**Control and Cybernetics**

vol. **42** (2013) No. 4

# Digital signature with secretly embedded warning*†

by

**Konrad Durnoga, Jacek Pomykała and Tomasz Trabszys**

Faculty of Mathematics, Informatics and Mechanics
University of Warsaw, Banacha 2, 02-097 Warszawa, Poland
kdr@mimuw.edu.pl

**Abstract:** We present a digital signature scheme with secretly embedded warning. The embedded warning is a protection mechanism in case of restraint or blackmail. Extending ordinary digital signatures we propose schemes where a signer, approached by a powerful adversary that demands handing over a signing key, can disclose his private key. In our solution the signer is able to generate a feigned key indistinguishable from the genuine one. Then such a key can be used to embed a special warning message within a signature to indicate coercion. Such warnings can be transferred via subliminal channel to some trusted authority.

**Keywords:** public key cryptography, digital signature, one-way permutation, subliminal channel,deniable encryption, blackmail, secretly embedded warning, translucent sets

## 1. Introduction

### 1.1. Motivation

In the classical approach to digital signatures, the security of a scheme relies exclusively on secrecy of signer's private key. If this key is compromised, then signatures are no longer secure. No security specialist disputes words of the world's most famous convert hacker Kevin Mitnick: people are the weakest link in the system security. Users are vulnerable to a number of attacks employing social engineering, computational power advantage or more sophisticated black-bag cryptoanalysis. Often, people do not even realize that their most precious data – private key, is stolen. And little can be done by theoretical cryptographers to prevent such scenarios from happening. Alas, recent years have brought worrisome examples of data coercion where individuals are perfectly aware of this as information is forcibly seized. In the most extreme case, keys can be

*The work was partially supported by the grant from National Center for Research and Development no 023/R/ID3/2012/02.

†Submitted: January 2013; Accepted: October 2013

literally beaten out from victims. This is euphemistically called *the rubber-hose cryptoanalysis*. Other than that, legal regulations in certain countries, like *The Regulation of Investigatory Powers Act* in the UK, entitle appropriate authorities to demand handing over encryption keys. With a caveat that refusal may be penalized, e.g., with a few years of imprisonment.

A handful of practical solutions that can protect users in the aforementioned scenarios have been proposed. These, for instance, include encrypting filesystems with an additional feature of hidden volumes. Using the popular TrueCrypt software one can easily create one or more decoy filesystem layers, with an encrypted material of minor importance that can be revealed to adversary to deceive him, and a proper layer with some valuable data beneath. It works like a double-bottom, or rather multi-bottom, hat where only the upper bottom is exposed if needed. This can be viewed as some kind of steganography that hides the very fact that some classified data is in play. A similar approach is taken in the Rubberhose filesystem (Dreyfus, no date) which is specially suited for political dissidents susceptible to rubber-hose cryptoanalysis. A fresh new idea has been introduced by Geambasu et al. (2009). The authors present a distributed self-destructing storage where data is encrypted using some random key, then the key is split into pieces using the well-known secret sharing schemes, and these parts get spread across a P2P network. The original key is discarded, and so a user has to download pieces to reconstruct his files. Moreover, distributed key parts are erased after some timeout and afterwards the encrypted data is rendered useless.

A somewhat more theoretical treatment on this matter has been given by Canetti et al. (1996)(see also the work by Klonowski et al., 2008). They propose an equivocal encryption where an encryptor is offered a possibility to open a ciphertext in multiple ways. In particular, when approached by an attacker, one can disclose a counterfeit key and claim that it is genuine with only a slim risk that the cat is out of the bag and the attacker knows the hoax. This is called *a deniable encryption*. We elaborate on this topic in the forthcoming sections.

### 1.2. Our solution

In this paper we address a problem that is quite similar to the one solved by a deniable encryption. But it exists in the other part of the public key cryptography, i.e., in the world of signature schemes. We consider what could happen if, at a certain moment, a signer, say, Alice would be somehow forced to hand over her private signing key to an adversary – a party that knows the signature scheme and all former signatures issued by the signer? Or to issue valid signatures on messages of the adversary's choice? The main purpose of ours is to give a signature scheme with the core property of being blackmail secure. Namely, Alice should be able to transfer a message to some trusted authorities subliminally alerting that a signature has been coerced. Such "a call for help" must be completely hidden from the point of view of the adversary, which can be quite a powerful entity. Thus, we introduce a notion of an *embedded secret*

*signature* (ESS).

We summarize several informal requirements for an embedded secret signature scheme below. Such a scheme should satisfy the following conditions:

- an ordinary signature receiver, say, Bob is able to verify correctness of a signature, but both types of signatures are considered valid by him: these assembled voluntarily and these coerced by an adversary;
- when Alice is forced by the adversary to create a signature, she can leak information subliminally that the signature is coerced;
- nobody, even the signer, except for some fixed trusted authority is able to extract the information about coercion from a signature; in particularly, the adversary cannot distinguish between both types of signatures even under the assumption that he possesses signatures issued in the past by Alice;
- with an overwhelming probability the coercer cannot craft a signature that is considered as a voluntary one by the trusted party (even though the adversary might be able to produce a correct signature, which is acceptable for Bob, on an arbitrary message), notwithstanding the fact of having access to previously intercepted signatures issued by Alice.

Clearly, no cryptographic solution can prevent the adversary from blocking the entire communication between Alice and the trusted party. However, it is reasonable to assume that the adversary wants to make use of the coerced signatures and will present them to some third party, e.g., a bank. In this case that party can contact the trusted authority to verify whether these signatures are legitimate.

The idea leading to our solution scheme that meets requirements sketched above is to establish a new, shared key $K$ between Alice and the trusted authority. The ability of hiding an embedded secret in a signature is assured if only $K$ is kept secret. Still, the adversary can demand $K$ from Alice but now she may present a fake key instead of the real one. If the data Alice gives to the adversary is coherent, then by no means can he tell whether the given key matches $K$ or not. Thus, Alice risks nothing here.

A downside of the solution we propose is that it introduces a substantial overhead factor and signatures produced by our scheme may be quite long in order to ensure an acceptable level of security for the signer. This disadvantage is inherited from the construction of a deniable encryption by Canetti et al. (1996), which we use internally. A problem of constructing a more efficient deniable encryption scheme, that offers sufficiently high security and deniability, remains open (see the discussion at the end of Section 2. Our security proof works in the random oracle model (Bellare and Rogaway, 1993).

### 1.3. Related work

In the recent years several concepts of digital signature security that concern compromising signer's private key have been developed. One idea avails evolution of secret keys in subsequent periods of time. The adversary that seized

signer's secret key in one period is not able to forge signatures assigned to the previous period. Such schemes, called *forward secure signatures*, defined by Bellare and Miner (1999), only include a secret key updating algorithm and the corresponding public key remains unchanged. A somewhat more general approach was presented in the work of Itkis (2003), where the idea of key evolving signature was considered. Here, an additional divergence test was introduced. It can provide a suitable tamper evidence proving key exposure. The construction is generic and relies on combinatorial methods, but the distinction between forger-generated and legitimate signatures is not possible.

