Tatra Mt. Math. Publ. **29** (2004), 57–67

TATRA MOUNTAINS
Mathematical Publications

HOW TO USE UN-TRUSTY CRYPTOGRAPHIC DEVICES

Daniel Kucner — Mirosław Kutyłowski

ABSTRACT. Devices implementing cryptographic operations such as electronic signatures can leak information on secret data using kleptography, even if they are claimed to be secure. Testing these devices is extremely hard, since on one hand a major design goal is to disable external access to the device, and on the other hand the output of a contaminated device might be indistinguishable from the legitimate one if a secret data is unknown.

We discuss a couple of simple and cheap techniques that may be used as a defense against such threats.

1. Introduction

1.1 Problem statement

One of the weakest points in cryptographic systems today is protection of secrets such as cryptographic keys. Since general purpose software systems do not offer sufficient security guarantees, special purpose hardware devices such as smart cards are used. In order to protect the secrets stored in such a device it should prevent any access to the data stored inside except for the access defined by the cryptographic protocol, in which the device participates.

This is a reasonable approach, but has a severe drawback. It also prevents verification what is happening inside the device. It turn out that this may lead to severe security threats: Young and Yung [8–11] present kleptography-attack based on modifications of standard algorithms executed by the device that enable the device to leak the secrets to the manufacturer (and only to him). The manufacturer needs only to eavesdrop cipher-texts generated by the device and sent by public channel according to the protocol.

This technique is extremely dangerous since no additional bits are sent and the output of a contaminated device is computationally indistinguishable from the output of a clean device. Even if analyzing the internal code of such a

2000 Mathematics Subject Classification: 94A60, 11T71.
Keywords: cryptographic protocols un-trusty cryptographic devices.

device may give an evidence of a presence of kleptographic code, it cannot help to intercept secrets transmitted by the device (so, from the point of view of dishonest manufacturer, it is much better solution than subliminal channel).

Kleptography works particularly well for discrete-log cryptosystem. For example the contaminated Diffie–Hellman protocol executed by Alice (Bob) looks as follows [10]:

Let X be the private key of the manufacturer and let $Y = \alpha^X \bmod p$ be his public key (not published, but to be used inside contaminated devices). Assume that $H\colon \mathbb{Z}_p \to \mathbb{Z}_{p-1}$ and let W, a, b be fixed odd integers.

1. for the first usage, generate random $c_1 \in \mathbb{Z}_{p-1}$;
2. return $m_1 = \alpha^{c_1} \bmod p$;
3. store c_1 in memory for the next usage;
4. for the second usage, generate random $t \in \{0, 1\}$;
5. $z := \alpha^{c_1 - Wt} \cdot Y^{-ac_1 - b} \bmod p$;
6. $c_2 := H(z)$;
7. return $m_2 = \alpha^{c_2} \bmod p$.

So when device is used for the first time the exponent c_1 is generated at random. The next time instead of a random value the device uses m_1 from previous round. Of course, putting $c_2 = m_1$ would be easily detectable. So the code uses value z which depends on $\alpha^{c_1} = m_1$ and is blinded by Y^{c_1}. Notice that $Y^{c_1} = (\alpha^x)^{c_1} = m_1^x \bmod p$, so the blinding factor can be easily computed by the owner of x. More precisely, if the attacker has intercepted m_1 and m_2, he can calculate c_2 as follows:

1. $r := m_1^a \cdot \alpha^b \bmod p$;
2. $z_1 := m_1 / r^X \bmod p$;
3. if $m_2 := \alpha^{H(z_1)} \bmod p$, then return $H(z_1)$;
4. $z_2 := z_1 / \alpha^W \bmod p$;
5. if $m_2 := \alpha^{H(z_2)} \bmod p$, then return $H(z_2)$.

Having the value of c_2, the attacker can easily calculate the secret key generated in the second key exchange.

For the owner of the private key x, c_2 is a simple function of m_1. So in order to simplify the description we assume that $c_2 = \mathcal{F}(m_1)$ for the rest of the paper.

