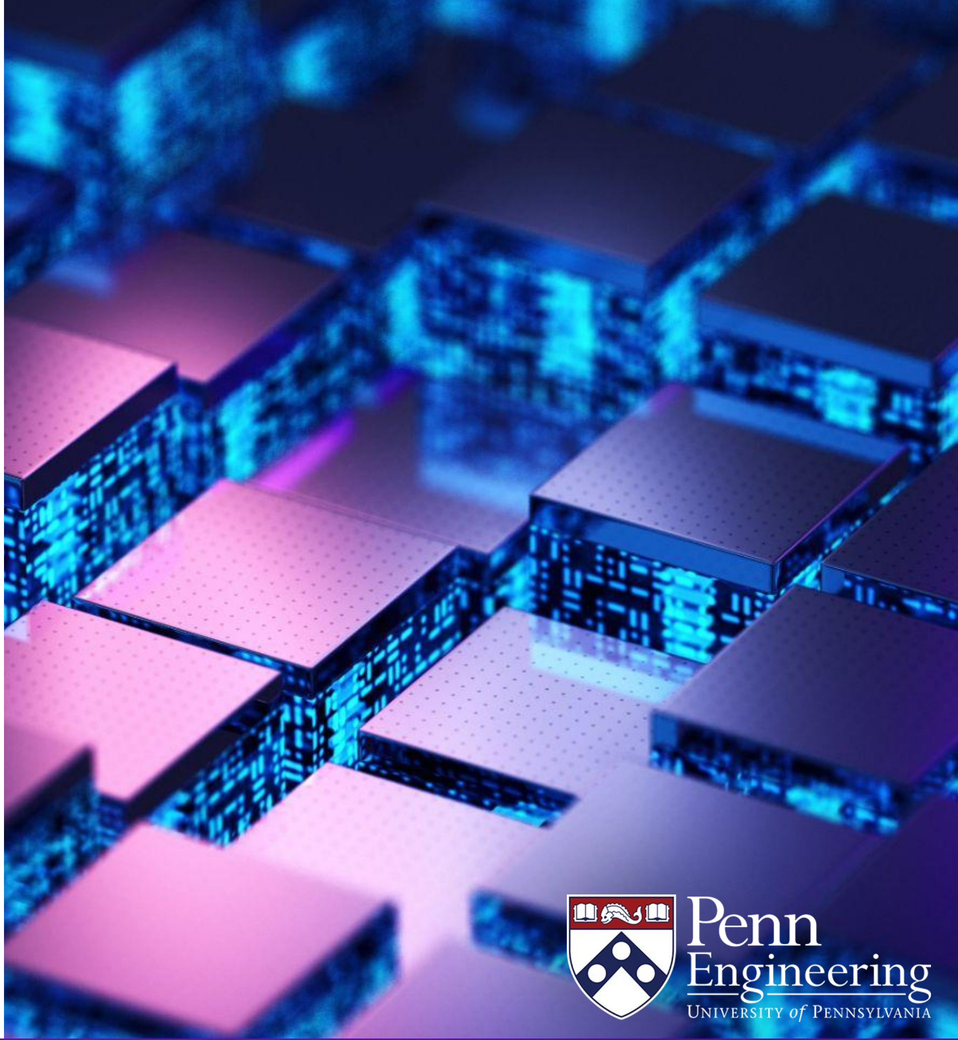
EAS 5830: BLOCKCHAINS

Verifiable Random Functions

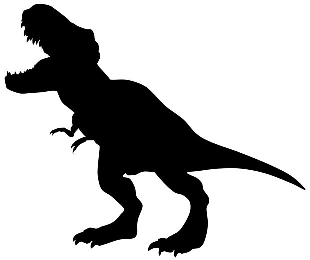
Dr. Brett Hemenway Falk



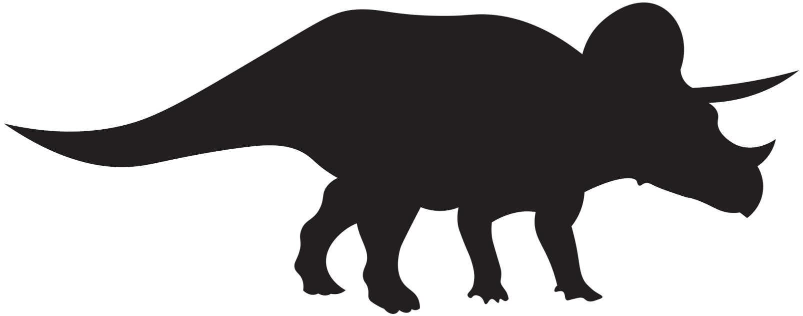
Random Functions

- Truly random function could be represented by a lookup table
- Consider 32-bit inputs 64-bit outputs
  - Table has 2^{32} entries, each requiring 64 bits
  - Requires 35 Gb to describe this function
- Consider 64-bit inputs 256-bit outputs
  - Table has 2^{64} entries, each requiring 256 bits
  - Requires 590 exabytes to describe this function