On the other hand, we may have a forgery of signature which is not caused by key coercion but rather the computational power of the adversary. In such an approach the idea of the fail-stop signatures has been invented by Pfitzman and Waidner (1991). Here, the legitimate signer can construct only one of many possible signatures. The adversary can generate all the signatures but can not determine which one is produced by the legitimate signer. In this connection security of the signature can be broken if either the legitimate signer can construct a signature that he can later prove to be a forgery, or the adversary with an unbounded computational power succeeds to construct a signature such that the legitimate signer can not prove to be counterfeit. Obviously, the fail-stop signature does not offer any help if signer's secret key is exposed to the adversary.

Another approach addresses the so called coercive exposures of private key. The monotone signature schemes (Naccache, Pointcheval and Tymen, 2002) allow the verification algorithm to be updated after attack so that signatures forged by the adversary are not valid according to the current verification algorithm. The solution proposed by Håstad et al. (2000) applies a subliminal channel to inform a receiver that a signature was issued under duress. The corresponding protocol was called *backward-malleable funkspiel scheme*. Here, in the key setup algorithm two secret keys $s_1$, $s_2$ and a sequence of random bits are generated and shared between the signer and the receiver. The receiver uses a key swapping algorithm to replace the correct configuration of keys by a fake one. A verification algorithm allows the receiver to decide whether a subsequent message $m_i$ was signed with the aid of the corresponding secret key $s_i$. This scheme, although very practical, admits, after a break-in, some forgeries with a non-negligible probability. Consequently, it does not satisfy our security requirements formulated in Sections 3.1.1 and 3.1.2 below. It is worth saying that a very attractive digital signature with secretly embedded warning scheme based on the Diffie–Hellman key exchange idea is proposed independently by Kubiak and Kutyłowski (2013).

### 1.4. Paper outline

First, we briefly recall the deniable encryption scheme. The ideas from the work of Canetti et al. (1996), of key importance for our paper, are summarized in Section 2. Then we describe a security model for ESS in Section 3 and present

the signature scheme in Section 4. The soundness proof follows in Section 5. Plugging the deniable encryption into the well-known subliminal channel construct of Simmons yields an example of a modified ESS in Section 6. Some notes on possible applications of ESS form the final part of this work.

## 2. Deniable encryption

Our scheme is built upon a scheme of deniable encryption. This neat concept has been introduced by Canetti et al. (1996). Their motivation flows from the following scenario. Suppose that Alice encrypts a message and sends it to Bob. After some period of time she can be asked or even forced, i.e., coerced by an adversary that demands information by force or being obliged by some law-abiding authorities to do so, to unveil a plaintext message together with all random choices involved in the encryption process. The authors are concerned in the situation, and this is quite similar to our case, where Alice, for her own good, cannot refuse. She has to obey and present some at least real-looking data. Ideally, Alice would be able to craft some new fake message that fits to the original one. The "real-looking" term means that the adversary gets persuaded by this data, and he does not realize that plaintext Alice has given him is a fake one. In particularly, a senseless content is not quite plausible here. Also note that all the ciphertexts previously transmitted to Bob could have been intercepted by the attacker, and so it may serve as some kind of commitment. The adversary can easily verify if data Alice has presented matches the ciphertext.

Let $\mathcal{M} = \{0,1\}^\ell$, for some parameter $\ell$, be a space of plaintext messages. The explicit statement of the problem sketched above is: given a (deterministic) encryption algorithm $E\colon \mathcal{M} \times \{0,1\}^* \to \{0,1\}^v$ (such that $E(m,r)$ is a ciphertext of a message $m$ using a pregenerated randomness $r \in \{0,1\}^*$ for some fixed parameter $v$) and two distinct messages $m_1, m_2 \in \mathcal{M}$, is it possible to compute $r_1, r_2 \in \{0,1\}^*$ such that $E(m_1, r_1) = E(m_2, r_2)$? In fact, we want to allow picking a fake message *a posteriori*, i.e., not at the time a ciphertext is generated, but rather when an adversary demands to reveal the data. Also the fake message should have some reasonable content as it is unlikely that an attacker accepts a senseless message even if the resulting ciphertext matches the one produced from the real message. So, the question becomes: given $E$, $m_1 \neq m_2$, $r_1$, is it possible to generate $r_2$ such that $E(m_1, r_1) = E(m_2, r_2)$?

At the first sight it seems unlikely that the answer be 'yes'. One can point out some obvious obstacles like: how can we assure that a receiver is able to correctly decrypt a ciphertext if two distinct messages can be encrypted to the same string — how can he distinguish between them? However, it turns out that, under some additional assumptions, construction of a scheme that meets the deniable encryption requirements presented above is obtainable. Canetti et al. (1996) have shown how an ingenious idea of translucent sets can be applied to create a provably secure deniable encryption scheme. Yet, it should be noted that there exists some practical difficulties that may prevent such a construction from being adopted.

We now give an informal definition of a translucent set and show, following the original work of Canetti et al. (1996), deniable encryption algorithms built upon such a set. We say that a function $\epsilon = \epsilon(t)$ is *negligible* in a security parameter $t$ if $\epsilon(t)$ vanishes faster than the inverse of any polynomial of $t$.

Definition 2.1 *Let $t$ be a parameter and $d(t)$ be a polynomial. A set $S_t \subset \{0,1\}^t$ together with an associated trapdoor information $d_t \in \{0,1\}^{d(t)}$ is said to be translucent iff for some sufficiently large $l = l(t)$ all of the following four assertions hold:*

- $\#S_t \leq 2^{t-l}$;
- *there exists a polynomial time algorithm that returns a random element from $S_t$, such that the distribution of algorithm's output is indistinguishable from the uniform one;*
- *for any polynomial-time distinguisher $\mathcal{D}$ the probability that $\mathcal{D}$ tells apart a random element of $\{0,1\}^t$ from random element of $S_t$ is $1/2 + \epsilon$, where $\epsilon$ is negligible in $t$;*
- *there exists a deterministic polynomial-time algorithm that given $x \in \{0,1\}^t$ and $d_t$ decides whether $x \in S_t$.*

