論文 / 著書情報
Article / Book Information

題目(和文)	完全構造保持署名と耐不可逆漏洩署名に関する構成
Title(English)	Constructions for Fully Structure-Preserving Signature and Uninvertible Leakage Resilient Signature
著者(和文)	王 煜宇
Author(English)	Yuyu Wang
出典(和文)	学位:博士(理学), 学位授与機関:東京工業大学, 報告番号:甲第10754号, 授与年月日:2018年3月26日, 学位の種別:課程博士, 審査員:田中 圭介,伊東 利哉,尾形 わかは,渡辺 治,鹿島 亮
Citation(English)	Degree:Doctor (Science), Conferring organization: Tokyo Institute of Technology, Report number:甲第10754号, Conferred date:2018/3/26, Degree Type:Course doctor, Examiner:,,,,
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

Constructions for Fully Structure-Preserving Signature and Uninvertible Leakage Resilient Signature

Yuyu Wang
Supervisor: Keisuke Tanaka

Department of Mathematical and Computing Sciences
Tokyo Institute of Technology

February 15, 2018

Abstract

In this thesis, we focus on fully structure-preserving signatures (FSPSs), which play important roles in many efficient modular protocols, and signatures resilient to uninvertible leakage, which provide strong security guarantee against side-channel attacks. Concretely, we achieve the following results.

Firstly, we propose a general way to obtain FSPSs. More specifically, we bridge the gap between standard structure-preserving signatures (SPSs), which have already been widely studied in prior works, and FSPSs. In FSPSs, all the messages, signatures, verification keys, and signing keys consist only of group elements, while in SPSs, signing keys are not required to be a collection of group elements. To achieve our goal, we introduce two new primitives called trapdoor signature and signature with auxiliary key, both of which can be derived from SPSs. By carefully combining both primitives, we obtain generic constructions of FSPS from SPSs. Upon instantiating the above two primitives, we get many instantiations of FSPS with unilateral and bilateral message spaces. Different from previously proposed FSPSs, many of our instantiations also have the automorphic property, which enables a signer to sign his own verification key. As by-product results, one of our instantiations has the shortest verification key size, signature size, and lowest verification cost among all previous constructions based on standard assumptions, and one of them is the first FSPS scheme in the type I bilinear group.

Then we propose a fully leakage resilient signature scheme in the selective auxiliary input model, which captures an extremely wide class of side-channel attacks that are based on physical implementations of algorithms rather than public parameters chosen. Our signature scheme remains existential unforgeable under chosen message attacks as long as the adversary cannot completely recover the entire secret state from leakage in polynomial time with non-negligible probability. Formally speaking, the leakage is allowed to be any computable uninvertible function on input the secret state, without any additional restriction. We instantiate such a signature scheme by exploiting a point-function obfuscator with auxiliary input (AIPO) and a differing-inputs obfuscator (diO). As far as we know, this is the first signature scheme secure against uninvertible leakage. Furthermore, our signature scheme is public-coin, in the sense that the randomness used in the signing procedure is a part of a signature and no additional secret randomness is used. Additionally, we provide a variant of the above signature scheme, for which leakage functions are additionally required to be injective, and the sizes of the circuits representing leakage functions are upper bounded. This scheme is resilient to uninvertible leakage that information-theoretically determines the

secret information, and can be constructed based only on diO, without exploiting AIPO.

Contents

1 Introduction **5**
- 1.1 Background . . . 5
  - 1.1.1 (Fully) Structure-Preserving Signatures . . . 5
  - 1.1.2 Leakage Resilient Signatures . . . 7
- 1.2 Our Results . . . 10
  - 1.2.1 Constructions of Fully Structure-Preserving (Automorphic) Signature . . . 10
  - 1.2.2 Constructions of Signature Resilient to Uninvertible Leakage . . . 13
- 1.3 Outline of This Paper . . . 17

2 Preliminaries **18**
- 2.1 Notations . . . 18
- 2.2 Pairing Group . . . 18
- 2.3 SXDH and $\mathcal{D}_k$-MDDH Assumptions . . . 19
- 2.4 One-way Function and Uninvertible Function . . . 20
- 2.5 Obfuscation . . . 20
- 2.6 Puncturable Pseudorandom Function . . . 22
- 2.7 Digital Signature . . . 22

3 Generic Constructions of Fully Structure-Preserving Signature **24**
- 3.1 (Fully) Structure-Preserving Signature . . . 24
- 3.2 Trapdoor Signature . . . 26
  - 3.2.1 Definition of Trapdoor Signature . . . 26
  - 3.2.2 Security of Trapdoor Signatures . . . 27
  - 3.2.3 Converting Structure-Preserving Signatures into Signing Key Structure-Preserving Trapdoor Signatures . . . 29
  - 3.2.4 Instantiations of Trapdoor Signature . . . 32
- 3.3 (Two-tier) Signature with Auxiliary Key(s) . . . 34
  - 3.3.1 Signature with Auxiliary Key . . . 35
  - 3.3.2 Two-tier Signature with Auxiliary Keys . . . 37
- 3.4 Generic Constructions of Fully Structure-Preserving Signature (and Fully Automorphic Signature) . . . 41
  - 3.4.1 Generic Construction Sig_1: Trapdoor Signature + Signature with Auxiliary Key . . . 41

3.4.2 Variation of Sig_1: Trapdoor Signature + Signature with Auxiliary Key (UF-CMA) . . . 48
3.4.3 Generic Construction Sig_2: Trapdoor Signature + Two-tier Signature with Auxiliary Keys . . . 49
3.4.4 Generic Construction Sig_3 (UF-RMA): Trapdoor Signature + Binding Trapdoor Commitment . . . 58
3.5 Instantiations of UF-CMA Secure FSPS (FAS) . . . 64
3.5.1 Sig_1: SKSP-TS + SP-AKS . . . 64
3.5.2 Sig_1^*: SKSP-TS + SP-AKS (UF-CMA) . . . 66
3.5.3 Sig_2: SKSP-TS + SP-TT-AKS . . . 66
3.6 Instantiations of One-time FSPS (FAS) . . . 67
3.7 Efficient Instantiations of FSPS and FAS Based on the $\mathcal{SC}_k$-MDDH Assumptions 68
3.7.1 Instantiation: UF-CMA Secure SKSP-TS + UF-otCMA Secure SP-AKS 68
3.7.2 Instantiation: UF-otRMA Secure SKSP-TS + UF-otCMA Secure SP-AKS . . . 68
3.7.3 Instantiation: UF-CMA Secure SKSP-TS + UF-TT-CMA Secure SP-TT-AKS . . . 68
3.7.4 Instantiation: UF-otRMA Secure SKSP-TS + UF-TT-CMA Secure SP-TT-AKS . . . 70
3.8 Signing Key Sizes . . . 72
3.9 Number of Pairings . . . 72

4 Signature Resilient to Uninvertible Leakage 74
4.1 Fully Leakage Resilient Signature in the Selective Auxiliary Input Model . . . 74
4.2 Uninvertible Leakage Resilient Hard Relation . . . 75
4.2.1 Definitions . . . 75
4.2.2 Constructions . . . 76
4.3 Constructions of Fully Leakage Resilient Signature in the Selective Auxiliary Input Model . . . 79
4.3.1 Fully Leakage Resilient Signature Scheme . . . 79
4.3.2 Weak Fully Leakage Resilient Signature Scheme . . . 92
4.3.3 Remarks on Our Constructions . . . 92

5 Conclusion and Open Problems 94

6 Acknowledgement 95

Chapter 1

Introduction

1.1 Background

Digital signatures are fundamental cryptographic primitives that give receivers the reason to believe that messages are admitted by claimed senders. Specifically, by using a signature scheme, senders can sign messages by using secret keys only known by themselves, and receivers can verify the validity of the signatures by using the corresponding public information. Roughly, the security of signature schemes ensures that an adversary who does not know secret keys cannot forge signatures. They are used as building blocks in many cryptographic protocols, and also exploited in many digital services where authenticity of digital messages is necessary, such as smart cards, ID cards, digital transactions, and digital contracts. Therefore, signatures compatible with other cryptographic elements and ones that can tolerate more powerful attacks in the real world are desirable.

1.1.1 (Fully) Structure-Preserving Signatures

Structure-preserving signatures (SPSs). To ensure conceptual simplicity, cryptographic protocols are usually expected to be designed in modular ways. However, when combining signatures with other building blocks such as encryption schemes and zero knowledge proof systems, resulting protocols may lose efficiency due to the difference between their structures. To overcome this problem, in [5], Abe et al. initiated the study of SPSs which denote pairing-based signatures where all the verification keys, messages, and signatures consist only of group elements and the verification algorithms only make use of pairing product equations (PPEs) to verify signatures. SPSs retain reasonable efficiency when combined with other structure-preserving (SP) primitives (e.g., ElGamal encryption [49] and Groth-Sahai proofs [70]), which results in efficient cryptographic protocols such as blind signatures [5, 56, 55, 54], group signatures [5, 56, 86], homomorphic signatures [85], delegatable anonymous credentials [53], compact verifiable shuffles [44], network coding [15], oblivious transfer [98, 38], tightly secure encryption [73, 4], and e-cash [16]. Following [5], there have been a large deal of works focusing on SPSs (e.g., [5, 3, 73, 6, 7, 4, 63, 9, 8, 55, 69, 64, 66, 65])

in the past few years, which provide us with various SPS schemes based on different assumptions and with high efficiency.

Automorphic signatures. In [5], Abe et al. noted that for elaborate applications, the SP property of a signature scheme is not sufficient. In addition, an SPS scheme has to be able to sign its own verification keys, i.e., verification keys have to lie in the message space. They called such kind of SPS automorphic signature and gave an instantiation of it, and also provided a generic transformation that converts automorphic signatures for messages of fixed length into ones for messages of arbitrary length.

As argued in [5], since automorphic signatures enable constructions of certification chain (i.e., sequences of verification keys linked by certificates from one key on the next one), they are useful in constructing anonymous proxy signatures and delegatable anonymous credentials. Abe et al. [5] also showed how to combine automorphic signatures with the Groth-Sahai proof system to construct a round-optimal blind signature scheme.

Fully structure-preserving signatures (FSPSs). In [10], Abe et al. introduced FSPS, where signing keys also consist only of group elements and the correctness of signing keys w.r.t. verification keys can be verified by PPEs. Since the fully structure-preserving (FSP) property enables efficient signing key extraction, it could help us prevent rogue-key attacks in the public-key infrastructures (PKIs) [90], make anonymous credentials UC-secure [40], achieve privacy in group and ring signatures [21, 22, 27] in the presence of adversarial keys, and extend delegatable anonymous credentials [18, 53, 45] with all-or-nothing transferability [41], as noted in [10]. Camenisch et al. [39] also showed that FSPSs help construct unlinkable redactable signatures. In this thesis, we call an automorphic signature scheme that is FSP a fully automorphic signature (FAS) scheme.

Abe et al. [10] gave two generic constructions by combining FSPSs unforgeable (UF) against extended random message attacks (xRMA) [3] with other primitives such as one-time SPSs, two-tier SPSs (also called partial one-time SPSs), and trapdoor commitment schemes. Although these constructions are novel and neat, they suffer from three shortcomings due to the use of specific primitives, which make them less generic.

1. As both constructions require a UF-xRMA secure FSPS scheme and one of them also requires a γ-blinding trapdoor commitment scheme, the underlying assumptions and bilinear map of their instantiations are limited. Concretely speaking, all the signature schemes derived from their constructions have to be based on at least the Symmetric External Diffie-Hellman (SXDH) and External Decision Linear (XDLIN) assumptions and be in the type III bilinear group.

2. For the same reason, the efficiency of their instantiations is also potentially limited by the underlying UF-xRMA secure FSPS scheme and the γ-blinding trapdoor commitment scheme. For example, the verification keys and signatures of their most efficient FSPS scheme consist of more than $10n$ group elements in total if messages consist of n^2 group elements.

3. Their instantiations are not automorphic. The reason is that verification keys of the UF-xRMA secure FSPS scheme (which are also verification keys of the resulting schemes) consist of elements in both source groups, while the resulting signature schemes can only sign messages consisting only of elements in one source group.

Note that Abe et al. [10] also gave a variant of their constructions by combining a UF-xRMA secure signature scheme and a trapdoor commitment scheme with SPSs, which can be treated as a generic transformation from SPSs to FSPSs. If the instantiation of SPS is with a bilateral message space (i.e., messages consist of elements in both source groups), then the resulting signature scheme could be automorphic. However, as far as we know, besides the aforementioned shortcomings, all the previously proposed SPS schemes with a bilateral message space require verification keys to consist of elements in both source groups (except for ones that sign messages of "DDH form" [63, 64, 66, 65]), which result in very inefficient FSPS schemes, as noted in [10]. The verification keys and signatures (respectively, the verification algorithm) of the most efficient automorphic instantiation that can be derived from their generic construction consist of more than $12n$ group elements in total (respectively, more than $3n$ PPEs) if messages consist of $2n^2$ group elements.

Following the work of Abe et al. [10], Groth [69] gave an elegant construction of FSPS, which has the shortest verification keys and signatures, and needs the fewest PPEs for verification. Although this FSPS scheme is the most efficient one as far as we know, it is only known to be secure in the generic group model and is not automorphic.

Up until now, a lot of results are devoted to constructing efficient SPSs under different assumptions, while there are very few FSPS schemes. If we can find a generic method to transform existing SPSs into FSPSs or even FASs without directly using specific primitives, it will greatly alleviate the efforts to construct them from scratch.

1.1.2 Leakage Resilient Signatures

Leakage resilient (LR) primitives. A cryptographic primitive is usually proved to be secure in the attack models where intermediate values, e.g., secret keys (or signing keys in the case of signatures) and randomizers used to encrypt or sign messages, are assumed to be completely hidden. However, it is becoming more and more unrealistic to rule out the possibility that an adversary learns leakage on secret information (including secret keys and secret randomizers) from the physical implementation of algorithms by executing the side-channel attacks with low cost (e.g. [81, 14, 28, 30, 82, 95, 57, 92, 71]).

Motivated by this scenario, Akavia et al. [11] introduced the bounded leakage model, in which a primitive is said to be LR if it is secure against an adversary who may learn partial information of the secret key. The leakage is denoted as $f(sk)$ where sk is the secret key, and f can be any efficiently computable function as long as the number of output bits of f is not larger than the leakage parameter ℓ. In [78], Katz and Vaikuntanathan introduced the notion of full leakage resilience (FLR), which is a stronger security notion against the adversary who may learn leakage on not only the secret key, but also the intermediate values during the whole lifetime of a signature scheme. It is obvious that ℓ must be smaller than

the length of the secret key. Otherwise, an adversary can easily break a system by letting a leakage function output the whole secret key. As an extension of the bounded leakage model, Dodis et al. [47] and Brakerski et al. [36] suggested the continual leakage model, which is the same as the bounded leakage model except that it requires the system to be able to update the secret key periodically without changing the public key. Another model called the noisy leakage model, which can be treated as a generalization of the bounded leakage model, was proposed by Naor and Segev [91]. In the noisy leakage model, there is no bound on the number of leaked bits. It is only required that the secret key keeps some min-entropy, given leakage.

Although these models are well defined, all of them assume that partial information of the secret key is information-theoretically hidden, while the leakage information-theoretically determines the secret information (including the secret key and other secret randomness) typically in the practical world [104]. Intrigued by this fact, Dodis et al. [48] initialized the research in the auxiliary input model (also called the hard-to-invert leakage model), in which, it is only assumed that it is hard to recover the secret key from the leakage, i.e., the secret key may be information-theoretically revealed by the leakage. There are several researches focusing on encryption in the auxiliary input model [46, 68, 35, 109, 111, 108], while proposing signatures in such a model seemed to be hopeless. For signatures in the auxiliary input model, a leakage function could be of the form $f(\cdot) = \mathsf{Sign}(pk, \cdot, m^*; r)$, which is the signing algorithm for a challenge message m^*. It is obvious that in this case, the leakage obtained by the adversary, which is $f(sk) = \mathsf{Sign}(pk, sk, m^*; r)$, is itself a successfully forged signature on m^*.

However, since signatures play a very important role in public-key cryptography and previously proposed LR signatures do not capture a large class of side channel attacks, defining and constructing signatures in the auxiliary input model have remained an important and practically-motivated problem. To avoid the aforementioned trivial attack, the followup works define the auxiliary input model for signatures by making some restrictions.

The auxiliary input model for signatures. Faust et al. [52] firstly defined LR signature in the auxiliary input model. The restriction they made is that the leakage should be exponentially hard-to-invert rather than polynomially hard-to-invert. The signature schemes they provided were not FLR since they only considered leakage on signing keys.

More specifically, the leakage in their work is denoted as $f(pk, sk)$, which is given to the adversary along with pk at the beginning of the security game, where f is the leakage function and (pk, sk) is the verification/signing key pair. To formalize the attack model, they followed [46] to define two classes of leakage functions. For a function f in the first class, it is required that given $(pk, f(pk, sk))$, it is hard to compute sk, while in the second class, the requirement is that it is hard to compute sk given only $f(pk, sk)$. They proposed two signature schemes. The first one is unforgeable against random message attacks (UF-RMA), and resilient to polynomially hard-to-invert leakage w.r.t. the first class of leakage functions and exponentially hard-to-invert leakage w.r.t. the second class. The second one is unforgeable against chosen message attacks (UF-CMA), and resilient to exponentially

hard-to-invert leakage w.r.t. both classes of leakage functions.

The selective auxiliary input model for signatures. Independently of the work of Faust et al., Yuen et al. [110] defined the selective auxiliary input model. This model avoids the aforementioned trivial attack by letting the adversary choose candidates of leakage functions before seeing the verification key. The signature scheme they gave is FLR and the leakage is allowed to be polynomially hard-to-invert.

The selective auxiliary input model is reasonably defined since it captures the implementation-based side-channel attacks which help an adversary learn leakage on the secret information independently of the public parameters chosen in the system [110] (e.g., the power analysis of the CPU). Furthermore, a signature scheme secure in this model can be typically proved to be secure in the model of [52] by making use of complexity leveraging.

However, the restriction made on the class of leakage functions in [110] is very strong. Roughly speaking, the leakage function f should satisfy $\Pr[sk \leftarrow \mathcal{A}(f(state), pk, \mathcal{S})] \approx 0$ in [110], where $\mathcal{A}$ denotes any adversary, (pk, sk) a randomly generated verification/signing key pair, $\mathcal{S}$ the set of signatures obtained from the signing oracle, and $state$ the secret state (including sk and the secret randomizers used to generate $\mathcal{S}$).

Our point of view that their restriction is too strong lies in: (a) By making this restriction, they ruled out the possibility that $\mathcal{A}$ may recover sk from the leakage in the presence of pk and $\mathcal{S}$, which in turn makes obtaining a secure signature scheme in this model much easier. (b) Hardness of recovering the secret state should not bypass the hardness of recovering the signing key itself (i.e., it is more practical to assume that it is hard for $\mathcal{A}$ to recover $state$ rather than sk).

Another signature scheme secure in the selective auxiliary input model was proposed by Yuen et al. [111] by exploiting the Goldreich-Levin randomness extractor [67], while this tool was also used by Yu et al. [108] to achieve a chosen-ciphertext public-key encryption scheme secure in the presence of hard-to-invert leakage. This method is based on the well-known fact that given $f(x)$ where f is hard-to-invert, the hard-core bit string is indistinguishable from randomness. Therefore, by making use of the secret key x, intermediate values can be generated by computing the hard-core bit string $h(x)$ instead of choosing the real randomness, while $f(x)$ can be learnt by the adversary as the leakage. However, f must be exponentially hard-to-invert or restricted in other ways. What is more, the more hard-core bits are generated, the more restrictions have to be applied to f. The only known randomness extractor that can provide poly-many hardcore bits for any one-way function was proposed by Bellare et al. [25], based on iO and diO. However, since their construction of hard-core bits generator depends on the one-way function, it cannot be used as a building block of LR primitives.

Signatures secure against uninvertible leakage. It is a natural question to ask if it is possible to construct a signature scheme in the selective auxiliary input model where the restriction on the class of leakage functions is extremely weak, especially when leakage functions are only required to be uninvertible. Note that in this case, the leakage function f is only required to satisfy $\Pr[state \leftarrow \mathcal{A}(f(state))] \approx 0$ rather than $\Pr[sk \leftarrow \mathcal{A}(f(state), pk, \mathcal{S})] \approx 0$.

It is obvious that such a signature scheme is secure against much wider class of side-channel attacks, compared with the one proposed in [110].

1.2 Our Results

1.2.1 Constructions of Fully Structure-Preserving (Automorphic) Signature

Generic construction of FSPS. In this thesis, we formalize two extensions to ordinary signatures called trapdoor signatures (TSs) and signatures with auxiliary key (AKSs). We show that any well-formed[1] SPS scheme can be converted into a TS scheme satisfying the signing key structure-preserving (SKSP) property, in which signing keys consist only of group elements and the correctness w.r.t. verification keys can be verified by PPEs, while messages are not necessarily group elements. Furthermore, it is relatively straightforward to show that any SPS scheme with an algebraic key generation algorithm can be converted into a structure-preserving signature with auxiliary keys (SP-AKS). By combining SKSP-TSs with SP-AKSs, we obtain a generic construction of FSPS.[2] Our construction implies that for any two SPS schemes, if verification keys of one lie in the message space of the other (which is well-formed), then basically, they can be used to construct an FSPS scheme, without using any other specific primitives or additional assumptions. It also implies that most well-formed SPS schemes with a bilateral message space or unilateral verification key space (i.e., the verification keys consist only of elements in one source group) can be converted into an FSPS scheme.

This generic construction is proved to be secure based on building blocks satisfying different security, which allows us to obtain various instantiations of FSPS based on different assumptions.

Efficient instantiations of FSPS. By extending the definition of AKS to two-tier signature with auxiliary key (TT-AKS) and substituting AKSs with TT-AKSs in the above generic construction, we obtain another generic construction, which enables us to obtain more efficient instantiations of FSPS. For instance, by using the TS scheme and TT-AKS scheme adapted from the SPS schemes proposed by Kiltz et al. [79, 80], we obtain instantiations of FSPS with unilateral and bilateral message spaces. We give an efficiency comparison between our instantiations and the ones proposed in [10] in Table 1.1.[3] Note that like the

[1] We refer the reader to Definition 3.2.5 for details of well-formed SPSs. As far as we know, all the existing SPS schemes are well-formed.

[2] As in [10], we assume the underlying SKSP-TS scheme and SP-AKS scheme share the common setup algorithm.

[3] The second instantiation in Table 1.1 is derived from the generic construction described in [10, Section 6.4], where the underlying SPS scheme is the one with bilateral message space in [79] (based on the SXDH assumption), and the parameters are computed following the equations in [10, Section 6.4]. In this instantiation, we have to add a group element denoting the sequence number to every message block. Furthermore, the underlying two-tier signature schemes of the first and third instantiations have the same efficiency, which

FSPS scheme proposed in [69], a signing key in our instantiations consists of $\Omega(n)$ group elements (concretely, $2n+1$ in [69] and $4n+9$ and $8n+13$ in our results), while that in "AKO+15" consists only of 4 elements. However, in many applications, the size of a signing key does not have to be "extremely short" since typically, a user generates only one proof for knowing a signing key (e.g., in PKIs and group/ring signatures), while proofs for knowing a signature or a verification key/signature pair are required to be generated for multiple times.[4]

	Security	Assumption	$\|m\|$	$\|pk\|+\|par\|$	$\|\sigma\|$	$\sharp$ PPE
AKO+15 [10]	Full	SXDH, XDLIN	$(n^2,0)$	$6n+17$	$4n+11$	$n+5$
	Full	SXDH, XDLIN	(n^2,n^2)	$6n+47$	$13n+30$	$5n+6$
Our results	Full	SXDH	$(n^2,0)$	$2n+7$	$4n+8$	$n+3$
	Full	SXDH	(n^2,n^2)	$4n+10$	$8n+12$	$2n+4$

Table 1.1: Comparison between the most efficient instantiations of FSPS based on standard assumptions derived from the main construction in [10] and the most efficient ones derived from our constructions. Notation (x,y) denotes x elements in $\mathbb{G}_1$ and y elements in $\mathbb{G}_2$. We do not count the two generators in the description of bilinear groups when giving the parameters.

Our FSPS schemes in Table 1.1 can also be based on the $\mathcal{D}_k$-matrix Diffie-Hellman (MDDH) assumptions [50] (ref., Section 2.3), while the parameters of the first scheme become $(|m|,|pk|+|par|,|\sigma|,\sharp\text{PPE}) = (n^2,(2nk+2k+3+\mathsf{RE}(\mathcal{D}_k))k+\mathsf{RE}(\mathcal{D}_k),(3k+1)n+4+3k+\mathsf{RE}(\mathcal{D}_k),kn+2k+1)$ and those of the second scheme become $(|m|,|pk|+|par|,|\sigma|,\sharp\text{PPE}) = (2n^2,(4nk+3k+3+2\mathsf{RE}(\mathcal{D}_k))k+2\mathsf{RE}(\mathcal{D})_k,2(3k+1)n+5k+5+2\mathsf{RE}(\mathcal{D}_k),2kn+3k+1)$, where $\mathsf{RE}(\mathcal{D}_k)$ denotes the minimal number of group elements needed to present a matrix sampled from $\mathcal{D}_k$ (see Section 3.5 for more details).

Since our constructions only require the underlying schemes to have properties naturally satisfied by SPSs, further improvement on SPS schemes will contribute to the efficiency of FSPSs more via our constructions than the constructions in [10]. Recently, Jutla and Roy [77] improved the efficiency of the SPS scheme in [79]. By adopting their instantiation, we can respectively reduce the signature size and the number of PPEs of our first result in Table 1.1 by 1. The same argument is made for all the other instantiations derived from the UF-CMA secure FSPS scheme with unilateral message space by Kiltz et al. [79] in this thesis.

makes sure that this comparison is fair. If we allow trusted setup besides the bilinear map generation, the sizes of common parameters $|par|$ in these four schemes are 6, 6, 1, and 2 respectively.

[4]The argument that the signing key size is not as important as verification/signature size does not spoil the motivation for FSPS. FSPS helps avoid extremely heavy key extraction, i.e., extracting a signing key bit by bit (see Introduction in [10]). However, this does not mean we have to make the extraction extremely light. Allowing checking signing keys by using PPEs and keeping the key size linear with message size are enough to achieve the goal.

FASs. Since we can convert any (well-formed) SPS scheme into an SKSP-TS scheme and an SP-AKS scheme, our generic constructions also derive many instantiations of FAS from various combinations (including the ones in Table 1.1). As long as verification keys of the underlying TS scheme consist of no more group elements than messages of the underlying AKS scheme in both source groups, the resulting scheme is usually fully automorphic.

We can instantiate our first generic construction with the TS scheme and AKS scheme adapted from the SPS scheme proposed by Groth et al. [69] to obtain our most efficient FAS scheme, while the most efficient one from the generic construction in [10] can be obtained by letting the underlying SPS scheme be the one in [6] and the underlying one-time SPS scheme the one in [66].[5] For ease of understanding, we give an efficiency comparison in Table 1.2.

	Security	Assumption	$\|m\|$	$\|pk\| + \|par\|$	$\|\sigma\|$	$\sharp$ PPE
`AKO+15` [10]	Full	Generic	(n^2, n^2)	$6n + 23$	$6n + 14$	$3n + 6$
Our result	Full	Generic	$(n^2, 0)$	$2n + 1$	$2n + 5$	$n + 3$

Table 1.2: Comparison between the most efficient instantiation of FAS derived from the main construction in [10] and the most efficient one derived from our constructions. Both of them are secure in the generic group model.

FSPS (FAS) schemes in the symmetric (type I) bilinear map. We also instantiate our generic constructions with the SPS scheme and the tag-based SPS scheme proposed in [4] to obtain the first FSPS and FAS schemes in the type I bilinear map, the most efficient one of which achieves $(|m|, |pk| + |par|, |\sigma|, \sharp\text{PPE}) = (n^2, 6n + 30, 6n + 12, 2n + 7)$.

UF-RMA secure FSPS scheme with short sigantures. Besides the above generic constructions, we give a generic construction of UF-RMA secure FSPS, which combines TSs with binding trapdoor commitments (BTCs) [10]. By instantiating the underlying TS scheme with the UF-CMA secure SPS scheme in [79], and the underlying BTC scheme with the one in [10], this generic construction achieves $(|m|, |pk| + |par|, |\sigma|, \sharp\text{PPE}) = (n^2, 2n + 6, 3n + 7, n + 3)$ based on the SXDH assumption. As far as we know, it has the shortest signature size among all the FSPSs for a vector of unilateral messages under standard assumptions.

High-level idea. Our generic construction can be treated as an extension of the well-known EGM paradigm [51]. In this paradigm a signer uses two signature schemes Σ_1 and Σ_2 to sign a message m. It first signs m by using the signing key sk_2 of Σ_2 and then signs the verfication key pk_2 of Σ_2 by using the signing key sk_1 of Σ_1. This paradigm was used to obtain SPSs in [3] and a generic construction of FSPS in [10]. To ensure that the resulting signature scheme is an FSPS scheme, it is natural to require sk_1 to consist only of group elements. This is the reason why Abe et al. [10] instantiated Σ_1 with the xRMA secure signature scheme proposed in [3], which was the only proposed FSPS scheme until then.

[5] The parameters are computed following the equations in [10, Section 6.4].

However, we observe that it is possible to instantiate Σ_1 with all the existing SPS schemes, which also provides us with more options when selecting instantiations of Σ_2 to match Σ_1.

Next, we explain how to choose Σ_1 and Σ_2, and the high level idea of our construction. Roughly speaking, starting from an SPS scheme with a signing key $x \in \mathbb{Z}_p$, we can always derive a signature scheme in which the signing key becomes a group element $X = G^x \in \mathbb{G}$ (where G denotes the generator of $\mathbb{G}$). It is obvious that in this case a message $M \in \mathbb{G}$ cannot be signed by using X since we are not able to compute M^x from X and M. Supposing that $M = G^m$, we can use X to sign m instead of M, i.e., compute X^m instead of M^x when generating a signature. Furthermore, since signatures generated in this way are the same as those generated by the real signing key, and the verification key and verification algorithm remain the same, one can verify the signature by using M. We formalize such a signature scheme as a TS scheme. Although such a signature scheme is only "semi"-structure-preserving, we use it to sign the exponent $v \in \mathbb{Z}_p$ of a verification key (called auxiliary key) of another SPS scheme and use the latter SPS scheme to sign a message $M' \in \mathbb{G}'$. This enables us to obtain an FSPS scheme. We formalize the latter signature scheme which generates auxiliary keys besides verification/signing key pairs as an AKS scheme.

To verify a signature, one only needs to know $V = G^v$ and M', rather than v. Furthermore, the original signing key x (called trapdoor key) of the TS scheme is never used in the signing process but is necessary as the reduction algorithm in the security proof signs verification keys without knowing the exponent.

Our main contributions lie in two aspects. First, we formalize the notions of TS and AKS in order to adapt the EGM paradigm to construct FSPSs. Second, we show that most of existing SPS schemes can be cast as our extended signatures, and consequently we can obtain a number of FSPSs and FASs based on existing SPSs.

Perhaps interestingly, although most of the previously proposed SPS schemes with a unilateral message space are not automorphic (since their verification keys and messages usually consist of elements in different source groups), when some of them are converted into FSPSs by using our method, the resulting schemes become automorphic.[6]

Independently with our work, Abe [2] proposed a variant of the EGM paradigm, which is essentially the same as ours. His construction is based on F-signatures, the security of which is formalized in [19], and partial one-time signatures. The F-signatures and partial one-time signatures play the same role as our TSs and AKSs, respectively.

1.2.2 Constructions of Signature Resilient to Uninvertible Leakage

In this thesis, we also study signatures secure against uninvertible leakage in the selective auxiliary input model and obtain the following results.

- We propose an FLR signature scheme, for which the leakage is allowed to be any

[6] When messages and verification keys of the underlying TS scheme consist of elements in $\mathbb{G}_2$ and $\mathbb{G}_1$ respectively and those of the underlying AKS scheme consist of elements in $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively, verification keys and messages of the resulting FSPS scheme consist of elements only in $\mathbb{G}_1$.

computable uninvertible function on input the secret information. To achieve our goal, we exploit a point-function obfuscator with auxiliary input (AIPO) and a differing-inputs obfuscator (diO) for circuits.[7]

As far as we know, this is the first FLR signature scheme secure in the presence of uninvertible leakage. It is also the first FLR signature scheme with public-coin construction, which does not make use of secret randomness in the signing procedure, as far as we know.

- We propose a weak version of the above signature scheme, for which leakage functions are additionally required to be injective and the sizes of (the circuits representing) them are upper bounded,[8] based on diO, without making use of AIPO. Such restriction makes sense since the leakage information-theoretically determines the secret information typically in the practical world [104] as we mentioned before, and the upper bound on the sizes of leakage functions can be set reasonably, depending on the computational ability of adversaries in the practical world.

Although our constructions are based on strong assumptions, they show that signature schemes resilient to uninvertible leakage are achievable. Furthermore, they can be treated as a solution to the open problem mentioned in [34], which is whether it is possible to achieve public-coin (or deterministic) constructions of FLR signature.[9] Constructing signature schemes with such strong security based on standard assumptions is an open problem we hope to address in future works.

High-level idea. A high-level idea about how we obtain the proposed signature schemes is as follows.

It is obvious that if a leakage function f is allowed to be any computable uninvertible function and *state* contains the signing key sk and the secret randomness $\mathcal{R}$, then an adversary may trivially obtain sk by setting a leakage function f as $f(sk, \mathcal{R}) = (sk, f'(\mathcal{R}))$, where f' is uninvertible. To avoid such attack, we choose the way mentioned by Boyle et al. [34] to achieve FLR signatures, which is letting *state* contain only sk. This requires the signature scheme to be deterministic or only make use of public coins in the signing procedure.

Furthermore, since uninvertible leakage helps an adversary obtain extremely large amount of information of sk, we have to make sure that the verification key and signatures from the signing oracle reveal no information about sk other than the leakage (which do not have to

[7] In this thesis, when we say an "obfuscator", we mean an obfuscator for circuits, unless we clearly state that it is an obfuscator for Turing machines or point functions.

[8] Note that for FLR signatures in the bounded leakage model, it is the number of total leaked bits that is upper bounded, while for FLR signatures in our model, it is the sizes of leakage functions that are upper bounded. Furthermore, the upper bound in the bounded leakage model must be smaller than the size of signing keys, or an adversary can let leakage queries (which can be any polynomially computational functions) output a whole signing key, while the upper bound in our model could be any polynomial.

[9] The only previously proposed FLR signature scheme with a deterministic construction is the one proposed by Katz and Vaikuntanathan [78], which is only one-time secure, and there are no known constructions of FLR signature with public-coin property.

be considered in [110] since they had already assumed that an adversary cannot recover sk from the leakage in the presence of the verification key and signatures, as explained above).

As the first step to achieve our goal, we define the notion of uninvertible leakage resilient (ULR) hard relation. Roughly speaking, this is a binary relation R_{HR} such that if a pair (y, x) satisfying R_{HR} is chosen randomly, then it is hard for any adversary to find x^* such that $R_{\mathsf{HR}}(y, x^*) = 1$, even given y and uninvertible leakage on x. Inspired by Brzuska and Mittelbach [37], who proposed a public key encryption scheme in the auxiliary input model by making use of weak multi-bit AIPO (which is based on AIPO and indistinguishability obfuscator (iO), where iO a special case of diO), we instantiate such a relation by making use of AIPO.

Next we let $(pk, sk) = ((y, \widetilde{\mathbf{Sign}}, \widetilde{\mathbf{Verify}}), x)$ be the verification/signing key pair of our signature scheme, where $\widetilde{\mathbf{Sign}}$ is a signing program obfuscated by diO, $\widetilde{\mathbf{Verify}}$ a verification program obfuscated by iO, and (y, x) is a public/secret key pair satisfying the ULR-hard relation. When signing a message m, $\widetilde{\mathbf{Sign}}$ takes as input (y, x, m) and checks if $R_{\mathsf{HR}}(y, x) = 1$. If the check works out, it outputs $F(K, y||m)$, which is a Sahai-Waters style signature [101] linked with y, where F is a puncturable pseudorandom function and K is a hard-wired value in both $\widetilde{\mathbf{Sign}}$ and $\widetilde{\mathbf{Verify}}$.[10] Otherwise, it aborts. When verifying a message/signature pair (m, σ), $\widetilde{\mathbf{Verify}}$ takes as input (y, m, σ) and checks if $\sigma = F(K, y||m)$. Since $\widetilde{\mathbf{Sign}}$ and $\widetilde{\mathbf{Verify}}$ are independent of (y, x) and signatures contain no information about x other than y, an adversary is not able to recover x, given pk, a set of signatures, and the leakage $f(x)$. As a result, the adversary has no "access" to K to obtain a forged signature linked with y. Since such a scheme is only selectively secure (i.e., an adversary is required to determine the challenge message, on which a signature will be forged, before seeing the verification key), we extend it into an adaptively secure one by letting $\widetilde{\mathbf{Sign}}$ output Ramchen-Waters style signatures [97] instead of Sahai-Waters style ones, linked with y.

If we generate y as an iO-obfuscated point-function that maps all inputs to 0 except for x, instead of AIPO, we obtain another primitive that we call injective uninvertible leakage resilient (IULR) hard relation, for which leakage functions are additionally required to be injective and the sizes of them are upper bounded. By substituting the ULR-hard relation with an IULR-hard relation in the signature scheme we described above, we immediately obtain a signature scheme resilient to injective uninvertible leakage, while the sizes of leakage functions are upper bounded.

Independently with our work, Komagardski [83] defined LR one-way relation in the selective and adaptive models, and showed positive (respectively, negative) results in the former (respectively, later) model.

Status of iO, diO, and AIPO. To achieve signatures secure against uninvertible (full) leakage, we make use of iO [59], diO [17, 12, 32], and AIPO [29], the existence of which is a strong assumption.

[10] K is deleted after generating $\widetilde{\mathbf{Sign}}$ and $\widetilde{\mathbf{Verify}}$.

iO can be used to obfuscate circuits without changing their functionality, and two iO-obfuscated circuits are indistinguishable (in the presence of auxiliary input) if they have the same functionality. diO is a natural extension of iO. The difference is that diO provides stronger guarantee such that two circuits are indistinguishable if it is hard to find an input that leads the underlying original circuits to different outputs, in the presence of auxiliary input. Boyle et al. [32] proved that iO can be used as diO, if the number of inputs leading the two circuits to different outputs is polynomial. AIPO focuses on obfuscating point functions that map all strings to 0 except for a single string mapped to 1, when auxiliary input is present.

The first candidate of iO was given in the breakthrough work by Garg et al. [59] based on multilinear maps. Following their work, many multilinear map based iO schemes have been proposed. Although a lot of works demonstrate that existing multilinear maps suffer from vulnerabilities, most of them have no direct impact on the security of iO candidates, as discussed by Ananth et al. [13, Appendix A]. Furthermore, Ananth et al. [13] showed how to build iO combiners by using the learning with errors and decisional Diffie-Hellman assumptions respectively. By using their combiners, we can produce an instantiation of iO from serval iO candidates, and the resulting instantiation is secure as long as one of the original candidates is secure. They also constructed a universal iO scheme, which is secure as long as any secure iO scheme exists.

Compared with iO, there are more negative results on diO. Garg et al. [60] showed that general-purpose diO for circuits does not exist if there exists some special-purpose obfuscator for Turing machine. However, the heuristic analysis they used to justify the special-purpose obfuscator is itself much stronger than assuming diO as discussed by Bellare et al. in [26]. Following this work, Boyle and Pass [33] showed some negative results on public-coin diO [74] which is a relaxed notion of diO. They proved that if extractable one-way functions w.r.t. some auxiliary input (respectively, succinct non-interactive arguments of knowledge) exist, then public-coin diO for Turing machines (respectively, for NC^1 circuits) does not exist. Recently, Bellare et al. [26] showed that sub-exponentially secure (respectively, polynomially secure) diO for Turing machines does not exist if sub-exponentially secure one-way function (respectively, sub-exponentially secure iO) exists. Although the status of diO is in flux, as far as we know, there is no negative results on diO for circuits (rather than Turing machines) based on weak or standard assumptions yet, beyond the known negative results on iO.

The notion of AIPO was firstly formalized by Bitansky and Paneth [29] while the first candidate of AIPO was proposed by Canetti [42]. Bitansky and Paneth extended the point-function obfuscator proposed by Wee [107] to a candidate of AIPO based on a novel assumption on a trapdoor permutation. The candidate by Canetti is based on the Auxiliary-Input Diffie-Hellman Inversion assumption. Lynn et al. [89] also showed that it is easy to obtain AIPO in the random oracle model. Recently, Bellare and Stepanovs [24] gave three candidates of AIPO respectively based on iO and one-way functions relative to target generators, deterministic public-key encryption, and universal computational extractors [20].

1.3 Outline of This Paper

In Chapter 2, we recall several terminologies and definitions that are necessary to describe our paper. In Chapter 3 we show how to generally construct FSPSs and FASs from SPSs via TSs and AKSs. In Chapter 4, we define FLR signature in the selective auxiliary input model and give constructions based on ULR and IULR hard relations. In Chapter 5, we conclude our results and discuss open problems.

Chapter 2

Preliminaries

2.1 Notations

negl denotes an unspecified negligible function, $x \leftarrow \mathcal{X}$ denotes sampling an element x from a set $\mathcal{X}$ at random, $[n]$ denotes the set $\{1, \ldots, n\}$, $\mathbb{N}$ denotes the set of natural numbers, $|X|$ denotes the number of elements in X (where X could be a space, a vector, or a matrix), and $\widetilde{\mathbf{A}}$ the $1 \times mn$ vector $(a_{11}, a_{12}, \ldots a_{1n}, a_{21}, a_{22}, \ldots a_{2n}, \ldots, a_{m1}, a_{m2}, \ldots a_{mn})$ where $\mathbf{A}$ denotes the $m \times n$ matrix $(a_{ij})_{i\in[m],j\in[n]}$. If $\mathbf{A} \in \mathbb{Z}_p^{(k+1)\times k}$ lies in the matrix distribution $\mathcal{D}_k$ (ref., Section 2.3), then we use $\overline{\mathbf{A}}$ to denote the upper square matrix of $\mathbf{A}$. Furthermore, $\vec{a} \in \mathbb{Z}_p^n$ denotes a column vector by default.

If $\mathcal{A}$ is a deterministic (respectively, probabilistic) algorithm, then $y = \mathcal{A}(x)$ (respectively, $y \leftarrow \mathcal{A}(x)$) means that $\mathcal{A}$ on input x outputs y. Letting the internal randomness space of a probabilistic algorithm $\mathcal{A}$ be $\mathbb{R}_a$, computing $y \leftarrow \mathcal{A}(x)$ is equivalent to sampling $r \leftarrow \mathbb{R}_a$ and then computing $y = \mathcal{A}(x; r)$.

2.2 Pairing Group

In this thesis, we let $\mathcal{G}$ be an algorithm that takes as input 1^λ and outputs $gk = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G_1, G_2)$ such that p is a prime satisfying $p = \Theta(2^\lambda)$, $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ are descriptions of groups of order p, G_1 and G_2 are generators that generate $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficiently computable (non-degenerate) bilinear map. Following [79] and [69], we use the additive notation in [50] such as $e((a+b)[x]_1, [y]_2) = a \cdot e([x]_1, [y]_2) + b \cdot e([x]_1, [y]_2)$ where $[x]_1$ and $[y]_2$ denote G_1^x and G_2^y respectively, and $e([x]_1, [y]_2)$ can be written as $[xy]_T$. Furthermore, $e([\vec{a}]_1^\top, [\vec{b}]_2)$ denotes $\sum_{i=1}^n e([a_i]_1, [b_i]_2)$ where $[\vec{a}]_1 = ([a_1]_1, \ldots, [a_n]_1)^\top$ and $[\vec{b}]_2 = ([b_1]_2, \ldots, [b_n]_2)^\top$, and $e([\mathbf{A}]_1^\top, [\mathbf{B}]_2)$ denotes $(e([\vec{a}_i]_1^\top, [\vec{b}_j]_2))_{i\in[n],j\in[n']}$ where $[\mathbf{A}]_1 = ([\vec{a}_1]_1, \ldots, [\vec{a}_n]_1)$ and $[\mathbf{B}]_2 = ([\vec{b}_1]_2, \ldots, [\vec{b}_{n'}]_2)$.

2.3 SXDH and $\mathcal{D}_k$-MDDH Assumptions

Now we recall the SXDH and $\mathcal{D}_k$-MDDH assumptions.

Definition 2.3.1 (Symmetric External Diffie-Hellman (SXDH) assumption [105].)**.** *Let* $gk = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, [1]_1, [1]_2)$ *be the tuple generated by* $\mathcal{G}$ *as we introduced earlier. We say that the* SXDH *assumption holds if the DDH problem is hard in both groups* $\mathbb{G}_1$ *and* $\mathbb{G}_2$*.*

Definition 2.3.2 (Matrix Distribution)**.** *Let* $k \in \mathbb{N}$*.* $\mathcal{D}_k$ *is said to be a* matrix distribution *if it returns matrices of full rank* k *in* $\mathbb{Z}_p^{(k+1)\times k}$ *in polynomial time.*

The $\mathcal{D}_k$-MDDH assumptions is as follows.

Definition 2.3.3 ($\mathcal{D}_k$-Matrix Decisional Diffie-Hellman (MDDH) assumption [50])**.** *Let* $s \in \{1, 2, T\}$*, and* $\mathcal{D}_k$ *be a matrix distribution. We say that the* $\mathcal{D}_k$-MDDH *assumption holds relative to* $\mathcal{G}$ *in group* $\mathbb{G}_s$ *if for any probabilistic polynomial-time (PPT) adversary* $\mathcal{A}$*,*[1] *we have*

$$|\Pr[\mathcal{A}(gk, [\mathbf{A}]_s, [\mathbf{A}\vec{w}]_s) = 1] - \Pr[\mathcal{A}(gk, [\mathbf{A}]_s, [\vec{u}]_s) = 1]| \leq negl(\lambda),$$

where the probability is taken over $gk \leftarrow \mathcal{G}(1^\lambda)$*,* $\mathbf{A} \leftarrow \mathcal{D}_k, \vec{w} \leftarrow \mathbb{Z}_p^k$*, and* $\vec{u} \leftarrow \mathbb{Z}_p^{k+1}$*.*

As in [79], we define the representation size of $\mathcal{D}_k$, denoted by $\mathsf{RE}(\mathcal{D}_k)$, as the minimal number of group elements needed to represent a matrix sampled from $\mathcal{D}_k$. Additionally, we require the space of the exponents of these group elements to be $\mathbb{M}_1 \times \mathbb{M}_2 \times \ldots \times \mathbb{M}_{\mathsf{RE}(\mathcal{D}_k)}$, where $\mathbb{M}_i \subseteq \mathbb{Z}_p$ and $|\mathbb{M}_i|$ is super-polynomially large for $i \in \{1, \ldots, \mathsf{RE}(\mathcal{D}_k)\}$.[2] We specify the following distributions $\mathcal{L}_k$, $\mathcal{SK}_k$, and $\mathcal{U}_k$ such that the corresponding $\mathcal{D}_k$-MDDH assumptions are generically secure in bilinear groups and form a hierarchy of increasingly weaker assumptions, as in [50, 79].

$$\mathcal{SC}_k = \begin{pmatrix} 1 & 0 & 0 & \ldots & 0 \\ a & 1 & 0 & \ldots & 0 \\ 0 & a & 1 & \ldots & 0 \\ 0 & 0 & a & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & a \end{pmatrix}, \mathcal{L}_k = \begin{pmatrix} 1 & 1 & 1 & \ldots & 1 \\ a_1 & 0 & 0 & \ldots & 0 \\ 0 & a_2 & 0 & \ldots & 0 \\ 0 & 0 & a_3 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & a_k \end{pmatrix}, \mathcal{U}_k = \begin{pmatrix} a_{1,1} & \ldots & a_{1,k} \\ \vdots & \ddots & \vdots \\ a_{k+1,1} & \ldots & a_{k+1,k} \end{pmatrix}.$$

where $a, a_i, a_{i,j} \leftarrow \mathbb{Z}_p$ for each $k \geq 1$. Note that the $\mathcal{D}_1$-MDDH assumption is the DDH assumption, the $\mathcal{L}_k$-MDDH assumption is the k-Linear assumption (for each k), and the $\mathcal{SC}_k$-MDDH assumption offers the same security guarantees as the $\mathcal{L}_k$-MDDH assumption (for each k). Here, $\mathsf{RE}(\mathcal{SC}_k) = 1$, $\mathsf{RE}(\mathcal{L}_k) = k$, and $\mathsf{RE}(\mathcal{U}_k) = (k+1)k$.

[1] In this thesis, when we say PPT adversary, we mean a non-uniform PPT adversary.

[2] This requirement is to ensure that when we change the message spaces of TSs to match the auxiliary key spaces of AKSs in our generic constructions, the TSs are still well-formed. However, when treating " Relaxed FSP property" described below Definition 3.1.3 in Section 3.1, this additionally requirement is not necessary.

2.4 One-way Function and Uninvertible Function

Now we recall the definitions of one-way function and uninvertible function.

Definition 2.4.1 (One-way function)**.** *A function $f : \{0,1\}^* \to \{0,1\}^*$ is said to be* one-way *if it is efficiently computable, and for any PPT adversary $\mathcal{A}$, we have*

$$\Pr[x \leftarrow \{0,1\}^\lambda, x^* \leftarrow \mathcal{A}(1^\lambda, f(x)) : f(x^*) = f(x)] \leq negl(\lambda).$$

Definition 2.4.2 (Uninvertible function)**.** *A function $f : \{0,1\}^* \to \{0,1\}^*$ is said to be* uninvertible *if it is efficiently computable, and for any PPT adversary $\mathcal{A}$, we have*

$$\Pr[x \leftarrow \{0,1\}^\lambda : x \leftarrow \mathcal{A}(1^\lambda, f(x))] \leq negl(\lambda).$$

A one-way or uninvertible function is said to be injective if it additionally satisfies that for all a and b in $\{0,1\}^*$, $f(a) = f(b)$ implies $a = b$. Note that a function is an injective one-way function iff it is injective and uninvertible.

It is not hard to see that an uninvertible function is not necessarily a one-way function while a one-way function must be uninvertible, which means that the class of uninvertible functions is larger than that of one-way functions.

2.5 Obfuscation

In this section, we recall the definitions of diO (for circuits), iO (for circuits), and AIPO. Below, $\mathcal{C}_\lambda$ denotes a family of circuits whose size is some polynomial of λ.

Definition 2.5.1 (Same-functionality sampler/Differing-inputs sampler)**.** *Let* Samp *be a (non-uniform) PPT algorithm that takes 1^λ as input, and outputs two circuits $C_0, C_1 \in \mathcal{C}_\lambda$ and a string $\alpha \in \{0,1\}^*$.* Samp *is said to be*

- *a* same-functionality sampler *for $\{\mathcal{C}_\lambda\}$ if the two circuits in the output of* Samp *have the same functionality (i.e. $C_0(x) = C_1(x)$ for all inputs x).*
- *a* differing-inputs sampler *for $\{\mathcal{C}_\lambda\}$ if for any PPT adversary $\mathcal{A}$, we have*

$$\Pr[(C_0, C_1, \alpha) \leftarrow \mathsf{Samp}(1^\lambda), x \leftarrow \mathcal{A}(1^\lambda, C_0, C_1, \alpha) : C_0(x) \neq C_1(x)] \leq negl(\lambda).$$

Definition 2.5.2 (Differing-inputs obfuscation (diO))**.** *A uniform PPT algorithm $\mathcal{DIO}$ is said to be* diO *for circuit class $\{\mathcal{C}_\lambda\}$, if it satisfies the* functionality preserving *property and the* differing-inputs *property.*

The functionality preserving property is satisfied if for all security parameters λ, all $C \in \mathcal{C}_\lambda$, all $C' \leftarrow \mathcal{DIO}(1^\lambda, C)$, and all inputs x, we have $C'(x) = C(x)$.

