Functional Encryption: Deterministic to Randomized Functions from Simple Assumptions

Shashank Agrawal and David J. Wu

The Olympic Games

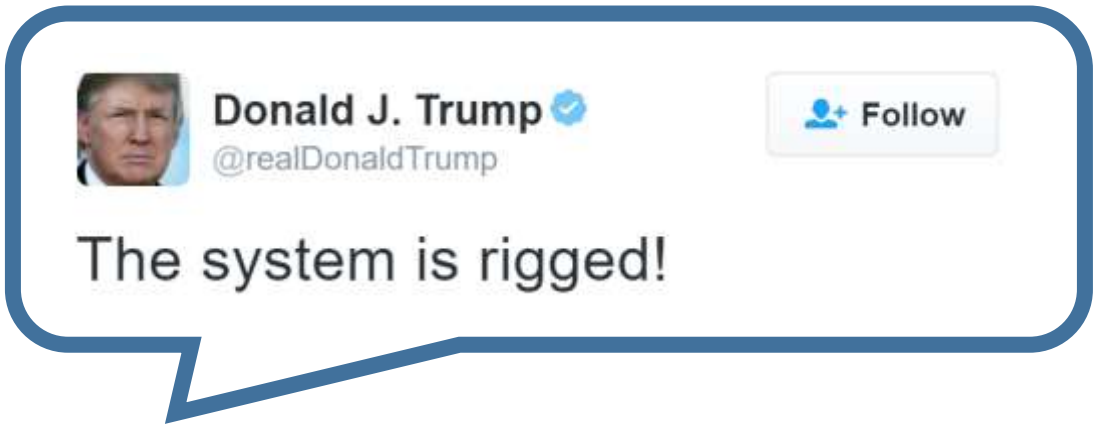
Vinicius, Tom, and friends are competing at the Olympic games…

But…

Vinicius, Tom, and friends are competing at the Olympic games…

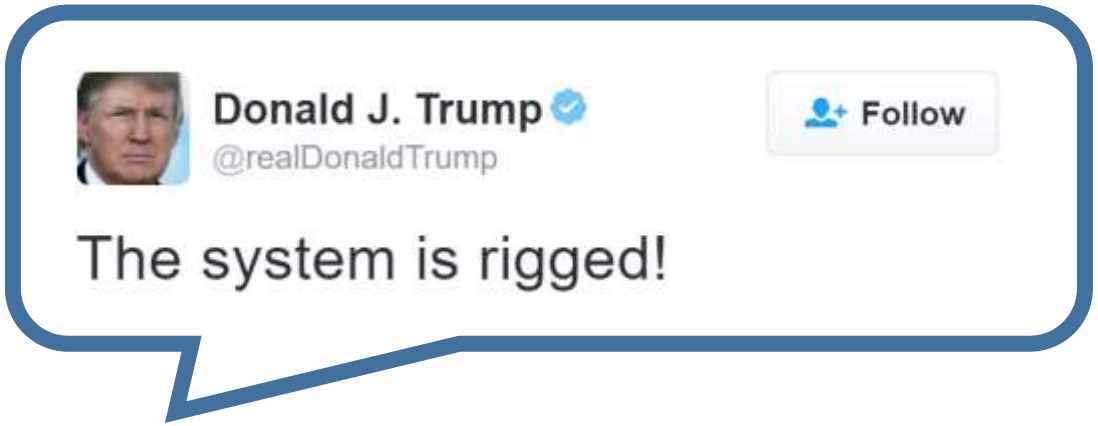
But…

Vinicius, Tom, and friends are competing at the Olympic games…



But… is it Rigged?

Vinicius, Tom, and friends are competing at the Olympic games…



Let’s Audit the Records!

Date: 01/09/2011

Date: 03/15/2012

Date: 06/11/2012

Weight: xxx
Blood Pressure: xxx
Performance: xxx
…

Solution: take each competitor’s health records, choose a couple at random and see if there is anything unusual

Let’s Audit the Records!

