Subliminal Hash Channels

George Teşeleanu[1,2]

[1] Simion Stoilow Institute of Mathematics of the Romanian Academy
21 Calea Grivitei, Bucharest, Romania
[2] Advanced Technologies Institute
10 Dinu Vintilă, Bucharest, Romania
tgeorge@dcti.ro

Abstract. Due to their nature, subliminal channels are mostly regarded as being malicious, but due to recent legislation efforts users' perception might change. Such channels can be used to subvert digital signature protocols without degrading the security of the underlying primitive. Thus, it is natural to find countermeasures and devise subliminal-free signatures. In this paper we discuss state-of-the-art countermeasures and introduce a generic method to bypass them.

1 Introduction

As more and more countries require individuals and providers to hand over passwords and decryption keys [7], we might observe an increase in the usage of *subliminal channels*. Subliminal channels are secondary channels of communication hidden inside a potentially compromised communication channel. The concept was introduced by Simmons [39–41] as a solution to the *prisoners' problem*. In the prisoners' problem *Alice* and *Bob* are incarcerated and wish to communicate confidentially and undetected by their guard *Walter* who imposes to read all their communication. Note that *Alice* and *Bob* can exchange a secret key before being incarcerated.

A special case of subliminal channels are secretly embedded trapdoor with universal protection (SETUP) attacks. By combining subliminal channels with public key cryptography Young and Yung devised a plethora of mechanisms [45–48] to leak a user's private key or message. Although the authors assume a black-box environment[3] in [15] is pointed out that these mechanism can also be implemented in open source software due to the code's sheer complexity and the small number of experts who review it. SETUP attacks are meant to capture the situation in which the manufacturer of a black-box device is also an adversary or employed by an adversary.

According to the classified documents leaked by Snowden [12, 34] the NSA made efforts for subverting cryptographic standards. More precisely, there are strong indications of the existence of a backdoor in the Dual-EC generator [13]. This backdoor is a direct application of Young and Yung's work. Snowden's revelations rekindled the study of backdoors. Thus, more examples of backdoor embedding methods were found [14, 22, 24, 42, 43], methods for protecting users against them were developed [11, 18, 21, 26, 36, 37] and implementations were exploited in the wild [19, 20].

Most subliminal channels or SETUP attacks use random numbers to convey information undetected. In consequence, all the proposed countermeasures focus on sanitizing the random numbers used by a system. In the case of digital signatures, a different but laborious method for inserting a subliminal channel in a system is presented in [44]. Instead of using random numbers as information carriers, *Alice* uses the hash of the message to convey the message for *Bob*. In order to achieve this, *Alice* makes small changes to the message until the hash has the desired properties. Note that the method presented in [44] bypasses all the countermeasures mentioned so far.

This paper studies a generic method that allows the prisoners to communicate through the subliminal-free signatures found in [11, 18, 21, 26, 36, 37]. To achieve our goal we work in a scenario where all messages are

[3] A black-box is a device, process or system, whose inputs and outputs are known, but its internal structure or working is not known or accessible to the user (*e.g.* tamper proof devices, closed source software).

time-stamped before signing. Note that we do not break any of the assumptions made by the subversion-free proposals. This work is motivated by the fact that most end-users to do not verify the claims made by manufacturers[4]. Moreover, users often do not know which should be the outputs of a device [30]. A notable incident in which users where not aware of the correct outputs and trusted the developers is the Debian incident [17].

Structure of the paper. We introduce notations and definitions in Section 2. By adapting and improving the mechanism from [44] we introduce new hash channels in Section 3. A series of experiments is conducted in Section 4. Applications are provided in Section 5. We conclude in Section 6. Additional definitions are given in Appendix A.

2 Preliminaries

Notations. Throughout the paper λ and κ will denote security parameters. We let ROM denote the random oracle model. The number of bits of an element x is denoted by $|x|$ and $x\|y$ represents the concatenation of the strings x and y. The set $\{0,1\}^\ell$ consists of bit strings of length ℓ.

The action of selecting a random element x from a sample space X is represented by $x \overset{\$}{\leftarrow} X$. We also denote by $x \leftarrow y$ the assignment of value y to variable x. The encryption of a message $m \in \{0,1\}$ using one-time pad is denoted by $\omega \leftarrow m \oplus b$, where b is a random bit used only once.

2.1 Diffie-Hellman Assumptions

Definition 1 (Computational Diffie-Hellman - CDH). *Let $\mathbb{G}$ be a cyclic group of order q, g a generator of $\mathbb{G}$ and let A be a probabilistic polynomial-time algorithm (PPT algorithm) that returns an element from $\mathbb{G}$. We define the advantage*

$$ADV^{\text{CDH}}_{\mathbb{G},g}(A) = Pr[A(g^x, g^y) = g^{xy} | x, y \overset{\$}{\leftarrow} \mathbb{Z}_q^*].$$

If $ADV^{\text{CDH}}_{\mathbb{G},g}(A)$ is negligible for any PPT algorithm A, we say that the Computational Diffie-Hellman problem is hard in $\mathbb{G}$.

Definition 2 (Hash Diffie-Hellman - HDH). *Let $\mathbb{G}$ be a cyclic group of order q, g a generator of $\mathbb{G}$, $\mathbb{G}_m$ a set and $H : \mathbb{G} \to \mathbb{G}_m$ a hash function. Let A be a PPT algorithm which returns 1 on input (g^x, g^y, z) if $H(g^{xy}) = z$. We define the advantage*

$$ADV^{\text{HDH}}_{\mathbb{G},g,H}(A) = |Pr[A(g^x, g^y, H(g^{xy})) = 1 | x, y \overset{\$}{\leftarrow} \mathbb{Z}_q^*] - Pr[A(g^x, g^y, z) = 1 | x, y \overset{\$}{\leftarrow} \mathbb{Z}_q^*, z \overset{\$}{\leftarrow} \mathbb{G}_m]|.$$

If $ADV^{\text{HDH}}_{\mathbb{G},g,H}(A)$ is negligible for any PPT algorithm A, we say that the Hash Diffie-Hellman problem is hard in $\mathbb{G}$.

