# Research Statement

Benedikt Schmidt

The goal of my research is to build tools that help developers to increase the security and the efficiency of the systems they design. To achieve this, my research applies techniques from theorem proving, programming languages and program verification to problems in information security and cryptography. The considered problems range from analyzing the security of key exchange protocols, through finding efficient and secure realizations of cryptographic primitives, to automatically analyzing cryptographic assumptions. Improving the security of key exchange protocols is highly important since they are one of most widely deployed cryptographic protocols and used in virtually any secure communication protocol such as TLS, Quic, or SSH. The problem is also far from solved as numerous realistic attacks on deployed protocols demonstrate.[1] To obtain meaningful security guarantees, it is also necessary to prove the security of the underlying cryptographic primitives such as signature schemes and public key encryption. The security of these primitives in turn relies on cryptographic assumptions, whose hardness must be also verified in idealized models. Furthermore, security guarantees should not be limited to simplified algorithmic descriptions, they should also apply to real-world implementations.

In all these areas, techniques from theorem proving, programming languages and program verification are proving to be highly useful to obtain security proofs with strong correctness guarantees and to achieve a high degree of automation. Strong correctness guarantees can be obtained by building machine-checked proofs where only the proven statement must be manually verified. After coming up with the right formalism to express proofs, it is also often possible to devise automated proof search procedures. Such procedures can then be used to evaluate the security of synthesized constructions to find optimal trade-offs between security and efficiency. In the remainder of this statement, I will first summarize my contributions to the areas of tool-supported analysis of security protocols and computer-aided cryptography and then conclude by describing ongoing and future work. A common goal of my work is to develop usable tools and publish them under open-source licenses to ensure that the results of my research can be used and extended by other research groups.[2]

## Analysis of security protocols in strong adversary models

In this section, I describe my contributions to the analysis of (physical) security protocols.

**Automated protocol analysis in the symbolic model.** Push-button tools for analyzing security protocols in symbolic models of cryptography have been highly successful in both discovering new flaws and proving security with respect to large classes of attackers. Unfortunately, existing tools cannot be used to prove security for state-of-the-art key exchange protocols with respect to an unbounded number of protocol sessions. Tools that provide the required combination of support for reasoning about Diffie-Hellman exponentiation and an input language that is expressive enough to formalize the strong adversaries considered for these protocols are restricted to a bounded number of sessions. To lift this restriction, we have developed a general approach for the symbolic analysis of security protocols that supports Diffie-Hellman exponentiation and an expressive input language [12]. We model protocols as multiset rewriting systems and security properties as first-order temporal formulas. We analyze them using a novel constraint-solving algorithm that supports both falsification and verification. The algorithm exploits the finite variant property and builds on ideas from strand spaces and proof normal forms. We

[1] For example, the Logjam attack combines a protocol flaw in TLS with advances in discrete log computation and was applicable to 8% of the most popular 1 million websites. [2] `http://beschmi.net/#tools`

have implemented the algorithm in the Tamarin tool[3] and demonstrate the scope and the effectiveness of our algorithm on numerous case studies.

In my dissertation, I have further extended the approach to support reasoning about bilinear pairings, natural numbers, multisets and finite maps. We have used this extension to obtain the first symbolic correctness proofs for group key agreement protocols that use Diffie-Hellman or bilinear pairing, loops, and recursion, while at the same time supporting advanced security properties, such as perfect forward secrecy and eCK-security [13]. To demonstrate the effectiveness of our approach, we automatically verify a set of protocols, including the STR protocol, the group Joux protocol and the GDH protocols. The Tamarin tool has also been successfully used as a backend for verifying protocols with global state by Kremer and Künnemann [11] and extended to observational equivalence by Basin, Dreier and Sasse [9].

**Machine-checked security proofs in the computational model.** Even though Tamarin uses a rich equational theory to model Diffie-Hellman groups, the approach still considers significantly fewer adversaries than proofs in the computational model. The main reason is that the symbolic model only considers adversaries that deduce new messages according to a fixed set of deduction rules, which is not justified in our setting since there is no applicable computational soundness result. To combine the strong security guarantees of a proof in the computational model and the strong correctness guarantees of a machine-checked proof, we have used EasyCrypt [6] to obtain security proofs of implicitly authenticated key exchange protocols in the eCK model. EasyCrypt is a machine-checked framework for building and verifying security proofs of cryptographic constructions mirroring the popular code-based, game-based reductionist arguments used by cryptographers. In EasyCrypt proofs, probabilistic claims are justified by using a probabilistic relational Hoare Logic and an ambient logic similar to higher-order logic. Our formal proof consists of two independent parts and critically relies on features added to the new version of EasyCrypt [6], to which I have contributed. We first develop a new generic proof of security for key-exchange protocols and then instantiate it to obtain security proofs for known protocols, such as Naxos and Nets, with respect to different adversary models and hardness assumptions.