Date: 01/09/2011

Privacy?

ompetitor’s health records and
eck if there is anything unusual

Let’s Audit the Records!

Let’s use CRYPTO!
I’ll just encrypt
my records!

Date: 01/09/2011

Date: 03/15/2012

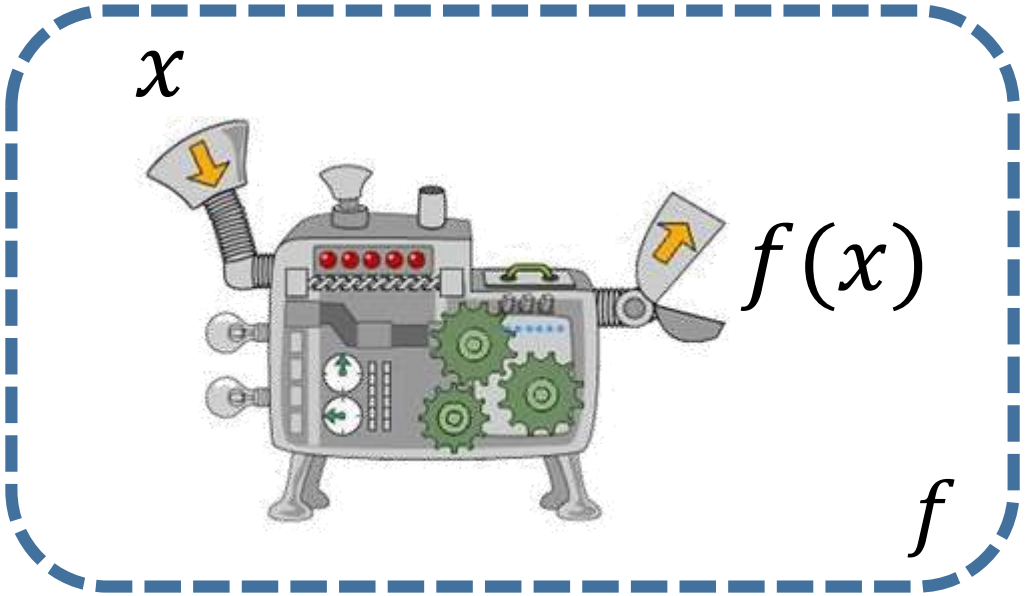
Date: 06/11/2012

Weight: xxx
Blood Pressure: xxx
Performance: xxx

But How Do We Audit?

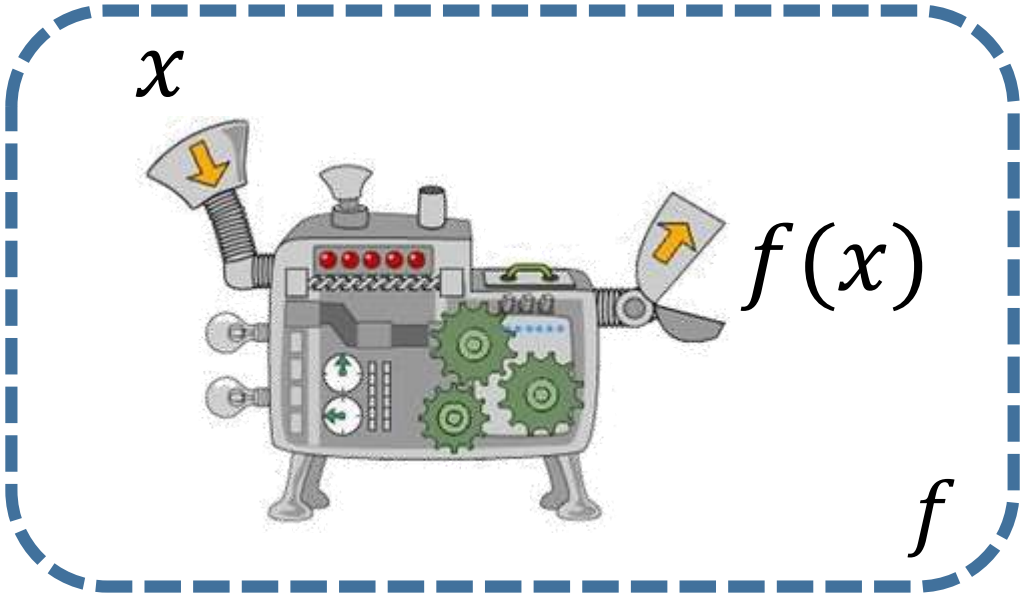
Abstract problem: sample a random record from an encrypted database

