International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:1, No:5, 2007

# A Computer Proven Application of the Discrete Logarithm Problem

Sebastian Kusch, Markus Kaiser

***Abstract*****—** In this paper we analyze the application of a formal proof system to the discrete logarithm problem used in public-key cryptography. That means, we explore a computer verification of the ElGamal encryption scheme with the formal proof system Isabelle/HOL. More precisely, the functional correctness of this algorithm is formally verified with computer support. Besides, we present a formalization of the DSA signature scheme in the Isabelle/HOL system. We show that this scheme is correct what is a necessary condition for the usefulness of any cryptographic signature scheme.

***Keywords*****—** formal proof system, higher-order logic, formal verification, cryptographic signature scheme

## I. Introduction

COMPUTER proofs (the application of formal proof systems) are a useful approach in the area of verification. Formal and computer verification augment the traditional concept of software engineering by providing techniques that guarantee trustiness as well as correctness of software systems in a mathematical way. There are many possible applications of formal and computer verification like automotive, medical technology, information technology security and cryptography.

In public-key cyrptography, many algorithm are based the factoring problem or the discrete logarithm problem. While the Rabin public-key scheme relies on the hardness of factoring an integer $n$ that is a product of two large primes $p$ and $q$ (compare [9]), the ElGamal scheme is based on the discrete logarithm problem. In this paper, we explore a computer verification of the ElGamal encryption scheme with the formal proof system Isabelle/HOL. More precisely, the functional correctness of this algorithm is formally verified with computer support. Besides, we present a formalization of the DSA signature scheme in the Isabelle/HOL system [7]. The Digital Signature Algorithm (DSA) was proposed by the United States Institute of Standards (NIST) in 1991 to be used in the Digital Signature Standard (DSS). The DSA is an efficient version of the ElGamal signature scheme based on discrete logarithms since it mainly reduces the bitlength of the exponents involved. In contrast to high-level examinations of cryptographic protocols like the key exchange schemes suggested by Needham and Schroeder for instance, formalizations involving also the single algebraic equations are a relatively new application of theorem provers like Isabelle.

This paper is organized as follows: We start in Chapter II with a description of the used formal proof system. In Chapter III we explore the ElGamal encryption scheme with a formal proof system, and in Chapter IV we present a formalization of the Digital Signature Algorithm. In Chapter V some conclusions, as well as some comments on future work are given.

The authors are 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.

## II. Foundation

In this chapter we give an overview on the used formal proof system Isabelle/HOL as well as the cryptographic knowledge concerning the Digital Signature Algorithm.

### A. The Isabelle System

The Isabelle system (written in ML) is a generic theorem proving environment invented at Cambridge University and TU Munich that can be applied to several logics. Isabelle/HOL is the specialization of Isabelle for higher-order logic (HOL) whereas other logics distributed with Isabelle include the usual first-order logic (FOL) or LCF which is a version of Scotts logic for computable functions. Isabelle is also often referred to as a "Proof Assistant" underlining the process of alternating automated reasoning with human intervention.

As we will see in detail in the following subsection theorem proving with Isabelle is often based on a "human-guided" manipulation of the proof state, where the system itself only executes the given commands and - of course - verifies their applicability until finally all subgoals have been proven (e.g. via reduction to already proven lemmata). On the other hand there are also strong tools that can be applied to handle (suitable) proofs (or at least considerable parts of it) automatically. The user is free to alternate these two paradigms arbitrarily.

Isabelle comes with a large theory library of formally verified mathematics, including elementary number theory (for example, Gauss's law of quadratic reciprocity), analysis (basic properties of limits, derivatives and integrals), algebra (up to Sylow's theorem) and set theory (the relative consistency of the Axiom of Choice). Using appropriate include-commands we may base our (new) theorems upon those in the libraries by referring to them during the proof process wherever they are applicable.

More information about Isabelle/HOL are given in [8], 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. Digital Signatures

Digital signatures are meant to prevent (or rather to detect) unauthorized modifications to data and to authenticate

the identity of the signatory. Besides, the signatory cannot repudiate a signature that was issued by him or her. The former means that - in contrast to conventional (handwritten) signatures - digital signatures are tightly connected to the data they refer to. A signature for document $m_1$ is different from that for document $m_2$ and it is (or should be) impossible for an attacker to create a signature for $m_2$ by knowing a valid signature for $m_1$; digital signatures are in this sense even stronger than their conventional counterparts.