Remark 1. The CDH assumption is standard and we include it for completeness. The HDH assumption was formally introduced in [8, 9], although it was informally described as a composite assumption in [16, 49]. According to [16], the HDH assumption is equivalent with the CDH assumption in the ROM. Although an equivalent of the HDH assumption exists in the standard model, in this paper we are working with the Schnorr signature scheme that is secure in the ROM. Thus, the security in the ROM suffices for our purposes.

[4] Manufacturers might implement subversion-free signatures just for marketing purposes, while still backdooring some of the devices produced.

Hashed Diffie-Hellman Key Exchange (HKE). Based on the HDH assumption we describe a key exchange protocol[5] in Figure 1. A formal analysis of this design can be found in [8,9,23].

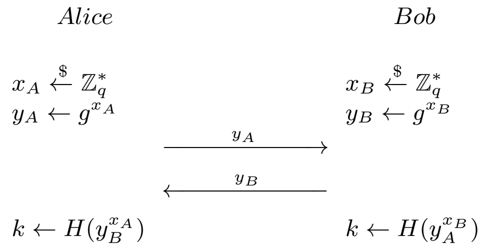
Alice
Bob
$x_A \xleftarrow{\$} \mathbb{Z}_q^*$
$y_A \leftarrow g^{x_A}$
$x_B \xleftarrow{\$} \mathbb{Z}_q^*$
$y_B \leftarrow g^{x_B}$
y_A
y_B
$k \leftarrow H(y_B^{x_A})$
$k \leftarrow H(y_A^{x_B})$

Fig. 1. The Hashed Diffie-Hellman key exchange protocol.

2.2 Digital Signatures

Definition 3 (Signature Scheme). *A Signature Scheme consists of four PPT algorithms: ParamGen, KeyGen, Sign and Verification. The first one takes as input a security parameter and outputs the system's parameters. Using these parameters, the second algorithm generates the public key and the matching secret key. The secret key together with the Sign algorithm are used to generate a signature σ for a message m. Using the public key, the last algorithm verifies if a signature σ for a message m is generated using the matching secret key.*

Remark 2. For simplicity, public parameters will further be considered implicit when describing an algorithm.

Schnorr Signature. In [38], Schnorr introduces a digital signature based on the discrete logarithm problem. Later on, the scheme was proven secure in the ROM by Stern and Pointcheval [35]. We further recall the Schnorr signature.

ParamGen(λ)*:* Generate two large prime numbers p, q, such that $q \geq 2^\lambda$ and $q | p-1$. Select a cyclic group $\mathbb{G}$ of order p and let $g \in \mathbb{G}$ be an element of order q. Let $h : \{0,1\}^* \to \mathbb{Z}_q^*$ be a hash function. Output the public parameters $pp = (p, q, g, \mathbb{G}, h)$.

KeyGen(pp)*:* Choose $x \xleftarrow{\$} \mathbb{Z}_q^*$ and compute $y \leftarrow g^x$. Output the public key $pk = y$. The secret key is $sk = x$.

Sign(m, sk)*:* To sign a message $m \in \{0,1\}^*$, first generate a random number $k \xleftarrow{\$} \mathbb{Z}_q^*$. Then compute the values $r \leftarrow g^k$, $e \leftarrow h(r \| m)$ and $s \leftarrow k - xe \bmod q$. Output the signature (e, s).

Verification(m, e, s, pk)*:* To verify the signature (e, s) of message m, compute $r \leftarrow g^s y^e$ and $u \leftarrow h(r \| m)$. Output `true` if and only if $u = e$. Otherwise, output `false`.

2.3 Subliminal Channels and SETUP attacks

Covert channels [31] have the capability of transporting information through system parameters apparently not intended for information transfer. Subliminal channels and SETUP attacks are special cases of covert channels and achieve information transfer by modifying the original specifications of cryptographic primitives[6]. We further restrict covert channels to two sub-cases: subliminal channels and SETUP attacks.

Definition 4 (Subliminal channel). *A Subliminal channel is an algorithm that can be inserted in a system such that it allows the system's owner to communicate*[7] *with a recipient without their communication being*

[5] a high level description of the IKE protocols [27,28]

[6] for example, by modifying the way random numbers are generated

[7] through the system's outputs

detected by a third party[8]. *It is assumed that the prisoners' communication is encrypted using a secret/public key encryption scheme and the decryption function is accessible to the recipient.*

Definition 5 (Secretly Embedded Trapdoor with Universal Protection - SETUP**).** *A Secretly Embedded Trapdoor with Universal Protection (SETUP) is an algorithm that can be inserted in a system such that it leaks encrypted private key information to an attacker through the system's outputs. Encryption of the private key is performed using a public key encryption scheme. It is assumed that the decryption function is accessible only to the attacker.*

Remark 3. Note that SETUP mechanisms are special cases of subliminal channels. In the SETUP case, the sender is the system, the recipient is the attacker, while the third party is the owner of the system.

Definition 6 (Covert channel indistinguishability - IND-COVERT**).** *Let C_0 be a black-box system that uses a secret key sk. Let $\mathcal{E}$ be the encryption scheme used by a covert channel as defined above, in Definitions 4 and 5. We consider C_1 an altered version of C_0 that contains a covert channel based on $\mathcal{E}$. Let A be a PPT algorithm which returns 1 if it detects that C_0 is altered. We define the advantage*

$$ADV^{\text{IND-COVERT}}_{\mathcal{E},C_0,C_1}(A) = |Pr[A^{C_1(sk,\cdot)}(\lambda) = 1] - Pr[A^{C_0(sk,\cdot)}(\lambda) = 1]|.$$

If $ADV^{\text{IND-COVERT}}_{\mathcal{E},C_0,C_1}(A)$ is negligible for any PPT algorithm A, we say that C_0 and C_1 are polynomially indistinguishable.

Remark 4. In some cases, if sk is known, the covert channel can be detected by using its description and parameters. Thus, depending on the context we will specify if A has access to sk or not. If $\mathcal{E}$ is a public key encryption scheme we always assume that A has access to the public key[9].

We consider that the covert channels presented from now on are implemented in a device D that digitally signs messages. In the case of subliminal channels, the prisoners are denoted as *Alice* (sender) and *Bob* (receiver), while $Walter$ is the guard. In the case of SETUP attacks, the owner of the device is referred to as $Charlie$ and the attacker is usually $Mallory$. When the secret key sk is not known to the PPT algorithm A we assume that sk is stored only in D's volatile memory. Note that $Walter$ and $Charlie$ believe that D signs messages using the original specifications of the signature scheme implemented in D. When one of the original signature's algorithm is not modified by the covert channel, the algorithm will be omitted when presenting the respective channel.

