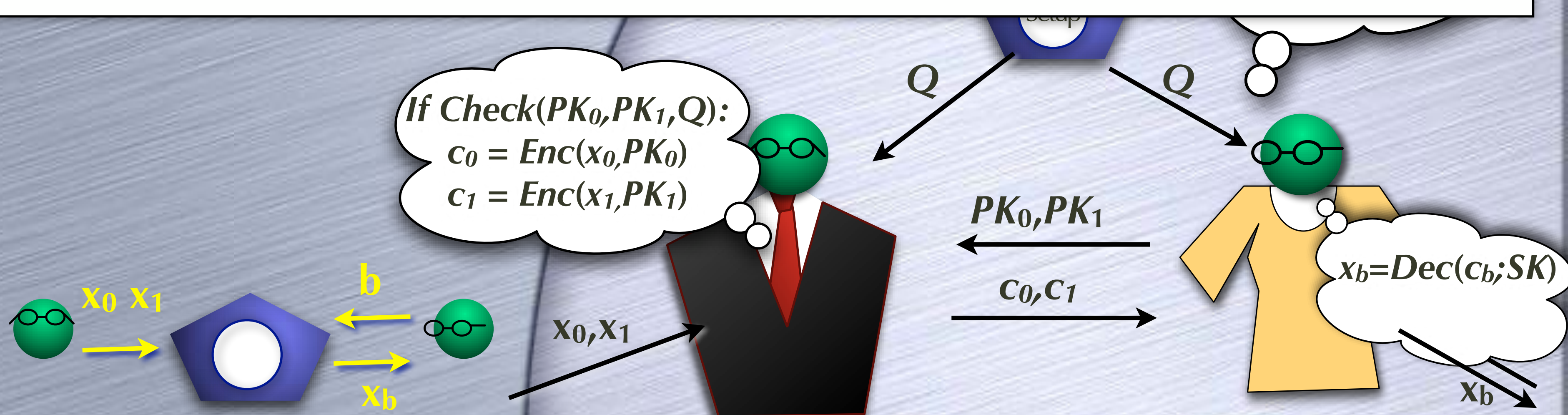
OT from DME

- Simulation for corrupt sender:

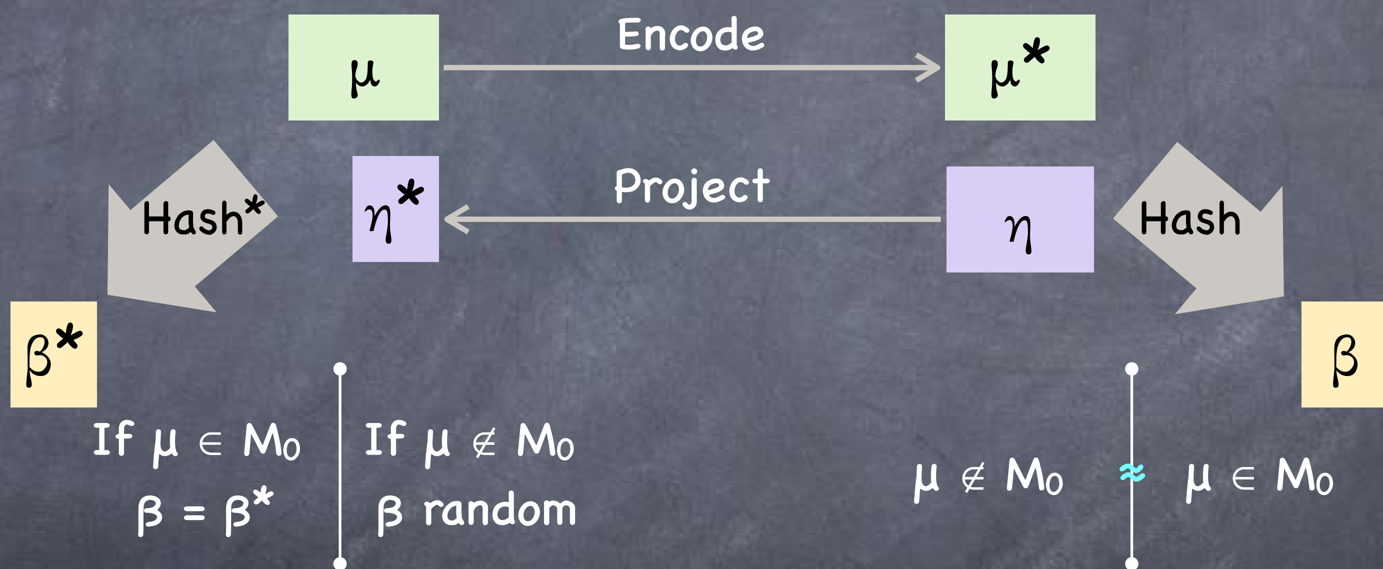
0. $(Q,T) \leftarrow \text{Setup}_{\text{Dec}}$, send Q.
1. Send $(PK_0,PK_1,SK_0,SK_1) \leftarrow \text{TrapKeyGen}(T)$
2. On getting (c_0,c_1), extract (x_0,x_1) using (SK_0,SK_1) and send to F_{OT}

