World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:1, No:9, 2007

Formal Analysis of a Public-Key Algorithm

Markus Kaiser, Johannes Buchmann

***Abstract*— In this article, a formal specification and verification of the Rabin public-key scheme in a formal proof system is presented. The idea is to use the two views of cryptographic verification: the computational approach relying on the vocabulary of probability theory and complexity theory and the formal approach based on ideas and techniques from logic and programming languages. A major objective of this article is the presentation of the first computer-proved implementation of the Rabin public-key scheme in Isabelle/HOL. Moreover, we explicate a (computer-proven) formalization of correctness as well as a computer verification of security properties using a straight-forward computation model in Isabelle/HOL. The analysis uses a given database to prove formal properties of our implemented functions with computer support. The main task in designing a practical formalization of correctness as well as efficient computer proofs of security properties is to cope with the complexity of cryptographic proving. We reduce this complexity by exploring a light-weight formalization that enables both appropriate formal definitions as well as efficient formal proofs. Consequently, we get reliable proofs with a minimal error rate augmenting the used database, what provides a formal basis for more computer proof constructions in this area.**

***Keywords*— public-key encryption, Rabin public-key scheme, formal proof system, higher-order logic, formal verification.**

I. Introduction

CRYPTOGRAPHIC algorithms are crucial security tools for guaranteeing secrecy of sensitive data. Moreover, their area of application is widely spread and growing. Consequently, in domains where security is a major issue, as in electronic commerce or electronic voting, the need of confidence in correct implementations is increasing dramatically. Leaks in security, for example in a communication between banks, could be very costly and should be prevented. Consequently, the use of correct implementations of cryptographic algorithms is essential for security, what means that implementing cryptographic algorithms must be done with extreme care. A verification of these algorithms with reliable methods is a central objective.

A verification of an algorithm can prove crucial properties or in the optimal case all relevant facts needed for its application. Verification approaches for cryptographic primitives have been directed in two distinct directions: the computational approach relying on the vocabulary of probability theory and complexity theory and the formal approach based on ideas and techniques from logic and programming languages (compare [1], where a formal approach and a computational approach as well as their combination are given, an example involving Isabelle/HOL is presented in [9]). A proof of functional correctness of a given implementation can be achieved by formal verification. But, in the area of cryptography the used algorithms are often very complex, hence formal verification is a big challenge and therefore only achieved with much effort.

The authors a re with the Technische Universität Darmstadt, 64289 Darmstadt, Germany. This work was partially funded by the German Federal Ministry of Education and Technology (BMBF) in the framework of the Verisoft project under grant 01 IS C38. The responsibility for this article lies with the authors.

We show that a formal verification with computer support in the area of cryptography is possible by exploring a light-weight formalization that enables both appropriate formal definitions as well as efficient formal proofs. Our formal analysis (functional correctness as well as arguments concerning security of the given implementation) yields the first computer-proved implementation of the Rabin public-key scheme in Isabelle/HOL. Consequently, we get reliable proofs with a minimal error rate augmenting the used database, what provides a formal basis for more computer proof constructions in this area.

A central objective of our research is a combination of the formal approach and the computational approach. This article is part of an effort to unify the formal and the computational views of cryptographic verification. The idea is to unify both verification approaches in cryptography by embedding one into the other: formalizing computational aspects as well as their computer verification embed the computational approach in the formal approach. We obtain a formal verified computational description of a cryptographic primitive, that can be used in practice (compare II-D). More specifically, this work continues our recent work that provides useful formal descriptions of mathematical background and cryptographic algorithms computer-proven with Isabelle/HOL (compare [4] and [3]). Besides, the formally verified cryptographic algorithms are components of a cryptographic client.

This article is organized as follows. We review the Rabin public-key scheme originally introduced in [7] (compare II). Therefore, we explicate the original definition of the Rabin public-key function (compare II-A), illustrate a corresponding decryption algorithm (compare II-B), and outline the application of the Rabin function for signature (compare II-C). After that, we show how the Rabin functions may be used in practice (compare II-D). In III Isabelle/HOL is outlined (compare III-A). Furthermore, in III-B, we introduce a computation model that allows formal verification of computational effort of a function. III-C provides an overview of the formalization and verification. Furthermore, we explore a formal description and verification of the Rabin functions (compare IV). We introduce our formal specification (compare IV-A) and a formal proof of its correctness (compare IV-B). Moreover, we computer-prove properties concerning security (compare IV-C). Therefore, we apply a formal version of our computation model. V provides a conclusion.

II. Rabin Public-Key Scheme

In 1979 Michael Rabin introduced in his well known paper ([7]) a new class of public-key functions, where a number

$n = p \cdot q$ with large primes p and q is involved. The number n is the public key, moreover p and q are the private keys that can be used for signing and decryption. A main result given in [7] is that for any given number n the efficient inversion of the function $y = E_n(x)$ that is described below, for even a small percentage of the values of y implies the efficient factorization of n.

A. Public-Key Function

For a given number $n = p \cdot q$ that is a product of two large prime numbers p, q, and for given b, $0 \leq b < n$ the function $E_{n,b}(x)$ is defined for $0 \leq x < n$ by $E_{n,b}(x) \equiv x(x+b) \bmod n$, where $E_{n,b}(x) < n$. In the case that $b = 0$, $E_{n,b}(x) = E_{n,0}(x) \equiv x^2 \bmod n$. Below, the Rabin public-key function E with $b = 0$ is refered by $A_{encrypt}$.

