The Generic Transformation from Standard Signatures to Identity-Based Aggregate Signatures

Bei Liang[1,2,3(✉)], Hongda Li[1,2,3], and Jinyong Chang[1,3]

[1] State Key Laboratory of Information Security,
Institute of Information Engineering of Chinese Academy of Sciences,
Beijing, China
{liangbei,lihongda,changjinyong}@iie.ac.cn
[2] Data Assurance and Communication Security Research Center
of Chinese Academy of Sciences, Beijing, China
[3] University of Chinese Academy of Sciences, Beijing, China

Abstract. Aggregate signature system allows a collection of signatures can be compressed into one short signature. Identity-based signature schemes (IBS) allow a signer to sign a message, in which the signature can be verified by his identity. The notion of identity-based aggregate signatures (IBAS) were formally introduced by Gentry and Ramzan (PKC'06). Over the past decade, several constructions of IBAS have been proposed, which are restricted to share a common token or require sequential additions. The problem about how to achieve IBAS from standard signatures still is not resolved.

In this work, we present a generic transformation that yields IBAS schemes starting with standard signature schemes. Specifically, we provide a generic construction of an n-bounded IBAS scheme that can be proven selectively secure in the standard model from any secure signature scheme by using indistinguishability obfuscation and selective one-time universal parameters scheme. The complexity leveraging requires sub-exponential hardness assumption of indistinguishability obfuscation, puncturable PRF and one-way functions.

Keywords: Aggregate signature · Identity-based signature · Identity-based aggregate signature · Indistinguishability obfuscation · Universal parameters.

1 Introduction

Aggregate signatures, as introduced by Boneh et al. [2], are digital signatures that allow n users (whose verification and secret signing key pair is

This research is supported by the Strategy Pilot Project of Chinese Academy of Sciences (Grant No. Y2W0012203).

J. Lopez and C.J. Mitchell (Eds.): ISC 2015, LNCS 9290, pp. 21–41, 2015.
DOI: 10.1007/978-3-319-23318-5_2

$\{(vk_i, sk_i)\}_{i\in[n]}$) of a given group of potential signers to sign n different messages $\{m_i\}_{i\in[n]}$ respectively, and all the signatures of those users on those messages can be aggregated into a single short signature σ. This single signature σ and the n original verification key/message pairs $\{(vk_i, m_i)\}_{i\in[n]}$ are enough to convince the verifier that the n signers did indeed sign the n original messages m_i respectively. Aggregate signatures are useful in many real-world applications where one needs to simultaneously verify several signatures from different signers on different messages in environments with communication or storage resource constraints, such as secure route attestation.

Identity-Based Aggregate Signatures. In 1984, Shamir proposed a new model for public key cryptography, the identity-based cryptography and constructed an identity-based signature scheme (IBS) [11]. The idea of identity based cryptography is to simplify the public key of the user by using user's identity, which uniquely defines the user. In an identity based signature scheme, each user is provided with a secret signing key corresponding to his identity and he/she signs their messages using the secret signing key. The signature can be verified by using the identity of the signer and public parameters of the system.

The features of an identity-based signature scheme make it particularly appealing for use in conjunction with aggregate signature schemes. Gentry and Ramzan first formally introduced the notion of identity-based aggregate signatures (IBAS) and corresponding security model [7]. In an identity-based aggregate signature scheme, a trusted private key generator generates a private signing key sk_{id} corresponding to user's identity id. Using private signing key sk_{id} user can obtain a signature σ_{id} for message corresponding to identity id. Furthermore a signature σ_1 on identity/message pair (id_1, m_1) can be combined with a signature σ_2 on (id_2, m_2) to produce a new signature $\tilde{\sigma}$ on the set $\{id_1, m_1), (id_2, m_2)\}$. Crucially, the size of aggregated signature $\tilde{\sigma}$ should be independent of the number of signatures aggregated. The aggregated signature $\tilde{\sigma}$ can be verified by using the identity/message pair of the signer and public parameters of the system. The system will be secure in the sense that it is hard to produce an aggregate signature on a identity/message list L that contains some (id_i, m_i) never queried before — i.e., for all the adversary's queries L', $(id_i, m_i) \notin L'$.

Current State of the Art. Standard signatures imply identity-based signatures following the "certification paradigm", e.g. [6] , i.e. by simply attaching signer's public key and certificate to each signature. However, it is not clear how to convert standard signatures into identity-based aggregate signatures. Although over the past decade many identity-based aggregate signature schemes have been proposed [3,4,7,10], all of these constructions are restricted to share a common token [7] (e.g., where a set of signatures can only be aggregated if they were created with the same common token) or require sequential additions [3] (e.g., where a group of signers sequentially form an aggregate by each adding their own signature to the aggregate-so-far).

In 2013, Hohenberger, Sahai and Waters [10] implemented the Full Domain Hash with a Naor-Reingold-type structure that is publicly computable by using leveled multilinear maps. And departing from this result they constructed the

first identity-based aggregate signature scheme that admits unrestricted aggregation. However, since their solution to identity-based aggregate signature scheme is firstly to build a BLS type-signature that admit unrestricted aggregation, the problem about how to achieve identity-based aggregate signatures from standard signatures still is not resolved.

Our Results in a Nutshell. In this work, we present a generic transformation that yields identity-based aggregate signature schemes based on standard signature schemes. Specifically, we provide a generic construction of an n-bounded identity-based aggregate signature scheme that can be proven selectively secure in the standard model from any secure signature scheme by using indistinguishability obfuscation and selective one-time universal parameters scheme. Although our IBAS scheme requires an a-priori bound n on the number of signatures that can be aggregated, the size of the public parameters and aggregated signatures are independent of it.

Before we describe our construction we briefly overview the underlying primitive: universal parameters scheme. Intuitively, a universal parameters (UP) scheme allows multiple paries to sample a consistent elements from arbitrary distributions while insuring that an adversary cannot learn the randomness that yields this element. In UP there is a universal parameter generation algorithm, $\mathsf{UniversalGen}$, which takes as input a security parameter and output "universal parameters" U. In addition, there is a second algorithm $\mathsf{InduceGen}$ which takes as input universal parameters U and a distribution specified by a circuit d, and outputs the induced parameters $d(z)$ for hidden random coins z that are pseudorandomly derived from U and d. The security definition states that it is computationally difficult to distinguish an honest execution of U from that generated by a simulator $\mathsf{SimUGen}$ that has access to the parameters oracle.

To transform standard signature scheme into identity-based aggregate signature, we proceed in two steps. In the first step, we show how to obtain IBS from standard signature scheme (SIG). The basic idea is to use one signature instance for each identity id of IBS by universal parameter, which is inspired by the application of universal parameter for transforming public key encryption (PKE) into identity-based encryption (IBE) [8]. Precisely, choose a universal parameter U and a key pair $(pk_{\mathsf{PKE}}, sk_{\mathsf{PKE}})$ of PKE. Let $\mathsf{Prog}\{pk_{\mathsf{PKE}}\}$ be a circuit that taking a random string $r = r_1 \| r_2$ as input, first samples $(vk_{\mathsf{SIG}}, sk_{\mathsf{SIG}}) \leftarrow \mathsf{SIG.Setup}(1^\lambda; r_1)$, then encrypts sk_{SIG} under pk_{PKE} via $c' \leftarrow \mathsf{PKE.Enc}(pk_{\mathsf{PKE}}, sk_{\mathsf{SIG}}; r_2)$, and finally outputs (vk_{SIG}, c'). Here we view pk_{PKE} as a constant hardwired into the circuit $\mathsf{Prog}\{pk_{\mathsf{PKE}}\}$ and $r = r_1 \| r_2$ as input, where we make the random coins of the $\mathsf{SIG.Setup}$ and $\mathsf{PKE.Enc}$ explicit. For identity id we compute $(vk_{id}, c'_{id}) \leftarrow \mathsf{InduceGen}(U, \mathsf{Prog}\{pk_{\mathsf{PKE}}\} \| id)$. This way, we can use sk_{PKE} as a master trapdoor to extract the signing key sk_{id} from c'_{id} and thus obtain individual user secret signing key for identity id. Using that secret signing key sk_{id} corresponding to id, user can sign their messages.

The second step is to make this IBS support aggregation. Our main solution idea departs fundamentally from Hohenberger, Koppula and Waters's method of aggregating signatures using indistinguishability obfuscation [9]. Basing on

their idea we provide an approach to make this IBS support aggregation. Our construction relies on the puncturable PRF and additively homomorphic encryption HE.

Our IBAS scheme is setup as follows. Randomly choose a key K of puncturable PRF and a key pair $(pk_{\mathsf{HE}}, sk_{\mathsf{HE}})$ of HE and obtain n encryption of 0 under pk_{HE}, e.g. $\mathsf{ct}_i \leftarrow \mathsf{HE.Enc}(pk_{\mathsf{HE}}, 0)$. Create program $\mathsf{Prog}\{K, \mathsf{ct}_1, \ldots, \mathsf{ct}_n\}$ which taking as inputs $\{vk_{id_i}, (id_i, m_i), \sigma_i\}_{i\in[n]}$, firstly verifies (m_i, σ_i) is valid under verification key vk_{id_i}, then computes $t = \sigma_1 \cdot \mathsf{ct}_1 + \ldots + \sigma_n \cdot \mathsf{ct}_n$ and $s_i = F(K, vk_{id_i} \| id_i \| m_i \| i \| t)$, and finally outputs $\sigma_{\mathsf{agg}} = (t, \oplus_i s_i)$. In addition, create program $\mathsf{Prog}\{K\}$ that taking as inputs $\{vk_{id_i}, (id_i, m_i)\}_i$ and $\sigma_{\mathsf{agg}} = (t, s)$, computes $s' = \oplus_i F(K, vk_{id_i} \| id_i \| m_i \| i \| t)$ and outputs 1 if $s' = s$, else outputs 0. Set obfuscated programs $P_1 = i\mathcal{O}\big(\mathsf{Prog}\{K, \mathsf{ct}_1, \ldots, \mathsf{ct}_n\}\big)$ and $P_2 = i\mathcal{O}(\mathsf{Prog}\{K\})$ as public parameters. To aggregate the signatures σ_i of identity/message pair (id_i, m_i) for all $i \in [n]$, firstly obtain verification key vk_{id_i} corresponding identity id_i by universal parameter U and run program $P_1\big(\{vk_{id_i}, (id_i, m_i), \sigma_i\}_i\big)$ to get $\sigma_{\mathsf{agg}} = (t, s)$. To verify an aggregate signature, $\sigma_{\mathsf{agg}} = (t, s)$, on $\{(id_i, m_i)\}_{i\in[n]}$, firstly obtain verification key vk_{id_i} corresponding identity id_i by universal parameter U and return the output of $P_2(\{vk_{id_i}, (id_i, m_i)\}_i, \sigma_{\mathsf{agg}})$.

