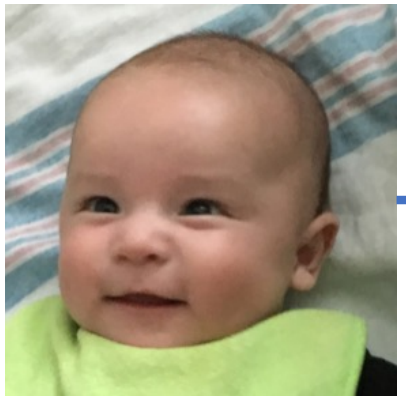
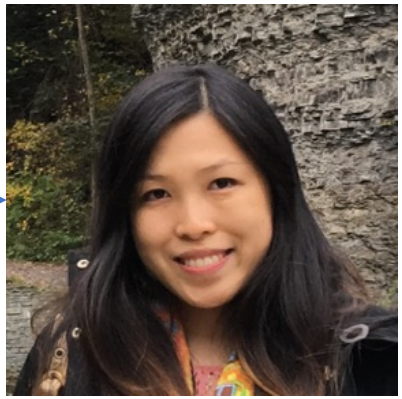
On ELFs, Deterministic Encryption, and Correlated Input Security

Mark Zhandry

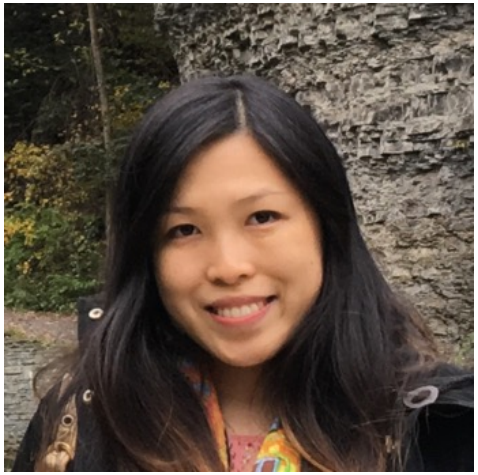
Princeton University



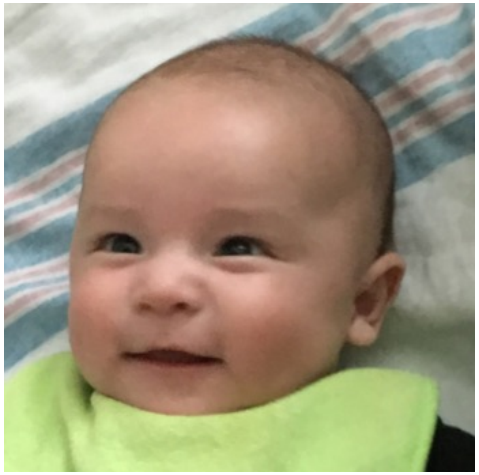
"mommy > daddy"



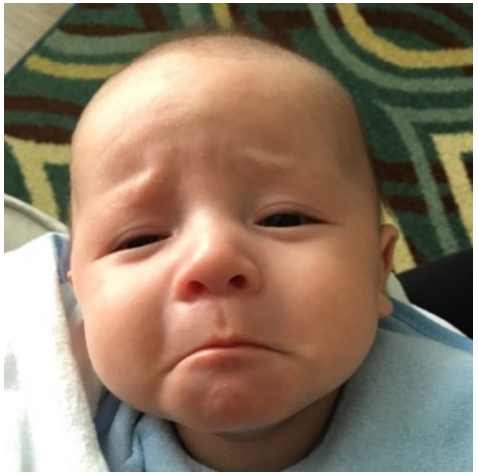
In reality…

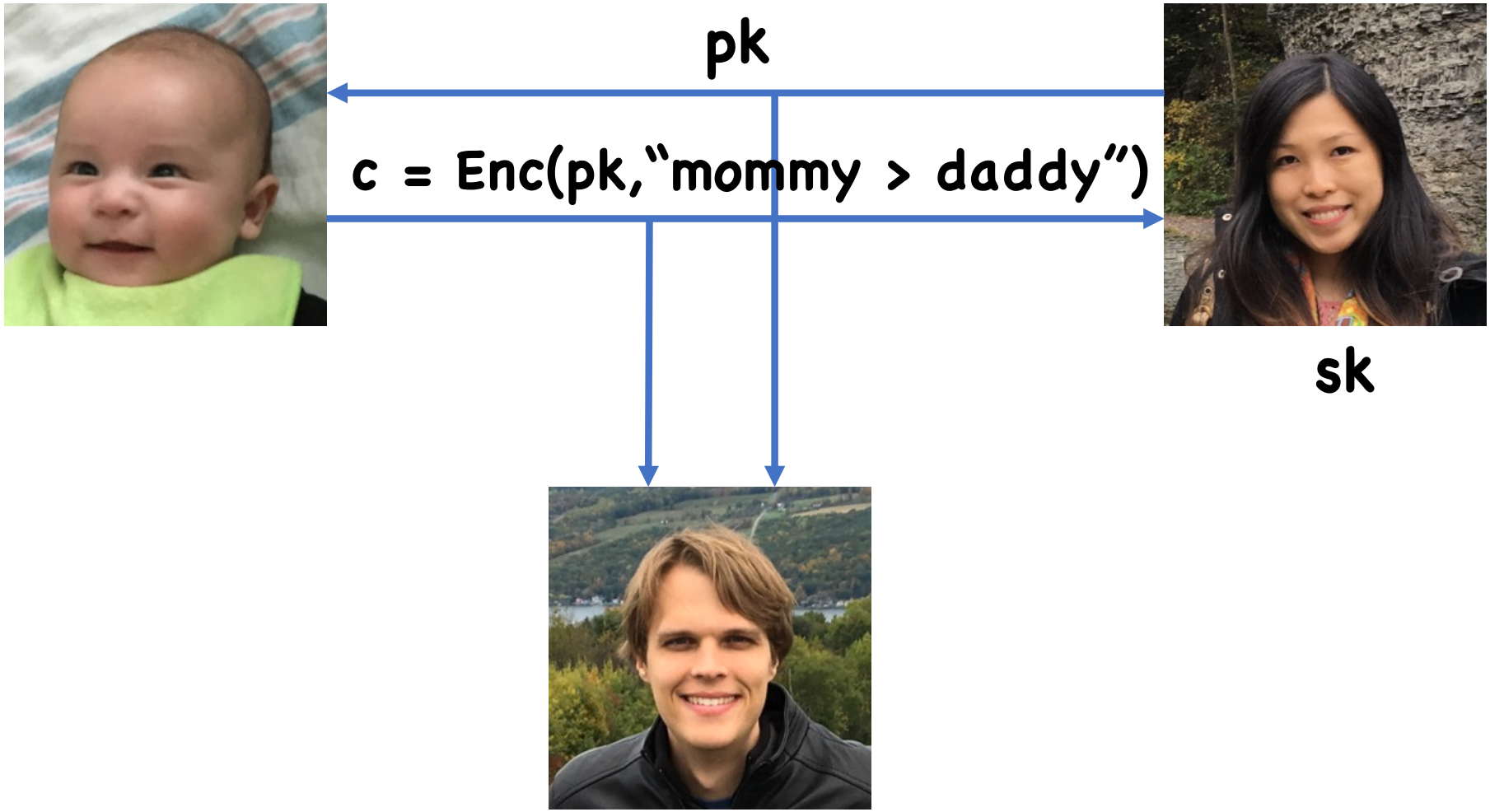


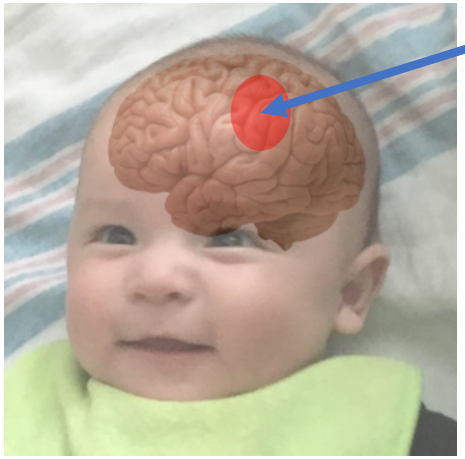
=



=




pk
c = Enc(pk,"mommy > daddy")
sk



Random Number Cortex:

r = 0000000000…….

