HOW TO TURN A PIN INTO AN IRON BEAM

A patent-free and practical protocol for secure communication using a weak common secret

Stefan Lucks

Theoretische Informatik, Universität Mannheim, Germany

luckst@th.informatik.uni-mannheim.de

Ruediger Weis

Department of Computer Systems, Vrije Universiteit Amsterdam, Netherlands

ruedi@cryptolabs.org

Abstract It is a hazardous fact of life that users are hardly ever willing and able to memorise strong passwords or long personal identification numbers (PINs). This paper describes a variant of the “open key exchange” (OKE) protocol, a cryptographic scheme to provide high security even when using low-entropy passwords. The variant has been dedicatedly designed for slow devices, such as smart cards.

1. Introduction

Speak “friend” and enter! (J. R. R. Tolkien)

Consider two principals, Alice and Bob, trying to communicate via an insecure network. Think of Alice being a host and Bob being a smart card¹. Alice, as the host, does not know whether she is connected to Bob or to a fake smart card, and Bob does not know whether he is connected to a fake host. Worse, Alice may be connected to a fake smart card, *and* Bob may be connected to a fake host *and* the adversary has simultaneously access to the fake smart card and to the fake host – i.e., the connection between Alice and Bob may be completely controlled by the adversary, whom we call Eve here².

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: [10.1007/978-0-387-35691-4_52](https://doi.org/10.1007/978-0-387-35691-4_52)

D. Gritzalis et al. (eds.), *Security and Privacy in the Age of Uncertainty*

© IFIP International Federation for Information Processing 2003

If Alice and Bob use a *strong secret key*, or if one of them knows the other's *authentic public key*, standard cryptographic techniques for key agreement allow both sides to establish a session key and to use it for secure communication (i.e., to guarantee both confidentiality and authenticity of the messages). Many practical applications, however, suffer from the following problem: An authentic public key is not known, and if both sides agree on a secret key, it is a *weak secret key*, generated from a simple password to be entered by the user.³ Since humans are quite bad at generating and memorising high-entropy secrets, searching for all likely (or even possible) keys – so-called *dictionary attacks* – is efficient and will probably reveal the secret key.

Fortunately, cryptographers have started developing “*password protocols*” to securely authenticate and exchange session keys, even if the participants only have agreed on a weak key in advance. The security of such protocols depends on counting how often an attempt to establish a secure connection fails. The failure may occur either because the other participant responds “wrongly”, or because no response is given at all (some “time-out” event). We describe this by a *reject counter*.

Here, we reconsider the Protected OKE protocol from Lucks [15]. While the security assurance from the proof of security may be smaller, especially in comparison to protocols provably secure in the standard model [11, 13], we argue that for some practical applications, only Protected OKE is efficient enough. The reason is, that the competing protocols all use either the Diffie-Hellman key exchange or (in the case of [13]) the Cramer-Shoup Cryptosystem as their main cryptographic primitive. In contrast to this, Protected OKE uses RSA and allows the use of small fixed public exponents, such as 3. Let n be a security parameter. Given a group base g , both Diffie-Hellman and Cramer-Shoup require to compute some values g^x (modulo an n -bit number). This is computationally much more expensive than computing x^3 (modulo a composed n -bit number). In the case of Protected OKE, only one of the parties needs to compute the RSA-decryption operation (i.e., to compute x^d for a large d). Think of an application, where one of the parties is computationally impaired, e.g. by using a slow smartcard. In this case, computing x^3 can be fast enough, even if general exponentiations are too slow.

Sometimes, even Protected OKE with the public exponent 3 may be too slow. We describe a much more efficient variant, by trading *some security* for *greatly improved efficiency*.

2. The Security Model

In this section, we give an informal description of the security model.⁴

The Attack Model. As mentioned above, Eve has complete control over the connection between Bob and Alice, see below. She can

- read all messages,
- modify the messages, delay them or send them multiple times,
- create any message she wants on her own and send it to Alice as “Bob’s message” and to Bob as a message seemingly from Alice.

