Communicating Through Subliminal-Free Signatures

George Teşeleanu

Institute of Mathematics of the Romanian Academy, Bucharest, Romania
george.teseleanu@imar.ro

Abstract. By exploiting the inherent randomness used by certain digital signature protocols, subliminal channels can subvert these protocols without degrading their security. Due to their nature, these channels cannot be easily detected by an outside observer. Therefore, they pose a severe challenge for protocol designers. More precisely, designers consider certain assumptions implicitly, but in reality these assumptions turn out to be false or cannot be enforced or verified. In this paper we exemplify exactly such a situation by presenting several subliminal channels with a small capacity in Zhang *et al.* and Dong *et al.*'s subliminal-free signature protocols.

1 Introduction

The notion of covert channels was introduced by Lampson in [8]. These channels have the capability of transporting information through system parameters apparently not intended for information transfer. In order to be efficient, covert channels should be hard to detect or control by the systems' security mechanisms.

The *prisoners' problem*, introduced by Simmons [12], captures the need of two parties to communicate secretly through normal-looking communication over an insecure channel. In the prisoners' problem *Alice* (sender) and *Bob* (receiver) are incarcerated and want 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 covert channels was introduced by Simmons [13,15–17] as a possible solution to the prisoners' problem. Subliminal channels achieve information transfer by modifying the original specifications of cryptographic primitives (for example, by modifying the way random numbers are generated). Hence, allowing *Alice* and *Bob* to communicate without being detected by *Walter*.

Within the scenario presented previously a natural question rises: how can one eliminate subliminal channels? To answer this question Simmons developed in [14] an interactive protocol between *Alice* and *Walter*. Other countermeasures against subliminal channels can be found in [1–3, 5, 6, 10, 11, 21]. Unfortunately, shortly after the publication of [14], Desmedt [4] found a flaw in the protocol. When running Simmons' protocol *Alice* can stop the protocol when certain conditions are not achieved. Thus, enabling her to subliminally send a bit. This method is called a *fail-stop* channel. To reduce the capacity of fail-stop channels, Simmons describes in [18] a cut-and-choose method. Note that fail-stop channels also exist in [6, 7][1] due to their similarity to [14].

Another problem with [14] was described by Simmons himself in [16]. Simmons suggests a method in which *Walter* can corrupt the protocol in such a way that he can subliminally communicate to a third party. Such channels are called *cuckoo's channels*.

In this paper we analyse the protocols presented in [5, 21]. We show that fail-stop channels exist, although the authors claim that the protocols are free of such channels. We also show that cuckoo's channels exist in both cases. Hence, we prove that their protocols are not subliminal-free. Due to their large communication overhead we suggest using other subliminal-free methods (for example, the methods proposed in [3, 6, 10, 11][2]).

Structure of the paper. We introduce notations and definitions in Section 2. In Section 3 we describe fail-stop and cuckoo's channels for the protocols proposed in [5, 21]. We conclude in Section 4.

2 Preliminaries

Notations. Throughout the paper λ and κ will denote security parameters. We denote by $x \| y$ the concatenation of the strings x and y. The set $\{0,1\}^*$ denotes the set of all bit strings and the cardinality of a set S is denoted by $|S|$,

The action of selecting a random element x from a sample space X is denoted by $x \xleftarrow{\$} X$. 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 random bit used only once.

2.1 Simmons' Signing Protocol

2.1.1 Description In [13, 15], Simmons introduced several subliminal channels that can be embedded into the DSA signature. These channels use as information carriers the ephemeral keys used by *Alice* in the signing process. A possible method for eliminating these channels was proposed by Simmons in [14]. He argued that covert communications can be stopped if the ephemeral keys where jointly generated by *Alice* and *Walter*. Thus, making the exact values indeterminate to both participants. Moreover, *Walter* must be able to check if *Alice* is honest.

We further describe the algorithms of Simmons' signing protocol. For simplicity, public parameters will further be considered implicit when describing an algorithm.

[1] In [6] a fail-stop channel is described, but it can be easily detected due to the protocol being implemented in devices with limited computational power.

[2] Note that in certain cases, hash channels [19, 20] create the capability of subliminal communication through these proposals.

Public Parameters' Generation(κ, λ): Select a prime number $q \geq 2^\kappa$ and a prime number $p \geq 2^\lambda$ such that $q | p-1$. Choose an element $g \in \mathbb{Z}_p$ of order q and a hash function $h : \{0,1\}^* \to \mathbb{Z}_q^*$. Output the public parameters $pp = (p, q, g, h)$.

Signer's Key Generation(pp): Choose $x \overset{\$}{\leftarrow} \mathbb{Z}_q^*$ and compute $y \leftarrow g^x \bmod p$. Output the public key $pk = y$. The secret key is $sk = x$.

Signing Protocol(m): To sign a message $m \in \{0,1\}^*$, the signer *Alice* and the warden *Walter* start the interactive protocol described in Figure 1. If the protocol succeeds, then *Walter* will relay (m, r, s) to *Bob*.