Deterministic Public Key Encryption (DPKE)

Pros:

- ✓ No randomness needed
- ✓ Public equality test

Cons:

- ✗ Harder to construct
- ✗ Semantic security impossible
- ✗ Need unpredictable messages
- ✗ Multiple messages?

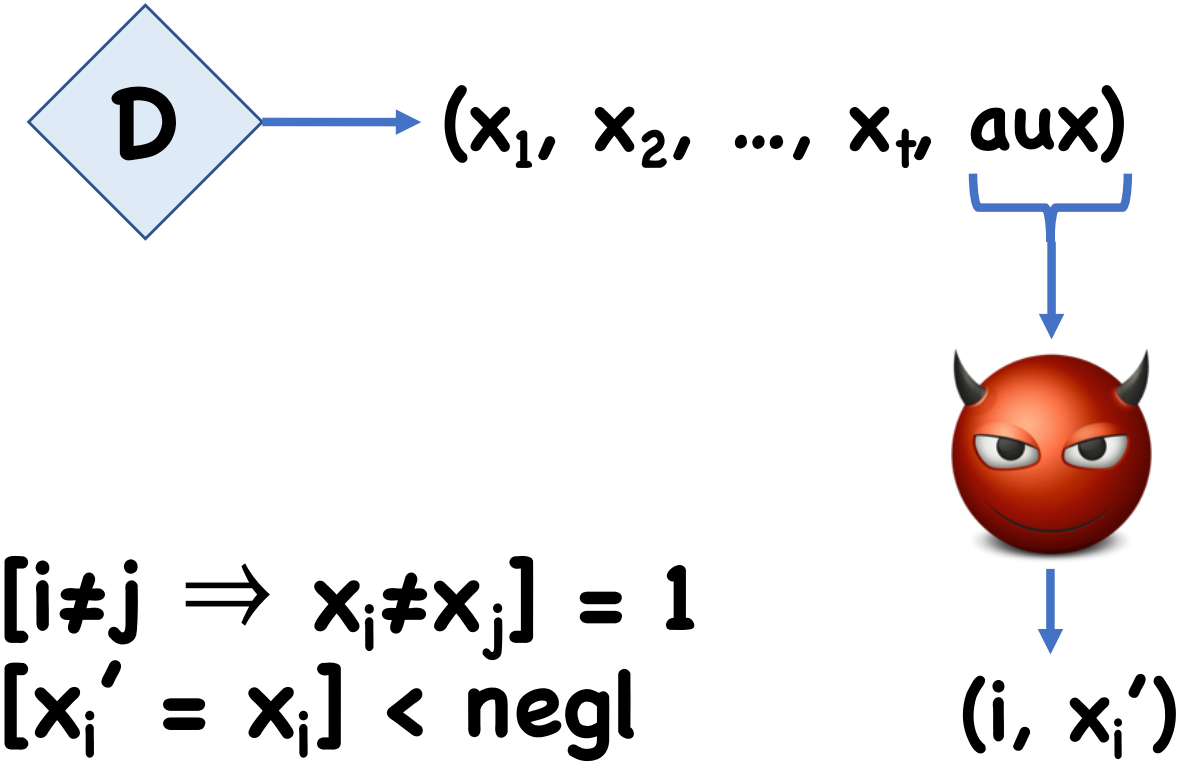
This Work

DPKE secure under

- Arbitrary computationally unpredictable sources
- *Constant* number of arbitrarily correlated sources
- Chosen ciphertext attacks

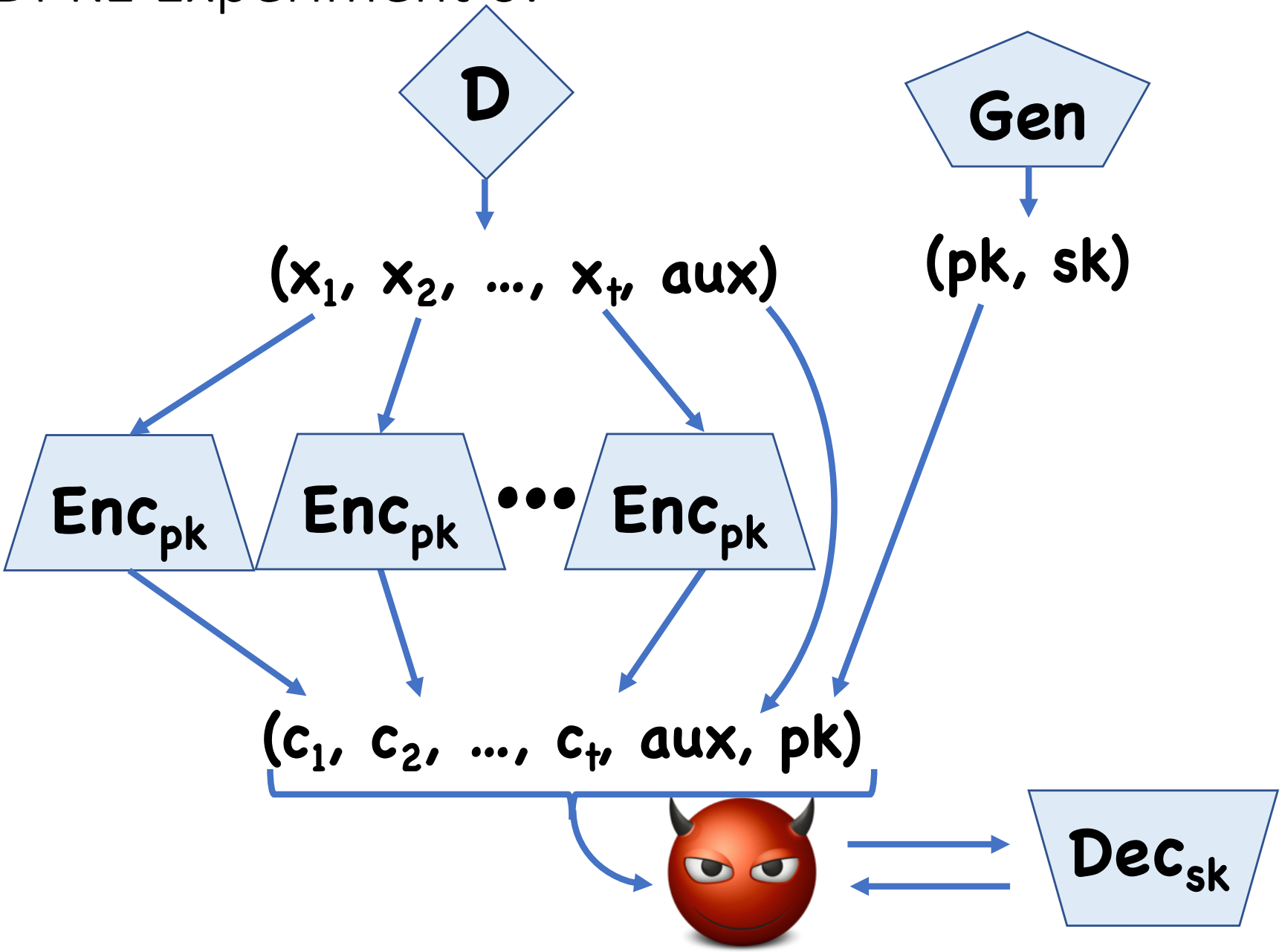
Computational assumption: exponential DDH