Throughout the paper, when presenting covert channels, we make use of the following additional algorithms:

- *Subliminal/Malicious ParamGen* – used by the prisoners/attacker to generate their (his) parameters;
- *Subliminal/Malicious KeyGen* – used by the prisoners/attacker to generate their (his) keys;
- *Extract* – used by the recipient to extract the secret message;
- *Recovering* – used by the attacker to recover $Charlie$'s secret key.

The algorithms above are not implemented in D. For simplicity, covert parameters will further be implicit when describing an algorithm.

Trivial Subliminal Channel. The Schnorr signature supports a subliminal channel based on rejection sampling. We further describe the trivial subliminal channel.

Sign(m, sk)*:* Choose $k \xleftarrow{\$} \mathbb{Z}_q^*$ and compute $r \leftarrow g^k$, until $\omega \equiv r \bmod 2$. To sign a message $m \in \{0,1\}^*$ compute the values $e \leftarrow h(r\|m)$ and $s \leftarrow k - xe \bmod q$. Output the signature (e, s).

Extract(e, s) : To extract the embedded message ω compute $\omega \leftarrow g^s y^e \bmod 2$.

[8] The sender and receiver will further be called prisoners and the third party warden.

[9] found by means of reverse engineering the system, for example

Young-Yung SETUP Attack In [45–48], the authors propose a kleptographic version of Schnorr signatures and prove it IND-COVERT secure in the standard model under the HDH assumption. The algorithms of the SETUP attack are shortly described below. Note that after D signs at least two messages, $Mallory$ can recover $Charlie$'s secret key and, thus, impersonate $Charlie$.

Malicious ParamGen(pp): Let $H : \mathbb{G} \to \mathbb{Z}_q^*$ be a hash function. Output the public parameter $sp_M = H$. Note that H will be stored in D's volatile memory.

Malicious KeyGen(pp): Choose $x_M \xleftarrow{\$} \mathbb{Z}_q^*$ and compute $y_M \leftarrow g^{x_M}$. Output the public key $pk_M = y_M$. The public key pk_M will be stored in D's volatile memory. The secret key is $sk_M = x_M$; it will only be known by $Mallory$ and will not be stored in the black-box.

Signing Sessions: The possible signing sessions performed by D are described below. Let $i \geq 1$.

Session$_0(m_0, sk)$: To sign message $m_0 \in \mathbb{G}$, D does the following

$$k_0 \xleftarrow{\$} \mathbb{Z}_q^*, r_0 \leftarrow g^{k_0}, e_0 \leftarrow h(r_0 \| m_0), s_0 \leftarrow k_0 - xe_0 \bmod q.$$

The value k_0 is stored in D's volatile memory until the end of *Session*$_1$. Output the signature (r_0, s_0).

Session$_i(m_i, sk, pk_M)$: To sign message $m_i \in \mathbb{G}$, D does the following

$$z_i \leftarrow y_M^{k_{i-1}}, k_i \leftarrow H(z_i), r_i \leftarrow g^{k_i}, e_i \leftarrow h(r_i \| m_i), s_i \leftarrow k_i - xe_i \bmod q.$$

The value k_i is stored in D's volatile memory until the end of *Session*$_{i+1}$. Output the signature (r_i, s_i).

Recovering$(m_i, e_{i-1}, e_i, s_i, sk_M)$*:* Compute $r_{i-1} \leftarrow g^{s_{i-1}} y^{e_{i-1}}$, $\alpha \leftarrow r_{i-1}^{x_M}$ and $k_i \leftarrow H(\alpha)$. Recover x by computing $x \leftarrow e_i^{-1}(k_i - s_i) \bmod q$.

3 Hash Channels

In order to be valid, legal documents need a timestamp appended to them before being digitally signed [10,25]. According to [10] the timestamp must include seconds. Note that if the timestamp module is independent from the *Sign* module, then $Walter$ or $Charlie$ can inject false timestamps into the signing module. Thus, we assume that the timestamp module is integrated in the signing module. Using this framework we achieve a subliminal channel by adapting and simplifying the idea from [44].

Let *lim* be an upper limit for the number of trials and u_t the smallest time unit used by the time stamping algorithm (*e.g.* seconds, milliseconds). We further present our proposed subliminal channel.

Time Stamp(u_t)*:* Output the current time τ including u_t.

Subliminal Sign(m, ω, sk)*:* Generate a random number $k \xleftarrow{\$} \mathbb{Z}_q^*$ and compute $r \leftarrow g^k$. Let *counter* $= 1$. Generate τ using the *Time Stamp* algorithm and compute $e \leftarrow h(r \| m \| \tau)$ and *counter* $=$ *counter* $+ 1$, until $e \equiv \omega \bmod 2$ or *counter* $=$ *lim*. Compute $s \leftarrow k - xe \bmod q$. Output the signature (e, s).

Extract(m, e, s, pk) : To extract the embedded message compute $\omega \leftarrow e \bmod 2$. Remark that the probability of event $e \equiv \omega \bmod 2$ is $1 - 1/2^{lim}$.

The security of the Schnorr signature scheme is preserved, since we are not modifying the scheme itself, but the way messages are processed. Let τ_h be the average time it takes device D to compute $h(r \| m \| \tau)$ for fixed bit-size bit_m messages. To avoid detection by $Walter$ or $Charlie$ the manufacturer writes in D's specification that for a message of size bit_m it takes $lim \cdot \tau$ to sign bit_m messages. Thus, D remains consistent with the specifications (*i.e.* IND-COVERT secure). The main restriction when choosing *lim* is users' usability. Due to the hash-rate statistics reported for SHA-256 in [2, 3] we can assume $\tau_h < 1$ second. Thus, the bottleneck becomes the time stamp (*i.e.* D can not output a signature for time t at time $t - 1$). This can be mitigated by including finer time units into the timestamp (*e.g* milliseconds).

