High-Assurance Post-Quantum Cryptography and Distributed Secure Computation (work plan)

Henrique José Faria

Abstract. Computer Aided Cryptography (CAC) aims to develop tools for the analysis and implementation of cryptographic protocols, including specification and implementation languages, compilers and formal verification tools. The formal verification dimension opens the way for High-Assurance Cryptographic Software. The objective of this work is to progress beyond the state of the art at both the foundational and applied research levels by developing new extensions to the Jasmin tool-chain and demonstrating its applicability to concrete examples that can have real-world impact.
A recent SoK paper [2] published at IEEE S&P 2021 summarises challenges in this area; two of these challenges serve as our main motivation: 1) removing scalability bottlenecks in the analysis of distributed cryptographic protocols and 2) enabling the transition to post-quantum cryptography. This proposal will focus on concrete cryptographic constructions and implementations of post-quantum cryptography (PQC) and secure distributed computation that can serve as inputs to standardisation processes or as contributions to widely deployed cryptographic libraries and protocol software stacks.

Keywords: High-Assurance Cryptographic Software · Program Verification · Post-Quantum Cryptography · Secure Distributed Computation.

State of the art

Formal methods progressed to an impressive level of maturity, and several tools for systematically preventing entire classes of bugs in cryptographic software now exist. Frameworks like F* [FSTR], EasyCrypt [EC], and Coq [FIAT] are used to verify high-performance cryptographic code written in C and assembly. Tools like CryptoVerif [CVRF] and EasyCrypt are used to verify the correctness of crypto security proofs. In practice, protocol stacks for TLS [TLS], Signal [SIGN], Key Management [KMS] and crypto standards [SHA3,HACL] have been verified with these tools.

This project will be focused on a tool-chain that includes EasyCrypt, and the Jasmin language. EasyCrypt is an interactive proof assistant for the verification of cryptographic security proofs. It adopts the code-based approach, where primitives, security goals and hardness assumptions are expressed as probabilistic programs. EasyCrypt offers logics to reason about programs written in an probabilistic imperative language, as well as establishing relations between two program executions. The EasyCrypt logics have been successfully used to machine-check a number of relevant cryptographic security proofs and implementations [SHA3,EUC,MASK,DIFP]. Jasmin [JASM] is a programming language designed to allow "assembly in the head" (a mixture of high-level and low-level, platform-specific, programming); it is supported by a formally verified (certified in Coq), predictable, compiler which empowers programmers to write highly efficient fine-tuned code. The generated verified assembly code is able to match the performance of the best available

implementations for the encoded cryptographic primitive. Moreover, Jasmin code can be proved correct and secure via an equivalence proof to a high-level specification in EasyCrypt: specifically, the Jasmin compiler is able to create an EasyCrypt translation of its source. End-to-end security and correctness follow from the certification of the Jasmin compiler (source-to-assembly) and from the EasyCrypt proof (source-to-spec).

Experience in using EasyCrypt and Jasmin [SoK] shows that both foundational and applied research are needed to tackle two classes of cryptographic protocols that have recently assumed a prominent relevance due to advances of Quantum Computing: 1) post-quantum cryptography (PQC) and 2) secure distributed (multiparty) computation. PQC should not be confused with quantum computing or quantum cryptography; its goal is to create cryptographic schemes that can be used today and resist potential quantum attacks in the future. PQC relies on different mathematical abstractions and computational assumptions than classical crypto such as lattice-based [CRYS] and isogeny-based assumptions [SIKE]. These imply reasoning about distributions and complex mathematical objects for which little has been done in the machine-checking setting; indeed, proof techniques are still maturing in the cryptographic community to permit fine-tuning parameter sizes and improve performance. The majority of these proofs assume a quantum adversary interacting with a classical system, which can be formalized as an extension of the current EasyCrypt semantics, but this is currently lacking. Implementing these primitives also raises new challenges for the Jasmin framework; e.g., it was not yet considered rejection sampling mechanisms, which are crucial in PQC. Orthogonal challenges arise in the verification of interactive protocols in Jasmin and EasyCrypt. Each cryptographic primitive is proved secure in a security model that captures its use in the real world. Primitives such as key exchange have been proved secure in a variety of models [KECR], with surprising complexity for two-party protocols: the goal is to capture concurrent executions and deal with composition within the security proof itself. General approaches to composability [UC] exist, but these exclude some of the more efficient instantiations, or require ad-hoc adaptations to capture the associated caveats. Recent works [KMS,EUC,LMPC] have highlighted a mismatch between the security semantics of multiparty computation and the EasyCrypt framework. This is not surprising, as main use cases for EasyCrypt are non-interactive primitives. What *is* surprising is that it can used EasyCrypt to reason about restricted classes of distributed protocols, such as those offering semi-honest security, or constant-round two-party computation [LMPC, EUC]. It has been explored more complex protocols [KMS], and identified the main bottlenecks: when the scheduling of executions becomes even moderately complex, the number of cases explodes and there is a large overhead in specifying and verifying global invariants by hand. These results point to new directions we aim to explore, while stressing the goal being of obtaining verified proofs matching those written by cryptographers, rather than using a symbolic model (e.g., as in Tamarin or ProVerif), which are incomparable.