The differing-inputs property is satisfied if for any differing-inputs sampler Samp *for $\{\mathcal{C}_\lambda\}$ and any PPT adversary $\mathcal{D}$, we have*

$$\begin{aligned} &|\Pr[(C_0, C_1, \alpha) \leftarrow \mathsf{Samp}(1^\lambda) : \mathcal{D}(1^\lambda, \mathcal{DIO}(1^\lambda, C_0), \alpha) = 1] \\ &- \Pr[(C_0, C_1, \alpha) \leftarrow \mathsf{Samp}(1^\lambda) : \mathcal{D}(1^\lambda, \mathcal{DIO}(1^\lambda, C_1), \alpha) = 1]| \leq negl(\lambda). \end{aligned}$$

Definition 2.5.3 (Indistinguishability obfuscation (iO))**.** *A uniform PPT algorithm $\mathcal{IO}$ is said to be* iO *for circuit class $\{\mathcal{C}_\lambda\}$ if it satisfies the* functionality preserving *property and* indistinguishability *property. The former property is defined in exactly the same way as that of diO. The indistinguishability property is also defined in the same way as the differing-inputs property of diO, except that we replace "for any differing-inputs sampler" with "for any same-functionality sampler".*

Definition 2.5.4 (Point function)**.** *A function p_x for a value $x \in \{0,1\}^*$ is called a* point-function *if for any $\check{x} \in \{0,1\}^*$, we have $p_x(\check{x}) = 1$ if $\check{x} = x$, and $p_x(\check{x}) = 0$ otherwise.*

Definition 2.5.5 (Unpredictable distribution)**.** *A distribution ensemble $\{Z_\lambda, X_\lambda\}$ associated with a PPT algorithm* Samp *is said to be* unpredictable *if for any PPT adversary $\mathcal{A}$, we have*

$$\Pr[(z,x) \leftarrow \mathsf{Samp}(1^\lambda) : x \leftarrow \mathcal{A}(1^\lambda, z)] \leq negl(\lambda).$$

Definition 2.5.6 (Point obfuscation with auxiliary input (AIPO))**.** *A PPT algorithm $\mathcal{AIPO}$ is said to be* AIPO *if on input x it outputs a polynomial-size circuit $\tilde{p}_x$ such that $\tilde{p}_x(\check{x}) = 1$ if $\check{x} = x$ and $\tilde{p}_x(\check{x}) = 0$ otherwise, and the following property is satisfied.*

For any unpredictable distribution associated with a PPT algorithm Samp *over $\{0,1\}^* \times \{0,1\}^\lambda$ and any PPT algorithm $\mathcal{D}$, we have*

$$\begin{aligned}&|\Pr[(z,x) \leftarrow \mathsf{Samp}(1^\lambda), r \leftarrow \{0,1\}^\lambda, \tilde{p} \leftarrow \mathcal{AIPO}(r) : \mathcal{D}(1^\lambda, \tilde{p}, z) = 1]\\ &- \Pr[(z,x) \leftarrow \mathsf{Samp}(1^\lambda), \tilde{p} \leftarrow \mathcal{AIPO}(x) : \mathcal{D}(1^\lambda, \tilde{p}, z) = 1]| \leq negl(\lambda).\end{aligned}$$

We will utilize the following simple fact about AIPO. Although its proof is straightforward, we give the formal proof of this lemma for completeness.

Lemma 2.5.1. *If a PPT algorithm $\mathcal{AIPO}$ is AIPO, then $\mathcal{AIPO}$ is a probabilistic uninvertible function, i.e., the distribution of $(\mathcal{AIPO}(r), r)$ where r is randomly chosen from $\{0,1\}^\lambda$ is unpredictable.*

Proof of Lemma 2.5.1. Let $\mathcal{AIPO}$ be any AIPO and $\mathcal{A}$ any PPT adversary. If $\mathcal{A}$ breaks the uninvertibility of $\mathcal{AIPO}$ with probability ϵ, we construct a sampler Samp that samples an unpredictable distribution and an adversary $\mathcal{D}$ that break the security of $\mathcal{AIPO}$ as follows with advantage $\epsilon - 2^{-\lambda/2}$ if $\epsilon \geq 2^{-\lambda/2}$, where λ denotes the security parameter. We show how to construct Samp and $\mathcal{D}$ as follows.

On input 1^λ, Samp randomly chooses $x \leftarrow \{0,1\}^\lambda$, where $x = x_1||x_2$ and $x_1, x_2 \in \{0,1\}^{\lambda/2}$, and outputs x_1. Since x_2 is information-theoretically hidden in the presence of x_1, the distribution of (x_1, x) is unpredictable.

Let r be a bit string randomly chosen from from $\{0,1\}^\lambda$ and (x_1, x) a tuple sampled by Samp on input 1^λ. $\mathcal{D}$ takes as input $(1^\lambda, \tilde{p}, x_1)$ where $\tilde{p}$ is generated as $\tilde{p} \leftarrow \mathcal{AIPO}(x)$ or $\tilde{p} \leftarrow \mathcal{AIPO}(r)$, and gives $(1^\lambda, \tilde{p})$ to $\mathcal{A}$. When $\mathcal{A}$ outputs $x^* = x_1^*||x_2^*$ where $x_1^*, x_2^* \in \{0,1\}^{\lambda/2}$, $\mathcal{D}$ checks if $x_1^* = x_1$. If the check works out, $\mathcal{D}$ outputs 1. Otherwise, $\mathcal{D}$ outputs 0.

If $\tilde{p}$ is generated as $\tilde{p} \leftarrow \mathcal{AIPO}(x)$, the probability that $x = x^*$ is ϵ since $\mathcal{A}$ breaks the uninvertibility of $\mathcal{AIPO}$ with advantage ϵ, i.e., $\mathcal{D}$ outputs 1 with probability at least ϵ.

Otherwise, the probability that $\mathcal{D}$ outputs 1 is $2^{-\lambda/2}$ since $\mathcal{A}$ learns no information about x at all. As a result, $(\mathsf{Samp}, \mathcal{D})$ breaks the security of $\mathcal{AIPO}$ with advantage at least $\epsilon - 2^{-\lambda/2}$.

Since the security of $\mathcal{AIPO}$ implies that $\epsilon - 2^{-\lambda/2}$ is negligible, we have that ϵ is negligible, completing the proof of Lemma 2.5.1. □

2.6 Puncturable Pseudorandom Function

Now we recall the definition of puncturable pseudorandom function (puncturable PRF) [31, 101], which is a variant of PRF.

Definition 2.6.1 (Puncturable pseudorandom function (puncturable PRF))**.** *A* puncturable PRF *consists of three algorithms* $(F, Puncture, Eval)$*.*

- $F : \mathcal{K} \times \{0,1\}^{m(\lambda)} \to \{0,1\}^{n(\lambda)}$ *is a PRF function that takes as input* $K \in \mathcal{K}$ *and a bit string* $x \in \{0,1\}^{m(\lambda)}$*, and outputs a string* $y \in \{0,1\}^{n(\lambda)}$*, where* m *and* n *are polynomial functions.*
- *Puncture takes as input* $K \in \mathcal{K}$ *and a bit string* $s \in \{0,1\}^{m(\lambda)}$*, and outputs a punctured key* $K\{s\}$*.*
- *Eval takes as input a punctured key* $K\{s\}$ *and a bit string* $x \in \{0,1\}^{m(\lambda)}$*, and outputs a string* $y \in \{0,1\}^{n(\lambda)}$*.*

The puncturable PRF must satisfy two properties, which are functionality preserved under puncturing *property and* pseudorandom at punctured point *property.*

The functionality preserved under puncturing property is satisfied if for all security parameters λ*, all* $s, x \in \{0,1\}^{m(\lambda)}$ *such that* $x \neq s$*, and all* $K \in \mathcal{K}$*, we have* $Eval(K\{s\}, x) = F(K, x)$ *where* $K\{s\} = Puncture(K, s)$*.*

The pseudorandom at punctured point property is satisfied if for any PPT adversary $(\mathcal{A}_1, \mathcal{A}_2)$*, we have*

$$\begin{aligned} |\Pr[(s,\alpha) \leftarrow \mathcal{A}_1(1^\lambda), K \leftarrow \mathcal{K}, K\{s\} = Puncture(K,s) : \mathcal{A}_2(K\{s\}, F(K,s), \alpha) = 1] \\ - \Pr[(s,\alpha) \leftarrow \mathcal{A}_1(1^\lambda), K \leftarrow \mathcal{K}, K\{s\} = Puncture(K,s), r \leftarrow \{0,1\}^{n(\lambda)} : \\ \mathcal{A}_2(K\{s\}, r, \alpha) = 1]| \leq negl(\lambda). \end{aligned}$$

2.7 Digital Signature

Definition 2.7.1 (Digital Signature)**.** *A* signature scheme *consists of four polynomial-time algorithms* Setup*,* Gen*,* Sign*, and* Verify*.*

- Setup *takes as input a security parameter* 1^λ *and returns a public parameter par, which determines the message space* $\mathcal{M}$ *and the randomness space* $\mathcal{R}$ *for signing.*
- Gen *is a randomized algorithm that takes as input a public parameter par, and returns a verification/signing key pair* (pk, sk)*.*

- Sign *is a randomized algorithm that takes as input a signing key* sk *and a message* m*, and returns a signature* σ*. It is also implicitly given* (par, pk) *as input.*
- Verify *is a deterministic algorithm that takes as input a verification key* pk*, a message* m*, and a signature* σ*, and returns* 1 *(accept) or* 0 *(reject).*

Correctness *is satisfied if we have* $\mathsf{Verify}(pk, m, \mathsf{Sign}(sk, m; r)) = 1$ *for all* $\lambda \in \mathbb{N}$*,* $par \leftarrow \mathsf{Setup}(1^\lambda)$*,* $(pk, sk) \leftarrow \mathsf{Gen}(par)$*,* $m \in \mathcal{M}$*, and* $r \in \mathcal{R}$*.*

We now recall the UF-CMA, UF-RMA, UF-otCMA, and UF-otRMA security of a signature scheme.

Definition 2.7.2 (UF-CMA)**.** *A signature scheme* $(\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *is said to be* UF-CMA *secure if for any PPT adversary* $\mathcal{A}$*, we have*

$$\Pr[par \leftarrow \mathsf{Setup}(1^\lambda), (pk, sk) \leftarrow \mathsf{Gen}(par), (m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathrm{SignO}(\cdot)}(par, pk) : m^* \notin \mathcal{Q}_m \wedge \mathsf{Verify}(pk, m^*, \sigma^*) = 1] \leq negl(\lambda),$$

where $\mathrm{SignO}(\cdot)$ *is the signing oracle that takes* m *as input, runs* $\sigma \leftarrow \mathsf{Sign}(sk, m)$*, adds* m *to* $\mathcal{Q}_m$ *(initialized with* $\emptyset$*), and returns* σ*.*

Definition 2.7.3 (UF-RMA, UF-otCMA and UF-otRMA)**.** UF-RMA *security is the same as UF-CMA security except that the signing oracle* $\mathrm{SignO}(\cdot)$ *randomly chooses* $m \leftarrow \mathcal{M}$ *by itself, adds* m *to* $\mathcal{Q}_m$*, and returns* m *along with the signature.*

UF-otCMA *(respectively,* UF-otRMA*) security is the same as UF-CMA (respectively, UF-RMA) security, except that* $\mathcal{A}$ *is only allowed to make one query to the signing oracle* $\mathrm{SignO}(\cdot)$*.*

Chapter 3

Generic Constructions of Fully Structure-Preserving Signature

In this chapter, we first recall the definitions of (F)SPS and FAS. We then formalize the definitions of TS and AKS and show how to instantiate them from any (well-formed) SPS scheme. Finally, we give generic constructions of FSPS and FAS based on TSs and AKSs, and also several instantiations.

3.1 (Fully) Structure-Preserving Signature

In [5], Abe et al. firstly defined *SPS*, in which verification keys, messages, and signatures consist only of group elements in $\mathbb{G}_1$ and $\mathbb{G}_2$, and signatures are verified by evaluating pairing product equations (PPEs), which are of the form $\sum_{ij} a_{ij} e([x_i]_1, [y_j]_2) = [0]_T$, where a_{ij} is an integer constant for all i and j.

Definition 3.1.1 (Structure-preserving signature (SPS))**.** *A signature scheme is said to be an* SPS *scheme over a bilinear group generator $\mathcal{G}$ if we have*

(*a*) *a public parameter includes a group description gk generated by $\mathcal{G}$,*

(*b*) *verification keys consist only of group elements in $\mathbb{G}_1$ and $\mathbb{G}_2$,*

(*c*) *messages consist only of group elements in $\mathbb{G}_1$ and $\mathbb{G}_2$,*

(*d*) *signatures consist only of group elements in $\mathbb{G}_1$ and $\mathbb{G}_2$, and*

(*e*) *the verification algorithm consists only of evaluating membership in $\mathbb{G}_1$ and $\mathbb{G}_2$ and relations described by PPEs.*

SPSs are versatile since they mix well with other pairing-based protocols. Especially, they are compatible with the Groth-Sahai proof system [70]. However, as argued by Abe et al. in [5], Groth-Sahai compatibility of a signature scheme is not sufficient for elaborate applications such as anonymous signatures and delegatable anonymous credentials, which

require signatures on verification keys to obtain anonymized certification chains. Abe et al. [5] called an SPS scheme that is able to sign its own verification keys an *automorphic signature* scheme.

Definition 3.1.2 (Automorphic signature)**.** *A signature scheme is said to be an* automorphic signature scheme *over a bilinear group generator $\mathcal{G}$ if it is structure-preserving and its (padded) verification keys lie in the message space.*

In [10], Abe et al. introduced *FSPS*, which also requires a signing key to be group elements in $\mathbb{G}_1$ and $\mathbb{G}_2$ and the correctness of a signing key w.r.t. a verification key can be verified by PPEs. Such signatures allow efficient key extraction when combined with non-interactive proofs (e.g., the Groth-Sahai proofs), which may help prevent rogue-key attacks [90], build UC-secure privacy preserving protocols [40], strengthen privacy in group and ring signatures [21, 22, 27] in the presence of adversarial keys, extend delegatable anonymous credential systems [18, 53, 45] with all-or-nothing transferability [41], and construct unlinkable redactable signatures [39].

Definition 3.1.3 (Fully structure-preserving signature (FSPS))**.** *An SPS scheme* $(\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *with the message space $\mathcal{M}$ and randomness space $\mathcal{R}$ for signing is said to be an* FSPS *scheme if we have*

(*a*) *signing keys consist only of group elements in $\mathbb{G}_1$ and $\mathbb{G}_2$, and additionally,*

(*b*) *there exists a polynomial-time deterministic algorithm* $\mathsf{VerifySK}$ *that takes as input a verification/signing key pair and consists only of evaluating membership in $\mathbb{G}_1$ and $\mathbb{G}_2$ and relations described by PPEs, and it is required that for sufficiently large $\lambda \in \mathbb{N}$, $par \leftarrow \mathsf{Setup}(1^\lambda)$, the following holds:*

- $\mathsf{VerifySK}(pk, sk) = 1$ *if and only if* $\mathsf{Verify}(pk, m, \mathsf{Sign}(sk, m; r)) = 1$ *holds for all $m \in \mathcal{M}$ and $r \in \mathcal{R}$.*

Relaxed FSP property. In practice, we can relax the condition (*b*) of Definition 3.1.3 as follows.

There exists a polynomial-time deterministic algorithm $\mathsf{VerifySK}$ that takes as input a verification/signing key pair and consists only of evaluating membership in $\mathbb{G}_1$ and $\mathbb{G}_2$ and relations described by PPEs, and it is required that for sufficiently large $\lambda \in \mathbb{N}$, $par \leftarrow \mathsf{Setup}(1^\lambda)$, the following holds:

- if $\mathsf{VerifySK}(pk, sk) = 1$, then $\mathsf{Verify}(pk, m, \mathsf{Sign}(sk, m; r)) = 1$ holds for all $m \in \mathcal{M}$ and $r \in \mathcal{R}$,
- for all $(pk, sk) \leftarrow \mathsf{Gen}(par)$, $\mathsf{VerifySK}(pk, sk) = 1$ holds.

This relaxed correctness of signing keys makes sure that if $\mathsf{VerifySK}(pk, sk) = 1$ holds, then sk is a valid signing key for pk, and on the other hand, all honestly generated (pk, sk) satisfy $\mathsf{VerifySK}(pk, sk) = 1$. When adopting this version of FPS property, we can convert an SPS

scheme to an FSPS one in a more direct way, without adjusting the message spaces (see the remark in Section 3.5.1).

In this thesis, we call an automorphic signature scheme which is also FSP a *fully automorphic signature (FAS) scheme.*

Definition 3.1.4 (Fully automorphic signature (FAS))**.** *An automorphic signature scheme is said to be an* FAS *scheme if it is also fully structure-preserving.*

3.2 Trapdoor Signature

In this section, we introduce *TS*, the instantiations of which are used as building blocks to obtain FSPSs. In Section 3.2.1 and Section 3.2.2, we give the definition and security of TS respectively. In Section 3.2.3, we show how to convert FSPSs from SPSs. In Section 3.2.4, we give instantiations of TS.

3.2.1 Definition of Trapdoor Signature

In this section, we formalize the notion of γ-TS. Different from standard signatures, a TS scheme verifies the validity of a signature σ on a message $m \in \mathcal{M}$ by taking as input $(\gamma(m) \in \mathcal{M}_\gamma, \sigma)$ where $\gamma : \mathcal{M} \mapsto \mathcal{M}_\gamma$ is an efficiently computable bijection. Furthermore, there exists a trapdoor key with which we can generate a signature on m if we have $\gamma(m)$ but not m itself.

Definition 3.2.1 (γ-Trapdoor signature (γ-TS))**.** *A* γ-TS *scheme consists of five polynomial-time algorithms* Setup*,* Gen*,* Sign*,* Verify*, and* TDSign*.*

- Setup *takes as input a security parameter* 1^λ *and returns a public parameter* par*, which determines the message space* $\mathcal{M}$ *for the signing algorithm, the message space* $\mathcal{M}_\gamma$ *for the verification algorithm, and an efficiently computable bijection* $\gamma : \mathcal{M} \mapsto \mathcal{M}_\gamma$*.*
- Gen *is a randomized algorithm that takes as input* par*, and returns a verification/signing key pair* (pk, sk) *and a trapdoor key* tk*.*
- Sign *is a randomized algorithm that takes as input a signing key* sk *and a message* $m \in \mathcal{M}$*, and returns a signature* σ*, where the randomness space is denoted by* $\mathcal{R}$*.*
- Verify *is a deterministic algorithm that takes as input a verification key* pk*, a message* $M \in \mathcal{M}_\gamma$*, and a signature* σ*, and returns* 1 *(accept) or* 0 *(reject).*
- TDSign *takes as input a trapdoor key* tk *and a message* $M \in \mathcal{M}_\gamma$*, and returns a signature* σ*. The randomness space of* TDSign *is also* $\mathcal{R}$*.*

Correctness *is satisfied if for all* $\lambda \in \mathbb{N}$*,* $par \leftarrow \mathsf{Setup}(1^\lambda)$*,* $((pk, sk), tk) \leftarrow \mathsf{Gen}(par)$*, and* $m \in \mathcal{M}$*, we have (a)* $\mathsf{Verify}(pk, \gamma(m), \mathsf{Sign}(sk, m)) = 1$*, and (b)* $\mathsf{Sign}(sk, m; r) = \mathsf{TDSign}(tk, \gamma(m); r)$ *for all* $r \in \mathcal{R}$*.*

Key generation algorithm $\mathcal{T}_{\mathsf{Gen}}$. We use $\mathcal{T}_{\mathsf{Gen}}$ to denote an algorithm that runs Gen, which is the key generation algorithm of a TS scheme, in the following way. Taking as input a public parameter par, $\mathcal{T}_{\mathsf{Gen}}$ gives par to Gen and obtains an output $((pk, sk), tk)$. Then $\mathcal{T}_{\mathsf{Gen}}$ outputs (pk, tk) as a verification/signing key pair.

For a TS scheme $\Sigma = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{TDSign})$, we denote $(\mathsf{Setup}, \mathcal{T}_{\mathsf{Gen}}, \mathsf{TDSign}, \mathsf{Verify})$ by $\mathcal{T}_{\Sigma}$. According to the syntax of TS, it is not hard to see that $\mathcal{T}_{\Sigma}$ forms a standard signature scheme whose message space is $\mathcal{M}_{\gamma}$.

Now we define SKSP-TS, in which verification keys, signing keys, and signatures (but not necessarily messages) consist only of group elements, and the correctness of signing keys w.r.t. verifications keys can be verified by PPEs.

Definition 3.2.2 (Signing key structure-preserving (SKSP))**.** *A γ-TS scheme $\Sigma = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{TDSign})$ with message space $\mathcal{M}$ is said to be satisfy the* SKSP *property over a bilinear group generator $\mathcal{G}$ if we have*

(*a*) *$\mathcal{T}_{\Sigma}$ is an SPS scheme,*

(*b*) *signing keys (rather than trapdoor keys) consist only of group elements in $\mathbb{G}_1$ and $\mathbb{G}_2$, and*

(*c*) *there exists a polynomial-time deterministic algorithm $\mathsf{VerifySK}$ that takes as input a verification/signing key pair and consists only of evaluating membership in $\mathbb{G}_1$ and $\mathbb{G}_2$ and relations described by PPEs, and it is required that for sufficiently large $\lambda \in \mathbb{N}$, $par \leftarrow \mathsf{Setup}(1^{\lambda})$, the following holds:*

- *$\mathsf{VerifySK}(pk, sk) = 1$ if and only if $\mathsf{Verify}(pk, \gamma(m), \mathsf{Sign}(sk, m; r)) = 1$ holds for all $m \in \mathcal{M}$ and $r \in \mathcal{R}$.*

Note that different from FSPSs, messages are not required to be group elements in SKSP-TSs.

3.2.2 Security of Trapdoor Signatures

We now define the UF-CMA security of TSs.

Definition 3.2.3 (UF-CMA of TSs)**.** *A γ-TS scheme $(\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{TDSign})$ is said to be* UF-CMA *secure if for any PPT adversary $\mathcal{A}$, we have*

$$\Pr[par \leftarrow \mathsf{Setup}(1^{\lambda}), ((pk, sk), tk) \leftarrow \mathsf{Gen}(par), (M^*, \sigma^*) \leftarrow \mathcal{A}^{\mathrm{SignO}(\cdot)}(par, pk) : M^* \notin \mathcal{Q}_m \wedge \mathsf{Verify}(pk, M^*, \sigma^*) = 1] \leq negl(\lambda)$$

where $\mathrm{SignO}(\cdot)$ *is the signing oracle that takes as input $m \in \mathcal{M}$, runs $\sigma \leftarrow \mathsf{Sign}(sk, m)$, adds $\gamma(m) \in \mathcal{M}_{\gamma}$ to $\mathcal{Q}_m$ (initialized with $\emptyset$), and returns σ.*

Unlike the UF-CMA security of standard signatures, a query m made by an adversary is in $\mathcal{M}$, the signing oracle records $\gamma(m) \in \mathcal{M}_\gamma$, and the message M^* output by the adversary is in $\mathcal{M}_\gamma$.

The UF-CMA security of TSs is similar to the F-unforgeability of standard signatures defined by Belenkiy et al. [19]. Moreover, Libert et al. [87] gave an instantiation of F-unforgeable signature and combined it with a tagged one-time signature scheme proposed by Abe et al. [4] to obtain a very efficient SPS scheme. However, they neither provided generic constructions nor considered the FSP property.

Now we show the relation between the UF-CMA security of $(\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{TDSign})$ and that of $(\mathsf{Setup}, \mathcal{T}_{\mathsf{Gen}}, \mathsf{TDSign}, \mathsf{Verify})$ in Theorem 3.2.1.

Theorem 3.2.1. *For a γ-TS scheme $\Sigma = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{TDSign})$, if $\mathcal{T}_\Sigma = (\mathsf{Setup}, \mathcal{T}_{\mathsf{Gen}}, \mathsf{TDSign}, \mathsf{Verify})$ is UF-CMA secure, then Σ is UF-CMA secure.*

Proof of Theorem 3.2.1. We argue that if there exists a PPT adversary $\mathcal{A}$ that breaks the UF-CMA security of Σ with non-negligible probability, then there exists a PPT adversary $\mathcal{B}$ that breaks the UF-CMA security of $\mathcal{T}_\Sigma$.

The challenger runs $\mathsf{Setup}(1^\lambda)$ to generate a public parameter par which defines the message spaces $\mathcal{M}$ and $\mathcal{M}_\gamma$ and the bijection $\gamma : \mathcal{M} \mapsto \mathcal{M}_\gamma$. Then it runs $(pk, sk) \leftarrow \mathsf{Gen}(par)$ and sends (par, pk) to $\mathcal{B}$.

Taking as input (par, pk), $\mathcal{B}$ gives it to $\mathcal{A}$. On receiving the ith signing query $m_i \in \mathcal{M}$ from $\mathcal{A}$, $\mathcal{B}$ sends $\gamma(m_i) \in \mathcal{M}_\gamma$ to the signing oracle and gets the answer σ_i which is computed as $\sigma_i = \mathsf{TDSign}(tk, \gamma(m_i); r_i)$ where r_i is the randomness used in the signing process. Then $\mathcal{B}$ sends σ_i back to $\mathcal{A}$ as the answer for the signing query. When $\mathcal{A}$ outputs a forgery (M^*, σ^*), $\mathcal{B}$ outputs (M^*, σ^*) where $M^* \in \mathcal{M}_\gamma$.

Since we have $\sigma_i = \mathsf{TDSign}(tk, \gamma(m_i); r_i) = \mathsf{Sign}(sk, m_i; r_i)$ according to the correctness of Σ, $\mathcal{B}$ perfectly simulates the signing oracle of $\mathcal{A}$. As a result, when $\mathcal{A}$ succeeds in outputting a valid forgery, we have $\mathsf{Verify}(pk, M^*, \sigma^*) = 1$ and $M^* \neq \gamma(m_i)$ for all i, which is exactly the condition of breaking the UF-CMA security of $\mathcal{T}_\Sigma$. This means that if $\mathcal{A}$ has non-negligible advantage in breaking the UF-CMA security of Σ, $\mathcal{B}$ breaks the UF-CMA security of $\mathcal{T}_\Sigma$ with non-negligible advantage, completing the proof of Theorem 3.2.1. □

Now we give the definitions of unforgeability against random message attacks (RMA), one-time chosen message attacks (otCMA), and one-time random message attacks (otRMA) of TSs.

Definition 3.2.4 (UF-RMA, UF-otCMA, and UF-otRMA of TSs)**.** *The* UF-RMA *security of TSs is the same as the UF-CMA security of TSs except that to answer a signing query,* $\mathrm{SignO}(\cdot)$ *randomly chooses $m \leftarrow \mathcal{M}$ itself, runs $\sigma \leftarrow \mathsf{Sign}(sk, m)$, adds $\gamma(m)$ to $\mathcal{Q}_m$ (initialized with $\emptyset$), and returns (m, σ).*

The UF-otCMA *(respectively,* UF-otRMA*) security is the same as the UF-CMA (respectively, UF-RMA) security of TSs, except that $\mathcal{A}$ is only allowed to make one query to the signing oracle* $\mathrm{SignO}(\cdot)$*.*

3.2.3 Converting Structure-Preserving Signatures into Signing Key Structure-Preserving Trapdoor Signatures

In this section, we show how to convert SPSs into SKSP-TSs. Before showing our conversion, we define a class of SPSs called well-formed SPSs. Roughly speaking, for a well-formed SPS scheme, it is required that the spaces of elements in randomness and exponents of messages are super-polynomially large in the security parameter, and generating a signature element only involves the group operation, while the scalars of group elements are computed as arithmetic circuits of elements in the signing key and the randomness.

Definition 3.2.5 (Well-formed SPS)**.** *For an SPS scheme Σ, let $\mathbb{M}_1 \times \mathbb{M}_2 \times \ldots \times \mathbb{M}_n$ be the space of exponents (with $[1]_1$ and $[1]_2$ for bases) of elements in a message,*[1] *and $\mathbb{R}_1 \times \mathbb{R}_2 \times \ldots \times \mathbb{R}_{n'}$ the randomness space (for signing), where $n, n' \in \mathbb{N}$. Σ is said to be* well-formed *if (a) for all i, $\mathbb{M}_i, \mathbb{R}_i \subseteq \mathbb{Z}_p$ and $|\mathbb{M}_i|$ and $|\mathbb{R}_i|$ are super-polynomial in the security parameter, and (b) generating a group element $[B]_b$ where $b \in \{1, 2\}$ in a signature only involves computing*

$$[B]_b = \sum_i (\prod_j a_{ij}^{c_{ij}})[A_i]_b, \tag{3.1}$$

where $\{[A_i]_b\}_i$ denotes elements appearing in the public parameters, the message, and the signing key, $\{a_{ij}\}_{ij}$ denotes elements (in $\mathbb{Z}_p$) appearing in the signing key and the randomness for signing, and integer constants, and $\{c_{ij}\}_{ij}$ denotes integer constants (independent of the security parameter). Here, elements in $\{[A_i]_b\}_i$ may represent the same variables, and the same argument is made for $\{a_{ij}\}_{ij}$.[2]

Note that there is no requirement on the distributions of the elements other than the space sizes in the above definition, and as far as we know, all the existing SPSs are well-formed.[3] Now we show that any well-formed SPS scheme can be converted into an SKSP-TS scheme.

Theorem 3.2.2. *Any well-formed SPS scheme, the messages of which are supposed to be of the form $([\vec{M}]_1, [\vec{N}]_2)$, can be converted into a γ-SKSP-TS scheme for γ defined by $\gamma(\vec{M}, \vec{N}) = ([\vec{M}]_1, [\vec{N}]_2)$.*

Schwartz-Zippel Lemma. Now we introduce Schwartz-Zippel Lemma [103], based on which we will give the proof of Theorem 3.2.2.

[1] We do not count repeated elements, e.g., when messages are of the form $([m]_1, [m]_2)$ where $m \in \mathbb{Z}_p$, we have $n = 1$ and $\mathbb{M}_1 = \mathbb{Z}_p$.

[2] For ease of understanding, we give an example here. Supposing that an element in a signature is generated as $(r_1 s_1 + r_2^2 r_1)[U]_1 + s_2^{-1}[M]_1 + [S]_1$, where (r_1, r_2), $(s_1, s_2, [S]_1)$, $[U]_1$, and $[M]_1$ are respectively element(s) in the randomness, signing key, verification key, and message, then we express the formula as $(a_{11}^{c_{11}} a_{12}^{c_{12}})[A_1]_1 + (a_{21}^{c_{21}} a_{22}^{c_{22}})[A_2]_1 + a_{31}^{c_{31}}[A_3]_1 + [A_4]_1$, where $([A_1]_1, [A_2]_1, [A_3]_1, [A_4]_1, a_{11}, a_{12}, a_{21}, a_{22}, a_{31})$ represents $([U]_1, [U]_1, [M]_1, [S]_1, r_1, s_1, r_2, r_1, s_2)$ and $(c_{11}, c_{12}, c_{21}, c_{22}, c_{31}) = (1, 1, 2, 1, -1)$.

[3] The automorphic signature in [5] seems not well-formed at first glance since the exponent of a group element in the signature is $1/(x + c)$ where x is the signing key and c is a randomness. However, we can easily convert it to a well-formed one by switching $(x, c, x + c)$ to $(x, c' - x, c')$ where c' is a randomness.

Lemma 3.2.1. *([103]) Let $P \in F[x]$ be a non-zero polynomial of total degree $d \geq 0$ over a field, S a finite subset of F, and r a randomness uniformly chosen from S. Then, we have*

$$\Pr[P(r) = 0] \leq d/|S|.$$

This lemma indicates that a polynomial of degree d over $\mathbb{Z}_p$ has at most d roots.

Proof of Theorem 3.2.2. We divide the proof of Theorem 3.2.2 into two parts. In the first part, we show that any well-formed SPS scheme can be converted into a γ-TS scheme satisfying the conditions (a) and (b) of the SKSP property in Definition 3.2.2. In the second part, we prove that the converted TS scheme also satisfies the condition (c).

Part I. Let a group element in a signature be generated as Equation (3.1). For all i such that $\{a_{ij}\}_j$ contains a set of variables in the signing key, denoted by $\{s_{ij}\}_j$, we use c'_{ij} to denote the exponent of s_{ij} in Equation (3.1), and do the following conversion.

- If $[A_i]_b$ is in the message, then we add $[(\prod_j s_{ij}^{c'_{ij}})]_b$ to the signing key.
- Otherwise (i.e., if $[A_i]_b$ is in the signing key or the verification key), then we add $[(\prod_j s_{ij}^{c'_{ij}})A_i]_b$ to the signing key,

For all other group elements in the signature, we execute the same conversions. Then we remove all elements in $\mathbb{Z}_p$, all repeated elements, and elements never used in signing procedures from the original signing key, and set the original signing key as the trapdoor key.

By using the new signing key, we can generate a signature consisting of group elements in the forms of Equation (3.1) when taking as input a message consisting of $M_1, M_2, \ldots, N_1, N_2, \ldots \in \mathbb{Z}_p$, which forms the signing algorithm for the resulting γ-TS scheme. Furthermore, taking as input $[M_1]_1, [M_2]_1, \ldots, [N_1]_2, [N_2]_2, \ldots$ and the trapdoor key, we can generate the same signature if the randomness is the same, by using the original signing algorithm. As a result, we have obtained a γ-TS scheme for $\gamma(\vec{M}, \vec{N}) = ([\vec{M}]_1, [\vec{N}]_2)$.

It is straightforward to see that in this γ-TS scheme, the verification keys, signing keys, and signatures consist only of group elements in $\mathbb{G}_1$ and $\mathbb{G}_2$ and the verification consists only of evaluating membership in $\mathbb{G}_1$ and $\mathbb{G}_2$ and relations described by PPEs. This completes the first part of the proof. Here, the verification key size, signature size, and number of PPEs do not change during the conversion, while the signing key size changes depending on the concrete construction of SPS.[4]

Part II. Next we prove that for the above γ-TS scheme, there exists an algorithm that can check the correctness of signing keys w.r.t. verification keys by using only PPEs.

[4] In the worst case, the resulting signing key size is the total number of elements in all $\{[A_i]_b\}_i$.

Since a group element in the signature is computed as Equation (3.1), and a group element in the message $[M]_1$ or $[N]_2$ can be treated as $M[1]_1$ or $N[1]_2$, a PPE in the verification algorithm can be written as

$$\sum_i \Big(\prod_j {x_{ij}}^{d_{ij}} \Big) [X_i]_T = [0]_T, \tag{3.2}$$

where $\{x_{ij}\}_{ij}$ denotes elements in the randomness, exponents of the message, and integer constants, $\{d_{ij}\}_{ij}$ denotes integer constants (independent of the security parameter), and $\{[X_i]_T\}_i$ denotes pairings between elements in the verification key and the signing key. Here, elements in $\{x_{ij}\}_{ij}$ may represent the same variables, and the same argument is made for $\{[X_i]_T\}_i$.

We now show how to obtain PPEs that check the correctness of signing keys w.r.t. verification keys as follows. Let $\mathcal{E}$ be the set of all the *distinct* variables in $\{x_{ij}\}_{ij}$ (not including constants). Then for any $x \in \mathcal{E}$, we rewrite Equation (3.2) as

$$\sum_i x^{d_i} [Y_i]_T = [0]_T, \tag{3.3}$$

where $\{d_i\}_i$ are fixed polynomials (in the security parameter) and $\{[Y_i]_T\}_i$ denotes elements in $\mathbb{G}_T$. Since the SPS scheme is well-formed, the left hand side of Equation (3.3) can be treated as a polynomial in x by fixing all $[Y_i]_T$. We rewrite Equation (3.3) as

$$[P_0]_T + x^1 [P_1]_T + \ldots + x^n [P_n]_T = [0]_T, \tag{3.4}$$

for some fixed polynomial n, where $[P_k]_T$ denotes the sum of coefficients of x^k. According to the definition of well-formed SPS, since the space of x is super-polynomial (in the security parameter) and n is a polynomial (in the security parameter), the number of possible values of x must be larger than n for sufficiently large security parameters. As a result, if Equation (3.4) holds for all possible value of x, we have

$$[P_0]_T = [0]_T, \ [P_1]_T = [0]_T, \ \ldots, \ [P_n]_T = [0]_T, \tag{3.5}$$

or the number of roots of Equation (3.4) could be larger than n, which is against Schwartz-Zippel Lemma. On the other hand, it is obvious that if PPEs in (3.5) hold, Equation (3.4) holds for any x. For each $[P_i]_T = 0$, we cancel another variable in $\mathcal{E}$ in the same way. Recursively, all the variables in PPEs in the verification algorithm can be cancelled, and we finally obtain a sequence of PPEs of the form

$$\sum_i c'_i [X'_i]_T = [0]_T,$$

where $\{c'_i\}_i$ denotes integer constants, and $\{[X'_i]_T\}_i$ denotes pairings between elements in the verification key and the signing key, and elements in $\{[X'_i]_T\}_i$ may represent the same variables. Since such collection of PPEs hold *if and only if* PPEs in the verification algorithm

hold for all possible randomness and messages, we obtain an algorithm that takes as input verification/signing key pairs and check their correctness using this collection of PPEs.[5]

In conclusion, any well-formed SPS scheme can be converted into an SKSP-TS scheme, completing the proof of Theorem 3.2.2. □

Remark 1. The latter half of the proof can also be adopted to show that for a well-formed SPS scheme, if signing keys consist only of group elements, then it is an FSPS scheme.

Remark 2. It is not hard to see that an SPS scheme whose messages are of the form, e.g., $([m_1]_1, [m_2]_2)$ where $m_1 = m_2 + 1$ and $m_1, m_2 \in \mathbb{Z}_p$, is not well-formed. However, such a scheme can be easily converted to a well-formed one by letting messages be of the form $([m_1]_1, [m_1]_2)$ and compute $[m_1 + 1]_2$ in signing and verification. Furthermore, if we relax the definition of well-formed SPS by allowing messages in the form of, e.g., $[m_1]_1, [m_2]_1, [m_3]_1, [m_1 m_2 + m_3]_1$, where the spaces of m_1, m_2, and m_3 have super-polynomially large spaces, Theorem 3.2.2 still holds.

3.2.4 Instantiations of Trapdoor Signature

UF-CMA secure TS scheme. Using the conversion described in the proof of Theorem 3.2.2, we can convert well-formed SPSs into SKSP-TSs. For ease of understanding, we give an instantiation of γ-TS $\Sigma = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{TDSign})$ in Fig. 3.1, which is converted from the SPS scheme (denoted by $\mathcal{T}_\Sigma = (\mathsf{Setup}, \mathcal{T}_{\mathsf{Gen}}, \mathsf{TDSign}, \mathsf{Verify})$) proposed by Kiltz et al. [79]. Here, $\mathcal{T}_\Sigma$ is UF-CMA secure under the $\mathcal{D}_k$-MDDH assumptions and $\gamma : \mathbb{Z}_p^n \mapsto \mathbb{G}_1^n$ is defined by $\gamma(x_1, \ldots, x_n) = ([x_1]_1, \ldots, [x_n]_1)$, where n denotes the number of group elements in a message.

To generate a signature of $\mathcal{T}_\Sigma$, $\sigma_1 = [(1, \vec{m}^\top)]_1 \mathbf{K} + \vec{r}^\top [\mathbf{P}_0 + \tau \mathbf{P}_1]_1$ is the only part that needs to be operated by using "$\mathbb{Z}_p$-elements" $\mathbf{K} \in \mathbb{Z}_p^{(n+1)\times(k+1)}$ of the signing key. Following our conversion, we replace $\mathbf{K}$ with $[\mathbf{K}]_1$ in the signing key, and keep the original signing key as the trapdoor key. By using $[\mathbf{K}]_1$, we can compute σ_1 as $(1, \vec{m}^\top)[\mathbf{K}]_1 + \vec{r}^\top [\mathbf{P}_0 + \tau \mathbf{P}_1]_1$.

[5] The number of PPEs we finally obtain is smaller than number of elements in $\{[X_i]_T\}_i$ in PPEs of the form Equation (3.2).

Furthermore, we obtain PPEs that check the correctness of signing keys as follows.

$$
\begin{aligned}
& e(\sigma_1, [\mathbf{A}]_2) = e([(1, \vec{m}^\top)]_1, [\mathbf{C}]_2) + e(\sigma_2, [\mathbf{C}_0]_2) + e(\sigma_3, [\mathbf{C}_1]_2), \\
\Rightarrow & e((1, \vec{m}^\top)[\mathbf{K}]_1 + \vec{r}^\top[\mathbf{P}_0 + \tau\mathbf{P}_1]_1, [\mathbf{A}]_2) = e([(1, \vec{m}^\top)]_1, [\mathbf{C}]_2) + e(\vec{r}^\top[\mathbf{B}^\top]_1, [\mathbf{C}_0]_2) \\
& \quad + e(\vec{r}^\top[\mathbf{B}^\top\tau]_1, [\mathbf{C}_1]_2), && \text{(Rewrite first equation in } \mathsf{Verify}\text{)} \\
\Rightarrow & \begin{cases} e((1, \vec{m}^\top)[\mathbf{K}]_1 + \vec{r}^\top[\mathbf{P}_0]_1, [\mathbf{A}]_2) = e([(1, \vec{m}^\top)]_1, [\mathbf{C}]_2) + e(\vec{r}^\top[\mathbf{B}^\top]_1, [\mathbf{C}_0]_2), \\ e(\vec{r}^\top[\mathbf{P}_1]_1, [\mathbf{A}]_2) = e(\vec{r}^\top[\mathbf{B}^\top]_1, [\mathbf{C}_1]_2), \end{cases} && \text{(Cancelling } \tau\text{)} \\
\Rightarrow & \begin{cases} e((1, \vec{m}^\top)[\mathbf{K}]_1, [\mathbf{A}]_2) = e([(1, \vec{m}^\top)]_1, [\mathbf{C}]_2), \\ e([\mathbf{P}_0]_1, [\mathbf{A}]_2) = e([\mathbf{B}^\top]_1, [\mathbf{C}_0]_2), \\ e([\mathbf{P}_1]_1, [\mathbf{A}]_2) = e([\mathbf{B}^\top]_1, [\mathbf{C}_1]_2), \end{cases} && \text{(Cancelling } \vec{r}\text{)} \\
\overset{*}{\Rightarrow} & \begin{cases} e([\mathbf{K}]_1, [\mathbf{A}]_2) = e([1]_1, [\mathbf{C}]_2)), \\ e([\mathbf{P}_0]_1, [\mathbf{A}]_2) = e([\mathbf{B}^\top]_1, [\mathbf{C}_0]_2), \\ e([\mathbf{P}_1]_1, [\mathbf{A}]_2) = e([\mathbf{B}^\top]_1, [\mathbf{C}_1]_2). \end{cases} && \text{(Cancelling } \vec{m}\text{)}
\end{aligned}
$$

One can see that when fixing $\vec{m}$ and $\vec{r}$, the PPEs generated in the second step (by cancelling τ) hold if and only if the PPE generated in the first step holds for all possible τ, due to the Schwartz-Zippel Lemma. In the same way, we have that the PPEs generated in the third (respectively, the last) step hold if and only if the PPEs generated in the second (respectively, the third) step hold for all possible $\vec{r}$ (respectively, $\vec{m}$). As a result, the three resulting PPEs hold if and only if the first equation in Verify hold for all possible $\vec{m}$, $\vec{r}$, and τ (i.e., all possible message and randomness elements).

Then we rewrite the second equation $e(\sigma_2, \sigma_4) = e(\sigma_3, [1]_2)$ as $e(\vec{r}^\top[\mathbf{B}^\top]_1, [\tau]_2) = e(\vec{r}^\top[\mathbf{B}^\top\tau]_1, [1]_2)$. By cancelling $\vec{r}$ and τ, we obtain $e([\mathbf{B}^\top]_1, [1]_2) = e([\mathbf{B}^\top]_1, [1]_2)$, which is trivial.[6]

Finally, we obtain the algorithm $\mathsf{VerifySK}$ checking correctness of signing keys w.r.t. verification keys via the above three PPEs (derived from $\overset{*}{\Rightarrow}$), which makes our scheme satisfy the condition (c) of Definition 3.2.2.

Theorem 3.2.3. *The instantiation described in Fig. 3.1 is a UF-CMA secure γ-SKSP-TS scheme under the $\mathcal{D}_k$-MDDH assumptions.*

The SKSP property of this instantiation is implied by Theorem 3.2.2 and UF-CMA security is implied by Theorem 3.2.1. The formal proof is as follows.

Proof of Theorem 3.2.3. Since $\mathcal{T}_\Sigma$ is actually the same as the UF-CMA secure signature scheme in [79], to prove the correctness, we only have to show that $\mathsf{Sign}(sk, m; r) = \mathsf{TDSign}(tk, \gamma(m); r)$. Proving this is straightforward since Sign and TDSign run in the same way except that to compute $[(1, \vec{m}^\top)\mathbf{K}]_1$, Sign makes use of $\vec{m}$ and $[\mathbf{K}]_1$ while TDSign makes use of $[\vec{m}]_1$ and $\mathbf{K}$.

[6] Note that for simplicity, we sometimes directly canceled vectors in the above conversion, instead of following the proof of Theorem 3.2.2 to cancel elements one by one.

$\mathsf{Setup}(1^\lambda)$:
$par = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, [1]_1, [1]_2) \leftarrow \mathcal{G}(1^\lambda)$.
For preliminary-fixed $n \in \mathbb{N}$,
define $\mathcal{M} = \mathbb{Z}_p^n$ and $\mathcal{M}_\gamma = \mathbb{G}_1^n$.
Define γ by $\gamma(m_1, \ldots, m_n) = ([m_1]_1, \ldots [m_n]_1)$.
Return par.

$\mathsf{Gen}(par)$:
$\mathbf{A}, \mathbf{B} \leftarrow \mathcal{D}_k$, $\mathbf{K} \leftarrow \mathbb{Z}_p^{(n+1)\times(k+1)}$, $\mathbf{K}_0, \mathbf{K}_1 \leftarrow \mathbb{Z}_p^{(k+1)\times(k+1)}$,
$\mathbf{C} = \mathbf{K}\mathbf{A} \in \mathbb{Z}_p^{(n+1)\times k}$,
$\mathbf{C}_0 = \mathbf{K}_0\mathbf{A} \in \mathbb{Z}_p^{(k+1)\times k}$, $\mathbf{C}_1 = \mathbf{K}_1\mathbf{A} \in \mathbb{Z}_p^{(k+1)\times k}$,
$\mathbf{P}_0 = \mathbf{B}^\top\mathbf{K}_0 \in \mathbb{Z}_p^{k\times(k+1)}$, $\mathbf{P}_1 = \mathbf{B}^\top\mathbf{K}_1 \in \mathbb{Z}_p^{k\times(k+1)}$.
$pk = ([\mathbf{C}_0]_2, [\mathbf{C}_1]_2, [\mathbf{C}]_2, [\mathbf{A}]_2)$,
$sk = (\boxed{[\mathbf{K}]_1}, [\mathbf{P}_0]_1, [\mathbf{P}_1]_1, [\mathbf{B}]_1)$,
$tk = \boxed{(\mathbf{K}, [\mathbf{P}_0]_1, [\mathbf{P}_1]_1, [\mathbf{B}]_1)}$.
Return (pk, sk) and tk.

$\mathsf{VerifySK}(pk, sk)$:
Return 1 if $e([\mathbf{K}]_1, [\mathbf{A}]_2) = e([1]_1, [\mathbf{C}]_2)$,
$e([\mathbf{P}_0]_1, [\mathbf{A}]_2) = e([\mathbf{B}^\top]_1, [\mathbf{C}_0]_2)$,
and $e([\mathbf{P}_1]_1, [\mathbf{A}]_2) = e([\mathbf{B}^\top]_1, [\mathbf{C}_1]_2)$.
Return 0 otherwise.

$\mathsf{Sign}(sk, \vec{m})$:
$\vec{r} \leftarrow \mathbb{Z}_p^k$, $\tau \leftarrow \mathbb{Z}_p$,
$\sigma_1 = \boxed{(1, \vec{m}^\top)[\mathbf{K}]_1} + \vec{r}^\top[\mathbf{P}_0 + \tau\mathbf{P}_1]_1$,
$\sigma_2 = \vec{r}^\top[\mathbf{B}^\top]_1$, $\sigma_3 = \vec{r}^\top[\mathbf{B}^\top\tau]_1$,
$\sigma_4 = [\tau]_2 \in \mathbb{G}_2$.
Return $(\sigma_1, \sigma_2, \sigma_3, \sigma_4) \in \mathbb{G}_1^{1\times(k+1)} \times \mathbb{G}_1^{1\times(k+1)} \times \mathbb{G}_1^{1\times(k+1)} \times \mathbb{G}_2$.

$\mathsf{Verify}(pk, [\vec{m}]_1, \sigma)$:
Parse $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4)$,
Return 1 if
$e(\sigma_1, [\mathbf{A}]_2) = e([(1, \vec{m}^\top)]_1, [\mathbf{C}]_2) + e(\sigma_2, [\mathbf{C}_0]_2) + e(\sigma_3, [\mathbf{C}_1]_2)$
and $e(\sigma_2, \sigma_4) = e(\sigma_3, [1]_2)$.
Return 0 otherwise.

$\mathsf{TDSign}(tk, [\vec{m}]_1)$:
$\vec{r} \leftarrow \mathbb{Z}_p^k$, $\tau \leftarrow \mathbb{Z}_p$.
$\sigma_1 = \boxed{[(1, \vec{m}^\top)]_1\mathbf{K}} + \vec{r}^\top[\mathbf{P}_0 + \tau\mathbf{P}_1]_1$
$\sigma_2 = \vec{r}^\top[\mathbf{B}^\top]_1$, $\sigma_3 = \vec{r}^\top[\mathbf{B}^\top\tau]_1$,
$\sigma_4 = [\tau]_2 \in \mathbb{G}_2$.
Return $(\sigma_1, \sigma_2, \sigma_3, \sigma_4) \in \mathbb{G}_1^{1\times(k+1)} \times \mathbb{G}_1^{1\times(k+1)} \times \mathbb{G}_1^{1\times(k+1)} \times \mathbb{G}_2$.

Figure 3.1: A UF-CMA secure γ-TS scheme adapted from [79, Section 4.2]. The boxes indicate the main differences from the original scheme in [79].

Furthermore, since $\mathcal{T}_\Sigma$ is well-formed (see Definition 3.2.5) and Σ is converted from $\mathcal{T}_\Sigma$ in the way we described in the proof of Theorem 3.2.2, this TS scheme is SKSP.