Remark 5. Let $z \leftarrow g^t$ be the public key of *Bob* and b a bit *Alice* wants to send to *Bob*. Then, we can easily transition to a public key subliminal channel by using HKE and computing $\omega \leftarrow b \oplus H(z^k)$, where $\mathbb{G}_m = \{0, 1\}$. Since k is fresh for each signature, *Alice* can continuously leak data to *Bob*. Note that we are using HKE to encrypt the message so, ω is indistinguishable from a random bit. Thus, the scheme is IND-COVERT secure under the HDH assumption. We further denote this public key subliminal channel by hash_p.

Remark 6. When dealing with longer messages there is a simpler way to transmit them. Thus, let m_i be the ith bit of m and $\mathbb{G}_m = \{0, 1\}^{|m|}$. The device D can leak m to *Bob* through $|m|$ signing sessions by computing $c \leftarrow m \oplus H(z^{k_0})$ and setting $\omega \leftarrow c_i$ for the ith signing session, where $0 \leq i < |m|$. Note that m is successfully transmitted with a probability of $(1 - 1/2^{lim})^{|m|}$. This channel is further denoted by hash_ℓ. Remark that if we replace *Bob* with *Mallory* and set $m \leftarrow x$, hash_ℓ is transformed into a SETUP attack.

Remark 7. If adversary A has access to x, then he can compute all k numbers. Thus, hash_p and hash_ℓ can be detected if x can be retrieved from D, while the regular hash channel it is not. Hence, in the public key setting we assume that x is only stored in D's volatile memory[10].

4 Stochastic Detection

In [29], the authors show that the execution time of the Young-Yung attack can be used to distinguish honest devices from backdoored devices. Using Kucner *et. al.* observations as a starting point, we run a series of experiments to see if our proposed methods can be detected by measuring their execution time.

We implemented in C using the GMP library [6] the Schnorr signature (normal), the trivial channel (trivial), the Young-Yung attack (yy), the hash channel (hash), the public key hash channel (hash_p) and hash_p's extension to long messages (hash_ℓ). The programs were run on a CPU Intel i7-4790 4.00 GHz and compiled with GCC with the O3 flag activated. In our experiments for each prime size of 2048, 3072, 4096 and 8192 bits, we ran the algorithms with 100 safe prime numbers from [5]. For each prime we measured the average running time for 128 random 2040 byte messages[11] using the function *omp_get_wtime*() [4]. Before signing each message we added a 8 byte timestamp with the current system time in milliseconds (*clock_gettime*()). The hash function used internally by the algorithms is either SHA256 or SHA512 [1].

When we implemented the hash channels we took advantage of the Merkle-Damgard structure of SHA256 or SHA512. Thus, we computed and stored the intermediary value h_{it} obtained after processing 1984 (SHA256) or 1920 (SHA512) bytes. Then for each trial we used h_{it} to process the last block of the message. Note that the size of the messages was selected such that after h_{it} the SHA functions must process one full message block and a full padding block (worst case scenario). Also, in our experiments lim tends towards infinity.

The results of our experiments are presented in Figures 2 to 9. We can see from the plots that the Schnorr signature and the hash channel have similar execution times. We further investigated this by computing the absolute time difference between a normal execution (t_{n_1}) and a hash channel execution (t_h) or another normal execution (t_{n_2}). The results are presented in Table 1. Note that the empirical evidence suggests that the normal and hash channel executions are indistinguishable due to the noise added by the operating system.

When we implemented hash_ℓ we distributed the HKE protocol execution over 128 Schnorr signature computations. The downside of this method is that first we need to use 128 Schnorr signatures for masking the HKE and then 128 hash channel signatures for leaking the message. The results presented in this section are only for the first part, since experimental data for hash channels is already presented. Note that the first part of hash_ℓ is indistinguishable from the normal execution. As in the case of the hash channel, we

[10] The same assumption is make in Young-Yung's attack, since their mechanism can also be detected when x is known to the attacker.

[11] By choosing 128 messages we simulated the following scenario: the secret key x is generated using a PRNG with a seed of 128 bits and D leaks the seed.

