Secretly Embedding Trapdoors into Contract Signing Protocols

Diana Maimuţ[1] and George Teşeleanu[1,2]

[1] Advanced Technologies Institute
10 Dinu Vintilă, Bucharest, Romania
ati@dcti.ro

[2] Department of Computer Science
"Al.I.Cuza" University of Iaşi 700506 Iaşi, Romania,
george.teseleanu@info.uaic.ro

Abstract. Contract signing protocols have been proposed and analyzed for more than three decades now. One of the main problems that appeared while studying such schemes is the impossibility of achieving both fairness and guaranteed output delivery. As workarounds, cryptographers have put forth three main categories of contract signing schemes: gradual release, optimistic and concurrent or legally fair schemes. Concurrent signature schemes or legally fair protocols do not rely on trusted arbitrators and, thus, may seem more attractive for users. Boosting user trust in such manner, an attacker may cleverly come up with specific applications. Thus, our work focuses on embedding trapdoors into contract signing protocols. In particular, we describe and analyze various SETUP (Secretly Embedded Trapdoor with Universal Protection) mechanisms which can be injected in concurrent signature schemes and legally fair protocols without keystones.

1 Introduction

Contract signing protocols have been proposed and extensively studied in the past. During the analysis of such schemes, the impossibility of achieving both fairness and guaranteed output delivery became a central problem for researchers. Trying to solve the aforementioned issue, cryptographers have developed various contract signing schemes which can be categorized having in mind three different design types: 1 *gradual release* [12, 14, 15, 18], 2 *optimistic* [2, 5, 17] and 3 *concurrent* [6] or *legally fair* [10] models. Concurrent signatures or legally fair protocols do not rely on trusted third parties. Also, concurrent signature models do not require too much interaction between users as compared to older paradigms like gradual release or optimistic models. Such features may seem much more attractive for users. Building upon user trust in the case of fair contract signing protocols, a (powerful) adversary may cleverly construct attack scenarios.

Digital signature schemes naturally arose as the central ingredient of modern contract signing protocols. The use of digital signatures as a channel to convey information (subliminal channel) was first introduced and studied by Simmons [21, 22]. Another step was taken by Young and Yung [23–27], who combined subliminal channels and public key cryptography in order to leak a user's private key (SETUP attacks). The two authors work in a black-box environment[3], pointing out that other scenarios exist. Such attacks may be considered if the manufacturer of the device is an accomplice, in the sense that he implements the mechanisms to recover the keys.

A SETUP attack of the previously mentioned form is likely to be applied in the case of auctions. To provide the reader with a possible scenario, we further assume that participants receive signing tokens from an auctioneer and they do not communicate using additional channels. The participants' bids are acknowledged by the auctioneer's co-signature. In this context, the auctioneer is able to leak lists containing fake bids for the competing participants. The value of the bids is, thus, maliciously raised.

Our work focuses on embedding trapdoors into contract signing protocols. In particular, we describe and analyze two main SETUP mechanisms which can be injected in the concurrent signature scheme presented in [6] and the legally fair protocol (without keystones) introduced in [10].

[3] *e.g.* tamper proof devices

Structure of the Paper. We introduce notations, definitions and protocols used throughout the paper in Section 2. In Sections 3 and 4 we present two main SETUP mechanisms which can be injected into concurrent or legally fair signature schemes and analyze their security in the standard model and, respectively, Random Oracle Model (ROM) [3]. We conclude in Section 5. We recall additional security models and Schnorr signatures in Appendix A and provide supplementary SETUP mechanisms in Appendices B and C.

2 Preliminaries

Notations. Let S be a finite set. We denote by $x \stackrel{\$}{\leftarrow} S$ the operation of picking an element uniformly from S.

$x||y$ represents the string obtained by concatenating y to x.

If and only if is further referred to as *iff*.

Unless otherwise specified, $\mathbb{G}$ is a cyclic group of order q, where q is a large prime number. Also, we denote by g a generator of $\mathbb{G}$.

x_i and y_i represent the private and public keys associated with user i: x_i is considered to be randomly chosen from $\mathbb{Z}_q^*$ and $y_i = g^{x_i}$.

The action of choosing a random element from an entropy smoothing[4] (ES) family $\mathcal{H}$ is further referred to as "H is ES".

We denote by PPT algorithm a probabilistic polynomial-time algorithm.

2.1 Security Assumptions

Definition 1 (Discrete Logarithm Problem - DLP). *Let $\mathbb{G}$ be a cyclic group of order n and g a generator $\mathbb{G}$. Given $g, h \stackrel{\$}{\leftarrow} \mathbb{G}$, find a such that $h = g^a$.*

The number a is called the discrete logarithm of h to the base g and is denoted by $\log_g h$.

Remark 1. Two users A and B can choose a DLP based protocol in order to compute a common secret key K. We further describe the Diffie-Hellman (DH) key exchange [7].

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

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

Definition 2 (Computational Diffie-Hellman - CDH and List Computational Diffie-Hellman of Order 2 - LCDH2). *Let $\mathbb{G}$ be a cyclic group of order n, g a generator $\mathbb{G}$ and let A be a PPT algorithm that returns either an element (CDH) or a list of elements (LCDH2) from $\mathbb{G}$. We define the advantages*

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

$$ADV^{\text{LCDH2}}_{\mathbb{G},g}(A) = Pr[g^{xy} \text{ or } g^{xz} \in A(g^x, g^y, g^z) | x, y, z \stackrel{\$}{\leftarrow} \mathbb{Z}_n^*].$$

If $ADV^{\text{CDH}}_{\mathbb{G},g}(A)$ or $ADV^{\text{LCDH2}}_{\mathbb{G},g}(A)$ is negligible for any PPT algorithm A, we say that the Computational Diffie-Hellman problem or List Computational Diffie-Hellman problem of Order 2 is hard in $\mathbb{G}$.

[4] We refer the reader to Appendix A for a definition of the concept.

Remark 2. A similar with LCDH2 concept was introduced in [20] and proven to be equivalent with CDH. Tweaking the proof from [20], we obtain that for an efficient PPT LCDH2 adversary A there exist an efficient PPT algorithm B such that

$$ADV^{\text{LCDH2}}_{\mathbb{G},g}(A) \le 2ADV^{\text{CDH}}_{\mathbb{G},g}(B). \tag{1}$$

It is easy to see that if the CDH assumption doesn't hold, then the LCDH2 assumption doesn't hold. If the LCDH2 assumption doesn't hold, then there exist a PPT algorithm A that has non-negligible LCDH2 advantage. We will use A to build an algorithm B that has non-negligible CDH advantage for (g^x, g^y) or (g^x, g^z). Algorithm B simply runs A and then outputs two random elements from the list returned by A. Thus we obtain the loose reduction (1).