Intuitively, $S_t$ is a small set, when compared to the size of $\{0,1\}^t$, and one can effectively generate random elements of $S_t$. On the other hand, $S_t$ should be large enough to make enumerating all elements from $S_t$ virtually impossible. Moreover, without knowing the secret $d_t$ it is infeasible to distinguish random elements from $\{0,1\}^t$ and $S_t$. However, using the trapdoor information $d_t$, one can easily tell if a given element belongs to $S_t$ or not.
There are several constructions of sample translucent sets proposed by Canetti et al. (1996). Two of them rely on the existence of a trapdoor permutation, say $f\colon \{0,1\}^s \to \{0,1\}^s$ for some $s$ that will be determined shortly. Let $B\colon \{0,1\}^s \to \{0,1\}$ denote a corresponding hard-core bit predicate for $f$. Now, if we relate $t$, $s$ and $l$ in such way that $t = s + l$, where $t$ and $l = l(t)$ have exactly the same meaning as in Definition 2.1, we can define

$$S_t = \big\{x_0\|b_1\|b_2\ldots b_l \in \{0,1\}^t\colon \quad x_0 \in \{0,1\}^s, b_1, b_2, \ldots, b_l \in \{0,1\}, \\ B\big(f^{(-i)}(x_0)\big) = b_i \quad \text{for each } i = 1, \ldots, l\big\},$$

where $f^{(-i)} \coloneqq \underbrace{f^{-1} \circ \cdots \circ f^{-1}}_{i \text{ times}}$ is a composition of permutation inverse $f^{-1}$.
Note that it is easy to generate a random $x \in S_t$ — it suffices to pick $x_l \in \{0,1\}^s$ at random and to compute

$$x_{i-1} \coloneqq f(x_i) \quad \text{and} \quad b_i \coloneqq B(x_i) \tag{1}$$

for $i = l$ down to 1 iteratively. Then $x_0\|b_1\|b_2\|\ldots\|b_l \in S_t$.

The translucent set's property of being indistinguishable from random elements manifests its potential when applying it to deniable encryption. Let us

fix $S_t$ temporarily. In the basic scheme, the encryption algorithm simply splits a plaintext message $m \in \mathcal{M} = \{0,1\}^\ell$ into independent bits and the encryption is conducted for each bit separately (resulting ciphertexts are concatenated to get ciphertext for $m$). In order to generate a ciphertext for bit $b = 0$ the algorithm simply outputs a random element from $\{0,1\}^t$, whilst for $b = 1$ it should return a random element from $S_t$. Throughout this paper we refer to strings drawn uniformly from $\{0,1\}^t$ and from $S_t$ as $R$-elements and $S$-elements, respectively.

We keep in mind that the encryption function introduced in Section 2 was $E\colon \mathcal{M} \times \{0,1\}^* \to \{0,1\}^v$, where the second parameter corresponds to source of randomness and there is no additional randomness in $E$. Here we change it slightly and replace $\{0,1\}^*$ with $\{0,1\}^{\ell t}$. Encrypting $b = 0$ simply means returning a corresponding substring of randomness passed to $E$. For $b = 1$ this substring may serve as a seed for a generator returning $S$-elements, e.g., by using (1).

Decrypting a given ciphertext for any party possessing $d_t$ is straightforward and boils down to determining whether given string is an $R$- or $S$-element. It is worth noting that decrypting an encrypted $b = 1$ will always succeed, but for $b = 0$ it may happen, with probability $1/2^l$, that the decryption will be erroneous, since an $R$-element can also belong to $S_t$. Opening an encryption means presenting all random choices used to generate a target ciphertext and pointing whether given string has been picked out as an $R$- or $S$-element. When a ciphertext for $b = 1$ is opened, a convincing proof of the fact that the resulting element was drawn from $S_t$ can be carried by Alice. This is be done by unveiling $x_0$ and $b_1, b_2, \ldots, b_l$ computed in the backtracking process (1). Of course, there is no such a compact confirmation possible for $b = 0$. It is also clear that ciphertext can be opened dishonestly. For $b = 1$ Alice may claim that the corresponding ciphertext is an $R$-element that does not belong to $S_t$. However lying in the opposite direction is, in this basic scheme, hardly feasible (chances that an $R$-element $x$ is in $S_t$ are slim and even if it happened by coincidence, the adversary could additionally demand the aforementioned evidence that $x$ had been computed as in (1)). One can easily get around this limitation by extending the encryption scheme.

Let us fix a positive integer $n$. To encrypt a bit $b$, pick a random $j$ such that $0 \le j \le n$ and $j \equiv b \pmod 2$. The ciphertext for $b$ now consists of exactly $n$ random strings from $\{0,1\}^t$ — the first $j$ consecutive $S$-elements are followed by $n - j$ $R$-elements. The encryptor can reveal the correct $j$ and present proofs for all $S$-elements as before to persuade which bit has been encrypted. But if Alice is deceitful she can simply flip the parity of $j$ and claim that it was $j - 1$ instead of $j$ that was chosen in the first step of the encryption. Consequently, the $j$th element of a ciphertext is declared to be an $R$-element. Of course, such allegements can only be made if only the initial, i.e., the real one, value $j$ was greater than 0. Otherwise Alice can be caught red-handed.

The last outlined algorithm for encrypting a single bit with security parameters $t, s$ and $n$ is called the *parity scheme*. This encryption may be viewed as an extension of a public encryption scheme. Here, the publicly known parameters,

like the one-way permutation $f$ and its hard-core predicate $B$, given, e.g., as Boolean circuits, play the role of a public key. The trapdoor $d_t$ is, in turn, a private key. Below, we use the capital letters $E$ and $D$ to denote the encrypting and decrypting algorithms for the parity scheme without writing these keys explicitly.

The following definition, which is borrowed from Canetti et al. (1996), summarizes the properties of deniable encryption schemes and provides a criterion to compare them. Traditionally, we say that two distribution ensembles $\mathcal{U} = \{U_k\}_{k\in\mathbb{N}}$ and $\mathcal{V} = \{V_k\}_{k\in\mathbb{N}}$ are $\delta(k)$-close, for some function $\delta$ supported on natural numbers, if for every polynomial-time distinguisher $\mathcal{D}$ and large enough $k$ we have that

$$\big|\Pr\big[\mathcal{D}(U_k) = 1\big] - \Pr\big[\mathcal{D}(V_k) = 1\big]\big| < \delta(k).$$

To express the fact that $\mathcal{U}$ and $\mathcal{V}$ are computationally indistinguishable, which is when $\delta(k)$ is negligible in $k$, we write that $\mathcal{U} \stackrel{c}{\approx} \mathcal{V}$. Also, let $\mathrm{COM}(m, r_A, r_B)$ denote a communication transcription between Alice and Bob when transmitting an encrypted message $m \in \mathcal{M}$ where $r_A$ and $r_B$ are random values used by Alice (for encrypting) and Bob (for decrypting) respectively. By $\mathcal{COM}(m)$ we refer to a random variable describing $\mathrm{COM}(m, r_A, r_B)$ when $r_A$ and $r_B$ are uniformly and independently chosen.