B. Decryption

An inversion of $E_{n,0}(x)$ can be calculated by Algorithm (1) (compare [7]) where the four square roots of c are obtained by the computation of solutions $\bmod p$ and $\bmod q$ that are composed to solutions $\bmod n$. For $a_0 \in \mathbf{Z}$ with $a_0 \equiv 1 \bmod p$ and $a_0 \equiv 0 \bmod q$, $a_1 \in \mathbf{Z}$ with $a_1 \equiv 1 \bmod q$ and $a_1 \equiv 0 \bmod p$ and for m_p and m_q with $c \equiv m_p^2 \bmod p$ and $c \equiv m_q^2 \bmod q$, $z_0 = a_0 m_p + a_1 m_q$ solves the congruence $c \equiv z^2 \bmod n$ and $x_0 = z_0 \bmod n$ is a solution $< n$.

For a fixed prime r and a quadratic residue $d \bmod r$ we write $\sqrt{d}$ for any of the two integer numbers with $(\sqrt{d})^2 \equiv d \bmod r$. Moreover $-\sqrt{d}$ denotes $r - \sqrt{d}$.

A computation of a number x with

$$x^2 + bx \equiv c \bmod r \tag{1}$$

can be realized by the extraction of square roots $\bmod r$.

For $d = b/2 \bmod r$ a computation of x given by (1) can be denoted by $(x+d)^2 \equiv c + d^2 \bmod r$ or by $x = -d + \sqrt{c+d^2}$ and $x = -d - \sqrt{c+d^2}$, respectively. Consequently, (1) (an extraction of square roots $\bmod r$) can be used to compute x).

In the following we concentrate on the case that $p \equiv q \equiv 3 \bmod 4$ ($p = 4n_p - 1$ and $q = 4n_q - 1$).

Algorithm (1).

Input: $c \equiv x^2 \bmod n$, p, q with $p \equiv q \equiv 3 \bmod 4$, $a_0 \in \mathbf{Z}$ with $a_0 \equiv 1 \bmod p$ and $a_0 \equiv 0 \bmod q$, $a_1 \in \mathbf{Z}$ with $a_1 \equiv 1 \bmod q$ and $a_1 \equiv 0 \bmod p$

Output: $0 \leq x_i < n$, $1 \leq i \leq 4$ with $E_{n,0}(x_i) = c$

1) $m_p = c^{\frac{p+1}{4}} \bmod p$
2) $m_q = c^{\frac{q+1}{4}} \bmod q$
3) $x_1 = (a_0 m_p + a_1 m_q) \bmod n$
4) $x_2 = (a_0 m_p - a_1 m_q) \bmod n$
5) $x_3 = n - x_1 \bmod n$
6) $x_4 = n - x_2 \bmod n$
7) return (x_1, x_2, x_3, x_4)

C. Application for Signature

The application of the Rabin function E (compare II-A) for signature needs two large prime numbers p and q that can be produced by a primality test (compare [3]). As before, (n, b), where $n = p \cdot q$ and $0 \leq b < n$, is public, but p and q remain private.

Algorithm (2).

Input: message $x_0 \in \mathbf{N}$, public key (n, b), prime numbers p and q, compression function h

Output: signature (z, x) of x_0

1) choice of a random number z with length k
2) $x_1 = x_0 \| z$
3) $c = h(x_1)$ with binary length of $c < n$
4) if there is a solution x of $x(x+b) \equiv c \bmod n$ then return (z, x) else 1

For given (n, b), (z, x), h the number $c = h(x \| z)$ as well as the congruence $x(x+b) \equiv c \bmod n$ can be computed or tested, respectively (notation: $A_{verification}$, Algorithm (1): $A_{decrypt}$, Algorithm (2): $A_{digital}$).

D. Use of the Rabin Functions

We remark how the given algorithms can be used in practice. The idea is that a message is signed (encryption) in order to determine the correct square root as plaintext (decryption).

Encryption:

For $n = p \cdot q$, $b = b_0 \cdot b_1$, where p, q, b_0 and b_1 are large prime numbers, $m \in \mathbf{Z}_n$, the encryption (c, z, x, h) of m is computed as follows.

- $A_{encryption}(m, n) = c$
- $A_{digital}(m, n, h, b_0, b_1) = (z, x)$
- encryption: (c, z, x, h)

Decryption:

- $A_{decryption}(c, z, x, h, p, q) = (x_1, x_2, x_3, x_4)$
- For ($j \in \{1, 2, 3, 4\}$): if $A_{verification}(x_j, z, x, b, h) = 1$: x_j

In the following we discuss our formalized version of the Rabin functions (compare II). Therefore we introduce a formal description of these functions that provide a formal base for a computer analysis of the Rabin encryption scheme as well as the Rabin signature scheme. Moreover, we show that computer verification of these schemes is practical.

III. Overview of the Formalization

We illustrate the formal proof system Isabelle/HOL and explore a computation model with respect to efficient algorithms. Moreover, we give an overview of the formalization and verification of the Rabin functions reviewed in II.