Definition 3 (Decisional Diffie-Hellman - DDH). *Let $\mathbb{G}$ be a cyclic group of order n, g a generator $\mathbb{G}$ and let A be a PPT algorithm. We define the advantage*

$$ADV^{DDH}_{\mathbb{G},g}(A) = \left| Pr[A(g^x, g^y, g^z) = 1 | x, y \xleftarrow{\$} \mathbb{Z}_n^*, z \leftarrow xy] - Pr[A(g^x, g^y, g^z) = 1 | x, y, z \xleftarrow{\$} \mathbb{Z}_n^*] \right|.$$

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

2.2 Security Models

Definition 4 (Pseudorandom Function - PRF). *A function $F : \{0,1\}^n \times \{0,1\}^s \to \{0,1\}^m$ is a (t,q)-PRF if:*

- *Given a key $K \in \{0,1\}^s$ and an input $X \in \{0,1\}^n$ there is an efficient algorithm to compute $F_K(X) = F(X,K)$.*
- *For any t-time oracle algorithm A, the PRF-advantage of A, defined as*

$$ADV^{PRF}_{F}(A) = \left| Pr[A^{F_K(\cdot)} = 1 | K \xleftarrow{\$} \{0,1\}^s] - Pr[A^{F(\cdot)} = 1 | F \xleftarrow{\$} \mathcal{F}] \right|$$

  is negligible for any PPT algorithm A, where $\mathcal{F} = \{F : \{0,1\}^n \to \{0,1\}^m\}$ and A makes at most q queries to the oracle.

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. The leakage is achieved through a public key exchange protocol between an unsuspecting victim and the attacker.*

Definition 6 (SETUP indistinguishability - IND-SETUP). *Let C_0 be a black-box system that uses a pair of keys (pk, sk), where pk is the public key and sk the corresponding secret key. Let pk_S be the public key of an attacker as defined in Definition 5. Let $\mathcal{KE}$ be a public key exchange protocol that takes as input pk and pk_S. We consider C_1 an altered version of C_0 that contains a SETUP mechanism based on $\mathcal{KE}$. Let A be a PPT algorithm. We define the advantage*

$$ADV^{IND\text{-}SETUP}_{\mathcal{KE},C_0,C_1}(A) = \left| Pr[A^{C_1(sk,\cdot)}(pk, pk_S) = 1] - Pr[A^{C_0(sk,\cdot)}(pk, pk_S) = 1] \right|.$$

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

2.3 Concurrent Signatures

In the case of classical contract signing protocols, users exchange complete signatures (*e.g.* [13]). Concurrent signature protocols [6, 16] use "ambiguous" signatures that do not bind their author. An additional piece of information called the *keystone* can later be used to lift the ambiguity. Thus, when the keystone is revealed, signatures become simultaneously binding.

The standard algorithms corresponding to a concurrent signature are shortly described in Table 1.

Table 1. The algorithms of a concurrent signature.

$\mathsf{Setup}(\ell)$	On input a security parameter ℓ, this algorithm outputs the private and public keys (x_i, y_i) of all participants and the public parameters $\mathsf{pp} = (\mathcal{M}, \mathcal{S}, \mathcal{K}, \mathcal{F}, \mathsf{KeyGen})$, where $\mathcal{M}$ is the message space, $\mathcal{S}$ is the signature space, $\mathcal{K}$ is the keystone space and $\mathsf{KeyGen} : \mathcal{K} \to \mathcal{F}$ is a selected function.
$\mathsf{aSign}(y_i, y_j, x_i, e_2, m)$	On input the public keys y_i and y_j, the private key x_i corresponding to y_i, an element $e_2 \in \mathcal{F}$ and a message $m \in \mathcal{M}$, this algorithm outputs an "ambiguous signature" $\sigma = \langle s, e_1, e_2 \rangle$, where $s \in \mathcal{S}$ and $e_1, e_2 \in \mathcal{F}$.
$\mathsf{aVerify}(\sigma, y_i, y_j, m)$	On input an ambiguous signature $\sigma = \langle s, e_1, e_2 \rangle$, public keys y_i, y_j and a message m this algorithm outputs a boolean value satisfying $$\mathsf{aVerify}\left(\sigma', y_j, y_i, m\right) = \mathsf{aVerify}\left(\sigma, y_i, y_j, m\right),$$ where $\sigma' = \langle s, e_2, e_1 \rangle$.
$\mathsf{Verify}(k, \sigma, y_i, y_j, m)$	On input $k \in \mathcal{K}$, $\sigma = \langle s, e_2, e_1 \rangle$, public keys y_i, y_j and message m, this algorithm checks whether $\mathsf{KeyGen}(k) = e_2$ and outputs False if not; otherwise it outputs the result of $\mathsf{aVerify}(\sigma, y_i, y_j, m)$.

Concurrent signatures are used by two parties *Alice* and *Bob* as depicted in Figure 2.

Alice		*Bob*
		$k \xleftarrow{\$} \mathcal{K}$ $f \leftarrow \mathsf{KeyGen}(k)$ $\sigma_B \leftarrow \mathsf{aSign}(y_B, y_A, x_B, f, m_B)$
	$\xleftarrow{\sigma_B}$	
$T_A \leftarrow \mathsf{aVerify}(\sigma_B, y_B, y_A, m_B)$ **if** $T_A = \mathsf{False}$ **then** abort $\sigma_A \leftarrow \mathsf{aSign}(y_A, y_B, x_A, f, m_A)$		
	$\xrightarrow{\sigma_A}$	
		$T_B \leftarrow \mathsf{aVerify}(\sigma_A, y_A, y_B, m_A)$ **if** $T_B = \mathsf{False}$ **then** abort
	$\xleftarrow{k}$	

Fig. 2. The concurrent signature of messages m_A and m_B.

At the end of this protocol, both $\langle k, \sigma_A \rangle$ and $\langle k, \sigma_B \rangle$ are binding, and accepted by the Verify algorithm.

A Concrete Construction. To mount our SETUP attacks, we further use a concrete concurrent signature, more precisely the protocol presented in [6]. The security of this protocol can be proven in the ROM, assuming the hardness of computing discrete logarithms in a group $\mathbb{G}$.

Chen *et. al*'s concurrent scheme is presented in Figure 3. The scheme makes use of two cryptographic hash functions $H_1, H_2 : \{0,1\}^* \to \mathbb{Z}_q^*$.

Alice		*Bob*
		$k \xleftarrow{\$} \{0,1\}^*$ $f \leftarrow H_1(k)$ $\delta_B \xleftarrow{\$} \mathbb{Z}_q^*$ $\eta_B \leftarrow H_2\left(g^{\delta_B} y_A^f \| m_B\right)$ $e_B \leftarrow \eta_B - f \bmod q$ $s_B \leftarrow \delta_B - e_B x_B \bmod q$ $\sigma_B \leftarrow \langle s_B, e_B, f \rangle$
	$\xleftarrow{\quad \sigma_B \quad}$	
$T_A \leftarrow H_2\left(g^{s_B} y_B^{e_B} y_A^f \| m_B\right) \bmod q$ **if** $T_A \not\equiv e_B + f \bmod q$ **then** abort $\delta_A \xleftarrow{\$} \mathbb{Z}_q^*$ $\eta_A \leftarrow H_2\left(g^{\delta_A} y_B^f \| m_A\right)$ $e_A \leftarrow \eta_A - f \bmod q$ $s_A \leftarrow \delta_A - e_A x_A \bmod q$ $\sigma_A \leftarrow \langle s_A, e_A, f \rangle$		
	$\xrightarrow{\quad \sigma_A \quad}$	
		$T_B \leftarrow H_2\left(g^{s_A} y_A^{e_A} y_B^f \| m_A\right) \bmod q$ **if** $T_B \not\equiv e_A + f \bmod q$ **then** abort
	$\xleftarrow{\quad k \quad}$	

Fig. 3. Chen *et al.* concurrent signature.

2.4 Legally Fair Signatures without Keystones

In [10] the authors present a new contract signing paradigm that does not require keystones to achieve legal fairness. Their provably secure co-signature construction recalled in Figure 4 is based on Schnorr digital signatures[5].