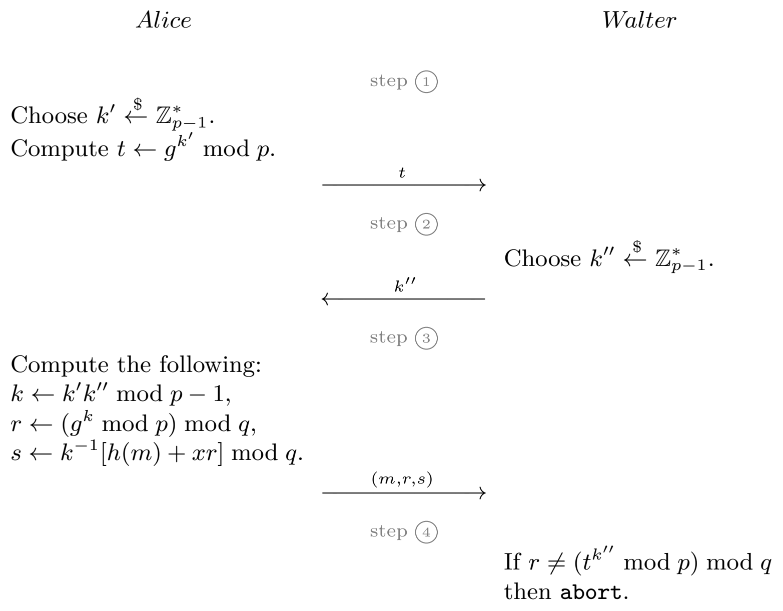
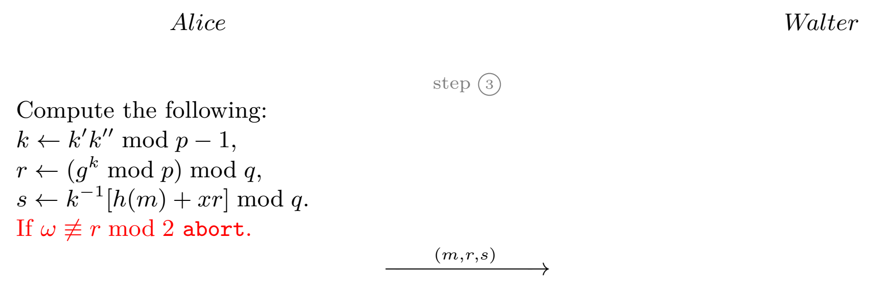

Alice
Walter
step ①
Choose $k' \overset{\$}{\leftarrow} \mathbb{Z}_{p-1}^*$.
Compute $t \leftarrow g^{k'} \bmod p$.
t
step ②
Choose $k'' \overset{\$}{\leftarrow} \mathbb{Z}_{p-1}^*$.
k''
step ③
Compute the following:
$k \leftarrow k'k'' \bmod p-1$,
$r \leftarrow (g^k \bmod p) \bmod q$,
$s \leftarrow k^{-1}[h(m) + xr] \bmod q$.
(m,r,s)
step ④
If $r \neq (t^{k''} \bmod p) \bmod q$
then `abort`.

Fig. 1. Simmons' Signing Protocol.

Verification(m, r, s, pk): To verify the signature (r, s) of message m, compute $u_1 \leftarrow h(m)s^{-1} \bmod q$ and $u_2 \leftarrow rs^{-1} \bmod q$. Then compute $v \leftarrow (g^{u_1} y^{u_2} \bmod p) \bmod q$ and output `true` if and only if $v = r$. Otherwise, output `false`.

2.1.2 Fail-Stop Channel Initially introduced in [4], this mechanism allows *Alice* to subliminally communicate with *Bob* even if *Walter* imposes a protocol like the one described in Figure 1. To communicate ω to *Bob*, *Alice* must stop the protocol if certain conditions are not achieved. If the protocol is stopped too often by *Alice*, *Walter* might become suspicious and cut off any communication between the prisoners. Thus, *Alice* can only send a few bits of data to *Bob* through this channel.

We further describe the fail-stop protocol in Figure 2 and the corresponding extraction algorithm (denoted by *Extract*). The changes made in the original protocol are marked with red in Figure 2.



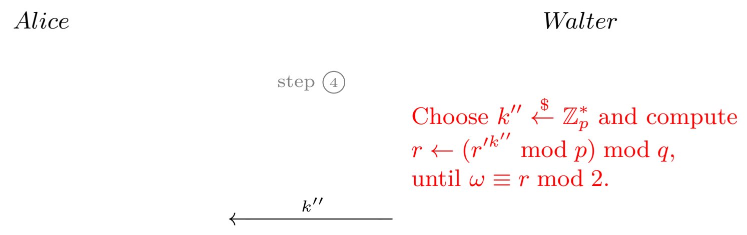
Compute the following:
$k \leftarrow k'k'' \bmod p-1$,
$r \leftarrow (g^k \bmod p) \bmod q$,
$s \leftarrow k^{-1}[h(m)+xr] \bmod q$.
If $\omega \not\equiv r \bmod 2$ `abort`.

Fig. 2. Desmedt's Fail-Stop Channel.

Extract(r) : To extract the embedded message ω compute $\omega \leftarrow r \bmod 2$.

2.1.3 Cuckoo's Channel In an article about protocol failures, Simmons describes a subliminal channel in his own protocol [16]. He called this type of channel the cuckoo's channel. Compared to fail-stop channels, cuckoo's channels are used by a dishonest *Walter* to convey information to a third party. Thus, just like a cuckoo that lays his eggs in the nests of unsuspecting birds, *Walter* inserts his message into *Alice*'s signature without her suspecting anything.

Let ω be the bit *Walter* subliminally embeds in Figure 1. We briefly describe the cuckoo's channel in Figure 3. As before, the changes made by *Walter* are written in red.