Functional Encryption



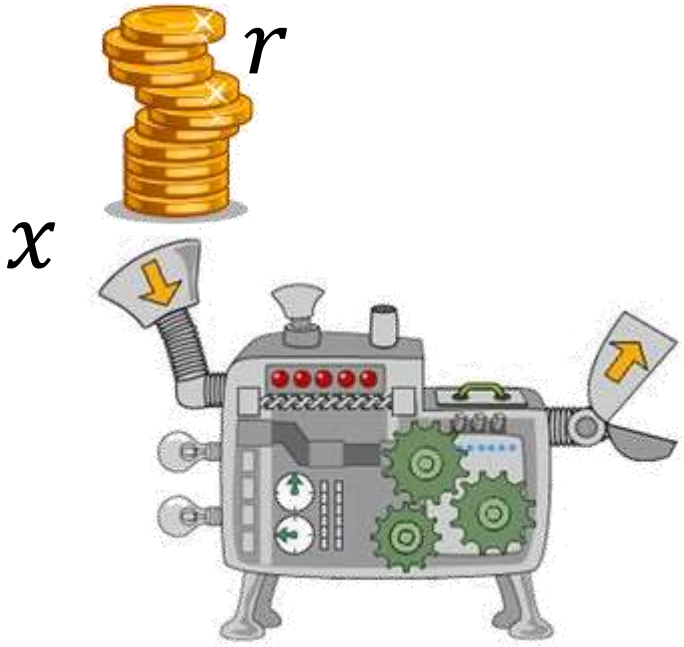
Keys associated with functions f

Functional Encryption



Keys associated with functions f

Randomized Functional Encryption [GJKS15]