In Figure 4, $\mathcal{L}$ represents a local non-volatile memory used by Bob and $H_1 : \{0,1\}^* \to \mathbb{Z}_q^*$ denotes a cryptographic hash functions. During the protocol, Alice makes use of a publicly known auxiliary signature scheme σ that uses her secret key x_A.

[5] recalled in Appendix A

Alice		*Bob*
$y_{A,B} \leftarrow y_A \cdot y_B$		$y_{A,B} \leftarrow y_A \cdot y_B$
$\delta_A \xleftarrow{\$} \mathbb{Z}_q^*$		$\delta_B \xleftarrow{\$} \mathbb{Z}_q^*$
$r_A \leftarrow g^{\delta_A}$		$r_B \leftarrow g^{\delta_B}$
		$\rho \leftarrow H_1(0 \Vert r_B)$
	$\xleftarrow{\quad \rho \quad}$	
$t \leftarrow \sigma(r_A \Vert \text{Alice} \Vert \text{Bob})$	$\xrightarrow{\quad r_A, t \quad}$	
		if t is incorrect **then** abort
		store t in $\mathcal{L}$
	$\xleftarrow{\quad r_B \quad}$	
if $H_1(0 \Vert r_B) \neq \rho$ **then** abort		
$r \leftarrow r_A \cdot r_B$		$r \leftarrow r_A \cdot r_B$
$e \leftarrow H_1(1 \Vert m \Vert r \Vert \text{Alice} \Vert \text{Bob})$		$e \leftarrow H_1(1 \Vert m \Vert r \Vert \text{Alice} \Vert \text{Bob})$
$s_A \leftarrow \delta_A - e x_A \bmod q$		$s_B \leftarrow \delta_B - e x_B \bmod q$
		store s_B in $\mathcal{L}$
	$\xleftarrow{\quad s_B \quad}$	
if s_B is incorrect **then** abort		
	$\xrightarrow{\quad s_A \quad}$	
		if s_A is incorrect **then** abort
$s \leftarrow s_A + s_B \bmod q$		$s \leftarrow s_A + s_B \bmod q$
		if $\{m, r, s\}$ is valid **then**
		erase t, s_B from $\mathcal{L}$

Fig. 4. The legally fair signature (without keystones) of message m.

3 SETUP Attacks on Concurrent Signatures

We present a SETUP mechanism[6] which can later be used by an external attacker Eve to recover either Alice's or Bob's secret key. To implement her attack, Eve needs a valid pair of (private and public) keys $(x_E, y_E = g^{x_E})$. The public key y_E is stored in a volatile memory on the victim's device. We further assume that Eve has access to the data transmitted during the protocol.

Changes required by the SETUP mechanisms will further be underlined using red colored text within Protocol 5.

Description. The SETUP mechanism requires:

- a pseudorandom function $\mathbb{PRF} : K \times \mathbb{Z}_q^* \to \mathbb{Z}_q^*$, where K is the key space;
- a function $H : \mathbb{G} \to K$;
- a protocol needs to reach breakpoint 1 for an attacker to recover Bob's secret key;
- a protocol needs to reach breakpoint 2 for an attacker to recover both secret key.

The value f is transmitted during the protocol and is available to Eve. Hence, she can recover user i's secret key simply by computing $\mathbb{PRF}(H(y_i^{x_E}), f)$ and extracting x_i from s_i by calculating $e_i^{-1}(\delta_i - s_i)$, where i denotes either Alice or Bob.

Compared to the mechanism presented in Appendix B, this SETUP attack requires only one successful protocol to recover Alice's and Bob's secret key.

Malicious Co-Signers. If Eve is replaced by Alice, a protocol needs to reach breakpoint 1. When replaced by Bob, a protocol needs to reach breakpoint 2.

Security Analysis. We present the main security results, more precisely Theorems 1 and 2, and provide the reader with the necessary proofs.

When referring to the security analysis presented in the current section, Θ is considered an additional security parameter and refers to the maximal number of protocol iterations.

Theorem 1. *If* DDH *is hard in* $\mathbb{G}$ *and* H *is a one-to-one function*[7]*, then the protocols presented in Figure 3 and Figure 5 are* IND-SETUP *in the standard model. Formally, let* A *be an efficient PPT* IND-SETUP *adversary. There exist two efficient PPT algorithms* B_1, B_2 *such that*

$$ADV^{IND\text{-}SETUP}_{DH,P_3,P_5}(A) \leq 4ADV^{DDH}_{\mathbb{G},g}(B_1) + 4ADV^{PRF}_{\mathbb{PRF}}(B_2).$$

Proof. We denote the protocols presented in Figure 3 and Figure 5 by P_3 and P_5. Let A be an IND-SETUP adversary trying to distinguish between P_3 and P_5. We show that A's advantage is negligible. We construct the proof as a sequence of games in which all the required changes are applied to P_5. Let W_i be the event that A wins game i.

Game 0. The first game is identical to the IND-SETUP game[8]. Thus, we have

$$|2Pr[W_0] - 1| = ADV^{\text{IND-SETUP}}_{\text{DH},P_3,P_5}(A). \quad (2)$$

Game 1. In this game, $y_E^{x_A}$ and $y_E^{x_B}$ from *Game 0* become g^{z_A} and g^{z_B}, where $z_A, z_B \xleftarrow{\$} \mathbb{Z}_q$. Since this is the only change between *Game 0* and *Game 1*, A will not notice the difference assuming the DDH assumption holds. Formally, this means that there exists an algorithm B_1 such that

$$|Pr[W_0] - Pr[W_1]| = 2ADV^{\text{DDH}}_{\mathbb{G},g}(B_1). \quad (3)$$

[6] Another mechanism (detailed in Appendix B) naturally arises.

[7] A function for which every element of the range of the function corresponds to precisely one element of the domain.

[8] as in Definition 6

$$
\begin{array}{lcl}
\textit{Alice} & & \textit{Bob} \\
 & & k \xleftarrow{\$} \{0,1\}^* \\
 & & f \leftarrow H_1(k) \\
 & & K_1 \leftarrow H(y_E^{x_B}) \\
 & & \delta_B \leftarrow \mathbb{PRF}(K_1, f) \\
 & & \eta_B \leftarrow H_2\left(g^{\delta_B} y_A^f \| m_B\right) \\
 & & e_B \leftarrow \eta_B - f \bmod q \\
 & & s_B \leftarrow \delta_B - e_B x_B \bmod q \\
 & & \sigma_B \leftarrow \langle s_B, e_B, f \rangle \\
 & \text{breakpoint } 1 & \\
 & \xleftarrow{\quad \sigma_B \quad} & \\
T_A \leftarrow H_2\left(g^{s_B} y_B^{e_B} y_A^f \| m_B\right) \bmod q & & \\
\textbf{if } T_A \not\equiv e_B + f \bmod q \textbf{ then } \mathsf{abort} & & \\
K_2 \leftarrow H(y_E^{x_A}) & & \\
\delta_A \leftarrow \mathbb{PRF}(K_2, f) & & \\
\eta_A \leftarrow H_2\left(g^{\delta_A} y_B^f \| m_A\right) & & \\
e_A \leftarrow \eta_A - f \bmod q & & \\
s_A \leftarrow \delta_A - e_A x_A \bmod q & & \\
\sigma_A \leftarrow \langle s_A, e_A, f \rangle & & \\
 & \text{breakpoint } 2 & \\
 & \xrightarrow{\quad \sigma_A \quad} & \\
 & & T_B \leftarrow H_2\left(g^{s_A} y_A^{e_A} y_B^f \| m_A\right) \bmod q \\
 & & \textbf{if } T_B \not\equiv e_A + f \bmod q \textbf{ then } \mathsf{abort} \\
 & \xleftarrow{\quad k \quad} &
\end{array}
$$

Fig. 5. Protocol 3 with a SETUP mechanism.

