Relation between Verifiable Random Functions and Convertible Undeniable Signatures, and New Constructions

Kaoru Kurosawa[1], Ryo Nojima[2], and Le Trieu Phong[2]

[1] Ibaraki University
[2] NICT, Japan

Abstract. Verifiable random functions (VRF) and selectively-convertible undeniable signature (SCUS) schemes were proposed independently in the literature. In this paper, we observe that they are tightly related. This directly yields several deterministic SCUS schemes based on existing VRF constructions. In addition, we create a new probabilistic SCUS scheme, which is very compact. The confirmation and disavowal protocols of these SCUS are efficient, and can be run either sequentially, concurrently, or arbitrarily. These protocols are based on what we call *zero-knowledge protocols for generalized DDH and non-DDH*, which are of independent interest.

Keywords: Selectively convertible undeniable signatures, verifiable random function, standard model.

1 Introduction

1.1 Background

Selectively convertible undeniable signatures (**SCUS**) were introduced by Boyar, Chaum, Damgård, and Pedersen [2] in 1990, extending the concept of undeniable signatures of Chaum and Antwerpen [5]. Recall that the verification of undeniable signatures is restricted, namely controlled by the signer. Any verifier needs to run an interactive protocol with the signer to check whether the purported signature is valid. However, in SCUS schemes, the verification of signatures can be additionally done non-interactively; namely the signer now can release a piece of information, often called a converter, to make his signature publicly checked.

SCUS schemes can be used directly, when one would like to allow some verifiers to freely check his signature, while the other cannot do so. For example, IACR members who receive the converter, can directly verify the signature, while non-members cannot. SCUS schemes can also be used as a the main building block in protocols such as fair payment [3]. There is a long list of work on SCUS schemes in the literature, including [10, 15, 17, 26, 28] in the standard model.

On the other hand, *verifiable random functions* (**VRF**s), later introduced by Micali, Rabin, and Vadhan [21] in 1999, are like pseudo-random functions in the sense that their outputs are indistinguishable from random. However, unlike standard pseudo-random functions, the output of VRFs can be proved coming from a given input if one (owning the secret key) releases an additional piece of information, often called the proof of correctness.

VRFs have been used in various contexts, including resettable zero-knowledge proofs [22], micro-payment schemes [23], updatable zeroknowledge databases [20], and verifiable transaction escrow schemes [14].

1.2 Our contributions

We first show VRF implies deterministic SCUS. The reverse side holds if SCUS has an additional property called uniqueness. Furthermore, probabilistic SCUS can be seen as "randomized" VRF

Table 1. Some recent SCUS schemes in the standard model, operating on a pairing group $\mathbb{PG} = (G, G_T, g, q, e)$, where g is a generator of G, and the order $|G| = |G_T| = q$ for prime q, and $e : G \times G \to G_T$. The notation 3_G means 3 group elements in G; 1_{G_T} means 1 group element in G_T. Typically, $s_G = 160_G$

SCUS schemes	**Signature**	**Converter**	**Public keys**
	(all sizes are in group elements)		
PKO [26]	$\geq 3_G$	2_G	$\geq 12_G$
SM [28]	4_G	2_G	$\approx s_G$
Our $\mathsf{SCUS}_{\mathsf{DY}}$ (Sect.4.2)	1_{G_T}	1_G	2_G
Our $\mathsf{SCUS}_{\mathsf{HW}}$ (Sect.4.2)	1_{G_T}	$\approx s_G$	$\approx s_G$
Our $\mathsf{SCUS}_{\mathsf{BB}}$ (Sect.5)	$1_{\mathbb{Z}_q} + 1_{G_T}$	1_G	4_G

(taking randomness as a function input). Perhaps surprisingly, these facts were not notified before in the literature of both SCUS and VRF. With these insights, one can use SCUS in protocols originally employed VRF, and vice versa. This broadens the tools one can use in designing protocols.

Then, also based on the relation, we construct new deterministic SCUS in the standard model. Going further, we build a probabilistic SCUS scheme with *very neat* parameters. A comparison with the best known schemes in the standard model is given in Table 1. More details are as follows.

Deterministic SCUS from VRF. We show that VRFs directly imply deterministic SCUS schemes. This result, coupling with our zero-knowledge protocols for generalized DDH and non-DDH, then yields several efficient SCUS schemes. We give in Sect.4.2 two concrete deterministic SCUS schemes denoted as $\mathsf{SCUS}_{\mathsf{DY}}$ and $\mathsf{SCUS}_{\mathsf{HW}}$, respectively based on the VRFs of Dodis, Yampolskiy [9] and Hohenberger, Waters [13]. We note that $\mathsf{SCUS}_{\mathsf{DY}}$ requires small signing space, while $\mathsf{SCUS}_{\mathsf{HW}}$ does not.

Interlude: A new probabilistic SCUS. The converters and/or public keys in the above deterministic SCUS schemes are a bit long (e.g., of at least 160 group elements in $\mathsf{SCUS}_{\mathsf{HW}}$). Inspired from $\mathsf{SCUS}_{\mathsf{DY}}$ and the technique of Boneh and Boyen [1], we newly construct a more compact probabilistic SCUS scheme, called $\mathsf{SCUS}_{\mathsf{BB}}$, with arbitrary signing space. The scheme is given in Sect.5.

VRF from SCUS. We establish the other side as well, namely VRFs from SCUS schemes with a unique property. The property requires, for *every*[3] public key, there is one valid signature for each message in SCUS. However, since there is no known unique SCUS scheme currently, this is just of theoretical interest. See Sect.4.3 for the construction.

Note that the confirmation and disavowal protocols for our SCUS constructions can be efficiently and elegantly realized from what we call *zero-knowledge protocols for generalized DDH and non-DDH* in Sect.3. These protocols are of independent interest, and may be useful in other context as well.

2 Definitions

By $a \xleftarrow{\$} U$ we mean a is chosen randomly from a set U, and by $\mathcal{A}^{\mathcal{O}(\cdot)}$ we mean the adversary $\mathcal{A}$ gets access to the oracle $\mathcal{O}(\cdot)$. The value $|a|$ is the length in bits of the element a, while $|\mathcal{H}|$ is the order of a group $\mathcal{H}$.

Verifiable random function. The function, described as $\mathsf{VRF} = (\mathsf{Gen}, \mathsf{Func}, \mathsf{V})$, is as follows.

[3] This unique property is a little stronger than "deterministic", since it deals also with malformed public keys. If the public key is honestly created, both are identical.

- $\mathsf{Gen}(1^\lambda)$: return the public key pk and the secret key sk.
- $\mathsf{Func}_{sk}(x)$: return the (pseudo-random) output $y = F_{sk}(x)$ for $F_{sk}(\cdot) : \mathsf{Domain}(\lambda, pk) \to \mathsf{Range}(\lambda, pk)$ [4], and its proof of correctness $\pi = \pi_{sk}(x)$.
- $\mathsf{V}_{pk}(x, y, \pi)$: return 1 (meaning $y = F_{sk}(x)$) or 0.

We require the following properties on VRF.

Provability: if $(y, \pi) = \mathsf{Func}_{sk}(x)$ then $\mathsf{V}_{pk}(x, y, \pi) = 1$ (except with negligible probability), for all $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$, and $x \in \mathsf{Domain}(\lambda)$.

Uniqueness: Except with negligible probability, there is no tuple $(x, y_1, y_2, \pi_1, \pi_2)$ satisfying

$$y_1 \neq y_2 \text{ and } \mathsf{V}_{pk}(x, y_1, \pi_1) = \mathsf{V}_{pk}(x, y_2, \pi_2) = 1,$$

for all pk, and $x \in \mathsf{Domain}(\lambda)$.

Pseudo-randomness: For all poly-time distinguisher $\mathcal{D} = (\mathcal{D}_1, \mathcal{D}_2)$, the following advantage

$$\mathbf{Adv}^{\text{rand}}_{\mathsf{VRF}}(\mathcal{D}) =$$
$$\left| \Pr\left[b' = b : \begin{array}{l} (pk, sk) \xleftarrow{\$} \mathsf{Gen}(1^\lambda), (x^*, \mathtt{st}) \leftarrow \mathcal{D}_0^{\mathsf{Func}_{sk}(\cdot)}(1^\lambda, pk) \\ y_0^* \leftarrow F_{sk}(x^*), y_1^* \xleftarrow{\$} \mathsf{Range}(\lambda), \\ b \xleftarrow{\$} \{0,1\}, b' \leftarrow \mathcal{D}_1^{\mathsf{Func}_{sk}(\cdot)}(y_b^*, \mathtt{st}) \end{array} \right] - \frac{1}{2} \right|,$$

where x^* is not queried to the oracle $\mathsf{Func}_{sk}(\cdot)$, is negligible in λ. Above, $\mathtt{st}$ stands for state.