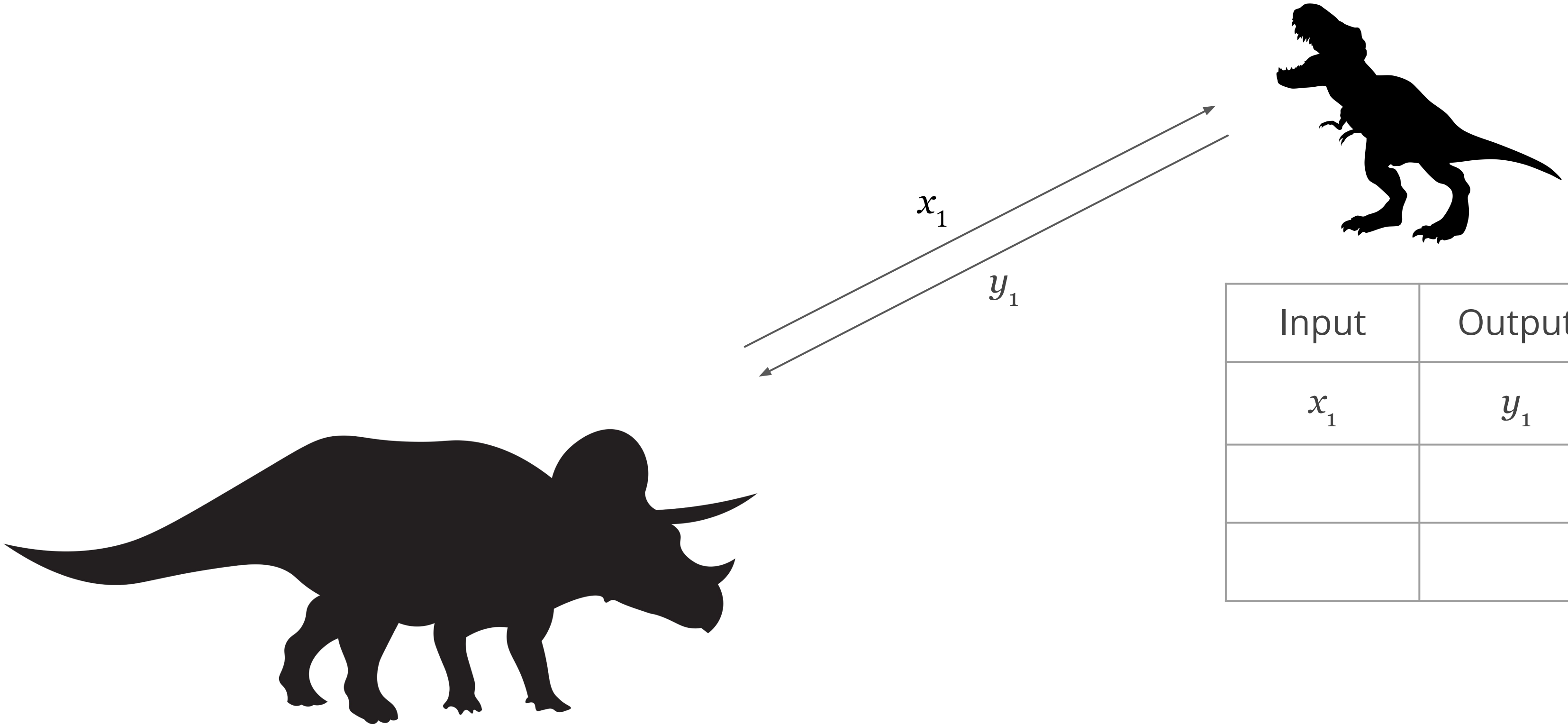
Stateful random functions



Input	Output

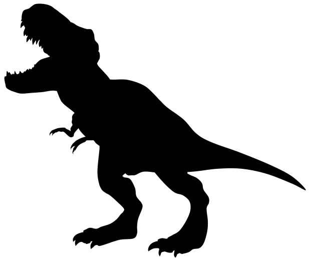


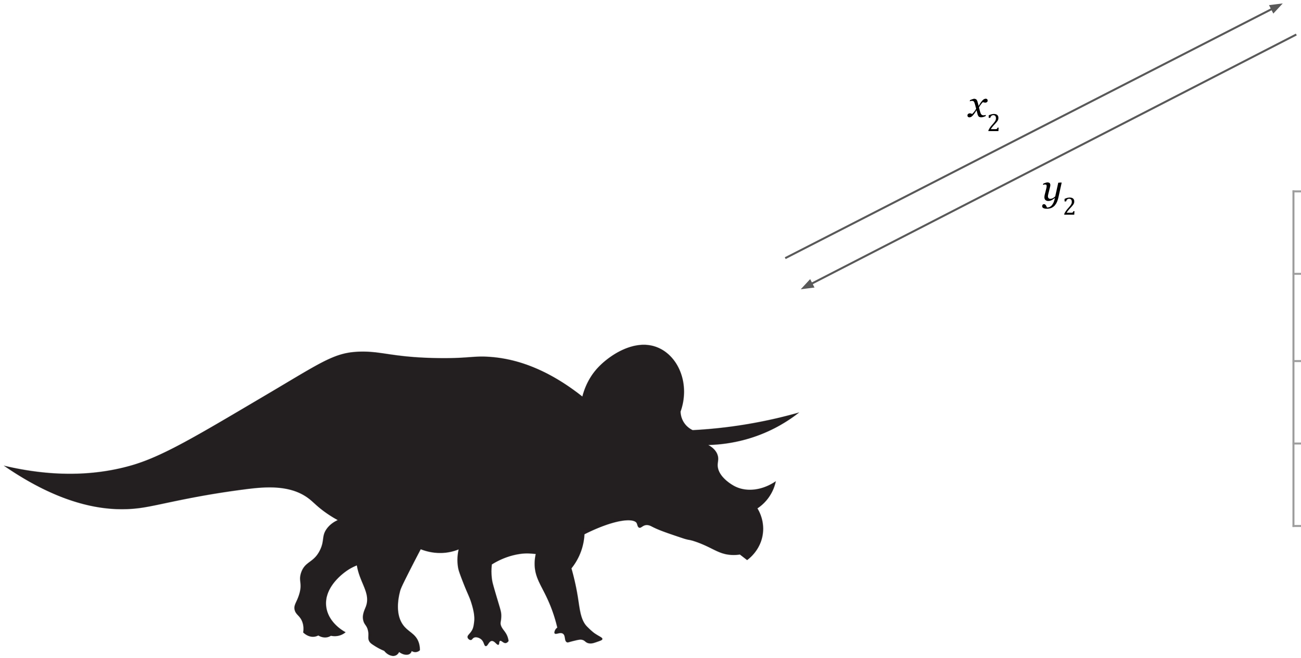