Definition 2.2 *An encryption protocol between Alice and Bob is said to be $\delta(k)$-sender-deniable if*

- *the probability that Bob decrypts a ciphertext erroneously is negligible in $k$;*
- *for any $m_1, m_2 \in \mathcal{M}$ we have that $\mathcal{COM}(m_1) \stackrel{c}{\approx} \mathcal{COM}(m_2)$;*
- *there exists an efficient algorithm that given $\mathrm{COM}(m_1, r_A, r_B)$ and $m_1, m_2, r_A$, for any $m_1, m_2 \in \mathcal{M}$ and uniformly chosen values of $r_A$ and $r_B$, produces a value $r'_A$ such that variables*

$$\big(m_2, r'_A, \mathrm{COM}(m_1, r_A, r_B)\big) \text{ and} \big(m_2, r_A, \mathrm{COM}(m_2, r_A, r_B)\big)$$

*are $\delta(k)$-close.*

Canetti et al. (1996) proved that the following proposition about the parity scheme holds with respect to Definition 2.2:

Theorem 2.1 *If a trapdoor permutation exists then the parity scheme is a $\frac{4}{n}$-sender-deniable encryption scheme.*

The fact that the scheme offers only deniability of order $1/O(n)$ is a major disadvantage. Recently, Dürmuth and Freeman (2011) have come out with a $\mu(n)$-sender-deniable encryption scheme where $\mu(n)$ is negligible in $n$. However, Peikert and Waters (2011) have pointed out a gap in their proof and shown an attack demonstrating that the construction is incorrect. Apparently, the problem of achieving deniability of negligible level is still open.

## 3. Embedded secret signature model

### 3.1. The model

Let $SS = (\text{KeyGen}, \Sigma, V, V^*)$ be a public-key signature scheme with security parameter $k$ (for example, $k$ could be the bit length of a signature). KeyGen is a key generation algorithm, $V$ is a public verification algorithm, and $V^*$ is a verification algorithm with a trapdoor information $d_t$ known only by the trusted authority $\mathcal{T}$. Any particular signer $\mathcal{A}$ with his public-private key pair $(upk, usk)$ signs messages using the $usk$ key in $\Sigma$ algorithm. A signature obtained in this way satisfies both verification algorithms, i.e., $V$ and $V^*$. We require that $\mathcal{A}$ does not communicate with $\mathcal{T}$, except for the KeyGen algorithm. Everywhere below, we assume that the verification algorithm $V$ uses the public key $upk$, and $V^*$ may have access to both: $upk$ and $d_t$.

When referring to $V$ as a "public" algorithm we mean that any signature receiver can obtain $V$ and verify if a signature has been assembled using the private key of $\mathcal{A}$, which could have been coerced by adversary. $V^*$ enables reading a secretly embedded message and thus $\mathcal{T}$ must conceal the trapdoor $d_t$.

Let $\mathcal{E}$ be a third party, who can adaptively query $\mathcal{A}$ to honestly sign any chosen message. Then $\mathcal{A}$ is asked or forced to show his private signing key to $\mathcal{E}$, but this time $\mathcal{A}$ can try to deceive $\mathcal{E}$. A signature scheme is embedded secret secure iff it is infeasible for $\mathcal{E}$ to determine, in polynomial time in the security parameter $k$, whether the signatures forged with the obtained private key $usk'$ would satisfy the verification algorithm $V^*$.

### 3.2. Security statements

In this section we describe formal security statements. The embedded secret security is divided into two properties: embedded secret indistinguishability and embedded secret unforgeability. First, we introduce a signing $\mathcal{O}_{\mathcal{A}}$:

- $SigQuery(m)$ : the signing oracle $\mathcal{O}_{\mathcal{A}}$ returns a signature $\sigma$ of $m$ under the corresponding private key $usk$

and the description of a simulator $\mathcal{S}im$:

- $Setup(usk)$ : given a private key $usk$ and public data $\mathcal{S}im$ initializes
- $UskQuery()$ : $\mathcal{S}im$ chooses a random bit $b \xleftarrow{\$} \{0,1\}$ and returns $UskQuery(b)$
- $UskQuery(b)$ : $\mathcal{S}im$ returns $usk_b$, where $usk_0 = usk$ (honest private key) and $usk_1 \neq usk$ (dishonest private key). If $b = 1$ then $usk_1$ is generated by $\mathcal{S}im$. (From the security definitions below, for an embedded secret secure signature scheme, signatures made even with the dishonest private key must pass the public verification algorithm).

Note that $\mathcal{S}im$ does not communicate with $\mathcal{T}$. A signature scheme is embedded secret secure if there exists a simulator $\mathcal{S}im$ matching the following two requirements.

### 3.2.1. Embedded secret indistinguishability

At the beginning, a random instance of signature scheme $Params$ is generated. In the second step, private-public key pair $(upk, usk)$ is generated. The adversary $\mathcal{E}$, breaking the embedded secret indistinguishability of public key $upk$ is allowed to query a signing oracle $\mathcal{O}_{\mathcal{A}}$ and finally ask the simulator $\mathcal{S}im$ for a challenge and prove that $\mathcal{E}$ is able to guess whether the $usk_b$ given by $\mathcal{S}im$ is honest.

The advantage in breaking embedded secret indistinguishability of an adversary $\mathcal{E}$, given access to $\mathcal{O}_{\mathcal{A}}$ and a simulator $\mathcal{S}im$, is

$$AdvESI_{(\mathcal{E},\mathcal{S}im)} \coloneqq \Pr\left[\begin{array}{lcl} & & Params \xleftarrow{\$} Setup(), \\ & & (upk, usk) \xleftarrow{\$} KeyGen(), \\ \tilde{b} = b & : & \mathcal{E}^{\mathcal{O}_{\mathcal{A}}}(Params, upk), \\ & & usk_b \overset{UskQuery()}{\leftarrow} \mathcal{S}im(usk), \\ & & \tilde{b} \leftarrow \mathcal{E}(usk_b) \end{array}\right] - 1/2.$$

The probability is taken over the coin tosses of the Setup algorithm, the key-generation algorithms, of the oracle, of $b$ and the algorithm $\mathcal{E}$.

A signature scheme is embedded secret indistinguishable (with respect to the simulator $\mathcal{S}im$) if, for every polynomial-time algorithm $\mathcal{E}$, the value of $AdvESI_{(\mathcal{E},\mathcal{S}im)}$ is negligible in the security parameter $k$. In other words - there exists no polynomial-time algorithm $\mathcal{E}$ with a significant advantage $AdvESI_{(\mathcal{E},\mathcal{S}im)}$.

### 3.2.2. Embedded secret unforgeability

At the beginning a random instance of signature scheme $Params$ is generated. In the second step, private-public key pair $(upk, usk)$ is generated. The adversary $\mathcal{E}$ breaking the embedded secret unforgeability of public key $upk$ is allowed to query a signing oracle $\mathcal{O}_{\mathcal{A}}$. Finally $\mathcal{E}$ queries the simulator $\mathcal{S}im$ for dishonest private key (bit $b$ is set: $b = 1$) and forges signature for a chosen, fresh message. $\mathcal{E}$'s advantage in breaking the embedded secret unforgeability is defined as

$$AdvESU_{(\mathcal{E},\mathcal{S}im)} \coloneqq \Pr\left[\begin{array}{lcl} & & Params \xleftarrow{\$} Setup(), \\ & & (upk, usk) \xleftarrow{\$} KeyGen(), \\ V^*(\sigma) = \texttt{YES} & : & \mathcal{E}^{\mathcal{O}_{\mathcal{A}}}(Params, upk), \\ & & usk_1 \overset{UskQuery(b=1)}{\leftarrow} \mathcal{S}im(usk), \\ & & \sigma \leftarrow \mathcal{E}(usk_1) \end{array}\right].$$

The probability is taken over the coin tosses of the Setup algorithm, the key-generation algorithms, the oracle, and the algorithm $\mathcal{E}$.

