Advanced Tools from Modern Cryptography

Lecture 12

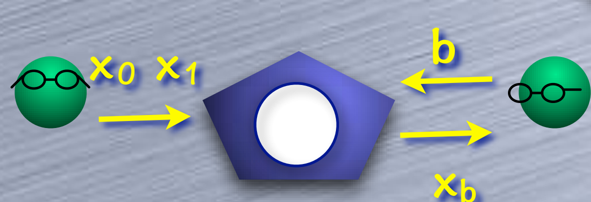
MPC: UC-secure OT

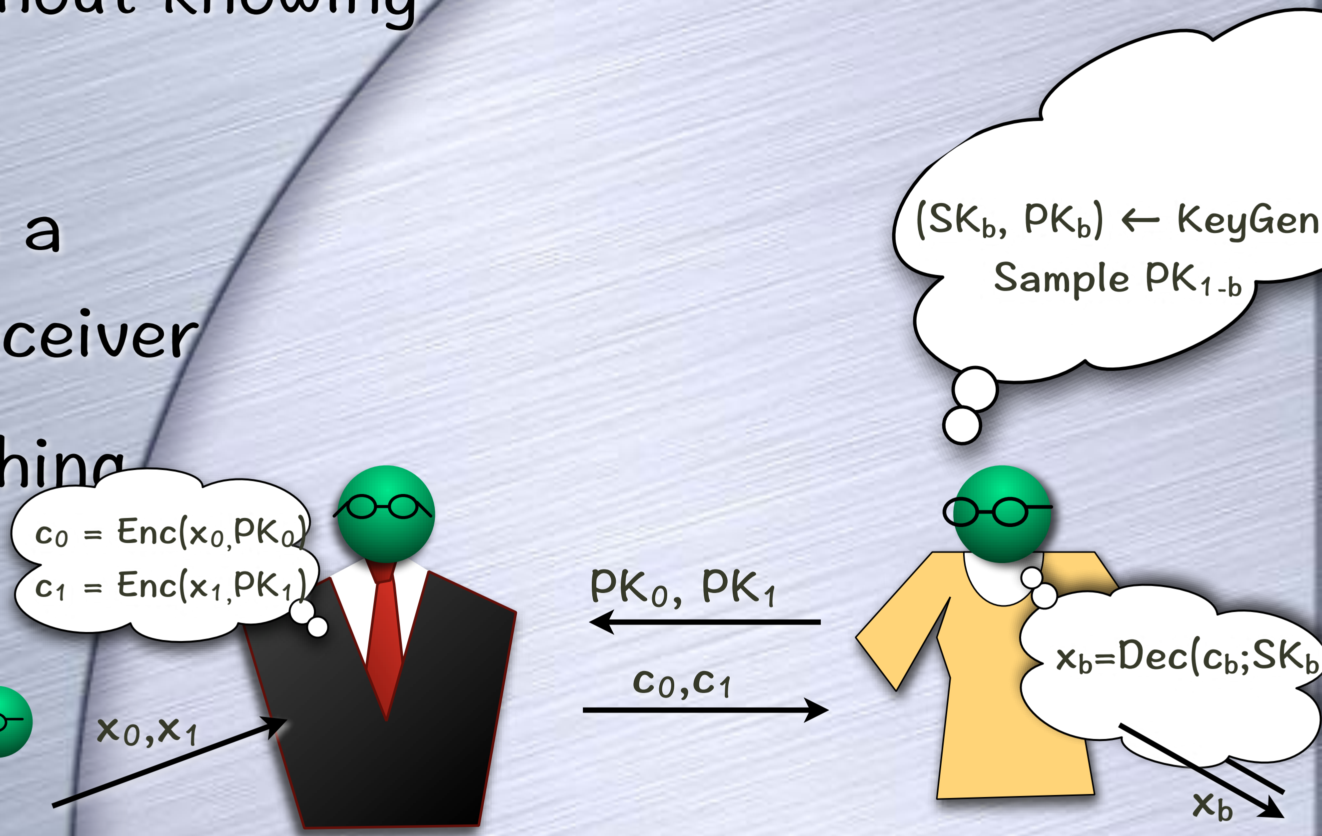
UC-Secure OT

- 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., common random coins (needs computational assumptions)
  - Today: from <u>Common random string</u>
    - Like common 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. $check(PK_0,PK_1) = True$
    - SK decrypts $Enc(m;PK_b)$, but not $Enc(m;PK_{1-b})$. (PK_0,PK_1) hides 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 <u>common random string</u> 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: $\mathrm{Gen}(Q,b)$ and $\mathrm{check}(PK_0,PK_1,Q)$ such that
  - If $(PK_0,PK_1,SK) \leftarrow \mathrm{Gen}(Q,b)$: SK decrypts $\mathrm{Enc}(m;PK_b)$, (PK_0,PK_1) hides b.