Game 2. Since H is one-to-one then we can make the change $K_1, K_2 \xleftarrow{\$} \mathbb{Z}_q$ and adversary A will not notice. Formally, this means that

$$Pr[W_1] = Pr[W_2]. \tag{4}$$

Game 3. The last change we make is $\delta_A, \delta_B \xleftarrow{\$} \mathbb{Z}_q$. Adversary A will not notice the difference, since $\mathbb{PRF}$ is a pseudorandom function. Formally, there exist an algorithms B_2 such that

$$|Pr[W_2] - Pr[W_3]| = 2ADV^{\text{PRF}}_{\mathbb{PRF}}(B_2). \tag{5}$$

The changes made to P_5 in *Game 1* - *Game 3*, transformed it into P_3. Thus, we have

$$Pr[W_3] = 1/2. \tag{6}$$

Finally, the statement is proven by combining the equalities (2) – (6). □

Remark 3. From Theorem 1, the maximum advantage an IND-SETUP adversary can obtain in the standard model is

$$ADV^{\text{IND-SETUP}}_{\text{DH},P_3,P_5}(A) \leq 4\Theta ADV^{\text{DDH}}_{\mathbb{G},g}(B_1) + 4\Theta ADV^{\text{PRF}}_{\mathbb{PRF}}(B_2).$$

The advantage remains negligible if parameter Θ is polynomial.

Theorem 2. *If* CDH *is hard in* $\mathbb{G}$ *and* H *is a hash function, then the protocols presented in Figure 3 and Figure 5 are* IND-SETUP *in the ROM. Formally, let* A *be an efficient PPT* IND-SETUP *adversary. There exist two efficient PPT algorithms* B_1, B_2 *such that*

$$ADV^{\text{IND-SETUP}}_{\text{DH},P_3,P_5}(A) \leq 4ADV^{\text{CDH}}_{\mathbb{G},g}(B_1) + 4ADV^{\text{PRF}}_{\mathbb{PRF}}(B_2).$$

Proof. We will use the same notations as in the proof for Theorem 1.

Game 0. The first game is identical to the IND-SETUP game[9]. Thus, we have

$$|2Pr[W_0] - 1| = ADV^{\text{IND-SETUP}}_{\text{DH},P_3,P_5}(A). \tag{7}$$

The challenger picks a random oracle $H : \mathbb{G} \to \mathbb{Z}_q^*$ at random from the set of all such functions. A can make a sequence of queries of the following type.

Hash oracle query[10]: A presents the challenger with $m \in \mathbb{G}$, who responds with $H(m)$.

Game 1. At the beginning of the game choose $K_1, K_2 \xleftarrow{\$} \mathbb{Z}_q^*$. The challenger's way to respond to queries becomes:

Hash oracle query[11]: A presents the challenger with $m \in \mathbb{G}$. The challenger responds with

- K_1, if $m = y_E^{x_A}$;
- K_2, if $m = y_E^{x_B}$;
- $H(m)$, otherwise.

Since we have replaced the values $y_E^{x_A}$ and $y_E^{x_B}$ throughout the game, we have

$$Pr[W_0] = Pr[W_1]. \tag{8}$$

Game 2. In this game, we revert to the original hash oracle query (*i.e* the challenger responds with $H(m)$ for all m). Let F be the event that the adversary makes a query with $m \leftarrow y_E^{x_A}$ or $m \leftarrow y_E^{x_B}$. *Game 1* and *Game 2* are identical until F occurs. Thus, we have

$$|Pr[W_1] - Pr[W_2]| \leq Pr[F]. \tag{9}$$

We need to prove that

$$Pr[F] = ADV^{\text{LCDH2}}_{\mathbb{G},g}(C), \tag{10}$$

where C is an algorithm that takes as input y_E, y_A and y_B. C will play the role of the challenger in *Game 2*. Algorithm C has a list of queries and responses, such that if A makes a query that matches one of the previous queries, C can return the previous output. At the end of the game, algorithm C will output a list with all the responses to A's queries. It is easy to see that the probability of C returning a list containing $y_E^{x_A}$ or $y_E^{x_B}$ is the same as $Pr[F]$.

Game 3. In this game we choose $\delta_A, \delta_B \xleftarrow{\$} \mathbb{Z}_q$. Adversary A will not notice the difference, since $\mathbb{PRF}$ is a pseudorandom function. Formally, there exist an algorithm B_2 such that

$$|Pr[W_2] - Pr[W_3]| = 2ADV^{\text{PRF}}_{\mathbb{PRF}}(B_2). \tag{11}$$

The changes made to P_5 in *Game 1* - *Game 3*, transformed it into P_3. Thus, we have

$$Pr[W_3] = 1/2. \tag{12}$$

Finally, the statement is proven by combining the equalities (7) – (12). □

Remark 4. From Theorem 6, the maximum advantage an IND-SETUP adversary can obtain in the ROM is

$$ADV^{\text{IND-SETUP}}_{\text{DH},P_3,P_5}(A) \leq 4\Theta ADV^{\text{CDH}}_{\mathbb{G},g}(B_1) + 4\Theta ADV^{\text{PRF}}_{\mathbb{PRF}}(B_2).$$

The advantage remains negligible if parameter Θ is polynomial.

[9] as in Definition 6

[10] *Game 0*

[11] *Game 1*

4 SETUP Attacks on Legally Fair Signatures without Keystones

To implement her attack[12], Eve works in the same environment described in Section 3.

As in Section 3, changes required by the SETUP mechanisms will further be underlined using red colored text in Protocol 6.

Description. The SETUP mechanism requires:

- a pseudorandom function $\mathbb{PRF} : K \times \mathbb{Z}_q^* \to \mathbb{Z}_q^*$, where K is the key space;
- a function $H : \mathbb{G} \to K$;
- a protocol needs to reach breakpoint 1 for an attacker to recover Bob's secret key;
- a protocol needs to reach breakpoint 2 for an attacker to recover both secret key.

By j_B we understand a counter incremented each time Bob runs the protocol.

The value ρ is transmitted during the protocol and is available to Eve. Hence, she can recover Alice's secret key simply by computing $\mathbb{PRF}(H(y_A^{x_E}), \rho)$ and extracting x_A from s_A by calculating $e^{-1}(\delta_A - s_A)$.

To find the value of j_B, Eve computes $\delta_{B,\ell} \leftarrow \mathbb{PRF}(H(y_B^{x_E}), \ell)$, $r_\ell = g^{\delta_{B,\ell}}$ and $\ell = \ell + 1$, until $r_\ell = r_B$. Once j_B is found, she can compute the secret key as $e^{-1}(\delta_{B,\ell} - s_B)$.

Compared to the mechanism presented in Appendix C, this SETUP attack requires only one successful protocol to recover Alice's secret key. Attacking Bob instead of Alice is less efficient, since Eve must find the current counter value.

Malicious Co-Signers. If Eve is replaced by Alice, a protocol needs to reach breakpoint 1. Also, Alice may choose to infect the protocols which directly involve her, keep an internal counter, and, thus, avoid the need to use brute-force methods to obtain j_B. When replaced by Bob, a protocol needs to reach breakpoint 2.

[12] Another attack (detailed in Appendix C) naturally arises.