A. Isabelle/HOL

Formal proof systems provide computer support for formal verification. But the correctnes of a formal proof using a formal proof system relies on the correctness of the applied computer system. A formal proof system works on automated

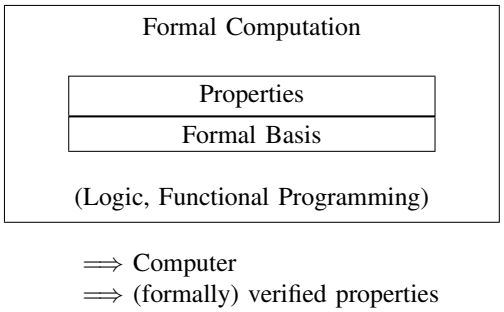

Fig. 1. Construction pattern in Isabelle/HOL

or interactive proof constructions. In our work, we use the interactive formal proof system Isabelle/HOL (proof assistant for higher-order logic, which can be used for interactive proof constructions, formal specifications, as well as verification in higher-order logic and functional programming).

The formal proof language of Isabelle/HOL that consists of higher-order logic and functional programming, is used to give definitions and lemmata, which are based on a large database. These definitions and (proven) lemmata can be used to prove further lemmata and theorems, which results in an augmented data base for the purpose of building up new theories (compare to Figure III-A).

In the lines below some Isabelle/HOL code examples are explained.

Remark: (Isabelle/HOL code)

- int: integer number datatype
- num: natural number datatype
- bool: boolean datatype
- (x::int): variable of type int
- f x or (f x): function f applied to x
- **consts** example :: "[int, num] ⇒ bool"; (declaration of a function/predicate)
- **defs** example_def: "example i n ≡ i < (int n)"; (definition of a function/predicate with an application of function int)
- **lemma** "(x::int) < (y::int) ⟹ x ≤ y"; (computer lemma)
- ^z: exponent z
- zprime: set of (integer) prime numbers
- zgcd: (integer) greatest common divisor

More information about Isabelle/HOL (that is successfully applied in the *Verisoft project* ([5]) are given in [6], which describes constructions with this tool. A further useful reference is [10]. There, parts of the large database are mapped. Besides [10] contains other references about Isabelle/HOL.

B. Computation Model

In this chapter, concepts of complexity theory with respect to cryptography are given. A major term concerning public-key cryptography is *polynomial complexity*. In cryptography, for encryption, decryption, signature, verification and other primtives, efficient algorithms are needed with respect to practical applications. But, breaking a cryptographic algorithm should remain hard (not efficient). In constucting new cryptographic algorithms, hard problems that are hard on average are needed. A definition of efficient can be given by polynomial complexity described by the complexity class P or by Landau-notation (O-notation).

A decision problem B is a problem with *yes* or *no* as answers (input x, where x is coded in a way that allows a measurement of the size of x, for example a binary code with length $|x| = n$). Usually, a given computation problem can be mapped to a decision problem.

P is the class of all decision problems with property (P).

(P) $B \in P$ if there is an algorithm that answers *yes* (*no*) for any input x where the answers is *yes* (*no*), where time is measured in terms of bit operations that are bounded by a polynomial function of $|x|$.

NP is the class of all decision problems with property (NP).

(NP) $B \in P$ if for any input x where the answers is *yes* there is an algorithm that can verify this fact in polynomial time. If the answers is *no*, termination of this algorithm is not certain, but in the case of termination, the answers is *no*.

A decision problem B is NP-*complete* if all problems in NP can be reduced to B in polynomial time: $B \in P \Longrightarrow P = NP$.

In cryptography it is mostly assumed that $P \neq NP$. Many hard problems on that public-key cryptography is based (integer factoring, discrete logarithm problem) lie in NP but are not related to a NP-complete problem. A main aspect of using these problems in cryptography is that they are hard on average ([8], p. 331 - 335, [2]).

Complexity can be given in another way, by O-notation.

For $k \in \mathbf{N}$, $X, Y \subseteq \mathbf{N}^k$ and $f : X \to \mathbf{R}$, $g : Y \to \mathbf{R}$, $f \in O(g)$ if there are $b, c \in \mathbf{R}$ with $b, c > 0$ where for all $x = (x_1, \ldots, x_k) \in \mathbf{N}^k$ with $x_i > b$, $1 \leq i \leq k$, (1) and (2) hold.

(1) $x \in X \cap Y$ ($f(x)$, $g(x)$ are definied)
(2) $f(x) \leq cg(x)$

([2], p. 5)

Example.

- $f \in O(1)$ if $g(x) = 1$ for all $x \in X \cap Y$
- $f \in O(x^k)$, $x, k \in \mathbf{N}$

A cryptographic algorithm A with input $x \in \mathbf{N}^k$ ($k \in \mathbf{N}$) has *polynomial complexity* if there are $e_1, \ldots, e_k \in \mathbf{N}$ where time complexity *cp* of A is $O(|x_1|^{e_1} \cdot \ldots \cdot |x_k|^{e_k})$. Algorithm A is regarded as *efficient* if $cp(A) \in O(|x_1|^{e_1} \cdot \ldots \cdot |x_k|^{e_k})$. In cryptographic practice, O-constants as well as $e_1, \ldots, e_k$ have to be small if an algorithm is regarded as efficient. A cyrptographic algorithm has to be efficient in functionality, but breaking it must not be efficient.

Example.

