Universal Signature Aggregators

Susan Hohenberger*
Johns Hopkins University
susan@cs.jhu.edu

Venkata Koppula
University of Texas at Austin
kvenkata@cs.utexas.edu

Brent Waters†
University of Texas at Austin
bwaters@cs.utexas.edu

Abstract

We introduce the concept of universal signature aggregators. In a universal signature aggregator system, a third party, using a set of common reference parameters, can aggregate a collection of signatures produced from *any* set of signing algorithms (subject to a chosen length constraint) into one short signature whose length is independent of the number of signatures aggregated. In prior aggregation works, signatures can only be aggregated if all signers use the same signing algorithm (e.g., BLS) and shared parameters. A universal aggregator can aggregate across schemes even in various algebraic settings (e.g., BLS, RSA, ECDSA), thus creating novel opportunities for compressing authentication overhead. It is especially compelling that *existing* public key infrastructures can be used and that the signers do not have to alter their behavior to enable aggregation of their signatures.

We provide multiple constructions and proofs of universal signature aggregators based on indistinguishability obfuscation and other supporting primitives. We detail our techniques as well as the tradeoffs in features and security of our solutions.

1 Introduction

An aggregate signature system, as introduced by Boneh, Gentry, Lynn and Shacham [BGLS03], allows a party to bundle a set of signatures together into a single short cryptographic signature. Aggregate signatures are motivated by applications where one needs to simultaneously verify several signatures from different users on different messages in environments with communication or storage resource constraints. For example, Boneh et al. [BGLS03] proposed applying aggregate signatures to Secure BGP [KLMS00] path authentication; later this idea was empirically evaluated by Zhao et al. [ZSN05].

Over the past several years many solutions to aggregate signatures [LOS+06, GR06, BGOY07, BNN07, AGH10] have been proposed that have explored tradeoffs regarding computational cost, security models, features (e.g. identity-based), limitations (e.g. sequential signing), and cryptographic assumptions. However, all of these constructions have one thing in common in that they require *all signers to adopt a common signature system and shared parameters.*

In practice, the common scheme and parameter requirements can be a large barrier to adoption. Existing users will already have established signing keys and algorithms which are entrenched in an existing public key infrastructure. The overhead of changing and re-certifying one's public keys may very well overwhelm

*Supported by the National Science Foundation (NSF) CNS-1154035 and CNS-1228443; the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL) under contract FA8750-11-2-0211, the Office of Naval Research under contract N00014-14-1-0333, and a Microsoft Faculty Fellowship.

†Supported by NSF CNS-0952692, CNS-1228599 and CNS-1414082. DARPA through the U.S. Office of Naval Research under Contract N00014-11-1-0382, Google Faculty Research award, the Alfred P. Sloan Fellowship, Microsoft Faculty Fellowship, and Packard Foundation Fellowship.

the perceived benefit of creating signatures that can be aggregated by a third party. Indeed the original signer might not even be incentivized to allow aggregation in the first place when the benefits fall to the aggregating party or verifier of the signatures. Furthermore, even if a user moved from one signature system to an aggregate signature system, all previously created signatures would be unaggregatable.[1]

Universal Signature Aggregators We introduce the concept of universal signature aggregators. In a universal signature aggregator system, a third party, using a set of common reference parameters, can aggregate a collection of signatures produced from *any* set of signing algorithms (subject to a chosen length constraint) into one short signature whose length is independent of the number of signatures aggregated. A verifier can use the common parameters to verify the aggregate signature. The system will be secure in the sense that it is hard to produce an aggregate signature on a verification algorithm, verification key, message tuple, $(\mathsf{Verify}, \mathrm{VK}, m)$ unless the holder of the corresponding secret key produced a signature on m. We emphasize that signers in the system need not do anything special to allow aggregation; indeed they could be unaware of the existence of such a system.

Our central challenge is to create a way to compress many signatures of varying types into one short object. Prior solutions required all signatures to reside in a common (often bilinear) group, where it was possible to leverage homomorphic properties of the group structure. Here we are afforded no such luxury as signatures will reside in different groups or even be based on a scheme with no algebraic structure.

Our approach will be to overcome these limitations by applying the tool of program obfuscation. At the highest level, a trusted setup routine will produce a pair of a global signature verification key for a universal signature aggregator and a shared obfuscated program. The job of the obfuscated program will be to take as input tuples of the form $(\mathsf{Verify}, \mathrm{VK}, m, \sigma)$ that respectively represent verification algorithm, verification key, message and signature 4-tuples. The program will first verify using algorithm Verify and key VK that σ is a signature on m. If this check passes, it will produce a signature using a master secret key on the message $\textsc{Msg} = (\mathsf{Verify}, \mathrm{VK}, m)$ — essentially transforming the arbitrary signature into one of an aggregateable form.

At first glance it might appear that obfuscation provides an open and close solution to our problem. Indeed, if we heuristically model the obfuscated program as an oracle to the program, the analysis is relatively straightforward. However, as noted by Hada [Had00] such a definition is impossible to achieve for any functionality. Our goal is to create probably secure constructions under a realizable definition of obfuscation — ideally indistingusihability obfuscation.

Achieving provable security under indistingusihability obfuscation (and without knowledge assumptions [2]) presents significant challenges. The primary technical challenge is how to design a construction and corresponding reduction that can extract a forgery on an arbitrary input signature scheme from an attacker that forges on the aggregate. We emphasize that without an oracle interface or knowledge assumption a reduction is not afforded the opportunity to simply "look at" the input signatures.

Universally Aggregating Unique Signatures We begin by exploring how to universally aggregate unique signatures — a unique signature system [GO93] is one where there is at most one signature that will verify per message. Notably, RSA based full domain hash [BR93, BR96] are unique signatures that form the basis of the widely deployed PKCS#1 standard [KS98]. As evidence of the wide scale deployment, Heninger et al. [HDWH12] performed an Internet-wide scan of machines responding on the TLS and SSH ports for IPv4 space and reported 3.9 million distinct RSA keys compared to only 1.9 thousand DSA keys.

Our construction will be parameterized by four polynomial functions over the security parameter: $\ell_{\mathrm{ver}}(\lambda)$, $\ell_{\mathrm{vk}}(\lambda)$,$\ell_{\mathrm{msg}}(\lambda)$, $\ell_{\mathrm{sig}}(\lambda)$. These respectively represent a bound on the size of verification circuits, verification keys, length of messages signed and size of signatures that are aggregated. While we are interested in signatures of arbitrary length messages, in practice almost all signature schemes will apply the "hash and

[1] We note that the concept of integrating "special property" cryptography into existing keys is relatively unexplored, but has been considered in ring signatures [BKM08] and deniable encryption [SW14].

[2] A different direction is to attempt to build universal aggregation from succinct arguments of knowledge (SNARKs)[BCCT13]. We aim to achieve our goals without applying knowledge assumptions.

sign" paradigm where a longer message is first hashed down to a fixed size hash value (dependent on the security parameter). The core signature scheme then signs this value.

In our first construction (see Section 4), the $\mathsf{UniversalSetup}$ first chooses an RSA modulus N and exponent $e \leftarrow \mathbb{Z}^*_{\phi(N)}$. Next, it chooses a puncturable PRF [BW13, BGI13, KPTZ13, SW14] key K for a function F that takes inputs of the form $(\mathsf{Verify}, \mathrm{VK}, m) \in \{0,1\}^{\ell_{\mathrm{ver}}} \times \{0,1\}^{\ell_{\mathrm{vk}}} \times \{0,1\}^{\ell_{\mathrm{msg}}}$ (i.e., 3-tuples representing a verification circuit, verification key and message). The puncturable PRF will output into $\mathbb{Z}_N$.

Finally, the setup will publish (indistinguishability) obfuscations of two programs. The first is $\mathsf{Transform}_{N,K}$. This program takes as input a 4-tuple $\mathsf{Verify}, \mathrm{VK}, m, \sigma$. It then computes $\mathsf{Verify}(\mathrm{VK}, m, \sigma)$, which verifies the signature under the algorithm. If the signature verifies, the program outputs $F(K, \mathsf{Verify}, \mathrm{VK}, m) \in \mathbb{Z}_N$. This can be thought of as a "transformed signature" where the obfuscated program maps the original signature into one over $\mathbb{Z}_N$. The second program is called $\mathsf{Transform\text{-}Image}_{N,K,e}$. On input $(\mathsf{Verify}, \mathrm{VK}, m)$, it computes $F(K, \mathsf{Verify}, \mathrm{VK}, m)^e \pmod N$.

One can now aggregate a sequence of signatures $(\mathsf{Verify}_i, \mathrm{VK}_i, m_i, \sigma_i)$ by transforming each one by computing[3] $s_i = \mathsf{Transform}_{N,K}(\mathsf{Verify}_i, \mathrm{VK}_i, m_i, \sigma_i)$ and then aggregating into one element of $\mathbb{Z}_N$ as $\sigma_{\mathrm{agg}} = \prod_i s_i$. To verify an aggregate signature, σ_{agg}, on $(\mathsf{Verify}_i, \mathrm{VK}_i, m_i)$ simply compute $t_i = \mathsf{Transform}_{N,K}(\mathsf{Verify}_i, \mathrm{VK}_i, m_i, \sigma_i)$ and test whether $\sigma^e_{\mathrm{agg}} \stackrel{?}{=} \prod_i t_i$. [4] Essentially the $\mathsf{Transform}$ program maps an arbitrary signature to an RSA FullDomain hash type signature on the "message" $(\mathsf{Verify}_i, \mathrm{VK}_i, m_i)$.

We prove selective security where the attacker declares before seeing the public parameters a message m^* that they will forge on.[5] Our security argument is centered around an alternative program $\mathsf{Transform\text{-}Reject}$ which is programmed to behave the same as $\mathsf{Transform}$ except on input $y = (\mathsf{Verify}^*, \mathrm{VK}^*, m^*)$ on which it always outputs $\perp$ *even if it is given a valid signature on* m^*. It also uses a PRF key that is punctured at y.

Security follows from two primary arguments about the program. We first establish that if an attack algorithm, Att, is successful when given $\mathsf{Transform}$, it must be almost as successful when given $\mathsf{Transform\text{-}Reject}$; otherwise, the underlying unique signature scheme is broken. Suppose that there is an attacker, Att, with a non-negligible difference in advantage between these two games, then we can build an reduction algorithm that extracts the unique signature on m^* in a bit by bit fashion. The reduction algorithm runs as the challenger in the aggregate signature game and receives a challenge verification key from the challenger in the standard signature security game. It runs to the point in the security game where the input public key and parameters are established and saves the state of the game (including the state of Att). Then for each bit of the signature it performs the following process multiple times. It runs a third program $\mathsf{TransformAlt}_{y,j}$. This program runs as $\mathsf{Transform}$, but rejects if the j-th bit of the input signature is 1. For each j, it runs the experiment multiple times with fresh randomness. If the measured advantage of the attacker drops when using $\mathsf{TransformAlt}_{y,j}$ then it guesses that the j-th bit of the signature is 1; otherwise it guesses that it is 0. It complies all of these guesses together to output a forgery. (The amount of rewinding needed depends on the difference in advantage. In addition, our actual analysis addresses other technical details.)

Since signatures are unique, the program $\mathsf{TransformAlt}_{y,j}$ is functionally equivalent to $\mathsf{Transform}$ if the j-th bit of the unique signature on m^* is 0 and thus by indistinguishability obfuscation the attacker's advantage should be negligibly close in these two cases. Similarly, $\mathsf{TransformAlt}_{y,j}$ is functionally equivalent to $\mathsf{Transform\text{-}Reject}$ if the j-th signature bit is 1 and again by indistinguishability obfuscation the advantage should be close to that of $\mathsf{Transform\text{-}Reject}$.

After we have established that the advantage when given $\mathsf{Transform\text{-}Reject}$ is close to that of $\mathsf{Transform}$,we show that an attacker that can win when given $\mathsf{Transform\text{-}Reject}$ will either break $i\mathcal{O}$, the punctured PRF or the RSA assumption and roughly follows [HSW14] using punctured programming [SW14] techniques. The main proof innovation is combining a rewinding argument with indistinguishability obfuscation to extract a unique signature.

We also show a variation of this idea in Appendix A that is a universal aggregator of unique signatures,

[3] We slightly abuse notation in the introduction for ease of exposition by using the names $\mathsf{Transform}$ and $\mathsf{Transform\text{-}Image}$ to refer both to the obfuscated and unobfuscated forms of the program. In the main body, we are careful about these distinctions.

[4] We require in verification that no 3-tuples are repeated. I.e., for all $i \neq j$, $(\mathsf{Verify}_i, \mathrm{VK}_i, m_i,) \neq (\mathsf{Verify}_j, \mathrm{VK}_j, m_j)$.

[5] The usual complexity leveraging arguments for adaptive security can be applied here if we are willing to make sub-exponential hardness assumptions.

but where we avoid using the RSA assumption. (Indistinguishability obfuscation and punctured PRFs are still used.) The tradeoff is that there is an a priori bound n on the number of signatures that can be aggregated. In the construction, the parameters will grow polynomially with n, but the size of the signatures is independent of n. We also conjecture that in our main construction the RSA-type transformed signature can be replaced by a BLS [BLS01] type signature (in an analogous way to [HSW14]), but do not formally show this.

Universal Aggregation of arbitrary signatures using VBB Obfuscation While covering unique signatures achieves progress, we want to push toward our central goal of aggregating arbitrary signatures. Our next step is to show that a slight tweak to the previous construction gives us a universal aggregator of arbitrary signatures under a specific virtual black box (VBB) assumption. This appears in Section 5.

It might first seem that a solution proven under a VBB assumption is not better than the oracle heuristic outlined earlier. However, achieving a VBB proof provides both a sounder justification and is more technically challenging than the oracle heuristic. First, modeling an obfuscated program as an oracle is a heuristic — a piece of code is clearly a different object than an oracle. In contrast, a VBB assumption could be true for many functionalities even though there exists certain functionalities for which it cannot hold [BGI$^+$01].

Proving our construction secure under a VBB definition presents an interesting technical barrier. A natural proof methodology is to first say that an obfuscator for a given circuit cannot be more successful than a simulator with oracle access to the same circuit using VBB. And then making further hybrid security arguments leveraging the fact that the simulator has oracle access. The primary problem with this strategy is that while the universal aggregator security game gives the attacker access to a signing oracle, there is no place to "put" this signing oracle when applying the VBB security game.

We overcome this obstacle by introducing a new technique that we call "oracle assimilation" which we believe might be of independent interest. In our construction, the Transform-VBB program behaves in almost the same way as Transform before except an extra mode bit is added to the input. If this mode bit b is set to 0, it indicates `normal` input and the Transform-VBB program operates roughly as described above. If the mode bit is set to 1, it indicates `query` input and the program outputs a rejecting $\perp$ on all inputs of this type. The `query` type input is only used in the proof and not in the construction.

Our proof of security proceeds by a sequence of games. In the initial security game, all `query` inputs output a rejecting $\perp$. The proof (in a couple of steps) then moves to a game where the `query` inputs will take a form of (a, m) and output a signature on m under the challenge input secret signing key if $\mathrm{PRG}(a) = \mathrm{PRG}(\alpha)$ for some value α chosen by the game, but hidden from the attacker. We can argue this change is indiscernable to the attacker by $i\mathcal{O}$ and pseudorandom generator security. At this point the security game will use the `query` interface of the obfuscated program to answer signing queries and we can say that the signing oracle was "assimilated" into the obfuscated program. Next, we can use VBB security to argue that there must exist a simulator with oracle access to the program that outputs 1 with close to the same probability that the attacker wins. Now that the input signing algorithm is accessed by an oracle we can use its security to argue that the game is indistinguishable from when the circuit refuses to transform on m^*, the challenge message.[6] Finally, we use VBB again to reason about the attack algorithm's advantage when given this second circuit that will not transform on m^*. From here, the proof follows as in the unique signature case.

Stepping back, the main innovation for this proof is to use punctured programming techniques to subliminally assimilate the signing oracle for one scheme into the obfuscated program, then use the VBB interface to execute the proof. We expect that this technique will be useful in other contexts. One interesting view is that we could apply either this VBB argument for arbitrary signatures or the previous $i\mathcal{O}$ argument for unique signatures to this single construction. So a user with any signature scheme would get VBB based security and if a user had a unique signature scheme, she would get the added benefit of $i\mathcal{O}$ based security.

[6] For ease of exposition, the proof in the main body proves selective security; however, we show how a minor transformation of the construction using admissible hash functions [BB04] gives adaptive security in Appendix B.

Aggregating arbitrary signatures using indistinguishability obfuscation For our final contribution, we return to our goal of aggregating arbitrary signatures using indistinguishability obfuscation. Our primary challenge again is how to extract an input forgery from the attacker in a proof. The previous two methods used the structure of a unique signature and an oracle interface, neither of which is available to us now.

We overview the main solution ideas and our proof approach. At a high level, we devise a means for being able to extract and check the validity of a single signature (from the aggregate) of our choice in the proof without the adversary being able to know which one we are "looking at". Thus, we build our confidence in the validity of all the signatures by being able to check any given one of them. We call this an "enforce all by one" technique.

To do this, we first use additively (or singly) homomorphic encryption to combine the encryptions of several signatures together into one object t. Then we will have an obfuscated program generate a PRF-type signature component s on a message representing ciphertext tag t along with tuples $\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}$ if the input contains valid signatures on each message. The output aggregate signature is $\sigma_{\mathrm{agg}} = (t, s)$. Although the homomorphic ciphertext t will not be large enough to contain all of the input signatures, in the proof it can be used to remember one of the input signatures and thus provide us with an opportunity to extract a forgery on the input signature. The difficulty is in using $i\mathcal{O}$ to ensure that an attacker can only output a verifying $\sigma_{\mathrm{agg}} = (t, s)$ on a ciphertext "tag" t that contains a proper forgery in the proof.

Diving in a little further in our solution, the setup algorithm will be parameterized by a polynominal $n(\cdot)$ that gives an a-priori bound on the number of signatures that can be verified. The size of the parameters will grow polynomially with n, but the signature size will be independent of it. The setup algorithm will output n ciphertexts $\{\mathsf{count}_i \leftarrow \mathsf{HE.enc}(\mathsf{pk}, 0)\}_{i=1,\ldots n}$ each of which is an encryption of 0.

The universal aggregation algorithm takes input $\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i, \sigma_i\}$. It then computes $t = \Sigma_i \mathsf{count}_i \cdot \sigma_i$. Next it will input t and the tuples $\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i, \sigma_i\}$ to an obfuscated program $\mathsf{AggSign}$ which will evaluate and output a punctured PRF on t and $\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}$ if the input signatures verify. (We will return shortly to where the obfuscated program comes from.)

In proving security we perform a sequence of hybrids, where the first step of the hybrid is to guess an index j (incurring a $1/n$ loss) where the forgery occurs. Next, we change count_j to be an encryption of 1. This causes an honestly computed value t to be an encryption of the j-th signature that we will eventually use for extraction.

The challenge at this point is to come up with a formulation of the program $\mathsf{AggSign}$ for which we can prove security using indistinguishability arguments. We provide two approaches. In the first one (see Section 6), we allow $\mathsf{AggSign}$ to be created by a Universal Sampler (also called a Universal Parameters Scheme) as defined by Hofheinz et al. [HJK+14]. A Universal Sampler is allowed to adaptively sample from an arbitrary (efficiently computable) distribution. In this case we sample from an obfuscation of the $\mathsf{AggSign}_t$ program that is parameterized to only work with a given tag value t. As noted in [HJK+14], Universal Samplers are realizable in the random oracle model from indistinguishability obfuscation. So this solution will exist in the random oracle model as well. An advantage of Universal Samplers is that they can define the $\mathsf{AggSign}_t$ program adaptively.

We also propose a second variation of this solution in Section 7 that does not need the random oracle heuristic. Instead it applies complexity leveraging that requires assuming sub-exponential hardness of some of the underlying computational assumptions.

1.1 Summary of our results

Our results are summarized in the following table. The first column labels the construction. The remaining columns indicate: type of signatures that can be aggregated, selective or adaptive security, standard or random oracle model proofs, whether the aggregator is bounded or not, and finally, the cryptographic assumptions used in the security proof. In our assumptions, we prefix them with "subexp" to indicate if sub-exponential hardness is required for complexity leveraging. Since PRFs, PRGs, and (selectively-secure) puncturable PRFs are constructible from one-way functions, we list OWF as the assumption. UPS stands for a universal parameters scheme [HJK+14] (implied by $i\mathcal{O}$ in the random oracle model), HE stands for

singly homomorphic encryption, $i\mathcal{O}$ stands for indistinguishability obfuscation, and VBB stands for virtual black-box obfuscation, where we assume that VBB holds only for a certain limited family of circuits.

Construction	**Type**	**Selective/ Adaptive**	**RO**	**Bounded Aggregator**	**Assumptions**
Section 4	Unique	Selective	No	No	$i\mathcal{O}$, RSA, OWF
Section 5	Arbitrary	Selective[7]	No	No	$i\mathcal{O}$, VBB, OWF
Section 6	Arbitrary	Adaptive	Yes	Yes	$i\mathcal{O}$, UPS, HE, OWF
Section 7	Arbitrary	Selective	No	Yes	subexp-$i\mathcal{O}$, HE, subexp-OWF

2 Preliminaries

2.1 Notations

For any set $\mathcal{X}$, $x \leftarrow \mathcal{X}$ denotes a uniformly random element drawn from $\mathcal{X}$. Given integers $\ell_{\text{ckt}}, \ell_{\text{inp}}, \ell_{\text{out}}$, let $\mathcal{C}[\ell_{\text{ckt}}, \ell_{\text{inp}}, \ell_{\text{out}}]$ denote the set of circuits that can be represented using ℓ_{ckt} bits, take ℓ_{inp} bits as input, and output ℓ_{out} bits.

2.2 Admissible Hash Functions

We recall the notion of *admissible hash functions* due to Boneh and Boyen [BB04]. Here we state a simplified definition from [HSW14].

Definition 2.1. Let l, n and θ be efficiently computable univariate polynomials. Let $h : \{0,1\}^{l(\lambda)} \to \{0,1\}^{n(\lambda)}$ be an efficiently computable function and $\mathsf{AdmSample}$ a PPT algorithm that takes as input 1^λ and an integer q, and outputs $u \in \{0,1,\perp\}^{n(\lambda)}$. For any $u \in \{0,1,\perp\}^{n(\lambda)}$, define $P_u : \{0,1\}^{l(\lambda)} \to \{0,1\}$ as follows: $P_u(x) = 0$ if for all $1 \le j \le n(\lambda), h(x)_j \neq u_j$, else $P_u(x) = 1$ (where u_j denotes the j^{th} bit of u).

We say that $(h, \mathsf{AdmSample})$ is θ-admissible if the following condition holds:

For any efficiently computable polynomial Q, for all $x^1, \ldots, x^{Q(\lambda)}, x^* \in \{0,1\}^{l(\lambda)}$, where $x^* \notin \{x^i\}_i$,

$$Pr[(\forall i \le Q(\lambda), P_u(x^i) = 1) \wedge P_u(x^*) = 0] \ge \frac{1}{\theta(Q(\lambda))}$$

where the probability is taken over $u \leftarrow \mathsf{AdmSample}(1^\lambda, Q(\lambda))$.

Theorem 2.1 (Admissible Hash Function Family [BB04], simplified proof in [FHPS13])**.** For any efficiently computable polynomial l, there exist efficiently computable polynomials n, θ such that there exist θ-admissible function families mapping l bits to n bits.

2.3 Signature Schemes

A signature scheme $\mathcal{S}$ with message space $\mathcal{M}(\lambda)$, signature key space $\mathcal{SK}(\lambda)$ and verification key space $\mathcal{VK}(\lambda)$ consists of the following algorithms.

- $\mathsf{Gen}(1^\lambda)$ The setup algorithm is a randomized algorithm that takes as input security parameter λ and outputs signing key $\text{SK} \in \mathcal{SK}$ and verification key $\text{VK} \in \mathcal{VK}$.
- $\mathsf{Sign}(\text{SK}, m)$ The signing algorithm takes as input the signing key $\text{SK} \in \mathcal{SK}$ and a message $m \in \mathcal{M}$ and outputs a signature σ.
- $\mathsf{Verify}(\text{VK}, m, \sigma)$ The verification algorithm takes as input a verification key $\text{VK} \in \mathcal{VK}$, message $m \in \mathcal{M}$ and signature σ and outputs either 0 or 1.

[7] In Appendix B, we modify this construction to achieve adaptive security without any additional assumptions.

Correctness For all $\lambda \in \mathbb{N}$, $(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathsf{Gen}(1^\lambda)$, messages $m \in \mathcal{M}(\lambda)$, we require that $\mathsf{Verify}(\mathrm{VK}, m, \mathsf{Sign}(\mathrm{SK}, m)) = 1$.

Security The security notion for signature schemes, formalized by Goldwasser, Micali and Rivest [GMR88], is based on the following game between an adversary $\mathcal{A}$ and a challenger.

1. **Setup Phase** Challenger chooses $(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathsf{Gen}(1^\lambda)$.
2. **Signing Phase** $\mathcal{A}$ sends signature query $m_i \in \mathcal{M}$ and receives $\sigma_i \leftarrow \mathsf{Sign}(\mathrm{SK}, m_i)$.
3. **Forgery Phase** $\mathcal{A}$ finally outputs a message m and signature σ.

$\mathcal{A}$ wins if m was not queried during the Signing Phase and $\mathsf{Verify}(\mathrm{VK}, m, \sigma) = 1$. Let $\mathsf{Adv}_{\mathcal{A}}(\lambda) = \Pr[\mathcal{A} \text{ wins }]$.

Definition 2.2. A signature scheme $\mathcal{S}$=($\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify}$) is *existentially unforgeable under a chosen message attack* if for all PPT adversaries $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}(\lambda)$ is negligible in λ.

Goldwasser and Ostrovsky [GO93] introduced the notion of *unique signature schemes* [8]. In a unique signature scheme, there is a unique valid signature corresponding to any message, verification key pair.

Definition 2.3 (Unique Signatures)**.** A signature scheme $\mathcal{S} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ is said to be *unique* if for all tuples $(\mathrm{VK}, m, \sigma_1, \sigma_2)$, either $\sigma_1 = \sigma_2$ or $\mathsf{Verify}(\mathrm{VK}, m, \sigma_1) = 0$ or $\mathsf{Verify}(\mathrm{VK}, m, \sigma_2) = 0$.

In this work, we will be considering signature schemes where the messages, signatures and verification keys have bounded length, and the verification algorithm is deterministic. In practice, most signature schemes use a collision resistant hash function to compress an arbitrary length message to bounded length. We will be dealing with these 'post-hash' messages.

Definition 2.4 ($(\ell_{\mathrm{vk}}, \ell_{\mathrm{msg}}, \ell_{\mathrm{sig}})$-*bounded length* signature scheme)**.** Let ℓ_{vk}, ℓ_{msg} and ℓ_{sig} be fixed polynomials. A signature scheme $\mathcal{S} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ is said to be $(\ell_{\mathrm{vk}}, \ell_{\mathrm{msg}}, \ell_{\mathrm{sig}})$-*bounded length* if all verification keys output by $\mathsf{Gen}(1^\lambda)$ have length at most $\ell_{\mathrm{vk}}(\lambda)$, Sign takes as input messages of length at most $\ell_{\mathrm{msg}}(\lambda)$ and outputs signatures of length bounded by $\ell_{\mathrm{sig}}(\lambda)$.

Since the verification keys, messages and signatures have bounded length, we can view Verify as a circuit with three inputs- verification key VK, message m and signature σ. We assume every circuit can be represented as a binary string.

Definition 2.5 ($(\ell_{\mathrm{ver}}, \ell_{\mathrm{vk}}, \ell_{\mathrm{msg}}, \ell_{\mathrm{sig}})$-*length qualified* signature scheme)**.** Let $\ell_{\mathrm{ver}}, \ell_{\mathrm{vk}}, \ell_{\mathrm{msg}}, \ell_{\mathrm{sig}}$ be fixed polynomials. A $(\ell_{\mathrm{vk}}, \ell_{\mathrm{msg}}, \ell_{\mathrm{sig}})$-bounded length signature scheme $\mathcal{S} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ is said to be (ℓ_{ver}, ℓ_{vk}, ℓ_{msg}, ℓ_{sig})-*length qualified* if the verification circuit Verify and signing circuit Sign can be represented as a binary string of length at most $\ell_{\mathrm{ver}}(\lambda)$ bits.

Abusing notation, we will say that a tuple $(\mathsf{Verify}, \mathrm{VK}, m, \sigma)$ is a $(\ell_{\mathrm{ver}}, \ell_{\mathrm{vk}}, \ell_{\mathrm{msg}}, \ell_{\mathrm{sig}})$-length qualified tuple if Verify is a circuit that can be represented using $\ell_{\mathrm{ver}}(\lambda)$ bits, and VK, m, σ are of length at most $\ell_{\mathrm{vk}}(\lambda)$, $\ell_{\mathrm{msg}}(\lambda)$ and $\ell_{\mathrm{sig}}(\lambda)$ respectively. Similarly, a tuple (Verify, VK, m) is (ℓ_{ver}, ℓ_{vk}, ℓ_{msg})-length qualified if Verify, VK and m have length at most $\ell_{\mathrm{ver}}(\lambda)$, $\ell_{\mathrm{vk}}(\lambda)$ and $\ell_{\mathrm{vk}}(\lambda)$ respectively.

2.4 Additively Homomorphic Encryption

In this work, we will be using encryption schemes which allow us to perform additive operations on ciphertexts. Many encryptions schemes [GM84, Ben87, NS98, OU98, Pai99, DJ03] have the 'additive homomorphism' property. We will now define the syntax and security definition for an additively homomorphic encryption scheme.

Let p be a prime[9]. An additively homomorphic encryption scheme $\mathcal{HE}$ with message space $\mathbb{F}_p$ and ciphertext space $\mathcal{C}_{\mathsf{HE}}$ consists of the following algorithms.

[8] Also known as invariant signature schemes.

[9] The prime p is a property of the encryption scheme.

- $\mathsf{HE.setup}(1^\lambda)$ The setup algorithm takes the security parameter as input and outputs public key pk, secret key sk.
- $\mathsf{HE.enc}(\mathsf{pk}, m)$ The encryption algorithm takes as input a public key pk and message $m \in \mathbb{F}_p$ and outputs a ciphertext $\mathrm{ct} \in \mathcal{C}_{\mathsf{HE}}$.
- $\mathsf{HE.dec}(\mathsf{sk}, \mathrm{ct})$ The decryption algorithm takes as input a secret key sk, a ciphertext $\mathrm{ct} \in \mathcal{C}_{\mathsf{HE}}$ and either outputs an element in $\mathbb{F}_p$ or $\perp$.
- $\mathsf{HE.add}(\mathsf{pk}, \mathrm{ct}_1, \mathrm{ct}_2)$ The addition algorithm takes as input a public key pk and two ciphertexts $\mathrm{ct}_1, \mathrm{ct}_2 \in \mathcal{C}_{\mathsf{HE}}$ and outputs a ciphertext ct.

For simplicity of notation, we will represent $\mathsf{HE.add}(\mathsf{pk}, \mathrm{ct}_1, \mathrm{ct}_2)$ as $\mathrm{ct}_1 + \mathrm{ct}_2$.

Correctness We require the following correctness property:

- Let p be any prime and q any polynomial in λ. For any $\lambda \in \mathbb{N}$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{HE.setup}(1^\lambda)$, q messages $m_1, \ldots, m_q \in \mathbb{F}_p$, the following must hold.

$$\mathsf{HE.dec}(\mathsf{sk}, \mathsf{HE.enc}(m_1) + \ldots + \mathsf{HE.enc}(m_q)) = m_1 + \ldots + m_q.$$

Note that given an encryption ct of message $m \in \mathbb{F}_p$, and a plaintext $a \in \mathbb{F}_p$, one can use $\mathsf{HE.add}$ to compute an encryption of $m \cdot a$ efficiently. Let $a \cdot \mathrm{ct}$ represent this operation.

Security The security game is the usual IND-CPA security game between a challenger and a PPT adversary Att.

1. The challenger chooses $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{HE.setup}(1^\lambda)$ and sends pk to Att.
2. Att sends messages $m_0, m_1 \in \mathbb{F}_p$ to the challenger.
3. The challenger chooses $b \leftarrow \{0, 1\}$, computes $\mathrm{ct}_b \leftarrow \mathsf{HE.enc}(\mathsf{pk}, m_b)$ and sends ct_b to Att.
4. Att finally outputs a guess b'.

Att wins if $b = b'$. Let $\mathsf{Adv}^{\mathcal{HE}}_{\mathsf{Att}} = \Pr[\mathsf{Att} \text{ wins }] - 1/2$.

Definition 2.6. An additively homomorphic encryption scheme $\mathcal{HE}$ is secure if for all PPT adversaries Att, $\mathsf{Adv}^{\mathcal{HE}}_{\mathsf{Att}}$ is negligible in λ.

2.5 Obfuscation

We recall the definition of indistinguishability obfuscation from [GGH+13, SW14].

Definition 2.7. (Indistinguishability Obfuscation) Let $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ be a family of polynomial-size circuits. Let $i\mathcal{O}$ be a uniform PPT algorithm that takes as input the security parameter λ, a circuit $C \in \mathcal{C}_\lambda$ and outputs a circuit C'. $i\mathcal{O}$ is called an indistinguishability obfuscator for a circuit class $\{\mathcal{C}_\lambda\}$ if it satisfies the following conditions:

- (Preserving Functionality) For all security parameters $\lambda \in \mathbb{N}$, for all $C \in \mathcal{C}_\lambda$, for all inputs x, we have that $C'(x) = C(x)$ where $C' \leftarrow i\mathcal{O}(1^\lambda, C)$.
- (Indistinguishability of Obfuscation) For any (not necessarily uniform) PPT distinguisher $\mathcal{B} = (Samp, \mathcal{D})$, there exists a negligible function $\mathrm{negl}(\cdot)$ such that the following holds: if for all security parameters $\lambda \in \mathbb{N}, \Pr[\forall x, C_0(x) = C_1(x) : (C_0; C_1; \sigma) \leftarrow Samp(1^\lambda)] > 1 - \mathrm{negl}(\lambda)$, then

$$\begin{aligned} &|\Pr[\mathcal{D}(\sigma, i\mathcal{O}(1^\lambda, C_0)) = 1 : (C_0; C_1; \sigma) \leftarrow Samp(1^\lambda)] - \\ &\Pr[\mathcal{D}(\sigma, i\mathcal{O}(1^\lambda, C_1)) = 1 : (C_0; C_1; \sigma) \leftarrow Samp(1^\lambda)]| \leq \mathrm{negl}(\lambda). \end{aligned}$$

In a recent work, [GGH+13] showed how indistinguishability obfuscators can be constructed for the circuit class $P/poly$. We remark that $(Samp, \mathcal{D})$ are two algorithms that pass state, which can be viewed equivalently as a single stateful algorithm $\mathcal{B}$. In our proofs we employ the latter approach, although here we state the definition as it appears in prior work.

A stronger notion of obfuscation, *virtual black box obfuscation* was proposed by Barak et al. [BGI+12].