We consider protocols for *the authenticated key exchange (for weak keys)* between Alice and Bob. Such a protocol is *sound*, if both partners accept with overwhelming probability, assuming each message from Alice to Bob and each message from Bob to Alice is faithfully transmitted.

Both Alice and Bob can either send a reply, or accept the communication and create a session key, or reject the communication. They reject, if they either find out that a reply they get is “wrong”, or if a connection is timed out when they are waiting for a reply. So, in order to avoid being rejected, Eve must give “correct” replies.

Eve’s target is to learn a session key, which has been accepted by either Alice or Bob or both of them.

The Reject Counter. We assume that both Alice and Bob will not continue any communication after a fixed number $\mathcal{R}$ of *rejections*.⁵ The number of allowed rejections is crucial for Eve’s probability of success.

The Trivial Attack. Consider the following trivial attack: *Eve guesses the password and runs the protocol with either Alice or Bob. If her guess has been correct, her partner accepts and she learns the session key, i.e., she succeeds.*

If she had guessed the wrong password, her partner rejects (hopefully!).

If the protocol is good, Eve learns nothing else about the correct password, except that it is different from her guess.

If the initial probability to guess the right password is $\leq \sigma$, if Alice stops communication after at most τ_a rejections, and if Bob stops communication after at most τ_b rejections, then Eve’s complete probability of success by this *trivial* attack is $\leq \sigma(\tau_a + \tau_b)$.⁶

Secure key exchange protocol for weak keys. A protocol to do the authenticated key exchange protocol for weak keys is *secure*, if no attack is “much”⁷ better than the trivial attack.

3. The generic open key exchange protocol

We fix two security parameters K and L with the following requirements: K is large enough such that doing any 2^K operations is infeasible. Let $L > K$ be large enough such that factorising an L -bit RSA-modulus N , which has secretly been chosen as the product of two $L/2$ -bit primes, is infeasible. Currently, we recommend $K \geq 80$ and $L \geq 1024$.

For the protocol, we need hash functions $h_i : \{0, 1\}^* \rightarrow \{0, 1\}^l$ ($i \in \{1, 2, 3\}$), an asymmetric key pair (E, D) , such that E describes the encryption operation $E : S_E \rightarrow S_E$, where S_E is a finite group, “ $*$ ” denotes the group operation, and “ $/$ ” its inverse, and a hash function $H : \{0, 1\}^* \rightarrow S_E$.⁸

The group operation needs to be compatible with the encryption function, i.e. for all $x, y \in S_E$: $E(x) * E(y) = E(x * y)$. Also, the encryption function must be invertible, i.e. $D(E(x)) = x$ must hold for $x \in S_E$. The group S_E can depend on the choice of E , as for RSA. The pair (E, D) and hence the group S_E can be chosen once by Alice and remains fixed during the experiment of running the protocol in the presence of Eve. D is kept secret. We use “ $|$ ” to indicate the concatenation of bit strings.

3.1 Protocol Description

The generic OKE protocol works like this [15]:

0. *Both in advance:* Alice and Bob agree on a common secret π .
Alice in advance: Alice generates a public/private-key pair (E, D) .
- 1 Alice chooses $m \in_{\mathbb{R}} \{0, 1\}^l$ and sends E and m to Bob.⁹
- 2 Bob chooses $\mu \in_{\mathbb{R}} \{0, 1\}^l$ and $a \in_{\mathbb{R}} S_E$, computes $p = H(E|m|\mu|\pi)$ and $q = E(a) * p$, and sends μ and q to Alice.
- 3 Alice computes p like Bob, $a' = D(q/p)$, and sends $r = h_1(a')$ to Bob. (Now $a' = a$.)
- 4 If $r \neq h_1(a)$, Bob rejects. Otherwise, he computes $k = h_2(a)$ and $t = h_3(a)$, sends t to Alice, and uses k as his session key.
- 5 Alice rejects if $t \neq h_3(a')$. Else, she uses $k' = h_2(a')$ as session key.