The UF-CMA security of Σ is directly implied by the UF-CMA security of $\mathcal{T}_\Sigma$ according to Theorem 3.2.1, while the security of $\mathcal{T}_\Sigma$ is implied by the $\mathcal{D}_k$-MDDH assumptions according to [79], completing the proof of Theorem 3.2.3. □

UF-otRMA secure TS scheme. In Fig. 3.2, we give another instantiation of TS which satisfies UF-otRMA security under the $\mathcal{D}_k$-MDDH assumptions. This scheme is converted from the UF-otRMA secure SPS scheme in [79]. The proof of correctness is straightforward and the correctness of a signing key w.r.t. a verification key can be verified by $\mathsf{VerifySK}$ via $e([\mathbf{K}]_1, [\overline{\mathbf{A}}]_2) = e([1]_1, [\mathbf{C}]_2)$.

Unlike UF-CMA security proved in Theorem 3.2.1, the UF-otRMA security of $\Sigma = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{TDSign})$ is not automatically implied by the UF-otRMA security of $\mathcal{T}_\Sigma = (\mathsf{Setup}, \mathcal{T}_{\mathsf{Gen}}, \mathsf{TDSign}, \mathsf{Verify})$. However, according to [79], the proof of the UF-otRMA security of $\mathcal{T}_\Sigma$ remains valid even when an adversary sees the exponents of the messages from the signing oracle, which implies the UF-otRMA security of Σ. We refer the reader to [79] for details of the proof.

3.3 (Two-tier) Signature with Auxiliary Key(s)

In this section, we introduce *AKS* which are used as building blocks to achieve our generic construction of FSPS. In Section 3.3.1, we give the definition of AKS, define their properties,

$\mathsf{Setup}(1^\lambda)$: $par = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, [1]_1, [1]_2) \leftarrow \mathcal{G}(1^\lambda)$. $\mathcal{M} = \mathbb{Z}_p^n$, $\mathcal{M}_\gamma = \mathbb{G}_1^n$. For preliminary-fixed $n \in \mathbb{N}$, define γ by $\gamma(m_1, \ldots, m_n) = ([m_1]_1, \ldots [m_n]_1)$. Return par.	$\mathsf{Sign}(sk, \vec{m})$: $\sigma = \boxed{(1, \vec{m}^\top)[\mathbf{K}]_1}$. Return $\sigma \in \mathbb{G}_1^{1\times k}$.
$\mathsf{Gen}(par)$: $\mathbf{A} \leftarrow \mathcal{D}_k$, $\mathbf{K} \leftarrow \mathbb{Z}_p^{(n+1)\times k}$, $\mathbf{C} = \mathbf{K}\overline{\mathbf{A}} \in \mathbb{Z}_p^{(n+1)\times k}$, $pk = ([\mathbf{C}]_2, [\overline{\mathbf{A}}]_2)$, $sk = \boxed{[\mathbf{K}]_1}$, $tk = \boxed{\mathbf{K}}$. Return (pk, sk) and tk.	$\mathsf{Verify}(pk, [\vec{m}]_1, \sigma)$: Return 1 if $e(\sigma, [\overline{\mathbf{A}}]_2) = e([(1, \vec{m}^\top)]_1, [\mathbf{C}]_2)$. Return 0 otherwise.
$\mathsf{VerifySK}(pk, sk)$: Return 1 if $e([\mathbf{K}]_1, [\overline{\mathbf{A}}]_2) = e([1]_1, [\mathbf{C}]_2)$. Return 0 otherwise.	$\mathsf{TDSign}(tk, [\vec{m}]_1)$: $\sigma = \boxed{[(1, \vec{m}^\top)]_1\mathbf{K}}$. Return $\sigma \in \mathbb{G}_1^{1\times k}$.

Figure 3.2: A UF-otRMA secure γ-SKSP-TS scheme adapted from [79, Section 5.2]. The boxes indicate the main differences from the original scheme in [79].

and give an instantiation of AKS. In Section 3.3.2, we extend AKS to TT-AKS and give an instantiation of TT-AKS.

3.3.1 Signature with Auxiliary Key

Definition. Roughly speaking, a γ-AKS scheme is a signature scheme in which the key generation algorithms additionally generate auxiliary keys, and the verification key space and the auxiliary key space have a special (but natural) structure related with γ.

Definition 3.3.1 (γ-signature with auxiliary key (γ-AKS))**.** *A signature scheme* $\Sigma = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *with verification key space* $\mathcal{P}_\gamma$ *is said to be a* γ-AKS *scheme for an efficiently computable bijection* $\gamma : \mathcal{P} \mapsto \mathcal{P}_\gamma$ *if in addition to the verification/signing key pair* (pk, sk), Gen *also outputs an auxiliary key* $ak \in \mathcal{P}$ *such that* $pk = \gamma(ak)$.

Security. The UF-(ot)CMA security and UF-(ot)RMA security of γ-AKSs are exactly the same as those of standard signatures except that Gen additionally generates ak.

Key generation algorithm $\mathcal{U}_{\mathsf{Gen}}$. Similarly to $\mathcal{T}_{\mathsf{Gen}}$ defined in Section 3.2.1, we use $\mathcal{U}_{\mathsf{Gen}}$ to denote an algorithm that runs Gen, which is the key generation algorithm of a γ-AKS scheme, in the following way.

Taking as input a public parameter par, $\mathcal{U}_{\mathsf{Gen}}$ gives par to Gen and obtains an output $((pk, sk), ak)$. Then $\mathcal{U}_{\mathsf{Gen}}$ outputs (pk, sk) as a verification/signing key pair, without outputting ak. We use $\mathcal{U}_\Sigma$ to denote $(\mathsf{Setup}, \mathcal{U}_{\mathsf{Gen}}, \mathsf{Sign}, \mathsf{Verify})$ when $\Sigma = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$.

Just like SPSs, we consider γ-AKSs with the SP property.

Definition 3.3.2 (γ-SP-AKS)**.** *A* γ*-AKS scheme* Σ *is said to be a* γ-SP-AKS *scheme if* $\mathcal{U}_\Sigma$ *is an SPS scheme.*

Converting SPSs into SP-AKSs. It is straightforward to see that any SPS scheme with an algebraic key generation algorithm, verification keys of which are supposed to be of the form $([\vec{u}]_1, [\vec{v}]_2)$, can be converted into a γ-SP-AKS scheme, where γ is defined by $\gamma(\vec{u}, \vec{v}) = ([\vec{u}]_1, [\vec{v}]_2)$, since we can force the setup of any SPS to output no common parameter except for the bilinear map description and let the key generation algorithm additionally output $(\vec{u}, \vec{v})$.

We now define the random auxiliary key property for AKSs. This property is only useful when combining AKSs with UF-RMA secure TSs (rather than UF-CMA ones).

Definition 3.3.3 (Random auxiliary key property)**.** *A γ-AKS scheme* (Setup, Gen, Sign, Verify) *with an auxiliary key space $\mathcal{P}$ is said to satisfy the* random auxiliary key *property if there exists an additional algorithm* AKGen *such that* AKGen *takes as input par and an auxiliary key ak, and outputs a verification/signing key pair (pk, sk) where $\gamma(ak) = pk$. Furthermore, for any PPT adversary $\mathcal{A}$ and all $\lambda \in \mathbb{N}$, we have*

$$|\Pr[par \leftarrow \mathsf{Setup}(1^\lambda) : \mathcal{A}^{\mathrm{GenO}}(par) = 1] - \Pr[par \leftarrow \mathsf{Setup}(1^\lambda) : \mathcal{A}^{\mathrm{AKGenO}}(par) = 1]| \leq negl(\lambda),$$

where GenO *runs $((pk, sk), ak) \leftarrow \mathsf{Gen}(par)$, and returns (pk, sk, ak), and* AKGenO *uniformly chooses ak from $\mathcal{P}$, runs $(pk, sk) \leftarrow \mathsf{AKGen}(par, ak)$, and returns (pk, sk, ak).*

Instantiation of AKS. Now we give an instantiation of AKS satisfying UF-otCMA security under the $\mathcal{U}_k$-MDDH assumptions (see Section 2.3) in Fig. 3.3. This signature scheme is actually the same as the UF-otCMA secure signature scheme in [79] except that Gen additionally generates exponents of a verification key as an auxiliary key. For this instantiation, the bijection γ is defined by $\gamma(\mathbf{X}) = [\mathbf{X}]_2 \in \mathbb{G}_2^{(n+1)\times k} \times \mathbb{G}_2^{(k+1)\times k}$ for n which denotes the length of a message.

We refer the reader to [79] for the proof of the UF-otCMA security of this instantiation.

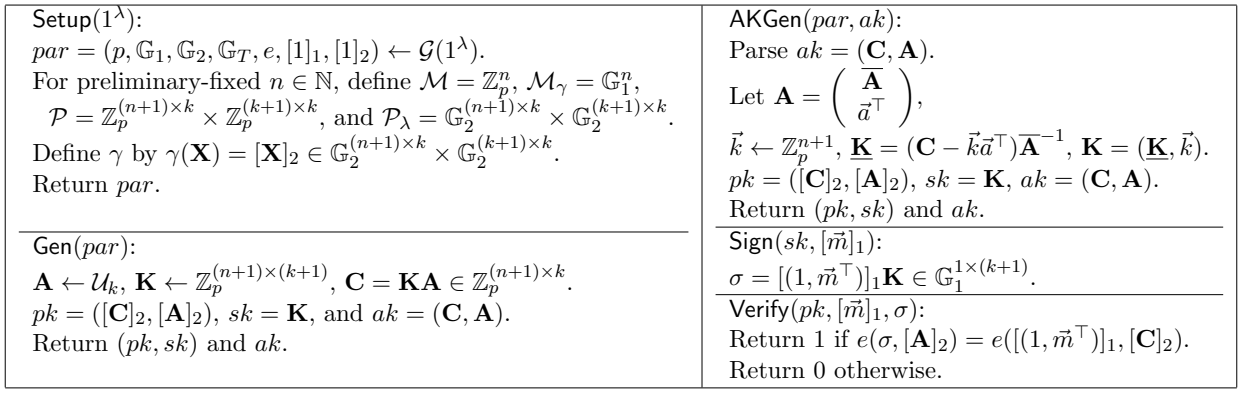

Figure 3.3: A UF-otCMA secure γ-SP-AKS scheme adapted from [79, Section 3].

Theorem 3.3.1. *The instantiation described in Fig. 3.3 satisfies the random auxiliary key property.*

The proof follows from the fact that when the distribution of $\mathbf{C}$ is uniform, the distribution of $\underline{\mathbf{K}} = (\mathbf{C} - \vec{k}\vec{a}^\top)\overline{\mathbf{A}}^{-1}$ is uniform as well. The formal proof is as follows.

Proof of Theorem 3.3.1. Parsing the exponent of a signing key $\mathbf{K} \in \mathbb{Z}_p^{(n+1)\times(k+1)}$ as $(\underline{\mathbf{K}}, \vec{k}) \in \mathbb{Z}_p^{(n+1)\times k} \times \mathbb{Z}_p^{(n+1)\times 1}$ and part of the exponent of the corresponding verification key $\mathbf{A} \in \mathbb{Z}_p^{(k+1)\times k}$ as $\begin{pmatrix} \overline{\mathbf{A}} \\ \vec{a}^\top \end{pmatrix}$, we have that $\underline{\mathbf{K}}\overline{\mathbf{A}} + \vec{k}\vec{a}^\top = \mathbf{C}$. If $\mathbf{C}$ is randomly chosen from $\mathbb{Z}_p^{(n+1)\times k}$, we have that $\underline{\mathbf{K}}\overline{\mathbf{A}} = (\mathbf{C} - \vec{k}\vec{a}^\top) \in \mathbb{Z}_p^{(n+1)\times k}$ looks random and independent of $\vec{k}$ (if we do not see $\mathbf{C}$ at first). Equivalently, we have that $\underline{\mathbf{K}} = (\mathbf{C} - \vec{k}\vec{a}^\top)\overline{\mathbf{A}}^{-1}$ is indistinguishable from a random matrix from $\mathbb{Z}_p^{(n+1)\times k}$. As a result, $\mathbf{K}$ generated by $\mathsf{AKGen}(par, ak)$ is indistinguishable from a random matrix in $\mathbb{Z}_p^{(n+1)\times(k+1)}$ if ak is randomly chosen from $\mathbb{Z}_p^{(n+1)\times k} \times \mathcal{U}_k$, which means that the tuples output by Gen are indistinguishable from the ones output by AKGen as long as AKGen takes as input randomly chosen auxiliary keys, completing the proof of Theorem 3.3.1. □

3.3.2 Two-tier Signature with Auxiliary Keys

Definition. Besides AKSs, we also give the definition of (γ_p, γ_s)-*TT-AKS*, which is the same as that of two-tier signature [23, 3, 80] except that the key generation algorithms additionally generate primary/secondary auxiliary keys. The primary/secondary verification key space and the primary/secondary auxiliary key space have a special (but natural) structure related with γ_p/γ_s. Combining SP-TT-AKSs with SKSP-TSs enables us to obtain more efficient instantiations of FSPS and FAS.

Definition 3.3.4 ((γ_p, γ_s)-TT-AKS)**.** *A* (γ_p, γ_s)-TT-AKS *scheme consists of five polynomial-time algorithms* Setup, PGen, SGen, TTSign, *and* $\mathsf{TTVerify}$.

- Setup *is a randomized algorithm that takes as input* 1^λ*, and outputs a public parameter* par*, which determines the message space* $\mathcal{M}$*, the primary/secondary verification key spaces* $\mathcal{P}_\gamma/\mathcal{S}_\gamma$*, the primary/secondary auxiliary key spaces* $\mathcal{P}/\mathcal{S}$*, and the efficiently computable bijections* $\gamma_p : \mathcal{P} \mapsto \mathcal{P}_\gamma$ *and* $\gamma_s : \mathcal{S} \mapsto \mathcal{S}_\gamma$*.*

- PGen *is a randomized algorithm that takes as input* par*, and outputs a primary verification/signing key pair* (Ppk, Psk) *where* $Ppk \in \mathcal{P}_\gamma$ *and a primary auxiliary key* $Pak \in \mathcal{P}$*.*

- SGen *is a randomized algorithm that takes as input a primary verification/signing key pair* (Ppk, Psk) *and a primary auxiliary key* Pak*, and outputs a secondary verification/signing key pair* (opk, osk) *where* $opk \in \mathcal{S}_\gamma$ *and a secondary auxiliary key* $oak \in \mathcal{S}$*.*

- TTSign *is a randomized algorithm that takes as input a primary signing key* Psk*, a secondary signing key* osk*, and a message* m*, and returns a signature* σ*.*

- $\mathsf{TTVerify}$ *is a deterministic algorithm that takes as input a primary verification key* Ppk*, a secondary verification key* opk*, a message* m*, and a signature* σ*, and returns* 1 *(accept) or* 0 *(reject).*

Correctness *is satisfied if for all* $\lambda \in \mathbb{N}$*,* $par \leftarrow \mathsf{Setup}(1^\lambda)$*,* $((Ppk, Psk), Pak) \leftarrow \mathsf{PGen}(par)$*, and* $((opk, osk), oak) \leftarrow \mathsf{SGen}(Ppk, Psk, Pak)$*, we have*

(*a*) $\mathsf{TTVerify}(Ppk, opk, m, \mathsf{TTSign}(Psk, osk, m)) = 1$ *for all messages* $m \in \mathcal{M}$*, and*

(*b*) $\gamma_p(Pak) = Ppk$ *and* $\gamma_s(oak) = opk$*.*

Unlike the definition of standard two-tier signature, SGen takes as input (Ppk, Psk, Pak) (instead of (Ppk, Psk)) in the above definition. However, the interface of SGen is not essentially changed since Pak can be treated as part of Psk.

Security. Now we give the definition of unforgeability against two-tier chosen message attacks (UF-TT-CMA).

Definition 3.3.5 (UF-TT-CMA [80])**.** *A TT-AKS scheme* $(\mathsf{PGen}, \mathsf{SGen}, \mathsf{TTSign}, \mathsf{TTVerify})$ *is said to be* UF-TT-CMA secure *if for any PPT adversary* $\mathcal{A}$*, we have*

$$\Pr[par \leftarrow \mathsf{Setup}(1^\lambda), ((Ppk, Psk), Pak) \leftarrow \mathsf{PGen}(par), (i^*, m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathrm{TTSignO}(\cdot)}(Ppk) : (i^*, m) \in \mathcal{TQ}_m \wedge m^* \neq m \wedge \mathsf{TTVerify}(Ppk, opk_{i^*}, m^*, \sigma^*) = 1] \leq negl(\lambda),$$

where $\mathrm{TTSignO}(\cdot)$ *is the signing oracle that takes a message* $m \in \mathcal{M}$ *as input, runs* $i = i + 1$ *(initialized with* 0*), samples* $(opk_i, osk_i) \leftarrow \mathsf{SGen}(Ppk, Psk, Pak)$*, and computes* $\sigma \leftarrow \mathsf{TTSign}(Psk, osk_i, m)$*. Then it adds* (i, m) *to* $\mathcal{TQ}_m$ *(initialized with* $\emptyset$*) and returns* (opk_i, σ)*.*

Next we define the SP property of TT-AKSs as follows.

Definition 3.3.6 (Structure-preserving TT-AKS (SP-TT-AKS))**.** *A TT-AKS scheme is said to be* structure-preserving *over a bilinear group generator* $\mathcal{G}$ *if we have*

(*a*) *a public parameter includes a group description* gk *generated by* $\mathcal{G}$*,*

(*b*) *primary and secondary verification keys consist only of group elements in* $\mathbb{G}_1$ *and* $\mathbb{G}_2$*,*

(*c*) *messages consist only of group elements in* $\mathbb{G}_1$ *and* $\mathbb{G}_2$*, and*

(*d*) *the verification algorithm consists only of evaluating membership in* $\mathbb{G}_1$ *and* $\mathbb{G}_2$ *and relations described by PPEs.*

Converting SP two-tier signatures into SP-TT-AKSs. Like SP-AKSs, an SP two-tier signature scheme, primary and secondary verification keys of which are supposed to be of the form $([\vec{u}]_1, [\vec{v}]_2)$ and $([\vec{u}']_1, [\vec{v}']_2)$ respectively, can be converted into a (γ_p, γ_s)-SP-TT-AKS scheme, where γ_p and γ_s are defined as $\gamma_p(\vec{u}, \vec{v}) = ([\vec{u}]_1, [\vec{v}]_2)$ and $\gamma_s(\vec{u}', \vec{v}') = ([\vec{u}']_1, [\vec{v}']_2)$ respectively, as long as the key generation algorithms are algebraic and primary signing keys consist only of elements in $\mathbb{Z}_p$.[7]

We define the random primary and secondary auxiliary key properties of TT-AKSs as follows. Like the random auxiliary key property, these properties are useful only when combining TT-AKSs with UF-RMA secure TSs (rather than UF-CMA ones).

Definition 3.3.7 (Random primary/secondary auxiliary key properties)**.** *A (γ_p, γ_s)-TT-AKS scheme* $(\mathsf{Setup}, \mathsf{PGen}, \mathsf{SGen}, \mathsf{TTSign}, \mathsf{TTVerify})$ *is said to satisfy the* random primary auxiliary key *property if there exists an additional polynomial-time algorithm* AKPGen *that takes as input par and a primary auxiliary key Pak, and outputs a primary verification/signing key pair (Ppk, Psk) where $\gamma_p(Pak) = Ppk$. Furthermore, for any PPT adversary $\mathcal{A}$ and all $\lambda \in \mathbb{N}$, we have*

$$|\Pr[par \leftarrow \mathsf{Setup}(1^\lambda) : \mathcal{A}^{\mathrm{PGenO}}(par) = 1] - \Pr[par \leftarrow \mathsf{Setup}(1^\lambda) : \mathcal{A}^{\mathrm{AKPGenO}}(par) = 1]| \leq negl(\lambda),$$

where PGenO *runs $((Ppk, Psk), Pak) \leftarrow \mathsf{PGen}(par)$ and returns $((Ppk, Psk), Pak)$, and* AKPGenO *uniformly chooses Pak from the primary auxiliary key space $\mathcal{P}$, runs $(Ppk, Psk) \leftarrow \mathsf{AKPGen}(par, Pak)$, and returns $((Ppk, Psk), Pak)$.*

Furthermore, it is said to satisfy the random secondary auxiliary key property *if there exists another polynomial-time algorithm* AKSGen *that takes as input a primary verification/signing key pair (Ppk, Psk), a primary auxiliary key Pak, and a secondary auxiliary key oak, and outputs a secondary verification/signing key pair (opk, osk) where $\gamma_s(oak) = opk$. Furthermore, for any PPT adversary $\mathcal{A}$ and all $\lambda \in \mathbb{N}$, we have*

$$|\Pr[par \leftarrow \mathsf{Setup}(1^\lambda) : \mathcal{A}^{\mathrm{SGenO}(\cdot)}(par) = 1] - \Pr[par \leftarrow \mathsf{Setup}(1^\lambda) : \mathcal{A}^{\mathrm{AKSGenO}(\cdot)}(par) = 1]| \leq negl(\lambda),$$

Here, on input a polynomial $n = n(\lambda)$, $\mathrm{SGenO}(\cdot)$ *runs $((Ppk, Psk), Pak) \leftarrow \mathsf{PGen}(par)$ and $((opk_i, osk_i), oak_i) \leftarrow \mathsf{SGen}(Ppk, Psk, Pak)$ for $i = 1, \ldots, n$, and returns $(Ppk, Psk, Pak, \{(opk_i, osk_i, oak_i)\}_{i=1}^n)$. On input a polynomial $n = n(\lambda)$,* $\mathrm{AKSGenO}(\cdot)$ *runs $((Ppk, Psk), Pak) \leftarrow \mathsf{PGen}(par)$, uniformly chooses oak_i from the secondary auxiliary key space $\mathcal{S}$, runs $(opk_i, osk_i) \leftarrow \mathsf{AKSGen}(Ppk, Psk, Pak, oak_i)$ for $i = 1, \ldots, n$, and returns $(Ppk, Psk, Pak, \{(opk_i, osk_i, oak_i)\}_{i=1}^n)$.*

[7] If a primary signing key consists of group elements, PGen may have trouble in outputting secondary auxiliary keys. However, this can be easily solved by forcing PGen to output the exponents of those group elements as part of a primary signing key.

Instantiation of (γ_p, γ_s)-SP-TT-AKS. Now we give an instantiation of (γ_p, γ_s)-SP-TT-AKS satisfying UF-TT-CMA security under the $\mathcal{U}_k$-MDDH assumptions. This signature scheme is the same as the SP two-tier signature scheme in [80] except that PGen and SGen additionally generate the auxiliary keys, and SGen additionally takes as input the primary auxiliary key. For this instantiation, the bijections (γ_p, γ_s) are defined by $\gamma_p(\mathbf{X}) = [\mathbf{X}]_2 \in \mathbb{G}_2^{n\times k} \times \mathbb{G}_2^{(k+1)\times k}$ and $\gamma_s(\vec{x}) = [\vec{x}]_2 \in \mathbb{G}_2^{1\times k}$ respectively for some fixed integer n which denotes the length of a message.

$\mathsf{Setup}(1^\lambda)$:
$par = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, [1]_1, [1]_2) \leftarrow \mathcal{G}(1^\lambda)$.
For preliminary-fixed $n \in \mathbb{N}$,
define $\mathcal{M} = \mathbb{Z}_p^n$, $\mathcal{M}_\gamma = \mathbb{G}_1^n$,
$\mathcal{P} = \mathbb{Z}_p^{n\times k} \times \mathbb{Z}_p^{(k+1)\times k}$, $\mathcal{P}_\gamma = \mathbb{G}_2^{n\times k} \times \mathbb{G}_2^{(k+1)\times k}$,
$\mathcal{S} = \mathbb{Z}_p^{1\times k}$, and $\mathcal{S}_\gamma = \mathbb{G}_2^{1\times k}$.
Define γ_p by $\gamma_p(\mathbf{X}) = [\mathbf{X}]_2 \in \mathbb{G}_2^{n\times k} \times \mathbb{G}_2^{(k+1)\times k}$
and γ_s by $\gamma_s(\vec{x}) = [\vec{x}]_2 \in \mathbb{G}_2^{1\times k}$.
Return par.

$\mathsf{PGen}(par)$:
$\mathbf{A} \leftarrow \mathcal{U}_k$, $\mathbf{K}' \leftarrow \mathbb{Z}_p^{n\times(k+1)}$, $\mathbf{C}' = \mathbf{K}'\mathbf{A} \in \mathbb{Z}_p^{n\times k}$.
$Ppk = ([\mathbf{C}']_2, [\mathbf{A}]_2)$, $Psk = \mathbf{K}'$, $Pak = (\mathbf{C}', \mathbf{A})$.
Return (Ppk, Psk) and Pak.

$\mathsf{SGen}(Ppk, Psk, Pak)$:
$\vec{k} \leftarrow \mathbb{Z}_p^{k+1}$, $\vec{c} = \vec{k}^\top \mathbf{A} \in \mathbb{Z}_p^{1\times k}$.
$opk = [\vec{c}]_2$, $osk = \vec{k}$, $oak = \vec{c}$.
Return (opk, osk) and oak.

$\mathsf{AKPGen}(par, Pak)$:
Parse $Pak = (\mathbf{C}', \mathbf{A})$.
Let $\mathbf{A} = \begin{pmatrix} \overline{\mathbf{A}} \\ \vec{a}^\top \end{pmatrix}$,
$\vec{k}' \leftarrow \mathbb{Z}_p^n$, $\underline{\mathbf{K}}' = (\mathbf{C}' - \vec{k}'\vec{a}^\top)\overline{\mathbf{A}}^{-1} \in \mathbb{Z}_p^{n\times k}$,
$\mathbf{K}' = (\underline{\mathbf{K}}', \vec{k}') \in \mathbb{Z}_p^{n\times(k+1)}$.
$Ppk = ([\mathbf{C}']_2, [\mathbf{A}]_2)$, $Psk = \mathbf{K}'$, $Pak = (\mathbf{C}', \mathbf{A})$.
Return (Ppk, Psk) and Pak.

$\mathsf{AKSGen}(Ppk, Psk, Pak, oak)$:
Parse $Ppk = ([\mathbf{C}']_2, [\mathbf{A}]_2))$, $Psk = \mathbf{K}'$, $Pak = (\mathbf{C}', \mathbf{A})$,
and $oak = \vec{c}$.
Let $\mathbf{A} = \begin{pmatrix} \overline{\mathbf{A}} \\ \vec{a}^\top \end{pmatrix}$, $k \leftarrow \mathbb{Z}_p$, $\vec{k}'^\top = (\vec{c} - k\vec{a}^\top)\overline{\mathbf{A}}^{-1}$, $\vec{k}^\top = (\vec{k}'^\top, k)$.
$opk = [\vec{c}]_2$, $osk = \vec{k}$, $oak = \vec{c}$.
Return (opk, osk) and oak.

$\mathsf{TTSign}(Psk, osk, [\vec{m}]_1)$:
$\mathbf{K} = (\vec{k}, \mathbf{K}'^\top)^\top$.
Return $\sigma = [(1, \vec{m}^\top)]_1\mathbf{K} \in \mathbb{G}_1^{1\times(k+1)}$.

$\mathsf{TTVerify}(Ppk, opk, [\vec{m}]_1, \sigma)$:
$[\mathbf{C}]_2 = ([\vec{c}]_2^\top, [\mathbf{C}']_2^\top)^\top$.
Return 1 if $e(\sigma, [\mathbf{A}]_2) = e([(1, \vec{m}^\top)]_1, [\mathbf{C}]_2)$.
Return 0 otherwise.

Figure 3.4: A UF-TT-CMA secure (γ_p, γ_s)-SP-TT-AKS scheme adapted from [80, Section 6.1].

Theorem 3.3.2. *The instantiation described in Fig. 3.4 satisfies the random primary and secondary auxiliary key properties.*

The proof follows from the fact that when the distributions of $\mathbf{C}'$ and c are uniform, the distribution of $\underline{\mathbf{K}'} = (\mathbf{C}' - \vec{k}\vec{a}^\top)\overline{\mathbf{A}}^{-1}$ and $\vec{k}' = (\vec{c} - k\vec{a}^\top)\overline{\mathbf{A}}^{-1}$ are uniform as well. The formal proof is as follows.

Proof of Theorem 3.3.2. Parsing the primary signing key $\mathbf{K}' \in \mathbb{Z}_p^{n\times(k+1)}$ as $(\underline{\mathbf{K}'}, \vec{k}') \in \mathbb{Z}_p^{n\times k} \times \mathbb{Z}_p^{n\times 1}$ and $\mathbf{A} \in \mathbb{Z}_p^{(k+1)\times k}$ as $\begin{pmatrix} \overline{\mathbf{A}} \\ \vec{a}^\top \end{pmatrix}$, we have that $\underline{\mathbf{K}'}\overline{\mathbf{A}} + \vec{k}'\vec{a}^\top = \mathbf{C}'$. If $\mathbf{C}'$ is randomly chosen from $\mathbb{Z}_p^{n\times k}$, we have that $\underline{\mathbf{K}'}\overline{\mathbf{A}} = (\mathbf{C}' - \vec{k}'\vec{a}^\top)$ looks random and independent of $\vec{k}'$ (if we do not see $\mathbf{C}'$ at first). Equivalently, we have that $\underline{\mathbf{K}'} = (\mathbf{C}' - \vec{k}'\vec{a}^\top)\overline{\mathbf{A}}^{-1}$ is indistinguishable

from a random matrix from $\mathbb{Z}_p^{n\times k}$. As a result, $\mathbf{K}'$ generated by $\mathsf{AKPGen}(par, Pak)$ is indistinguishable from a random matrix in $\mathbb{Z}_p^{n\times(k+1)}$ if Pak is randomly chosen from $\mathbb{Z}_p^{n\times k} \times \mathcal{U}_k$, which means that this TT-AKS scheme satisfies the random primary auxiliary key property.

Now we show that this signature scheme satisfies the random secondary auxiliary key property. Parse the secondary signing keys $\vec{k}_i \in \mathbb{Z}_p^{k+1}$ as $(\vec{k}_i'^{\top}, k_i) \in \mathbb{Z}_p^{1\times k} \times \mathbb{Z}_p$ and $\mathbf{A}$ as $\begin{pmatrix} \overline{\mathbf{A}} \\ \vec{a}^{\top} \end{pmatrix}$, we have that $\vec{k}_i'^{\top}\overline{\mathbf{A}} + k_i\vec{a}^{\top} = \vec{c}_i$ for $i = 1, \ldots, q$ where $q = q(\lambda)$ is a polynomial in λ. If $\vec{c}_i$ is randomly chosen from $\mathbb{Z}_p^{1\times k}$, we have that $\vec{k}_i'^{\top} = (\vec{c}_i - k_i\vec{a}^{\top})\overline{\mathbf{A}}^{-1}$ looks random and independent of k_i for all i (if we do not see $\vec{c}_i$ at first). Equivalently, we have that $\vec{k}_i'^{\top} = (\vec{c}_i - k_i\vec{a}^{\top})\overline{\mathbf{A}}^{-1}$ has the same distribution as a random vector in $\mathbb{Z}_p^{1\times k}$ for all i. As a result, all $\vec{k}_i$ generated by $\mathsf{AKSGen}(Ppk, Psk, Pak, oak_i)$ are indistinguishable from fresh randomness in $\mathbb{Z}_p^{k+1}$ if oak_i is randomly chosen from $\mathbb{Z}_p^{1\times k}$ for all i, which means that this signature scheme satisfies the random secondary auxiliary key property, completing the proof of Theorem 3.3.2. □

3.4 Generic Constructions of Fully Structure-Preserving Signature (and Fully Automorphic Signature)

In this section, we give generic constructions of FSPS and FAS from SKSP-TSs and (TT-)AKSs. Such constructions can be derived from SPSs that are based on various assumptions and with different efficiency performance. In Section 3.4.1, Section 3.4.2, and Section 3.4.3, we give three generic constructions of UF-CMA secure FSPS respectively. The first two constructions are based on SKSP-TSs and SP-AKSs, and the third one is based on SKSP-TSs and SP-TT-AKSs. In Section 3.4.4, we introduce a generic construction of UF-RMA secure FSPS based on the BTC scheme proposed in [10].

3.4.1 Generic Construction Sig_1: Trapdoor Signature + Signature with Auxiliary Key

We give a generic construction of FSPS (and FAS) based on a γ-SKSP-TS scheme and a γ'-SP-AKS scheme, where γ and γ' satisfy a suitable compatibility that we explain shortly.

Let $\Sigma_t = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{TDSign}, \mathsf{VerifySK})$ be a γ-SKSP-TS scheme with message spaces $\mathcal{M}$ and $\mathcal{M}_\gamma$, and $\Sigma_s = (\mathsf{Setup}, \mathsf{Gen}', \mathsf{Sign}', \mathsf{Verify}')$[8] a γ'-SP-AKS scheme with verification key space $\mathcal{M}_\gamma$, auxiliary key space $\mathcal{M}$, and message space $\mathcal{M}'$, and we have $\gamma'(x) = \gamma(x)$. Then a generic construction of FSPS denoted by $\mathsf{Sig}_1 = (\widehat{\mathsf{Setup}}, \widehat{\mathsf{Gen}}, \widehat{\mathsf{Sign}}, \widehat{\mathsf{Verify}}, \widehat{\mathsf{VerifySK}})$ with message space $\mathcal{M}'$ is described as in Fig. 3.5.

Next we give a theorem for this generic construction.

[8] As in [10], we assume that Σ_t and Σ_s share the common setup algorithm Setup.

$\widehat{\mathsf{Setup}}(1^\lambda)$: Run $par \leftarrow \mathsf{Setup}(1^\lambda)$. Determine the message spaces $\mathcal{M}$ and $\mathcal{M}_\gamma$ for Σ_t. Define $\gamma : \mathcal{M} \mapsto \mathcal{M}_\gamma$. Determine the message space $\mathcal{M}'$, verification key space $\mathcal{M}_\gamma$, and auxiliary key space $\mathcal{M}$ for Σ_s. Define $\gamma' : \mathcal{M} \mapsto \mathcal{M}_\gamma$ where $\gamma'(x) = \gamma(x)$. Return par.	$\widehat{\mathsf{Sign}}(sk, M)$: $((pk', sk'), ak') \leftarrow \mathsf{Gen}'(par)$. $\sigma_1 \leftarrow \mathsf{Sign}(sk, ak')$. $\sigma_2 = pk'$. $\sigma_3 \leftarrow \mathsf{Sign}'(sk', M)$. Return $\sigma = (\sigma_1, \sigma_2, \sigma_3)$.
$\widehat{\mathsf{Gen}}(par)$: $((pk, sk), tk) \leftarrow \mathsf{Gen}(par)$. Return (pk, sk).	$\widehat{\mathsf{Verify}}(pk, M, \sigma)$: Parse $\sigma = (\sigma_1, \sigma_2, \sigma_3)$ and $\sigma_2 = pk'$. Return 1 if $\mathsf{Verify}(pk, \sigma_2, \sigma_1) = 1$ and $\mathsf{Verify}'(pk', M, \sigma_3) = 1$. Return 0 otherwise.
$\widehat{\mathsf{VerifySK}}(pk, sk)$: Return 1 if $\mathsf{VerifySK}(pk, sk) = 1$. Return 0 otherwise.	

Figure 3.5: Generic construction Sig_1: TS + AKS (UF-otCMA).

Theorem 3.4.1. *If Σ_t is a UF-CMA secure SKSP-TS scheme, and Σ_s a UF-otCMA secure SP-AKS scheme, then $\mathsf{Sig}_1 = (\widehat{\mathsf{Setup}}, \widehat{\mathsf{Gen}}, \widehat{\mathsf{Sign}}, \widehat{\mathsf{Verify}}, \widehat{\mathsf{VerifySK}})$ is a UF-CMA secure FSPS scheme.*

Overview of the proof. The proof of Theorem 3.4.1 follows from the fact that if there exists a PPT adversary $\mathcal{A}$ that outputs a successful forgery $(\sigma_1^*, \sigma_2^*, \sigma_3^*)$, where σ_2^* was not queried before (respectively, was queried before), with non-negligible probability, then we can construct a PPT adversary $\mathcal{B}_1$ (respectively, $\mathcal{B}_2$) that breaks the UF-CMA security of Σ_t (respectively, the UF-otCMA security of Σ_s). Note that to answer a query from $\mathcal{A}$, $\mathcal{B}_2$ may have to use the signing key of Σ_t to sign an auxiliary key ak' of Σ_s, while it only learns the corresponding verification key pk' from the challenger. In this case, it signs pk' by using the trapdoor key of Σ_t instead. According to the correctness of a TS scheme, $\mathcal{A}$ cannot distinguish such a signature with an honestly generated one, which means that $\mathcal{B}_2$ can perfectly simulate the signing oracle of $\mathcal{A}$. The formal proof is as follows.

Proof of Theorem 3.4.1. Proving that Sig_1 is FSPS is straightforward. Since Σ_t is SKSP, we have that verification keys and signing keys consist only of group elements in $\mathbb{G}_1$ and $\mathbb{G}_2$, and $\mathsf{VerifySK}$ and Verify consist only of PPEs. Furthermore, since Σ_s is SP, we have that messages and signatures consist only of group elements in $\mathbb{G}_1$ and $\mathbb{G}_2$, and Verify' consists only of PPEs. What is left to do is to show that $\mathsf{VerifySK}$ is able to verify the correctness of signing keys w.r.t. verification keys.

According to the SKSP property of Σ_t, for all sufficiently large $\lambda \in \mathbb{N}$, $par \leftarrow \mathsf{Setup}(1^\lambda)$, and verification/signing keys pair (pk, sk), if $\mathsf{VerifySK}(pk, sk) = 1$ holds, then $\mathsf{Verify}(pk, \gamma(ak'), \mathsf{Sign}(sk, ak')) = 1$ holds for all $ak' \in \mathcal{M}$ i.e., for all $(pk', sk', ak') \leftarrow \mathsf{Gen}'(par)$ (and all possible randomness used by Sign). Furthermore, $\mathsf{Verify}'(pk', M, \sigma_3) = 1$

holds for all $(pk', sk', ak') \leftarrow \mathsf{Gen}'(par)$, $M \in \mathcal{M}'$, and $\sigma_3 \leftarrow \mathsf{Sign}'(sk', M)$ according to the correctness of Σ_s. As a result, for all sufficiently large $\lambda \in \mathbb{N}$, $par \leftarrow \mathsf{Setup}(1^\lambda)$, and (pk, sk), if $\mathsf{VerifySK}(pk, sk) = 1$, then $\widehat{\mathsf{Verify}}(pk, M, \widehat{\mathsf{Sign}}(sk, M)) = 1$ holds for all $M \in \mathcal{M}'$ (and all possible randomness used by $\widehat{\mathsf{Sign}}$). One the other hand, if for all sufficiently large $\lambda \in \mathbb{N}$, $par \leftarrow \mathsf{Setup}(1^\lambda)$, and verification/signing keys pair (pk, sk), we have $\widehat{\mathsf{Verify}}(pk, M, \widehat{\mathsf{Sign}}(sk, M)) = 1$ for all $M \in \mathcal{M}'$ (and all possible randomness used by $\widehat{\mathsf{Sign}}$), then $\mathsf{Verify}(pk, \gamma(ak'), \mathsf{Sign}(sk, ak')) = 1$ holds for all $(pk', sk', ak') \leftarrow \mathsf{Gen}'(par)$ (and all possible randomness used by Sign). As a result, $\mathsf{Verify}(pk, \gamma(ak'), \mathsf{Sign}(sk, ak')) = 1$ holds for all $ak' \in \mathcal{M}$ (and all possible randomness),[9] which means that we have $\mathsf{VerifySK}(pk, sk) = 1$, according to the SKSP property of Σ_t, completing the proof that Sig_1 is FSPS.

Next we prove that this signature scheme satisfies UF-CMA security. Let $\mathcal{A}$ be any PPT adversary. For $j \in \{1, \ldots, q\}$ where q denotes the number of the queries made by $\mathcal{A}$, let $M^{(j)}$ denote the jth signing query and $\sigma^{(j)} = (\sigma_1^{(j)}, \sigma_2^{(j)}, \sigma_3^{(j)})$ the answer to the jth signing query. At some point, $\mathcal{A}$ outputs $(M^*, (\sigma_1^*, \sigma_2^*, \sigma_3^*))$ as the forgery.

To win the UF-CMA security game, $\mathcal{A}$ has to output a forgery, which is one of the following two types.

- **type I** forgery: $M^* \notin \{M^{(1)}, \ldots, M^{(q)}\}$, $\sigma_2^* \notin (\sigma_2^{(1)}, \ldots, \sigma_2^{(q)})$, $\mathsf{Verify}(pk, \sigma_2^*, \sigma_1^*) = 1$, and $\mathsf{Verify}'(\sigma_2^*, M^*, \sigma_3^*) = 1$.
- **type II** forgery: $M^* \notin \{M^{(1)}, \ldots, M^{(q)}\}$, $\sigma_2^* \in \{\sigma_2^{(1)}, \ldots, \sigma_2^{(q)}\}$, $\mathsf{Verify}(pk, \sigma_2^*, \sigma_1^*) = 1$, and $\mathsf{Verify}'(\sigma_2^*, M^*, \sigma_3^*) = 1$.

type I. We show that $\mathcal{A}$ outputs a **type I** forgery with negligible probability.

Lemma 3.4.1. *If $\mathcal{A}$ outputs a* **type I** *forgery with probability ϵ, then we can construct a PPT adversary $\mathcal{B}$ that breaks the UF-CMA security of Σ_t with advantage ϵ.*

Proof of Lemma 3.4.1. The challenger samples $par \leftarrow \mathsf{Setup}(1^\lambda)$ and $((pk, sk), tk) \leftarrow \mathsf{Gen}(par)$, and gives (par, pk) to $\mathcal{B}$. Then $\mathcal{B}$ gives (par, pk) to $\mathcal{A}$.

Signing queries. For all $j \in \{1, \ldots, q\}$, when $\mathcal{A}$ makes the jth query to $\mathcal{B}$, $\mathcal{B}$ responds as follows:

1. Sample $((pk'^{(j)}, sk'^{(j)}), ak'^{(j)}) \leftarrow \mathsf{Gen}'(par)$ where $pk'^{(j)} \in \mathcal{M}_\gamma$ and $ak'^{(j)} \in \mathcal{M}$.
2. Make a signing query $ak'^{(j)}$ to the challenger who returns $\sigma_1^{(j)} \leftarrow \mathsf{Sign}(sk, ak'^{(j)})$.
3. Let $\sigma_2^{(j)} = pk'^{(j)}$.
4. Compute $\sigma_3^{(j)} \leftarrow \mathsf{Sign}'(sk'^{(j)}, M^{(j)})$.
5. Return $\sigma^{(j)} = (\sigma_1^{(j)}, \sigma_2^{(j)}, \sigma_3^{(j)})$ to $\mathcal{A}$.

[9] This is because that the auxiliary key space of Σ_s perfectly matches the message space of Σ_t.

Output. When $\mathcal{A}$ outputs $(M^*, \sigma_1^*, \sigma_2^*, \sigma_3^*)$, $\mathcal{B}$ outputs (σ_2^*, σ_1^*).

Since the view of $\mathcal{A}$ is identical to its view in the original UF-CMA game of Sig_1, the probability that $\sigma_2^* \notin (\sigma_2^{(1)}, \ldots, \sigma_2^{(q)})$ and $\mathsf{Verify}(pk, \sigma_2^*, \sigma_1^*) = 1$ is ϵ. As a result, $\mathcal{B}$ breaks the UF-CMA security of Σ_t with advantage ϵ, completing the proof of Lemma 3.4.1. □

Since the UF-CMA security of Σ_t implies that ϵ is negligible, the probability that $\mathcal{A}$ outputs a **type I** forgery is negligible, completing this part of the proof.

type II. We give hybrid games to show that $\mathcal{A}$ outputs a **type II** forgery with negligible probability.

Game 0: This is the original UF-CMA security game for $\mathcal{A}$. At the beginning of the game, the challenger samples $par \leftarrow \mathsf{Setup}(1^\lambda)$ and $((pk, sk), tk) \leftarrow \mathsf{Gen}(par)$, and gives (par, pk) to $\mathcal{A}$.

Signing queries. For all $j \in \{1, \ldots, q\}$, when $\mathcal{A}$ sends the jth query to the challenger, the challenger responds as follows:

1. Sample $((pk'^{(j)}, sk'^{(j)}), ak'^{(j)}) \leftarrow \mathsf{Gen}'(par)$ where $pk'^{(j)} \in \mathcal{M}_\gamma$ and $ak'^{(j)} \in \mathcal{M}$.
2. Compute $\sigma_1^{(j)} \leftarrow \mathsf{Sign}(sk, ak'^{(j)})$ and sends $\sigma_1^{(j)}$ back to $\mathcal{A}$.
3. Let $\sigma_2^{(j)} = pk'^{(j)}$.
4. Compute $\sigma_3^{(j)} \leftarrow \mathsf{Sign}'(sk'^{(j)}, M^{(j)})$.
5. Return $\sigma^{(j)} = (\sigma_1^{(j)}, \sigma_2^{(j)}, \sigma_3^{(j)})$ to $\mathcal{A}$.

Output. At some point, $\mathcal{A}$ outputs $(M^*, \sigma^* = (\sigma_1^*, \sigma_2^*, \sigma_3^*))$. $\mathcal{A}$ succeeds if (M^*, σ^*) is a **type II** forgery.

Game 1: This game is the same as **Game 0** except that the challenger randomly chooses $\hat{j} \leftarrow \{1, \ldots, q\}$ at the beginning of the game, and $\mathcal{A}$ succeeds if its forgery is a **type II** forgery and $\sigma_2^* = \sigma_2^{(\hat{j})}$.

Lemma 3.4.2. *If $\mathcal{A}$ succeeds with probability ϵ_0 in* **Game 0** *and ϵ_1 in* **Game 1**, *then we have $\epsilon_1 \geq \epsilon_0/q$.*

Proof of Lemma 3.4.2. Since we have $\sigma_2^* \in \{\sigma_2^{(1)}, \ldots, \sigma_2^{(q)}\}$ if the forgery is a **type II** forgery, j such that $\sigma_2^* = \sigma_2^{(j)}$ must exists. Then this lemma follows from the fact that the view of $\mathcal{A}$ is identical to its view in **Game 0** and $\mathcal{A}$ learns no information on which $\hat{j}$ is chosen. □

Game 2: This game is the same as **Game 1** except that the challenger generates $\sigma_1^{(\hat{j})}$ by computing $\sigma_1^{(\hat{j})} \leftarrow \mathsf{TDSign}(tk, pk'^{(\hat{j})})$. For all $j \neq \hat{j}$, $\sigma_1^{(j)}$ is honestly generated.

Lemma 3.4.3. *If $\mathcal{A}$ succeeds with probability ϵ_1 in* **Game 1** *and ϵ_2 in* **Game 2**, *then we have $\epsilon_1 = \epsilon_2$.*

Proof of Lemma 3.4.3. This lemma follows from the fact that $\mathsf{TDSign}(tk, pk'^{(\hat{j})}; r) = \mathsf{Sign}(sk, ak'^{(\hat{j})}; r)$ for all r in the randomness space according to the correctness of a γ-TS scheme. □

Lemma 3.4.4. *If $\mathcal{A}$ succeeds with probability ϵ_2 in* **Game 2**, *then we can construct a PPT adversary $\mathcal{B}$ that breaks the UF-otCMA security of Σ_s with advantage at least ϵ_2.*

Proof of Lemma 3.4.4. $\mathcal{B}$ takes par and pk' from the UF-otCMA challenger who samples $par \leftarrow \mathsf{Setup}(1^\lambda)$ and $((pk', sk'), ak') \leftarrow \mathsf{Gen}'(par)$. Then $\mathcal{B}$ samples $((pk, sk), tk) \leftarrow \mathsf{Gen}(par)$ and gives (par, pk) to $\mathcal{A}$. $\mathcal{B}$ also picks $\hat{j} \leftarrow \{1, \ldots, q\}$ uniformly, and answers the $\hat{j}$th query $M^{(\hat{j})}$ from $\mathcal{A}$ as follows:

1. Compute $\sigma_1^{(\hat{j})} \leftarrow \mathsf{TDSign}(tk, pk')$.
2. Let $\sigma_2^{(\hat{j})} = pk'$.
3. Make a signing query $M^{(\hat{j})}$ to the challenger who returns $\sigma_3^{(\hat{j})} \leftarrow \mathsf{Sign}'(sk', M^{(\hat{j})})$.
4. Return $\sigma^{(\hat{j})} = (\sigma_1^{(\hat{j})}, \sigma_2^{(\hat{j})}, \sigma_3^{(\hat{j})})$ to $\mathcal{A}$.

Furthermore, $\mathcal{B}$ answers the jth signing query $M^{(j)}$ where $j \neq \hat{j}$ by honestly computing $\sigma^{(j)} \leftarrow \widehat{\mathsf{Sign}}(sk, M^{(j)})$ and returning $\sigma^{(j)}$ to $\mathcal{A}$.

When $\mathcal{A}$ outputs $(M^*, \sigma_1^*, \sigma_2^*, \sigma_3^*)$, $\mathcal{B}$ outputs (M^*, σ_3^*).

Since the view of $\mathcal{A}$ is identical to its view in **Game 2**, the probability that $\mathcal{A}$ succeeds is ϵ_2, i.e., the probability that $M^* \neq M^{(\hat{j})}$ and $\mathsf{Verify}'(pk', M^*, \sigma_3^*) = 1$ is ϵ_2, completing the proof of Lemma 3.4.4. □

Let ϵ_i be the probability that $\mathcal{A}$ succeeds in **Game i**. The UF-otCMA security of Σ_s implies that ϵ_2 is negligible. Furthermore, since $\epsilon_2 = \epsilon_1$ and $\epsilon_1 \geq \epsilon_0/q$, we have that ϵ_0, which is the probability that $\mathcal{A}$ outputs a **type II** forgery, is negligible. This completes this part of the proof.

In conclusion, $\mathcal{A}$ breaks the UF-CMA security of Sig_1 with only negligible advantage, completing the proof of Theorem 3.4.1. □

UF-RMA secure TSs + UF-otCMA secure AKSs. Now we give another theorem showing that for the generic construction in Fig. 3.5, the security of the TS scheme can be weakened to UF-RMA security if the AKS scheme satisfies the random auxiliary key property.

Theorem 3.4.2. *If Σ_t is a UF-RMA secure SKSP-TS scheme, and Σ_s a UF-otCMA secure SP-AKS scheme satisfying the random auxiliary key property, then* $\mathsf{Sig}_1 = (\widehat{\mathsf{Setup}}, \widehat{\mathsf{Gen}}, \widehat{\mathsf{Sign}}, \widehat{\mathsf{Verify}}, \widehat{\mathsf{VerifySK}})$ *is a UF-CMA secure FSPS scheme.*

Overview of the proof. The proof overview of Theorem 3.4.2 is the same as that of Theorem 3.4.1 except that $\mathcal{B}_1$ is against the UF-RMA security of Σ_t instead of UF-CMA security. To answer a query from $\mathcal{A}$, $\mathcal{B}_1$ makes a query to the signing oracle of Σ_t to obtain a randomly chosen auxiliary key ak' and the corresponding signature σ_1. Then $\mathcal{B}_1$ runs the additional algorithm AKGen (defined in Definition 3.3.3) on input (par, ak') to generate a verification/signing key pair (pk', sk'), which is indistinguishable from an honestly generated one according to the random auxiliary key property. Then it lets pk' be σ_2 and use sk' to sign the message. The formal proof is as follows.