Definition 2.8 (Virtual Black-Box Obfuscator)**.** Let $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ be a family of polynomial-size circuits. Let $\mathcal{O}$ be a PPT algorithm that takes as input the security parameter λ, a circuit $C \in \mathcal{C}_\lambda$ and outputs a circuit C'. $\mathcal{O}$ is called a virtual black-box obfuscator for a circuit class $\{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ if it satisfies the following conditions:

- (Preserving Functionality) For all security parameters $\lambda \in \mathbb{N}$, for all $C \in \mathcal{C}_\lambda$, for all inputs x, we have that $C'(x) = C(x)$ where $C' \leftarrow \mathcal{O}(1^\lambda, C)$.
- (Virtual Black-Box) For every (non-uniform) PPT algorithm $\mathcal{A}$, there exists a PPT simulator S such that, for all $C \in \mathcal{C}_\lambda$,

$$\Pr[\mathcal{A}(\mathcal{O}(1^\lambda, C)) = 1] - \Pr[S^C(1^\lambda, 1^{|C|}) = 1] \leq \mathrm{negl}(\lambda)$$

For simplicity of notation, we will drop the dependence of $i\mathcal{O}$ and $\mathcal{O}$ on 1^λ.

2.6 Puncturable Pseudorandom Functions

The notion of constrained PRFs was introduced in the concurrent works of [BW13, BGI13, KPTZ13]. Punctured PRFs, first termed by [SW14] are a special class of constrained PRFs.

A PRF $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is a puncturable pseudorandom function if there is an additional key space $\mathcal{K}_p$ and three polynomial time algorithms $F.\mathsf{setup}$, $F.\mathsf{eval}$ and $F.\mathsf{puncture}$ as follows:

- $F.\mathsf{setup}(1^\lambda)$ is a randomized algorithm that takes the security parameter λ as input and outputs a description of the key space $\mathcal{K}$, the punctured key space $\mathcal{K}_p$ and the PRF F.
- $F.\mathsf{puncture}(K, x)$ is a randomized algorithm that takes as input a PRF key $K \in \mathcal{K}$ and $x \in \mathcal{X}$, and outputs a key $K_x \in \mathcal{K}_p$.
- $F.\mathsf{eval}(K_x, x')$ is a deterministic algorithm that takes as input a punctured key $K_x \in \mathcal{K}_p$ and $x' \in \mathcal{X}$. Let $K \in \mathcal{K}$, $x \in \mathcal{X}$ and $K_x \leftarrow F.\mathsf{puncture}(K, x)$. For correctness, we need the following property:

$$F.\mathsf{eval}(K_x, x') = \begin{cases} F(K, x') & \text{if } x \neq x' \\ \perp & \text{otherwise} \end{cases}$$

In this work, we will only need selectively secure puncturable PRFs. The selective security game between the challenger and the adversary A consists of the following phases.

Challenge Phase A sends a challenge $x^* \in \mathcal{X}$. The challenger chooses uniformly at random a PRF key $K \leftarrow \mathcal{K}$ and a bit $b \leftarrow \{0, 1\}$. It computes $K\{x^*\} \leftarrow F.\mathsf{puncture}(K, x^*)$. If $b = 0$, the challenger sets $y = F(K, x^*)$, else $y \leftarrow \mathcal{Y}$. It sends $K\{x^*\}, y$ to A.

Guess A outputs a guess b' of b.

A wins if $b = b'$. The advantage of A is defined to be $\mathsf{Adv}^F_A(\lambda) = \Pr[A \text{ wins}]$.

Definition 2.9. The PRF F is a selectively secure puncturable PRF if for all probabilistic polynomial time adversaries $\mathcal{A}$, $\mathsf{Adv}^F_{\mathcal{A}}(\lambda)$ is negligible in λ.

2.7 Universal Parameters

In a recent work, Hofheinz et al. [HJK+14] introduced the notion of universal parameters. A universal parameters scheme $\mathcal{U}$, parameterized by polynomials $\ell_{\text{ckt}}, \ell_{\text{inp}}$ 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 of size at most ℓ_{ckt} bits. The circuit d takes as input ℓ_{inp} bits and outputs ℓ_{out} bits.

In this work, we will be using a universal parameter scheme that is adaptively secure in the random oracle model. In order to define adaptive security for universal parameters, let us first define the notion of an admissible adversary $\mathcal{A}$.

An admissible adversary $\mathcal{A}$ is defined to be an efficient interactive Turing Machine that outputs one bit, with the following input/output behavior:

- $\mathcal{A}$ takes as input security parameter λ and a universal parameter U.
- $\mathcal{A}$ can send a random oracle query (RO, x), and receives the output of the random oracle on input x.
- $\mathcal{A}$ can send a message of the form (params, d) where $d \in \mathcal{C}[\ell_{\text{ckt}}, \ell_{\text{inp}}, \ell_{\text{out}}]$. Upon sending this message, $\mathcal{A}$ is required to honestly compute $p_d = \mathsf{InduceGen}(U, d)$, making use of any additional random oracle queries, and $\mathcal{A}$ appends (d, p_d) to an auxiliary tape.

Let $\mathsf{SimUGen}$ and SimRO be PPT algorithms. Consider the following two experiments:

$\mathsf{Real}^{\mathcal{A}}(1^\lambda)$:

1. The random oracle RO is implemented by assigning random outputs to each unique query made to RO.
2. $U \leftarrow \mathsf{UniversalGen}^{\mathsf{RO}}(1^\lambda)$.
3. $\mathcal{A}(1^\lambda, U)$ is executed, where every message of the form (RO, x) receives the response $\mathsf{RO}(x)$.
4. Upon termination of $\mathcal{A}$, the output of the experiment is the final output of the execution of $\mathcal{A}$.

$\mathsf{Ideal}^{\mathcal{A}}_{\mathsf{SimUGen},\mathsf{SimRO}}(1^\lambda)$:

1. A truly random function F that maps ℓ_{ckt} bits to ℓ_{out} bits is implemented by assigning random ℓ_{out}-bit outputs to each unique query made to F . Throughout this experiment, a Parameters Oracle O is implemented as follows: On input d, where $d \in \mathcal{C}[\ell_{\text{ckt}}, \ell_{\text{inp}}, \ell_{\text{out}}]$, O outputs $d(F(d))$.
2. $(U, \tau) \leftarrow \mathsf{SimUGen}(1^\lambda)$. Here, $\mathsf{SimUGen}$ can make arbitrary queries to the Parameters Oracle O.
3. $\mathcal{A}(1^\lambda, U)$ and $\mathsf{SimRO}(\tau)$ begin simultaneous execution.
   - Whenever $\mathcal{A}$ sends a message of the form (RO, x), this is forwarded to SimRO, which produces a response to be sent back to $\mathcal{A}$.
   - SimRO can make any number of queries to the Parameter Oracle O.
   - Finally, after $\mathcal{A}$ sends any message of the form (params, d), the auxiliary tape of $\mathcal{A}$ is examined until an entry of the form (d, p_d) is added to it. At this point, if p_d is not equal to $d(F(d))$, then experiment aborts, resulting in an *Honest Parameter Violation*.
4. Upon termination of $\mathcal{A}$, the output of the experiment is the final output of the execution of $\mathcal{A}$.

Definition 2.10. A universal parameters scheme $\mathcal{U} = (\mathsf{UniversalGen}, \mathsf{InduceGen})$, parameterized by polynomials $\ell_{\text{ckt}}, \ell_{\text{inp}}$ and ℓ_{out}, is said to be adaptively secure in the random oracle model if there exist PPT algorithms $\mathsf{SimUGen}$ and SimRO such that for all PPT adversaries $\mathcal{A}$, the following hold:

$$\Pr[\mathsf{Ideal}^{\mathcal{A}}_{\mathsf{SimUGen},\mathsf{SimRO}}(1^\lambda) \text{ aborts }] = 0^{10}$$

and

$$|\Pr[\mathsf{Real}^{\mathcal{A}}(1^\lambda) = 1] - \Pr[\mathsf{Ideal}^{\mathcal{A}}_{\mathsf{SimUGen},\mathsf{SimRO}}(1^\lambda) = 1]| \leq \text{negl}(\lambda)$$

[10]The definition in [HJK+14] only requires this probability to be negligible in λ. However, the construction actually achieves zero probability of Honest Parameter Violation. Hence, for the simplicity of our proof, we will use this definition.

Hofheinz et al. [HJK+14] construct a universal parameters scheme that is adaptively secure in the random oracle model, assuming a secure indistinguishability obfuscator, a selectively secure puncturable PRF and an injective one way function.

2.8 RSA Assumption

Assumption 1 (RSA [RSA78])**.** Let λ be the security parameter. Let $N = pq$ be the RSA modulus, where p, q are randomly chosen, distinct, λ-bit primes. Let e be a randomly chosen positive integer less than and relatively prime to $\phi(N) = (p-1)(q-1)$ and $y \leftarrow \mathbb{Z}_N$. For any PPT algorithm $\mathcal{A}$, $\Pr[x \leftarrow \mathcal{A}(N, e, y)\textbf{and } x^e = y] \leq \text{negl}(\lambda)$.

3 Universal Signature Aggregators

In this section, we define the notion of universal signature aggregators. Let ℓ_{ver}, ℓ_{vk}, ℓ_{msg}, ℓ_{sig} be polynomials. Given any security parameter λ, $\ell_{\text{ver}}(\lambda)$ represents a bound on the size of verification circuits, $\ell_{\text{vk}}(\lambda)$ represents a bound on the size of verification key, $\ell_{\text{msg}}(\lambda)$ is a bound on the length of messages signed and $\ell_{\text{sig}}(\lambda)$ is a bound on the size of signatures. For simplicity of notation, we will drop the dependence on λ when the context is clear.

A universal signature aggregator $(\ell_{\text{ver}}, \ell_{\text{vk}}, \ell_{\text{msg}}, \ell_{\text{sig}})$-$\mathsf{UniversalSigAgg}$ consists of three algorithms $\mathsf{UniversalSetup}$, $\mathsf{UniversalAgg}$ and $\mathsf{UniversalVerify}$ defined as follows.

- $\mathsf{UniversalSetup}(1^\lambda)$ is a randomized algorithm that takes as input security parameter λ and outputs public parameters PP.
- $\mathsf{UniversalAgg}(\text{PP}, \{(\mathsf{Verify}_i, \text{VK}_i, m_i, \sigma_i)\}_{i=1}^t)$ is a deterministic algorithm that takes as input security parameter λ, public parameters PP and t tuples $(\mathsf{Verify}_i, \text{VK}_i, m_i, \sigma_i)$ (for some arbitrary t) where each tuple is $(\ell_{\text{ver}}, \ell_{\text{vk}}, \ell_{\text{msg}}, \ell_{\text{sig}})$-length qualified. It outputs an aggregate signature σ_{agg} whose length is polynomial in λ, but independent of t.
- $\mathsf{UniversalVerify}(\text{PP}, \{(\mathsf{Verify}_i, \text{VK}_i, m_i)\}_{i=1}^t, \sigma_{\text{agg}})$ is a deterministic algorithm that takes as input security parameter λ, public parameters PP, t tuples $(\mathsf{Verify}_i, \text{VK}_i, m_i)$ that are $(\ell_{\text{ver}}, \ell_{\text{vk}}, \ell_{\text{msg}})$-length qualified, and an aggregated signature σ_{agg}. It outputs either 0 or 1.

For our constructions, we will assume that all verification circuits have ℓ_{ver} bit representation, all verification keys have length ℓ_{vk}, all messages signed have length ℓ_{msg} and the corresponding signatures have length ℓ_{sig}.

Correctness Let $\{(\mathsf{Verify}_i, \text{VK}_i, m_i, \sigma_i)\}_{i=1}^t$ be any t distinct tuples that are $(\ell_{\text{ver}}, \ell_{\text{vk}}, \ell_{\text{msg}}, \ell_{\text{sig}})$-length qualified and for all $i \leq t$, $\mathsf{Verify}_i(\text{VK}_i, m_i, \sigma_i) = 1$. For all $\lambda \in \mathbb{N}$, $\text{PP} \leftarrow \mathsf{UniversalSetup}(1^\lambda)$ and $\sigma_{\text{agg}} \leftarrow \mathsf{UniversalAgg}(1^\lambda, \text{PP}, \{(\mathsf{Verify}_i, \text{VK}_i, m_i, \sigma_i)\}_i)$, we require that $\mathsf{UniversalVerify}(\text{PP}, \{(\mathsf{Verify}_i, \text{VK}_i, m_i)\}_i, \sigma_{\text{agg}}) = 1$.

3.1 Security of Universal Signature Aggregators

We now proceed to the formal security definition for universal signature aggregators.

Let $\mathcal{S} = (\mathcal{S}.\mathsf{Gen}, \mathcal{S}.\mathsf{Sign}, \mathcal{S}.\mathsf{Verify})$ be a secure $(\ell_{\text{ver}}, \ell_{\text{vk}}, \ell_{\text{msg}}, \ell_{\text{sig}})$-length qualified signature scheme. Consider the following security game between an adversary $\mathcal{A}$ and the challenger.

$\mathsf{Exp}_{\mathcal{A},\mathcal{S}}(\lambda)$:

- **Setup Phase** Challenger chooses $(\text{SK}, \text{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$, computes $\text{PP} \leftarrow \mathsf{UniversalSetup}(1^\lambda)$ and sends PP, VK to $\mathcal{A}$.

- **Signing Phase** $\mathcal{A}$ sends signing query x_i, and receives $\sigma_i \leftarrow \mathcal{S}.\mathsf{Sign}(\mathrm{SK}, x_i)$.
- **Forgery** $\mathcal{A}$ finally outputs t tuples $(\mathsf{Verify}_i, \mathrm{VK}_i, m_i)$ and an aggregated forgery σ_{agg}.

$\mathcal{A}$ wins if there exists $i^* \in [t]$ such that $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}$, $\mathrm{VK}_{i^*} = \mathrm{VK}$, message m_{i^*} was not queried during the signing phase and $\mathsf{UniversalVerify}(\mathrm{PP}, \{(\mathsf{Verify}_i, \mathrm{VK}_i, m_i)\}, \sigma_{\mathrm{agg}}) = 1$. Let $\mathsf{Adv}_{\mathcal{A},\mathcal{S}}(\lambda) = \Pr[\mathcal{A} \text{ wins } \mathsf{Exp}_{\mathcal{A},\mathcal{S}}(\lambda)]$.

Definition 3.1. Let $\mathcal{S}$ be a $(\ell_{\mathrm{ver}}, \ell_{\mathrm{vk}}, \ell_{\mathrm{msg}}, \ell_{\mathrm{sig}})$- length qualified secure signature scheme. A universal signature aggregator scheme $(\ell_{\mathrm{ver}}, \ell_{\mathrm{vk}}, \ell_{\mathrm{msg}}, \ell_{\mathrm{sig}})$-$\mathsf{UniversalSigAgg}$ is secure with respect to scheme $\mathcal{S}$ if for all PPT adversaries $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A},\mathcal{S}}(\lambda)$ is negligible in λ.

We can also define a weaker *selective* notion where the adversary $\mathcal{A}$ chooses the message m corresponding to $(\mathcal{S}.\mathsf{Verify}, \mathrm{VK})$ before receiving the public parameters PP. More formally, the selective experiment $\mathsf{Exp}^{\mathrm{sel}}_{\mathcal{A},\mathcal{S}}(\lambda)$ is defined as follows.

$\mathsf{Exp}^{\mathrm{sel}}_{\mathcal{A},\mathcal{S}}(\lambda)$:

- $\mathcal{A}$ sends a message m to the challenger.
- **Setup Phase** Challenger computes $(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$, $\mathrm{PP} \leftarrow \mathsf{UniversalSetup}(1^\lambda)$ and sends PP, VK to $\mathcal{A}$.
- **Signing Phase** $\mathcal{A}$ sends signing query $x_i \neq m$, and receives $\sigma_i \leftarrow \mathcal{S}.\mathsf{Sign}(\mathrm{SK}, x_i)$.
- **Forgery** $\mathcal{A}$ finally outputs t tuples $(\mathsf{Verify}_i, \mathrm{VK}_i, m_i)$ and an aggregated forgery σ_{agg}.

$\mathcal{A}$ wins if there exists an $i^* \in [t]$ such that $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}$, $\mathrm{VK}_{i^*} = \mathrm{VK}$, $m_{i^*} = m$ and $\mathsf{UniversalVerify}(\mathrm{PP}, \{(\mathsf{Verify}_i, \mathrm{VK}_i, m_i)\}, \sigma_{\mathrm{agg}}) = 1$. Let $\mathsf{Adv}^{\mathrm{sel}}_{\mathcal{A},\mathcal{S}}(\lambda) = \Pr[\mathcal{A} \text{ wins } \mathsf{Exp}^{\mathrm{sel}}_{\mathcal{A},\mathcal{S}}(\lambda)]$.

Definition 3.2. Let $\mathcal{S}$ be a $(\ell_{\mathrm{ver}}, \ell_{\mathrm{vk}}, \ell_{\mathrm{msg}}, \ell_{\mathrm{sig}})$- length qualified secure signature scheme. A universal signature aggregator scheme $(\ell_{\mathrm{ver}}, \ell_{\mathrm{vk}}, \ell_{\mathrm{msg}}, \ell_{\mathrm{sig}})$-$\mathsf{UniversalSigAgg}$ is *selectively* secure with respect to scheme $\mathcal{S}$ if for all PPT adversaries $\mathcal{A}$, $\mathsf{Adv}^{\mathrm{sel}}_{\mathcal{A},\mathcal{S}}(\lambda)$ is negligible in λ.

In certain situations, it may be possible that the number of signatures to be aggregated is known in advance. In such a scenario, we can use bounded universal signature aggregators (defined below).

Definition 3.3. An n-bounded universal signature aggregator scheme $(\ell_{\mathrm{ver}}, \ell_{\mathrm{vk}}, \ell_{\mathrm{msg}}, \ell_{\mathrm{sig}})$-$\mathsf{UniversalSigAgg}$ = ($\mathsf{UniversalSetup}$, $\mathsf{UniversalAgg}$, $\mathsf{UniversalVerify}$) is a universal signature aggregator in which $\mathsf{UniversalSetup}$ takes an additional input 1^n. The public parameters output by $\mathsf{UniversalSetup}$ 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.

4 Universally Aggregating Unique Signatures

We will now describe our universal signature aggregator $(\ell_{\mathrm{ver}}, \ell_{\mathrm{vk}}, \ell_{\mathrm{msg}}, \ell_{\mathrm{sig}})$-$\mathsf{UniversalSigAgg}$. Let $i\mathcal{O}$ be a secure indistinguishability obfuscation scheme, F a puncturable PRF with key space $\mathcal{K}$, punctured key space $\mathcal{K}_p$, domain $\mathcal{X} = \{0,1\}^{\ell_{\mathrm{ver}}} \times \{0,1\}^{\ell_{\mathrm{vk}}} \times \{0,1\}^{\ell_{\mathrm{msg}}}$ and range $\mathcal{Y} = \mathbb{Z}_N^*$ for some randomly chosen RSA modulus N, and algorithms $F.\mathsf{setup}$, $F.\mathsf{puncture}$, $F.\mathsf{eval}$. Our scheme consists of the three algorithms $\mathsf{UniversalSetup}$, $\mathsf{UniversalAgg}$ and $\mathsf{UniversalVerify}$.

$\mathsf{UniversalSetup}(1^\lambda)$ $\mathsf{UniversalSetup}$ first chooses an RSA modulus N and $e \leftarrow \mathbb{Z}^*_{\phi(N)}$. Next, it chooses a PRF key $K \leftarrow F.\mathsf{setup}(1^\lambda)$ and computes obfuscations of the programs $\mathsf{Transform}_{N,K}$[11] and $\mathsf{Transform\text{-}Image}_{N,K,e}$[12] defined below. It sets the public parameters to be $\mathrm{PP} = (i\mathcal{O}(\mathsf{Transform}_{N,K}), i\mathcal{O}(\mathsf{Transform\text{-}Image}_{N,K,e}), N, e)$.

$\mathsf{Transform}_{N,K}$:

Inputs: $\mathsf{Verify}' \in \{0,1\}^{\ell_{\mathrm{ver}}}$, $\mathrm{VK}' \in \{0,1\}^{\ell_{\mathrm{vk}}}$, $m' \in \{0,1\}^{\ell_{\mathrm{msg}}}$, $\sigma' \in \{0,1\}^{\ell_{\mathrm{sig}}}$.
Constants : RSA modulus $N \in \mathbb{N}$, $K \in \mathcal{K}$.

if $\mathsf{Verify}'(\mathrm{VK}', m', \sigma') = 0$ **then**
  Output $\perp$.
else
  Output $F(K, \mathsf{Verify}'||\mathrm{VK}'||m')$.
end if

$\mathsf{Transform\text{-}Image}_{N,K,e}$:

Inputs: $\mathsf{Verify}' \in \{0,1\}^{\ell_{\mathrm{ver}}}$, $\mathrm{VK}' \in \{0,1\}^{\ell_{\mathrm{vk}}}$, $m' \in \{0,1\}^{\ell_{\mathrm{msg}}}$.
Constants : RSA modulus $N \in \mathbb{N}$, $K \in \mathcal{K}$, $e \in \mathbb{Z}_{\phi(N)}$.

Let $w = F(K, \mathsf{Verify}'||\mathrm{VK}'||m')$. Output $w^e \pmod N$.

$\mathsf{UniversalAgg}(\mathrm{PP}, \{(\mathsf{Verify}_i, \mathrm{VK}_i, m_i, \sigma_i)\}_{i=1}^n)$**:** Let $\mathrm{PP} = (P_1, P_2, N, e)$. $\mathsf{UniversalAgg}$ first checks if the n tuples are distinct. If not, it outputs $\perp$. Else, it computes $t_i = P_1(\mathsf{Verify}_i, \mathrm{VK}_i, m_i, \sigma_i)$ for each $i \le n$. If $t_i = \perp$ for some i, then $\mathsf{UniversalAgg}$ outputs $\perp$, else it outputs $\sigma_{\mathrm{agg}} = \prod_i t_i \pmod N$.

$\mathsf{UniversalVerify}(\mathrm{PP}, \{(\mathsf{Verify}_i, \mathrm{VK}_i, m_i)\}_{i=1}^n, \sigma_{\mathrm{agg}})$**:** Let $\mathrm{PP} = (P_1, P_2, N, e)$. $\mathsf{UniversalVerify}$ first checks if all n tuples are distinct. If not, it outputs 0. Else, it computes, for all $i \le n$, $s_i = \mathsf{Transform\text{-}Image}(\mathsf{Verify}_i, \mathrm{VK}_i, m_i)$. If $(\prod_i s_i) = \sigma^e_{\mathrm{agg}} \pmod N$, it outputs 1, else it outputs 0.

Correctness: Let $\{(\mathsf{Verify}_i, \mathrm{VK}_i, m_i, \sigma_i)\}_{i=1}^n$ be n tuples such that they are all distinct and $\mathsf{Verify}_i(\mathrm{VK}_i, m_i, \sigma_i) = 1$ for all $i \le n$. Fix any $\lambda \in \mathbb{N}$, $\mathrm{PP} \leftarrow \mathsf{UniversalSetup}(1^\lambda)$, $(\sigma_{\mathrm{agg}}) \leftarrow \mathsf{UniversalAgg}(\mathrm{PP}, \{(\mathsf{Verify}_i, \mathrm{VK}_i, m_i, \sigma_i)\})$. Then,

$$\begin{aligned}
\sigma^e_{\mathrm{agg}} &= (\prod \mathsf{Transform}(\mathsf{Verify}_i, \mathrm{VK}_i, m_i, \sigma_i))^e \pmod N \\
&= (\prod F(K, \mathsf{Verify}_i||\mathrm{VK}_i||m_i))^e \pmod N \\
&= (\prod F(K, \mathsf{Verify}_i||\mathrm{VK}_i||m_i)^e) \pmod N \\
&= (\prod \mathsf{Transform\text{-}Image}_{N,K,e}(\mathsf{Verify}_i, \mathrm{VK}_i, m_i)) \pmod N
\end{aligned}$$

Also, note that the size of the aggregated signature ($\sigma_{\mathrm{agg}} \in \mathbb{Z}^*_N$) depends only on the security parameter λ, but not on the number of signatures aggregated.

4.1 Proof of Security

In this subsection, we will show that our construction from Section 4 is selectively secure with respect to secure unique signature schemes.

[11] Padded to be of the same size as $\mathsf{TransformAlt}$ and $\mathsf{Transform\text{-}Reject}$.
[12] Padded to be of the same size as $\mathsf{Transform\text{-}Image\text{-}1}$.

Theorem 4.1. Assuming $i\mathcal{O}$ is a secure indistinguishability obfuscator, F is a selectively secure puncturable PRF and RSA is secure, for all $(\ell_{\text{ver}}, \ell_{\text{vk}}, \ell_{\text{msg}}, \ell_{\text{sig}})$-length qualified secure unique signature schemes $\mathcal{S}$, the universal signature aggregator $(\ell_{\text{ver}}, \ell_{\text{vk}}, \ell_{\text{msg}}, \ell_{\text{sig}})$-$\mathsf{UniversalSigAgg}$ is selectively secure with respect to $\mathcal{S}$.

Let $\mathcal{S} = (\mathcal{S}.\mathsf{Gen}, \mathcal{S}.\mathsf{Sign}, \mathcal{S}.\mathsf{Verify})$ be a secure $(\ell_{\text{ver}}, \ell_{\text{vk}}, \ell_{\text{msg}}, \ell_{\text{sig}})$-length qualified unique signature scheme, and Att a PPT adversary. In order to prove this theorem, we will define a sequence of experiments Game 0, ..., Game 3, where Game 0 $= \mathsf{Exp}^{\text{sel}}_{\mathsf{Att},\mathcal{S}}$.

4.1.1 Sequence of Games

Game 0**:** This game corresponds to $\mathsf{Exp}^{\text{sel}}_{\mathsf{Att},\mathcal{S}}$. The adversary Att first selectively sends message m, and then receives the verification key and public parameters for the aggregator. Next, the adversary makes signing queries, and finally submits the forgery.

1. Att sends message m.
2. Compute $(\text{SK}, \text{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$. Choose an RSA modulus N, $e \leftarrow \mathbb{Z}^*_{\phi(N)}$, $K \leftarrow F.\mathsf{setup}(1^\lambda)$ and set $\text{PP} = (i\mathcal{O}(\mathsf{Transform}_{N,K}), i\mathcal{O}(\mathsf{Transform\text{-}Image}_{N,K,e}), N, e)$. Send PP, VK to Att.
3. For each signing query $x_i \neq m$, compute $\sigma_i \leftarrow \mathcal{S}.\mathsf{Sign}(\text{SK}, x_i)$ and send σ_i to Att.
4. Att sends forgery σ_{agg} and n tuples $\{(\mathsf{Verify}_i, \text{VK}_i, m_i)\}$. Att wins if $\exists i^* \in [n]$ such that $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}$, $\text{VK}_{i^*} = \text{VK}$ and $m_{i^*} = m$ and $\mathsf{UniversalVerify}(\text{PP}, \{(\mathsf{Verify}_i, \text{VK}_i, m_i)\}, \sigma_{\text{agg}}) = 1$.

Game 1**:** This game is exactly similar to the previous one, except that the program $\mathsf{Transform}$ is replaced by $\mathsf{Transform\text{-}Reject}$[13] which outputs $\perp$ if the input tuples is $(\mathcal{S}.\mathsf{Verify}, \text{VK}, m, \sigma)$ even if $\mathcal{S}.\mathsf{Verify}(\text{VK}, m, \sigma) = 1$. Also, it uses a PRF key punctured at $y = \mathcal{S}.\mathsf{Verify}||\text{VK}||m$.

1. Att sends message m.
2. Compute $(\text{SK}, \text{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$. Choose an RSA modulus N, $e \leftarrow \mathbb{Z}^*_{\phi(N)}$, $K \leftarrow F.\mathsf{setup}(1^\lambda)$. Set $y = \mathcal{S}.\mathsf{Verify}||\text{VK}||m$, compute $K\{y\} \leftarrow F.\mathsf{puncture}(K, y)$ and $\text{PP} = (i\mathcal{O}(\mathsf{Transform\text{-}Reject}_{y,N,K\{y\}}), i\mathcal{O}(\mathsf{Transform\text{-}Image}_{N,K,e}), N, e)$. Send PP, VK to Att.
3. For each signing query $x_i \neq m$, compute $\sigma_i \leftarrow \mathcal{S}.\mathsf{Sign}(\text{SK}, x_i)$ and send σ_i to Att.
4. Att sends forgery σ_{agg} and n tuples $\{(\mathsf{Verify}_i, \text{VK}_i, m_i)\}$. Att wins if $\exists i^* \in [n]$ such that $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}$, $\text{VK}_{i^*} = \text{VK}$ and $m_{i^*} = m$ and $\mathsf{UniversalVerify}(\text{PP}, \{(\mathsf{Verify}_i, \text{VK}_i, m_i)\}, \sigma_{\text{agg}}) = 1$.

$\mathsf{Transform\text{-}Reject}_{y,N,K\{y\}}$:

Inputs: $\mathsf{Verify}' \in \{0,1\}^{\ell_{\text{ver}}}$, $\text{VK}' \in \{0,1\}^{\ell_{\text{vk}}}$, $m' \in \{0,1\}^{\ell_{\text{msg}}}$, $\sigma' \in \{0,1\}^{\ell_{\text{sig}}}$.
Constants : $y \in \{0,1\}^{\ell_{\text{ver}}} \times \{0,1\}^{\ell_{\text{vk}}} \times \{0,1\}^{\ell_{\text{msg}}}$, RSA modulus $N \in \mathbb{N}$, $K\{y\} \in \mathcal{K}_p$.

if $\mathsf{Verify}'(\text{VK}', m', \sigma') = 0$ **then**
  Output $\perp$.
else if $\mathsf{Verify}'||\text{VK}'||m' = y$ **then**
  Output $\perp$.
else
  Output $F.\mathsf{eval}(K\{y\}, \mathsf{Verify}'||\text{VK}'||m')$.
end if

Game 2**:** This game is similar to the previous one, except that the program $\mathsf{Transform\text{-}Image}$ is replaced by $\mathsf{Transform\text{-}Image\text{-}1}$[14]. It uses a PRF key punctured at $y = \mathcal{S}.\mathsf{Verify}||\text{VK}||m$. For input y, it outputs a hardwired constant z. In this game, z is set to be $F(K, y)^e$.

[13] Padded appropriately to be of the same size as $\mathsf{Transform}$ and $\mathsf{TransformAlt}$.
[14] Padded appropriately to be of the same size as $\mathsf{Transform\text{-}Image}$.

1. Att sends message m.
2. Compute $(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$. Choose an RSA modulus N, $e \leftarrow \mathbb{Z}^*_{\phi(N)}$ and $K \leftarrow F.\mathsf{setup}(1^\lambda)$. Set $y = \mathcal{S}.\mathsf{Verify}||\mathrm{VK}||m$. Compute $K\{y\} \leftarrow F.\mathsf{puncture}(K, y)$, $w = F(K, y)$ and $z = w^e \pmod N$. Set $\mathrm{PP} = (i\mathcal{O}(\mathsf{Transform\text{-}Reject}_{y,N,K\{y\}}), i\mathcal{O}(\mathsf{Transform\text{-}Image\text{-}1}_{y,N,K\{y\},z,e}), N, e)$ and send PP, VK to Att.
3. For each signing query $x_i \neq m$, compute $\sigma_i \leftarrow \mathcal{S}.\mathsf{Sign}(\mathrm{SK}, x_i)$ and send σ_i to Att.
4. Att sends forgery σ_{agg} and n tuples $\{(\mathsf{Verify}_i, \mathrm{VK}_i, m_i)\}$. Att wins if $\exists i^* \in [n]$ such that $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}$, $\mathrm{VK}_{i^*} = \mathrm{VK}$ and $m_{i^*} = m$ and $\mathsf{UniversalVerify}(\mathrm{PP}, \{(\mathsf{Verify}_i, \mathrm{VK}_i, m_i)\}, \sigma_{\mathrm{agg}}) = 1$.

$\mathsf{Transform\text{-}Image\text{-}1}_{y,N,K\{y\},z,e}$:

Inputs: $\mathsf{Verify}' \in \{0,1\}^{\ell_{\mathrm{ver}}}, \mathrm{VK}' \in \{0,1\}^{\ell_{\mathrm{vk}}}, m' \in \{0,1\}^{\ell_{\mathrm{msg}}}$.
Constants: $y \in \{0,1\}^{\ell_{\mathrm{ver}}} \times \{0,1\}^{\ell_{\mathrm{vk}}} \times \{0,1\}^{\ell_{\mathrm{msg}}}$, RSA modulus $N \in \mathbb{N}$, $K\{y\} \in \mathcal{K}_p$, $z \in \mathbb{Z}^*_N$, $e \in \mathbb{Z}^*_{\phi(N)}$.

if $\mathsf{Verify}_i||\mathrm{VK}_i||m_i = y$ **then**
  Output z.
else
  Let $w = F.\mathsf{eval}(K\{y\}, \mathsf{Verify}'||\mathrm{VK}'||m')$.
  Output w^e.
end if

Game 3: In this game, the challenger chooses z at random.

1. Att sends message m.
2. Compute $(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$. Choose an RSA modulus N, $e \leftarrow \mathbb{Z}^*_{\phi(N)}$ and $K \leftarrow F.\mathsf{setup}(1^\lambda)$. Set $y = \mathcal{S}.\mathsf{Verify}||\mathrm{VK}||m$. Compute $K\{y\} \leftarrow F.\mathsf{puncture}(K, y)$ and $z \leftarrow \mathbb{Z}^*_N$. Set $\mathrm{PP} = (i\mathcal{O}(\mathsf{Transform\text{-}Reject}_{y,N,K\{y\}}), i\mathcal{O}(\mathsf{Transform\text{-}Image\text{-}1}_{y,N,K\{y\},z,e}), N, e)$ and send PP, VK to Att.
3. For each signing query $x_i \neq m$, compute $\sigma_i \leftarrow \mathcal{S}.\mathsf{Sign}(\mathrm{SK}, x_i)$ and send σ_i to Att.
4. Att sends forgery σ_{agg} and n tuples $\{(\mathsf{Verify}_i, \mathrm{VK}_i, m_i)\}$. Att wins if all the n tuples are distinct and $\exists i^* \in [n]$ such that $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}$, $\mathrm{VK}_{i^*} = \mathrm{VK}$ and $m_{i^*} = m$ and $\mathsf{UniversalVerify}(\mathrm{PP}, \{(\mathsf{Verify}_i, \mathrm{VK}_i, m_i)\}, \sigma_{\mathrm{agg}}) = 1$.

4.1.2 Analysis

Let $\mathsf{Adv}^j_{\mathsf{Att}}$ denote the advantage of adversary Att in Game j.