3.2 Security Theorem

We make the following two cryptographic assumptions:

- A1:** The hash functions H , h_1 , h_2 , and h_3 are treated as “random oracles” [4], i.e. Eve can send queries $x_1, x_2, \dots$ to the random oracle h_i resp. H and receives answers $h_i(x_j)$, resp. $H(x_j)$, all

independent random values. Only when a query $x_c = x_j$ is sent to the oracle h_i (or H) and Eve was given the oracle response $h_i(x_j)$ ($H(x_j)$) before, the result is predictable: $h_i(x_c) = h_i(x_j)$ (or $H(x_c) = H(x_j)$).

A2: Given a value x , randomly chosen from the set S_E , it is infeasible to find a value y with $y = D(x)$.

THEOREM 1 *Under the assumptions A1 and A2, the generic OKE protocol is a secure key exchange protocol for weak common secrets, i.e., it has the following properties:*

- 1 *If all participants behave as specified, both Alice and Bob accept with overwhelming probability and compute identical session keys.*
- 2 *If Eve faithfully transmits all messages between Alice and Bob and only listens passively, her probability of success is negligible.*
- 3 *If Eve mounts any active attacks, her probability of success does not significantly exceed the trivial attack.*

Proof: See [15].

4. When RSA comes into play

At a first look, the RSA public-key cryptosystem appears to be ideal for open key exchange. The public key is the pair $E = (e, N)$, where $N = PQ$ is the product of two huge primes P and Q and e is relatively prime to $\varphi(N) = \varphi(P)\varphi(Q) = (P-1)(Q-1)$. The encryption function is defined by $E(x) = x^e \bmod N$, the decryption function is $D(x) = x^d \bmod N$, where the secret exponent d is chosen such that $ed \equiv 1 \pmod{\varphi(N)}$. The natural choice for the group operation $*$ is the multiplication mod N , the group S_E is the set of all numbers in $\{1, \dots, N-1\}$ which are not divisible by P or Q , i.e. the size of S_E is $|S_E| = \varphi(N)$.

4.1 A Security Problem

Note that the function E must be invertible. We had required this above. This property is actually used in the proof of security [15].

If Eve, when impersonating Alice, chooses E such that it is not invertible, i.e., no function D exists at all, Theorem 1 is no longer applicable.¹⁰ In the case of RSA, unfortunately, Eve may choose her public key such that encryption is not invertible (and thus, not really “encryption” any more, since decryption is impossible). Specifically, if N is the product of two distinct odd primes P and Q , the RSA-function $y = x^e \pmod{N}$ is invertible, if and only if e divides neither $P-1$ nor $Q-1$.

Even though it may seem counter-intuitive, it may be useful for Eve to choose a non-invertible function E . We describe an attack:

For all likely passwords π^ , Eve computes $p = H(E|m|\mu|\pi^*)$. If the equivalence $a^e \equiv (q * (p^*)^{-1}) \pmod{N}$ has no solution, Eve concludes that π^* is wrong. By elementary group theory, we know that the number of such solutions is either 0 or a constant c , which depends on N .¹¹*

Thus, if here had been $|\Pi|$ passwords before the attack, this attack allows Eve to reduce the number of passwords down to $|\Pi|/c$, on the average.

4.2 Protected OKE

To defend against the above attack, we need an additional *protection step* when using RSA-based OKE variants. Alice (or whoever communicates with Bob) must prove that the function E in use is actually invertible. If the proof fails, Bob rejects.

The basic idea for such a proof is to require Alice to sign a couple of random values, using N and her secret exponent d . Bob checks the validity of these signatures, using N and the public exponent e . Lucks [15] suggested to use K such signatures, i.e., Alice had to sign K random values, and Bob had to verify K signatures. Each random value can only be signed with probability $1/c$, and if Alice (or Eve) fails to sign any given random values, Bob rejects.