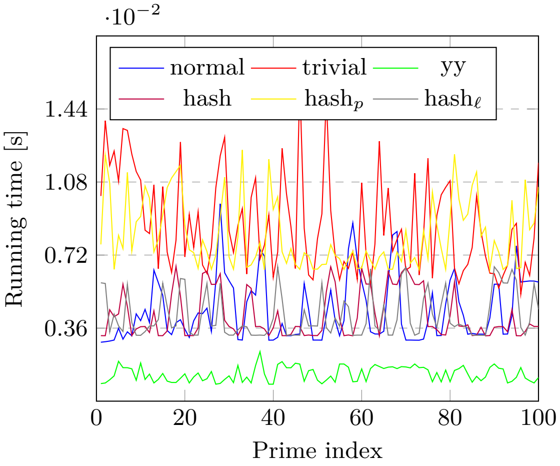

Fig. 2. Prime's size 2048 bits with SHA256

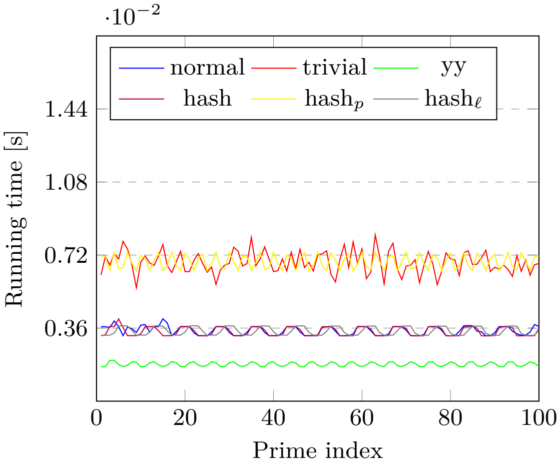

Fig. 3. Prime's size 2048 bits with SHA512

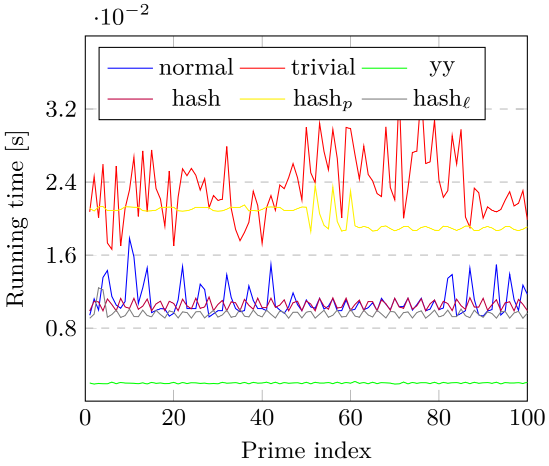

Fig. 4. Prime's size 3072 bits with SHA256

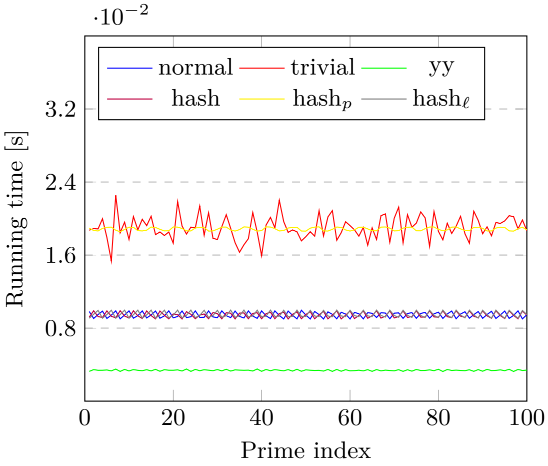

Fig. 5. Prime's size 3072 bits with SHA512

Prime's size	SHA	$\|t_{n_1} - t_{n_2}\|$	$\|t_{n_1} - t_h\|$	$\|t_{n_1} - t_\ell\|$
2048	256	0.185621	0.138921	0.149650
	512	0.122050	0.097460	0.101445
3072	256	0.462406	0.105996	0.150646
	512	0.156232	0.156667	0.160375
4096	256	0.523229	0.354953	0.358049
	512	0.118666	0.134868	0.085795
8192	256	1.381028	1.548863	1.586020
	512	0.483661	0.629464	0.468742

Table 1. Time comparison

further investigated the indistinguishability claim by computing the absolute time difference between a normal execution (t_{n_1}) and a hash$_\ell$ channel execution (t_ℓ). The results are presented in Table 1. Note that the empirical evidence suggests that the normal and hash$_\ell$ channel executions are indistinguishable due to the noise added by the operating system.

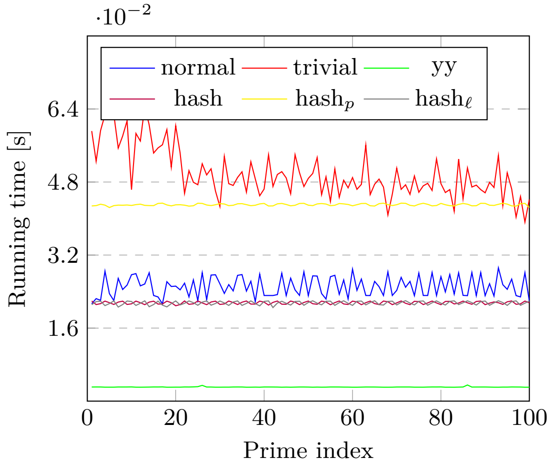

Fig. 6. Prime's size 4096 bits with SHA256

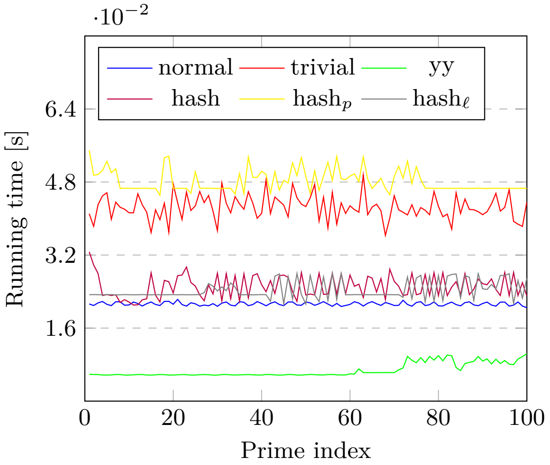

Fig. 7. Prime's size 4096 bits with SHA512

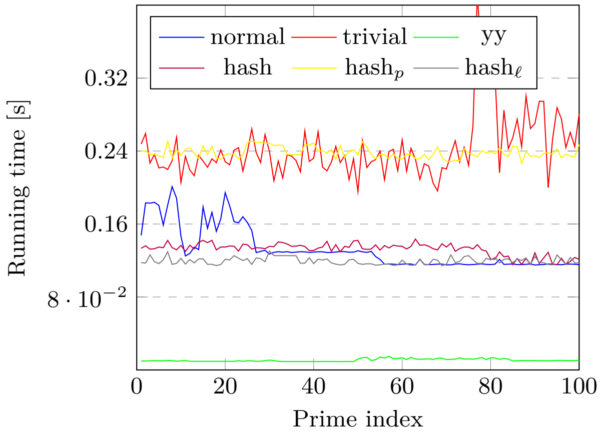

Fig. 8. Prime's size 8192 bits with SHA256

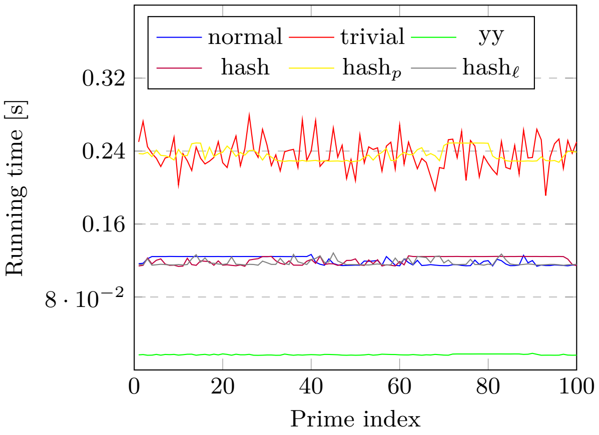

Fig. 9. Prime's size 8192 bits with SHA512

Another remark is that the rest of the channels can be easily detected by measuring their execution time. Thus, noise must be added to the Young-Yung attack or to the Schnorr signature in order to make the subliminal channels undetectable. Note that the trivial channel and the public key hash channel have similar execution times. Thus, any technique used to mask the execution time of the trivial channel can also be used for the public key hash channel.