$a : \mathbf{Z}^2 \to \mathbf{Z}$ with $a(x, y) = x + y$ ($- : \mathbf{Z}^2 \to \mathbf{Z}$ with $-(x, y) = x - y$) is of time complexity $O(\max\{|x|, |y|\})$, $mp : \mathbf{Z}^2 \to \mathbf{Z}$ with $mp(x, y) = x \cdot y$ is of time complexity $O(|x||y|)$, mod : $\mathbf{Z}^2 \to \mathbf{Z}$ with $\text{mod}(x, y) = x \bmod y$, div : $\mathbf{Z}^2 \to \mathbf{Z}$ with $\text{div}(x, y) = x/y$ are of time complexity $O(|x||y|)$, exp :

$\mathbf{Z}^2 \to \mathbf{Z}$ with $exp(x, y) = x^y$ is efficient, a computation of the greatest common divisor of x and y ($\gcd(x, y)$) is efficient ([2], p. 7 - 10, p. 37, 38)

With respect to the Rabin public-key algorithm, that means, the computation of $f(x) = x^2 \bmod n$ is efficient where $n = p \cdot q$. Moreover, extracting x from $f(x)$ with knowledge of p, q is efficient.

A main aim of this article is to keep formal verification easy, but a formalization and verification of P, NP, and O-notation, respectively, is impractical in order to reach this main aim. Consequently, this article explicates the definition of a straight-forward computation model.

For any $z \in \mathbf{Z}$, there are functions

- $a_z^{(0)} : \mathbf{Z} \to \mathbf{Z}$ with $a_z^{(0)}(x) = z + x$
- $mp_z^{(0)} : \mathbf{Z} \to \mathbf{Z}$ with $mp_z^{(0)}(x) = z \cdot x$
- $\mathrm{mod}_z^{(0)} : \mathbf{Z} \to \mathbf{Z}$ with $\mathrm{mod}_z^{(0)}(x) = x \bmod z$, $\mathrm{div}_z^{(0)} : \mathbf{Z} \to \mathbf{Z}$ with $\mathrm{div}_z^{(0)}(x) = x/z$
- $\gcd_z^{(0)} : \mathbf{Z} \to \mathbf{Z}$ with $\gcd_z^{(0)}(x) = \gcd(x, z)$

and

- $a_z^{(1)} : \mathbf{Z} \to \mathbf{Z}$ with $a_z^1(x) = x + z$
- $mp_z^{(1)} : \mathbf{Z} \to \mathbf{Z}$ with $mp_z^1(x) = x \cdot z$
- $\mathrm{mod}_z^{(1)} : \mathbf{Z} \to \mathbf{Z}$ with $\mathrm{mod}^{(1)}z(c) = z \bmod c$, $\mathrm{div}_z^{(1)} : \mathbf{Z} \to \mathbf{Z}$ with $\mathrm{div}_z^{(1)}(c) = z \div c$
- $\gcd_z^{(1)} : \mathbf{Z} \to \mathbf{Z}$ with $\gcd_z^{(1)}(c) = \gcd(z, c)$

that are efficient. In this context, more efficient functions, for example $-_z : \mathbf{Z} \to \mathbf{Z}$ with $-_z(x) = z - x$ and $exp_z : \mathbf{Z} \to \mathbf{Z}$ with $exp_z(x) = z^x$ for any $z \in \mathbf{Z}$, can be used.

For any $f : \mathbf{Z} \to \mathbf{Z}$ with $cp(f) \in O(|x|^{e_0})$ and $g : \mathbf{Z} \to \mathbf{Z}$ with $cp(g) \in O(|x|^{e_1})$, $cp(f), cp(g) \in O(|x|^{e_{max}})$ where $e_{max} = \max\{e_0, e_1\}$: for f, g there are $b_0, c_0, b_1, c_1 \in \mathbf{R}$ with $b_0, c_0, b_1, c_1 > 0$ where for all $x \in \mathbf{N}$ with $x > b_0$, $cp(f) \leq c_0|x|^{e_0}$ and for all $x \in \mathbf{N}$ with $x > b_1$, $cp(g) \leq c_1|x|^{e_1}$. Therefore $cp(f), cp(g) \leq c_{max}|x|^{e_{max}}$, $c_{max} = \max\{c_0, c_1\}$ for all $x \in \mathbf{N}$ with $x > b_{max} = \max\{b_0, b_1\}$. Furthermore, $cp(f \circ g) \in O(|x|^{e_{max}})$ ($cp(f \circ g) \leq c_{max}|\max\{x, g(x)\}|^{e_{max}}$, $b_0 = b_0 = 1$).

A computation model for efficiency computing can be defined by the following calculus.

$\mathcal{C}$ is based on function *efficient* (compare to Figure III-B):

- $Z = \{f : \mathbf{Z} \to \mathbf{Z}\}$
- efficient : $Z \to \{0, 1\}$
- For all integer numbers $z \in \mathbf{Z}$, $E(z) = \{a_z^{(0)}, mp_z^{(0)}, \mathrm{mod}_z^{(0)}, \mathrm{div}_z^{(0)}, \gcd_z^{(0)}, a_z^{(1)}, mp_z^{(1)}, \mathrm{mod}_z^{(1)}, \mathrm{div}_z^{(1)}, \gcd_z^{(1)}\}$
- For all $z \in \mathbf{Z}$ and for all $f \in E(z)$, efficient$(f) = 1$ can be derived by $\mathcal{C}$
- For all $z \in \mathbf{Z}$ and for all $f, g \in E(z)$: from efficient$(f) = 1$ and efficient$(g) = 1$, efficient$(f \circ g) = 1$ can be derived by $\mathcal{C}$

In general, the only inference rule of $\mathcal{C}$ indicates that the composition of efficient functions remains efficient. A formal application of $\mathcal{C}$ is explained in IV-C.