[15] also proposed a technique to save bandwidth, avoiding the transmission of $K - 1$ of these signatures. Eve's chances to trick Bob into accepting a non-invertible encryption function are negligible.

4.3 Fixing the Exponent $e = 3$

In the context of this paper, N is an RSA-modulus, and for the sake of efficiency, we suggest to use $e = 3$ as the public exponent.¹²

If Eve makes the attempt to “cheat” by using a non-invertible encryption function, this means choosing $N' = PQ'$ with $Q' - 1$ divisible by 3. Then no value d exists with $3d \equiv 1 \pmod{Q'}$, and, in fact, an equivalence $y = x^3 \pmod{N'}$ with $\text{lcm}(y, N) = 1$ has either none or at least three solutions for x . (E.g., set $N' = 35 = 5 * 7$: the equation $29 = x^3 \pmod{N'}$ has three solutions, namely 5, 9, and 29.)

4.4 A Trade-Off between Security and Efficiency

With $K \geq 80$, Protected OKE requires Bob (the smart card) to do more than 80 public-key operations. This is quite slow, even if all these operations are of the form $y = x^3 \pmod{N}$. The bandwidth-saving technique uses multiplications mod N , which slows down things additionally.

In order to speed up the protocol, we propose to do only $K' \ll K$ signatures. Eve's chance to convince Bob that she can decrypt mod N may then be small, but not negligible. Consider Eve having chosen a modulus N such that $y = x^c \pmod{N}$ is not invertible, i.e., $c > 1$.

- With K' signatures, and if Bob stops communication after having rejected τ_b times, Eve is accepted

$$\frac{\tau_b}{c^{K'}} \quad \text{times on the average.}$$

- Each time her “cheated” modulus N is accepted, the number of passwords (or PINs) still possible

decreases from a value $|\Pi|$ down to $\frac{|\Pi|}{c}$ on the average.

Thus, $\log_2(c)$ bit of the Password- or PIN-entropy are lost.

Recall that c is under Eve's control. If Eve chooses a large c , her forged modulus N is unlikely to ever be accepted. If c is small, such as $c = 3$, she has some chance to reduce the password-space, but only by a moderate amount, such as decreasing the password-entropy by $\log_2(3) \approx 1.6$ bit. Consider, say, a random 6-digit decimal PIN. The initial entropy of such a PIN is $\log_2(1000000) \approx 20$ bit. In this case, the (unlikely but possible) loss of 1.6 bits of entropy may be tolerable.

How likely is the event, that Eve succeeds with her cheating, and some of the entropy of π is lost? We provide two examples:

- Consider $K' = 5$, and assume that Bob stops all communication after $\tau_b = 5$ rejections. In this case, Eve's probability of success is no more than $5/(c^5)$, i.e., even for $c = 3$ it is only about 2.1 %.
- With $K' = 7$ and $\tau_b = 10$, Eve's probability of success is less than 0.46 %, even for $c = 3$.

We stress that the security of the scheme against this type of attack only depends on the number K' of signatures required, and on the number τ_b of rejections Bob tolerates.

For most applications, $K' = 5$ and $\tau_b = 5$ seem to provide a reasonable compromise between security and efficiency. We stress that the event with probability 2.1 % is not “Eve succeeds in finding π ”, but only “Eve succeeds in reducing the entropy of π by 1.6 bit”.