The only way to detect kleptographic code is to use side-channel analysis, such as stochastic analysis of computation time [5] using the fact that different number of exponentiations must change stochastic characteristics of computation time. However, there are countermeasures that can be deployed by the manufacturer.

What can we do, when we cannot control the manufacturer or are forced to use the devices that are already on the market? We discuss here a few simple techniques that are useful in this case. We work under the following assumptions:

- no special devices are to be used,
- if a protocol is executed by two parties, then one party must be able to defend itself without enforcing the second part to change its behavior.

Organization of the paper. The paper is composed of three sections: the first one discusses key agreement, the second one is on public key encryption algorithms, the last one deals with signature schemes. Additionally, the rest of this section contains notations and primitives used in further parts.

1.2 Preliminaries

Throughout the paper we use the following notations: D_i stands for the ith cryptographic black box device used; $\mathbb{Z}_n$ denotes the set $\{0, 1, \ldots, n-1\}$ with addition and multiplication modulo n. If n is prime, we use symbol p instead of n and assume that *discrete-log problem* is hard for p: for a given generator α of $\mathbb{Z}_p^*$ and a (random) number $\beta \in \mathbb{Z}_p^*$ it is computationally infeasible to find k such that $\alpha^k = \beta \bmod p$. Diffie-Hellman problem is closely related to discrete--log and it is defined as follows: for given prime number p, generator α of $\mathbb{Z}_p^*$ and numbers x_1, x_2 find $\alpha^{k_1 k_2}$, where $x_1 = \alpha_1^k \bmod p$ and $x_2 = \alpha_2^k \bmod p$.

Let us formulate the following task closely related to Diffie-Hellman problem:

Task 1. *Assume that p is a prime number and α is a generator of $\mathbb{Z}_p^*$. Let α_1, p_1 and α_2, p_2 be chosen such that p_i is a large prime and α_i is a generator of $\mathbb{Z}_{p_i}$, for $i = 1, 2$. Then for any $x_1 = \alpha_1^{k_1} \bmod p_1$ and $x_2 = \alpha_2^{k_2} \bmod p_2$, compute $x = \alpha^{k_1 k_2} \bmod p$.*

Let us remark that solving Task 1 for arbitrary parameters would give us a solution to Diffie-Hellman problem for p, where $x = \alpha^{c_1} \bmod p$ and $x' = \alpha^{c_2} \bmod p$ are exchanged during the Diffie-Hellman protocol. This is done as follows:

- choose p_1, α_1 and p_2, α_2 so that Task 1 can be solved;
- for $x_1 = x$ take $x_2 = \alpha_2^1 \bmod p_2$ and compute $y = \alpha_1^{c_1 \cdot 1} = \alpha_1^{c_1} \bmod p_1$ using Task 1 for p, p_2 and p_1;
- reverse the roles of p_1 and p_2 for x' and compute $y' = \alpha_2^{c_2} \bmod p_2$;
- execute Task 1 for p_1, p_2 and p for y, y' and compute $x'' \in \mathbb{Z}_p$ such that $x'' = \alpha^{c_1 \cdot c_2} \bmod p$.

2. Diffie–Hellman key exchange with un-trusty devices

Let us fix notation for Diffie–Hellman protocol [1]: Alice and Bob agree upon initial parameters p, large prime, and α, a generator in $\mathbb{Z}_p$. Then they proceed as follows:

Alice	**Bob**
1. generate random $a < p-1$,	1. generate random $b < p-1$,
2. $x \leftarrow \alpha^a \bmod p$,	2. $y \leftarrow \alpha^b \bmod p$,
3. send x to Bob,	3. send y to Alice,
4. $k \leftarrow y^a \bmod p$;	4. $k \leftarrow x^b \bmod p$.