Stateful random functions



Input	Output
x_1	y_1

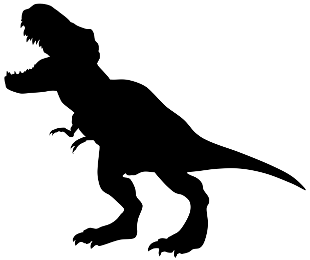
Stateful random functions

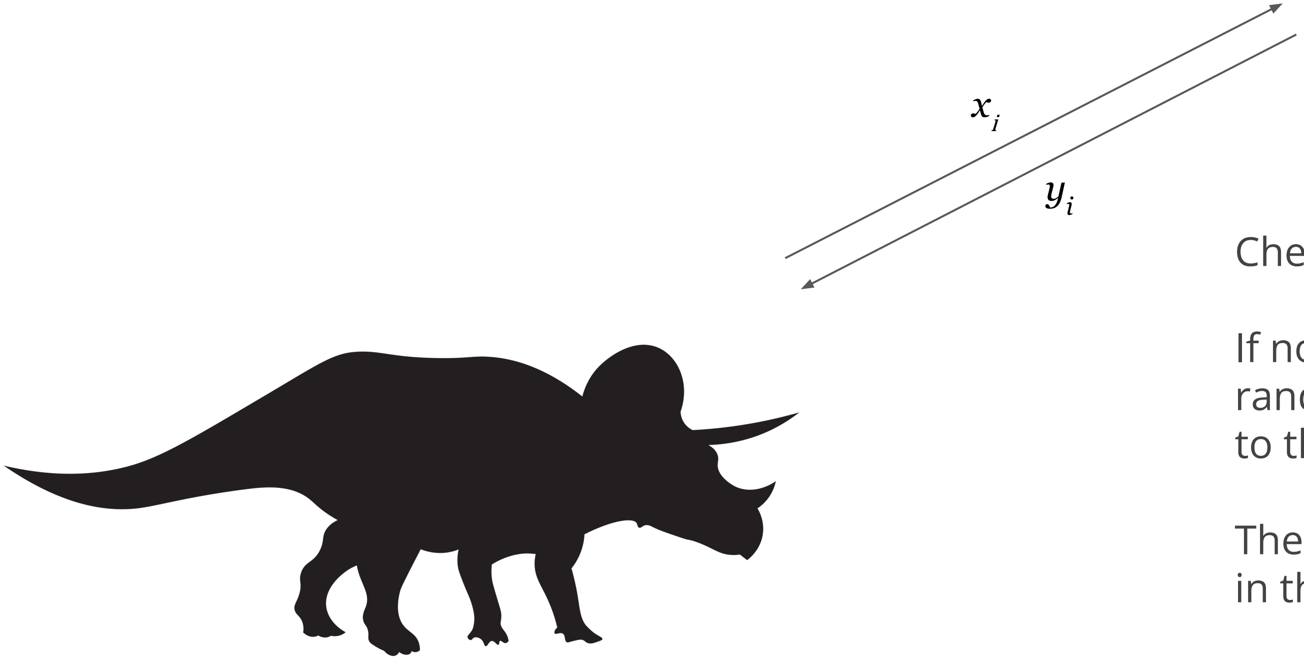




Input	Output
x_1	y_1
x_2	y_2

