Report from Dagstuhl Seminar 19301

Secure Composition for Hardware Systems

Edited by

Divya Arora[1], Ilia Polian[2], Francesco Regazzoni[3], and Patrick Schaumont[4]

1 **Intel – Santa Clara, US,** divya.arora@intel.com
2 **Universität Stuttgart, DE,** ilia.polian@informatik.uni-stuttgart.de
3 **University of Lugano, CH,** regazzoni@alari.ch
4 **Virginia Polytechnic Institute – Blacksburg, US,** schaum@vt.edu

Abstract

The goal of the Dagstuhl Seminar 19301 "Secure Composition for Hardware Systems" was to establish a common understanding of principles and techniques that can facilitate composition and integration of hardware systems to achieve specified security guarantees.

Theoretical foundations of secure composition have been laid out in the past, but they are limited to software systems. New and unique security challenges arise when a real system composed of a range of hardware components, including application-specific blocks, programmable microcontrollers, and reconfigurable fabrics, are put together. For example, these components may have different owners, different trust assumptions and may not even have a common language to describe their security properties to each other. Physical and side-channel attacks that take advantage of various physical properties to undermine a system's security objectives add another level of complexity to the secure composition problem. Moreover, practical hardware systems include software of tremendous size and complexity, and hardware-software interaction can create new security challenges.

The seminar considered secure composition both from a pure hardware perspective, where multiple hardware blocks are composed in, *e.g.*, a system on chip (SoC), and from a hardware-software perspective where hardware is integrated within a system that includes software. The seminar brought together researchers and industry practitioners from fields that have to deal with secure composition: Secure hardware architectures, hardware-oriented security, applied cryptography, test and verification of security properties. By involving industrial participants, we were able to get insights on real-world challenges, heuristics, and methodologies employed to address them and initiate a discussion towards new solutions.

Seminar July 21–26, 2019 – http://www.dagstuhl.de/19301
2012 ACM Subject Classification Hardware → Methodologies for EDA, Hardware → Integrated circuits, Security and privacy → Formal security models
Keywords and phrases Hardware, Secure composition, Security, Software
Digital Object Identifier 10.4230/DagRep.9.7.94
Edited in cooperation with Elif Bilge Kavun, The University of Sheffield, UK, e.kavun@sheffield.ac.uk

1 Executive Summary

Ilia Polian (Universität Stuttgart, DE)
Divya Arora (Intel – Santa Clara, US)
Francesco Regazzoni (University of Lugano, CH)
Patrick Schaumont (Virginia Polytechnic Institute – Blacksburg, US)

Today's electronic systems consist of mixtures of programmable, reconfigurable and application-specific hardware components, tied together by tremendously complex software. At the same time, systems are increasingly integrated such that a system that was traditionally regarded "harmless" (*e.g.*, an entertainment system in a car) finds itself tightly coupled with safety-critical driving-assistance systems and security-sensitive online payment systems. Moreover, a system's hardware components are now often directly accessible to the users, making the system vulnerable to physical attacks via its hardware which becomes the system's "Achille's heel". This necessitates a new look on system security from hardware perspective.

The Dagstuhl seminar "Secure Composition for Hardware Systems", which took place on July 21-26, 2019, focused on secure composition of systems which contain hardware blocks. This is a practically important but a theoretically challenging problem where several foundational questions still lack an adequate answer.

Several formats were used during the seminar. The first phase of the seminar, which focused on prior findings, started with presentations by five pre-selected experts giving their view on secure composition from different perspectives: theory, design automation, trusted execution environments and attacks countermeasures. Then, small-group discussions of relevant state of the art were held, focusing on questions such as "What does it mean to securely compose two elements?" or "What is the role of models in secure composition?" The findings of the small groups were intensively discussed in plenary sessions.

The second phase of the seminar was devoted to discussing research questions. Some of the questions were prepared by the seminar organizers (*e.g.*, "Which protocol-level secure composition methods are applicable in hardware domain?" or "How to counter possible loss of security due to abstraction of hardware components?") and some additional questions were proposed by the participants (*e.g.*, "How to bootstrap trust in a distributed hardware system?"). The questions were discussed again in small groups, intertwined by individual presentations in plenum (for instance, an in-depth study on the applicability of Universal Composability (UC) in the hardware domain).

Two immediate outcomes grew out of the seminar. First, some participants are organizing a special session on secure compositions in one of the leading scientific conferences; a respective proposal was recently accepted by the "Design, Automation, and Test in Europe Conference" (DATE). Second, there is an ambitious plan to prepare a manuscript on the full variety of aspects in secure composition of electronic systems and submit it as a "Systematization of Knowledge" (SoK) paper to the IEEE Symposium on Security and Privacy (S&P); this effort is ongoing at the time of writing this report.

Overall, we believe that this seminar has provided entirely new insights to most of the participants and has opened new avenues for research on the intersection of security and hardware systems. It brought together researchers from communities who rarely interacted with each other in the past. The seminar helped define new research challenges, and activities are underway to put the topic of secure composition higher on the agenda of the respective communities.

The organizers are thankful to the Dagstuhl team (and in particular to Dr. Andreas Dolzmann who handled the scientific part and Mrs. Heike Clemens who was of invaluable help in organizing the social event and masterly handled all practical issues); to Dr. Elif Bilge Kavun who did a great job in collecting and organizing the documents from participants and in preparing the summarizing texts; and to all the participants for making this seminar a success.

2 Table of Contents

3 Overview of Pre-selected Talks

3.1 Commercial Trusted Execution Environments (TEEs) – An Overview

Divya Arora (Intel – Santa Clara, US)

A Trusted Execution Environment (TEE) is a hardware/software/firmware framework to allow isolated execution of security-sensitive code and its aim is to reduce Trusted Computing Base (TCB) of sensitive code. Isolated execution, secure storage, remote attestation, secure provisioning, and trusted input output are general properties of a TEE. Today, many commercial TEE solutions are available and different approaches exist for reducing application TCB; however, not all TEEs support all of the listed TEE properties.