In detail we expect a signature scheme to meet the following requirement [2]:

- **Identity:** The signature needs to prove the signatorys identity beyond any doubt.
- **Non-Reusability:** The signature is only valid together with the original document.
- **Non-Modifibiliy:** Any subsequent manipulation of the document will be detected.
- **Non-Repudiation:** The signatory cannot deny to have signed the document.

In a symmetric system, the key for encrypting a message is the same as the one that is needed to decrypt it, or - more generally - the keys might not be the same, but it is easy - from knowing one of them - to compute the other.

In contrast, an asymmetric scheme is even based on the impossibility (or rather difficulty) to compute the decryption key from knowing the one used for encryption. Usually the encryption key is called the public key that may be known to anyone who wants to send secrete messages to participant $X$, whereas $X$ is the only one who has access to his private key which is a prerequisite for decrypting them again.

**Digital Signatures based on Symmetric Cryptography**

A signature scheme based on the symmetric case necessitates a trusted third party $T$. Furthermore, we need to assume that every participant $A$ has exchanged a secrete symmetric key $K_A$ with $T$.

If a participant $A$ wants to sign a document for participant $B$, he encrypts it using his key $K_A$ and sends it to $T$. Since $T$ assumes that $A$ is the only person (besides himself) that has access to $K_A$, he can confirm that the document originates from $A$. After decrypting it using $K_A$ and attaching a corresponding confirmation for $B$ expressing that the document was actually sent by $A$, he encrypts it (together with the confirmation) using $K_B$ and gives it to recipient $B$.
By decrypting it and reading the confirmation, $B$ can be sure that that $A$ was the sender (and the signatory) since he knows that $T$ confirms this claim and $T$ is (by definition) trustworthy.

It is obvious that the assumptions involved here are quite ambitious. First of all we just need to believe that $T$ is not only trustworthy but also resistant to any malfunction or attack. Since it is involved in every exchange of signed documents it is on the other hand required to work relatively quickly. We also need a secure exchange of secrete symmetric keys. And finally we need to assume that these keys are also stored safely (inaccessible for any other party) by the participants. The latter is basically the only difficulty arising here that also applies to asymmetric systems.

**Digital Signatures based on Asymmetric Cryptography**

For an asymmetric cryptosystem, let $e$ be an arbitrary encryption key, $d$ the corresponding decryption key, and $E_e$ as well as $D_d$ the associated encryption (or decryption) functions. Then $D_d(E_e(m)) = m$ holds for every message $m$.

The system can be directly used for digital signatures if the encryption and decryption functions commute, that means if - under the above assumptions - the following is also valid:

$$E_e(D_d(m)) = m$$

A participant $A$ applies his (private) decryption key $d$ (that is only known to himself) to a message $m$, yielding the signature $c := D_d(m)$, and every recipient may use $A$'s (public) encryption key $e$ to verify the signature, i.e. to check whether $E_e(c) = m$. The security requirements are the same is in the original application of the system in terms of a confidential exchange of messages:

- The private keys need to be stored safely. Only the legitimate users are allowed to have access to them.
- The public keys should be protected from unauthorized manipulation, stored and managed at a safe place like a trust center. If an attacker succeeds in changing the public key data, faked signatures might be mistaken for legitimate ones since a correct verification requires the correct public key.

Of course, usually one would not sign the actual messages (or documents) but their hash values, also for security reasons. In addition, it is not recommendable at all to use the same key pair for signatures *and* for encryption / decryption operations. But this is the idea how it works.

## III. Discrete Logarithm Problem

In this chapter, the ElGamal encryption scheme that is based on the discrete logarithm problem is explicated. Moreover, we explore a formal anaylsis with Isabelle/HOL.