  - If $\mathrm{check}(PK_0,PK_1,Q) = \mathrm{True}$: $\mathrm{Enc}(m;PK_c)$ 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: Honestly generated (PK_0,PK_1) <u>statistically</u> hides b.
  Trapdoor decrypts both $\mathrm{Enc}(m;PK_0)$ and $\mathrm{Enc}(m;PK_1)$.
  Type 2 setup: Honest $\mathrm{Enc}(m;PK_c)$ <u>statistically</u> hides m for some c.
  Trapdoor extracts such a c from any (PK_0,PK_1).
  Type 1 setup ≈ Type 2 setup (computationally)

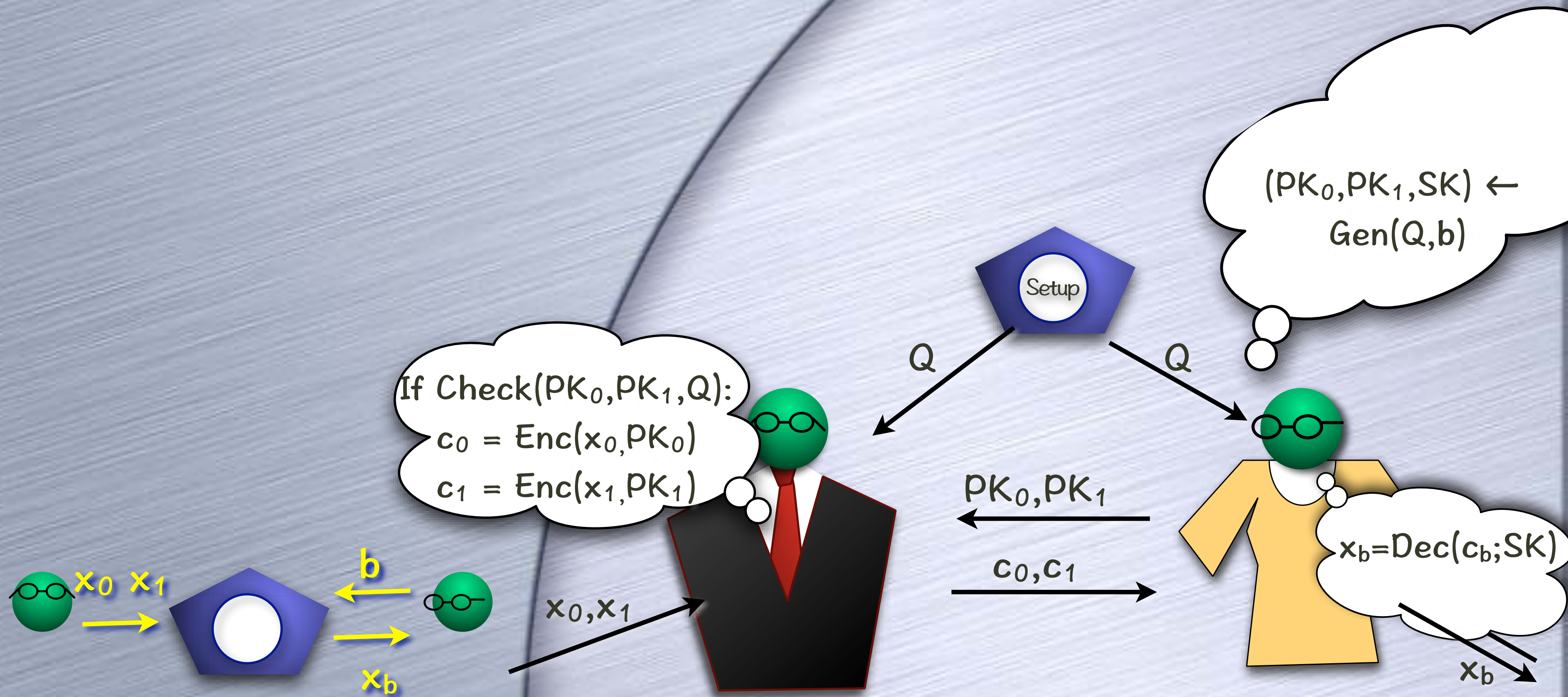
  PK_c said to be "lossy"
- (PK_0,PK_1) computationally hides b in Type 2 setup too.
  $\mathrm{Enc}(m;PK_c)$ computationally hides m for some c in Type 1 setup too.
- Simulation when Sender corrupt: Use Type 1 setup
- Simulation when Receiver corrupt: Use Type 2 setup

