



Publishable Summary of the CACE project

Project number:	216499
Project acronym:	CACE
Project title:	Computer Aided Cryptography Engineering
Start date of the project:	January 1 st , 2008
Funding scheme:	FP7 ICT STREP

Date of the reference Annex I:	September 26 th , 2007
Deliverable:	Publishable Summary of the 2nd period of the CACE Project (as part of the "2nd Periodic Report of the CACE project") <i>V1.1 (Updated Version)</i>
Period covered:	01.01.2009 – 31.12.2009 (M13-M24)
WPs contributing to the deliverable	All
Due date	31 st December 2009 (M24)
Actual submission date	12 th March 2010

Responsible organisation:	Project coordinator: Technikon Forschungsgesellschaft mbH (TEC)
Tel.:	+43 4242 233 55
Fax:	+43 4242 233 55 77
E-mail:	coordination@cace-project.eu
Project website:	www.cace-project.eu

2 Publishable summary



Project Name: CACE
Grant Agreement: 216499
Project Website:
Contact:

Start date: 1 Jan. 2008
Duration: 36 months
<http://www.cace-project.eu/coordination@cace-project.eu>

Mission of CACE

"To enable verifiable secure cryptographic software engineering to non-experts by developing a toolbox which automatically produces high-performance solutions from natural specifications."

The CACE Project

Development of hardware devices and software products is facilitated by a design flow, and a set of tools (e.g., compilers and debuggers), which automate tasks normally performed by experienced and highly skilled developers. However, in both hardware and software examples the tools are generic since they seldom provide specific support for a particular domain. Within the CACE project a toolbox, that will support the specific domain of cryptographic software engineering, will be designed, developed and deployed.

Motivation

Ordinarily, development of cryptographic software is a huge challenge: security and trust is mission critical and modern applications processing sensitive data typically require the deployment of sophisticated cryptographic techniques. The proposed toolbox will allow non-experts to develop high-level cryptographic applications and business models by means of cryptography-aware high-level programming languages and compilers. The description of such applications in this way will allow automatic analysis and transformation of cryptographic software to detect security critical implementation failures, e.g. software and hardware based side-channel attacks when realising low-level cryptographic primitives and protocols. Ultimately, the end result will be better quality and more robust software at a much lower cost; this provides both a clear economic benefit to the European industry in the short term, and positions it better in dealing with any future roadblocks to ICT development in the longer term.

Objectives & Overall Strategy

The CACE project aims to target the lack of support currently offered to cryptographic software engineers. The central objective is the development of a toolbox that supports the production of high quality cryptographic software. The aim is that specific components within the toolbox will address specific software development problems and processes; combined use of the constituent tools is enabled by designed integration between their interfaces. A representative example use of the toolbox might be to develop an online voting system by a natural, high-level description of the system properties. The CACE toolbox would take this description and produce an efficient, executable implementation, which has verifiable security properties both at the semantic and physical levels.

The main technical objectives of CACE are therefore as follows:

- Development of a toolbox, which automates cryptographic tasks and therefore supports developers in implementing cryptographic schemes
- Automatic translation from natural specifications
- Automatic security awareness, analysis and correction
- Automatic optimization for diverse platforms

The CACE project is divided into three stages, which roughly correspond to the tasks of requirements analysis during the first year, development during the second year and evaluation during the third year. Within the first project stage (M01-M12), all languages, compilers and static libraries are specified. Interfaces between WP deliverables are stabilized. Theoretical hurdles are identified and feasible solutions are outlined. Prototypes of lower-level compilers and shared library are released. The second stage (M13-M24) covers the completion of the full implementations of lower-level compilers and shared libraries. Further the theoretical framework is completed and novel results are published. Prototype implementations of higher-level compilers and run-time environments are

released (building on lower-level results). Within the third and last stage (M25-M36), the optimization and extension of lower-level compilers and shared library is completed. Full implementation of higher-level compilers and run-time environments are released.

In order to ensure that these stages are completed in a timely manner, the overall execution of the project can be easily monitored by the following 6 major milestones, constituting central points in the course of the project across the technical work packages:

After 12 months:

- Milestone 1 – System analysis and specification completed.
- Milestone 2 – Proof-of-concept completed.