Selectively convertible undeniable signature scheme. The scheme, denoted as SCUS, consists of the algorithms (KGen, USign, $\mathsf{Convert}$, Verify) and the protocols ($\mathsf{Confirm}$, $\mathsf{Disavowal}$) described as follows.

- $\mathsf{KGen}(1^\lambda)$: return the public key pk and the secret key (signing key) sk.
- $\mathsf{USign}_{sk}(m)$: return a signature σ on a message m. We sometimes require that it is deterministic so that a signature σ is valid on m if and only if $\sigma = \mathsf{USign}_{sk}(m)$. If the algorithm is probabilistic with randomness r, we sometimes write $\sigma = \mathsf{USign}_{sk}(m; r)$.
- $\mathsf{Convert}_{sk}(m, \sigma)$: release a converter cvt if (m, σ) is valid, and $\perp$ otherwise.
- $\mathsf{Verify}_{pk}(m, \sigma, cvt)$: return 1 (meaning (m, σ) is valid) or 0.
- $\mathsf{Confirm}$: This is a protocol between the signer and a verifier, on common input (pk, m, σ), the signer with sk proves that (m, σ) is a valid message-signature pair in zero-knowledge.
- $\mathsf{Disavowal}$: This is a protocol between the signer and a verifier, on common input (pk, m, σ), the signer with sk proves that (m, σ) is an invalid message-signature pair in zero-knowledge.

Definition 1 (Unforgeability of SCUS) *The scheme* SCUS *is unforgeable if for all poly-time adversary* $\mathcal{A}$*, the advantage*

$$\mathbf{Adv}^{\text{forge}}_{\mathsf{SCUS}}(\mathcal{A}) =$$
$$\Pr\left[(m^*, \sigma^*) \textit{ is valid } : \begin{array}{l} (pk, sk) \leftarrow \mathsf{KGen}(1^\lambda), \\ (m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{USign}_{sk}(\cdot), \mathsf{Convert}_{sk}(\cdot,\cdot), \mathcal{P}}(pk), \\ m^* \textit{ was not queried to } \mathsf{USign}_{sk}(\cdot) \end{array} \right]$$

is negligible in λ*. Above and below,* $\mathcal{P}$ *stands for the confirmation/disavowal oracle working as follows:* $\mathcal{A}$ *submits a message-signature pair of the form* (m, σ) *to* $\mathcal{P}$*, which first checks the validity of* (m, σ)*. If it is a valid pair, the oracle returns* 1 *and executes the confirmation protocol with* $\mathcal{A}$

[4] We will be mainly interested in the case $\mathsf{Range}(\lambda)$ is big enough, e.g., $\mathsf{Range}(\lambda) = \{0,1\}^\lambda$ for $\lambda = 170$.

(acting as a cheating verifier). Otherwise, the oracle returns 0 *and executes the disavowal protocol with* $\mathcal{A}$.

For strong unforgeability, the forged pair (m^*, σ^*) *is different from the pairs appeared at the oracle* $\mathsf{USign}_{sk}(\cdot)$. *(Yet* m^* *can be queried to* $\mathsf{USign}_{sk}(\cdot)$, *yielding for example* σ_*. *In that case,* $\sigma^* \neq \sigma_*$.*).*

We note that all SCUS schemes in this paper meet the notion of strong unforgeability.

Definition 2 (Invisibility of SCUS [11]) *The scheme* SCUS *has invisibility if for all poly-time distinguisher* $\mathcal{D} = (\mathcal{D}_0, \mathcal{D}_1)$, *the advantage*

$$\mathbf{Adv}^{\text{inv}}_{\mathsf{SCUS}}(\mathcal{D}) = \left| \Pr\left[b' = b : \begin{array}{l} (pk, sk) \leftarrow \mathsf{KGen}(1^\lambda), \\ (m^*, \mathtt{st}) \leftarrow \mathcal{D}_0^{\mathsf{USign}_{sk}(\cdot), \mathsf{Convert}_{sk}(\cdot,\cdot), \mathcal{P}}(pk), \\ \sigma_0^* \leftarrow \mathsf{USign}_{sk}(m^*), \sigma_1^* \xleftarrow{\$} \mathsf{SigSpace}(\lambda), \\ b' \leftarrow \mathcal{D}_1^{\mathsf{USign}_{sk}(\cdot), \mathsf{Convert}_{sk}(\cdot,\cdot), \mathcal{P}}(\sigma^* \overset{\text{def}}{=} \sigma_b^*, \mathtt{st}) \end{array} \right] - 1/2 \right|$$

is negligible in λ, *where* $\mathsf{SigSpace}(\lambda)$ *is the signature space of the scheme. Also, there are some natural restrictions on* $\mathcal{D}_1$*'s queries: no confirmation/disavowal query and conversion query* (m^*, σ^*), *and no signing query* m^* *(needed only if* USign *is deterministic), are allowed.*

3 Zero-knowledge protocols for generalized DDH and non-DDH

Protocols from q-oneway homomorphism [6]. Let Dom and Rng be finite Abelian groups, and $f : \mathsf{Dom} \to \mathsf{Rng}$ be a group homomorphism. Following Cramer, Damgård, and MacKenzie, we say a one-way function f is q-oneway for a fixed prime q, if given f and b in f's range, one can efficiently find $a \in \mathsf{Dom}$ such that $f(a) = b^q$. From such f, it is shown in [6] how to build a 4-move zero-knowledge (ZK) protocol proving the knowledge of f's pre-image. Namely, for public x, the prover shows the knowledge of ω such that $f(\omega) = x$. The zero-knowledge protocol is based on the following Σ-protocol.

Σ-protocol proving $f(\omega) = x$

1. The prover chooses $\omega' \xleftarrow{\$} \mathsf{Dom}$, and sends $x' = f(\omega')$ to the verifier.
2. The verifier sends back a random challenge $c \in \{0, \ldots, q-1\}$.
3. The prover returns $\omega'' = \omega' + c\omega$ to the verifier who checks $f(\omega'') = x'x^c$.

The well-known Schnorr protocol [27] becomes a special case of the above, when considering $f : \mathbb{Z}_q \to \mathcal{H}$, and $f(\omega) = h^\omega$, where the order $|\mathcal{H}| = q$ and h is a generator of $\mathcal{H}$. It is easy to see that f is an oneway homomorphism. For $y \in \mathcal{H}$, we have $y^q = 1$, thus $f(0) = y^q$, and hence f is q-oneway.

ZK protocol for generalized DDH. Consider two generators $g, h \in \mathcal{H}$. The elements $X_i = h^{x_i}$ for $1 \leq i \leq n$, and Y are given in public. The prover with secrets x_i wants to show in ZK that $Y = g^{\prod_{i=1}^n x_i}$. When $g = h$ and $n = 1$, this is exactly the Schnorr protocol.

Let $y_j = \prod_{i=1}^j x_i$ for $1 \leq j \leq n$. For the following q-oneway homomorphism

$$f(y_1, \ldots, y_n) = \left(h^{y_1}, h^{y_2} X_2^{-y_1}, \ldots, h^{y_n} X_n^{-y_{n-1}}, g^{y_n} \right),$$

our prover proves that he knows $(y_1, \ldots, y_n)$ such that $f(y_1, \ldots, y_n) = (X_1, 1, \ldots, 1, Y)$ by using the above Σ-protocol. The protocol is a 3-move ZK protocol against honest verifier. It can be transformed to a 4-move ZK protocol against any verifier.

(Completeness) It holds that $h^{y_j} X_j^{-y_{j-1}} = 1$ because $h^{y_j} = X_j^{y_{j-1}}$ for all $2 \leq j \leq n$. (Soundness) Note that $h^{y_1} = X_1$ implies $y_1 = x_1$, and $h^{y_2} X_2^{-y_1} = 1$ implies $y_2 = y_1 x_2 = x_1 x_2$, and so on, so that $y_n = \prod_{i=1}^n x_i$ as required.