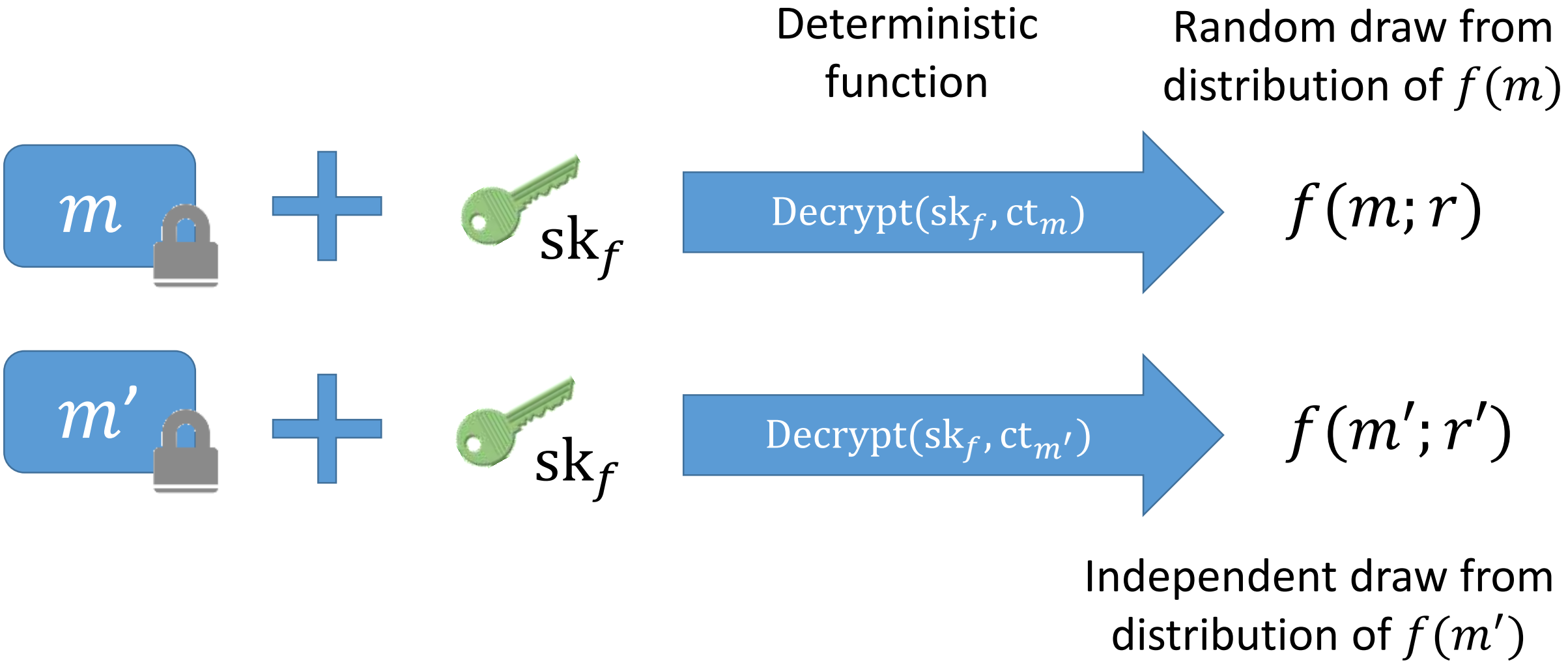
$f(x\,;r)$

But what if f is *randomized*?

Auditing an encrypted database reduces to sampling a **random** element.

Randomized Functional Encryption

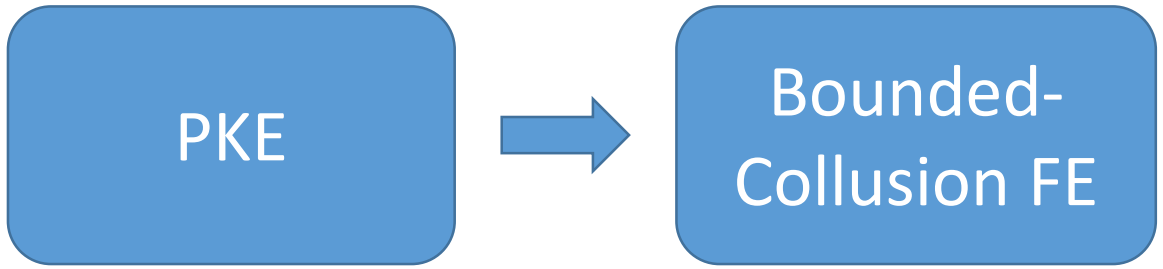
Randomized Functional Encryption



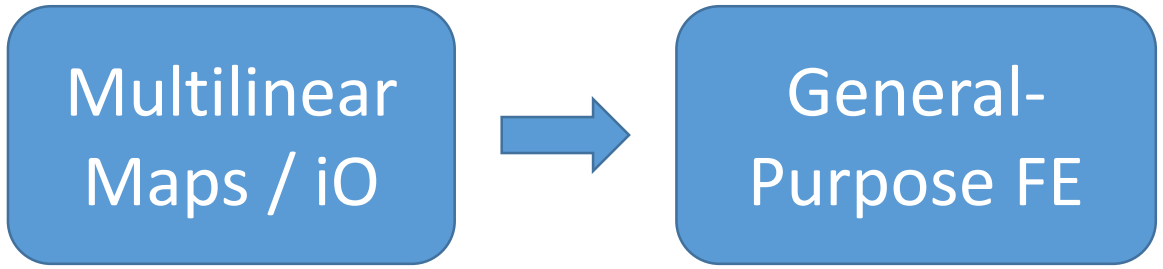
The State of (Public-Key) Functional Encryption

deterministic functionalities

[SS10, GVW12, …]