Computationally Unpredictable Sources

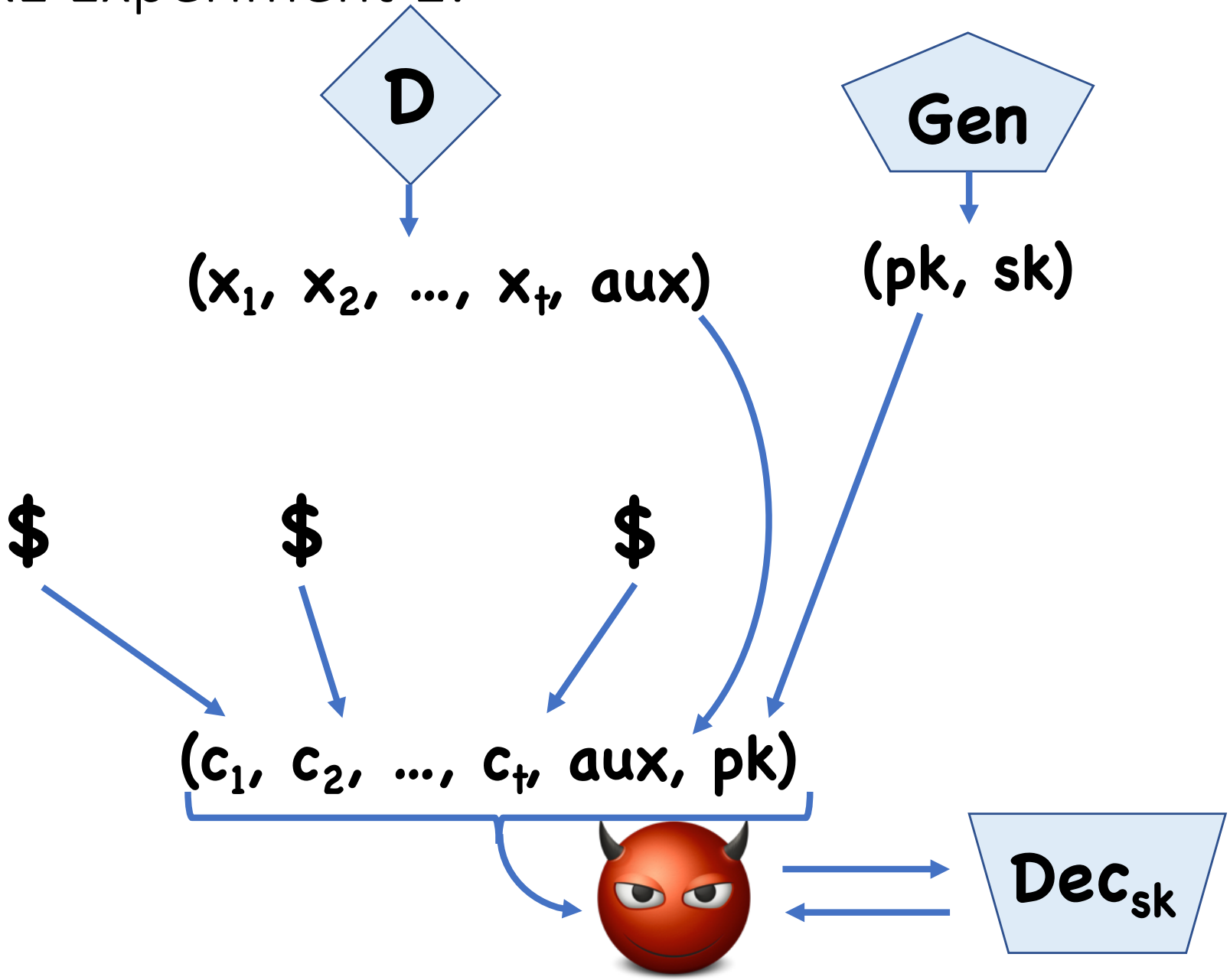


$$\Pr[i \neq j \Rightarrow x_i \neq x_j] = 1$$
$$\Pr[x_i' = x_i] < negl$$

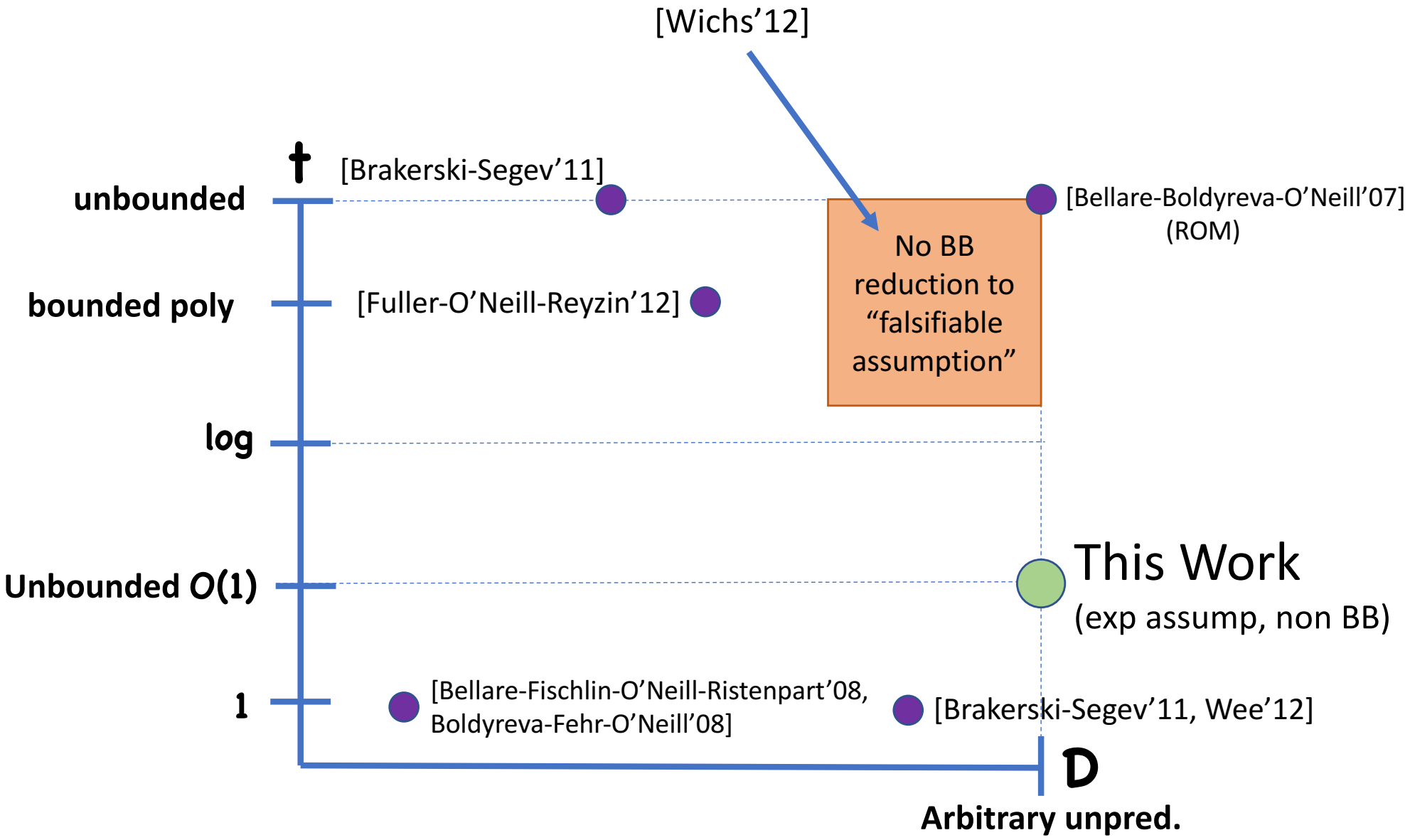
DPKE Experiment 0:



DPKE Experiment 1:



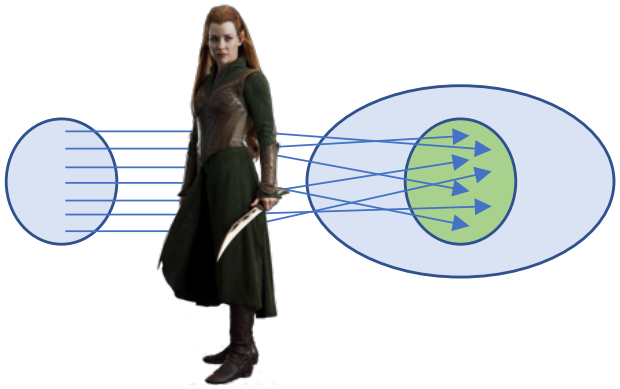
Some Prior Work



Step 1: **t=1**, No CCA queries

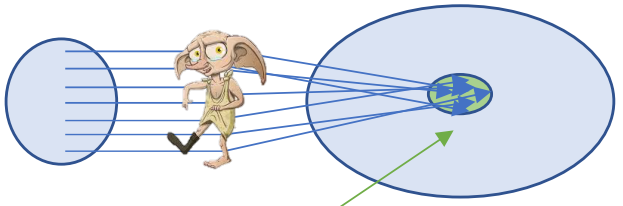
Extremely Lossy Functions (ELFs) [Z'16]

Injective Mode:



$\approx_c$

Lossy Mode:

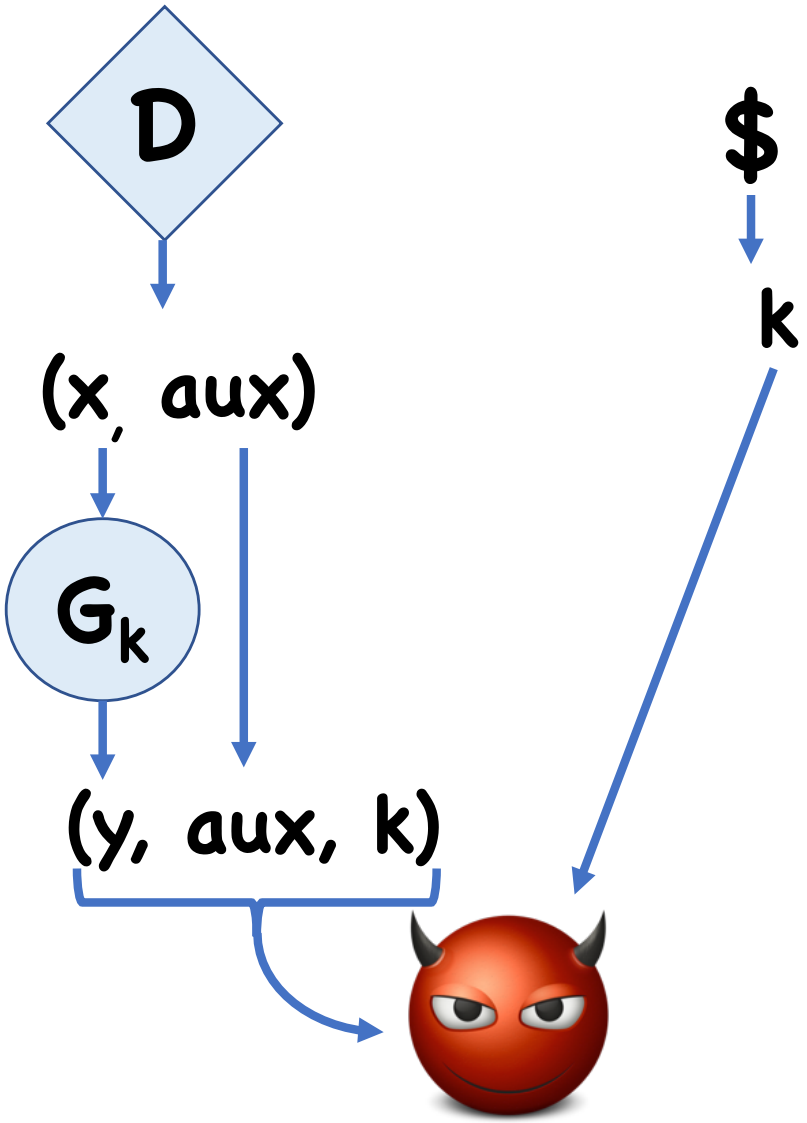


| Img | = polynomial*

Thm [Z'16]: Exponential DDH $\Rightarrow$ ELFs

*Technically **|Img|** depends on adversary

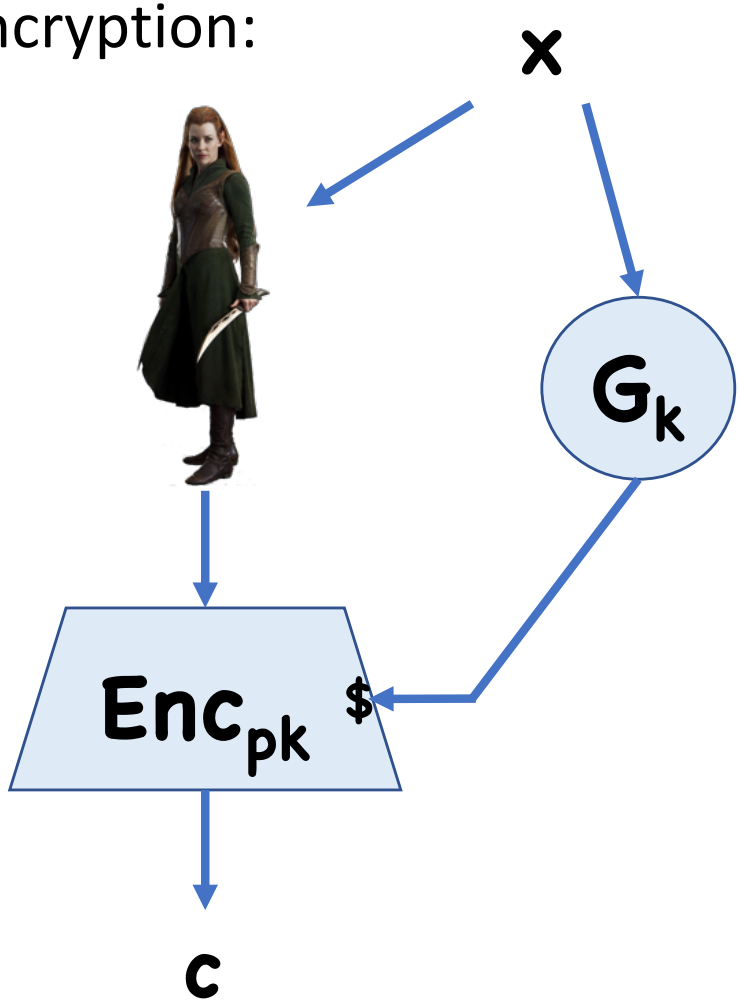
PRGs for Comp. Unpred. Sources, **t=1**



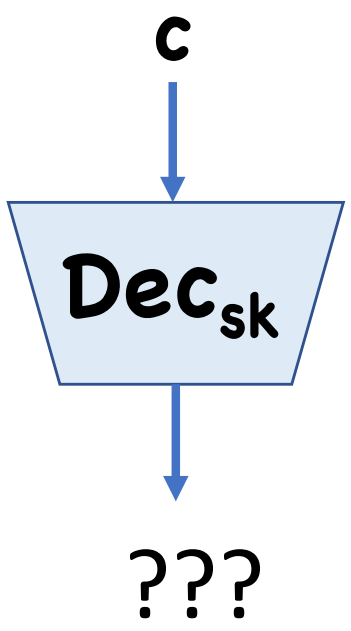
Thm [Z'16]: ELFs $\Rightarrow$ PRGs for arbitrary **1**-CU sources

Upgrading to DPKE

Encryption:

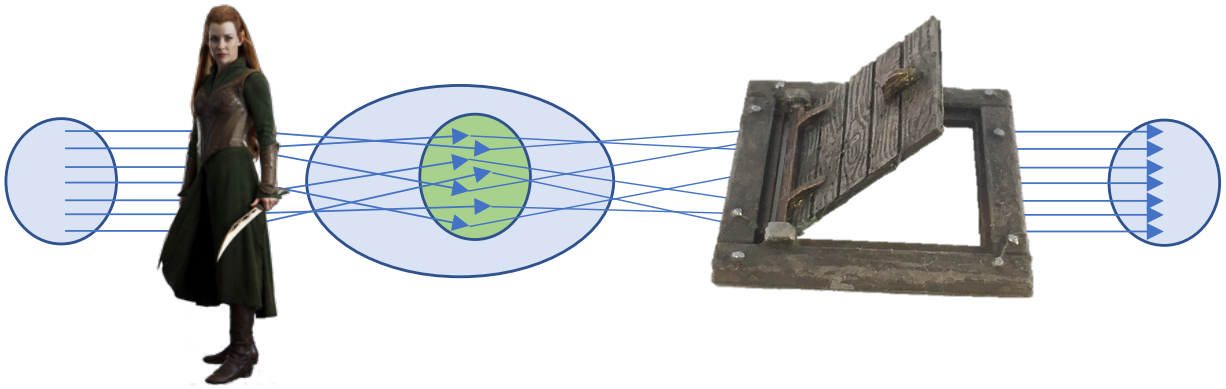


Decryption:

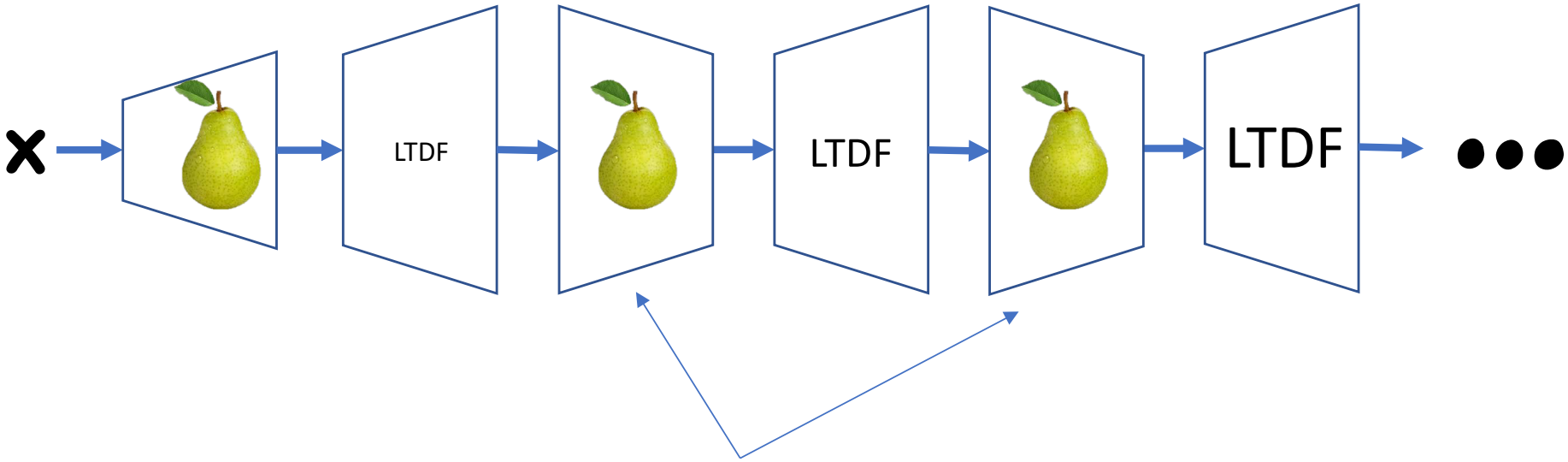


New Tool: Trapdoor ELFs

Injective Mode:



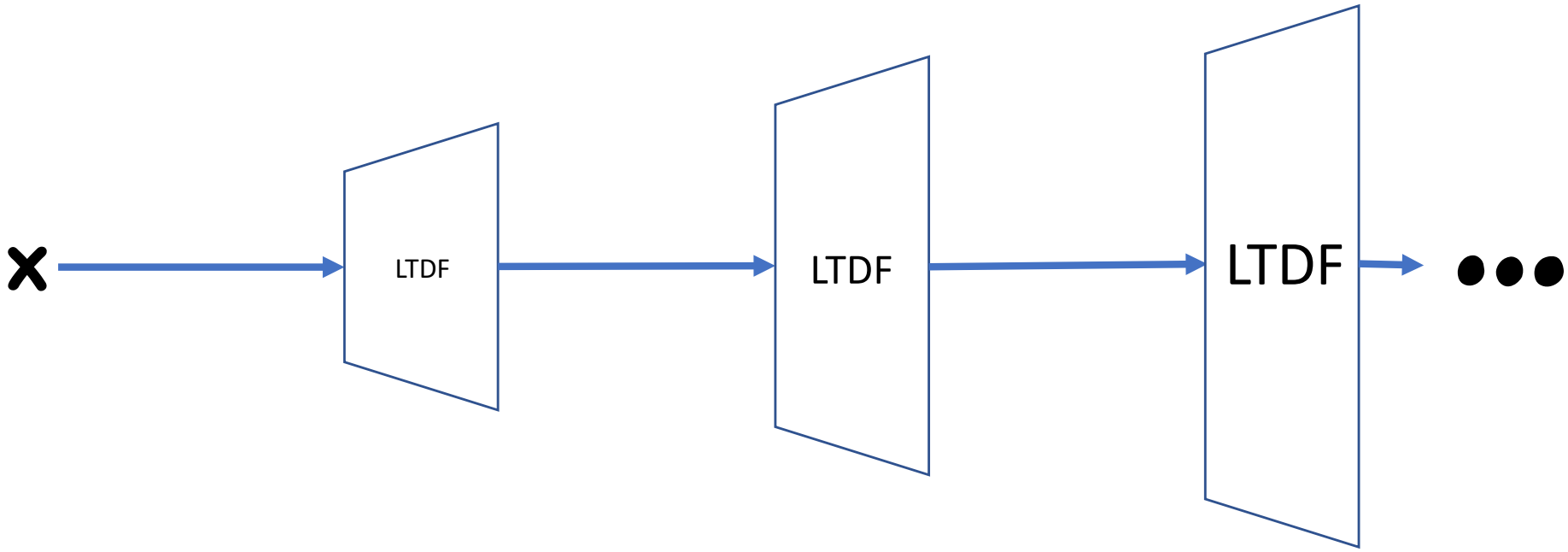
Constructing T-ELFs



Compression kills trapdoor

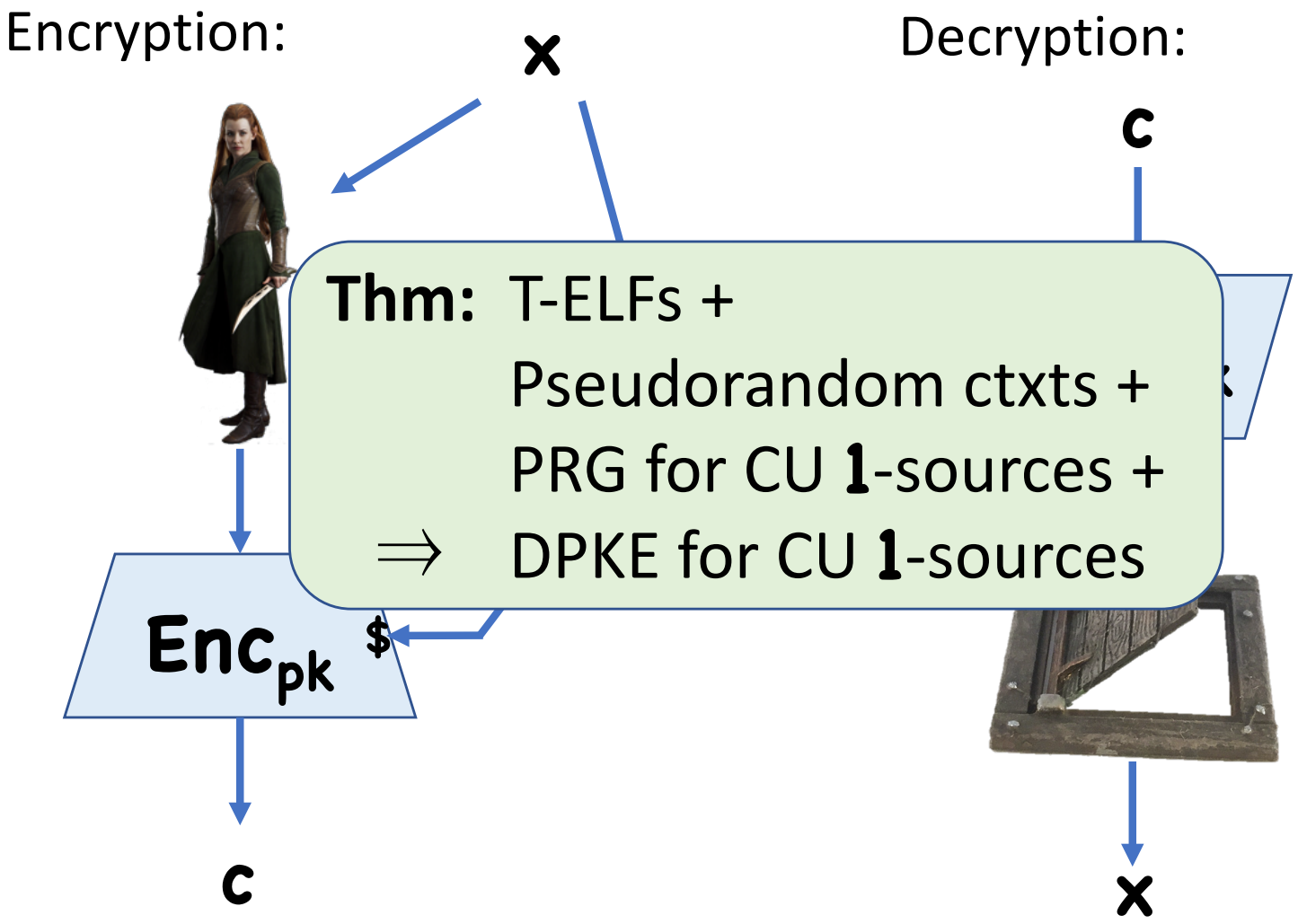
= Pairwise independent function

Constructing T-ELFs



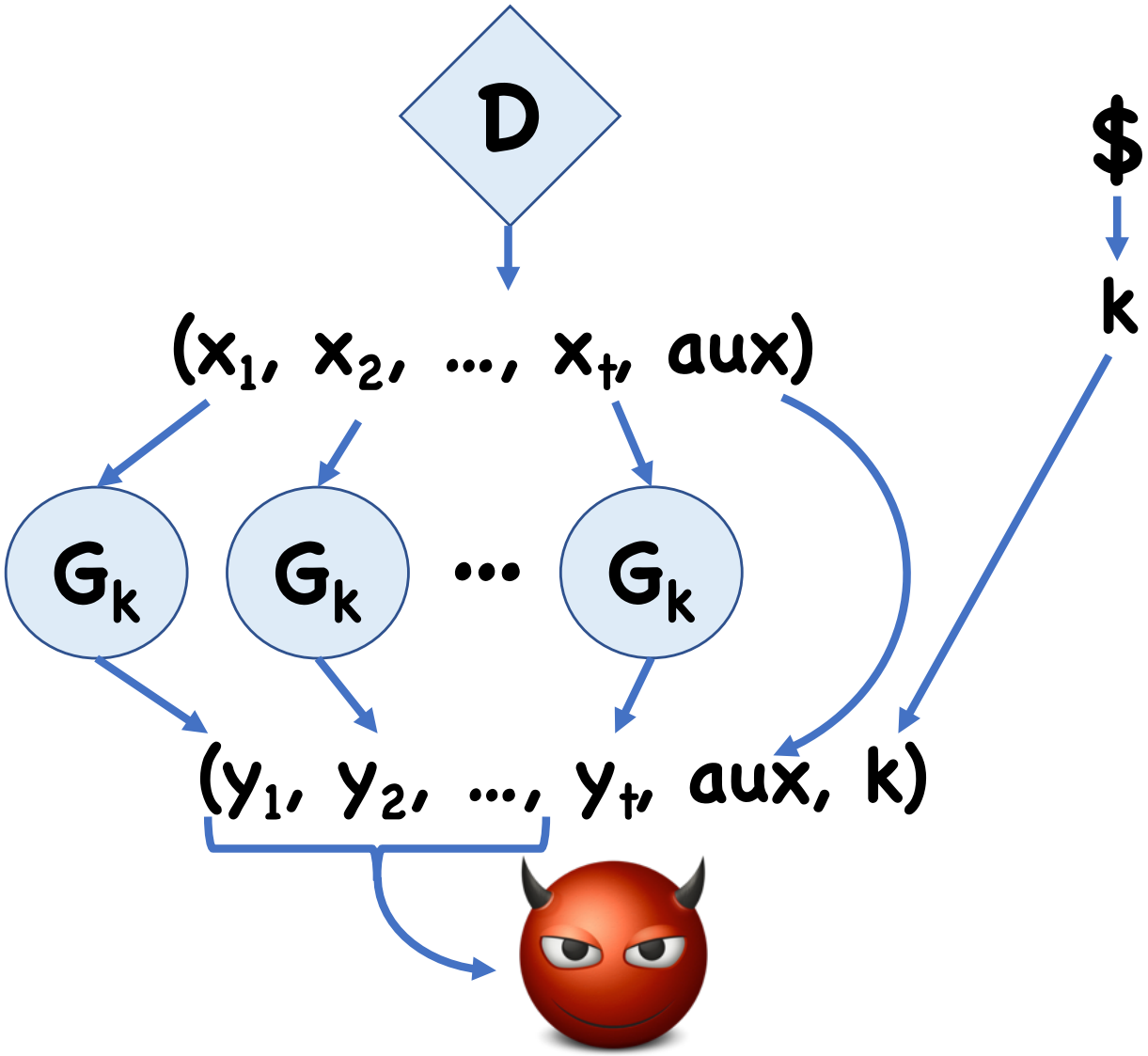
In paper: instantiate parameters such that growth isn't too big