Let T be the computation time for one signature. We denote by $E[T]$ and $\sigma[T]$ the expected value and the standard deviation of T. Kucner *et. al.* introduce the $R[T] = \sigma[T]/E[T]$ characteristic in order to measure computation time independently of the actual speed of the processor. We computed $R[T]$ for all channels and the results are presented in Table 2. We can observe from our experiments that the $R[T]$ characteristic fluctuates in practice. Also, from Table 2, it is easy to observe that the $R[T]$ characteristic for the Schnorr signature is always smaller than the one for the trivial channel and the Young-Yung attack. Thus, we can distinguish these two channels from an honest execution. Unfortunately, the results are inconclusive for the rest of the channels.

Prime's size	SHA	normal	trivial	yy	hash	hash$_p$	hash$_\ell$
2048	256	0.128196	0.722334	0.322048	0.076789	0.138364	0.147544
	512	0.035308	0.695684	0.131866	0.017010	0.033390	0.094461
3072	256	0.094644	0.751065	0.430531	0.044940	0.063633	0.095584
	512	0.024800	0.718313	0.207609	0.024917	0.044754	0.092278
4096	256	0.131263	0.700937	0.582434	0.043187	0.045583	0.101174
	512	0.051705	0.691449	0.326993	0.125705	0.089238	0.140214
8192	256	0.116920	0.704363	1.156188	0.172762	0.101328	0.132343
	512	0.056456	0.708207	0.594154	0.057846	0.086518	0.121710

Table 2. $R[T]$ characteristic

5 Marketing Backdoors

In this section we provide the reader with state-of-the-art countermeasures used to obtain subliminal-free signatures and show that three proposals are vulnerable to the hash and hash$_\ell$ channels without masking the channels' execution time, while for the rest the channels must be masked. Thus, a manufacturer can market a product as being subliminal free[12], while in reality it is not. Note that our proposed scenario does not violate the assumptions made by the subversion-free protocols.

5.1 Russel *et al.* Subversion-Free Proposal

The authors of [36, 37] assume that all the random numbers used by a signature scheme are generated by a malicious RNG (including the key generation step). Based on this assumption, the authors describe and prove secure a generic method for protecting users. Note that both the trivial channel and the Young-Yung attack can be modeled as malicious RNGs. Unfortunately, in the hash channel scenario the security of their method breaks down.

The philosophy behind Russel *et al.* method is to split every generation algorithm into two parts: a random string generation part RG and a deterministic part DG. By extensively testing DG the user can be ensured that the deterministic part is *almost consistent* with the specifications. By using two independent RNG modules $Source_1, Source_2$ and hashing their concatenated outputs, any backdoors implemented in the RNGs will not propagate into DG. We further describe an instantiation of [37] using the Schnorr signature scheme.

Random$(Source_1, Source_2)$*:* Generate $s_1 \overset{\$}{\leftarrow} Source_1$ and $s_2 \overset{\$}{\leftarrow} Source_2$. Output $h(s_1 \| s_2)$.

KeyGen(pp)*:* Generate x using the *Random* algorithm and compute $y \leftarrow g^x$. Output the public key $pk = y$. The secret key is $sk = x$.

Sign(m, sk)*:* To sign a message $m \in \{0,1\}^*$, first generate k using the *Random* algorithm. Then, compute the values $r \leftarrow g^k$, $e \leftarrow h(r\|m)$ and $s \leftarrow k - xe \bmod q$. Output the signature (e, s).

5.2 Hanzlik *et al.* Controlled Randomness Proposal

A method for controlling the quality of k is proposed in [26]. In order to do this, the authors use a blinding factor $U \leftarrow g^u$ that is installed by the owner of the device and a counter i. The owner accepts a signature produced by D if and only if *Check* returns `true`. Note that the Young-Yung SETUP attack is not possible due to the blinding factor. We further present their modifications on the *Sign* algorithm.

[12] by implementing one of these countermeasures

Sign(m, U, i, sk)*:* To sign a message $m \in \{0,1\}^*$, first generate $k_0 \xleftarrow{\$} \mathbb{Z}_q^*$, compute $r' \leftarrow g^{k_0}$, $k_1 \leftarrow H(U^{k_0}, i)$ and increment i. Let $k \leftarrow k_0 k_1$. Compute the values $r \leftarrow g^k$, $e \leftarrow h(r \| m)$ and $s \leftarrow k - xe \bmod q$. Output the signature (e, s) and the control data (r', i).

Check(e, s, r', u)*:* Compute $r \leftarrow g^s y^e$ and $\alpha \leftarrow H(r'^u, i)$. Output `true` only if and only if $r = r'^\alpha$. Otherwise, output `false`.

The authors underline that a subliminal channel[13] exists, but due to the limited memory of the signing device, hiding the time needed to implement their proposed channel is difficult. Note that our timestamp method proposed in Section 3 is much faster[14] and, thus, in some cases is feasible for bypassing Hanzlik *et al.* mechanism.

5.3 Choi *et al.* Tamper-Evident Digital Signatures

Choi *et al.* [21] introduce the notion of tamper-evidence for digital signatures in order to prevent corrupted nodes to covertly leak secret information. We further provide the tamper-evident Schnorr signature.

ParamGen(λ)*:* Generate two large prime numbers p, q, such that $q \geq 2^\lambda$ and $q | p-1$. Select a cyclic group $\mathbb{G}$ of order p and let $g \in \mathbb{G}$ be an element of order q. Let $h : \{0,1\}^* \to \mathbb{Z}_q^*$ be a hash function and let ℓ be the number of permitted signatures. Output the public parameters $pp = (p, q, g, \mathbb{G}, h, \ell)$.

KeyGen(pp, κ)*:* Choose $x \xleftarrow{\$} \mathbb{Z}_q^*$ and compute $y \leftarrow g^x$. Also, choose $\omega_\ell \xleftarrow{\$} \{0,1\}^\kappa$. For $1 \leq i \leq \ell$, generate $k_i \xleftarrow{\$} \mathbb{Z}_q^*$ and compute $\omega_{i-1} \leftarrow h(g^{k_i} \| \omega_i)$. Output the public key $pk = (y, \omega_0)$. The secret key is $sk = (x, k_1, \ldots, k_\ell, \omega_1, \ldots, \omega_\ell)$.

Sign(m_i, sk)*:* To sign the ith message $m_i \in \{0,1\}^*$, compute the values $r_i \leftarrow g^{k_i}$, $e_i \leftarrow h(r_i \| \omega_i \| m_i)$ and $s_i \leftarrow k_i - xe_i \bmod q$. Output the signature (e_i, s_i, ω_i).