**Machine-checked security proofs of physical security protocols.** In another related line of work [8], we initiate the formal analysis of protocols that establish and rely on properties of the physical environment. A prominent example of such protocols are distance bounding protocols where a prover must convince a verifier that they are sufficiently close. To achieve this, distance bounding protocols combine cryptography with time-of-flight measurements. Our formal model to analyze such protocols uses a trace-based operational semantics that captures location, network connectivity, time and a symbolic model of cryptography. To evaluate the approach, we have formalized our framework and several case studies, ranging from distance bounding protocols to secure time synchronization protocols, in Isabelle/HOL. Building on this work, we observe in [10] that the standard security definitions for distance bounding do not account for an important class of attacks which we coin Distance Hijacking. Such attacks pose a serious threat in many practical scenarios and we show that many proposed distance bounding protocols are vulnerable. We show how to make these protocols resilient to Distance Hijacking and formally prove the effectiveness of our countermeasures in an extension of our Isabelle/HOL framework.

## Computer-aided Cryptography

In this section, I describe my contributions to the area of computer-aided cryptography which focus on the areas of automated proofs and synthesis of efficient and secure constructions.

**Analyzing assumptions in generic group models.** Generic group models are one of the main methods to evaluate non-standard hardness assumptions in cyclic groups, bilinear groups, and the emerging field of multilinear groups. In [4], we initiate the study of principled, automated, methods for analyzing hardness assumptions in generic group models. We follow the approach of the symbolic model of cryptography and first prove a novel computational soundness theorem that allows us to translate security experiments in generic group models (including models of multilinear groups) to problems in polynomial algebra. Our Generic Group Analyzer tool[4] then uses SMT solvers and methods from computer algebra, such as Gröbner Bases, to perform both falsification and verification.

[3] `http://tamarin-prover.github.io` [4] `http://generic-group-analyzer.github.io`

**Automated proofs and synthesis of signature schemes in the generic group model.** A signature scheme is structure preserving if message, public key, and signature all consist of group elements and the verification algorithm evaluates a pairing product equation. Security of structure preserving signature (SPS) schemes in the generic group model is mainly used to find SPS schemes of minimal size or with maximal efficiency. In [5], we refine previous work and introduce a new (more realistic) efficiency measure. We prove lower bounds for this measure and build a tool that automates the generation and security analysis of SPS schemes. Moreover, we leverage the tool to find new SPS schemes that match our new lower bounds and improve over previous work. The main limitation of this work is that our analysis is restricted to adversaries that perform a fixed number of signing queries. We haved lifted this restriction by developing a method that can deal with an unbounded number of oracle queries in a paper that is currently under submission.

**Automated computational proofs and synthesis for cryptographic primitives.** Public-key encryption schemes built from hash functions and trapdoor permutations are highly popular and used in many real-world system. Even though the building blocks are well understood, obtaining formal proofs and finding the right balance between simplicity, efficiency and security is challenging. To address this challenge, we have developed [3] new proof systems for automatically analyzing the chosen-plaintext and chosen-ciphertext security of such schemes in the random oracle model. Our approach uses a novel combination of techniques from computational and symbolic cryptography. Building on these proof systems, we develop the ZooCrypt toolset[5] that bundles together fully automated proof and attack finding algorithms. We use this toolset to build a comprehensive database of encryption schemes that records attacks against insecure schemes, and proofs with concrete bounds for secure ones. In [7], we build on this work to obtain highly automated proofs for pairing-based cryptography in the standard model. The resulting AutoG&P tool[6] supports extraction of independently verifiable proofs to EasyCrypt and is much more general than ZooCrypt. For example, the user can specify assumptions and security experiments.

## Future work

In general, I expect that the two lines of work on security protocols and computer-aided cryptography will converge in the future and use similar methods. Additionally, I will increase my focus on transferring proofs from the algorithmic to the implementation level, for example by working on the following topics.