After 24 months:

- Milestone 3 – Full prototype of CACE toolbox available.
- Milestone 4 – Research results finalised and published.

After 32 months:

- Milestone 5 – System integration completed.

After 36 months:

- Milestone 6 – Final CACE toolbox available, evaluation published.

Figure 1 illustrates the 3 stages of the CACE project and the corresponding major milestones.

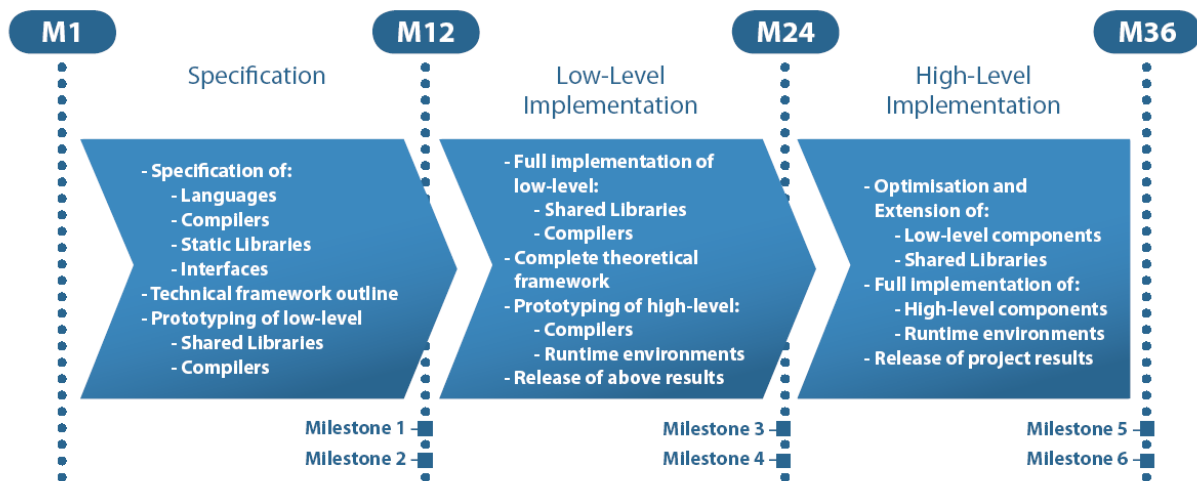


Figure 1: Stages of the CACE project

Technical Approach

The work plan of CACE includes six work packages. Work packages 1-4 deliver the tools, libraries and compilers for the toolbox. Formal verification and validation will be addressed in WP5. Work package 6 provides the organisational framework.

WP1: Automating Cryptographic Implementation

WP2: Accelerating Secure Networking

WP3: Bringing Proofs of Knowledge to Practice

WP4: Securing Distributed Management of Information

WP5: Formal Verification and Validation

WP6: Project Management, Dissemination and Standardisation

Description of the work done and the results until now

The CACE project started in January 2008 and is going to run for 36 months.

For the technical work packages there were two general objectives for the first project year that were pursued and achieved across the work package boundaries. First, the main aim was to define the tools and the interfaces between the tools which constitute the CACE toolbox (e.g. languages, compilers and run-time libraries). Beyond this, the consortium aimed at producing early prototypes of tools and libraries as proof-of-concept and validation of the system analysis and specification results.

Even though these prototypes were for internal use by project partners only, they were essential for the overall consistency of the project. This is particularly important for shared components such as run-time libraries, supporting the work of multiple work packages. One of the main goals of the CACE project in the second year was to release a full prototype of the CACE toolbox which allows for a basic development of cryptographic software. This implied that implementations of the tools within the CACE toolbox have been developed with run-time libraries and interfaces have been matured. In addition, in order to enable the automatic translation from natural specifications, the prototype provided a platform into which later research has been integrated. Further, alongside the development of the toolbox prototype, novel research has been carried out by identifying relevant technical challenges and methods for overcoming them. The aim was to develop, for example, algorithms and technical knowledge that could later be codified. Our current publications of these initial results highlight the CACE project within the community, and foster collaborative work to advance the before mentioned results.