A signature scheme is embedded secret unforgeable if for any polynomial-time algorithm $\mathcal{E}$ the value $AdvESU_{(\mathcal{E},\mathcal{S}im)}$ is negligible in the security parameter $k$. In other words, there exists no polynomial-time algorithm $\mathcal{E}$ with a significant advantage $AdvESU_{(\mathcal{E},\mathcal{S}im)}$.

## 4. The Scheme

In this section we propose a construction of an embedded secret signature scheme. First, however, we outline problems that arise in a simple and intuitive approach to building such a signature.

### 4.1. Naïve approach

The most straightforward way to embed a warning is to place it in a message that gets signed afterwards. To make it look innocuous this warning could be appended as some random bits to the base message, e.g., to pad it to some prescribed block length. We note that such a padding should vary when multiple messages are signed, i.e., different messages should not use the same padding. Another way is utilizing random components that come as parts of a signature scheme. For instance, we can put an alert message into a random exponent, call it $a$, of the El Gamal signature (see Section 6 where the El Gamal scheme is recalled). In this case the signer $\mathcal{A}$ and the trusted party $\mathcal{T}$ should first agree on a space of exponents that indicate a coercion. However, if the space is small, say, of constant size, the adversary can easily issue a correct signature that does not contain a warning after obtaining $\mathcal{A}$'s private key and picking a random $a$. He can safely hope that $a$ that he has chosen does not fall into the space of alerting exponents. Even if we swap roles of warning-indicator and warning-free exponents, so that $\mathcal{A}$ picks $a$ from a prechosen, shared, yet still small, space if he does *not* want to place an alarm in a signature, there is still a problem with such a scheme. First of all, the adversary can use some kind of repetition attack. When warning-free exponents do not depend on a message that gets signed, the adversary can simply reuse exponents from previously issued signatures that are supposed to contain no warning. This implies that $a$ should also be correlated with a message. For instance, we can let $a = H(M \oplus K)$, where $M$ is a message to sign, $K$ is a secret key known to $\mathcal{A}$ and $\mathcal{T}$, and $H$ denotes a hash function. A difficulty arising here is that $a$ itself may serve as a commitment. Namely, the adversary may want to determine if $K$ handed over to him is genuine by demanding $K$ and $a$ from the previously issued signatures. The key observation towards our solution is that the signer has to be able to "deny" all such commitments and present *any* fake $K$. This justifies turning our attention to deniable encryption schemes.

### 4.2. Construction

Our scheme is based on almost any randomized signature scheme

$$SS_{\text{inner}} = \big(Kg, \text{Sig}, \text{Ver}\big) \ .$$

The only additional requirement is that such a signature, more specifically: the signing function Sig, must utilize some significant part of randomness source in a non-trivial way (the randomness is given as $R$ argument passed to Sig).

$SS_{\text{inner}}$ is thus an inner scheme, and we build an embedded secret scheme $SS = (\text{KeyGen}, \Sigma, V, V^*)$ on the top of it.

Let $\ell$, $n$, $t$ and $s$ be parameters (the exact meaning of these should be clear from context they appear in) and $\mathcal{M} = \{0,1\}^\ell$ be (as in Section 2) a space of allowable messages. We shall assume that for all sufficiently large choices of $t$ there exists a family $\{S_t^{(i)}\}_i$ of translucent sets $S_t^{(i)} \subset \{0,1\}^t$. Let $H$ be a random oracle computing the hash function.

**Setup phase (**KeyGen**)**:
- $\mathcal{T}$ picks up a random string $K$ from $\{0,1\}^\ell$ and gives it to the signer $\mathcal{A}$;
- $\mathcal{T}$ chooses $S_t^{(i)}$ randomly and publishes the associated deniable encryption algorithm $E$; trapdoor information $d_t$, used in the decryption algorithm $D$, is kept secret;
- $\mathcal{A}$ fixes her/his signature function Sig using key generation algorithm $Kg()$ from signature scheme $SS_{\text{inner}}$. $\text{Sig}(M, R)$ forms a signature on message $M$ using randomness $R$;
- a verification algorithm Ver (returning either `YES` or `NO`) corresponding to Sig is published.

It should be noted that the key $K$ is shared between parties $\mathcal{T}$ and $\mathcal{A}$. Security of the system is assured as long as $K$ remains secret, so that nobody except $\mathcal{T}$ and $\mathcal{A}$, knows $K$.

**Signature ($\Sigma$)**:
In order to sign a message $M \in \mathcal{M}$ the party $\mathcal{A}$ computes

$$\begin{aligned} R &= E\big(H(M \oplus K), r\big) \\ S &= \text{Sig}\big(M, R\big) \end{aligned} \tag{2}$$

using a random seed $r$ fed into the deniable encryption function $E$.
The signature on $M$ is then

$$\sigma = (M, R, S). \tag{3}$$

**Verification ($V$ and $V^*$)**:
- given $\sigma' = (M', R', S')$ the standard verification algorithm $V$ checks if the following holds:
$$\text{Ver}(M', S') \stackrel{?}{=} \texttt{YES}$$
- given $\sigma' = (M', R', S')$ the opening algorithm $V^*$ applies $V$ and if $\sigma'$ is correct, i.e., when $V(\sigma') = \texttt{YES}$, then $V^*$ does the additional checking:
$$H(M' \oplus K) \stackrel{?}{=} D(R')$$
using the trapdoor $d_t$ to decrypt $R'$ using $D$.
The signature is claimed to be forged, i.e., $\mathcal{A}$ was forced to sign $M'$ by the adversary, iff the above condition does not hold.

## 5. Security

In this section we prove that our construction preserves inner signature scheme's unforgeability properties (security against, e.g., existential forgery under known message attack, total break under chosen-message attack, etc.; see Goldwasser et al., 1988), and that our new scheme is embedded secret secure.

The security of the system is based on the fact that $V^*$ is not known to the legitimate signer and therefore he would not be able to produce the irrefutable proof (for any third party) that a signature was produced with a real key.

Intuitively, the adversary should know nothing about $K$ (in particular, he must not know $H(M \oplus K)$ — a commitment for $K$ when some signed message $M$ is fixed). Then, the signer $\mathcal{A}$ may very well try to fool the adversary. The signer has to be consistent in his testimonies, nonetheless. In order to do this, $\mathcal{A}$ can choose a single $K'$ that $\mathcal{A}$ can claim to be the legitimate shared key. Now, if $\mathcal{A}$ wants to convince the adversary that some signature does not contain an embedded warning, $\mathcal{A}$ can open a given $R = E\big(H(M \oplus K), r\big)$ (for some random seed $r$ and a fixed message $M$) as $H(M \oplus K')$ claiming that $R$ is equal to $E\big(H(M \oplus K'), r'\big)$ (for some $r'$). The deniable encryption has to allow virtually all openings but the *parity scheme* based encryption algorithm discussed in Section 2 has this property. The adversary cannot check that the claim is invalid since he does not see $H(M \oplus K)$ and $R$ looks completely random. The signer shows him the following values: $M$, $K'$, and demonstrates how $E\big(H(M \oplus K'), r'\big)$ was produced using $r'$.