Since security of the scheme depends on quality of random exponent a (the external observer should not be able to derive it), it is desirable to execute at least Step 1 inside a cryptographic device. So let *DH-device* be the black-box device used by Alice for this purpose. The problem is that a contaminated DH--device [9–11] uses the value a that depends on the messages already sent by the device through the channel (this makes it possible to derive a from these messages for anyone who knows an appropriate key).

We propose two solutions which prevent the dangers mentioned. Both of them allow to use contaminated devices without loosing security.

Solution 1 (*Diffie-Hellman*). Alice uses two DH-devices D_1, D_2. First she initiates all of them with the same parameters p and α. Then she proceed as follows:

1. compute $x_i \leftarrow \alpha^{k_i} \bmod p$ using D_i for $i = 1, 2$;
2. compute $x \leftarrow x_1 x_2 \bmod p$ (on a PC) and send it to Bob;
3. after getting y from Bob compute $z_i \leftarrow y^{k_i} \bmod p$ using D_i for $i = 1, 2$;
4. compute the final key $z \leftarrow z_1 z_2 \bmod p$ on a PC.

Observe that $x = x_1 x_2 = \alpha^{k_1+k_2}$ and $z_1 z_2 = y^{k_1+k_2}$ so for some random k used by Bob the final key is $\alpha^{k(k_1+k_2)}$.

Lemma 1. *Solution 1 prevents kleptographic attacks on Diffie-Hellman key exchange.*

P r o o f. First observe that if at least one of k_1, k_2 is random, then $k_1 + k_2 \bmod (p-1)$ is random. So if at least one of devices D_i is secure, then whole computation is secure. So for the rest of the proof we assume that both devices are contaminated. Assume that the attacker knows the devices and he is able to see all values sent through public channels. So he gets x. Consider the model where all devices are identical (if the devices are of different manufacturers, then the output of one device is random from the point of view of the manufacturer of the second device and the previous case applies). Then from the first round the attacker gets $x^{1)}$ and knows that it has the form $\alpha^{k1)} = \alpha^{k_1^{1)}+k_2^{1)}}$, but knows neither $k^{1)}$ nor the numbers $k_i^{1)}$ for $i = 1, 2$. Then he eavesdrops the second round and gets $x^{2)}$. Observe that $x^{2)} = \alpha^{k^{2)}}$ and because devices are infected $k^{2)}$ depends on $k^{1)}$. Notice that $k^{2)} = \alpha^{\mathcal{F}(k_1^{1)})} + \alpha^{\mathcal{F}(k_2^{1)})} \neq \alpha^{\mathcal{F}(k^{1)})} \bmod p$.

It follows that to perform the kleptographic attack the attacker has to solve the following problem: find $r = u + v \bmod (p-1)$ provided that $w = u \cdot v \bmod p$ is known. Obviously, for each u we may solve $w = u \cdot v \bmod p$ for v and substitute it in $u + v \bmod p - 1$ and get one solution for r in this way. However, for how many numbers u it is possible to get the same r? Assume that it happens for u, v and u', v'. Then w.l.o.g $u' = u + \delta$ and $v' = v - \delta$, where $\delta < p - 1$. Then $(u + \delta)(v - \delta) = w = uv \bmod p$, so $\delta(\delta + u - v) = 0 \bmod p$ which yields $\delta = v - u \bmod p$. Hence at most two different numbers u lead to the same solution r. It follows that that the attacker cannot say much about $u + v \bmod (p-1)$ even if he knows $uv \bmod p$. □

In order to show that one has to be very careful in designing the protocols for contaminated devices we discuss the following incorrect solution:

Solution 2 (*Incorrect solution for Diffie-Hellman*). We assume that it impossible to modify the α parameter of device. Now the procedure executed by Alice looks as follows:

1. set the generator to α in D_1;
2. compute $x_1 \leftarrow \alpha^{k_1}$ using D_1;
3. set a generator of D_2 to x_1;
4. compute $x_2 \leftarrow x_1^{k_2}$ using D_2;
5. send x_2 to Bob and obtain y (note that $x_2 = \alpha^{k_1 k_2}$);
6. put y into D_2 and compute $y_2 \leftarrow y^{k_2}$;
7. put y_2 into D_1 and compute the key $y \leftarrow y_2^{k_1}$.

