# Subliminal Channels in Cryptographic Systems

Christian Bäcker

23.07.2009

Seminar Topic: Covert Channels and Embedded Forensics
Advisor: Timo Kasper

Seminararbeit
Ruhr-Universität Bochum

Chair for Embedded Security
Prof. Dr.-Ing. Christof Paar

# Contents

# 1 Introduction

Covert channels transport information by using a system parameter that was not intentionally made for communication. System parameters can be, e.g., jitter, crosstalk, modulation, key material, padding, nonce or random values. Subliminal channels are a subgroup of covert channels. They are hard to detect and can be used to communicate secretly over an insecure channel with help of digital signatures.

In [Sim84] Gustavus J. Simmons presented how the "Prisoner Problem" can be solved with help of digital signature crypto systems. The prisoner problem describes the problem how two prisoners can communicate confidentially and undetected over an insecure channel, the warden. Remaining undetected is the main problem here because the warden must read all communication.

In short, Simmons sorted the secret private key (in this paper referred to as authentication key) into two different sets, where keys out of one set mean "1" and the others mean "0". The prisoners discussed the set partitioning before, what the warden does not know. The warden knows all keys and can check the message signatures successfully. For the warden everything is alright, but the prisoners can communicate with their choice of the keys.

Later Simmons describes in [Sim85] how the problem can be solved through parameter substitution of signature algorithms. In signature algorithms like ElGamal/DSA exist parameters which have to be set with random. He shows how one can set these parameter with a message and how the receiver can recover the message out of the signature with knowledge of the secret key. Additionally the secret key can be used to hold the message, then the random parameter has a predefined value and the secret key is computable. Of course, when using ElGamal in this way the algorithm does not fulfill the security requirements of signature algorithms any longer. The receiver can impersonate the sender because both know the secret key. Therefore, signature algorithms that are secure even if used for subliminal channels were developed.

Without the knowledge of the secret, a subliminal channel is hard to detect because the signature still is verifiable but the hidden message cannot be reconstructed. At trying to cheat the algorithm with a randomly chosen signature value an infeasible small difference can be detected, between the success probability when the algorithm is used normally and when it is used as a subliminal message transporter. Apart from that a subliminal channel can be made as hard to discover as the crypto system is hard to break [Sim84]. Algorithms can be searched for possibilities of subliminal channels but it is arbitrarily hard to say wether the channel was used. This problem led to the development of authentication systems without the possibility of subliminal channels, called “subliminal channel-free” crypto system.

Simmons divides subliminal channels into broadband and narrow-band channels. A broadband channel uses $\alpha - \beta$ bits, where $\alpha$ are the signature bits and $\beta$ the bits needed for protection against signature forgery, alteration or transplantation. Every channel which uses less bits is called a narrow-band channel. The additional used bits are needed for further protection. The broadband and the narrow-band channels can use different algorithm parameters. A narrow-band channel cannot transport maximal information, but it preserves the authentication key. Because of improvements, Simmons authentication key sharing, for his broadband channel, can be avoided.

### Definition of Covert Channels

Yvo Desmedt generalizes subliminal channels in [Des90] and calls them “abuses”. He found abuses in the Feige-Fiat-Shamir identification system [FFS87]. Further he argues that these abuses, including subliminal channels, are no covert channels because the signature is intended to transfer information, namely integrity, authentication and non-repudiation.Desmedt cites B.W. Lampson, who defined in [Lam73, p. 614] that covert channels are not intended for information transfer at all. Such covert channels transfer information unintendedly, e.g., time jitter, crosstalk and amplitude modulation. However, Desmedt admits that this all is debatable.

According to the title of this seminar “Covert Channels and Embedded Forensics” this seminar paper treats a subliminal channel as a covert channel.

### Steganography vs. Subliminal Channels [Sch95]

Steganography hides a secret message in another message, such that the existence of the secret message is concealed. This can be done in various ways. Historically this has been done with invisible ink, tiny pin punctures or pencil marks on typewritten characters etc.. Today it is popular to hide a message in computer graphics, where the least significant bit of each byte is replaced. Modern graphic standards specify more gradations of color than the human eye can differ. By changing the least significant bit the graphic do not change noticeable and it is improbable that the change is noticed by the human eye. In this way a 64 kilobyte message can be stored in a 1024 × 1024 grey-scale picture.

The subliminal channel uses an algorithm or protocol to hide messages in a normal looking communication. Because the algorithm’s signature creation procedure is unchanged, the subliminal signature is indistinguishable from a normal signature. Without the knowledge of the secret the subliminal message cannot be read nor detected.

To summarize, a steganographic message is hidden in an object while a subliminal message is embedded in an algorithm.

### Applications

Obvious applications of subliminal channels are spy networks or circumventing a censorship. Because of the undetectability, no one will be noticed sending subliminal messages if everyone is signing their messages and documents.

A more subtle application is to “mark” digital cash or to track a document throughout its lifespan.Also possible are hidden information in personal ID’s or licenses about the health, creditworthy, political reliability or arrest record of the owner.