We prove the selective security where the attacker declares before seeing the public parameters a idenntity/message pair (id^*, m^*) by performing a sequence of games. In game 1 challenger first guesses an index i^* (incurring a $1/n$ loss) where the forgery occurs. In game 2, we change ct_{i^*} to be an encryption of 1. This causes an honestly computed value t to be an encryption of the i^*-th signature that we will eventually use for extraction. In game 3 we use the programmed generated algorithm $\mathsf{SimUGen}$ to produce U such that $(vk_{id_{i^*}}, c_{id_{i^*}}) \leftarrow \mathsf{InduceGen}(U, \mathsf{Prog}\{pk_{\mathsf{PKE}}\} \| id_{i^*})$ where $c_{id_{i^*}} = \mathsf{PKE.Enc}(pk_{\mathsf{PKE}}, sk_{id_{i^*}})$. In game 4 we replace $c_{id_{i^*}}$ with an encryption of 1^λ. At this time the simulator cannot answer the $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$ and $\mathsf{Sign}(\cdot, \cdot)$ queries, since it cannot decrypt the ciphertext c_{id_i}. We overcome this obstacle by employing a wCCA-secure PKE that requires that the attacker has access to decryption oracle only after seeing the challenge ciphertext. When using wCCA-secure PKE, simulator can use the wCCA decryption oracle to answer $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$ and $\mathsf{Sign}(\cdot, \cdot)$ queries. For forgery $\sigma^*_{\mathsf{agg}} = (t^*, s^*)$, since t^* is an encryption of σ_{i^*} under sk_{HE}, if $\mathsf{SIG.Vefy}(vk_{id_{i^*}}, m_{i^*}, \mathsf{HE.Dec}(sk_{\mathsf{HE}}, t^*)) = 1$, then $(m_{i^*}, \mathsf{HE.Dec}(sk_{\mathsf{HE}}, t^*))$ is a forgery for basic signature scheme SIG, which contradicts with the existential unforgeability of SIG. Therefore when $\mathsf{SIG.Vefy}(vk_{id_{i^*}}, m_{i^*}, \mathsf{HE.Dec}(sk_{\mathsf{HE}}, t^*)) = 0$, we can use the punctured key $K\{y\}$ at punctured point $y = vk_{id_{i^*}} \| id_{i^*} \| m_{i^*} \| i^* \| t^*$ to replace program $\mathsf{Prog}\{K, \mathsf{ct}_1, \ldots, \mathsf{ct}_n\}$ with $\mathsf{Prog}\{K\{y\}, \mathsf{ct}_1, \ldots, \mathsf{ct}_n\}$. In addition, we replace $i\mathcal{O}\big(\mathsf{Prog}\{K\}\big)$ with $i\mathcal{O}\big(\mathsf{Prog}\{y, z = F(K, y), K\{y\}\}\big)$, where program $\mathsf{Prog}\{y, z = F(K, y), K\{y\}\}$ employs an one-way function to check the correctness for $F(K, vk_{id_{i^*}} \| id_{i^*} \| m_{i^*} \| i^* \| t^*)$ and $F(K\{y\}, vk_{id_{i^*}} \| id_{i^*} \| m_{i^*} \| i^* \| t)$ that in turn can be computed by $\oplus_{i\neq i^*} F(K\{y\}, vk_{id_i} \| id_i \| m_i \| i \| t) \oplus s$. By the

pseudorandomness property of puncturable PRF we replace $F(K, y)$ with a random strings z. This perfectly simulates the game.

Since there will be an exponential number of intermediate hybrid games, we will be using stronger security for the indistinguishability obfuscation, the puncturable PRF and the one way function, which requires sub-exponential hardness assumption.

Organization. The rest of this paper is organized as follows. In Sect. 2 we describe the basic tools which will be used in our construction. In Sect. 3 we introduce the notions of identity-based aggregate signatures that are considered in this work. In Sect. 4 we present our generic transformation to build IBAS from standard signature scheme and prove the security of our IBAS scheme.

2 Preliminaries

In this section, we give the definitions of cryptographic primitives that will be used in our constructions. Below, we recall the notions of indistinguishability obfuscation, puncturable pseudorandom functions and universal parameters.

2.1 Indistinguishability Obfuscation

Here we recall the notion of indistinguishability obfuscation which was originally proposed by Barak et al. [1]. The formal definition we present below is from [5].

Definition 1 (Indistinguishability Obfuscation [5]). *A PPT algorithm $i\mathcal{O}$ is said to be an indistinguishability obfuscator for a circuits class $\{\mathcal{C}_\lambda\}$, if the following conditions are satisfied:*

- *For all security parameters $\lambda \in \mathbb{N}$, for all $C \in C_\lambda$, for all inputs x, we have that*
$$\Pr[C'(x) = C(x) : C' \leftarrow i\mathcal{O}(\lambda, C)] = 1.$$
- *For any (not necessarily uniform) PPT adversaries (Samp, D), there exists a negligible function $negl(\cdot)$ such that the following holds: if $\Pr[\forall x, C_0(x) = C_1(x) : (C_0, C_1, \sigma) \leftarrow \mathsf{Samp}(1^\lambda)] > 1 - negl(\lambda)$, then we have:*
$$\begin{aligned}&\big|\Pr[D(\sigma, i\mathcal{O}(\lambda, C_0)) = 1 : (C_0, C_1, \sigma) \leftarrow \mathsf{Samp}(1^\lambda)]\\ &\quad -\Pr[D(\sigma, i\mathcal{O}(\lambda, C_1)) = 1 : (C_0, C_1, \sigma) \leftarrow \mathsf{Samp}(1^\lambda)]\big| \leq negl(\lambda).\end{aligned}$$

In a recent work, Garg et al. [5] gave the first candidate construction of indistinguishability obfuscator $i\mathcal{O}$ for all polynomial size circuits under novel algebraic hardness assumptions. In this paper, we will take advantage of such indistinguishability obfuscators for all polynomial size circuits.

2.2 Puncturable PRFs

Puncturable PRFs, as introduced by Sahai and Waters [12], are PRFs that a punctured key can be derived to allow evaluation of the PRF on all inputs, except for any polynomial-size set of inputs. The definition is formulated as in [12].

Definition 2. *A puncturable family of PRFs F mapping is given by a triple of Turing Machines (Key_F, $\mathsf{Puncture}_F$, and Eval_F), and a pair of computable functions $\tau_1(\cdot)$ and $\tau_2(\cdot)$, satisfying the following conditions:*

- ***[Functionality preserved under puncturing].*** *For every PPT adversary $\mathcal{A}$ such that $\mathcal{A}(1^\lambda)$ outputs a set $S \subseteq \{0,1\}^{\tau_1(\lambda)}$, then for all $x \in \{0,1\}^{\tau_1(\lambda)}$ where $x \notin S$, we have that:*

$$\Pr[\mathsf{Eval}_F(K,x)=\mathsf{Eval}_F(K_S,x) : K \leftarrow \mathsf{Key}_F(1^\lambda), K_S=\mathsf{Puncture}_F(K,S)]=1$$

- ***[Pseudorandom at punctured points].*** *For every PPT adversary $(\mathcal{A}_1, \mathcal{A}_2)$ such that $\mathcal{A}_1(1^\lambda)$ outputs a set $S \subseteq \{0,1\}^{\tau_1(\lambda)}$ and state σ, consider an experiment where $K \leftarrow \mathsf{Key}_F(1^\lambda)$ and $K_S = \mathsf{Puncture}_F(K,S)$. Then we have*

$$\left|\Pr[\mathcal{A}_2(\sigma, K_S, S, \mathsf{Eval}_F(K,S)) = 1] - \Pr[\mathcal{A}_2(\sigma, K_S, S, U_{\tau_2(\lambda)\cdot|S|}) = 1]\right| = negl(\lambda),$$

where $\mathsf{Eval}_F(K,S)$ denotes the concatenation of $\mathsf{Eval}_F(K,x_1),\ldots,\mathsf{Eval}_F(K,x_k)$ where $S=\{x_1,\ldots,x_k\}$ is the enumeration of the elements of S in lexicographic order, $negl(\cdot)$ is a negligible function, and $U_{\tau_2(\lambda)\cdot|S|}$ denotes the uniform distribution over $\tau_2(\lambda)\cdot|S|$ bits.

Theorem 1 [12]. *If one-way functions exist, then for all efficiently computable functions $\tau_1(\lambda)$ and $\tau_2(\lambda)$, there exists a puncturable PRFs family that maps $\tau_1(\lambda)$ bits to $\tau_2(\lambda)$ bits.*

2.3 Universal Parameters

In a recent work, Hofheinz et al. [8] introduced the notion of universal parameters. A universal parameters scheme UP, parameterized by polynomials ℓ_{ckt}, ℓ_{inp} and ℓ_{out}, consists of algorithms $\mathsf{UniversalGen}$ and $\mathsf{InduceGen}$ defined below.

- $\mathsf{UniversalGen}(1^\lambda)$ takes as input the security parameter λ and outputs the universal parameters U.
- $\mathsf{InduceGen}(U,d)$ takes as input the universal parameters U and a circuit d which takes as input ℓ_{inp} bits and outputs ℓ_{out} bits. The size of circuit d is at most ℓ_{ckt} bits.

Definition 3 (Selectively-Secure One-Time Universal Parameters Scheme). *Let ℓ_{ckt}, ℓ_{inp}, ℓ_{out} be efficiently computable polynomials. A pair of efficient algorithms ($\mathsf{UniversalGen}$, $\mathsf{InduceGen}$) is a selectively-secure one-time universal parameters scheme if there exists an efficient algorithm $\mathsf{SimUGen}$ such that:*

– *There exists a negligible function* $negl(\cdot)$ *such that for all circuits* d *of length* ℓ_{ckt}*, taking* ℓ_{inp} *bits of input, and outputting* ℓ_{out} *bits, and for all strings* $p_d \in \{0,1\}^k$*, we have that:*

$$\Pr[\mathsf{InduceGen}(\mathsf{SimUGen}(1^\lambda, d, p_d), d) = p_d] = 1 - negl(\lambda).$$

– *For every efficient adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, where* $\mathcal{A}_2$ *outputs one bit, there exists a negligible function* $negl(\cdot)$ *such that the following holds. Consider the following two experiments:*

The experiment $\mathsf{Real}(1^\lambda)$ *is as follows:*

1. $(d^*, \sigma) \leftarrow \mathcal{A}_1(1^\lambda)$
2. *Output* $\mathcal{A}_2(\mathsf{UniversalGen}(1^\lambda), \sigma)$.

The experiment $\mathsf{Ideal}(1^\lambda)$ *is as follows:*

1. $(d^*, \sigma) \leftarrow \mathcal{A}_1(1^\lambda)$
2. *Choose* r *uniformly from* $\{0,1\}^{\ell_{inp}}$.
3. *Let* $p_d = d^*(r)$.
4. *Output* $\mathcal{A}_2(\mathsf{SimUGen}(1^\lambda, d^*, p_d), \sigma)$.

Then we have:

$$\left|\Pr[\mathsf{Real}(1^\lambda) = 1] - \Pr[\mathsf{Ideal}(1^\lambda) = 1]\right| = negl(\lambda).$$

Hofheinz et al. [9] construct a selectively secure one-time universal parameters scheme, assuming a secure indistinguishability obfuscator and a selectively secure puncturable PRF.

3 Identity-Based Aggregate Signatures

Syntax. An identity-based aggregate signatures (IBAS) scheme can be described as a tuple of polynomial time algorithms $\mathsf{IBAS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Aggregate}, \mathsf{Verify})$ as follows:

– $\mathsf{Setup}(1^\lambda)$ The setup algorithm takes as input the security parameter and outputs the public parameters PP of the scheme and master secret key msk.
– $\mathsf{KeyGen}(\mathsf{msk}, id \in \{0,1\}^{\ell_{\mathsf{id}}})$ The key generation algorithm run by the master entity, takes as input the master secret key msk and an identity id, and outputs a secret signing key sk_{id} corresponding to id.
– $\mathsf{Sign}(\mathsf{sk}_{id}, m \in \{0,1\}^{\ell_{\mathsf{msg}}})$ The signing algorithm takes as input a secret signing key sk_{id} as well as a message $m \in \{0,1\}^{\ell_{\mathsf{msg}}})$, and outputs a signature $\sigma \in \ell_{\mathsf{sig}}$ for identity id on m.
– $\mathsf{Aggregate}(\mathsf{PP}, \{(id_i, m_i), \sigma_i\}_{i=1}^t)$ The aggregation algorithm takes as input t tuples $\{(id_i, m_i), \sigma_i\}$ (for some arbitrary t) where each tuple is $(\ell_{\mathsf{id}}, \ell_{\mathsf{msg}}, \ell_{\mathsf{sig}})$-length. It outputs an aggregate signature σ_{agg} whose length is polynomial in λ, but independent of t.
– $\mathsf{Verify}(\mathsf{PP}, \{(id_i, m_i)\}_{i=1}^t, \sigma_{\mathsf{agg}})$ The verification algorithm takes as input the public parameters PP, t tuples $\{(id_i, m_i)\}$ that are $(\ell_{\mathsf{id}}, \ell_{\mathsf{msg}})$-length, and an aggregate signature σ_{agg}. It outputs 0 or 1 to indicate whether verification succeeded.