Lemma 4.1. Assuming $i\mathcal{O}$ is a secure indistinguishability obfuscator and $\mathcal{S}$ is a secure $(\ell_{\mathrm{ver}}, \ell_{\mathrm{vk}}, \ell_{\mathrm{msg}}, \ell_{\mathrm{sig}})$-length qualified unique signature scheme, for any PPT adversary Att,

$$\mathsf{Adv}^0_{\mathsf{Att}} - \mathsf{Adv}^1_{\mathsf{Att}} \leq \mathrm{negl}(\lambda).$$

Proof. Suppose, on the contrary, there exists a PPT adversary Att such that $\mathsf{Adv}^0_{\mathsf{Att}} - \mathsf{Adv}^1_{\mathsf{Att}} = \epsilon$, where ϵ is non-negligible in λ. Assuming $\mathcal{O}$ is a secure indistinguishability obfuscator, we will use Att to construct a PPT algorithm $\mathcal{B}$ that breaks the security of $\mathcal{S}$. Here, we will crucially use the fact that $\mathcal{S}$ is a unique signature scheme; that is, there is a unique accepting signature $\sigma \in \{0,1\}^{\ell_{\mathrm{sig}}}$ corresponding to verification key VK and message m.

First, let us consider the following altered circuit $\mathsf{TransformAlt}_{j,b,y,N,K}$[15] which takes as input a tuple $(\mathsf{Verify}', \mathrm{VK}', m', \sigma')$ and outputs $\perp$ if $\mathsf{Verify}' = \mathcal{S}.\mathsf{Verify}$, $\mathrm{VK}' = \mathrm{VK}$ and the j^{th} bit of σ' is b.

[15] Padded appropriately to be of the same size as $\mathsf{Transform}$ and $\mathsf{Transform\text{-}Reject}$.

$\mathsf{TransformAlt}_{j,b,y,N,K}$:

Inputs: $\mathsf{Verify}' \in \{0,1\}^{\ell_{\mathrm{ver}}}$, $\mathrm{VK}' \in \{0,1\}^{\ell_{\mathrm{vk}}}$, $m' \in \{0,1\}^{\ell_{\mathrm{msg}}}$, $\sigma' \in \{0,1\}^{\ell_{\mathrm{sig}}}$.
Constants : $j \in [\ell_{\mathrm{sig}}]$, $b \in \{0,1\}$, $y \in \{0,1\}^{\ell_{\mathrm{ver}}} \times \{0,1\}^{\ell_{\mathrm{vk}}} \times \{0,1\}^{\ell_{\mathrm{msg}}}$, RSA modulus $N \in \mathbb{N}, K \in \mathcal{K}$.

if $\mathsf{Verify}'(\mathrm{VK}', m', \sigma') = 0$ **then**
  Output $\perp$.
else if $\mathsf{Verify}'||\mathrm{VK}'||m' = y$ **and** $\sigma'[j] = b$ **then**
  Output $\perp$.
else
  Output $F(K, \mathsf{Verify}'||\mathrm{VK}'||m')$.
end if

We will now state two observations which will be useful for proving our claim. Fix a message $m \in \{0,1\}^{\ell_{\mathrm{msg}}}$. Let $(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathcal{S}.\mathsf{Setup}(1^\lambda)$, $\sigma \leftarrow \mathcal{S}.\mathsf{Sign}(\mathrm{SK}, m)$ and $y = \mathcal{S}.\mathsf{Verify}||\mathrm{VK}||m$. Let $\sigma[j]$ denote the j^{th} bit of σ.

Observation 4.1. For all $j \in [\ell_{\mathrm{sig}}]$, the circuits $\mathsf{Transform}_{N,K}$ and $\mathsf{TransformAlt}_{j,1-\sigma[j],y,N,K}$ are functionally identical.

Observation 4.2. For all $j \in [\ell_{\mathrm{sig}}]$, the circuits $\mathsf{Transform\text{-}Reject}_{y,N,K\{y\}}$ and $\mathsf{TransformAlt}_{j,\sigma[j],y,N,K}$ are functionally identical.

Both these observations follow from the fact that $\mathcal{S}$ is a unique signature scheme and the correctness of punctured key on non-punctured inputs.

Next, we define $\mathsf{Game\text{-}Alt}\ j, b$, which is exactly similar to $\mathsf{Game}\ 0$ and $\mathsf{Game}\ 1$, except that the challenger outputs $\mathcal{O}(1^\lambda, \mathsf{TransformAlt}_{j,b,y,N,K})$ (instead of $\mathcal{O}(1^\lambda, \mathsf{Transform}_{N,K})$ or $\mathcal{O}(1^\lambda, \mathsf{Transform\text{-}Reject}_{y,N,K\{y\}})$) as part of the public parameters PP. Let $\mathcal{E}_j^\sigma$ be the event that $\sigma[j] = 1$, where σ is the unique signature corresponding to challenge message m output by Att.

From these observations, it follows that $\mathcal{O}(1^\lambda, \mathsf{Transform}_{N,K})$ and $\mathcal{O}(1^\lambda, \mathsf{TransformAlt}_{j,1-\sigma[j],y,N,K})$ are computationally indistinguishable (by the security of $\mathcal{O}$) and similarly, $\mathcal{O}(1^\lambda, \mathsf{TransformAlt}_{j,\sigma[j],y,N,K})$ and $\mathcal{O}(1^\lambda, \mathsf{Transform\text{-}Reject}_{y,N,K\{y\}})$ are computationally indistinguishable. Hence, for all $j \leq \ell_{\mathrm{sig}}$, we get the following equations:

$$\left|\Pr[(\mathsf{Att}\text{ wins in }\mathsf{Game}\ 0)\,|\mathcal{E}_j^\sigma] - \Pr[(\mathsf{Att}\text{ wins in }\mathsf{Game\text{-}Alt}\ j, 0)\,|\mathcal{E}_j^\sigma]\right| \leq \mathrm{negl}(\lambda), \tag{1}$$

$$\left|\Pr[(\mathsf{Att}\text{ wins in }\mathsf{Game}\ 0)\,|\neg\mathcal{E}_j^\sigma] - \Pr[(\mathsf{Att}\text{ wins in }\mathsf{Game\text{-}Alt}\ j, 1)\,|\neg\mathcal{E}_j^\sigma]\right| \leq \mathrm{negl}(\lambda), \tag{2}$$

$$\left|\Pr[(\mathsf{Att}\text{ wins in }\mathsf{Game}\ 1)\,|\mathcal{E}_j^\sigma] - \Pr[(\mathsf{Att}\text{ wins in }\mathsf{Game\text{-}Alt}\ j, 1)\,|\mathcal{E}_j^\sigma]\right| \leq \mathrm{negl}(\lambda) \tag{3}$$

$$\left|\Pr[(\mathsf{Att}\text{ wins in }\mathsf{Game}\ 1)\,|\neg\mathcal{E}_j^\sigma] - \Pr[(\mathsf{Att}\text{ wins in }\mathsf{Game\text{-}Alt}\ j, 0)\,|\neg\mathcal{E}_j^\sigma]\right| \leq \mathrm{negl}(\lambda) \tag{4}$$

Continuing with our proof, let us assume $\mathsf{Att} = (\mathsf{Att}_1, \mathsf{Att}_2)$. Att_1 is a randomized algorithm that on input 1^λ, outputs message $m \in \{0,1\}^{\ell_{\mathrm{msg}}}$ which it sends to the challenger, and state st which is sent to Att_2. Att_2 is a randomized algorithm that receives state m, st from Att_1 and inputs PP, VK from challenger. It makes signature queries before outputting the forgery. We will now describe algorithm $\mathcal{B}$ that interacts with a unique signature $\mathcal{S}$ challenger. Let $\tau = \frac{32\lambda}{\epsilon^2}$.

1. $\mathcal{B}$ first runs $\mathsf{Att}_1(1^\lambda)$ and receives message m and state st. It sends m to $\mathcal{S}$ challenger, and receives VK.
2. For $j = 1$ to ℓ_{sig}, do

(a) Set $\mathsf{count}_{j,0} = 0$. For $i = 1$ to τ,

i. Choose RSA modulus N, $e \leftarrow \mathbb{Z}^*_{\phi(N)}$ and $K \leftarrow F.\mathsf{setup}(1^\lambda)$. Set $y = \mathcal{S}.\mathsf{Verify}||\mathrm{VK}||m$, $\mathrm{PP} = (\mathcal{O}(1^\lambda, \mathsf{TransformAlt}_{j,0,y,N,K}), \mathcal{O}(1^\lambda, \mathsf{Transform\text{-}Image}_{N,K,e}), N, e)$ and send $\mathrm{PP}, \mathrm{VK}, m, \mathsf{st}$ to Att_2 as input. Att_2 uses fresh randomness for each run.

ii. For each signing query x_r, $\mathcal{B}$ forwards x_r to the challenger, receives σ_r, which it sends to Att_2.

iii. Finally, Att_2 outputs σ_{agg} and n tuples. If Att wins, $\mathcal{B}$ increments $\mathsf{count}_{j,0}$.

(b) Set $\mathsf{count}_{j,1} = 0$. For $i = 1$ to τ

i. Choose RSA modulus N, $e \leftarrow \mathbb{Z}^*_{\phi(N)}$ and $K \leftarrow F.\mathsf{setup}(1^\lambda)$. Set $y = \mathcal{S}.\mathsf{Verify}||\mathrm{VK}||m$, $\mathrm{PP} = (\mathcal{O}(1^\lambda, \mathsf{TransformAlt}_{j,1,y,N,K}), \mathcal{O}(1^\lambda, \mathsf{Transform\text{-}Image}_{N,K,e}), N, e)$ and send $\mathrm{PP}, \mathrm{VK}, m, \mathsf{st}$ to Att_2 as input. Att_2 uses fresh randomness for each run.

ii. For each signing query x_r, $\mathcal{B}$ forwards x_r to the challenger, receives σ_r, which it sends to Att_2.

iii. Finally, Att_2 outputs σ_{agg} and n tuples. If Att wins, $\mathcal{B}$ increments $\mathsf{count}_{j,1}$.

(c) If $\mathsf{count}_{j,0} > \mathsf{count}_{j,1}$, $\mathcal{B}$ sets $\alpha_j = 1$, else it sets $\alpha_j = 0$.

3. Finally, $\mathcal{B}$ outputs $\sigma' = \alpha_1 \dots \alpha_{\ell_{\mathrm{sig}}}$ as forgery to challenger.

We will now analyze the winning probability of $\mathcal{B}$. In order to do this, we will first define a subset of verification keys which are 'good' (i.e. verification keys for which the difference between the advantages of Att in Game 0 and Game 1 is 'large') and then show that a non-negligible fraction of the verification keys are 'good'. This technique is similar to the *heavy row lemma* used in [OO98].

For any $(m, \mathsf{st}) \leftarrow \mathsf{Att}_1(1^\lambda)$, let $\mathsf{Good}_{m,\mathsf{st}} \subset \mathcal{VK}$ be the set of verification keys VK such that the following holds:

$$\Pr[\mathsf{Att}_2(\mathrm{PP}, \mathrm{VK}, m, \mathsf{st}) \text{ wins in } \mathsf{Game}\ 0] - \Pr[\mathsf{Att}_2(\mathrm{PP}, \mathrm{VK}, m, \mathsf{st}) \text{ wins in } \mathsf{Game}\ 1] \geq \epsilon/2,$$

where the probability is taken over the random coins used by Att_2 and the random coins used by the challenger to compute PP and the signatures.

Let $\mathcal{E}$ denote the event $(m, \mathsf{st}) \leftarrow \mathsf{Att}_1(1^\lambda)$ **and** $(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$ **and** $\mathrm{VK} \in \mathsf{Good}_{m,\mathsf{st}}$. We can also view $\mathcal{E}$ as the set of all tuples $(m, \mathsf{st}, \mathrm{VK})$ such that $(m, \mathsf{st}) \leftarrow \mathsf{Att}_1(1^\lambda)$ and $(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$ and $\mathrm{VK} \in \mathsf{Good}_{m,\mathsf{st}}$.

Claim 4.1. $\Pr[\mathcal{E}] \geq \epsilon/2$, where the probability is over the random coins of Att_1 and $\mathcal{S}.\mathsf{Gen}$.

Proof.

$$\begin{aligned}
\epsilon &= \Pr[\mathsf{Att} \text{ wins in } \mathsf{Game}\ 0] - \Pr[\mathsf{Att} \text{ wins in } \mathsf{Game}\ 1] \\
&= \Pr[\mathsf{Att} \text{ wins in } \mathsf{Game}\ 0|\mathcal{E}] \Pr[\mathcal{E}] + \Pr[\mathsf{Att} \text{ wins in } \mathsf{Game}\ 0|\neg\mathcal{E}] \Pr[\neg\mathcal{E}] \\
&\quad - \Pr[\mathsf{Att} \text{ wins in } \mathsf{Game}\ 1|\mathcal{E}] \Pr[\mathcal{E}] - \Pr[\mathsf{Att} \text{ wins in } \mathsf{Game}\ 1|\neg\mathcal{E}] \Pr[\neg\mathcal{E}] \\
&= \Pr[\mathcal{E}](\Pr[\mathsf{Att} \text{ wins in } \mathsf{Game}\ 0|\mathcal{E}] - \Pr[\mathsf{Att} \text{ wins in } \mathsf{Game}\ 1|\mathcal{E}]) \\
&\quad + \Pr[\neg\mathcal{E}](\Pr[\mathsf{Att} \text{ wins in } \mathsf{Game}\ 0|\neg\mathcal{E}] - \Pr[\mathsf{Att} \text{ wins in } \mathsf{Game}\ 1|\neg\mathcal{E}]) \\
&\leq \Pr[\mathcal{E}] + \epsilon/2
\end{aligned}$$

This implies $\Pr[\mathcal{E}] \geq \epsilon/2$. ∎

Let us assume event $\mathcal{E}$. We will now compute the probability that $\mathcal{B}$ fails to recover forgery, given $\mathcal{E}$. Let p_j denote the probability that the j^{th} bit of forgery σ' is incorrect, given $\mathcal{E}$.

Let $v = (m, \mathsf{st}, \mathrm{VK}) \in \mathcal{E}$. Define $\theta^v_{j,b} = \Pr[\mathsf{Att}_2(\mathrm{PP}, \mathrm{VK}, m, \mathsf{st}) \text{ wins in } \mathsf{Game\text{-}Alt}\ j, b|v]$. Then, the expected value of $\mathsf{count}_{j,b}$ given v, $E[\mathsf{count}_{j,b}|v] = \theta^v_{j,b} \cdot \tau$. Note that in each of the runs, the random coins used by Att_2 and the random coins used by the challenger to compute PP and the signatures are chosen afresh. By Chernoff-Hoeffding bounds,

$$\Pr\left[\left|\mathsf{count}_{j,0} - \theta^v_{j,0} \cdot \tau\right| > \left(\frac{\epsilon}{4}\right) \cdot \tau \middle| v\right] \leq \exp\left(-\left(\frac{\epsilon^2}{32}\right) \cdot \tau\right) \tag{5}$$

$$\Pr\left[\left|\mathsf{count}_{j,1} - \theta^v_{j,1} \cdot \tau\right| > \left(\frac{\epsilon}{4}\right) \cdot \tau \middle| v\right] \leq \exp\left(-\left(\frac{\epsilon^2}{32}\right) \cdot \tau\right) \tag{6}$$

Setting $\tau = \frac{32\lambda}{\epsilon^2}$, we get that the above probabilities are bounded by a negligible function in λ.

We will now compute p_j.

$p_j = \Pr\left[\alpha_j \neq \sigma[j] \middle| \mathcal{E}\right] = \sum_{v \in \mathcal{E}} \Pr\left[\alpha_j \neq \sigma[j] \middle| v\right] \cdot \Pr\left[v \middle| \mathcal{E}\right]$. Let us focus on one such term $\Pr\left[\alpha_j \neq \sigma[j] \middle| v\right]$ for some $v \in \mathcal{E}$. $\Pr\left[\alpha_j \neq \sigma[j] \middle| v\right] = \Pr\left[\alpha_j = 0 \textbf{ and } \sigma[j] = 1 \middle| v\right] + \Pr\left[\alpha_j = 1 \textbf{ and } \sigma[j] = 0 \middle| v\right]$.

Since $\mathcal{S}$ is a unique signature scheme, given v, $\sigma[j]$ is fixed. If $\sigma[j] = 1$, then,

$$\begin{aligned}
&\Pr\left[\alpha_j \neq \sigma[j] \middle| v\right] \\
&= \Pr\left[\alpha_j = 0 \textbf{ and } \sigma[j] = 1 \middle| v\right] \\
&= \Pr\left[\alpha_j = 0 \middle| v\right] \\
&= \Pr\left[\mathsf{count}_{j,0} \leq \mathsf{count}_{j,1} \middle| v\right] \\
&\leq \Pr\left[\mathsf{count}_{j,0} \leq (\theta^v_{j,0} + \theta^v_{j,1})\tau/2 \ \middle| v\right] + \Pr\left[\mathsf{count}_{j,1} \geq (\theta^v_{j,0} + \theta^v_{j,1})\tau/2 \middle| v\right] \\
&= \Pr\left[\mathsf{count}_{j,0} \leq \theta^v_{j,0}\tau/2 - (\theta^v_{j,0} - \theta^v_{j,1})\tau/2 \ \middle| v\right] + \Pr\left[\mathsf{count}_{j,1} \geq \theta^v_{j,1} + (\theta^v_{j,0} - \theta^v_{j,1})\tau/2 \ \middle| v\right]
\end{aligned} \tag{7}$$

Now, note that if $v \in \mathcal{E}$ and $\sigma[j] = 1$, then

$$\begin{aligned}
\theta^v_{j,0} &= \Pr[\mathsf{Att}_2(\mathrm{PP}, \mathrm{VK}, m, \mathsf{st}) \text{ wins in } \mathsf{Game\text{-}Alt}\ j, 0 \middle| v] && (8)\\
&\geq \Pr[\mathsf{Att}_2(\mathrm{PP}, \mathrm{VK}, m, \mathsf{st}) \text{ wins in } \mathsf{Game}\ 0 \middle| v] - \mathrm{negl}(\lambda) && (9)\\
&\geq \Pr[\mathsf{Att}_2(\mathrm{PP}, \mathrm{VK}, m, \mathsf{st}) \text{ wins in } \mathsf{Game}\ 1 \middle| v] + \epsilon/2 - \mathrm{negl}(\lambda) && (10)\\
&\geq \Pr[\mathsf{Att}_2(\mathrm{PP}, \mathrm{VK}, m, \mathsf{st}) \text{ wins in } \mathsf{Game\text{-}Alt}\ j, 1 \middle| v] + \epsilon/2 - \mathrm{negl}(\lambda) && (11)\\
&= \theta^v_{j,1} + \epsilon/2 - \mathrm{negl}(\lambda) && (12)
\end{aligned}$$

The transitions from Equation 8 and 9, and from Equation 10 to 11 follow from Equations 1 and 3 respectively, while the transition from Equation 9 to 10 uses the fact that $v \in \mathcal{E}$. Hence, getting back to Equation 7,

$$\begin{aligned}
&\Pr\left[\mathsf{count}_{j,0} \leq \theta^v_{j,0}\tau - (\theta^v_{j,0} - \theta^v_{j,1})\tau/2 \ \middle| v\right] + \Pr\left[\mathsf{count}_{j,1} \geq \theta^v_{j,1}\tau + (\theta^v_{j,0} - \theta^v_{j,1})\tau/2 \ \middle| v\right] \\
&\leq \Pr\left[\mathsf{count}_{j,0} \leq \theta^v_{j,0}\tau - \epsilon \cdot \tau/4 \ \middle| v\right] + \Pr\left[\mathsf{count}_{j,1} \geq \theta^v_{j,1}\tau + \epsilon \cdot \tau/4 \ \middle| v\right]
\end{aligned}$$

Now, we can use Equations 5 and 6 to conclude that $\Pr\left[\alpha_j \neq \sigma[j] \middle| v\right] \leq \mathrm{negl}(\lambda)$. A similar argument follows if v is such that $\sigma[j] = 0$. Therefore, $\Pr\left[\alpha_j \neq \sigma[j] \middle| \mathcal{E}\right] \leq \mathrm{negl}(\lambda)$. Hence, $\Pr[\mathcal{B} \text{ fails } | \mathcal{E}] \leq \mathrm{negl}(\lambda)$. This implies $\Pr[\mathcal{B} \text{ wins}] \geq \Pr[\mathcal{B} \text{ wins } | \mathcal{E}] \Pr[\mathcal{E}] \geq \epsilon/2 - \mathrm{negl}(\lambda)$. Since this violates the unforgeability of the signature scheme, we have our contradiction.

■

Claim 4.2. Assuming $i\mathcal{O}$ is a secure indistinguishability obfuscator, for any PPT adversary Att,

$$\mathsf{Adv}^1_{\mathsf{Att}} - \mathsf{Adv}^2_{\mathsf{Att}} \leq \mathrm{negl}(\lambda).$$

Proof. Suppose there exists a PPT adversary Att such that $\mathsf{Adv}^1_{\mathsf{Att}} - \mathsf{Adv}^2_{\mathsf{Att}} = \epsilon$. We will construct a PPT algorithm $\mathcal{B}$ that constructs two circuits C_0 and C_1 with identical functionality, and uses Att to distinguish between $i\mathcal{O}(C_0)$ and $i\mathcal{O}(C_1)$, thereby breaking the security of $i\mathcal{O}$.

$\mathcal{B}$ receives m from Att, chooses $(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$, RSA modulus N, $e \leftarrow \mathbb{Z}^*_{\phi(N)}$ and $K \leftarrow F.\mathsf{setup}(1^\lambda)$. It sets $y = \mathcal{S}.\mathsf{Verify}||\mathrm{VK}||m$ and computes $K\{y\} \leftarrow F.\mathsf{puncture}(K, y)$. It sets $C_0 = \mathsf{Transform\text{-}Image}_{N,K,e}$

and $C_1 = \mathsf{Transform\text{-}Image\text{-}1}_{y,N,K\{y\},z,e}$, and sends C_0, C_1 to the $i\mathcal{O}$ challenger. It receives $C = i\mathcal{O}(C_b)$. $\mathcal{B}$ sets $\mathrm{PP} = (i\mathcal{O}(\mathsf{Transform\text{-}Reject}_{y,N,K\{y\}}), C, N, e)$ and sends PP, VK to Att.

Note that $\mathcal{B}$ can respond to the signing queries perfectly, since it has SK. Finally, if Att wins, then $\mathcal{B}$ outputs 0, else it outputs 1. Clearly, if $C = i\mathcal{O}(C_0)$, then it corresponds to Game 1, else it corresponds to Game 2.

To conclude, we need to argue that C_0 and C_1 have identical functionality. This follows from the correctness property of puncturable PRFs. ∎

Claim 4.3. Assuming F is a selectively secure puncturable PRF, for any PPT adversary Att,

$$\mathsf{Adv}^2_{\mathsf{Att}} - \mathsf{Adv}^3_{\mathsf{Att}} \leq \mathrm{negl}(\lambda).$$

Proof. Suppose there exists a PPT adversary Att such that $\mathsf{Adv}^2_{\mathsf{Att}} - \mathsf{Adv}^3_{\mathsf{Att}} = \epsilon$. We will construct a PPT algorithm $\mathcal{B}$ that uses Att to break the security of puncturable PRF F with advantage ϵ.

First, $\mathcal{B}$ receives the message m from Att. As in Game 2 and Game 3, it computes $(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$, chooses an RSA modulus N and $e \leftarrow \mathbb{Z}^*_{\phi(N)}$. Next, it sends $y = \mathcal{S}.\mathsf{Verify}||\mathrm{VK}||m$ as the challenge to the PRF challenger. $\mathcal{B}$ receives a punctured key $K\{y\}$ and $z \in \mathbb{Z}^*_N$, where $z = F(K, y)$ or $z \leftarrow \mathbb{Z}^*_N$. $\mathcal{B}$ sets the public parameters PP=($i\mathcal{O}(\mathsf{Transform\text{-}Reject}_{y,N,K\{y\}})$,$i\mathcal{O}(\mathsf{Transform\text{-}Image\text{-}1}_{y,N,K\{y\},z,e})$, N, e). It sends PP, VK to Att.

The signing phase and forgery phase are exactly similar in Game 2 and Game 3. For each signing query x_i, $\mathcal{B}$ sends $\mathcal{S}.\mathsf{Sign}(\mathrm{SK}, x_i)$ to Att. Finally, Att outputs the forgery σ_{agg} and n tuples $\{(\mathsf{Verify}_i, \mathrm{VK}_i, m_i)\}$.

Note that if $z = F(K, y)$, then $\mathcal{B}$ simulates Game 2 perfectly. If $z \leftarrow \mathbb{Z}^*_N$, $\mathcal{B}$ simulates Game 3 perfectly. This concludes our proof. ∎

Claim 4.4. Assuming RSA is secure, for any PPT adversary Att, $\mathsf{Adv}^3_{\mathsf{Att}} \leq \mathrm{negl}(\lambda)$.

Proof. Suppose there exists a PPT adversary Att such that $\mathsf{Adv}^3_{\mathsf{Att}} = \epsilon$. We will construct a PPT algorithm $\mathcal{B}$ that breaks the RSA assumption with advantage ϵ.

$\mathcal{B}$ receives message m from Att. It receives the RSA tuple (N, e, z) from the RSA challenger. $\mathcal{B}$ chooses $(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$, $K \leftarrow F.\mathsf{setup}(1^\lambda)$. Next, it sets $y = \mathsf{Verify}||\mathrm{VK}||m$ and computes $K\{y\} \leftarrow F.\mathsf{puncture}(K, y)$. It sets $\mathrm{PP} = (i\mathcal{O}(1^\lambda(\mathsf{Transform\text{-}Reject}_{y,N,K\{y\}}), i\mathcal{O}(\mathsf{Transform\text{-}Image\text{-}1}_{y,N,K\{y\},z,e}), N, e)$ and sends PP, VK to Att.

Att sends signature queries, which $\mathcal{B}$ can compute by itself, since it has the signing key SK. Finally, Att outputs forgery σ_{agg} along with n tuples $\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}$. If Att wins, then all n tuples are distinct, and there exists $i^* \in [n]$ such that $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}$, $\mathrm{VK}_{i^*} = \mathrm{VK}$, $m_{i^*} = m$ and $\sigma^e_{\mathrm{agg}} = \left(\prod_{i \neq i^*} \mathsf{Transform\text{-}Image\text{-}1}_{y,N,K\{y\},z,e}(\mathsf{Verify}_i, \mathrm{VK}_i, m_i)\right) z \pmod N$. For all $i \neq i^*$, $\mathsf{Transform\text{-}Image\text{-}1}_{y,N,K\{y\},z,e}$ outputs $F(K, \mathsf{Verify}_i||\mathrm{VK}_i||m_i)^e$ on input $(\mathsf{Verify}_i, \mathrm{VK}_i, m_i)$. Therefore, $\left(\frac{\sigma_{\mathrm{agg}}}{\prod_{i \neq i^*} F(K, \mathsf{Verify}_i||\mathrm{VK}_i||m_i)}\right)^e = z \pmod N$. ∎

Using the above claims, it follows that any PPT adversary has negligible advantage in Game 0, assuming $i\mathcal{O}$ is a secure indistinguishability obfuscator, F is a selectively secure puncturable PRF and the RSA assumption holds. Therefore, the construction in Section 4 is selectively secure with respect to all secure unique signature schemes.

5 Universal Aggregation of Arbitrary Signatures Using VBB Obfuscation

In this section, we will describe our construction based on *virtual black box* obfuscation. The construction is similar to the one in Section 4, the only difference being in program $\mathsf{Transform\text{-}VBB}$, which now takes

some additional inputs and has additional constants hardwired. The additional inputs/constants are used for "oracle assimilation" (see Section 1 for a discussion on this technical issue).

We will assume that all signing algorithms (corresponding to schemes whose signatures need to be aggregated) use at most ℓ_{rnd} random bits to compute signatures, for some polynomial ℓ_{rnd}. We use a pseudorandom generator $\text{PRG} : \{0,1\}^{\ell} \leftarrow \{0,1\}^{2\ell}$ (where ℓ is some polynomial in λ), a (standard) PRF $\tilde{F}$ with key space $\tilde{\mathcal{K}}$, domain $\tilde{\mathcal{X}}$ and range $\tilde{\mathcal{Y}} = \{0,1\}^{\ell_{\text{rnd}}}$ and a puncturable PRF F as in Section 4.

Our universal signature aggregator consists of the three algorithms $\mathsf{UniversalSetup}$, $\mathsf{UniversalAgg}$ and $\mathsf{UniversalVerify}$ described below.

$\mathsf{UniversalSetup}(1^{\lambda})$ $\mathsf{UniversalSetup}$ first chooses random primes $p, q \in \Theta(2^{\lambda})$, sets the RSA modulus $N = pq$. It chooses $e \leftarrow \mathbb{Z}^*_{\phi(N)}$, PRF key $K \leftarrow F.\mathsf{setup}(1^{\lambda})$ as in Section 4. It computes obfuscations of the programs $\mathsf{Transform\text{-}VBB}_{N,K}$ [16] and $\mathsf{Transform\text{-}Image}_{N,K,e}$ [17], where $\mathsf{Transform\text{-}VBB}_{N,K}$ is defined below, while $\mathsf{Transform\text{-}Image}_{N,K,e}$ is the same as in Section 4. It sets the public parameters to be $\text{PP} = (\mathcal{O}(\mathsf{Transform\text{-}VBB}_{N,K}), \mathcal{O}(\mathsf{Transform\text{-}Image}_{N,K,e}), N, e)$.

$\mathsf{Transform\text{-}VBB}_{N,K}$:

Inputs: $b \in \{0,1\}, a \in \{0,1\}^{\ell}, \mathsf{Verify}' \in \{0,1\}^{\ell_{\text{ver}}}$, $\text{VK}' \in \{0,1\}^{\ell_{\text{vk}}}$, $m' \in \{0,1\}^{\ell_{\text{msg}}}$, $\sigma' \in \{0,1\}^{\ell_{\text{sig}}}$.

Constants : RSA modulus $N \in \mathbb{N}$, $K \in \mathcal{K}$.

if $b = 0$ **then**
  Output $\perp$.
else if $\mathsf{Verify}'(\text{VK}', m', \sigma') = 0$ **then**
  Output $\perp$.
else
  Output $F(K, \mathsf{Verify}'||\text{VK}'||m')$.
end if

$\mathsf{UniversalAgg}(\text{PP}, \{(\mathsf{Verify}_i, \text{VK}_i, m_i, \sigma_i)\}_{i=1}^{n})$**:** Let $\text{PP} = (P_1, P_2, N, e)$. $\mathsf{UniversalAgg}$ first checks that all the n tuples are distinct. If not, it outputs $\perp$. Else, it computes $t_i = P_1(\mathsf{Verify}_i, \text{VK}_i, m_i, \sigma_i)$ for each $i \leq n$. If $t_i = \perp$ for some i, then $\mathsf{UniversalAgg}$ outputs $\perp$, else it outputs $\sigma_{\text{agg}} = \prod_i t_i \pmod N$.

$\mathsf{UniversalVerify}(\text{PP}, \{(\mathsf{Verify}_i, \text{VK}_i, m_i)\}_{i=1}^{n}, \sigma_{\text{agg}})$**:** Let $\text{PP} = (P_1, P_2, N, e)$. $\mathsf{UniversalVerify}$ checks that the n tuples are distinct. If not, it outputs 0. Else, it computes, for all $i \leq n$, $s_i = \mathsf{Transform\text{-}Image}(\mathsf{Verify}_i, \text{VK}_i, m_i)$. If $(\prod_i s_i) = \sigma_{\text{agg}}^{e} \pmod N$, it outputs 1, else it outputs 0.

5.1 Proof of Security

We will now prove that the construction in Section 5 is selectively secure with respect to all secure signature schemes.

Theorem 5.1. Assuming $\mathcal{O}$ is a secure virtual black-box obfuscator for a class of circuits $\mathcal{C}$ (as defined in Section 5.1.3), F is a selectively secure puncturable PRF, $\tilde{F}$ is a secure PRF, PRG is a secure pseudorandom generator and RSA is secure, for all $(\ell_{\text{ver}}, \ell_{\text{vk}}, \ell_{\text{msg}}, \ell_{\text{sig}})$-length qualified signature schemes $\mathcal{S}$, the universal signature aggregator $(\ell_{\text{ver}}, \ell_{\text{vk}}, \ell_{\text{msg}}, \ell_{\text{sig}})$-$\mathsf{UniversalSigAgg}$ is selectively secure with respect to $\mathcal{S}$.

We will now describe the intermediate hybrid experiments.

[16] Padded appropriately to be of the same size as $\mathsf{Transform\text{-}VBB\text{-}1}$, $\mathsf{Transform\text{-}VBB\text{-}2}$, $\mathsf{Transform\text{-}VBB\text{-}3}$ defined later in this section.

[17] Padded appropriately to be of the same size as $\mathsf{Transform\text{-}Image\text{-}1}$ as in Section 4.

5.1.1 Sequence of Games

Game 0: This game corresponds to $\mathsf{Exp}^{\mathrm{sel}}_{\mathsf{Att},\mathcal{S}}$.

1. Att sends message m.
2. Compute $(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$. Choose an RSA modulus N, $e \leftarrow \mathbb{Z}^*_{\phi(N)}$, $K \leftarrow F.\mathsf{setup}(1^\lambda)$ and set $\mathrm{PP} = (\mathcal{O}(\mathsf{Transform\text{-}VBB}_{N,K}), \mathcal{O}(\mathsf{Transform\text{-}Image}_{N,K,e}), N, e)$. Send PP, VK to Att.
3. For each signing query $x_i \neq m$, compute $\sigma_i \leftarrow \mathcal{S}.\mathsf{Sign}(\mathrm{SK}, x_i)$ and send σ_i to Att.
4. Att sends forgery σ_{agg} and n tuples $\{(\mathsf{Verify}_i, \mathrm{VK}_i, m_i)\}$. Att wins if $\exists i^* \in [n]$ such that $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}$, $\mathrm{VK}_{i^*} = \mathrm{VK}$ and $m_{i^*} = m$ and $\mathsf{UniversalVerify}(\mathrm{PP}, \{(\mathsf{Verify}_i, \mathrm{VK}_i, m_i)\}, \sigma_{\mathrm{agg}}) = 1$.

Game 1: In this game, the challenger uses pseudorandomly generated strings as randomness for the signature queries.

1. Att sends message m.
2. Compute $(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$. Choose an RSA modulus N, $e \leftarrow \mathbb{Z}^*_{\phi(N)}$, $K \leftarrow F.\mathsf{setup}(1^\lambda)$.
   Choose standard PRF key $\tilde{K} \leftarrow \tilde{F}.\mathsf{setup}(1^\lambda)$.
   Set $\mathrm{PP} = (\mathcal{O}(\mathsf{Transform\text{-}VBB}_{N,K}), \mathcal{O}(\mathsf{Transform\text{-}Image}_{N,K,e}), N, e)$. Send PP, VK to Att.
3. For each signing query $x_i \neq m$, choose $\rho_i \leftarrow \{0,1\}^{\ell_{\mathrm{sig}}}$, compute $r_i = \tilde{F}(\tilde{K}, \rho_i)$, $\sigma_i \leftarrow \mathcal{S}.\mathsf{Sign}(\mathrm{SK}, x_i; r_i)$ and send σ_i to Att.
4. Att sends forgery σ_{agg} and n tuples $\{(\mathsf{Verify}_i, \mathrm{VK}_i, m_i)\}$. Att wins if $\exists i^* \in [n]$ such that $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}$, $\mathrm{VK}_{i^*} = \mathrm{VK}$ and $m_{i^*} = m$ and $\mathsf{UniversalVerify}(\mathrm{PP}, \{(\mathsf{Verify}_i, \mathrm{VK}_i, m_i)\}, \sigma_{\mathrm{agg}}) = 1$.