Neff [25], with other techniques, also gave a protocol called iterated logarithmic multiplication protocol realizing the above task when $g = h$. Ours, described above, is more compact and general.

ZK protocol for generalized non-DDH. Given $h, X_i = h^{x_i}, Y \in \mathcal{H}$, and now the prover wants to show that $Y \neq h^{\prod_{i=1}^n x_i}$. We call this *generalized non-DDH protocol*, which will be used for disavowal in later SCUS schemes. Employing a trick of Camenisch and Shoup [4], the prover takes $x_0 \xleftarrow{\$} \mathbb{Z}_q$ and sends $T = \left(Y^{-1} h^{\prod_{i=1}^n x_i}\right)^{x_0}$ to the verifier who checks $T \neq 1$. Let $y_j = \prod_{i=0}^j x_i$ so that $T = Y^{-x_0} h^{y_n}$. Note that $h^{y_1} X_1^{-x_0} = 1$ and $h^{y_j} X_j^{-y_{j-1}} = 1$ for $2 \leq i \leq n$. We then consider the following q-oneway homomorphism

$$f(x_0, y_1, \ldots, y_n) = \left(h^{y_1} X_1^{-x_0}, h^{y_2} X_2^{-y_1}, \ldots, h^{y_n} X_n^{-y_{n-1}}, Y^{-x_0} h^{y_n}\right),$$

so that the prover just needs to prove the knowledge of the pre-image satisfying $f(x_0, y_1, \ldots, y_n) = (1, \ldots, 1, T)$. The protocol is 3-move against honest verifier, and 4-move against any verifier.

Full zero-knowledgeness from the Σ-protocols is described in Appendix A.

4 Relation between SCUS and VRF

4.1 SCUS from VRF: a Generic Construction

Let $\mathsf{VRF} = (\mathsf{Gen}, \mathsf{Func}, \mathsf{V})$ be a VRF for $\mathsf{Func}_{sk}(\cdot) = \left(F_{sk}(\cdot), \pi_{sk}(\cdot)\right)$ where the range of $F_{sk}(\cdot)$ is considered as $\{0,1\}^\kappa$ ($\kappa \approx 170$ in later sections). We now construct a scheme $\mathsf{SCUS} = (\mathsf{KGen}, \mathsf{USign}, \mathsf{Convert}, \mathsf{Verify}, \mathsf{Confirm}, \mathsf{Disavowal})$ described as follows.

<u>The construction of SCUS:</u>

– $\mathsf{KGen}(1^\lambda)$: Return $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$.
– $\mathsf{USign}_{sk}(m)$: Return $\sigma = F_{sk}(m)$ as the undeniable signature.
– $\mathsf{Convert}_{sk}(m, \sigma)$: Return $\pi = \pi_{sk}(m)$ as the converter if $\mathsf{V}_{pk}(m, \sigma, \pi) = 1$, else return $\perp$.
– $\mathsf{Verify}_{pk}(m, \sigma, \pi)$: Return $\mathsf{V}_{pk}(m, \sigma, \pi)$.
– $\mathsf{Confirm}$: The common input is (pk, m, σ). With private input sk, the signer proves in zero-knowledge that $\sigma = F_{sk}(m)$.
– $\mathsf{Disavowal}$: The common input is (pk, m, σ). With private input sk, the signer proves in zero-knowledge that $\sigma \neq F_{sk}(m)$.

In general, the Confirm and Disavowal protocols can be realized by general techniques for zero-knowledge, but not very efficient. Later, we will show efficient protocols for specific cases.

Theorem 3 *The* SCUS *construction above satisfies unforgeability.*

Proof. Given $\mathcal{A}$ against the unforgeability of SCUS, we build a distinguisher $\mathcal{D}$ against the pseudo-randomness of VRF. The distinguisher $\mathcal{D}$, on input pk, runs $\mathcal{A}$ also with pk and answers $\mathcal{A}$'s queries as follows.

– Query m to $\mathsf{USign}_{sk}(\cdot)$: $\mathcal{D}$ calls its own oracle $\mathsf{Func}_{sk}(\cdot)$ on m to obtain $\big(\sigma = F_{sk}(m), \pi = \pi_{sk}(m)\big)$. Return σ as the undeniable signature. Furthermore, $\mathcal{D}$ adds (m, σ, π) into a list $\mathcal{Q}_{\text{usign}}$, which is initially empty.

Before using a technique as in [16] to continue the simulation, let us add some notations. Let n_{qr} be the total number of convert and confirmation/disavowal queries, each of which is indexed by a number i. Without loss of generality, consider the final output of $\mathcal{A}$ as the final confirmation/disavowal query (since, otherwise, $\mathcal{D}$ can call the oracle with the query). Let i^* be *the first* valid message-signature pair, denoted as (m_*, σ_*), queried by $\mathcal{A}$ to the convert or the confirmation/disavowal oracle so that $(m_*, \sigma_*, \cdot) \notin \mathcal{Q}_{\text{usign}}$. Certainly, i^* exists if $\mathcal{A}$ wins, but we do not know the index exactly. We then guess i^* by choosing $\mathsf{guess} \xleftarrow{\$} \{1, \ldots, n_{qr}+1\}$, so that $\Pr[i^* = \mathsf{guess}] = 1/(n_{qr}+1)$, which is non-negligible. Below simulation is conditioned on the event $i^* = \mathsf{guess}$.

– Query (m, σ) indexed $i < \mathsf{guess}$ to either $\mathsf{Convert}_{sk}(\cdot, \cdot)$ or $\mathcal{P}$: If $(m, \sigma, \pi) \in \mathcal{Q}_{\text{usign}}$ for some π, then return π in case of conversion query, or return 1 and execute the confirmation protocol in case of confirmation/disavowal query.
  Otherwise, if $(m, \sigma, \cdot) \notin \mathcal{Q}_{\text{usign}}$, this must be an invalid message-signature pair (since $i < \mathsf{guess}$), so $\mathcal{D}$ just returns $\perp$ (for conversion query), or returns 0 and runs the disavowal protocol. The protocols can be simulated by the rewinding technique.
– The case $i = \mathsf{guess}(= i^*)$: As denoted above, we have the pair $(m_*, \sigma_*) \notin \mathcal{Q}_{\text{usign}}$. Since (m_*, σ_*) is valid, we have $\sigma_* = F_{sk}(m_*)$. Now $\mathcal{D}$ sends m_* as the challenge message, receiving the challenge τ_*, which is either $F_{sk}(m_*)$ or a random element in $\{0,1\}^\kappa$. Note that if the hidden bit is 1 then τ_* is random, and hence $\Pr[\sigma_* = \tau_*] \leq 2^{-\kappa}$, which is negligible. Therefore, if $\sigma_* = \tau_*$, $\mathcal{D}$ return 0 as its guess of the hidden bit; otherwise $\mathcal{D}$ returns 1.

Note that there is no need to consider $i > \mathsf{guess}$. Furthermore, it is easy to see that $\mathbf{Adv}^{\text{rand}}_{\mathsf{VRF}}(\mathcal{D}) = \frac{1}{n_{qr}+1} \cdot \mathbf{Adv}^{\text{forge}}_{\mathsf{SCUS}}(\mathcal{A})$, ending the proof. □

Theorem 4 *The* SCUS *construction above has invisibility. In particular, if there is a poly-time $\mathcal{D}_{\text{scus}}$ against* SCUS*, then there are poly-time $\mathcal{D}_{\text{vrf}}$ and $\mathcal{A}$ satisfying*

$$\mathbf{Adv}^{\text{inv}}_{\mathsf{SCUS}}(\mathcal{D}_{\text{scus}}) \leq \mathbf{Adv}^{\text{rand}}_{\mathsf{VRF}}(\mathcal{D}_{\text{vrf}}) + \mathbf{Adv}^{\text{forge}}_{\mathsf{SCUS}}(\mathcal{A})$$

Proof. Given $\mathcal{D}_{\text{scus}}$ against the SCUS scheme, we build a distinguisher $\mathcal{D}_{\text{vrf}}$ against the pseudo-randomness of VRF. The distinguisher $\mathcal{D}_{\text{vrf}}$, on input pk, runs $\mathcal{D}_{\text{scus}}$ and answers its queries as follows.