Correctness. For all $\lambda, n \in \mathbb{N}$, $(\mathsf{PP}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, n)$, t tuples $\{(id_i, m_i)\}$ that are $(\ell_{\mathsf{id}}, \ell_{\mathsf{msg}})$-length, for all $i \in [t]$, $sk_{id_i} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, id_i)$, $\sigma_i \leftarrow \mathsf{Sign}(\mathsf{sk}_{id_i}, m_i)$, and $\sigma_{\mathsf{agg}} \leftarrow \mathsf{Aggregate}(\mathsf{PP}, \{(id_i, m_i), \sigma_i\}_{i=1}^t)$, we require that $\mathsf{Verify}(\mathsf{PP}, \{(id_i, m_i)\}_{i=1}^t, \sigma_{\mathsf{agg}}) = 1$.

Selective Security. We consider a weaker attack (selective in both the identity and the message) where a forger is challenged on a given identity/message pair (id^*, m^*) chosen by the adversary before receiving the public parameters. More formally, the selective experiment $\mathbf{Exp}^{\mathsf{sel\text{-}uf}}_{\mathsf{IBAS},\mathcal{A}}(\lambda)$ between a challenger and an adversary $\mathcal{A}$ with respect to scheme $\mathsf{IBAS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Aggregate}, \mathsf{Verify})$ is defined as follows:

Experiment $\mathbf{Exp}^{\mathit{sel\text{-}uf}}_{\mathit{IBAS},\mathcal{A}}(\lambda)$

1. $(id^*, m^*) \leftarrow \mathcal{A}(\mathsf{PP})$;
2. $(\mathsf{PP}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$;
3. $(L^* = \{(id_i, m_i)\}_{i=1}^t, \sigma^*_{\mathsf{agg}}) \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot),\mathsf{Sign}(\cdot,\cdot)}(\mathsf{PP})$;
   - $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$ oracle: on input an identity id, returns secret keys for arbitrary identities.
   - $\mathsf{Sign}(\cdot, \cdot)$ oracle: on input an identity id and a message m, sets $sk_{id} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, id)$ and returns $\mathsf{Sign}(sk_{id}, m)$.
4. The adversary $\mathcal{A}$ wins or the output of this experiment is 1 if the following hold true:
   (a) $\mathsf{Verify}(\mathsf{PP}, \{(id_i, m_i)\}_{i=1}^t, \sigma^*_{\mathsf{agg}})=1$,
   (b) $\exists i^* \in [t]$ such that
      i. $(id^*, m^*) = (id_{i^*}, m_{i^*}) \in L^*$,
      ii. id^* has not been asked to the $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$ oracle,
      iii. (id^*, m^*) has not been submitted to the $\mathsf{Sign}(\cdot, \cdot)$ oracle.

The advantage of an adversary $\mathcal{A}$ in the above game is defined to be

$$\mathsf{Adv}^{\mathsf{sel\text{-}uf}}_{\mathsf{IBAS},\mathcal{A}} = \Pr[\mathbf{Exp}^{\mathsf{sel\text{-}uf}}_{\mathsf{IBAS},\mathcal{A}}(\lambda) = 1],$$

where the probability is taken over all coin tosses of the Setup, KeyGen, and Sign algorithm and of $\mathcal{A}$.

Definition 4 *(Selective Unforgeability). An identity-based aggregate signature scheme IBAS is existentially unforgeable with respect to selectively chosen identiry/message pair attacks if for all probabilistic polynomial time adversaries $\mathcal{A}$, the advantage $\mathit{Adv}^{\mathit{sel\text{-}uf}}_{\mathit{IBAS},\mathcal{A}}$ in the experiment $\mathbf{Exp}^{\mathit{sel\text{-}uf}}_{\mathit{IBAS},\mathcal{A}}(\lambda)$ is negligible in λ.*

In our setting, we define an n-bounded identity-based aggregate signatures scheme, which means that at most n signatures can be aggregated.

Definition 5. *An n-bounded identity-based aggregate signatures scheme IBAS=(Setup, KeyGen, Sign, Aggregate,Verify) is an IBAS in which Setup algorithm takes an additional input 1^n and Aggregate algorithm takes in t tuples $\{(id_i, m_i), \sigma_i\}_{i\in[t]}$ satisfying $t \leq n$. The public parameters output by Setup have size bounded by some polynomial in λ and n. However, the aggregated signature has size bounded by a polynomial in λ, but is independent of n.*

Comparison to Previous Definitions. Our definition of IBAS make the requirement that there is an a-priori bound n on the number of signatures that can be aggregated. It is different from the definition described in [10], which allows any two aggregate signatures can be combined into a new aggregate signature.

4 Generic Construction of Identity-Based Aggregate Signatures

In this section, we present our generic transformation to build n-bounded IBAS from length-bounded signature scheme, which can be proven selectively secure in the standard model. Besides indistinguishability obfuscator $i\mathcal{O}$, we will use the following primitives.

- A selectively one-time secure $(\ell_{\mathsf{ckt}}, \ell_{\mathsf{inp}}, \ell_{\mathsf{out}})$ universal parameter scheme UP=($\mathsf{UniversalGen}$,$\mathsf{InduceGen}$).
- A wCCA-secure public-key encryption scheme $\mathsf{PKE}_{\mathsf{wCCA}} = (\mathsf{PKE.Setup}_{\mathsf{wCCA}}, \mathsf{PKE.Enc}_{\mathsf{wCCA}}, \mathsf{PKE.Dec}_{\mathsf{wCCA}})$. Let the randomness space of $\mathsf{PKE.Enc}_{\mathsf{wCCA}}$ be $\{0,1\}^{\ell_{\mathsf{inp}}/2}$. We give a formal definition of wCCA-secure PKE scheme in Appendix 1.
- A $(\ell_{\mathsf{vk}}, \ell_{\mathsf{msg}}, \ell_{\mathsf{sig}})$-length signature scheme $\mathsf{SIG} = (\mathsf{SIG.Setup}, \mathsf{SIG.Sign}, \mathsf{SIG.Vefy})$ that the verification keys output by $\mathsf{SIG.Setup}$ have length at most $\ell_{\mathsf{vk}}(\lambda)$, $\mathsf{SIG.Sign}$ takes as input messages of length at most $\ell_{\mathsf{msg}}(\lambda)$ and outputs signatures of length bounded by $\ell_{\mathsf{sig}}(\lambda)$. Let the randomness space of $\mathsf{SIG.Setup}$ be $\{0,1\}^{\ell_{\mathsf{inp}}/2}$. We give a formal definition of signature scheme in Appendix 2.
- An additively homomorphic encryption scheme HE=($\mathsf{HE.Setup}$, $\mathsf{HE.Enc}$,$\mathsf{HE.Dec}$, $\mathsf{HE.Add}$) with message space $\mathbb{F}_p$ for some prime $p > 2^{\ell_{\mathsf{sig}}}$ and ciphertext space $\mathcal{C}_{\mathsf{HE}}$, where each ciphertext in $\mathcal{C}_{\mathsf{HE}}$ can be represented using ℓ_{HEct} bits. We give a formal definition of additively homomorphic encryption scheme in Appendix 3.
- A puncturable PRF F with key space $\mathcal{K}$, input space $\{0,1\}^{\ell_{\mathsf{vk}}+\ell_{\mathsf{id}}+\ell_{\mathsf{msg}}+\log n+\ell_{\mathsf{HEct}}}$ and range $\{0,1\}^{\ell}$.
- An injective one-way function $f : \{0,1\}^{\ell} \to \{0,1\}^{2\ell}$.

Our n-bounded identity-based aggregate signature scheme consists of algorithms $\mathsf{IBAS.Setup}$, $\mathsf{IBAS.KeyGen}$, $\mathsf{IBAS.Sign}$, $\mathsf{IBAS.Aggregate}$ and $\mathsf{IBAS.Verify}$ described below.

$\mathsf{IBAS.Setup}(1^\lambda, n)$: On input 1^λ, the $\mathsf{IBAS.Setup}$ algorithm works as follows.

1. It runs $(pk_{\mathsf{HE}}, sk_{\mathsf{HE}}) \leftarrow \mathsf{HE.Setup}(1^\lambda)$ and computes ciphertext $\mathsf{ct}_i \leftarrow \mathsf{HE.Enc}(pk_{\mathsf{HE}}, 0)$ for all $i \in [n]$.
2. It runs $(pk_{\mathsf{wCCA}}, sk_{\mathsf{wCCA}}) \leftarrow \mathsf{PKE.Setup}_{\mathsf{wCCA}}(1^\lambda)$ to generate a key pair for $\mathsf{PKE}_{\mathsf{wCCA}}$.
3. Then it creates a program $\mathsf{Prog}\{pk_{\mathsf{wCCA}}\}$ which is defined below as Fig. 1.

4. Choose a puncturable PRF key K, create programs $\mathsf{Prog}\{K, \mathsf{ct}_1, \ldots, \mathsf{ct}_n\}$ and $\mathsf{Prog}\{K\}$ described below as Figs. 2 and 3 respectively, and set $P_1 = i\mathcal{O}(\mathsf{Prog}\{K, \mathsf{ct}_1, \ldots, \mathsf{ct}_n\})$ and $P_2 = i\mathcal{O}(\mathsf{Prog}\{K\})$.
5. Finally it computes $U \leftarrow \mathsf{UniversalGen}(1^\lambda)$.

The public parameters PP is $(pk_{\mathsf{HE}}, U, \mathsf{Prog}\{pk_{\mathsf{wCCA}}\}, P_1, P_2)$ and the master secret key msk is sk_{wCCA}.

IBAS.KeyGen(msk, id) : On input the master secret key msk and $id \in \{0,1\}^{\ell_{\mathsf{id}}}$, it computes $(vk_{id}, c_{id}) \leftarrow \mathsf{InduceGen}(U, \mathsf{Prog}\{pk_{\mathsf{wCCA}}\} \| id)$ and returns $\mathsf{sk}_{id} \leftarrow \mathsf{PKE.Dec}_{\mathsf{wCCA}}(\mathsf{msk}, c_{id})$.

Remark. For any program $\mathsf{Prog}\{pk_{\mathsf{wCCA}}\}$, we let $\mathsf{Prog}\{pk_{\mathsf{wCCA}}\} \| id$ denote the program $\mathsf{Prog}\{pk_{\mathsf{wCCA}}\}$ extended with an additional string $id \in \{0,1\}^{\ell_{\mathsf{id}}}$. Although their description is different, program $\mathsf{Prog}\{pk_{\mathsf{wCCA}}\} \| id$ has the same functionality as program $\mathsf{Prog}\{pk_{\mathsf{wCCA}}\}$. We require that this extension is performed in some standard and deterministic way, for instance by always adding the id string at the end of the code.

IBAS.Sign(sk_{id}, m)**:** On input a secret signing key sk_{id} and a message $m \in \{0,1\}^{\ell_{\mathsf{msg}}}$, it runs $\sigma \leftarrow \mathsf{SIG.Sign}(\mathsf{sk}_{id}, m)$ and returns σ.