Game 2: In this game, the challenger uses the program Transform-VBB-1 instead of Transform-VBB. Unlike Transform-VBB, Transform-VBB-1 uses the input a to check if $\mathrm{PRG}(a)$ is equal to the hardwired α. If the 'mode' bit is 0 and $\mathrm{PRG}(a) = \alpha$, then the program outputs the verification key VK and a signature on the desired message.

1. Att sends message m.
2. Compute $(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$. Choose an RSA modulus N, $e \leftarrow \mathbb{Z}^*_{\phi(N)}$, $K \leftarrow F.\mathsf{setup}(1^\lambda)$.
   Choose PRF key $\tilde{K} \leftarrow \tilde{F}.\mathsf{setup}(1^\lambda)$, $\alpha \leftarrow \{0,1\}^{2\ell}$.
   Let Transform-VBB-1[18] be the circuit defined below.
   Set $\mathrm{PP} = (\mathcal{O}(\mathsf{Transform\text{-}VBB\text{-}1}_{N,K,\alpha,\mathrm{SK},\tilde{K}}), \mathcal{O}(\mathsf{Transform\text{-}Image}_{N,K,e}), N, e)$. Send PP, VK to Att.
3. For each signing query $x_i \neq m$, choose $\rho_i \leftarrow \{0,1\}^{\ell_{\mathrm{sig}}}$, compute $r_i = \tilde{F}(\tilde{K}, \rho_i)$, $\sigma_i \leftarrow \mathcal{S}.\mathsf{Sign}(\mathrm{SK}, x_i; r_i)$ and send σ_i to Att.
4. Att sends forgery σ_{agg} and n tuples $\{(\mathsf{Verify}_i, \mathrm{VK}_i, m_i)\}$. Att wins if $\exists i^* \in [n]$ such that $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}$, $\mathrm{VK}_{i^*} = \mathrm{VK}$ and $m_{i^*} = m$ and $\mathsf{UniversalVerify}(\mathrm{PP}, \{(\mathsf{Verify}_i, \mathrm{VK}_i, m_i)\}, \sigma_{\mathrm{agg}}) = 1$.

[18] Padded appropriately to be of the same size as Transform-VBB, Transform-VBB-2 and Transform-VBB-3.

$\mathsf{Transform\text{-}VBB\text{-}1}_{N,K,\alpha,\mathrm{SK},\tilde{K}}$:

Inputs: $b \in \{0,1\}, a \in \{0,1\}^{\ell}, \mathsf{Verify}' \in \{0,1\}^{\ell_{\mathrm{ver}}}$, $\mathrm{VK}' \in \{0,1\}^{\ell_{\mathrm{vk}}}$, $m' \in \{0,1\}^{\ell_{\mathrm{msg}}}$, $\sigma' \in \{0,1\}^{\ell_{\mathrm{sig}}}$.

Constants : RSA modulus $N \in \mathbb{N}$, $K \in \mathcal{K}$, $\alpha \in \{0,1\}^{2\ell}$, $\mathrm{SK} \in \mathcal{SK}$, $\tilde{K} \in \tilde{\mathcal{K}}$.

if $b = 0$ **then**
  if $\mathrm{PRG}(a) \neq \alpha$ **then**
    Output $\perp$.
  else
    Output $(\mathrm{VK}, \mathcal{S}.\mathsf{Sign}(\mathrm{SK}, m'; \tilde{F}(\tilde{K}, \sigma')))$.
  end if
else if $\mathsf{Verify}'(\mathrm{VK}', m', \sigma') = 0$ **then**
  Output $\perp$.
else
  Output $F(K, \mathsf{Verify}'||\mathrm{VK}'||m')$.
end if

Game 3: In this experiment, α is a pseudorandom string; i.e. $\alpha = \mathrm{PRG}(a)$, where $a \leftarrow \{0,1\}^{\ell}$.

1. Att sends message m.
2. Compute $(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^{\lambda})$. Choose an RSA modulus N, $e \leftarrow \mathbb{Z}^*_{\phi(N)}$, $K \leftarrow F.\mathsf{setup}(1^{\lambda})$. Choose $a \leftarrow \{0,1\}^{\ell}$ and set $\alpha = \mathrm{PRG}(a)$. Choose PRF key $\tilde{K} \leftarrow \tilde{F}.\mathsf{setup}(1^{\lambda})$. Set $\mathrm{PP} = (\mathcal{O}(\mathsf{Transform\text{-}VBB\text{-}1}_{N,K,\alpha,\mathrm{SK},\tilde{K}}), \mathcal{O}(\mathsf{Transform\text{-}Image}_{N,K,e}), N, e)$. Send PP, VK to Att.
3. For each signing query $x_i \neq m$, choose $\rho_i \leftarrow \{0,1\}^{\ell_{\mathrm{sig}}}$, compute $r_i = \tilde{F}(\tilde{K}, \rho_i)$, $\sigma_i \leftarrow \mathcal{S}.\mathsf{Sign}(\mathrm{SK}, x_i; r_i)$ and send σ_i to Att.
4. Att sends forgery σ_{agg} and n tuples $\{(\mathsf{Verify}_i, \mathrm{VK}_i, m_i)\}$. Att wins if $\exists i^* \in [n]$ such that $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}$, $\mathrm{VK}_{i^*} = \mathrm{VK}$ and $m_{i^*} = m$ and $\mathsf{UniversalVerify}(\mathrm{PP}, \{(\mathsf{Verify}_i, \mathrm{VK}_i, m_i)\}, \sigma_{\mathrm{agg}}) = 1$.

Game 4: This experiment is similar to the previous one, except that the challenger uses Transform-VBB-2 instead of Transform-VBB-1.

1. Att sends message m.
2. Compute $(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^{\lambda})$. Choose an RSA modulus N, $e \leftarrow \mathbb{Z}^*_{\phi(N)}$, $K \leftarrow F.\mathsf{setup}(1^{\lambda})$. Choose $a \leftarrow \{0,1\}^{\ell}$ and set $\alpha = \mathrm{PRG}(a)$. Choose $\tilde{K} \leftarrow \tilde{F}.\mathsf{setup}(1^{\lambda})$. Let Transform-VBB-2[19] be the circuit defined below. Set $y = \mathcal{S}.\mathsf{Verify}||\mathrm{VK}||m$, $\mathrm{PP} = (\mathcal{O}(\mathsf{Transform\text{-}VBB\text{-}2}_{y,N,K,\alpha,\mathrm{SK},\tilde{K}}), \mathcal{O}(\mathsf{Transform\text{-}Image}_{N,K,e}), N, e)$. Send PP, VK to Att.
3. For each signing query $x_i \neq m$, choose $\rho_i \leftarrow \{0,1\}^{\ell_{\mathrm{sig}}}$, compute $r_i = \tilde{F}(\tilde{K}, \rho_i)$ and send $\sigma_i = \mathcal{S}.\mathsf{Sign}(\mathrm{SK}, x_i; r_i)$ to Att.
4. Att sends forgery σ_{agg} and n tuples $\{(\mathsf{Verify}_i, \mathrm{VK}_i, m_i)\}$. Att wins if $\exists i^* \in [n]$ such that $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}$, $\mathrm{VK}_{i^*} = \mathrm{VK}$ and $m_{i^*} = m$ and $\mathsf{UniversalVerify}(\mathrm{PP}, \{(\mathsf{Verify}_i, \mathrm{VK}_i, m_i)\}, \sigma_{\mathrm{agg}}) = 1$.

[19] Padded appropriately to be of the same size as Transform-VBB, Transform-VBB-1 and Transform-VBB-3.

$\mathsf{Transform\text{-}VBB\text{-}2}_{y,N,K,\alpha,\mathrm{SK},\tilde{K}}$:

Inputs: $b \in \{0,1\}, a \in \{0,1\}^{\ell}, \mathsf{Verify}' \in \{0,1\}^{\ell_{\mathrm{ver}}}$, $\mathrm{VK}' \in \{0,1\}^{\ell_{\mathrm{vk}}}$, $m' \in \{0,1\}^{\ell_{\mathrm{msg}}}$, $\sigma' \in \{0,1\}^{\ell_{\mathrm{sig}}}$.

Constants : $y \in \{0,1\}^{\ell_{\mathrm{ver}}} \times \{0,1\}^{\ell_{\mathrm{vk}}} \times \{0,1\}^{\ell_{\mathrm{msg}}}$,RSA modulus $N \in \mathbb{N}$, $K \in \mathcal{K}$, $\alpha \in \{0,1\}^{2\ell}$, $\mathrm{SK} \in \mathcal{SK}$, $\tilde{K} \in \tilde{\mathcal{K}}$.

if $b = 0$ **then**
  if $\mathrm{PRG}(a) \neq \alpha$ **then**
    Output $\perp$.
  else
    Output $(\mathrm{VK}, \mathcal{S}.\mathsf{Sign}(\mathrm{SK}, m'; \tilde{F}(\tilde{K}, \sigma')))$.
  end if
else if $\mathsf{Verify}'(\mathrm{VK}', m', \sigma') = 0$ **then**
  Output $\perp$.
else if $\mathsf{Verify}'||\mathrm{VK}'||m' = y$ **then**
  Output $\perp$.
else
  Output $F(K, \mathsf{Verify}'||\mathrm{VK}'||m')$.
end if

Game 5: In this experiment, the challenger uses a key punctured at y instead of the master PRF key.

1. Att sends message m.
2. Compute $(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^{\lambda})$. Choose an RSA modulus N, $e \leftarrow \mathbb{Z}^*_{\phi(N)}$, $K \leftarrow F.\mathsf{setup}(1^{\lambda})$. Choose $a \leftarrow \{0,1\}^{\ell}$ and set $\alpha = \mathrm{PRG}(a)$. Choose $\tilde{K} \leftarrow \tilde{F}.\mathsf{setup}(1^{\lambda})$. Set $y = \mathcal{S}.\mathsf{Verify}||\mathrm{VK}||m$, compute $K\{y\} \leftarrow F.\mathsf{puncture}(K, y)$ and $z = F(K, y)^e$. Let $\mathsf{Transform\text{-}VBB\text{-}3}$[20] be the circuit defined below, while $\mathsf{Transform\text{-}Image\text{-}1}$[21]$,e)$ is the same as in Section 4.1 Set PP $=(\mathcal{O}(\mathsf{Transform\text{-}VBB\text{-}3}_{y,N,K\{y\},\alpha,\mathrm{SK},\tilde{K}}), \mathcal{O}(\mathsf{Transform\text{-}Image\text{-}1}_{y,N,K\{y\},z,e})$. Send PP, VK to Att.
3. For each signing query $x_i \neq m$, choose $\rho_i \leftarrow \{0,1\}^{\ell_{\mathrm{sig}}}$, compute $r_i = \tilde{F}(\tilde{K}, \rho_i)$ and send $\sigma_i = \mathcal{S}.\mathsf{Sign}(\mathrm{SK}, x_i; r_i)$ to Att.
4. Att sends forgery σ_{agg} and n tuples $\{(\mathsf{Verify}_i, \mathrm{VK}_i, m_i)\}$. Att wins if $\exists i^* \in [n]$ such that $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}$, $\mathrm{VK}_{i^*} = \mathrm{VK}$ and $m_{i^*} = m$ and $\mathsf{UniversalVerify}(\ \mathrm{PP}, \{(\mathsf{Verify}_i, \mathrm{VK}_i, m_i)\ \}, \sigma_{\mathrm{agg}}) = 1$.

[20] Padded appropriately to be of the same size as Transform-VBB, Transform-VBB-1 and Transform-VBB-2.
[21] Padded appropriately to be of the same size as Transform-Image-1.

$\mathsf{Transform\text{-}VBB\text{-}3}_{y,N,K\{y\},\alpha,\mathrm{SK},\tilde{K}}$:

Inputs: $b \in \{0,1\}, a \in \{0,1\}^\ell, \mathsf{Verify}' \in \{0,1\}^{\ell_{\mathrm{ver}}}$, $\mathrm{VK}' \in \{0,1\}^{\ell_{\mathrm{vk}}}$, $m' \in \{0,1\}^{\ell_{\mathrm{msg}}}$, $\sigma' \in \{0,1\}^{\ell_{\mathrm{sig}}}$.

Constants : $y \in \{0,1\}^{\ell_{\mathrm{ver}}} \times \{0,1\}^{\ell_{\mathrm{vk}}} \times \{0,1\}^{\ell_{\mathrm{msg}}}$, RSA modulus $N \in \mathbb{N}$, $K\{y\} \in \mathcal{K}_p$, $\alpha \in \{0,1\}^{2\ell}$, $\mathrm{SK} \in \mathcal{SK}$, $\tilde{K} \in \tilde{\mathcal{K}}$.

if $b = 0$ **then**
  if $\mathrm{PRG}(a) \neq \alpha$ **then**
    Output $\perp$.
  else
    Output $(\mathrm{VK}, \mathcal{S}.\mathsf{Sign}(\mathrm{SK}, m'; \tilde{F}(\tilde{K}, \sigma')))$.
  end if
else if $\mathsf{Verify}'(\mathrm{VK}', m', \sigma') = 0$ **then**
  Output $\perp$.
else if $\mathsf{Verify}'||\mathrm{VK}'||m' = y$ **then**
  Output $\perp$.
else
  Output $F.\mathsf{eval}(K\{y\}, \mathsf{Verify}'||\mathrm{VK}'||m')$.
end if

Game 6: Here the challenger chooses a uniformly random $z \leftarrow \mathbb{Z}_N^*$.

1. Att sends message m.
2. Compute $(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$. Choose an RSA modulus N, $e \leftarrow \mathbb{Z}_{\phi(N)}^*$, $K \leftarrow F.\mathsf{setup}(1^\lambda)$. Choose $a \leftarrow \{0,1\}^\ell$ and set $\alpha = \mathrm{PRG}(a)$. Choose $\tilde{K} \leftarrow \tilde{F}.\mathsf{setup}(1^\lambda)$. Set $y = \mathcal{S}.\mathsf{Verify}||\mathrm{VK}||m$, compute $K\{y\} \leftarrow F.\mathsf{puncture}(K, y)$ and $z \leftarrow \mathbb{Z}_N^*$. Set PP $=(\mathcal{O}(\mathsf{Transform\text{-}VBB\text{-}3}_{y,N,K\{y\},\alpha,\mathrm{SK},\tilde{K}}), \mathcal{O}(\mathsf{Transform\text{-}Image\text{-}1}_{y,N,K\{y\},z,e}), e)$. Send PP, VK to Att.
3. For each signing query $x_i \neq m$, compute $\sigma_i \leftarrow \mathcal{S}.\mathsf{Sign}(\mathrm{SK}, x_i)$ and send σ_i to Att.
4. Att sends forgery σ_{agg} and n tuples $\{(\mathsf{Verify}_i, \mathrm{VK}_i, m_i)\}$. Att wins if $\exists i^* \in [n]$ such that $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}$, $\mathrm{VK}_{i^*} = \mathrm{VK}$ and $m_{i^*} = m$ and $\mathsf{UniversalVerify}(\mathrm{PP}, \{(\mathsf{Verify}_i, \mathrm{VK}_i, m_i)\}, \sigma_{\mathrm{agg}}) = 1$.

5.1.2 Analysis

We will now show that if a PPT adversary has non negligible advantage in Game i, then it has non-negligible advantage in the next game. Some of the proofs are exactly similar to the corresponding ones in Section 4.1, and hence we skip them in this section. Except the part involving oracle assimilation, the remaining proofs are relatively easier. The step involving oracle assimilation is discussed in a separate subsection (Section 5.1.3).

Let $\mathsf{Adv}_{\mathsf{Att}}^j$ denote the advantage of adversary Att in Game j.

Claim 5.1. Assuming $\tilde{F}$ is a secure PRF, for any PPT adversary Att,

$$\mathsf{Adv}_{\mathsf{Att}}^0 - \mathsf{Adv}_{\mathsf{Att}}^1 \leq \mathrm{negl}(\lambda).$$

Proof. Suppose there exists an adversary Att such that $\mathsf{Adv}_{\mathsf{Att}}^0 - \mathsf{Adv}_{\mathsf{Att}}^1 = \epsilon$. We will construct a PPT algorithm $\mathcal{B}$ that uses Att and breaks the security of $\tilde{F}$ with advantage ϵ.

$\mathcal{B}$ receives message m from Att. It chooses $(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$, RSA modulus N, $e \leftarrow \mathbb{Z}_{\phi(N)}^*$ and $K \leftarrow F.\mathsf{setup}(1^\lambda)$. It sets PP $=(\mathcal{O}(\mathsf{Transform\text{-}VBB}_{N,K}), \mathcal{O}(\mathsf{Transform\text{-}Image}_{N,K,e}), e)$ and sends PP, VK to Att.

For each signing query x_i, $\mathcal{B}$ first chooses $\rho_i \leftarrow \{0,1\}^{\ell_{\text{sig}}}$ and sends ρ_i to the PRF challenger. In response, it receives r_i. $\mathcal{B}$ sends $\sigma_i = \mathcal{S}.\mathsf{Sign}(\text{SK}, x_i; r_i)$ to Att.

Finally, Att outputs a forgery. If Att wins, then $\mathcal{B}$ outputs 1, indicating that the PRF challenger's responses were truly random. Else it outputs 0.

If the PRF challenger's responses were truly random, then for each query ρ_i, r_i is a truly random string. Therefore, this corresponds to Game 0. If the PRF challenger's responses were pseudorandom, then there exists a PRF key $\tilde{K}$ such that for each query ρ_i, $r_i = \tilde{F}(\tilde{K}, \rho_i)$. This corresponds to Game 1. Therefore, $\mathsf{Adv}^{\tilde{F}}_{\mathcal{B}} = \epsilon$. ∎

Claim 5.2. Assuming $\mathcal{O}$ is a secure indistinguishability obfuscator, for any PPT adversary Att,

$$\mathsf{Adv}^1_{\mathsf{Att}} - \mathsf{Adv}^2_{\mathsf{Att}} \leq \text{negl}(\lambda).$$

Proof. Suppose there exists a PPT adversary Att such that $\mathsf{Adv}^1_{\mathsf{Att}} - \mathsf{Adv}^1_{\mathsf{Att}} = \epsilon$. We will construct a PPT algorithm $\mathcal{B}$ that breaks the security of $\mathcal{O}$ with advantage ϵ.

$\mathcal{B}$ receives message m from Att. It chooses $(\text{SK}, \text{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$, RSA modulus N, $e \leftarrow \mathbb{Z}^*_{\phi(N)}$, $K \leftarrow F.\mathsf{setup}(1^\lambda)$, $\tilde{K} \leftarrow \tilde{F}.\mathsf{setup}(1^\lambda)$ and $\alpha \leftarrow \{0,1\}^{2\ell}$. It sets $C_0 = \mathsf{Transform\text{-}VBB}_{N,K}$, $C_1 = \mathsf{Transform\text{-}VBB\text{-}1}_{N,K,\alpha,\text{SK},\tilde{K}}$ and sends $\mathcal{C}_0, C_1$ to the $\mathcal{O}$ challenger. It receives an obfuscated circuit $C' = \mathcal{O}(C_b)$ in response, and sets $\text{PP} = (C', \mathcal{O}(\mathsf{Transform\text{-}Image}_{N,K,e}), e)$ and sends PP, VK to Att.

For each signing query x_i, $\mathcal{B}$ first chooses $\rho_i \leftarrow \{0,1\}^{\ell_{\text{sig}}}$ and computes $r_i = \tilde{F}(\tilde{K}, \rho_i)$. $\mathcal{B}$ sends $\sigma_i = \mathcal{S}.\mathsf{Sign}(\text{SK}, x_i; r_i)$ to Att.

Finally, Att outputs a forgery. If Att wins, then $\mathcal{B}$ outputs 0, else it outputs 1. Clearly, if $b = 0$, then this corresponds to Game 1, else it corresponds to Game 2. Therefore, in order to show that $\mathsf{Adv}^{\mathcal{O}}_{\mathcal{B}} = \epsilon$, we need to show that C_0 and C_1 have identical functionality.

This follows from the observation that with overwhelming probability, there exists no $a \in \{0,1\}^\ell$ such that $\alpha = \text{PRG}(a)$, since α is chosen uniformly at random. As a result, on input $(0, a, \mathsf{Verify}', \text{VK}', m', \sigma')$, both circuits output $\perp$ for all $a, \mathsf{Verify}', \text{VK}', m', \sigma'$. This concludes our proof. ∎

Claim 5.3. Assuming PRG is a secure pseudorandom generator, for any PPT adversary Att,

$$\mathsf{Adv}^2_{\mathsf{Att}} - \mathsf{Adv}^3_{\mathsf{Att}} \leq \text{negl}(\lambda).$$

Proof. Suppose there exists a PPT adversary Att such that $\mathsf{Adv}^2_{\mathsf{Att}} - \mathsf{Adv}^3_{\mathsf{Att}} = \epsilon$. We will construct a PPT algorithm $\mathcal{B}$ that breaks the security of PRG with advantage ϵ.

$\mathcal{B}$ receives α from the PRG challenger, where $\alpha \leftarrow \{0,1\}^\ell$ or $\alpha = \text{PRG}(a)$ for some $a \leftarrow \{0,1\}^\ell$. Note that $\mathcal{B}$ can simulate either Game 1 or Game 2 perfectly using α. It chooses $(\text{SK}, \text{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$, RSA modulus N, $e \leftarrow \mathbb{Z}^*_{\phi(N)}$, $K \leftarrow F.\mathsf{setup}(1^\lambda)$, $\tilde{K} \leftarrow \tilde{F}.\mathsf{setup}(1^\lambda)$. $\mathcal{B}$ sets $\text{PP} = (\mathcal{O}(\mathsf{Transform\text{-}VBB\text{-}1}_{N,K,\alpha,\text{SK},\tilde{K}}), \mathcal{O}(\mathsf{Transform\text{-}Image}_{N,K,e}), e)$ and sends PP, VK to Att.

For the signature queries, $\mathcal{B}$ uses SK. Finally, if Att wins, $\mathcal{B}$ outputs 1 (indicating that $\alpha \leftarrow \{0,1\}^{2\ell}$). Else it outputs 0. Clearly, $\mathsf{Adv}_{\mathcal{B}} = \epsilon$. This concludes our proof. ∎

Lemma 5.1. Assuming $\mathcal{O}$ is a secure virtual black box obfuscator for a class of circuits $\mathcal{C}$ (defined in Section 5.1.3), $\tilde{F}$ is a secure pseudorandom function, PRG is a secure pseudorandom generator and $\mathcal{S}$ is a (ℓ_{ver}, ℓ_{vk}, ℓ_{msg}, ℓ_{sig})-length qualified secure signature scheme,

$$\mathsf{Adv}^3_{\mathsf{Att}} - \mathsf{Adv}^4_{\mathsf{Att}} \leq \text{negl}(\lambda).$$

The proof of this lemma consists of multiple intermediate hybrids, and is contained in Section 5.1.3.

Claim 5.4. Assuming $\mathcal{O}$ is a secure indistinguishability obfuscator, for any PPT adversary Att,

$$\mathsf{Adv}^4_{\mathsf{Att}} - \mathsf{Adv}^5_{\mathsf{Att}} \leq \text{negl}(\lambda).$$

Proof. Similar to proof of Claim 4.2. ∎

Claim 5.5. Assuming F is a selectively secure puncturable PRF, for any PPT adversary Att,

$$\mathsf{Adv}^5_{\mathsf{Att}} - \mathsf{Adv}^6_{\mathsf{Att}} \leq \mathrm{negl}(\lambda).$$

Proof. Similar to proof of Claim 4.3. ∎

Claim 5.6. Assuming RSA is secure, for any PPT adversary Att,

$$\mathsf{Adv}^6_{\mathsf{Att}} \leq \mathrm{negl}(\lambda).$$

Proof. Similar to proof of Claim 4.4. ∎

Using the above claims, we can conclude that any PPT adversary has at most negligible advantage in Game 0, assuming $\mathcal{O}$ is a secure virtual black-box obfuscator for circuit family $\mathcal{C}$, F is a selectively secure puncturable PRF, $\tilde{F}$ is a secure (standard) PRF, PRG is a secure pseudorandom generator, and RSA is secure. Therefore, the construction described in Section 5 is selectively secure with respect to all secure length-qualified signature schemes.

5.1.3 Proof of Lemma 5.1

Proof. Let Att be a PPT adversary such that $\mathsf{Adv}^3_{\mathsf{Att}} - \mathsf{Adv}^4_{\mathsf{Att}} = \epsilon$. As in proof of Lemma 4.1, we will assume that $\mathsf{Att} = (\mathsf{Att}_1, \mathsf{Att}_2)$ where Att_1 takes as input the security parameter λ and outputs (m, st), where st denotes some state information. Att_2 takes as input $m, \mathsf{st}, \mathrm{PP}, \mathrm{VK}$, issues signature queries and finally outputs a forgery.

Let us assume $\mathsf{rnd}_{\mathsf{RSA}} = \mathsf{rnd}_{\mathsf{RSA}}(\lambda)$ bits are used to choose the RSA modulus N, $\mathsf{rnd}_{\mathsf{F}} = \mathsf{rnd}_{\mathsf{F}}(\lambda)$ bits are used by $F.\mathsf{setup}(1^\lambda)$ to choose a PRF key $K \in \mathcal{K}$ and $\mathsf{rnd}_{\mathsf{Att}} = \mathsf{rnd}_{\mathsf{Att}}(\lambda)$ bits are used by Att_1 to compute (m, st). Let $\mathcal{V}_\lambda = \{(a, r_N, r_K, r_{\mathsf{Att}}) \mid a \in \{0,1\}^\ell,\ r_N \in \{0,1\}^{\mathsf{rnd}_{\mathsf{RSA}}},\ r_K \in \{0,1\}^{\mathsf{rnd}_{\mathsf{F}}},\ r_{\mathsf{Att}} \in \{0,1\}^{\mathsf{rnd}_{\mathsf{Att}}}\}$. For any $v = (a, r_N, r_K, r_{\mathsf{Att}}) \in \mathcal{V}_\lambda$, let N_v denote the RSA modulus generated by r_N, $K_v = F.\mathsf{setup}(1^\lambda; r_K)$ and $(m_v, \mathsf{st}_v) = \mathsf{Att}_1(1^\lambda; r_{\mathsf{Att}})$. Let $\mathcal{C}^0_{\lambda,v}$ denote the family of circuits corresponding to $\mathsf{Transform\text{-}VBB\text{-}1}$; that is

$$\mathcal{C}^0_{\lambda,v} = \{\mathsf{Transform\text{-}VBB\text{-}1}_{N_v, K_v, \alpha, \mathrm{SK}, \tilde{K}} : \alpha = \mathrm{PRG}(a), \mathrm{SK} \in \mathcal{SK}, \mathrm{VK} \in \mathcal{VK}, \tilde{K} \in \tilde{\mathcal{K}}\}.$$

Similarly, $\mathcal{C}^1_{\lambda,v}$ denotes the circuits corresponding to $\mathsf{Transform\text{-}VBB\text{-}2}$; that is

$$\mathcal{C}^1_{\lambda,v} = \{\mathsf{Transform\text{-}VBB\text{-}2}_{y, N_v, K_v, \alpha, \mathrm{SK}, \tilde{K}} : y = \mathcal{S}.\mathsf{Verify}||\mathrm{VK}||m_v, \alpha = \mathrm{PRG}(a), \mathrm{SK} \in \mathcal{SK}, \mathrm{VK} \in \mathcal{VK}, \tilde{K} \in \tilde{\mathcal{K}}\}.$$

When the context is clear, we will drop the dependence of N_v, K_v, m_v and st_v on v. We will now define a PPT algorithm Alg_v that takes as input a circuit $C' \in \mathcal{C}^0_{\lambda,v} \cup \mathcal{C}^1_{\lambda,v}$, has v hardwired, interacts with Att_2 and outputs a bit b'.

Alg_v:

Inputs: Circuit $C' \in \mathcal{C}^0_{\lambda,v} \cup \mathcal{C}^1_{\lambda,v}$
Constants: $v = (a, r_N, r_K, r_{\mathsf{Att}}) \in \{0,1\}^\ell \times \{0,1\}^{\mathsf{rnd}_{\mathsf{RSA}}} \times \{0,1\}^{\mathsf{rnd}_{\mathsf{F}}} \times \{0,1\}^{\mathsf{rnd}_{\mathsf{Att}}}$

1. Compute p, q using r_N, set $N = pq$ and choose $e \leftarrow \mathbb{Z}^*_{\phi(N)}$. Compute $K \leftarrow F.\mathsf{setup}(1^\lambda; r_K)$.
2. Choose $\mathsf{Verify}' \leftarrow \{0,1\}^{\ell_{\mathrm{ver}}}$, $\mathrm{VK}' \leftarrow \{0,1\}^{\ell_{\mathrm{vk}}}$, $m' \leftarrow \{0,1\}^{\ell_{\mathrm{msg}}}$, $\sigma' \leftarrow \{0,1\}^{\ell_{\mathrm{sig}}}$ and compute $(\mathrm{VK}, \rho) = C'(0, a, \mathsf{Verify}', \mathrm{VK}', m', \sigma')$.
3. Compute $P_2 \leftarrow \mathcal{O}(\mathsf{Transform\text{-}Image}_{N,K,e})$. Set $\mathrm{PP} = (C', P_2, e)$ and send $\mathrm{PP}, \mathrm{VK}, m, \mathsf{st}$ to Att_2.
4. For each signing query x_i, $\mathcal{A}_v$ chooses $\sigma' \leftarrow \{0,1\}^{\ell_{\mathrm{sig}}}$ computes $C'(0, a, \mathcal{S}.\mathsf{Verify}, \mathrm{VK}, x_i, \sigma') = \sigma_i$ and sends σ_i to Att.
5. Att_2 sends forgery σ_{agg} and n tuples $\{(\mathsf{Verify}_i, \mathrm{VK}_i, m_i)\}$. If $\exists i^* \in [n]$ such that $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}$, $\mathrm{VK}_{i^*} = \mathrm{VK}$ and $m_{i^*} = m$ and $\mathsf{UniversalVerify}(\mathrm{PP}, \{(\mathsf{Verify}_i, \mathrm{VK}_i, m_i)\}, \sigma_{\mathrm{agg}}) = 1$, then $\mathcal{A}_v$ outputs 1. Else it outputs 0.

Consider the following experiment Exp^b_v: Compute RSA modulus N using r_N, $K = F.\mathsf{setup}(1^\lambda; r_K)$, $\alpha = \mathrm{PRG}(a)$ and $(m, \mathsf{st}) = \mathsf{Att}_1(1^\lambda; r_{\mathsf{Att}})$. Choose $(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$ and $\tilde{K} \leftarrow \tilde{F}.\mathsf{setup}(1^\lambda)$. If $b = 0$, set $C' \leftarrow \mathcal{O}(\mathsf{Transform\text{-}VBB\text{-}1}_{N,K,\alpha,\mathrm{SK},\tilde{K}})$, else set $C' \leftarrow \mathcal{O}(\mathsf{Transform\text{-}VBB\text{-}2}_{y,N,K,\alpha,\mathrm{SK},\tilde{K}})$, where $y = \mathcal{S}.\mathsf{Verify}||\mathrm{VK}||m$. Output $\mathsf{Alg}_v(C')$.

From the definition of Exp^0_v and Alg_v, it follows that $\Pr\left[1 \leftarrow \mathsf{Exp}^0_v\right] = \Pr\left[\mathsf{Att} \text{ wins in } \mathsf{Game}\ 3 | v\right]$. Similarly, $\Pr\left[1 \leftarrow \mathsf{Exp}^1_v\right] = \Pr\left[\mathsf{Att} \text{ wins in } \mathsf{Game}\ 4\right]$. Hence, $E\left[\Pr\left[1 \leftarrow \mathsf{Exp}^0_v\right] - \Pr\left[1 \leftarrow \mathsf{Exp}^1_v\right]\right] = \epsilon$, where the expectation is over the choice of $v \leftarrow \mathcal{V}_\lambda$. Let $v^* = v^*(\lambda) = \arg\max_{v \in \mathcal{V}} \{\Pr[1 \leftarrow \mathsf{Exp}^0_v] - \Pr[1 \leftarrow \mathsf{Exp}^1_v]\}$. Then, it follows that

$$\Pr[1 \leftarrow \mathsf{Exp}^0_{v^*}] - \Pr[1 \leftarrow \mathsf{Exp}^1_{v^*}] \geq \epsilon. \tag{13}$$

Using Alg, we can now define our non-uniform algorithm $\mathcal{A}$. For each security parameter λ, $\mathcal{A}(1^\lambda) = \mathsf{Alg}_{v^*(\lambda)}$.

Now, consider the class of circuits $\mathcal{C}_\lambda = \mathcal{C}^0_{\lambda,v^*} \cup \mathcal{C}^1_{\lambda,v^*}$. We will require our obfuscator $\mathcal{O}$ to be a virtual black box obfuscator for circuit class $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$.

From the security property of VBB obfuscator, it follows that there exists a PPT simulator S corresponding to $\mathcal{A}$ such that

$$\Pr\left[\mathcal{A}\left(\mathcal{O}(C)\right) = 1\right] - \Pr\left[S^C\left(1^{|C|}\right) = 1\right] \leq \mathrm{negl}(\lambda) \tag{14}$$

for all circuits $C \in \mathcal{C}_\lambda$ and the probabilities are over the random coins of $\mathcal{A}$ and S respectively.

Therefore, from Equations 13 and 14, we get the following observation.

Observation 5.1. Let $(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$ and $\tilde{K} \leftarrow \tilde{F}.\mathsf{setup}(1^\lambda)$. Let $v^* = (a, r_N, r_K, r_{\mathsf{Att}})$, $\alpha = \mathrm{PRG}(a)$. N_{v^*}, K_{v^*} and $(m_{v^*}, \mathsf{st}_{v^*})$ computed using r_N, r_K and r_{Att} respectively, and $y = \mathcal{S}.\mathsf{Verify}||\mathrm{VK}||m_{v^*}$. Let $C_0 = \mathsf{Transform\text{-}VBB\text{-}1}_{N_{v^*},K_{v^*},\alpha,\mathrm{SK},\tilde{K}}$ and $C_1 = \mathsf{Transform\text{-}VBB\text{-}2}_{y,N_{v^*},K_{v^*},\alpha,\mathrm{SK},\tilde{K}}$. Then

$$\left|\Pr\left[S^{C_0}\left(1^{|C_0|}\right) = 1\right] - \Pr\left[S^{C_1}\left(1^{|C_1|}\right) = 1\right]\right| \geq \epsilon - \mathrm{negl}(\lambda)$$

where the probabilities are over the choice of $(\mathrm{SK}, \mathrm{VK})$, $\tilde{K}$ and the random coins of S.