– Query m to $\mathsf{USign}(\cdot)$: $\mathcal{D}_{\text{vrf}}$ calls its own oracle $\mathsf{Func}_{sk}(m)$ to obtain $\sigma = F_{sk}(m)$ and $\pi = \pi_{sk}(m)$. Return $F_{sk}(m)$ as the signature. Also, add the pair (m, σ, π) to $\mathcal{Q}_{\text{usign}}$, which is an initially-empty set.
– Query (m, σ) to $\mathsf{Convert}_{sk}(\cdot, \cdot)$: If for some π, $(m, \sigma, \pi) \in \mathcal{Q}_{\text{usign}}$, then return π; otherwise, return $\perp$. The reason behind this simulation is that if $(m, \sigma, \cdot) \notin \mathcal{Q}_{\text{usign}}$, then (m, σ) must be invalid thanks to unforgeability.
– Query (m, σ) to protocol $\mathcal{P}$: Similarly to the above, if $(m, \sigma, \cdot) \in \mathcal{Q}_{\text{usign}}$, then $\mathcal{D}_{\text{vrf}}$ returns 1 and runs the confirmation protocol with $\mathcal{D}_{\text{scus}}$; otherwise it returns 0 and executes the disavowal protocol. The protocols are zero-knowledge, and then simulatable by the rewinding technique.
– Challenge query m^*: $\mathcal{D}_{\text{vrf}}$ forwards m^* to its own challenge oracle to obtain σ^* which is either $F_{sk}(m^*)$ or a random element from the range $\mathsf{Range}(\lambda)$. The element σ^* is given to $\mathcal{D}_{\text{scus}}$.

At the end, $\mathcal{D}_{\text{vrf}}$ outputs what $\mathcal{D}_{\text{scus}}$ does. It is straightforward to see that the simulation is correct except with negligible probability, and $\mathbf{Adv}^{\text{inv}}_{\mathsf{SCUS}}(\mathcal{D}_{\text{scus}}) \leq \mathbf{Adv}^{\text{rand}}_{\mathsf{VRF}}(\mathcal{D}_{\text{vrf}}) + \mathbf{Adv}^{\text{forge}}_{\mathsf{SCUS}}(\mathcal{A})$, finishing the proof. □

4.2 Concrete instantiations

In this section, we will use a pairing group $\mathbb{PG} = (G, G_T, g, q, e)$, where g is a generator of G, and the order $|G| = |G_T| = q$ for prime q, and $e : G \times G \to G_T$.

SCUS from the VRF of Dodis and Yampolskiy [9] The VRF works on a pairing group $\mathbb{PG} = (G, G_T, g, q, e)$, and is pseudo-random under the q-DBDHI assumption [24]. The secret key is $sk = s \in \mathbb{Z}_q$ and the public key is $pk = g^s$. On input $m \in \{0,1\}^{\ell_m} \subset \mathbb{Z}_q$ (in which 2^{ℓ_m} must be polynomial), define $F_s(m) = e(g,g)^{1/(m+s)}$, and $\pi = \pi_s(m) = g^{1/(m+s)}$. The function $F_s(\cdot)$ serves as a random function, and $\pi_s(\cdot)$ as the proof of its correctness. To ensure that $y(= F_s(m))$ was computed correctly, one checks

$$e(\pi_m, g^m \cdot pk) \stackrel{?}{=} e(g,g) \text{ and } e(\pi_m, g) \stackrel{?}{=} y.$$

We now show how to turn the above VRF into a SCUS scheme. The public and secret keys are the same as above. The signature on m is $\sigma = F_s(m)$, which is pseudo-random, so invisible, under the q-DBDHI assumption. For selective conversion on (m, σ), release $\pi_s(m)$.

Confirm: To show that (m, σ) is valid in the confirmation protocol, the signer with secret s proves that

$$\sigma = e(g,g)^{1/(m+s)} \iff \sigma^{m+s} = e(g,g) \iff \sigma^s = \sigma^{-m} e(g,g) \in G_T.$$

Thus the signer just proves $\big(e(g,g), e(pk,g), \sigma, \sigma^{-m}e(g,g)\big)$ is a DDH tuple in G_T, which can be realized in 4 moves as follows. Using ideas in Sect.3, the q-oneway homomorphism is $f(s) = (e(g,g)^s, \sigma^s)$, and the signer needs to prove for secret $s \in \mathbb{Z}_q$ that $f(s) = (e(pk,g), \sigma^{-m}e(g,g))$.

Disavowal: The signer needs to prove that the value $\sigma^{m+s} \cdot e(g,g) \neq 1$. The signer sets $U = (\sigma^{m+s} \cdot e(g,g))^u \in G_T$ for random $u \in \mathbb{Z}_q$ and sends U to the verifier who checks $U \neq 1$. With secrets u, s, the signer shows in zero-knowledge that

$$U = \big(\sigma^{m+s} \cdot e(g,g)\big)^u.$$

Imagine $v = us$, then what must be proven becomes, for secrets (u, v),

$$U = (\sigma^m)^u \cdot \sigma^v \cdot e(g,g)^u \quad \wedge \quad e(g,g)^v \cdot e(g,pk)^{-u} = 1,$$

which can be again realized in 4 moves by the following q-oneway homomorphism (mentioned in Sect.3)

$$\begin{aligned} f\colon \mathbb{Z}_q \times \mathbb{Z}_q &\longrightarrow G_T \times G_T \\ (u,v) \quad &\longmapsto \big((\sigma^m)^u \cdot \sigma^v \cdot e(g,g)^u, e(g,g)^v \cdot e(g,pk)^{-u}\big) \end{aligned}$$

Then, the Disavowal protocol just becomes proving $f(u,v) = (U, 1)$ for $U \neq 1$.

It is worth noting that security results given in [9] hold only if the message length ℓ_m is logarithmic in the security parameter. However, other SCUS schemes below will not suffer from the shortcoming.

SCUS from the VRF of Hohenberger and Waters [13] The VRF also works on a pairing group $\mathbb{PG} = (G, G_T, g, q, e)$, and is pseudo-random under the $\mathfrak{q}$-DDHE assumption. The secret key is $sk = (\tilde{u}, u_0, \dots, u_n)$ for $n \approx 160$ typically (or more, depending on the output of a hash function), and the public key is $pk = (\mathbb{PG}, h, \tilde{U}, U_0, \dots, U_n)$ where $\tilde{U} = g^{\tilde{u}}, U_0 = g^{u_0}, \dots, U_n = g^{u_n}$. The function $F_{sk}(m \in \{0,1\}^n)$ is

$$\sigma = F_{sk}(m) = e(g,h)^{\tilde{u}u_0 \prod_{i=1}^{n} u_i^{m[i]}},$$

for $m = m[1] \dots m[n]$, which serves as the undeniable signature on m. (For arbitrary $m \in \{0,1\}^*$, one can apply a hash function first to get an n-bit string, so that $n = 160$ typically.)

The proof of correctness $\pi_{sk}(m)$, which serves as the converter in the SCUS, consists of $\pi = (\pi_1, \dots, \pi_n, \pi_0)$, in which

$$\pi_k = g^{\tilde{u} \prod_{i=1}^{k} u_i^{m[i]}} (1 \le k \le n), \text{ and } \pi_0 = g^{\tilde{u}u_0 \prod_{i=1}^{n} u_i^{m[i]}}.$$

To verify, $\mathsf{Verify}_{pk}(m, \sigma, \pi)$ of the SCUS works step-by-step, checking

$$e(\pi_1, g) \stackrel{?}{=} \begin{cases} e(\tilde{U}, g) & \text{if } m[1] = 0 \\ e(\tilde{U}, U_1) & \text{if } m[1] = 1 \end{cases}$$

and for $2 \le i \le n$,

$$e(\pi_i, g) \stackrel{?}{=} \begin{cases} e(\pi_{i-1}, g) & \text{if } m[i] = 0 \\ e(\pi_{i-1}, U_i) & \text{if } m[i] = 1 \end{cases}$$

and finally

$$e(\pi_0, g) \stackrel{?}{=} e(\pi_n, U_0), \text{ and } e(\pi_0, h) \stackrel{?}{=} \sigma,$$

and return 1 if and only if all checks pass.