Alice		*Bob*
$y_{A,B} \leftarrow y_A \cdot y_B$		$y_{A,B} \leftarrow y_A \cdot y_B$ $\delta_B \leftarrow \mathbb{PRF}(H(y_E^{x_B}), j_B)$ $r_B \leftarrow g^{\delta_B}$ $\rho \leftarrow H_1(0 \| r_B)$
	$\xleftarrow{\rho}$	
$\delta_A \leftarrow \mathbb{PRF}(H(y_E^{x_A}), \rho)$ $r_A \leftarrow g^{\delta_A}$ $t \leftarrow \sigma(r_A \| \text{Alice} \| \text{Bob})$	$\xrightarrow{r_A, t}$	
		if t is incorrect **then** abort store t in $\mathcal{L}$
	$\xleftarrow{r_B}$	
if $H_1(0 \| r_B) \neq \rho$ **then** abort $r_i \leftarrow r_A \cdot r_B$ $e \leftarrow H_1(1 \| m \| r \| \text{Alice} \| \text{Bob})$ $s_A \leftarrow \delta_A - e x_A \bmod q$		$r \leftarrow r_A \cdot r_B$ $e \leftarrow H_1(1 \| m \| r \| \text{Alice} \| \text{Bob})$ $s_B \leftarrow \delta_B - e x_B \bmod q$ store s_B in $\mathcal{L}$
	breakpoint 1 $\xleftarrow{s_B}$	
if s_B is incorrect **then** abort		
	breakpoint 2 $\xrightarrow{s_A}$	
		if s_A is incorrect **then** abort
$s_i \leftarrow s_A + s_B \bmod q$		$s \leftarrow s_A + s_B \bmod q$ **if** $\{m, r, s\}$ is valid **then** erase t, s_B from $\mathcal{L}$

Fig. 6. Protocol 4 with a SETUP mechanism.

Security Analysis. The main security results are presented in Theorems 3 and 4. The proofs are omitted given the similarity with the ones presented in Section 3.

Theorem 3. *If* DDH *is hard in $\mathbb{G}$ and H is a one-to-one function, then the protocols presented in Figure 4 and Figure 6 are* IND-SETUP *in the standard model. Formally, let A be an efficient PPT* IND-SETUP *adversary. There exist two efficient PPT algorithms B_1, B_2 such that*

$$ADV^{\text{IND-SETUP}}_{\text{DH},P_4,P_6}(A) \leq 4ADV^{\text{DDH}}_{\mathbb{G},g}(B_1) + 4ADV^{\text{PRF}}_{\mathbb{PRF}}(B_2).$$

Remark 5. From Theorem 3, the maximum advantage an IND-SETUP adversary can obtain in the standard model is

$$ADV^{\text{IND-SETUP}}_{\text{DH},P_4,P_6}(A) \leq 4\Theta ADV^{\text{DDH}}_{\mathbb{G},g}(B_1) + 4\Theta ADV^{\text{PRF}}_{\mathbb{PRF}}(B_2).$$

The advantage remains negligible if parameter Θ is polynomial.

Theorem 4. *If* CDH *is hard in $\mathbb{G}$ and H is a hash function, then the protocols presented in Figure 4 and Figure 6 are* IND-SETUP *in the ROM. Formally, let A be an efficient PPT* IND-SETUP *adversary. There exist three efficient PPT algorithms B_1, B_2 such that*

$$ADV^{\text{IND-SETUP}}_{\text{DH},P_4,P_6}(A) \leq 4ADV^{\text{CDH}}_{\mathbb{G},g}(B_1) + 4ADV^{\text{PRF}}_{\mathbb{PRF}}(B_2).$$

Remark 6. From Theorem 4, the maximum advantage an IND-SETUP adversary can obtain in the ROM is

$$ADV^{\text{IND-SETUP}}_{\text{DH},P_4,P_6}(A) \leq 4\Theta ADV^{\text{CDH}}_{G,g}(C) + 4\Theta ADV^{\text{PRF}}_{\mathbb{PRF}}(B_2).$$

The advantage remains negligible if parameter Θ is polynomial.

5 Conclusions

In this paper we presented various SETUP mechanisms which can be injected in contract signing protocols. We also analyzed the security of the proposed attack scenarios. The reader may easily observe that finding Bob's secret key requires less resources in the scenario described in Section 3 than the one described in Section 4. These two main attacks can be implemented within independent protocol runs and maintain their efficiency, while the mechanisms proposed in Appendices B and C need two consecutive runs to achieve[13] the same efficiency.

Acknowledgments

The authors would like to thank Adrian Atanasiu and the anonymous reviewers for their helpful comments.

References

1. Michel Abdalla, Jee Hea An, Mihir Bellare, and Chanathip Namprempre. From Identification to Signatures via the Fiat-Shamir Transform: Minimizing Assumptions for Security and Forward-Security. In *Advances in Cryptology - EUROCRYPT'02*, volume 2332 of *Lecture Notes in Computer Science*, pages 418–433. Springer, 2002.
2. N. Asokan, Matthias Schunter, and Michael Waidner. Optimistic Protocols for Fair Exchange. In *Proceedings of the 4^{th} ACM Conference on Computer and Communications Security - CCS'97*, pages 7–17. ACM, 1997.
3. Mihir Bellare and Phillip Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security - CCS'93*, pages 62–73. ACM, 1993.
4. Mihir Bellare and Phillip Rogaway. Introduction to Modern Cryptography. *UCSD CSE*, 207:207, 2005.
5. Christian Cachin and Jan Camenisch. Optimistic Fair Secure Computation. In *Advances in Cryptology - CRYPTO'00*, volume 1880 of *Lecture Notes in Computer Science*, pages 93–111. Springer, 2000.
6. Liqun Chen, Caroline Kudla, and Kenneth G. Paterson. Concurrent Signatures. In *Advances in Cryptology - EUROCRYPT'04*, volume 3027 of *Lecture Notes in Computer Science*, pages 287–305. Springer, 2004.
7. Whitfield Diffie and Martin Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.
8. Yevgeniy Dodis, Rosario Gennaro, Johan Håstad, Hugo Krawczyk, and Tal Rabin. Randomness Extraction and Key Derivation Using the CBC, Cascade and HMAC Modes. In *Advances in Cryptology - CRYPTO'04*, volume 3152 of *Lecture Notes in Computer Science*, pages 494–510. Springer, 2004.
9. Taher ElGamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.
10. Houda Ferradi, Rémi Géraud, Diana Maimuţ, David Naccache, and David Pointcheval. Legally Fair Contract Signing Without Keystones. In *International Conference on Applied Cryptography and Network Security - ACNS'16*, volume 9696 of *Lecture Notes in Computer Science*, pages 175–190. Springer, 2016.
11. Uriel Fiege, Amos Fiat, and Adi Shamir. Zero Knowledge Proofs of Identity. In *Proceedings of the 19^{th} Annual ACM Symposium on Theory of Computing - STOC'87*, pages 210–217. ACM, 1987.
12. Juan Garay, Philip MacKenzie, Manoj Prabhakaran, and Ke Yang. Resource Fairness and Composability of Cryptographic Protocols. In *Proceedings of the 3^{rd} Theory of Cryptography Conference - TCC'06*, volume 3876 of *Lecture Notes in Computer Science*, pages 404–428. Springer, 2006.
13. Oded Goldreich. A Simple Protocol for Signing Contracts. In *Advances in Cryptology - CRYPTO'83*, pages 133–136. Springer, 1984.