This talk provided an overview of commercially available TEEs like ARM® TrustZone® (TZ), Microsoft Virtualization Based Security (VBS), AMD Secure Encrypted Virtualization (SEV), and Intel® Software Guard Extensions (SGX) and asked the question "How do we reason about security of TEEs including hardware/firmware/software?". Also, a comparison of the supported properties in the commercial TEEs is also provided in the talk: For example, ARM®TZ and Microsoft VBS do not support remote attestation property and Intel®SGX does not support trusted input output. Finally, some examples of challenges that many TEEs face in terms of ecosystem deployment are listed:

- Not all TEEs are available to regular users (*e.g.*, selected usages deployed as part of VTL1 in VBS)
- There may still be a large software attack surface in some cases (*e.g.*, integer overflow in TZ Secure OS)
- Memory integrity & anti-replay are very hard on performance/area
- Many TEEs rely on hardware sharing to amortize the cost of creating a separate environment which may lead to side-channels
- Some TEEs require partitioning of existing applications or "enlightenment" of existing virtual machines which is harder to deploy in the ecosystem

3.2 Introduction to Universal Composability

Ran Canetti (Tel Aviv University, IL)

In this talk, a general universal composability framework for describing cryptographic protocols and analyzing their security is presented. The framework allows specifying the security requirements of practically any cryptographic task in a unified and systematic way. Furthermore, in this framework the security of protocols is preserved under a general protocol composition operation, called universal composition.

The proposed framework with its security-preserving composition operation allows for modular design and analysis of complex cryptographic protocols from simpler building blocks. Moreover, within this framework, protocols are guaranteed to maintain their security in any context, even in the presence of an unbounded number of arbitrary protocol instances that run concurrently in an adversarially controlled manner. This is a useful guarantee, which allows arguing about the security of cryptographic protocols in complex and unpredictable environments such as modern communication networks.

3.3 State-of-the-art Implementation Attacks

Elke De Mulder (Rambus – Sunnyvale, US)

This talk focused on the state-of-the-art in side-channel leakage analysis and mitigation from the point-of-view of compositional security.

Three large research directions are discussed. The first is the non-completeness of our understanding of the physical leakage of devices, whether it is a pure hardware implementation or a software implementation running on an embedded processor or larger system on chip. The second one is the use of formal proofs to guarantee security properties of implementations where composability plays a role for combining smaller provable secure components to create a larger implementations and for combining countermeasures for different types of attack. Are the security properties still valid? The last research direction discussed in this talk is testing. With a growing arsenal of attacks, how can one practically test whether an implementation resist all of part of them in a reasonable amount of time?

3.4 Cryptographic Hardware Design – Challenges and Remarks

Tim Erhan Güneysu (Ruhr-Universität Bochum, DE)

As a result of digital evolution, today's systems consist of more software than hardware. In contrast, security demand of hardware systems does not decrease due to physical exposure, enhanced security requirements, and advanced networking and connectivity.

This talk focused on challenges in cryptographic hardware design in the presence of attacks and protection measures. Secure hardware elements have to provide security guarantees according to defined attacker model while keeping a trade-off between security, efficiency and cost. Security guarantees often (implicitly) bound to technical/physical device limitations which is important for composability.

It is possible to divide the challenges in cryptographic hardware into two groups: Crypto-level and system-level. On the crypto side, new computing paradigms (*e.g.*, quantum computers makes existing public-key cryptography obsolete) and new/changed requirements (*e.g.*, fault tolerance and verifiable & delegated computation) cause challenges. On the system side, static nature of hardware, non-trivial upgrade/migration, and security validation & testing are the main problems. Some solutions to crypto-level and system-level challenges are also presented in the talk (together with example applications): Hardware implementations of post-quantum cryptography (crypto-level) and integration of cryptographic hardware (system-level).

As a final remark, existing challenges are listed:

- Imperfections of hardware within abstract models and requirements in higher layers
- Composition and multiplicity of (low) confidence from practical security element evaluation
- Lack of a realistic simulator for complex systems
- Long term security, efficiency & cost

3.5 State-of-the-art in EDA Security

Francesco Regazzoni (University of Lugano, CH)

Security is one of the most important properties that should be provided by a system. Unfortunately, due to physical attacks, the presence of cryptographic primitives is not sufficient to fulfill this requirement.

In recent years, researchers invested significant efforts implementing optimized security primitives. These blocks are generally produced by expert designers and they are integrated manually into the whole system. This approach however is not optimal, since manual integration is a time consuming and error prone process. Furthermore, this approach is particularly dangerous when used for implementing side-channel resistant designs.

A more effective way to implement secure cryptographic algorithms would enable the automatic application of side-channel countermeasures and would support the verification of their correct application.

This talk revised and summarized the research efforts in this important research direction, starting from the first works implementing hardware design flow for security to the initial steps of automatically driving design tools using security variables and highlights future research direction in design automation for security.

4 Overview of Individual Presentations

4.1 Composability of Machine-Learning Resistant PUFs – When Yao Fails -or- Can we build secure composite PUFs?

Fatemeh Ganji (University of Florida – Gainesville, US)

When it comes to composability in the context of cryptography, Yao's lemma [1] plays an important role. This lemma states that if several instances of a somewhat-hard function are XORed together, the resulting function is harder to compute. Moreover, in cryptography and machine learning (ML) theory, it is well-known how to make a connection between the security and provable ML. Here we quote from the seminal work of Rivest, published in 1991 [2]: "In cryptography, the major goal is to 'prove' security under the broadest possible definition of security, [...]. [...], in the typical paradigm, it is shown that there is no polynomial-time [learning] algorithm that can 'break' the security of the [secure] system." From these two principles, we can conclude that a physical primitive can become more robust against ML attack if we combine some instances of that by applying the XOR function.