# 2 Two Examples

The following examples illustrate how a subliminal channel is embedded in an algorithm. The first is the classical example from Simmons “The Prisoner Problem and the Subliminal Channel” [Sim84]. The second is explained in more detail in Simmons “Subliminal Communication is Easy Using the DSA” [Sim94].

## 2.1 Prisoner Problem

This example is constructed only to show the idea behind a subliminal channel and has no claims to fulfill cryptographic security requirements. The prisoners agree on the following scheme before they are arrested, such that the warden has no knowledge about this agreement. The setting is that two prisoners want to communicate and the only way is the warden. The prisoners cannot use encryption because the warden wants to read all communication or he will not deliver the message. They decide to sign the outcome of a fair-coin toss and use the signature of the authentication system to hide their communication. The warden accepts the use of an authentication system, but insists on an encoding rule related to the outcome, such that he can verify the signature. For simplicity a three bit message is used and the warden only accepts an even parity message when head is the outcome of the coin toss and at tail an odd parity message. The eight ($2^3 = 8$) possible messages are now sorted into two unique sets. Both sets contain two even and two odd parity messages, e.g., $\phi = \{000, 101, 010, 111\}$ and $\psi = \{110, 011, 100, 001\}$. With these sets the following eight keys can be constructed (details follows).

| | Head | | Tail | |
|---|---|---|---|---|
| Key | $\phi$ | $\psi$ | $\phi$ | $\psi$ |
| 1. | 000 | 011 | 111 | 100 |
| 2. | 000 | 011 | 010 | 001 |
| 3. | 000 | 110 | 010 | 100 |
| 4. | 000 | 110 | 111 | 001 |
| 5. | 101 | 110 | 111 | 100 |
| 6. | 101 | 110 | 010 | 001 |
| 7. | 101 | 011 | 010 | 100 |
| 8. | 101 | 011 | 111 | 001 |

From the keys above the prisoners can choose one and tell the warden their choice, such that he can verify the signature. This is analog to digital signatures, where the prisoners would give the warden the verification keys.

If the set $\phi$ is associated with "1" and $\psi$ with "0" the prisoners can communicate a 1-bit subliminal message independently from the outcome of the toss and the warden cannot detect this message. E.g., they choose key 8, then the message "101" is send for the subliminal message "1" and a outcome of head.

In this general scheme the warden (opponent) has two options to deceive the prisoners. He can deliver a message of his own choice (impersonation) or he can substitute (substitution) the message.

## 2.2 The Subliminal Channel in DSA

In 1991 NIST proposed the digital signature algorithm (DSA) [NIS94] derived from the El Gamal scheme [EG85]. Analogous to the El Gamal scheme DSA contains a broadband subliminal channel. For further explanations the knowledge of the DSA is necessary. We repeat the algorithm from Appendix B of [Sim94]. The algorithm involves four distinct steps.

1. Set-Up, the in this step generated $p$, $q$ and $g$ are made public.
    - a large prime (512–1024 bits), $p$, is randomly selected, subject to the condition that $p-1$ is divisible by a 160-bit prime, $q$.
    - an element, $g$, of order $q$ in $GF(p)$ is constructed by choosing any element $z \in GF^*(p)$ for which $g \equiv z^{(p-1)/q} \bmod p > 1$.
2. Key Generation, the verification key, $y$, is associated with the signer identity in a public, certified, directory.
    - user chooses a random $x \in GF^*(p)$, which is the authentication key.
    - user publishes $y \equiv g^x \bmod p$ as the verification key.
3. Signing, $m$ is the message to sign, $H(\cdot)$ a cryptographic hash function and the triple $(m; r, s)$ is the signed message.
    - calculate the message digest $h = H(m)$.
    - choose a random $k \in GF^*(q)$. ($k$ is essentially a session key.)
    - calculate $r \equiv (g^k \bmod p) \bmod q$.
    - calculate $s \equiv k^{-1}(h + xr) \bmod q$, where $kk^{-1} \equiv 1 \bmod q$.
4. Verifying, the receiver receives a triple $(\underline{m}; \underline{r}, \underline{s})$ and assumes that it comes by the user whose public key in the directory is $y$.
    - receiver first calculates the message digest $\underline{h} = H(\underline{m})$.
    - receiver calculates $t \equiv \underline{s}^{-1} \bmod q$.

- receiver calculates $u_1 \equiv \underline{h}t \bmod q$.
- receiver calculates $u_2 \equiv \underline{r}t \bmod q$.
- receiver calculates $v \equiv (g^{u_1} y^{u_2} \bmod p) \bmod q$.
- The triple $(\underline{m}; \underline{r}, \underline{s})$ is accepted as having been signed by the user, associated with the public key $y$, if and only if $\underline{r} = v$.

### The Broadband Channel

The easiest way to establish a subliminal channel between sender and receiver using DSA is to share the authentication key $x$. One can then use $k = m'$ as the subliminal message. Hence, it is impossible to recover the subliminal message $m'$ without knowing $x$ or to detect that the subliminal channel is being used, even if the adversary knows the subliminal message.