Stateful random functions



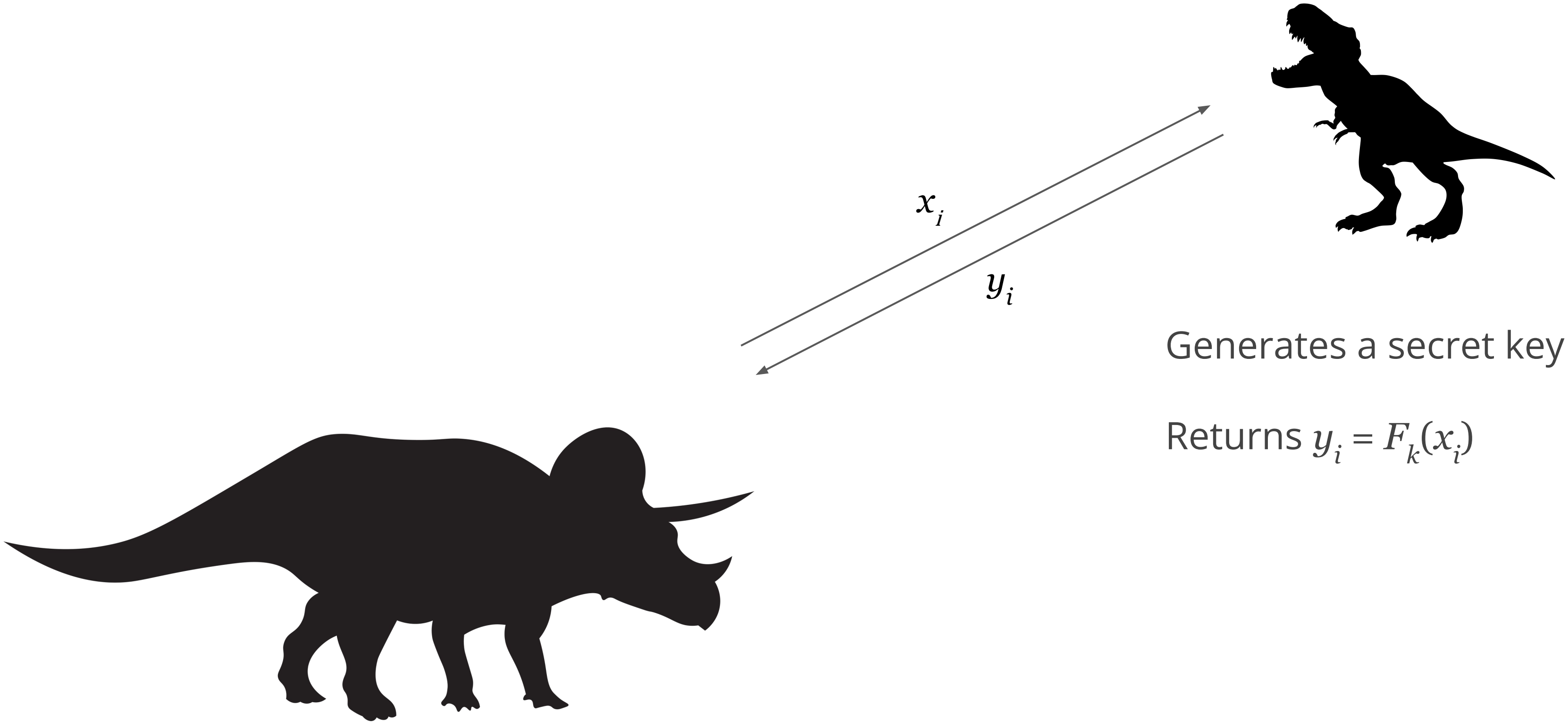


Check if x_i is in table

If not, generate a random y_i and add (x_i, y_i) to the table