Proof of Theorem 3.4.2. This proof is the same as the proof of Theorem 3.4.1 except that we show that $\mathcal{A}$ outputs a **type I** forgery with negligible probability by constructing a PPT adversary $\mathcal{B}$ against the UF-RMA security instead of the UF-CMA security of Σ_t in a different way as follows.

type I. We give hybrid games to show that $\mathcal{A}$ outputs a **type I** forgery with negligible probability.

Game 0: This game is the same as the original UF-CMA security game for $\mathcal{A}$. The challenger samples $par \leftarrow \mathsf{Setup}(1^\lambda)$, $((pk, sk), tk) \leftarrow \mathsf{Gen}(par)$, and gives (par, pk) to $\mathcal{A}$.

Signing queries. For all $j \in \{1, \ldots, q\}$, when $\mathcal{A}$ sends the jth query $M^{(j)}$ to the challenger, the challenger responds as follows:

1. Sample $((pk'^{(j)}, sk'^{(j)}), ak'^{(j)}) \leftarrow \mathsf{Gen}'(par)$ where $pk'^{(j)} \in \mathcal{M}_\gamma$ and $ak'^{(j)} \in \mathcal{M}$.
2. Compute $\sigma_1^{(j)} \leftarrow \mathsf{Sign}(sk, ak'^{(j)})$.
3. Let $\sigma_2^{(j)} = pk'^{(j)}$.
4. Compute $\sigma_3^{(j)} \leftarrow \mathsf{Sign}'(sk'^{(j)}, M^{(j)})$.
5. Return $\sigma^{(j)} = (\sigma_1^{(j)}, \sigma_2^{(j)}, \sigma_3^{(j)})$.

Output. At some point, $\mathcal{A}$ outputs $(M^*, \sigma^* = (\sigma_1^*, \sigma_2^*, \sigma_3^*))$. $\mathcal{A}$ succeeds if (M^*, σ^*) is a **type I** forgery.

Game 1: This game is the same as **Game 0** except that to answer the jth query from $\mathcal{A}$ for all j, the challenger uniformly samples auxiliary keys at first and then generates verification/signing key pairs by running $\mathsf{AKGen}'(par, ak'^{(j)})$. Concretely speaking, for all $j \in \{1, \ldots, q\}$, the jth signing query is answered by the challenger as follows:

1. Sample $ak'^{(j)} \leftarrow \mathcal{M}$ and compute $(pk'^{(j)}, sk'^{(j)}) \leftarrow \mathsf{AKGen}'(par, ak'^{(j)})$ where $pk'^{(j)} \in \mathcal{M}_\gamma$.
2. Compute $\sigma_1^{(j)} \leftarrow \mathsf{Sign}(sk, ak'^{(j)})$.
3. Let $\sigma_2^{(j)} = pk'^{(j)}$.
4. Compute $\sigma_3^{(j)} \leftarrow \mathsf{Sign}'(sk'^{(j)}, M'^{(j)})$.
5. Return $\sigma^{(j)} = (\sigma_1^{(j)}, \sigma_2^{(j)}, \sigma_3^{(j)})$ to $\mathcal{A}$.

Lemma 3.4.5. *If $\mathcal{A}$ succeeds with probability ϵ_0 in* **Game 0** *and ϵ_1 in* **Game 1**, *then we can construct a PPT adversary $\mathcal{B}$ that breaks the random auxiliary key property of Σ_s with advantage $|\epsilon_1 - \epsilon_0|$.*

Proof of Lemma 3.4.5. Taking as input par, $\mathcal{B}$ samples $((pk, sk), tk) \leftarrow \mathsf{Gen}(par)$ and gives (par, pk) to $\mathcal{A}$. For all $j \in \{1, \ldots, q\}$, when $\mathcal{A}$ sends the jth signing query $M^{(j)}$ to $\mathcal{B}$, $\mathcal{B}$ answers the query as follows:

1. Make a query to the oracle which is GenO or AKGenO as described in Definition 3.3.3. The oracle returns the results $(pk'^{(j)}, sk'^{(j)}, ak'^{(j)})$, where $pk'^{(j)} \in \mathcal{M}_\gamma$ and $ak'^{(j)} \in \mathcal{M}$.
2. Compute $\sigma_1^{(j)} \leftarrow \mathsf{Sign}(sk, ak'^{(j)})$.
3. Let $\sigma_2^{(j)} = pk'^{(j)}$.
4. Compute $\sigma_3^{(j)} \leftarrow \mathsf{Sign}'(sk'^{(j)}, M^{(j)})$.
5. Return $\sigma^{(j)} = (\sigma_1^{(j)}, \sigma_2^{(j)}, \sigma_3^{(j)})$.

At some point, $\mathcal{A}$ outputs the forgery (M^*, σ^*). If the forgery is a **type I** forgery, $\mathcal{B}$ outputs 1. Otherwise, $\mathcal{B}$ outputs 0.

Since the view of $\mathcal{A}$ is identical to its view in **Game 0** if the oracle is GenO or in **Game 1** if the oracle is AKGenO, we have that $\mathcal{B}$ breaks the random auxiliary key property with advantage $|\epsilon_1 - \epsilon_0|$, completing the proof of Lemma 3.4.5. □

Lemma 3.4.6. *If $\mathcal{A}$ succeeds with probability ϵ_1 in* **Game 1**, *then we can construct a PPT adversary $\mathcal{B}$ that breaks the UF-RMA security of Σ_t with advantage at least ϵ_1.*

Proof of Lemma 3.4.6. $\mathcal{B}$ takes par, pk, $(ak'^{(1)}, \ldots, ak'^{(q)})$, and $(\sigma_1^{(1)}, \ldots, \sigma_1^{(q)})$ from the UF-RMA challenger who samples $((pk, sk), tk) \leftarrow \mathsf{Gen}(par)$, randomly chooses $ak'^{(j)} \leftarrow \mathcal{M}$, and computes $\sigma_1^{(j)} \leftarrow \mathsf{Sign}(sk, ak'^{(j)})$, for all $j \in \{1, \ldots, q\}$. Then $\mathcal{B}$ gives (par, pk) to $\mathcal{A}$. When $\mathcal{A}$ makes the jth signing query, $\mathcal{B}$ answers as follows:

1. Compute $(pk'^{(j)}, sk'^{(j)}) \leftarrow \mathsf{AKGen}'(par, ak'^{(j)})$ where $pk'^{(j)} \in \mathcal{M}_\gamma$.
2. Let $\sigma_2^{(j)} = pk'^{(j)}$.
3. Compute $\sigma_3^{(j)} \leftarrow \mathsf{Sign}'(sk'^{(j)}, M^{(j)})$.
4. Return $\sigma^{(j)} = (\sigma_1^{(j)}, \sigma_2^{(j)}, \sigma_3^{(j)})$ to $\mathcal{A}$.

When $\mathcal{A}$ outputs $(M^*, \sigma_1^*, \sigma_2^*, \sigma_3^*)$, $\mathcal{B}$ outputs (σ_2^*, σ_1^*) as the forgery.

Since the view of $\mathcal{A}$ is identical to its view in **Game 1**, the probability that $\mathcal{A}$ succeeds is ϵ_1. i.e., that probability that $\mathsf{Verify}(pk, \sigma_2^*, \sigma_1^*) = 1$ and $\sigma_2^* \notin (\sigma_2^{(1)}, \ldots, \sigma_2^{(q)})$ is at least ϵ_1, completing the proof of Lemma 3.4.6. □

Let ϵ_i be the probability that $\mathcal{A}$ succeeds in **Game i**. The UF-RMA security of Σ_t and the random auxiliary key property of Σ_s imply that ϵ_1 and $|\epsilon_1 - \epsilon_0|$ are negligible respectively, i.e., ϵ_0, which is the probability that $\mathcal{A}$ outputs a **type I** forgery, is negligible, completing this part of the proof.

Since the other parts of this proof follow from the corresponding parts of the proof of Theorem 3.4.1, $\mathcal{B}$ breaks the UF-RMA security of Σ_t with negligible advantage, completing the proof of Theorem 3.4.2. □

Remark. In our construction, we require the message space of Σ_t to be the same as the auxiliary key space of Σ_s, or the FSP property is not strictly satisfied. If we relax the definition of FSP property (as described below Definition 3.1.3), we only need Σ_t to be able to sign auxiliary keys of Σ_s (i.e., the auxiliary key space of Σ_s can be smaller than the message space of Σ_t).[10]

Instantiations of Sig_1. By combining the UF-CMA (respectively, UF-otRMA) secure TS scheme in Fig. 3.1 (respectively, Fig. 3.2) with the UF-otCMA secure AKS scheme in Fig. 3.3 (where $\mathbb{G}_1$ and $\mathbb{G}_2$ are swapped), we obtain an FSPS scheme satisfying UF-CMA (respectively, UF-otCMA) security. We refer the reader to Fig. 3.9 and Fig. 3.10 (in Section 3.7) for the resulting signature schemes.

Furthermore, by converting other previously proposed SPSs into SKSP-TSs and SP-AKSs, we obtain various FSPSs. We list some of them in Table 3.1 in Section 3.5.

3.4.2 Variation of Sig_1: Trapdoor Signature + Signature with Auxiliary Key (UF-CMA)

Now we give a variation of the generic construction in Fig. 3.6 by letting Σ_s be a UF-CMA secure SP-AKS scheme and sign n message blocks with one signing key. Each block is signed with an element indicating its number. This change reduces the signature and verification key sizes from $\Omega(n^2)$ to $\Omega(n)$ when signing n^2 group elements.

[10] This argument also holds for our other generic constructions.

Let $\Sigma_t = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{TDSign}, \mathsf{VerifySK})$ be a γ-SKSP-TS scheme with message spaces $\mathcal{M}$ and $\mathcal{M}_\gamma$, and $\Sigma_s = (\mathsf{Setup}, \mathsf{Gen}', \mathsf{Sign}', \mathsf{Verify}')$[11] a γ-SP-AKS scheme with verification key space $\mathcal{M}_\gamma$, auxiliary key space $\mathcal{M}$, and message space $\mathcal{M}' \times \mathcal{M}_I$, where $\mathcal{M}_I$ is the space for elements indicating the numbers of blocks. Then a generic construction of FSPS denoted by $\mathsf{Sig}_1^* = (\widehat{\mathsf{Setup}}, \widehat{\mathsf{Gen}}, \widehat{\mathsf{Sign}}, \widehat{\mathsf{Verify}}, \widehat{\mathsf{VerifySK}})$ with message space $\mathcal{M}'^n$, where n is some fixed integer, is described as in Fig. 3.6.

$\widehat{\mathsf{Setup}}(1^\lambda)$: Run $par \leftarrow \mathsf{Setup}(1^\lambda)$. Determine the message spaces $\mathcal{M}$ and $\mathcal{M}_\gamma$ for Σ_t. Determine the message space $\mathcal{M}' \times \mathcal{M}_I$, verification key space $\mathcal{M}_\gamma$, and auxiliary key space $\mathcal{M}$ for Σ_s. Define $\gamma : \mathcal{M} \mapsto \mathcal{M}_\gamma$. Return par.	$\widehat{\mathsf{Sign}}(sk, \vec{M})$: Parse $\vec{M} = (M_1, \ldots, M_n) \in \mathcal{M}'^n$. $((pk', sk'), ak') \leftarrow \mathsf{Gen}'(par)$. $\sigma_1 \leftarrow \mathsf{Sign}(sk, ak')$. $\sigma_2 = pk'$. $\sigma_{3i} \leftarrow \mathsf{Sign}'(sk', (M_i, I(i)))$ where $I(i) \in \mathcal{M}_I$ for $i = 1, \ldots, n$. $\sigma_3 = (\sigma_{31}, \ldots, \sigma_{3n})$. Return $\sigma = (\sigma_1, \sigma_2, \sigma_3)$.
$\widehat{\mathsf{Gen}}(par)$: $((pk, sk), tk) \leftarrow \mathsf{Gen}(par)$. Return (pk, sk).	$\widehat{\mathsf{Verify}}(pk, \vec{M}, \sigma)$: Parse $\vec{M} = (M_1, \ldots, M_n) \in \mathcal{M}'^n$, $\sigma = (\sigma_1, \sigma_2, \sigma_3)$, and $\sigma_3 = (\sigma_{31}, \ldots, \sigma_{3n})$. Return 1 if $\mathsf{Verify}(pk, \sigma_2, \sigma_1) = 1$ and $\mathsf{Verify}'(pk', (M_i, I(i)), \sigma_{3i}) = 1$ for all i. Return 0 otherwise.
$\widehat{\mathsf{VerifySK}}(pk, sk)$: Return 1 if $\mathsf{VerifySK}(pk, sk) = 1$. Return 0 otherwise.	

Figure 3.6: Generic construction Sig_1^*: TS + AKS (UF-CMA).

For this generic construction, the following two theorems hold.

Theorem 3.4.3. *If Σ_t is a UF-CMA secure SKSP-TS scheme, and Σ_s a UF-CMA secure SP-AKS scheme, then $\mathsf{Sig}_1^* = (\widehat{\mathsf{Setup}}, \widehat{\mathsf{Gen}}, \widehat{\mathsf{Sign}}, \widehat{\mathsf{Verify}}, \widehat{\mathsf{VerifySK}})$ is a UF-CMA secure FSPS scheme.*

Theorem 3.4.4. *If Σ_t is a UF-RMA secure SKSP-TS scheme, and Σ_s a UF-CMA secure SP-AKS scheme satisfying the random auxiliary key property, then $\mathsf{Sig}_1^* = (\widehat{\mathsf{Setup}}, \widehat{\mathsf{Gen}}, \widehat{\mathsf{Sign}}, \widehat{\mathsf{Verify}}, \widehat{\mathsf{VerifySK}})$ is a UF-CMA secure FSPS scheme.*

We omit the proofs of Theorem 3.4.3 and Theorem 3.4.4 since they are similar to the proofs of Theorem 3.4.1 and Theorem 3.4.2, respectively. We list several instantiations of Sig^* in Table 3.1 in Section 3.5. Most of them achieve better efficiency than instantiations obtained from Sig_1, and are automorphic.

3.4.3 Generic Construction Sig_2: Trapdoor Signature + Two-tier Signature with Auxiliary Keys

In this section, we give another generic construction of FSPS which provides us with FSPSs and FASs based on standard assumptions that have shorter verification keys and signatures.

[11] As in [10], we assume that Σ_t and Σ_s share the common setup algorithm Setup.

Let $\Sigma_t = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{TDSign}, \mathsf{VerifySK})$ be a γ-TS scheme with message spaces $\mathcal{M}_p \times \mathcal{M}_s^n$ and $\mathcal{M}_{\gamma p} \times \mathcal{M}_{\gamma s}^n$, $\Sigma_s = (\mathsf{Setup}, \mathsf{PGen}, \mathsf{SGen}, \mathsf{TTSign}, \mathsf{TTVerify})$[12] a (γ_p, γ_s)-TT-AKS with primary/secondary verification key spaces $\mathcal{M}_{\gamma p}/\mathcal{M}_{\gamma s}$, auxiliary key spaces $\mathcal{M}_p/\mathcal{M}_s$, and message space $\mathcal{M}'$, where n is some fixed integer and $(\gamma_p(x_0), \gamma_s(x_1), \ldots, \gamma_s(x_n)) = \gamma(x_0, x_1 \ldots, x_n)$. A generic construction of FSPS denoted by $\mathsf{Sig}_2 = (\widehat{\mathsf{Setup}}, \widehat{\mathsf{Gen}}, \widehat{\mathsf{Sign}}, \widehat{\mathsf{Verify}}, \widehat{\mathsf{VerifySK}})$ with message space $\mathcal{M}'^n$ is as described as in Fig. 3.7.

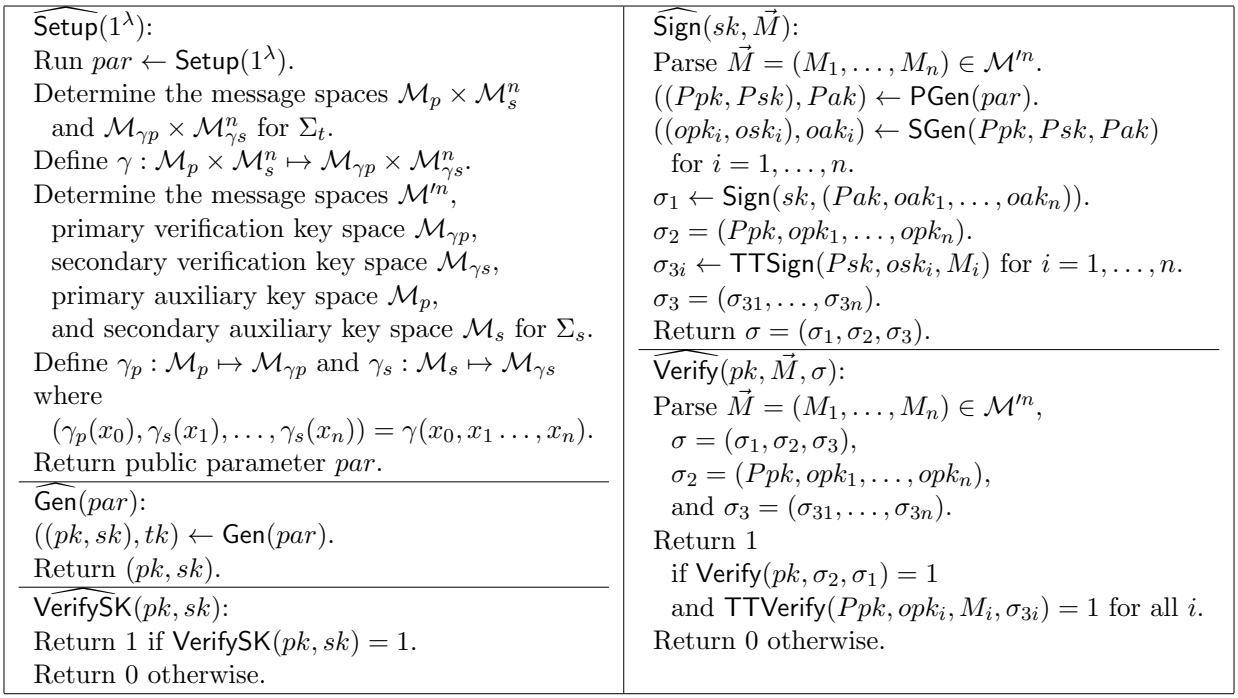

Figure 3.7: Generic construction Sig_2: TS + TT-AKS.

For this generic construction, the following theorem holds.

Theorem 3.4.5. *If Σ_t is a UF-CMA secure SKSP-TS scheme, and Σ_s a UF-TT-CMA secure SP-TT-AKS scheme, then $\mathsf{Sig}_2 = (\widehat{\mathsf{Setup}}, \widehat{\mathsf{Gen}}, \widehat{\mathsf{Sign}}, \widehat{\mathsf{Verify}}, \widehat{\mathsf{VerifySK}})$ is a UF-CMA secure FSPS scheme.*

Proof of Theorem 3.4.5. Since the proof that $\mathsf{Sig}_2 = (\widehat{\mathsf{Setup}}, \widehat{\mathsf{Gen}}, \widehat{\mathsf{Sign}}, \widehat{\mathsf{Verify}}, \widehat{\mathsf{VerifySK}})$ is FSP is straightforward and similar to the corresponding part of the proof of Theorem 3.4.1, we omit it here.

Now we prove that Sig_2 satisfies UF-CMA security. Let $\mathcal{A}$ be any PPT adversary. For $i = 1, \ldots, q$ where q denotes the number of the queries made by the adversary, let $\vec{M}^{(j)} = (M_1^{(j)}, \ldots, M_n^{(j)})$ be the jth signing query, $\sigma^{(j)} = (\sigma_1^{(j)}, \sigma_2^{(j)}, \sigma_3^{(j)})$ (where

[12] As in [10], we assume that Σ_t and Σ_s share the common setup algorithm Setup.

$\sigma_2^{(j)} = (Ppk^{(j)}, opk_1^{(j)}, \ldots, opk_n^{(j)})$ and $\sigma_3^{(j)} = (\sigma_{31}^{(j)}, \ldots, \sigma_{3n}^{(j)})$) the answer to the jth signing query. At some point, $\mathcal{A}$ outputs $(\vec{M}^*, (\sigma_1^*, \sigma_2^*, \sigma_3^*))$ (where $\vec{M}^* = (M_1^*, \ldots, M_n^*)$, $\sigma_2^* = (Ppk^*, opk_1^*, \ldots, opk_n^*)$, and $\sigma_3^* = (\sigma_{31}^*, \ldots, \sigma_{3n}^*)$) as the forgery.

To win the UF-CMA game with non-negligible probability ϵ, $\mathcal{A}$ has to output a forgery, which is one of the following two types.

- **type I** forgery: $\vec{M}^* \notin \{\vec{M}^{(1)}, \ldots, \vec{M}^{(q)}\}$, $\sigma_2^* \notin (\sigma_2^{(1)}, \ldots, \sigma_2^{(q)})$, $\mathsf{Verify}(pk, \sigma_2^*, \sigma_1^*) = 1$, and $\mathsf{TTVerify}(Ppk^*, opk_i^*, M_i^*, \sigma_{3i}^*) = 1$ for all i.
- **type II** forgery: $\vec{M}^* \notin \{\vec{M}^{(1)}, \ldots, \vec{M}^{(q)}\}$, $\sigma_2^* \in \{\sigma_2^{(1)}, \ldots, \sigma_2^{(q)}\}$, $\mathsf{Verify}(pk, \sigma_2^*, \sigma_1^*) = 1$, and $\mathsf{TTVerify}(Ppk^*, opk_i^*, M_i^*, \sigma_{3i}^*) = 1$ for all i.

type I. We show that $\mathcal{A}$ outputs a **type I** forgery with negligible probability.

Lemma 3.4.7. *If $\mathcal{A}$ outputs a* **type I** *forgery with probability ϵ, then we can construct a PPT adversary $\mathcal{B}$ that breaks the UF-CMA security of Σ_t with advantage ϵ.*

Proof of Lemma 3.4.7. The challenger samples $par \leftarrow \mathsf{Setup}(1^\lambda)$, $((pk, sk), tk) \leftarrow \mathsf{Gen}(par)$, and gives (par, pk) to $\mathcal{B}$. Then $\mathcal{B}$ gives (par, pk) to $\mathcal{A}$.

Signing queries. For all $j \in \{1, \ldots, q\}$, when $\mathcal{A}$ sends the jth query to $\mathcal{B}$, $\mathcal{B}$ responds as follows:

1. Sample $((Ppk^{(j)}, Psk^{(j)}), Pak^{(j)}) \leftarrow \mathsf{PGen}(par)$ where $Ppk^{(j)} \in \mathcal{M}_{\gamma p}$ and $Pak^{(j)} \in \mathcal{M}_p$.
2. Sample $((opk_i^{(j)}, osk_i^{(j)}), oak_i^{(j)}) \leftarrow \mathsf{SGen}(Ppk^{(j)}, Psk^{(j)}, Pak^{(j)})$ where $opk_i^{(j)} \in \mathcal{M}_{\gamma s}$ and $oak_i^{(j)} \in \mathcal{M}_s$ for $i = 1, \ldots, n$.
3. Make a query $(Pak^{(j)}, (oak_1^{(j)}, \ldots, oak_n^{(j)}))$ to the challenger who returns $\sigma_1^{(j)} \leftarrow \mathsf{Sign}(sk, (Pak^{(j)}, oak_1^{(j)}, \ldots, oak_n^{(j)}))$.
4. Let $\sigma_2^{(j)} = (Ppk^{(j)}, opk_1^{(j)}, \ldots, opk_n^{(j)})$.
5. Compute $\sigma_{3i}^{(j)} \leftarrow \mathsf{TTSign}(Psk^{(j)}, osk_i^{(j)}, M_i^{(j)})$ for $i = 1, \ldots, n$, and set $\sigma_3^{(j)} = (\sigma_{31}^{(j)}, \ldots, \sigma_{3n}^{(j)})$.
6. Return $\sigma^{(j)} = (\sigma_1^{(j)}, \sigma_2^{(j)}, \sigma_3^{(j)})$ to $\mathcal{A}$.

Output. When $\mathcal{A}$ outputs $(\vec{M}^*, \sigma_1^*, \sigma_2^*, \sigma_3^*)$, $\mathcal{B}$ outputs (σ_2^*, σ_1^*).

Since the view of $\mathcal{A}$ is identical to its view in the original UF-CMA game of Σ_t, the probability that $\sigma_2^* \notin (\sigma_2^{(1)}, \ldots, \sigma_2^{(q)})$ and $\mathsf{Verify}(pk, \sigma_2^*, \sigma_1^*) = 1$ is ϵ. As a result, $\mathcal{B}$ breaks the UF-CMA security of Σ_t with advantage ϵ, completing the proof of Lemma 3.4.7. □

Since the UF-CMA security of Σ_t implies that ϵ is negligible, the probability that $\mathcal{A}$ outputs a **type I** forgery is negligible, completing this part of the proof.

type II. We give hybrid games to show that $\mathcal{A}$ outputs a **type II** forgery with negligible probability.

Game 0: This is the original UF-CMA security game for $\mathcal{A}$. At the beginning of the game, the challenger samples $par \leftarrow \mathsf{Setup}(1^\lambda)$ and $((pk, sk), tk) \leftarrow \mathsf{Gen}(par)$, and gives (par, pk) to $\mathcal{A}$.

Signing queries. For all $j \in \{1, \ldots, q\}$, when $\mathcal{A}$ sends the jth query to the challenger, the challenger responds as follows:

1. Sample $((Ppk^{(j)}, Psk^{(j)}), Pak^{(j)}) \leftarrow \mathsf{PGen}(par)$ where $Ppk^{(j)} \in \mathcal{M}_{\gamma p}$ and $Pak^{(j)} \in \mathcal{M}_p$.
2. Sample $((opk_i^{(j)}, osk_i^{(j)}), oak_i^{(j)}) \leftarrow \mathsf{SGen}(Ppk^{(j)}, Psk^{(j)}, Pak_i^{(j)})$ where $opk_i^{(j)} \in \mathcal{M}_{\gamma s}$ and $oak_i^{(j)} \in \mathcal{M}_s$ for $i = 1, \ldots, n$.
3. Compute $\sigma_1^{(j)} \leftarrow \mathsf{Sign}(sk, (Pak^{(j)}, oak_1^{(j)}, \ldots, oak_n^{(j)}))$.
4. Let $\sigma_2^{(j)} = (Ppk^{(j)}, opk_1^{(j)}, \ldots, opk_n^{(j)})$.
5. Compute $\sigma_{3i}^{(j)} \leftarrow \mathsf{TTSign}(Psk^{(j)}, osk_i^{(j)}, M_i^{(j)})$ for $i = 1, \ldots, n$, and set $\sigma_3^{(j)} = (\sigma_{31}^{(j)}, \ldots, \sigma_{3n}^{(j)})$.
6. Return $\sigma^{(j)} = (\sigma_1^{(j)}, \sigma_2^{(j)}, \sigma_3^{(j)})$ to $\mathcal{A}$.

Output. At some point, $\mathcal{A}$ outputs $(\vec{M}^*, \sigma^* = (\sigma_1^*, \sigma_2^*, \sigma_3^*))$. $\mathcal{A}$ succeeds if $(\vec{M}^*, \sigma^*)$ is a **type II** forgery.

Game 1: This game is the same as **Game 0** except that the challenger randomly chooses $\hat{j} \leftarrow \{1, \ldots, q\}$ at the beginning of the game, and $\mathcal{A}$ succeeds if its forgery is a **type II** forgery and $\sigma_2^* = \sigma_2^{(\hat{j})}$.

Lemma 3.4.8. *If $\mathcal{A}$ succeeds with probability ϵ_0 in* **Game 0** *and ϵ_1 in* **Game 1**, *then we have $\epsilon_1 \geq \epsilon_0/q$.*

Proof of Lemma 3.4.8. Since we have $\sigma_2^* \in \{\sigma_2^{(1)}, \ldots, \sigma_2^{(q)}\}$ if the forgery is a **type II** forgery, j such that $\sigma_2^* = \sigma_2^{(j)}$ must exist. Then this lemma follows from the fact that the view of $\mathcal{A}$ is identical to its view in **Game 0**, and $\mathcal{A}$ learns no information on which $\hat{j}$ is chosen. □

Game 2: This game is the same as **Game 1** except that the challenger generates $\sigma_1^{(\hat{j})}$ by computing $\sigma_1^{(\hat{j})} \leftarrow \mathsf{TDSign}(tk, (Ppk^{(\hat{j})}, opk_1^{(\hat{j})}, \ldots, opk_n^{(\hat{j})}))$. For all $j \neq \hat{j}$, $\sigma_1^{(j)}$ is honestly generated.

Lemma 3.4.9. *If $\mathcal{A}$ succeeds with probability ϵ_1 in* **Game 1** *and ϵ_2 in* **Game 2**, *then we have $\epsilon_1 = \epsilon_2$.*

Proof of Lemma 3.4.9. This lemma follows from the fact that $\mathsf{TDSign}(tk, (Ppk^{(\hat{j})}, opk_1^{(\hat{j})}, \ldots, opk_n^{(\hat{j})}); r) = \mathsf{Sign}(sk, (Pak^{(\hat{j})}, oak_1^{(\hat{j})}, \ldots, oak_n^{(\hat{j})}); r)$ according to the correctness of a γ-TS scheme. □

Lemma 3.4.10. *If $\mathcal{A}$ succeeds with probability ϵ_2 in* **Game 2**, *then we can construct a PPT adversary $\mathcal{B}$ that breaks the UF-TT-CMA security of Σ_s with advantage at least ϵ_2.*

Proof of Lemma 3.4.10. $\mathcal{B}$ takes par and Ppk from the UF-TT-CMA challenger who samples $par \leftarrow \mathsf{Setup}(1^\lambda)$, $((Ppk, Psk), Pak) \leftarrow \mathsf{PGen}(par)$ where $Ppk \in \mathcal{M}_{\gamma p}$ and $Pak \in \mathcal{M}_p$. Then $\mathcal{B}$ samples $((pk, sk), tk) \leftarrow \mathsf{Gen}(par)$ and gives (par, pk) to $\mathcal{A}$. $\mathcal{B}$ also picks $\hat{j} \leftarrow \{1, \ldots, q\}$ uniformly, and answers the $\hat{j}$th query $\vec{M}^{(\hat{j})}$ from $\mathcal{A}$ as follows:

1. Sends the query $\vec{M}^{(\hat{j})}$ to the challenger who samples $(opk_i, osk_i, oak_i) \leftarrow \mathsf{SGen}(Ppk, Psk, Pak)$ where $opk_i \in \mathcal{M}_{\gamma s}$ and $oak_i \in \mathcal{M}_s$, and computes $\sigma_{3i}^{(\hat{j})} \leftarrow \mathsf{TTSign}(Psk, osk_i, M_i^{(\hat{j})})$ for $i = 1, \ldots, n$. Then the challenger returns $(opk_1, \ldots, opk_n)$ and $\sigma_3^{(\hat{j})} = (\sigma_{31}^{(\hat{j})}, \ldots, \sigma_{3n}^{(\hat{j})})$ to $\mathcal{B}$.
2. Compute $\sigma_1^{(\hat{j})} \leftarrow \mathsf{TDSign}(tk, (Ppk, opk_1, \ldots, opk_n))$.
3. Let $\sigma_2^{(\hat{j})} = (Ppk, opk_1, \ldots, opk_n)$.
4. Return $\sigma^{(\hat{j})} = (\sigma_1^{(\hat{j})}, \sigma_2^{(\hat{j})}, \sigma_3^{(\hat{j})})$ to $\mathcal{A}$.

Furthermore, $\mathcal{B}$ answers the jth signing query $M^{(j)}$ where $j \neq \hat{j}$ by honestly computing $\sigma^{(j)} \leftarrow \widehat{\mathsf{Sign}}(sk, M^{(j)})$ and returning $\sigma^{(j)}$ to $\mathcal{A}$.

When $\mathcal{A}$ outputs $(\vec{M}^*, \sigma_1^*, \sigma_2^*, \sigma_3^*)$, $\mathcal{B}$ finds i^* such that $M_{i^*}^* \neq M_{i^*}^{(\hat{j})}$ and outputs $(i^*, M_{i^*}^*, \sigma_{3i^*}^*)$.

Since the view of $\mathcal{A}$ is identical to its view in **Game 2**, the probability that $\mathcal{A}$ succeeds is ϵ_2, i.e., the probability that there exists i^* such that $M_{i^*}^* \neq M_{i^*}^{(\hat{j})}$ and $\mathsf{TTVerify}(Ppk, opk_{i^*}, M_{i^*}^*, \sigma_{3i^*}^*) = 1$ is ϵ_3, completing the proof of Lemma 3.4.10. □

Let ϵ_i be the probability that $\mathcal{A}$ succeeds in **Game i**. The UF-TT-CMA security of Σ_s implies that ϵ_2 is negligible. Furthermore, since $\epsilon_2 = \epsilon_1$ and $\epsilon_1 \geq \epsilon_0/q$, we have that ϵ_0, which is the probability that $\mathcal{A}$ outputs a **type II** forgery, is negligible. This completes this part of the proof.

In conclusion, $\mathcal{A}$ breaks the UF-CMA security of Sig_2 with only negligible advantage, completing the proof of Theorem 3.4.5. □

UF-RMA secure TSs + UF-TT-CMA secure TT-AKSs. Similarly to the generic constructions Sig_1 and Sig_1^*, the security of the TS scheme can be weakened to UF-RMA security if the TT-AKS scheme satisfies the random primary and secondary auxiliary key properties.

Theorem 3.4.6. *If Σ_t is a UF-RMA secure SKSP-TS scheme, and Σ_s a UF-TT-CMA secure SP-TT-AKS scheme satisfying the random primary and secondary auxiliary key properties, then* $\mathsf{Sig}_2 = (\widehat{\mathsf{Setup}}, \widehat{\mathsf{Gen}}, \widehat{\mathsf{Sign}}, \widehat{\mathsf{Verify}}, \widehat{\mathsf{VerifySK}})$ *is a UF-CMA secure FSPS scheme.*

Proof of Theorem 3.4.6. This proof is the same as the proof of Theorem 3.4.5 except that we show that $\mathcal{A}$ outputs a **type I** forgery with negligible probability by constructing a PPT adversary $\mathcal{B}$ against the UF-RMA security instead of the UF-CMA security of Σ_t in a different way as follows.

type I. We give hybrid games to show that $\mathcal{A}$ outputs a **type I** forgery with negligible probability.

Game 0: This game is the same as the original UF-CMA security game for $\mathcal{A}$. The challenger samples $par \leftarrow \mathsf{Setup}(1^\lambda)$, $((pk, sk), tk) \leftarrow \mathsf{Gen}(par)$, and gives (par, pk) to $\mathcal{A}$.

Signing queries. For all $j \in \{1, \ldots, q\}$, when $\mathcal{A}$ sends the jth query to the challenger, the challenger responds as follows:

1. Sample $((Ppk^{(j)}, Psk^{(j)}), Pak^{(j)}) \leftarrow \mathsf{PGen}(par)$ where $Ppk^{(j)} \in \mathcal{M}_{\gamma p}$ and $Pak^{(j)} \in \mathcal{M}_p$.
2. Sample $((opk_i^{(j)}, osk_i^{(j)}), oak_i^{(j)}) \leftarrow \mathsf{SGen}(Ppk^{(j)}, Psk^{(j)}, Pak^{(j)})$ where $opk_i^{(j)} \in \mathcal{M}_{\gamma s}$ and $oak_i^{(j)} \in \mathcal{M}_s$ for $i = 1, \ldots, n$.
3. Compute $\sigma_1^{(j)} \leftarrow \mathsf{Sign}(sk, (Pak^{(j)}, oak_1^{(j)}, \ldots, oak_n^{(j)}))$.
4. Let $\sigma_2^{(j)} = (Ppk^{(j)}, opk_1^{(j)}, \ldots, opk_n^{(j)})$.
5. Compute $\sigma_{3i}^{(j)} \leftarrow \mathsf{TTSign}(Psk^{(j)}, osk_i^{(j)}, M_i^{(j)})$ for $i = 1, \ldots, n$, and set $\sigma_3^{(j)} = (\sigma_{31}^{(j)}, \ldots, \sigma_{3n}^{(j)})$.
6. Return $\sigma^{(j)} = (\sigma_1^{(j)}, \sigma_2^{(j)}, \sigma_3^{(j)})$ to $\mathcal{A}$.

Output. At some point, $\mathcal{A}$ outputs $(\vec{M}^*, \sigma^* = (\sigma_1^*, \sigma_2^*, \sigma_3^*))$. $\mathcal{A}$ succeeds if $(\vec{M}^*, \sigma^*)$ is a **type I** forgery.

Game 1: This game is the same as **Game 0** except that to answer the jth query from $\mathcal{A}$, the challenger uniformly samples secondary auxiliary keys at first and then generates secondary verification/signing key pairs by running $\mathsf{AKSGen}(Ppk^{(j)}, Psk^{(j)}, Pak^{(j)}, oak_i^{(j)})$ for $i = 1, \ldots, n$. Concretely speaking, the jth signing query is answered by the challenger as follows:

1. Sample $((Ppk^{(j)}, Psk^{(j)}), Pak^{(j)}) \leftarrow \mathsf{PGen}(par)$ where $Ppk^{(j)} \in \mathcal{M}_{\gamma p}$ and $Pak^{(j)} \in \mathcal{M}_p$.

2. Sample $oak_i^{(j)} \leftarrow \mathcal{M}_s$ and compute $(opk_i^{(j)}, osk_i^{(j)}) \leftarrow \mathsf{AKSGen}(Ppk^{(j)}, Psk^{(j)}, Pak^{(j)}, oak_i^{(j)})$ where $opk^{(j)} \in \mathcal{M}_{\gamma s}$ for all $i = 1, \ldots, n$.

3. Compute $\sigma_1^{(j)} \leftarrow \mathsf{Sign}(sk, (Pak^{(j)}, oak_1^{(j)}, \ldots, oak_n^{(j)}))$.

4. Let $\sigma_2^{(j)} = (Ppk^{(j)}, opk_1^{(j)}, \ldots, opk_n^{(j)}) \in \mathcal{M}_{\gamma p} \times \mathcal{M}_{\gamma s}^n$.

5. Compute $\sigma_{3i}^{(j)} \leftarrow \mathsf{TTSign}(Psk^{(j)}, osk_i^{(j)}, M_i^{(j)})$ for $i = 1, \ldots, n$, and set $\sigma_3^{(j)} = (\sigma_{31}^{(j)}, \ldots, \sigma_{3n}^{(j)})$.

6. Return $\sigma^{(j)} = (\sigma_1^{(j)}, \sigma_2^{(j)}, \sigma_3^{(j)})$ to $\mathcal{A}$.

Lemma 3.4.11. *If $\mathcal{A}$ succeeds with probability ϵ_0 in* **Game 0** *and ϵ_1 in* **Game 1**, *then we can construct a PPT adversary $\mathcal{B}$ that breaks the random secondary auxiliary key property of Σ_s with advantage $|\epsilon_1 - \epsilon_0|$.*

Proof of Lemma 3.4.11. Taking as input par, $\mathcal{B}$ samples $((pk, sk), tk) \leftarrow \mathsf{Gen}(par)$, and gives (par, pk) to $\mathcal{A}$. For all $j \in \{1, \ldots, q\}$, when $\mathcal{A}$ sends the jth signing query $\vec{M}^{(j)} = (M_1^{(j)}, \ldots, M_n^{(j)})$ to $\mathcal{B}$, $\mathcal{B}$ answers the query as follows:

1. Send n to the oracle which is SGenO$(\cdot)$ or AKSGenO$(\cdot)$ as described in Definition 3.3.7. The oracle returns a set $(Ppk^{(j)}, Psk^{(j)}, Pak^{(j)}, \{(opk_i^{(j)}, osk_i^{(j)}, oak_i^{(j)})\}_{i=1}^n)$ where $(Ppk^{(j)}, \{opk_i^{(j)}\}_{i=1}^n) \in \mathcal{M}_{\gamma p} \times \mathcal{M}_{\gamma s}^n$ and $(Pak^{(j)}, \{oak_i^{(j)}\}_{i=1}^n) \in \mathcal{M}_p \times \mathcal{M}_s^n$.

2. Compute $\sigma_1^{(j)} \leftarrow \mathsf{Sign}(sk, (Pak^{(j)}, oak_1^{(j)}, \ldots, oak_n^{(j)}))$.

3. Let $\sigma_2^{(j)} = (Ppk^{(j)}, opk_1^{(j)}, \ldots, opk_n^{(j)})$.

4. Compute $\sigma_{3i}^{(j)} \leftarrow \mathsf{TTSign}(Psk^{(j)}, osk_i^{(j)}, M_i^{(j)})$ for $i = 1, \ldots, n$, and set $\sigma_3^{(j)} = (\sigma_{31}^{(j)}, \ldots, \sigma_{3n}^{(j)})$.

5. Return $\sigma^{(j)} = (\sigma_1^{(j)}, \sigma_2^{(j)}, \sigma_3^{(j)})$ to $\mathcal{A}$.

At some point, $\mathcal{A}$ outputs the forgery $(\vec{M}^*, \sigma^*)$. If the forgery is a **type I** forgery, $\mathcal{B}$ outputs 1. Otherwise, $\mathcal{B}$ outputs 0.

Since the view of $\mathcal{A}$ is identical to its view in **Game 0** if the oracle is SGenO$(\cdot)$ or in **Game 1** if the oracle is AKSGenO$(\cdot)$, we have that $\mathcal{B}$ breaks the random secondary auxiliary key property with advantage $|\epsilon_1 - \epsilon_0|$. Completing the proof of Lemma 3.4.11. □

Game 2: This game is the same as **Game 1** except that to answer the jth query from $\mathcal{A}$, the challenger uniformly samples primary auxiliary keys at first and then generates primary verification/signing key pairs by running $\mathsf{AKPGen}(par, Pak^{(j)})$. Concretely speaking, for all $j \in \{1, \ldots, q\}$ the jth signing query is answered by the challenger as follows:

1. Sample $Pak^{(j)} \leftarrow \mathcal{M}_p$ and compute $(Ppk^{(j)}, Psk^{(j)}) \leftarrow \mathsf{AKPGen}(par, Pak^{(j)})$ where $Ppk^{(j)} \in \mathcal{M}_{\gamma p}$.
2. Sample $oak_i^{(j)} \leftarrow \mathcal{M}_s$ and compute $(opk_i^{(j)}, osk_i^{(j)}) \leftarrow \mathsf{AKSGen}(Ppk^{(j)}, Psk^{(j)}, Pak^{(j)}, oak_i^{(j)})$ where $opk_i^{(j)} \in \mathcal{M}_{\gamma s}$ for all $i = 1, \ldots, n$.
3. Compute $\sigma_1^{(j)} \leftarrow \mathsf{Sign}(sk, (Pak^{(j)}, oak_1^{(j)}, \ldots, oak_n^{(j)}))$.
4. Let $\sigma_2^{(j)} = (Ppk^{(j)}, opk_1^{(j)}, \ldots, opk_n^{(j)})$.
5. Compute $\sigma_{3i}^{(j)} \leftarrow \mathsf{TTSign}(Psk^{(j)}, osk_i^{(j)}, M_i^{(j)})$ for $i = 1, \ldots, n$, and set $\sigma_3^{(j)} = (\sigma_{31}^{(j)}, \ldots, \sigma_{3n}^{(j)})$.
6. Return $\sigma^{(j)} = (\sigma_1^{(j)}, \sigma_2^{(j)}, \sigma_3^{(j)})$ to $\mathcal{A}$.

Lemma 3.4.12. *If $\mathcal{A}$ succeeds with probability ϵ_1 in* **Game 1** *and ϵ_2 in* **Game 2**, *then we can construct a PPT adversary $\mathcal{B}$ that breaks the random primary auxiliary key property of Σ_s with advantage $|\epsilon_2 - \epsilon_1|$.*

Proof of Lemma 3.4.12. Taking as input par, $\mathcal{B}$ samples $((pk, sk), tk) \leftarrow \mathsf{Gen}(par)$, and gives (par, pk) to $\mathcal{A}$. For all $j \in \{1, \ldots, q\}$, when $\mathcal{A}$ sends the jth signing query $\vec{M}^{(j)} = (M_1^{(j)}, \ldots, M_n^{(j)})$ to $\mathcal{B}$, $\mathcal{B}$ answers the query as follows:

1. Make a query to the oracle which is PGenO or AKPGenO as described in Definition 3.3.7. The oracle returns a set $(Ppk^{(j)}, Psk^{(j)}, Pak^{(j)})$ where $Ppk^{(j)} \in \mathcal{M}_{\gamma p}$ and $Pak^{(j)} \in \mathcal{M}_p$.
2. Sample $oak_i^{(j)} \leftarrow \mathcal{M}_s$ and compute $(opk_i^{(j)}, osk_i^{(j)}) \leftarrow \mathsf{AKSGen}(Ppk^{(j)}, Psk^{(j)}, Pak^{(j)}, oak_i^{(j)})$ where $opk_i^{(j)} \in \mathcal{M}_{\gamma s}$ for all $i = 1, \ldots, n$.
3. Compute $\sigma_1^{(j)} \leftarrow \mathsf{Sign}(sk, (Pak^{(j)}, oak_1^{(j)}, \ldots, oak_n^{(j)}))$.
4. Let $\sigma_2^{(j)} = (Ppk^{(j)}, opk_1^{(j)}, \ldots, opk_n^{(j)})$.
5. Compute $\sigma_{3i}^{(j)} \leftarrow \mathsf{TTSign}(Psk^{(j)}, osk_i^{(j)}, M_i^{(j)})$ for $i = 1, \ldots, n$, and set $\sigma_3^{(j)} = (\sigma_{31}^{(j)}, \ldots, \sigma_{3n}^{(j)})$.
6. Return $\sigma^{(j)} = (\sigma_1^{(j)}, \sigma_2^{(j)}, \sigma_3^{(j)})$ to $\mathcal{A}$.

At some point, $\mathcal{A}$ outputs the forgery $(\vec{M}^*, \sigma^*)$. If the forgery is a **type I** forgery, $\mathcal{B}$ outputs 1. Otherwise, $\mathcal{B}$ outputs 0.

Since the view of $\mathcal{A}$ is identical to its view in **Game 1** if the oracle is PGenO or in **Game 2** if the oracle is AKPGenO, we have that $\mathcal{B}$ breaks the random primary auxiliary key property with advantage $|\epsilon_2 - \epsilon_1|$, completing the proof of Lemma 3.4.12. □

Lemma 3.4.13. *If $\mathcal{A}$ succeeds with probability ϵ_2 in* **Game 2**, *then we can construct a PPT adversary $\mathcal{B}$ that breaks the UF-RMA security of Σ_t with advantage at least ϵ_2.*

Proof of Lemma 3.4.13. $\mathcal{B}$ takes par, pk, $((Pak^{(1)}, \{oak_i^{(1)}\}_{i=1}^n), \ldots, (Pak^{(q)}, \{oak_i^{(q)}\}_{i=1}^n))$, and $(\sigma_1^{(1)}, \ldots, \sigma_1^{(q)})$ from the UF-RMA challenger who samples $((pk, sk), tk) \leftarrow \mathsf{Gen}(par)$, randomly chooses $(Pak^{(j)}, (oak_1^{(j)}, \ldots, oak_n^{(j)})) \leftarrow \mathcal{M}_p \times \mathcal{M}_s^n$, and computes $\sigma^{(j)} \leftarrow \mathsf{Sign}(sk, Pak^{(j)}, (oak_1^{(j)}, \ldots, oak_n^{(j)}))$ for all $j \in \{1, \ldots, q\}$. Then $\mathcal{B}$ gives (par, pk) to $\mathcal{A}$. When $\mathcal{A}$ makes the jth signing query, $\mathcal{B}$ answers as follows:

1. Compute $(Ppk^{(j)}, Psk^{(j)}) \leftarrow \mathsf{AKPGen}(par, Pak^{(j)})$ where $Ppk^{(j)} \in \mathcal{M}_{\gamma p}$.
2. Compute $(opk_i^{(j)}, osk_i^{(j)}) \leftarrow \mathsf{AKSGen}(Ppk^{(j)}, Psk^{(j)}, Pak^{(j)}, oak_i^{(j)})$ where $opk_i^{(j)} \in \mathcal{M}_{\gamma s}$ for all $i = 1, \ldots, n$.
3. Let $\sigma_2^{(j)} = (Ppk^{(j)}, opk_1^{(j)}, \ldots, opk_n^{(j)})$.
4. Compute $\sigma_{3i}^{(j)} \leftarrow \mathsf{TTSign}(Psk^{(j)}, osk_i^{(j)}, M_i^{(j)})$ for $i = 1, \ldots, n$, and set $\sigma_3^{(j)} = (\sigma_{31}^{(j)}, \ldots, \sigma_{3n}^{(j)})$.
5. Return $\sigma^{(j)} = (\sigma_1^{(j)}, \sigma_2^{(j)}, \sigma_3^{(j)})$ to $\mathcal{A}$.

When $\mathcal{A}$ outputs $(\vec{M}^*, \sigma_1^*, \sigma_2^*, \sigma_3^*)$, $\mathcal{B}$ outputs (σ_2^*, σ_1^*) as the forgery.

Since the view of $\mathcal{A}$ is identical to its view in **Game 2**, the probability that $\mathcal{A}$ succeeds is ϵ_2, i.e., the probability that $\mathsf{Verify}(pk, \sigma_2^*, \sigma_1^*) = 1$ and $\sigma_2^* \notin \{\sigma_2^{(1)}, \ldots, \sigma_2^{(q)}\}$ is at least ϵ_2, completing the proof of Lemma 3.4.13. □

Let ϵ_i be the probability that $\mathcal{A}$ succeeds in **Game i**. The UF-RMA security of Σ_t, the random primary auxiliary key property of Σ_s, and the random secondary auxiliary key property of Σ_s imply that ϵ_2, $|\epsilon_2 - \epsilon_1|$, and $|\epsilon_1 - \epsilon_0|$ are negligible respectively, i.e., ϵ_0, which is the probability that $\mathcal{A}$ outputs a **type I** forgery, is negligible, completing this part of the proof.

Since the other parts of this proof follows from the corresponding parts of the proof of Theorem 3.4.5, $\mathcal{B}$ breaks the UF-RMA security of Σ_t with negligible advantage, completing the proof of Theorem 3.4.6. □

Instantiations of Sig_2. We give two signature schemes in Fig. 3.11 and Fig. 3.12 (in Section 3.7), which can be viewed as more efficient versions of the schemes in Fig. 3.9 and Fig. 3.10, respectively. Furthermore, we list several instantiations of Sig_2 in Table 3.1 in Section 3.5.

3.4.4 Generic Construction Sig_3 (UF-RMA): Trapdoor Signature + Binding Trapdoor Commitment

By combining a TS scheme with a BTC scheme firstly proposed in [10], we obtain a generic construction of UF-RMA secure FSPS. As in [10], a BTC scheme verifies the correctness of a commitment to a message $m \in \mathcal{M}$ by taking as input $\gamma(m) \in \mathcal{M}_\gamma$ and the opening, where $\gamma : \mathcal{M} \mapsto \mathcal{M}_\gamma$ is an efficiently computable bijection. Although the security of an instantiation derived from this construction is weakened to UF-RMA, it may achieve shorter signature size. Especially, if we instantiate the underlying TS scheme with the UF-CMA secure SPS scheme in [79] and the underlying BTC scheme with the one in [10], we have $(|pk| + |par|, |\sigma|, \sharp\text{PPE}) = (2n + 6, 3n + 7, n + 3)$ when signing n^2 group elements.[13]

Before giving the generic construction, we recall the definition of BTC. Slightly different from the original definition in [10], we define two additional bijections γ_p and γ_s, which are from the trapdoor key space to the commitment key space and from the equivocation key space to the commitment space, respectively.