We formalize this idea below.

### 5.1. Inner signature's unforgeability preservation

From the definition of encryption $E$, $R$ is a concatenation of $t$-bit strings from sets $S$ and $R$. $R$ is a set of random $t$-bit strings and $S$-elements are indistinguishable from $R$-elements. Therefore, the distribution of $E$ taken over the set of messages is indistinguishable from the random distribution.

The quasi-random element $R = E\big(H(M \oplus K), r\big)$ is indistinguishable from a random element for any third party, therefore forging (with respect to the unforgeability property) the signature

$$\sigma = \Big(\mathrm{Sig}(M, R), E\big(H(M \oplus K), r\big)\Big), \qquad \text{where } R = E\big(H(M \oplus K), r\big)$$

is at least as hard as forging the inner signature $\mathrm{Sig}(M, R)$ with $R$ generated according to the protocol, i.e., randomly.

### 5.2. Embedded secret

We introduce the simulator $\mathcal{S}im$ that can be understood as an algorithm in the signer's mind. Given all signer's private data it generates dishonest answers in case of coercion (e.g. blackmail). If the signer $\mathcal{A}$ was forced by the adversary $\mathcal{E}$ to

reveal messages, private keys and random elements used in signing the messages, he would use the $\mathcal{S}im$ algorithm. $\mathcal{S}im$ generates the following output:

- honest private keys and random elements for the inner signature
- random $\tilde{K}$
- for each given signature $\sigma = \big(\mathrm{Sig}(M,R), E(H(M \oplus K), r)\big)$ return an element $\tilde{r}$ such that:

$$E\big(H(M \oplus K), r\big) = E\big(H(M \oplus \tilde{K}), \tilde{r}\big).$$

As argued by Canetti et al. (1996) it is feasible to generate such an element with probability $1 - \frac{1}{n}$.

Now we show that the scheme is embedded secret indistinguishable (3.2.1) and embedded secret unforgeable (3.2.2).

### 5.2.1. Embedded secret indistinguishability

The signer $\mathcal{A}$ passes the output generated by the $\mathcal{S}im$ to $\mathcal{E}$. Since $K$ and $\tilde{K}$ (of length $\ell$) were chosen randomly, they are indistinguishable for $\mathcal{E}$. Now, we have to show that honest encryptions are indistinguishable from dishonest ones. First, we introduce more convenient variables:

$$\begin{aligned} m_1 &= H(M \oplus K) \qquad \text{and} \\ m_2 &= H(M \oplus \tilde{K}), \end{aligned}$$

plus $b_i^{m_1}$ and $b_i^{m_2}$ (for $i = 1, \ldots, \ell$), which indicate consecutive bits of $m_1$ and $m_2$, respectively. We also write $r_{\mathcal{A}} = r_1^{\mathcal{A}} \| \ldots \| r_\ell^{\mathcal{A}}$ and $r'_{\mathcal{A}} = r_1'^{\mathcal{A}} \| \ldots \| r_\ell'^{\mathcal{A}}$ to denote bit representation of $r$ and $\tilde{r}$, respectively. Similarly, by $r_i^{\mathcal{B}}$ for $i = 1, \ldots, \ell$ we mean corresponding verifier's random bits.

From the deniability of encryption (see 2.2), for each $i = 1, \ldots, \ell$, i.e., for each bit of a "message" we have that

$$\big(b_i^{m_2}, r_i'^{\mathcal{A}}, \mathrm{COM}(b_i^{m_1}, r_i^{\mathcal{A}}, r_i^{\mathcal{B}})\big) \quad \text{and} \quad \big(b_i^{m_2}, r_i^{\mathcal{A}}, \mathrm{COM}(b_i^{m_2}, r_i^{\mathcal{A}}, r_i^{\mathcal{B}})\big)$$

are $4/n$-close.

Since $m_1$ and $m_2$ have the same distribution ($m_1$ and $m_2$ are generated by the hash function $H$), we have that

$$\big(b_i^{m_2}, r_i'^{\mathcal{A}}, \mathrm{COM}(b_i^{m_1}, r_i^{\mathcal{A}}, r_i^{\mathcal{B}})\big) \quad \text{and} \quad \big(b_i^{m_1}, r_i^{\mathcal{A}}, \mathrm{COM}(b_i^{m_1}, r_i^{\mathcal{A}}, r_i^{\mathcal{B}})\big)$$

are $4/n$-close.

According to the statement above, probability that a polynomial distinguisher will not distinguish an encrypted bit (which is a sequence of $n$ elements from $\{0,1\}^t$) of $m_1$ and $m_2$ is at least $1 - 4/n$. Since $m_1$ has length $\ell$, every bit

is encrypted independently, hence probability that it will not distinguish any of $\ell$ bits is at least $(1-4/n)^{\ell}$. Therefore we have that

$$\big(m_2, r'_{\mathcal{A}}, \mathrm{COM}(m_1, r_{\mathcal{A}}, r_{\mathcal{B}})\big) \quad \text{and} \quad \big(m_1, r_{\mathcal{A}}, \mathrm{COM}(m_1, r_{\mathcal{A}}, r_{\mathcal{B}})\big) \tag{4}$$

are $1-(1-4/n)^{\ell}$-close.

We would like to determine the security parameter $k_I = k_I(n, \ell)$, such that for every constant $\mathcal{C}$:

$$1 - \left(1 - \frac{4}{n}\right)^{\ell} < \frac{1}{k_I^{\mathcal{C}}} \ , \tag{5}$$

when $n, \ell \to \infty$.

We prove that the inequality (5) holds for any fixed $\epsilon, \delta > 0$ when we set $\ell \leq n^{1-\delta}$ and $k_I \leq \exp\big((\ln n)^{1-\epsilon}\big)$. Assume that $n > n_0(\epsilon, \delta, \mathcal{C})$ is sufficiently large and let $\ell \leq n^{1-\delta}$. By the binomial formula, we have:

$$1 - \left(1 - \frac{4}{n}\right)^{\ell} \leq 1 - \left(1 - \frac{4}{n}\right)^{n^{1-\delta}} < \frac{5}{n^{\delta}} < k_I^{-\mathcal{C}} \ ,$$

provided $k_I \leq \exp\big((\ln n)^{1-\epsilon}\big)$.