The progress achieved in all work-packages within the first and the second project year is in line with the initial plan and can be summarized as follows:

Within WP1 among other things deliverable D1.1, detailing the language definitions for CAO and qasm tools and enabling the implementation of shared run-time library components to commence, has been prepared. The significance of deliverable D1.1 is that with concrete and well considered language definitions for CAO and qasm, work on the associated compiler tools can accelerate. Based on early requirements analysis, a significant amount of work has gone into development of compiler/interpreter prototypes for CAO and qasm and the result is a solid source code framework which can be built upon. In the second year the main objective was the implementation of the shared run-time library components in CAO and qasm. Here the work focused mainly on tasks 1.2 (tool integration) and 1.3 (tool extension), including the completion of D1.2. The syntax and type system of CAO have been formalized in a Haskell-based type checker and modeled type system in Alloy to permit generation of test cases. Also, various tasks to improve the reproducibility of cache-based side-channel attacks have been carried out. Moreover, a program transformation has been developed that allows a single-source AES program to be compiled into low-footprint and high-performance versions. Within WP1 twelve publications have been released and 2 more are currently pending for review. The CAO portion of WP1 changed direction a little. It was decided to concentrate on only some of the specific research challenges appearing to be important for both, academic and industry. This concerns the effective memorization for AES and automated "masking" of AES.

WP2 produced in the first year 2 deliverables In the first deliverable D2.1 in M06 a prototype implementation of the NaCl library with a C interface has been presented. It contains functionality for both secret-key and public-key cryptography. It does not yet contain networking functionality. The prototype is mostly written in C and the only supported interface language is C. An updated version of the prototype library has been implemented and presented in deliverable D2.2 which was due in M12 and which added networking elements to the prototype produced in D2.1. Further high speed implementations of different cryptosystems (AES, elliptic-curve Diffie-Hellman key exchange) have been written in qasm, demonstrating the usability of WP1 tools in WP2. Further a python wrapper was implemented and an independently developed python NaCl was studied. Also a C++ version of NaCl has been prepared. So far hash functions and AES are implemented. In the second year WP2 concentrated on the enhancement of the prototype created in the first period, with regard to cryptography and networking elements. The main objective was to improve and extend the functionalities of the NaCl library. The objectives matched the tasks defined in the work plan. A functionality assessment has been carried out where, together with WP3 and WP4, the list of required functionalities was extended and a ranking of urgency was developed. The implementation provides more functions as part of the library; in particular the networking part was extended. In the speed evaluation and improvement all cryptographic functions were included in the benchmarking framework eBACS and measured for performance in time and size. During the second year several improvements in speed were noticed, most notably in code geared towards particular platforms. The deliverable D2.3 covering the networking and cryptography library has been finalized until the end of the second period.

The work done by WP3 during the first year was following three major lines of research. One was focusing on reviewing and unifying the existing theory of efficient zero-knowledge proofs of knowledge (ZK-POK) (this work corresponds to Deliverable D3.1). The second line of work was to investigate and implement a first prototype of a compiler to generate implementations of ZK-POK (this

work corresponds to Deliverables D3.2 and D3.3). The third line of work concerned the investigation of formal verification techniques for assuring the correctness of the compiler suite to be developed (this work corresponds to deliverables which are due in a later phase of the project). In the second year WP3 finished 3 deliverables, the first one covered the initial specification of the ZK-POK language and the initial design of the protocol compiler. Then a final report on unified theoretical framework of efficient ZK-POK was compiled. Another important aim of the second project period included the prototype release of the ZK-POK language and runtime compiler, able to process arbitrary ZK-POK programs and to generate the corresponding protocol language code. The existing ZK-PoK compiler ("runtime compiler") has been redeveloped at many places, and a significant functionality ("protocol compiler") was added. Now essentially all practically relevant ZK-PoK protocols are supported. The compiler is publicly available via the Web frontend. Further, a formal verification tool plug-in has been implemented, which, using Isabelle theorem prover, verifies the soundness (proof of knowledge property) of protocols generated by the compiler. The current work aims at integrating this into the compiler. Further, several contributions to the theory of ZK-PoK using Sigma and related protocols have been made, even more than initially expected. Also existing literature has been critically considered, improved and extended (e.g., the work by UNIVBRIS on Groth-Sahai proofs). In this context the publication record consists of 8 publications, and more publications are in the pipeline.