Dual-Mode Encryption (DME)

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

OT from DME

- Protocol could use either $\text{Setup}_{\text{Dec}}$ or $\text{Setup}_{\text{Ext}}$



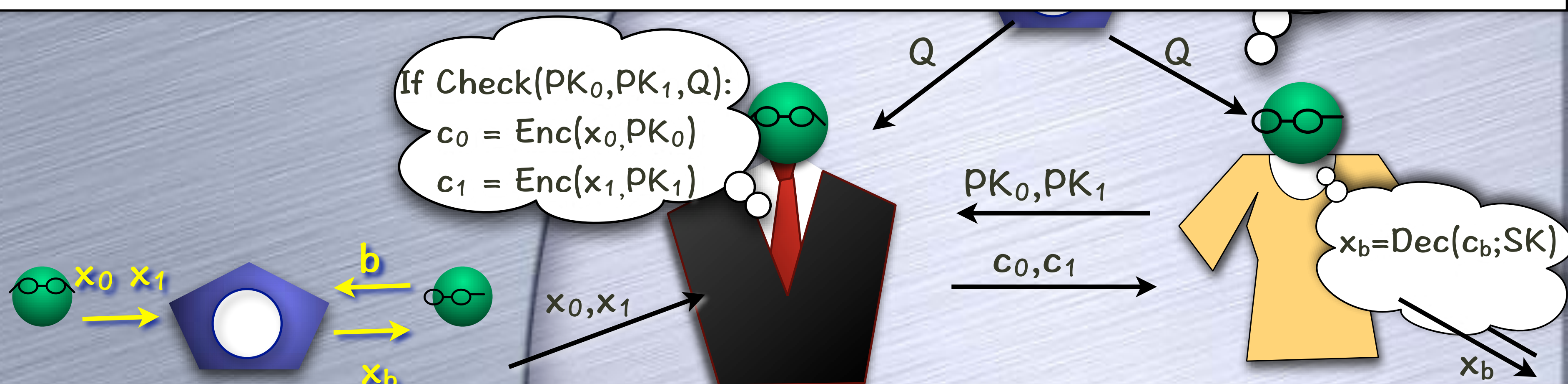
OT from DME

- Simulation for corrupt sender:

0. $(Q,T) \leftarrow \text{Setup}_{\text{Dec}}$, send Q.
1. $(PK_0,PK_1,SK_0,SK_1) \leftarrow \text{TrapKeyGen}(T)$, and send (PK_0,PK_1)
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:

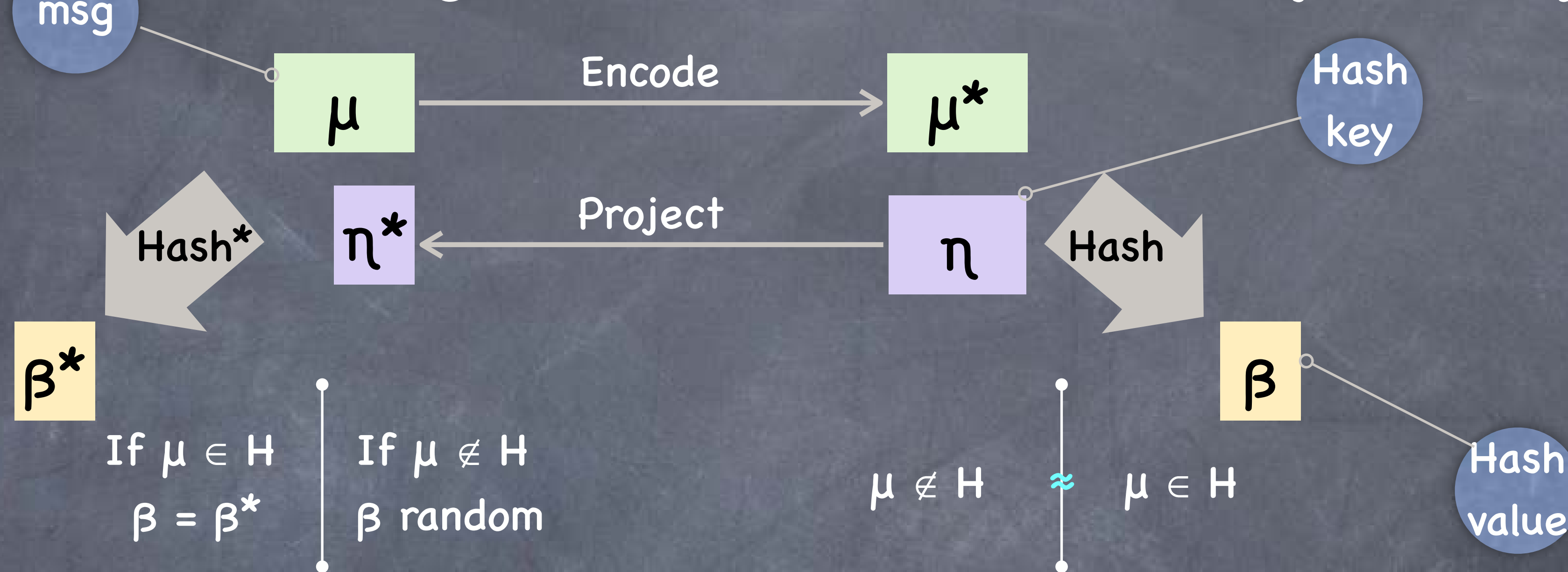
0. $(Q,T) \leftarrow \text{Setup}_{\text{Ext}}$, send Q.
1. On getting (PK_0,PK_1), send $b{:=}1\text{-FindLossy}(PK_0,PK_1,T)$ to F_{OT}, get x_b
2. Send $c_b = \text{Enc}(x_b, PK_b)$ and $c_{1-b} = \text{Enc}(0, PK_{1-b})$



Dual-Mode Encryption (DME)