Choose $k'' \overset{\$}{\leftarrow} \mathbb{Z}_p^*$ and compute
$r \leftarrow (r'^{k''} \bmod p) \bmod q$,
until $\omega \equiv r \bmod 2$.

Fig. 3. Simmons' Cuckoo's Channel.

Extract(r) : To extract the embedded message ω compute $\omega \leftarrow r \bmod 2$.

To achieve indistinguishablility from Simmons' protocol, *Walter* must use sufficient parallel computing power. Thus, the more power *Walter* has, the longer the conveyed message can be. Let assume that for Simmons' protocol, *Walter* uses one computing unit CU. In the case of the cuckoo's protocol presented in Figure 3, if *Walter* uses α CU, then the probability of *Walter* transmitting his message undetected is $1 - 1/2^\alpha$. Hence, we can consider the cuckoo's channel as a noisy channel with an error probability of $1/2^\alpha$.

We further state without proof a security result from [16].

Lemma 1. *The cuckoo's channel presented in Figure 3 preserves the distribution of r.*

3 Novel Fail-Stop and Cuckoo's Channels

By using an interactive protocol between the signer and the warden, the authors of [5,21] try to eliminate existing subliminal channels from the Schnorr signature [21] and the ECDSA signature [5]. As we will later see, the protocols presented in [5,21] do not manage to completely eliminate covert channels, although the authors claim that they are subliminal-free.

3.1 Zhang *et al.*'s Signing Protocol

3.1.1 Description The first subliminal-free proposal that we describe was presented in [21]. According to the authors, the signer cannot control the outputs of the signature. Hence, the protocol is subliminal-free. We will see in the subsequent subsections that this is not true. Note that Zhang *et al.* assume that *Walter* is an *honest-but-curious*[3] warden that is disallowed to sign messages independently.

We further state Zhang *et al.*'s interactive protocol (Figure 4) and the associated algorithms, as presented in [21].

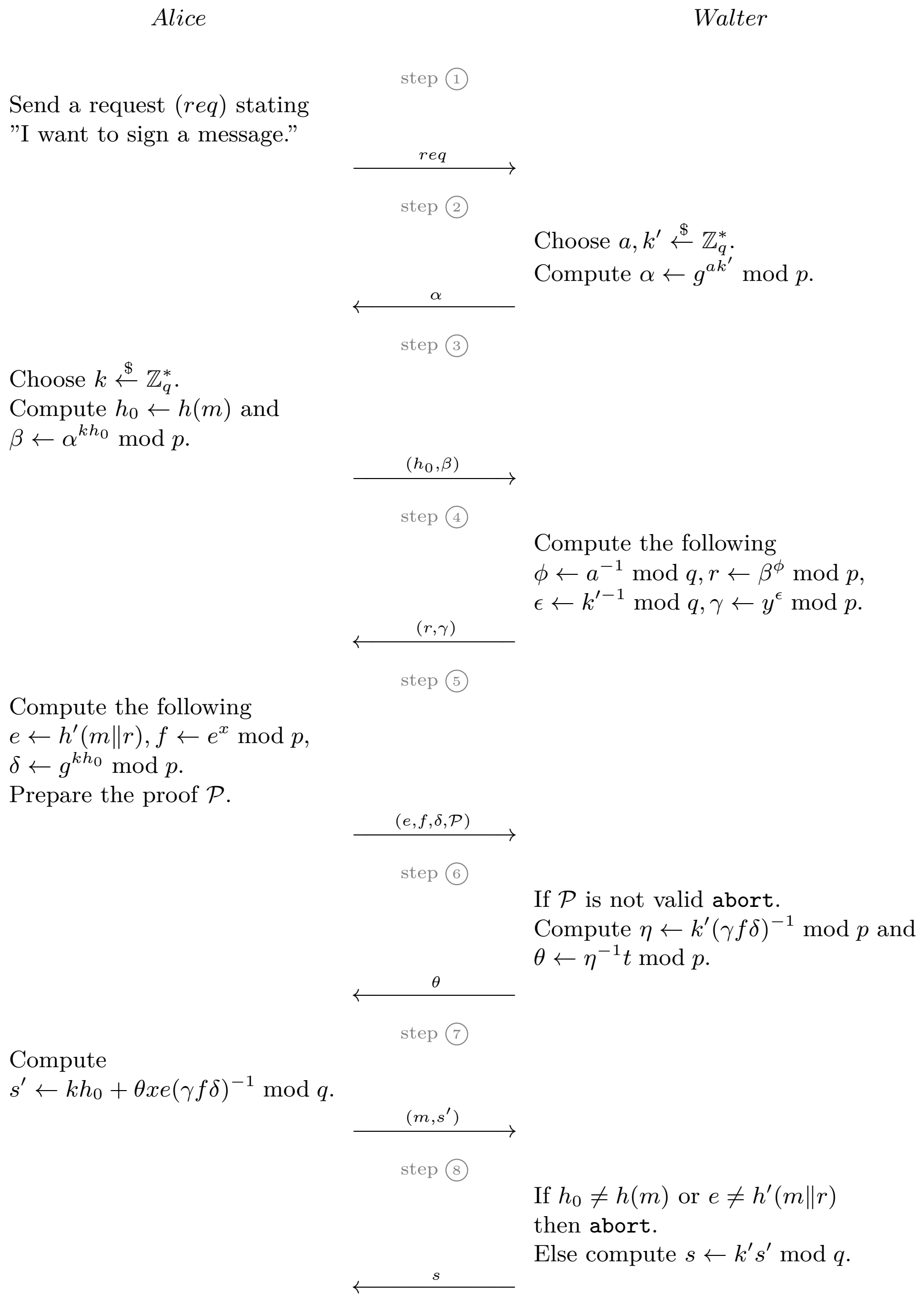
Public Parameters' Generation(κ, λ): Select a prime number $q \geq 2^\kappa$ and a prime number $p \geq 2^\lambda$ such that $q | p-1$. Choose an element $g \in \mathbb{Z}_p$ of order q and two hash functions $h : \{0,1\}^* \to \mathbb{G}$ and $h' : \{0,1\}^* \times \mathbb{G} \to \mathbb{Z}_q^*$. Output the public parameters $pp = (p, q, g, h, h')$.