Then return the value y_i in the table

Pseudorandom functions (PRFs)



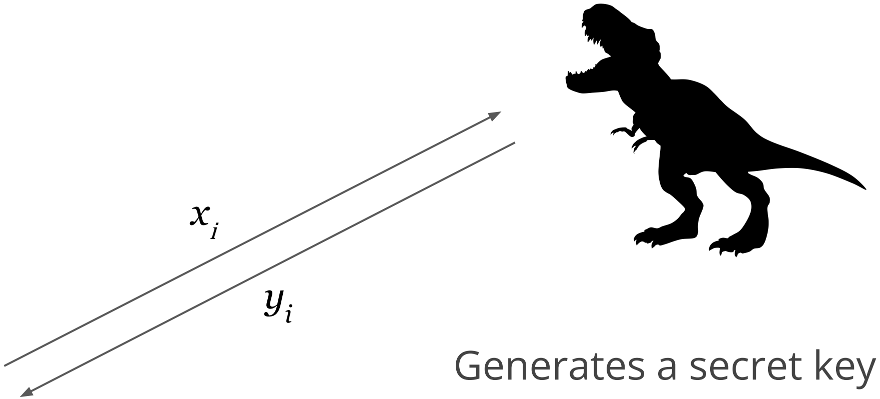
Generates a secret key k

Returns $y_i = F_k(x_i)$

Pseudorandom functions (PRFs)

