Crash course in computer-checked crypto proofs

in the Foundational Cryptography Framework

The game-playing approach leads to proofs that are

less error-prone
more easily verifiable
mechanically verifiable

main idea:
adversary guesses what "world" it's in

Bellare (2004)

Prove security of small scheme:
apply PRF once to generate bits that are
indistinguishable from random

Game-based crypto proof

Assume f is a PRF.

Game START:

1. randomly sample an initial vector v
2. compute s = (f v)
3. give s to the adversary
4. adversary guesses whether s was randomly sampled or came from PRF

Pr[Start] = hopefully 1/2 + (small)

Game-based crypto proof

Game END:

1. randomly sample s
2. give s to the adversary
3. adversary guesses whether s was randomly sampled or came from PRF

Pr[End] = 1/2

Want to prove:
difference b/t
guessing world START and
world END is small

Pr[Start] - Pr[End] = epsilon

Games as probabilistic imperative code!

Game to code

Assume f is a PRF

Game START:

1. randomly sample an initial vector v
2. compute s = (f v)
3. give s to the adversary
4. adversary guesses whether s was randomly sampled or came from PRF

```
Variable RndS : Comp S.

Definition DRBG_G0 :
Comp Bool :=

  s <-$ RndS;
  A (f s).
```

Game to code

Game END:

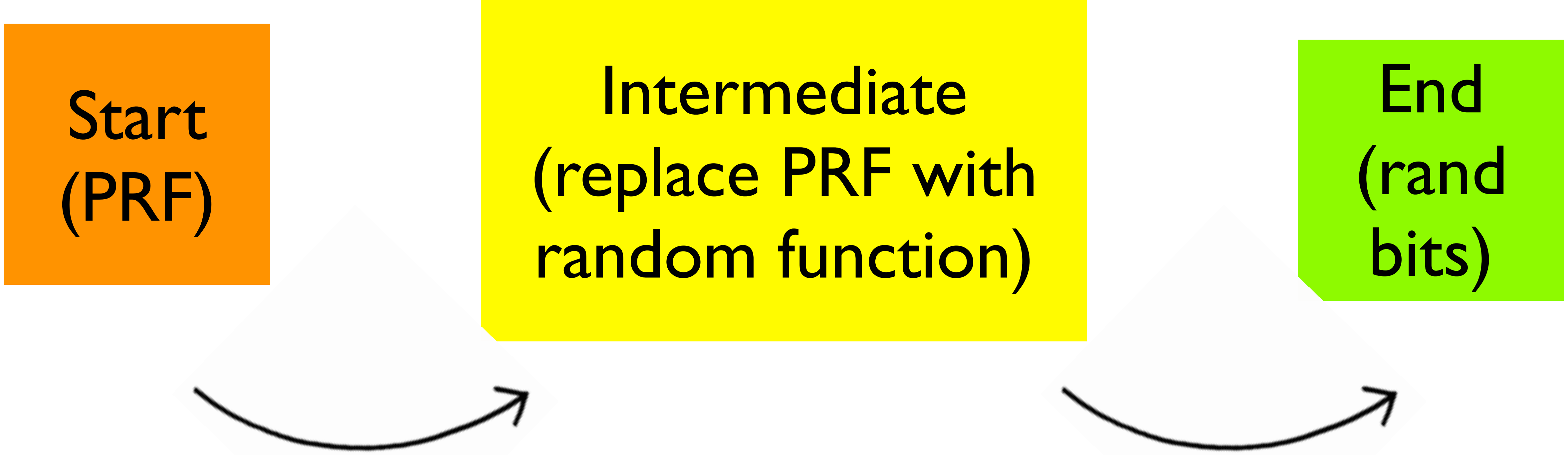
1. randomly sample s
2. give s to the adversary
3. adversary guesses whether s was randomly sampled or came from PRF

```
Variable RndR : Comp R.

Definition DRBG_G1 :
Comp Bool :=

  r <-$ RndR;
  A r.
```