Upgrading to DPKE



Step 2: Constant $\dagger$, No CCA queries

PRGs for Comp. Unpred. Sources, **t=O(1)**

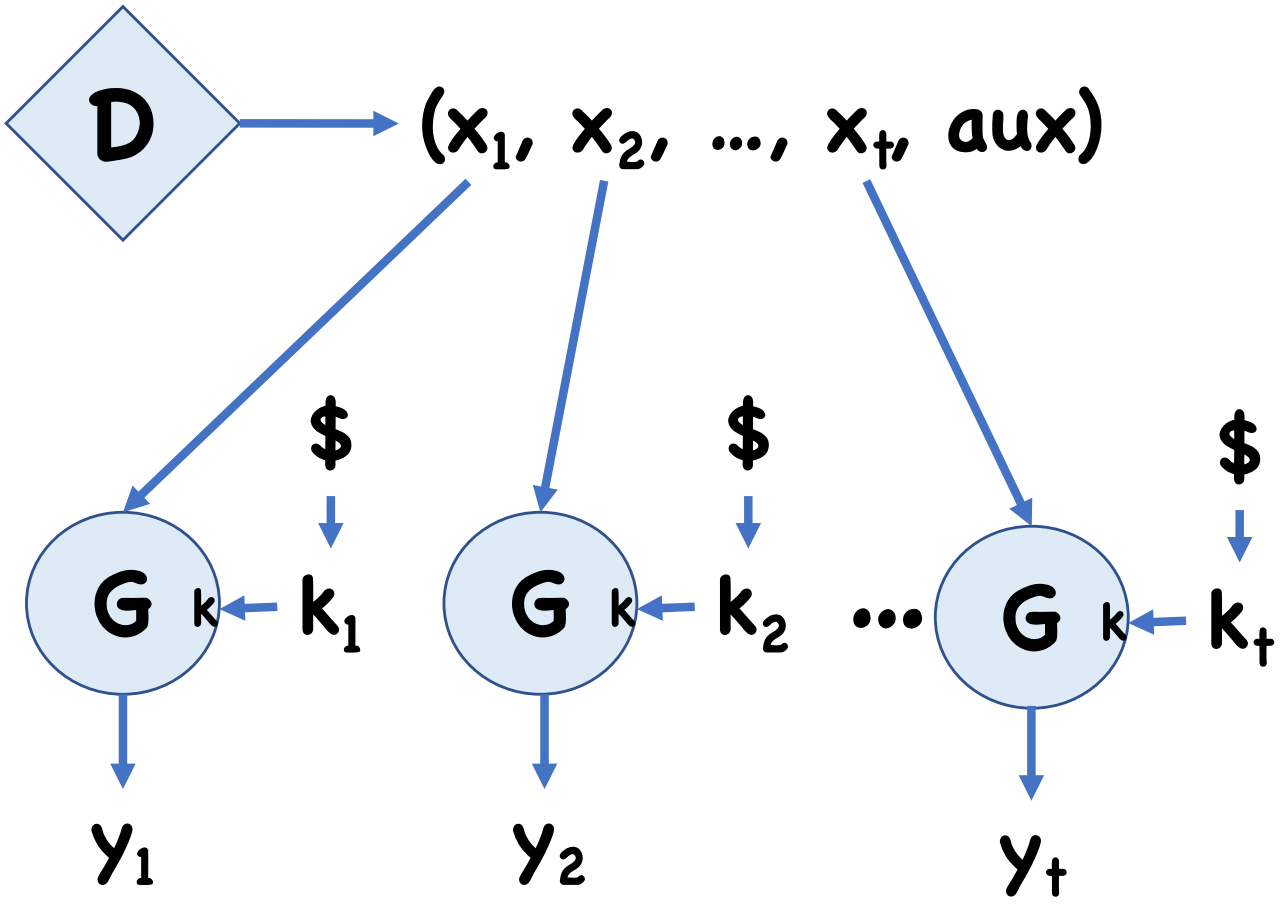


Step 2: Constant $\dagger$, No CCA queries

Thm: T-ELFs +
Pseudorandom ctxts +
PRG for CU $O(1)$-sources +
$\Rightarrow$ DPKE for CU $O(1)$-sources