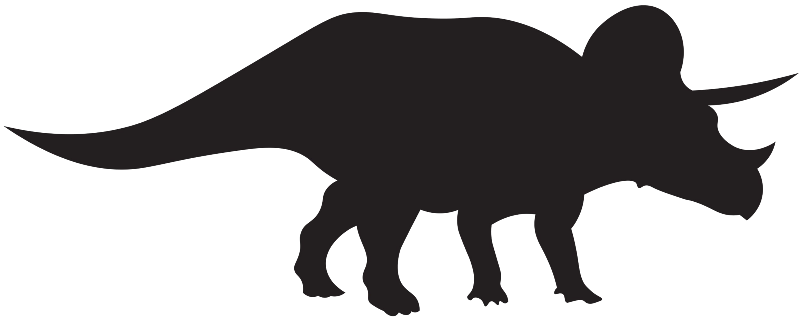
Security:

Triceratops can't tell whether it's interacting with a pseudorandom function or a stateful random function



Generates a secret key k

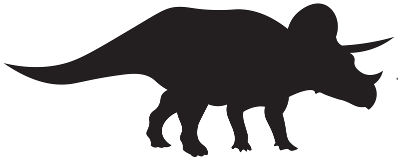
Returns $y_i = F_k(x_i)$



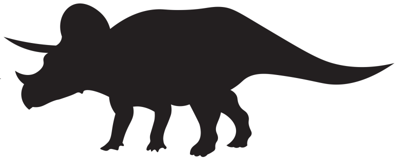
Examples of PRFs

- Fix a hash function, H (e.g. SHA-256)
- Choose a secret "nonce," k
- Input is a bit string x
- Output is $H(k \,||\, x)$

Application: Symmetric-key encryption



$(r, F_k(r) \text{ XOR } m)$



k

k

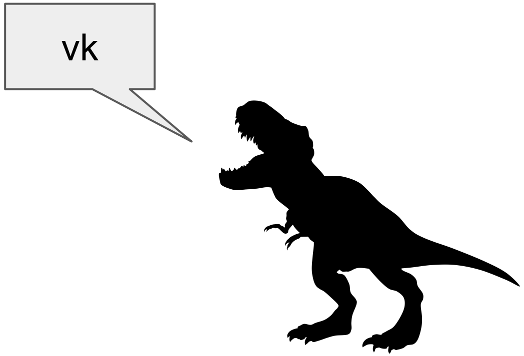
Receives (c_1, c_2)

Decrypts $m = F_k(c_1) \text{ XOR } c_2$

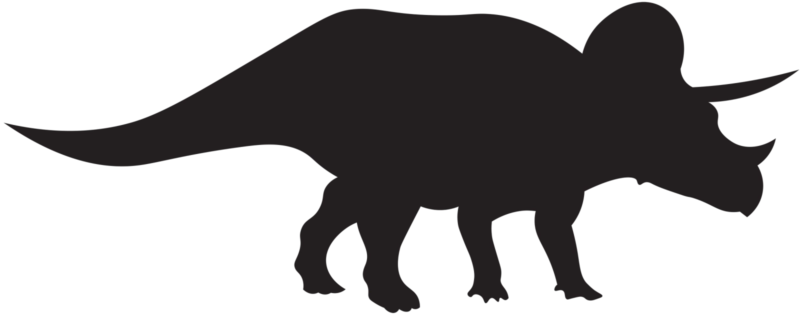
Verifiable Random Functions

- Public-key allows users to verify output came from PRF
- Public-key is a “commitment” to all future values of the function