In this talk, we show that this is, unfortunately, not the case. In particular, although the above approach makes physical primitives more resilient to ML attacks, it still requires drastic practical measures to be taken to achieve the ultimate level of security, where the attacker cannot learn the functionality of the respective primitive. We elaborate on this in the context of physically unclonable functions.

References

1 A.C. Yao *Theory and Application of Trapdoor Functions*. In 23rd Annual Symposium on Foundations of Computer Science (sfcs 1982) (pp. 80-91), IEEE, 1982, November

2 R.L. Rivest *Cryptography and Machine Learning*. In International Conference on the Theory and Application of Cryptology (pp. 427-439), Springer, Berlin, Heidelberg, 1991, November

4.2 Attacks Through Externally-amplified Couplings

Itamar Levi (University of Louvain, BE)

Joint work of Itamar Levi, Davide Bellizia, François-Xavier Standaert
Main reference Itamar Levi, Davide Bellizia, François-Xavier Standaert: "Reducing a Masked Implementation's Effective Security Order with Setup Manipulations And an Explanation Based on Externally-Amplified Couplings", IACR Trans. Cryptogr. Hardw. Embed. Syst., Vol. 2019(2), pp. 293–317, 2019.
URL https://doi.org/10.13154/tches.v2019.i2.293-317

Couplings are a type of physical default that can violate the independence assumption needed for the secure implementation of the masking countermeasure. Recent works put forward qualitatively that couplings can cause information leakages of lower order than theoretically expected. However, the (quantitative) amplitude of these lower-order leakages (*e.g.*, measured as the amplitude of a detection metric such as Welch's T statistic) was usually lower than the one of the (theoretically expected) d^{th} order leakages. So, the actual security level of these implementations remained unaffected. In addition, the couplings had to be internally amplified in order to make them visible (*e.g.*, by tweaking the placement and routing or iterating linear operations on the shares).

In this talk, firstly, how the amplitude of low-order leakages in masked implementations can be externally amplified by tweaking side-channel measurement setups in a way that they are under control of a power analysis adversary is explained. The experiments put forward that the "effective security order" of both hardware (Field Programmable Gate Array – FPGA) and software (ARM-32) implementations can be reduced, leading to concrete reductions of their security level. For this purpose, instead of the detection-based analyses of previous works, attack-based evaluations are performed in order to allow the confirmation of the exploitability of the amplified lower-order leakages. In the talk, a tentative explanation for the effects based on couplings is provided and a model that can be used to predict them in function of the measurement setup's external resistor and implementation's supply voltage is described. In conclusion, the effective security orders observed are mainly due to "externally-amplified couplings" that can be systematically exploited by actual adversaries.

4.3 CAD for Physical Attacks – A Fault Attack Perspective

Debdeep Mukhopadhyay (Indian Institute of Technology – Kharagpur, IN)

License 🅭 Creative Commons BY 3.0 Unported license
© Debdeep Mukhopadhyay
Joint work of Sayandeep Saha, S. Nishok Kumar, Sikhar Patranabis, Debdeep Mukhopadhyay, Pallab Dasgupta
Main reference Sayandeep Saha, S. Nishok Kumar, Sikhar Patranabis, Debdeep Mukhopadhyay, Pallab Dasgupta: "ALAFA: Automatic Leakage Assessment for Fault Attack Countermeasures", in Proc. of the 56th Annual Design Automation Conference 2019, DAC 2019, Las Vegas, NV, USA, June 02-06, 2019, p. 136, ACM, 2019.
URL https://doi.org/10.1145/3316781.3317763

The talk presents an overview on automation for fault analysis attacks on cryptosystems. Fault attacks have emerged as a strong attack vector for crypto-implementations and thus need to be properly mitigated using suitable countermeasures. Test and analysis of these countermeasures, particularly in the black-box setting is thus of demand. The talk outlines two approaches: first a prototype tool called ExpFault to analyze differential fault analysis of ciphers at the algorithm level. Secondly, ALAFA, an automated leakage assessment framework was presented which derives its root from classical non-interference theorem and uses t-test based identification of leakage. The tool can be promising for security evaluation for protected crypto-designs and hardware security modules. More details can be found in [1, 2].

References

1 Sayandeep Saha, S. Nishok Kumar, Sikhar Patranabis, Debdeep Mukhopadhyay, Pallab Dasgupta: ALAFA: Automatic Leakage Assessment for Fault Attack Countermeasures. DAC 2019: 136
2 Sayandeep Saha, Debdeep Mukhopadhyay, Pallab Dasgupta: ExpFault: An Automated Framework for Exploitable Fault Characterization in Block Ciphers. IACR Trans. Cryptogr. Hardw. Embed. Syst. 2018(2): 242-276 (2018)

4.4 Securing Cyber-physical Control Systems – A Formal Perspective

Dey Soumyajit (Indian Institute of Technology – Kharagpur, IN)

License 🅭 Creative Commons BY 3.0 Unported license
© Dey Soumyajit
Joint work of Saurav K. Ghosh, Dey Soumyajit

Given the widespread deployment of cyber-physical systems and their safety-critical nature, reliability and security guarantees offered by such systems are of paramount importance. While security of such systems against sensor attacks have garnered significant attention from researchers in recent times, improving the reliability of a control software implementation against transient environmental disturbances need to be investigated further. Scalable formal methods for verification of actual control performance guarantee offered by software implementations of control laws in the face of sensory faults have been explored in recent works. However, the formal verification of the improvement of system reliability by incorporating sensor fault mitigation techniques like Kalman Filtering and Sensor Fusion remains to be explored. Moreover, system designers are bound to face complex trade-off choices for deciding upon the usage of fault and attack mitigation techniques and scheduling them on available system resources as they incur extra computation load.