C. Overview of the Formalization and Verification

Formal functions beeing useful for realizing the Rabin encryption scheme as well as the Rabin signature scheme are easy to define. They can directly be compiled from the functions given in II. We implemented the functions *modq*, *mod_prim*, *mod_key0*, and *mod_key1* as basic elements of our formal description. These formalized functions correspond to the functions $A_{encrypt}$, $A_{decrypt}$, $A_{digital}$, and $A_{verification}$ (function E with $b = 0$, Algorithm (1), Algorithm (2)). In the definitions of these functions the parameter b is considered as 0. Furthermore, we consider prime numbers p, q, where $p \equiv 3 \bmod 4$ and $q \equiv 3 \bmod 4$. $A_{encrypt}$, $A_{decrypt}$, $A_{digital}$, and $A_{verification}$ can be formally specified as follows.

For $n \in \mathbf{Z}$, $n = p \cdot q$ and $m \in \{1, \ldots, n-1\}$:

- $A_{encrypt}(m, n) = m^2 \bmod n$
- $A_{decrypt}(c, p, q) = (m_1, m_2, m_3, m_4)$ where $c = m_j^2 \bmod (p \cdot q)$ for $j \in \{1, 2, 3, 4\}$, $|\{m_1, m_2, m_3, m_4\}| = 4$ or $A_{decrypt}(c, p, q) = x$ where $c = x^2 \bmod (p \cdot q)$ (compare to Figure III-C)
- $A_{digital}(m, n, h, p, q) = (z, x)$ where $x \in \mathbf{Z}$ random number with length $k \in \mathbf{N}$, and $h(m, x) = z^2 \bmod n$ (h compression function with output length $< n$)
- $A_{verification}(m, z, x, n, h) = 0$ or 1, whether $h(m, x) \neq z^2 \bmod n$ or $h(m, x) = z^2 \bmod n$

In this article, a formal verification of the functions *modq*, *mod_prim*, *mod_key0*, and *mod_key1* provides computer-proven lemmata describing functional correctness as well as arguments concerning security.

Our formal analysis is a light-weight verification, since in some cases needed mathematical foundations are imported to the computer system as facts. If these (mathematical) facts are given as lemmata, they could be proven formally. Consequently, this article describes a reliable formal verification from the area of cryptography and provides computer support in education as well as in research in this area.

$$\boxed{\text{efficient}(f) = 1,\ z \in \mathbf{Z},\ f \in E(z)}$$

$$\boxed{\frac{\text{efficient}(f) = 1, \text{efficient}(g) = 1}{\text{efficient}(f \circ g)},\ f, g \in E(z)}$$

Fig. 2. Computation model for efficiency $\mathcal{C}$

$$\boxed{c = m^2 \bmod n,\ a_0 \cdot p + a_1 \cdot q = 1}$$

$z_p = c^{\frac{p+1}{4}} \bmod p$
$z_q = c^{\frac{q+1}{4}} \bmod q$
$m_1 = (a_0 m_p + a_1 m_q) \bmod n$
$m_2 = (a_0 m_p - a_1 m_q) \bmod n$
$m_3 = n - m_1 \bmod n$
$m_4 = n - m_2 \bmod n$
return (m_1, m_2, m_3, m_4)

Fig. 3. Algorithm (1)

IV. Formal Analysis of the Rabin Functions

With respect to the main objective this article from I, we provide a formal analysis of the Rabin functions given in II

and III-C in the formal proof system Isabelle/HOL. The idea is to unify the two views of cryptographic verification by a construction of a Isabelle/HOL theory involving both, functional correctness as well as computational aspects. Therefore we discuss a formal description of these functions. Besides, we explore computer proofs of properties expressing aspects of functional correctness or security. In order to achieve efficient (practical) computer verification, a formal compilation of the computation model from III-B is applied.

A. Implementation

We present Isabelle/HOL functions that are based on the Rabin functions. A formal function is given by its declaration and its definitions as a constant.

Function modq

Function *modq*, given as follows, realizes the algorithm $A_{encrypt}$ and can be used to represent the algorithm $A_{verification}$. Fundamental datatype is *int* (type of integer numbers).

consts modq :: "[int, int] ⇒ int";

defs modq_def: "modq m n ≡ (m∗m) mod n";

This function implements the squaring of an integer m modulo n.

Function mod_prim

Function *mod_prim* can be used for decryption and signing. While a ciphertext is of type *int*, a prime is of type *num* and must in some cases be mapped to type *int* by the function *int*.

consts mod_prim :: "[int, num] ⇒ int";

defs mod_prim_def: "mod_prim c k ≡ c^ ((k+1) div 4) mod (int k)";

Function *mod_prim* realizes the computation of a square root modulo a prime number. That means, for a prime number k, this function computes a square root of c mod k, if c is a quadratic residue mod k and $k \equiv 3 \bmod 4$ (the other square root of c mod k is (k−(mod_prim c k)) mod (int k)).

Function mod_key0 and Function mod_key1

Function *mod_key0* and function *mod_key1* using function *mod_prim* realize algorithm $A_{decrypt}$. Besides, they can be used for a realization of algorithm $A_{digital}$. The main datatype of the given computation is *int*. Primes are of type *num*.