We will show that this leads to a contradiction. Consider the algorithm $\mathsf{Transform\text{-}VBB}'\text{-}1$ which is exactly similar to the circuit $\mathsf{Transform\text{-}VBB\text{-}1}$, except that the signature is computed using true randomness instead of using $\tilde{F}$.

$\mathsf{Transform\text{-}VBB'\text{-}1}_{N,K,\alpha,\mathrm{SK}}$:

Inputs: $b \in \{0,1\}, a \in \{0,1\}^{\ell}, \mathsf{Verify}' \in \{0,1\}^{\ell_{\mathrm{ver}}}$, $\mathrm{VK}' \in \{0,1\}^{\ell_{\mathrm{vk}}}$, $m' \in \{0,1\}^{\ell_{\mathrm{msg}}}$, $\sigma' \in \{0,1\}^{\ell_{\mathrm{sig}}}$.

Constants : RSA modulus $N \in \mathbb{N}$, $K \in \mathcal{K}$, $\alpha \in \{0,1\}^{2\ell}$, $\mathrm{SK} \in \mathcal{SK}$.

if $b = 0$ **then**
  if $\mathrm{PRG}(a) \neq \alpha$ **then**
    Output $\perp$.
  else
    Choose $r \in \{0,1\}^{\ell_{\mathrm{rnd}}}$ and output $(\mathrm{VK}, \mathcal{S}.\mathsf{Sign}(\mathrm{SK}, m'; r))$.
  end if
else if $\mathsf{Verify}'(\mathrm{VK}', m', \sigma') = 0$ **then**
  Output $\perp$.
else
  Output $F(K, \mathsf{Verify}'||\mathrm{VK}'||m')$.
end if

From the security of $\tilde{F}$, we get the following claim:

Claim 5.7. Let $(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathcal{S}.\mathsf{Sign}(1^{\lambda})$ and $\tilde{K} \leftarrow \tilde{F}.\mathsf{setup}(1^{\lambda})$. Let $v^* = (a, r_N, r_K, r_{\mathsf{Att}})$, $\alpha = \mathrm{PRG}(a)$. N_{v^*}, $K_{v^*}, (m_{v^*}, \mathsf{st}_{v^*})$ are computed using $r_N, r_K, r_{\mathsf{Att}}$ respectively. Let $C_0 = \mathsf{Transform\text{-}VBB\text{-}1}_{N,K,\alpha,\mathrm{SK},\tilde{K}}$ and $C_0' = \mathsf{Transform\text{-}VBB'\text{-}1}_{N,K,\alpha,\mathrm{SK}}$. Assuming $\tilde{F}$ is a secure PRF, for any PPT algorithm S,

$$Pr\left[S^{C_0}\left(1^{|C_0|}\right) = 1\right] - Pr\left[S^{C_0'}\left(1^{|C_0'|}\right) = 1\right] \leq \mathrm{negl}(\lambda).$$

Similarly, we define an algorithm $\mathsf{Transform\text{-}VBB'\text{-}2}$ which is exactly similar to $\mathsf{Transform\text{-}VBB\text{-}2}$, except that the signature is computed using true randomness.

Claim 5.8. Let $(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathcal{S}.\mathsf{Sign}(1^{\lambda})$ and $\tilde{K} \leftarrow \tilde{F}.\mathsf{setup}(1^{\lambda})$. Let $v^* = (a, r_N, r_K, r_{\mathsf{Att}})$, $\alpha = \mathrm{PRG}(a)$. N_{v^*}, $K_{v^*}, (m_{v^*}, \mathsf{st}_{v^*})$ are computed using $r_N, r_K, r_{\mathsf{Att}}$ respectively and $y = \mathcal{S}.\mathsf{Verify}||\mathrm{VK}||m_{v^*}$. Let $C_1 = \mathsf{Transform\text{-}VBB\text{-}2}_{y,N,K,\alpha,\mathrm{SK},\tilde{K}}$ and $C_1' = \mathsf{Transform\text{-}VBB'\text{-}2}_{y,N,K,\alpha,\mathrm{SK}}$. Assuming $\tilde{F}$ is a secure PRF, for any PPT algorithm S,

$$\left|\Pr\left[S^{C_1}\left(1^{|C_1|}\right) = 1\right] - \Pr\left[S^{C_1'}\left(1^{|C_1'|}\right) = 1\right]\right| \leq \mathrm{negl}(\lambda).$$

Therefore, if we can show that no PPT algorithm can distinguish between C_0' and C_1' given only oracle access, then together with Equation 14 and Claims 5.7, 5.8, this leads to a contradiction. Note that if any algorithm S has only oracle access to C_0' and C_1', then in order to distinguish between the two, S must send a query $(1, a', \mathcal{S}.\mathsf{Verify}, \mathrm{VK}, m, \sigma)$ such that $\mathcal{S}.\mathsf{Verify}(\mathrm{VK}, m, \sigma) = 1$. This breaks the security of signature scheme $\mathcal{S}$.

Claim 5.9. Let $(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathcal{S}.\mathsf{Sign}(1^{\lambda})$ and $\tilde{K} \leftarrow \tilde{F}.\mathsf{setup}(1^{\lambda})$. Let $v^* = (a, r_N, r_K, r_{\mathsf{Att}})$, $\alpha = \mathrm{PRG}(a)$. N_{v^*}, $K_{v^*}, (m_{v^*}, \mathsf{st}_{v^*})$ are computed using $r_N, r_K, r_{\mathsf{Att}}$ respectively and $y = \mathcal{S}.\mathsf{Verify}||\mathrm{VK}||m_{v^*}$. Let $C_0' = \mathsf{Transform\text{-}VBB'\text{-}1}_{N,K,\alpha,\mathrm{SK}}$ and $C_1' = \mathsf{Transform\text{-}VBB'\text{-}2}_{y,N,K,\alpha,\mathrm{SK}}$. Assuming $\mathcal{S}$ is a secure signature scheme, for any PPT algorithm S,

$$\left|\Pr\left[S^{C_0'}\left(1^{|C_0'|}\right) = 1\right] - \Pr\left[S^{C_1'}\left(1^{|C_1'|}\right) = 1\right]\right| \leq \mathrm{negl}(\lambda).$$

■

6 Universal Aggregation of Arbitrary Signatures from $i\mathcal{O}$ in the Random Oracle Model

In this section, we describe our n-bounded universal signature aggregator $(\ell_{\text{ver}}, \ell_{\text{vk}}, \ell_{\text{msg}}, \ell_{\text{sig}})$-$\mathsf{UniversalSigAgg}$. By n-bounded, we mean that at most n signatures can be aggregated.

We will use a secure $(\ell_{\text{ckt}}, \ell_{\text{inp}}, \ell_{\text{out}})$ universal parameters scheme $\mathcal{U} = (\mathsf{UniversalGen}, \mathsf{InduceGen})$ (where the parameters $\ell_{\text{ckt}}, \ell_{\text{inp}}$ and ℓ_{out} will be specified later), an additively homomorphic encryption scheme $(\mathsf{HE.setup}, \mathsf{HE.enc}, \mathsf{HE.dec}, \mathsf{HE.add})$ with message space $\mathbb{F}_p$ for some prime $p > 2^{\ell_{\text{sig}}}$ and ciphertext space $\mathcal{C}_{\mathsf{HE}}$. We will assume each $\text{ct} \in \mathcal{C}_{\mathsf{HE}}$ can be represented using ℓ_{ct} bits. Finally, we will also use a one-way function $f : \{0,1\}^{\ell} \to \{0,1\}^{2\ell}$ and a secure indistinguishability obfuscator $i\mathcal{O}$.

Our construction consists of three algorithms $\mathsf{UniversalSetup}$, $\mathsf{UniversalAgg}$ and $\mathsf{UniversalVerify}$ described as follows.

$\mathsf{UniversalSetup}(1^\lambda, 1^n)$ Let $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{HE.setup}(1^\lambda)$. It computes n ciphertexts $\text{ct}_i \leftarrow \mathsf{HE.enc}(\mathsf{pk}, 0)$ and $U \leftarrow \mathsf{UniversalGen}(1^\lambda)$. It sets the public parameters to be $\text{PP} = (\mathsf{pk}, \text{ct}_1, \ldots, \text{ct}_n, U)$. Let us assume PP can be represented using ℓ_{pp} bits.

$\mathsf{UniversalAgg}(\text{PP} = (\mathsf{pk}, \text{ct}_1, \ldots, \text{ct}_n, U), \{\mathsf{Verify}_i, \text{VK}_i, m_i, \sigma_i\}_{i=1}^n)$ We will view each signature σ_i as an integer in $[0, 2^{\ell_{\text{sig}}} - 1]$.

The universal aggregator first checks if all n tuples are distinct. If not, it outputs $\perp$. Else, it computes $t = \sigma_1 \cdot \text{ct}_1 + \ldots + \sigma_n \cdot \text{ct}_n$.

Let $\mathsf{AggSetup}$ be the (randomized) algorithm (defined below) that takes as input security parameter λ, and outputs a program C_{agg} and $\tilde{s} \in \{0,1\}^{2\ell}$. It uses ℓ_{inp} bits of randomness, and its output has length ℓ_{out}. Let $\mathcal{C}\text{-}\mathsf{AggSetup}_{t,\text{PP},\{\mathsf{Verify}_i,\text{VK}_i,m_i\}_i} \in \{0,1\}^{\ell_{\text{ckt}}}$ be a string corresponding to canonical description of $\mathsf{AggSetup}_{t,\text{PP},\{\mathsf{Verify}_i,\text{VK}_i,m_i\}_i}$. We will assume that given $\mathcal{C}\text{-}\mathsf{AggSetup}_{t,\text{PP},\{\mathsf{Verify}_i,\text{VK}_i,m_i\}_i}$, one can efficiently extract the hardwired constants t, PP and the n tuples $\{\mathsf{Verify}_i, \text{VK}_i, m_i\}_i$.

The aggregator algorithm first computes $(C_{\mathsf{agg}}, \tilde{s}) = \mathsf{InduceGen}(\mathcal{C}\text{-}\mathsf{AggSetup}_{t,\text{PP},\{\mathsf{Verify}_i,\text{VK}_i,m_i\}_i})$. Next, it computes $s = C_{\mathsf{agg}}(\sigma_1, \ldots, \sigma_n)$ and outputs $\sigma_{\text{agg}} = (t, s)$.

$\mathsf{AggSetup}_{t,\mathrm{PP},\{\mathsf{Verify}_i,\mathrm{VK}_i,m_i\}_i}$:

Inputs: Security parameter 1^λ, $r \in \{0,1\}^{\ell_{\mathrm{inp}}}$.

Constants: $t \in \mathcal{C}_{\mathsf{HE}}$, $\mathrm{PP} = (\mathsf{pk}, \mathrm{ct}_1, \ldots, \mathrm{ct}_n, U) \in \{0,1\}^{\ell_{\mathrm{pp}}}$, $\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}_i \in (\{0,1\}^{\ell_{\mathrm{ver}}} \times \{0,1\}^{\ell_{\mathrm{vk}}} \times \{0,1\}^{\ell_{\mathrm{msg}}})^n$.

1. Choose $s \leftarrow \{0,1\}^\ell$ using r.
2. Compute $C_{\mathsf{agg}} \leftarrow i\mathcal{O}(\mathsf{AggSign}_{s,t,\mathrm{PP},\{\mathsf{Verify}_i,\mathrm{VK}_i,m_i\}_i})$, where $\mathsf{AggSign}$ is the circuit described below.

   $\mathsf{AggSign}_{s,t,\mathrm{PP},\{\mathsf{Verify}_i,\mathrm{VK}_i,m_i\}_i}$:

   Inputs: $\sigma_1, \ldots, \sigma_n$, where $\sigma_i \in \{0,1\}^{\ell_{\mathrm{sig}}}$.

   Constants: $s \in \{0,1\}^\ell$, $t \in \mathcal{C}_{\mathsf{HE}}$, $\mathrm{PP} = (\mathsf{pk}, \mathrm{ct}_1, \ldots, \mathrm{ct}_n, U)$, $\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}_i$.

   if $\exists i$ such that $\mathsf{Verify}_i(\mathrm{VK}_i, m_i, \sigma_i) = 0$ **then**
   Output $\perp$.
   end if
   if $t \neq \sigma_1 \cdot \mathrm{ct}_1 + \ldots + \sigma_n \cdot \mathrm{ct}_n$ **then**
   Output $\perp$.
   end if
   Output s.

3. Compute $\tilde{s} = f(s)$.
4. Output $(C_{\mathsf{agg}}, \tilde{s})$.

$\mathsf{UniversalVerify}(\mathrm{PP} = (\mathsf{pk}, \mathrm{ct}_1, \ldots, \mathrm{ct}_n, U), \{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}_{i=1}^n, \sigma_{\mathrm{agg}} = (t, s'))$ The verification algorithm first checks if all n tuples are distinct. If not, it outputs 0. Else, let $\mathcal{C}$-$\mathsf{AggSetup}$ be the canonical description of $\mathsf{AggSetup}$ as defined above. It computes $(C_{\mathsf{agg}}, \tilde{s}) = \mathsf{InduceGen}(U, \mathcal{C}\text{-}\mathsf{AggSetup}_{t,\mathrm{PP},\{\mathsf{Verify}_i,\mathrm{VK}_i,m_i\}_i})$. If $\tilde{s} = f(s')$, output 1, else output 0.

Correctness follows directly from the observation that $\mathsf{InduceGen}$ is a deterministic algorithm.

6.1 Proof of Security

Theorem 6.1. Assuming $i\mathcal{O}$ is a secure indistinguishability obfuscator, $(\mathsf{UniversalGen}, \mathsf{InduceGen})$ is a secure universal parameters scheme in the random oracle model, $\mathcal{HE}$ is a secure additively homomorphic encryption scheme and f is a secure one-way function, for all $(\ell_{\mathrm{ver}}, \ell_{\mathrm{vk}}, \ell_{\mathrm{msg}}, \ell_{\mathrm{sig}})$-length qualified secure signature schemes $\mathcal{S}$, the bounded universal signature aggregator described in Section 6 is adaptively secure in the random oracle model with respect to $\mathcal{S}$.

We will first describe a sequence of intermediate experiments $\mathsf{Game}\ 0, \ldots, \mathsf{Game}\ 5$, where $\mathsf{Game}\ 0$ is the adaptive security game in random oracle model. From $\mathsf{Game}\ 3$ onwards, the challenger starts simulating the universal parameters and the responses to random oracle queries. In order to do so, the challenger implements a parameter oracle O, and the simulation algorithms are allowed to make random oracle queries to O. Let us assume the simulator algorithms $\mathsf{SimUGen}$ and SimRO makes at most q_{par} calls to the Parameters Oracle.

6.1.1 Sequence of Games

Game 0: In this game, the challenger first sends PP, VK to the adversary Att. Att then makes polynomially many signature and random oracle queries. Finally, Att outputs forgery σ_{agg} and n tuples $\{\mathsf{Verify}_i, \text{VK}_i, m_i\}_i$.

1. Choose $(\text{SK}, \text{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{HE.setup}(1^\lambda)$ and $U \leftarrow \mathsf{UniversalGen}(1^\lambda)$.
   Compute $\text{ct}_i \leftarrow \mathsf{HE.enc}(\mathsf{pk}, 0)$ for all $i \in [n]$ and set $\text{PP} = (\mathsf{pk}, \text{ct}_1, \ldots, \text{ct}_n, U)$.
   Send PP, VK to Att.
2. For each signature query x_i, compute $\sigma_i = \mathcal{S}.\mathsf{Sign}(\text{SK}, x_i)$ and send σ_i to Att.
3. For each random oracle query y_i, check if y_i has already been queried.
   If yes, let (y_i, α_i) be the tuple corresponding to y_i. Send α_i to Att.
   If not, choose $\alpha_i \leftarrow \{0,1\}^{\ell_{\text{RO}}}$, send α_i to Att and add (y_i, α_i) to table.
4. Finally, Att sends a forgery $\sigma_{\text{agg}} = (t^*, s^*)$ and n tuples $\{\mathsf{Verify}_i, \text{VK}_i, m_i\}_i$. Att wins if
   (a) $\exists i^*$ such that $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}$ and $\text{VK}_{i^*} = \text{VK}$,
   (b) m_{i^*} was not queried during the signing phase,
   (c) $f(s^*) = \tilde{s}$ and $\mathsf{InduceGen}(U, \mathcal{C}\text{-}\mathsf{AggSetup}_{t^*, \text{PP}, \{\mathsf{Verify}_i, \text{VK}_i, m_i\}_i}) = (C, \tilde{s})$.

Game 1: This game is exactly similar to the previous one, except that the challenger guesses a position $i^* \in [n]$, and the attacker wins only if the forgery verifies, and the i^*th tuple corresponds to $\mathcal{S}.\mathsf{Verify}, \text{VK}$.

1. Choose $i^* \leftarrow [n]$.
   Choose $(\text{SK}, \text{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{HE.setup}(1^\lambda)$ and $U \leftarrow \mathsf{UniversalGen}(1^\lambda)$.
   Compute $\text{ct}_i \leftarrow \mathsf{HE.enc}(\mathsf{pk}, 0)$ and set $\text{PP} = (\mathsf{pk}, \text{ct}_1, \ldots, \text{ct}_n, U)$.
   Send PP, VK to Att.
2. For each signature query x_i, compute $\sigma_i = \mathcal{S}.\mathsf{Sign}(\text{SK}, x_i)$ and send σ_i to Att.
3. For each random oracle query y_i, check if y_i has already been queried.
   If yes, let (y_i, α_i) be the tuple corresponding to y_i. Send α_i to Att.
   If not, choose $\alpha_i \leftarrow \{0,1\}^{\ell_{\text{RO}}}$, send α_i to Att and add (y_i, α_i) to table.
4. Finally, Att sends a forgery $\sigma_{\text{agg}} = (t^*, s^*)$ and n tuples $\{\mathsf{Verify}_i, \text{VK}_i, m_i\}_i$. Att wins if
   (a) $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}$ and $\text{VK}_{i^*} = \text{VK}$,
   (b) m_{i^*} was not queried during the signing phase,
   (c) $f(s^*) = \tilde{s}$ and $\mathsf{InduceGen}(U, \mathcal{C}\text{-}\mathsf{AggSetup}_{t^*, \text{PP}, \{\mathsf{Verify}_i, \text{VK}_i, m_i\}_i}) = (C, \tilde{s})$.

Game 2: In this game, the challenger modifies the public parameters PP. Instead of outputting n encryptions of 0, the challenger outputs an encryption of 1 at position i^*.

1. Choose $i^* \leftarrow [n]$.
   Choose $(\text{SK}, \text{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{HE.setup}(1^\lambda)$ and $U \leftarrow \mathsf{UniversalGen}(1^\lambda)$.
   Compute $\text{ct}_i \leftarrow \mathsf{HE.enc}(\mathsf{pk}, 0)$ for all $i \in [n], i \neq i^*$. Let $\text{ct}_{i^*} \leftarrow \mathsf{HE.enc}(\mathsf{pk}, 1)$.
   Set $\text{PP} = (\mathsf{pk}, \text{ct}_1, \ldots, \text{ct}_n, U)$.
   Send PP, VK to Att.
2. For each signature query x_i, compute $\sigma_i = \mathcal{S}.\mathsf{Sign}(\text{SK}, x_i)$ and send σ_i to Att.
3. For each random oracle query y_i, check if y_i has already been queried.
   If yes, let (y_i, α_i) be the tuple corresponding to y_i. Send α_i to Att.
   If not, choose $\alpha_i \leftarrow \{0,1\}^{\ell_{\text{RO}}}$, send α_i to Att and add (y_i, α_i) to table.
4. Finally, Att sends a forgery $\sigma_{\text{agg}} = (t^*, s^*)$ and n tuples $\{\mathsf{Verify}_i, \text{VK}_i, m_i\}_i$. Att wins if
   (a) $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}$ and $\text{VK}_{i^*} = \text{VK}$,
   (b) m_{i^*} was not queried during the signing phase,
   (c) $f(s^*) = \tilde{s}$ and $\mathsf{InduceGen}(U, \mathcal{C}\text{-}\mathsf{AggSetup}_{t^*, \text{PP}, \{\mathsf{Verify}_i, \text{VK}_i, m_i\}_i}) = (C, \tilde{s})$.

Game 3 In this game, the challenger 'simulates' both the universal parameters U and the responses to random oracle queries. Let $\mathsf{SimUGen}$ and SimRO be the simulation algorithms corresponding to the universal parameters scheme $(\mathsf{UniversalGen}, \mathsf{InduceGen})$. The challenger also implements the Parameters Oracle O. O takes as input a circuit $d \in \mathcal{C}[\ell_{\text{ckt}}, \ell_{\text{inp}}, \ell_{\text{out}}]$. If d has already been queried, O returns the same response. Else, it chooses $r \leftarrow \{0,1\}^{\ell_{\text{inp}}}$, outputs $d(r)$, and adds $(d, d(r))$ to its table T. Though the parameters oracle O is described in the Setup Phase, it is used in all the later phases as well.

1. Choose $i^* \leftarrow [n]$.
   Choose $(\text{SK}, \text{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{HE.setup}(1^\lambda)$.
   Compute $U \leftarrow \mathsf{SimUGen}(1^\lambda)$.
   Compute $\text{ct}_i \leftarrow \mathsf{HE.enc}(\mathsf{pk}, 0)$ for all $i \in [n], i \neq i^*$. Let $\text{ct}_{i^*} \leftarrow \mathsf{HE.enc}(\mathsf{pk}, 1)$.
   Set $\text{PP} = (\mathsf{pk}, \text{ct}_1, \ldots, \text{ct}_n, U)$.
   Implement the Parameters Oracle O as follows.
   - Maintain a table T. Initially, T is empty.
   - For the i^{th} query $d \in \mathcal{C}[\ell_{\text{ckt}}, \ell_{\text{inp}}, \ell_{\text{out}}]$, check if T contains an entry corresponding to d.
   - If T contains an entry of the form (d, δ), output δ.
   - Else choose $r \leftarrow \{0,1\}^{\ell_{\text{inp}}}$ and output $d(r)$. Add $(d, d(r))$ to T.

   Send PP, VK to Att.
2. For each signature query x_i, compute $\sigma_i = \mathcal{S}.\mathsf{Sign}(\text{SK}, x_i)$ and send σ_i to Att.
3. For each random oracle query y_i, output $\mathsf{SimRO}(y_i)$ [22].
4. Finally, Att sends a forgery σ_{agg} and n tuples $\{\mathsf{Verify}_i, \text{VK}_i, m_i\}_i$.
   Let $\mathsf{O\text{-}Queries}_i$ denote the set of first i queries to O. Att wins if
   (a) $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}$ and $\text{VK}_{i^*} = \text{VK}$,
   (b) m_{i^*} was not queried during the signing phase,
   (c) $f(s^*) = \tilde{s}$ and $O(\mathcal{C}\text{-}\mathsf{AggSetup}_{t^*, \text{PP}, \{\mathsf{Verify}_i, \text{VK}_i, m_i\}_i}) = (C, \tilde{s})$.

Recall from Section 6 that $\mathcal{C}\text{-}\mathsf{AggSetup}_{t, \text{PP}, \{\mathsf{Verify}_i, \text{VK}_i, m_i\}} \in \{0,1\}^{\ell_{\text{ckt}}}$ allows efficient extraction of t, PP and $(\mathsf{Verify}_i, \text{VK}_i, m_i)$ for all $i \leq n$. Without loss of generality, we can assume that if Att outputs $\sigma_{\text{agg}} = (t^*, s^*)$ as forgery, along with n tuples $\{\mathsf{Verify}_i, \text{VK}_i, m_i\}_i$, then the circuit $\mathcal{C}\text{-}\mathsf{AggSetup}_{t^*, \text{PP}, \{\mathsf{Verify}_i, \text{VK}_i, m_i\}_i}$ was sent as query to the Parameters Oracle O. We will now define games Game 4-j-a and Game 4-j-b for $j \leq q_{\mathsf{par}}$. Let us first define some notations. Given a canonical circuit $\mathcal{C}\text{-}\mathsf{AggSetup}_{t, \text{PP}, \{\mathsf{Verify}_i, \text{VK}_i, m_i\}_i}$, call it (i^*, sk)-rejecting if $\mathsf{Verify}_{i^*}(\text{VK}_{i^*}, m_{i^*}, \mathsf{HE.dec}(\mathsf{sk}, t)) = 0$. Let $\mathsf{Reject\text{-}ckt}$ be a circuit of size same as $\mathsf{AggSign}$ that outputs $\perp$ for all inputs.

Game 4-j-a

1. Choose $i^* \leftarrow [n]$.
   Choose $(\text{SK}, \text{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{HE.setup}(1^\lambda)$.
   Compute $U \leftarrow \mathsf{SimUGen}(1^\lambda)$.
   Compute $\text{ct}_i \leftarrow \mathsf{HE.enc}(\mathsf{pk}, 0)$ for all $i \in [n], i \neq i^*$. Let $\text{ct}_{i^*} \leftarrow \mathsf{HE.enc}(\mathsf{pk}, 1)$.
   Set $\text{PP} = (\mathsf{pk}, \text{ct}_1, \ldots, \text{ct}_n, U)$.
   Implement the Parameters Oracle O as follows.
   - Maintain a table T. Initially, T is empty.
   - For the i^{th} query $d \in \mathcal{C}[\ell_{\text{ckt}}, \ell_{\text{inp}}, \ell_{\text{out}}]$, check if T contains an entry corresponding to d.
   - If T contains an entry of the form (d, δ), output δ.
   - Else if $i \leq j$ and $d = \mathcal{C}\text{-}\mathsf{AggSetup}_{t, \text{PP}, \{\mathsf{Verify}_i, \text{VK}_i, m_i\}}$ is (i^*, sk)-rejecting, output $i\mathcal{O}(\mathsf{Reject\text{-}ckt})$ and $f(s)$ for $s \leftarrow \{0,1\}^\ell$.
   - Else, choose $r \leftarrow \{0,1\}^{\ell_{\text{inp}}}$ and output $d(r)$. Add $(d, d(r))$ to T.

[22] Note that SimRO can make polynomially many queries to O.

Send PP, VK to Att.

2. For each signature query x_i, compute $\sigma_i = \mathcal{S}.\mathsf{Sign}(\mathrm{SK}, x_i)$ and send σ_i to Att.
3. For each random oracle query y_i, output $\mathsf{SimRO}(y_i)$.
4. Finally, Att sends a forgery σ_{agg} and n tuples $\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}_i$. Let $\mathsf{O\text{-}Queries}_i$ denote the set of first i queries to O. Att wins if
   (a) $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}$ and $\mathrm{VK}_{i^*} = \mathrm{VK}$,
   (b) m_{i^*} was not queried during the signing phase,
   (c) ($\mathcal{C}$-$\mathsf{AggSetup}_{t^*,\mathrm{PP},\{\mathsf{Verify}_i,\mathrm{VK}_i,m_i\}}$ is not (i^*,sk)-rejecting) **or** ($\mathcal{C}$-$\mathsf{AggSetup}_{t^*,\mathrm{PP},\{\mathsf{Verify}_i,\mathrm{VK}_i,m_i\}} \notin \mathsf{O\text{-}Queries}_{j-1}$),
   (d) $f(s^*) = \tilde{s}$ and $O(\mathcal{C}\text{-}\mathsf{AggSetup}_{t^*,\mathrm{PP},\{\mathsf{Verify}_i,\mathrm{VK}_i,m_i\}_i}) = (C, \tilde{s})$.

Game 4-j-b

1. Choose $i^* \leftarrow [n]$.
   Choose $(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{HE.setup}(1^\lambda)$.
   Compute $U \leftarrow \mathsf{SimUGen}(1^\lambda)$.
   Compute $\mathrm{ct}_i \leftarrow \mathsf{HE.enc}(\mathsf{pk}, 0)$ for all $i \in [n], i \neq i^*$. Let $\mathrm{ct}_{i^*} \leftarrow \mathsf{HE.enc}(\mathsf{pk}, 1)$.
   Set $\mathrm{PP} = (\mathsf{pk}, \mathrm{ct}_1, \ldots, \mathrm{ct}_n, U)$.
   Implement the Parameters Oracle O as follows.
   - Maintain a table T. Initially, T is empty.
   - For the i^{th} query $d \in \mathcal{C}[\ell_{\mathrm{ckt}}, \ell_{\mathrm{inp}}, \ell_{\mathrm{out}}]$, check if T contains an entry corresponding to d.
   - If T contains an entry of the form (d, δ), output δ.
   - Else if $i \leq j$ and $d = \mathcal{C}\text{-}\mathsf{AggSetup}_{t,\mathrm{PP},\{\mathsf{Verify}_i,\mathrm{VK}_i,m_i\}}$ is (i^*,sk)-rejecting, output $i\mathcal{O}(\mathsf{Reject\text{-}ckt})$ and $f(s)$ for $s \leftarrow \{0,1\}^\ell$.
   - Else, choose $r \leftarrow \{0,1\}^{\ell_{\mathrm{inp}}}$ and output $d(r)$. Add $(d, d(r))$ to T.

   Send PP, VK to Att.
2. For each signature query x_i, compute $\sigma_i = \mathcal{S}.\mathsf{Sign}(\mathrm{SK}, x_i)$ and send σ_i to Att.
3. For each random oracle query y_i, output $\mathsf{SimRO}(y_i)$.
4. Finally, Att sends a forgery $\sigma_{\mathrm{agg}} = (t^*, s^*)$ and n tuples $\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}_i$. Att wins if
   (a) $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}$ and $\mathrm{VK}_{i^*} = \mathrm{VK}$,
   (b) m_{i^*} was not queried during the signing phase,
   (c) ($\mathcal{C}$-$\mathsf{AggSetup}_{t^*,\mathrm{PP},\{\mathsf{Verify}_i,\mathrm{VK}_i,m_i\}}$ is not (i^*,sk)-rejecting) **or** ($\mathcal{C}$-$\mathsf{AggSetup}_{t^*,\mathrm{PP},\{\mathsf{Verify}_i,\mathrm{VK}_i,m_i\}} \notin \mathsf{O\text{-}Queries}_j$),
   (d) $f(s^*) = \tilde{s}$ and $O(\mathcal{C}\text{-}\mathsf{AggSetup}_{t^*,\mathrm{PP},\{\mathsf{Verify}_i,\mathrm{VK}_i,m_i\}_i}) = (C, \tilde{s})$.

Game 5 This game is exactly similar to Game 4-q_{par}-b.

1. Choose $i^* \leftarrow [n]$.
   Choose $(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{HE.setup}(1^\lambda)$.
   Compute $U \leftarrow \mathsf{SimUGen}(1^\lambda)$.
   Compute $\mathrm{ct}_i \leftarrow \mathsf{HE.enc}(\mathsf{pk}, 0)$ for all $i \in [n], i \neq i^*$. Let $\mathrm{ct}_{i^*} \leftarrow \mathsf{HE.enc}(\mathsf{pk}, 1)$.
   Set $\mathrm{PP} = (\mathsf{pk}, \mathrm{ct}_1, \ldots, \mathrm{ct}_n, U)$.
   Implement the Parameters Oracle O as follows.
   - Maintain a table T. Initially, T is empty.
   - For the i^{th} query $d \in \mathcal{C}[\ell_{\mathrm{ckt}}, \ell_{\mathrm{inp}}, \ell_{\mathrm{out}}]$, check if T contains an entry corresponding to d.
   - If T contains an entry of the form (d, δ), output δ.
   - Else if $d = \mathcal{C}\text{-}\mathsf{AggSetup}_{t,\mathrm{PP},\{\mathsf{Verify}_i,\mathrm{VK}_i,m_i\}}$ is (i^*,sk)-rejecting, output $i\mathcal{O}(\mathsf{Reject\text{-}ckt})$ and $f(s)$ for $s \leftarrow \{0,1\}^\ell$.
   - Else, choose $r \leftarrow \{0,1\}^{\ell_{\mathrm{inp}}}$ and output $d(r)$. Add $(d, d(r))$ to T.

   Send PP, VK to Att.
2. For each signature query x_i, compute $\sigma_i = \mathcal{S}.\mathsf{Sign}(\mathrm{SK}, x_i)$ and send σ_i to Att.

3. For each random oracle query y_i, output $\mathsf{SimRO}(y_i)$.
4. Finally, Att sends a forgery $\sigma_{\text{agg}} = (t^*, s^*)$ and n tuples $\{\mathsf{Verify}_i, \text{VK}_i, m_i\}_i$. Att wins if
   (a) $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}$ and $\text{VK}_{i^*} = \text{VK}$,
   (b) m_{i^*} was not queried during the signing phase,
   (c) $\mathcal{S}.\mathsf{Verify}(\text{VK}, m_{i^*}, \mathsf{HE.dec}(\mathsf{sk}, t^*)) = 1$,
   (d) $f(s^*) = \tilde{s}$ and $O(\mathcal{C}\text{-}\mathsf{AggSetup}_{t^*,\text{PP},\{\mathsf{Verify}_i,\text{VK}_i,m_i\}_i}) = (C, \tilde{s})$.

6.1.2 Analysis

Let $\mathsf{Adv}^j_{\mathsf{Att}}$ denote the advantage of Att in Game j.

Claim 6.1. For any adversary Att,

$$\mathsf{Adv}^1_{\mathsf{Att}} = \mathsf{Adv}^0_{\mathsf{Att}}/n.$$

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 $1/n$. ∎

Claim 6.2. Assuming $(\mathsf{HE.setup}, \mathsf{HE.enc}, \mathsf{HE.dec})$ is a secure additively homomorphic encryption scheme, for any PPT adversary Att,

$$\mathsf{Adv}^1_{\mathsf{Att}} - \mathsf{Adv}^2_{\mathsf{Att}} \leq \text{negl}(\lambda).$$

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

$\mathcal{B}$ receives the public key pk. It sends $0, 1$ as challenge messages to the HE challenger, and receives ct in response. It chooses $i^* \leftarrow [n]$, (SK, VK), computes $n-1$ encryptions of 0, that is, $\text{ct}_i \leftarrow \mathsf{HE.enc}(\mathsf{pk}, 0)$ for $i \neq i^*$. It sets $\text{ct}_{i^*} = \text{ct}$. It computes $U \leftarrow \mathsf{UniversalGen}(1^\lambda)$ and sends $\text{PP} = (\mathsf{pk}, \text{ct}_1, \ldots, \text{ct}_n, U)$ and VK to Att.

Att then asks for signature/random oracle queries, which $\mathcal{B}$ can simulate perfectly. Finally, Att outputs a forgery σ_{agg} and n tuples $\{\mathsf{Verify}_i, \text{VK}_i, m_i\}$. If Att wins as per the winning conditions (which are the same in both Game 1 and Game 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. ∎

Claim 6.3. Assuming $\mathcal{U} = (\mathsf{UniversalGen}, \mathsf{InduceGen})$ is a secure $(\ell_{\text{ckt}}, \ell_{\text{inp}}, \ell_{\text{out}})$ universal parameters scheme, for any PPT adversary Att,

$$\mathsf{Adv}^2_{\mathsf{Att}} - \mathsf{Adv}^3_{\mathsf{Att}} \leq \text{negl}(\lambda).$$

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

$\mathcal{A}$ interacts with Att and participates in either the Real or Ideal game. It receives the universal parameters U. It chooses $(\text{SK}, \text{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{HE.setup}(1^\lambda)$, computes ciphertexts $\text{ct}_1, \ldots, \text{ct}_n$ and sets $\text{PP} = (\mathsf{pk}, \text{ct}_1, \ldots, \text{ct}_n, U)$. It sends PP, VK to Att.