IBAS.Aggregate$(\mathsf{PP}, \{(id_i, m_i), \sigma_i\}_i)$**:** On input public parameters PP and tuples $\{(id_i, m_i), \sigma_i\}_i$, if tuples $\{(id_i, m_i), \sigma_i\}$ are not distinct, the algorithm outputs $\perp$. Else, it computes $(vk_{id_i}, c_{id}) \leftarrow \mathsf{InduceGen}(U, \mathsf{Prog}\{pk_{\mathsf{wCCA}}\} \| id_i)$ and outputs $P_1(\{vk_{id_i}, (id_i, m_i), \sigma_i\}_i)$.

IBAS.Verify$(\mathsf{PP}, \{(id_i, m_i)\}_i, \sigma_{\mathsf{agg}} = (t, s))$**:** The verification algorithm checks if the tuples $\{(id_i, m_i)\}_i$ are distinct. If not, it outputs 0. Else, it computes $(vk_{id_i}, c_{id}) \leftarrow \mathsf{InduceGen}(U, \mathsf{Prog}\{pk_{\mathsf{wCCA}}\} \| id_i)$, and outputs $P_2(\{(vk_{id_i}, id_i, m_i)\}_i, \sigma_{\mathsf{agg}} = (t, s))$.

Prog$\{pk_{\mathsf{wCCA}}\}$

Hardwired into the circuit: pk_{wCCA}.
Input to the circuit: $r = r_0 \| r_1 \in \{0,1\}^{\ell_{\mathsf{inp}}}$.
Algorithm:
1. Compute $(vk_{\mathsf{SIG}}, sk_{\mathsf{SIG}}) \leftarrow \mathsf{SIG.Setup}(1^\lambda; r_0)$ and $c \leftarrow \mathsf{PKE.Enc}_{\mathsf{wCCA}}(pk_{\mathsf{wCCA}}, sk_{\mathsf{SIG}}; r_1)$.
2. Output (vk_{SIG}, c).

Fig. 1. Program $\mathrm{Prog}\{pk_{w\mathsf{CCA}}\}$.

The correctness of this scheme follows immediately from the correctness of SIG, $\mathsf{PKE}_{\mathsf{wCCA}}$, HE and $(\mathsf{UniversalGen}, \mathsf{InduceGen})$.

Remark. The setup algorithm is parameterized by a polynominal n that gives an a-priori bound on the number of signatures that can be aggregated. The size of the parameters and aggregate signatures are independent of it.

$\mathbf{Prog}\{K, \mathsf{ct}_1, \ldots, \mathsf{ct}_n\}$

Hardwired into the circuit: $K, \mathsf{ct}_1, \ldots, \mathsf{ct}_n$.
Input to the circuit: $\{vk_{id_i}, (id_i, m_i), \sigma_i\}_i$.
Algorithm:

1. If $\exists i$ such that $\mathsf{SIG.Vefy}(vk_{id_i}, m_i, \sigma_i) = 0$, then output $\perp$. End if.
2. Compute $t = \sum_i \sigma_i \cdot \mathsf{ct}_i$.
3. Compute $s_i = F(K, vk_{id_i} \| id_i \| m_i \| i \| t)$.
4. Output $\sigma_{\mathsf{agg}} = (t, \oplus_i s_i)$.

Fig. 2. Program $\mathrm{Prog}\{K, \mathsf{ct}_1, \ldots, \mathsf{ct}_n\}$.

$\mathbf{Prog}\{K\}$

Hardwired into the circuit: K.
Input to the circuit: $\{(vk_{id_i}, id_i, m_i)\}_i, \sigma_{\mathsf{agg}} = (t, s)$.
Algorithm:

1. Compute $s' = \oplus_i F(K, vk_{id_i} \| id_i \| m_i \| i \| t)$.
2. Output 1 if $s' = s$, else output 0.

Fig. 3. Program $\mathrm{Prog}\{K\}$.

Theorem 2. *Let $\mathcal{A}$ be any PPT adversary, and SIG a $(\ell_{vk}, \ell_{msg}, \ell_{sig})$-length secure signature scheme. Let $\mathsf{Adv}^{\mathsf{sel\text{-}uf}}_{\mathsf{IBAS},\mathcal{A}}$ denote the advantage of $\mathcal{A}$ in the identity-based aggregate signatures. Let $\mathsf{Adv}_{\mathsf{UP}}$, $\mathsf{Adv}_{\mathsf{SIG}}$, $\mathsf{Adv}_{\mathsf{HE}}$, $\mathsf{Adv}_{\mathsf{PKE}_{\mathsf{wCCA}}}$, $\mathsf{Adv}_{i\mathcal{O}}$, $\mathsf{Adv}_{\mathsf{PRF}}$ and Adv_f denote the maximum advantage of a PPT adversary against universal parameters scheme UP, signature scheme SIG, additively homomorphic encryption scheme HE, wCCA secure public key encryption $\mathsf{PKE}_{\mathsf{wCCA}}$, indistinguishability obfuscator $i\mathcal{O}$, selectively secure puncturable PRF F and one way function f respectively. Then,*

$$\mathsf{Adv}^{\mathsf{sel\text{-}uf}}_{\mathsf{IBAS},\mathcal{A}} \leq n\big(\mathsf{Adv}_{\mathsf{HE}} + \mathsf{Adv}_{\mathsf{UP}} + \mathsf{Adv}_{\mathsf{PKE}_{\mathsf{wCCA}}} + 2^{\ell_{\mathsf{HEct}}}(6\mathsf{Adv}_{i\mathcal{O}} + 2\mathsf{Adv}_{\mathsf{PRF}} + \mathsf{Adv}_f) + \mathsf{Adv}_{\mathsf{SIG}}\big)$$

where ℓ_{HEct} is the length of ciphertexts in $\mathcal{C}_{\mathsf{HE}}$.

We now prove via a sequence of exponential number hybrid games Game 0, Game 1, Game 2, Game 3, Game 4, Game 5,0, Game 5,0-1, Game 5,0-2,..., Game 5,0-6, Game 5,1, Game 5,1-1, Game 5,1-2,..., Game 5,1-6, Game 5,2,..., Game $5{,}2^{\ell_{\mathsf{HEct}}}$, each of which we prove to be indistinguishable from the previous one.

Sequence of Games.

Game 0. This game is the original selective security game $\mathbf{Exp}^{\mathsf{sel\text{-}uf}}_{\mathsf{IBAS},\mathcal{A}}(\lambda)$ in Sect. 3 instantiated by our construction.

1. $\mathcal{A}$ first choose a challenge (id^*, m^*).
2. The challenger chooses $(pk_{\mathsf{HE}}, sk_{\mathsf{HE}})$, $(pk_{\mathsf{wCCA}}, sk_{\mathsf{wCCA}})$, $U \leftarrow \mathsf{UniversalGen}(1^\lambda)$ and $K \leftarrow \mathsf{PRF.Setup}(1^\lambda)$. Compute $\mathsf{ct}_i \leftarrow \mathsf{HE.Enc}(pk_{\mathsf{HE}}, 0)$ for all $i \in [n]$, $P_1 \leftarrow i\mathcal{O}\big(\mathsf{Prog}\{K, \mathsf{ct}_1, \ldots, \mathsf{ct}_n\}\big)$ and $P_2 \leftarrow i\mathcal{O}\big(\mathsf{Prog}\{K\}\big)$. Let $\mathsf{Prog}\{pk_{\mathsf{wCCA}}\}$ be circuit as defined in the Fig. 1. Set $\mathsf{PP} = (pk_{\mathsf{HE}}, U, \mathsf{Prog}\{pk_{\mathsf{wCCA}}\}, P_1, P_2)$ and $\mathsf{msk} = sk_{\mathsf{wCCA}}$, and send PP to $\mathcal{A}$.
3. On attacker's $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$ queries and $\mathsf{Sign}(\cdot, \cdot)$ queries, the challenger responds as follows:
   - On $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$ query for id, the challenger computes $(vk_{id}, c_{id}) \leftarrow \mathsf{InduceGen}(U, \mathsf{Prog}\{pk_{\mathsf{wCCA}}\}\|id)$ and returns $\mathsf{sk}_{id} \leftarrow \mathsf{PKE.Dec}_{\mathsf{wCCA}}(\mathsf{msk}, c_{id})$.
   - On $\mathsf{Sign}(\cdot, \cdot)$ query for identity id and message m, the challenger first runs $\mathsf{sk}_{id} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, id)$ and then returns $\sigma \leftarrow \mathsf{SIG.Sign}(\mathsf{sk}_{id}, m)$.
4. Finally the adversary outputs $(L^* = \{(id_i, m_i)\}_i, \sigma^*_{\mathsf{agg}})$. The adversary $\mathcal{A}$ wins or the output of this experiment is 1 if the following hold true:
   (a) $\mathsf{IBAS.Verify}(\mathsf{PP}, L^* = \{(id_i, m_i)\}_i, \sigma^*_{\mathsf{agg}}) = 1$,
   (b) $\exists i^*$ such that
      i. $(id^*, m^*) = (id_{i^*}, m_{i^*}) \in L^*$,
      ii. id^* has not been asked to the $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$ oracle,
      iii. (id^*, m^*) has not been submitted to the $\mathsf{Sign}(\cdot, \cdot)$ oracle.

Game 1. This game is exactly similar to the previous one, except that the challenger guesses a position $i^* \leftarrow [n]$, and the attacker wins if $id^* = id_{i^*}$ and $m^* = m_{i^*}$.

Game 2. This game is similar to the previous one, except that ct_{i^*} is an encryption of 1 under pk_{HE}, instead of 0. That is, compute $\mathsf{ct}_i \leftarrow \mathsf{HE.Enc}(pk_{\mathsf{HE}}, 0)$ for all $i \in [n]$ and $i \neq i^*$, and $\mathsf{ct}_{i^*} \leftarrow \mathsf{HE.Enc}(pk_{\mathsf{HE}}, 1)$.

Game 3. This game is identical to Game 2, except for the following. The experiment generates parameters as $U \leftarrow \mathsf{SimUGen}(1^\lambda, \mathsf{Prog}\{pk_{\mathsf{wCCA}}\}\|id_{i^*}, (vk_{id_{i^*}}, c_{id_{i^*}}))$, where $(vk_{id_{i^*}}, c_{id_{i^*}}) \leftarrow \mathsf{Prog}\{pk_{\mathsf{wCCA}}\}\|id_{i^*}(r)$ for uniformly random $r \in \{0,1\}^{\ell_{\mathsf{inp}}}$. And on attacker's $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$ query for id, the challenger computes $(vk_{id}, c_{id}) \leftarrow \mathsf{InduceGen}(U, \mathsf{Prog}\{pk_{\mathsf{wCCA}}\}\|id)$, where $U \leftarrow \mathsf{SimUGen}(1^\lambda, \mathsf{Prog}\{pk_{\mathsf{wCCA}}\}\|id_{i^*}, (vk_{id_{i^*}}, c_{id_{i^*}}))$, and returns $\mathsf{sk}_{id} \leftarrow \mathsf{PKE.Dec}_{\mathsf{wCCA}}(\mathsf{msk}, c_{id})$.

Game 4. The only difference between this game and the previous one is in the behavior of evaluation on the $d_{id_{i^*}} = \mathsf{Prog}\{pk_{\mathsf{wCCA}}\}\|id_{i^*}$. In Game 3, the entry corresponding to $d_{id_{i^*}}$ is of the form $(d_{id_{i^*}}, (vk_{id_{i^*}}, c_{id_{i^*}}))$ where $r = r_0\|r_1 \in \{0,1\}^{\ell_{\mathsf{inp}}}$, $(vk_{id_{i^*}}, sk_{id_{i^*}}) \leftarrow \mathsf{SIG.Setup}(1^\lambda; r_0)$ and $c_{id_{i^*}} \leftarrow \mathsf{PKE.Enc}_{\mathsf{wCCA}}(pk_{\mathsf{wCCA}}, sk_{id_{i^*}}; r_1)$. In this game, the entry corresponding to $d_{id_{i^*}}$ is $(d_{id_{i^*}}, (vk_{id_{i^*}}, c_{id_{i^*}}))$, where $c_{id_{i^*}} \leftarrow \mathsf{PKE.Enc}_{\mathsf{wCCA}}(pk_{\mathsf{wCCA}}, 1^\lambda; r_1)$.
We will now describe an exponential number of hybrid experiments Game 5,j for $j \leq 2^{\ell_{\mathsf{HEct}}}$. Let us define some notations. Recall $\mathsf{Prog}\{K\}$ takes as input tuples of the form $(\{(vk_{id_i}, id_i, m_i)\}_i, (t, s))$. We say tuple $\big(\{(vk_{id_i}, id_i, m_i)\}_i, (t, s)\big)$ is (i^*, sk_{HE})-rejecting if $\mathsf{SIG.\ Vefy}(vk_{id_{i^*}}, m_{i^*}, \mathsf{HE.Dec}(sk_{\mathsf{HE}}, t)) = 0$.