The sender substitutes $k$ with $m'$ and calculates as before $r \equiv (g^{m'} \bmod p) \bmod q$ and $s \equiv m'^{-1}(h + xr) \bmod q$. The inverse of $m'$ exists because the modulus is prime and any non-zero element of $GF^*(q)$ has a multiplicative inverse. With knowledge of $x$ the receiver is now able to compute $m'$ by

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

It is important to note that $h + xr$ can be 0, so that $s = 0$. In this case it is impossible to recover $m'$. Simmons shows that the probability of $s$ being 0 is $1/q$, i.e., $\approx 2^{-160}$ and that the expected number of subliminal messages that cannot be communicated is only $|e/(1 - e)| \approx 1.58$. Although not all messages can be communicated subliminally the channel capacity is given by $l - \epsilon$, where $\epsilon < 10^{-47}$ and $l = \lceil \log_2(q) \rceil$ [Sim94].

### Three Narrowband Channel

The narrowband channel of DSA eliminates the necessity to share the authentication key. Simmons describes three narrowband channels in [Sim93].

For the first narrowband channel sender and receiver have to create a shared secret prime $P$ with $P > q$. The public prime $p$ should not be used, because anybody can check the following explanations with $p$ and therefore would be able to disclose the usage of the subliminal channel. One bit can be transmitted subliminally if the parameter $r$ is or is not a quadratic residue modulo $P$. The receiver can easily check that with Euler's criterion

$$r^{(P-1)/2} \equiv 1 \bmod P.$$

The transmitter can chose $k$ until $r = (g^k \bmod p) \bmod q$ is a quadratic (non-)residue. Since there exists $(P-1)/2$ quadratic residues and $(P-1)/2$ quadratic non-residues the chance to get the wished result is 50 percent. Because the values for the parameter $r$ cannot be distinguished by looking at the signature, this is a 1-bit subliminal channel which does not need the sharing of the authentication key.

The second narrowband channel is also a 1-bit channel. To use this channel, the transmitter and the receiver share a random binary sequence $B = b_1, \dots, b_t$ which can be used as a one-time key. First both agree on a bit position of $r$ which is used to transport the subliminal message, e.g., the lowest or the $w-$th lowest bit. Then the $i-$th subliminal message bit in the $i-$th signature is XORed with the $i-$th random bit $b_i$. This method is much easier then the first one, because the complete signature can be used to hide the subliminal message. E.g., the message itself could be chosen as subliminal message channel.

The last narrowband channel is an $l$-bit channel. The sender chooses a random $k' \in GF^*(q)$ and computes $r' \equiv g^{k'} \bmod p$. To set up the subliminal channel the sender has to transmit $r'$ secretly to the receiver. Before starting the digital signature algorithm the sender computes $k^* \equiv m' + k' \bmod q$, where $m'$ is the subliminal message of lenght $l$. Instead of the random value $k$ the constructed $k^*$ is used in the DSA algorithm to calculate $r$. The sender transmits as before the triple $(m; r, s)$. The receiver knows $r'$ and has to do a total search, testing all possible values for $m'$, to find $m'$ until

$$(g^{m'+k'} \bmod p) \bmod q \equiv \underline{r} = r \equiv (r' g^{m'} \bmod p) \bmod q,$$

where $\underline{r}$ is from the received triple $(\underline{m}; \underline{r}, \underline{s})$. This is feasible for small messages $m'$ of length $l$. Since $k' \in GF^*(q)$ and $k^* = m' + k' \bmod q$ it follows that $l$ is bounded to a maximum of 159 bit and therefore a narrowband channel. Considering that today a 148-bit subgroup discrete logarithm key size are assumed to be secure to a brute force attack, because the computation/testing of all keys needs to much time and computational power [Len04]. It can be assumed that a realistic useable maximal message length is less than 140 bits. A weakness of this channel is the constant value $r$ when sending the same subliminal message $m'$. This can be avoided by adding an increasing value to $k^*$ with the drawback that now the order of the messages is important [HMP94].

One should note that the construction of the last narrowband channel is very similar to the broadband channel. The narrowband channel has the benefit that it does not need the revealing of the authentication key. It trades this advantage for computing power at the receiver side.

### 2.2.1 Calculation Example of the Broadband Channel

Here we will demonstrate on an example how the broadband channel in DSA works. For simplicity we will use a little bit smaller values instead of 512-bit numbers.

1. Set-Up:

- We chose the primes $p = 2347$, $q = 23$ and the element $z = 1979$.
- It follows that the generator $g \equiv z^{(p-1)/q} \bmod p = 1979^{102} \bmod 2347 = 266$
- Publishing $(p, q, g)$.