For the signature queries, $\mathcal{A}$ computes the signatures using SK. For any random oracle query x, it forwards x to the challenger in the $\mathsf{Real}/\mathsf{Ideal}$ game, and receives either $\mathsf{RO}(x)$ or $\mathsf{SimRO}(x)$. Finally, it receives a forgery $\sigma_{\text{agg}} = (t^*, s^*)$ and n tuples $\{\mathsf{Verify}_i, \text{VK}_i, m_i\}_i$. Note that since there is no Honest Parameter Violation, $\mathsf{InduceGen}(U, \mathcal{C}\text{-}\mathsf{AggSetup}_{t^*,\text{PP},\{\mathsf{Verify}_i,\text{VK}_i,m_i\}_i}) = O(\mathcal{C}\text{-}\mathsf{AggSetup}_{t^*,\text{PP},\{\mathsf{Verify}_i,\text{VK}_i,m_i\}_i})$. Therefore, Game 2 corresponds to $\mathsf{Real}^{\mathcal{A}}(1^\lambda)$ experiment, while Game 3 corresponds to $\mathsf{Ideal}^{\mathcal{A}}_{\mathsf{SimUGen},\mathsf{SimRO}}(1^\lambda)$. Hence, $\Pr[\mathsf{Real}^{\mathcal{A}}(1^\lambda) = 1] - \Pr[\mathsf{Ideal}^{\mathcal{A}}_{\mathsf{SimUGen},\mathsf{SimRO}}(1^\lambda) = 1] = \mathsf{Adv}^2_{\mathsf{Att}} - \mathsf{Adv}^3_{\mathsf{Att}}$.

∎

Claim 6.4. Assuming $i\mathcal{O}$ is a secure indistinguishability obfuscator, for any $j \leq q_{\mathsf{par}}$, for any PPT adversary Att,

$$\mathsf{Adv}_{\mathsf{Att}}^{4\text{-}(j-1)\text{-}b} - \mathsf{Adv}_{\mathsf{Att}}^{4\text{-}j\text{-}a} \leq \mathrm{negl}(\lambda).$$

Proof. The only difference between Game 4-$(j-1)$-b and Game 4-j-a is with respect to the j^{th} query to the parameters oracle O. If the j^{th} query is not of the form $\mathcal{C}\text{-}\mathsf{AggSetup}_{t,\mathrm{PP},\{\mathsf{Verify}_i,\mathrm{VK}_i,m_i\}}$, or if it is not (i^*,sk)-rejecting, then both games are identical. Therefore, let us consider the case where the j^{th} query to O is $\mathcal{C}\text{-}\mathsf{AggSetup}_{t,\mathrm{PP},\{\mathsf{Verify}_i,\mathrm{VK}_i,m_i\}}$ for some $t, \{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}$, and it is (i^*,sk)-rejecting. In Game 4-$(j-1)$-b, O outputs $(i\mathcal{O}(\mathsf{AggSign}_{t,s,\mathrm{PP},\{\mathsf{Verify}_i,\mathrm{VK}_i,m_i\}}), f(s))$ while in Game 4-j-a, it outputs $(i\mathcal{O}(\mathsf{Reject\text{-}ckt}), f(s))$. Hence, if we can show that $\mathcal{C}\text{-}\mathsf{AggSetup}_{t,\mathrm{PP},\{\mathsf{Verify}_i,\mathrm{VK}_i,m_i\}}$ and $\mathsf{Reject\text{-}ckt}$ are functionally identical for (i^*,sk)-rejecting circuit, then we can use the security of $i\mathcal{O}$ to prove our claim.

Consider any input $\sigma_1, \ldots, \sigma_n$ to $\mathcal{C}\text{-}\mathsf{AggSetup}_{t,\mathrm{PP},\{\mathsf{Verify}_i,\mathrm{VK}_i,m_i\}}$. If $\exists i$ such that $\mathsf{Verify}_i(\mathrm{VK}_i, m_i, \sigma_i) = 0$, then it outputs $\perp$. If $t \neq \sigma_1 \cdot \mathrm{ct}_1 + \ldots + \sigma_n \cdot \mathrm{ct}_n$, then it output $\perp$. However, note that if $t = \sigma_1 \cdot \mathrm{ct}_1 + \ldots + \sigma_n \cdot \mathrm{ct}_n$, then t is an encryption of σ_{i^*}. Since $\mathcal{C}\text{-}\mathsf{AggSetup}_{t,\mathrm{PP},\{\mathsf{Verify}_i,\mathrm{VK}_i,m_i\}}$ is (i^*,sk)-rejecting, $\mathsf{Verify}_{i^*}(\mathrm{VK}_{i^*}, m_{i^*}, \mathsf{HE.dec}(\mathsf{sk}, t)) = 0$. Therefore, this circuit outputs $\perp$ on all inputs, and is functionally identical to $\mathsf{Reject\text{-}ckt}$. ∎

Claim 6.5. Assuming f is a secure one way function, for any $j \leq q_{\mathsf{par}}$, for any PPT adversary Att,

$$\mathsf{Adv}_{\mathsf{Att}}^{4\text{-}j\text{-}a} - \mathsf{Adv}_{\mathsf{Att}}^{4\text{-}j\text{-}b} \leq \mathrm{negl}(\lambda).$$

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

Note that the only way an adversary can distinguish between Game 4-j-a and Game 4-j-b is by submitting a forgery $\sigma_{\mathrm{agg}} = (t^*, s^*)$ and n tuples $\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}$ such that $\mathcal{C}\text{-}\mathsf{AggSetup}_{t,\mathrm{PP},\{\mathsf{Verify}_i,\mathrm{VK}_i,m_i\}}$ is (i^*,sk)-rejecting and $\mathcal{C}\text{-}\mathsf{AggSetup}_{t^*,\mathrm{PP},\{\mathsf{Verify}_i,\mathrm{VK}_i,m_i\}}$ was sent as j^{th} query to O.

$\mathcal{B}$ receives as input $\tilde{s}$. It chooses $i^* \leftarrow [n]$, chooses $(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$ and sets PP as in Game 4-j-a and Game 4-j-b. It sends PP, VK to Att. For each signature query x_i, it sends $\sigma_i \leftarrow \mathcal{S}.\mathsf{Sign}(\mathrm{SK}, x_i)$ to Att. For each random oracle query y_i, $\mathcal{B}$ uses SimRO. SimRO, in turn, makes a number of queries to the Parameters Oracle O. If the j^{th} query to O is $\mathcal{C}\text{-}\mathsf{AggSetup}_{t,\mathrm{PP},\{\mathsf{Verify}_i,\mathrm{VK}_i,m_i\}}$ and is (i^*,sk)-rejecting, send $(i\mathcal{O}(\mathsf{Reject\text{-}ckt}), \tilde{s})$ as response. All other oracle queries are computed as before. Finally, if Att wins, then $\mathcal{B}$ can use the forgery $\sigma_{\mathrm{agg}} = (t^*, s^*)$ and send s^* as inverse of $\tilde{s}$.

∎

Claim 6.6. Assuming $\mathcal{S}$ is a $(\ell_{\mathrm{ver}}, \ell_{\mathrm{vk}}, \ell_{\mathrm{msg}}, \ell_{\mathrm{sig}})$-length qualified secure signature scheme, for any adversary Att,

$$\mathsf{Adv}_{\mathsf{Att}}^{5} \leq \mathrm{negl}(\lambda).$$

Proof. Suppose $\mathsf{Adv}_{\mathsf{Att}}^{5} = \epsilon$. We will construct a PPT algorithm $\mathcal{B}$ that breaks the security of $\mathcal{S}$ with advantage ϵ.

$\mathcal{B}$ receives VK from the challenger. It chooses $i^* \leftarrow [n]$, PP as in Game 5 and sends PP.VK to Att. For each signature query x_i sent by Att, $\mathcal{B}$ sends it to the challenger, receives σ_i, which it forwards to Att. It simulates the oracle queries using SimRO, as in Game 5. Finally, Att outputs a forgery $\sigma_{\mathrm{agg}} = (t^*, s^*)$ and n tuples $\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}_i$. Att wins if $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}_{i^*}$, $\mathrm{VK}_{i^*} = \mathrm{VK}$, m_{i^*} was not queried during the signature phase and $\mathcal{S}.\mathsf{Verify}(\mathrm{VK}, m_{i^*}, \mathsf{HE.dec}(\mathsf{sk}, t^*)) = 1$. It sends $(m_{i^*}, \mathsf{HE.dec}(\mathsf{sk}, t^*))$ as forgery. Note that $\mathcal{B}$ wins the signature game if Att wins Game 5. This concludes our proof.

∎

Using the above claims, it follows that any PPT adversary has negligible advantage in Game 0, assuming the universal parameters scheme is secure (in the random oracle model), $\mathcal{HE}$ is a secure additively homomorphic encryption scheme and f is a secure one-way function. Therefore, the universal signature aggregator described in Section 6 is adaptively secure with respect to all secure signature schemes in the random oracle model.

7 Universal Aggregation of Arbitrary Signatures from $i\mathcal{O}$ in the Standard Model

In this section, we will describe a construction for an n-bounded universal signature aggregator that can be proven selective secure with respect to all secure length-qualified signature schemes using complexity leveraging. We will use an additively HE scheme $\mathcal{HE}$ 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_{ct} bits. We will also use an indistinguishability obfuscator $i\mathcal{O}$, a puncturable pseudorandom function F with key space $\mathcal{K}$, input space $\{0,1\}^{\ell_{\mathrm{ver}}+\ell_{\mathrm{vk}}+\ell_{\mathrm{msg}}+\log n+\log p}$ and range $\{0,1\}^{\ell}$ for $\ell > 2\ell_{\mathsf{ct}}$ and an injective one-way function $f : \{0,1\}^{\ell} \to \{0,1\}^{2\ell}$. The universal signature aggregator consists of three algorithms $\mathsf{UniversalSetup}$, $\mathsf{UniversalAgg}$ and $\mathsf{UniversalVerify}$ described below.

$\mathsf{UniversalSetup}(1^\lambda, 1^n)$ The setup algorithm takes λ, n as input, and chooses $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{HE.setup}(1^\lambda)$. It then computes n encryptions of 0, that is, $\mathrm{ct}_i \leftarrow \mathsf{HE.enc}(\mathsf{pk}, 0)$ for $i \in [n]$.

Let $\sigma_i \in F_p$ for $i \in [n]$. Let $C_{\sigma_1,\ldots,\sigma_n}$ be a circuit that takes as input n bits $x_1, \ldots, x_n$ and outputs $\sum \sigma_i x_i \bmod p$. The setup algorithm computes $P_1 = i\mathcal{O}(\mathsf{AggSign}_{K,\mathsf{pk},\mathrm{ct}_1,\ldots,\mathrm{ct}_n})$ and $P_2 = i\mathcal{O}(\mathsf{AggVerify}_K)$, where the programs $\mathsf{AggSign}$[23] and $\mathsf{AggVerify}$[24] are defined below. It outputs $\mathrm{PP} = (P_1, P_2)$.

$\mathsf{AggSign}_{K,\mathsf{pk},\mathrm{ct}_1,\ldots,\mathrm{ct}_n}$

Inputs: $\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i, \sigma_i\}_i$.

Constants: PRF Key $K \in \mathcal{K}$, pk, $(\mathrm{ct}_1, \ldots, \mathrm{ct}_n) \in \mathcal{C}_{\mathsf{HE}}^n$.

if $\exists i$ such that $\mathsf{Verify}_i(\mathrm{VK}_i, m_i, \sigma_i) = 0$ **then**
Output $\perp$.
end if
Compute $t = \sigma_1 \cdot \mathrm{ct}_1 + \ldots + \sigma_n \cdot \mathrm{ct}_n$.
Let $s_i = F(K, \mathsf{Verify}_i || \mathrm{VK}_i || m_i || i || t)$.
Output $\sigma_{\mathrm{agg}} = (t, \oplus_i s_i)$.

$\mathsf{AggVerify}_K$

Inputs: $\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}_i, (t^*, s^*) \in \mathcal{C}_{\mathsf{HE}} \times \{0,1\}^{\ell}$

Constants: PRF key K

Compute $s = \oplus_i F(K, \mathsf{Verify}_i || \mathrm{VK}_i || m_i || i || t^*)$.
Output 1 if $s = s^*$, else output 0.

$\mathsf{UniversalAgg}(\mathrm{PP} = (P_1, P_2), \{\mathsf{Verify}_i, \mathrm{VK}_i, m_i, \sigma_i\}_i)$ The aggregator algorithm receives as input the public parameters PP and n tuples $\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i, \sigma_i\}_i$. Without loss of generality, we will assume the n tuples are lexicographically ordered. If the n tuples are not distinct, the algorithm outputs $\perp$. Else, it outputs $P_1(\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i, \sigma_i\}_i)$.

$\mathsf{UniversalVerify}(\mathrm{PP} = (P_1, P_2), \{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}_{i=1}^n, \sigma_{\mathrm{agg}} = (t^*, s^*))$ Assume the n tuples are sorted in lexicographic order. The verification algorithm checks that the n tuples are distinct. If not, it outputs 0. Else, it outputs $P_2(\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}, (t^*, s^*))$.

[23] Padded to be of same size as AggSign-1.
[24] Padded to be of same size as AggVerify-1 and AggVerify-2.

7.1 Proof of Security

Let $\mathcal{S}$ be a secure signature scheme. In order to prove the construction in Section 7 selectively secure with respect to $\mathcal{S}$, we will describe a sequence of intermediate hybrid experiments. Looking ahead, there will be an exponential number of intermediate hybrid experiments, and hence we will be using stronger security for the indistinguishability obfuscator $i\mathcal{O}$, the puncturable PRF F and the one way function f.

Theorem 7.1. Let Att be any PPT adversary, and $\mathcal{S}$ a $(\ell_{\text{ver}}, \ell_{\text{vk}}, \ell_{\text{msg}}, \ell_{\text{sig}})$-length qualified secure signature scheme. Let $\mathsf{Adv}^{\text{sel}}_{\mathsf{Att},\mathcal{S}}$ denote the advantage of Att in the universal signature aggregator selective security game with respect to $\mathcal{S}$. Let $\mathsf{Adv}^{\mathcal{S}}, \mathsf{Adv}^{\mathcal{HE}}, \mathsf{Adv}^{i\mathcal{O}}$, Adv^{F} and Adv^{f} denote the maximum advantage of a PPT adversary against signature scheme $\mathcal{S}$, HE scheme $\mathcal{HE}$, indistinguishability obfuscator $i\mathcal{O}$, selectively secure puncturable PRF F and one way function f respectively. Then,

$$\mathsf{Adv}^{\text{sel}}_{\mathsf{Att},\mathcal{S}} \leq n(\mathsf{Adv}^{\mathcal{HE}} + 2^{\ell_{\text{ct}}}(6\mathsf{Adv}^{i\mathcal{O}} + 2\mathsf{Adv}^{F} + \mathsf{Adv}^{f}) + \mathsf{Adv}^{\mathcal{S}})$$

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

7.1.1 Sequence of Games

Game 0: This corresponds to the selective security game. The challenger receives m^* from Att, chooses $(\text{SK}, \text{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$, the public parameters PP and sends PP, VK to the adversary Att. Att then queries for signatures, which the challenger can compute using SK. Finally, Att outputs forgery σ_{agg} and n tuples $\{\mathsf{Verify}_i, \text{VK}_i, m_i\}$.

1. Att sends message m^*.
2. Choose $(\text{SK}, \text{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{HE.setup}(1^\lambda)$ and $K \leftarrow F.\mathsf{setup}(1^\lambda)$.
   Compute $\text{ct}_i \leftarrow \mathsf{HE.enc}(\mathsf{pk}, 0)$ for all $i \in [n]$ and $P_1 \leftarrow i\mathcal{O}(\mathsf{AggSign}_{K,\mathsf{pk},\text{ct}_1,\ldots,\text{ct}_n})$, $P_2 \leftarrow i\mathcal{O}(\mathsf{AggVerify}_K)$.
   Set $\text{PP} = (P_1, P_2)$. Send PP, VK to Att.
3. For each signature query $x_i \neq m^*$, compute $\sigma_i = \mathcal{S}.\mathsf{Sign}(\text{SK}, x_i)$ and send σ_i to Att.
4. Finally, Att sends a forgery $\sigma_{\text{agg}} = (t^*, s^*)$ and n tuples $\{\mathsf{Verify}_i, \text{VK}_i, m_i\}_i$. Att wins
   (a) $\exists i^*$ such that $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}$, $\text{VK}_{i^*} = \text{VK}$, $m_{i^*} = m^*$,
   (b) $\mathsf{AggVerify}_K(\text{PP}, \{\mathsf{Verify}_i, \text{VK}_i, m_i\}, \sigma_{\text{agg}}) = 1$.

Game 1: In this experiment, the challenger chooses $i^* \leftarrow [n]$, and the adversary wins if $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}$, $\text{VK}_{i^*} = \text{VK}$ and $m_{i^*} = m^*$.

1. Att sends m^*.
2. Choose $i^* \leftarrow [n]$.
   Choose $(\text{SK}, \text{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{HE.setup}(1^\lambda)$ and $K \leftarrow F.\mathsf{setup}(1^\lambda)$.
   Compute $\text{ct}_i \leftarrow \mathsf{HE.enc}(\mathsf{pk}, 0)$ for all $i \in [n]$, $P_1 \leftarrow i\mathcal{O}(\mathsf{AggSign}_{K,\mathsf{pk},\text{ct}_1,\ldots,\text{ct}_n})$, $P_2 \leftarrow i\mathcal{O}(\mathsf{AggVerify}_K)$.
   Set $\text{PP} = (P_1, P_2)$. Send PP, VK to Att.
3. For each signature query $x_i \neq m^*$, compute $\sigma_i = \mathcal{S}.\mathsf{Sign}(\text{SK}, x_i)$ and send σ_i to Att.
4. Finally, Att sends a forgery $\sigma_{\text{agg}} = (t^*, s^*)$ and n tuples $\{\mathsf{Verify}_i, \text{VK}_i, m_i\}_i$. Att wins
   (a) $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}$, $\text{VK}_{i^*} = \text{VK}$, $m_{i^*} = m^*$,
   (b) $\mathsf{AggVerify}_K(\text{PP}, \{\mathsf{Verify}_i, \text{VK}_i, m_i\}, \sigma_{\text{agg}}) = 1$.

Game 2: This game is similar to the previous one, except that ct_{i^*} is an encryption of 1, instead of 0.

1. Att sends m^*.
2. Choose $i^* \leftarrow [n]$.
   Choose $(\text{SK}, \text{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{HE.setup}(1^\lambda)$ and $K \leftarrow F.\mathsf{setup}(1^\lambda)$.
   Compute $\text{ct}_i \leftarrow \mathsf{HE.enc}(\mathsf{pk}, 0)$ for all $i \neq i^*$, $\text{ct}_{i^*} \leftarrow \mathsf{HE.enc}(\mathsf{pk}, 1)$, $P_1 \leftarrow i\mathcal{O}(\mathsf{AggSign}_{K,\mathsf{pk},\text{ct}_1,\ldots,\text{ct}_n})$, $P_2 \leftarrow i\mathcal{O}(\mathsf{AggVerify}_K)$.
   Set $\text{PP} = (P_1, P_2)$. Send PP, VK to Att.

3. For each signature query $x_i \neq m^*$, compute $\sigma_i = \mathcal{S}.\mathsf{Sign}(\mathrm{SK}, x_i)$ and send σ_i to Att.
4. Finally, Att sends a forgery $\sigma_{\mathrm{agg}} = (t^*, s^*)$ and n tuples $\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}_i$. Att wins
   (a) $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}$, $\mathrm{VK}_{i^*} = \mathrm{VK}$, $m_{i^*} = m^*$,
   (b) $\mathsf{AggVerify}_K(\mathrm{PP}, \{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}, \sigma_{\mathrm{agg}}) = 1$.

We will now describe an exponential number of hybrid experiments $\mathsf{Game}\ 3, j$ for $j \leq 2^{\ell_{\mathsf{ct}}}$. Before describing these intermediate hybrids, we will define some notations. Recall $\mathsf{AggVerify}_K$ takes as input tuples of the form $(\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}, (t^*, s^*))$. Call such a tuple (i^*, sk)-rejecting if $\mathsf{Verify}_{i^*}(\mathrm{VK}_{i^*}, m_{i^*}, \mathsf{HE.dec}(\mathsf{sk}, t^*)) = 0$.

$\mathsf{Game}\ 3, j$: In this game, the adversary does not win if the forgery input $(\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}, (t^*, s^*))$ is (i^*, sk)-rejecting and $t^* \leq j$.

1. Att sends m^*.
2. Choose $i^* \leftarrow [n]$.
   Choose $(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{HE.setup}(1^\lambda)$ and $K \leftarrow F.\mathsf{setup}(1^\lambda)$.
   Compute $\mathrm{ct}_i \leftarrow \mathsf{HE.enc}(\mathsf{pk}, 0)$ for all $i \neq i^*$, $\mathrm{ct}_{i^*} \leftarrow \mathsf{HE.enc}(\mathsf{pk}, 1)$, $P_1 \leftarrow i\mathcal{O}(\mathsf{AggSign}_{K,\mathsf{pk},\mathrm{ct}_1,\ldots,\mathrm{ct}_n})$, $P_2 \leftarrow i\mathcal{O}(\mathsf{AggVerify}_K)$.
   Set $\mathrm{PP} = (P_1, P_2)$. Send PP, VK to Att.
3. For each signature query $x_i \neq m^*$, compute $\sigma_i = \mathcal{S}.\mathsf{Sign}(\mathrm{SK}, x_i)$ and send σ_i to Att.
4. Finally, Att sends a forgery $\sigma_{\mathrm{agg}} = (t^*, s^*)$ and n tuples $\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}_i$. Att wins
   (a) $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}$, $\mathrm{VK}_{i^*} = \mathrm{VK}$, $m_{i^*} = m^*$,
   (b) $(\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}, (t^*, s^*))$ is not (i^*, sk)-rejecting or $t^* > j$,
   (c) $\mathsf{AggVerify}_K(\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}, \sigma_{\mathrm{agg}}) = 1$.

$\mathsf{Game}\ 4$: This game is identical to $\mathsf{Game}\ 3, 2^{\ell_{\mathsf{ct}}}$.

1. Att sends m^*.
2. Choose $i^* \leftarrow [n]$.
   Choose $(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{HE.setup}(1^\lambda)$ and $K \leftarrow F.\mathsf{setup}(1^\lambda)$.
   Compute $\mathrm{ct}_i \leftarrow \mathsf{HE.enc}(\mathsf{pk}, 0)$ for all $i \neq i^*$, $\mathrm{ct}_{i^*} \leftarrow \mathsf{HE.enc}(\mathsf{pk}, 1)$, $P_1 \leftarrow i\mathcal{O}(\mathsf{AggSign}_{K,\mathsf{pk},\mathrm{ct}_1,\ldots,\mathrm{ct}_n})$, $P_2 \leftarrow i\mathcal{O}(\mathsf{AggVerify}_K)$.
   Set $\mathrm{PP} = (P_1, P_2)$. Send PP, VK to Att.
3. For each signature query x_i, compute $\sigma_i = \mathcal{S}.\mathsf{Sign}(\mathrm{SK}, x_i)$ and send σ_i to Att.
4. Finally, Att sends a forgery $\sigma_{\mathrm{agg}} = (t^*, s^*)$ and n tuples $\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}_i$. Att wins
   (a) $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}$, $\mathrm{VK}_{i^*} = \mathrm{VK}$, $m_{i^*} = m^*$,
   (b) $(\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}, (t^*, s^*))$ is not (i^*, sk)-rejecting,
   (c) $\mathsf{AggVerify}_K(\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}, \sigma_{\mathrm{agg}}) = 1$.

7.1.2 Analysis

Let $\mathsf{Adv}^j_{\mathsf{Att}}$ denote the advantage of Att in $\mathsf{Game}\ j$.

Claim 7.1. For any adversary Att,

$$\mathsf{Adv}^1_{\mathsf{Att}} = \mathsf{Adv}^0_{\mathsf{Att}}/n.$$

Proof. This follows from the definitions of $\mathsf{Game}\ 0$ and $\mathsf{Game}\ 1$. 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 $1/n$. ∎

Claim 7.2. For any PPT adversary Att,

$$\mathsf{Adv}^1_{\mathsf{Att}} - \mathsf{Adv}^2_{\mathsf{Att}} \leq \mathsf{Adv}_{\mathcal{HE}}(\lambda).$$

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

$\mathcal{B}$ receives the public key pk. It sends $0, 1$ as challenge messages to the HE challenger, and receives ct in response. It chooses $i^* \leftarrow [n]$, $(\mathrm{SK}, \mathrm{VK})$, computes $n-1$ encryptions of 0, that is, $\mathrm{ct}_i \leftarrow \mathsf{HE.enc}(\mathsf{pk}, 0)$ for $i \neq i^*$. It sets $\mathrm{ct}_{i^*} = \mathrm{ct}$. It chooses $K \leftarrow F.\mathsf{setup}(1^\lambda)$, computes $P_1 \leftarrow i\mathcal{O}(\mathsf{AggSign}_{K,\mathsf{pk},\mathrm{ct}_1,\ldots,\mathrm{ct}_n})$, $P_2 \leftarrow i\mathcal{O}(\mathsf{AggVerify}_K)$ and sends $\mathrm{PP} = (P_1, P_2)$ and VK to Att.

Att then asks for signature/random oracle queries, which $\mathcal{B}$ can simulate perfectly. Finally, Att outputs a forgery σ_{agg} and n tuples $\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}$. If Att wins as per the winning conditions (which are the same in both Game 1 and Game 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. ∎

Observation 7.1. For any PPT adversary Att,

$$\mathsf{Adv}^2_{\mathsf{Att}} = \mathsf{Adv}^{3,0}_{\mathsf{Att}}.$$

Claim 7.3. For any $j < 2^{\ell_{\mathrm{ct}}}$,

$$\mathsf{Adv}^{3,j}_{\mathsf{Att}} - \mathsf{Adv}^{3,j+1}_{\mathsf{Att}} \leq 6\mathsf{Adv}^{i\mathcal{O}} + 2\mathsf{Adv}^{F} + \mathsf{Adv}^{f}.$$

Proof. The proof of this claim involves a sequence of intermediate hybrids described below. Note that the only difference between the two hybrids is Step 4b. Both games are identical if $j+1$ is not (i^*, sk)-rejecting. Hence, we will consider the case where $\mathcal{S}.\mathsf{Verify}(\mathrm{VK}, m^*, \mathsf{HE.dec}(\mathsf{sk}, j+1)) = 0$.

Game $3, j, a$ In this game, the challenger uses obfuscations of circuit $\mathsf{AggVerify\text{-}1}$ instead of $\mathsf{AggVerify}$. Instead of checking whether $s^* = \oplus_i s_i$, $\mathsf{AggVerify\text{-}1}$ uses an injective one way function f to check if $f(s \oplus (\oplus_{i \neq i^*} s_i)) = f(s_{i^*})$.

1. Att sends m^*.
2. Choose $i^* \leftarrow [n]$.
   Choose $(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{HE.setup}(1^\lambda)$ and $K \leftarrow F.\mathsf{setup}(1^\lambda)$.
   Compute $\mathrm{ct}_i \leftarrow \mathsf{HE.enc}(\mathsf{pk}, 0)$ for all $i \neq i^*$, $\mathrm{ct}_{i^*} \leftarrow \mathsf{HE.enc}(\mathsf{pk}, 1)$, $P_1 \leftarrow i\mathcal{O}(\mathsf{AggSign}_{K,\mathsf{pk},\mathrm{ct}_1,\ldots,\mathrm{ct}_n})$, $P_2 \leftarrow i\mathcal{O}(\mathsf{AggVerify\text{-}1}_K)$.
   Set $\mathrm{PP} = (P_1, P_2)$. Send PP, VK to Att.
3. For each signature query $x_i \neq m^*$, compute $\sigma_i = \mathcal{S}.\mathsf{Sign}(\mathrm{SK}, x_i)$ and send σ_i to Att.
4. Finally, Att sends a forgery $\sigma_{\mathrm{agg}} = (t^*, s^*)$ and n tuples $\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}_i$. Att wins
   (a) $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}$, $\mathrm{VK}_{i^*} = \mathrm{VK}$, $m_{i^*} = m^*$,
   (b) $(\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}, (t^*, s^*))$ is not (i^*, sk)-rejecting or $t^* > j$,
   (c) $\mathsf{AggVerify\text{-}1}_K(\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}, \sigma_{\mathrm{agg}}) = 1$.

$\mathsf{AggVerify\text{-}1}_K$

Inputs: $\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}_i, (t^*, s^*) \in \mathcal{C}_{\mathsf{HE}} \times \{0,1\}^\ell$

Constants: PRF key K

Compute $\tilde{s} = (\oplus_{i \neq i^*} F(K, \mathsf{Verify}_i || \mathrm{VK}_i || m_i || i || t)) \oplus s^*$.
Output 1 if $f(F(K, \mathsf{Verify}_{i^*} || \mathrm{VK}_{i^*} || m_{i^*} || i^* || t^*) = f(\tilde{s})$, else output 0.

Game $3, j, b$: In this game, AggSign and AggVerify-1 are replaced by AggSign-1 and AggVerify-2. Both the replaced programs use a PRF key punctured at $y = \mathcal{S}.\mathsf{Verify}||\mathrm{VK}||m_{i^*}||i^*||j+1$.

1. Att sends m^*.
2. Choose $i^* \leftarrow [n]$.
   Choose $(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{HE.setup}(1^\lambda)$ and $K \leftarrow F.\mathsf{setup}(1^\lambda)$.
   Let $y = \mathcal{S}.\mathsf{Verify}||\mathrm{VK}||m^*||i^*||j+1$, $K\{y\} \leftarrow F.\mathsf{puncture}(K, y)$ and $z = f(F(K, y))$.
   Compute $\mathrm{ct}_i \leftarrow \mathsf{HE.enc}(\mathsf{pk}, 0)$ for all $i \neq i^*$, $\mathrm{ct}_{i^*} \leftarrow \mathsf{HE.enc}(\mathsf{pk}, 1)$, $P_1 \leftarrow i\mathcal{O}(\mathsf{AggSign\text{-}1}_{K\{y\},\mathsf{pk},\mathrm{ct}_1,\ldots,\mathrm{ct}_n})$,
   $P_2 \leftarrow i\mathcal{O}(\mathsf{AggVerify\text{-}2}_{y,K\{y\},z})$.
   Set $\mathrm{PP} = (P_1, P_2)$. Send PP, VK to Att.
3. For each signature query $x_i \neq m^*$, compute $\sigma_i = \mathcal{S}.\mathsf{Sign}(\mathrm{SK}, x_i)$ and send σ_i to Att.
4. Finally, Att sends a forgery $\sigma_{\mathrm{agg}} = (t^*, s^*)$ and n tuples $\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}_i$. Att wins
   (a) $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}$, $\mathrm{VK}_{i^*} = \mathrm{VK}$, $m_{i^*} = m^*$,
   (b) $(\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}, (t^*, s^*))$ is not (i^*, sk)-rejecting or $t^* > j$,
   (c) $\mathsf{AggVerify\text{-}2}_{y,K\{y\},z}(\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}, \sigma_{\mathrm{agg}}) = 1$.

$\mathsf{AggSign\text{-}1}_{K\{y\},\mathsf{pk},\mathrm{ct}_1,\ldots,\mathrm{ct}_n}$

Inputs: $\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i, \sigma_i\}_i$.

Constants: PRF Key $K\{y\}, \mathsf{pk}, (\mathrm{ct}_1, \ldots, \mathrm{ct}_n) \in \mathcal{C}^n_{\mathsf{HE}}$.

if $\exists i$ such that $\mathsf{Verify}_i(\mathrm{VK}_i, m_i, \sigma_i) = 0$ **then**
  Output $\perp$.
end if
Compute $t = \sigma_1 \cdot \mathrm{ct}_1 + \ldots + \sigma_n \mathrm{ct}_n$.
Let $s_i = F.\mathsf{eval}(K\{y\}, \mathsf{Verify}_i||\mathrm{VK}_i||m_i||i||t)$.
Output $\sigma_{\mathrm{agg}} = (t, \oplus_i s_i)$.

$\mathsf{AggVerify\text{-}2}_{y,K\{y\},z}$

Inputs: $\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}_i, (t^*, s^*) \in \mathcal{C}_{\mathsf{HE}} \times \{0,1\}^\ell$

Constants: y, PRF key $K\{y\}$, $z \in \{0,1\}^{2\ell}$.

Compute $\tilde{s} = (\oplus_{i \neq i^*} F(K, \mathsf{Verify}_i||\mathrm{VK}_i||m_i||i||t)) \oplus s^*$.
if $\mathsf{Verify}_{i^*}||\mathrm{VK}_{i^*}||m_{i^*}||i^*||t^* = y$ **then**
  Output 1 if $z = f(\tilde{s})$, else output 0.
else
  Output 1 if $f(F.\mathsf{eval}(K, \mathsf{Verify}_{i^*}||\mathrm{VK}_{i^*}||m_{i^*}||i^*||t^*) = f(\tilde{s})$, else output 0.
end if

Game $3, j, c$: This game is similar to the previous one, except that z is a uniformly random string.

1. Att sends m^*.
2. Choose $i^* \leftarrow [n]$.
   Choose $(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{HE.setup}(1^\lambda)$ and $K \leftarrow F.\mathsf{setup}(1^\lambda)$.
   Let $y = \mathcal{S}.\mathsf{Verify}||\mathrm{VK}||m^*||i^*||j+1$, $K\{y\} \leftarrow F.\mathsf{puncture}(K, y)$ and $z' \leftarrow \{0,1\}^\ell$, $z = f(z')$.
   Compute $\mathrm{ct}_i \leftarrow \mathsf{HE.enc}(\mathsf{pk}, 0)$ for all $i \neq i^*$, $\mathrm{ct}_{i^*} \leftarrow \mathsf{HE.enc}(\mathsf{pk}, 1)$, $P_1 \leftarrow i\mathcal{O}(\mathsf{AggSign\text{-}1}_{K\{y\},\mathsf{pk},\mathrm{ct}_1,\ldots,\mathrm{ct}_n})$,
   $P_2 \leftarrow i\mathcal{O}(\mathsf{AggVerify\text{-}2}_{y,K\{y\},z})$.
   Set $\mathrm{PP} = (P_1, P_2)$. Send PP, VK to Att.

3. For each signature query $x_i \neq m^*$, compute $\sigma_i = \mathcal{S}.\mathsf{Sign}(\mathrm{SK}, x_i)$ and send σ_i to Att.
4. Finally, Att sends a forgery $\sigma_{\mathrm{agg}} = (t^*, s^*)$ and n tuples $\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}_i$. Att wins
   (a) $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}$, $\mathrm{VK}_{i^*} = \mathrm{VK}$, $m_{i^*} = m^*$,
   (b) $(\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}, (t^*, s^*))$ is not (i^*, sk)-rejecting or $t^* > j$,
   (c) $\mathsf{AggVerify\text{-}2}_{y,K\{y\},z}(\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}, \sigma_{\mathrm{agg}}) = 1$.