Verification$(m_i, e_i, s_i, \omega_i, pk)$*:* To verify the signature (e_i, s_i, ω_i) of message m_i, compute $r_i \leftarrow g^{s_i} y^{e_i}$ and $u \leftarrow h(r_i \| \omega_i \| m_i)$. Output `true` if and only if $u = e_i$ and $\omega_{i-1} \leftarrow h(r_i \| \omega_i)$. Otherwise, output `false`.

The authors work in the honest key generation model. Thus, the nodes can not manipulate the k_is in any way. Fortunately, our hash channels use messages to leak confidential data. Thus, the nodes can still subliminally transmit data by using our proposed channels.

5.4 Ateniese *et al.* and Bohli *et al.* Subversion-Free proposals

The authors of [11] propose the usage of re-randomizable signatures[15] and unique signatures[15] as countermeasures to backdoors induced by malicious RNGs. These proposals are secure according to their security model [11]. A similar approach can be found in [18], where the authors convert the Digital Signature Algorithm into a deterministic signature. Note that both approaches assume honest key generation.

All these signature schemes work on fixed length messages and internally use a number theoretic hash function[16]. In order to work on variable length messages a standard hash function is used to process the message and the resulting hash is used as input for the Naor-Reingold function. Thus, for each small change in the message we have to recompute the hash $h(m)$, multiply $|h(m)/2|$ integers from $\mathbb{Z}_q^*$ and perform an exponentiation in $\mathbb{G}$. So, our proposed hash channel on average doubles the time necessary to process a message. In this case, the execution time of a hash channel is no longer similar to an honest implementation and, thus, noise must be added to mask the backdoor.

[13] similar to the trivial channel described in Section 2.3

[14] *i.e.* computing a hash is faster than computing a modular exponentiation

[15] See Appendix A for a definition of the concept.

[16] more precisely, the Naor-Reingold pseudo-random function [32, 33]

6 Conclusions

In this paper we introduced a scenario in which the security of the subliminal-free methods presented in [21,26,36,37] degrades. We also conducted a series of experiments to show that the hash and hash$_\ell$ channels' executions are indistinguishable from the normal signature executions due to the noise produced by the operating system. Hence, we proved that users must request justifications for every security choice made by the manufacturer and that testing centers must never let the DG modules to modify inputs.

References

1. mbed TLS. https://tls.mbed.org
2. Mining Hardware Comparison. https://en.bitcoin.it/wiki/Mining_hardware_comparison
3. Non-Specialized Hardware Comparison. https://en.bitcoin.it/wiki/Non-specialized_hardware_comparison
4. OpenMP. https://www.openmp.org/
5. Safe Prime Database. https://2ton.com.au/safeprimes/
6. The GNU Multiple Precision Arithmetic Library. https://gmplib.org/
7. World Map of Encryption Laws and Policies. https://www.gp-digital.org/world-map-of-encryption/
8. Abdalla, M., Bellare, M., Rogaway, P.: DHAES: An Encryption Scheme Based on the Diffie-Hellman Problem. IACR Cryptology ePrint Archive **1999/7** (1999)
9. Abdalla, M., Bellare, M., Rogaway, P.: The Oracle Diffie-Hellman Assumptions and an Analysis of DHIES. In: CT-RSA 2001. Lecture Notes in Computer Science, vol. 2020, pp. 143–158. Springer (2001)
10. Adams, C., Cain, P., Pinkas, D., Zuccherato, R.: RFC 3161: Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP). Tech. rep., Internet Engineering Task Force (2001)
11. Ateniese, G., Magri, B., Venturi, D.: Subversion-Resilient Signature Schemes. In: ACM-CCS 2015. pp. 364–375. ACM (2015)
12. Ball, J., Borger, J., Greenwald, G.: Revealed: How US and UK Spy Agencies Defeat Internet Privacy and Security. The Guardian **6** (2013)
13. Barker, E., Kelsey, J.: SP 800-90A. Recommendations for Random Number Generation Using Deterministic Random Bit Generators (2012)
14. Bellare, M., Jaeger, J., Kane, D.: Mass-Surveillance without the State: Strongly Undetectable Algorithm-Substitution Attacks. In: ACM-CCS 2015. pp. 1431–1440. ACM (2015)
15. Bellare, M., Paterson, K.G., Rogaway, P.: Security of Symmetric Encryption Against Mass Surveillance. In: CRYPTO 2014. Lecture Notes in Computer Science, vol. 8616, pp. 1–19. Springer (2014)
16. Bellare, M., Rogaway, P.: Minimizing the Use of Random Oracles in Authenticated Encryption Schemes. In: ICICS 1997. Lecture Notes in Computer Science, vol. 1334, pp. 1–16. Springer (1997)
17. Bello, L.: DSA-1571-1 OpenSSL—Predictable Random Number Generator. https://www.debian.org/security/2008/dsa-1571 (2008)
18. Bohli, J., Vasco, M.I.G., Steinwandt, R.: A subliminal-free variant of ECDSA. In: IH 2006. Lecture Notes in Computer Science, vol. 4437, pp. 375–387. Springer (2006)
19. Checkoway, S., Maskiewicz, J., Garman, C., Fried, J., Cohney, S., Green, M., Heninger, N., Weinmann, R.P., Rescorla, E., Shacham, H.: A Systematic Analysis of the Juniper Dual EC Incident. In: ACM-CCS 2016. pp. 468–479. ACM (2016)
20. Checkoway, S., Niederhagen, R., Everspaugh, A., Green, M., Lange, T., Ristenpart, T., Bernstein, D.J., Maskiewicz, J., Shacham, H., Fredrikson, M.: On the Practical Exploitability of Dual EC in TLS Implementations. In: USENIX Security Symposium. pp. 319–335. USENIX Association (2014)
21. Choi, J.Y., Golle, P., Jakobsson, M.: Tamper-Evident Digital Signature Protecting Certification Authorities Against Malware. In: DASC 2006. pp. 37–44. IEEE (2006)
22. Dodis, Y., Ganesh, C., Golovnev, A., Juels, A., Ristenpart, T.: A Formal Treatment of Backdoored Pseudorandom Generators. In: EUROCRYPT 2015. Lecture Notes in Computer Science, vol. 9056, pp. 101–126. Springer (2015)
23. Dodis, Y., Gennaro, R., Håstad, J., Krawczyk, H., Rabin, T.: Randomness Extraction and Key Derivation Using the CBC, Cascade and HMAC Modes. In: CRYPTO 2004. Lecture Notes in Computer Science, vol. 3152, pp. 494–510. Springer (2004)
24. Fried, J., Gaudry, P., Heninger, N., Thomé, E.: A Kilobit Hidden SNFS Discrete Logarithm Computation. In: EUROCRYPT 2017. Lecture Notes in Computer Science, vol. 10210, pp. 202–231. Springer (2017)