- For corrupt receiver:

0. $(Q,T) \leftarrow \text{Setup}_{\text{Ext}}$, send Q.
1. On getting (PK_0,PK_1), send b:=1-FindLossy(PK_0,PK_1,T) to F_{OT}, get x_b
2. Send $c_b = \text{Enc}(x_b,\ PK_b)$ and $c_{1-b} = \text{Enc}(0,\ PK_{1-b})$



Smooth Projective Hash (SPH)

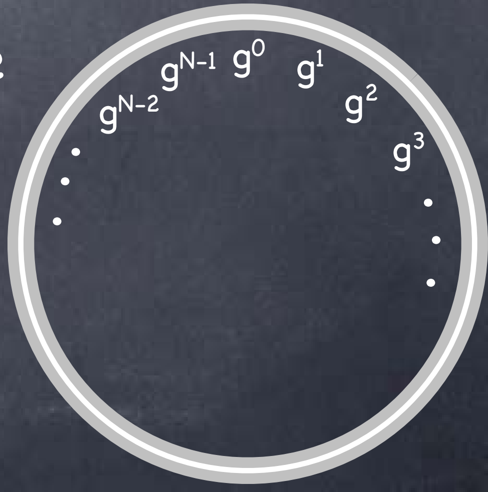


Smooth Projective Hash (SPH)

- Public parameters θ. Trapdoor parameters τ.
- Messages $\mu \in M$. Efficient Encode$_\theta$: $\mu \mapsto \mu^*$, a group homom. $M \rightarrow M^*$
  - Subgroup $M_0 \subseteq M$. Given τ and μ^*, can efficiently check if $\mu \in M_0$
- Hash key η with efficient Project$_\theta$: $\eta \mapsto \eta^*$
- Efficient Hash(μ^*,η) and Hash*(μ,η^*) s.t. $\forall\mu$, for random η:
  - If $\mu \in M_0$, then Hash(μ^*,η) = Hash*(μ,η^*)
  - If $\mu \notin M_0$, Hash(μ^*,η) statistically close to uniform, even given η^*
- Distributions $\{\mu^*\}_{\mu \leftarrow M_0} \approx \{\mu^*\}_{\mu \leftarrow M \setminus M_0}$
- Hash output is in a group too