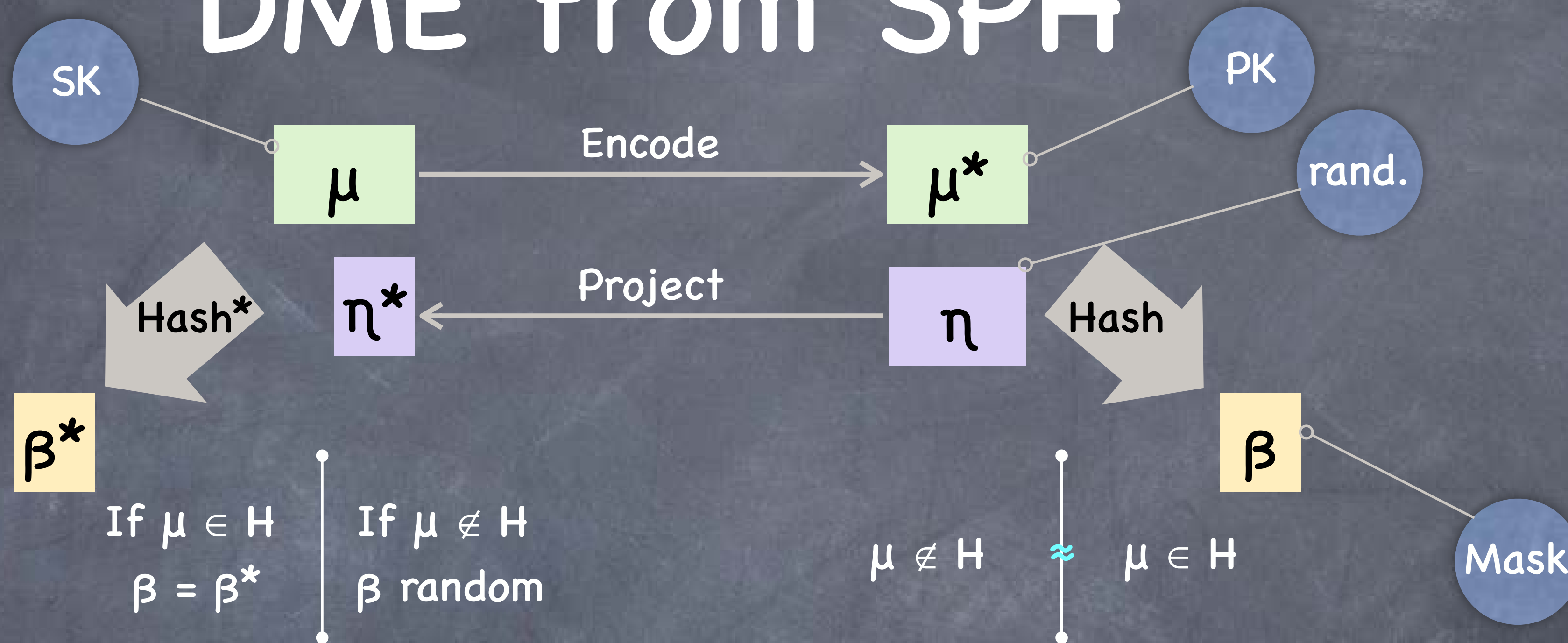
- High-level idea for constructing a DME
  - PKE s.t. a (hidden) subset of the PK-space is "lossy"
  - Q = PK. Require that $PK_0 \cdot PK_1 = PK$
    - Receiver can pick only one PK_b. Other gets determined by Q
    - But maybe both can still be non-lossy!
  - Fix: Non-lossy subset is a sub-group, and Q = PK, a lossy key
    - $PK_0 \cdot PK_1 = PK \Rightarrow$ not both in the non-lossy subgroup!
- Coming up: A primitive called SPH which allows a DME construction as above
  - And a construction of SPH from "Decisional Diffie-Hellman" assumption

Smooth Projective Hash (SPH)



- Public parameters θ used by all algorithms. Trapdoor τ
- Encode: $M \longrightarrow M^*$ is a group homomorphism
- $H \subseteq M$ group s.t. given only θ, distributions $\{\mu^*\}_{\mu \leftarrow H} \approx \{\mu^*\}_{\mu \leftarrow M \setminus H}$
  - But using τ, can perfectly distinguish the two distributions
    - So, $\mu \in H \Leftrightarrow \mu^* \in H^*$, where $H^* = \{ \mu^* \mid \mu \in H \}$ a group

DME from SPH



- SPH gives a PKE scheme, with Hash as Enc, Hash* as Dec
- Setup: Sample SPH params (θ,τ). Let $\mu \leftarrow M$. Let $Q=(\mu^*,\theta)$, $T=(\mu,\tau)$
  - $\text{Setup}_{\text{Dec}}$: $\mu \in H$. $\text{Setup}_{\text{Ext}}$: $\mu \notin H$.
- If $\mu^* \notin H^*$, given (μ_0^*,μ_1^*) s.t. $\mu_0^* \cdot \mu_1^* = \mu^*$, at least one of $\mu_0,\mu_1 \notin H$. Can find using τ. (FindLossy)