Definition 3.4.1 ($(\gamma, \gamma_p, \gamma_s)$-Binding Trapdoor Commitment (BTC))**.** *A* $(\gamma, \gamma_p, \gamma_s)$-BTC *scheme consists of six polynomial-time algorithms* Setup*,* CGen*,* Com*,* CVerify*,* Sim*, and* Equiv*.*

- Setup *is a randomized algorithm that takes as input* 1^λ*, and outputs a public parameter* par*, which determines the message space* $\mathcal{M}$ *for the commitment algorithm, the message space* $\mathcal{M}_\gamma$ *for the verification algorithm, the trapdoor key space* $\mathcal{M}_p$*, the commitment key space* $\mathcal{M}_{\gamma p}$*, the equivocation key space* $\mathcal{M}_s$*, the commitment space* $\mathcal{M}_{\gamma s}$*, and efficiently computable bijections* $\gamma : \mathcal{M} \mapsto \mathcal{M}_\gamma$*,* $\gamma_p : \mathcal{M}_p \mapsto \mathcal{M}_{\gamma p}$*, and* $\gamma_s : \mathcal{M}_s \mapsto \mathcal{M}_{\gamma s}$*.*
- CGen *is a randomized algorithm that takes as input* par*, and outputs a commitment key* ck *and a trapdoor key* $\widetilde{tk}$*.*
- Com *is a randomized algorithm that takes as input a commitment key* ck *and a message* $m \in \mathcal{M}$*, and returns a commitment* c *and an opening* op*.*
- CVerify *is a deterministic algorithm that takes as input a commitment key* ck*, a commitment* c*, a message* $M \in \mathcal{M}_\gamma$*, and an opening* op*, and returns* 1 *(accept) or* 0 *(reject).*
- Sim *takes as input* par *and returns a commitment* c *and an equivocation key* ek*.*
- Equiv *takes as input* $M \in \mathcal{M}_\gamma$*,* ek*,* $\widetilde{tk}$*, and returns an opening* op*.*

A BTC scheme is required to satisfy correctness *and the* statistical trapdoor *property.*

- *Correctness is satisfied if for all* $\lambda \in \mathbb{N}$*,* $par \leftarrow \mathsf{Setup}(1^\lambda)$*,* $(ck, \widetilde{tk}) \leftarrow \mathsf{CGen}(par)$*,* $m \in \mathcal{M}$*, and* $(c, op) \leftarrow \mathsf{Com}(ck, m)$*, we have* $\mathsf{Verify}(ck, c, \gamma(m), op) = 1$*.*

[13] Actually, this construction also satisfies UF-xRMA security [3, 10], where the auxiliary hints for the adversary are messages in $\mathcal{M}$.

- *The statistical trapdoor property is satisfied if for all* $\lambda \in \mathbb{N}$, $par \leftarrow \mathsf{Setup}(1^\lambda)$, $(ck, \widetilde{tk}) \leftarrow \mathsf{CGen}(par)$, $m \in \mathcal{M}$, *the two distributions* (ck, m, c, op) *and* (ck, m, c', op') *are statistically close, where* $(c, op) \leftarrow \mathsf{Com}(ck, m)$, $(c', ek) \leftarrow \mathsf{Sim}(par)$, *and* $op' \leftarrow \mathsf{Equiv}(\gamma(m), ek, \widetilde{tk})$.

Next we recall the SP and target collision resistance properties of a BTC scheme.[14]

Definition 3.4.2 (Structure-preserving BTC (SP-BTC))**.** *A BTC scheme is said to be* structure-preserving *over a bilinear group generator* $\mathcal{G}$ *if we have*

(*a*) *a public parameter includes a group description* gk *generated by* $\mathcal{G}$,

(*b*) *commitment keys consist only of group elements in* $\mathbb{G}_1$ *and* $\mathbb{G}_2$,

(*c*) *messages for the verification algorithm and openings consist only of group elements in* $\mathbb{G}_1$ *and* $\mathbb{G}_2$, *and*

(*d*) *the verification algorithm consists only of evaluating membership in* $\mathbb{G}_1$ *and* $\mathbb{G}_2$ *and relations described by PPEs.*

Definition 3.4.3 (Target collision resistance)**.** *A* $(\gamma, \gamma_p, \gamma_s)$*-BTC scheme* $(\mathsf{Setup}, \mathsf{CGen}, \mathsf{Com}, \mathsf{CVerify}, \mathsf{Sim}, \mathsf{Equiv})$ *is said to satisfy the* target collision resistance *property if for any PPT adversary* $\mathcal{A}$, *we have*

$$\begin{aligned}\Pr[gk \leftarrow &\mathsf{Setup}(1^\lambda), (ck, \widetilde{tk}) \leftarrow \mathsf{CGen}(gk), (m_i)_{i=1}^n \leftarrow \mathcal{M}^n,\\ &((c_i, op_i) \leftarrow \mathsf{Com}(ck, m_i))_{i=1}^n, (i^*, c^*, M^*, op^*) \leftarrow \mathcal{A}(ck, (m_i)_{i=1}^n, (c_i)_{i=1}^n, (op_i)_{i=1}^n) :\\ &c^* = c_{i^*} \wedge M^* \neq \gamma(m_{i^*}) \wedge \mathsf{CVerify}(ck, c^*, M^*, op^*) = 1] \leq negl(\lambda),\end{aligned}$$

where n *is a polynomial in* λ.

Let $\Sigma_t = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{TDSign}, \mathsf{VerifySK})$ be a γ-SKSP-TS scheme with message spaces $\mathcal{M}_p \times \mathcal{M}_s^n$ and $\mathcal{M}_{\gamma p} \times \mathcal{M}_{\gamma s}^n$, and $\Sigma_s = (\mathsf{Setup}, \mathsf{CGen}, \mathsf{Com}, \mathsf{CVerify}, \mathsf{Sim}, \mathsf{Equiv})$[15] a $(\gamma, \gamma_p, \gamma_s)$-BTC scheme with message space $\mathcal{M}$ for the commitment algorithm, the message space $\mathcal{M}_\gamma$ for the verification algorithm, the trapdoor key space $\mathcal{M}_p$, the commitment key space $\mathcal{M}_{\gamma p}$, the equivocation key space $\mathcal{M}_s$, the commitment space $\mathcal{M}_{\gamma s}$, and efficiently computable bijections $\gamma : \mathcal{M} \mapsto \mathcal{M}_\gamma$, $\gamma_p : \mathcal{M}_p \mapsto \mathcal{M}_{\gamma p}$, and $\gamma_s : \mathcal{M}_s \mapsto \mathcal{M}_{\gamma s}$. Then a generic construction of FSPS denoted by $\mathsf{Sig}_3 = (\widehat{\mathsf{Setup}}, \widehat{\mathsf{Gen}}, \widehat{\mathsf{Sign}}, \widehat{\mathsf{Verify}}, \widehat{\mathsf{VerifySK}})$ with message space $\mathcal{M}^n$, where n is some fixed integer, is described as in Fig. 3.8.

Theorem 3.4.7. *If* Σ_t *is a UF-CMA secure SKSP-TS scheme, and* Σ_s *a target collision resistant SP-BTC scheme, then* $\mathsf{Sig}_3 = (\widehat{\mathsf{Setup}}, \widehat{\mathsf{Gen}}, \widehat{\mathsf{Sign}}, \widehat{\mathsf{Verify}}, \widehat{\mathsf{VerifySK}})$ *is a UF-RMA secure FSPS scheme.*

[14] The target collision resistance property defined in our thesis is stronger than the original one in [10], but still weaker than the collision resistance property in [10].

[15] As in [10], we assume that Σ_t and Σ_s share the common setup algorithm Setup.

$\widehat{\mathsf{Setup}}(1^\lambda)$:
Run $par \leftarrow \mathsf{Setup}(1^\lambda)$.
Determine the message spaces $\mathcal{M}_p \times \mathcal{M}_s^n$
and $\mathcal{M}_{\gamma p} \times \mathcal{M}_{\gamma s}^n$ for Σ_t.
Define $\gamma : \mathcal{M}_p \times \mathcal{M}_s^n \mapsto \mathcal{M}_{\gamma p} \times \mathcal{M}_{\gamma s}^n$.
Determine the message spaces $\mathcal{M}^n$ and $\mathcal{M}_\gamma^n$,
commitment key space $\mathcal{M}_{\gamma p}$,
commitment space $\mathcal{M}_{\gamma s}$,
trapdoor key space $\mathcal{M}_p$,
and equivacation key space $\mathcal{M}_s$ for Σ_s.
Define $\gamma : \mathcal{M} \mapsto \mathcal{M}_\gamma$,
$\gamma_p : \mathcal{M}_p \mapsto \mathcal{M}_{\gamma p}$, and $\gamma_s : \mathcal{M}_s \mapsto \mathcal{M}_{\gamma s}$
where
$(\gamma_p(x_0), \gamma_s(x_1), \ldots, \gamma_s(x_n)) = \gamma(x_0, x_1 \ldots, x_n)$.
Return public parameter par.

$\widehat{\mathsf{Gen}}(par)$:
$((pk, sk), tk) \leftarrow \mathsf{Gen}(par)$.
Return (pk, sk).

$\widehat{\mathsf{VerifySK}}(pk, sk)$:
Return 1 if $\mathsf{VerifySK}(pk, sk) = 1$.
Return 0 otherwise.

$\widehat{\mathsf{Sign}}(sk, \vec{M})$:
Parse $\vec{M} = (M_1, \ldots, M_n) \in \mathcal{M}^n$.
$(ck, \widetilde{tk}) \leftarrow \mathsf{CGen}(par)$.
$(c_i, ek_i) \leftarrow \mathsf{Sim}(gk)$ for $i = 1, \ldots, n$.
$\sigma_1 \leftarrow \mathsf{Sign}(sk, (\widetilde{tk}, ek_1, \ldots, ek_n))$.
$\sigma_2 = (ck, c_1, \ldots, c_n)$.
$\sigma_{3i} \leftarrow \mathsf{Equiv}(M_i, ek_i, \widetilde{tk})$ for $i = 1, \ldots, n$.
$\sigma_3 = (\sigma_{31}, \ldots, \sigma_{3n})$.
Return $\sigma = (\sigma_1, \sigma_2, \sigma_3)$.

$\widehat{\mathsf{Verify}}(pk, \vec{M}, \sigma)$:
Parse $\vec{M} = (M_1, \ldots, M_n) \in \mathcal{M}^n$,
$\sigma = (\sigma_1, \sigma_2, \sigma_3)$,
$\sigma_2 = (ck, c_1, \ldots, c_n)$,
and $\sigma_3 = (\sigma_{31}, \ldots, \sigma_{3n})$.
Return 1
if $\mathsf{Verify}(pk, ck, c_1, \ldots, c_n) = 1$
and $\mathsf{CVerify}(ck, c_i, M_i, \sigma_{3i}) = 1$ for all i.
Return 0 otherwise.

Figure 3.8: Generic construction Sig_3: TS + BS.

Proof of Theorem 3.4.7. Since the proof that Sig_3 is FSP is straightforward and similar to the corresponding part of the proof of Theorem 3.4.1, we omit it here.

Now we prove that Sig_3 satisfies UF-RMA security. Let $\mathcal{A}$ be any PPT adversary. For $i = 1, \ldots, q$ where q denotes the number of the queries made by the adversary, let $\vec{M}^{(j)} = (M_1^{(j)}, \ldots, M_n^{(j)})$ and $\sigma^{(j)} = (\sigma_1^{(j)}, \sigma_2^{(j)}, \sigma_3^{(j)})$ (where $\sigma_2^{(j)} = (ck^{(j)}, c_1^{(j)}, \ldots, c_n^{(j)})$ and $\sigma_3^{(j)} = (\sigma_{31}^{(j)}, \ldots, \sigma_{3n}^{(j)})$) be the answer to the jth signing query. At some point, $\mathcal{A}$ outputs $(\vec{M}^*, (\sigma_1^*, \sigma_2^*, \sigma_3^*))$ (where $\vec{M}^* = (M_1^*, \ldots, M_n^*)$, $\sigma_2^* = (ck^*, c_1^*, \ldots, c_n^*)$, and $\sigma_3^* = (\sigma_{31}^*, \ldots, \sigma_{3n}^*)$) as the forgery.

To win the UF-RMA game with non-negligible probability ϵ, $\mathcal{A}$ has to output a forgery, which is one of the following two types.

- **type I** forgery: $\vec{M}^* \notin \{\vec{M}^{(1)}, \ldots, \vec{M}^{(q)}\}$, $\sigma_2^* \notin (\sigma_2^{(1)}, \ldots, \sigma_2^{(q)})$, $\mathsf{Verify}(pk, \sigma_2^*, \sigma_1^*) = 1$, and $\mathsf{CVerify}(ck^*, c_i^*, M_i^*, \sigma_{3i}^*) = 1$ for all i.
- **type II** forgery: $\vec{M}^* \notin \{\vec{M}^{(1)}, \ldots, \vec{M}^{(q)}\}$, $\sigma_2^* \in \{\sigma_2^{(1)}, \ldots, \sigma_2^{(q)}\}$, $\mathsf{Verify}(pk, \sigma_2^*, \sigma_1^*) = 1$, and $\mathsf{CVerify}(ck^*, c_i^*, M_i^*, \sigma_{3i}^*) = 1$ for all i.

type I. We show that $\mathcal{A}$ outputs a **type I** forgery with negligible probability.

Lemma 3.4.14. *If $\mathcal{A}$ outputs a* **type I** *forgery with probability ϵ, then we can construct a PPT adversary $\mathcal{B}$ that breaks the UF-CMA security of Σ_t with advantage ϵ.*

Proof of Lemma 3.4.14. The challenger samples $par \leftarrow \mathsf{Setup}(1^\lambda)$, $((pk, sk), tk) \leftarrow \mathsf{Gen}(par)$, and gives (par, pk) to $\mathcal{B}$. Then $\mathcal{B}$ gives (par, pk) to $\mathcal{A}$.

Signing queries. For all $j \in \{1, \ldots, q\}$, when $\mathcal{A}$ sends the jth query to $\mathcal{B}$, $\mathcal{B}$ responds as follows:

1. Randomly choose $\vec{M}^{(j)}$ from $\mathcal{M}_\gamma^n$.
2. Sample $(ck^{(j)}, \widetilde{tk}^{(j)}) \leftarrow \mathsf{CGen}(par)$ where $ct^{(j)} \in \mathcal{M}_{\gamma p}$ and $\widetilde{tk}^{(j)} \in \mathcal{M}_p$.
3. Sample $(c_i^{(j)}, ek_i^{(j)}) \leftarrow \mathsf{Sim}(gk)$ where $c_i^{(j)} \in \mathcal{M}_{\gamma s}$ and $ek_i^{(j)} \in \mathcal{M}_s$ for $i = 1, \ldots, n$.
4. Make a query $(\widetilde{tk}^{(j)}, (ek_1^{(j)}, \ldots, ek_n^{(j)}))$ to the challenger who returns $\sigma_1^{(j)} \leftarrow \mathsf{Sign}(sk, (\widetilde{tk}^{(j)}, ek_1^{(j)}, \ldots, ek_n^{(j)}))$.
5. Let $\sigma_2^{(j)} = (ck^{(j)}, c_1^{(j)}, \ldots, c_n^{(j)})$.
6. Compute $\sigma_{3i}^{(j)} \leftarrow \mathsf{Equiv}(M_i^{(j)}, ek_i^{(j)}, \widetilde{tk}^{(j)})$ for $i = 1, \ldots, n$, and set $\sigma_3^{(j)} = (\sigma_{31}^{(j)}, \ldots, \sigma_{3n}^{(j)})$.
7. Return $\vec{M}^{(j)}$ and $\sigma^{(j)} = (\sigma_1^{(j)}, \sigma_2^{(j)}, \sigma_3^{(j)})$ to $\mathcal{A}$.

Output. When $\mathcal{A}$ outputs $(\vec{M}^*, \sigma_1^*, \sigma_2^*, \sigma_3^*)$, $\mathcal{B}$ outputs (σ_2^*, σ_1^*).

Since the view of $\mathcal{A}$ is identical to its view in the original UF-CMA game of Σ_t, the probability that $\sigma_2^* \notin (\sigma_2^{(1)}, \ldots, \sigma_2^{(q)})$ and $\mathsf{Verify}(pk, \sigma_2^*, \sigma_1^*) = 1$ is ϵ. As a result, $\mathcal{B}$ breaks the UF-CMA security of Σ_t with advantage ϵ, completing the proof of Lemma 3.4.14. □

Since the UF-CMA security of Σ_t implies that ϵ is negligible, the probability that $\mathcal{A}$ outputs a **type I** forgery is negligible, completing this part of the proof.

type II. We give hybrid games to show that $\mathcal{A}$ outputs a **type II** forgery with negligible probability.

Game 0: This is the original UF-RMA security game for $\mathcal{A}$. At the beginning of the game, the challenger samples $par \leftarrow \mathsf{Setup}(1^\lambda)$ and $((pk, sk), tk) \leftarrow \mathsf{Gen}(par)$, and gives (par, pk) to $\mathcal{A}$.

Signing queries. For all $j \in \{1, \ldots, q\}$, when $\mathcal{A}$ sends the jth query to the challenger, the challenger responds as follows:

1. Randomly choose $\vec{M}^{(j)}$ from $\mathcal{M}_\gamma^n$.
2. Sample $(ck^{(j)}, \widetilde{tk}^{(j)}) \leftarrow \mathsf{CGen}(par)$ where $ck^{(j)} \in \mathcal{M}_{\gamma p}$ and $ek^{(j)} \in \mathcal{M}_p$.
3. Sample $(c_i^{(j)}, ek_i^{(j)}) \leftarrow \mathsf{Sim}(gk)$ where $c_i^{(j)} \in \mathcal{M}_{\gamma s}$ and $ek_i^{(j)} \in \mathcal{M}_s$ for $i = 1, \ldots, n$.

4. Compute $\sigma_1^{(j)} \leftarrow \mathsf{Sign}(sk, (\widetilde{tk}^{(j)}, ek_1^{(j)}, \ldots, ek_n^{(j)}))$.

5. Let $\sigma_2^{(j)} = (ck^{(j)}, c_1^{(j)}, \ldots, c_n^{(j)})$.

6. Compute $\sigma_{3i}^{(j)} \leftarrow \mathsf{Equiv}(M_i^{(j)}, ek_i^{(j)}, \widetilde{tk}^{(j)})$ for $i = 1, \ldots, n$, and set $\sigma_3^{(j)} = (\sigma_{31}^{(j)}, \ldots, \sigma_{3n}^{(j)})$.

7. Return $\vec{M}^{(j)}$ and $\sigma^{(j)} = (\sigma_1^{(j)}, \sigma_2^{(j)}, \sigma_3^{(j)})$ to $\mathcal{A}$.

Output. At some point, $\mathcal{A}$ outputs $(\vec{M}^*, \sigma^* = (\sigma_1^*, \sigma_2^*, \sigma_3^*))$. $\mathcal{A}$ succeeds if $(\vec{M}^*, \sigma^*)$ is a **type II** forgery.

Game 1: This game is the same as **Game 0** except that the challenger randomly chooses $\hat{j} \leftarrow \{1, \ldots, q\}$ at the beginning of the game and $\mathcal{A}$ succeeds if its forgery is a **type II** forgery and $\sigma_2^* = \sigma_2^{(\hat{j})}$.

Lemma 3.4.15. *If $\mathcal{A}$ succeeds with probability ϵ_0 in* **Game 0** *and ϵ_1 in* **Game 1**, *then we have $\epsilon_1 \geq \epsilon_0/q$.*

Proof of Lemma 3.4.15. Since we have $\sigma_2^* \in \{\sigma_2^{(1)}, \ldots, \sigma_2^{(q)}\}$ if the forgery is a **type II** forgery, j such that $\sigma_2^* = \sigma_2^{(j)}$ must exist. Then this lemma follows from the fact that the view of $\mathcal{A}$ is identical to its view in **Game 0**, and $\mathcal{A}$ learns no information on which $\hat{j}$ is chosen. □

Game 2: This game is the same as **Game 1** except that the challenger generates $\sigma_1^{(\hat{j})}$ as follows:

1. Randomly choose $\vec{m}^{(\hat{j})} = (m_1^{(\hat{j})}, \ldots, m_n^{(\hat{j})})$ from $\mathcal{M}^n$.

2. Sample $(ck^{(\hat{j})}, \widetilde{tk}^{(\hat{j})}) \leftarrow \mathsf{CGen}(par)$ where $ck^{(\hat{j})} \in \mathcal{M}_{\gamma p}$ and $ek^{(\hat{j})} \in \mathcal{M}_p$.

3. Compute $(c_i^{(\hat{j})}, \sigma_{3i}^{(\hat{j})}) \leftarrow \mathsf{Com}(ck^{(\hat{j})}, m_i^{(\hat{j})})$ for $i = 1, \ldots, n$, and set $\sigma_2^{(\hat{j})} = (ck^{(\hat{j})}, c_1^{(\hat{j})}, \ldots, c_n^{(\hat{j})})$ and $\sigma_3^{(\hat{j})} = (\sigma_{31}^{(\hat{j})}, \ldots, \sigma_{3n}^{(\hat{j})})$.

4. Compute $\sigma_1^{(\hat{j})} \leftarrow \mathsf{TDSign}(tk, (ck^{(\hat{j})}, c_1^{(\hat{j})}, \ldots, c_n^{(\hat{j})}))$.

5. Return $\sigma^{(\hat{j})} = (\sigma_1^{(\hat{j})}, \sigma_2^{(\hat{j})}, \sigma_3^{(\hat{j})})$ to $\mathcal{A}$.

Furthermore, for all $j \neq \hat{j}$, $\sigma^{(j)}$ is honestly generated.

Lemma 3.4.16. *If $\mathcal{A}$ succeeds with probability ϵ_1 in* **Game 1** *and ϵ_2 in* **Game 2**, *then we have that $|\epsilon_2 - \epsilon_1|$ is negligible.*

Proof of Lemma 3.4.16. This lemma follows from the fact that $\mathsf{TDSign}(tk, (ck^{(\hat{j})}, c_1^{(\hat{j})}, \ldots, c_n^{(\hat{j})}); r) = \mathsf{Sign}(sk, (\widetilde{tk}^{(j)}, ek_1^{(j)}, \ldots, ek_n^{(j)}); r)$ for all r according to the correctness of a γ-TS scheme, and the distribution of $(ck^{(j)}, (m_i^{(j)})_{i=1}^n, (c_i^{(j)})_{i=1}^n, (\sigma_{3i}^{(j)})_{i=1}^n)$ generated in **Game 2** is the statistically close to that of **Game 1** according to the statistical trapdoor property of a BTC scheme. □

Lemma 3.4.17. *If $\mathcal{A}$ succeeds with probability ϵ_2 in* **Game 2**, *then we can construct a PPT adversary $\mathcal{B}$ that breaks the target collision resistance property of Σ_s with advantage at least ϵ_2.*

Proof of Lemma 3.4.17. $\mathcal{B}$ takes $(par, ck, \{m_i\}_{i=1}^n, \{c_i\}_{i=1}^n, \{op_i\}_{i=1}^n)$ from the target collision resistance challenger who samples $par \leftarrow \mathsf{Setup}(1^\lambda)$, $(ck, \widetilde{tk}) \leftarrow \mathsf{CGen}(par)$ (where $ck \in \mathcal{M}_{\gamma p}$ and $\widetilde{tk} \in \mathcal{M}_p$), and $m_i \leftarrow \mathcal{M}$ for $i = 1, \ldots, n$, and computes $(c_i, op_i) \leftarrow \mathsf{Com}(ck, m_i)$ for $i = 1, \ldots, n$. Then $\mathcal{B}$ samples $((pk, sk), tk) \leftarrow \mathsf{Gen}(par)$ and $\hat{j} \leftarrow \{1, \ldots, q\}$, gives (par, pk) to $\mathcal{A}$, and answers the $\hat{j}$th query as follows:

1. Set $ck^{(\hat{j})} = ck$, $(c_i^{(\hat{j})}, \sigma_{3i}^{(\hat{j})}) = (c_i, op_i)$ for $i = 1, \ldots, n$, $\sigma_2^{(\hat{j})} = (ck^{(\hat{j})}, c_1^{(\hat{j})}, \ldots, c_n^{(\hat{j})})$, and $\sigma_3^{(\hat{j})} = (\sigma_{31}^{(\hat{j})}, \ldots, \sigma_{3n}^{(\hat{j})})$.
2. Compute $\sigma_1^{(\hat{j})} \leftarrow \mathsf{TDSign}(tk, (ck^{(\hat{j})}, c_1^{(\hat{j})}, \ldots, c_n^{(\hat{j})}))$.
3. Return $\vec{M}^{(\hat{j})} = (M_i^{(\hat{j})})_{i=1}^n$, where $M_i^{(\hat{j})} = \gamma(m_i)$ for $i = 1, \ldots, n$, and $\sigma^{(\hat{j})} = (\sigma_1^{(\hat{j})}, \sigma_2^{(\hat{j})}, \sigma_3^{(\hat{j})})$ to $\mathcal{A}$.

Furthermore, $\mathcal{B}$ answers the jth signing query where $j \neq \hat{j}$ by sampling $\vec{M}^{(j)} \leftarrow \mathcal{M}^n$ and honestly computing $\sigma^{(j)} \leftarrow \widehat{\mathsf{Sign}}(sk, \vec{M}^{(j)})$ and returning $\sigma^{(j)}$ to $\mathcal{A}$.

When $\mathcal{A}$ outputs $(\vec{M}^*, \sigma_1^*, \sigma_2^*, \sigma_3^*)$, $\mathcal{B}$ finds i^* such that $M_{i^*}^* \neq M_{i^*}^{(\hat{j})}$ and $c_{i^*}^* = c_{i^*}^{(\hat{j})}$ and outputs $(c_{i^*}^*, M_{i^*}^*, \sigma_{3i^*}^*)$. If such i^* does not exists, $\mathcal{B}$ aborts.

Since the view of $\mathcal{A}$ is identical to its view in **Game 2**, the probability that $\mathcal{A}$ succeeds is ϵ_2, i.e., the probability that there exists i^* such that $c_{i^*}^* = c_{i^*}^{(\hat{j})}$, $M_{i^*}^* \neq M_{i^*}^{(\hat{j})}$, and $\mathsf{CVerify}(ck, c_{i^*}^*, M_{i^*}^*, \sigma_{3i^*}^*) = 1$ is ϵ_2. As a result, the probability that $\mathcal{B}$ breaks the target collision resistance property is ϵ_2, completing the proof of Lemma 3.4.17. □

Let ϵ_i be the probability that $\mathcal{A}$ succeeds in **Game i**. The target collision resistance property of Σ_s implies that ϵ_2 is negligible. Furthermore, since $|\epsilon_2 - \epsilon_1|$ is negligible and $\epsilon_1 \geq \epsilon_0/q$, we have that ϵ_0, which is the probability that $\mathcal{A}$ outputs a **type II** forgery, is negligible, completing this part of the proof.

In conclusion, $\mathcal{A}$ breaks the UF-RMA security of Sig_3 with only negligible advantage, completing the proof of Theorem 3.4.7. □

Remark. For the BTC scheme in [10], we can generate commitment keys by using randomly chosen trapdoor keys, and commitments by using randomly chosen equivocation keys, while keeping the distributions unchanged. This allows us to relax the security of Σ_t from UF-CMA to UF-RMA. Furthermore, this construction satisfies UF-xRMA security, where the auxiliary hints for the adversary are the pre-images of messages w.r.t. the injection γ.

Instantiation of Sig_3. As mentioned before, if we instantiate the underlying TS scheme with the UF-CMA secure SPS scheme in [79] (based on the SXDH assumption) and the underlying BTC scheme with the one in [10] (based on the SXDH assumption), this construction achieves $(|m|, |pk| + |par|, |\sigma|, \sharp\text{PPE}) = (n^2, 2n + 6, 3n + 7, n + 3)$. As far as we know, it achieves the shortest signature size among all the FSPSs for a vector of unilateral messages under standard assumptions.

3.5 Instantiations of UF-CMA Secure FSPS (FAS)

In this section, we give several instantiations derived from our generic constructions of UF-CMA secure FSPS, which are summarized in Table 3.1. For notational convenience, we denote these schemes as `(A)`, `(B)`, `(C)`, `(D)`, `(E)`, `(F)`, `(G)`, `(H)`, and `(I)`, respectively (see the first column of Table 3.1). `(F)` and `(G)` correspond to our results given in Table 1.1 in Introduction, and `(C)` corresponds to our result given in Table 1.2. Many of our instantiations are FAS schemes,[16] and typically, when signing n^2 group elements, Sig_1 needs $O(n^2)$ verification/signature key elements and $O(1)$ PPEs,[17] while Sig_1^* and Sig_2 need $O(n)$ verification/signature key elements and PPEs.

Signing key sizes and number of pairings. In Section 3.5.1, Section 3.5.2, and Section 3.5.3, we give remarks on the instantiations of Sig_1, Sig_1^*, and Sig_2, respectively. We refer the reader to Section 3.8 for signing key sizes, and Section 3.9 for the numbers of pairings required in verification.

3.5.1 Sig_1: SKSP-TS + SP-AKS

We give parameters of three instantiations for Sig_1, which are `(A)`, `(B)`, and `(C)`. Especially, `(B)` is an FSPS scheme in the type I bilinear map and `(C)` is an FSPS scheme in the generic group model.

The verification key size $|pk|$ of `(C)` is $(n_1, 0) \leq (n_1^2, 0)$, which makes it automorphic, while its efficiency (considering public parameter size, signature size, and verification cost) is very close to `(G)` (i.e., the FSPS scheme in [69]). As far as we know, `(C)` is the most efficient FAS scheme by now. Note that if we follow the definition of basic signature in [5], which

[16] It is not hard to see that FAS schemes in Table 3.1 may lose the automorphic property when n_1 (or n_2 or n) is an extremely small number. Furthermore, when k (which is independent of the message size) is a large number, the message size has to be made reasonably large to keep the automorphic property.

[17] There are some exceptions, e.g., `(C)` in Table 3.1.

	Const.	Auto.	Assumption	Parameter	♯ Group element (PPE)
AKO+15 [10]	Generic construction 1	×	SXDH XDLIN	$\|m\|$ $\|pk\| + \|par\|$ $\|\sigma\|$ ♯ PPE	$(n_1^2, 0)$ (n_1^2+5, n_1^2+11) $(7, 3n_1^2+7)$ $2n_1^2+7$
AKO+15 [10]	Generic construction 2	×	SXDH XDLIN	$\|m\|$ $\|pk\| + \|par\|$ $\|\sigma\|$ ♯ PPE	$(n_1^2, 0)$ $(6n_1+13, 4)$ $(2n_1+4, 2n_1+7)$ n_1+5
Gro15 [69]	FSPS scheme	×	Generic	$\|m\|$ $\|pk\| + \|par\|$ $\|\sigma\|$ ♯ PPE	$(n_1^2, 0)$ $(2n_1-1, 1)$ (n_1+1, n_1) n_1+1
(A): KPW15 [79] (CMA) + KPW15 [79](otCMA)	Sig_1	×	$\mathcal{D}_k$-MDDH $(\mathbb{G}_1, \mathbb{G}_2)$	$\|m\|$ $\|pk\| + \|par\|$ $\|\sigma\|$ ♯ PPE	$(n_1^2, 0)$ $\Big((n_1^2k+3k+3+\mathsf{RE}(\mathcal{D}_k))k+\mathsf{RE}(\mathcal{D}_k), 0\Big)$ $\Big(k+2, (n_1^2+4)k+3+\mathsf{RE}(\mathcal{D}_k)\Big)$ $3k+1$
(B): ADK+13 [4] (CMA) + ADK+13 [4] (CMA)	Sig_1	×	2-Lin $(\mathbb{G}_1 = \mathbb{G}_2)$	$\|m\|$ $\|pk\| + \|par\|$ $\|\sigma\|$ ♯ PPE	n^2 $4n^2+60$ $2n^2+48$ 14
(C): Gro15 [69] (CMA) + Gro15 [69] (CMA)	Sig_1	√	Generic	$\|m\|$ $\|pk\| + \|par\|$ $\|\sigma\|$ ♯ PPE	$(n_1^2, 0)$ $(2n_1, 1)$ (n_1+2, n_1+3) n_1+3
(D): KPW15 [79] (CMA) + KPW15 [79](CMA)	Sig_1^*	√	$\mathcal{D}_k$-MDDH $(\mathbb{G}_1, \mathbb{G}_2)$	$\|m\|$ $\|pk\| + \|par\|$ $\|\sigma\|$ ♯ PPE	$(n_1^2, 0)$ $\Big((n_1k+2k^2+6k+3+\mathsf{RE}(\mathcal{D}_k))k+\mathsf{RE}(\mathcal{D}_k), 0\Big)$ $\Big(3n_1k+3n_1+1, (n_1+2k+7)k+n_1+3+\mathsf{RE}(\mathcal{D}_k)\Big)$ $(2k+1)(n_1+1)$
(E): ADK+13 [4] (CMA) + ADK+13 [4] (CMA)	Sig_1^*	√	2-Lin $(\mathbb{G}_1 = \mathbb{G}_2)$	$\|m\|$ $\|pk\| + \|par\|$ $\|\sigma\|$ ♯ PPE	n^2 $4n+64$ $16n+36$ $7(n+1)$
(F): KPW15 [79] (CMA) + KPW15 [80] (TT)	Sig_2	√	$\mathcal{D}_k$-MDDH $(\mathbb{G}_1, \mathbb{G}_2)$	$\|m\|$ $\|pk\| + \|par\|$ $\|\sigma\|$ ♯ PPE	$(n_1^2, 0)$ $\Big((2n_1k+2k+3+\mathsf{RE}(\mathcal{D}_k))k+\mathsf{RE}(\mathcal{D}_k), 0\Big)$ $\Big((k+1)n_1+1, 2n_1k+3k+3+\mathsf{RE}(\mathcal{D}_k)\Big)$ kn_1+2k+1
(G): KPW15 [79] (CMA) (bilateral) + KPW15 [80] (TT)	Sig_2	√	$\mathcal{D}_k$-MDDH $(\mathbb{G}_1, \mathbb{G}_2)$	$\|m\|$ $\|pk\| + \|par\|$ $\|\sigma\|$ ♯ PPE	(n_1^2, n_2^2) $\Big((2n_1k+3k+3+\mathsf{RE}(\mathcal{D}_k))k+\mathsf{RE}(\mathcal{D}_k),$ $(2n_2k+\mathsf{RE}(\mathcal{D}_k))k+\mathsf{RE}(\mathcal{D}_k)\Big)$ $\Big((k+1)n_1+2n_2k+k+2+\mathsf{RE}(\mathcal{D}_k),$ $(k+1)n_2+2n_1k+4k+3+\mathsf{RE}(\mathcal{D}_k)\Big)$ $k(n_1+n_2)+3k+1$
(H): ADK+13 [4] (CMA) + ADK+13 [4] (TT(TOS))	Sig_2	√	2-Lin $(\mathbb{G}_1 = \mathbb{G}_2)$	$\|m\|$ $\|pk\| + \|par\|$ $\|\sigma\|$ ♯ PPE	n^2 $6n+30$ $6n+12$ $2n+7$
(I) ACD+12 [3] (CMA) + ACD+12 [3] (TT)	Sig_2	×	SXDH XDLIN	$\|m\|$ $\|pk\| + \|par\|$ $\|\sigma\|$ ♯ PPE	$(n_1^2, 0)$ $(2n_1+14, 7)$ $(2n_1+4, 2n_1+8)$ n_1+4

Table 3.1: Previously proposed FSPSs and FSPSs derived from our work. "Const." is short for "Construction" and "Auto." is short for "Automorphic". We use "(A): KPW15 [79] (CMA) + KPW15 [79] (otCMA)" to denote that the underlying TS (respectively, AKS) scheme of (A) is adapted from the UF-CMA secure (respectively, UF-otCMA secure) SPS scheme in [79]. We use the same argument for others except that the three FSPSs in the top denote the ones proposed in [10] and [69]. Especially, "ADK+13 [4] (TT(TOS))" denotes the tagged one-time signature scheme in [4]. Notation (x, y) denotes x elements in $\mathbb{G}_1$ and y elements in $\mathbb{G}_2$. As noted in Introduction, we do not count the two generators in the bilinear groups in the parameters.

allows no trusted setup except for bilinear group generation, then (C) is not automorphic, and the most efficient FAS scheme becomes (F), in Table 3.1.

Remark. Note that to make the underlying TS scheme compatible with the underlying AKS scheme in (B) (and (E)), we need to adjust the message space of the TS scheme. This means that the SKSP property of this TS scheme is not implied by the SPS property of the original scheme (i.e., "ADK+13 [4] (CMA)"). However, we can still show that it is SKSP by using the conversion in part II of Theorem 3.2.2. Furthermore, if we relax the definition of FSP property as described below Definition 3.1.3, we do not need this adjustment.

3.5.2 Sig_1^*: SKSP-TS + SP-AKS (UF-CMA)

We give parameters of two instantiations for Sig_1^*, which are (D) and (E), where (E) is in the type I bilinear map. Both of them are automorphic.

It is obvious that most instantiations derived from Sig_1 have verification key and signature sizes linear in the message size, which makes them less efficient and not automorphic (since verification keys have larger size than messages). However, as shown in Table 3.1, as a variation of Sig_1, Sig_1^* allows us to obtain FSPSs with shorter signatures and verification keys if the underlying SP-AKS scheme is UF-CMA secure. This fact shows that many existing SPSs imply the existence of a corresponding efficient FSPS scheme since any well-formed SPS scheme (respectively, SPS scheme with an algebraic key generation algorithm) can be converted into an SKSP-TS (respectively, SP-AKS) scheme.

3.5.3 Sig_2: SKSP-TS + SP-TT-AKS

We give parameters of four instantiations for Sig_2, which are (F), (G), (H), and (I), where (H) is in the type I bilinear map. The only one that is *not* automorphic among them is (I). Here, (G) is achieved by using a UF-CMA secure SKSP-TS scheme to sign auxiliary keys of two SP-TT-AKS schemes with verification keys consisting of elements in $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively, and (H) is achieved by using a SKSP-TS scheme to sign auxiliary keys of the tag-based one-time signature scheme in [4]. Tag based one-time signatures can be treated as a special case of two-tier signatures where secondary signing keys are the same as secondary verification keys.

For $k = 1$ (SXDH), we have $(|m|, |pk + par|, |\sigma|, \sharp\text{PPEs}) = (n_1^2, 2n_1 + 7, 4n_1 + 8, n_1 + 3)$ in (F), while the most efficient instantiation given in [10] achieves $(|m|, |pk + par|, |\sigma|, \sharp\text{PPEs}) = (n_1^2, 6n_1 + 17, 4n_1 + 11, n_1 + 5)$ and is not automorphic. Furthermore, by sacrificing efficiency, (F) can be based on weaker assumptions.

(G) achieves $(|m|, |pk| + |par|, |\sigma|, \sharp\text{PPEs}) = (n_1^2, 2n_1 + 2n_2 + 10, 4n_1 + 4n_2 + 12, n_1 + n_2 + 4)$ for $k = 1$ (SDXH), which has the shortest verification key size, signature size, and lowest cost in verification among all FSPS and FAS schemes with a bilateral message space based on standard assumptions by now, as far as we know.

(H) is the most efficient FSPS and FAS scheme in the type I bilinear map, as far as we know.

3.6 Instantiations of One-time FSPS (FAS)

In this section, we give several UF-otCMA secure instantiations derived from our generic constructions, which have relatively better efficiency compared with the UF-CMA secure ones in Section 3.5. We summarize them in Table 3.2 and denote them as `(J)`, `(K)`, `(L)`, `(M)`, `(N)`, and `(O)`, respectively.

The underlying SKSP-TS schemes (respectively, SP-AKS schemes) are adapted from the UF-otCMA and UF-otRMA secure SPS schemes (respectively, UF-otCMA and UF-TT-CMA secure schemes) in [79, 80]. Note that the underlying assumptions of `(K)`, `(M)`, and `(O)` are $\mathcal{D}_k$-MDDH assumptions, where we specify $\mathcal{D}_k$ as $\mathcal{SC}_k$, $\mathcal{L}_k$, or $\mathcal{U}_k$ (see Section 2.3).[18] The reason is that to make AKSs compatible with (ot)UF-RMA secure TSs, we have to make sure that the auxiliary keys are sampled from uniform distributions, while (representations of) matrices sampled from $\mathcal{SC}_k$, $\mathcal{L}_k$, and $\mathcal{U}_k$ satisfy our requirement.

The most efficient one is adapted from the otRMA secure scheme and the UF-TT-CMA secure one in [79, 80] (see "`KPW15` [79] (otRMA) + `KPW15` [80] (TT)", Table 3.2), which achieves $(|pk| + |par|, |\sigma|, \sharp\text{PPEs}) = (2n_1 + 3, 4n_1 + 2, n_1 + 1)$ under the SXDH assumption.

	Const.	Auto.	Assumption	Parameter	♯ Group element (PPE)
`(J)`: `KPW15` [79] (otCMA) + `KPW15` [79] (otCMA)	Sig_1	×	$\mathcal{D}_k$-MDDH $(\mathbb{G}_1, \mathbb{G}_2)$	$\lvert m\rvert$ $\lvert pk\rvert + \lvert par\rvert$ $\lvert\sigma\rvert$ ♯ PPE	$(n_1^2, 0)$ $\left((n_1^2k + k + 1 + \mathsf{RE}(\mathcal{D}_k))k + \mathsf{RE}(\mathcal{D}_k), 0\right)$ $\left(k + 1, (n_1^2 + 2)k + 1 + \mathsf{RE}(\mathcal{D}_k)\right)$ $2k$
`(K)`: `KPW15` [79] (otRMA) + `KPW15` [79] (otCMA)	Sig_1	×	$\mathcal{D}_k$-MDDH $(\mathbb{G}_1, \mathbb{G}_2)$	$\lvert m\rvert$ $\lvert pk\rvert + \lvert par\rvert$ $\lvert\sigma\rvert$ ♯ PPE	$(n_1^2, 0)$ $\left((n_1^2k + k + 1 + \mathsf{RE}(\mathcal{D}_k))k + \mathsf{RE}(\mathcal{D}_k), 0\right)$ $\left(k + 1, (n_1^2 + 2)k + \mathsf{RE}(\mathcal{D}_k)\right)$ $2k$
`(L)`: `KPW15` [79] (otCMA) + `KPW15` [79] (CMA)	Sig_1^*	√	$\mathcal{D}_k$-MDDH $(\mathbb{G}_1, \mathbb{G}_2)$	$\lvert m\rvert$ $\lvert pk\rvert + \lvert par\rvert$ $\lvert\sigma\rvert$ ♯ PPE	$(n_1^2, 0)$ $\left((n_1k + 2k^2 + 4k + \mathsf{RE}(\mathcal{D}_k) + 1)k + \mathsf{RE}(\mathcal{D}_k), 0\right)$ $\left(3n_1k + 3n_1, (n_1 + 2k + 4)k + n_1 + k + 1 + \mathsf{RE}(\mathcal{D}_k)\right)$ $(2k + 1)n_1 + k$
`(M)`: `KPW15` [79] (otRMA) + `KPW15` [79] (CMA)	Sig_1^*	√	$\mathcal{D}_k$-MDDH $(\mathbb{G}_1, \mathbb{G}_2)$	$\lvert m\rvert$ $\lvert pk\rvert + \lvert par\rvert$ $\lvert\sigma\rvert$ ♯ PPE	$(n_1^2, 0)$ $\left((n_1k + 2k^2 + 4k + \mathsf{RE}(\mathcal{D}_k) + 1)k + \mathsf{RE}(\mathcal{D}_k), 0\right)$ $\left(3n_1k + 3n_1, (n_1 + 2k + 4)k + n_1 + k + \mathsf{RE}(\mathcal{D}_k)\right)$ $(2k + 1)n_1 + k$
`(N)`: `KPW15` [79] (otCMA) + `KPW15` [80] (TT)	Sig_2	√	$\mathcal{D}_k$-MDDH $(\mathbb{G}_1, \mathbb{G}_2)$	$\lvert m\rvert$ $\lvert pk\rvert + \lvert par\rvert$ $\lvert\sigma\rvert$ ♯ PPE	$(n_1^2, 0)$ $\left((2n_1k + 1 + \mathsf{RE}(\mathcal{D}_k))k + \mathsf{RE}(\mathcal{D}_k), 0\right)$ $\left((k + 1)n_1, 2kn_1 + k + 1 + \mathsf{RE}(\mathcal{D}_k)\right)$ $kn_1 + k$
`(O)`: `KPW15` [79] (otRMA) + `KPW15` [80] (TT)	Sig_2	√	$\mathcal{D}_k$-MDDH $(\mathbb{G}_1, \mathbb{G}_2)$	$\lvert m\rvert$ $\lvert pk\rvert + \lvert par\rvert$ $\lvert\sigma\rvert$ ♯ PPE	$(n_1^2, 0)$ $\left((2n_1k + 1 + \mathsf{RE}(\mathcal{D}_k))k + \mathsf{RE}(\mathcal{D}_k), 0\right)$ $\left((k + 1)n_1, 2kn_1 + k + \mathsf{RE}(\mathcal{D}_k)\right)$ $kn_1 + k$

Table 3.2: UF-otCMA secure FSPS schemes constructed from TSs and (TT-)AKSs. Here we use the same notation as in Table 3.1. Scheme `(A)`, `(B)`, ..., `(O)` correspond to those of Table 3.1 and Table 3.2.

[18] We do not specify this for other schemes.

3.7 Efficient Instantiations of FSPS and FAS Based on the $\mathcal{SC}_k$-MDDH Assumptions

In this section, we give four instantiations of our generic constructions Sig_1 and Sig_2 by combining the SPSP-TS schemes in Fig. 3.1 and Fig. 3.2 with the AKS schemes in Fig. 3.3 and Fig. 3.4. Although these instantiations are based on the $\mathcal{D}_k$-MDDH ($\mathcal{U}_k$-MDDH) assumptions, for simplicity, we let them be based on the $\mathcal{SC}_k$-MDDH assumptions here, and use $SC_k(a)$ to denote a $(k+1) \times k$ matrix $\begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ a & 1 & 0 & \dots & 0 \\ 0 & a & 1 & \dots & 0 \\ 0 & 0 & a & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & a \end{pmatrix}$.

3.7.1 Instantiation: UF-CMA Secure SKSP-TS + UF-otCMA Secure SP-AKS

In Fig. 3.9, we give an instantiation of UF-CMA secure FSPS scheme based on the $\mathcal{SC}_k$-MDDH assumptions in $\mathbb{G}_1$ and $\mathbb{G}_2$. This instantiation is derived from Sig_1 (see Fig. 3.5). The underlying TS scheme is the one in Fig. 3.1 and the underlying AKS scheme is the one in Fig. 3.3. The efficiency of this instantiation is $(|m|, |pk| + |par|, |\sigma|, \sharp\text{PPE}) = (n^2, (n^2k + 3k + 3 + \mathsf{RE}(\mathcal{SC}_k))k + \mathsf{RE}(\mathcal{SC}_k), n^2k + 5 + 5k + \mathsf{RE}(\mathcal{SC}_k), 3k + 1)$ where $\mathsf{RE}(\mathcal{SC}_k) = 1$. Note that $\widetilde{\mathbf{C}}$ denotes a vector consisting of all the elements in $\mathbf{C}$ as noted in Section 2.1, and we use the underlying TS scheme to sign $(\widetilde{\mathbf{C}}, a')$ instead of $(\mathbf{C}, a')$, which does not affect the correctness, security, and the structure of this instantiation. We use the same argument for constructions in Fig. 3.10, Fig. 3.11, and Fig. 3.12.

3.7.2 Instantiation: UF-otRMA Secure SKSP-TS + UF-otCMA Secure SP-AKS

In Fig. 3.10, we give a UF-otCMA secure FSPS scheme based on the $\mathcal{SC}_k$-MDDH assumptions in $\mathbb{G}_1$ and $\mathbb{G}_2$. This instantiation is derived from Sig_1 (see Fig. 3.5). The underlying TS scheme is the one in Fig. 3.2 and the underlying AKS scheme is the one in Fig. 3.3. The efficiency of this instantiation is $(|m|, |pk|+|par|, |\sigma|, \sharp\text{PPE}) = (n^2, (n^2k+k+1+\mathsf{RE}(\mathcal{SC}_k))k+ \mathsf{RE}(\mathcal{SC}_k), n^2k + 1 + 3k + \mathsf{RE}(\mathcal{SC}_k), 2k)$ where $\mathsf{RE}(\mathcal{SC}_k) = 1$.

3.7.3 Instantiation: UF-CMA Secure SKSP-TS + UF-TT-CMA Secure SP-TT-AKS

In Fig. 3.11, we give a UF-CMA secure FSPS scheme based on the $\mathcal{SC}_k$-MDDH assumptions in $\mathbb{G}_1$ and $\mathbb{G}_2$. This instantiation is derived from Sig_2 (see Fig. 3.7). The underlying TS

$\widehat{\mathsf{Setup}}(1^\lambda)$:
$par = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, [1]_1, [1]_2) \leftarrow \mathcal{G}(1^\lambda)$,
For preliminary-fixed $n \in \mathbb{N}$,
  determine the message space $\mathbb{G}_1^n$.
Return par.

$\widehat{\mathsf{Gen}}(par)$:
$a, b \leftarrow \mathbb{Z}_p$, $\mathbf{A} = SC_k(a)$, $\mathbf{B} = SC_k(b)$,
$\mathbf{K} \leftarrow \mathbb{Z}_p^{((n+1)k+2)\times(k+1)}$,
$\mathbf{K_0}, \mathbf{K_1} \leftarrow \mathbb{Z}_p^{(k+1)\times(k+1)}$.
$\mathbf{C} = \mathbf{KA} \in \mathbb{Z}_p^{((n+1)k+2)\times k}$,
$(\mathbf{C}_0, \mathbf{C}_1) = (\mathbf{K}_0\mathbf{A}, \mathbf{K}_1\mathbf{A}) \in (\mathbb{Z}_p^{(k+1)\times k})^2$,
$(\mathbf{P}_0, \mathbf{P}_1) = (\mathbf{B}^\top\mathbf{K}_0, \mathbf{B}^\top\mathbf{K}_1) \in (\mathbb{Z}_p^{k\times(k+1)})^2$.
$pk = ([\mathbf{C}_0]_1, [\mathbf{C}_1]_1, [\mathbf{C}]_1, [a]_1)$,
$sk = ([\mathbf{K}]_2, [\mathbf{P}_0]_2, [\mathbf{P}_1]_2, [b]_2)$.
Return (pk, sk).

$\widehat{\mathsf{VerifySK}}(pk, sk)$:
$\mathbf{A} = SC_k(a)$, $\mathbf{B} = SC_k(b)$,
Return 1 if
  $e([\mathbf{A}]_1^\top, [\mathbf{K}]_2^\top) = e([\mathbf{C}]_1^\top, [1]_2)$,
  $e([\mathbf{A}]_1^\top, [\mathbf{P}_0]_2^\top) = e([\mathbf{C}_0]_1^\top, [\mathbf{B}^\top]_2^\top)$,
  and $e([\mathbf{A}]_1^\top, [\mathbf{P}_1]_2^\top) = e([\mathbf{C}_1]_1^\top, [\mathbf{B}]_2)$.
Return 0 otherwise.