2. Key Generation:
   - We chose the authentication key $x = 1468$, so that the verification key $y \equiv g^x \bmod p = 266^{1468} \bmod 2347 = 2100$.
   - Publishing $y$.
3. Signing:
   - The message $m$ is 1337 and as hash function we chose for simplicity a modulo reduction by 107.
   - The message hash is then calculated as $h \equiv 1337 \bmod 107 = 53$.
   - We chose randomly the session key $k = 17$ and since $kk^{-1} = 1 \bmod q$ it follows that $k^{-1} = 19$.
   - $r \equiv (g^k \bmod p) \bmod q = (266^{17} \bmod 2347) \bmod 23 = 12$
   - $s \equiv k^{-1}(h + xr) \bmod q = 19(53 + 1468 \cdot 12) \bmod 23 = 3$
   - Sending the triple $(m = 1337; r = 12, s = 3)$.
4. Verifying:
   - Receiving the triple $(m = 1337; r = 12, s = 3)$.
   - $h \equiv 1337 \bmod 107 = 53$
   - $t \equiv s^{-1} \bmod q = 8$
   - $u_1 \equiv ht \bmod q = 53 \cdot 8 \bmod 23 = 10$
   - $u_2 \equiv rt \bmod q = 12 \cdot 8 \bmod 23 = 4$
   - $v = (g^{u_1} y^{u_2} \bmod p) \bmod q = (266^{10} \cdot 2100^4 \bmod 2347) \bmod 23 = 12$
   - Since $v = r$ it is accepted that the message was signed by the user, associated with the public key $y$.

This example shows that the public information $p$, $q$, $g$, $y$, $m$, $r$, and $s$ are exchanged. The verification of the signature succeeded. Therefore, it is hard for an attacker to distinguish if a subliminal message is embedded.

Assume that the session key $k$ was not chosen randomly, but with purpose 17 by the transmitter and that the receiver also knows the authentication key $x$, which was transmitted secretly before. Hence, the receiver can calculate $k \equiv s^{-1}(h + xr) \bmod q$

$$k \equiv 8 \cdot (53 + 1468 \cdot 12) \bmod 23 = 17.$$

Since the attacker does not know the authentication key $x$ it is not possible for him to calculate $k$.

## 2.2.2 Calculation Example of the Narrowband Channel

In this section we will exemplary show how the narrowband channel works. For this we will use the same parameter values as in the broadband channel example.

### The First Narrowband Channel

For the first narrowband channel assume the transmitter and the receiver has securely exchanged the prime $P = 2237$. For the broadband channel example value $r = 12$ we calculate $r^{(P-1)/2} \bmod P = 12^{1118} \bmod 2237 = -1$. This is a quadratic non-residue and if we would associate quadratic residues with the value 1 we would have transmitted a 0 in our broadband example.

For $k = 15$ we get $r = (g^k \bmod p) \bmod q = (266^{15} \bmod 2347) \bmod 23 = 11$. If we check $r^{(P-1)/2} \bmod P = 11^{1118} \bmod 2237 = 1$ and therefore this is a quadratic residue.

One can see that the usage of this channel is very easy.

### The Second Narrowband Channel

For this channel we need a binary sequence $B = b_1, b_2, \ldots, b_t = 100101110_2$ which is exchanged between the transmitter and the receiver. Further we want to use the least signifiant bit of the message $m$ as our subliminal channel. In our representation the least signifiant bit is the rightmost bit, e.g., in $B$ the last 0.

Assume we want to communicate the subliminal message $m' = 0$ in our first message. Since $b_1 = 1$ we must chose a message $m$ which ends in binary on 1, see Table 2.1 for coding rules.

| | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 0 |

Table 2.1: XOR Table

If we want to communicate again a subliminal message $m' = 0$ in our second message we must chose a message $m$ which ends in binary on 0, because $b_2 = 0$. For a subliminal message $m' = 1$ we can reuse the first message $m$ which ends on 1, because $b_3 = 0$. This demonstrates that the meaning of the transmitted message $m$ is not relevant when used together with the subliminal channel technique.

### The Third Narrowband Channel

For the last narrowband channel at first the transmitter sends the receiver $r' = g^{k'} \bmod p$. Where $k'$ is chosen randomly by the transmitter. For this example the transmitter chooses $k' = 10$, hence $r' = 266^{10} \bmod 2347 = 515$ and as subliminal message $m' = 7$, so that $k^* = 7 + 10 \bmod 23 = 17$.

For simplicity and to not repeat the whole DSA calculations again and again $k^*$ was constructed to be 17 like in the broadband channel example. For calculations please see subsection 2.2.1.

The receiver get the triple $(\underline{m}; \underline{r}, \underline{s}) = (1337; 12, 3)$ and starts to search for the correct $m'$, after verify that the signature is valid, see step 4 of subsection 2.2.1.

The calculations for the total search are shown in Table 2.2.

| $m'$ | $r = (r'g^{m'} \bmod p) \bmod q$ |
|---|---|
| 1 | $(515 \cdot 266^1 \bmod 2347) \bmod 23 = 13$ |
| 2 | $(515 \cdot 266^2 \bmod 2347) \bmod 23 = 3$ |
| 3 | $(515 \cdot 266^3 \bmod 2347) \bmod 23 = 1$ |
| 4 | $(515 \cdot 266^4 \bmod 2347) \bmod 23 = 6$ |
| 5 | $(515 \cdot 266^5 \bmod 2347) \bmod 23 = 11$ |
| 6 | $(515 \cdot 266^6 \bmod 2347) \bmod 23 = 17$ |
| 7 | $(515 \cdot 266^7 \bmod 2347) \bmod 23 = 12$ |

Table 2.2: Total Search for the Subliminal Message $m'$.