This is Check(PK_0,PK_1)

- If $\mu^* \in H^*$, using μ, can find (μ_0,μ_1) s.t. $\mu_0^* \cdot \mu_1^* = \mu^*$ and both $\mu_0,\mu_1 \in H$ (TrapKeyGen)

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),

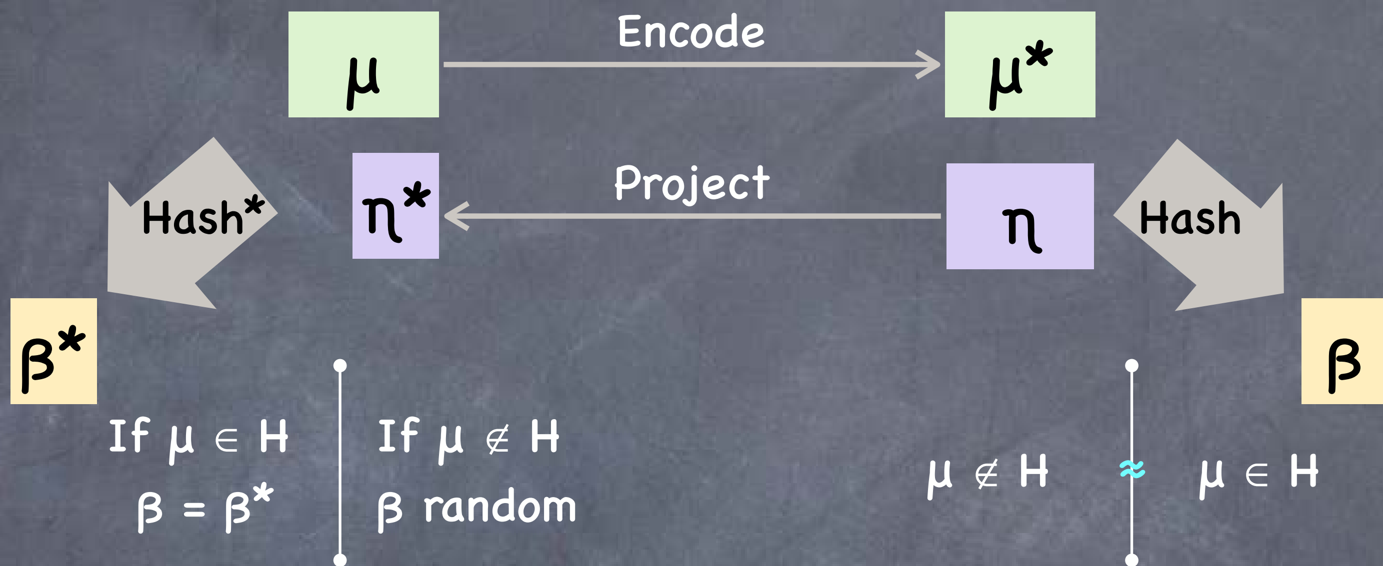
  G^n = (Cartesian product of a group G, coordinate-wise operation)
- Order of a group G: $|G|$ = number of elements in G
- For any $a \in G$, $a^{|G|} = a * a * \dots * a$ (|G| times) = identity
- Finite Cyclic group (in multiplicative notation): there is one element g such that $G = \{g^0, g^1, g^2, \dots g^{|G|-1}\}$
  - 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
- $\{(G, g, g^x, g^y, g^{xy})\}_{(G,g)\leftarrow Gen;\ x,y\leftarrow[|G|]} \approx \{(G, g, g^x, g^y, g^r)\}_{(G,g)\leftarrow Gen;\ x,y,r\leftarrow[|G|]}$
- 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.
- A 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 $\mu^* = (g^{a.u}, g^{b.v})$. $\mu \in H$ iff $u=v$.
  $\mathrm{Hash}(\mu^*,\eta) = g^{a.u.s}\, g^{b.v.t}$ and $\mathrm{Hash}^*(\mu,\eta^*) = g^{(as+bt).u}$

For random s,t, and u≠v, and non-zero a,b, aus+bvt is random given only (as+bt,u,v,a,b)