A discrete logaritm of a multiplicative group $G$ with group order $e \in \mathbf{N}$ can be given by a generator $g$ of $G$ (for all $z \in \mathbf{N}$ with $z < e$, $g^z \neq 1$ and $g^e = 1$). If $y = g^x$ for $x \in \mathbf{N}$, $x$ is the discrete logarithm of $y$ ($y = g^x \bmod p$, if $p$ prime and the group given by $p$)

If $|G| = p$, then $m \in \mathbf{N}$ with $m < p$ can be encrypted by the following computation:

1) choice of a random number $r \in \{1, \ldots, p-1\}$
2) $a = g^r$
3) $z = y^r \cdot m$
4) return $(a, z)$

Applying the private key decrypts the ciphertext $(a, z)$:

$$za^{-x} = y \cdot m \cdot g^{-rx} = g^{xr-xr} \cdot m = m$$

A formal compilation of of the correctness of this algorithm can be given by the following computer lemma, where the datatypes are given by $num$, $int$ and $bool$.

**consts** *mod_generator* :: $[int, int] \Rightarrow bool$

**defs** *mod_generator_def*: $mod\_generator\ g\ p \equiv 0 < g \wedge g < p \wedge \text{zprime}\ p \wedge \text{zgcd}(g,p) = 1 \wedge (\forall(x :: num).\forall(z :: num).(g^x \bmod p) = (g^z \bmod p) \longrightarrow ((\text{int}\ x \bmod (p - (1 :: int))) = (\text{int}\ z \bmod (p - (1 :: int))))) \wedge (\forall(x :: num).0 < x \wedge x < (\text{num}\ (p - (1 :: int))) \longrightarrow (g^x \bmod p \neq 1)) \wedge (g^{(\text{num}\ (p-(1::int)))} \bmod p = 1)$
**lemma** *formal_lemma*: $[|\ mod\_generator\ g\ p; (h :: int) = g^{(x::num)} \bmod p; (a :: int) = g^{(k::num)} mod p; (a' :: int) = g^{(l::num)} \bmod p; (b :: int) = h^k * (m :: int); (\text{num}\ (p - (1 :: int))) = k + l; 1 \bmod p = 1\ |] \Longrightarrow (b * a'^x) \bmod p = m \bmod p$

In [4] other formal verifcation work concerning the ElGamal scheme can be found.

## IV. The DSA Signature Scheme

The Digital Signature Algorithm is a more efficient version of the ElGamal scheme [3], proposed by the National Institute of Standards and Technology (NIST) in August 1991 and based on discrete logarithms in the group $(Z/pZ)^*$ where $p$ is a (large) prime number. In the DSA case the exponentiations concern only a subgroup of $(Z/pZ)^*$ which reduces the size of the exponents involved. In addition, the verification requires only two modular exponentiations instead of three. The United States Institute of Standards (NIST) recommends the Secure Hash Algorithm (SHA-1) to be applied to the data first. That means the signing / verifying does not refer to the actual data, but to its hash value using SHA-1.

### A. DSA Parameters

In order to generate DSA signatures we need the following ingredients [1]:

- $p$: a prime modulus, where $2^{L-1} < p < 2^L$ for $512 \leq L \leq 1024$ and $L$ a multiple of 64
- $q$: a prime divisor of $p-1$, where $2^{159} < q < 2^{160}$
- $g := h^{(p-1)/q} \bmod p$, where $h$ is any integer with $1 < h < p-1$ such that $h^{(p-1)/q} \bmod p > 1$ (which implies that $g$ has order $q \bmod p$)
- $x$: a randomly or pseudo-randomly generated integer with $0 < x < q$
- $y := g^x \bmod p$
- $k$: a randomly or pseudo-randomly generated integer with $0 < k < q$

**Note**: Usually one would first choose the prime $q$, then the larger prime $p$, as the required conditions are in this case easier to satisfy. Moreover, the parameter $k$ is allowed to be used for *one* signature, only, and must then be replaced by a new (pseudo-)random value.

$p$, $q$, $g$ and $y$ can be made public, whereas $x$ and $k$ need to be kept secret and $k$ must even be regenerated for each new signature as already mentioned.

### B. Signature Generation

Let $h(m)$ denote the hash value of a message $m$, i.e. the output of the SHA-1 algorithm as recommended by the NIST. The signature of the message $m$ is the pair of numbers $r$ and $s$ computed according to the equations below.

