NON-INTERACTIVE VERIFIABLE COMPUTATION

Presentation by Mohd Arslan Sheikh

IIITG

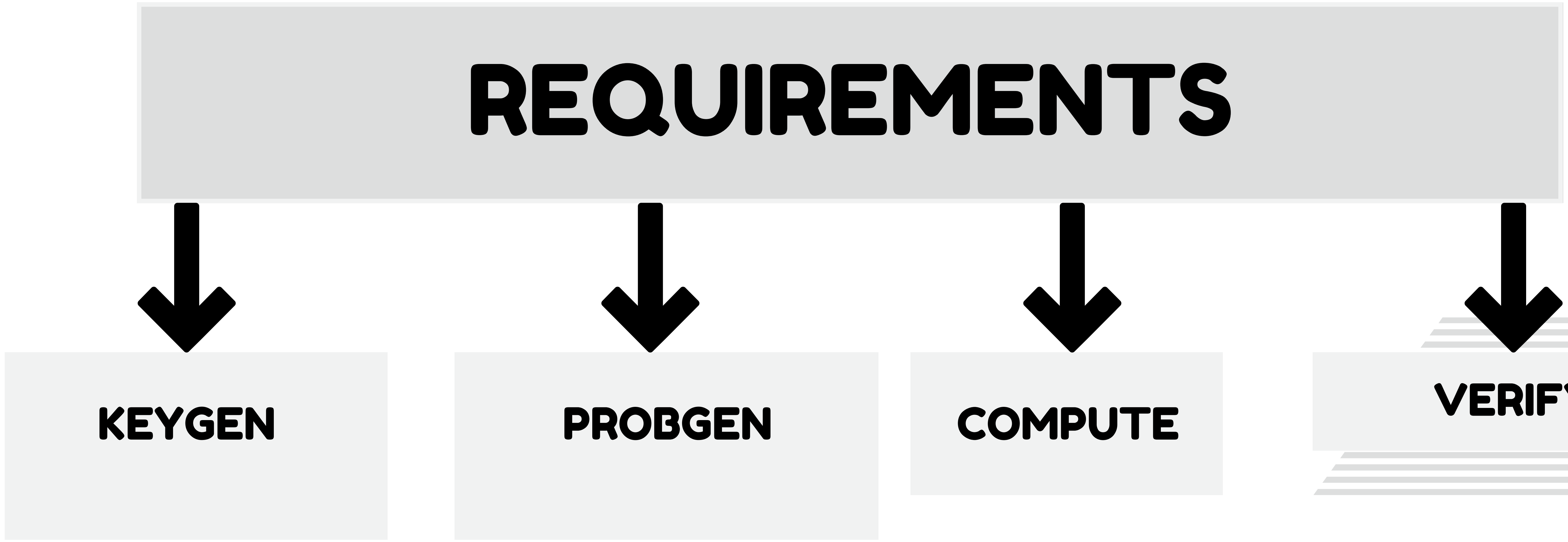
INTRODUCTION

Non-Interactive Verifiable Computation is a powerful tool that enables a client to outsource computation to an untrusted worker without the need for interactive communication between the two parties

In a non-interactive verifiable computation scheme, the client provides the worker with an encoded version of the computation task, along with a proof of correctness that can be verified by the client without requiring any further interaction with the worker

PROBLEM

At a high-level, a verifiable computation scheme is a two-party protocol in which a client chooses a function and then provides an encoding of the function and inputs to the function to a worker. The worker is expected to evaluate the function on the input and respond with the output. The client then verifies that the output provided by the worker is indeed the output of the function computed on the input provided


REQUIREMENTS
KEYGEN
PROBGEN
COMPUTE
VERIFY

KEYGEN(F, Λ) → (PK, SK)

- This requirement involves generating a public key (PK) and a matching secret key (SK) based on the security parameter λ. The public key encodes the target function F, which is used by the worker to compute F. The secret key is kept private by the client. The purpose of this requirement is to ensure that the worker can compute F on the input without revealing any information about the function or the input to the client or any other party

PROBGEN SK(X)

- This requirement involves using the secret key (SK) to encode the function input x as a public value σx and a secret value τx. The public value σx is given to the worker to compute with, and the secret value τx is kept private by the client. The purpose of this requirement is to ensure that the worker can compute F on the input without learning anything about the input or the function other than what is necessary to compute the output.

COMPUTE PK

- This requirement involves using the client's public key and the encoded input (σx) to compute an encoded version of the function's output (σy). The worker provides σy to the client as the result of the computation. The purpose of this requirement is to enable the worker to compute F on the input and provide a verifiable proof of the correctness of the result to the client

VERIFY SK

- This requirement involves using the secret key (SK) and the secret decoding τx to verify the correctness of the result provided by the worker. The verification algorithm converts the worker's encoded output (σy) into the output of the function, e.g., $y = F(x)$, or outputs $\perp$ indicating that σy does not represent the valid output of F on x. The purpose of this requirement is to enable the client to verify the correctness and validity of the result provided by the worker without needing to recompute the result from scratch

HOMOMORPHIC

Homomorphic encryption is a type of encryption that allows computations to be performed on ciphertext, without requiring access to the plaintext. In other words, it enables data to be processed while it is still encrypted, without ever needing to decrypt it

THERE ARE TWO TYPES OF HOMOMORPHIC ENCRYPTION

FHE : In FHE, arbitrary computations can be performed on encrypted data. This means that complex computations can be performed on encrypted data without ever having to decrypt it.

PHE : In PHE, only certain types of computations can be performed on encrypted data. For example, a PHE system might allow for addition or multiplication of encrypted numbers, but not both.

IMPLEMENTATION

- PREPROCESSING
- INPUT PREPARATION
- OUTPUT COMPUTATION AND VERIFICATION.

INPUT AND OUTPUT PRIVACY

While the basic definition of a verifiable computation protects the integrity of the computation, it is also desirable that the scheme protect the secrecy of the input given to the worker(s). We define input privacy based on a typical indistinguishability argument that guarantees that no information about the inputs is leaked

CONCLUSION

n this work, we introduced the notion of Verifiable Computation as a natural formulation for the increasingly common phenomenon of outsourcing computational tasks to untrusted workers. We describe a scheme that combines Yao’s Garbled Circuits with a fully-homomorphic. encryption scheme to provide extremely efficient outsourcing, even in the presence of an adaptive adversary. As an additional benefit, our scheme maintains the privacy of the client’s inputs and outputs.

THANK YOU