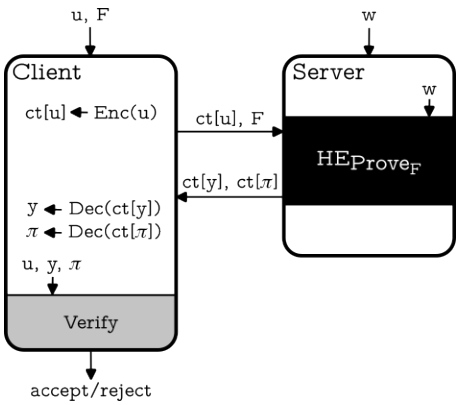
Computing Proof Systems using Homomorphic Encryption



Verifiable Computation (VC) is a cryptographic technique that allows you to verify other people's computations without having to redo them yourself. Let's say you want to perform a computation F on some input u but don't posses the required computational resources so you delegate the computation to some server (e.g. a cloud computing platform). The server computes the output y and then uses a VC scheme to generate a "proof" $\pi \leftarrow \mathsf{Prove}_F(u, y)$. The proof π can be used to verify whether the provided y is actually equal to $F(u)$. Naturally a VC scheme should have the property that verifying π is less costly than computing F, which is called succinctness.

By using this protocol you would not have to trust the server to actually perform the computation. However, your privacy is not well protected since input u and output y are visible. Luckily there is another cryptographic technique that provides this functionality named Homomorphic Encryption (HE). It allows you to encrypt u to $\mathsf{Enc}(u)$ in a way that still allows for computation. So now the server could compute $\mathsf{Enc}(y) = F(\mathsf{Enc}(u))$ without the ability to see u and y (we say it blindly computes F). However, HE schemes do not natively provide a guarantee that the server honestly computed the requested computation F.

Ideally we would have a protocol that has the properties of VC and HE combined. One way to achieve this is to use an HE scheme on top of a VC scheme, i.e. to blindly compute Prove_F inside the HE scheme. This is easier said than done. Both are already advanced cryptographic primitives with many moving parts and relatively bad performance. Combining them does not help in this regard. Recent work proposes new protocols that are estimated to drastically improve performance. In this thesis, the student will either: 1) try to discover (subcomponents of) VC protocols that lend themselves to blind computation, i.e. homomorphic computation using an HE scheme, 2) compare the different approaches both asymptotically and by means of software implementation and/or 3) design/implement a blind proof for a specific application F. The student will have to become familiar with complex cryptographic protocols and new open-source software libraries.

Practicalities

Promotor:	Frederik Vercauteren		
Daily supervisor:	Jannik Spiessens	jannik.spiessens@esat.kuleuven.be	office B01.15
	Valerio Ardizio	valerio.ardizio@esat.kuleuven.be	office B01.29
Nature of the work:	40% literature, 20% theoretical work, 40% software		
Number of students:	1		

 © 2026 – 2027 COSIC