Game 5, j. In this game, the adversary does not win if the forgery input $(\{(vk_{id_i}, id_i, m_i)\}_i, (t^*, s^*))$ is (i^*, sk_{HE})-rejecting and $t^* \leq j$. That is, finally the adversary $\mathcal{A}$ outputs $(L^* = \{(id_i, m_i)\}_i, \sigma^*_{\mathsf{agg}} = (t^*, s^*))$, and $\mathcal{A}$ wins if the following hold true:

1. $\mathsf{IBAS.Verify}(\mathsf{PP}, L^* = \{(id_i, m_i)\}_i, \sigma^*_{\mathsf{agg}}) = 1$,
2. $(\{(vk_{id_i}, id_i, m_i)\}_i, (t^*, s^*))$ is not (i^*, sk_{HE})-rejecting or $t^* > j$,
3. id_{i^*} has not been asked to the $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$ oracle,
4. (id_{i^*}, m_{i^*}) has not been submitted to the $\mathsf{Sign}(\cdot, \cdot)$ oracle.

Game 5, j-1. In this game, the challenger replace P_2 with obfuscation of program $\mathsf{Prog\text{-}1}\{K\}$ instead of $\mathsf{Prog}\{K\}$. That is $P_2 = i\mathcal{O}(\mathsf{Prog\text{-}1}\{K\})$. In program $\mathsf{Prog\text{-}1}\{K\}$ as described in Fig. 4, instead of checking whether $s = \oplus_i s_i$, it uses an injective one way function f to check if $f(s \oplus (\oplus_{i \neq i^*} s_i)) = f(s_{i^*})$.

Prog-1$\{K\}$

Hardwired into the circuit: K.
Input to the circuit: $\{(vk_{id_i}, id_i, m_i)\}_i, \sigma_{\mathsf{agg}} = (t, s)$.
Algorithm:
1. Compute $s' = \oplus_{i \neq i^*} F(K, vk_{id_i} \| id_i \| m_i \| i \| t) \oplus s$.
2. Output 1 if $f(F(K, vk_{id_{i^*}} \| id_{i^*} \| m_{i^*} \| i^* \| t)) = f(s')$, else output 0.

Fig. 4. Program Prog-1$\{K\}$.

Game 5, j-2. In this game, $\mathsf{Prog}\{K, \mathsf{ct}_1, \ldots, \mathsf{ct}_n\}$ and $\mathsf{Prog\text{-}1}\{K\}$ are replaced by $\mathsf{Prog\text{-}1}\{K\{y\}, \mathsf{ct}_1, \ldots, \mathsf{ct}_n\}$ (described in Fig. 5) and $\mathsf{Prog\text{-}2}\{y, z, K\{y\}\}$ (described in Fig. 6) respectively. Both the replaced programs use the punctured key at punctured point $y = vk_{id_{i^*}} \| id_{i^*} \| m_{i^*} \| i^* \| (j+1)$. More precisely, the challenger computes $y = vk_{id_{i^*}} \| id_{i^*} \| m_{i^*} \| i^* \| (j+1)$, $K\{y\} \leftarrow \mathsf{PRF.Puncture}(K, y)$ and $z = f(F(K, y))$. Let $P_1 = i\mathcal{O}(\mathsf{Prog\text{-}1}\{K, \mathsf{ct}_1, \ldots, \mathsf{ct}_n\})$ and $P_2 = i\mathcal{O}(\mathsf{Prog\text{-}2}\{K\})$.
Game 5, j-3. This game is similar to the previous one, except that z is a uniformly random string. That is, the challenger randomly chooses z', and computes $z = f(z')$.
Game 5, j-4. In this game, the challenger modifies the winning condition. That is, $(\{(vk_{id_i}, id_i, m_i)\}_i, (t^*, s^*))$ is not (i^*, sk_{HE})-rejecting or $t^* > j+1$.
Game 5, j-5. In this game, the challenger sets $z = f(F(K, y))$ as in Game 4-j-2.
Game 5, j-6. In this game, the challenger changes program $\mathsf{Prog\text{-}1}\{K, \mathsf{ct}_1, \ldots, \mathsf{ct}_n\}$ and program $\mathsf{Prog\text{-}2}\{K\}$ back to $\mathsf{Prog}\{K, \mathsf{ct}_1, \ldots, \mathsf{ct}_n\}$ and $\mathsf{Prog\text{-}1}\{K\}$ respectively.

Game 6. This game is identical to Game 5, $2^{\ell_{\mathsf{HEct}}}$.

Prog-1$\{K\{y\}, \mathsf{ct}_1, \ldots, \mathsf{ct}_n\}$

Hardwired into the circuit: $K\{y\}, \mathsf{ct}_1, \ldots, \mathsf{ct}_n$.
Input to the circuit: $\{vk_{id_i}, (id_i, m_i), \sigma_i\}_i$.
Algorithm:

1. If $\exists i$ such that $\mathsf{SIG.Vefy}(vk_{id_i}, m_i, \sigma_i) = 0$, then output $\perp$. End if.
2. Compute $t = \sigma_1 \cdot \mathsf{ct}_1 + \ldots + \sigma_n \cdot \mathsf{ct}_n$.
3. Compute $s_i = F(K\{y\}, vk_{id_i} \| id_i \| m_i \| i \| t)$.
4. Output $\sigma_{\mathsf{agg}} = (t, \oplus_i s_i)$.

Fig. 5. Program Prog-1$\{K\{y\}, \mathsf{ct}_1, \ldots, \mathsf{ct}_n\}$.

Prog-2$\{y, z, K\{y\}\}$

Hardwired into the circuit: $y, z, K\{y\}$.
Input to the circuit: $\{(vk_{id_i}, id_i, m_i)\}_i, \sigma_{\mathsf{agg}} = (t, s)$.
Algorithm:

1. Compute $s' = \oplus_{i \neq i^*} F(K\{y\}, vk_{id_i} \| id_i \| m_i \| i \| t) \oplus s$.
2. If $vk_{id_{i^*}} \| id_{i^*} \| m_{i^*} \| i^* \| t = y$, then output 1 if $z = f(s')$, else output 0.
3. Else output 1 if $F(K\{y\}, vk_{id_{i^*}} \| id_{i^*} \| m_{i^*} \| i^* \| t) = f(s')$, else output 0.

Fig. 6. Program Prog-2$\{y, z, K\{y\}\}$.

Analysis. Let $\mathsf{Adv}^i_{\mathcal{A}}$ denote the advantage of adversary $\mathcal{A}$ in Game i. We now establish the difference of the attacker's advantage between each adjacent game via a sequence of lemmas.

Lemma 1. *For any adversary $\mathcal{A}$, $Adv^1_{\mathcal{A}} = Adv^0_{\mathcal{A}}/n$.*

Proof. This follows from the definitions of Game 0 and Game 1. The only difference between the two experiments is the change in winning condition, which now includes the guess i^*. This guess is correct with probability $1/n$.

Lemma 2. *For any PPT adversary $\mathcal{A}$, $Adv^1_{\mathcal{A}} - Adv^2_{\mathcal{A}} \leq Adv_{HE}(\lambda)$.*

Proof. Suppose there exists an adversary $\mathcal{A}$ such that $\mathsf{Adv}^1_{\mathcal{A}} - \mathsf{Adv}^2_{\mathcal{A}} = \epsilon$. We will construct a PPT algorithm $\mathcal{B}$ that breaks the semantic security of HE scheme using $\mathcal{A}$.

$\mathcal{B}$ receives the public key pk_{HE}. It sends $0, 1$ as challenge messages to the HE challenger, and receives ct in response. On receiving (id^*, m^*) from $\mathcal{A}$, $\mathcal{B}$ chooses $i^* \leftarrow [n]$, $(pk_{\mathsf{wCCA}}, sk_{\mathsf{wCCA}})$, $U \leftarrow \mathsf{UniversalGen}(1^\lambda)$ and computes ciphertext $\mathsf{ct}_i \leftarrow \mathsf{HE.Enc}(pk_{\mathsf{HE}}, 0)$ for all $i \neq i^*$. It sets $\mathsf{ct}_{i^*} = \mathsf{ct}$. Let $\mathsf{Prog}\{pk_{\mathsf{wCCA}}\}$ be circuit as defined in the Fig. 1. It chooses $K \leftarrow \mathsf{PRF.Setup}(1^\lambda)$ and computes $P_1 = i\mathcal{O}\big(\mathsf{Prog}\{K, \mathsf{ct}_1, \ldots, \mathsf{ct}_n\}\big)$ and $P_2 = i\mathcal{O}(\mathsf{Prog}\{K\})$. $\mathcal{B}$ sends $\mathsf{PP} = \big(pk_{\mathsf{HE}}, U, P_1, P_2, \mathsf{Prog}\{pk_{\mathsf{wCCA}}\}\big)$ to $\mathcal{A}$.

$\mathcal{A}$ then asks for $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$, $\mathsf{Sign}(\cdot, \cdot)$ queries, which $\mathcal{B}$ can simulate perfectly. Finally, $\mathcal{A}$ outputs a forgery $\sigma^*_{\mathsf{agg}} = (t^*, s^*)$ and tuples $\{(id_i, m_i)\}_i$. If $\mathcal{A}$ wins as per the winning conditions (which are the same in both Games 1 and 2), output 0, else output 1.

Clearly, if ct is an encryption of 0, then this corresponds to Game 1, else it corresponds to Game 2. This completes our proof.

Lemma 3. *For any PPT adversary $\mathcal{A}$, $Adv^2_{\mathcal{A}} - Adv^3_{\mathcal{A}} \leq Adv_{UP}(\lambda)$.*

Proof. Suppose there exists a PPT adversary $\mathcal{A}$ such that $\mathsf{Adv}^2_{\mathcal{A}} - \mathsf{Adv}^3_{\mathcal{A}} = \epsilon$. We will construct a PPT algorithm $\mathcal{B}$ such that $\left|\Pr[\mathsf{Real}^{\mathcal{B}}(1^\lambda) = 1] - \Pr[\mathsf{Ideal}^{\mathcal{B}}_{\mathsf{SimUGen}}(1^\lambda) = 1]\right| = \epsilon$.