Observe that finally the key $y = y_2^{k_1} = y^{k_1 k_2} = \alpha^{k k_1 k_2}$. The above solution looks fine: the products in the exponent hide k_1 and k_2. However this is not true. Assume that we use two contaminated devices and both work as described and the protocol is executed three times. The attacker overhears the parameters sent which are $x_2^{1)}, x_2^{2)}, x_2^{3)}$ where the upper index states for the round number. Then the following equations hold

$$x_2^{1)} = \left(x_1^{1)}\right)^{k_2^{1)}},$$
$$x_2^{2)} = \left(x_1^{2)}\right)^{k_2^{2)}} = \left(x_1^{2)}\right)^{\mathcal{F}(x_2^{1)})},$$

$$x_2^{3)} = \left(x_1^{3)}\right)^{k_2^{3)}} = \left(x_1^{3)}\right)^{\mathcal{F}(x_2^{2)})}.$$

Then the attacker can compute $x_1^{2)}$ and $x_1^{3)}$:

$$x_1^{2)} = \left(\mathcal{F}(x_2^{2)})\right)^{f_1} \bmod p \qquad \text{and} \qquad x_1^{3)} = \left(x_2^{3)}\right)^{f_2} \bmod p$$

for $f_i = \left(x_2^{i)}\right)^{-1} \bmod (p-1)$. Since D_1 is contaminated now, the attacker gets $x_1^{3)} = \alpha^{x_1^{2)}}$ and can compute $x_1^{2)} \cdot x_2^{2)} \bmod (p-1)$, which is the secret of the third round.

3. Encryption protocols

3.1 ElGamal encryption

First we recall the ElGamal encryption algorithm [2]. The initial parameters for this algorithm are p which is a large random prime number, and α which is a generator of $\mathbb{Z}_p$. The private key of Alice is a random number $a \in \mathbb{Z}_p$. Her public key y is given by equation $y = \alpha^a \bmod p$.

Assume that Bob would like to send a message m to Alice so that nobody can read it. First he fetches the public key y and the parameters p and α. Then he performs following operations:

1. pick a random integer k such that $0 < k < p-1$,
2. compute $r \leftarrow \alpha^k \bmod p$,
3. compute $s \leftarrow m \cdot y^k \bmod p$.

The pair (r, s) is a ciphertext of m. Alice deciphers it by computing $r^{-a} \cdot s \bmod p$, which equals $(\alpha^k)^{-a} \cdot m \cdot y^k \bmod p = m \cdot \alpha^{-ak} \cdot (\alpha^a)^k \bmod p = m \bmod p$.

A kleptographic attack against ElGamal encryption is analogous to the one for Diffie–Hellman. We can defend ourselves in the following way:

Solution 3 (*ElGamal Encryption*). One can apply technique of combining two ElGamal ciphertexts from [3]:

(1) compute a ciphertext (r', s') of m using device D;

1. compute a ciphertext (r'', s'') of message 1 on a PC;
2. compute $r \leftarrow r' \cdot r'' \bmod p$ and $s \leftarrow s' \cdot s'' \bmod p$;
3. send (r, s) to Alice.

The method is based on a simple phenomenon that (r, s) is a ciphertext of $m \cdot 1 = m \bmod p$ ([3]).

If either PC or D is secure, then $r = \alpha^k$, where k is random. So the attacker has no advantage of knowing the discrete-log of either r' or r''. However, since

PC is by definition un-trusty, we should implement this algorithm with at least two devices:

Solution 4 (*ElGamal Encryption*). Assume that we have devices D_1, D_2. First we initiate all of them with the same parameters p, α and the public key y. In order to encrypt a message m we perform the following protocol:

1. find messages m_1, m_2 such that $m = m_1 m_2 \bmod p$ (for instance $m_1 = m \cdot u \bmod p$, $m_2 = u^{-1} \bmod p$ for a random u);
2. compute $(r_i, s_i) \leftarrow \mathrm{Enc}_{D_i}(m_i)$ for $i = 1, 2$, where Enc_D denotes encryption with device D;
3. compute $r \leftarrow r_1 r_2 \bmod p$ on a PC;
4. compute $s \leftarrow s_1 s_2 \bmod p$ on a PC.