$\mathsf{Confirm}$: On common input (pk, m, σ), the signer with secret $sk = (\tilde{u}, u_0, \dots, u_n)$ proves in zero-knowledge

$$\sigma = e(g,h)^{\tilde{u}u_0 \prod_{i=1}^{n} u_i^{m[i]}},$$

which can be implemented by the generalized DDH protocol (see Sect.3) on G_T, with $e(\tilde{U}, h)$, $e(U_0, h)$, ..., $e(U_n, h)$ and σ as public elements.

$\mathsf{Disavowal}$: On common input (pk, m, σ), the signer with secret $sk = (\tilde{u}, u_0, \dots, u_n)$ proves in zero-knowledge

$$\sigma \neq e(g,h)^{\tilde{u}u_0 \prod_{i=1}^{n} u_i^{m[i]}},$$

which can be implemented using the generalized non-DDH protocol in Sect.3.

4.3 VRF from deterministic SCUS

Consider a deterministic SCUS consisting of the algorithms (KGen, USign, $\mathsf{Convert}$, Verify) and two protocols for confirmation and disavowal of signatures. The protocols are not used in the below construction of VRF. The signing space is $\{0,1\}^*$, and the signature space is $\mathsf{SigSpace}(\lambda)$.

The construction of VRF from deterministic SCUS:

– $\mathsf{Gen}(1^\lambda)$: run $\mathsf{KGen}(1^\lambda)$ of SCUS to obtain (pk, sk).

– $\mathsf{Func}_{sk}(x)$: return $y = \mathsf{USign}_{sk}(x)$, and the proof $\pi = \mathsf{Convert}_{sk}\big(x, y\big)$. By construction, $F_{sk}(\cdot) : \{0,1\}^* \to \mathsf{SigSpace}(\lambda)$.
– $\mathsf{V}_{pk}(x, y, \pi)$: return $\mathsf{Verify}_{pk}(x, y, \pi)$.

We now check the properties of the VRF. Provability is easy: if $(y, \pi) = \mathsf{Func}_{sk}(x)$, then y is a valid signature on x, and π is the converter, so $\mathsf{V}_{pk}(x, y, \pi) = 1$.

Uniqueness holds if we require USign produces only one valid signature on each message. Then, if $\mathsf{V}_{pk}(x, y_1, \pi_1) = \mathsf{V}_{pk}(x, y_2, \pi_2) = 1$ then y_1, y_2 are valid signatures on the same message x. Since there is only one valid signature on each message, we have $y_1 = y_2$.

Pseudo-randomness is ensured by the following theorem.

Theorem 5 *If* SCUS *has the property of invisibility, then* VRF *has the property of pseudo-randomness. Moreover, if* $\mathcal{D}_{\mathrm{vrf}}$ *is a distinguisher of* VRF*, then there exists* $\mathcal{D}_{\mathrm{scus}}$ *against* SCUS *such that*

$$\mathbf{Adv}^{\mathrm{rand}}_{\mathsf{VRF}}(\mathcal{D}_{\mathrm{vrf}}) \le \mathbf{Adv}^{\mathrm{inv}}_{\mathsf{SCUS}}(\mathcal{D}_{\mathrm{scus}}),$$
$$\mathbf{T}(\mathcal{D}_{\mathrm{vrf}}) \approx \mathbf{T}(\mathcal{D}_{\mathrm{scus}}),$$

where $\mathbf{T}(\cdot)$ *expresses the running time.*

Proof. Given the distinguisher $\mathcal{D}_{\mathrm{vrf}}$, we will build $\mathcal{D}_{\mathrm{scus}}$, which receives input pk and runs $\mathcal{D}_{\mathrm{vrf}}$ on that input. Recalled that, by Definition 2, $\mathcal{D}_{\mathrm{scus}}$ has access to oracles $\mathsf{USign}_{sk}(\cdot)$, and $\mathsf{Convert}_{sk}(\cdot)$. Whenever $\mathcal{D}_{\mathrm{vrf}}$ makes a query x on $\mathsf{Func}_{sk}(\cdot)$, the distinguisher $\mathcal{D}_{\mathrm{scus}}$ calls its own oracles to obtain $y = \mathsf{USign}_{sk}(x)$ and $\pi = \mathsf{Convert}_{sk}(x, y)$, and then returns (y, π) to $\mathcal{D}_{\mathrm{vrf}}$.

When $\mathcal{D}_{\mathrm{vrf}}$ submits the challenge query x^*, $\mathcal{D}_{\mathrm{scus}}$ forwards the query to its own challenge oracle to get y^*. By definition, y^* is either $\mathsf{USign}_{sk}(x^*)$ or a random element. The distinguisher $\mathcal{D}_{\mathrm{scus}}$ then returns y^* to $\mathcal{D}_{\mathrm{vrf}}$.

Finally, $\mathcal{D}_{\mathrm{scus}}$ outputs what $\mathcal{D}_{\mathrm{vrf}}$ does. It is clear that $\mathcal{D}_{\mathrm{scus}}$ succeeds with the same advantage and running time as $\mathcal{D}_{\mathrm{vrf}}$ does. □

5 A new probabilistic SCUS with neat converters and signatures

In Sect.4.2, we have seen the deterministic SCUS resulting from the VRF of Dodis and Yampolskiy [9] with small signing space. In this section, we aim at increasing the signing space to arbitrary one, while keeping the converters and signatures as short as possible. We will use the result of Boneh and Boyen [1] to build a probabilistic SCUS scheme called $\mathsf{SCUS}_{\mathsf{BB}}$ in this section.

Let us provide some intuition first. Recall that a Boneh-Boyen signature is of two elements $\Big(r, \pi = g_1^{1/(x+m+ry)}\Big)$ for random $r \in \mathbb{Z}_q$, secrets $x, y \in \mathbb{Z}_q$ and message $m \in \mathbb{Z}_q$ (for $m \in \{0,1\}^*$, just applying a collision-resistant hash to $\mathbb{Z}_q$). The element π will serve as the converter[5]. To achieve invisibility, we hide π in G_T, namely apply the pairing to compute $e(\pi, g_2)$, which will, together with r, be the undeniable signature. The confirmation and disavowal protocols can be efficiently designed thanks to the algebraic structure of the construction.

We now proceed with the concrete description of the scheme, denoted as $\mathsf{SCUS}_{\mathsf{BB}}$. To be compatible with [1], we will be more general than previous schemes by considering the pairing $e : G_1 \times G_2 \to G_T$ in which $|G_1| = |G_2| = |G_T| = q$, yet G_1 may be different from G_2.

The scheme $\mathsf{SCUS}_{\mathsf{BB}}$:

[5] Considering groups *without* pairings so the DDH assumption holds, Laguillaumie and Vergnaud [18] observed that π is pseudo-random, on which they built an undeniable scheme. The scheme, however, is not convertible. More schemes based on the Boneh-Boyen signature scheme are in Vergnaud's PhD thesis [30].

KGen: Generate the generators g_1 for G_1, and g_2 for G_2. Pick the secret key $sk = (x, y) \xleftarrow{\$} \mathbb{Z}_q^2$. The public key $pk = (u, v, g_1, g_2)$ for $u = g_2^x, v = g_2^y$.

USign$_{sk}(m)$: For a message $m \in \mathbb{Z}_q$, pick $r \xleftarrow{\$} \mathbb{Z}_q$, and return the undeniable signature $\sigma = \left(r, e(g_1, g_2)^{\frac{1}{x+m+ry}}\right)$.

Convert$_{sk}(m, \sigma)$: Parse $\sigma = (r, \rho) \in \mathbb{Z}_q \times G_T$. If $\rho = e(g_1, g_2)^{\frac{1}{x+m+ry}}$ then return $\pi = g_1^{\frac{1}{x+m+ry}}$ as the converter.

Verify$_{pk}(m, \sigma, \pi)$: Let $\sigma = (r, \rho)$. Return 1 iff $\rho = e(\pi, g_2)$ and $e(\pi, ug_2^m v^r) = e(g_1, g_2)$.

Confirm: On common input pk, m, and $\sigma = (r, \rho)$, the signer shows in ZK that $\rho = e(g_1, g_2)^{\frac{1}{x+m+ry}}$, namely $\rho^{x+m+ry} = e(g_1, g_2)$, or equivalently $\rho^x (\rho^r)^y = e(g_1, g_2)\rho^{-m}$.
Consider the q-oneway homomorphism $f(x, y) = (\rho^x \cdot (\rho^r)^y, g_2^x, g_2^y)$. The protocol is equivalent to proving $f(x, y) = (e(g_1, g_2)\rho^{-m}, u, v)$ for secret (x, y), which achieves 4 moves with full zero-knowledge.