(1)

consts mod_key0 :: "[int, int, num, num, int] ⇒ int";
defs mod_key0_def:
"mod_key0_def x z p q c ≡ ((x∗(int p)∗(mod_prim c q))+((z∗(int q)∗(mod_prim c p))";

(2)

consts mod_key1 :: "[int, int, num, num, int] ⇒ int";
defs mod_key1_def:
"mod_key1_def x z p q c ≡ ((x∗(int p)∗(mod_prim c q))−(z∗(int q)∗(mod_prim c p))";

These functions compute a composition of a square root of c mod p and a square root of c mod q to a square root of c mod $p \cdot q$, if c is a quadratic residue mod $p \cdot q$.

Formal Verification

A formal verification of these functions under given preconditions in Isabelle/HOL is realized by a computer verification of appropriate lemmata with consequences determined by postconditions. A central aspect of this article is to provide a corresponding computer verification.

As mentioned before, we determine $n \in \mathbf{Z}$ given by the product of two primes p, q, that is $n = p \cdot q$. Furthermore, $m \in \{1, \ldots, n-1\}$ (message space). These conditions guarantee that *modq* implements the Rabin public-key encryption $A_{encrypt}(m, n) = m^2 \bmod n$. Moreover, the Rabin decryption algorithm $A_{decrypt}(j, c, p, q) = m_j$ where $m_j \in \{x \in \mathbf{Z}_n : c = x^2 \bmod n\}$ for $j \in \{1, 2, 3, 4\}$ (or $A_{decrypt}(c, p, q) = (m_1, m_2, m_3, m_4)$) can be realized by *mod_prim*, *mod_key0* and *mod_key1*.

If h is a compression function with output length $< n$, the Rabin algorithm for digital signature $A_{digital}(m, n, h, p, q) = (z, x)$ where $x \in \mathbf{Z}$ random number with length $k \in \mathbf{N}$, $h(m, x) = z^2 \bmod n$ as well as the Rabin algorithm for verification $A_{verification}(m, z, x, n, h) = 0$ or 1, whether $h(m, x) \neq z^2 \bmod n$ or $h(m, x) = z^2 \bmod n$ can be realized by a composition of *modq*, *mod_prim*, *mod_key0* and *mod_key1*.

The square roots of c mod $p \cdot q$ are:

- (mod_key0 x z p q c) mod n
- (n − (mod_key0 x z p q c)) mod n
- (mod_key1 x z p q c) mod n
- (n − (mod_key1 x z p q c)) mod n

Beside the formal definitions of the Rabin functions, a formulation (with proofs) of computer lemmata expressing interesting properties of these functions, is of great relevance with respect to a formal analysis. A property of interest in this context provides information concerning the functional correctness of a given implementation or the security of used functions. Both aspects are illustrated below.

B. Correctness

In general, correctness of cryptographic encryption/decryption functions is formulated as follows.

$$A_{decrypt}(A_{encrypt}(x)) = x$$

But, the output of the Rabin decryption function is not unique, what implies that this formulation is not used in this paper. Therefore, we prove a different property to verify the functional correctness of the Rabin encryption/decryption scheme.

$$A_{encrypt}(A_{decrypt}(A_{encrypt}(x))) = A_{encrypt}(x)$$

This correctness property can be proven in the case of the Rabin encryption/decryption scheme. A more precise formulation is given below.

Correctness Property: If p and q are prime numbers, where $p \equiv 3 \bmod 4$ and $q \equiv 3 \bmod 4$, $n = p \cdot q$, and $x, z \in \mathbf{Z}$ with

$x \cdot p + z \cdot q = 1$, then for $m \in \{1, \ldots, n-1\}$ with $\gcd(m, n) = 1$, $(A_{decrypt}(j, m^2 \bmod n, p, q)^2 \bmod n = m^2 \bmod n$, where each $A_{decrypt}(j, m^2 \bmod n, p, q) \in \{i : i^2 \bmod n = m^2 \bmod n\}$ $(j \in \{1, 2, 3, 4\})$.

A formal compilation of this property is given below.

lemma "[|prime p; prime q; p mod 4 = 3; q mod 4 = 3; ¬((int p) dvd (m::int)); ¬((int q) dvd (m::int)); 0 <= m; ¬((int p) dvd (m·m) mod ((int p)·(int q))); ¬((int q) dvd (m·m) mod ((int p)·(int q))); ¬(2 dvd (int p)); ¬(2 dvd (int q)); ¬((int p) dvd (int q)); ¬((int q) dvd (int p)); x·(int p) + z·(int q) = 1|]
⟹
((modq (mod_key0 x z p q ((m·m) mod ((int p)·(int q)))) ((int p)·(int q)))) = ((modq m ((int p)·(int q))))";

lemma "[|prime p; prime q; p mod 4 = 3; q mod 4 = 3; ¬((int p) dvd (m::int)); ¬((int q) dvd (m::int)); 0 <= m; ¬((int p) dvd (m·m) mod ((int p)·(int q))); ¬((int q) dvd (m·m) mod ((int p)·(int q))); ¬(2 dvd (int p)); ¬(2 dvd (int q)); ¬((int p) dvd (int q)); ¬((int q) dvd (int p)); x·(int p) + z·(int q) = 1|]
⟹
((modq (mod_key1 x z p q ((m·m) mod ((int p)·(int q)))) ((int p)·(int q)))) = ((modq m ((int p)·(int q))))";

For a better understanding of these lemmata we skipped the datatypes of the variables, furthermore we write · for a multiplication.

C. Further Properties

Beside the correctness property we computer proved further interesting lemmata given below. A main result are computer lemma expressing security arguments. More interesting lemmata are given below.