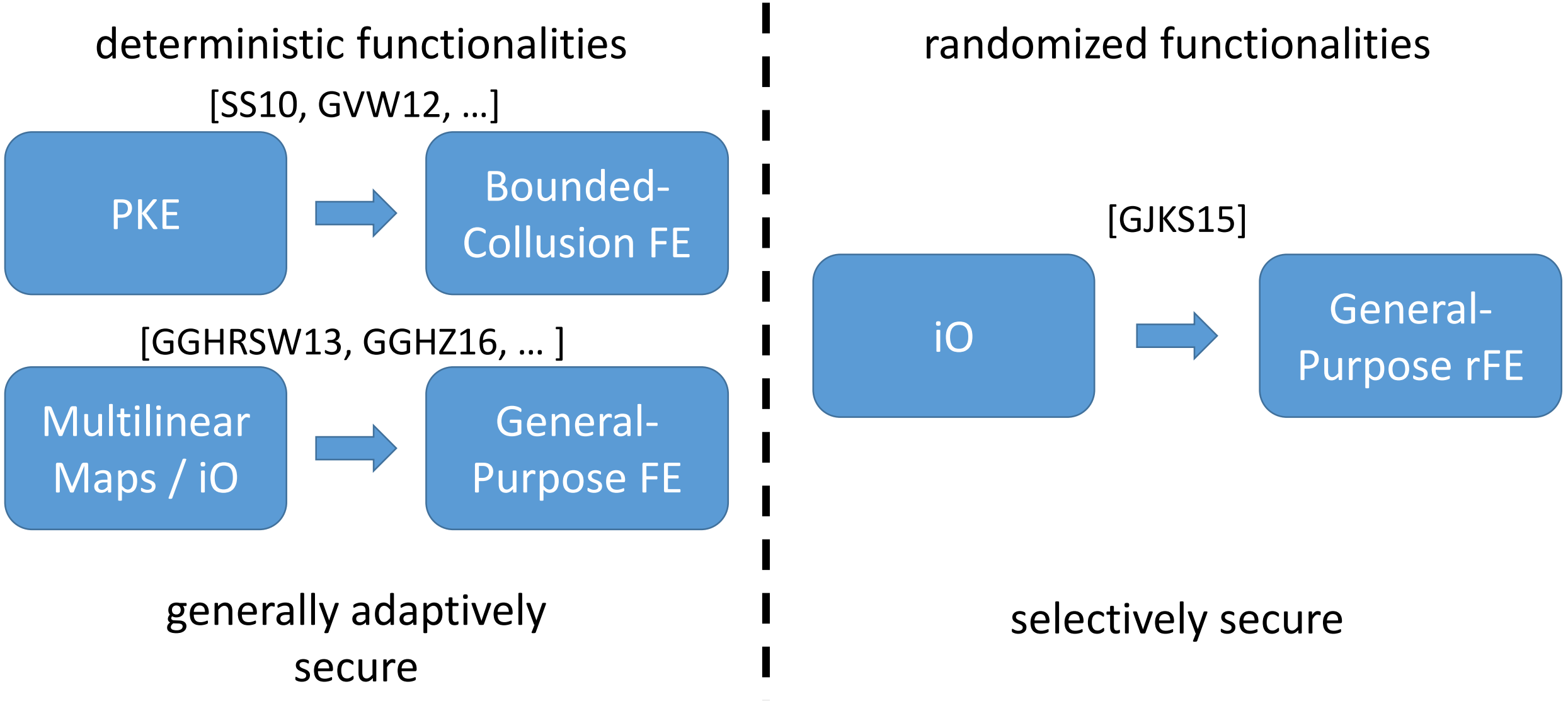


[GGHRSW13, GGHZ16, …]