4.5 Description of Modified Protocol

Our modified RSA-based Protected OKE protocol works like this:

0. *Both in advance:* Alice and Bob agree on a common secret π .
Alice in advance: Alice generates an RSA-modulus $N = PQ$, with the prime factors P and Q not being divisible by 3.
- 1 Alice chooses $m \in_{\mathbb{R}} \{0, 1\}^l$ and sends E and m to Bob.
- 1.1 Bob sends a random challenge x_0 to Alice.
- 1.2 For $i := 1$ to K' , Alice calculates $x_i = H(x_{i-1})^d \pmod{N}$. Alice sends the values $x_1, \dots, x_{K'}$ to Bob.
- 1.3 Bob checks $(x_{i-1}) = x_i^3 \pmod{N}$ (for $1 \leq i \leq K'$) and rejects if this fails for any such i .
- 2 Bob chooses $\mu \in_{\mathbb{R}} \{0, 1\}^l$ and $a \in_{\mathbb{R}} \{1, \dots, N-1\}$, computes $p = H(N|m|\mu|\pi)$ and $q = E(a) * p \pmod{N}$, and sends μ and q to Alice.
- 3 Alice computes p like Bob, $a' = (q/p)^d \pmod{N}$, and sends $r = h_1(a')$ to Bob.
- 4 If $r \neq h_1(a)$, Bob rejects. Otherwise, he computes $k = h_2(a)$ and $t = h_3(a)$, sends t to Alice, and uses k as his session key.
- 5 Alice rejects if $t \neq h_3(a')$. Else, she uses the session key $k' = h_2(a')$.

Note that the steps are numbered as their counterparts in the generic protocol. The additional “protection” steps are numbered as 1.x.

5. Final Remarks and Conclusion

5.1 Communication Time $\leftrightarrow$ Computations

We proposed a “reasonably secure” variant of RSA-based protected OKE, which improved on the original version [15] by reducing the amount of computations. The two *disadvantages* of this variant are:

- 1 the security is *somewhat* reduced (as quantified above), and
- 2 the amount of communication grows.

When implementing smartcard-based protocols in practice, the amount of computations dome the smartcards are not always the performance bottleneck. Sometimes, the performance is mainly determined by the amount of communication instead. In this case, we recommend to use the original RSA-based protected OKE protocol [15], which is more secure and even more efficient under such circumstances.

5.2 On Proofs of Security

The fact that the Generic OKE protocol is provably secure, but its seemingly natural instantiation with RSA can be insecure, raises some questions about “provably secure” cyptosystems. An application based on a “provably secure” cryptosystem may nevertheless be insecure, for one or more of the following reasons:

- The application fails to enforce some of the system’s environmental requirements. This is the case for RSA-based generic OKE.
- An assumption made to prove the scheme’s security is false.
- The “proven” security does not satisfy the application’s needs.

Is “provable security” cryptography of no use, then? We do not think so. Providing a proof of security for a cryptosystem forces its author

- to explicitly describe the system’s environmental requirements,
- to explicitly declare the assumptions made, and
- to exactly specify the what kind of “security” the scheme is supposed to provide.

A “provably secure” cryptosystem tells the security architect, what she has to watch for, when implementing it. This is different from “heuristic” cryptosystems, published without a proof of security. Note that the problems of RSA-based generic OKE have been described in the same paper as OKE and its proof of security [15], and are in fact quite obvious from the specification of the environmental requirements. On the other hand, security problems of variants of EKE, e.g. for RSA-based EKE [17] have been found years after the publication of these schemes.

5.3 The Random Oracle Model

Proofs of security in the *random oracle model* assume that the adversary treats hash functions like random oracles. The corresponding security results, showing the infeasibility of some attacks, are thus only valid with respect to restricted, so-called *generic* attacks. All currently known and practically relevant attacks against cryptographic protocols are generic. But in theory, non-generic attacks could exist. So a proof of security in the random oracle model is good, but a proof of security in the *standard model* is more assuring and better. See [9] for details.

5.4 Related Work

The formal verification of the security of all kinds of cryptographic protocols has been intensively studied before. We refer the reader to [8] for a “classical” example and to [2] and references therein for more up-to-date information.

The first password protocol has been developed in 1989 [14]. Some protocols with similar goals followed. An important step has been the “Encrypted Key Exchange” (EKE) protocol from Bellare and Merritt 1992 [6]. The core idea behind EKE is to use the secret (symmetric) key π to encrypt the public key E of a randomly chosen asymmetric key pair (E, D) , then to use E to encrypt the randomly chosen (symmetric) session key k and finally to decrypt $E(k)$ using D . The security of EKE depends on the fact that two parties using different values for π are likely to use different public keys E and E' , and that knowing an asymmetric key pair (E', D') does not help to decrypt $E(k)$. It is important for EKE protocols, that every public key is used only once.