Warden's Key Generation(pp): Choose $t \xleftarrow{\$} \mathbb{Z}_q^*$ and compute $z \leftarrow g^t$. Output the public key $pk_w = z$. The secret key is $sk_w = t$.

Signer's Key Generation(pk_w): Choose $x \xleftarrow{\$} \mathbb{Z}_q^*$ and compute $y \leftarrow z^x$. Output the public key $pk = y$. The secret key is $sk = x$.

Signing Protocol(m): To sign a message $m \in \{0,1\}^*$, the signer *Alice* and the warden *Walter* start the interactive protocol described in Figure 4. Note that in Step 5, Figure 4 *Alice* uses a non-interactive zero-knowledge proof $\mathcal{P}$ to convince W that $\log_e(f) = \log_z(y)$.

[3] According to [9,21], an *honest-but-curious* adversary is a legitimate participant in a communication protocol who will not deviate from the defined protocol but will attempt to learn all possible information from legitimately received messages.



Alice | *Walter*

step ①

Send a request (req) stating
"I want to sign a message."

$\xrightarrow{req}$

step ②

Choose $a, k' \overset{\$}{\leftarrow} \mathbb{Z}_q^*$.
Compute $\alpha \leftarrow g^{ak'} \bmod p$.

$\xleftarrow{\alpha}$

step ③

Choose $k \overset{\$}{\leftarrow} \mathbb{Z}_q^*$.
Compute $h_0 \leftarrow h(m)$ and
$\beta \leftarrow \alpha^{kh_0} \bmod p$.

$\xrightarrow{(h_0,\beta)}$

step ④

Compute the following
$\phi \leftarrow a^{-1} \bmod q, r \leftarrow \beta^{\phi} \bmod p$,
$\epsilon \leftarrow k'^{-1} \bmod q, \gamma \leftarrow y^{\epsilon} \bmod p$.

$\xleftarrow{(r,\gamma)}$

step ⑤

Compute the following
$e \leftarrow h'(m\|r), f \leftarrow e^x \bmod p$,
$\delta \leftarrow g^{kh_0} \bmod p$.
Prepare the proof $\mathcal{P}$.

$\xrightarrow{(e,f,\delta,\mathcal{P})}$

step ⑥

If $\mathcal{P}$ is not valid `abort`.
Compute $\eta \leftarrow k'(\gamma f \delta)^{-1} \bmod p$ and
$\theta \leftarrow \eta^{-1} t \bmod p$.

$\xleftarrow{\theta}$

step ⑦

Compute
$s' \leftarrow kh_0 + \theta x e(\gamma f \delta)^{-1} \bmod q$.

$\xrightarrow{(m,s')}$

step ⑧

If $h_0 \neq h(m)$ or $e \neq h'(m\|r)$
then `abort`.
Else compute $s \leftarrow k's' \bmod q$.

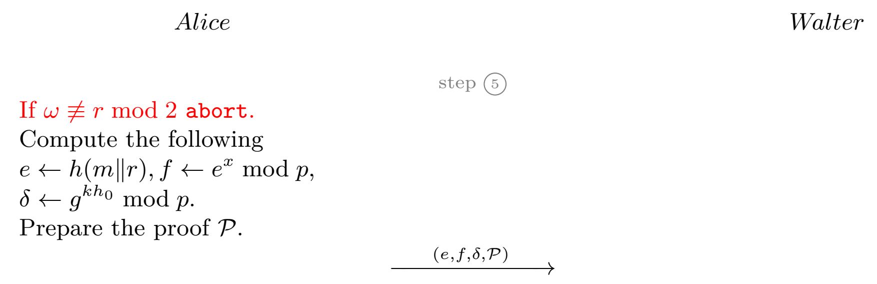
$\xleftarrow{s}$

Fig. 4. Zhang *et al.* Signing Protocol.

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

3.1.2 Fail-Stop Channel To bypass the protections set in place by Zhang *et al.* we use a fail-stop channel. Although *Alice* cannot control e, r and s, she can control if the protocol is successful or not. Hence, since the final value of r is not modified by *Walter* after Step 4, Figure 4 she can use it to carry out her message.

We further describe our proposed fail-stop protocol (Figure 5) and its corresponding extraction algorithm. The changes made in the original protocol are marked with red in Figure 5.