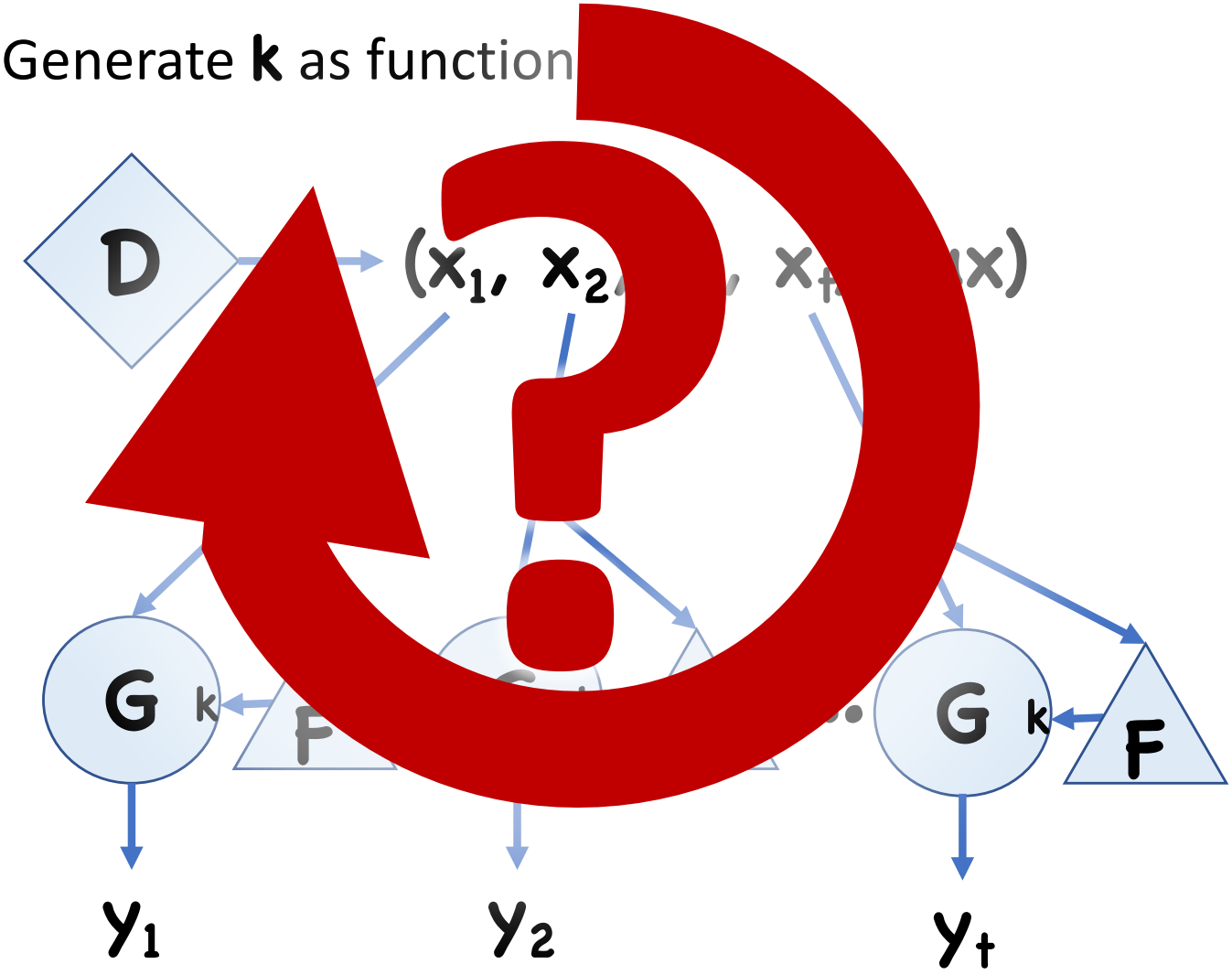
PRG for CU **O(1)**-sources

Idea 1: each $\mathbf{x_i}$ gets it's own PRG for CU **1**-sources



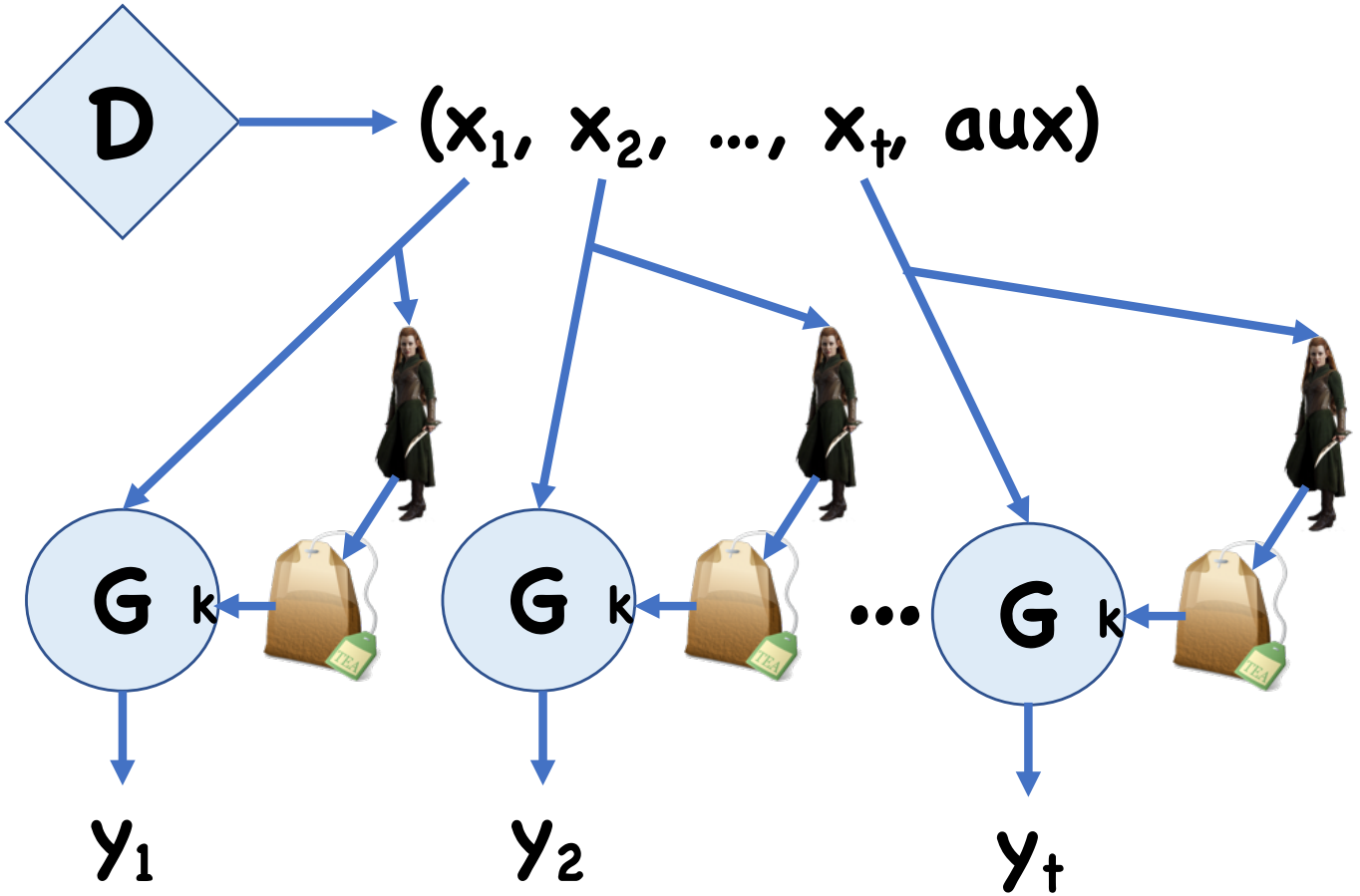
PRG for CU **O(1)**-sources

Idea 2: Generate **k** as function



PRG for CU **O(1)**-sources

Idea 3: Break circularity using **t**-wise independence + ELFs



= **t**-wise independent func

Step 3: CCA Security

See paper…

Difficulties arise:

- Need “branched” T-ELFs

- T-ELFs are much more delicate than LTDFs

  $\Rightarrow$ Generic approaches don’t work

- Instead, modify construction directly

Now time for a nap …