- $r := (g^k \bmod p) \bmod q$
- $s := (k^{-1}(h(m) + xr)) \bmod q$

In the above, $k^{-1}$ is the multiplicative inverse of $k$, mod $q$. In the unlikely case that either $r$ or $s$ turn out to be 0, the signatory is (strongly) advised to choose a new (pseudo-)random value for $k$. The signature is transmitted along with the message to the verifier.

### C. Signature Verification

Prior to verifying the signature in a signed message, $p$, $q$, $g$ and $y$ (together with the sender's identity, indeed) are made available to the verifier in an authenticated manner.

Let $m'$, $r'$ and $s'$ be the received versions of the message and the signature parameters, respectively. To verify the signature, the verifier first checks to see that $0 < r' < q$ and $0 < s' < q$; if either condition is violated the signature shall be rejected. If these two conditions are satisfied the verifier computes

- $w := s'^{-1} \bmod q$
- $u_1 := (h(m')w) \bmod q$
- $u_2 := (r'w) \bmod q$
- $v := ((g^{u_1} y^{u_2}) \bmod p) \bmod q$

The signature is verified if $v = r'$. Otherwise, the message may have been modified, the message may have been incorrectly signed by the signatory, or the message may have been signed by an impostor. The message should be considered invalid. For a proof that $v = r'$ if $m = m'$, $r = r'$ and $s = s'$ (both in a "human readable" style and in a formalized version in the Isabelle system).

### D. The Roadmap of the Proof

What follows is a summary of the most important steps of the formal proof in Isabelle/HOL in order to outline the general course.

Instead of directly proving the verification equation in the group $(Z/pZ)^*$ we decided to consider the more general case of an arbitrary finite cyclic group. On the one hand we yield stronger (more general) statements this way. On the other hand the reasoning gets a bit simpler since we can eliminate the reductions modulo $p$ (appearing in the handwrittten proof) where $p-1$ is the group order.

However, the verifying congruency holds even if we regard $r$ and $g^k$ as independent of each other, as long as the second component of the signature, namely $s$, is computed as defined in the specification. Therefore, my *verifying* theorem later on is strictly speaking a generalization of the actual verification of DSA, concerning the group we are regarding as well as concerning the relationships between the parameters involved.

The following elements will appear in our encoding:

- natural numbers $n$ and $q$, as defined above - such that our group order gets $n$ - along with the quotient $u := n/q$, or equivalently, as it will appear in the actual encoding, $uq = n$

- a primitive root (i.e. especially a group element) $h$ and a group element $g$ such that $g := h^u$, which is the same definition as in section IV since $u := n/q$
- natural number $x$ and group element $y := g^x$ playing the same role as in the specification
- natural numbers $k$, $r$ and $s$ (according to the specification) along with natural numbers $k^{-1}$ and $s^{-1}$ (or $k\_inv$ and $s\_inv$, as written in the theory file, respectively), denoting the inverse (modulo $q$) of the $k$ and the $s$ values

Using the above "ingredients" the correctness proof of the DSA signature algorithm) in the theory file dsa.thy comprises the following steps:

- **constants definitions**
  - the natural number $n$ such that $n$ represents the order of our finite group
  - a primitive root *h* for a finite cyclic group (which is from Isabelle's point of view of no specific type but annotated by a *type variable*)
  - the functions *inv* and *pow* where *inv(g)* yields the inverse of a group element $g$ and *pow(g,k)* for a natural number $k$ is the $n$-th exponentiation of $g$ regarding the group operation

  In our theory these steps are handled using the following encoding:

  **consts**
  *h::"'a"*
  *n::"nat"*
  *inv::"'a" ⇒ "'a"*
  *pow::"['a, nat]⇒ 'a"*

  As already explained *'a* is a type variable, which means the functions *inv* for instance may be applied to arbitrary types. However, characteristic properties like the (semantic) meaning of *inv* as the inverse of a group element will probably hold only in the particular axiomatic type classes that we can define now.