Since we get $r = \underline{r}$ for $m' = 7$, we have successfully extracted the subliminal message $m' = 7$. The last narrowband channel is an $l$-bit channel, where $l$ is the length of $m$. As sketched in this example, for big $l$ the total search needs more time.

# 3 Improvements

Besides the development of countermeasures against subliminal channels, as detailed in section 4, a development on improving the channel has happened. One goal was to get the advantage of the narrowband channel in a broadband subliminal channel. The broadband channel should not need to make it necessary to reveal nor make it possible to compute the authentication key.

### A Modification of the Brickell and DeLaurentis Signature Scheme

Simmons [Sim86] himself presents an improvement to keep the authentication key secret, such that the receiver is not able to impersonate the sender. He modifies the Brickell and DeLaurentis signature scheme [BD86]. The scheme rests on the difficulty of extracting approximate $k^{th}$ roots in $Z_n$, where $k \geq 4$ (see [SPMlS02]). In the following we show the modified version of the scheme and point out the differences.

1. Key Generation (Set-Up)
   - The sender chooses three primes $p > q > r$ sufficient large that $p^2q$ is computational infeasible to factor. $p$ and $q$ are kept secret while $r$ is secretly given to the receiver.
   - The sender publishes $n = p^2qr$ as his authentication key.
   - A one-way hashing function $(H(\cdot))$ on messages is known by the sender and receiver and an exponent $k \geq 4$.

2. Signature Generation, $m \in Z_n$ is the signed message and $m' \in Z_r$ the subliminal message
   - Instead of a randomly chosen $x \in_R Z^*_{pqr}$ the sender chooses a $u \in_R Z^*_{pq}$ and calculates $x' = m' + ur$.
   - Then the sender first calculates the one-way hashing function $h(m)$, and then calculates the signature $s$ of $m$ as follows.

     a) $w' = \left\lceil \frac{h(m) - x'^k \bmod n}{pqr} \right\rceil$

     b) $y' = \frac{w'}{kx'^{k-1}} \bmod p$

     c) $s' = x' + y'pqr$

- The pair $(m; s')$ is transmitted as the "signed" message.

3. Decoding the Subliminal Message

   - Since the receiver knows $r$ and got the pair $(m; s)$ he can calculate

   $$s = x' + y'pqr = m' + ur + y'pqr \equiv m' \bmod r.$$

4. Verification of Signature (only for completeness)

   a) Receiver gets $(m; s)$.

   b) The receiver calculates the hashing function $h(m)$.

   c) The message is accepted as authentic if and only if

   $$h(m) \leq s^k \bmod n < h(m) + \delta$$

   where $\delta$ is a bound to $O(n^{2/3})$. A proof that this statement holds is given in [Sim86, Appendix].

The authentication key is not revealed to the receiver and he is not able to compute the key because he still must factor $p^2q$ to recover $pqr$. An additional advantage of this scheme is that signing the same message repeatedly results in a random appearing set of signatures, instead of resulting in the same signature, because of the random chosen $u$.

### Message Converting and Self-Synchronization

In some algorithms it is possible to differentiate if random is used or not. A method presented in [KI97] shows how a message can be converted by a special converter, such that it looks like indistinguishable random. A special de-converter transforms the pseudo-randomness back into the message. The authors decided to create their own converter and deconverter instead using a self-synchronizing stream or CBC and CFB block mode ciphers because "most of the output sequences can be distinguished from real or well-designed random number sequences by using birthday paradox distinguishers". These distinguishers are proposed in the paper. Additionally the converter and deconverter synchronizes automatically because information for synchronization reduces indistinguishability. It is also possible to trade-off indistinguishability for data rate. To achieve the goals self-synchronization and random indistinguishability, they make use of nondeterministic input, hash and encryption functions. However, the encryption function makes it necessary to exchange a secret key between sender and receiver. In contrary this message conversion makes it possible to send the same message repeatedly without ending in the same signature.

### The Newton Channel

The Newton channel [AVPN96] can be viewed as an improvement of the subliminal channel because it is constructed similarly and gives an answer to a question from Simmons. Simmons asks if there exists a signature scheme with a broadband channel that does not require the sender to compromise the security of her signing key. Instead of operating in a group of prime order digital signatures are performed in a composite group. From this it follows that operating in a group of prime order would avoid the Newton channel. The channel exists in many discrete logarithm based algorithm [AV96] and it provides a broadband and a scalable narrowband channel. For the broadband channel the modulus of the group $F_p^*$ is chosen as $p = qm + 1$, where $m$ is smooth. Extracting discrete logarithms is hard in the subgroup of order $q$ that is generated by exponentiate a generator $g$ with $m$. If $p - 1 = mq_1q_2 + 1$ of $F_p^*$, and the discrete logarithm problem is hard in the groups of order $q_1$ and $q_2$, then the authentication key can be kept secret by modulo $q_1$ and reveals the message modulo $q_2$. By scaling $q_2$ this narrowband channel can be adjusted to the needed size. It is also shown that many discrete logarithm based systems are insecure because they operate in more than one group at a time in which discrete log can be easy, such that secret key material may leak. With this they can show that the DSA, contrary to earlier belief, does not maximize, but minimizes the subliminal usability of its signature and that it is not vulnerable to the mentioned key material leakage problem [AVPN96]. The narrowband key must be shared explicit with the recipient. Additionally an utilization of the generator $g$ is explained in [AV96].