In this case we use a PC despite it was considered to be un-trusty. However, on a PC we perform operations. Moreover, a PC does not generate any random parameters and since it has no access to private keys it cannot betray them. The only thing a PC could do would be to pass some parameters, say r_1, s_1. However, sending extra messages goes beyond kleptographic model we are talking about.

Lemma 2. *Solution 4 prevents kleptographic attacks against ElGamal encryption.*

Proof. Observe that $r_i = \alpha^{k_i}$ for $i = 1, 2$ and $r = \alpha^{k_1 + k_2} \bmod p$. The second element of ciphertext pair is $s = m_1 y^{k_1} m_2 y^{k_2} = m y^{k_1 + k_2} \bmod p$.

Kleptography implemented on devices D_1, D_2 yields $k_1^{(2)} = r_1^{(1)}$ and $k_2^{(2)} = r_2^{(1)}$. But the attacker knows only $r^{(1)} = r_1^{(1)} \cdot r_2^{(1)} \bmod p$. As in the proof of Lemma 1, the attacker has to derive $k^{(2)} = r_1^{(1)} + r_2^{(1)} \bmod (p-1)$ based on the known product of $r_1^{(1)} \cdot r_2^{(1)} \bmod p$. But this is impossible, as we have seen before. □

It is an interesting question whether it is possible to replace the sum of random exponents (chosen by the devices) in the final result by another function of these exponents, for instance by their product. The answer is partially positive:

Solution 5 (*ElGamal Encryption*). We assume that it is possible to set the α parameter of a device. Our goal is to replace device D_2 of Solution 4 by two devices D_2' and D_2''. They cooperate as follows to simulate D_2:

1. D_2' computes (r_1, s_1), a ciphertext of 1;
2. we set the parameter α of D_2'' to r_1 and the public key to s_1;
3. D_2'' computes (r_2, s_2) as a ciphertext of m_2.

Note that $r_2 = r_1^{k_2''} = (\alpha^{k_2'})^{k_2''} = \alpha^{k_2' \cdot k_2''}$ and $s_2 = m \cdot s_1^{k_2''} = m \cdot (y^{k_2'})^{k_2''} = m \cdot y^{k_2' \cdot k_2''}$.

Of course, as already seen for Diffie-Hellman protocol, cascading technique used in Solution 4 cannot be applied alone. Are there any other ways to get products of random exponents chosen by the devices?

First consider the case when both devices have the same parameters p, α, y and the final ciphertext is to be formed for these parameters. Assume that there is an algorithm A which computes (r,s) based on (r_1, s_1) and (r_2, s_2). We use A to break Diffie-Hellman protocol. First we choose a random a and set $y = \alpha^a \bmod p$. Then we eavesdrop the key exchange messages so we know two values $v_1 (= \alpha^{x_1} \bmod p)$ and $v_2 (= \alpha^{x_2} \bmod p)$. Then we set $r_1 = v_1$ and $r_2 = v_2$ and compute easily s_1, s_2, since $s_1 = mr_1^a$ and $s = mr_2^a$. Then we give these parameters to A and we get (r,s). Since $r = \alpha^{x_1 x_2}$ we have broken the Diffie-Hellman protocol.