In the first year WP4 has produced an overview of practical applications of secure multiparty computation, and has used this to define the essential properties that protocols should have to support the applications. This will be used to guide the protocol design work in the rest of the project. A domain specific language for secure multiparty computation has been specified, this specification will be the point of departure for the implementation in the coming periods. Further a virtual machine for multiparty computation has been designed, and a prototype has been implemented. Design and implementations of a number of cryptographic protocols have been done, these will be used for supporting the virtual machine. In summary, 4 deliverables have been completed in the first year. Within the second year the focus of WP4 was set on the implementation of the appropriate compilers and interpreters, as well as on the design of a common virtual machine and byte code language. The first implementation of PySMCL, high-level programming language for secure multiparty computation (MPC) has been completed. The implementation of the VIFF framework, the virtual machine for MPC, has also been completed and so has been the first suite of protocols for MPC, integrating code from WP2. One further aspect here are the additional contributions to WP4. During the year 2009, several steps have been taken to bring secure two-party computation towards practice as well: A domain specific language and compiler have been developed that allow automatic generation of semi-private functions. New protocols and building blocks have been developed with suitable security and performance properties, including Secure Evaluation of Private Linear Branching Programs, improved garbled circuit building blocks, and Secure Hamming Distance based Computation. The practical performance of secure two-party computation protocols has been demonstrated by showing how to securely evaluate the AES functionality. For further performance improvements, secure two-party computation protocols can be extended with secure hardware. The combination of the new compiler, the improved building blocks and the highly efficient new protocols allowed to show that several applications such as Secure Classification of Medical ECG Data and Privacy-Preserving Face Recognition are efficient enough to be used in practice today. All in all WP4 has published 20 scientific papers until now.

Within WP5 in Y1 the milestone M5.1 - End-user and partner meeting for security policy identification – has successfully been organised in Porto in July 2008, which allowed the collection of an exhaustive list of types of security requirements needed to be addressed from all partners in the project. Further deliverable D5.1 has been completed within schedule. It contains a taxonomy of security policies that are relevant for the CACE project, and identifies to which CACE tool-box component each policy may apply. This document is an important reference for future work in WP5 (and other work packages) as it delimits the design goals for the formal verification functionality to be developed in years 2 and 3. Additionally a technical report associated with M5.2 was finalized within schedule. It contains the specifications of the formal verification and validation tools that will be developed in WP5. This was an important achievement because the planned progress of WP5 relied on the fact that this could be done by the end of year 1. WP5's major goal for the second project year was the formal specification of language definitions and security policy extensions, and the implementation of prototypes of machine-assisted software verification and validation tools. Another important goal in WP5 was the identification of a NaCl library core that will be verified in the third and final period of the CACE project. Along with these goals, the requirements analysis for formal verification tools has been

completed. Full specifications of the formal verification tools have been released at M18 (D5.2). Early prototypes of the formal verification tools have been made available to the CACE partners at M20. In line with the project plan, WP5 activity currently focused on tool implementation. The definition of the verified NaCl core functionality has been completed at month 24 (M5.3). Three publications on the formal verification of cryptographic code have been published.

As stated above, all the relevant overall project objectives have been successfully achieved and finished; the project made excellent progress and is consistent with the original planning. This paves the way for a successful third project year without any deviations from the original schedule.

The CACE consortium

The EC FP7 project CACE brings together leading companies and academic institutions in the area of cryptography. Together they represent a vertically integrated consortium, with knowledge stretching from the basic research (academic partners) to the design and marketing of products (industrial partners and SMEs). To foster the cooperation with the industry, several experts support the project consortium as Advisory Board members. As a lot of basic research and basic input on cryptography is needed for the CACE project, seven European universities as well as one Asian university participate in this project. Furthermore, a global industrial player and three SMEs contribute with their expertise and knowledge to this project. The 12 project partners are located in eight European countries (Austria, Denmark, Finland, Germany, United Kingdom, Netherlands, Portugal, and Switzerland) and one Asian country (Israel).



Figure 2: The CACE consortium at the Kick-off meeting in Bochum in February 2008

CACE Disclaimer

All public information will be marked with the following CACE project disclaimer:

The information in this document is provided "as is", and no guarantee or warranty is given that the information is fit for any particular purpose. The user uses the information at its sole risk and liability.