In this talk, recent contributions for securing cyber-physical control systems are explained. These are threefold:

- Formally analyzing the actual performance guarantee of control software implementations enabled with additional fault mitigation techniques
- Considering task-level models of such implementations enabled with security and fault tolerance primitives and constructing a time-automata based model which checks for schedulability on heterogeneous multi-core platforms
- Leveraging these methodologies in the context of a novel Design-Space-Exploration (DSE) framework that considers target reliability and security guarantees for a control system, and computes schedulable design options while considering well-known platform level security improvement and fault mitigation techniques

The contributions are validated over several case studies from the automotive domain.

4.5 Challenges in Secure Composition from a Practical Perspective

Marc Stöttinger (Continental AG – Frankfurt, DE)

Nowadays systems require security controls and countermeasures became system of systems with certain level of complexity due to composition of multiple software and hardware components. In this talk, practical challenges of putting security in composed systems were discussed on two examples. The first example demonstrates how an isolation layer for separation (established by two individual electrical components) is overcome by exploitations in the software domain. The second example discusses multiple exploitations on the example of an authentic communication and data exchange between a sensor node and a processing unit. The major issue in this example is the distributed development of both system components with too relaxed requirements.

4.6 Definition of "Root of Trust (RoT)"

Ingrid Verbauwhede (KU Leuven, BE)

This was an impromptu presentation of four slides to introduce Ingrid Verbauwhede's definition of a "root of trust."

First there is a clear distinction between what can be 'trusted' and what is 'trustworthy'. Trusted cannot be verified and if the trust is broken, the system can fail. In the context of hardware design, we want to minimize what needs to be trusted. This is mapped on the design pyramid. So, "a root of trust is a component at a lower abstraction layer, upon which the system relies for its security."

Some feedback received from the audience on the presentation:

- A component at a lower abstraction layer should be refined to "a component with an associated behavior or usage" for higher abstraction layers.

- A level below PUFs and TRNGs should be added to also include security problems at processing and technology levels (*e.g.*, to protect against Trojan circuits).
- This approach allows for reasoning on 'defense in depth'. If a security violation is detected at some interface, the consequences can be systematically evaluated.

5 Discussions on State-of-the-art Questions

5.1 Working Group A1

Divya Arora (Intel – Santa Clara, US), Gaetan Cassiers (University of Louvain, BE), Johann Heyszl (Fraunhofer AISEC – München, DE), Itamar Levi (University of Louvain, BE), Debdeep Mukhopadhyay (Indian Institute of Technology – Kharagpur, IN), Kazuo Sakiyama (The University of Electro-Communications – Tokyo, JP), and Dey Soumyajit (Indian Institute of Technology – Kharagpur, IN)

Group A1 summarized their answers to below questions as follows.

1. How would you handle composability?
2. What does it mean to securely compose two elements?
3. Can you give an example of a security failure due to insecure composition?
4. What is the foundation of composition?
5. How do you verify remote identity of a connected device?

- **Horizontal and vertical composition:**
  The composition of elements can be of two general forms which are different in terms of their interaction possibility. In a horizontal composition, the elements are independent from each other in the sense of running different execution environments (CPU or *e.g.* a state machine). Examples for this are embedded system's printed circuit boards containing multiple chips. Another example are SoCs including CPUs and peripherals. Interaction is achieved through dedicated interfaces.
  In a vertical composition, layers directly depend, respectively run on each other. They would typically share an execution flow. Examples are software layers conceptually running on top of each other (hardware, hypervisor, OS for a running system or hardware, bootloader and OS for the startup process). With vertical compositions, the interactions between elements (software layers) are usually much more complex.
- **Example for a system failure due to insecure composition:**
  In an example for a composed system, the CPU is fetching code from an external ROM and verifying its content before executing it from internal memory (cache). In a sense this can be seen as that the code is included into a trust bubble created by a RoT in the CPU for the purpose of system security. This code is part of the TCB for the system in the sense that the running system can jump into the code's routines for security purposes. The issue is that the code might be flushed from internal memory (cache) during runtime meaning that it leaves the trust bubble. In the example, the code is then simply fetched from external ROM again but without a repeated verification. The example shows how the composition of the system lead to a critical time of check – time of use issue.

- **Solutions:**
  State of the art threat modelling helps to assess the security of composed systems.
  On top of this, it seems like modelling and formulization are required, if composed systems are designed. This could result in a set of assertion-based checks which ensure security throughout the design flow. The drawback is that this assumes situations where the entire system is designed from scratch using this approach.
  Runtime filters, possible configured using the above derived assertions, seem as a valid countermeasure to prevent such situations.

5.2 Working Group A2

Georg T. Becker (ESMT – Berlin, DE), Yaacov Belenky (Intel Israel – Haifa, IL), Shivam Bhasin (Nanyang TU – Singapore, SG), and Shahin Tajik (University of Florida – Gainesville, US)

- **How would you handle composability?**
  By formalization of threat models and security feature for a composable system and developing frameworks to verify these properties at different levels (*e.g.*, netlist generation, route & placement, *etc.*)
- **What does it mean to securely compose two elements?**
  1- Preserving the functionalities of secure elements A and B according to the security guarantees of A and B: *e.g.*, shared resources on FPGAs leading to side-channel sources and fault injections
  2- Combining secure elements A and B as secure element C to achieve new security feature and functionalities.
- **Can you give an example of a security failure due to insecure composition?**
  Secure Element A is plugged to some non-security element B:
  - Meltdown/Spectre (Memory protection + Speculative execution)
  - Error Messages (Error reveals information and leaks sensitive information; *e.g.*, IEEE P1735 » Padding Oracle Attack on CBC mode)
- **What is the foundation of composition?**
  A possible foundation is a common language/specification in different levels of design and fabrication to assure the coherency between different blocks and assure the security (*e.g.*, constraints for CAD tools, Design Rule Checks (DRC)). However, first, one should overcome the challenge of describing the threat model for the composition.
- **How do you verify remote identity of a connected device?**
  This verification depends on threat model: There should at least be some secrets and this secret should be bounded to the identity (*e.g.*, identity-based cryptography, public-key infrastructure, device DNA, PUFs, *etc.*)