Decryption and Factoring:

A main result given in [7] is that for any given number n the efficient inversion of the function $y = E_{n,0}(x)$, for even a small percentage of the values of y implies the efficient factorization of n. A paper proof of this result is given in [7].

We explored a computer analysis of this interesting mathematical property illustrated below. That means we formulate appropriate formal lemmata proven with computer support. For a better understanding of these formal lemmata, we give these properties in a less formal way. A main aspect of this formal verification is the easy construction compared with a complete formal verification approach.

A basis for our computer proof construction are Property (1) and Property (2).

Property (1). If algorithm A_0 computes p from input n efficiently, and algorithm A_1 computes q from input n efficiently, where $n = p \cdot q$ and p, q are prime, and $x \cdot p + z \cdot q = 1$ for $x, z \in \mathbf{Z}$, then a computation of
modq(mod_key0$(x, z, p, q,$ modq$(m, n)), n))$ = modq(m, n) and
modq(mod_key1$(x, z, p, q,$ modq$(m, n)), n))$ = modq(m, n)
is efficient.

(A computation of p, q implies the recoverage of a message m.)

Property (2). If $n = p \cdot q$, where p, q are prime and $p \equiv 3 \bmod 4$, $q \equiv 3 \bmod 4$, for $m, x \in \{1, \cdots, n-1\}$ and $\gcd(m, n) = 1$, $\gcd(x, n) = 1$, $c = m^2 \bmod n$, $c = x^2 \bmod n$, $m \bmod p = x \bmod p$, $m \bmod q = (q-x) \bmod q$ $m \bmod p = (p-x) \bmod p$, $m \bmod q = x \bmod q$ and algorithm A computes m from input c efficiently, then a computation of
1) $\gcd(A(c) - x, n) = p$, $n / \gcd(A(c) - x, n) = q$
2) $\gcd(A(c) - x, n) = q$, $n / \gcd(A(c) - x, n) = p$
is efficient.

(A computation of a plaintext from a ciphertext c implies the factorization of n.)

A computer compilation of Property (1) and Property (2) can be done by the following lemmata. Complexity of the involved functions is handled by an extra theory. That means, we implemented a predicate *efficient* that holds when a function f can be computed efficiently ($f \in P$) in a defined way. *efficient* is given in a minimal way in order to keep formal proving practical.

Computation Model:

We express computational complexity of the formalized Rabin functions by a formal light-weight computational model (**??**). This computational model is very easy but effective. We formalized a predicate *efficient* that holds when a function f can be computed efficiently ($f \in P$) in a defined way. *efficient* is given in a minimal way in order to keep formal proving practical.

efficient: $(\mathbf{Z} \longrightarrow \mathbf{Z}) \longrightarrow \{0, 1\}$,

where *efficient*$(f) = 1$
if $f \in \{f_{add,z}, f_{diff,z}, f_{mult,z}, f_{x,z}, f_{div,z}, f_{zdiv,z}, f_{mod,z}, f_{zmod,z}, f_{gcd,z}$ for all $z \in \mathbf{Z}\}$ $((\mathbf{Z} \longrightarrow \mathbf{Z}) = \{f : \mathbf{Z} \longrightarrow \mathbf{Z}\})$.

- $f_{add,z}(z') = z + z'$
- $f_{diff,z}(z') = z - z'$
- $f_{mult,z}(z') = z \cdot z'$
- $f_{x,z}(z') = z^{z'}$
- $f_{div,z}(z') = z / z'$
- $f_{zdiv,z}(z') = z' / z$
- $f_{mod,z}(z') = z \bmod z'$
- $f_{zmod,z}(z') = z' \bmod z$
- $f_{gcd,z}(z') = \gcd(z, z')$

Moreover *efficient*$(f) \wedge$ *efficient*$(g) \implies$ *efficient*$(f \circ g)$.

Computer Lemmata (Property (1)):

A formal version of Property (1) is expressed by the following computer lemmata.

lemma [| prime (p::num); prime (q::num); p mod 4 = 3; q mod 4 = 3;
A0((n::int))=p; A1(n)=q; n=(int p)·(int q);
zgcd((m::int),(int n)) = 1; (x·(int p) + z·int q)) = 1|]
⟹ (modq (mod_key0 x z A0(n) A1(n) (modq m n)) n) = (modq m n)";

lemma "[| prime (p::num); prime (q::num); p mod 4 = 3; q mod 4 = 3;
A0((n::int))=p; A1(n)=q; n=(int p)·(int q);
zgcd((m::int),(int n)) = 1; (x·(int p) + z·(int q)) = 1|]
$\Longrightarrow$ (modq (mod_key1 x z A0(n) A1(n) (modq m n)) n) = (modq m n)";

lemma "[| efficient A0(n); efficient A1(n)|] $\Longrightarrow$ efficient ((add ((mult x) ((mult A0(n)) ((mod_operator n) ((exponent c) ((div_operator 4) ((add A1(n)) 1)))))) o ((mult z) o (mult A1(n)) o ((mod_operator n) o ((exponent c) o ((div_operator 4) o (add A0(n)))))))";

lemma "[| efficient A0(n); efficient A1(n)|] $\Longrightarrow$ efficient ((diff ((mult x) ((mult A0(n)) ((mod_operator n) ((exponent c) ((div_operator 4) ((add A1(n)) 1)))))) o ((mult z) o (mult (A1(n)) o ((mod_operator n) o ((exponent c) o ((div_operator 4) o (add A0(n)))))))";