- **definitions of the axiomatic typeclasses**
  - axiomatic typeclass *group_mult*, which is a subtype of *monoid_mult* in the Isabelle library, where *mult* means basically that the group operation is denoted by "$*$"
  - subclass *cyclic_group_mult*, the class of all cyclic groups, i.e. there is a primitive root *h*, such that each element can be written as a power of *h*
  - *finite_cyclic_group_mult* that a cyclic group of fixed (but arbitrary) finite cardinality $n$ and the "universe" of our proof, i.e. we do not consider $(Z/pZ)^*$ but arbitrary finite cyclic groups

  Basic definitions of the whole Isabelle theory:
  **axclass** *group_mult* ⊆ *monoid_mult*
  *left_inverse: "inv a * a = 1"*
  *pow_def_0: "pow a 0 = 1"*
  *pow_def_suc: "pow a (Suc k) = (pow a k) * a"*

  **axclass** *cyclic_group_mult* ⊆ *group_mult*
  *generator: "∃k. pow h k = a"*

  **axclass** *finite_cyclic_group_mult* ⊆ *cyclic_group_mult*
  *finite: "pow h n = 1"*

  A group is just a monoid with the additional property of the *left_inverse* rule. (From this one could also derive a corresponding *right_inverse* rule if necessary, i.e. the latter already follows from this and does not need to be added as an axiom.) In the definition of cyclic groups the group element $a$ is implicitly $\forall$-quantified like all variable symbols occurring in such formulae, unless they are equipped with another (explicit) quantifier or defined as a constant.

  With these definitions we can now derive first lemmata, mainly concerned with the *pow* function:

- **basic associativity lemmata for groups**
  - lemma *pow_assoc1* stating that $a^k * a^l = a^{k+l}$
  - lemma *pow_assoc2* revealing the hardly surprising finding
    $(a^k)^l = g^{kl}$
    We tried to keep these lemmata as general as possible. The first one, for instance, is valid (and defined) for arbitrary groups, not only for finite cyclic groups (even if we only need it there). In between there is from time to time a collection of relatively uninteresting lemmata (that are ommited here).
- **lemmata about the order of group elements**
  - lemma *subgroup_generator*, mainly concerned with the role of the element we denoted by $g$ in the DSA specification generating the subgroup in which all the DSA computations finally take place:
    $uq = n \wedge g = h^u \Longrightarrow g^q = 1$, i.e. element $g$ generating our subgroup has an order that devides $q$ (and as a corollary $uq = n \wedge g = h^u \Longrightarrow g^{qk} = 1$)

Based on some more auxiliary lemmata we now approach the discovery that reducing the exponent modulo $q$ is irrelevant for exponentiations with this element $g$ as basis. Note that $g$ is always defined in terms of the primitive root $h$ which has been defined as a constant.

- **lemmata for reduced exponents**
  - lemma *pow_mod1* : $uq = n \wedge g = h^u \Longrightarrow g^{r+kq} = g^r$
  - lemma *pow_mod2* : $uq = n \wedge g = h^u \wedge \exists l(k = r + lq) \Longrightarrow g^k = g^r$
  - lemma *pow_mod3* : $uq = n \wedge g = h^u \Longrightarrow g^k = g^{k \bmod q}$
  - lemma *pow_mod4* : $uq = n \wedge g = h^u \wedge k \bmod q = l \bmod q$
    $\Longrightarrow g^k = g^l$

The lemmata we have proven so far are almost sufficient to approach the verification congruency. But there is still a little gap we have to close. We need from the defining equation of natural number $s$, that is

$$s := (k^{-1}(h(m) + xr)) \bmod q$$

the following congruency modulo $q$:

$$k \equiv s^{-1}(h(m) + xr) \bmod q$$

which looks for the "human reader" like a matter of course, since this seems to follow simply by multiplying both sides with $s^{-1}$ and then with $k$. However, if we have a closer look we realize that we also need the commutativity of multiplication along with the finding that the equivalency modulo $q$ for a natural number $q$ is actually a *congruency* relation, both for addition and multiplication. Furthermore, we need to apply those rules several times and (in a formal proof) we cannot simply say something like "analogously it follows ..." skipping the details in the second round.

Even if it is not about a technically difficult deduction these reasons cause the formal proof in Isabelle to turn out relatively long and bulky, hard to read for a human eye. Of course, it might be possible to prove the following lemma - basically expressing the above consequence - in a slightly more elegant fashion. If we regard our proof scheme in retrospect this applies especially to this proof. However, writing *elegant* proves in Isabelle is beyond the scope of this paper.