5.3 Working Group A3

Lucas Davi (Universität Duisburg-Essen, DE), Fatemeh Ganji (University of Florida – Gainesville, US), Tim Erhan Güneysu (Ruhr-Universität Bochum, DE), Ahmad-Reza Sadeghi (TU Darmstadt, DE), and Fareena Saqib (University of North Carolina – Charlotte, US)

- **How would you handle composability?**
  In an ideal world, systematic security integration, analysis models, and penetration testing tools for whole system emulation are the ways to handle composability.
  However, as there are integration issues in real world, a unified threat model with realistic assumptions together with divide and conquer approach considering security requirements can be the ways to handle composability.
- **What does it mean to securely compose two elements?**
  In *supply chain:* Security features on chip and the design house/foundry
  At *microarchitectural level (Rowhammer attack):* DRAM and access control & integrity checks on DRAM using software
  In *cryptographic designs:* Mathematically strong algorithms and secure interfaces & implementation
  At *system level (CANBus)*: Abstract isolation and standard security measures
- **Can you give an example of a security failure due to insecure composition?**
  Examples lie at different levels of abstraction. For example, there are supply chain threats and piracy. Other examples are microarchitecture level failures (Rowhammer bugs) and cryptographic algorithm failures due to implementation errors (memory corruption).
- **What is the foundation of composition?**
  - Clear requirements and assumptions that fit reality,
  - Security metrics and confidence of metrics,
  - Formal construction flow of integrating countermeasures (side effects of the protection countermeasures needs to be modeled),
  - New opportunities for attack (for example, self healing logic can be exploited),
  - Formal methods to measure and verify the security assumptions,
  - Stress testing in order to test all the corner cases for full coverage and assurance.
- **How do you verify remote identity of a connected device?**
  This can be verified via mutual authentication, remote attestation, hardware fingerprints, and secure hardware protocols.

5.4 Working Group B1

Ran Canetti (Tel Aviv University, IL), Jean-Luc Danger (Telecom ParisTech, FR), Elif Bilge Kavun (University of Sheffield, GB), Osnat Keren (Bar-Ilan University, IL), Johannes Mittmann (BSI – Bonn, DE), and Ilia Polian (Universität Stuttgart, DE)

- **How can we verify that mathematically proven properties are correctly implemented?**
  In the current certification practice, highest level (EAL7) includes requirements on formal techniques being used. On top of this, security properties can be checked at run time via

monitoring security properties of the entire system, making sure that secret key never appears on an SoC's bus, and detecting local attacks via sensors. Also, separation of system into small components may help as security then would be considered individually and future attacks would be contained to one component. Finally, techniques like modular redundancy that are used in safety context can be used to verify correct implementation.

- **What is the starting point of trust?**
  Here, the answer depends on the precise definition of "trust".
  We can converge towards root of trust through clarity, so the root of trust can be defined as the "clear" design. However, one has to trust vendor and the whole supply-chain where clarity is not possible (*e.g.*, design details kept secret due to certification).
- **Does time play a role in composition?**
  Time does play a role in composition; for example, timing of the composed system (order of events) is a side-channel. Delays also play a role in security solutions like blockchains.
- **Can security (strictly) increase, decrease or stay constant as a result of composition?**
  All three cases are possible – the point of concepts like Universal Composition is to prevent bad cases. Using composition to increase security sometimes looks obvious but in reality is difficult to prove rigorously. Security is often determined by weakest link of the system (composition may lead to security decrease). Sometimes, a system composed of imperfectly secure components becomes more secure due to composition (*e.g.*, hybrid key exchange via two mechanisms + XOR of the results), or secure components yield an insecure composition.
- **What is the role of models in secure composition?**
  A solution depends on how the problem is modeled. For example, in our context, security analysis will be based on (explicit or implicit) model of the attacker. However, while defining the models, we have to make sure that the unimportant details are omitted from the models and models on different levels of abstraction must be connected with each other.

5.5 Working Group B2

Elke De Mulder (Rambus – Sunnyvale, US), Elena Dubrova (KTH Royal Institute of Technology – Stockholm, SE), Yunsi Fei (Northeastern University – Boston, US), Paolo Palmieri (University College Cork, IE), and Milos Prvulovic (Georgia Institute of Technology – Atlanta, US)

Group B2 answered below questions along with an identification of similarities and differences between them.

- **Common aspects of questions:**
  All of the questions have multiple interpretations in the context of secure composition, which makes them excellent from the perspective of identifying research problems and seeing the big picture rather than focusing on a particular set of solutions.
  The group found that all the questions related to models of various aspects of security and models of security-relevant aspects of hardware/software systems. In particular, all the questions related to how assumptions built into the hardware designers' models

of their designs, and into models of potential attacks on those designs, are broken by successful attacks, and how composition tends to make these models' assumptions more problematic.

- **Differences between questions:**
  As the questions are highly inter-related, the group found it difficult to discuss the questions separately unless the problem space is significantly more constrained. However, the group also found that some of the questions, specifically questions 6 and 7, are about problems that are difficult even without composition, *i.e.*, the questions are about problems that exist (and are difficult to address) even in single-component systems.

- **How can we verify that mathematically proven properties are correctly implemented?**
  The group does not believe that it can be done for an arbitrary implementation, they believe that there should be a restriction so that the properties can be proven/verified. In order to verify the correctness of the implementation or properties, one needs to control the *entire* flow including not only the design itself, but also the supply chain.
- **What is the starting point of trust?**
  The group answers this question by asking if there is a need for a root of trust. Ideally, they would design a secure system without a RoT; however, in practice, it is not possible to avoid it. In that case, the group believes that the design, supply chain, and the user all need to be trusted.
- **Does time play a role in composition?**
  If time is understood as how much time an attacker has; yes, it does matter because time is a way of measuring security.
- **Can security (strictly) increase, decrease or stay constant as a result of composition?**
  The question is whether there are systematic ways of composing that maintain or improve security. This probably requires models with proofs and possibly prevents combinatorial explosion by finding which properties are provable for individual components.
- **What is the role of models in secure composition?**
  It is not really possible to have secure composition without models (there would be no systematic security without models). However, composition makes building models much more difficult. In order to deal with this, a hierarchy of models would be necessary. Also, problem-specific models are needed and countermeasures should be designed with respect to problems/models. However, there are still open questions like "Do multiple countermeasures work well together?"