If $\omega \not\equiv r \bmod 2$ `abort`.
Compute the following
$e \leftarrow h(m\|r)$, $f \leftarrow e^x \bmod p$,
$\delta \leftarrow g^{kh_0} \bmod p$.
Prepare the proof $\mathcal{P}$.

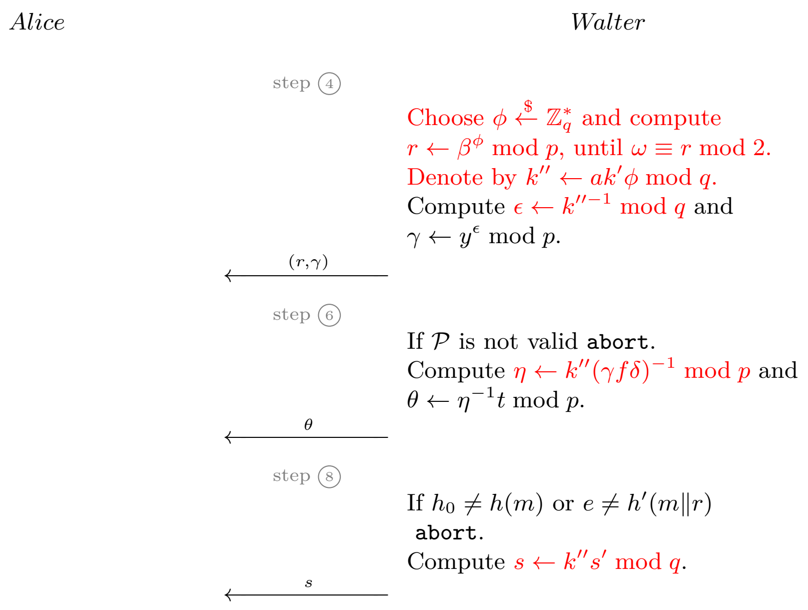
$\xrightarrow{(e,f,\delta,\mathcal{P})}$

Fig. 5. A Fail-Stop Channel Embedded into Zhang *et al.*'s protocol.

Extract(e, s, pk) : To extract the embedded message ω compute $r \leftarrow g^s y^{-e} \bmod p$ and $\omega \leftarrow r \bmod 2$.

3.1.3 Cuckoo's Channel According to [21], *Walter* will not deviate from the signing protocol. Thus, in Step 4, Figure 4 *Walter* has to supply *Alice* with (r, γ), θ and s of a given distribution. Keeping this restriction in mind, we have developed a cuckoo's channel in Zhang *et al.*'s protocol.

We briefly describe our proposed cuckoo's channel in Figure 6. As before, the changes made by *Walter* are written in red.



Choose $\phi \overset{\$}{\leftarrow} \mathbb{Z}_q^*$ and compute $r \leftarrow \beta^\phi \bmod p$, until $\omega \equiv r \bmod 2$.
Denote by $k'' \leftarrow ak'\phi \bmod q$.
Compute $\epsilon \leftarrow k''^{-1} \bmod q$ and $\gamma \leftarrow y^\epsilon \bmod p$.

If $\mathcal{P}$ is not valid `abort`.
Compute $\eta \leftarrow k''(\gamma f \delta)^{-1} \bmod p$ and $\theta \leftarrow \eta^{-1} t \bmod p$.

If $h_0 \neq h(m)$ or $e \neq h'(m\|r)$ `abort`.
Compute $s \leftarrow k'' s' \bmod q$.

Fig. 6. A Cuckoo's Channel Embedded into Zhang *et al.*'s protocol.

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

Correctness. The correctness of the *Verification* algorithm follows from the equality

$$\begin{aligned} s &\equiv k''s' \equiv k''[kh_0 + \theta x e(\gamma f \delta)^{-1}] \\ &\equiv kk''h_0 + k''(\eta^{-1}t)xe(\gamma f\delta)^{-1} \\ &\equiv kk''h_0 + k''[k''^{-1}(\gamma f \delta)t]xe(\gamma f \delta)^{-1} \\ &\equiv kk''h_0 + txe \bmod q, \end{aligned}$$

which leads to

$$r \equiv g^s y^{-e} \equiv g^{kk''h_0} g^{txe} y^{-e} \equiv g^{kk''h_0} \equiv g^{k(ak'\phi)h_0} \equiv \alpha^{k\phi h_0} \equiv \beta^\phi \bmod p.$$

The following lemma proves that no matter how much computing power *Alice* has, she will not be able to detect *Walter*'s cuckoo's channel and she will not be able to accuse *Walter* of being dishonest. Therefore, from the point of view of *Alice*, *Walter* is honest-but-curious, even though he is not.

Lemma 2. *The cuckoo's channel preserves the distributions of* (r, γ), θ *and* s.

Proof. In Zhang *et al.*'s protocol we have

$$r \equiv g^{kk'h_0} \bmod p,\ \epsilon \equiv k'^{-1} \bmod q,\ \eta \equiv k'(\gamma f \delta)^{-1} \bmod p \text{ and } s \equiv k's' \bmod q,$$

while in the cuckoo's version we have

$$r \equiv g^{kk''h_0} \bmod p,\ \epsilon \equiv k''^{-1} \bmod q,\ \eta \equiv k''(\gamma f \delta)^{-1} \bmod p \text{ and } s \equiv k''s' \bmod q.$$

Since $\phi \in \mathbb{Z}_q^*$ is chosen at random in the cuckoo's version, then $k'' \equiv ak'\phi \bmod q$ is also a random element from $\mathbb{Z}_q^*$. Therefore, k'' has the same distribution as k' value from Zhang *et al.*'s protocol. Thus, the distributions of (r, γ), θ and s are preserved. □

3.2 Dong *et al.*'s Signing Protocol

3.2.1 Description The authors of [5] use a similar approach to Zhang *et al.*'s for eliminating subliminal channels. Note that in this case, the authors do not impose that *Walter* is honest-but-curious. Fortunately for us, we were able to devise a fail-stop channel and a cuckoo's channel.