The next trial would be to make the parameters α, α_1, α_2 different, say $\alpha_1 = \alpha^{x_1} \bmod p$ and $\alpha_2 = \alpha^{x_2} \bmod p$. Then we feed the devices with public keys $y_1 = \alpha_1^x \bmod p$ and $y_2 = \alpha_2^x \bmod p$ (determining the public keys is due to the owner of the private key, so a device must accept such keys). Then if we get ciphertexts from D_1 and D_2, then they are equal to $(r_1, s_1) = \left(\alpha_1^{k_1}, m_1 \cdot y_1^{k_1}\right)$ and $(r_2, s_2) = \left(\alpha_2^{k_2}, m_2 \cdot y_2^{k_2}\right)$ for some random k_1, k_2. But notice that $(r_i, s_i) = (\alpha^{x_i k_i}, m_i \cdot \alpha^{x \cdot x_i k_i})$. Since numbers $x_i k_i$ are random as well as numbers k_i, we are back in the previous case, when all parameters are equal.

The last hope would be to concern different p_1, p_2 and p. This is in fact the Task 1 so we cannot hope to find a general algorithm for computing $\alpha^{k_1 k_2} \bmod p$ given $\alpha_1^{k_1} \bmod p_1$ and $\alpha_2^{k_2} \bmod p_2$. The only hope is to get some special values p_1, p_2, p for which Task 1 must be solvable in only "one direction".

4. ElGamal signature protocols

The standard ElGamal signature scheme [3] looks as follows. Let p be a prime number for which discrete-log problem is hard. Let α be a generator of $\mathbb{Z}_p$. A random integer a $(1 \leq a \leq p-2)$ is a private key and $y = \alpha^a \bmod p$ is a public key. The parameters p and α are public as well. To sign a message m Alice performs the following steps:

1. choose integer k, $1 \leq k \leq p-1$, at random;
2. $r \leftarrow \alpha^k \bmod p$;
3. $s = k^{-1}(H(m) - ar) \bmod (p-1)$, where H is a secure hash function;
4. output the signature $S(m) = (r,s)$.

To verify the signature Bob checks if $\alpha^{H(m)} = y^r r^s \bmod p$.

Our goal is to generate two signatures and combine them to obtain a valid signature (r, s) of m for parameters p, α, a, y. For this purpose we use two devices that are able to create ElGamal signatures.

Quite often the signing devices are not computing hash functions, since this is an operation that could involve sending huge data to the device (such as a smart card with very slow communication channel)—this operation is executed on the PC preparing the data to be signed. Therefore it might be possible to send any chosen value to the device—the device cannot verify which value is indeed a value of hash function H.

Solution 6 (*ElGamal Signature*). Initiate device D_1 with parameters p, α and an arbitrary private key u. (A device should offer such an option of loading the keys for the case when very strong demands on randomness of the keys are enforced by law.) In fact D_1 will be used as a source of randomness. The real private key x of Alice is contained in device D_2. Signing a message m by Alice looks as follows:

- Alice sends an arbitrary hash h to D_1;
- D_1 generates a signature (r_1, s_1), where $r_1 = \alpha^{k_1} \bmod p$ and $s_1 = k_1^{-1}(h - ur_1) \bmod (p-1)$.
- Alice computes k_1 from s_1, r_1, u and h (on her PC);
- Alice sets the generator on device D_2 to r_1;
- D_2 computes signature (r_2, s_2) for message m;
- Alice presents $\big(r_2, s_2/k_1 \bmod (p-1)\big)$ as a signature of m for public parameters p, α, α^x.

According to the algorithm we obtain $r_2 = r_1^{k_2} = \alpha^{k_1 k_2} \bmod p$ and $s_2/k_1 = (k_1 k_2)^{-1}(H(m) - r_2 x) \bmod (p-1)$, so the signature is valid.

We remark here that the same does not work if D_1 and D_2 are loaded with the secret key x: even if we manage to compute somehow s, it must satisfy the equality $s = s_2/k_1 \bmod (p-1)$, and therefore k_1 could be derived. Then it would be possible to get the secret key x from s_1 computed by D_1. So ElGamal would be broken.