5.6 Working Group B3

Annelie Heuser (IRISA – Rennes, FR), Michail Maniatakos (New York University – Abu Dhabi, AE), Wenjing Rao (University of Illinois – Chicago, US), Patrick Schaumont (Virginia Polytechnic Institute – Blacksburg, US), Werner Schindler (BSI – Bonn, DE), and Marc Stöttinger (Continental AG – Frankfurt, DE)

- **How can we verify that mathematically proven properties are correctly implemented?**
  More precisely, one usually tries to confirm model assumptions but not the conclusions of the mathematical model. The answer to the modified question depends on the particular feature or functionality. The correctness of implemented functions (*e.g.*, cryptographic algorithms) may be verified by known-answer tests. A known answer test constitutes a special case of a tests for particular properties, which itself is part of the implementation (see FIPS140-2). Another verification path could be the application of formally verified construction methods (see Common Criteria EAL6 or higher). The resistance against implementation attacks (side-channel attacks, fault attacks, *etc.*) may be confirmed by empirical analysis and experimental evaluation although this certainly is not a verification in a strict (mathematical) sense.
  Moreover, we identified several open problems, which should be discussed. One question concerns the quantitative impact on the security of the system if some model assumptions are at least to some degree invalid. Another problem is how the completeness of the model assumptions can be/should be verified. Finally, it is not obvious at which stage one should check whether the implementation fulfils the model assumptions.
- **What is the starting point of trust?**
  There can be potentially several different starting points of trust, depending on the threat model, the amount of resources available, and the solution cost. During the phase of statement collection, many different opinions on the starting point of trust were discussed: No trust, Mathematical theorem and properties, Hardcoded reference value, RTL, hardware, secure tamper proof storage, TPM, smart card, bootloader routine, and BIOS. Some of them (*e.g.*, hardware, TPM) are currently used in practice as starting points of trust. The question, however, is what the optimal starting point of trust is. Approaching the question from the philosophical side, the starting point of trust can be defined as an entity that cannot be divided into smaller entities without affecting the trust assumption of the mathematical model. Or, the starting point of trust is an entity that is believed to guarantee central mathematical security assumptions; depending on the model assumptions it may be advantageous if its complexity is low. Following the complexity discussion, the starting point of trust should be as simple as possible to the extent that it does not need to be verified.
- **Does time play a role in composition?**
  This question was ranked at the bottom of the list for the group discussion, according to a vote at the initial phase in the interests of prioritizing the questions. The main issue seems to be that there is not a clearly understood foundation for this question, perhaps due to the general nature of "time" – what exactly does "time" mean, in the specific context of this question (about composition)? It would have been more helpful if this question had been narrowed down in the context of security composition, or with some examples.

- **Can security (strictly) increase, decrease or stay constant as a result of composition?**
  Security can either decrease or increase because of composition. First, the security properties of components may not scale up to the composition of the components because the security properties may not be transferable to the composition. An example of that phenomenon can be seen in the composition of individual PUF using an XOR operation into a so-called XOR-PUF. The XOR-PUF construction was proposed to harden the PUF against model-building attempts for the individual PUF. For example, while the arbiter PUF is susceptible to model building, the XOR-composition of arbiter PUF was thought to prevent model-building of the components. However, recent progress in machine learning attacks on XOR-PUF has shown this assumption to be incorrect. The problem is that the XOR operation is linear, and machine learning tools can still classify individual components under such a linear composition. Hence, in this case, one can argue that security decreases as the result of the composition. It decreases because the XOR-PUF offers the illusion of hardening against model-building. The composition is ineffective.
  Second, the security properties of components may combine and strengthen the overall composition. An example can be seen in the composition of a better random number generator out of two biases random number generator using an XOR operation. In this case, the XOR result will generally show less bias and better overall distribution. The XOR is able to combine the entropy of each individual random number generator.
  These two examples show that an identical operation (XOR) can be detrimental or beneficial to the security properties of the composition. Therefore, the composition of secure elements must be analyzed as well, even when the security properties of individual secure elements is well understood.
- **What is the role of models in secure composition?**
  Models are a key point in secure composition. Models make the analysis of a system with several components feasible and they are the starting point in many scenarios. They are a means for communication between different parties to achieve a common understanding of properties, threats, vulnerabilities, or interfaces. Having precise models make complex problems manageable, but also impose risks to overlook side-effects if an invalid or imprecise model is used. This is a particular risk the field of secure composition as even though individual layers may have been modeled precisely, their composition may include additional unexpected side-effects.

6 Discussions on Research Questions

6.1 Research Questions: "What models and description languages are useful for formalization of security properties?" "Which protocol-level secure composition methods are applicable in hardware domain?"

Patrick Schaumont (Virginia Polytechnic Institute – Blacksburg, US), Shivam Bhasin (Nanyang TU – Singapore, SG), Debdeep Mukhopadhyay (Indian Institute of Technology – Kharagpur, IN), Francesco Regazzoni (University of Lugano, CH), Kazuo Sakiyama (The University of Electro-Communications – Tokyo, JP), Dey Soumyajit (Indian Institute of Technology – Kharagpur, IN), and Ingrid Verbauwhede (KU Leuven, BE)

The group discussed these research questions and built the following "Layered Approach to Secure Composition of Electronic Systems" flow in Fig. 1 as a visual answer.