Disavowal: The notation is as above, and the signer proves in ZK that $\rho^x(\rho^r)^y \neq e(g_1, g_2)\rho^{-m}$, namely $\rho^x(\rho^r)^y \cdot \rho^m e(g_1, g_2)^{-1} \neq 1$. The prover takes $t \xleftarrow{\$} \mathbb{Z}_q$ and sends

$$T = \left(\rho^x(\rho^r)^y \cdot \rho^m e(g_1, g_2)^{-1}\right)^t,$$

to the verifier who checks $T \neq 1$. Let $x' = xt$ and $y' = yt$, then $T = \rho^{x'}(\rho^r)^{y'}\left(\rho^m e(g_1, g_2)^{-1}\right)^t$, and $g_2^{x'} u^{-t} = 1$, $g_2^{y'} v^{-t} = 1$. The q-oneway homomorphism is as follows

$$f(x', y', t) = \left(\rho^{x'}(\rho^r)^{y'}\left(\rho^m e(g_1, g_2)^{-1}\right)^t, g_2^{x'} u^{-t}, g_2^{y'} v^{-t}\right),$$

so that the protocol becomes proving $f(x', y', t) = (T, 1, 1)$ for published $T \neq 1$, which is done in 4 moves.

To prove security of $\mathsf{SCUS}_{\mathsf{BB}}$, we need the following assumptions, which are variants of the strong Diffie-Hellman assumption [1].

Computational bilinear strong Diffie-Hellman (CBSDH). Given $g_1 \in G_1$, $g_2 \in G_2$, and $g_1^x, \ldots, g_1^{x^\ell}$, g_2^x, it's hard to compute $(c, e(g_1, g_2)^{\frac{1}{x+c}})$ for some $c \in \mathbb{Z}_q \setminus \{-x\}$.

Decisional bilinear strong Diffie-Hellman (DBSDH). Given $g_1 \in G_1$, $g_2 \in G_2$, and $g_1^x, \ldots, g_1^{x^\ell}$, g_2^x, and random $c \in \mathbb{Z}_q \setminus \{-x\}$, it's hard to distinguish $e(g_1, g_2)^{\frac{1}{x+c}}$ from a random element in G_T.

The DBSDH assumption can be shown equivalent to the decisional bilinear Diffie-Hellman inversion assumption [24], since c is fixed (see [1, Sect.3.3]). These assumptions can be evaluated in the generic group model [29].

Theorem 6 *The* $\mathsf{SCUS}_{\mathsf{BB}}$ *scheme is strongly unforgeable under the CBSDH assumption.*

Proof. Given a forger against the $\mathsf{SCUS}_{\mathsf{BB}}$ scheme, we show how to use it to break the computational bilinear SDH assumption. Let us denote m^*, $\sigma^* = (r^*, \rho^*)$ as the forgery; m_i for $1 \leq i \leq \ell$ as the signing queries (whose total number is ℓ), and $\sigma_i = (r_i, \rho_i)$ as the corresponding answers. We will consider two types of forgers:

- Forger $\mathcal{A}_1$: $[m^* + r^*y \neq m_i + r_i y$ for all $1 \leq i \leq \ell]$, or $[m_i = -x$ for some $i]$.
- Forger $\mathcal{A}_2$: $[m^* + r^*y = m_i + r_i y$ for some $i]$, and $[m_i \neq -x$ for all $i]$.

Let us begin with forger $\mathcal{A}_1$, from which we build an algorithm $\mathcal{B}_1$ solving the CBSDH assumption. The input of $\mathcal{B}_1$ is $g_1, g_1^x, \ldots, g_1^{x^\ell}, g_2, g_2^x$ and it wants to compute $\left(c, e(g_1,g_2)^{1/(x+c)}\right)$ for $c \neq -x$. The algorithm $\mathcal{B}_1$ takes $s_0, \ldots, s_\ell \xleftarrow{\$} \mathbb{Z}_q$ and computes $g_1' = g_1^{f(x)}$ in which $f(x) = s_0(x+s_1)\cdots(x+s_\ell)$. $\mathcal{B}_1$ then runs $\mathcal{A}_1$ with the public key $pk = (u = g_2^x, v = g_2^y, g_1', g_2)$ for $y \xleftarrow{\$} \mathbb{Z}_q$. The implicit secret key is (x, y) in which x is unknown to $\mathcal{B}_1$ (yet pk is clearly computable from the input of $\mathcal{B}_1$). The queries of $\mathcal{A}_1$ are processed as follows.

Signing queries: for the i-th message m_i, $\mathcal{B}_1$ sets $r_i = (s_i - m_i)y^{-1}$ so that $m_i + r_i y = s_i$. It sets $\pi_i = (g_1')^{1/(x+s_i)}$ and $\rho_i = e(\pi_i, g_2)$, then returns $\sigma_i = (r_i, \rho_i)$ to $\mathcal{A}_1$ while adding (m_i, σ_i, π_i) to an initially-empty list $\mathcal{L}$. If $m_i = -x$ for some i, then obviously $\mathcal{B}_1$ can solve the problem instance without any further action, so below we assume the harder case that $m^* + r^* y \neq m_i + r_i y$ for all $1 \le i \le \ell$.

Conversion queries (or confirmation/disavowal queries) (m, σ): Let's first consider a conversion query. If $(m, \sigma) = (m_i, \sigma_i)$ for some $(m_i, \sigma_i, \pi_i) \in \mathcal{L}$, then return π_i to $\mathcal{A}_1$; otherwise, return $\perp$. The reasoning behind this simulation is that if (m, σ) is not found on the list, and yet the pair is valid, then it serves exactly as a forgery, which can be used to solve the problem instance (as will be seen later). The problem is that it is unknown when $\mathcal{A}_1$ submits such a valid pair, and yet the technique as in Theorem 3 can be used exactly in the same way, inducing a (polynomial) loss factor in the reduction. The simulation of confirmation/disavowal queries is also along the lines of Theorem 3.

Now consider $\mathcal{A}_1$'s forgery $(m^*, \sigma^* = (r^*, \rho^*)) \notin \mathcal{L}$ in which $s^* = m^* + r^* y \neq s_i = m_i + r_i y$ for all i. Thus we can express

$$\frac{f(x)}{x+s^*} = \frac{t^*}{x+s^*} + \sum_{i=0}^{\ell-1} t_i x^i,$$

in which $t^*(\neq 0), t_0, \ldots, t_{\ell-1} \in \mathbb{Z}_q$ are known to $\mathcal{B}_1$. Then

$$\begin{aligned}\rho^* &= e(g_1', g_2)^{\frac{1}{x+s^*}} = e(g_1, g_2)^{\frac{f(x)}{x+s^*}} \\ &= \left(e(g_1,g_2)^{\frac{1}{x+s^*}}\right)^{t^*} e\left(g_1^{\sum_{i=0}^{\ell-1} t_i x^i}, g_2\right),\end{aligned}$$

enabling $\mathcal{B}_1$ to compute $e(g_1,g_2)^{\frac{1}{x+s^*}}$, and then output $\left(s^*, e(g_1,g_2)^{\frac{1}{x+s^*}}\right)$ as required.

Now consider forger $\mathcal{A}_2$, from which we build an algorithm $\mathcal{B}_2$ solving the CBSDH assumption. The input of $\mathcal{B}_2$ is $g_1, g_1^y, \ldots, g_1^{y^\ell}, g_2, g_2^y$ and it wants to compute $\left(c, e(g_1,g_2)^{1/(y+c)}\right)$ for $c \neq -y$. In fact, $\mathcal{B}_2$ below does a little more by computing y. Set $g_1' = g_1^{f(y)}$ in which $f(y) = s_0(y + s_1)\cdots(y+s_\ell)$ for now. The public key is $pk = (u = g_2^x, v = g_2^y, g_1', g_2)$ for $x \xleftarrow{\$} \mathbb{Z}_q$. The conversion and confirmation/disavowal queries are treated as before, so let us focus on the signing queries. Note that $x + m_i + r_i y = \left(y + (x+m_i)r_i^{-1}\right) r_i$. Let $r_i = (x+m_i)s_i^{-1} (\neq 0)$ so that $x + m_i + r_i y = (y+s_i)r_i$. This means

$$\rho_i = e\left((g_1')^{\frac{1}{x+m_i+r_i y}}, g_2\right) = e\left((g_1')^{\frac{1}{(y+s_i)r_i}}, g_2\right) = e\left((g_1')^{\frac{1}{(y+s_i)}}, g_2\right)^{1/r_i}$$

is computable by $\mathcal{B}_2$, so that the signature (r_i, ρ_i) can be returned to $\mathcal{A}_2$.