For further reading, more subliminal and similar channels were found, e.g., in pairing-based digital signatures [WG06], access control systems [CW06], and key escrow crypto systems [Wan98].

# 4 Countermeasures

To prevent subliminal channels in crypto systems they must be considered in the design of the algorithm which leads to subliminal-free algorithms. As the prisoner problem example illustrates, preventing such a channel can be hard. Therefore we will show some possibilities which eliminate the subliminal channel and can fix existing algorithms.

### Assurance of the Randomness and Zero-Knowledge

The main problem is the free choice of random parameters, which is why most countermeasures limit or eliminate this freedom.

For example, Desmedt showed in [Des90] a public key generation in an abuse-free way. Assuming that a public key $n = pq$ is published and either $p$ or $q$ is known by someone, then he can compute the counterpart and therewith a secret message. To prevent this Desmedt lets a second party B, e.g., the warden, actively influence A's randomness, which is used for key generation. For that B has to generate some randomness himself and send it to A. B's randomness is then XORed with the randomness of A. The outcome $R$ of the XOR operation is given into the key generating function. Of course A can ignore B's randomness and omit the XOR operation. Hence, A and B have to agree on a commitment algorithm, which is used by both to commit their randomness. Further they also agree on two more commitment algorithms to verify that $R$ satisfies and not satisfies certain key material conditions, e.g., that A has done the XOR operation with B's randomness. Hence, A is able to proof to B that $R$ satisfies or not satisfies these key material conditions. B can verify A's proof and stops the protocol if the verification fails, such that A cannot use a possible not abuse-free public key. If the verification succeeded, then an abuse-free public key was generated or contrary A has proven to B that the key does not satisfy the key material conditions, such that the protocol restarts.

In the same paper Desmedt presents briefly an abuse-free signature system based on zero-knowledge protocols. First an abuse-free seed $S = R_A \oplus R_B$ is generated in the way of the above abuse-free public key. Then an abuse-free public key $(n, I_1 \ldots, I_k)$ is generated, such that $n = pq$, where both $p$ and $q$ are congruent to 3 modulo 4 and $I_i = s_i^2 \bmod n$. It is assumed that $n$ has indeed this form and that B has been convinced of this. To sign a message $m$ A has to do the following five steps.

1. B chooses a true random $r_i \in QR_n$, and random $k_i$ and sends $E_{k_i}(r_i)$ $(1 \leq i \leq t)$ as commitments to A. $E(\cdot)$ is the commitment function.

2. A picks (random) $\rho_i \in QR_n$ and computes $x_i = \rho_i^2 \bmod n$ and sends these $x_i$ to B.

3. B reveals $(r_i, k_i)$ to A

4. A verifies the commitment. If satisfied, A computes $x'_i = r_i^2 x_i$ and computes $e_{ij}$ starting from $x'_i$, in a similar way as in [FS87].

5. B continues in a similar way as [FS87] with the $x'_i$. B verifies if A has used the $r_i$. A has to prove (using zero-knowledge) to B that all the numbers that were sent, except the $e_{ij}$, are quadratic residues mod $n$. Then B publishes the signature.

This algorithm is not yet abuse-free. Changes to the protocol to overcome this were proposed by Desmedt, but he himself discovered that even with the changes it is not abuse-free [Des96]. Hence, the changed algorithm is not discussed here, but instead we show why it is not abuse-free even with the changes. We show it as an example to demonstrate the difficulty to create a subliminal-free algorithm. Desmedt himself discovered that this protocol has a 1-bit subliminal channel. In step 4 A verifies the commitment and has the ability to stop or continue the protocol. Assumed that A and C agreed to associate a protocol failure with “0” then this can be used to communicate one bit [Des96].

### Meta-ElGamal

Another countermeasure method is presented in [HMP94] where the random parameters are substituted by the outcome of an “Meta-ElGamal” process. The sender chooses a random number $k_1 \in GF^*(q)$ and computes $r_1 \equiv g^{k_1} \bmod p$ and transmits $r_1$ to the warden. The warden can verify that $r_1$ is a generator and chooses $k_2$ which is directly transmitted to the sender. After receiving $k_2$ both parties compute $r_1^{k_2} \bmod p$ and use it in the signature scheme described in the paper. Besides the commitment solution from Desmedt it is also possible to substitute the randomness with help of the Diffie-Hellman key exchange protocol or similar. The derived key is then used instead of the freely chosen random seed.

### Detection of Malicious Behavior

The problem of detecting (potentially covert) malicious behavior by corrupt certification authorities and other generators of digital signatures is considered in [AC05]. They propose “tamper-evident” digital signatures to solve this problem. It is important to clarify that tamper-evident signatures do not prevent corruption. Instead they ensure the detection of any corruption that causes a signer to output signatures that are in any way different from those of an honest signer [AC05, section 4 and 5].