generally adaptively secure

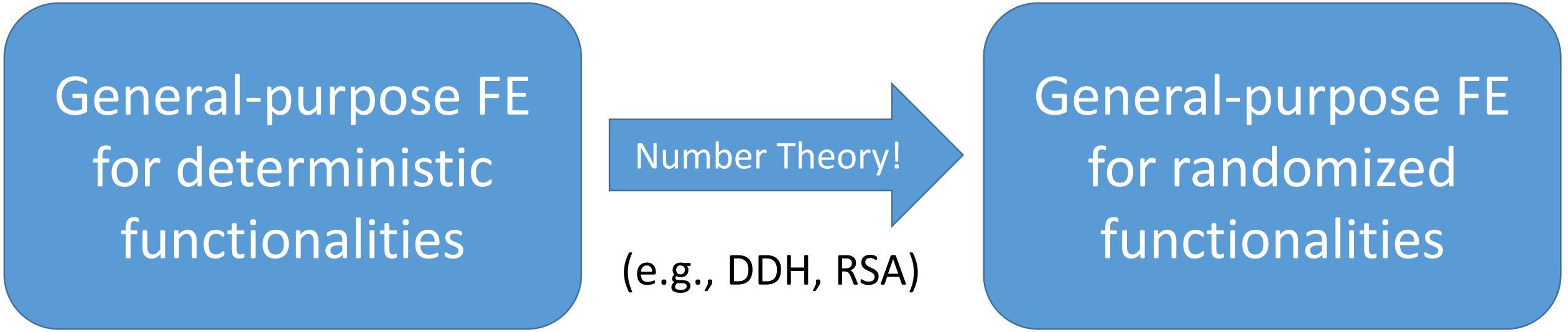
The State of (Public-Key) Functional Encryption



The State of (Public-Key) Functional Encryption

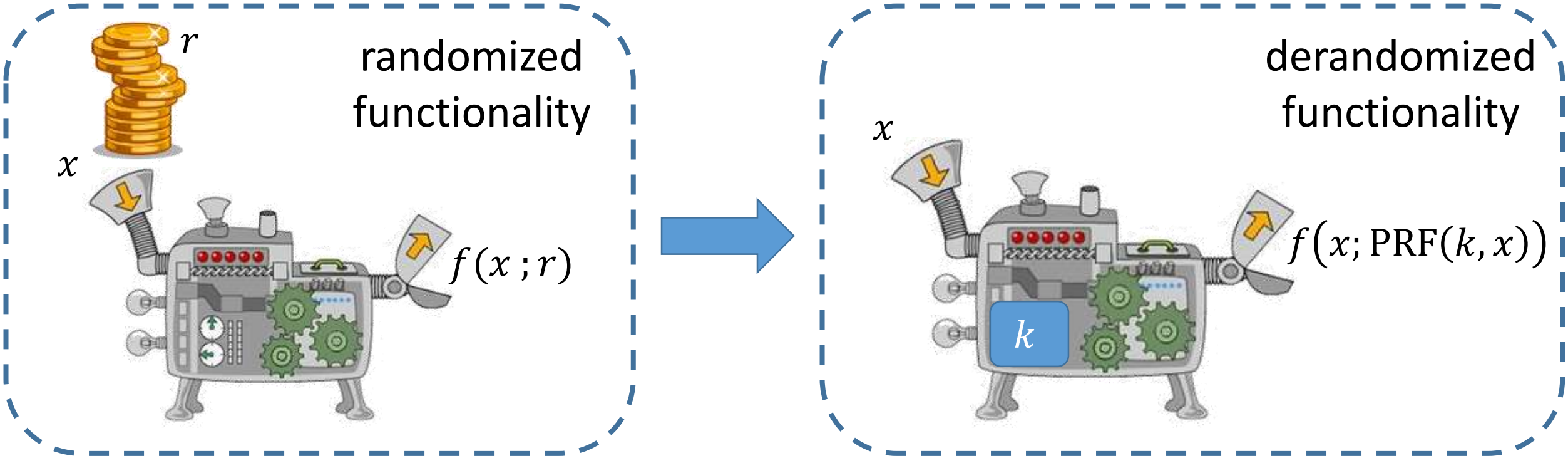
Does extending FE to support randomized functionalities require much stronger tools?