Objectives

The aim of this project is to develop extensions to the Jasmin-EasyCrypt tool chain, so that it can be used to formally verify the security and correctness of state of the art post-quantum cryptography (PQC) implementations. For concreteness, most of the effort will concentrate on the candidate submissions to the ongoing NIST competition for PQC. Specifically, this project aims at addressing the following set of challenges:

- To develop EasyCrypt libraries supporting for currently lacking data types and operators. To support the most efficient lattice-based schemes, there's need to cover polynomials over rings and finite fields, cyclotomic polynomials, matrices and vectors thereof. Important operations include the NTT transform, an analogue of the Fourier transform, norm computations, and sampling from non-uniform distributions, namely

for dealing with low-norm noise. In parallel, Jasmin must be extended to deal with machine instructions that are sometimes used to optimize the implementations of these operations, namely (vectorized) floating point instructions.

- To extend EasyCrypt with new program logics to deal with quantum attackers, i.e. an attacker in possession of a quantum computer (say a large organization or a country) is trying to break cryptography implemented in classical computers. This model is particularly important for long-term security of data protected today. There is a growing number of results [PQPR,PQRO] in cryptography that extend classical constructions and generic transformations (e.g., those based on random oracles) to this setting, and some seminal work in formalizing some of these results [PQEC]. Our goal will be to distill these developments into extended EasyCrypt and Why3 logics, to enable the machine-checking of formal security proofs of some of the NIST candidates.

- To develop proof techniques and automation to deal with optimizations. PQC proofs are also challenging because they use aggressive parameter optimization in order to obtain a level of performance compatible with real-world use. This often implies adopting new proof techniques to improve the bounds, which deviate from the standard game-hopping approach. Two examples of this are the use of truncation and rounding to introduce noise and simultaneously compress public and secret keys. These techniques imply that schemes do not always work correctly, and it is necessary to perform intricate analysis of complex distributions to bound the probability of error. Similarly, improving bounds when adversaries interact with quantum random oracles and similar abstractions require reasoning about amortized bad event analysis, i.e., avoiding the use of coarse union bounds across potential occurrences when bounding the probability of a bad event.

Work description

In what follows, we detail some of the activities planned in this projects, and specifically focusing on the first year of the doctoral programme.

The primary objective of the first year of the project is, of course, formative. As such, it will be followed a comprehensive program of supervised studies on foundational subjects in the area, such as theoretical cryptography; required mathematical structures; and deductive reasoning. Moreover, it is planned to attend some international advanced schools in the area, such as the "Summer School on real-world crypto and privacy" (https://summerschool-croatia.cs.ru.nl/2022/) and/or "BIU Winter School on Cryptography" (https://www.youtube.com/watch?v=fV3oD4sarNA).

Having in mind that the main goal of the first year is formative, it is nevertheless planned to start a *seed project* to be developed in parallel with the study mentioned above, and encompassing already some of the ingredients that shall be addressed during the research. This seed project will consist in implementing in Jasmin, and experimenting with, different versions of the ZKBoo protocol [ZKBoo], which is the basis of PQC Picnic signature scheme. The choice of the aforementioned protocol is justified by the fact that it is based on the so called MPC-in-the-Head transformation, which jointly require reasoning about distributed protocols and post-quantum cryptography – the main challenges addressed earlier. Moreover, it is used as a sub-component for a signature scheme through the Fiat-Shamir construction whose formalisation in EasyCrypt is, by itself, of independent interest. But even if this seed project already addresses some of the central points of the research line planned for the PhD, the main motivation for undertaking it is indeed still formative in its nature. Specifically:

- the Picnic signature scheme was at the heart of the author's MSc thesis [MScThesis], and as such this implementation can be seen a natural follow-up of that work, benefiting from the author's familiarity with the subject;
- it could, and should, be understood as an opportunity for an hands-on approach to the EasyCrypt/Jasmin development framework. Such a non-trivial development shall certainly demand a deep understanding of the underlying tools, allowing for a first assessment of what are their strengths and limitations in targeting the aimed schemes;
- last but not least, it is also an opportunity to get knowledge and interact with the research community around the EasyCrypt/Jasmin framework, as well as the development teams of the tools.

References

[1] J. Almeida et al. "Jasmin: High-Assurance and High-Speed Cryptography." In: *ACM CCS* (2017).

[2] M. Barbosa et al. "SoK: Computer-Aided Cryptography." In: (2021).

[3] M. Bellare, D. Pointcheval, and P. Rogaway. "Authenticated Key Exchange Secure against Dictionary Attacks." In: *IACR EUROCRYPT* (2000).

[4] K. Bhargavan, F. Kiefer, and P.-Y. Strub. "HACSpec: Towards Verifiable Crypto Standards." In: *SSR* (2018).

[5] R. Canetti. "Universally Composable Security." In: *J. ACM* (2020).

[6] *CRYSTALS: Cryptographic Suite for Algebraic Lattices.* 2021.

[7] *SIKE: Supersingular Isogeny Key Encapsulation.* 2021.