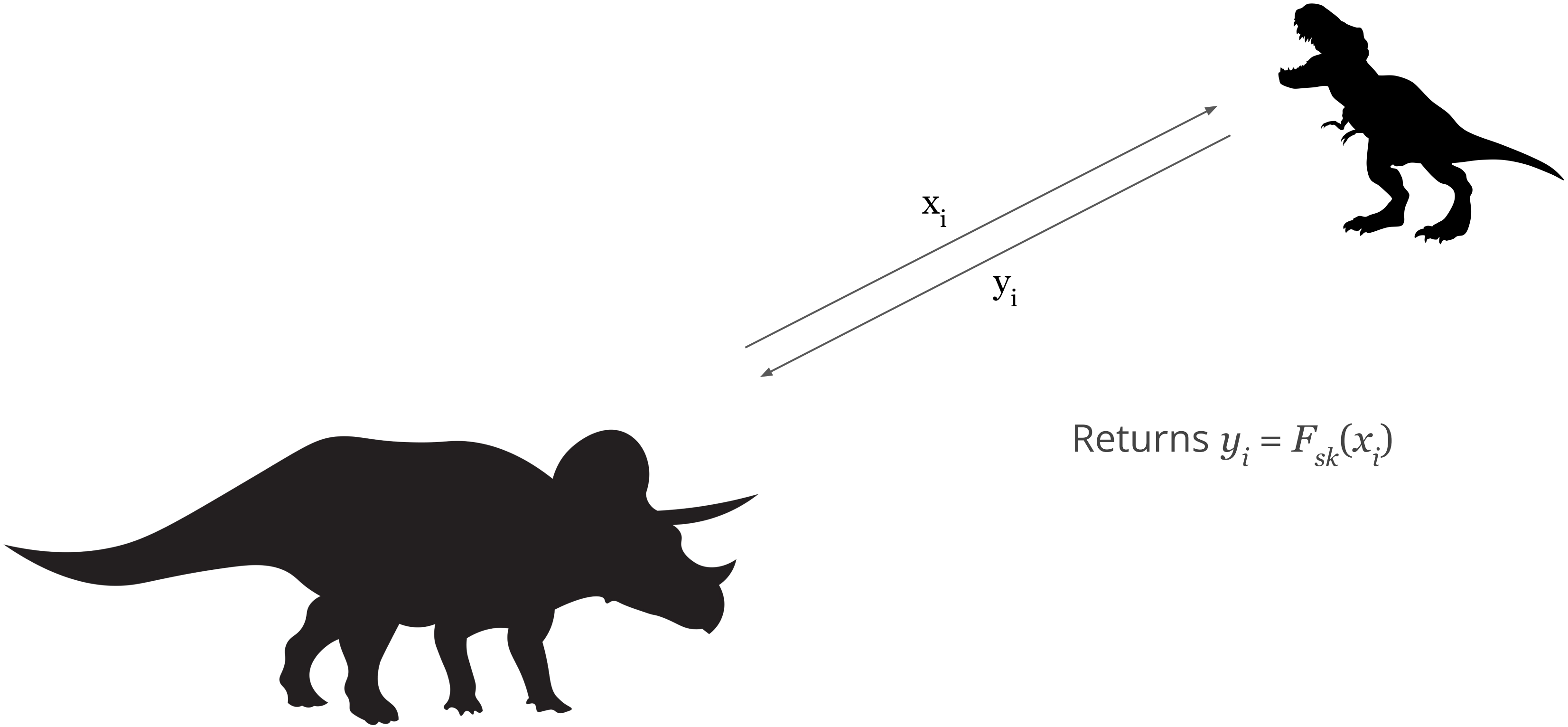
Verifiable random functions (VRFs)