[13] more or less

14. Shafi Goldwasser, Leonid Levin, and Scott A. Vanstone. Fair Computation of General Functions in Presence of Immoral Majority. In *Advances in Cryptology - CRYPTO'90*, volume 537 of *Lecture Notes in Computer Science*, pages 77–93. Springer, 1991.
15. S. Dov Gordon, Carmit Hazay, Jonathan Katz, and Yehuda Lindell. Complete Fairness in Secure Two-Party Computation. *Jornal of the ACM*, 58(6):1–37, December 2011.
16. Andrew Y. Lindell. Legally-Enforceable Fairness in Secure Two-Party Computation. In *Topics in Cryptology - CT-RSA'08*, volume 4964 of *Lecture Notes in Computer Science*, pages 121–137. Springer, 2008.
17. Silvio Micali. Simple and Fast Optimistic Protocols for Fair Electronic Exchange. In *Proceedings of the 22nd Annual Symposium on Principles of Distributed Computing - PODC'03*, pages 12–19. ACM, 2003.
18. Benny Pinkas. Fair Secure Two-Party Computation. In *Advances in Cryptology - EUROCRYPT'03*, volume 2656 of *Lecture Notes in Computer Science*, pages 87–105. Springer, 2003.
19. Claus-Peter Schnorr. Efficient Identification and Signatures For Smart Cards. In *Advances in Cryptology - CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252. Springer, 1989.
20. Victor Shoup. Sequences of Games: A Tool for Taming Complexity in Security Proofs. *IACR Cryptology ePrint Archive*, 2004:332, 2004.
21. Gustavus J. Simmons. The Subliminal Channel and Digital Signatures. In *Advances in Cryptology - EUROCRYPT'84*, volume 209 of *Lecture Notes in Computer Science*, pages 364–378. Springer, 1985.
22. Gustavus J. Simmons. Subliminal Communication is Easy Using the DSA. In *Advances in Cryptology - EUROCRYPT'93*, volume 765 of *Lecture Notes in Computer Science*, pages 218–232. Springer, 1993.
23. Adam Young and Moti Yung. The Dark Side of "Black-Box" Cryptography or: Should We Trust Capstone? In *Advances in Cryptology - CRYPTO'96*, volume 1109 of *Lecture Notes in Computer Science*, pages 89–103. Springer, 1996.
24. Adam Young and Moti Yung. Kleptography: Using Cryptography Against Cryptography. In *Advances in Cryptology — EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 62–74. Springer, 1997.
25. Adam Young and Moti Yung. The Prevalence of Kleptographic Attacks on Discrete-Log Based Cryptosystems. In *Advances in Cryptology — CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 264–276. Springer, 1997.
26. Adam Young and Moti Yung. *Malicious Cryptography: Exposing Cryptovirology*. John Wiley & Sons, 2004.
27. Adam Young and Moti Yung. Malicious Cryptography: Kleptographic Aspects. In *Topics in Cryptology - CT-RSA'05*, volume 3376 of *Lecture Notes in Computer Science*, pages 7–18. Springer, 2005.

A Additional Preliminaries

Security Models

Definition 7 (Entropy Smoothing - ES). *Let $\mathbb{G}$ be a cyclic group of order n, $\mathcal{K}$ the key space and A a PPT algorithm. Also, let $\mathcal{H} = \{h_i\}_{i\in\mathcal{K}}$ be a family of keyed hash functions, where each h_i maps $\mathbb{G}$ to $\mathbb{Z}_n^*$. We define the advantage*

$$ADV_{\mathcal{H}}^{ES}(A) = \left| Pr[A(i, h_i(z)) = 1 | i \xleftarrow{\$} \mathcal{K}, z \xleftarrow{\$} \mathbb{G}] - Pr[A(i, h) = 1 | i \xleftarrow{\$} \mathcal{K}, h \xleftarrow{\$} \mathbb{Z}_n^*] \right|.$$

If $ADV_{\mathcal{H}}^{ES}(A)$ is negligible for any PPT algorithm A, we say that $\mathcal{H}$ is entropy smoothing.

Remark 7. In [8], the authors prove that the CBC-MAC, HMAC and Merkle-Damgård constructions satisfy the definition above as long as the underling primitives satisfy certain security properties.

Schnorr Signatures

ElGamal signatures [9] inspired the construction of many other DLP based signatures. We particularly refer to Schnorr signatures [19] for the purpose of our current work. This family of signatures is obtained by converting interactive identification protocols into signatures[14].

We shortly describe the algorithms of the Schnorr digital signature scheme in Table 2.

Table 2. Schnorr digital signature.

$\mathsf{Setup}(\ell)$	On input a security parameter ℓ, this algorithm selects large primes p, q such that $q \geq 2^\ell$ and $p - 1 \bmod q = 0$, as well as an element $g \in \mathbb{G}$ of order q in some multiplicative group $\mathbb{G}$ of order $p-1$, and a hash function $H_1 : \{0,1\}^* \to \{0,1\}^\ell$. The output is a set of public parameters $\mathsf{pp} = (p, q, g, \mathbb{G}, H_1)$.
$\mathsf{KeyGen}(\mathsf{pp})$	On input the public parameters pp, this algorithm chooses uniformly at random $x \xleftarrow{\$} \mathbb{Z}_q^*$ and computes $y \leftarrow g^x$. The output is the couple $(\mathsf{sk}, \mathsf{pk})$ where $\mathsf{sk} = x$ is kept private, and $\mathsf{pk} = y$ is made public.
$\mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, m)$	On input public parameters, a secret key sk, and a message m this algorithm selects a random $\delta \xleftarrow{\$} \mathbb{Z}_q^*$, computes $r \leftarrow g^\delta \quad e \leftarrow H_1(m\|r) \quad s \leftarrow \delta - ex \bmod q$ and outputs $\langle r, s\rangle$ as the signature of m.
$\mathsf{Verify}(\mathsf{pp}, \mathsf{pk}, m, \sigma)$	On input public parameters, a public key, a message m and a signature $\sigma = \langle r, s\rangle$, this algorithm computes $e \leftarrow H_1(m\|r)$ and returns True iff $g^s y^e = r$; otherwise it returns False.

[14] as previously described in [1,11] and implicitly used by ElGamal

B A Supplementary SETUP Attack on Concurrent Signatures

Description. Let $H : \mathbb{G} \to \mathbb{Z}_q^*$ be a hash function. Let α be either Alice or Bob. Then, $\delta_{\alpha,0}$ represents α's secret key x_α, $r_{\alpha,0}$ represents α's public key y_α and $r_{\alpha,i} \leftarrow g^{\delta_{\alpha,i}}$. As in Section 3, Eve has a valid pair of keys (x_E, y_E), where y_E is stored on the victim's device.

Again, changes required by the SETUP mechanisms will further be underlined using red colored text in Figure 7.