Hence, given (4), it is infeasible for the adversary to judge whether $m_2$ and $\tilde{r}$ were the honest message and a random value used by $\mathcal{A}$. Therefore, it is also infeasible for him to judge whether $\tilde{K}$ and $\tilde{r}$ were actually used by $\mathcal{A}$ in signing messages. It is infeasible with respect to the security parameters $k_I$, $s$ (describing the trapdoor permutation $f\colon \{0,1\}^s \to \{0,1\}^s$) and $t$, both used in the construction of translucent set (as given by Canetti et al., 1996). As a result, for any polynomial-time algorithm $\mathcal{E}$ the corresponding value of $AdvESI_{(\mathcal{E},\mathcal{S}im)}$ is negligible in $k_{ESS} = \min\{k_I, t, s\}$. Note that the length of the signature grows faster than the security parameter.

### 5.2.2. Embedded secret unforgeability

An adversary knowing only $\tilde{K}$ is not able to compute $H(M \oplus K)$ for any fresh message $M$, since $K$ and $\tilde{K}$ were chosen at random, and $H$ is a random oracle. Therefore, generating a valid encryption of the unknown text $H(M \oplus K)$ is infeasible for the adversary.

We summarize the results above with the following theorem:

THEOREM 5.1 *Under the assumption that for all sufficiently large choices of $t$ there exists a family $\{S_t^{(i)}\}_i$ of translucent sets $S_t^{(i)} \subset \{0,1\}^t$, our scheme, based on any signature scheme $SS_{inner}$, is embedded secret secure and preserves security properties of the inner scheme $SS_{inner}$.*

## 6. Modified example

The noticeable disadvantage of the signature presented in Section 4 is that (3) contains an explicit reference to random bits $R$ generated by a deniable encryption algorithm. The purpose of such a binding is to ensure that the trusted party $\mathcal{T}$ is able to apply $V^*$ and extract an embedded secret message. However, for Bob, i.e., any person that uses the standard verification algorithm $V$, placing $R$ in $\sigma$ can be superfluous. In some cases we have a possibility to get rid of the passing parameters that could be unnecessary at the first sight (of course we do not want to eliminate $R$ completely — it will still be sealed in $S = \mathrm{Sig}\big(M, R\big)$).

The construction is built upon El Gamal signature (El Gamal, 1985). In his classical work Simmons (1985) demonstrated that there exists a broad subliminal channel that can be hidden in a signature. We briefly present his idea here. Recall that El Gamal signature operates on a cyclic group $\mathbb{Z}_p^*$, where $p$ is some large prime (having several additional properties that make signatures cryptographically secure, in particularly $p-1$ has a large prime factor). Let $g$ be a generator of $\mathbb{Z}_p^*$. By $x \in \mathbb{Z}_{p-1}^*$ we denote the signer's private key with $y = g^x \in \mathbb{Z}_p^*$ being the associated public key. We assume that $p$, $g$ and $y$ are publicly known. To sign $M \in \mathcal{M}$ the party $\mathcal{A}$ simply picks $a \in \mathbb{Z}_{p-1}^*$ at random and computes

$$\alpha \coloneqq g^a \bmod p$$
$$\beta \coloneqq \big(h(M) - x\alpha\big)a^{-1} \bmod p-1,$$

where $h\colon \mathcal{M} \to \mathbb{Z}_{p-1}^*$ is a collision-resistant hash function. A signature is a tuple $(\alpha, \beta)$ and its validity can be verified by any party by checking if the following holds

$$y^\alpha \alpha^\beta \stackrel{?}{=} g^{h(M)}$$

in $\mathbb{Z}_p^*$.

To embed a secret message $m \in \mathbb{Z}_{p-1}^*$, the party $\mathcal{A}$ can alter this scheme and use the random part of $(\alpha, \beta)$ as a container for $m$, so to speak. Namely, instead of picking a random $a$, $\mathcal{A}$ can set $\alpha \coloneqq g^m \bmod p$. The verification phase of the innocuous message $M$ remains the same. But anyone who possesses the key $x$ can recover $m = \big(h(M) - x\alpha\big)\beta^{-1} \bmod p-1$ sent subliminally (provided $\beta$ is invertible $\bmod p-1$). Here, the capability of reading from the subliminal channel comes as a trade-off for sharing signer's "private" data $x$. Another assertion that should be made here is that some significant part of $m$ should consist of random bits to make guessing $m$ virtually impossible. Then to extract $m$ it is essential to know $x$, since computing a discrete logarithm in $\mathbb{Z}_p^*$ is believed, in general case, to be hard.

In our embedded secret signature scheme we can adopt Simmon's construction in a straightforward way. Let Sig be El Gamal signature (in some group $\mathbb{Z}_p^*$) with $x$ being the "private" key of $\mathcal{A}$. In fact, in the setup phase $\mathcal{A}$ should transfer $x$ in secure way to $\mathcal{T}$. Signing means computing $S = \mathrm{Sig}(M, R) = \big(\alpha, (h(M) - x\alpha)R^{-1}\big)$, where $\alpha = g^R \bmod p$ and $R$ is a ciphertext of a deniably

encrypted secret message indicating whether signature has been forced or not. Thus, both $R$ and $S$ are given in the same way as in (2). However, in opposition to (3), now the signature is only $\sigma = (M, S)$, i.e. it does not contain an explicit reference to $R$ (in fact, an acquaintance of $R$ implies extracting $x$ from signatures, so in this particular case placing $R$ in $\sigma$ would compromise the whole scheme). Also, note that $E$ proposed before produces a stream of bits that are indistinguishable from random bits. Using $E$ as a blinding function is required in order to make an attacker unable to read subliminal information even when $\mathcal{A}$ is forced to unveil $x$ to the adversary.

## 7. Possible applications

The presented cryptographic primitive has a natural appearance in the signature schemes with the trusted party involved in the verification process. A typical example concerns the situation when the trusted party legitimates a voluntary signature, or is able to discover the embedded secret in a signature. In some applications the corresponding trusted authority should be involved in the preparation of a suitable "proof of coercion".

Let us consider as an example the e-delegation of signing rights (the corresponding proxy signature primitive has been defined in the work by Mambo et al., 1996). Assume that the original signer, who delegates his signing ability to the proxy signer, is equipped with a suitable verification algorithm $V^*$. Then, in the case when the proxy signer is forced to sign a given message, or simply to expose his private key to the adversary, the corresponding signature would be discovered by the designator as a coerced one. Obviously, the strong unforgeability condition (see, e.g., the work by Boldyreva et al., 2003) should imply that the designator is not able to generate voluntary signatures on behalf of the proxy signer.

Another application towards the group signature schemes, introduced by Chaum and van Heyst (2003), may regard the manager as being equipped with the suitable verification algorithm $V^*$, since he is the party that can recognize the identity of signer being coerced. Here the full traceability condition (see, e.g., Bellare and Miner, 1999) should imply that the manager is not able to compute the voluntary signature on behalf of a group member.