Groups

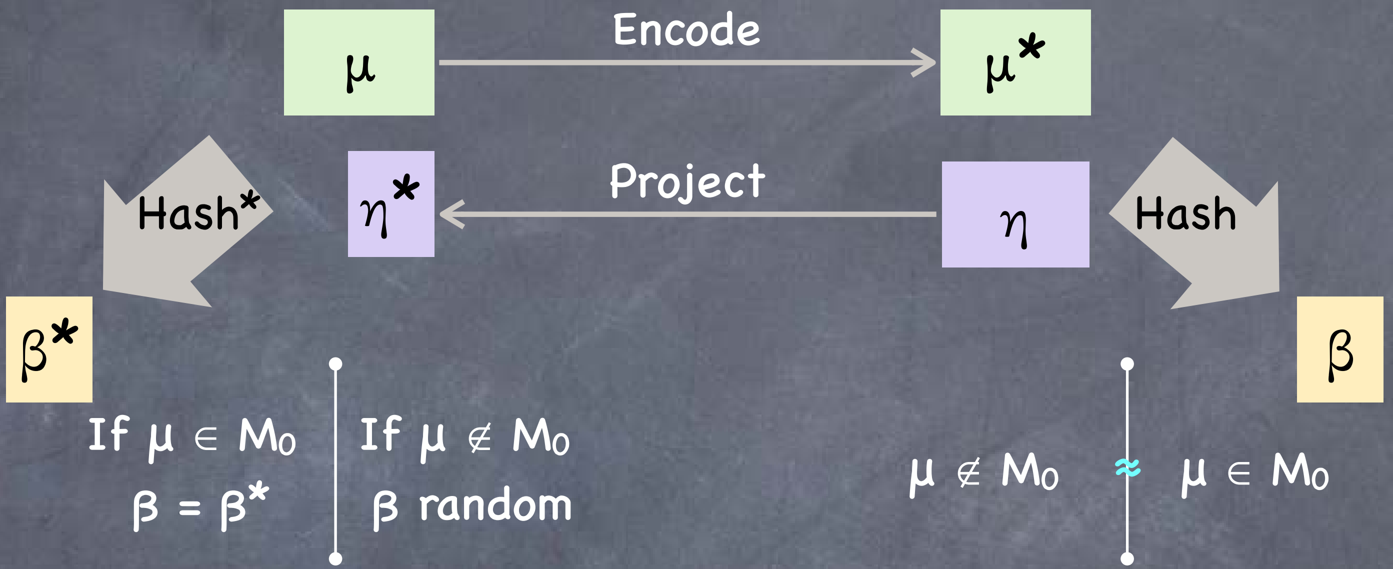
- A set G (for us finite, unless otherwise specified) and a "group operation" $*$ that is associative, has an identity, is invertible, and (for us) commutative
- Examples: $\mathbb{Z}$ = (integers, +) (this is an infinite group),
  $\mathbb{Z}_N$ = (integers modulo N, + mod N),
  G^n = (Cartesian product of a group G, coordinate-wise operation)
- Order of a group G: $|G|$ = number of elements in G
- For any $a \in G$, $a^{|G|} = a * a * \ldots * a$ ($|G|$ times) = identity
- Finite Cyclic group (in multiplicative notation): there is one element g such that $G = \{g^0, g^1, g^2, \ldots g^{|G|-1}\}$
  - Prototype: $\mathbb{Z}_N$ (additive group), with g=1.
    Corresponds to arithmetic in the exponent.



Decisional Diffie-Hellman (DDH) Assumption

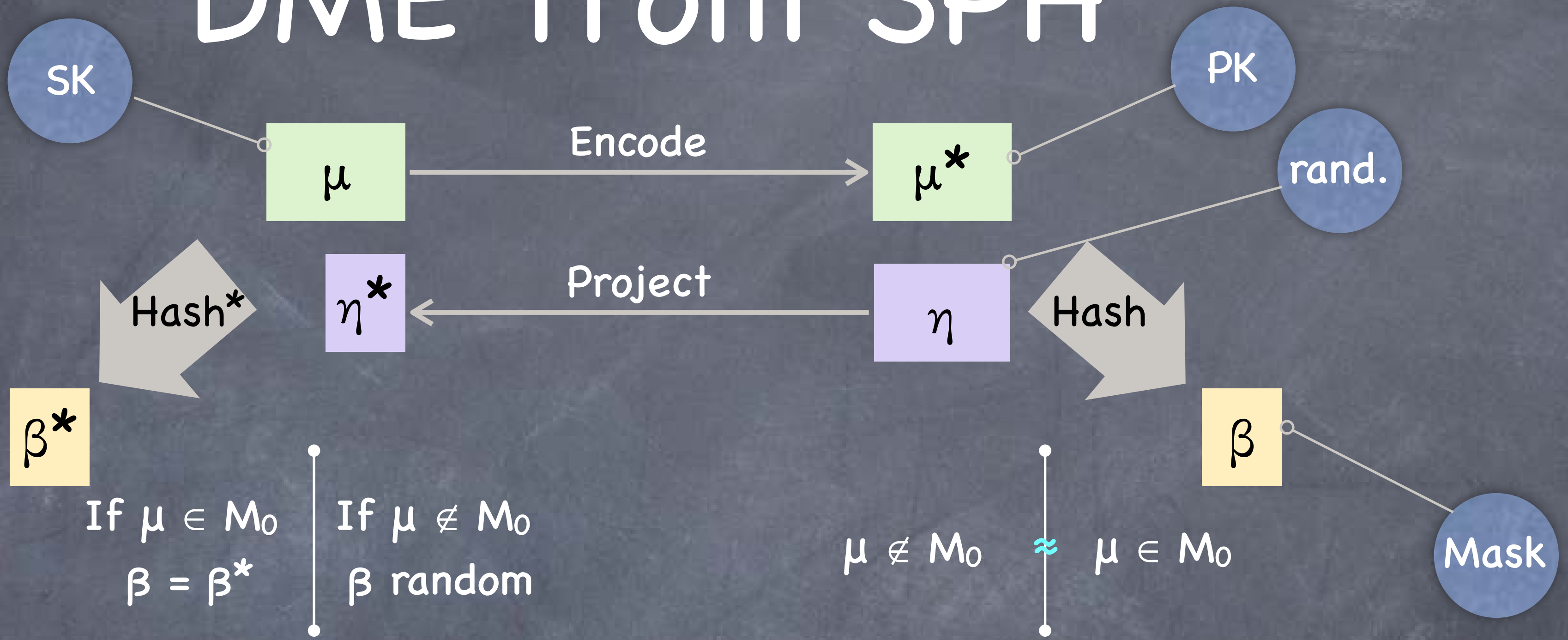
- Assumption about a distribution of finite cyclic groups and generators
- $\{(G, g, g^x, g^y, g^{xy})\}_{(G,g)\leftarrow\mathbf{Gen};\ x,y\leftarrow[|G|]} \approx \{(G, g, g^x, g^y, g^r)\}_{(G,g)\leftarrow\mathbf{Gen};\ x,y,r\leftarrow[|G|]}$
- Note: Requires that it is hard to find x from g^x
- Typically, G required to be a prime-order group. So arithmetic in the exponent is in a field.
- Formulation equivalent to DDH in prime-order groups:
  - $\{(G, g, g^a, g^b, g^{au}, g^{bu})\}_{(G,g),a,b,u} \approx \{(G, g, g^a, g^b, g^{au}, g^{bv})\}_{(G,g),a,b,u,v}$
    - If can distinguish the above, then can break DDH: map $(G, g, g^x, g^y, h) \mapsto (G, g, g^a, g^x, g^{y.a}, h)$