Alice | *Bob*

Bob:

$k_i \xleftarrow{\$} \{0,1\}^*$
$f_i \leftarrow H_1(k_i)$
$\delta_{B,i} \leftarrow H(y_E^{\delta_{B,i-1}})$
$\eta_{B,i} \leftarrow H_2\left(g^{\delta_{B,i}} y_A^{f_i} \| m_{B,i}\right)$
$e_{B,i} \leftarrow \eta_{B,i} - f_i \bmod q$
$s_{B,i} \leftarrow \delta_{B,i} - e_{B,i} x_B \bmod q$
$\sigma_{B,i} \leftarrow \langle s_{B,i}, e_{B,i}, f_i \rangle$

breakpoint (1)

Bob → Alice: $\sigma_{B,i}$

Alice:

$T_A \leftarrow H_2\left(g^{s_{B,i}} y_B^{e_{B,i}} y_A^{f_i} \| m_{B,i}\right) \bmod q$
if $T_A \not\equiv e_{B,i} + f_i \bmod q$ **then** abort
$\delta_{A,i} \leftarrow H(y_E^{\delta_{A,i-1}})$
$\eta_{A,i} \leftarrow H_2\left(g^{\delta_{A,i}} y_B^{f_i} \| m_{A,i}\right)$
$e_{A,i} \leftarrow \eta_{A,i} - f_i \bmod q$
$s_{A,i} \leftarrow \delta_{A,i} - e_{A,i} x_A \bmod q$
$\sigma_{A,i} \leftarrow \langle s_{A,i}, e_{A,i}, f_i \rangle$

breakpoint (2)

Alice → Bob: $\sigma_{A,i}$

Bob:

$T_B \leftarrow H_2\left(g^{s_{A,i}} y_A^{e_{A,i}} y_B^{f_i} \| m_{A,i}\right) \bmod q$
if $T_B \not\equiv e_{A,i} + f_i \bmod q$ **then** abort

Bob → Alice: k_i

Fig. 7. Iteration i of Protocol 3 with a supplementary SETUP mechanism.

Eve can decide to recover Alice's secret key whenever she wants. To do that, she must first compute $\delta_{A,i} = H(r_{A,i-1}^{x_E})$. Eve recovers $r_{A,i-1}$ from an older protocol in which Alice was involved, more precisely the $i-1$ one. Thus, Eve calculates

$$g^{s_{A,i-1}} y_A^{e_{A,i-1}} \equiv g^{s_{A,i-1} + e_{A,i-1} x_A} \equiv g^{\delta_{A,i-1}} \equiv r_{A,i-1}.$$

Eve's final goal is finding x_A which can be achieved by computing $e_{A,i}^{-1}(\delta_{A,i} - s_{A,i})$. The values $e_{A,i}$ and $s_{A,i}$ are transmitted during the protocol and are public. Similarly, she can recover Bob's secret key.

The most efficient way to recover secret keys is by observing two consecutive protocol iterations that need to reach breakpoint (2).

Exceptions. An exception is iteration 1, since $\delta_{\alpha,0}$ is already known. Thus, only protocol 1 needs to reach breakpoint (2). Eve can also recover secret keys at iteration i by computing all intermediary values, $\delta_{\alpha,j}$ for $0 \le j < i$. This method is computationally costly.

Malicious Co-Signers. If Eve is replaced by Alice, the most efficient way to recover secret keys is by observing two protocol iterations that need to reach breakpoint ①.

If Eve is replaced by Bob, the most efficient way to recover secret keys is by running two protocol iterations that need to reach breakpoint ②.

Security Analysis. We present the main security results, more precisely Theorems 5 and 6, and provide the reader with the necessary proofs.

When referring to the security analysis presented in the current section, Θ is considered an additional security parameter and refers to the maximal number of protocol iterations.

Theorem 5. *Let i be an integer smaller than Θ. If* DDH *is hard in $\mathbb{G}$ and H is* ES*, then iterations i of the protocols presented in Figure 3 and Figure 7 are* IND-SETUP *in the standard model. Formally, let A be an efficient PPT* IND-SETUP *adversary then there exist two efficient PPT algorithms B_1, B_2 such that*

$$ADV^{\text{IND-SETUP}}_{DH,P_3,P_7}(A) \leq 4ADV^{DDH}_{\mathbb{G},g}(B_1) + 4ADV^{ES}_{\mathcal{H}}(B_2).$$

Proof. We denote iterations i of the protocols presented in Figure 3 and Figure 7 by P_3 and P_7. Let A be an IND-SETUP adversary trying to distinguish between P_3 and P_7. We show that his advantage is negligible. We present the proof as a sequence of games and all the required changes are made to P_7. Let W_i be the event that A wins game i.

Game 0. The first game is identical to the IND-SETUP game[15]. Thus, we have

$$|2Pr[W_0] - 1| = ADV^{\text{IND-SETUP}}_{\text{DH},P_3,P_7}(A). \tag{13}$$

Game 1. In this game, $y_E^{\delta_{A,i-1}}$ and $y_E^{\delta_{B,i-1}}$ from *Game 0* become $g^{z_{A,i}}$ and $g^{z_{B,i}}$, where $z_{A,i}, z_{B,i} \xleftarrow{\$} \mathbb{Z}_q$. Since this is the only change between *Game 0* and *Game 1*, A will not notice the difference assuming the DDH assumption holds. Formally, this means that there exists an algorithm B_1 such that

$$|Pr[W_0] - Pr[W_1]| = 2ADV^{\text{DDH}}_{\mathbb{G},g}(B_1). \tag{14}$$

Game 2. Since H is ES then we can make the change $\delta_{A,i}, \delta_{B,i} \xleftarrow{\$} \mathbb{Z}_q$ and adversary A will not notice. Formally, this means that there exists an algorithm B_2 such that

$$|Pr[W_1] - Pr[W_2]| = 2ADV^{\text{ES}}_{\mathcal{H}}(B_2) \tag{15}$$

The changes made to P_7 in *Game 1* and *Game 2*, transformed it into P_3. Thus, we have

$$Pr[W_2] = 1/2. \tag{16}$$

Finally, the statement is proven by combining the equalities (13) – (16). □

Remark 8. From Theorem 5, the maximum advantage an IND-SETUP adversary can obtain in the standard model is

$$ADV^{\text{IND-SETUP}}_{\text{DH},P_3,P_7}(A) \leq 4\Theta ADV^{\text{DDH}}_{\mathbb{G},g}(B_1) + 4\Theta ADV^{\text{ES}}_{\mathcal{H}}(B_2).$$

The advantage remains negligible if parameter Θ is polynomial.

Theorem 6. *Let i be an integer smaller than Θ. If* CDH *is hard in $\mathbb{G}$, then iterations i of the protocols presented in Figure 3 and Figure 7 are* IND-SETUP *in the ROM. Formally, let A be an efficient PPT* IND-SETUP *adversary then there exist an efficient PPT algorithms C such that*

$$ADV^{\text{IND-SETUP}}_{DH,P_3,P_7}(A) \leq 4ADV^{CDH}_{\mathbb{G},g}(C).$$

[15] as in Definition 6

Proof. We will use the same notations as in the proof for Theorem 5.

Game 0. The first game is identical to the IND-SETUP game[16]. Thus, we have

$$|2Pr[W_0] - 1| = ADV^{\text{IND-SETUP}}_{\text{DH},P_3,P_7}(A). \tag{17}$$

The challenger picks a random oracle $H : \mathbb{G} \to \mathbb{Z}_q^*$ at random from the set of all such functions. A can make a sequence of queries of the following type:

Hash oracle query[17]: A presents the challenger with $m \in \mathbb{G}$, who responds with $H(m)$.

Game 1. At the beginning of the game choose $z_{A,i}, z_{B,i} \xleftarrow{\$} \mathbb{Z}_q^*$. We change the challenger's way to respond to queries as follows:

Hash oracle query[18]: A presents the challenger with $m \in \mathbb{G}$. The challenger responds with:

- $z_{A,i}$, if $m = y_E^{\delta_{A,i-1}}$;
- $z_{B,i}$, if $m = y_E^{\delta_{B,i-1}}$;
- $H(m)$, otherwise.