Before stating our results, we first describe Dong *et al.*'s protocol (Figure 7) and the associated algorithms, as presented in [5].

Public Parameters' Generation(λ): Select an elliptic curve $E(\mathbb{Z}_p)$ defined over $\mathbb{Z}_p$, where p is prime. Generate a prime number $q \geq 2^\lambda$, such that q divides $|E(\mathbb{Z}_p)|$. Generate a point $P \in E(\mathbb{Z}_p)$ of order q and select a hash function $h : \{0,1\}^* \to \mathbb{Z}_q^*$. Output the public parameters $pp = (q, P, E(\mathbb{Z}_p), h)$.

Signer's Key Generation(pp): Choose $d \overset{\$}{\leftarrow} \mathbb{Z}_q^*$ and compute $Q \leftarrow dP$. Output the public key $pk = Q$. The secret key is $sk = d$.

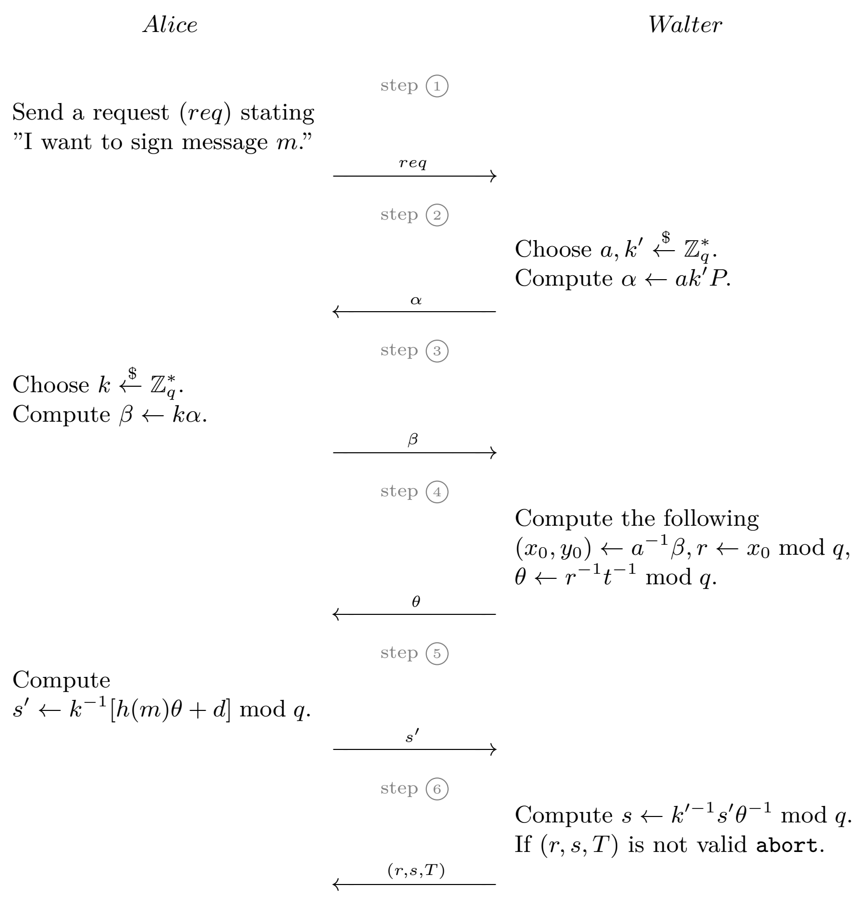
Warden's Key Generation(pk): Choose $t \overset{\$}{\leftarrow} \mathbb{Z}_q^*$ and compute $T \leftarrow tQ = (x_t, y_t)$. Let $h_t = h(x_t \| y_t)$. Output the public key $pk_w = h_t$. The secret key is $sk_w = t$.

Signing Protocol(m): To sign a message $m \in \{0,1\}^*$, the signer *Alice* and the warden *Walter* start the interactive protocol described in Figure 7. Note that in Step 6, Figure 7 *Walter* uses the *Verification* algorithm to check the validity of (r, s, T).

Verification(m, r, s, T, pk_w): To verify the signature (r, s, T) of message m, compute $u_1 \leftarrow h(m)s^{-1} \bmod q$, $u_2 \leftarrow rs^{-1} \bmod q$ and $h_t^* = h(x_t \| y_t)$. Then compute $u_1P + u_2T = (x_1, y_1)$ and $v \leftarrow x_1 \bmod q$. Output `true` if and only if $v = r$ and $h_t^* = h_t$. Otherwise, output `false`.

3.2.2 Fail-Stop Channel The authors of [5] claim that they eliminate fail-stop channels. Their main argument is that *Alice* does not know any information about (r, s) before *Walter* finishes the signature and thus she cannot use r as a carrier. Contrary to their statement, we managed to find such a channel.