At the end, $\mathcal{A}_2$ produces $(m^*, \sigma^* = (r^*, \rho^*))$ in which $m^* + r^* y = m_i + r_i y$ for some $1 \le i \le \ell$. Such index i can be found by checking $g_2^{m^*} v^{r^*} \stackrel{?}{=} g_2^{m_i} v^{r_i}$ for all signing queries m_i. Also note that for such index, $(m^*, r^*) \neq (m_i, r_i)$, since otherwise $(m^*, \sigma^* = (r^*, \rho^*))$ is not a valid forgery. Thus we have $m^* - m_i = y(r_i - r^*)$ and hence y is computable by $\mathcal{B}_2$ as stated. $\square$

Theorem 7 *The* $\mathsf{SCUS_{BB}}$ *scheme is invisible under the CBSDH assumption and the DBSDH assumption.*

Proof. By Theorem 6, we know that $\mathsf{SCUS_{BB}}$ is strongly unforgeable under the CBSDH assumption. The fact will be used in this proof of invisibility. Let $\mathcal{D}$ be a distinguisher, from which we will build an algorithm $\mathcal{D}'$ solving the DBSDH assumption. $\mathcal{D}'$ is given $g_1 \in G_1$, $g_2 \in G_2$, and $g_1^x, \dots, g_1^{x^\ell}$, g_2^x, and $c \in \mathbb{Z}_q \setminus \{-x\}$, and $\mathit{\Gamma} \in G_T$, and its task is to tell whether $\mathit{\Gamma} = e(g_1, g_2)^{\frac{1}{x+c}}$ or random. As above, $\mathcal{D}'$ takes $s_0, \dots, s_\ell \xleftarrow{\$} \mathbb{Z}_q$ and computes $g_1' = g_1^{f(x)}$ in which $f(x) = s_0(x+s_1)\cdots(x+s_\ell)$. Then $\mathcal{D}'$ runs $\mathcal{D}$ with public key $pk = (u = g_2^x, v = g_2^y, g_1', g_2)$ for $y \xleftarrow{\$} \mathbb{Z}_q$, and answers the queries as follows.

Signing query $m_i (1 \le i \le \ell)$**:** set $r_i = (s_i - m_i)y^{-1}$ so that $m_i + r_i y = s_i$. Then

$$\rho_i = e\left((g_1')^{\frac{1}{x+m_i+r_i y}}, g_2\right) = e\left(\pi_i, g_2\right)$$

for $\pi_i = (g_1')^{\frac{1}{x+s_i}}$ are computable by $\mathcal{D}'$, who returns (r_i, ρ_i) and keeps (r_i, ρ_i, π_i) in an initially-empty list $\mathcal{L}$.

Conversion query (m, σ)**:** if $(m, \sigma, \pi) \in \mathcal{L}$ for some π then return π; otherwise return $\perp$. The reason is that if $(m, \sigma, \cdot) \notin \mathcal{L}$ then (m, σ) must be invalid thanks to the strong unforgeability of the scheme.

Confirmation/disavowal query (m, σ)**:** similarly to the above, if $(m, \sigma, \pi) \in \mathcal{L}$ for some π then return 1 and run the simulated protocol for confirmation. Otherwise return 0 and run the simulated protocol for disavowal.

Challenge query m^***:** let $r^* = (c - m^*)y^{-1}$ so that $m^* + r^* y = c \neq -x$. We can express

$$\frac{f(x)}{x+c} = \frac{t^*}{x+c} + \sum_{i=0}^{\ell-1} t_i x^i,$$

in which $t^*, t_0, \dots, t_{\ell-1} \in \mathbb{Z}_q$ are known to $\mathcal{D}'$. Note that

$$e\left((g_1')^{\frac{1}{x+m^*+r^*y}}, g_2\right) = e\left(g_1^{\frac{f(x)}{x+c}}, g_2\right) = e\left(g_1^{\frac{1}{x+c}}, g_2\right)^{t^*} \cdot e\left(g_1^{\sum_{i=0}^{\ell-1} t_i x^i}, g_2\right),$$

so that $\mathcal{D}'$ sets

$$\rho^* = \mathit{\Gamma}^{t^*} \cdot e\left(g_1^{\sum_{i=0}^{\ell-1} t_i x^i}, g_2\right),$$

which is random if $\mathit{\Gamma}$ is random; and together with r^*, is a valid signature if $\mathit{\Gamma} = e(g_1, g_2)^{\frac{1}{x+c}}$. Then (r^*, ρ^*) is given to $\mathcal{D}$ as the challenge.

Finally, $\mathcal{D}'$ outputs what $\mathcal{D}$ does. It is clear that if $\mathcal{D}$ succeeds in distinguishing the challenge, then $\mathcal{D}'$ can break the DBSDH assumption as desired. □

References

1. D. Boneh and X. Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *J. Cryptology*, 21(2):149–177, 2008.
2. J. Boyar, D. Chaum, I. Damgård, and T. P. Pedersen. Convertible undeniable signatures. In A. Menezes and S. A. Vanstone, editors, *CRYPTO*, volume 537 of *Lecture Notes in Computer Science*, pages 189–205. Springer, 1990.