$\mathcal{B}$ interacts with $\mathcal{A}$ and participates in either the Real or Ideal game. On receiving (id^*, m^*) from $\mathcal{A}$, $\mathcal{B}$ chooses $i^* \leftarrow [n]$, and sets $d_{id_{i^*}} = \mathsf{Prog}\{pk_{\mathsf{wCCA}}\} \| id_{i^*}$. $\mathcal{B}$ sends $d_{id_{i^*}}$ to the challenger of universal parameters UP. The challenger of universal parameters UP computes $(vk_{id_{i^*}}, c_{id_{i^*}}) \leftarrow d_{id_{i^*}}(r)$ for uniformly random $r = r_0 \| r_1 \in \{0,1\}^{\ell_{\mathsf{inp}}}$, where $(vk_{id_{i^*}}, sk_{id_{i^*}}) \leftarrow \mathsf{SIG.Setup}(1^\lambda; r_0)$ and $c_{id_{i^*}} \leftarrow \mathsf{PKE.Enc}_{\mathsf{wCCA}}(pk_{\mathsf{wCCA}}, sk_{id_{i^*}}; r_1)$. $\mathcal{B}$ receives U from the challenger of universal parameters. $\mathcal{B}$ then chooses $(pk_{\mathsf{wCCA}}, sk_{\mathsf{wCCA}})$, $(pk_{\mathsf{HE}}, sk_{\mathsf{HE}})$ and compute ciphertext $\mathsf{ct}_i \leftarrow \mathsf{HE.Enc}(pk_{\mathsf{HE}}, 0)$ for all $i \in [n]$ and $i \neq i^*$, $\mathsf{ct}_{i^*} \leftarrow \mathsf{HE.Enc}(pk_{\mathsf{HE}}, 1)$. Let $\mathsf{Prog}\{pk_{\mathsf{wCCA}}\}$ be circuit as defined in the Fig. 1. It chooses $K \leftarrow \mathsf{PRF.Setup}(1^\lambda)$ and computes $P_1 = i\mathcal{O}(\mathsf{Prog}\{K, \mathsf{ct}_1, \ldots, \mathsf{ct}_n\})$ and $P_2 = i\mathcal{O}(\mathsf{Prog}\{K\})$. $\mathcal{B}$ sends $\mathsf{PP} = (pk_{\mathsf{HE}}, U, \mathsf{Prog}\{pk_{\mathsf{wCCA}}\}, P_1, P_2)$ to $\mathcal{A}$.

For the $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$ and $\mathsf{Sign}(\cdot, \cdot)$ queries, $\mathcal{B}$ computes $(vk_{id}, c_{id}) \leftarrow \mathsf{InduceGen}(U, d_{id})$, where $d_{id} = \mathsf{Prog}\{pk_{\mathsf{wCCA}}\} \| id$, and returns $\mathsf{sk}_{id} \leftarrow \mathsf{PKE.Dec}_{\mathsf{wCCA}}(sk_{\mathsf{wCCA}}, c_{id})$ by using sk_{wCCA}. Finally, it receives a forgery $\sigma^*_{\mathsf{agg}} = (t^*, s^*)$ and tuples $\{(id_i, m_i)\}_i$. Note that since there is no $\mathsf{Honest\ Parameter\ Violation}$, $\mathsf{InduceGen}(U, \mathsf{Prog}\{pk_{\mathsf{wCCA}}\} \| id_{i^*}) = (vk_{id_{i^*}}, c_{id_{i^*}})$. Therefore, Game 2 corresponds to $\mathsf{Real}^{\mathcal{B}}(1^\lambda)$ experiment, while Game 3 corresponds to $\mathsf{Ideal}^{\mathcal{B}}_{\mathsf{SimUGen}}$. Hence, $\left|\Pr[\mathsf{Real}^{\mathcal{B}}(1^\lambda) = 1] - \Pr[\mathsf{Ideal}^{\mathcal{B}}_{\mathsf{SimUGen}}(1^\lambda) = 1]\right| = \epsilon$.

Lemma 4. *For any PPT adversary $\mathcal{A}$, $Adv^3_{\mathcal{A}} - Adv^4_{\mathcal{A}} \leq Adv_{PKE_{wCCA}}(\lambda)$.*

Proof. Note that the only difference between Games 3 and 4 is in the behavior of evaluation on the $d_{id_{i^*}} = \mathsf{Prog}\{pk_{\mathsf{wCCA}}\} \| id_{i^*}$. Suppose there exists an adversary $\mathcal{A}$ such that $\mathsf{Adv}^3_{\mathcal{A}} - \mathsf{Adv}^4_{\mathcal{A}} = \epsilon$. We will construct a PPT algorithm $\mathcal{B}$ that breaks the wCCA security of $\mathsf{PKE}_{\mathsf{wCCA}}$ scheme using $\mathcal{A}$.

$\mathcal{B}$ receives the public key pk_{wCCA}. On receiving (id^*, m^*) from $\mathcal{A}$, $\mathcal{B}$ chooses $i^* \leftarrow [n]$, $(pk_{\mathsf{HE}}, sk_{\mathsf{HE}})$ and compute ciphertext $\mathsf{ct}_i \leftarrow \mathsf{HE.Enc}(pk_{\mathsf{HE}}, 0)$ for all $i \in [n]$ and $i \neq i^*$, $\mathsf{ct}_{i^*} \leftarrow \mathsf{HE.Enc}(pk_{\mathsf{HE}}, 1)$. Let $\mathsf{Prog}\{pk_{\mathsf{wCCA}}\}$ be circuit as defined in the Fig. 1. It chooses $K \leftarrow \mathsf{PRF.Setup}(1^\lambda)$ and computes $P_1 = i\mathcal{O}(\mathsf{Prog}\{K, \mathsf{ct}_1, \ldots, \mathsf{ct}_n\})$ and $P_2 = i\mathcal{O}(\mathsf{Prog}\{K\})$.

$\mathcal{B}$ chooses $r = r_0 \| r_1 \in \{0,1\}^{\ell_{\mathsf{inp}}}$ and computes $(vk_{id_{i^*}}, sk_{id_{i^*}}) \leftarrow \mathsf{SIG.Setup}(1^\lambda; r_0)$. $\mathcal{B}$ sends $sk_{id_{i^*}}, 1^\lambda$ as the challenge messages to the wCCA challenger. It receives in response a ciphertext $\mathsf{ct}^*_{\mathsf{wCCA}}$. Let $d_{id_{i^*}}(r) = (vk_{id_{i^*}}, \mathsf{ct}^*_{\mathsf{wCCA}})$, where $d_{id_{i^*}} = \mathsf{Prog}\{pk_{\mathsf{wCCA}}\} \| id_{i^*}$. Then $\mathcal{B}$ compute $U \leftarrow \mathsf{SimUGen}(1^\lambda)$ and sends $\mathsf{PP} = (pk_{\mathsf{HE}}, U, \mathsf{Prog}\{pk_{\mathsf{wCCA}}\}, P_1, P_2)$ to $\mathcal{A}$.

On receiving $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$ query for id, $\mathcal{B}$ computes $(vk_{id}, c_{id}) \leftarrow \mathsf{InduceGen}(U, d_{id})$, where $d_{id} = \mathsf{Prog}\{pk_{\mathsf{wCCA}}\}\|id$, then submits a decryption query to oracle $\mathsf{PKE.Dec}_{\mathsf{wCCA}}(sk_{\mathsf{wCCA}}, \cdot)$ for c_{id}, and returns whatever $\mathsf{PKE.Dec}_{\mathsf{wCCA}}(sk_{\mathsf{wCCA}}, \cdot)$ returns. Finally, $\mathcal{A}$ outputs a forgery $\sigma^*_{\mathsf{agg}} = (t^*, s^*)$ and tuples $\{(id_i, m_i)\}_i$. If $\mathcal{A}$ wins as per the winning conditions (which are the same in both Games 3 and 4), output 0, else output 1.

If $\mathsf{ct}^*_{\mathsf{wCCA}}$ is an encryption of $sk_{id_{i^*}}$, then this is a perfect simulation of Game 3, while if $\mathsf{ct}^*_{\mathsf{wCCA}}$ is an encryption of 1^λ, then this is a perfect simulation of Game 4. This completes our proof.

Observation 1. *For any PPT adversary $\mathcal{A}$, $\mathsf{Adv}^4_{\mathcal{A}} = \mathsf{Adv}^{5,0}_{\mathcal{A}}$.*

Lemma 5. *For any j, any PPT adversary $\mathcal{A}$, $\mathsf{Adv}^{5,j}_{\mathcal{A}} - \mathsf{Adv}^{5,j\text{-}1}_{\mathcal{A}} \leq \mathsf{Adv}_{i\mathcal{O}}(\lambda)$.*

Proof. To prove this lemma, we need to show that the programs $\mathsf{Prog}\{K\}$ and $\mathsf{Prog\text{-}1}\{K\}$ are functionally identical. This follows from the observation that f is an injective function, and hence, for any t, s,

$$s = \oplus_i F(K, vk_{id_i}\|id_i\|m_i\|i\|t) = \oplus_{i\neq i^*} F(K, vk_{id_i}\|id_i\|m_i\|i\|t) \oplus F(K, vk_{id_{i^*}}\|id_{i^*}\|m_{i^*}\|i^*\|t)$$
$$\iff \oplus_{i\neq i^*} F(K, vk_{id_i}\|id_i\|m_i\|i\|t) \oplus s = F(K, vk_{id_{i^*}}\|id_{i^*}\|m_{i^*}\|i^*\|t)$$
$$\iff f(\oplus_{i\neq i^*} F(K, vk_{id_i}\|id_i\|m_i\|i\|t) \oplus s) = f(F(K, vk_{id_{i^*}}\|id_{i^*}\|m_{i^*}\|i^*\|t)).$$

Lemma 6. *For any j, any PPT adversary $\mathcal{A}$, $\mathsf{Adv}^{5,j\text{-}1}_{\mathcal{A}} - \mathsf{Adv}^{5,j\text{-}2}_{\mathcal{A}} \leq 2\mathsf{Adv}_{i\mathcal{O}}(\lambda)$.*

Proof. Let $K \leftarrow \mathsf{PRF.Setup}(1^\lambda)$, $y = vk_{id_{i^*}}\|id_{i^*}\|m_{i^*}\|i^*\|(j+1)$, $K\{y\} \leftarrow \mathsf{PRF.Puncture}(K, y)$ and $z = f(F(K, y))$. As in the previous proof, it suffices to show that $\mathsf{Prog}\{K, \mathsf{ct}_1, \ldots, \mathsf{ct}_n\}$ and $\mathsf{Prog\text{-}1}\{K\{y\}, \mathsf{ct}_1, \ldots, \mathsf{ct}_n\}$ have identical functionality, and $\mathsf{Prog\text{-}1}\{K\}$ and $\mathsf{Prog\text{-}2}\{y, z, K\{y\}\}$ have identical functionality.

Let us first consider $\mathsf{Prog}\{K, \mathsf{ct}_1, \ldots, \mathsf{ct}_n\}$ and $\mathsf{Prog\text{-}1}\{K\{y\}, \mathsf{ct}_1, \ldots, \mathsf{ct}_n\}$. Consider input $\{vk_{id_i}, (id_i, m_i), \sigma_i\}_i$. Let $t = \sigma_1 \cdot \mathsf{ct}_1 + \ldots + \sigma_n \cdot \mathsf{ct}_n$. From the correctness property of puncturable PRFs, it follows that the only case in which $\mathsf{Prog}\{K, \mathsf{ct}_1, \ldots, \mathsf{ct}_n\}$ and $\mathsf{Prog\text{-}1}\{K\{y\}, \mathsf{ct}_1, \ldots, \mathsf{ct}_n\}$ can possibly differ is when $\mathsf{SIG.Vefy}(vk_{id_i}, m_i, \sigma_i)=1$ for all $i \in [n]$, and $id^*=id_{i^*}$, $m^*=m_{i^*}$ and $t=j+1$. But this case is not possible, since $\mathsf{SIG.Vefy}(vk_{id_{i^*}}, m_{i^*}, \mathsf{HE.Dec}(sk_{\mathsf{HE}}, t))=\mathsf{SIG.Vefy}(vk_{id_{i^*}}, m_{i^*}, \sigma_{i^*})=1$, while $\mathsf{SIG.Vefy}(vk_{id_{i^*}}, m_{i^*}, \mathsf{HE.Dec}(sk_{\mathsf{HE}}, j+1))=0$.

