Commodity computing devices, such as laptops and smartphones, are an essential part of today's society. We routinely rely on them for both our professional and personal lives. Furthermore, many of the client applications they run are security critical. The trusted computing base of these applications includes the devices’ underlying system software (operating system, hypervisor and firmware), which is large, complex, and vulnerable to compromise. To mitigate this risk, Trusted Execution Environments, TEEs, offer a security primitive which protects the confidentiality and integrity of user applications’ code and data, against the device’s underlying system software which can be malicious.

Online user authentication is a prominent example of user applications which can benefit from the security guarantees of TEEs. While different realizations of hardware TEEs are commonly deployed within today's commodity devices, user authentication applications do not use their services. Instead, TEEs are heavily used by few premium Service Providers, SPs, such as original equipment manufacturers. In fact, TEEs have been primarily designed to meet the requirements of such premium SPs, and have consequently under-prioritized the needs of end-users and application developers. As a result, TEEs lack important functionalities, such as secure input/output channels to end-users.

This thesis shows that we can use available commodity TEEs as primitives to build systems which meet the specific security requirements for user authentication developers and their end-users.

We first study the security of passwords and Fast Identity online (FIDO) as two prominent user authentication modalities. We then present TrustUI, a solution that enables secure input/output channels from end-users to online service providers. TrustUI uses Intel SGX and personal security devices that are based on secure elements as TEE primitives. We finally present SMMDecoy, a new architecture which leverages System Management Mode and security by deception techniques, to detect firmware keyloggers that can compromise the confidentiality of the keyboard’s user interface.