The solution proposed should defend against kleptographic setup: the "random" number $k_1 k_2$ depends on both devices. While the second device might use the parameter r_2 of the final signature seen by the attacker and therefore make k_2 computable for the attacker at the next round, the first device does not know any part of the final signature which is blinded by k_2, and therefore cannot relate a new k_1 to the previous signature.

Conclusions and open problems

There are many open questions that gain importance in the context of cryptographic devices. We have shown how to prevent a specific kind of kleptographic attack. However, it does not mean that it excludes any kind of kleptography, yet unknown one.

An other example of such a question is the following: given two ElGamal signatures of the same m composed using the same private key x. Is it possible to construct a third signature of m valid for key x without using the key x? While getting such a signature for a different message m is regarded as a severe fault of a signature scheme, deriving a signature for the same message could be very helpful as a defense against kleptographic attacks.

REFERENCES

[1] DIFFIE, W.—HELLMAN, M.: *New directions in cryptography*, IEEE Trans. Inform. Theory **22** (1976), 644–654.

[2] ELGAMAL, T.: *A public–key cryptosystem and a signature scheme based on discrete logarithms*, in: CRYPTO '84, Lecture Notes in Comput. Sci., Vol. 196, Springer-Verlag, Berlin, 1985, pp. 10–18.

[3] JAKOBSSON, M.—JUELS, A.: *Mix and match: Secure function evaluation via ciphertexts*, in: ASIACRYPT '00, Lecture Notes in Comput. Sci., Vol. 1976, Springer-Verlag, Berlin, 2000, pp. 162–177.

[4] KOCHER, P. C.: *Timing attacks on implementations of Diffie–Hellman, RSA, DSS, and other systems*, in: CRYPTO '96, Lecture Notes in Comput. Sci., Vol. 1109, Springer-Verlag, Berlin, 1996, pp. 104–113.

[5] KUCNER, D.—KUTYLOWSKI, M.: *Stochastic kleptography detection*, in: Proc. Public-Key Cryptography Comput. Number Theory (Warszawa, Sept. 11–15 2000), Walter de Gruyter, Berlin, 2001, pp. 137–149.

[6] RABIN, M. P.: *Digitalized signatures and public-key functions as intractable as factorization*, Tech. rep. MIT/LCS/TR-212, Laboratory for Computer Science, MIT, January 1979.

[7] SIMMONS, G. J.: *The subliminal channel and digital signatures*, in: EUROCRYPT '84, Lecture Notes in Comput. Sci., Vol. 209, Springer-Verlag, Berlin, 1985, pp. 364–378.

[8] YOUNG, A.—YUNG, M.: *The dark side of black-box cryptography*, in: CRYPTO '96, Lecture Notes in Comput. Sci., Vol. 1109, Springer-Verlag, Berlin, 1996, pp. 89–103.

[9] YOUNG, A.—YUNG, M.: *The prevalence of kleptographic attacks on discrete-log based cryptosystems*, in: CRYPTO '97, Lecture Notes in Comput. Sci., Vol 1294, Springer-Verlag, Berlin, 1997, pp. 264–276.

[10] YOUNG, A.—YUNG, M.: *Kleptography: Using cryptography against cryptography*, in: EUROCRYPT '97, Lecture Notes in Comput. Sci., Vol. 1233, Springer-Verlag, Berlin, 1997, pp. 62–74.

[11] YOUNG, A.—YUNG, M.: *Bandwidth-optimal kleptographic attacks*, in: CHES '01, Lecture Notes in Comput. Sci., Vol. 2162, Springer-Verlag, Berlin, 2001, pp. 235–250.

Received October 13, 2003

Daniel Kucner
Institute of Computer Science
University of Wrocław
Przesmyckiego 20
Pl–51-151 Wrocław
POLAND
E-mail: Daniel@ii.uni.wroc.pl

Mirosław Kutyłowski
Institute of Mathematics
University of Technology, and CC Signet
Wybrzeże Wyspiańskiego 27
PL–50-370 Wrocław
POLAND
E-mail: Miroslaw.Kutylowski@pwr.wroc.pl