We also make the changes $\delta_{A,i} \leftarrow z_{A,i}$ and $\delta_{B,i} \leftarrow z_{B,i}$ in P_7.

Since we have replaced the values $y_E^{\delta_{A,i-1}}$ and $y_E^{\delta_{B,i-1}}$ throughout the game, we have

$$Pr[W_0] = Pr[W_1]. \tag{18}$$

Game 2. In this game, we revert to the original hash oracle query (*i.e* the challenger responds with $H(m)$ for all m). Let F be the event that the adversary makes a query with $m \leftarrow y_E^{\delta_{A,i-1}}$ or $m \leftarrow y_E^{\delta_{B,i-1}}$. *Game 1* and *Game 2* are identical until F occurs. Thus, we have

$$|Pr[W_1] - Pr[W_2]| \leq Pr[F]. \tag{19}$$

We need to prove that

$$Pr[F] = ADV^{\text{LCDH2}}_{\mathbb{G},g}(C), \tag{20}$$

where C is an algorithm that takes as input y_E, $r_{A,i-1}$ and $r_{B,i-1}$. C will play the role of the challenger in *Game 2*. Algorithm C has a list of queries and responses, such that if A makes a query that matches one of the previous queries, C can return the previous output. At the end of the game, algorithm C will output a list with all the responses to A's queries. It is easy to see that the probability of C returning a list containing $y_E^{\delta_{A,i-1}}$ or $y_E^{\delta_{B,i-1}}$ is the same as $Pr[F]$.

The changes made to P_7 in *Game 1* and *Game 2*, transformed it into P_3. Thus, we have

$$Pr[W_2] = 1/2. \tag{21}$$

Finally, the statement is proven by combining the equalities (17) – (21), □

Remark 9. From Theorem 6, the maximum advantage an IND-SETUP adversary can obtain in the ROM is

$$ADV^{\text{IND-SETUP}}_{\text{DH},P_3,P_7}(A) \leq 4\Theta ADV^{\text{CDH}}_{\mathbb{G},g}(C).$$

The advantage remains negligible if parameter Θ is polynomial.

[16] as in Definition 6

[17] *Game 0*

[18] *Game 1*

C A Supplementary SETUP Attack on Legally Fair Signatures without Keystones

Description. To implement an attack, Eve will work in almost the same environment as in Appendix B. Thus, we only mention the differences between the environments.

As in Section 3, changes required by the SETUP mechanisms are further underlined using red colored text in Figure 8.

Alice		*Bob*
$y_{A,B} \leftarrow y_A \times y_B$		$y_{A,B} \leftarrow y_A \times y_B$
$\delta_{A,i} \leftarrow H(y_E^{\delta_{A,i-1}})$		$\delta_{B,i} \leftarrow H(y_E^{\delta_{B,i-1}})$
$r_{A,i} \leftarrow g^{\delta_{A,i}}$		$r_{B,i} \leftarrow g^{\delta_{B,i}}$
		$\rho_i \leftarrow H_1(0 \Vert r_{B,i})$
	$\xleftarrow{\rho_i}$	
	breakpoint 1	
$t_i \leftarrow \sigma(r_{A,i} \Vert \text{Alice} \Vert \text{Bob})$	$\xrightarrow{r_{A,i},t_i}$	
		if t_i is incorrect **then** abort
		store t_i in $\mathcal{L}$
	breakpoint 2	
	$\xleftarrow{r_{B,i}}$	
if $H_1(0 \Vert r_{B,i}) \neq \rho_i$ **then** abort		
$r_i \leftarrow r_{A,i} \times r_{B,i}$		$r_i \leftarrow r_{A,i} \times r_{B,i}$
$e_i \leftarrow H_1(1 \Vert m_i \Vert r_i \Vert \text{Alice} \Vert \text{Bob})$		$e_i \leftarrow H_1(1 \Vert m_i \Vert r_i \Vert \text{Alice} \Vert \text{Bob})$
$s_{A_i} \leftarrow \delta_{A,i} - e_i x_A \bmod q$		$s_{B,i} \leftarrow \delta_{B,i} - e_i x_B \bmod q$
		store $s_{B,i}$ in $\mathcal{L}$
	breakpoint 3	
	$\xleftarrow{s_{B,i}}$	
if $s_{B,i}$ is incorrect **then** abort		
	breakpoint 4	
	$\xrightarrow{s_{A,i}}$	
		if $s_{A,i}$ is incorrect **then** abort
$s_i \leftarrow s_{A,i} + s_{B,i} \bmod q$		$s_i \leftarrow s_{A,i} + s_{B,i} \bmod q$
		if $\{m_i, r_i, s_i\}$ is valid **then**
		erase $t_i, s_{B,i}$ from $\mathcal{L}$

Fig. 8. Iteration i of Protocol 4 with a supplementary SETUP mechanism.

The most efficient way for Eve to recover secret keys is taking into account the following requirements:

1. an iteration needs to reach breakpoint 4;
2. the previous protocol iteration needs to reach breakpoint 2.

Malicious Co-Signers. If Eve is replaced by Alice, the most efficient way to recover secret keys is taking into account the following requirements:

1. an iteration needs to reach breakpoint 3;
2. the previous protocol iteration needs to reach breakpoint 2.

If Eve is replaced by Bob, the most efficient way to recover secret keys is taking into account the following requirements:

1. an iteration needs to reach breakpoint 4;
2. the previous protocol iteration needs to reach breakpoint 1.

Security Analysis. The main security results are presented in Theorems 7 and 8. The proofs are omitted given their similarities with the ones constructed in Appendix B.

Theorem 7. *Let i be an integer smaller than Θ. If* DDH *is hard in $\mathbb{G}$ and H is* ES, *then iterations i of the protocols presented in Figure 4 and Figure 8 are* IND-SETUP *in the standard model. Formally, let A be an efficiet PPT* IND-SETUP *adversary. There exist two efficient PPT algorithms B_1, B_2 such that*

$$ADV^{\textit{IND-SETUP}}_{\mathrm{DH},P_4,P_8}(A) \leq 4ADV^{\textit{DDH}}_{\mathbb{G},g}(B_1) + 4ADV^{\textit{ES}}_{\mathcal{H}}(B_2).$$

Remark 10. From Theorem 7, the maximum advantage an IND-SETUP adversary can obtain in the standard model is

$$ADV^{\text{IND-SETUP}}_{\mathrm{DH},P_4,P_8}(A) \leq 4\Theta ADV^{\text{DDH}}_{\mathbb{G},g}(B_1) + 4\Theta ADV^{\text{ES}}_{\mathcal{H}}(B_2).$$

The advantage remains negligible if parameter Θ is polynomial.

Theorem 8. *Let i be an integer smaller than Θ. If* CDH *is hard in $\mathbb{G}$, then iterations i of the protocols presented in Figure 4 and Figure 8 are* IND-SETUP *in the ROM. Formally, let A be an efficient PPT* IND-SETUP *adversary. There exist an efficient PPT algorithms C such that*

$$ADV^{\textit{IND-SETUP}}_{\mathit{DH},P_4,P_8}(A) \leq 4ADV^{\textit{CDH}}_{\mathbb{G},g}(C).$$

Remark 11. From Theorem 8, the maximum advantage an IND-SETUP adversary can obtain in the ROM is

$$ADV^{\text{IND-SETUP}}_{\mathrm{DH},P_4,P_8}(A) \leq 4\Theta ADV^{\text{CDH}}_{\mathbb{G},g}(C).$$

The advantage remains negligible if parameter Θ is polynomial.