We further describe our proposed channel (Figure 8) and its corresponding extraction algorithm. The changes made to the original protocol are written in red in Figure 8.



Alice *Walter*

step ①

Send a request (req) stating
"I want to sign message m."

$\xrightarrow{req}$

step ②

Choose $a, k' \overset{\$}{\leftarrow} \mathbb{Z}_q^*$.
Compute $\alpha \leftarrow ak'P$.

$\xleftarrow{\alpha}$

step ③

Choose $k \overset{\$}{\leftarrow} \mathbb{Z}_q^*$.
Compute $\beta \leftarrow k\alpha$.

$\xrightarrow{\beta}$

step ④

Compute the following
$(x_0, y_0) \leftarrow a^{-1}\beta, r \leftarrow x_0 \bmod q$,
$\theta \leftarrow r^{-1}t^{-1} \bmod q$.

$\xleftarrow{\theta}$

step ⑤

Compute
$s' \leftarrow k^{-1}[h(m)\theta + d] \bmod q$.

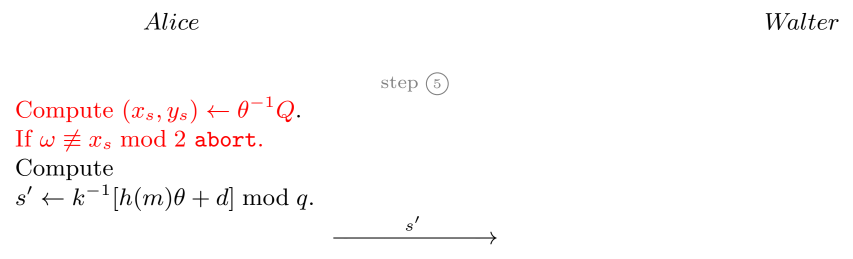
$\xrightarrow{s'}$

step ⑥

Compute $s \leftarrow k'^{-1}s'\theta^{-1} \bmod q$.
If (r, s, T) is not valid `abort`.

$\xleftarrow{(r,s,T)}$

Fig. 7. Dong *et al.* Signing Protocol.


Alice *Walter*

step 5

Compute $(x_s, y_s) \leftarrow \theta^{-1}Q$.
If $\omega \not\equiv x_s \bmod 2$ `abort`.
Compute
$s' \leftarrow k^{-1}[h(m)\theta + d] \bmod q$.

$\xrightarrow{s'}$

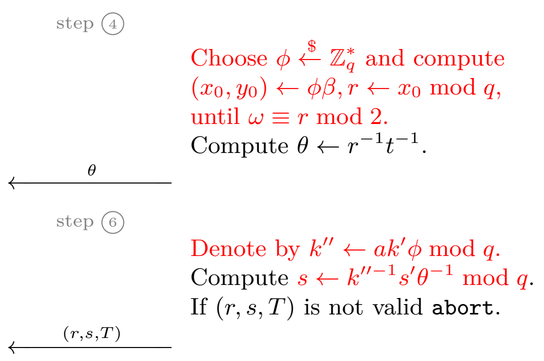
Fig. 8. A Fail-Stop Channel Embedded into Dong *et al.*'s protocol.

Extract(r, T) : To extract the embedded message ω compute $rT = (x_s, y_s)$ and $\omega \leftarrow x_s \bmod 2$.

Correctness. The correctness of the *Extract* algorithm follows from the following equality

$$\theta^{-1}Q = rtQ = rT.$$

3.2.3 Cuckoo's Channel Using a technique similar to the Zhang *et al.* cuckoo's channel, we further present in Figure 9 a cuckoo's channel that can be inserted into the Dong *et al.*'s protocol. As before, the changes made by *Walter* are written in red.


step 4

Choose $\phi \xleftarrow{\$} \mathbb{Z}_q^*$ and compute
$(x_0, y_0) \leftarrow \phi\beta, r \leftarrow x_0 \bmod q$,
until $\omega \equiv r \bmod 2$.
Compute $\theta \leftarrow r^{-1}t^{-1}$.

$\xleftarrow{\theta}$

step 6

Denote by $k'' \leftarrow ak'\phi \bmod q$.
Compute $s \leftarrow k''^{-1}s'\theta^{-1} \bmod q$.
If (r, s, T) is not valid `abort`.

$\xleftarrow{(r,s,T)}$

Fig. 9. A Cuckoo's Channel Embedded into Dong *et al.*'s protocol.

Extract(r) : To extract the embedded message compute $\omega \leftarrow r \bmod 2$.

Correctness. The check the correctness of the *Verification* algorithm we first compute

$$\begin{aligned} s &\equiv k''^{-1}s'\theta^{-1} \equiv k''^{-1}k^{-1}[h(m)\theta + d]\theta^{-1} \\ &\equiv k^{-1}k''^{-1}[h(m) + d\theta^{-1}] \\ &\equiv k^{-1}k''^{-1}[h(m) + drt] \bmod q, \end{aligned}$$

which leads to

$$\begin{aligned} u_1P + u_2T &= s^{-1}[h(m)P + rT] = s^{-1}[h(m) + rtd]P \\ &= kk''P = k(ak'\phi)P = k\phi\alpha = \phi\beta. \end{aligned}$$

In order to be secure, we need to prove that our proposal cannot be detected by *Alice* no matter how much computing power she has at her disposal. This is proven in the following lemma.