Game $3, j, d$: In this game, the challenger modifies the winning condition in Step 4b.

1. Att sends m^*.
2. Choose $i^* \leftarrow [n]$.
   Choose $(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{HE.setup}(1^\lambda)$ and $K \leftarrow F.\mathsf{setup}(1^\lambda)$.
   Let $y = \mathcal{S}.\mathsf{Verify}||\mathrm{VK}||m^*||i^*||j+1$, $K\{y\} \leftarrow F.\mathsf{puncture}(K, y)$, $z' \leftarrow \{0,1\}^\ell$ and $z = f(z')$.
   Compute $\mathrm{ct}_i \leftarrow \mathsf{HE.enc}(\mathsf{pk}, 0)$ for all $i \neq i^*$, $\mathrm{ct}_{i^*} \leftarrow \mathsf{HE.enc}(\mathsf{pk}, 1)$, $P_1 \leftarrow i\mathcal{O}(\mathsf{AggSign\text{-}1}_{K\{y\},\mathsf{pk},\mathrm{ct}_1,\ldots,\mathrm{ct}_n})$,
   $P_2 \leftarrow i\mathcal{O}(\mathsf{AggVerify\text{-}2}_{y,K\{y\},z})$.
   Set $\mathrm{PP} = (P_1, P_2, \mathsf{pk}, \mathrm{ct}_1, \ldots, \mathrm{ct}_n)$. Send PP, VK to Att.
3. For each signature query $x_i \neq m^*$, compute $\sigma_i = \mathcal{S}.\mathsf{Sign}(\mathrm{SK}, x_i)$ and send σ_i to Att.
4. Finally, Att sends a forgery $\sigma_{\mathrm{agg}} = (t^*, s^*)$ and n tuples $\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}_i$. Att wins
   (a) $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}$, $\mathrm{VK}_{i^*} = \mathrm{VK}$, $m_{i^*} = m^*$,
   (b) $(\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}, (t^*, s^*))$ is not (i^*, sk)-rejecting or $t^* > j+1$,
   (c) $\mathsf{AggVerify\text{-}2}_{y,K\{y\},z}(\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}, \sigma_{\mathrm{agg}}) = 1$.

Game $3, j, e$: In this game, the challenger sets $z = f(F(K, y))$ as in **Game** $3, j, c$.

1. Att sends m^*.
2. Choose $i^* \leftarrow [n]$.
   Choose $(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{HE.setup}(1^\lambda)$ and $K \leftarrow F.\mathsf{setup}(1^\lambda)$.
   Let $y = \mathcal{S}.\mathsf{Verify}||\mathrm{VK}||m^*||i^*||j+1$, $K\{y\} \leftarrow F.\mathsf{puncture}(K, y)$, and $z = f(F(K, y))$.
   Compute $\mathrm{ct}_i \leftarrow \mathsf{HE.enc}(\mathsf{pk}, 0)$ for all $i \neq i^*$, $\mathrm{ct}_{i^*} \leftarrow \mathsf{HE.enc}(\mathsf{pk}, 1)$, $P_1 \leftarrow i\mathcal{O}(\mathsf{AggSign\text{-}1}_{K\{y\},\mathsf{pk},\mathrm{ct}_1,\ldots,\mathrm{ct}_n})$,
   $P_2 \leftarrow i\mathcal{O}(\mathsf{AggVerify\text{-}2}_{y,K\{y\},z})$.
   Set $\mathrm{PP} = (P_1, P_2)$. Send PP, VK to Att.
3. For each signature query $x_i \neq m^*$, compute $\sigma_i = \mathcal{S}.\mathsf{Sign}(\mathrm{SK}, x_i)$ and send σ_i to Att.
4. Finally, Att sends a forgery $\sigma_{\mathrm{agg}} = (t^*, s^*)$ and n tuples $\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}_i$. Att wins
   (a) $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}$, $\mathrm{VK}_{i^*} = \mathrm{VK}$, $m_{i^*} = m^*$,
   (b) $(\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}, (t^*, s^*))$ is not (i^*, sk)-rejecting or $t^* > j+1$,
   (c) $\mathsf{AggVerify\text{-}2}_{y,K\{y\},z}(\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}, \sigma_{\mathrm{agg}}) = 1$.

Game $3, j, f$**:** In this game, the challenger uses PRF key K in both $\mathsf{AggSign}$ and $\mathsf{AggVerify\text{-}1}$ instead of using $K\{y\}$ in $\mathsf{AggSign\text{-}1}$ and $\mathsf{AggVerify\text{-}2}$.

1. Att sends m^*.
2. Choose $i^* \leftarrow [n]$.
   Choose $(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{HE.setup}(1^\lambda)$ and $K \leftarrow F.\mathsf{setup}(1^\lambda)$.
   Compute $\mathrm{ct}_i \leftarrow \mathsf{HE.enc}(\mathsf{pk}, 0)$ for all $i \neq i^*$, $\mathrm{ct}_{i^*} \leftarrow \mathsf{HE.enc}(\mathsf{pk}, 1)$, $P_1 \leftarrow i\mathcal{O}(\mathsf{AggSign}_{K,\mathsf{pk},\mathrm{ct}_1,\ldots,\mathrm{ct}_n})$,
   $P_2 \leftarrow i\mathcal{O}(\mathsf{AggVerify\text{-}1}_K)$.
   Set $\mathrm{PP} = (P_1, P_2)$. Send PP, VK to Att.
3. For each signature query $x_i \neq m^*$, compute $\sigma_i = \mathcal{S}.\mathsf{Sign}(\mathrm{SK}, x_i)$ and send σ_i to Att.
4. Finally, Att sends a forgery $\sigma_{\mathrm{agg}} = (t^*, s^*)$ and n tuples $\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}_i$. Att wins
   (a) $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}$, $\mathrm{VK}_{i^*} = \mathrm{VK}$, $m_{i^*} = m^*$,
   (b) $(\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}, (t^*, s^*))$ is not (i^*, sk)-rejecting or $t^* > j+1$,
   (c) $\mathsf{AggVerify\text{-}1}_K(\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}, \sigma_{\mathrm{agg}}) = 1$.

We will now relate the difference in Att's advantages in these games to either $\mathsf{Adv}^{i\mathcal{O}}$, Adv^{F} or Adv^{f}.

Claim 7.4. For any PPT adversary Att,

$$\mathsf{Adv}^{3,j}_{\mathsf{Att}} - \mathsf{Adv}^{3,j,a}_{\mathsf{Att}} \leq \mathsf{Adv}^{i\mathcal{O}}.$$

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

$$\begin{aligned}
& s^* = \oplus_i F(K, \mathsf{Verify}_i || \mathrm{VK}_i || m_i || i || t^*) \\
\iff & (\oplus_{i \neq i^*} F(K, \mathsf{Verify}_i || \mathrm{VK}_i || m_i || i || t^*)) \oplus s^* = F(K, \mathsf{Verify}_{i^*} || \mathrm{VK}_{i^*} || m_{i^*} || i^* || t^*) \\
\iff & f((\oplus_{i \neq i^*} F(K, \mathsf{Verify}_i || \mathrm{VK}_i || m_i || i || t^*)) \oplus s^*) = f(F(K, \mathsf{Verify}_{i^*} || \mathrm{VK}_{i^*} || m_{i^*} || i^* || t^*))
\end{aligned}$$

■

Claim 7.5. For any PPT adversary Att,

$$\mathsf{Adv}^{3,j,a}_{\mathsf{Att}} - \mathsf{Adv}^{3,j,b}_{\mathsf{Att}} \leq 2\mathsf{Adv}^{i\mathcal{O}}.$$

Proof. Let $K \leftarrow F.\mathsf{setup}(1^\lambda)$, $y = \mathcal{S}.\mathsf{Verify}||\mathrm{VK}||m^*||i^*||j+1$, $K\{y\} \leftarrow F.\mathsf{puncture}(K, y)$ and $z = f(F(K, y))$. As in the previous proof, it suffices to show that $\mathsf{AggSign}_{K,\mathsf{pk},\mathrm{ct}_1,\ldots,\mathrm{ct}_n}$ and $\mathsf{AggSign\text{-}1}_{K\{y\},\mathsf{pk},\mathrm{ct}_1,\ldots,\mathrm{ct}_n}$ have identical functionality, and $\mathsf{AggVerify\text{-}1}_K$ and $\mathsf{AggVerify\text{-}2}_{y,K\{y\},z}$ have identical functionality.

Let us first consider $\mathsf{AggSign}_{K,\mathsf{pk},\mathrm{ct}_1,\ldots,\mathrm{ct}_n}$ and $\mathsf{AggSign\text{-}1}_{K\{y\},\mathsf{pk},\mathrm{ct}_1,\ldots,\mathrm{ct}_n}$. Consider input $\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i, \sigma_i\}_i$. Let $t = \sigma_1 \cdot \mathrm{ct}_1 + \ldots + \sigma_n \mathrm{ct}_n$. From the correctness property of puncturable PRFs, it follows that the only case in which $\mathsf{AggSign}_{K,\mathsf{pk},\mathrm{ct}_1,\ldots,\mathrm{ct}_n}$ and $\mathsf{AggSign}_{y,K\{y\},\mathsf{pk},\mathrm{ct}_1,\ldots,\mathrm{ct}_n}$ can possibly differ is when $\mathsf{Verify}_i(\mathrm{VK}_i, m_i, \sigma_i) = 1$ for all $i \leq n$, $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}$, $\mathrm{VK}_{i^*} = \mathrm{VK}$, $m_{i^*} = m^*$ and $t = j+1$. But this case is not possible, since $\mathcal{S}.\mathsf{Verify}(\mathrm{VK}, m^*, \mathsf{HE.dec}(\mathsf{sk}, t)) = \mathcal{S}.\mathsf{Verify}(\mathrm{VK}, m^*, \sigma_{i^*}) = 1$, while $\mathcal{S}.\mathsf{Verify}(\mathrm{VK}, m^*, \mathsf{HE.dec}(\mathsf{sk}, j+1)) = 0$.

Next, let us consider the programs $\mathsf{AggVerify\text{-}1}_K$ and $\mathsf{AggVerify}_{y,K\{y\},z}$. 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. ■

Claim 7.6. For any PPT adversary Att,

$$\mathsf{Adv}^{3,j,b}_{\mathsf{Att}} - \mathsf{Adv}^{3,j,c}_{\mathsf{Att}} \leq \mathsf{Adv}^{F}.$$

Proof. We will construct a PPT algorithm $\mathcal{B}$ such that $\mathsf{Adv}^{F}_{\mathcal{B}} = \mathsf{Adv}^{3,j,b}_{\mathsf{Att}} - \mathsf{Adv}^{3,j,c}_{\mathsf{Att}}$. $\mathcal{B}$ interacts with Att, and receives m^*. It chooses $i^* \leftarrow [n]$, chooses $(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{HE.setup}(1^\lambda)$. Next, it computes $\mathrm{ct}_i \leftarrow \mathsf{HE.enc}(\mathsf{pk}, 0)$ for all $i \neq i^*$, $\mathrm{ct}_{i^*} \leftarrow \mathsf{HE.enc}(\mathsf{pk}, 1)$. It sends $y = \mathcal{S}.\mathsf{Verify}||\mathrm{VK}||m^*||i^*||j+1$ to the PRF challenger, and receives $K\{y\}, z'$, where either $z' = F(K, y)$ or $z' \leftarrow \{0,1\}^{2\ell}$. It computes $z = f(z')$, $P_1 \leftarrow i\mathcal{O}(\mathsf{AggSign\text{-}1}_{K\{y\},\mathsf{pk},\mathrm{ct}_1,\ldots,\mathrm{ct}_n})$, $P_2 \leftarrow i\mathcal{O}(\mathsf{AggVerify\text{-}2}_{y,K\{y\},z})$ and sets $\mathrm{PP} = (P_1, P_2)$. It sends PP, VK to Att.

Next, it receives signature queries, and it computes the signature using SK. Finally, it receives $\sigma_{\mathrm{agg}} = (t^*, s^*)$ and n tuples $\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}_i$. If Att wins, it outputs 0, indicating $z' = F(K, y)$. Else, it outputs 1. Since both games have the same winning condition, it follows $\mathsf{Adv}^{F}_{\mathcal{B}} = \mathsf{Adv}^{3,j,b}_{\mathsf{Att}} - \mathsf{Adv}^{3,j,c}_{\mathsf{Att}}$. ■

Claim 7.7. For any PPT adversary Att,

$$\mathsf{Adv}^{3,j,c}_{\mathsf{Att}} - \mathsf{Adv}^{3,j,d}_{\mathsf{Att}} \leq \mathsf{Adv}^{f}.$$

Proof. Suppose $\mathsf{Adv}^{3,j,c}_{\mathsf{Att}} - \mathsf{Adv}^{3,j,d}_{\mathsf{Att}} = \epsilon$. Then, with probability ϵ, Att receives PP, VK, sends signature queries, and outputs forgery $\sigma_{\mathrm{agg}} = (j+1, s^*)$ and n tuples $\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}_i$ such that $\mathcal{S}.\mathsf{Verify}_{i^*} = \mathsf{Verify}_{i^*}$, $\mathrm{VK}_{i^*} = \mathrm{VK}$, $m_{i^*} = m^*$, the output forgery is (i^*, sk)-rejecting and $\mathsf{AggVerify\text{-}2}_{y,K\{y\},z}(\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}, \sigma_{\mathrm{agg}}) = 1$. From the definition of $\mathsf{AggVerify\text{-}2}$, it follows that $f((\oplus_{i \neq i^*} F.\mathsf{eval}(K\{y\}, \mathsf{Verify}_i || \mathrm{VK}_i || m_i || i || j+1)) \oplus s^*) = z$. Therefore, using Att, we can construct a PPT algorithm $\mathcal{B}$ that breaks the security of one way function f with advantage ϵ. $\mathcal{B}$ receives z from the OWF challenger, and uses it to compute PP as in $\mathsf{Game}\ 3, j, c$ and $\mathsf{Game}\ 3, j, d$. It sends PP, VK to Att, responds to signature queries, and finally receives forgery $(j+1, s^*)$[25].

[25] If $\mathcal{B}$ receives any other forgery, then it simply quits.

and n tuples. It sends $z' = (\oplus_{i \neq i^*} F.\mathsf{eval}(K\{y\}, \mathsf{Verify}_i || \mathrm{VK}_i || m_i || i || j+1)) \oplus s^*$ to the OWF challenger, and clearly, $\mathcal{B}$ wins if Att wins. This completes our proof. ∎

Claim 7.8. For any PPT adversary Att,

$$\mathsf{Adv}^{3,j,d}_{\mathsf{Att}} - \mathsf{Adv}^{3,j,e}_{\mathsf{Att}} \leq \mathsf{Adv}^{F}.$$

Proof. Similar to the proof of Claim 7.6. ∎

Claim 7.9. For any PPT adversary Att,

$$\mathsf{Adv}^{3,j,e}_{\mathsf{Att}} - \mathsf{Adv}^{3,j,f}_{\mathsf{Att}} \leq 2\mathsf{Adv}^{i\mathcal{O}}.$$

Proof. Similar to the proof of Claim 7.5. ∎

Claim 7.10. For any PPT adversary Att,

$$\mathsf{Adv}^{3,j,f}_{\mathsf{Att}} - \mathsf{Adv}^{3,j+1}_{\mathsf{Att}} \leq \mathsf{Adv}^{i\mathcal{O}}.$$

Proof. Similar to the proof of Claim 7.4. ∎

Summing it up, from the above claims, it follows that for any PPT adversary Att, $\mathsf{Adv}^{3,j}_{\mathsf{Att}} - \mathsf{Adv}^{3,j+1}_{\mathsf{Att}} \leq 6\mathsf{Adv}^{i\mathcal{O}} + 2\mathsf{Adv}^{F} + \mathsf{Adv}^{f}$.

∎

Claim 7.11. For any PPT adversary Att,

$$\mathsf{Adv}^{4}_{\mathsf{Att}} \leq \mathsf{Adv}^{\mathcal{S}}.$$

Proof. Suppose there exists a PPT adversary Att such that $\mathsf{Adv}^{4}_{\mathsf{Att}} = \epsilon$. We will construct a PPT algorithm $\mathcal{B}$ that breaks the security of $\mathcal{S}$ with advantage ϵ.

$\mathcal{B}$ interacts with Att and the challenger for $\mathcal{S}$. First, it receives m^* from $\mathcal{S}$ and VK from the challenger. It chooses $i^* \leftarrow [n]$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{HE.setup}(1^\lambda)$, $K \leftarrow F.\mathsf{setup}(1^\lambda)$. It computes $\mathrm{ct}_i \leftarrow \mathsf{HE.enc}(\mathsf{pk}, 0)$ for all $i \neq i^*$, $\mathrm{ct}_{i^*} \leftarrow \mathsf{HE.enc}(\mathsf{pk}, 1)$, $P_1 \leftarrow i\mathcal{O}(\mathsf{AggSign}_{K,\mathsf{pk},\mathrm{ct}_1,\ldots,\mathrm{ct}_n})$ and $P_2 \leftarrow i\mathcal{O}(\mathsf{AggVerify}_K)$. It sends $\mathrm{PP} = (P_1, P_2), \mathrm{VK}$ to Att.

For each signature query $x_i \neq m^*$ sent by Att, it forwards x_i to the challenger, and receives σ_i, which it sends to Att.

Finally, Att outputs a forgery $\sigma_{\mathrm{agg}} = (t^*, s^*)$ along with n tuples $\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}$. If Att wins in Game 4, then $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}$, $\mathrm{VK}_{i^*} = \mathrm{VK}$, $m_{i^*} = m^*$ and $(\sigma_{\mathrm{agg}}, \{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\})$ must not be (i^*, sk)-rejecting. In other words, $\mathcal{S}.\mathsf{Verify}(\mathrm{VK}, m^*, \mathsf{HE.dec}(\mathsf{sk}, t^*)) = 1$. $\mathcal{B}$ sends $m^*, \mathsf{HE.dec}(\mathsf{sk}, t^*)$ as a forgery to the challenger. This completes our proof. ∎

Summing up, it follows that any adversary Att has advantage at most $n(\mathsf{Adv}^{\mathcal{HE}}_{\mathsf{Att}} + 2^{\ell_{\mathrm{ct}}}(6\mathsf{Adv}^{i\mathcal{O}}_{\mathsf{Att}} + 2\mathsf{Adv}^{F}_{\mathsf{Att}} + \mathsf{Adv}^{f}_{\mathsf{Att}}) + \mathsf{Adv}^{\mathcal{S}}_{\mathsf{Att}})$ in Game 0, where $\mathsf{Adv}^{\mathcal{HE}}_{\mathsf{Att}}$, $\mathsf{Adv}^{i\mathcal{O}}_{\mathsf{Att}}$, $\mathsf{Adv}^{F}_{\mathsf{Att}}$, $\mathsf{Adv}^{f}_{\mathsf{Att}}$ and $\mathsf{Adv}^{\mathcal{S}}_{\mathsf{Att}}$ denote the advantages of Att in the security games for HE scheme $\mathcal{HE}$, indistinguishability obfuscator $i\mathcal{O}$, (selectively secure) puncturable PRF F, one-way function f and signature scheme $\mathcal{S}$ respectively. Therefore, if $2^{\ell_{\mathrm{ct}}}(\mathsf{Adv}^{i\mathcal{O}}_{\mathsf{Att}} + \mathsf{Adv}^{F}_{\mathsf{Att}} + \mathsf{Adv}^{f}_{\mathsf{Att}})$ is negligible in λ, then the aggregator scheme described in Section 7 is adaptively secure with respect to all signature schemes $\mathcal{S}$. Note that we require sub-exponential hardness assumption for the indistinguishability obfuscator $i\mathcal{O}$, puncturable PRF F and one-way function f.

References

[AGH10] Jae Hyun Ahn, Matthew Green, and Susan Hohenberger. Synchronized aggregate signatures: new definitions, constructions and applications. In *Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010*, pages 473–484, 2010.

[BB04] Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In *CRYPTO*, pages 443–459, 2004.

[BCCT13] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for SNARKS and proof-carrying data. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 111–120, 2013.

[Ben87] Josh Daniel Cohen Benaloh. *Verifiable Secret-ballot Elections*. PhD thesis, Yale University, 1987.

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In *CRYPTO*, pages 1–18, 2001.

[BGI+12] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6, 2012.

[BGI13] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. *IACR Cryptology ePrint Archive*, 2013:401, 2013.

[BGLS03] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *EUROCRYPT*, pages 416–432, 2003.

[BGOY07] Alexandra Boldyreva, Craig Gentry, Adam O'Neill, and Dae Hyun Yum. Ordered multisignatures and identity-based sequential aggregate signatures, with applications to secure routing. In *Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28-31, 2007*, pages 276–285, 2007.

[BKM08] Adam Bender, Jonathan Katz, and Ruggero Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. *J. Cryptol.*, 22(1):114–138, December 2008.

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In *ASIACRYPT*, pages 514–532, 2001.

[BNN07] Mihir Bellare, Chanathip Namprempre, and Gregory Neven. Unrestricted aggregate signatures. In *ICALP*, pages 411–422, 2007.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[BR96] Mihir Bellare and Phillip Rogaway. The exact security of digital signatures - how to sign with rsa and rabin. In *EUROCRYPT*, pages 399–416, 1996.

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In *ASIACRYPT*, pages 280–300, 2013.

[DJ03] Ivan Damgård and Mads Jurik. A length-flexible threshold cryptosystem with applications. In *Information Security and Privacy, 8th Australasian Conference, ACISP 2003, Wollongong, Australia, July 9-11, 2003, Proceedings*, pages 350–364, 2003.

[FHPS13] Eduarda S. V. Freire, Dennis Hofheinz, Kenneth G. Paterson, and Christoph Striecks. Programmable hash functions in the multilinear setting. In *CRYPTO*, pages 513–530, 2013.

[GGH+13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, 2013.

[GM84] S. Goldwasser and S. Micali. Probabilistic encryption. *Jour. of Computer and System Science*, 28(2):270–299, 1984.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.

[GO93] Shafi Goldwasser and Rafail Ostrovsky. Invariant signatures and non-interactive zero-knowledge proofs are equivalent (extended abstract). In *Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '92, pages 228–245, London, UK, UK, 1993. Springer-Verlag.

[GR06] Craig Gentry and Zulfikar Ramzan. Identity-based aggregate signatures. In *Public Key Cryptography - PKC 2006, 9th International Conference on Theory and Practice of Public-Key Cryptography, New York, NY, USA, April 24-26, 2006, Proceedings*, pages 257–273, 2006.

[Had00] Satoshi Hada. Zero-knowledge and code obfuscation. In *Proceedings of the 6th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*, ASIACRYPT '00, pages 443–457, 2000.

[HDWH12] Nadia Heninger, Zakir Durumeric, Eric Wustrow, and J. Alex Halderman. Mining your ps and qs: Detection of widespread weak keys in network devices. In *Proceedings of the 21th USENIX Security Symposium, Bellevue, WA, USA, August 8-10, 2012*, pages 205–220, 2012.

[HJK+14] Dennis Hofheinz, Tibor Jager, Dakshita Khurana, Amit Sahai, Brent Waters, and Mark Zhandry. How to generate and use universal parameters. Cryptology ePrint Archive, Report 2014/507, 2014. `http://eprint.iacr.org/`.

[HSW14] Susan Hohenberger, Amit Sahai, and Brent Waters. Replacing a random oracle: Full domain hash from indistinguishability obfuscation. In *EUROCRYPT*, pages 201–220, 2014.

[KLMS00] Stephen Kent, Charles Lynn, Joanne Mikkelson, and Karen Seo. Secure border gateway protocol (s-bgp. *IEEE Journal on Selected Areas in Communications*, 18:103–116, 2000.

[KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In *ACM Conference on Computer and Communications Security*, pages 669–684, 2013.

[KS98] B. Kaliski and J. Staddon. PKCS #1: RSA Cryptography Specifications Version 2.0. In *RFC Editor*, United States, 1998.

[LOS+06] Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters. Sequential aggregate signatures and multisignatures without random oracles. In *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, pages 465–485, 2006.

[NS98] David Naccache and Jacques Stern. A new public key cryptosystem based on higher residues. In *Proceedings of the 5th ACM Conference on Computer and Communications Security*, CCS '98, pages 59–66, 1998.

[OO98] Kazuo Ohta and Tatsuaki Okamoto. On concrete security treatment of signatures derived from identification. In *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings*, pages 354–369, 1998.

[OU98] Tatsuaki Okamoto and Shigenori Uchiyama. A new public-key cryptosystem as secure as factoring. In *Advances in Cryptology - EUROCRYPT '98, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, May 31 - June 4, 1998, Proceeding*, pages 308–318, 1998.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proceedings of the 17th International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT'99, pages 223–238, 1999.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *STOC*, pages 475–484, 2014.

[ZSN05] Meiyuan Zhao, Sean W. Smith, and David M. Nicol. Aggregated path authentication for efficient bgp security. In *ACM Conference on Computer and Communications Security*, pages 128–138, 2005.

A Universally Aggregating Unique Signatures without the RSA Assumption

In this section we show a modification of our universal aggregation of unique signatures construction and proof from Section 4 The primary diffence is that the transformed siganture output will be a bit string as opposed to an RSA-type group element in $\mathbb{Z}_N$. Thus, we are able to prove security without using the RSA assumption (but keeping indistinguishability obfuscation and punctured PRF security assumptions.) The primary tradeoff is that the setup must commit to an a-priori bound, n on the number of signatures that can be aggregated. The signature length is independent of n.

We will now describe our n-bounded universal signature aggregator $(\ell_{\text{ver}}, \ell_{\text{vk}}, \ell_{\text{msg}}, \ell_{\text{sig}})$-$\mathsf{UniversalSigAgg}$. Let ℓ and ℓ_{owf} be polynomials such that $\ell(\lambda) \geq \lambda$. We will use a puncturable PRF F with key space $\mathcal{K}$, punctured key space $\mathcal{K}_p$, domain $\mathcal{X} = \{0,1\}^{\ell_{\text{ver}}} \times \{0,1\}^{\ell_{\text{vk}}} \times \{0,1\}^{\ell_{\text{msg}}}$ and range $\mathcal{Y} = \{0,1\}^{\ell}$, a one-way function $f : \{0,1\}^{\ell} \to \{0,1\}^{\ell_{\text{owf}}}$ and an indistinguishability obfuscator $i\mathcal{O}$. Our scheme consists of the three algorithms $\mathsf{UniversalSetup}$, $\mathsf{UniversalAgg}$ and $\mathsf{UniversalVerify}$.

$\mathsf{UniversalSetup}(1^\lambda, 1^n)$**:** $\mathsf{UniversalSetup}$ takes as input the security parameter λ and a bound n on the number of signatures to be aggregated. It chooses a puncturable PRF key $K \leftarrow F.\mathsf{setup}(1^\lambda)$ and computes obfuscations of the circuits $\mathsf{Transform}_K$ and $\mathsf{AggVerify}_K$ defined below. It sets the public parameters to be $\text{PP} = (i\mathcal{O}(\mathsf{Transform}_K), i\mathcal{O}(\mathsf{AggVerify}_K))$.

$\mathsf{Transform}_K$:

Inputs: $\mathsf{Verify}' \in \{0,1\}^{\ell_{\text{ver}}}$, $\text{VK}' \in \{0,1\}^{\ell_{\text{vk}}}$, $m' \in \{0,1\}^{\ell_{\text{msg}}}$, $\sigma' \in \{0,1\}^{\ell_{\text{sig}}}$.
Constants : $K \in \mathcal{K}$.

if $\mathsf{Verify}'(\text{VK}', m', \sigma') = 0$ **then**
  Output $\perp$.
else
  Output $F(K, \mathsf{Verify}'||\text{VK}'||m')$.
end if

$\mathsf{AggVerify}_K$:

Inputs: $\{(\mathsf{Verify}_i, \mathrm{VK}_i, m_i)\}_{i=1}^n$ where $(\mathsf{Verify}_i, \mathrm{VK}_i, m_i) \in \{0,1\}^{\ell_{\mathrm{ver}}} \times \{0,1\}^{\ell_{\mathrm{vk}}} \times \{0,1\}^{\ell_{\mathrm{msg}}}$ for all $i \leq n$, $\sigma_{\mathrm{agg}} \in \{0,1\}^{\ell}$.
Constants : $K \in \mathcal{K}$.

for all $i \leq n$ **do**
Compute $s_i = F(K, \mathsf{Verify}_i || \mathrm{VK}_i || m_i)$.
end for
Output 1 if $\oplus_{i=1}^n s_i = \sigma_{\mathrm{agg}}$, 0 otherwise.

$\mathsf{UniversalAgg}(\mathrm{PP}, \{(\mathsf{Verify}_i, \mathrm{VK}_i, m_i, \sigma_i)\}_{i=1}^n)$**:** Let $\mathrm{PP} = (P_1, P_2)$. UniversalAgg first checks that the n tuples are distinct. If not, it outputs $\perp$. Else, it computes $t_i = P_1(\mathsf{Verify}_i, \mathrm{VK}_i, m_i, \sigma_i)$ for each $i \leq n$. If $t_i = \perp$ for some i, then UniversalAgg outputs $\perp$, else it outputs $\sigma_{\mathrm{agg}} = \oplus_i t_i$.

$\mathsf{UniversalVerify}(\mathrm{PP}, \{(\mathsf{Verify}_i, \mathrm{VK}_i, m_i)\}_{i=1}^n, \sigma_{\mathrm{agg}})$**:** Let $\mathrm{PP} = (P_1, P_2)$. UniversalVerify first checks if the n tuples are distinct. If not, it outputs 0. Else, it outputs $P_2(\{(\mathsf{Verify}_i, \mathrm{VK}_i, m_i)\}_{i=1}^n, \sigma_{\mathrm{agg}})$.

A.1 Proof of security

In this section, we will show that our construction is selectively secure with respect to unique signature schemes.

Theorem A.1. Assuming $i\mathcal{O}$ is a secure indistinguishability obfuscator, $(F, F.\mathsf{setup}, F.\mathsf{puncture}, F.\mathsf{eval})$ is a puncturable PRF and f is an injective one way function, for all $(\ell_{\mathrm{ver}}, \ell_{\mathrm{vk}}, \ell_{\mathrm{msg}}, \ell_{\mathrm{sig}})$-length qualified secure unique signature schemes $\mathcal{S}$, the n-bounded universal signature aggregator $(\ell_{\mathrm{ver}}, \ell_{\mathrm{vk}}, \ell_{\mathrm{msg}}, \ell_{\mathrm{sig}})$-UniversalSigAgg is selectively secure with respect to $\mathcal{S}$.

Let $\mathcal{S} = (\mathcal{S}.\mathsf{Gen}, \mathcal{S}.\mathsf{Sign}, \mathcal{S}.\mathsf{Verify})$ be a secure $(\ell_{\mathrm{ver}}, \ell_{\mathrm{vk}}, \ell_{\mathrm{msg}}, \ell_{\mathrm{sig}})$-length qualified unique signature scheme, and Att a PPT adversary. Assume Att sends q signing queries during the signing phase. In order to prove this theorem, we will define a sequence of experiments Game 0, ..., Game 4, where Game $0 = \mathsf{Exp}^{\mathrm{sel}}_{\mathsf{Att},\mathcal{S}}$.

A.1.1 Sequence of Games

Game 0**:** This game corresponds to $\mathsf{Exp}^{\mathrm{sel}}_{\mathsf{Att},\mathcal{S}}$. The adversary Att first sends message m, and then receives the verification key and public parameters for the aggregator. Next, the adversary makes signing queries, and finally submits the forgery.

1. Att sends message m.
2. Compute $(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda, 1^n)$. Choose $K \leftarrow F.\mathsf{setup}(1^\lambda)$ and set $\mathrm{PP} = (i\mathcal{O}(\mathsf{Transform}_K), i\mathcal{O}(\mathsf{AggVerify}_K))$. Send PP, VK to Att.
3. For each signing query $x_i \neq m$, compute $\sigma_i \leftarrow \mathcal{S}.\mathsf{Sign}(\mathrm{SK}, x_i)$ and send σ_i to Att.
4. Att sends forgery σ_{agg} and n tuples $\{(\mathsf{Verify}_i, \mathrm{VK}_i, m_i)\}$. Att wins if $\exists i^* \in [n]$ such that $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}$, $\mathrm{VK}_{i^*} = \mathrm{VK}$ and $m_{i^*} = m$ and $\mathsf{UniversalVerify}(\mathrm{PP}, \{(\mathsf{Verify}_i, \mathrm{VK}_i, m_i)\}, \sigma_{\mathrm{agg}}) = 1$.

Game 1**:** This game is exactly similar to the previous one, except that the program $\mathsf{Transform}_K$ is replaced by $\mathsf{Transform}'_K$ which outputs $\perp$ if the input tuples is $(\mathcal{S}.\mathsf{Verify}, \mathrm{VK}, m, \sigma)$ where $\mathcal{S}.\mathsf{Verify}(\mathrm{VK}, m, \sigma) = 1$.

1. Att sends message m.
2. Compute $(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda, 1^n)$. Choose $K \leftarrow F.\mathsf{setup}(1^\lambda)$.
Set $y = \mathcal{S}.\mathsf{Verify}||\mathrm{VK}||m$ and $\mathrm{PP} = (i\mathcal{O}(\mathsf{Transform}'_{y,K}), i\mathcal{O}(\mathsf{AggVerify}_K))$. Send PP, VK to Att.
3. For each signing query $x_i \neq m$, compute $\sigma_i \leftarrow \mathcal{S}.\mathsf{Sign}(\mathrm{SK}, x_i)$ and send σ_i to Att.

4. Att sends forgery σ_{agg} and n tuples $\{(\mathsf{Verify}_i, \text{VK}_i, m_i)\}$. Att wins if $\exists i^* \in [n]$ such that $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}$, $\text{VK}_{i^*} = \text{VK}$ and $m_{i^*} = m$ and $\mathsf{UniversalVerify}(\text{PP}, \{(\mathsf{Verify}_i, \text{VK}_i, m_i)\}, \sigma_{\text{agg}}) = 1$.

$\mathsf{Transform}'_{y,K}$:

Inputs: $\mathsf{Verify}' \in \{0,1\}^{\ell_{\text{ver}}}$, $\text{VK}' \in \{0,1\}^{\ell_{\text{vk}}}$, $m' \in \{0,1\}^{\ell_{\text{msg}}}$, $\sigma' \in \{0,1\}^{\ell_{\text{sig}}}$.
Constants : $y \in \{0,1\}^{\ell_{\text{ver}}} \times \{0,1\}^{\ell_{\text{vk}}} \times \{0,1\}^{\ell_{\text{msg}}}$, $K \in \mathcal{K}$.

if $\mathsf{Verify}'(\text{VK}', m', \sigma') = 0$ **then**
  Output $\perp$.
else if $\mathsf{Verify}'||\text{VK}'||m' = y$ **then**
  Output $\perp$.
else
  Output $F(K, \mathsf{Verify}'||\text{VK}'||m')$.
end if

Game 2: This game is similar to the previous one, except that the programs $\mathsf{Transform}'$ and $\mathsf{AggVerify}$ are replaced by $\mathsf{Transform}$-1 and $\mathsf{AggVerify}$-1 respectively. Each of these programs uses a PRF key punctured at $y = \mathcal{S}.\mathsf{Verify}||\text{VK}||m$.