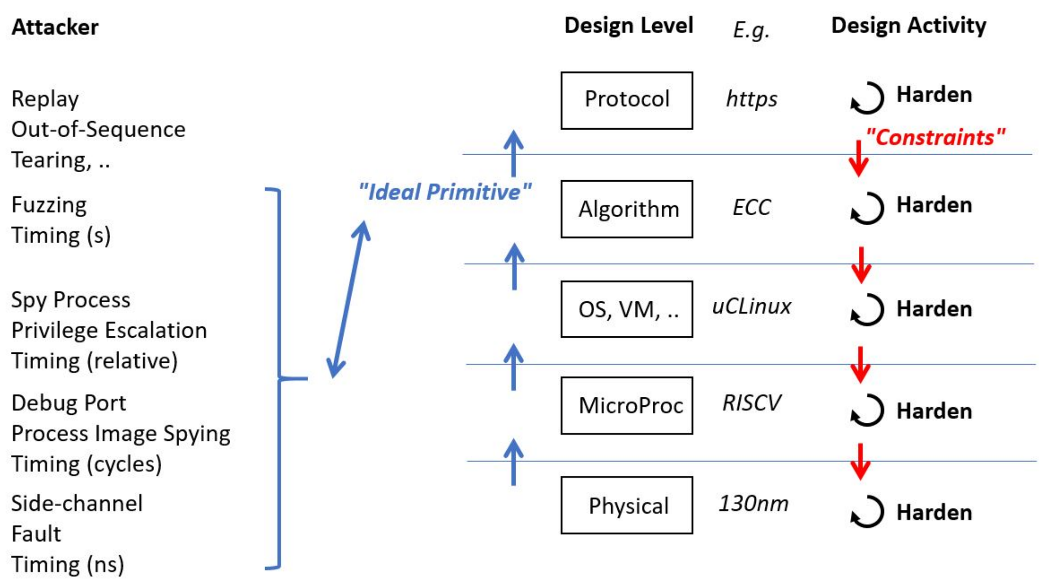

Figure 1 Layered Approach to Secure Composition of Electronic Systems.

6.2 Research Questions: “Can trust start in software, or are hardware roots and anchors of trust indispensable?” “How to bootstrap trust in a distributed hardware system?”

Johann Heyszl (Fraunhofer AISEC – München, DE), Ran Canetti (Tel Aviv University, IL), Fatemeh Ganji (University of Florida – Gainesville, US), Michail Maniatakos (New York University – Abu Dhabi, AE), Marcel Medwed (NXP Semiconductors – Gratkorn, AT), Shahin Tajik (University of Florida – Gainesville, US), and Marten Van Dijk (University of Connecticut – Storrs, US)

1. **There is no HW-free root of trust in embedded computing units**
   A root of trust is the minimal set of required building blocks (hardware and/or software; keys and/or routines; in all cases immutable) at the lowest possible abstraction layer, upon which the system’s security properties rely upon. The system’s higher level security properties (can we trust the system is behaving as intended) are built upon this RoT, similar like proofs are built upon axioms, by using more software as a TCB (secure boot helps to establish a TCB from a RoT). In the case of embedded security, the attacker is powerful and capable of hardware as well as software attacks. The attacker may, *e.g.*, replace code memory contents. Under these circumstances, to the best of our knowledge, there is no way to create such a RoT without the support of the hardware, hence, its manufacturer. Instead, the executing CPU needs to incorporate a minimal hardware RoT within the same chip (at least an executed routine optionally including either a fixed hash or routines for cryptographic verification of MACs and according symmetric or public key). It seems impossible to bootstrap a RoT on a system purely by supplying a piece of SW to be executed by the system. In such cases, the attacker is always able to execute higher-privileged code (*e.g.*, before and after) which is able to manipulate all system’s behaviour, hence, its security.
2. **There is no extension of trust to other executing units without HW based RoTs**
   We consider multiple computing units within a system which are interconnected through (*e.g.*, low-bandwidth) interfaces. To the best of our knowledge, there is no way of bootstrapping or extending trust from a first one containing a hardware RoT to a second one which does not contain a hardware based RoT. This is under the assumption that one unit does not have highest privileged access to the memory of the other unit (and could hence have complete control over the execution of the other). For example, say one unit S1 is a secure element providing key storage and cryptographic operations. S1 can only trust a connected S2 based on the notion that this S2 would itself contain an equivalently trusted RoT. Hence the composed RoT essentially comprises both individual RoT. Otherwise, an attacker may, *e.g.*, replace the software of S2 to another one providing all correct answers to S1 during a phase of trust establishment while running manipulated code before and/or afterwards. Similar examples are TPMs connected to CPUs (TPM 2.0 authenticated connection requires a key stored in the CPU – a RoT).

We discussed whether verifiable computing would help to run a small software RoT on S2 but came to the conclusion that it would be impossible since this software can be, *e.g.* run within a hypervisor by a manipulated version of S2. The topic of asking for software-only

RoT has a relation to white-box crypto which is essentially the attempt to have a secure software-based key storage without hardware support. It seems that similar limitations as with white box crypto would apply (white boxes can be lifted and executed in an emulator, hence need obfuscated interconnect to the application as much as possible) leading to the fact that a software-based RoT would only support trust under reduced attacker assumptions. Also the binding to the hardware, distance bounding in a sense, would be important. PUF instantiations help as long as the integrity of all relevant functionality of S2 (for the system's behaviour) is influencing the PUF response. PUFs used as keys storage can be part of a hardware RoT. Ideally, the challenge is, however, to devise a RoT without hardware reliance.

In summary, a hardware RoT is required in all system parts which are not fully controlled by one instance already containing a hardware RoT.

6.3 Research Question: "Under what circumstances is security additive, and how can this be proven and validated?"

Jean-Luc Danger (Telecom ParisTech, FR), Yaacov Belenky (Intel Israel – Haifa, IL), Elke De Mulder (Rambus – Sunnyvale, US), Elena Dubrova (KTH Royal Institute of Technology – Stockholm, SE), Osnat Keren (Bar-Ilan University, IL), Johannes Mittmann (BSI – Bonn, DE), and Werner Schindler (BSI – Bonn, DE)

A desirable aim is to combine independent security evaluations of component A and of component B (or of countermeasures against attack type A and attack type B, *etc.*) as this would reduce the complexity of the overall evaluation and thereby (hopefully) the probability of evaluation bugs and finally also the costs.