This method is not only useful against malware, but also against insider attacks. An insider attack is for example performed by an implementor of a crypto application which operates correctly under test conditions, but switches to a corrupt mode after deployment. To achieve this the existence of one or more observers is assumed, who do not know any secrets of the signer and therefore can use only public information to detect the presence of a covert channel. An observer is an external node whose task is to inspect all signature transcripts produced by a signer, to detect any variance and to prove the existence of the covert channel to avoid false alarms. As an illustrating example a crypto algorithm which makes use of a sequence of random salt[1] like RSA-PSS is assumed. The random salt is replaced by a hash chain of a one-way hash function to achieve tamper-evidence. The hash chain $s_0, \ldots, s_n$ is build by $s_i = h(s_{i+1})$ $(0 \leq i \leq n-1)$ for some secret seed $s_n$. In the setup phase the value $s_0$ is made public by the signer. The values $s_i$ for $i = 1, \ldots, n$ are now used as a salt for the $i^{th}$ signature. To verify the signature an observer can check the correctness with the equation $s_{i-1} = h(s_i)$. In the following an appropriately modified DSA algorithm is given.

1. Generation algorithm
   - The pair $(x, y = g^x)$ is the private/public key for DSA.
   - A sequence of random values $\{k_i\}$ $(1 \leq i \leq n)$ is generated and later be used for signature generation.
   - Witnesses $\{w_i\}$ $(0 \leq i \leq n)$ are generated by choosing $w_n$ uniformly at random from $\{0,1\}^\kappa$. Where $\kappa$ is a security parameter associated with the choice of the publicly known hash function $h$ used for witness generation.
   - Consecutive witnesses are generated by $w_{i-1} = h(r_i \parallel w_i)$ $(1 \leq i \leq n)$ where $r_i = (g^{k_i} \bmod p) \bmod q$.
   - The new public key is given by $(y, w_0)$ and the private key by $(x, \{k_i\}, \{w_i\})$.
2. Signing
   - The $i^{th}$ message $m_i$ for $i \geq 1$ is signed by $s_i = k_i^{-1}(h(m_i) + xr_i) \bmod q$.
   - The standard DSA signature $(m_i, r_i, s_i)$ along with the witness $w_i$ is given by the signer.
3. Verification
   - The triple $(m_i, r_i, s_i)$ is verified exactly as for regular DSA signatures with respect to the signer's public key $y$.

[1] In cryptography, a salt can be random bits or a previous generated value. It is used to complicate dictionary attacks. [Wik09]

- The covert-validity is checked with the previous witness by $w_{i-1} = h(r_i \parallel w_i)$.

No observer needs to communicate with the signer in order to verify the signatures. Thus, observers can work undercover. In the same paper it is also shown how Schnorr and Feige-Fiat-Shamir signature schemes can be made tamper-evident.

# 5 Conclusion

We discussed what subliminal channels are, and gave examples and intentions for possible use. Developments in both directions, preventing and improving the subliminal channel have been introduced. The existence of the subliminal channel is a practical threat for the crypto system itself and makes it possible to leak secret information. This problem must be considered when choosing appropriate algorithms for crypto systems. Because many crypto systems make use of digital signatures or similar methods, the channel can be found in several protocols, so that it is not sufficient only to suppose it in digital signature systems.

# Bibliography

[AC05] Protecting Certification Authorities and Jong Youl Choi. Tamper-Evident Digital Signatures:. Technical report, In Proceedings of the Symposium on Dependable Autonomic and Secure Computing 2006, 2005. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.61.9340&rep=rep1&type=pdf.

[AV96] Ross Anderson and Serge Vaudenay. Minding Your P's and Q's. In *In Advances in Cryptology - ASIACRYPT'96, LNCS 1163*, pages 26–35. Springer-Verlag, 1996. http://www.cl.cam.ac.uk/~rja14/Papers/psandqs.pdf.

[AVPN96] Ross J. Anderson, Serge Vaudenay, Bart Preneel, and Kaisa Nyberg. The Newton Channel. In *Proceedings of the First International Workshop on Information Hiding*, pages 151–156, London, UK, 1996. Springer-Verlag. http://www.cosic.esat.kuleuven.be/publications/article-54.pdf.

[BD86] Ernest F Brickell and John M DeLaurentis. An attack on a signature scheme proposed by Okamoto and Shiraishi. In *Lecture notes in computer sciences; 218 on Advances in cryptology—CRYPTO 85*, pages 28–32, New York, NY, USA, 1986. Springer-Verlag New York, Inc. http://dsns.csie.nctu.edu.tw/research/crypto/HTML/PDF/C85/28.PDF.

[CW06] Chin-Chen Chang and Chia-Chi Wu. An Improvement of the Design of Integrating Subliminal Channel with Access Control. In *Proceedings of the International MultiConference of Engineers and Computer Scientists 2006*, pages 276–279, Hong Kong, China, 2006. http://www.iaeng.org/IJCS/issues_v32/issue_4/IJCS_32_4_9.pdf.

[Des90] Yvo Desmedt. Abuses in Cryptography and How to Fight Them. In *CRYPTO '88: Proceedings of the 8th Annual International Cryptology Conference on Advances in Cryptology*, pages 375–389, London, UK, 1990. Springer-Verlag. http://dsns.csie.nctu.edu.tw/research/crypto/HTML/PDF/C88/375.PDF.