3. C. Boyd and E. Foo. Off-line fair payment protocols using convertible signatures. In K. Ohta and D. Pei, editors, *ASIACRYPT*, volume 1514 of *Lecture Notes in Computer Science*, pages 271–285. Springer, 1998.
4. J. Camenisch and V. Shoup. Practical verifiable encryption and decryption of discrete logarithms. In D. Boneh, editor, *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 126–144. Springer, 2003.
5. D. Chaum and H. V. Antwerpen. Undeniable signatures. In G. Brassard, editor, *CRYPTO*, volume 435 of *Lecture Notes in Computer Science*, pages 212–216. Springer, 1989.
6. R. Cramer, I. Damgård, and P. D. MacKenzie. Efficient zero-knowledge proofs of knowledge without intractability assumptions. In H. Imai and Y. Zheng, editors, *Public Key Cryptography*, volume 1751 of *Lecture Notes in Computer Science*, pages 354–373. Springer, 2000.
7. I. Damgård. On Sigma protocols. Notes for Cryptographic Protocol Theory course. Available from `https://services.brics.dk/java/courseadmin/CPT/documents/getDocument/Sigma.pdf?d=33739`.
8. I. Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In B. Preneel, editor, *EUROCRYPT*, volume 1807 of *Lecture Notes in Computer Science*, pages 418–430. Springer, 2000.
9. Y. Dodis and A. Yampolskiy. A verifiable random function with short proofs and keys. In S. Vaudenay, editor, *Public Key Cryptography*, volume 3386 of *Lecture Notes in Computer Science*, pages 416–431. Springer, 2005.
10. L. El Aimani. Anonymity from public key encryption to undeniable signatures. In B. Preneel, editor, *AFRICACRYPT*, volume 5580 of *Lecture Notes in Computer Science*, pages 217–234. Springer, 2009.
11. S. D. Galbraith and W. Mao. Invisibility and anonymity of undeniable and confirmer signatures. In M. Joye, editor, *CT-RSA*, volume 2612 of *Lecture Notes in Computer Science*, pages 80–97. Springer, 2003.
12. C. Hazay and K. Nissim. Efficient set operations in the presence of malicious adversaries. In P. Q. Nguyen and D. Pointcheval, editors, *Public Key Cryptography*, volume 6056 of *Lecture Notes in Computer Science*, pages 312–331. Springer, 2010.
13. S. Hohenberger and B. Waters. Constructing verifiable random functions with large input spaces. In H. Gilbert, editor, *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 656–672. Springer, 2010.
14. S. Jarecki and V. Shmatikov. Handcuffing big brother: an abuse-resilient transaction escrow scheme. In C. Cachin and J. Camenisch, editors, *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 590–608. Springer, 2004.
15. R. Kikuchi, L. T. Phong, and W. Ogata. A framework for constructing convertible undeniable signatures. In S.-H. Heng and K. Kurosawa, editors, *ProvSec*, volume 6402 of *Lecture Notes in Computer Science*, pages 70–86. Springer, 2010.
16. K. Kurosawa and S.-H. Heng. 3-move undeniable signature scheme. In R. Cramer, editor, *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 181–197. Springer, 2005.
17. K. Kurosawa and T. Takagi. New approach for selectively convertible undeniable signature schemes. In X. Lai and K. Chen, editors, *ASIACRYPT*, volume 4284 of *Lecture Notes in Computer Science*, pages 428–443. Springer, 2006.
18. F. Laguillaumie and D. Vergnaud. Short undeniable signatures without random oracles: The missing link. In S. Maitra, C. E. V. Madhavan, and R. Venkatesan, editors, *INDOCRYPT*, volume 3797 of *Lecture Notes in Computer Science*, pages 283–296. Springer, 2005.
19. Y. Lindell. Highly-efficient universally-composable commitments based on the DDH assumption. In K. G. Paterson, editor, *EUROCRYPT*, volume 6632 of *Lecture Notes in Computer Science*, pages 446–466. Springer, 2011.
20. M. Liskov. Updatable zero-knowledge databases. In B. K. Roy, editor, *ASIACRYPT*, volume 3788 of *Lecture Notes in Computer Science*, pages 174–198. Springer, 2005.
21. S. Micali, M. O. Rabin, and S. P. Vadhan. Verifiable random functions. In *FOCS*, pages 120–130, 1999.
22. S. Micali and L. Reyzin. Soundness in the public-key model. In J. Kilian, editor, *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 542–565. Springer, 2001.
23. S. Micali and R. L. Rivest. Micropayments revisited. In B. Preneel, editor, *CT-RSA*, volume 2271 of *Lecture Notes in Computer Science*, pages 149–163. Springer, 2002.
24. S. Mitsunari, R. Sakai, and M. Kasahara. A new traitor tracing. *IEICE Trans. Fundamentals*, E85-A(2):481–484, 2002.
25. C. A. Neff. A verifiable secret shuffle and its application to e-voting. In *ACM Conference on Computer and Communications Security*, pages 116–125, 2001.
26. L. T. Phong, K. Kurosawa, and W. Ogata. Provably secure convertible undeniable signatures with unambiguity. In J. A. Garay and R. D. Prisco, editors, *SCN*, volume 6280 of *Lecture Notes in Computer Science*, pages 291–308. Springer, 2010.
27. C.-P. Schnorr. Efficient signature generation by smart cards. *J. Cryptology*, 4(3):161–174, 1991.
28. J. C. N. Schuldt and K. Matsuura. An efficient convertible undeniable signature scheme with delegatable verification. In J. Kwak, R. H. Deng, Y. Won, and G. Wang, editors, *ISPEC*, volume 6047 of *Lecture Notes in Computer Science*, pages 276–293. Springer, 2010.

29. V. Shoup. Lower bounds for discrete logarithms and related problems. In *EUROCRYPT*, pages 256–266, 1997.
30. D. Vergnaud. Approximation diophantienne et courbes elliptiques. Protocoles asymétriques d'authentification non-transférable. PhD Thesis. Available from `http://www.di.ens.fr/~vergnaud/TheseVergnaud.ps.gz`.

A Full zero-knowledgeness from Σ-protocol

Sequential zero-knowledgeness. Following Damgård [7], we now show how to turn the above Σ-protocol proving $f(\omega) = x$ into a full-fledged zero-knowledge one proving the same statement, which is presented below. Recall that x is the common input.

1. The verifier takes $\omega_V \overset{\$}{\leftarrow} \mathbb{Z}_q$ and sends $x_V = f(\omega_V)$ to the prover.
2. The verifier then proves the knowledge of ω_V using the above Σ-protocol.
3. The prover proves the knowledge of ω satisfying $f(\omega) = x \vee f(\omega) = x_V$. This is exactly the standard OR-proof, whose details can be found in [7].

By merging moves, the above is of 4 moves. To show full-fledged zero-knowledgeness of the protocol, let us consider an arbitrary verifier V. Rewinding the verifier, from step 2, we can extract the witness ω_V satisfying $x_V = f(\omega_V)$. This witness is then utilized in the OR proof at step 3 to complete the protocol.

To show soundness, consider step 3. Rewinding the prover, we can extract ω^* satisfying $f(\omega^*) = x \vee f(\omega^*) = x_V$. If the equation $f(\omega^*) = x_V$ holds, then we can use the prover to attack the onewayness of f. Thus $f(\omega^*) = x$ with all but negligible probability, and the soundness follows.

Concurrent zero-knowledgeness. We will mainly assume that the zero-knowledge protocols are run sequentially in this paper. However, if concurrency becomes a concern, then the technique of Damgård [8] can be used to turn the Σ-protocols into concurrent zero-knowledge ones. The trade-off is that we have to use a publicly-available auxiliary string. The transformation is simple enough to be recalled here. Let $(\mathsf{Kg}, \mathsf{Com}, \mathsf{Decom})$ be a trapdoor commitment scheme, in which Kg generates a trapdoor t, and a public key pk. The public key pk will be the auxiliary string, available to both the prover and the verifier. With identical notations as the above Σ-protocol from q-oneway homomorphism, consider the following protocol proving the knowledge of ω satisfying $f(\omega) = x$.

1. The prover chooses $\omega' \overset{\$}{\leftarrow} \mathsf{Dom}$, and computes $x' = f(\omega')$. It then sends the commitment $u = \mathsf{Com}_{pk}(x'; r)$ to the verifier.
2. The verifier sends back a random challenge $c \in \{0, \ldots, q-1\}$.
3. The prover returns x', r and $\omega'' = \omega' + c\omega$ to the verifier who checks $f(\omega'') = x'x^c$ and $u = \mathsf{Com}_{pk}(x'; r)$.

To show concurrent zero-knowledgeness of the protocol, consider an arbitrary verifier V^*, and we will show how to simulate it without any rewinding. The simulator works as follows.

– It generates (t, pk) using Kg of the commitment scheme, and gives pk and x to V^*. At the same time, it sends a random $u = \mathsf{Com}_{pk}(x'_{\text{fake}}; r_{\text{fake}})$ to V^*.
– V^* sends back a challenge $c \in \{0, \ldots, q-1\}$.
– The simulator takes $\omega'' \overset{\$}{\leftarrow} \mathbb{Z}_q$ and sets $x' = f(\omega'')x^{-c}$. Using the trapdoor t, it finds r satisfying $u = \mathsf{Com}_{pk}(x'; r)$, and then returns x', r, and ω'' to V^*. It is clear that $f(\omega'') = x'x^c$ and $u = \mathsf{Com}_{pk}(x'; r)$ and hence V^* accepts (while the simulator does not hold the witness ω).

Universally composable zero-knowledgeness. Following [12], we can turn the basis Σ-protocol into a UC-secure one. To do so, let us restrict the challenge space to $\{0, 1\}$. This will allows us to *not* rewind the prover and the verifier, albeit yielding $1/2$ soundness error. Repeating the protocol L times will result in a protocol with 2^{-L} soundness error. Let us just provide some intuition below. For details, see [12].

1. The prover takes $\omega' \overset{\$}{\leftarrow} \mathsf{Dom}$ and lets $x' = f(\omega')$. It also computes the answers $\omega''_i = \omega' + i\omega$ for $i \in \{0,1\}$. The prover then commits $x', \omega''_0, \omega''_1$ to the verifier via a UC-secure commitment scheme (e.g. [19]).
2. The verifier sends back a challenge $c \overset{\$}{\leftarrow} \{0,1\}$.
3. The prover decommits x', ω''_c. The verifier then checks $f(\omega''_c) = x'x^c$ as in the basic Σ-protocol.

Consider a corrupted prover. Opening its commitments, the simulator gets $(x', 0, \omega''_0)$ and $(x', 1, \omega''_1)$, from which the witness can be extracted with probability $1/2$.

Now consider a corrupted verifier. The simulator sends a random message for the first step. After receiving $c \in \{0,1\}$ from the verifier, it takes $\omega''_c \overset{\$}{\leftarrow} \mathbb{Z}_q$ and lets $x' = f(\omega'')x^{-c}$. The simulator also opens the commitments according to these values.