$\widehat{\mathsf{Sign}}(sk, [\vec{m}]_1)$:
Parse $sk = ([\mathbf{K}]_2, [\mathbf{P}_0]_2, [\mathbf{P}_1]_2, [b]_2)$.
$a' \leftarrow \mathbb{Z}_p$, $\mathbf{A'} = SC_k(a')$, $\mathbf{B} = SC_k(b)$,
$\mathbf{K'} \leftarrow \mathbb{Z}_p^{(n+1)\times(k+1)}$, $\mathbf{C'} = \mathbf{K'A'} \in \mathbb{Z}_p^{(n+1)\times k}$,
$\vec{r} \leftarrow \mathbb{Z}_p^k$, $\tau \leftarrow \mathbb{Z}_p$.
$\sigma_{11} = [(1, \widetilde{\mathbf{C}}'^\top, a')\mathbf{K} + \vec{r}^\top(\mathbf{P}_0 + \tau\mathbf{P}_1)]_2 \in \mathbb{G}_2^{1\times(k+1)}$,
$\sigma_{12} = ([\vec{r}^\top\mathbf{B}^\top]_2) \in \mathbb{G}_2^{1\times(k+1)}$, $\sigma_{13} = ([\vec{r}^\top\mathbf{B}^\top\tau]_2) \in \mathbb{G}_2^{1\times(k+1)}$,
$\sigma_{14} = [\tau]_1 \in \mathbb{G}_1$,
$\sigma_2 = ([\mathbf{C'}]_2, [a']_2) \in \mathbb{G}_2^{(n+1)\times k} \times \mathbb{G}_2$,
$\sigma_3 = [(1, \vec{m}^\top)\mathbf{K'}]_1 \in \mathbb{G}_1^{1\times(k+1)}$,
Return $\sigma = ((\sigma_{11}, \sigma_{12}, \sigma_{13}, \sigma_{14}), \sigma_2, \sigma_3)$.

$\widehat{\mathsf{Verify}}(pk, [\vec{m}]_1, \sigma)$:
Parse $pk = ([\mathbf{C}_0]_1, [\mathbf{C}_1]_1, [\mathbf{C}]_1, [a]_1)$,
  $\sigma = ((\sigma_{11}, \sigma_{12}, \sigma_{13}, \sigma_{14}), \sigma_2, \sigma_3)$,
  and $\sigma_2 = ([\mathbf{C'}]_2, [a']_2) \in \mathbb{G}_2^{(n+1)\times k} \times \mathbb{G}_2$.
$[\mathbf{A}]_1 = SC_k([a]_1)$, $[\mathbf{A'}]_2 = SC_k([a']_2)$.
Return 1 if
$e([\mathbf{A}]_1^\top, \sigma_{11}^\top) = e([\mathbf{C}]_1^\top, [(1, \widetilde{\mathbf{C}}', a')]_2^\top) + e([\mathbf{C}_0]_1^\top, \sigma_{12}^\top) + e([\mathbf{C}_1]_1^\top, \sigma_{13}^\top)$,
$e(\sigma_{14}, \sigma_{12}^\top) = e([1]_1, \sigma_{13}^\top)$,
and $e(\sigma_3, [\mathbf{A'}]_2) = e([1, \vec{m}^\top]_1, [\mathbf{C'}]_2)$.
Return 0 otherwise.

Figure 3.9: UF-CMA secure FSPS scheme (TS (UF-CMA) + AKS (UF-otCMA)).

$\widehat{\mathsf{Setup}}(1^\lambda)$:
$par = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, [1]_1, [1]_2) \leftarrow \mathcal{G}(1^\lambda)$,
For preliminary-fixed $n \in \mathbb{N}$,
  determine the message space $\mathbb{G}_1^n$.
Return par.

$\widehat{\mathsf{Gen}}(par)$:
$a \leftarrow \mathbb{Z}_p$, $\mathbf{K} \leftarrow \mathbb{Z}_p^{((n+1)k+2)\times k}$,
$\overline{\mathbf{A}} = \overline{SC_k(a)}$,
$\mathbf{C} = \mathbf{K}\overline{\mathbf{A}} \in \mathbb{Z}_p^{((n+1)k+2)\times k}$,
$pk = ([\mathbf{C}]_1, [a]_1)$,
$sk = [\mathbf{K}]_2$.
Return (pk, sk).

$\widehat{\mathsf{VerifySK}}(pk, sk)$:
Return 1 if $e([\overline{\mathbf{A}}]_1^\top, [\mathbf{K}]_2^\top) = e([\mathbf{C}]_1^\top, [1]_2)$.
Return 0 otherwise.

$\widehat{\mathsf{Sign}}(sk, [\vec{m}]_1)$:
$a' \leftarrow \mathbb{Z}_p$, $\mathbf{A'} = SC_k(a')$,
$\mathbf{K'} \leftarrow \mathbb{Z}_p^{(n+1)\times(k+1)}$, $\mathbf{C'} = \mathbf{K'A'} \in \mathbb{Z}_p^{(n+1)\times k}$.
$\sigma_1 = [(1, \widetilde{\mathbf{C}}', a')\mathbf{K}]_2 \in \mathbb{G}_2^{1\times k}$,
$\sigma_2 = ([\mathbf{C'}]_2, [a']_2) \in \mathbb{G}_2^{(n+1)\times k} \times \mathbb{G}_2$,
$\sigma_3 = [(1, \vec{m}^\top)\mathbf{K'}]_1 \in \mathbb{G}_1^{1\times(k+1)}$,
Return $\sigma = (\sigma_1, \sigma_2, \sigma_3)$.

$\widehat{\mathsf{Verify}}(pk, [\vec{m}]_1, \sigma)$:
Parse $pk = ([\mathbf{C}]_1, [a]_1)$, $\sigma = ((\sigma_1, \sigma_2, \sigma_3)$,
  and $\sigma_2 = ([\mathbf{C'}]_2, [a']_2) \in \mathbb{G}_2^{(n+1)\times k} \times \mathbb{G}_2$.
$[\overline{\mathbf{A}}]_1 = \overline{SC_k([a]_1)}$, $[\mathbf{A'}]_2 = SC_k([a']_2)$.
Return 1 if $e([\overline{\mathbf{A}}]_1^\top, \sigma_1^\top) = e([\mathbf{C}]_1^\top, [(1, \widetilde{\mathbf{C}}', a')]_2^\top)$
  and $e(\sigma_3, [\mathbf{A'}]_2) = e([1, \vec{m}^\top]_1, [\mathbf{C'}]_2)$.
Return 0 otherwise.

Figure 3.10: UF-otCMA secure FSPS scheme (TS (UF-otRMA) + AKS (UF-otCMA)).

scheme is the one in Fig. 3.1 and the underlying TT-AKS scheme is the one in Fig. 3.4. The efficiency of this instantiation is $(|m|, |pk| + |par|, |\sigma|, \sharp\mathrm{PPE}) = (n^2, (2nk + 2k + 3 + \mathsf{RE}(\mathcal{SC}_k))k + \mathsf{RE}(\mathcal{SC}_k), (3k+1)n + 4 + 3k + \mathsf{RE}(\mathcal{SC}_k), kn + 2k + 1)$ where $\mathsf{RE}(\mathcal{SC}_k) = 1$.

$\widehat{\mathsf{Setup}}(1^\lambda)$:
$par = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, [1]_1, [1]_2) \leftarrow \mathcal{G}(1^\lambda)$,
For preliminary-fixed $n \in \mathbb{N}$,
determine the message space $\mathbb{G}_1^{n\times n}$.
Return par.

$\widehat{\mathsf{Gen}}(par)$:
$a, b \leftarrow \mathbb{Z}_p$, $\mathbf{A} = SC_k(a)$, $\mathbf{B} = SC_k(b)$,
$\mathbf{K} \leftarrow \mathbb{Z}_p^{(2nk+2)\times(k+1)}$.
$\mathbf{K_0}, \mathbf{K_1} \leftarrow \mathbb{Z}_p^{(k+1)\times(k+1)}$.
$\mathbf{C} = \mathbf{KA} \in \mathbb{Z}_p^{(2nk+2)\times k}$,
$(\mathbf{C}_0, \mathbf{C}_1) = (\mathbf{K}_0\mathbf{A}, \mathbf{K}_1\mathbf{A}) \in (\mathbb{Z}_p^{(k+1)\times k})^2$,
$(\mathbf{P}_0, \mathbf{P}_1) = (\mathbf{B}^\top\mathbf{K}_0, \mathbf{B}^\top\mathbf{K}_1) \in (\mathbb{Z}_p^{k\times(k+1)})^2$.
$pk = ([\mathbf{C}_0]_1, [\mathbf{C}_1]_1, [\mathbf{C}]_1, [a]_1)$,
$sk = ([\mathbf{K}]_2, [\mathbf{P}_0]_2, [\mathbf{P}_1]_2, [b]_2)$.
Return (pk, sk).

$\widehat{\mathsf{VerifySK}}(pk, sk)$:
$\mathbf{A} = SC_k(a)$, $\mathbf{B} = SC_k(b)$,
Return 1 if
$e([\mathbf{A}]_1^\top, [\mathbf{K}]_2^\top) = e([\mathbf{C}]_1^\top, [1]_2)$,
$e([\mathbf{A}]_1^\top, [\mathbf{P}_0]_2^\top) = e([\mathbf{C}_0]_1^\top, [\mathbf{B}]_2)$,
and $e([\mathbf{A}]_1^\top, [\mathbf{P}_1^\top]_2) = e([\mathbf{C}_1]_1^\top, [\mathbf{B}]_2)$.
Return 0 otherwise.

$\widehat{\mathsf{Sign}}(sk, [\mathbf{M}]_1)$:
Parse $[\mathbf{M}]_1 = ([\vec{m}_1]_1, \ldots, [\vec{m}_n]_1)$
and $sk = ([\mathbf{K}]_2, [\mathbf{P}_0]_2, [\mathbf{P}_1]_2, [b]_2)$.
$a' \leftarrow \mathbb{Z}_p$, $\mathbf{A}' = SC_k(a')$, $\mathbf{B} = SC_k(b)$,
$\mathbf{K}' \leftarrow \mathbb{Z}_p^{n\times(k+1)}$, $\vec{k}_1 \leftarrow \mathbb{Z}_p^{k+1}, \ldots, \vec{k}_n \leftarrow \mathbb{Z}_p^{k+1}$,

$$\mathbf{K}_1 = \begin{pmatrix} \vec{k}_1^\top \\ \mathbf{K}' \end{pmatrix}, \ldots, \mathbf{K}_n = \begin{pmatrix} \vec{k}_n^\top \\ \mathbf{K}' \end{pmatrix}, \mathbf{D} = \begin{pmatrix} \mathbf{K}'\mathbf{A}' \\ \vec{k}_1^\top\mathbf{A}' \\ \vdots \\ \vec{k}_n^\top\mathbf{A}' \end{pmatrix} \in \mathbb{Z}_p^{2n\times k},$$

$\vec{r} \leftarrow \mathbb{Z}_p^k$, $\tau \leftarrow \mathbb{Z}_p$.
$\sigma_{11} = [(1, \widetilde{\mathbf{D}}^\top, a')\mathbf{K} + \vec{r}^\top(\mathbf{P}_0 + \tau\mathbf{P}_1)]_2 \in \mathbb{G}_2^{1\times(k+1)}$,
$\sigma_{12} = [\vec{r}^\top\mathbf{B}^\top]_2 \in \mathbb{G}_2^{1\times(k+1)}$, $\sigma_{13} = [\vec{r}^\top\mathbf{B}^\top\tau]_2 \in \mathbb{G}_2^{1\times(k+1)}$,
$\sigma_{14} = [\tau]_1 \in \mathbb{G}_1$,
$\sigma_2 = ([\mathbf{D}]_2, [a']_2) \in \mathbb{G}_2^{2n\times k} \times \mathbb{G}_2$,
$\sigma_{3i} = [(1, m_i^\top)\mathbf{K}_i]_1 \in \mathbb{G}_1^{1\times(k+1)}$ for all i.
Return $\sigma = ((\sigma_{11}, \sigma_{12}, \sigma_{13}, \sigma_{14}), \sigma_2, (\sigma_{31}, \ldots, \sigma_{3n}))$.

$\widehat{\mathsf{Verify}}(pk, [\mathbf{M}]_1, \sigma)$:
Parse $pk = ([\mathbf{C}_0]_1, [\mathbf{C}_1]_1, [\mathbf{C}]_1, [a]_1)$,
$[\mathbf{M}]_1 = ([\vec{m}_1]_1, \ldots, [\vec{m}_n]_1)$,
and $\sigma = ((\sigma_{11}, \sigma_{12}, \sigma_{13}, \sigma_{14}), \sigma_2, (\sigma_{31}, \ldots, \sigma_{3n}))$.
Parse $\sigma_2 = [\mathbf{D}]_2 = (\begin{bmatrix} \mathbf{C}' \\ \vec{c}_1 \\ \vdots \\ \vec{c}_n \end{bmatrix}_2, [a']_2) \in \mathbb{G}_2^{2n\times k} \times \mathbb{G}_2$.
Let $[\mathbf{C}_i]_2 = \begin{bmatrix} \vec{c}_i \\ \mathbf{C}' \end{bmatrix}_2$ for $i = 1, \ldots, n$,
$[\mathbf{A}]_1 = SC_k([a]_1)$, and $[\mathbf{A}']_2 = SC_k([a']_2)$.
Return 1 if
$e([\mathbf{A}]_1^\top, \sigma_{11}^\top) = e([\mathbf{C}]_1^\top, [1, \widetilde{\mathbf{D}}, a']_2^\top) + e([\mathbf{C}_0^\top]_1, \sigma_{12}^\top) + e([\mathbf{C}_1]_1^\top, \sigma_{13}^\top)$,
$e(\sigma_{14}, \sigma_{12}^\top) = e([1]_1, \sigma_{13}^\top)$,
and $e(\sigma_{3i}, [\mathbf{A}']_2) = e([1, \vec{m}_i^\top]_1, [\mathbf{C}_i]_2)$ for $i = 1, \ldots, n$.
Return 0 otherwise.

Figure 3.11: UF-CMA secure FAS scheme (TS (UF-CMA) + AKS (UF-TT-CMA)).

3.7.4 Instantiation: UF-otRMA Secure SKSP-TS + UF-TT-CMA Secure SP-TT-AKS

In Fig. 3.12, we give a UF-otCMA secure FSPS scheme based on the $\mathcal{SC}_k$-MDDH assumptions in $\mathbb{G}_1$ and $\mathbb{G}_2$. This instantiation is derived from Sig_2 (see Fig. 3.7). The underlying TS scheme is the one in Fig. 3.2 and the underlying TT-AKS scheme is the one in Fig. 3.4. The efficiency of this instantiation is $(|m|, |pk| + |par|, |\sigma|, \sharp\mathrm{PPE}) = (n^2, (2nk + 1 + \mathsf{RE}(\mathcal{SC}_k))k + \mathsf{RE}(\mathcal{SC}_k), (3k+1)n + k + \mathsf{RE}(\mathcal{SC}_k), kn + k)$ where $\mathsf{RE}(\mathcal{SC}_k) = 1$.

$\widehat{\mathsf{Setup}}(1^\lambda)$:
$par = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, [1]_1, [1]_2) \leftarrow \mathcal{G}(1^\lambda)$,
For preliminary-fixed $n \in \mathbb{N}$,
determine the message space $\mathbb{G}_1^{n\times n}$.
Return par.

$\widehat{\mathsf{Gen}}(par)$:
$a \leftarrow \mathbb{Z}_p$, $\mathbf{K} \leftarrow \mathbb{Z}_p^{(2nk+2)\times k}$,
$\overline{\mathbf{A}} = \overline{SC_k(a)}$,
$\mathbf{C} = \mathbf{K}\overline{\mathbf{A}} \in \mathbb{Z}_p^{(2nk+2)\times k}$.
$pk = ([\mathbf{C}]_1, [a]_1)$,
$sk = ([\mathbf{K}]_2)$.
Return (pk, sk).

$\widehat{\mathsf{VerifySK}}(pk, sk)$:
$\overline{\mathbf{A}} = \overline{SC_k(a)}$.
Return 1 if $e([\overline{\mathbf{A}}]_1^\top, [\mathbf{K}]_2^\top) = e([\mathbf{C}]_1^\top, [1]_2)$.
Return 0 otherwise.

$\widehat{\mathsf{Sign}}(sk, [\mathbf{M}]_\mathbf{1})$:
Parse $[\mathbf{M}]_1 = ([\vec{m}_1]_1, \ldots, [\vec{m}_n]_1)$.
$a' \leftarrow \mathbb{Z}_p$, $\mathbf{A}' = SC_k(a')$, $\mathbf{K}' \leftarrow \mathbb{Z}_p^{n\times(k+1)}$,
$\vec{k}_1 \leftarrow \mathbb{Z}_p^{k+1}, \ldots, \vec{k}_n \leftarrow \mathbb{Z}_p^{k+1}$.

$$\mathbf{K}_1 = \begin{pmatrix} \vec{k}_1^\top \\ \mathbf{K}' \end{pmatrix}, \ldots, \mathbf{K}_n = \begin{pmatrix} \vec{k}_n^\top \\ \mathbf{K}' \end{pmatrix}, \mathbf{D} = \begin{pmatrix} \mathbf{K}'\mathbf{A}' \\ \vec{k}_1^\top\mathbf{A}' \\ \vdots \\ \vec{k}_n^\top\mathbf{A}' \end{pmatrix} \in \mathbb{Z}_p^{2n\times k},$$

$\sigma_1 = [(1, \widetilde{\mathbf{D}}, a')\mathbf{K}]_2 \in \mathbb{G}_2^{1\times k}$,
$\sigma_2 = ([\mathbf{D}]_2, [a']_2) \in \mathbb{G}_2^{2n\times k} \times \mathbb{G}_2$,
$\sigma_{3i} = [(1, \vec{m}_i^\top)\mathbf{K}_i]_1 \in \mathbb{G}_1^{1\times(k+1)}$ for all i.
Return $\sigma = (\sigma_1, \sigma_2, (\sigma_{31}, \ldots, \sigma_{3n}))$.

$\widehat{\mathsf{Verify}}(pk, [\mathbf{M}]_1, \sigma)$:
Parse $pk = ([\mathbf{C}]_1, [a]_1)$, $[\mathbf{M}]_1 = ([\vec{m}_1]_1, \ldots, [\vec{m}_n]_1)$,
and $\sigma = (\sigma_1, \sigma_2, (\sigma_{31}, \ldots, \sigma_{3n}))$.

$$\text{Parse } \sigma_2 = [\mathbf{D}]_2 = \left(\begin{bmatrix} \mathbf{C}' \\ \vec{c}_1 \\ \vdots \\ \vec{c}_n \end{bmatrix}_2, [a']_2 \right) \in \mathbb{G}_2^{2n\times k} \times \mathbb{G}_2.$$

$$\text{Let } [\mathbf{C}_i]_2 = \begin{bmatrix} \vec{c}_i \\ \mathbf{C}' \end{bmatrix}_2 \text{ for } i = 1, \ldots, n,$$

$[\mathbf{A}]_1 = SC_k([a]_1)$, and $[\mathbf{A}']_2 = SC_k([a']_2)$.
Return 1 if $e([\overline{\mathbf{A}}]_1^\top, \sigma_1^\top) = e([\mathbf{C}]_1^\top, [1, \widetilde{\mathbf{D}}, a']_2^\top)$,
and $e(\sigma_{3i}, [\mathbf{A}']_2) = e([1, \vec{m}_i^\top]_1, [\mathbf{C}_i]_2)$ for $i = 1, \ldots, n$.
Return 0 otherwise.

Figure 3.12: UF-otCMA secure FAS scheme (TS (UF-otRMA) + AKS (UF-TT-CMA)).

3.8 Signing Key Sizes

In Table 3.3, we give the the signing key sizes for all the instantiations in Table 3.1 and Table 3.2, and also for the instantiations in [10] and [69].

	$\lvert m\rvert$	Assumption	$\lvert sk\rvert$
AKO+15 [10]	$(n_1^2, 0)$	SXDH, XDLIN	$(4, 0)$
AKO+15 [10]	$(n_1^2, 0)$	SXDH, XDLIN	$(0, 4)$
Gro15 [69]	$(n_1^2, 0)$	Generic	$(2n_1 + 1, 0)$
(A)	$(n_1^2, 0)$	$\mathcal{D}_k$-MDDH$(\mathbb{G}_1, \mathbb{G}_2)$	$\Big(0, ((n_1^2+1)k + \mathsf{RE}(\mathcal{D}_k) + 1)(k+1) + 2(k+1)k + \mathsf{RE}(\mathcal{D}_k)\Big)$
(B)	n^2	2-Lin$(\mathbb{G}_1 = \mathbb{G}_2)$	$6n^2 + 65$
(C)	$(n_1^2, 0)$	Generic	$(0, n_1)$
(D)	$(n_1^2, 0)$	$\mathcal{D}_k$-MDDH$(\mathbb{G}_1, \mathbb{G}_2)$	$\Big(0, ((n_1 + 2k + 4)k + \mathsf{RE}(\mathcal{D}_k) + 1)(k+1) + 2(k+1)k + \mathsf{RE}(\mathcal{D}_k)\Big)$
(E)	n^2	2-Lin$(\mathbb{G}_1 = \mathbb{G}_2)$	$6n + 71$
(F)	$(n_1^2, 0)$	$\mathcal{D}_k$-MDDH$(\mathbb{G}_1, \mathbb{G}_2)$	$\Big(0, (2n_1k + \mathsf{RE}(\mathcal{D}_k) + 1)(k+1) + 2(k+1)k + \mathsf{RE}(\mathcal{D}_k)\Big)$
(G)	(n_1^2, n_2^2)	$\mathcal{D}_k$-MDDH$(\mathbb{G}_1, \mathbb{G}_2)$	$\Big((2n_2k + \mathsf{RE}(\mathcal{D}_k))(k+1), (2n_1k + \mathsf{RE}(\mathcal{D}_k) + k + 1)(k+1) + 2(k+1)k + \mathsf{RE}(\mathcal{D}_k)\Big)$
(H)	n^2	2-Lin$(\mathbb{G}_1 = \mathbb{G}_2)$	$9n + 20$
(I)	$(n_1^2, 0)$	SXDH, XDLIN	$(0, 4n_1 + 6)$
(J)	$(n_1^2, 0)$	$\mathcal{D}_k$-MDDH	$\Big(0, ((n_1^2+1)k + \mathsf{RE}(\mathcal{D}_k) + 1)(k+1)\Big)$
(K)	$(n_1^2, 0)$	$\mathcal{D}_k$-MDDH	$\Big(0, ((n_1^2+1)k + \mathsf{RE}(\mathcal{D}_k) + 1)k\Big)$
(L)	$(n_1^2, 0)$	$\mathcal{D}_k$-MDDH	$\Big(0, ((n_1^2 + 2k + 4)k + \mathsf{RE}(\mathcal{D}_k) + 1)(k+1)\Big)$
(M)	$(n_1^2, 0)$	$\mathcal{D}_k$-MDDH	$\Big(0, ((n_1^2 + 2k + 4)k + \mathsf{RE}(\mathcal{D}_k) + 1)k\Big)$
(N)	$(n_1^2, 0)$	$\mathcal{D}_k$-MDDH	$\Big(0, (2n_1k + \mathsf{RE}(\mathcal{D}_k) + 1)(k+1)\Big)$
(O)	$(n_1^2, 0)$	$\mathcal{D}_k$-MDDH	$\Big(0, (2n_1k + \mathsf{RE}(\mathcal{D}_k) + 1)k\Big)$

Table 3.3: Signing key sizes

3.9 Number of Pairings

In [64], Ghadafi argued that the number of pairings required in verification is as important as the number of PPEs. The reason is that when combining an SPS scheme with the Groth-Sahai proof system, the number of pairings required for Groth-Sahai proofs grows linearly with that required in verification. In Table 3.4, we give this parameter for all the FSPS schemes in Table 3.1 and Table 3.2.

Scheme	$\|m\|$	Assumption	♯ Pairing
AKO+15 [10]	$(n_1^2, 0)$	SXDH, XDLIN	$n_1^2 + 5n_1 + 15$
AKO+15 [10]	$(n_1^2, 0)$	SXDH, XDLIN	$5n_1^2 + 19$
Gro15 [69]	$(n_1^2, 0)$	Generic	$n_1^2 + 3n_1 + 2$
(A)	$(n_1^2, 0)$	$\mathcal{D}_k$-MDDH SXDH	$((n_1^2+1)k + \mathsf{RE}(\mathcal{D}_k) + 8 + n_1^2)k + 4k^2 + 2$ $2n_1^2 + 16$
(B)	n^2	2-Lin	$6n^2 + 84$
(C)	$(n_1^2, 0)$	Generic	$n_1^2 + 3n_1 + 8$
(D)	$(n_1^2, 0)$	$\mathcal{D}_k$-MDDH SXDH	$(3k^2 + (n_1+7)k + 2)n_1 + ((n_1 + 2k + 4)k + \mathsf{RE}(\mathcal{D}_k) + 6)k + 2 + 3k^2)$ $n_1^2 + 13n_1 + 18$
(E)	n^2	2-Lin	$24n^2 + 6n + 66$
(F)	$(n_1^2, 0)$	$\mathcal{D}_k$-MDDH SXDH	$(k^2 + (n_1+2)k)n_1 + 3k^2 + ((2n_1k + \mathsf{RE}(\mathcal{D}_k)) + 6)k + 2$ $n_1^2 + 5n_1 + 12$
(G)	(n_1^2, n_2^2)	$\mathcal{D}_k$-MDDH SXDH	$(k^2 + (n_1+2)k)n_1 + (k^2 + (n_2+2)k)n_2 + 5k^2 + (2(n_1+n_2)k + 2\mathsf{RE}(\mathcal{D}_k) + 8)k + 2$ $n_1^2 + n_2^2 + 5(n_1+n_2) + 17$
(H)	n^2	2-Lin	$2n^2 + 11n + 32$
(I)	$(n_1^2, 0)$	SXDH, XDLIN	$n_1^2 + 5n_1 + 16$
(J)	$(n_1^2, 0)$	$\mathcal{D}_k$-MDDH SXDH	$k^2 + (n_1^2+2)k + ((n_1^2+1)k + \mathsf{RE}(\mathcal{D}_k) + 2)k + k^2$ $2n_1^2 + 8$
(K)	$(n_1^2, 0)$	$\mathcal{D}_k$-MDDH SXDH	$k^2 + (n_1^2+2)k + ((n_1^2+1)k + \mathsf{RE}(\mathcal{D}_k) + 1)k + k^2$ $2n_1^2 + 7$
(L)	$(n_1^2, 0)$	$\mathcal{D}_k$-MDDH SXDH	$(3k^2 + (n_1+7)k + 2)n_1 + ((n_1 + 2k + 4)k + \mathsf{RE}(\mathcal{D}_k) + 2)k + k^2$ $n_1^2 + 13n_1 + 10$
(M)	$(n_1^2, 0)$	$\mathcal{D}_k$-MDDH SXDH	$(3k^2 + (n_1+7)k + 2)n_1 + ((n_1 + 2k + 4)k + \mathsf{RE}(\mathcal{D}_k) + 1)k + k^2$ $n_1^2 + 13n_1 + 9$
(N)	$(n_1^2, 0)$	$\mathcal{D}_k$-MDDH SXDH	$(k^2 + (n_1+2)k)n_1 + (2n_1k + \mathsf{RE}(\mathcal{D}_k) + 2)k + k^2$ $n_1^2 + 5n_1 + 4$
(O)	$(n_1^2, 0)$	$\mathcal{D}_k$-MDDH SXDH	$(k^2 + (n_1+2)k)n_1 + (2n_1k + \mathsf{RE}(\mathcal{D}_k) + 1)k + k^2$ $n_1^2 + 5n_1 + 3$

Table 3.4: Numbers of pairings required in verification.

Chapter 4

Signature Resilient to Uninvertible Leakage

In this chapter, we first give definitions of FLR signature in the selective auxiliary model. Then we define and instantiate two new LR primitives called ULR-hard relation and IULR-hard relation, which will be used as building blocks in our proposed signature schemes, and show how to construct the former (respectively, the latter) from AIPO (respectively, iO). Finally, we propose an FLR signature scheme against uninvertible leakage (respectively, injective uninvertible leakage) based on a ULR-hard relation (respectively, an IULR-hard relation) and diO.

4.1 Fully Leakage Resilient Signature in the Selective Auxiliary Input Model

Now we give the definition of *FLR signature in the selective auxiliary input model*. In this model, we allow an adversary to learn any uninvertible leakage, which is selective (i.e., independent of the verification key), on the signing key, and learn all the randomizers used in the signing procedure. Furthermore, we let the signing oracle return (σ, r) when answering an adaptive signing query m, where σ is a signature on m and r is the randomizer used to generate σ. Since r is public information, the secret state for the signature scheme only contains the signing key, which means that a signature scheme satisfying this security is FLR. We do not consider leakage during the key generation procedure since the verification/signing key pair can be generated "off-line" [34].

Definition 4.1.1 (UF-CMA security in the selective auxiliary input model)**.** *Let $\mathcal{F}$ denote a polynomial-time computable function family.*[1] *A signature scheme* $(\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *is said to be* UF-CMA secure in the selective auxiliary input model *w.r.t. $\mathcal{F}$ if for any PPT*

[1] In this thesis, when we say functions, we mean the descriptions of them, which are of the form of circuits.

adversary $\mathcal{A}$ *and any* $f \in \mathcal{F}$*, we have*

$$\Pr[par \leftarrow \mathsf{Setup}(1^\lambda), (pk, sk) \leftarrow \mathsf{Gen}(par), (m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathrm{SignO}(\cdot)}(par, pk, f(sk)) : m^* \notin \mathcal{Q}_m \wedge \mathsf{Verify}(pk, m^*, \sigma^*) = 1] \leq negl(\lambda),$$

where $\mathrm{SignO}(\cdot)$ *is the signing oracle that takes* m *as input, runs* $\sigma = \mathsf{Sign}(sk, m; r)$*, adds* m *to* $\mathcal{Q}_m$ *(initialized with* $\emptyset$*), and returns* (σ, r)*.*

Now we give the definition of FLR signature and a variant, called weak FLR signature, in the selective auxiliary input model. For an FLR signature scheme, leakage functions are allowed to be any computable uninvertible function, while for a weak FLR one, they are additionally required to be injective and the sizes of them are upper bounded.

Definition 4.1.2 (FLR in the selective auxiliary input model)**.** *A signature scheme is said to be* FLR in the selective auxiliary input model *if it is correct and UF-CMA secure in the selective auxiliary input model w.r.t.* $\mathcal{F}_{uf}$*, where* $\mathcal{F}_{uf}$ *denotes the family of all the (polynomial-time computable) uninvertible functions.*

Definition 4.1.3 (Weak FLR in the selective auxiliary input model)**.** *Let* $k = k(\lambda)$ *be a polynomial. A signature scheme is said to be* k-weak FLR in the selective auxiliary input model *if it is correct and UF-CMA secure in the selective auxiliary input model w.r.t.* $\mathcal{F}_{k-iuf}$*, where* $\mathcal{F}_{k-iuf}$ *denotes the family of all (polynomial-time computable)* injective *uninvertible functions whose sizes are less than or equal to* k*.*

4.2 Uninvertible Leakage Resilient Hard Relation

We define two new primitives called a *ULR-hard relation* and an *IULR-hard relation* in Section 4.2.1, and give the constructions of them in Section 4.2.2. They will be used as building blocks to achieve our proposed signature schemes.

4.2.1 Definitions

Now we give the definition of ULR-hard relation. Roughly speaking, for a randomly chosen public/secret key pair (y, x) satisfying the ULR-hard relation, it is hard for any adversary to find a valid secret key w.r.t. y, even given y and uninvertible leakage on x. The formal definition is as follows.

Definition 4.2.1 (ULR-hard relation)**.** *A* ULR-hard relation *consists of two algorithms* $(\mathsf{Gen}_{\mathsf{HR}}, R_{\mathsf{HR}})$*.*

- $\mathsf{Gen}_{\mathsf{HR}}$ *takes as input* 1^λ *and outputs a public/secret key pair* (y, x)*.*
- R_{HR} *takes as input a public/secret key pair* (y, x) *and outputs either* 1 *("accept") or* 0 *("reject").*

A ULR-hard relation must satisfy correctness *and* security.

Correctness is satisfied if we have $R_{\mathsf{HR}}(y, x) = 1$ *for all security parameters* λ *and all* $(y, x) \leftarrow \mathsf{Gen}_{\mathsf{HR}}(1^\lambda)$.

Let $\mathcal{F}_{uf}$ *denote the family of all the (polynomial-time computable) uninvertible functions. Security is satisfied if for any PPT adversary* $\mathcal{A}$ *and any* $f \in \mathcal{F}_{uf}$*, we have*

$$\Pr[(y, x) \leftarrow \mathsf{Gen}_{\mathsf{HR}}(1^\lambda), x^* \leftarrow \mathcal{A}(1^\lambda, y, f(x)) : R_{\mathsf{HR}}(y, x^*) = 1] \leq negl(\lambda).$$

Now we give the definition of IULR-hard relation, which is the same as that of a ULR-hard relation, except that leakage functions are required to be injective and the sizes of them are upper bounded.

Definition 4.2.2 (IULR-hard relation)**.** *A pair of algorithms* $(\mathsf{Gen}_{\mathsf{HR}}, R_{\mathsf{HR}})$*, whose syntax is the same as that of a ULR-hard relation, is said to be a* k-IULR-hard relation *if it is correct and secure. Correctness is defined in exactly the same way as that of a ULR-hard relation. Security is also defined in the same way as the that of a ULR-hard relation, except that we replace "*$\mathcal{F}_{uf}$*" with "*$\mathcal{F}_{k-iuf}$*" which denotes the family of all (polynomial-time computable) injective uninvertible functions whose sizes are less than or equal to* k*.*

4.2.2 Constructions

In this section, we give our constructions of ULR-hard relation and IULR-hard relation.

ULR-hard relation based on AIPO. Let $\mathcal{AIPO}$ be AIPO. Then the construction of ULR-hard relation is as follows.

- $\mathsf{Gen}_{\mathsf{HR}}(1^\lambda)$: Randomly select $x \leftarrow \{0,1\}^\lambda$, compute $y \leftarrow \mathcal{AIPO}(x)$, and output (y, x).
- $R_{\mathsf{HR}}(y, x)$: Output $y(x)$.

Theorem 4.2.1. *The above scheme* $(\mathsf{Gen}_{\mathsf{HR}}, R_{\mathsf{HR}})$ *is a ULR-hard relation if* $\mathcal{AIPO}$ *is AIPO.*

The high-level idea of the proof of Theorem 4.2.1 is as follows.

An adversary $\mathcal{A}$ wins the security game if it outputs x^* such that $y(x^*) = 1$, which happens if and only if $x^* = x$ since y is a point function. As a result, the goal of $\mathcal{A}$ is to find x, given 1^λ, y, and $f(x)$. However, according to Lemma 2.5.1, $\mathcal{A}$ cannot find x when seeing only 1^λ and y, and intuitively, seeing $f(x)$ does little to help $\mathcal{A}$ since f is uninvertible. The formal proof is as follows.

Proof of Theorem 4.2.1. According to the definition of AIPO, we have $R_{\mathsf{HR}}(y, x) = 1$ for all $(y, x) \leftarrow \mathsf{Gen}_{\mathsf{HR}}(1^\lambda)$, i.e., $(\mathsf{Gen}_{\mathsf{HR}}, R_{\mathsf{HR}})$ satisfies the correctness property.

Let $\mathcal{A}$ be any PPT adversary and f any computable uninvertible function used as a leakage function. We give hybrid games to show that $\mathcal{A}$ has negligible advantage in breaking the security of $(\mathsf{Gen}_{\mathsf{HR}}, R_{\mathsf{HR}})$.

Game 1: This is the original security game of the ULR-hard relation for $\mathcal{A}$. The challenger samples $(y, x) \leftarrow \mathsf{Gen}_{\mathsf{HR}}(1^\lambda)$ and gives $(1^\lambda, y, f(x))$ to $\mathcal{A}$. $\mathcal{A}$ succeeds if it outputs x^* such that $R_{\mathsf{HR}}(y, x^*) = 1$.

Game 2: This game is the same as **Game 1** except that the challenger generates y as $y \leftarrow \mathcal{AIPO}(r)$ where $r \leftarrow \{0,1\}^\lambda$.

Lemma 4.2.1. *If $\mathcal{A}$ succeeds with probability ϵ_1 in* **Game 1** *and ϵ_2 in* **Game 2**, *then we can construct a sampler* Samp *that samples an unpredictable distribution and an adversary $\mathcal{D}$ that break the security of $\mathcal{AIPO}$ with advantage $|\epsilon_2 - \epsilon_1|$.*

Proof of Lemma 4.2.1. We show how to construct Samp and $\mathcal{D}$ as follows.

On input 1^λ, Samp randomly chooses $x \leftarrow \{0,1\}^\lambda$, and outputs $(f(x), x)$. Since f is an uninvertible function, the distribution of $(f(x), x)$ is an unpredictable distribution.

Taking as input $(1^\lambda, y, f(x))$ where y is generated as $y \leftarrow \mathcal{AIPO}(r)$ or $y \leftarrow \mathcal{AIPO}(x)$, $\mathcal{D}$ runs $x^* \leftarrow \mathcal{A}(1^\lambda, y, f(x))$. If $R_{\mathsf{HR}}(y, x^*) = 1$, $\mathcal{D}$ outputs 1. Otherwise, it outputs 0.

If y is generated as $y \leftarrow \mathcal{AIPO}(r)$, then $\mathcal{A}$ is in **Game 2**, i.e., $\mathcal{D}$ outputs 1 with probability ϵ_2. Otherwise, $\mathcal{A}$ is in **Game 1**, i.e., $\mathcal{D}$ outputs 1 with probability ϵ_1. As a result, $(\mathsf{Samp}, \mathcal{D})$ breaks the security of $\mathcal{AIPO}$ with advantage $|\epsilon_2 - \epsilon_1|$, completing the proof of Lemma 4.2.1. □

Lemma 4.2.2. *If $\mathcal{A}$ succeeds with probability ϵ_2 in* **Game 2**, *then we can construct a PPT adversary $\mathcal{B}$ that breaks the (probabilistic) uninvertibility of $\mathcal{AIPO}$ with advantage ϵ_2.*

Proof of Lemma 4.2.2. Taking as input y generated as $y \leftarrow \mathcal{AIPO}(r)$ where $r \leftarrow \{0,1\}^\lambda$, $\mathcal{B}$ randomly chooses $x \leftarrow \{0,1\}^\lambda$ and gives $(1^\lambda, y, f(x))$ to $\mathcal{A}$. When $\mathcal{A}$ outputs x^*, $\mathcal{B}$ outputs x^*. Since the view of $\mathcal{A}$ is identical to its view in **Game 2**, the probability that $R_{\mathsf{HR}}(y, x^*) = 1$ is ϵ_2. Since $R_{\mathsf{HR}}(y, x^*) = 1$ implies $x^* = r$, we have that $\mathcal{B}$ breaks the uninvertibility of $\mathcal{AIPO}$ successfully with probability ϵ_2, completing the proof of Lemma 4.2.2. □

Let ϵ_i denote the probability that $\mathcal{A}$ succeeds in **Game** i. The security of $\mathcal{AIPO}$ and the uninvertibility of $\mathcal{AIPO}$ respectively imply that $|\epsilon_2 - \epsilon_1|$ and ϵ_2 are negligible. As a result, we have that ϵ_1, which is the probability that $\mathcal{A}$ breaks the security of the ULR-hard relation, is negligible, completing the proof of Theorem 4.2.1. □

IULR-hard relation based on iO. Let $\mathcal{IO}$ be iO, p_a a point-function for a, and $\mathbf{y}$ the program given in Figure 4.1. The construction of IULR-hard relation is as follows.

- $\mathsf{Gen}_{\mathsf{HR}}(1^\lambda)$: Randomly select $x \leftarrow \{0,1\}^\lambda$, compute $y \leftarrow \mathcal{IO}(1^\lambda, \mathbf{y})$ where $\mathbf{y}$ is the program described in Figure 4.1, and output (y, x).
- $R_{\mathsf{HR}}(y, x)$: Output $y(x)$.

Theorem 4.2.2. *Let $\{\mathcal{C}_\lambda\}$ denote a family of circuits whose size is equal to the size of $\mathbf{y}$. If $\mathcal{IO}$ is iO for $\{\mathcal{C}_\lambda\}$, then the above scheme $(\mathsf{Gen}_{\mathsf{HR}}, R_{\mathsf{HR}})$ is a k-IULR-hard relation.*

$\mathbf{y}$
Constant: x.
Input: $\check{x}$.
Output $p_x(\check{x})$.

$\mathbf{y}'$
Constant: $f, f(x)$ where $\lvert f\rvert \leq k$ and f is injective uninvertible.
Input: $\check{x}$.
Output 1 if $f(\check{x}) = f(x)$. Output 0 otherwise.

Figure 4.1: Programs $\mathbf{y}$ and $\mathbf{y}'$. Here, $\mathbf{y}$ is padded so that its size is equal to ℓ which denotes the maximum possible size of $\mathbf{y}'$.

The high-level idea of the proof of Theorem 4.2.2 is as follows.

An adversary $\mathcal{A}$ wins the security game if it outputs x^* such that $y(x^*) = 1$, which happens if and only if $x^* = x$ since y is a point function. As a result, the goal of $\mathcal{A}$ is to find x, given 1^λ, y, and $f(x)$. However, since f is uninvertible, $\mathcal{A}$ cannot find x when it sees only 1^λ and $f(x)$, and intuitively, y contains no more information on x than $f(x)$ due to the power of iO.

Next we give the formal proof. Note that in the formal proof, we define hybrid games in which y denotes obfuscations of different but functionally equivalent circuits $\mathbf{y}$ and $\mathbf{y}'$ (see Figure 4.1). In our construction and in all these hybrids, we pad the circuits so that their sizes are equal to ℓ, which denotes the maximum possible size of $\mathbf{y}'$.

Proof of Theorem 4.2.2. According to the functionality preserving property of $\mathcal{IO}$, $(\mathsf{Gen}_{\mathsf{HR}}, R_{\mathsf{HR}})$ satisfies the correctness property.

Let $\mathcal{A}$ be any PPT adversary and f any injective uninvertible function such that $|f| \leq k$ used as a leakage function. We give hybrid games to show that $\mathcal{A}$ has negligible advantage in breaking the security of $(\mathsf{Gen}_{\mathsf{HR}}, R_{\mathsf{HR}})$.

Game 1: This is the original security game of an IULR-hard relation for $\mathcal{A}$. The challenger samples $(y, x) \leftarrow \mathsf{Gen}_{\mathsf{HR}}(1^\lambda)$ and gives $(1^\lambda, y, f(x))$ to $\mathcal{A}$. $\mathcal{A}$ succeeds if it outputs x^* such that $R_{\mathsf{HR}}(y, x^*) = 1$.

Game 2: This game is the same as **Game 1** except that y is generated as $y \leftarrow \mathcal{IO}(1^\lambda, \mathbf{y}')$ where $\mathbf{y}'$ is the program in Figure 4.1. Here, $\mathbf{y}'$ is padded so that its size is equal to ℓ (which denotes the maximum possible size of $\mathbf{y}'$).

Lemma 4.2.3. *If $\mathcal{A}$ wins with probability ϵ_1 in* **Game 1** *and ϵ_2 in* **Game 2**, *then we can construct a same-circuits sampler* Samp *and an adversary $\mathcal{D}$ that break the indistinguishability property of $\mathcal{IO}$ with advantage $|\epsilon_2 - \epsilon_1|$.*

Proof of Lemma 4.2.3. We show how to construct Samp and $\mathcal{D}$ as follows.

On input 1^λ, Samp randomly chooses $x \leftarrow \{0,1\}^\lambda$, and generates (padded) $\mathbf{y}$ and (padded) $\mathbf{y}'$ respectively. The output of Samp is $(\mathbf{y}, \mathbf{y}', f(x))$. Since f is an injective function, the functionality of $\mathbf{y}$ and $\mathbf{y}'$ are the same.

Taking as input $(1^\lambda, y, f(x))$ where $y \leftarrow \mathcal{IO}(1^\lambda, \mathbf{y})$ or $y \leftarrow \mathcal{IO}(1^\lambda, \mathbf{y}')$, $\mathcal{D}$ runs $x^* \leftarrow \mathcal{A}(1^\lambda, y, f(x))$. If $R_{\mathsf{HR}}(y, x^*) = 1$, $\mathcal{D}$ outputs 1. Otherwise, it outputs 0. If y is an obfuscation of $\mathbf{y}$, then $\mathcal{A}$ is in **Game 1**, i.e., the probability that $\mathcal{D}$ outputs 1 is ϵ_1. Otherwise, $\mathcal{A}$ is in **Game 2**, i.e., $\mathcal{D}$ outputs 1 with probability ϵ_2. As a result, $(\mathsf{Samp}, \mathcal{D})$ breaks the indistinguishability property of $\mathcal{IO}$ with advantage $|\epsilon_2 - \epsilon_1|$, completing the proof of Lemma 4.2.3. □

Lemma 4.2.4. *If $\mathcal{A}$ succeeds with probability ϵ_2 in* **Game 2***, then we can construct a PPT adversary $\mathcal{B}$ that breaks the uninvertibility of f with advantage ϵ_2.*

Proof of Lemma 4.2.4. We show how to construct $\mathcal{B}$ as follows.

Taking as input $f(x)$ where x is randomly chosen from $\{0,1\}^\lambda$, $\mathcal{B}$ generates y as $y \leftarrow \mathcal{IO}(1^\lambda, \mathbf{y}')$, where $\mathbf{y}'$ is the program in Figure 4.1, by making use of $f(x)$ as the constant. Here $\mathbf{y}'$ is padded so that its size is equal to ℓ. Then $\mathcal{B}$ runs $x^* \leftarrow \mathcal{A}(1^\lambda, y, f(x))$ and outputs x^*. Since the view of $\mathcal{A}$ is identical to its view in **Game 2**, the probability that $R_{\mathsf{HR}}(y, x^*) = 1$ is ϵ_2. Since $R_{\mathsf{HR}}(y, x^*) = 1$ implies $x^* = x$, we have that $\mathcal{B}$ breaks the uninvertibility of f successfully with probability ϵ_2, completing the proof of Lemma 4.2.4. □

Let ϵ_i denote the probability that $\mathcal{A}$ succeeds in **Game** i. The indistinguishability property of $\mathcal{IO}$ and the uninvertibility of f respectively imply that $|\epsilon_2 - \epsilon_1|$ and ϵ_2 are negligible. As a result, we have that ϵ_1, which is the probability that $\mathcal{A}$ breaks the security of the IULR-hard relation, is negligible, completing the proof of Theorem 4.2.2. □

4.3 Constructions of Fully Leakage Resilient Signature in the Selective Auxiliary Input Model

In this section, we give our main results in this chapter, which are constructions of FLR signature in the selective auxiliary input model.

In Section 4.3.1, we give a construction of FLR signature (by making use of a ULR-hard relation). In Section 4.3.2, we explain that by substituting the underlying ULR-hard relation with an IULR-hard relation in our FLR signature scheme, we can immediately obtain a weak FLR signature scheme. In Section 4.3.3, we give remarks on our constructions.

4.3.1 Fully Leakage Resilient Signature Scheme

Construction. Let $\mathcal{DIO}$ be diO, $\mathcal{IO}$ iO, and $(\mathsf{Gen}_{\mathsf{HR}}, R_{\mathsf{HR}})$ a ULR-hard relation, while the output size of $\mathsf{Gen}_{\mathsf{HR}}$ is $(l + \lambda)$-bit (where l is the size of public keys and λ the size of secret keys). Let $(F, \mathit{Puncture}, \mathit{Eval})$, $(F_1, \mathit{Puncture}_1, \mathit{Eval}_1)$, $\cdots$, $(F_\lambda, \mathit{Puncture}_\lambda, \mathit{Eval}_\lambda)$ be puncturable PRFs respectively with key spaces $\mathcal{K}$, $\mathcal{K}_1$, $\cdots$, $\mathcal{K}_\lambda$, where $F(K, \cdot)$ maps $(l + \log\lambda + 1 + \lambda)$-bit inputs to λ-bit outputs and $F_j(K_j, \cdot)$ maps $(l + j)$-bit inputs to λ-bit outputs for $j = 1, \cdots, \lambda$.[2] Then our signature scheme $(\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ with message

[2]We do not necessarily have to let the size of messages, number of PRFs (excluding $(F, \mathit{Puncture}, \mathit{Eval})$), and size of outputs of puncturable PRFs be λ. We do this only for simplicity.

space $\{0,1\}^\lambda$ is as follows. In the following, for strings $m, t \in \{0,1\}^\lambda$ we denote by $m[j]$ the jth bit of m, and by $t^{(j)}$ the first j bits of t.

- $\mathsf{Setup}(1^\lambda)$:
  1. Choose $K \leftarrow \mathcal{K}$, $K_1 \leftarrow \mathcal{K}_1$, $\cdots$, $K_\lambda \leftarrow \mathcal{K}_\lambda$.
  2. Compute $\widetilde{\mathbf{Sign}} \leftarrow \mathcal{DIO}(1^\lambda, \mathbf{Sign})$ and $\widetilde{\mathbf{Verify}} \leftarrow \mathcal{IO}(1^\lambda, \mathbf{Verify})$ where **Sign** and **Verify** are the programs in Figure 4.2. Here, **Sign** (respectively, **Verify**) is padded so that its size is equal to the maximum of the sizes of the programs $\mathbf{Sign}_I$ and $\mathbf{Sign}_{II}$ described in Figure 4.5 and Figure 4.8 (respectively, $\mathbf{Verify}_I$ and $\mathbf{Verify}_{II}$ described in Figure 4.6 and Figure 4.9).
  3. Output $par = (1^\lambda, \widetilde{\mathbf{Sign}}, \widetilde{\mathbf{Verify}})$.[3]
- $\mathsf{Gen}(par)$:
  1. Compute $(y, x) \leftarrow \mathsf{Gen}_{\mathsf{HR}}(1^\lambda)$.
  2. Output $(pk, sk) = (y, x)$.

Sign
Constant: $K, (K_j)_{j=1}^\lambda$.
Input: $\check{y}, \check{x}, \check{m}, \check{t}$.
If $R_{\mathsf{HR}}(\check{y}, \check{x}) = 0$, output $\perp$.
Compute $\check{s}_1 = \oplus_{j=1}^\lambda F(K, \check{y}\|\|j\|\|\check{m}[j]\|\|\check{t})$.
Compute $\check{s}_2 = \oplus_{j=1}^\lambda F_j(K_j, \check{y}\|\|\check{t}^{(j)})$.
Output $\check{\sigma} = (\check{s}_1, \check{s}_2, \check{t})$.

Verify
Constant: $K, (K_j)_{j=1}^\lambda$.
Input: $\check{y}, \check{m}, \check{\sigma}$.
Parse $\check{\sigma} = (\check{s}_1, \check{s}_2, \check{t})$.
If $\check{s}_1 = \oplus_{j=1}^\lambda F(K, \check{y}\|\|j\|\|\check{m}[j]\|\|\check{t})$
and $\check{s}_2 = \oplus_{j=1}^\lambda F_j(K_j, \check{y}\|\|\check{t}^{(j)})$, output 1.
Otherwise, output 0.