[Des96] Yvo Desmedt. Simmons' Protocol is Not Free of Subliminal Channels. In *In Proc. of 9th IEEE Computer Security Foundations Workshop*, pages 170–175, 1996. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.56.4816&rep=rep1&type=pdf.

[EG85] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 10–18, New York, NY, USA, 1985. Springer-Verlag New York, Inc. http://groups.csail.mit.edu/cis/crypto/classes/6.857/papers/elgamal.pdf.

[FFS87] U. Feige, A. Fiat, and A. Shamir. Zero knowledge proofs of identity. In *STOC '87: Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 210–217, New York, NY, USA, 1987. ACM.

[FS87] Amos Fiat and Adi Shamir. How To Prove Yourself: Practical Solutions to Identification and Signature Problems. In *Advances in Cryptology CRYPTO 86*, pages 186–194. Springer-Verlag, 1987. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.13.8796&rep=rep1&type=pdf.

[HMP94] Patrick Horster, Markus Michels, and Holger Petersen. Subliminal Channels in discrete logarithm based signature schemes and how to avoid them, 1994. http://citeseer.ist.psu.edu/cache/papers/cs/17652/ftp:zSzzSzftp.cert.dfn.dezSzpubzSzdocszSzcryptzSzTR-94-13.pdf/horster94subliminal.pdf.

[KI97] Kazukuni Kobara and Hideki Imai. Self-Synchronized Message Randomization Methods for Subliminal Channels. In *In Proc. of International Conference on Information and Communications Security (ICICS'97) : LNCS 1334*, pages 325–334. Springer– Verlag, 1997. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.58.660&rep=rep1&type=pdf.

[Lam73] Butler W. Lampson. A note on the confinement problem. *Commun. ACM*, 16(10):613–615, 1973. http://www.cs.cornell.edu/andru/cs711/2003fa/reading/lampson73note.pdf.

[Len04] Arjen K. Lenstra. Key length, 2004. PDF-URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.90.213&rep=rep1&type=url&i=0 and Website with calculation tool: http://www.keylength.com/en/2/.

[NIS94] NIST. Digital Signature Standard, 1994. http://www.itl.nist.gov/fipspubs/fip186.htm.

[Sch95] Bruce Schneier. *Applied Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C, 2. Ed.* Wiley Computer Publishing, John Wiley & Sons, Inc., 1995.

[Sim84] Gustavus J. Simmons. The Prisoners Problem and the Subliminal Channel. In *Advances in Cryptology – CRYPTO '83*, pages 51–67, New York, 1984. Lecture Notes in Computer Science, ed. D. Chaum. `http://dsns.csie.nctu.edu.tw/research/crypto/HTML/PDF/C83/51.PDF`.

[Sim85] G J Simmons. The subliminal channel and digital signatures. In *Proc. of the EUROCRYPT 84 workshop on Advances in cryptology: theory and application of cryptographic techniques*, pages 364–378, New York, NY, USA, 1985. Springer-Verlag New York, Inc. `http://dsns.csie.nctu.edu.tw/research/crypto/HTML/PDF/E84/364.PDF`.

[Sim86] Gustavus J. Simmons. A Secure Subliminal Channel (?). In *CRYPTO '85: Advances in Cryptology*, pages 33–41, London, UK, 1986. Springer-Verlag. `http://dsns.csie.nctu.edu.tw/research/crypto/HTML/PDF/C85/33.PDF`.

[Sim93] Gustavus J. Simmons. The subliminal channel in the U.S. Digital Signature Algorithm (DSA), 1993.

[Sim94] Gustavus J. Simmons. Subliminal communication is easy using the DSA. In *EUROCRYPT '93: Workshop on the theory and application of cryptographic techniques on Advances in cryptology*, pages 218–232, Secaucus, NJ, USA, 1994. Springer-Verlag New York, Inc. `http://dsns.csie.nctu.edu.tw/research/crypto/HTML/PDF/E93/218.PDF`.

[SPMlS02] Jacques Stern, David Pointcheval, John Malone-lee, and Nigel P. Smart. Flaws in Applying Proof Methodologies to Signature Schemes. In *In Advances in Cryptology crypto'02, Santa Barbara, Lectures Notes in Computer Science 2442*, pages 93–110. Springer-Verlag, 2002. `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.108.163&rep=rep1&type=pdf`.

[Wan98] Yongge Wang. Abuses of Probabilistic Encryption Schemes, 1998. `http://www.sis.uncc.edu/~yonwang/papers/Eletter.pdf`.

[WG06] Zhenyou Wang and Wei Gao. Perfect Subliminal Channel in a Paring-Based Digital Signature. *Asian Journal of Information Technology*, 5(4):392–395, 2006. `http://medwelljournals.com/fulltext/ajit/2006/392-395.pdf`.

[Wik09] Wikipedia. Salt (cryptography) — wikipedia, the free encyclopedia, 2009. `http://en.wikipedia.org/w/index.php?title=Salt_(cryptography)&oldid=302579445` [Online; accessed 17-July-2009].