None of the early password protocols came with a formal proof of security, and in fact, some have been shown to be insecure [1, 10, 17]. Thus, researchers started to look for *provably secure* protocols. The *Open Key Exchange* (OKE) protocol by Lucks [15] has been the first such protocol.¹³ The approach in [15] is based on the work of Bellare and Rogaway on key exchange protocols for *strong secret keys* [3]. In the sequel other protocols have been proven to be secure [16, 7, 5].

The proofs of security in [15, 7, 5] are all in the random oracle model. A proof in the standard model would be more meaningful and provide an improved security assurance. Quite recently, password protocols provably secure in the standard model have been developed. Goldreich and Lindell [11] described such a protocol and a proof of security based only on the assumption that trapdoor one-way permutations exist. The protocol from [11] borrows techniques from general multi-party computation and thus is quite impractical. Nevertheless, [11] can be seen as a theoretical break-through. Another recent protocol [13] is quite practical and provably secure in the standard model, but it is based on the Cramer-Shoup Cryptosystem and less efficient than our protocol.

5.5 On Smartcard Programmer Interfaces

When trying to implement new cryptographic schemes (such as OKE or other password protocols, or blinding for anonymous digital money, or ...) on smartcards with support for public-key cryptography, one may get disillusioned. Often, the interface for plain number-theoretic operations such as “given e , N , and x , compute $x^e \bmod N$ ” or “given N , x , and y , compute $xy \bmod N$ ” is just missing. This is, e.g., the case for the JAVA Card 2.0 API. This has two implications:

- 1 Without access to the raw number-theoretic functions, the application programmer may be locked into using the given primitives, making it very hard to implement new ones.

- 2 Some of the pre-defined primitives may turn out to be insecure (e.g., PKCS#1 v. 1.5 padding for public-key encryption). The application programmer may be locked into using an insecure scheme!

Of course, the application programmer may implement a “BigNum” library on her own. However, this is expensive, error-prone and terribly unnecessary. Also, a self-written “BigNum” library may be much less efficient than a pre-installed one, especially on a smartcard with a hardware accelerator for public-key cryptography.

5.6 Conclusion

We presented a new variant of the Protected Open Key Exchange password protocol, based on the provably secure Generic Open Key Exchange protocol. The goal is to enable secure communications using slow devices, low-entropy PINs or passwords and no PKI. A trade-off between security and efficiency allows the security architect to meet reasonable security requirements while outperforming other password protocols.

Notes

1. Cryptographers are very unhappy if they have to use other names.
2. All this scenarios can happen in the real world if we have a manipulated smart card reader or we have a scenario like the Internet Smart Card [12].
3. I.e., the password is stored on the smart card, and when needed typed in at the host.
4. A random oracle based formal security model can be found in [15].
5. This is consistent with a typical smart card application, where the smart card stops all operations if the wrong password has been typed in a fixed number of times and the user should not (and probably will not) type in her password hundreds or thousands of times.
6. As a concrete example, think of the password being a random 6-digit decimal number, i.e. $\sigma = 1/1\,000\,000$. If the smart card stops communication after 10 failed identification attempts, and the human user gives up after at most 90 failed attempts, then Eve’s probability of success is at most $(1/1\,000\,000)(10 + 90)$, i.e. $\leq 0.01\%$.
7. See [15] for a more formal description.
8. The four *independent* hash functions can be implemented based on on single hash function, such as the well-known SHA-1.
9. Note that though the pair (E, D) can be chosen once, it does not make sense to assume Bob can remember E . If Bob already did know Alice’s authentic public key E , he could simply choose the session key k_B and send $E(k_B)$ to Alice, instead of running the protocol.
10. This has some implications for the use of “provably secure” cryptosystems. We discuss this in Section 5.2.
11. If the public exponent $e = 3$ and $c > 1$, then $c \geq 3$.
12. Technically, $N = PQ$ is the product of two distinct large primes P and Q , and neither $(P - 1)$ nor $(Q - 1)$ is divisible by 3.
13. In the context of this paper, and in general when dealing with cryptographic protocols, any proof of security assumes the security of the cryptographic primitives in use. E.g., if the instantiation of a cryptographic protocol uses a 400-bit RSA key, this can hardly be secure, because 400-bit RSA is insecure. The protocol by itself may nevertheless be secure, such as an instantiation with, say, 2000-bit RSA.