A similar functionality could be also adopted in a more involved context of fair exchange protocols. The hybrid solution (Pomykała and Trabszys, 2009) joining the idea of anonymous-signer signatures (Yao and Tamassia, 2006) and verifiably encrypted signature (Boneh and Gentry, 2003) is another useful appearance of the signature scheme with the trusted party involved in the verification process $V^*$.

The concept of the subliminally embedded warning can be also adopted to the certificateless systems (Pomykała, 2009). Unlike the typical Id-based digital signature scheme, this approach avoids regarding the Private Key Generator (PKG) as the trusted party $\mathcal{T}$. Instead, PKG generates a verification key related

to the secret key created by the signer. However, the shared knowledge between $\mathcal{T}$ and the signer allows $\mathcal{T}$ to detect an embedded secret in a corresponding signature (see the work by Hołyst and Pomykała, 2010).

Let us also quote another approach to the construction of secretly embedding warning in digital signature, based on the Diffie-Hellman key agreement protocol as proposed by Kubiak and Kutyłowski (2013). Their scheme is simpler and more efficient, but does not apply to some well known signature schemes. In particular, our approach is easily adopted to universal padding scheme for RSA (Coron et al., 2002) and Feige–Fiat–Shamir (1988) signature scheme. In the first case the corresponding scheme is safe even if the same pair private/public keys are used for signing and encrypting. In the second one the additional (subliminal) information is created on the basis of the two additional primes dividing the modulus, which are known only by the trusted party $\mathcal{T}$ (see Pomykała, 2009; Hołyst and Pomykała, 2010). Summing up, the deniable encrypting approach might be useful even in the case of the partial leakage of signer's private key.

## Acknowledgement

The authors are grateful to Dr. Rene Peralta for interesting personal discussions and e-mail correspondence on the subject. Moreover we are indebted to the reviewers for their suggestions that allowed for improvement of the final version of the paper.

## References

Bellare, M. and Rogaway, P. (1993) Random oracles are practical: a paradigm for designing efficient protocols. In: *Proceedings of the 1st ACM Conference on Computer and Communications Security.* ACM, New York, USA, 62-73.

Bellare, M. and Miner, S. (1999) A forward-secure digital signature scheme. *Advances in Cryptology – CRYPTO '99.* **1666** (1999) Springer-Verlag, 431–448.

Bellare, M., Micciancio, D. and Warinschi, B. (2003) Foundations of Group Signatures: Formal Definitions, Simplified Requirements, and a Construction Based on General Assumptions. *Advances in Cryptology – Eurocrypt '03.* **2656** (2003) Springer-Verlag, available at http://www-cse.ucsd.edu/ mihir/papers/gs.pdf.

Boldyreva, A., Palacio, A. and Warinschi, B. (2013) Secure Proxy Signature Schemes for Delegation of Signing Rights. *Journal of Cryptology*, **25**, 1, 57-115.

Boneh, D., Gentry, C. (2003) Aggregate and Verifiability Encrypted Signatures from Bilinear Maps. In: Advances in Cryptology – Eurocrypt '03, **2656**, Springer-Verlag, 416–432.

Canetti, R., Dwork, C., Naor, M. and Ostrovsky, R. (1996) Deniable Encryption. In: *Lecture Notes in Computer Science*, **1294**, 90–104.

Chaum, D. and van Heyst, E. (2003) Group Signatures. In: *Advances in Cryptology – Eurocrypt '91.* Springer-Verlag, 257–265.

Dreyfus, S. The Idiot Savants' Guide to Rubberhose. Available at http://iq.org/ proff/ rubberhose.org/current/src/doc/maruguide.pdf

Coron, J. S., Joye, M., Paillier, P. and Naccache, D. (2002) Universal Padding Schemes for RSA. *Proc. Crypto'02.* **2442**, LNCS, 226-241.

Dürmuth, M., Freeman, D. M. (2011) Deniable encryption with negligible detection probability: an interactive construction. *Proceedings of the 30th Annual international conference on Theory and applications of cryptographic techniques: advances in cryptology, EUROCRYPT'11.* Springer, Tallin, Estonia, 610–626.

El Gamal, T. (1985) A Public Key Cryptosystem and a Signature Scheme based on Discrete Logarithms. *IEEE Transactions on Information Theory* **31**, 4, 469–472.

Feige, U., Fiat, A., Shamir, A. (1988) Zero Knowledge Proof of Identity. *Journal of Cryptology* **1**, 77-94.

Geambasu, R., Kohno, T., Levy, A. and Levy, H. M. (2009) Vanish: Increasing Data Privacy with Self-Destructing Data. *Proceedings of the USENIX Security Symposium.* Available at http://vanish.cs.washington.edu/pubs/usenixsec09-geambasu.pdf

Goldwasser, S., Micali, S., Rivest, R. (1988) A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. *SIAM Journal on Computing* **17**, 2, 281–308.

Håstad, J., Jonnson, J., Juels, A. and Yung, M. (2000) Funkspiel schemes: an alternative to conventional tamper resistance. *CCS '00*, *Proc. of the* $7^{th}$ *ACM Conference on Computer and Communications Security.* ACM, New York, 125–133.

Hołyst, B. and Pomykała, J. (2010) *Electronic Signature and Biometric Methods of Identification* (in Polish). WSM publications, Warsaw.

Itkis, G. (2003) Cryptographic tamper evidence. *CCS '03*, 355–364.

Kubiak, P. and Kutyłowski, M. (2013) Lightweight Digital Signature with Secretly Embedded Warning. *Control and Cybernetics* **42**, 4, 825–827.

Klonowski, M., Kubiak, P. and Kutyłowski, M. (2008) Practical Deniable Encryption. *SOFSEM 2008: Proc. of the* $34^{th}$ *Conference on Current trends in Theory and Practice of Computer Science.* Springer, Berlin–Heidelberg, 599–609.

Mambo, M., Usuda, K. and Okamoto, E. (1996) Proxy Signatures for Delegating Signing Operation. *3rd ACM Conference on Computer and Communications Security.* ACM, 48–57.

Naccache, D., Pointcheval, D. and Tymen, C. (2002) Monotone signatures. *Financial Cryptography*, LNCS **2339**, 305–318.

Pfitzman, B. and Waidner, M. (1991) Fail-stop signatures and their application. *SECURICOM 91:* $9^{th}$ *Worldwide Congress on Computer and Communications Security and Protection.* SEDEP/Blenheim, Paris, 145–160.

Pomykała, J. and Trabszys, T. (2009) Anonymous signer verifiable encrypted signature from bilinear pairing. *Control and Cybernetics* **38** (3), 705–712.

Pomykała, J. (2009) Id-based Digital Signatures with Security Enhanced Approach. *Journal of Telecommunications and Information Technology* **4**, 146–153.

Simmons, G. (1985) The Subliminal Channel and Digital Signatures. *Advances in Cryptology – Eurocrypt '84 Proceedings.* Springer, 364–378.

Yao, D. and Tamassia , R. (2006) Cascaded Authorization with Anonymous-Signer Aggregate Signatures. *proc. of the 2006 IEEE Information Assurance Workshop*, IEEE, 84–91.