Game-hopping proof



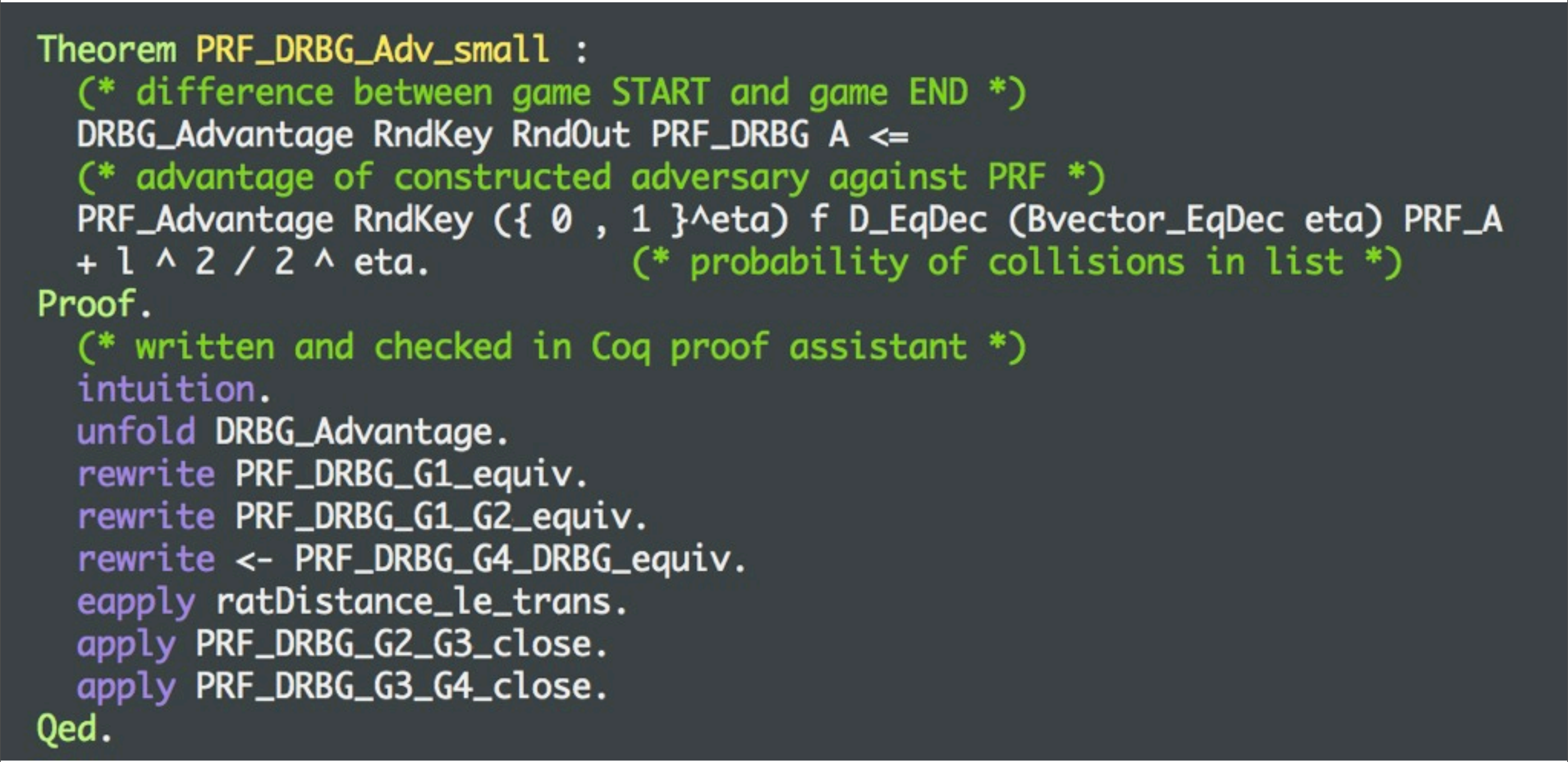
Start (PRF)

Intermediate (replace PRF with random function)

End (rand bits)

Difference:
PRF advantage

Difference:
prob. of collisions in a list



```
Theorem PRF_DRBG_Adv_small :
  (* difference between game START and game END *)
  DRBG_Advantage RndKey RndOut PRF_DRBG A <=
  (* advantage of constructed adversary against PRF *)
  PRF_Advantage RndKey ({ 0 , 1 }^eta) f D_EqDec (Bvector_EqDec eta) PRF_A
  + l ^ 2 / 2 ^ eta.          (* probability of collisions in list *)
Proof.
  (* written and checked in Coq proof assistant *)
  intuition.
  unfold DRBG_Advantage.
  rewrite PRF_DRBG_G1_equiv.
  rewrite PRF_DRBG_G1_G2_equiv.
  rewrite <- PRF_DRBG_G4_DRBG_equiv.
  eapply ratDistance_le_trans.
  apply PRF_DRBG_G2_G3_close.
  apply PRF_DRBG_G3_G4_close.
Qed.
```