- **a necessary ingredient**
  - lemma *inv_exp:*
    $a = b^{-1}c \bmod q \wedge$
    $aa^{-1} \bmod q = 1 \wedge$
    $bb^{-1} \bmod q = 1$
    $\Longrightarrow b \bmod q = a^{-1}c \bmod q$
- **the final step**
  - theorem *verify:*
    $uq = n \wedge$
    $h^u = g \wedge$
    $y = g^x \wedge$
    $s = k^{-1}(h + xr) \bmod q \wedge$
    $ss^{-1} \bmod q = 1 \wedge$
    $kk^{-1} \bmod q = 1$
    $\Longrightarrow g^{((s^{-1})h) \ mod \ q} y^{(r(s^{-1})) \ mod \ q} = g^k$

These two steps take both a bit longer than the previous lemmata. This is mainly due to term-rewriting operations, rather than technically interesting reasoning.

Furtehrmore, we can transfer the theorem to the special case $(Z/pZ)^*$ in the specification under the following assumptions:

- $(Z/pZ)^*$ is a finite cyclic group.
- Reductions modulo a natural number $q$ "preserve" equality, i.e.
  $a = b \Longrightarrow a \bmod q = b \bmod q$.
- Adding an assumption does not reduce the set of consequences, in short,
  $(\Phi \Longrightarrow \phi) \Longrightarrow (\Psi \Longrightarrow \phi)$ for $\Phi \subseteq \Psi$.

Concerning these aspects we will not use the term "obvious" (since this would be exactly what we criticized concerning common informal proofs at the beginning of this section). Instead, we claim that the first assumption could also proven in Isabelle/HOL relatively easily. The second one appeared in my encoding as a simple lemma (and if this statement did not hold, "mod" would not even be a function), whereas the third "meta-claim" is part of any reasonable deduction system - especially present in Isabelle.

Therefore, we assert that our theorem expressing the verifying equation (or verifying congruency) actually entails the special case suggested in the DSA specification. In addition, it is more generic and can therefore be applied to a wider spectrum of instances.

## V. Conclusion

We explored the application of a formal proof system to the discrete logarithm problem used in public-key cryptography. Therefore, we gave a computer verification of the ElGamal encryption scheme with the computer system Isabelle/HOL. More precisely, we proved the functional correctness of this algorithm formally. Besides, we present a formal description and computer verification of the DSA signature scheme with Isabelle/HOL. In this case, we proved formally that this scheme is correct what is a necessary condition for the usefulness of any cryptographic signature scheme. Moreover, we gave a roadmap of our formal verification in order to provide a better overview. 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 paper 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 [5] and [6]).

## References

[1] FIPS-PUB 186-2. Digital Signature Standard, January 27, 2000. United States Department of Commerce/National Institute of Standads and Technology.
[2] Claudia Eckert. *IT-Sicherheit*. Oldenbourg Wissenschaftsverlag, 2 edition, 2003.
[3] Taher ElGamal. A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In *Advances in Cryptology – CRYPTO 84*, pages 10–18. Springer-Verlag, 1984/1985. IEEE Transaction of Information Technology, v. IT-31, n.4, 1985.
[4] Joe Hurd. Elliptic Curve Cryptography. A case study.
[5] Markus Kaiser and Johannes Buchmann. Computer Verification in Cryptography. In *International Conference of Computer Science, Vienna, Austria*, volume 12, 2006.
[6] Markus Kaiser, Johannes Buchmann, and Tsuyoshi Takagi. A Framework for Machinery Proofs in Probability Theory for Use in Cryptography, 2005. Kryptotag in Darmstadt.
[7] Sebastian Kusch. Formalizing the DSA Signature Scheme in Isabelle/HOL. Diplomarbeit, Technische Universität Darmstadt, 2007.
[8] 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.
[9] Michael Rabin. Digitalized signatures and public key functions as intractable as factorization, 1979. Massachusetts Institute of Technology, Laboratory for Computer Science, Cambridge, Massachusetts.
[10] http://isabelle.in.tum.de.