References

- [1] R. ANDERSON, S. VAUDENAY: "Minding your p 's and q 's", Asiacrypt 96, Springer LNCS.
- [2] M. ABADI, P. ROGAWAY: "Reconciling Two Views of Cryptography (The Computational Soundness of Formal Encryption)", IFIP International Conference on Theoretical Computer Science, IFIP TCS2000, (2000).
- [3] M. BELLARE, P. ROGAWAY: "Entity Authentication and Key Distribution", Crypto 93, Springer LNCS 773.
- [4] M. BELLARE, P. ROGAWAY: "Random Oracles are Practical: A Paradigm for Designing Efficient Protocols", First ACM Conference on Computer and Communications Security, ACM, 1993.
- [5] M. BELLARE, D. POINTCHEVAL, P. ROGAWAY: "Authenticated Key Exchange Secure against Dictionary Attacks", Eurocrypt 2000, Springer LNCS 1807.
- [6] S. BELLOVIN, M. MERRIT: "Encrypted Key Exchange: Password-Based Protocols Secure against Dictionary Attacks", IEEE Computer Society Symposium on Research in Security and Privacy, 1992, 72–84.
- [7] V. BOYKO, P. MACKENZIE, S. PATEL: "Provably Secure Password-Authenticated Key Exchange Using Diffie-Hellman", Eurocrypt 2000, Springer LNCS 1807.
- [8] M. BURROWS, M. ABADI, R. NEEDHAM: "A logic of authentication", ACM Transactions on Computer Systems 8(1), Feb. 1990, 18–36.
- [9] R. CANTTI, O. GOLDBREICH, S. HALIEVI: "The Random Oracle Method Revisited", Proceedings of the 13th Annual ACM Symposium on Theory of Computing (STOC), 1998.
- [10] Y. DING, P. HORSTER: "Undetectable on-line Password Guessing Attacks", Operating Systems Review, Vol. 29, No. 4, Oct. 1995.
- [11] O. GOLDBREICH, Y. LINDELL: "Session-Key Generation using Human Passwords Only", Crypto 2001, Springer LNCS 2139.
- [12] N. ITOI, T. FUKUZAWA, P. HONEYMAN: "Secure Internet Smartcards", Proc. Java Card Workshop, Cannes, 2000.
- [13] J. KATZ, R. OSTROVSKY, M. YUNG: "Efficient Password-Authenticated Key Exchange using Human-Memorable Passwords", Eurocrypt 2001, Springer LNCS 2045.
- [14] M. LOMAS, L. GONG, J. SALTZER, R. NEEDHAM: "Reducing Risks from Poorly Chosen Keys", ACM Symposium on Operation System Principles, 1989.
- [15] S. LUCKS: "Open Key Exchange: How to Defeat Dictionary Attacks Without Encrypting Public Keys", Security Protocols 1997, Springer LNCS 1361.
- [16] P. MACKENZIE, R. SWAMINATHAN: "Secure Authentication with a Short Secret", Submission to IEEE P1363, <http://grouper.ieee.org/groups/1363/addendum.html>.
- [17] S. PATEL: "Number-Theoretic Attacks on Secure Password Schemes", IEEE Symposium on Security and Privacy, 1997.