**Verified implementations of security protocols.** The first first challenge consists of obtaining succinct and highly automated proofs of channel establishment protocols such as Quic or TLS. To achieve this, I am currently extending the approach used in the AutoG&P tool to security protocols. Lifting such proofs to implementations in a low-level language such as C, Rust or assembly or a combination of those poses additional challenges which I plan to address as follows. To prove functional correctness and side-channel resilience of low-level crypto code, I am currently developing a domain specific language and a verified compiler to write highly optimized code that exploits modern processor features like vector extensions (Intel AVX, Arm Neon). The language simultaneously provides a high-level semantics for simple correctness proofs and a low-level semantics that is used to reason about performance and cache and timing side-channels. For the message parsing and protocol logic, I plan to develop tools for proving equivalence between high-level algorithmic specifications and low-level implementations using relational program logics.

**Proof-guided synthesis.** Existing tools for synthesis of new schemes or automated translation of schemes to new settings such as AutoGroup+ [2] rely on generic proofs to obtain security of their outputs. I plan to exploit that in tools like ZooCrypt and AutoG&P, proofs are formal objects that can be inspected to guide synthesis and that can be manipulated to obtain translated proofs. In addition to translation, such an approach is also highly promising for optimizing concrete schemes obtained by instantiating generic constructions. For example, it is often possible for different components to share randomness, but this requires combining and adapting the generic proof and the proofs of the individual components. Another advantage of such an approach is that we can obtain independently verifiable evidence for the security of the synthesized scheme similar to our work on certified synthesis of batch verifiers [1].

[5] `http://zoocrypt.github.io` [6] `http://autognp.github.io`

## References

[1] J. A. Akinyele, G. Barthe, B. Grégoire, B. Schmidt, and P. Strub. Certified synthesis of efficient batch verifiers. In *IEEE Computer Security Foundations Symposium (CSF)*, 2014.

[2] J. A. Akinyele, C. Garman, and S. Hohenberger. Automating fast and secure translations from type-I to type-III pairing schemes. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2015.

[3] G. Barthe, J. M. Crespo, B. Grégoire, C. Kunz, Y. Lakhnech, B. Schmidt, and S. Z. Béguelin. Fully automated analysis of padding-based encryption in the computational model. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2013.

[4] G. Barthe, E. Fagerholm, D. Fiore, J. C. Mitchell, A. Scedrov, and B. Schmidt. Automated analysis of cryptographic assumptions in generic group models. In *Advances in Cryptology - CRYPTO*, 2014.

[5] G. Barthe, E. Fagerholm, D. Fiore, A. Scedrov, B. Schmidt, and M. Tibouchi. Strongly-optimal structure preserving signatures from type II pairings: Synthesis and lower bounds. In *Workshop on Theory and Practice in Public Key Cryptography (PKC)*, 2015.

[6] G. Barthe, B. Grégoire, S. Heraud, and S. Béguelin. Computer-aided security proofs for the working cryptographer. *Advances in Cryptology–CRYPTO 2011*, pages 71–90, 2011.

[7] G. Barthe, B. Grégoire, and B. Schmidt. Automated proofs of pairing-based cryptography. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2015.

[8] D. Basin, S. Capkun, P. Schaller, and B. Schmidt. Formal reasoning about physical properties of security protocols. *ACM Transactions on Information and System Security (TISSEC)*, 2011.

[9] D. Basin, J. Dreier, and R. Sasse. Automated symbolic proofs of observational equivalence. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2015.

[10] C. Cremers, K. Rasmussen, B. Schmidt, and S. Capkun. Distance hijacking attacks on distance bounding protocols. In *IEEE Symposium on Security and Privacy (S&P)*, 2012.

[11] S. Kremer and R. Künnemann. Automated analysis of security protocols with global state. In *IEEE Symposium on Security and Privacy (S&P)*, 2014.

[12] B. Schmidt, S. Meier, C. Cremers, , and D. Basin. Automated analysis of Diffie-Hellman protocols and advanced security properties. In *IEEE Computer Security Foundations Symposium (CSF)*, 2012.

[13] B. Schmidt, R. Sasse, C. Cremers, and D. Basin. Automated verification of group key agreement protocols. In *IEEE Symposium on Security and Privacy (S&P)*, 2014.