Next, let us consider the programs $\mathsf{Prog\text{-}1}\{K\}$ and $\mathsf{Prog\text{-}2}\{y, z, K\{y\}\}$. Both programs have identical functionality, because $z = f(F(K, y))$ and for all $y' \neq y$, $F(K, y') = F.\mathsf{eval}(K\{y\}, y')$. This concludes our proof.

Lemma 7. *For any j, any PPT adversary $\mathcal{A}$, $\mathsf{Adv}^{5,j\text{-}2}_{\mathcal{A}} - \mathsf{Adv}^{5,j\text{-}3}_{\mathcal{A}} \leq \mathsf{Adv}_{\mathsf{PRF}}(\lambda)$.*

Proof. We will construct a PPT algorithm $\mathcal{B}$ such that $\mathsf{Adv}^{\mathcal{B}}_{\mathsf{PRF}} = \mathsf{Adv}^{5,j\text{-}2}_{\mathcal{A}} - \mathsf{Adv}^{5,j\text{-}3}_{\mathcal{A}}$. On receiving (id^*, m^*) from $\mathcal{A}$, $\mathcal{B}$ chooses $i^* \leftarrow [n]$, $(pk_{\mathsf{wCCA}}, sk_{\mathsf{wCCA}})$, $(pk_{\mathsf{HE}}, sk_{\mathsf{HE}})$ and compute ciphertext $\mathsf{ct}_i \leftarrow \mathsf{HE.Enc}(pk_{\mathsf{HE}}, 0)$ for all $i \in [n]$ and $i \neq i^*$, $\mathsf{ct}_{i^*} \leftarrow \mathsf{HE.Enc}(pk_{\mathsf{HE}}, 1)$. Let $\mathsf{Prog}\{pk_{\mathsf{wCCA}}\}$ be circuit as defined in the

Fig. 1. It chooses $K \leftarrow \mathsf{PRF.Setup}(1^\lambda)$, and let $y = vk_{id_{i^*}} \| id_{i^*} \| m_{i^*} \| i^* \| (j+1)$. $\mathcal{B}$ sends y to the PRF challenger, and receives $K\{y\}, z'$, where either $z' = F(K, y)$ or $z' \leftarrow \{0,1\}^\ell$. It computes $z = f(z')$, $P_1 = i\mathcal{O}(\mathsf{Prog}\{K\{y\}, \mathsf{ct}_1, \ldots, \mathsf{ct}_n\})$ and $P_2 = i\mathcal{O}(\mathsf{Prog}\{y, z, K\{y\}\})$. $\mathcal{B}$ computes $(vk_{id_{i^*}}, c_{id_{i^*}}) \leftarrow \mathsf{Prog}\{pk_{\mathsf{wCCA}} \| id_{i^*}(r)$ for uniformly random $r = r_0 \| r_1 \in \{0,1\}^{\ell_{\mathsf{inp}}}$, where $(vk_{id_{i^*}}, sk_{id_{i^*}}) \leftarrow \mathsf{SIG.Setup}(1^\lambda; r_0)$ and $c_{id_{i^*}} \leftarrow \mathsf{PKE.Enc}_{\mathsf{wCCA}}(pk_{\mathsf{wCCA}}, 1^\lambda; r_1)$. Then $\mathcal{B}$ compute $U \leftarrow \mathsf{SimUGen}(1^\lambda)$ and sends $\mathsf{PP} = \big(pk_{\mathsf{HE}}, U, \mathsf{Prog}\{pk_{\mathsf{wCCA}}\}, P_1, P_2\big)$ to $\mathcal{A}$.

$\mathcal{A}$ then asks for $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$, $\mathsf{Sign}(\cdot, \cdot)$ queries, which $\mathcal{B}$ can simulate perfectly by using sk_{wCCA}. Finally, $\mathcal{A}$ outputs a forgery $\sigma^*_{\mathsf{agg}} = (t^*, s^*)$ and tuples $\{(id_i, m_i)\}_i$. If $\mathcal{A}$ wins as per the winning conditions (which are the same in both Game $5, j$-2 and Game $5, j$-3), output 0, else output 1.

Clearly, if $z' = F(K, y)$, then this corresponds to Game $5, j$-2; if $z' \leftarrow \{0,1\}^\ell$, it corresponds to Game $5, j$-3. This completes our proof.

Lemma 8. *For any j, any PPT adversary $\mathcal{A}$, $\mathsf{Adv}^{5,j\text{-}3}_{\mathcal{A}} - \mathsf{Adv}^{5,j\text{-}4}_{\mathcal{A}} \leq \mathsf{Adv}_f(\lambda)$.*

Proof. Suppose there exists a PPT adversary $\mathcal{A}$ such that $\mathsf{Adv}^{5,j\text{-}3}_{\mathcal{A}} - \mathsf{Adv}^{5,j\text{-}4}_{\mathcal{A}} = \epsilon$. We will construct a PPT algorithm $\mathcal{B}$ that inverts the one way function f using $\mathcal{A}$.

Note that the only way an adversary can distinguish between Game $5, j$-3 and Game $5, j$-4 is by submitting a forgery $\sigma^*_{\mathsf{agg}} = (t^* = j+1, s^*)$ and tuples $\{(id_i, m_i)\}_i$ such that $\big(\{(vk_{id_i}, id_i, m_i)\}_i, (t^* = j+1, s^*)\big)$ is (i^*, sk_{HE})-rejecting and $\mathsf{Prog\text{-}2}\big\{y, z, K\{y\}\big\}\big(\{(vk_{id_i}, id_i, m_i)\}_i, (t^* = j+1, s^*)\big) = 1$. From the definition of $\mathsf{Prog\text{-}2}\big\{y, z, K\{y\}\big\}$, it follows that $f(\oplus_{i \neq i^*} F(K, vk_{id_i} \| id_i \| m_i \| i \| t) \oplus s^*) = z$.

$\mathcal{B}$ receives z from the OWF challenger. On receiving (id^*, m^*) from $\mathcal{A}$, $\mathcal{B}$ chooses $i^* \leftarrow [n]$, $(pk_{\mathsf{wCCA}}, sk_{\mathsf{wCCA}})$, $(pk_{\mathsf{HE}}, sk_{\mathsf{HE}})$ and compute ciphertext $\mathsf{ct}_i \leftarrow \mathsf{HE.Enc}(pk_{\mathsf{HE}}, 0)$ for all $i \in [n]$ and $i \neq i^*$, $\mathsf{ct}_{i^*} \leftarrow \mathsf{HE.Enc}(pk_{\mathsf{HE}}, 1)$. Let $\mathsf{Prog}\{pk_{\mathsf{wCCA}}\}$ be circuit as defined in the Fig. 1. It chooses $K \leftarrow \mathsf{PRF.Setup}(1^\lambda)$, and let $y = vk_{id_{i^*}} \| id_{i^*} \| m_{i^*} \| i^* \| (j+1)$, $K\{y\} \leftarrow F.\mathsf{Puncture}(K, y)$. It computes $P_1 = i\mathcal{O}(\mathsf{Prog}\big\{K\{y\}, \mathsf{ct}_1, \ldots, \mathsf{ct}_n\big\})$ and $P_2 = i\mathcal{O}(\mathsf{Prog}\{y, z, K\{y\}\})$. $\mathcal{B}$ computes $(vk_{id_{i^*}}, c_{id_{i^*}}) \leftarrow \mathsf{Prog}\{pk_{\mathsf{wCCA}} \| id_{i^*}(r)$ for uniformly random $r = r_0 \| r_1 \in \{0,1\}^{\ell_{\mathsf{inp}}}$, where $(vk_{id_{i^*}}, sk_{id_{i^*}}) \leftarrow \mathsf{SIG.Setup}(1^\lambda; r_0)$ and $c_{id_{i^*}} \leftarrow \mathsf{PKE.Enc}_{\mathsf{wCCA}}(pk_{\mathsf{wCCA}}, 1^\lambda; r_1)$. Then $\mathcal{B}$ compute $U \leftarrow \mathsf{SimUGen}(1^\lambda)$ and sends $\mathsf{PP} = \big(pk_{\mathsf{HE}}, U, \mathsf{Prog}\{pk_{\mathsf{wCCA}}\}, P_1, P_2\big)$ to $\mathcal{A}$.

$\mathcal{A}$ then asks for $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$, $\mathsf{Sign}(\cdot, \cdot)$ queries, which $\mathcal{B}$ can simulate perfectly by using sk_{wCCA}. Finally, $\mathcal{A}$ outputs a forgery $\sigma^*_{\mathsf{agg}} = (t^* = j+1, s^*)$ and tuples $\{(id_i, m_i)\}_i$. $\mathcal{B}$ sends $\oplus_{i \neq i^*} F(K, vk_{id_i} \| id_i \| m_i \| i \| t) \oplus s^*$ as inverse of z to the OWF challenger, and clearly, $\mathcal{B}$ wins if $\mathcal{A}$ wins. This completes our proof.

Lemma 9. *For any j, any PPT adversary $\mathcal{A}$, $\mathsf{Adv}^{5,j\text{-}4}_{\mathcal{A}} - \mathsf{Adv}^{5,j\text{-}5}_{\mathcal{A}} \leq \mathsf{Adv}_{PRF}(\lambda)$.*

Proof. Similar to the proof of Lemma 7.

Lemma 10. *For any j, any PPT adversary $\mathcal{A}$, $\mathsf{Adv}^{5,j\text{-}5}_{\mathcal{A}} - \mathsf{Adv}^{5,j\text{-}6}_{\mathcal{A}} \leq 2\mathsf{Adv}_{i\mathcal{O}}(\lambda)$.*

Proof. Similar to the proof of Lemma 6.

Lemma 11. *For any j, any PPT adversary $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{5,j\text{-}6} - \mathsf{Adv}_{\mathcal{A}}^{5,(j+1)} \leq \mathsf{Adv}_{i\mathcal{O}}(\lambda)$.*

Proof. Similar to the proof of Lemma 5.

Lemma 12. *For any PPT adversary $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{6} \leq \mathsf{Adv}_{\mathsf{SIG}}(\lambda)$.*

Proof. Suppose $\mathsf{Adv}_{\mathcal{A}}^{6} = \epsilon$. We will construct a PPT algorithm $\mathcal{B}$ that breaks the security of SIG with advantage ϵ.

$\mathcal{B}$ receives vk from the challenger of signature scheme SIG. On receiving (id^*, m^*) from $\mathcal{A}$, $\mathcal{B}$ chooses $i^* \leftarrow [n]$, $(pk_{\mathsf{wCCA}}, sk_{\mathsf{wCCA}})$, $(pk_{\mathsf{HE}}, sk_{\mathsf{HE}})$ and compute ciphertext $\mathsf{ct}_i \leftarrow \mathsf{HE.Enc}(pk_{\mathsf{HE}}, 0)$ for all $i \in [n]$ and $i \neq i^*$, $\mathsf{ct}_{i^*} \leftarrow \mathsf{HE.Enc}(pk_{\mathsf{HE}}, 1)$. Let $\mathsf{Prog}\{pk_{\mathsf{wCCA}}\}$ be circuit as defined in the Fig. 1. It chooses $K \leftarrow \mathsf{PRF.Setup}(1^\lambda)$ and computes $P_1 = i\mathcal{O}(\mathsf{Prog}\{K, \mathsf{ct}_1, \ldots, \mathsf{ct}_n\})$ and $P_2 = i\mathcal{O}(\mathsf{Prog}\{K\})$. $\mathcal{B}$ computes $c_{id_{i^*}} \leftarrow \mathsf{PKE.Enc}_{\mathsf{wCCA}}(pk_{\mathsf{wCCA}}, 1^\lambda; r_1)$ for uniformly random $r = r_0 \| r_1 \in \{0,1\}^{\ell_{\mathsf{inp}}}$ and sets $\mathsf{Prog}\{pk_{\mathsf{wCCA}} \| id_{i^*}(r) = (vk, c_{id_{i^*}})$. Then $\mathcal{B}$ compute $U \leftarrow \mathsf{SimUGen}(1^\lambda)$ and sends $\mathsf{PP} = \big(pk_{\mathsf{HE}}, U, \mathsf{Prog}\{pk_{\mathsf{wCCA}}\}, P_1, P_2\big)$ to $\mathcal{A}$.