Generates a key pair (*vk*, *sk)*

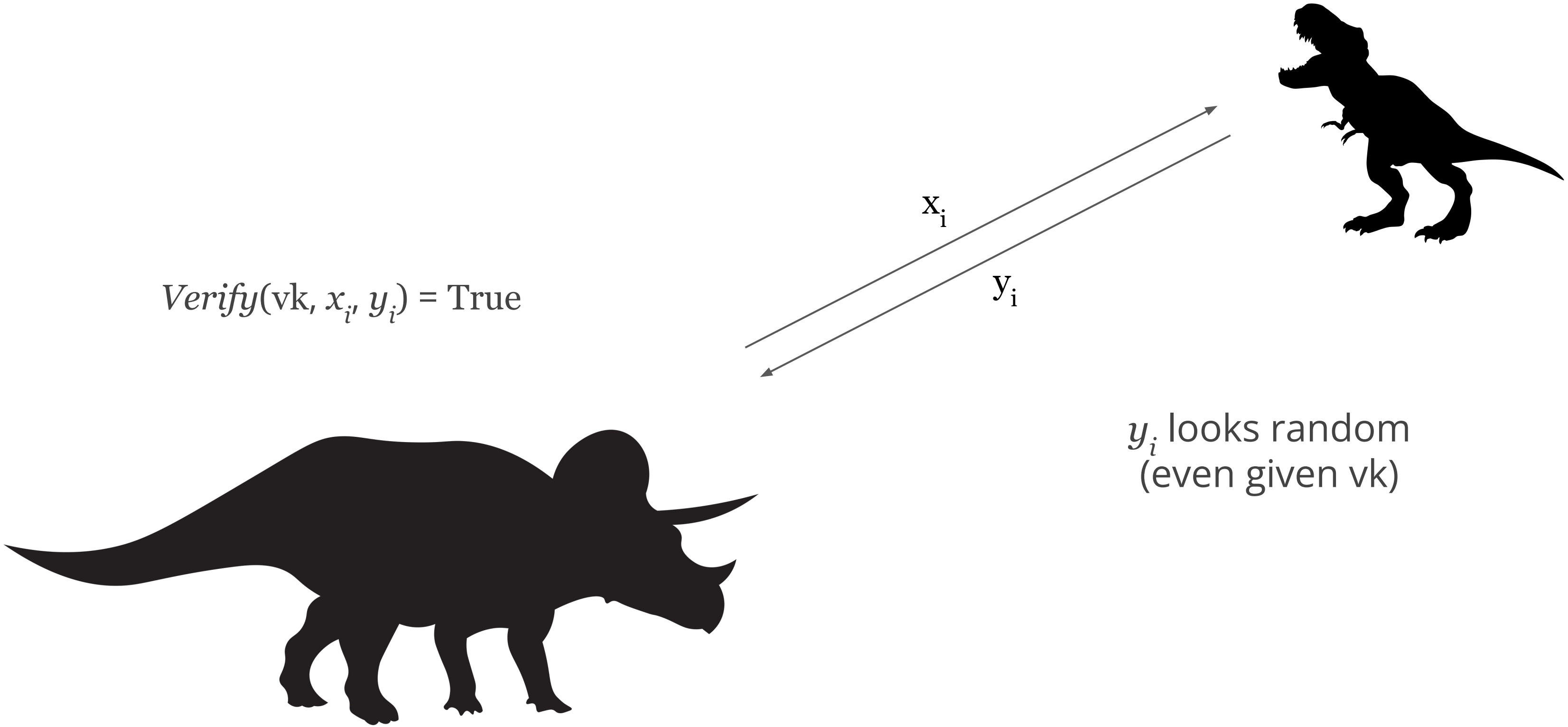


Verifiable random functions (VRFs)



Returns $y_i = F_{sk}(x_i)$

Verifiable random functions (VRFs)



Examples of VRFs

- Fix a hash function, H (e.g. SHA-256)
- Generate VK, SK pair for a digital signature scheme
  - VK will be verification key for VRF
- Input is a bit string x
- Output is $H(\text{sign}_{SK}(x))$

Examples of VRFs

- Fix a hash function, H (e.g. SHA-256)
- Generate VK, SK pair for a digital signature scheme
  - VK will be verification key for VRF
- Input is a bit string x
- Output is $\mathrm{H}(\mathrm{sign}_{SK}(x))$

- Signature scheme needs to be deterministic
- $\mathrm{sign}_{SK}(x)$ is unpredictable
  - "Predicting" is forging a signature
- Hashing it makes it uniformly random
- Given, x, $\mathrm{sign}_{SK}(x)$, VK you can verify that $\mathrm{H}(\mathrm{sign}_{SK}(x))$ was computed correctly

Decentralized lotteries using VRFs

- Every participant generates a (vk,sk) pair for a VRF
- Participants broadcast their vks
- Participants agree on a random string r
  - E.g. from the NIST randomness beacon
- Each player broadcasts their “lottery ticket” $F_{sk}(r)$
- Player with lowest value wins
- Everyone can use vks to check that all players generated their lottery tickets correctly (i.e., with the VRF)

VRFs on the Blockchain

- Selecting Block Producers
  - Ethereum RANDAO
  - Algorand
  - Cardano
  - Polkadot
- Getting randomness into smart-contracts
  - Chainlink VRFs