First, such an approach requires the definition of a suitable evaluation metric that is applicable to both the components and to the composed system. An evaluation metric might consider, *e.g.*, the resources required to carry out the most efficient (known) attacks. In evaluations according to the Common Criteria (CC), for example, numeric values for the factors "Elapsed Time", "Expertise", "Knowledge of the TOE" (TOE = target of evaluation), "Window of Opportunity", and "Equipment" are used to derive an attack rating. Moreover, the components or countermeasures need to be 'independent' (to be defined) with regard to security properties.

Countermeasures against (A) side-channel attacks and (B) fault attacks, for example, are usually not independent because the latter often use redundancy to detect successfully induced faults. Redundancy, however, might favour side-channel attacks and thus both sets of countermeasures should not be evaluated independently but jointly. (A positive example might be the verification of an RSA-based signature by the exponentiation with the public exponent to prevent the Bellcore attack.)

Discussions suggested that components should allow such an 'independence splitting' more often than countermeasures against different attack types. Independence between components usually may not be valid in an information theoretical sense. Instead, it might be the conclusion of a careful 'best-practice' evaluation that the components A and B do not interfere in terms of security in an exploitable way. An example might be hardware sensors and the implementation of cryptographic algorithms. In a strict sense, an attacker might gain some local information about the implementation if he is able to identify the position of hardware sensors. In many scenarios this knowledge yet might not allow to mount a successful attack. The evaluator often yet may be faced with 'nested' scenarios where the security evaluation may consider the component A first and then component B under consideration of A.

Finally, we formulate several heuristic criteria, which might justify a 'practical' independence assumption between different components A and B.

The components A and B

- are not nested
- have no common functionality related to the processing of secrets
- do not share resources that are used in the processing of secrets
- have no side-channels, which allow to combine information
- ...

These criteria are in general neither necessary nor sufficient for 'practical independence' of components but should support the decision making process.

6.4 Research Questions: "How can existing hardware fulfill expectations and idealistic assumptions of protocols?" "How to counter possible loss of security due to abstraction of hardware components?"

Elif Bilge Kavun (University of Sheffield, GB), Anupam Chattopadhyay (Nanyang TU – Singapore, SG), Annelie Heuser (IRISA – Rennes, FR), Johann Knechtel (New York University – Abu Dhabi, AE), and Itamar Levi (University of Louvain, BE)

Protocol-level solutions require certain assumptions; however, these assumptions may not always (or even never) be met by hardware. An example to this is the perfect randomness versus device-level randomness: The expected level of randomness by the algorithm/scheme may not be provided by the randomness source on device. A solution to such problems could be:

1. Transfer of requirements to hardware in a way that they are also accessible to system architects/designers,
2. Finding methodologies so that system architects/designers can verify that the requirements are met.

Notion of abstraction is crucial in modern chip design. Hardware-related vulnerabilities are not well-defined in higher abstraction layers. "Secure abstraction" can be a solution – system design must not introduce vulnerabilities by the fact that some relevant lower-level details are invisible/encapsulated on higher layers.

Participants

- Divya Arora
Intel – Santa Clara, US
- Georg T. Becker
ESMT – Berlin, DE
- Yaacov Belenky
Intel Israel – Haifa, IL
- Shivam Bhasin
Nanyang TU – Singapore, SG
- Ran Canetti
Tel Aviv University, IL
- Gaetan Cassiers
University of Louvain, BE
- Anupam Chattopadhyay
Nanyang TU – Singapore, SG
- Jean-Luc Danger
Telecom ParisTech, FR
- Lucas Davi
Universität Duisburg-Essen, DE
- Elke De Mulder
Rambus – Sunnyvale, US
- Elena Dubrova
KTH Royal Institute of Technology – Stockholm, SE
- Yunsi Fei
Northeastern University – Boston, US
- Fatemeh Ganji
University of Florida – Gainesville, US
- Tim Erhan Güneysu
Ruhr-Universität Bochum, DE
- Annelie Heuser
IRISA – Rennes, FR
- Johann Heyszl
Fraunhofer AISEC – München, DE
- Elif Bilge Kavun
University of Sheffield, GB
- Osnat Keren
Bar-Ilan University, IL
- Johann Knechtel
New York University – Abu Dhabi, AE
- Itamar Levi
University of Louvain, BE
- Michail Maniatakos
New York University – Abu Dhabi, AE
- Marcel Medwed
NXP Semiconductors – Gratkorn, AT
- Nele Mentens
KU Leuven, BE
- Johannes Mittmann
BSI – Bonn, DE
- Debdeep Mukhopadhyay
Indian Institute of Technology – Kharagpur, IN
- Paolo Palmieri
University College Cork, IE
- Ilia Polian
Universität Stuttgart, DE
- Milos Prvulovic
Georgia Institute of Technology – Atlanta, US
- Wenjing Rao
University of Illinois – Chicago, US
- Francesco Regazzoni
University of Lugano, CH
- Ahmad-Reza Sadeghi
TU Darmstadt, DE
- Kazuo Sakiyama
The University of Electro-Communications – Tokyo, JP
- Fareena Saqib
University of North Carolina – Charlotte, US
- Patrick Schaumont
Virginia Polytechnic Institute – Blacksburg, US
- Werner Schindler
BSI – Bonn, DE
- Georg Sigl
TU München, DE
- Dey Soumyajit
Indian Institute of Technology – Kharagpur, IN
- Marc Stöttinger
Continental AG – Frankfurt, DE
- Shahin Tajik
University of Florida – Gainesville, US
- Marten Van Dijk
University of Connecticut – Storrs, US
- Ingrid Verbauwhede
KU Leuven, BE