Lemma 3. *The cuckoo's channel preserves the distributions of θ and (r, s).*

Proof. In Dong *et al.*'s protocol we have

$$(x_0, y_0) \leftarrow kk'P, \quad \text{and} \quad s \leftarrow k'^{-1}s'\theta^{-1} \bmod q,$$

while in the cuckoo's version we have

$$(x_0, y_0) \leftarrow kk''P, \quad \text{and} \quad s \leftarrow k''^{-1}s'\theta^{-1} \bmod q.$$

Since $\phi \in \mathbb{Z}_q^*$ is chosen at random in the cuckoo's version, then $k'' \equiv ak'\phi \bmod q$ is also a random element from $\mathbb{Z}_q^*$. Therefore, k'' has the same distribution as k' value from Dong *et al.*'s protocol. Thus, the distributions of θ and (r, s) are preserved. □

4 Conclusions

Zhang *et al.* [21] and Dong *et al.* [5] propose two signature protocols that they claim to be subliminal-free. In this paper, we have proved that their claims are false. Since, the main utility of these protocols was to be subliminal-free and they failed to be so, we suggest that users employ other means of protection against subliminal channels with a lower communication overhead (*e.g.* the methods proposed in [3, 6, 10, 11]).

References

1. Giuseppe Ateniese, Bernardo Magri, and Daniele Venturi. Subversion-Resilient Signature Schemes. In *ACM-CCS 2015*, pages 364–375. ACM, 2015.
2. Jens-Matthias Bohli, Maria Isabel Gonzalez Vasco, and Rainer Steinwandt. A subliminal-free variant of ECDSA. In *IH 2006*, volume 4437 of *Lecture Notes in Computer Science*, pages 375–387. Springer, 2006.

3. Jong Youl Choi, Philippe Golle, and Markus Jakobsson. Tamper-Evident Digital Signature Protecting Certification Authorities Against Malware. In *DASC 2006*, pages 37–44. IEEE, 2006.
4. Yvo Desmedt. Simmons' protocol is not free of subliminal channels. In *Ninth IEEE Computer Security Foundations Workshop*, pages 170–175. IEEE, 1996.
5. Qingkuan Dong and Guozhen Xiao. A Subliminal-Free Variant of ECDSA Using Interactive Protocol. In *ICEEE 2010*, pages 1–3. IEEE, 2010.
6. Lucjan Hanzlik, Kamil Kluczniak, and Mirosław Kutyłowski. Controlled Randomness - A Defense against Backdoors in Cryptographic Devices. In *MyCrypt 2016*, volume 10311 of *Lecture Notes in Computer Science*, pages 215–232. Springer, 2016.
7. Patrick Horster, Markus Michels, and Holger Petersen. Subliminal Channels in Digital Logarithm Based Signature Schemes and How to Avoid Them. Technical Report TR-94-13, 1994.
8. Butler W Lampson. A Note on the Confinement Problem. *Communications of the ACM*, 16(10):613–615, 1973.
9. Andrew Paverd, Andrew Martin, and Ian Brown. Modelling and Automatically Analysing Privacy Properties for Honest-but-Curious Adversaries. Technical report, 2014.
10. Alexander Russell, Qiang Tang, Moti Yung, and Hong-Sheng Zhou. Cliptography: Clipping the power of kleptographic attacks. In *ASIACRYPT 2016*, volume 10032 of *Lecture Notes in Computer Science*, pages 34–64. Springer, 2016.
11. Alexander Russell, Qiang Tang, Moti Yung, and Hong-Sheng Zhou. Generic Semantic Security against a Kleptographic Adversary. In *ACM-CCS 2017*, pages 907–922. ACM, 2017.
12. Gustavus J Simmons. The Prisoners' Problem and the Subliminal Channel. In *CRYPTO 1983*, pages 51–67. Plenum Press, New York, 1983.
13. Gustavus J. Simmons. The Subliminal Channel and Digital Signatures. In *EUROCRYPT 1984*, volume 209 of *Lecture Notes in Computer Science*, pages 364–378. Springer, 1984.
14. Gustavus J Simmons. An Introductions to the Mathematics of Trust in Security Protocols. In *CSFW 1993*, pages 121–127. IEEE, 1993.
15. Gustavus J. Simmons. Subliminal Communication is Easy Using the DSA. In *EUROCRYPT 1993*, volume 765 of *Lecture Notes in Computer Science*, pages 218–232. Springer, 1993.
16. Gustavus J Simmons. Cryptanalysis and Protocol Failures. *Communications of the ACM*, 37(11):56–65, 1994.
17. Gustavus J Simmons. Subliminal Channels; Past and Present. *European Transactions on Telecommunications*, 5(4):459–474, 1994.
18. Gustavus J Simmons. Results concerning the bandwidth of subliminal channels. *IEEE Journal on Selected Areas in Communications*, 16(4):463–473, 1998.
19. George Teşeleanu. Subliminal Hash Channels. In *A2C 2019*, volume 1133 of *Communications in Computer and Information Science*, pages 149–165. Springer, 2019.
20. Chuan-Kun Wu. Hash channels. *Computers & Security*, 24(8):653–661, 2005.
21. Yinghui Zhang, Hui Li, Xiaoqing Li, and Hui Zhu. Provably Secure and Subliminal-Free Variant of Schnorr Signature. In *IICT-EurAsia 2013*, volume 7804 of *Lecture Notes in Computer Science*, pages 383–391. Springer, 2013.