Figure 4.2: Programs **Sign** and **Verify**. **Sign** and **Verify** are respectively padded so that their sizes are equal to the programs in the security proof.

- $\mathsf{Sign}(sk, m)$:
  1. Randomly choose $t \leftarrow \{0,1\}^\lambda$ and output $\sigma = \widetilde{\mathbf{Sign}}(y, x, m, t)$ where $y = pk$ and $x = sk$.[4]
- $\mathsf{Verify}(pk, m, \sigma)$:
  1. Output $\widetilde{\mathbf{Verify}}(y, m, \sigma)$ where $y = pk$.

[3] $\widetilde{\mathbf{Sign}}$ and $\widetilde{\mathbf{Verify}}$ do not have to be generated in every key generation procedure since they do not depend on (y, x). Instead, they can be used as global parameters for this scheme.

[4] Note that Sign is implicitly given (par, pk) as input (see Definition 2.7).

It is obvious that our construction is public-coin, since like the Ramchen-Waters style signature scheme, we only use a randomness t, which is part of a signature, in the signing procedure.

The security of our proposed scheme is guaranteed by the following theorem.

Theorem 4.3.1. *Let $\mathcal{C}_\lambda$ denote a family of circuits whose size is equal to the size of* **Sign** *and $\mathcal{C}'_\lambda$ a family of circuits whose size is equal to the size of* **Verify**. *If $(\mathsf{Gen}_{\mathsf{HR}}, R_{\mathsf{HR}})$ is a ULR-hard relation, $\mathcal{DIO}$ is diO for $\{\mathcal{C}_\lambda\}$, $\mathcal{IO}$ is iO for $\{\mathcal{C}'_\lambda\}$, $(F, Puncture, Eval)$ and $\{(F_j, Puncture_j, Eval_j)\}_{j=1}^{\lambda}$ are puncturable PRFs, and there exists an injective one-way function $h : \{0,1\}^\lambda \to \{0,1\}^*$,*[5] *then* $(\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *is an FLR signature scheme in the selective auxiliary input model.*

In the security proof of Theorem 4.3.1, we define hybrid games in which $\widetilde{\textbf{Sign}}$ (respectively, $\widetilde{\textbf{Verify}}$) denotes obfuscations of different circuits, and we pad the underlying circuits of $\widetilde{\textbf{Sign}}$ (respectively, $\widetilde{\textbf{Verify}}$) in our construction and in all these hybrids so that they have the same size. The high level idea of this proof is as follows.

Outline of the security proof. To explain the outline, we consider a simpler version of our construction where **Sign** outputs a Sahai-Waters style [101] signature linked with y instead. This construction is the same as the original one except that we skip sampling $\{K_j\}_{j=1}^{\lambda}$, and **Sign** and **Verify** are simplified as described in Figure 4.3. Note that in this case, the signature scheme is only selectively unforgeable, i.e., an adversary is required to determine the challenge message, on which a signature will be forged, before seeing the verification key, since the Sahai-Water signature scheme is selectively unforgeable.

Sign
Constant: K.
Input: $\overline{y}, \overline{x}, \overline{m}$.
If $R_{\mathsf{HR}}(\overline{y}, \overline{x}) = 0$, output $\perp$.
Output $\sigma = F(K, \overline{y}\|\|\overline{m})$.

Verify
Constant: K.
Input: $\overline{y}, \overline{m}, \overline{\sigma}$.
If $\overline{\sigma} = F(K, \overline{y}\|\|\overline{m})$, output 1.
Otherwise, output 0.

Figure 4.3: Simplified **Sign** and **Verify**.

Now we give another pair of programs $(\textbf{Sign}', \textbf{Verify}')$ in Figure 4.4.

Given $(\textbf{Sign}, \textbf{Sign}', \alpha)$, where α is auxiliary information (including y, constants used to generate signatures, the verification algorithm, and leakage $f(x)$), an adversary is not able find a correct secret key w.r.t y, due to the security of the ULR-hard relation. The reason is that the tuple $(\textbf{Sign}, \textbf{Sign}', \alpha)$ contains no information about x other than y and $f(x)$. As a result, an adversary $\mathcal{A}$ is not able to find an input $(\overline{y}, \overline{x}, \overline{m})$ that leads **Sign** and $\textbf{Sign}'$ to different outputs, since such an input must satisfy $\overline{y}||\overline{m} = y||m^*$ and $R_{\mathsf{HR}}(\overline{y}, \overline{x}) = 1$, i.e., $R_{\mathsf{HR}}(y, \overline{x}) = 1$. According to the property of diO, $\mathcal{A}$ is not able to distinguish $\widetilde{\textbf{Sign}}$

[5] h appears in the security proof.

Sign$'$
Constant: $K\{y\|\|m^*\}$.
Input: $\overline{y}, \overline{x}, \overline{m}$.
If $R_{\mathsf{HR}}(\overline{y}, \overline{x}) = 0$, output $\perp$.
Output $\sigma = F(K\{y\|\|m^*\}, \overline{y}\|\|\overline{m})$.

Verify$'$
Constant: $K\{y\|\|m^*\}$, $h(F(K\{y\|\|m^*\}))$, $y\|\|m^*$ where h is an injective one-way function.
Input: $\overline{y}, \overline{m}, \overline{\sigma}$.
If $\overline{y}\|\|\overline{m} = y\|\|m^*$:
If $h(\overline{\sigma}) = h(F(K, y\|\|m^*))$, output 1.
Else:
If $\overline{\sigma} = F(K\{y\|\|m^*\}, \overline{y}\|\|\overline{m})$, output 1.
Else output 0.

Figure 4.4: Programs **Sign**$'$ and **Verify**$'$.

and $\widetilde{\textbf{Sign}}'$, which are respectively differing-inputs obfuscations of **Sign** and **Sign**$'$. Furthermore, since the functionality of **Verify**$'$ is the same as that of **Verify**, $\mathcal{A}$ cannot distinguish $\widetilde{\textbf{Verify}}$ and $\widetilde{\textbf{Verify}}'$, which are respectively indistinguishability obfuscations of **Verify** and **Verify**$'$. However, if we substitute $(\widetilde{\textbf{Sign}}, \widetilde{\textbf{Verify}})$ with $(\widetilde{\textbf{Sign}}', \widetilde{\textbf{Verify}}')$, there is only a negligible chance for $\mathcal{A}$ to obtain the forgery $F(K, y||m^*)$, since $F(K, y||m^*)$ is independent of $K\{y||m^*\}$, and the only information about $F(K, y||m^*)$ $\mathcal{A}$ may learn is $h(F(K, y||m^*))$ while h is one-way. As a result, $\mathcal{A}$ cannot obtain the forged signature $F(K, y||m^*)$ on m^*.

Since the above signature scheme only achieves selective unforgeable, we adopt Ramchen-Waters style signatures [97] (linked with $\overline{y}$) instead of $F(K, \overline{y}||\overline{m})$ to achieve adaptive security. In this case, the security proof is more complicated. However, the basic idea does not change.

The formal proof is as follows.

Proof of Theorem 4.3.1. Let $\mathcal{A}$ be any PPT adversary and f any computable uninvertible function used as a leakage function. For $i = 1, \cdots, q$ where $q = q(\lambda)$ is a polynomial denoting the maximum of the number of messages queried by $\mathcal{A}$, let m_i be the ith signing query, $\sigma_i = (s_{1i}, s_{2i}, t_i)$ the answer to the ith signing query, and $(m^*, \sigma^* = (s_1^*, s_2^*, t^*))$ the forgery generated by $\mathcal{A}$.

To win the FLR security experiment, $\mathcal{A}$ has to output a forgery which is one of the following two types.

- **type I** forgery: $t^* \notin \{t_1, t_2, \cdots, t_q\} \wedge \mathsf{Verify}(pk, m^*, \sigma^*) = 1$.
- **type II** forgery: $t^* \in \{t_1, t_2, \cdots, t_q\} \wedge \mathsf{Verify}(pk, m^*, \sigma^*) = 1$.

type I. We give hybrid games to show that $\mathcal{A}$ outputs a **type I** forgery with negligible probability.

Game 1: This is the original security game for $\mathcal{A}$. The challenger runs $par \leftarrow \mathsf{Setup}(1^\lambda)$, $(pk, sk) \leftarrow \mathsf{Gen}(par)$, and gives $(par, pk, f(sk))$ to $\mathcal{A}$. When $\mathcal{A}$ makes signing queries, the challenger answers them honestly. At some point, $\mathcal{A}$ outputs a forgery (m^*, σ^*). $\mathcal{A}$ succeeds if (m^*, σ^*) is a **type I** forgery.

Game 2: This game is the same as **Game 1** except that the challenger randomly chooses $\hat{i} \leftarrow \{1, \cdots, q\}$ and $\hat{j} \leftarrow \{1, \cdots, \lambda\}$ at the beginning of the game. Furthermore, $\mathcal{A}$ succeeds if its forgery (m^*, σ^*) is a **type I** forgery such that $t^{*(\hat{j})} = t_{\hat{i}}^{(\hat{j})} \oplus e_{\hat{j}}$ and $t^{*(\hat{j})} \neq t_i^{(\hat{j})}$ for all $i \in \{1, \cdots, q\}$. In other words, $\mathcal{A}$ fails if (m^*, σ^*) is not a **type I** forgery or $t_{\hat{i}}^{(\hat{j})}$ is not the longest $t_i^{(j)}$ such that $t_i^{(j)} \oplus e_j = t^{*(j)}$ for $i \in \{1, \cdots, q\}$ and $j \in \{1, \cdots, \lambda\}$. Here, $e_{\hat{j}}$ denotes the $\hat{j}$-bit string $0 \cdots 01$.

Lemma 4.3.1. *If $\mathcal{A}$ succeeds with probability ϵ_1 in* **Game 1** *and ϵ_2 in* **Game 2**, *then we have $\epsilon_2 \geq \epsilon_1/(q \cdot k)$.*

Proof of Lemma 4.3.1. Since if $\mathcal{A}$ outputs a **type I** forgery, we have $t^* \notin \{t_1, \cdots, t_q\}$, $(\hat{i}, \hat{j})$ such that $t^{*(\hat{j})} = t_{\hat{i}}^{(\hat{j})} \oplus e_{\hat{j}}$ and $t^{*(\hat{j})} \neq t_i^{(\hat{j})}$ for all $i \in \{1, \cdots, q\}$ must exist. Then this lemma follows from the fact that the view of $\mathcal{A}$ in **Game 1** is identical to its view in **Game 2** and $\mathcal{A}$ learns no information on which $(\hat{i}, \hat{j})$ is chosen. □

Game 3: This game is the same as **Game 2** except that the challenger chooses $t_1, \cdots, t_q \leftarrow \{0,1\}^\lambda$ at the beginning of the game, and $\widetilde{\mathbf{Sign}}$ is the obfuscation of program $\mathbf{Sign}_I$ given in Figure 4.5 instead of **Sign**. Here, $\mathbf{Sign}_I$ is padded so that its size is equal to the size of **Sign** (see Figure 4.2) and $\mathbf{Sign}_{II}$ (see Figure 4.8), if necessary.

$\mathbf{Sign}_I$
Constant: $\hat{j}$, K, $K_{\hat{j}}\{s\}$, $(K_j)_{j \neq \hat{j}}$, and s, where $s = y||(t_{\hat{i}}^{(\hat{j})} \oplus e_{\hat{j}})$.
Input: $\overline{y}, \overline{x}, \overline{m}, \overline{t}$.
If $R_{\mathsf{HR}}(\overline{y}, \overline{x}) = 0$ or $\overline{y}||\overline{t}^{(\hat{j})} = s$, output $\perp$.
Compute $\overline{s}_1 = \oplus_{j=1}^{\lambda} F(K, \overline{y}||j||\overline{m}[j]||\overline{t})$.
Compute $\overline{s}_2 = \left(\oplus_{j \neq \hat{j}} F_j(K_j, \overline{y}||\overline{t}^{(j)}) \right) \oplus Eval_{\hat{j}}(K_{\hat{j}}\{s\}, \overline{y}||\overline{t}^{(\hat{j})})$.
Output $\overline{\sigma} = (\overline{s}_1, \overline{s}_2, \overline{t})$.

Figure 4.5: Program $\mathbf{Sign}_I$. $\mathbf{Sign}_I$ is the same as **Sign** except that the constant $K_{\hat{j}}$ is substituted with $(\hat{j}, K_{\hat{j}}\{s\}, s)$, $\mathbf{Sign}_I$ aborts if $\overline{y}||\overline{t}^{(\hat{j})} = s$, and $\overline{s}_2$ is evaluated by computing $\left(\oplus_{j \neq \hat{j}} F_j(K_j, \overline{y}||\overline{t}^{(j)}) \right) \oplus Eval_{\hat{j}}(K_{\hat{j}}\{s\}, \overline{y}||\overline{t}^{(\hat{j})})$ instead.

Lemma 4.3.2. *If $\mathcal{A}$ succeeds with probability ϵ_2 in* **Game 2** *and ϵ_3 in* **Game 3**, *then we can construct a differing-inputs sampler* Samp *and an adversary $\mathcal{D}$ that break the differing-inputs property of $\mathcal{DIO}$ with advantage $|\epsilon_3 - \epsilon_2|$.*

Proof of Lemma 4.3.2. We first give the description of Samp.

Taking as input 1^λ, Samp randomly chooses $t_1, \cdots, t_q \leftarrow \{0,1\}^\lambda$, $K \leftarrow \mathcal{K}$, $K_1 \leftarrow \mathcal{K}_1$, $\cdots$, $K_\lambda \leftarrow \mathcal{K}_\lambda$, $\hat{i} \leftarrow \{1, \cdots, q\}$, and $\hat{j} \leftarrow \{1, \cdots, \lambda\}$, and runs $(y, x) \leftarrow \mathsf{Gen}_{\mathsf{HR}}(1^\lambda)$. Next it computes $K_{\hat{j}}\{s\} = Puncture_{\hat{j}}(K_{\hat{j}}, s)$ where $s = y||(t_{\hat{i}}^{(\hat{j})} \oplus e_{\hat{j}})$, generates $(\mathbf{Sign}, \mathbf{Sign}_I, \mathbf{Verify})$, and outputs $(\mathbf{Sign}, \mathbf{Sign}_I, \alpha)$ where $\alpha = (f(x), (K_j)_{j \neq \hat{j}}, K_{\hat{j}}\{s\}, K, s, (t_i)_{i=1}^q, y, \mathbf{Verify}, \hat{i}, \hat{j})$.

Claim 4.3.1. Samp *is a differing-inputs sampler for* $\{\mathcal{C}_\lambda\}$.

Proof of Claim 4.3.1. If a tuple (y^*, x^*, m^*, t^*) satisfies $\mathbf{Sign}(y^*, x^*, m^*, t^*) \neq \mathbf{Sign}_I(y^*, x^*, m^*, t^*)$, it must satisfy $y^* = y \wedge R_{\mathsf{HR}}(y^*, x^*) = 1$ (i.e., $R_{\mathsf{HR}}(y, x^*) = 1$). The reason is that if $R_{\mathsf{HR}}(y^*, x^*) = 0$, $\mathbf{Sign}$ and $\mathbf{Sign}_I$ output $\perp$, and if $R_{\mathsf{HR}}(y^*, x^*) = 1 \wedge y^* \neq y$ (i.e., $R_{\mathsf{HR}}(y^*, x^*) = 1 \wedge y^* || t^{*(\hat{j})} \neq s$), they output the same signature.

Now we argue that if there exists a PPT adversary $\mathcal{B}$, taking as input $(\mathbf{Sign}, \mathbf{Sign}_I, \alpha)$ generated by Samp, can output x^* such that $R_{\mathsf{HR}}(y, x^*) = 1$ with advantage ϵ, then we can construct an adversary $\mathcal{B}'$ that breaks the security of $(\mathsf{Gen}_{\mathsf{HR}}, R_{\mathsf{HR}})$ in the presence of uninvertible leakage generated from f with advantage ϵ. The proof is as follows.

Taking as input $(1^\lambda, y, f(x))$ from the ULR-hard relation challenger who runs $(y, x) \leftarrow \mathsf{Gen}_{\mathsf{HR}}(1^\lambda)$, $\mathcal{B}'$ randomly chooses $t_1, \cdots, t_q \leftarrow \{0,1\}^\lambda$, $K \leftarrow \mathcal{K}$, $K_1 \leftarrow \mathcal{K}_1$, $\cdots$, $K_\lambda \leftarrow \mathcal{K}_\lambda$, $\hat{i} \leftarrow \{1, \cdots, q\}$, and $\hat{j} \leftarrow \{1, \cdots, \lambda\}$, and runs $K_{\hat{j}}\{s\} = \mathit{Puncture}_{\hat{j}}(K_{\hat{j}}, s)$ where $s = y || (t_{\hat{i}}^{(\hat{j})} \oplus e_{\hat{j}})$. Then it generates $(\mathbf{Sign}, \mathbf{Sign}_I, \mathbf{Verify})$ and gives $(1^\lambda, \mathbf{Sign}, \mathbf{Sign}_I, \alpha)$ where $\alpha = (f(x), (K_j)_{j \neq \hat{j}}, K_{\hat{j}}\{s\}, K, s, (t_i)_{i=1}^q, y, \mathbf{Verify}, \hat{i}, \hat{j})$, the distribution of which is completely the same as the distribution of the output of Samp, to $\mathcal{B}$. When $\mathcal{B}$ outputs x^*, $\mathcal{B}'$ outputs x^*.

Since the probability that $R_{\mathsf{HR}}(y, x^*) = 1$ is ϵ, we have that $\mathcal{B}'$ breaks the security of $(\mathsf{Gen}_{\mathsf{HR}}, R_{\mathsf{HR}})$ with probability ϵ. Since $(\mathsf{Gen}_{\mathsf{HR}}, R_{\mathsf{HR}})$ is a ULR-hard relation and f is an uninvertible function, ϵ is negligible. Hence, Samp is a differing-inputs sampler for $\{\mathcal{C}_\lambda\}$, completing the proof of Claim 4.3.1. □

We now give the description of $\mathcal{D}$. Taking as input $(1^\lambda, \widetilde{\mathbf{Sign}}, \alpha)$ where $\widetilde{\mathbf{Sign}} \leftarrow \mathcal{DIO}(1^\lambda, \mathbf{Sign})$ or $\widetilde{\mathbf{Sign}} \leftarrow \mathcal{DIO}(1^\lambda, \mathbf{Sign}_I)$, $\mathcal{D}$ computes $\widetilde{\mathbf{Verify}} \leftarrow \mathcal{IO}(1^\lambda, \mathbf{Verify})$, gives $(par, pk, f(x))$ where $par = (1^\lambda, \widetilde{\mathbf{Sign}}, \widetilde{\mathbf{Verify}})$ and $pk = y$ to $\mathcal{A}$, and answers the ith signing query by returning (s_{1i}, s_{2i}, t_i) where $s_{1i} = \oplus_{j=1}^{\lambda} F(K, y||j||m_i[j]||t_i)$ and $s_{2i} = \oplus_{j \neq \hat{j}} F_j(K_j, y||t_i^{(j)}) \oplus \mathit{Eval}_{\hat{j}}(K_{\hat{j}}\{s\}, y||t_i^{(\hat{j})})$. Note that $\mathit{Eval}_{\hat{j}}(K_{\hat{j}}\{s\}, y||(t_{\hat{i}}^{(\hat{j})} \oplus e_{\hat{j}}))$ will not be called by $\mathcal{D}$, or we do not have $t_{\hat{i}}^{(\hat{j})} \oplus e_{\hat{j}} \neq t_i^{(\hat{j})}$ for all i. $\mathcal{D}$ outputs 1 if $\mathcal{A}$ outputs a **type I** forgery such that $t^{*(\hat{j})} = t_{\hat{i}}^{(\hat{j})} \oplus e_{\hat{j}}$ and $t^{*(\hat{j})} \neq t_i^{(\hat{j})}$ for all $i \in \{1, \cdots, q\}$. Otherwise, $\mathcal{D}$ outputs 0.

If $\widetilde{\mathbf{Sign}}$ is generated as $\widetilde{\mathbf{Sign}} \leftarrow \mathcal{DIO}(1^\lambda, \mathbf{Sign})$, the probability that $\mathcal{D}$ outputs 1 is the same as the probability that $\mathcal{A}$ succeeds in **Game 2**. Otherwise, it is the same as the probability that $\mathcal{A}$ succeeds in **Game 3**. Since $\mathcal{A}$ succeeds with probability ϵ_2 in **Game 2** and ϵ_3 in **Game 3**, we have that $(\mathsf{Samp}, \mathcal{D})$ breaks the differing-inputs property of diO with advantage $|\epsilon_3 - \epsilon_2|$, completing the proof of Lemma 4.3.2. □

Game 4: This game is the same as **Game 3** except that $\widetilde{\mathbf{Verify}}$ is the obfuscation of program $\mathbf{Verify}_I$ given in Figure 4.6 instead of $\mathbf{Verify}$, where h is an injective one-way function. Here, $\mathbf{Verify}_I$ is padded so that its size is equal to the sizes of the program $\mathbf{Verify}$ (see Figure 4.2) and $\mathbf{Verify}_{II}$ (see Figure 4.9), if necessary.

Verify$_I$

Constant: $\hat{j}$, K, $K_{\hat{j}}\{s\}$, $(K_j)_{j\neq\hat{j}}$, $h(F_{\hat{j}}(K_{\hat{j}}, s))$, and s, where $s = y||(t_{\hat{i}}^{(\hat{j})} \oplus e_{\hat{j}})$.

Input: $\overline{y}, \overline{m}, \overline{\sigma}$.

Parse $\overline{\sigma} = (\overline{s}_1, \overline{s}_2, \overline{t})$.

If $\overline{y}||\overline{t}^{(\hat{j})} = s$:

  If $\overline{s}_1 = \oplus_{j=1}^{\lambda} F(K, \overline{y}||j||\overline{m}[j]||\overline{t})$ and $h(\overline{s}_2 \oplus_{j\neq\hat{j}} F_j(K_j, \overline{y}||\overline{t}^{(j)})) = h(F_{\hat{j}}(K_{\hat{j}}, s))$, output 1.

  Else, output 0.

Else:

  If $\overline{s}_1 = \oplus_{j=1}^{\lambda} F(K, \overline{y}||j||\overline{m}[j]||\overline{t})$ and $\overline{s}_2 = \left(\oplus_{j\neq\hat{j}} F_j(K_j, \overline{y}||\overline{t}^{(j)})\right) \oplus Eval_{\hat{j}}(K_{\hat{j}}\{s\}, \overline{y}||\overline{t}^{(\hat{j})})$, output 1.

  Else, output 0.

Figure 4.6: Program **Verify**$_I$ in **Game 4**. **Verify**$_I$ is the same as **Verify** except that the constant $K_{\hat{j}}$ is substituted with $(\hat{j}, K_{\hat{j}}\{s\}, h(F_{\hat{j}}(K_{\hat{j}}, s)), s)$, and $\overline{s}_2$ is verified by checking $h(\overline{s}_2 \oplus_{j\neq\hat{j}} F_j(K_j, \overline{y}||\overline{t}^{(j)})) = h(F_{\hat{j}}(K_{\hat{j}}, s))$ if $\overline{y}||\overline{t}^{(\hat{j})} = s$ and $\overline{s}_2 = \left(\oplus_{j\neq\hat{j}} F_j(K_j, \overline{y}||\overline{t}^{(j)})\right) \oplus Eval_{\hat{j}}(K_{\hat{j}}\{s\}, \overline{y}||\overline{t}^{(\hat{j})})$ if $\overline{y}||\overline{t}^{(\hat{j})} \neq s$.

Lemma 4.3.3. *If $\mathcal{A}$ succeeds with probability ϵ_3 in* **Game 3** *and ϵ_4 in* **Game 4**, *then we can construct a same-circuits sampler* Samp *and an adversary $\mathcal{D}$ that break the indistinguishability property of $\mathcal{IO}$ with advantage $|\epsilon_4 - \epsilon_3|$.*

Proof of Lemma 4.3.3. We show how to construct Samp and $\mathcal{D}$ as follows.

Taking as input 1^λ, Samp randomly chooses $t_1, \cdots, t_q \leftarrow \{0,1\}^\lambda$, $K \leftarrow \mathcal{K}$, $K_1 \leftarrow \mathcal{K}_1$, $\cdots$, $K_\lambda \leftarrow \mathcal{K}_\lambda$, $\hat{i} \leftarrow \{1, \cdots, q\}$, and $\hat{j} \leftarrow \{1, \cdots, \lambda\}$, and runs $(y, x) \leftarrow \mathsf{Gen}_{\mathsf{HR}}(1^\lambda)$. Next it computes $K_{\hat{j}}\{s\} = Puncture_{\hat{j}}(K_{\hat{j}}, s)$ where $s = y||(t_{\hat{i}}^{(\hat{j})} \oplus e_{\hat{j}})$, generates $(\mathbf{Verify}, \mathbf{Verify}_I, \mathbf{Sign}_I)$, and outputs $(\mathbf{Verify}, \mathbf{Verify}_I, \alpha)$ where $\alpha = (x, (t_i)_{i=1}^q, y, \mathbf{Sign}_I, \hat{i}, \hat{j})$. Since $\overline{s}_2 = \oplus_{j=1}^{\lambda} F_j(K_j, \overline{y}||\overline{t}^{(j)})$ is equivalent to $\overline{s}_2 \oplus_{j\neq\hat{j}} F_j(K_j, \overline{y}||\overline{t}^{(j)}) = F_{\hat{j}}(K_{\hat{j}}, s)$ when $\overline{y}||\overline{t}^{(\hat{j})} = s$, h is injective, and $Eval_{\hat{j}}(K_{\hat{j}}\{s\}, \overline{y}||\overline{t}^{(\hat{j})})$ is equivalent to $F_{\hat{j}}(K_{\hat{j}}, \overline{y}||\overline{t}^{(\hat{j})})$ when $\overline{y}||\overline{t}^{(\hat{j})} \neq s$, **Verify** and **Verify**$_I$ have the same functionality. Hence, Samp is a same-circuits sampler for $\{\mathcal{C}'_\lambda\}$.

Taking as input $(1^\lambda, \widetilde{\mathbf{Verify}}, \alpha)$ where $\widetilde{\mathbf{Verify}} \leftarrow \mathcal{IO}(1^\lambda, \mathbf{Verify})$ or $\widetilde{\mathbf{Verify}} \leftarrow \mathcal{IO}(1^\lambda, \mathbf{Verify}_I)$, $\mathcal{D}$ computes $\widetilde{\mathbf{Sign}} \leftarrow \mathcal{DIO}(1^\lambda, \mathbf{Sign}_I)$, gives $(par, pk, f(x))$ where $par = (1^\lambda, \widetilde{\mathbf{Sign}}, \widetilde{\mathbf{Verify}})$ and $pk = y$ to $\mathcal{A}$, and answers the ith signing query m_i by returning σ_i where $\sigma_i = \widetilde{\mathbf{Sign}}(y, x, m_i, t_i)$. $\mathcal{D}$ outputs 1 if $\mathcal{A}$ outputs a **type I** forgery such that $t^{*(\hat{j})} = t_{\hat{i}}^{(\hat{j})} \oplus e_{\hat{j}}$ and $t^{*(\hat{j})} \neq t_i^{(\hat{j})}$ for all $i \in \{1, \cdots, q\}$. Otherwise, $\mathcal{D}$ outputs 0.

If $\widetilde{\mathbf{Verify}} \leftarrow \mathcal{IO}(1^\lambda, \mathbf{Verify})$, then $\mathcal{A}$ is in **Game 3**. Otherwise, it is in **Game 4**. Since $\mathcal{A}$ succeeds with probability ϵ_3 in **Game 3** and ϵ_4 in **Game 4**, we have that $(\mathsf{Samp}, \mathcal{D})$ breaks the indistinguishability property of iO with advantage $|\epsilon_4 - \epsilon_3|$, completing the proof of Lemma 4.3.3. □

Game 5: This game is the same as **Game 4** except that the constant $h(F(K_{\hat{j}}, s))$ is substituted with $h(r)$ where r is a randomness chosen from $\{0,1\}^\lambda$, i.e., $\mathbf{Verify}_I$ is generated as shown in Figure 4.7.

$\mathbf{Verify}_I$
Constant: $\hat{j}$, K, $K_{\hat{j}}\{s\}$, $(K_j)_{j\neq\hat{j}}$, $h(r)$, and s, where $s = y||(t_{\hat{i}}^{(\hat{j})} \oplus e_{\hat{j}})$.
Input: $\overline{y}, \overline{m}, \overline{\sigma}$.
Parse $\overline{\sigma} = (\overline{s}_1, \overline{s}_2, \overline{t})$.
If $\overline{y}||\overline{t}^{(\hat{j})} = s$:
  If $\overline{s}_1 = \oplus_{j=1}^{\lambda} F(K, \overline{y}||j||\overline{m}[j]||\overline{t})$ and $h(\overline{s}_2 \oplus_{j\neq\hat{j}} F_j(K_j, \overline{y}||\overline{t}^{(j)})) = h(r)$, output 1.
  Else, output 0.
Else:
  If $\overline{s}_1 = \oplus_{j=1}^{\lambda} F(K, \overline{y}||j||\overline{m}[j]||\overline{t})$ and $\overline{s}_2 = \left(\oplus_{j\neq\hat{j}} F_j(K_j, \overline{y}||\overline{t}^{(j)})\right) \oplus Eval_{\hat{j}}(K_{\hat{j}}\{s\}, \overline{y}||\overline{t}^{(\hat{j})})$, output 1.
  Else, output 0.

Figure 4.7: Program $\mathbf{Verify}_I$ in **Game 5**. This $\mathbf{Verify}_I$ is the same as the one in **Game 4**, except that $h(F(K_{\hat{j}}, s))$ is substituted with $h(r)$ where r is a randomness.

Lemma 4.3.4. *If $\mathcal{A}$ succeeds with probability ϵ_4 in* **Game 4** *and ϵ_5 in* **Game 5**, *then we can construct an adversary $(\mathcal{B}_1, \mathcal{B}_2)$ that breaks the pseudorandom at punctured point property of $(F_{\hat{j}}, Puncture_{\hat{j}}, Eval_{\hat{j}})$ with advantage $|\epsilon_5 - \epsilon_4|$.*

Proof of Lemma 4.3.4. We construct $(\mathcal{B}_1, \mathcal{B}_2)$ as follows.

$\mathcal{B}_1$ randomly chooses $t_1, \cdots, t_q \leftarrow \{0,1\}^\lambda$, $K \leftarrow \mathcal{K}$, $K_1 \leftarrow \mathcal{K}_1$, $\cdots$, $K_{\hat{j}-1} \leftarrow \mathcal{K}_{\hat{j}-1}$, $K_{\hat{j}+1} \leftarrow \mathcal{K}_{\hat{j}+1}$, $\cdots$, $K_\lambda \leftarrow \mathcal{K}_\lambda$, $\hat{i} \leftarrow \{1, \cdots, q\}$, and $\hat{j} \leftarrow \{1, \cdots, \lambda\}$, and runs $(y, x) \leftarrow \mathsf{Gen}_{\mathsf{HR}}(1^\lambda)$. Then it sets $s = y||(t_{\hat{i}}^{(\hat{j})} \oplus e_{\hat{j}})$, and outputs s along with an auxiliary input α which contains all the elements sampled by $\mathcal{B}_1$.

Taking as input $(K_{\hat{j}}\{s\}, F_{\hat{j}}(K_{\hat{j}}, s), \alpha)$ or $(K_{\hat{j}}\{s\}, r, \alpha)$ where $K_{\hat{j}} \leftarrow \mathcal{K}$, $r \leftarrow \{0,1\}^\lambda$, and $K_{\hat{j}}\{s\} = Puncture_{\hat{j}}(K_{\hat{j}}, s)$ from the challenger of $(\mathcal{B}_1, \mathcal{B}_2)$, $\mathcal{B}_2$ generates $\mathbf{Sign}_I$ and $\mathbf{Verify}_I$ with $h(F_{\hat{j}}(K_{\hat{j}}, s))$ or $h(r)$, and runs $\widetilde{\mathbf{Sign}} \leftarrow \mathcal{DIO}(1^\lambda, \mathbf{Sign}_I)$ and $\widetilde{\mathbf{Verify}} \leftarrow \mathcal{IO}(1^\lambda, \mathbf{Verify}_I)$. Next $\mathcal{B}_2$ gives $(par, pk, f(x))$ where $par = (1^\lambda, \widetilde{\mathbf{Sign}}, \widetilde{\mathbf{Verify}})$ and $pk = y$ to $\mathcal{A}$. When receiving the ith signing query m_i from $\mathcal{A}$, $\mathcal{B}_2$ runs $\sigma_i = \widetilde{\mathbf{Sign}}(y, x, m_i, t_i)$ and returns σ_i to $\mathcal{A}$. If $\mathcal{A}$ outputs a **type I** forgery such that $t^{*(\hat{j})} = t_{\hat{i}}^{(\hat{j})} \oplus e_{\hat{j}}$ and $t^{*(\hat{j})} \neq t_i^{(\hat{j})}$ for all $i \in \{1, \cdots, q\}$, $\mathcal{B}_2$ outputs 1. Otherwise, $\mathcal{B}_2$ outputs 0.

When the input of $\mathcal{B}_2$ is $(K_{\hat{j}}\{s\}, F_{\hat{j}}(K_{\hat{j}}, s), \alpha)$, $\mathcal{A}$ is in **Game 4**, i.e., $\mathcal{B}_2$ outputs 1 with probability ϵ_4. Otherwise, $\mathcal{A}$ is in **Game 5**, i.e., $\mathcal{B}_2$ outputs 1 with probability ϵ_5. As a result, $(\mathcal{B}_1, \mathcal{B}_2)$ breaks the pseudorandom at punctured point property of $(F_{\hat{j}}, Puncture_{\hat{j}}, Eval_{\hat{j}})$ with advantage $|\epsilon_5 - \epsilon_4|$, completing the proof of Lemma 4.3.4. □

Lemma 4.3.5. *If $\mathcal{A}$ succeeds with probability ϵ_5 in* **Game 5**, *then we can construct a PPT adversary $\mathcal{B}_I$ that breaks the one-wayness of h with advantage ϵ_5.*

Proof of Lemma 4.3.5. Given $h(r)$ from the challenger who randomly chooses $r \leftarrow \{0,1\}^\lambda$, $\mathcal{B}_I$ randomly chooses $t_1, \cdots, t_q \leftarrow \{0,1\}^\lambda$, $K \leftarrow \mathcal{K}$, $K_1 \leftarrow \mathcal{K}_1, \cdots, K_\lambda \leftarrow \mathcal{K}_\lambda$, $\hat{i} \leftarrow \{1, \cdots, q\}$, and $\hat{j} \leftarrow \{1, \cdots, \lambda\}$, runs $(y, x) \leftarrow \mathsf{Gen}_{\mathsf{HR}}(1^\lambda)$, and computes $K_{\hat{j}}\{s\} = \mathit{Puncture}_{\hat{j}}(K_{\hat{j}}, s)$ where $s = y||(t_{\hat{i}}^{(\hat{j})} \oplus e_{\hat{j}})$. Then it generates $\mathbf{Sign}_I$ (see Figure 4.5) and $\mathbf{Verify}_I$ (see Figure 4.7) and runs $\widetilde{\mathbf{Sign}} \leftarrow \mathcal{DIO}(1^\lambda, \mathbf{Sign}_I)$ and $\widetilde{\mathbf{Verify}} \leftarrow \mathcal{IO}(1^\lambda, \mathbf{Verify}_I)$. Next $\mathcal{B}_I$ gives $(par, pk, f(x))$ where $par = (1^\lambda, \widetilde{\mathbf{Sign}}, \widetilde{\mathbf{Verify}})$ and $pk = y$ to $\mathcal{A}$. When receiving the ith signing query m_i from $\mathcal{A}$, $\mathcal{B}_I$ runs $\sigma_i = \widetilde{\mathbf{Sign}}(y, x, m_i, t_i)$ and returns σ_i to $\mathcal{A}$. When $\mathcal{A}$ outputs a forgery $(m^*, \sigma^* = (s_1^*, s_2^*, t^*))$, $\mathcal{B}_I$ outputs $s_2^* \oplus_{j \neq \hat{j}} F_j(K_j, y||t^{*(j)})$ if it is a **type I** forgery such that $t^{*(\hat{j})} = t_{\hat{i}}^{(\hat{j})} \oplus e_{\hat{j}}$ and $t^{*(\hat{j})} \neq t_i^{(\hat{j})}$ for all i. Otherwise, $\mathcal{B}_I$ aborts.

If $\mathcal{B}_I$ does not abort, we have $t^{*(\hat{j})} = t_{\hat{i}}^{(\hat{j})} \oplus e_{\hat{j}}$, which implies $h(s_2^* \oplus_{j \neq \hat{j}} F_j(K_j, y||t^{*(j)})) = h(r)$, since the forgery has to make $\mathbf{Verify}_I$ output 1. In other words, if $\mathcal{B}_I$ does not abort, it breaks the one-wayness of h. Since the view of $\mathcal{A}$ is identical to its view in **Game 5**, the probability that $\mathcal{B}_I$ does not abort is ϵ_5, completing the proof of Lemma 4.3.5. □

Let ϵ_i denote the probability that $\mathcal{A}$ succeeds in **Game** i. The differing-inputs property of $\mathcal{DIO}$, the indistinguishability property of $\mathcal{IO}$, the pseudorandom at punctured point property of $(F_{\hat{j}}, \mathit{Puncture}_{\hat{j}}, \mathit{Eval}_{\hat{j}})$, and the one-wayness of h respectively imply that $|\epsilon_3 - \epsilon_2|$, $|\epsilon_4 - \epsilon_3|$, $|\epsilon_5 - \epsilon_4|$, and ϵ_5 are negligible. Furthermore, since $\epsilon_2 \geq \epsilon_1/(q \cdot k)$, we have that ϵ_1, which is the probability that $\mathcal{A}$ outputs a **type I** forgery in the original FLR security game, is negligible, completing this part of proof.

type II. We give hybrid games to show that $\mathcal{A}$ outputs a **type II** forgery with negligible probability. Note that **Game 1**, $\cdots$, **Game 5** respectively correspond to the games for **type I** forgery, and **Game 2.5** is an extra game in this part of proof.

Game 1. This is the original security game for $\mathcal{A}$. The challenger runs $par \leftarrow \mathsf{Setup}(1^\lambda)$, $(pk, sk) \leftarrow \mathsf{Gen}(par)$, and gives $(par, pk, f(sk))$ to $\mathcal{A}$. When $\mathcal{A}$ makes signing queries, the challenger answers them honestly. At some point, $\mathcal{A}$ outputs the forgery (m^*, σ^*). $\mathcal{A}$ succeeds if (m^*, σ^*) is a **type II** forgery.

Game 2: This game is the same as **Game 1** except that the challenger randomly chooses $\hat{i} \leftarrow \{1, \cdots, q\}$, $\hat{j} \leftarrow \{1, \cdots, \lambda\}$, and $\hat{b} \leftarrow \{0,1\}$ at the beginning of the game. Furthermore, $\mathcal{A}$ succeeds if its forgery (m^*, σ^*) is a **type II** forgery, and $m^*[\hat{j}] = \hat{b} \wedge m_{\hat{i}}[\hat{j}] \neq \hat{b} \wedge t_{\hat{i}} = t^*$.

Lemma 4.3.6. *If $\mathcal{A}$ succeeds with probability ϵ_1 in* **Game 1**, *then we have $\epsilon_2 \geq \epsilon_1/(2 \cdot q \cdot k)$ where ϵ_2 denotes the probability that $\mathcal{A}$ succeeds in* **Game 2**.

Proof of Lemma 4.3.6. Since if $\mathcal{A}$ outputs a **type II** forgery, we have $t^* \in \{t_i\}_{i=1}^q$ and $m^* \neq m_{\hat{i}}$, $(\hat{i}, \hat{j}, \hat{b})$ such that $m^*[\hat{j}] = \hat{b} \wedge m_{\hat{i}}[\hat{j}] \neq \hat{b} \wedge t_{\hat{i}} = t^*$ must exist. Then this lemma

follows from the fact that the view of $\mathcal{A}$ in **Game 2** is identical to its view in **Game 1** and $\mathcal{A}$ learns no information on which $(\hat{i}, \hat{j}, \hat{b})$ is chosen. □

Game 2.5: This game is the same as **Game 2** except that $\mathcal{A}$ is said to be successful if its forgery (m^*, σ^*) is a **type II** forgery such that $m^*[\hat{j}] = \hat{b} \wedge m_{\hat{i}}[\hat{j}] \neq \hat{b} \wedge t_{\hat{i}} = t^*$ and $t_{\hat{i}} \neq t_i$ for all $i \in \{1, \cdots, q\}$.

Lemma 4.3.7. *If $\mathcal{A}$ succeeds with probability ϵ_2 in* **Game 2** *and $\epsilon_{2.5}$ in* **Game 2.5**, *then we have $\epsilon_{2.5} \geq \epsilon_2 - q/2^\lambda$ where ϵ_2 denotes the probability that $\mathcal{A}$ succeeds in* **Game 2**.

Proof of Lemma 4.3.7. This lemma follows from the fact that $\{t_i\}_{i=1}^q$ are randomly chosen from $\{0,1\}^\lambda$. □

Game 3: This game is the same as **Game 2.5** except that the challenger chooses $t_1, \cdots, t_q \leftarrow \{0,1\}^\lambda$ at the beginning of the game, and $\widetilde{\textbf{Sign}}$ is the obfuscation of program $\textbf{Sign}_{II}$ given in Figure 4.8 instead of **Sign**. Here, $\textbf{Sign}_{II}$ is padded so that its size is equal to the size of **Sign** and $\textbf{Sign}_I$, if necessary.

$\textbf{Sign}_{II}$
Constant: $K\{s\}, (K_j)_{j=1}^\lambda$, and s, where $s = y||\hat{j}||\hat{b}||t_{\hat{i}}$.
Input: $\overline{y}, \overline{x}, \overline{m}, \overline{t}$.
If $R_{\mathsf{HR}}(\overline{y}, \overline{x}) = 0$ or $\overline{y}||\hat{j}||\overline{m}[\hat{j}]||\overline{t} = s$, output $\perp$.
Compute $\overline{s}_1 = \oplus_{j=1}^\lambda Eval(K\{s\}, \overline{y}||j||\overline{m}[j]||\overline{t})$.
Compute $\overline{s}_2 = \oplus_{j=1}^\lambda F_j(K_j, \overline{y}||\overline{t}^{(j)})$.
Output $\overline{\sigma} = (\overline{s}_1, \overline{s}_2, \overline{t})$.

Figure 4.8: Program $\textbf{Sign}_{II}$. $\textbf{Sign}_{II}$ is the same as **Sign** except that the constant K is substituted with $(K\{s\}, s)$, $\textbf{Sign}_{II}$ aborts if $\overline{y}||\hat{j}||\overline{m}[\hat{j}]||\overline{t} = s$, and $\overline{s}_1$ is evaluated by computing $\overline{s}_1 = \oplus_{j=1}^\lambda Eval(K\{s\}, \overline{y}||j||\overline{m}[j]||\overline{t})$ instead.

Lemma 4.3.8. *If $\mathcal{A}$ succeeds with probability $\epsilon_{2.5}$ in* **Game 2.5** *and ϵ_3 in* **Game 3**, *then we can construct a differing-inputs sampler* Samp *and an adversary $\mathcal{D}$ that break the differing-inputs property of $\mathcal{DIO}$ with advantage $|\epsilon_3 - \epsilon_{2.5}|$.*

Proof of Lemma 4.3.8. We first give the description of Samp.

Taking as input 1^λ, Samp randomly chooses $t_1, \cdots, t_q \leftarrow \{0,1\}^\lambda$, $K \leftarrow \mathcal{K}$, $K_1 \leftarrow \mathcal{K}_1, \cdots, K_\lambda \leftarrow \mathcal{K}_\lambda$, $\hat{i} \leftarrow \{1, \cdots, q\}$, $\hat{j} \leftarrow \{1, \cdots, \lambda\}$, and $b \leftarrow \{0,1\}$, and runs $(y, x) \leftarrow \mathsf{Gen}_{\mathsf{HR}}(1^\lambda)$. Next it runs $K\{s\} = Puncture(K, s)$ where $s = y||\hat{j}||\hat{b}||t_{\hat{i}}$, generates $(\textbf{Sign}, \textbf{Sign}_{II}, \textbf{Verify})$, and outputs $(\textbf{Sign}, \textbf{Sign}_{II}, \alpha)$ where $\alpha = (f(x), (K_j)_{j=1}^\lambda, K\{s\}, s, (t_i)_{i=1}^q, y, \textbf{Verify}, \hat{i}, \hat{j}, \hat{b})$.

Claim 4.3.2. Samp *is a differing-inputs sampler for $\{\mathcal{C}_\lambda\}$.*

Proof of Claim 4.3.2. If a tuple (y^*, x^*, m^*, t^*) satisfies $\mathbf{Sign}(y^*, x^*, m^*, t^*) \neq \mathbf{Sign}_{II}(y^*, x^*, m^*, t^*)$, it must satisfy $y^* = y \wedge R_{\mathsf{HR}}(y^*, x^*) = 1$ (i.e., $R_{\mathsf{HR}}(y, x^*) = 1$). The reason is that if $R_{\mathsf{HR}}(y^*, x^*) = 0$, **Sign** and $\mathbf{Sign}_{II}$ output $\perp$, and if $R_{\mathsf{HR}}(y^*, x^*) = 1 \wedge y^* \neq y$ (i.e., $R_{\mathsf{HR}}(y^*, x^*) = 1 \wedge y^* || \hat{j} || m^*[j] || t^* \neq s$), they output the same signature.

Now we argue that if there exists a PPT adversary $\mathcal{B}$, taking as input $(\mathbf{Sign}, \mathbf{Sign}_{II}, \alpha)$ generated by Samp, can output x^* such that $R_{\mathsf{HR}}(y, x^*) = 1$ with advantage ϵ, then we can construct an adversary $\mathcal{B}'$ that breaks the security of $(\mathsf{Gen}_{\mathsf{HR}}, R_{\mathsf{HR}})$ with advantage ϵ. The proof is as follows.

Taking as input $(1^\lambda, y, f(x))$ from the ULR-hard relation challenger who runs $(y, x) \leftarrow \mathsf{Gen}_{\mathsf{HR}}(1^\lambda)$, $\mathcal{B}'$ randomly chooses $t_1, \cdots, t_q \leftarrow \{0,1\}^\lambda$, $K \leftarrow \mathcal{K}$, $K_1 \leftarrow \mathcal{K}_1$, $\cdots$, $K_\lambda \leftarrow \mathcal{K}_\lambda$, $\hat{i} \leftarrow \{1, \cdots, q\}$, $\hat{j} \leftarrow \{1, \cdots, \lambda\}$, and $\hat{b} \leftarrow \{0,1\}$, and runs $K\{s\} = \mathit{Puncture}(K, s)$ where $s = y || \hat{j} || \hat{b} || t_{\hat{i}}$. Then it generates $(\mathbf{Sign}, \mathbf{Sign}_{II}, \mathbf{Verify})$ and gives $(1^\lambda, \mathbf{Sign}, \mathbf{Sign}_{II}, \alpha)$ where $\alpha = (f(x), (K_j)_{j=1}^\lambda, K\{s\}, s, (t_i)_{i=1}^q, y, \mathbf{Verify}, \hat{i}, \hat{j}, \hat{b})$, the distribution of which is completely the same as the distribution of the output of Samp, to $\mathcal{B}$. When $\mathcal{B}$ outputs x^*, $\mathcal{B}'$ outputs x^*.

Since the probability that $R_{\mathsf{HR}}(y, x^*) = 1$ is ϵ, we have that $\mathcal{B}'$ breaks the security of $(\mathsf{Gen}_{\mathsf{HR}}, R_{\mathsf{HR}})$ with advantage ϵ. Since $(\mathsf{Gen}_{\mathsf{HR}}, R_{\mathsf{HR}})$ is a ULR-hard relation and f is an uninvertible function, ϵ is negligible. Hence, Samp is a differing-inputs sampler for $\{\mathcal{C}_\lambda\}$, completing the proof of Claim 4.3.2. □

We now give the description of $\mathcal{D}$. Taking as input $(1^\lambda, \widetilde{\mathbf{Sign}}, \alpha)$ where $\widetilde{\mathbf{Sign}} \leftarrow \mathcal{DIO}(1^\lambda, \mathbf{Sign})$ or $\widetilde{\mathbf{Sign}} \leftarrow \mathcal{DIO}(1^\lambda, \mathbf{Sign}_{II})$, $\mathcal{D}$ computes $\widetilde{\mathbf{Verify}} \leftarrow \mathcal{IO}(1^\lambda, \mathbf{Verify})$, gives $(par, pk, f(x))$ where $par = (1^\lambda, \widetilde{\mathbf{Sign}}, \widetilde{\mathbf{Verify}})$ and $pk = y$ to $\mathcal{A}$, and answers the ith signing query m_i by computing $s_{1i} = \oplus_{j=1}^\lambda \mathit{Eval}(K\{s\}, y || j || m_i[j] || t_i)$ and $s_{2i} = \oplus_{j=1}^\lambda F_j(K_j, y || t_i^{(j)})$. Note that $\mathit{Eval}(K\{s\}, y || \hat{j} || \hat{b} || t_{\hat{i}})$ will not be called by $\mathcal{D}$, or we do not have $m_{\hat{i}}[\hat{j}] \neq \hat{b}$ or $t_{\hat{i}} \neq t_i$ for all i. $\mathcal{D}$ outputs 1 if $\mathcal{A}$ outputs a **type II** forgery such that $m^*[\hat{j}] = \hat{b} \wedge m_{\hat{i}}[\hat{j}] \neq \hat{b} \wedge t_{\hat{i}} = t^*$ and $t_{\hat{i}} \neq t_i$ for all $i \in \{1, \cdots, q\}$. Otherwise, $\mathcal{D}$ outputs 0.

If $\widetilde{\mathbf{Sign}}$ is generated as $\widetilde{\mathbf{Sign}} \leftarrow \mathcal{DIO}(1^\lambda, \mathbf{Sign})$, the probability that $\mathcal{D}$ outputs 1 is the same as the probability that $\mathcal{A}$ succeeds in **Game 2.5**. Otherwise, it is the same as the probability that $\mathcal{A}$ succeeds in **Game 3**. Since $\mathcal{A}$ succeeds with probability $\epsilon_{2.5}$ in **Game 2.5** and ϵ_3 in **Game 3**, we have that $(\mathsf{Samp}, \mathcal{D})$ breaks the differing-inputs property of diO with advantage $|\epsilon_3 - \epsilon_{2.5}|$, completing the proof of Lemma 4.3.8. □

Game 4: This game is the same as **Game 3** except that $\widetilde{\mathbf{Verify}}$ is the obfuscation of program $\mathbf{Verify}_{II}$ given in Figure 4.9 instead of **Verify**, where h is an injective one-way function. Here, $\mathbf{Verify}_{II}$ is padded so that its size is equal to the size of **Verify** and $\mathbf{Verify}_I$, if necessary.