SPH from DDH Assumption



- SPH from DDH assumption on a prime order group G
  - $\{(G, g, g^a, g^b, g^{au}, g^{bu})\}_{(G,g),a,b,u} \approx \{(G, g, g^a, g^b, g^{au}, g^{bv})\}_{(G,g),a,b,u,v}$
- $\theta = (G,g,g^a,g^b)$, $\tau = (a,b)$
  $\eta = (s,t)$ and $\eta^* = g^{as+bt}$.
  $\mu = (u,v)$ and $\mu^* = (g^{a.u}, g^{b.v})$. $\mu \in M_0$ iff u=v.
  $\text{Hash}(\mu^*,\eta) = g^{a.u.s} \cdot g^{b.v.t}$ and $\text{Hash}^*(\mu,\eta^*) = g^{(as+bt).u}$

DME from SPH



- SPH gives a PKE scheme, with Hash as Enc, Hash* as Dec
- How to check that at least one of two PKs μ_0^*, μ_1^* is lossy?
  - Lossy means not in M_0^*
  - Setup contains $\mu^* \notin M_0^*$, and require that $\mu_0^* \cdot \mu_1^* = \mu^*$

DME from SPH

- Setup: Sample SPH params (θ,τ). Let $\mu \leftarrow M$. Let $Q=(\mu^*,\theta)$, $T=(\mu,\tau)$
  - $\text{Setup}_{\text{Dec}}$: $\mu \in M_0$. $\text{Setup}_{\text{Ext}}$: $\mu \notin M_0$.
- Gen(Q,b): $(PK_0,PK_1) = (\mu_0^*,\mu_1^*)$ where $\mu_b \leftarrow M_0$ and $\mu_{1-b}^* = \mu^* \mu_b^{*-1}$
Check (PK_0,PK_1,Q): check $\mu_0^* \cdot \mu_1^* = \mu^*$.
  - If $\mu \notin M_0$, given (μ_0^*,μ_1^*) s.t. $\mu_0^* \cdot \mu_1^* = \mu^*$, at least one of μ_0,μ_1 not in M_0. Can find using τ. (FindLossy)
  - If $\mu \in M_0$, using μ can find (μ_0,μ_1) s.t. $\mu_0^* \cdot \mu_1^* = \mu^*$ and both $\mu_0,\mu_1 \in M_0$ (TrapKeyGen)
- $\text{Enc}(x,\mu_b^*)$: $(\eta^*, x \cdot \text{Hash}(\mu_b^*,\eta))$ where η random
  - x assumed to be in the group of Hash output
- $\text{Dec}(c,\mu_b)$ where $c=(\eta^*,\alpha)$ and $\mu_b \in M_0$: Ouput $\alpha.(\text{Hash}^*(\mu_b,\eta^*))^{-1}$

OT from DME

- Protocol could use either $Setup_{Dec}$ or $Setup_{Ext}$