Remark:

- (add x) z = x+z
- (diff x) z = x−z
- (mult x) z = x∗z
- (mod_operator n) x = x mod n
- (exponent x) n = x^n
- (div_operator n) x = x div n
- (add ((mult x) ((mult A0(n)) ((mod_operator n) ((exponent c) ((div_operator 4) ((add A1(n)) 1)))))) o ((mult z) o (mult A1(n)) o ((mod_operator n) o ((exponent c) o ((div_operator 4) o (add A0(n)))))))) = (modq (mod_key0 x z A0(n) A1(n) (modq m n)) n)
- (diff ((mult x) ((mult A0(n)) ((mod_operator n) ((exponent c) ((div_operator 4) ((add A1(n)) 1)))))) o ((mult z) o (mult (A1(n)) o ((mod_operator n) o ((exponent c) o ((div_operator 4) o (add A0(n)))))))) = (modq (mod_key1 x z A0(n) A1(n) (modq m n)) n)

More Computer Lemmata (Property (2)):

A formal version of Property (2) is given by the following computer lemmata.

lemma "[|(n::num) = (p::num)·(q::num); zgcd((m::int),(int n)) = 1;
prime p; prime q; (c::int) = ((m::int)·m) mod ((int p)·(int q));
(c::int) = ((x::int)·x) mod ((int p)·(int q));
m mod (int p) = x mod (int p); (m mod (int q)) = (((int q)−x) mod (int q)); (A c) = m|]
$\Longrightarrow$ (zgcd((A c)−x,n) = (int p)) $\wedge$ ((int n) div zgcd((A c)−x,n) = (int q))";

lemma "[|(n::num) = (p::num)·(q::num); zgcd((m::int),(int n)) = 1;
prime p; prime q; (c::int) = ((m::int)·m) mod ((int p)·(int q));
(c::int) = ((x::int)·x) mod ((int p)·(int q));
m mod (int p) = ((int p)−x) mod (int p); (m mod (int q)) = (x mod (int q)); (A c) = m|]
$\Longrightarrow$ (zgcd((A c)−x,n) = (int q)) $\wedge$ ((int n) div zgcd((A c)−x,n) = (int p))";

lemma "efficient ((zgcd n) o (diff m))";

lemma "efficient ((div_alg n) o ((zgcd n) o (diff m)))";

Remark:

- (div_alg n) x = n/x
- (zgcd n) x = zgcd n x
- (diff m) x = m−x
- ((div_alg n) o (zgcd n) o (diff m)) = n/(zgcd n (m−x))

V. Conclusion

We formally described and computer-proved the Rabin functions introduced in [7]. Therefore we reviewed relevant aspects from [7] to illustrate the mathematical background of the given formalized public-key functions.

A formal analysis with computer support provides a complex, but useful approach to verify the functional correctness of implementations of cryptographic algorithms. Moreover, the computer-proven lemmata augment the given database that is basic for many Isabelle theories. This implementation is a component of a formally verified cryptographic client compiled in the Verisoft project.

This article is part of an effort to unify the formal and the computational views of cryptographic verification. More specifically, this work continues our recent work that provides useful formal descriptions of mathematical background and cryptographic algorithms computer-proven with Isabelle/HOL (compare [4] and [3]).

In this article we explored a formal specification and verification of the Rabin encryption and signing scheme in order to computer-prove interesting properties of this formal description. Therefore we used Isabelle/HOL to formally verify functional correctness as well as arguments concerning security of the given implementation, what means that we proved formal properties of the investigated functions with computer support. This shows that formal verification in the area of cryptography is possible. More exactly, we explored a construction of proofs with a minmal error rate. Furthermore, we augmented a proof database that can be used for further proof constructions.

The idea is to unify both verification approaches in cryptography by embedding one into the other: formalizing computational aspects as well as their computer verification embed the computational approach in the formal approach. We obtained a formal verified computational description of a cryptographic primitive, that can be used in practice (compare II-D).

References

[1] Martin Abadi and Jan Jürjens. Formal Eavesdropping and its Computational Interpretation, 2001.
[2] Johannes Buchmann. *Einführung in die Kryptographie*. Springer-Verlag, 2 edition, 2001.
[3] Johannes Buchmann and Markus Kaiser. Computer Verification in Cryptography. In *International Conference of Computer Science, Vienna, Austria*, volume 12, 2006.
[4] Johannes Buchmann, Tsuyoshi Takagi, and Markus Kaiser. A Framework for Machinery Proofs in Probability Theory for Use in Cryptography, 2005. Kryptotag in Darmstadt.
[5] http://www.verisoft.de.
[6] Tobias Nipkow, Lawrence Paulson, and Markus Wenzel. *Isabelle/HOL – A Proof Assistant for Higher-Order Logic*, volume 2283 of *Lecture Notes in Computer Science*. Springer-Verlag, 2002.

[7] Michael Rabin. Digitalized signatures and public key functions as intractable as factorization, 1979. Massachusetts Institute of Technology, Laboratory for Computer Science, Cambridge, Massachusetts.
[8] Nigel Smart. *Cryptography: An Indroduction*. McGraw-Hill Education, 2003.
[9] Christoph Sprenger, Michael Backes, Birgit Pfitzmann, and Michael Waidner. Cryptographically Sound Theorem Proving, 2006.
[10] http://isabelle.in.tum.de.