25. Haber, S., Stornetta, W.S.: How to Time-Stamp a Digital Document. In: CRYPTO 1990. Lecture Notes in Computer Science, vol. 537, pp. 437–455. Springer (1990)
26. Hanzlik, L., Kluczniak, K., Kutyłowski, M.: Controlled Randomness - A Defense against Backdoors in Cryptographic Devices. In: MyCrypt 2016. Lecture Notes in Computer Science, vol. 10311, pp. 215–232. Springer (2016)
27. Harkins, D., Carrel, D.: RFC 2409: The Internet Key Exchange (IKE). Tech. rep., Internet Engineering Task Force (1998)
28. Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., Kivinen, T.: RFC7296: Internet Key Exchange Protocol Version 2 (IKEv2). Tech. rep., Internet Engineering Task Force (2014)
29. Kucner, D., Kutyłowski, M.: Stochastic kleptography detection. In: Public-Key Cryptography and Computational Number Theory. pp. 137–149 (2001)
30. Kwant, R., Lange, T., Thissen, K.: Lattice Klepto - Turning Post-Quantum Crypto Against Itself. In: SAC 2017. Lecture Notes in Computer Science, vol. 10719, pp. 336–354. Springer (2017)
31. Lampson, B.W.: A Note on the Confinement Problem. Communications of the ACM **16**(10), 613–615 (1973)
32. Naor, M., Reingold, O.: Number-theoretic constructions of efficient pseudo-random functions. In: FOCS 1997. pp. 458–467. IEEE Computer Society (1997)
33. Naor, M., Reingold, O.: Number-Theoretic Constructions of Efficient Pseudo-Random Functions. Journal of the ACM (JACM) **51**(2), 231–262 (2004)
34. Perlroth, N., Larson, J., Shane, S.: NSA Able to Foil Basic Safeguards of Privacy on Web. The New York Times **5** (2013)
35. Pointcheval, D., Stern, J.: Security Proofs for Signature Schemes. In: EUROCRYPT 1996. Lecture Notes in Computer Science, vol. 1070, pp. 387–398. Springer (1996)
36. Russell, A., Tang, Q., Yung, M., Zhou, H.S.: Cliptography: Clipping the power of kleptographic attacks. In: ASIACRYPT 2016. Lecture Notes in Computer Science, vol. 10032, pp. 34–64. Springer (2016)
37. Russell, A., Tang, Q., Yung, M., Zhou, H.S.: Generic Semantic Security against a Kleptographic Adversary. In: ACM-CCS 2017. pp. 907–922. ACM (2017)
38. Schnorr, C.P.: Efficient Identification and Signatures For Smart Cards. In: CRYPTO 1989. Lecture Notes in Computer Science, vol. 435, pp. 239–252. Springer (1989)
39. Simmons, G.J.: The Subliminal Channel and Digital Signatures. In: EUROCRYPT 1984. Lecture Notes in Computer Science, vol. 209, pp. 364–378. Springer (1984)
40. Simmons, G.J.: Subliminal Communication is Easy Using the DSA. In: EUROCRYPT 1993. Lecture Notes in Computer Science, vol. 765, pp. 218–232. Springer (1993)
41. Simmons, G.J.: Subliminal Channels; Past and Present. European Transactions on Telecommunications **5**(4), 459–474 (1994)
42. Teşeleanu, G.: Unifying Kleptographic Attacks. In: NordSec 2018. Lecture Notes in Computer Science, vol. 11252, pp. 73–87. Springer (2018)
43. Teşeleanu, G.: Managing Your Kleptographic Subscription Plan. In: C2SI 2019. Lecture Notes in Computer Science, vol. 11445, pp. 452–461. Springer (2019)
44. Wu, C.K.: Hash channels. Computers & Security **24**(8), 653–661 (2005)
45. Young, A., Yung, M.: The Dark Side of "Black-Box" Cryptography or: Should We Trust Capstone? In: CRYPTO 1996. Lecture Notes in Computer Science, vol. 1109, pp. 89–103. Springer (1996)
46. Young, A., Yung, M.: Kleptography: Using Cryptography Against Cryptography. In: EUROCRYPT 1997. Lecture Notes in Computer Science, vol. 1233, pp. 62–74. Springer (1997)
47. Young, A., Yung, M.: The Prevalence of Kleptographic Attacks on Discrete-Log Based Cryptosystems. In: CRYPTO 1997. Lecture Notes in Computer Science, vol. 1294, pp. 264–276. Springer (1997)
48. Young, A., Yung, M.: Malicious Cryptography: Exposing Cryptovirology. John Wiley & Sons (2004)
49. Zheng, Y., Seberry, J.: Immunizing Public Key Cryptosystems Against Chosen Ciphertext Attacks. IEEE Journal on Selected Areas in Communications **11**(5), 715–724 (1993)

A Additional Preliminaries

Definition 7 (Unique Signature Scheme). *Let S be a signature scheme and pk be a public key generated by the KeyGen algorithm of S. We say that S is a Unique Signature Scheme if for any message m and any signatures of m, $\sigma_1 \neq \sigma_2$*

$$Pr[\,\mathit{Verification}(m, \sigma_1, pk) = \mathit{Verification}(m, \sigma_2, pk) = \texttt{true}]$$

is negligible.

Definition 8 (Re-Randomizable Signature Scheme). *Let S be a signature scheme and (pk, sk) be a public/secret key pair generated by the $KeyGen$ algorithm of S. We say that S is a Re-Randomizable Signature Scheme if there exists a PPT algorithm ReRand such that for all messages m the output of ReRand(m, σ, pk) is statistically indistinguishable from Sign(m, sk).*