$\mathcal{A}$ then asks for $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$ queries for $id \neq id_{i^*}$, which $\mathcal{B}$ can simulate perfectly by using sk_{wCCA}. On $\mathcal{A}$'s $\mathsf{Sign}(\cdot, \cdot)$ queries, if $id = id_{i^*}$ AND $m \neq m_{i^*}$, $\mathcal{B}$ forwards m to the signing oracle of signature scheme SIG, and receives σ, which is sent to $\mathcal{A}$ as response; if $id \neq id_{i^*}$, $\mathcal{B}$ generates the signature of m by using $\mathsf{sk}_{id} \leftarrow \mathsf{PKE.Dec}_{\mathsf{wCCA}}(sk_{\mathsf{wCCA}}, c_{id})$.

Finally, $\mathcal{A}$ outputs a forgery $\sigma^*_{\mathsf{agg}} = (t^*, s^*)$ and tuples $\{(id_i, m_i)\}_i$. $\mathcal{A}$ wins if id^* has not been asked to the $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$ oracle, (id^*, m^*) has not been submitted to the $\mathsf{Sign}(\cdot, \cdot)$ oracle, and $\mathsf{SIG.Vefy}(vk, m^*, \mathsf{HE.Dec}(sk_{\mathsf{HE}}, t^*)) = 1$. It sends $(m^*, \mathsf{HE.Dec}(sk_{\mathsf{HE}}, t^*))$ as forgery. Note that $\mathcal{B}$ wins the signature game if $\mathcal{A}$ wins Game 6. This concludes our proof.

5 Conclusions

In this work, we consider n-bounded identity-based aggregate signatures (IBAS), which requires at most n signatures can be aggregated. We also provide a generic transformation to build n-bounded IBAS scheme from any secure signature scheme by using indistinguishability obfuscation and selective one-time universal parameters scheme. Based on the sub-exponential hardness of indistinguishability obfuscation, puncturable PRF and one-way functions, we prove that our n-bounded IBAS scheme is selectively secure in the standard model.

A Appendix

1 Public Key Encryption

Definition 6. *A public-key encryption scheme (PKE) consists of PPT algorithms $\mathsf{PKE} = (\mathsf{PKE.Setup}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$.*

- **Key Generation**. *PKE.Setup takes as input security parameter* 1^λ *and returns a key pair* (pk, sk).
- **Encryption**. *PKE.Enc takes as input public key* pk *and message* m, *and returns a ciphertext* $c \leftarrow \mathsf{PKE.Enc}(pk, m)$.
- **Decryption**. *PKE.Dec takes as input secret key* sk *and ciphertext* c, *and returns a message* $m \leftarrow \mathsf{PKE.Dec}(sk, c)$.

Correctness. *For all* $\lambda \in \mathbb{N}$, $(pk, sk) \leftarrow \mathsf{PKE.Setup}(1^\lambda)$, *messages* $m \in \mathcal{M}(\lambda)$, *we require that* $\mathsf{PKE.Dec}(sk, \mathsf{PKE.Enc}(pk, m)) = m$.
We say that public-key encryption scheme PKE is wCCA secure, if

$$\left|\Pr[\mathsf{Exp}^{\mathsf{wCCA\text{-}0}}_{\mathsf{PKE},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{wCCA\text{-}1}}_{\mathsf{PKE},\mathcal{A}}(\lambda) = 1]\right| \leq negl(\lambda)$$

for some negligible function negl and for all PPT attackers $\mathcal{A}$, *where* $\mathsf{Exp}^{\mathsf{wCCA\text{-}b}}_{\mathsf{PKE},\mathcal{A}}(\lambda)$ *is the following experiment with scheme PKE and attacker* $\mathcal{A}$:

1. $(pk, sk) \leftarrow \mathsf{PKE.Setup}(1^\lambda)$.
2. $(m_0, m_1) \leftarrow \mathcal{A}(1^\lambda, pk)$.
3. $b \leftarrow \{0,1\}$ *and compute* $c^* \leftarrow \mathsf{PKE.Enc}(pk, m_b)$.
4. $b' \leftarrow \mathcal{A}^{\mathcal{O}_{wCCA}}(1^\lambda, c^*)$.
   Here $\mathcal{O}_{\mathsf{wCCA}}$ *is an oracle that on input* c *returns* $\mathsf{PKE.Dec}(sk, c)$ *for all* $c \neq c^*$.

Note that this is a weakened version of standard IND-CCA security, because the attacker has access to $\mathcal{O}_{\mathsf{wCCA}}$ only after seeing the challenge ciphertext.

2 Signature Schemes

Definition 7. *A signature scheme with message space* $M(\lambda)$, *signature key space* $\mathcal{SK}(\lambda)$ *and verification key space* $\mathcal{VK}(\lambda)$ *consists of PPT algorithms* $\mathsf{SIG} = (\mathsf{SIG.Setup}, \mathsf{SIG.Sign}, \mathsf{SIG.Vefy})$:

- **Key Generation**. *SIG.Setup is a randomized algorithm that takes as input security parameter* 1^λ *and outputs signing key* $\mathsf{sk} \in \mathcal{SK}$ *and verification key* $\mathsf{vk} \in \mathcal{VK}$.
- **Signature Generation**. *SIG.Sign takes as input the signing key* $\mathsf{sk} \in \mathcal{SK}$ *and a message* $m \in M$ *and outputs a signature* σ.
- **Verification**. *SIG.Vefy takes as input a verification key* $\mathsf{vk} \in \mathcal{VK}$, *message* $m \in \mathcal{M}$ *and signature* σ *and outputs either 0 or 1.*

Correctness. *For all* $\lambda \in \mathbb{N}$, $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{SIG.Setup}(1^\lambda)$, *messages* $m \in \mathcal{M}(\lambda)$, *we require that* $\mathsf{SIG.Vefy}(\mathsf{vk}, \mathsf{SIG.Sign}(\mathsf{sk}, m)) = 1$.
We say that signature scheme $\mathsf{SIG} = (\mathsf{SIG.Setup}, \mathsf{SIG.Sign}, \mathsf{SIG.Vefy})$ *is existentially unforgeable under a chosen message attack if*

$$\Pr[\mathsf{Exp}^{\mathsf{uf\text{-}cma0}}_{\mathsf{SIG},\mathcal{A}}(\lambda) = 1] \leq negl(\lambda)$$

for some negligible function negl and for all PPT attackers $\mathcal{A}$, *where* $\mathsf{Exp}^{\mathsf{uf\text{-}cma}}_{\mathsf{SIG},\mathcal{A}}(\lambda)$ *is the following experiment with scheme SIG and attacker* $\mathcal{A}$:

1. $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{SIG.Setup}(1^\lambda)$.
2. $(m, \sigma) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{sk},\cdot)}(1^\lambda, pk)$.
   If $\mathsf{SIG.Vefy}(\mathsf{vk}, m, \sigma) = 1$ *and* m *was not queried to* $\mathsf{Sign}(\mathsf{sk}, \cdot)$ *oracle Then return 1 else return 0.*

3 Additively Homomorphic Encryption

Definition 8. *An additively homomorphic encryption scheme with message space $\mathbb{F}_p$ and ciphertext space $\mathcal{C}_{HE}$ consists of PPT algorithms HE=(HE.Setup, HE.Enc,HE.Dec, HE.Add).*

- $HE.Setup(1^\lambda)$ *takes the security parameter* 1^λ *as input and outputs public key* pk*, secret key* sk*.*
- $HE.Enc(pk, m)$ *takes as input a public key* pk *and message* $m \in \mathbb{F}_p$ *and outputs a ciphertext* $ct \in \mathcal{C}_{HE}$*.*
- $HE.Dec(sk, ct)$ *takes as input a secret key* sk*, a ciphertext* $ct \in \mathcal{C}_{HE}$ *and either outputs an element in* $\mathbb{F}_p$ *or* $\perp$*.*
- $HE.Add(pk, ct_1, ct_2)$ *takes as input a public key* pk *and two ciphertexts* $ct_1, ct_2 \in \mathcal{C}_{HE}$ *and outputs a ciphertext* ct*.*

Correctness. *Let p be any prime and q any polynomial in λ. For all $\lambda \in \mathbb{N}$, $(pk, sk) \leftarrow HE.Setup(1^\lambda)$, q messages $m_1, \ldots, m_q \in \mathbb{F}_p$, the following holds*

$$HE.Dec(sk, HE.Enc(pk, m_1) + \ldots + HE.Enc(pk, m_q)) = m_1 + \ldots + m_q.$$

We say that additively homomorphic encryption scheme HE is IND-CPA secure, if

$$\left|\Pr[Exp^{CPA\text{-}0}_{HE,\mathcal{A}}(\lambda) = 1] - \Pr[Exp^{CPA\text{-}1}_{HE,\mathcal{A}}(\lambda) = 1]\right| \leq negl(\lambda)$$

for some negligible function negl and for all PPT attackers $\mathcal{A}$, where $Exp^{CPA\text{-}b}_{HE,\mathcal{A}}(\lambda)$ is the following experiment with scheme HE and attacker $\mathcal{A}$:

1. $(pk, sk) \leftarrow HE.Setup(1^\lambda)$.
2. $(m_0, m_1) \leftarrow \mathcal{A}(1^\lambda, pk)$.
3. $b \leftarrow \{0, 1\}$ *and compute* $c^* \leftarrow HE.Enc(pk, m_b)$.
4. $b' \leftarrow \mathcal{A}(1^\lambda, c^*)$.

References

1. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S.P., Yang, K.: On the (im)possibility of obfuscating programs. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, p. 1. Springer, Heidelberg (2001)
2. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 416–432. Springer, Heidelberg (2003)
3. Boldyreva, A., Gentry, C., ONeill, A., Yum, D.H.: Ordered multisignatures and identity-based sequential aggregate signatures, with applications to secure routing. Cryptology ePrint Archive, report 2007/438 (2010). Revised 21 February 2010
4. Bagherzandi, A., Jarecki, S.: Identity-based aggregate and multi-signature schemes based on RSA. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 480–498. Springer, Heidelberg (2010)

5. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: FOCS (2013)
6. Galindo, D., Herranz, J., Kiltz, E.: On the generic construction of identity-based signatures with additional properties. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 178–193. Springer, Heidelberg (2006)
7. Gentry, C., Ramzan, Z.: Identity-based aggregate signatures. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958, pp. 257–273. Springer, Heidelberg (2006)
8. Hofheinz, D., Jager, T., Khurana, D., Sahai, A., Waters, B., Zhandry, M.: How to generate and use universal parameters. Cryptology ePrint Archive, report 2014/507 (2014). http://eprint.iacr.org/
9. Hohenberger, S., Koppula, V., Waters, B.: Universal signature aggregators. Cryptology ePrint Archive, report 2014/745 (2014). http://eprint.iacr.org/
10. Hohenberger, S., Sahai, A., Waters, B.: Full domain hash from (leveled) multilinear maps and identity-based aggregate signatures. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 494–512. Springer, Heidelberg (2013)
11. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
12. Sahai, A., Waters, B.: How to use indistinguishability obfuscation: deniable encryption, and more. In: STOC, pp. 475–484 (2014)

http://www.springer.com/978-3-319-23317-8

Information Security
18th International Conference, ISC 2015, Trondheim, Norway, September 9-11, 2015, Proceedings
Lopez, J.; Mitchell, C.J. (Eds.)
2015, XIII, 570 p. 110 illus., Softcover
ISBN: 978-3-319-23317-8