Our Compiler



Implication: *Randomized FE is not much more difficult to construct than standard FE.*

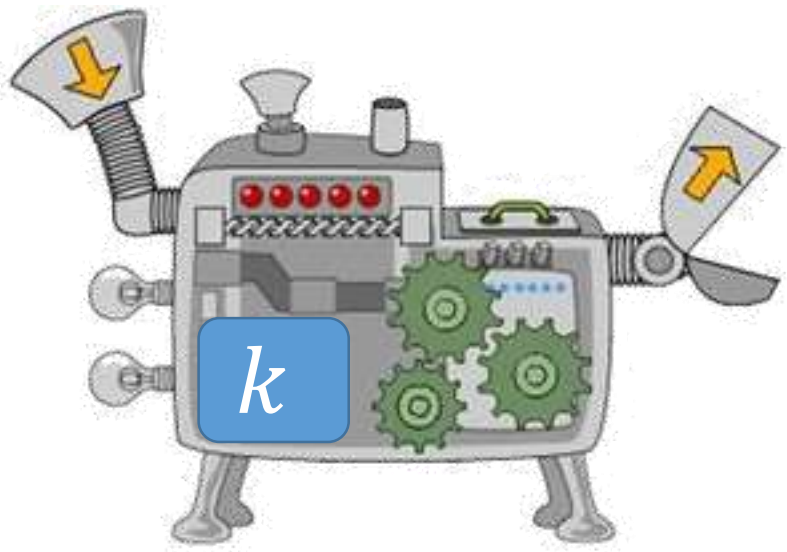
First Attempt



Starting point: Construct "derandomized function" where randomness for f derived from outputs of a PRF