Lemma 4.3.9. *If $\mathcal{A}$ succeeds with probability ϵ_3 in* **Game 3** *and ϵ_4 in* **Game 4**, *then we can construct a same-circuits sampler* Samp *and an adversary $\mathcal{D}$ that break the indistinguishability property of $\mathcal{IO}$ with advantage $|\epsilon_4 - \epsilon_3|$.*

Verify$_{II}$
Constant: $K\{s\}, (K_j)_{j=1}^{\lambda}$, $h(F(K,s))$, and s, where $s = y||\hat{j}||\hat{b}||t_{\hat{i}}$.
Input: $\overline{y}, \overline{m}, \overline{\sigma}$.
Parse $\overline{\sigma} = (\overline{s}_1, \overline{s}_2, \overline{t})$.
If $\overline{y}||\hat{j}||\overline{m}[\hat{j}]||\overline{t} = s$:
  If $h(\overline{s}_1 \oplus_{j\neq\hat{j}} Eval(K\{s\}, \overline{y}||j||\overline{m}[j]||\overline{t})) = h(F(K,s))$ and $\overline{s}_2 = \oplus_{j=1}^{\lambda} F_j(K_j, \overline{y}||\overline{t}^{(j)})$, output 1.
  Else, output 0.
Else:
  If $\overline{s}_1 = \oplus_{j=1}^{\lambda} Eval(K\{s\}, \overline{y}||j||\overline{m}[j]||\overline{t})$ and $\overline{s}_2 = \oplus_{j=1}^{\lambda} F_j(K_j, \overline{y}||\overline{t}^{(j)})$, output 1.
  Else, output 0.

Figure 4.9: Program **Verify**$_{II}$ in **Game 4**. **Verify**$_{II}$ is the same as **Verify** except that the constant K is substituted with $(K\{s\}, h(F(K,s)), s)$, and $\overline{s}_1$ is verified by checking $h(\overline{s}_1 \oplus_{j\neq\hat{j}} Eval(K\{s\}, \overline{y}||j||\overline{m}[j]||\overline{t})) = h(F(K,s))$ if $\overline{y}||\hat{j}||\overline{m}[\hat{j}]||\overline{t} = s$ and $\overline{s}_1 = \oplus_{j=1}^{\lambda} Eval(K\{s\}, \overline{y}||j||\overline{m}[j]||\overline{t})$ if $\overline{y}||\hat{j}||\overline{m}[\hat{j}]||\overline{t} \neq s$.

Proof of Lemma 4.3.9. We show how to construct Samp and $\mathcal{D}$ as follows.

Taking as input 1^λ, Samp randomly chooses $t_1, \cdots, t_q \leftarrow \{0,1\}^\lambda$, $K \leftarrow \mathcal{K}$, $K_1 \leftarrow \mathcal{K}_1$, $\cdots$, $K_\lambda \leftarrow \mathcal{K}_\lambda$, $\hat{i} \leftarrow \{1, \cdots, q\}$, $\hat{j} \leftarrow \{1, \cdots, \lambda\}$, and $\hat{b} \leftarrow \{0,1\}$, and runs $(y, x) \leftarrow \mathsf{Gen}_{\mathsf{HR}}(1^\lambda)$. Next it runs $K\{s\} = Puncture(K, s)$ where $s = y||\hat{j}||\hat{b}||t_{\hat{i}}$, generates $(\mathbf{Verify}, \mathbf{Verify}_{II}, \mathbf{Sign}_{II})$, and outputs $(\mathbf{Verify}, \mathbf{Verify}_{II}, \alpha)$ where $\alpha = (x, (t_i)_{i=1}^q, y, \mathbf{Sign}_{II}, \hat{i}, \hat{j}, \hat{b})$. Since $\overline{s}_1 = \oplus_{j=1}^{\lambda} F(K, \overline{y}||j||\overline{m}[j]||\overline{t})$ is equivalent to $\overline{s}_1 \oplus_{j\neq\hat{j}} Eval(K\{s\}, \overline{y}||j||\overline{m}[j]||\overline{t}) = F(K,s)$ when $\overline{y}||\hat{j}||\overline{m}[\hat{j}]||\overline{t} = s$, h is injective, and $\overline{s}_1 = \oplus_{j=1}^{\lambda} Eval(K\{s\}, \overline{y}||j||\overline{m}[j]||\overline{t})$ is equivalent to $\overline{s}_1 = \oplus_{j=1}^{\lambda} F(K, \overline{y}||j||\overline{m}[j]||\overline{t})$ when $\overline{y}||\hat{j}||\overline{m}[\hat{j}]||\overline{t} \neq s$, the functionality of **Verify** is equivalent to **Verify**$_{II}$. Hence, Samp is a same-circuits sampler for $\{\mathcal{C}'_\lambda\}$.

Taking as input $(1^\lambda, \widetilde{\mathbf{Verify}}, \alpha)$ where $\widetilde{\mathbf{Verify}} \leftarrow \mathcal{IO}(1^\lambda, \mathbf{Verify})$ or $\widetilde{\mathbf{Verify}} \leftarrow \mathcal{IO}(1^\lambda, \mathbf{Verify}_{II})$, $\mathcal{D}$ computes $\widetilde{\mathbf{Sign}} \leftarrow \mathcal{DIO}(1^\lambda, \mathbf{Sign}_{II})$, gives $(par, pk, f(x))$ where $par = (1^\lambda, \widetilde{\mathbf{Sign}}, \widetilde{\mathbf{Verify}})$ and $pk = y$ to $\mathcal{A}$, and answers the ith signing query by returning σ_i where $\sigma_i = \widetilde{\mathbf{Sign}}(y, x, m_i, t_i)$. $\mathcal{D}$ outputs 1 if $\mathcal{A}$ outputs a **type II** forgery such that $m^*[\hat{j}] = \hat{b} \wedge m_{\hat{i}}[\hat{j}] \neq \hat{b} \wedge t_{\hat{i}} = t^*$ and $t_{\hat{i}} \neq t_i$ for all $i \in \{1, \cdots, q\}$. Otherwise, $\mathcal{D}$ outputs 0.

If $\widetilde{\mathbf{Verify}}$ is generated as $\widetilde{\mathbf{Verify}} \leftarrow \mathcal{IO}(1^\lambda, \mathbf{Verify})$, then $\mathcal{A}$ is in **Game 3**. Otherwise, it is in **Game 4**. Since $\mathcal{A}$ succeeds with probability ϵ_3 in **Game 3** and ϵ_4 in **Game 4**, we have that $(\mathsf{Samp}, \mathcal{D})$ breaks the indistinguishability of iO with advantage $|\epsilon_4 - \epsilon_3|$, completing the proof of Lemma 4.3.9. □

Game 5: This game is the same as **Game 4** except that the constant $h(F(K,s))$ is substituted with $h(r)$ where r is a randomness chosen from $\{0,1\}^\lambda$, i.e., **Verify**$_{II}$ is generated as shown in Figure 4.10.

Verify$_{II}$
Constant: $K\{s\}, (K_j)_{j=1}^{\lambda}$, $h(r)$ and s, where $s = y||\hat{j}||\hat{b}||t_{\hat{i}}$.
Input: $\overline{y}, \overline{m}, \overline{\sigma}$.
Parse $\overline{\sigma} = (\overline{s}_1, \overline{s}_2, \overline{t})$.
If $\overline{y}||\hat{j}||\overline{m}[\hat{j}]||\overline{t} = s$:
  If $h(\overline{s}_1 \oplus_{j \neq \hat{j}} Eval(K\{s\}, \overline{y}||j||\overline{m}[j]||\overline{t})) = h(r)$ and $\overline{s}_2 = \oplus_{j=1}^{\lambda} F_j(K_j, \overline{y}||\overline{t}^{(j)})$, output 1.
  Else, output 0.
Else:
  If $\overline{s}_1 = \oplus_{j=1}^{\lambda} Eval(K\{s\}, \overline{y}||j||\overline{m}[j]||\overline{t})$ and $\overline{s}_2 = \oplus_{j=1}^{\lambda} F_j(K_j, \overline{y}||\overline{t}^{(j)})$, output 1.
  Else, output 0.

Figure 4.10: Program **Verify**$_{II}$ in **game 6**. This **Verify**$_{II}$ is the same as the one in **Game 5**, except that $h(F(K, s))$ is substituted with $h(r)$ where r is a randomness.

Lemma 4.3.10. *If $\mathcal{A}$ succeeds with probability ϵ_4 in* **Game 4** *and ϵ_5 in* **Game 5**, *then we can construct an adversary $(\mathcal{B}_1, \mathcal{B}_2)$ that breaks the pseudorandom at punctured point property of $(F, Puncture, Eval)$ with advantage $|\epsilon_5 - \epsilon_4|$.*

Proof of Lemma 4.3.10. We construct $(\mathcal{B}_1, \mathcal{B}_2)$ as follows.

$\mathcal{B}_1$ randomly chooses $t_1, \cdots, t_q \leftarrow \{0,1\}^{\lambda}$, $K_1 \leftarrow \mathcal{K}_1$, $\cdots$, $K_\lambda \leftarrow \mathcal{K}_\lambda$, $\hat{i} \leftarrow \{1, \cdots, q\}$, $\hat{j} \leftarrow \{1, \cdots, \lambda\}$, and $\hat{b} \leftarrow \{0,1\}$, and runs $(y, x) \leftarrow \mathsf{Gen}_{\mathsf{HR}}(1^\lambda)$. Then it sets s as $s = y||\hat{j}||\hat{b}||t_{\hat{i}}$, and outputs s along with an auxiliary input α which contains the information of all the elements sampled by $\mathcal{B}_1$.

Taking as input $(K\{s\}, F(K, s), \alpha)$ (or $(K\{s\}, r, \alpha)$) where $K \leftarrow \mathcal{K}$, $r \leftarrow \{0,1\}^\lambda$, and $K\{s\} = Puncture(K, s)$ from the challenger of $(\mathcal{B}_1, \mathcal{B}_2)$, $\mathcal{B}_2$ generates **Sign**$_{II}$ and **Verify**$_{II}$ with $h(F(K, s))$ or $h(r)$, and runs $\widetilde{\textbf{Sign}} \leftarrow \mathcal{DIO}(1^\lambda, \textbf{Sign}_{II})$ and $\widetilde{\textbf{Verify}} \leftarrow \mathcal{DIO}(1^\lambda, \textbf{Verify}_{II})$. Next $\mathcal{B}_2$ gives $(par, pk, f(x))$, where $par = (1^\lambda, \widetilde{\textbf{Sign}}, \widetilde{\textbf{Verify}})$ and $pk = y$, to $\mathcal{A}$. When receiving the ith signing query m_i from $\mathcal{A}$, $\mathcal{B}_2$ runs $\sigma_i = \widetilde{\textbf{Sign}}(y, x, m_i, t_i)$ and returns σ_i to $\mathcal{A}$. If $\mathcal{A}$ outputs a **type II** forgery and $m^*[\hat{j}] = \hat{b} \wedge m_{\hat{i}}[\hat{j}] \neq \hat{b} \wedge t_{\hat{i}} = t^*$ and $t_{\hat{i}} \neq t_i$ for all $i \in \{1, \cdots, q\}$, $\mathcal{B}_2$ outputs 1. Otherwise, $\mathcal{B}_2$ outputs 0.

When the input of $\mathcal{B}_2$ is $(K\{s\}, F(K, s), \alpha)$, we have that $\mathcal{A}$ is in **Game 4**, i.e., $\mathcal{B}_2$ outputs 1 with probability ϵ_4. Otherwise, $\mathcal{A}$ is in **Game 5**, i.e., $\mathcal{B}_2$ outputs 1 with probability ϵ_5. Since $(\mathcal{B}_1, \mathcal{B}_2)$ breaks the pseudorandom at punctured point property of the puncturable PRF $(F, Puncture, Eval)$ with advantage $|\epsilon_5 - \epsilon_4|$, we have that $|\epsilon_5 - \epsilon_4|$ is negligible, completing the proof of Lemma 4.3.10. □

Lemma 4.3.11. *If $\mathcal{A}$ succeeds with probability ϵ_5 in* **Game 5**, *then we can construct a PPT adversary $\mathcal{B}_{II}$ that breaks the one-wayness of h with advantage ϵ_5.*

Proof of Lemma 4.3.11. Given $h(r)$ from the challenger who randomly chooses $r \leftarrow \{0,1\}^\lambda$, $\mathcal{B}_{II}$ randomly chooses $t_1, \cdots, t_q \leftarrow \{0,1\}^\lambda$, $K \leftarrow \mathcal{K}$, $K_1 \leftarrow \mathcal{K}_1, \cdots, K_\lambda \leftarrow \mathcal{K}_\lambda$, $\hat{i} \leftarrow \{1, \cdots, q\}$, and $\hat{j} \leftarrow \{1, \cdots, \lambda\}$, runs $(y, x) \leftarrow \mathsf{Gen}_{\mathsf{HR}}(1^\lambda)$, and computes $K\{s\} = Puncture(K, s)$ where $s = y||\hat{j}||\hat{b}||t_{\hat{i}}$. Then it generates **Sign**$_{II}$ (see Figure 4.8) and **Verify**$_{II}$

(see Figure 4.10), and runs $\widetilde{\mathbf{Sign}} \leftarrow \mathcal{DIO}(1^\lambda, \mathbf{Sign}_{II})$ and $\widetilde{\mathbf{Verify}} \leftarrow \mathcal{IO}(1^\lambda, \mathbf{Verify}_{II})$. Next $\mathcal{B}_{II}$ gives $(par, pk, f(x))$, where $par = (1^\lambda, \widetilde{\mathbf{Sign}}, \widetilde{\mathbf{Verify}})$ and $pk = y$, to $\mathcal{A}$. When receiving the ith signing query m_i from $\mathcal{A}$, $\mathcal{B}_{II}$ runs $\sigma_i = \widetilde{\mathbf{Sign}}(y, x, m_i, t_i)$ and returns σ_i to $\mathcal{A}$. When $\mathcal{A}$ outputs a forgery $(m^*, \sigma^* = (s_1^*, s_2^*, t^*))$, $\mathcal{B}_{II}$ computes $s_1^* \oplus_{j \neq \hat{j}} Eval(K\{s\}, y||j||m^*[j]||t^*)$ if it is a **type II** forgery such that $m^*[\hat{j}] = \hat{b} \wedge m_{\hat{i}}[\hat{j}] \neq \hat{b} \wedge t_{\hat{i}} = t^*$ and $t_{\hat{i}} \neq t_i$ for all $i \in \{1, \cdots, q\}$. Otherwise, $\mathcal{B}_{II}$ aborts.

If $\mathcal{B}_{II}$ does not abort, we have $m^*[\hat{j}] = \hat{b} \wedge t_{\hat{i}} = t^*$, which implies $h(s_1^* \oplus_{j \neq \hat{j}} Eval(K\{s\}, y||j||m^*[j]||t^*)) = h(r)$, since the forgery has to make $\mathbf{Verify}_{II}$ output 1. In other words, if $\mathcal{B}_{II}$ does not abort, it breaks the one-wayness of h. Since the view of $\mathcal{A}$ is identical to its view in **Game 5**, the probability that $\mathcal{B}_{II}$ does not abort is ϵ_5, completing the proof of Lemma 4.3.11. □

Let ϵ_i denote the probability that $\mathcal{A}$ succeeds in **Game** i. The differing-inputs property of $\mathcal{DIO}$, the indistinguishability property of $\mathcal{IO}$, the pseudorandom at punctured point property of $(F, Puncture, Eval)$, and the one-wayness of h respectively imply that $|\epsilon_3 - \epsilon_{2.5}|$, $|\epsilon_4 - \epsilon_3|$, $|\epsilon_5 - \epsilon_4|$, and ϵ_5 are negligible. Furthermore, since $\epsilon_{2.5} \geq \epsilon_2 - q/2^\lambda$ and $\epsilon_2 \geq \epsilon_1/(2 \cdot q \cdot \lambda)$, we have that ϵ_1, which is the probability that $\mathcal{A}$ outputs a **type II** forgery in the original FLR security game, is negligible, completing this part of proof.

In conclusion, $\mathcal{A}$ breaks the FLR security of our scheme with only negligible advantage, completing the proof of Theorem 4.3.1. □

4.3.2 Weak Fully Leakage Resilient Signature Scheme

If we substitute the ULR-hard relation with a k-IULR-hard relation in the FLR signature scheme $(\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ in Section 4.3.1, we immediately obtain a k-weak FLR signature scheme in the selective auxiliary input model, as described in the following theorem.

Theorem 4.3.2. *Let $\mathcal{C}_\lambda$ denote a family of circuits whose size is equal to the size of* **Sign** *and $\mathcal{C}'_\lambda$ a family of circuits whose size is equal to the size of* **Verify**. *If $(\mathsf{Gen}_{\mathsf{HR}}, R_{\mathsf{HR}})$ is a k-IULR-hard relation, $\mathcal{DIO}$ is diO for $\mathcal{C}_\lambda$, $\mathcal{IO}$ is iO for $\mathcal{C}'_\lambda$, $(F, Puncture, Eval)$ and $\{(F_j, Puncture_j, Eval_j)\}_{j=1}^\lambda$ are puncturable PRFs, and there exists an injective one-way function $h : \{0,1\}^\lambda \to \{0,1\}^*$, then $(\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ is a k-weak FLR signature scheme in the selective auxiliary input model.*

We omit the proof of Theorem 4.3.2 since it is the same as the proof of Theorem 4.3.1 except that the uninvertible (leakage) function is substituted with an injective uninvertible one, the size of which is upper bounded by k.

4.3.3 Remarks on Our Constructions

In this section, we give several remarks on our proposed signature schemes as follows.

Strong existential unforgeability. In the presence of leakage, our proposed signature schemes also satisfy a stronger security notion called strong existential unforgeability against chosen message attacks (sEUF-CMA), which guarantees that $\mathcal{A}$ is not able to come up with a successful forgery pair (m^*, σ^*) as long as $(m^*, \sigma^*) \neq (m_i, \sigma_i)$ for all i (where m_i denotes the ith signing query and σ_i the answer for m_i). The reason is that if $\mathcal{A}$ outputs a successful forgery such that $t^* = t_i$ for some i (where t^* and t_i respectively denote the randomizers contained in σ^* and σ_i), m^* must be different from m_i, or σ^* (which is determined by m^* and t^*) must be the same as σ_i (which is determined by m_i and t_i). As a result, we have $t^* \notin \{t_1, \cdots, t_q\}$, or $t^* = t_i$ for some i but $m^*[j] \neq m_i[j]$ for some j. Therefore, the proof for the EUF-CMA security can be directly applied to the proof for the sEUF-CMA security.

Hash-and-sign. By exploiting collision resistant hash functions, we can extend the message spaces of our signature schemes to the full domain. Furthermore, like the short signature scheme in [97], we can make use of one puncturable PRF with variable-length domain instead of k puncturable PRFs to generate s_2 (see Figure 4.2).

Variant of leakage function family. Our proposed FLR signature scheme will remain secure if we let the leakage function take as input the state including the public randomizers $(t_1, \cdots, t_i)$ as the definitions of the previous FLR signature [34, 61], while the restriction becomes that given $(f(sk, t_1 || \cdots || t_i), t_1, \cdots, t_i)$, an adversary cannot recover sk except with negligible probability. The reason is that the distribution of $((f(sk, t_1 || \cdots || t_i), t_1, \cdots, t_i), sk)$ is unpredictable and $t_1, \cdots, t_i$ are public coins.

Selective unforgeability. As we mentioned in the outline of the security proof, we can achieve a selectively unforgeable signature scheme if we let **Sign** output $F(K, \overline{y} || \overline{m})$ instead. In this case, there will be only one input that leads **Sign**$'$,[6] which is the signing program used in the hybrid game of the security proof, and **Sign** to different outputs. This means that **Sign** and **Sign**$'$ are indistinguishable when they are obfuscated by iO, according to the result by Boyle et al. [32]. As a result, we can achieve an FLR signature scheme with selective unforgeability, in the selective auxiliary input model, based on iO and AIPO, without using diO.

[6] This algorithm is described in the "Outline of the security proof" paragraph in Section 4.3.1.

Chapter 5

Conclusion and Open Problems

In Chapter 3, we have formalized TSs and (TT-)AKSs, and shown how to convert (TT-)SPSs into SKSP-TSs and SP-(TT-)AKSs. By combining SKSP-TSs with SP-(TT-)AKSs (or SP-BTCs), we have obtained generic constructions of FSPS and FAS, which help us obtain many instantiations of FSPS and FAS with various advantages. As extensions of the EGM paradigm, our generic constructions are of independent interest. There are several open problems left by this part of work. Firstly, the signature of our most efficient UF-CMA secure FSPS scheme based on standard assumptions is still longer than $4n$ when signing a message consisting of n^2 elements. It remains open how to make the signature size close to $3n$ or even shorter. Secondly, it would be interesting to prove the existence of a non-trivial lower bound on the signature size of FSPSs, regardless the underlying assumptions. Notice that the lower bound given in [10] is on the sum of the signature size and the verification key size. Thirdly, it is desirable to find more concrete applications of FSPSs.

In Chapter 4, we have formalized and constructed ULR-hard relation and IULR-hard relation. Based on a ULR-hard relation (respectively, an IULR-hard relation) and diO, we have proposed an signature scheme secure against uninvertible leakage (respectively, injective uninvertible leakage). It would be interesting to find whether it is possible to achieve signatures with such security under standard assumptions.

Chapter 6

Acknowledgement

I would like to show my deepest gratitude to my supervisor Prof. Keisuke Tanaka for his constant guidance, support, and encouragement. He gave me invaluable suggestions not only on my study but also on my career. I consider myself extremely lucky to have been his student for these years.

I am grateful to Dr. Takahiro Matsuda for the insightful advices and timely feedbacks throughout the thesis. I was continually amazed by his rigorous attitude, which helped to improve the thesis and my scientific writing.

I would also like to thank Dr. Zongyang Zhang for the helpful discussions, the effort in improving the thesis, and also his friendship.

I appreciate Dr. Goichiro Hanaoka for the useful comments on the thesis. He provided me the chance to work in his group, and also supported me in many ways during my doctor course.

Special thanks also to all the members of Tanaka Laboratory and Advanced Cryptosystems Research Group of AIST. They created a friendly atmosphere for working and having fun.

At last, I must express my gratitude to my parents for their unwavering support and continuous encouragement.

Bibliography

[1] 51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA. IEEE Computer Society (2010), http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=5669376

[2] Abe, M.: Variations of Even-Goldreich-Micali framework for signature schemes. IEICE Transactions 100-A(1), 12–17 (2017), http://search.ieice.org/bin/summary.php?id=e100-a_1_12

[3] Abe, M., Chase, M., David, B., Kohlweiss, M., Nishimaki, R., Ohkubo, M.: Constant-size structure-preserving signatures: Generic constructions and simple assumptions. In: Wang and Sako [106], pp. 4–24, https://doi.org/10.1007/978-3-642-34961-4_3

[4] Abe, M., David, B., Kohlweiss, M., Nishimaki, R., Ohkubo, M.: Tagged one-time signatures: Tight security and optimal tag size. In: Kurosawa, K., Hanaoka, G. (eds.) Public-Key Cryptography - PKC 2013 - 16th International Conference on Practice and Theory in Public-Key Cryptography, Nara, Japan, February 26 - March 1, 2013. Proceedings. Lecture Notes in Computer Science, vol. 7778, pp. 312–331. Springer (2013), https://doi.org/10.1007/978-3-642-36362-7_20

[5] Abe, M., Fuchsbauer, G., Groth, J., Haralambiev, K., Ohkubo, M.: Structure-preserving signatures and commitments to group elements. In: Rabin [96], pp. 209–236, https://doi.org/10.1007/978-3-642-14623-7_12

[6] Abe, M., Groth, J., Haralambiev, K., Ohkubo, M.: Optimal structure-preserving signatures in asymmetric bilinear groups. In: Rogaway [99], pp. 649–666, https://doi.org/10.1007/978-3-642-22792-9_37

[7] Abe, M., Groth, J., Ohkubo, M.: Separating short structure-preserving signatures from non-interactive assumptions. In: Lee and Wang [84], pp. 628–646, https://doi.org/10.1007/978-3-642-25385-0_34

[8] Abe, M., Groth, J., Ohkubo, M., Tibouchi, M.: Structure-preserving signatures from type II pairings. In: Garay and Gennaro [58], pp. 390–407, https://doi.org/10.1007/978-3-662-44371-2_22

[9] Abe, M., Groth, J., Ohkubo, M., Tibouchi, M.: Unified, minimal and selectively randomizable structure-preserving signatures. In: Lindell [88], pp. 688–712, https://doi.org/10.1007/978-3-642-54242-8_29

[10] Abe, M., Kohlweiss, M., Ohkubo, M., Tibouchi, M.: Fully structure-preserving signatures and shrinking commitments. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II. Lecture Notes in Computer Science, vol. 9057, pp. 35–65. Springer (2015), https://doi.org/10.1007/978-3-662-46803-6_2

[11] Akavia, A., Goldwasser, S., Vaikuntanathan, V.: Simultaneous hardcore bits and cryptography against memory attacks. In: Reingold, O. (ed.) Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009, San Francisco, CA, USA, March 15-17, 2009. Proceedings. Lecture Notes in Computer Science, vol. 5444, pp. 474–495. Springer (2009), https://doi.org/10.1007/978-3-642-00457-5_28

[12] Ananth, P., Boneh, D., Garg, S., Sahai, A., Zhandry, M.: Differing-inputs obfuscation and applications. IACR Cryptology ePrint Archive 2013, 689 (2013), http://eprint.iacr.org/2013/689

[13] Ananth, P., Jain, A., Naor, M., Sahai, A., Yogev, E.: Universal obfuscation and witness encryption: Boosting correctness and combining security. IACR Cryptology ePrint Archive 2016, 281 (2016), http://eprint.iacr.org/2016/281

[14] Anderson, R., Kuhn, M.: Tamper resistance: A cautionary note. In: Proceedings of the 2Nd Conference on Proceedings of the Second USENIX Workshop on Electronic Commerce - Volume 2. pp. 1–1. WOEC'96, USENIX Association, Berkeley, CA, USA (1996), http://dl.acm.org/citation.cfm?id=1267167.1267168

[15] Attrapadung, N., Libert, B., Peters, T.: Computing on authenticated data: New privacy definitions and constructions. In: Wang and Sako [106], pp. 367–385, https://doi.org/10.1007/978-3-642-34961-4_23

[16] Baldimtsi, F., Chase, M., Fuchsbauer, G., Kohlweiss, M.: Anonymous transferable e-cash. In: Katz, J. (ed.) Public-Key Cryptography - PKC 2015 - 18th IACR International Conference on Practice and Theory in Public-Key Cryptography, Gaithersburg, MD, USA, March 30 - April 1, 2015, Proceedings. Lecture Notes in Computer Science, vol. 9020, pp. 101–124. Springer (2015), https://doi.org/10.1007/978-3-662-46447-2_5

[17] Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S.P., Yang, K.: On the (im)possibility of obfuscating programs. vol. 59, pp. 6:1–6:48 (2012), http://doi.acm.org/10.1145/2160158.2160159

[18] Belenkiy, M., Camenisch, J., Chase, M., Kohlweiss, M., Lysyanskaya, A., Shacham, H.: Randomizable proofs and delegatable anonymous credentials. In: Halevi [72], pp. 108–125, https://doi.org/10.1007/978-3-642-03356-8_7

[19] Belenkiy, M., Chase, M., Kohlweiss, M., Lysyanskaya, A.: P-signatures and non-interactive anonymous credentials. In: Canetti, R. (ed.) Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008. Lecture Notes in Computer Science, vol. 4948, pp. 356–374. Springer (2008), https://doi.org/10.1007/978-3-540-78524-8_20

[20] Bellare, M., Hoang, V.T., Keelveedhi, S.: Instantiating random oracles via uces. In: Canetti and Garay [43], pp. 398–415, https://doi.org/10.1007/978-3-642-40084-1_23

[21] Bellare, M., Micciancio, D., Warinschi, B.: Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In: Biham, E. (ed.) Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings. Lecture Notes in Computer Science, vol. 2656, pp. 614–629. Springer (2003), https://doi.org/10.1007/3-540-39200-9_38

[22] Bellare, M., Shi, H., Zhang, C.: Foundations of group signatures: The case of dynamic groups. In: Menezes, A. (ed.) Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3376, pp. 136–153. Springer (2005), https://doi.org/10.1007/978-3-540-30574-3_11

[23] Bellare, M., Shoup, S.: Two-tier signatures, strongly unforgeable signatures, and Fiat-Shamir without random oracles. In: Okamoto, T., Wang, X. (eds.) Public Key Cryptography - PKC 2007, 10th International Conference on Practice and Theory in Public-Key Cryptography, Beijing, China, April 16-20, 2007, Proceedings. Lecture Notes in Computer Science, vol. 4450, pp. 201–216. Springer (2007), https://doi.org/10.1007/978-3-540-71677-8_14

[24] Bellare, M., Stepanovs, I.: Point-function obfuscation: A framework and generic constructions. In: Kushilevitz, E., Malkin, T. (eds.) Theory of Cryptography - 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part II. Lecture Notes in Computer Science, vol. 9563, pp. 565–594. Springer (2016), https://doi.org/10.1007/978-3-662-49099-0_21

[25] Bellare, M., Stepanovs, I., Tessaro, S.: Poly-many hardcore bits for any one-way function and a framework for differing-inputs obfuscation. In: Sarkar and Iwata [102], pp. 102–121, https://doi.org/10.1007/978-3-662-45608-8_6

[26] Bellare, M., Stepanovs, I., Waters, B.: New negative results on differing-inputs obfuscation 9666, 792–821 (2016), https://doi.org/10.1007/978-3-662-49896-5_28

[27] Bender, A., Katz, J., Morselli, R.: Ring signatures: Stronger definitions, and constructions without random oracles. J. Cryptology 22(1), 114–138 (2009), https://doi.org/10.1007/s00145-007-9011-9

[28] Biham, E., Shamir, A.: Differential fault analysis of secret key cryptosystems. In: Jr. [76], pp. 513–525, https://doi.org/10.1007/BFb0052259

[29] Bitansky, N., Paneth, O.: Point obfuscation and 3-round zero-knowledge. In: Cramer, R. (ed.) Theory of Cryptography - 9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, March 19-21, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7194, pp. 190–208. Springer (2012), https://doi.org/10.1007/978-3-642-28914-9_11

[30] Boneh, D., DeMillo, R.A., Lipton, R.J.: On the importance of checking cryptographic protocols for faults (extended abstract). In: Fumy, W. (ed.) Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding. Lecture Notes in Computer Science, vol. 1233, pp. 37–51. Springer (1997), https://doi.org/10.1007/3-540-69053-0_4

[31] Boneh, D., Waters, B.: Constrained pseudorandom functions and their applications. In: Sako, K., Sarkar, P. (eds.) Advances in Cryptology - ASIACRYPT 2013 - 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part II. Lecture Notes in Computer Science, vol. 8270, pp. 280–300. Springer (2013), https://doi.org/10.1007/978-3-642-42045-0_15

[32] Boyle, E., Chung, K., Pass, R.: On extractability obfuscation. In: Lindell [88], pp. 52–73, https://doi.org/10.1007/978-3-642-54242-8_3

[33] Boyle, E., Pass, R.: Limits of extractability assumptions with distributional auxiliary input. In: Iwata and Cheon [75], pp. 236–261, https://doi.org/10.1007/978-3-662-48800-3_10

[34] Boyle, E., Segev, G., Wichs, D.: Fully leakage-resilient signatures. In: Paterson [93], pp. 89–108, https://doi.org/10.1007/978-3-642-20465-4_7

[35] Brakerski, Z., Goldwasser, S.: Circular and leakage resilient public-key encryption under subgroup indistinguishability - (or: Quadratic residuosity strikes back). In: Rabin [96], pp. 1–20, https://doi.org/10.1007/978-3-642-14623-7_1

[36] Brakerski, Z., Kalai, Y.T., Katz, J., Vaikuntanathan, V.: Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage. In: 51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA [1], pp. 501–510, https://doi.org/10.1109/FOCS.2010.55

[37] Brzuska, C., Mittelbach, A.: Indistinguishability obfuscation versus multi-bit point obfuscation with auxiliary input. In: Sarkar and Iwata [102], pp. 142–161, https://doi.org/10.1007/978-3-662-45608-8_8

[38] Camenisch, J., Dubovitskaya, M., Enderlein, R.R., Neven, G.: Oblivious transfer with hidden access control from attribute-based encryption. In: Visconti, I., Prisco, R.D. (eds.) Security and Cryptography for Networks - 8th International Conference, SCN 2012, Amalfi, Italy, September 5-7, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7485, pp. 559–579. Springer (2012), https://doi.org/10.1007/978-3-642-32928-9_31

[39] Camenisch, J., Dubovitskaya, M., Haralambiev, K., Kohlweiss, M.: Composable and modular anonymous credentials: Definitions and practical constructions. In: Iwata and Cheon [75], pp. 262–288, https://doi.org/10.1007/978-3-662-48800-3_11

[40] Camenisch, J., Krenn, S., Shoup, V.: A framework for practical universally composable zero-knowledge protocols. In: Lee and Wang [84], pp. 449–467, https://doi.org/10.1007/978-3-642-25385-0_24

[41] Camenisch, J., Lysyanskaya, A.: An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In: Pfitzmann, B. (ed.) Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceeding. Lecture Notes in Computer Science, vol. 2045, pp. 93–118. Springer (2001), https://doi.org/10.1007/3-540-44987-6_7

[42] Canetti, R.: Towards realizing random oracles: Hash functions that hide all partial information. In: Jr. [76], pp. 455–469, https://doi.org/10.1007/BFb0052255

[43] Canetti, R., Garay, J.A. (eds.): Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II, Lecture Notes in Computer Science, vol. 8043. Springer (2013), https://doi.org/10.1007/978-3-642-40084-1

[44] Chase, M., Kohlweiss, M., Lysyanskaya, A., Meiklejohn, S.: Malleable proof systems and applications. In: Pointcheval and Johansson [94], pp. 281–300, https://doi.org/10.1007/978-3-642-29011-4_18

[45] Chase, M., Kohlweiss, M., Lysyanskaya, A., Meiklejohn, S.: Malleable signatures: New definitions and delegatable anonymous credentials. In: IEEE 27th Computer Security Foundations Symposium, CSF 2014, Vienna, Austria, 19-22 July, 2014. pp. 199–213. IEEE Computer Society (2014), https://doi.org/10.1109/CSF.2014.22

[46] Dodis, Y., Goldwasser, S., Kalai, Y.T., Peikert, C., Vaikuntanathan, V.: Public-key encryption schemes with auxiliary inputs. In: Micciancio, D. (ed.) Theory of Cryptography, 7th Theory of Cryptography Conference, TCC 2010, Zurich, Switzerland,

February 9-11, 2010. Proceedings. Lecture Notes in Computer Science, vol. 5978, pp. 361–381. Springer (2010), https://doi.org/10.1007/978-3-642-11799-2_22

[47] Dodis, Y., Haralambiev, K., López-Alt, A., Wichs, D.: Cryptography against continuous memory attacks. In: 51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA [1], pp. 511–520, https://doi.org/10.1109/FOCS.2010.56

[48] Dodis, Y., Kalai, Y.T., Lovett, S.: On cryptography with auxiliary input. In: Mitzenmacher, M. (ed.) Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009. pp. 621–630. ACM (2009), http://doi.acm.org/10.1145/1536414.1536498

[49] ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans. Information Theory 31(4), 469–472 (1985), https://doi.org/10.1109/TIT.1985.1057074

[50] Escala, A., Herold, G., Kiltz, E., Ràfols, C., Villar, J.L.: An algebraic framework for Diffie-Hellman assumptions. In: Canetti and Garay [43], pp. 129–147, https://doi.org/10.1007/978-3-642-40084-1_8

[51] Even, S., Goldreich, O., Micali, S.: On-line/off-line digital signatures. J. Cryptology 9(1), 35–67 (1996), https://doi.org/10.1007/BF02254791

[52] Faust, S., Hazay, C., Nielsen, J.B., Nordholt, P.S., Zottarel, A.: Signature schemes secure against hard-to-invert leakage. In: Wang and Sako [106], pp. 98–115, https://doi.org/10.1007/978-3-642-34961-4_8

[53] Fuchsbauer, G.: Commuting signatures and verifiable encryption. In: Paterson [93], pp. 224–245, https://doi.org/10.1007/978-3-642-20465-4_14

[54] Fuchsbauer, G., Hanser, C., Kamath, C., Slamanig, D.: Practical round-optimal blind signatures in the standard model from weaker assumptions. In: Zikas, V., Prisco, R.D. (eds.) Security and Cryptography for Networks - 10th International Conference, SCN 2016, Amalfi, Italy, August 31 - September 2, 2016, Proceedings. Lecture Notes in Computer Science, vol. 9841, pp. 391–408. Springer (2016), https://doi.org/10.1007/978-3-319-44618-9_21

[55] Fuchsbauer, G., Hanser, C., Slamanig, D.: Practical round-optimal blind signatures in the standard model. In: Gennaro and Robshaw [62], pp. 233–253, https://doi.org/10.1007/978-3-662-48000-7_12

[56] Fuchsbauer, G., Vergnaud, D.: Fair blind signatures without random oracles. In: Bernstein, D.J., Lange, T. (eds.) Progress in Cryptology - AFRICACRYPT 2010, Third International Conference on Cryptology in Africa, Stellenbosch, South Africa, May 3-6, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6055, pp. 16–33. Springer (2010), https://doi.org/10.1007/978-3-642-12678-9_2

[57] Gandolfi, K., Mourtel, C., Olivier, F.: Electromagnetic analysis: Concrete results. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings. Lecture Notes in Computer Science, vol. 2162, pp. 251–261. Springer (2001), https://doi.org/10.1007/3-540-44709-1_21

[58] Garay, J.A., Gennaro, R. (eds.): Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I, Lecture Notes in Computer Science, vol. 8616. Springer (2014), https://doi.org/10.1007/978-3-662-44371-2

[59] Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: 54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA. pp. 40–49. IEEE Computer Society (2013), https://doi.org/10.1109/FOCS.2013.13

[60] Garg, S., Gentry, C., Halevi, S., Wichs, D.: On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. In: Garay and Gennaro [58], pp. 518–535, https://doi.org/10.1007/978-3-662-44371-2_29

[61] Garg, S., Jain, A., Sahai, A.: Leakage-resilient zero knowledge. In: Rogaway [99], pp. 297–315, https://doi.org/10.1007/978-3-642-22792-9_17

[62] Gennaro, R., Robshaw, M. (eds.): Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II, Lecture Notes in Computer Science, vol. 9216. Springer (2015), https://doi.org/10.1007/978-3-662-48000-7

[63] Ghadafi, E.: Formalizing group blind signatures and practical constructions without random oracles. In: Boyd, C., Simpson, L. (eds.) Information Security and Privacy - 18th Australasian Conference, ACISP 2013, Brisbane, Australia, July 1-3, 2013. Proceedings. Lecture Notes in Computer Science, vol. 7959, pp. 330–346. Springer (2013), https://doi.org/10.1007/978-3-642-39059-3_23

[64] Ghadafi, E.: Short structure-preserving signatures. In: Sako, K. (ed.) Topics in Cryptology - CT-RSA 2016 - The Cryptographers' Track at the RSA Conference 2016, San Francisco, CA, USA, February 29 - March 4, 2016, Proceedings. Lecture Notes in Computer Science, vol. 9610, pp. 305–321. Springer (2016), https://doi.org/10.1007/978-3-319-29485-8_18

[65] Ghadafi, E.: How low can you go? short structure-preserving signatures for Diffie-Hellman vectors. In: O'Neill, M. (ed.) Cryptography and Coding - 16th IMA International Conference, IMACC 2017, Oxford, UK, December 12-14, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10655, pp. 185–204. Springer (2017), https://doi.org/10.1007/978-3-319-71045-7_10

[66] Ghadafi, E.: More efficient structure-preserving signatures - or: Bypassing the type-III lower bounds. In: Foley, S.N., Gollmann, D., Snekkenes, E. (eds.) Computer Security - ESORICS 2017 - 22nd European Symposium on Research in Computer Security, Oslo, Norway, September 11-15, 2017, Proceedings, Part II. Lecture Notes in Computer Science, vol. 10493, pp. 43–61. Springer (2017), https://doi.org/10.1007/978-3-319-66399-9_3

[67] Goldreich, O., Levin, L.A.: A hard-core predicate for all one-way functions. In: Johnson, D.S. (ed.) Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washigton, USA. pp. 25–32. ACM (1989), http://doi.acm.org/10.1145/73007.73010

[68] Goldwasser, S., Kalai, Y.T., Peikert, C., Vaikuntanathan, V.: Robustness of the learning with errors assumption. In: Yao, A.C. (ed.) Innovations in Computer Science - ICS 2010, Tsinghua University, Beijing, China, January 5-7, 2010. Proceedings. pp. 230–240. Tsinghua University Press (2010), http://conference.itcs.tsinghua.edu.cn/ICS2010/content/papers/19.html

[69] Groth, J.: Efficient fully structure-preserving signatures for large messages. In: Iwata, T., Cheon, J.H. (eds.) Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9452, pp. 239–259. Springer (2015), https://doi.org/10.1007/978-3-662-48797-6_11

[70] Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. SIAM J. Comput. 41(5), 1193–1232 (2012), https://doi.org/10.1137/080725386

[71] Halderman, J.A., Schoen, S.D., Heninger, N., Clarkson, W., Paul, W., Calandrino, J.A., Feldman, A.J., Appelbaum, J., Felten, E.W.: Lest we remember: Cold boot attacks on encryption keys. In: van Oorschot, P.C. (ed.) Proceedings of the 17th USENIX Security Symposium, July 28-August 1, 2008, San Jose, CA, USA. pp. 45–60. USENIX Association (2008), http://www.usenix.org/events/sec08/tech/full_papers/halderman/halderman.pdf

[72] Halevi, S. (ed.): Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings, Lecture Notes in Computer Science, vol. 5677. Springer (2009), https://doi.org/10.1007/978-3-642-03356-8

[73] Hofheinz, D., Jager, T.: Tightly secure signatures and public-key encryption. In: Safavi-Naini and Canetti [100], pp. 590–607, https://doi.org/10.1007/978-3-642-32009-5_35

[74] Ishai, Y., Pandey, O., Sahai, A.: Public-coin differing-inputs obfuscation and its applications. In: Dodis, Y., Nielsen, J.B. (eds.) Theory of Cryptography - 12th Theory

of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part II. Lecture Notes in Computer Science, vol. 9015, pp. 668–697. Springer (2015), https://doi.org/10.1007/978-3-662-46497-7_26

[75] Iwata, T., Cheon, J.H. (eds.): Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II, Lecture Notes in Computer Science, vol. 9453. Springer (2015), https://doi.org/10.1007/978-3-662-48800-3

[76] Jr., B.S.K. (ed.): Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings, Lecture Notes in Computer Science, vol. 1294. Springer (1997), https://doi.org/10.1007/BFb0052223

[77] Jutla, C.S., Roy, A.: Improved structure preserving signatures under standard bilinear assumptions. In: Fehr, S. (ed.) Public-Key Cryptography - PKC 2017 - 20th IACR International Conference on Practice and Theory in Public-Key Cryptography, Amsterdam, The Netherlands, March 28-31, 2017, Proceedings, Part II. Lecture Notes in Computer Science, vol. 10175, pp. 183–209. Springer (2017), https://doi.org/10.1007/978-3-662-54388-7_7

[78] Katz, J., Vaikuntanathan, V.: Signature schemes with bounded leakage resilience. In: Matsui, M. (ed.) Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings. Lecture Notes in Computer Science, vol. 5912, pp. 703–720. Springer (2009), https://doi.org/10.1007/978-3-642-10366-7_41

[79] Kiltz, E., Pan, J., Wee, H.: Structure-preserving signatures from standard assumptions, revisited. In: Gennaro and Robshaw [62], pp. 275–295, https://doi.org/10.1007/978-3-662-48000-7_14

[80] Kiltz, E., Pan, J., Wee, H.: Structure-preserving signatures from standard assumptions, revisited. IACR Cryptology ePrint Archive 2015, 604 (2015), http://eprint.iacr.org/2015/604

[81] Kocher, P.C.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings. Lecture Notes in Computer Science, vol. 1109, pp. 104–113. Springer (1996), https://doi.org/10.1007/3-540-68697-5_9

[82] Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M.J. (ed.) Advances in Cryptology - CRYPTO '99, 19th Annual International Cryp-

tology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings. Lecture Notes in Computer Science, vol. 1666, pp. 388–397. Springer (1999), https://doi.org/10.1007/3-540-48405-1_25

[83] Komargodski, I.: Leakage resilient one-way functions: The auxiliary-input setting. In: Hirt, M., Smith, A.D. (eds.) Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9985, pp. 139–158 (2016), https://doi.org/10.1007/978-3-662-53641-4_6

[84] Lee, D.H., Wang, X. (eds.): Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings, Lecture Notes in Computer Science, vol. 7073. Springer (2011), https://doi.org/10.1007/978-3-642-25385-0

[85] Libert, B., Peters, T., Joye, M., Yung, M.: Linearly homomorphic structure-preserving signatures and their applications. In: Canetti and Garay [43], pp. 289–307, https://doi.org/10.1007/978-3-642-40084-1_17

[86] Libert, B., Peters, T., Yung, M.: Group signatures with almost-for-free revocation. In: Safavi-Naini and Canetti [100], pp. 571–589, https://doi.org/10.1007/978-3-642-32009-5_34

[87] Libert, B., Peters, T., Yung, M.: Short group signatures via structure-preserving signatures: Standard model security from simple assumptions. In: Gennaro and Robshaw [62], pp. 296–316, https://doi.org/10.1007/978-3-662-48000-7_15

[88] Lindell, Y. (ed.): Theory of Cryptography - 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA, February 24-26, 2014. Proceedings, Lecture Notes in Computer Science, vol. 8349. Springer (2014), https://doi.org/10.1007/978-3-642-54242-8

[89] Lynn, B., Prabhakaran, M., Sahai, A.: Positive results and techniques for obfuscation. In: Cachin, C., Camenisch, J. (eds.) Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings. Lecture Notes in Computer Science, vol. 3027, pp. 20–39. Springer (2004), https://doi.org/10.1007/978-3-540-24676-3_2

[90] Micali, S., Ohta, K., Reyzin, L.: Accountable-subgroup multisignatures: extended abstract. In: Reiter, M.K., Samarati, P. (eds.) CCS 2001, Proceedings of the 8th ACM Conference on Computer and Communications Security, Philadelphia, Pennsylvania, USA, November 6-8, 2001. pp. 245–254. ACM (2001), http://doi.acm.org/10.1145/501983.502017

[91] Naor, M., Segev, G.: Public-key cryptosystems resilient to key leakage. In: Halevi [72], pp. 18–35, https://doi.org/10.1007/978-3-642-03356-8_2

[92] Osvik, D.A., Shamir, A., Tromer, E.: Cache attacks and countermeasures: The case of AES. In: Pointcheval, D. (ed.) Topics in Cryptology - CT-RSA 2006, The Cryptographers' Track at the RSA Conference 2006, San Jose, CA, USA, February 13-17, 2006, Proceedings. Lecture Notes in Computer Science, vol. 3860, pp. 1–20. Springer (2006), https://doi.org/10.1007/11605805_1

[93] Paterson, K.G. (ed.): Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings, Lecture Notes in Computer Science, vol. 6632. Springer (2011), https://doi.org/10.1007/978-3-642-20465-4

[94] Pointcheval, D., Johansson, T. (eds.): Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings, Lecture Notes in Computer Science, vol. 7237. Springer (2012), https://doi.org/10.1007/978-3-642-29011-4

[95] Quisquater, J., Samyde, D.: Electromagnetic analysis (EMA): measures and countermeasures for smart cards. In: Attali, I., Jensen, T.P. (eds.) Smart Card Programming and Security, International Conference on Research in Smart Cards, E-smart 2001, Cannes, France, September 19-21, 2001, Proceedings. Lecture Notes in Computer Science, vol. 2140, pp. 200–210. Springer (2001), https://doi.org/10.1007/3-540-45418-7_17

[96] Rabin, T. (ed.): Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings, Lecture Notes in Computer Science, vol. 6223. Springer (2010), https://doi.org/10.1007/978-3-642-14623-7

[97] Ramchen, K., Waters, B.: Fully secure and fast signing from obfuscation. In: Ahn, G., Yung, M., Li, N. (eds.) Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014. pp. 659–673. ACM (2014), http://doi.acm.org/10.1145/2660267.2660306

[98] Rial, A., Kohlweiss, M., Preneel, B.: Universally composable adaptive priced oblivious transfer. In: Shacham, H., Waters, B. (eds.) Pairing-Based Cryptography - Pairing 2009, Third International Conference, Palo Alto, CA, USA, August 12-14, 2009, Proceedings. Lecture Notes in Computer Science, vol. 5671, pp. 231–247. Springer (2009), https://doi.org/10.1007/978-3-642-03298-1_15

[99] Rogaway, P. (ed.): Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings, Lecture

Notes in Computer Science, vol. 6841. Springer (2011), https://doi.org/10.1007/978-3-642-22792-9

[100] Safavi-Naini, R., Canetti, R. (eds.): Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings, Lecture Notes in Computer Science, vol. 7417. Springer (2012), https://doi.org/10.1007/978-3-642-32009-5

[101] Sahai, A., Waters, B.: How to use indistinguishability obfuscation: deniable encryption, and more. In: Shmoys, D.B. (ed.) Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014. pp. 475–484. ACM (2014), http://doi.acm.org/10.1145/2591796.2591825

[102] Sarkar, P., Iwata, T. (eds.): Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014, Proceedings, Part II, Lecture Notes in Computer Science, vol. 8874. Springer (2014), https://doi.org/10.1007/978-3-662-45608-8

[103] Schwartz, J.T.: Fast probabilistic algorithms for verification of polynomial identities. J. ACM 27(4), 701–717 (1980), http://doi.acm.org/10.1145/322217.322225

[104] Standaert, F.X.: Leakage resilient cryptography: a practical overview. In: invited talk, SKEW 2011 (2011)

[105] Verheul, E.R.: Evidence that XTR is more secure than supersingular elliptic curve cryptosystems. J. Cryptology 17(4), 277–296 (2004), https://doi.org/10.1007/s00145-004-0313-x

[106] Wang, X., Sako, K. (eds.): Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings, Lecture Notes in Computer Science, vol. 7658. Springer (2012), https://doi.org/10.1007/978-3-642-34961-4

[107] Wee, H.: On obfuscating point functions. In: Gabow, H.N., Fagin, R. (eds.) Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005. pp. 523–532. ACM (2005), http://doi.acm.org/10.1145/1060590.1060669

[108] Yu, Z., Xu, Q., Zhou, Y., Hu, C., Yang, R., Fan, G.: Weak-key leakage resilient cryptography. IACR Cryptology ePrint Archive 2014, 159 (2014), http://eprint.iacr.org/2014/159

[109] Yuen, T.H., Chow, S.S.M., Zhang, Y., Yiu, S.: Identity-based encryption resilient to continual auxiliary leakage. In: Pointcheval and Johansson [94], pp. 117–134, https://doi.org/10.1007/978-3-642-29011-4_9

[110] Yuen, T.H., Yiu, S., Hui, L.C.K.: Fully leakage-resilient signatures with auxiliary inputs. In: Susilo, W., Mu, Y., Seberry, J. (eds.) Information Security and Privacy - 17th Australasian Conference, ACISP 2012, Wollongong, NSW, Australia, July 9-11, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7372, pp. 294–307. Springer (2012), https://doi.org/10.1007/978-3-642-31448-3_22

[111] Yuen, T.H., Zhang, Y., Yiu, S.: Encryption schemes with post-challenge auxiliary inputs. IACR Cryptology ePrint Archive 2013, 323 (2013), http://eprint.iacr.org/2013/323