First Attempt

x

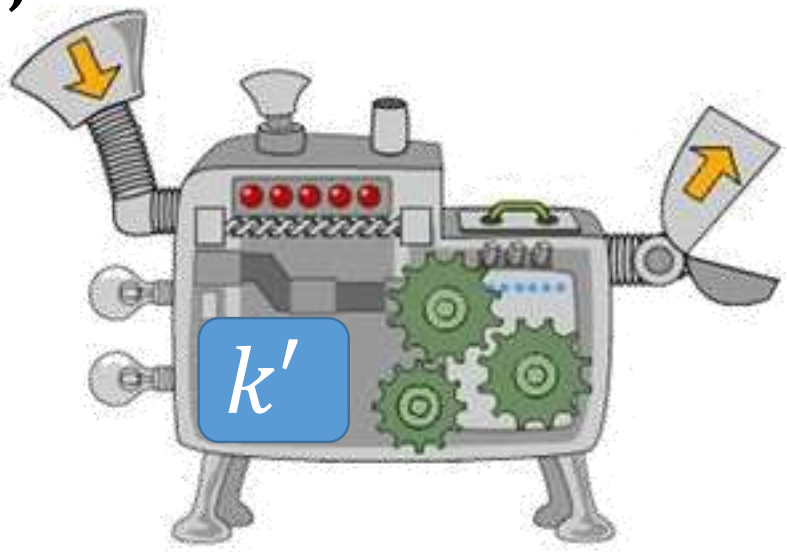


$f\left(x; \mathrm{PRF}(k, x)\right)$

But… function keys do not hide the function…

Second Attempt

k, x



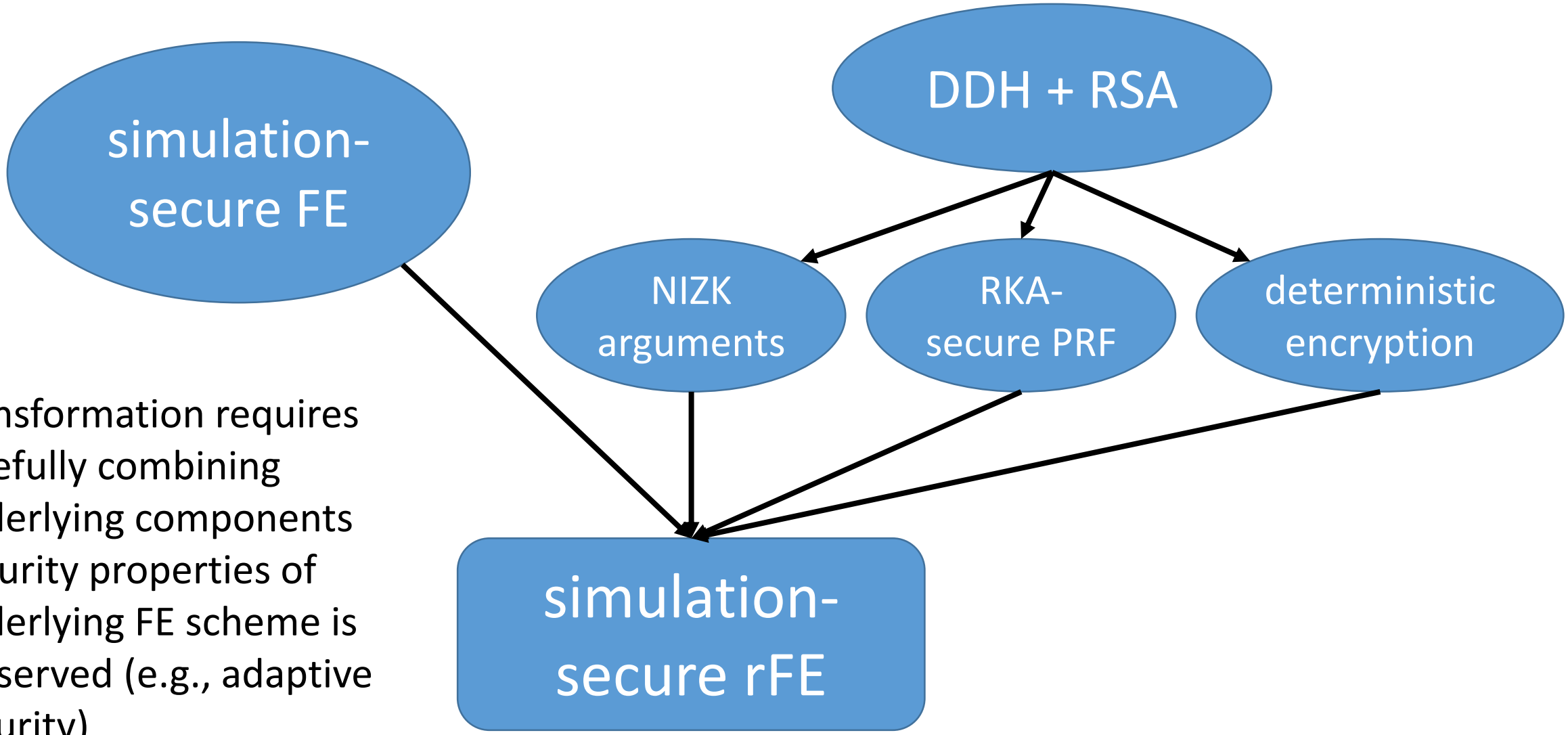
$$f\left(x; \mathrm{PRF}(k \oplus k', x)\right)$$

So split (secret share) the PRF key!

But Remember, it's Rigged…

Encrypter can misbehave! Can influence key-share for PRF and encryption randomness to induce bad distributions

Our Transformation



- Transformation requires carefully combining underlying components
- Security properties of underlying FE scheme is preserved (e.g., adaptive security)

Thanks!

Check out our paper on ePrint!

http://eprint.iacr.org/2016/482