1. Att sends message m.
2. Compute $(\text{SK}, \text{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda, 1^n)$. Choose $K \leftarrow F.\mathsf{setup}(1^\lambda)$.
   Set $y = \mathcal{S}.\mathsf{Verify}||\text{VK}||m$. Compute $K\{y\} \leftarrow F.\mathsf{puncture}(K, y)$ and $z = F(K, y)$.
   Set $\text{PP} = (i\mathcal{O}(\mathsf{Transform}\text{-}1_{y,K\{y\}}), i\mathcal{O}(\mathsf{AggVerify}\text{-}1_{y,K\{y\},z}))$ and send PP, VK to Att.
3. For each signing query $x_i \neq m$, compute $\sigma_i \leftarrow \mathcal{S}.\mathsf{Sign}(\text{SK}, x_i)$ and send σ_i to Att.
4. Att sends forgery σ_{agg} and n tuples $\{(\mathsf{Verify}_i, \text{VK}_i, m_i)\}$. Att wins if $\exists i^* \in [n]$ such that $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}$, $\text{VK}_{i^*} = \text{VK}$ and $m_{i^*} = m$ and $\mathsf{UniversalVerify}(\text{PP}, \{(\mathsf{Verify}_i, \text{VK}_i, m_i)\}, \sigma_{\text{agg}}) = 1$.

$\mathsf{Transform}\text{-}1_{y,K\{y\}}$:

Inputs: $\mathsf{Verify}' \in \{0,1\}^{\ell_{\text{ver}}}$, $\text{VK}' \in \{0,1\}^{\ell_{\text{vk}}}$, $m' \in \{0,1\}^{\ell_{\text{msg}}}$, $\sigma \in \{0,1\}^{\ell_{\text{sig}}}$.
Constants : $y \in \{0,1\}^{\ell_{\text{ver}}} \times \{0,1\}^{\ell_{\text{vk}}} \times \{0,1\}^{\ell_{\text{msg}}}$, $K\{y\} \in \mathcal{K}_p$.

if $\mathsf{Verify}'(\text{VK}', m', \sigma') = 0$ **then**
  Output $\perp$.
else if $\mathsf{Verify}'||\text{VK}'||m' = y$ **then**
  Output $\perp$.
else
  Output $F.\mathsf{eval}(K\{y\}, \mathsf{Verify}'||\text{VK}'||m')$.
end if

$\mathsf{AggVerify\text{-}1}_{y,K\{y\},z}$:

Inputs: $\{(\mathsf{Verify}_i, \mathrm{VK}_i, m_i)\}_{i=1}^n$ where $(\mathsf{Verify}_i, \mathrm{VK}_i, m_i) \in \{0,1\}^{\ell_{\mathrm{ver}}} \times \{0,1\}^{\ell_{\mathrm{vk}}} \times \{0,1\}^{\ell_{\mathrm{msg}}}$ for all $i \leq n$, $\sigma_{\mathrm{agg}} \in \{0,1\}^{\ell}$.
Constants : $y \in \{0,1\}^{\ell_{\mathrm{ver}}} \times \{0,1\}^{\ell_{\mathrm{vk}}} \times \{0,1\}^{lmsg}$, $K\{y\} \in \mathcal{K}_p$, $z \in \{0,1\}^{\ell}$.

for all $i \leq n$ **do**
  if $\mathsf{Verify}_i||\mathrm{VK}_i||m_i = y$ **then**
    $s_i = z$
  else
    Compute $s_i = F.\mathsf{eval}(K\{y\}, \mathsf{Verify}_i||\mathrm{VK}_i||m_i)$.
  end if
end for
Output 1 if $\oplus_{i=1}^n s_i = \sigma_{\mathrm{agg}}$, 0 otherwise.

Game 3: In this game, the program AggVerify-1 is replaced by AggVerify-2. As before, a punctured key is used in the program. However, instead of directly checking whether $\sigma_{\mathrm{agg}} = \oplus s_i$, AggVerify-2 uses a injective one way function f.

1. Att sends message m.
2. Compute $(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda, 1^n)$. Choose $K \leftarrow F.\mathsf{setup}(1^\lambda)$.
   Set $y = \mathcal{S}.\mathsf{Verify}||\mathrm{VK}||m$. Compute $K\{y\} \leftarrow F.\mathsf{puncture}(K, y)$ and $z = F(K, y)$.
   Compute $w = f(z)$, set PP = $(i\mathcal{O}(\mathsf{Transform\text{-}1}_{y,K\{y\}}), i\mathcal{O}(\mathsf{AggVerify\text{-}2}_{y,K\{y\},w}))$ and send PP, VK to Att.
3. For each signing query $x_i \neq m$, compute $\sigma_i \leftarrow \mathcal{S}.\mathsf{Sign}(\mathrm{SK}, x_i)$ and send σ_i to Att.
4. Att sends forgery σ_{agg} and n tuples $\{(\mathsf{Verify}_i, \mathrm{VK}_i, m_i)\}$. Att wins if $\exists i^* \in [n]$ such that $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}$, $\mathrm{VK}_{i^*} = \mathrm{VK}$ and $m_{i^*} = m$ and $\mathsf{AggVerify\text{-}2}_{y,K\{y\},w}(\{(\mathsf{Verify}_i, \mathrm{VK}_i, m_i)\}, \sigma_{\mathrm{agg}}) = 1$.

$\mathsf{AggVerify\text{-}2}_{y,K\{y\},w}$:

Inputs: $\{(\mathsf{Verify}_i, \mathrm{VK}_i, m_i)\}_{i=1}^n$ where $(\mathsf{Verify}_i, \mathrm{VK}_i, m_i) \in \{0,1\}^{\ell_{\mathrm{ver}}} \times \{0,1\}^{\ell_{\mathrm{vk}}} \times \{0,1\}^{\ell_{\mathrm{msg}}}$ for all $i \leq n$, $\sigma_{\mathrm{agg}} \in \{0,1\}^{\ell}$.
Constants : $y \in \{0,1\}^{\ell_{\mathrm{ver}}} \times \{0,1\}^{\ell_{\mathrm{vk}}} \times \{0,1\}^{\ell_{\mathrm{msg}}}$, $K\{y\} \in \mathcal{K}_p$, $w \in \{0,1\}^{\ell_{\mathrm{owf}}}$.

Set $\mathsf{present} = \mathsf{False}$, $\mathsf{pos} = 0$.
for all $i \leq n$ **do**
  if $\mathsf{Verify}_i||\mathrm{VK}_i||m_i = y$ **then**
    Set $\mathsf{present} = \mathsf{True}$, $\mathsf{pos} = i$.
  else
    Compute $s_i = F.\mathsf{eval}(K\{y\}, \mathsf{Verify}_i||\mathrm{VK}_i||m_i)$.
  end if
end for
if $\mathsf{present} = \mathsf{False}$ **then**
  Output 1 if $\oplus_{i=1}^n s_i = \sigma_{\mathrm{agg}}$, 0 otherwise.
else
  Output 1 if $f(\sigma_{\mathrm{agg}} \oplus_{i \neq \mathsf{pos}} s_i) = w$, 0 otherwise.
end if

Game 4: This game is exactly similar to the previous one, except that z is chosen at random.

1. Att sends message m.

2. Compute $(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda, 1^n)$. Choose $K \leftarrow F.\mathsf{setup}(1^\lambda)$.
   Set $y = \mathcal{S}.\mathsf{Verify}||\mathrm{VK}||m$. Compute $K\{y\} \leftarrow F.\mathsf{puncture}(K, y)$ and $z \leftarrow \{0,1\}^\ell$.
   Compute $w = f(z)$, set $\mathrm{PP} = (i\mathcal{O}(\mathsf{Transform\text{-}1}_{y,K\{y\}}), i\mathcal{O}(\mathsf{AggVerify\text{-}2}_{y,K\{y\},w}))$ and send PP, VK to Att.
3. For each signing query $x_i \neq m$, compute $\sigma_i \leftarrow \mathcal{S}.\mathsf{Sign}(\mathrm{SK}, x_i)$ and send σ_i to Att.
4. Att sends forgery σ_{agg} and n tuples $\{(\mathsf{Verify}_i, \mathrm{VK}_i, m_i)\}$. Att wins if $\exists i^* \in [n]$ such that $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}$, $\mathrm{VK}_{i^*} = \mathrm{VK}$ and $m_{i^*} = m$ and $\mathsf{AggVerify\text{-}2}_{y,K\{y\},w}(\{(\mathsf{Verify}_i, \mathrm{VK}_i, m_i)\}_i, \sigma_{\mathrm{agg}}) = 1$.

A.1.2 Analysis

Let $\mathsf{Adv}^j_{\mathsf{Att}}$ denote the advantage of adversary Att in Game j.

Claim A.1. Assuming $i\mathcal{O}$ is a secure indistinguishability obfuscator and $\mathcal{S}$ is a secure $(\ell_{\mathrm{ver}}, \ell_{\mathrm{vk}}, \ell_{\mathrm{msg}}, \ell_{\mathrm{sig}})$-length qualified unique signature scheme, for any PPT adversary Att,

$$\mathsf{Adv}^0_{\mathsf{Att}} - \mathsf{Adv}^1_{\mathsf{Att}} \leq \mathrm{negl}(\lambda).$$

Proof. The proof of this claim is similar to the one for Lemma 4.1. ∎

Claim A.2. Assuming $i\mathcal{O}$ is a secure indistinguishability obfuscator, for any PPT adversary Att,

$$\mathsf{Adv}^1_{\mathsf{Att}} - \mathsf{Adv}^2_{\mathsf{Att}} \leq \mathrm{negl}(\lambda).$$

Proof. In order to prove this claim, we will define an intermediate hybrid Game 1.5 which is exactly same as Game 1 and Game 2, except that the challenger sets $\mathrm{PP} = (i\mathcal{O}(\mathsf{Transform\text{-}1}_{y,K\{y\}}), i\mathcal{O}(\mathsf{AggVerify}_K))$. We will show (a) Game 1 and Game 1.5 are computationally indistinguishable, (b) Game 1.5 and Game 2 are computationally indistinguishable.

Proof of (a). Suppose there exists a PPT adversary Att such that $\mathsf{Adv}^1_{\mathsf{Att}} - \mathsf{Adv}^{1.5}_{\mathsf{Att}} = \epsilon$. We will construct a PPT algorithm $\mathcal{B}$ that constructs two circuits C_0 and C_1 with identical functionality, and uses Att to distinguish between $i\mathcal{O}(C_0)$ and $i\mathcal{O}(C_1)$, thereby breaking the security of $i\mathcal{O}$.

$\mathcal{B}$ receives m from Att, chooses $(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$ and $K \leftarrow F.\mathsf{setup}(1^\lambda)$. It sets $y = \mathcal{S}.\mathsf{Verify}||\mathrm{VK}||m$ and computes $K\{y\} \leftarrow F.\mathsf{puncture}(K, y)$. It sets $C_0 = \mathsf{Transform}'_{y,K}$ and $C_1 = \mathsf{Transform\text{-}1}_{y,K\{y\}}$, and sends C_0, C_1 to the $i\mathcal{O}$ challenger. It receives $C = i\mathcal{O}(C_b)$. $\mathcal{B}$ sets $\mathrm{PP} = (C, i\mathcal{O}(\mathsf{AggVerify}_K))$ and sends PP, VK to Att.

Note that $\mathcal{B}$ can respond to the signing queries perfectly, since it has SK. Finally, if Att wins, then $\mathcal{B}$ outputs 0, else it outputs 1. Clearly, if $C = i\mathcal{O}(C_0)$, then it corresponds to Game 1, else it corresponds to Game 1.5.

To conclude, we need to argue that C_0 and C_1 have identical functionality. This follows from the correctness property of puncturable PRFs. Note that both programs output $\perp$ if one of the input tuples is $(\mathcal{S}.\mathsf{Verify}, \mathrm{VK}, m, \sigma)$. For all other tuples $(\mathsf{Verify}', \mathrm{VK}', m', \sigma')$, $F(K, \mathsf{Verify}'||\mathrm{VK}'||m') = F.\mathsf{eval}(K\{y\}, \mathsf{Verify}'||\mathrm{VK}'||m')$. This completes the first step of our proof.

Proof of (b). The second step (showing that Game 1.5 and Game 2 are computationally indistinguishable) follows along similar lines. ∎

Claim A.3. Assuming $i\mathcal{O}$ is a secure indistinguishability obfuscator and f is an injective function, for any PPT adversary Att,

$$\mathsf{Adv}^2_{\mathsf{Att}} - \mathsf{Adv}^3_{\mathsf{Att}} \leq \mathrm{negl}(\lambda).$$

Proof. Suppose there exists a PPT adversary Att such that $\mathsf{Adv}^2_{\mathsf{Att}} - \mathsf{Adv}^3_{\mathsf{Att}} = \epsilon$. As in the previous proof, we will construct a PPT algorithm $\mathcal{B}$ that constructs two circuits C_0 and C_1 with identical functionality, and uses Att to distinguish between $i\mathcal{O}(C_0)$ and $i\mathcal{O}(C_1)$, thereby breaking the security of $i\mathcal{O}$.

The only difference between Game 2 and Game 3 is that in Game 2, circuit $\mathsf{AggVerify\text{-}1}_{y,K\{y\}.z}$ is used, while in Game 3, circuit $\mathsf{AggVerify\text{-}2}_{y,K\{y\},w}$ is used. $\mathcal{B}$ interacts with Att and receives m. It chooses

$(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$ and $K \leftarrow F.\mathsf{setup}(1^\lambda)$. Next, it computes a key punctured at $y = \mathcal{S}.\mathsf{Verify}||\mathrm{VK}||m$, i.e. $K\{y\} \leftarrow F.\mathsf{puncture}(K, y)$ and sets $z = F(K, y)$ and $w = f(z)$. Given $y, K\{y\}, z, w$, $\mathcal{B}$ can now construct circuits $C_0 = \mathsf{AggVerify\text{-}1}_{y,K\{y\},z}$ and $C_1 = \mathsf{AggVerify\text{-}2}_{y,K\{y\},w}$. $\mathcal{B}$ sends C_0 and C_1 to the $i\mathcal{O}$ challenger, and receives $C = i\mathcal{O}(C_b)$. $\mathcal{B}$ sets $\mathrm{PP} = (i\mathcal{O}(\mathsf{Transform\text{-}1}_{y,K\{y\}}), C)$ and sends PP, VK to Att.

$\mathcal{B}$ now responds to signing queries using SK. Finally Att sends forgery σ_{agg}, along with n tuples $\{(\mathsf{Verify}_i, \mathrm{VK}_i, m_i)\}$. If $C(\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}, \sigma_{\mathrm{agg}}) = 1$, output 0, else output 1. Clearly, if $C = i\mathcal{O}(C_0)$, then this corresponds to Game 2, else it corresponds to Game 3. Therefore, all that remains is to prove that C_0 and C_1 have identical functionality.

Consider any input $(\{(\mathsf{Verify}_i, \mathrm{VK}_i, m_i)\}, \sigma_{\mathrm{agg}})$. If there is no i^* such that $\mathsf{Verify}_{i^*}||\mathrm{VK}_{i^*}||m_{i^*} = y$, then both circuits check if $\sigma_{\mathrm{agg}} = \oplus_i F(K\{y\}, \mathsf{Verify}_i||\mathrm{VK}_i||m_i)$. If there exists an $i^* \in [n]$ such that $\mathsf{Verify}_{i^*}||\mathrm{VK}_{i^*}||m_{i^*} = y$, then C_0 accepts iff

$$\begin{aligned}
&(\oplus_{i \neq i^*} F(K\{y\}, \mathsf{Verify}_i||\mathrm{VK}_i||m_i)) \oplus F(K, y) = \sigma_{\mathrm{agg}} \\
&\iff (\oplus_{i \neq i^*} F(K\{y\}, \mathsf{Verify}_i||\mathrm{VK}_i||m_i)) \oplus \sigma_{\mathrm{agg}} = F(K, y) = z \\
&\iff f((\oplus_{i \neq i^*} F(K\{y\}, \mathsf{Verify}_i||\mathrm{VK}_i||m_i)) \oplus \sigma_{\mathrm{agg}}) = f(z) = w.
\end{aligned}$$

The last equivalence follows from the fact that f is an injective function. However, note that the last statement is the condition for C_1 accepting. This proves that both C_0 and C_1 have identical functionality, which proves our claim. ∎

Claim A.4. Assuming F is a puncturable PRF, for any PPT adversary Att,

$$\mathsf{Adv}^3_{\mathsf{Att}} - \mathsf{Adv}^4_{\mathsf{Att}} \leq \mathrm{negl}(\lambda).$$

Proof. Suppose there exists a PPT adversary Att such that $\mathsf{Adv}^3_{\mathsf{Att}} - \mathsf{Adv}^4_{\mathsf{Att}} = \epsilon$. We will construct a PPT algorithm $\mathcal{B}$ that uses Att to break the security of puncturable PRF (F, $F.\mathsf{setup}$, $F.\mathsf{puncture}$, $F.\mathsf{eval}$) with advantage ϵ.

First, $\mathcal{B}$ receives the message m from Att. As in Game 3 and Game 4, it computes $(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$. Next, it sends $y = \mathcal{S}.\mathsf{Verify}||\mathrm{VK}||m$ as the challenge to the PRF challenger. $\mathcal{B}$ receives a punctured key $K\{y\}$ and $z \in \{0,1\}^\ell$, where $z = F(K, y)$ or $z \leftarrow \{0,1\}^\ell$. $\mathcal{B}$ computes $w = f(z)$ and sets the public parameters $\mathrm{PP} = (i\mathcal{O}(\mathsf{Transform\text{-}1}_{y,K\{y\}}), i\mathcal{O}(\mathsf{AggVerify\text{-}2}_{y,K\{y\},w}))$. It sends PP, VK to Att.

The signing phase and forgery phase are exactly similar in Game 3 and Game 4. For each signing query x_i, $\mathcal{B}$ sends $\mathcal{S}.\mathsf{Sign}(\mathrm{SK}, x_i)$ to Att. Finally, Att outputs the forgery σ_{agg} and n tuples $\{(\mathsf{Verify}_i, \mathrm{VK}_i, m_i)\}$.

Note that if $z = F(K, y)$, then $\mathcal{B}$ simulates Game 3 perfectly. If $z \leftarrow \{0,1\}^\ell$, $\mathcal{B}$ simulates Game 4 perfectly. This concludes our proof. ∎

Claim A.5. Assuming f is a one way function, for any PPT adversary Att,

$$\mathsf{Adv}^4_{\mathsf{Att}} \leq \mathrm{negl}(\lambda).$$

Proof. Suppose there exists a PPT adversary Att such that $\mathsf{Adv}^4_{\mathsf{Att}} = \epsilon$. We will construct a PPT algorithm $\mathcal{B}$ that, using Att, inverts the one way function f with probability ϵ.

$\mathcal{B}$ receives w from the one way function challenger and m from Att. It chooses $(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$, $K \leftarrow F.\mathsf{setup}(1^\lambda)$ and computes $K\{y\} \leftarrow F.\mathsf{puncture}(K, y)$ (where $y = \mathcal{S}.\mathsf{Verify}||\mathrm{VK}||m$). It sets the public parameters $\mathrm{PP} = (i\mathcal{O}(\mathsf{Transform\text{-}1}_{y,K\{y\}}), i\mathcal{O}(\mathsf{AggVerify\text{-}2}_{y,K\{y\},w}))$ and sends PP, VK to Att.

For each signing query x_i, it computes $\mathcal{S}.\mathsf{Sign}(\mathrm{SK}, x_i)$.

Finally, $\mathcal{B}$ receives $\sigma_{\mathrm{agg}} \in \{0,1\}^\ell$ and n tuples $\{(\mathsf{Verify}_i, \mathrm{VK}_i, m_i)\}$. If $\mathsf{AggVerify\text{-}2}_{y,K\{y\},w}(\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}, \sigma_{\mathrm{agg}}) = 1$ and $\exists i^*$ such that $\mathsf{Verify}_{i^*}||\mathrm{VK}_{i^*}||m_{i^*} = y$, then $\mathcal{B}$ can successfully find an inverse for w. $\mathcal{B}$ computes $s_i = F(K, \mathsf{Verify}_i||\mathrm{VK}_i||m_i)$ for $i \neq i^*$ and sends $\sigma_{\mathrm{agg}} \oplus_{i \neq i^*} s_i$ to the one way function challenger. Clearly, if Att wins in Game 4, then $\mathcal{B}$ inverts the one way function. ∎

To conclude, it follows from the above claims that any PPT adversary has at most negligible advantage in Game 0 (assuming $i\mathcal{O}$, F and f are secure), and therefore the n-bounded aggregator described in A is selectively secure with respect to secure unique signature schemes.

B Making our VBB proof Adaptively Secure

We now show how a minor adaptiation of our selectively secure universal aggregator from VBB of Section 5 can be proven adaptively secure. The primary change is to first hash every message with an "admissible hash function" introduced by Boneh and Boyen [BB04]. From there the additonal RSA-type techniques needed fall in line with those used by Hohenberger, Sahai and Waters [HSW14].

We now describe our construction and proof. Let $\mathcal{O}$ be a virtual black-box obfuscator, $\tilde{F}$ a secure PRF with key space $\tilde{K}$, domain $\{0,1\}^{\ell_{\text{sig}}}$ and range $\{0,1\}^{\ell_{\text{rnd}}}$, PRG a secure pseudorandom generator and h a θ-admissible hash function mapping ℓ_{msg} bits to d_1 bits. Let $d_2 = d_1 + \ell_{\text{ver}} + \ell_{\text{vk}}$. Our universal signature aggregator $(\ell_{\text{ver}}, \ell_{\text{vk}}, \ell_{\text{msg}}, \ell_{\text{sig}})$-UniversalSigAgg consists of three algorithms UniversalSetup, UniversalAgg and UniversalVerify described below.

UniversalSetup(1^λ): The setup algorithm first chooses an RSA modulus N, $v \in \mathbb{Z}_N^*$ and $e \in \mathbb{Z}_{\phi(N)}^*$. Next, it chooses $2d_2$ constants $c_{i,b} \leftarrow \mathbb{Z}_{\phi(N)}$. Let $\mathbf{c} = \{c_{i,b}\}_{i\in[d_2],b\in\{0,1\}}$. It sets PP $=(\mathcal{O}(\mathsf{Transform}_{N,v,\mathbf{c}}), \mathcal{O}(\text{Transform-Image}_{N,v,\mathbf{c},e}), N, e)$, where Transform[26] and Transform-Image[27] are as follows.

$\mathsf{Transform}_{N,v,\mathbf{c}}$:

Inputs: $b \in \{0,1\}, a' \in \{0,1\}^\ell, \mathsf{Verify}' \in \{0,1\}^{\ell_{\text{ver}}}$, $\text{VK}' \in \{0,1\}^{\ell_{\text{vk}}}$, $m' \in \{0,1\}^{\ell_{\text{msg}}}$, $\sigma' \in \{0,1\}^{\ell_{\text{sig}}}$.

Constants : RSA modulus $N \in \mathbb{N}$, $v \in \mathbb{Z}_N^*$, $\mathbf{c} = \{c_{i,b}\} \in \mathbb{Z}_{\phi(N)}^{2d_2}$.

if $b = 0$ **then**
  Output $\perp$.
else if $\mathsf{Verify}'(\text{VK}', m', \sigma') = 0$ **then**
  Output $\perp$.
else
  Compute $h(m') = x$ and let $z = x||\mathsf{Verify}'||\text{VK}'$.
  Output $v^{\prod_i c_{i,z_i}} \pmod N$.
end if

$\text{Transform-Image}_{N,v,\mathbf{c},e}$:

Inputs: $\mathsf{Verify}' \in \{0,1\}^{\ell_{\text{ver}}}$, $\text{VK}' \in \{0,1\}^{\ell_{\text{vk}}}$, $m' \in \{0,1\}^{\ell_{\text{msg}}}$.
Constants : RSA modulus $N \in \mathbb{N}$, $v \in \mathbb{Z}_N^*$, $\mathbf{c} = \{c_{i,b}\} \in \mathbb{Z}_{\phi(N)}^{2d_2}$, $e \in \mathbb{Z}_{\phi(N)}^*$.

Compute $h(m') = x$ and let $z = x||\mathsf{Verify}'||\text{VK}'$.
Output $(v^{\prod c_{i,z_i}})^e \pmod N$.

UniversalAgg$(\text{PP}, \{(\mathsf{Verify}_i, \text{VK}_i, m_i, \sigma_i)\}_{i=1}^n)$: Let PP $= (P_1, P_2, N, e)$. UniversalAgg first checks if the n tuples are distinct. If not, it outputs $\perp$. Else, it computes $t_i = P_1(\mathsf{Verify}_i, \text{VK}_i, m_i, \sigma_i)$ for each $i \le n$. If $t_i = \perp$ for some i, then UniversalAgg outputs $\perp$, else it outputs $\sigma_{\text{agg}} = \prod_i t_i \pmod N$.

UniversalVerify$(\text{PP}, \{(\mathsf{Verify}_i, \text{VK}_i, m_i)\}_{i=1}^n, \sigma_{\text{agg}})$: Let PP $= (P_1, P_2, N, e)$. UniversalVerify first checks if all n tuples are distinct. If not, it outputs 0. Else, it computes, for all $i \le n$, $s_i =$ Transform-Image$(\mathsf{Verify}_i, \text{VK}_i, m_i)$. If $(\prod_i s_i) = \sigma_{\text{agg}}^e \pmod N$, it outputs 1, else it outputs 0.

[26] Padded appropriately to be of the same size as Transform-1, Transform-2, Transform-3 and Transform-4.
[27] Padded appropriately to be of the same size as Transform-Image-1 and Transform-Image-2.

B.1 Proof of Security

We will now prove that the scheme described in Section B is an adaptively secure universal signature aggregator with respect to all secure length-qualified signature schemes.

Theorem B.1. Assuming $\mathcal{O}$ is a secure virtual black-box obfuscator, F is a secure puncturable PRF, $\tilde{F}$ is a secure PRF, PRG is a secure pseudorandom generator and RSA is secure, for all $(\ell_{\text{ver}}, \ell_{\text{vk}}, \ell_{\text{msg}}, \ell_{\text{sig}})$-length qualified secure signature schemes $\mathcal{S}$, the universal signature aggregator $(\ell_{\text{ver}}, \ell_{\text{vk}}, \ell_{\text{msg}}, \ell_{\text{sig}})$-$\mathsf{UniversalSigAgg}$ is adaptively secure with respect to $\mathcal{S}$.

To prove the above theorem, we will first describe a sequence of hybrid experiments.

B.1.1 Sequence of Games

$\mathsf{Game}\ 0$ This corresponds to the adaptive security game $\mathsf{Exp}_{\mathsf{Att},\mathcal{S}}(\lambda)$ in which the challenger interacts with adversary Att.

1. Compute $(\text{SK}, \text{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$. Choose an RSA modulus N, $e \in \mathbb{Z}^*_{\phi(N)}$, $v \in \mathbb{Z}^*_N$ and $c_{i,b} \in \mathbb{Z}_{\phi(N)}$ for all $i \leq d_2, b \in \{0,1\}$. Choose $(\text{SK}, \text{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$ and set $\text{PP} = (\mathcal{O}(\mathsf{Transform}_{N,v,\mathbf{c}}), \mathcal{O}(\mathsf{Transform\text{-}Image}_{N,v,\mathbf{c},e}), N, e)$. Send (PP, VK) to Att.
2. For each signature query x_i, choose $r_i \leftarrow \{0,1\}^{\ell_{\text{rnd}}}$ and compute $\sigma_i \leftarrow \mathcal{S}.\mathsf{Sign}(\text{SK}, x_i)$. Send σ_i to Att.
3. Att sends a forgery σ_{agg} along with n tuples $\{\mathsf{Verify}_i, \text{VK}_i, m_i\}_{i=1}^n$. Att wins if
   (a) $\exists i^* \in [n]$ such that $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}$, $\text{VK}_{i^*} = \text{VK}$ and m_{i^*} was not queried during the signature phase
   (b) $\mathsf{UniversalVerify}(\text{PP}, \{\mathsf{Verify}_i, \text{VK}_i, m_i\}, \sigma_{\text{agg}}) = 1$

$\mathsf{Game}\ 1$ In this game, the challenger computes the signatures using the PRF $\tilde{F}$.

1. Compute $(\text{SK}, \text{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$. Choose an RSA modulus N, $e \in \mathbb{Z}^*_{\phi(N)}$, $v \in \mathbb{Z}^*_N$ and $c_{i,b} \in \mathbb{Z}_{\phi(N)}$ for all $i \leq d_2, b \in \{0,1\}$.
   Choose $\tilde{K} \leftarrow \tilde{F}.\mathsf{setup}(1^\lambda)$.
   Set $\text{PP} = (\mathcal{O}(\mathsf{Transform}_{N,v,\mathbf{c}}), \mathcal{O}(\mathsf{Transform\text{-}Image}_{N,v,\mathbf{c},e}), N, e)$. Send (PP, VK) to Att.
2. For each signature query x_i, choose $\rho_i \leftarrow \{0,1\}^{\ell_{\text{sig}}}$, compute $r_i \leftarrow \tilde{F}(\tilde{K}, \rho_i)$ and $\sigma_i = \mathcal{S}.\mathsf{Sign}(\text{SK}, x_i; r_i)$. Send σ_i to Att.
3. Att sends a forgery σ_{agg} along with n tuples $\{\mathsf{Verify}_i, \text{VK}_i, m_i\}_{i=1}^n$. Att wins if
   (a) $\exists i^* \in [n]$ such that $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}$, $\text{VK}_{i^*} = \text{VK}$ and m_{i^*} was not queried during the signature phase
   (b) $\mathsf{UniversalVerify}(\text{PP}, \{\mathsf{Verify}_i, \text{VK}_i, m_i\}, \sigma_{\text{agg}}) = 1$

$\mathsf{Game}\ 2$ In this game, the challenger uses program $\mathsf{Transform\text{-}1}$ to compute the public parameters PP. This program is similar to the program $\mathsf{Transform}$. However, it has the additional functionality that it allows user to receive signatures using secret key SK, provided the user has a 'trapdoor' for $\mathsf{Transform\text{-}1}$. In this game, since $\alpha \leftarrow \{0,1\}^{2\ell}$, it is unlikely that there exists a 'trapdoor'.

1. Compute $(\text{SK}, \text{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$. Choose an RSA modulus N, $e \in \mathbb{Z}^*_{\phi(N)}$, $v \in \mathbb{Z}^*_N$ and $c_{i,b} \in \mathbb{Z}_{\phi(N)}$ for all $i \leq d_2, b \in \{0,1\}$.
   Choose $\tilde{K} \leftarrow \tilde{F}.\mathsf{setup}(1^\lambda)$ and $\alpha \leftarrow \{0,1\}^{2\ell}$.
   Let $\mathsf{Transform\text{-}1}$[28] be the circuit defined below.
   Set $\text{PP} = (\mathcal{O}(\mathsf{Transform\text{-}1}_{N,v,\mathbf{c},\alpha,\text{SK},\tilde{K}}), \mathcal{O}(\mathsf{Transform\text{-}Image}_{N,v,\mathbf{c},e}), N, e)$. Send (PP, VK) to Att.

[28] Padded appropriately to be of the same size as $\mathsf{Transform}$, $\mathsf{Transform\text{-}2}$, $\mathsf{Transform\text{-}3}$ and $\mathsf{Transform\text{-}4}$.

2. For each signature query x_i, choose $\rho_i \leftarrow \{0,1\}^{\ell_{\text{sig}}}$, compute $r_i \leftarrow \tilde{F}(\tilde{K}, \rho_i)$ and $\sigma_i = \mathcal{S}.\mathsf{Sign}(\text{SK}, x_i; r_i)$. Send σ_i to Att.
3. Att sends a forgery σ_{agg} along with n tuples $\{\mathsf{Verify}_i, \text{VK}_i, m_i\}_{i=1}^n$. Att wins if
   (a) $\exists i^* \in [n]$ such that $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}$, $\text{VK}_{i^*} = \text{VK}$ and m_{i^*} was not queried during the signature phase
   (b) $\mathsf{UniversalVerify}(\text{PP}, \{\mathsf{Verify}_i, \text{VK}_i, m_i\}, \sigma_{\text{agg}}) = 1$

$\mathsf{Transform\text{-}1}_{N,v,\mathbf{c},\alpha,\text{SK},\tilde{K}}$:

Inputs: $b \in \{0,1\}, a' \in \{0,1\}^\ell, \mathsf{Verify}' \in \{0,1\}^{\ell_{\text{ver}}}$, $\text{VK}' \in \{0,1\}^{\ell_{\text{vk}}}$, $m' \in \{0,1\}^{\ell_{\text{msg}}}$, $\sigma' \in \{0,1\}^{\ell_{\text{sig}}}$.

Constants : RSA modulus $N \in \mathbb{N}$, $v \in \mathbb{Z}_N^*$, $\mathbf{c} = \{c_{i,b}\} \in \mathbb{Z}_{\phi(N)}^{2d_2}$, $\alpha \in \{0,1\}^{2\ell}$, $\text{SK} \in \mathcal{SK}$, $\tilde{K} \in \tilde{\mathcal{K}}$.

if $b = 0$ **then**
  if $\text{PRG}(a') \neq \alpha$ **then**
    Output $\perp$.
  else
    Output $(\text{VK}, \mathcal{S}.\mathsf{Sign}(\text{SK}, m'; \tilde{F}(\tilde{K}, \sigma')))$.
  end if
else if $\mathsf{Verify}'(\text{VK}', m', \sigma') = 0$ **then**
  Output $\perp$.
else
  Compute $h(m') = x$ and let $z = x||\mathsf{Verify}'||\text{VK}'$.
  Output $v^{\prod c_{i,z_i}} \pmod N$
end if

Game 3 This game is exactly similar to the previous experiment, except that the challenger chooses α such that there exists a trapdoor for program $\mathsf{Transform\text{-}1}$. For this, the challenger chooses $a \leftarrow \{0,1\}^\ell$ and sets $\alpha = \text{PRG}(a)$.

1. Compute $(\text{SK}, \text{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$. Choose an RSA modulus N, $e \in \mathbb{Z}_{\phi(N)}^*$, $v \in \mathbb{Z}_N^*$ and $c_{i,b} \in \mathbb{Z}_{\phi(N)}$ for all $i \leq d_2, b \in \{0,1\}$.
   Choose $\tilde{K} \leftarrow \tilde{F}.\mathsf{setup}(1^\lambda)$, $a \leftarrow \{0,1\}^\ell$ and set $\alpha = \text{PRG}(a)$.
   Set $\text{PP} = (\mathcal{O}(\mathsf{Transform\text{-}1}_{N,v,\mathbf{c},\alpha,\text{SK},\tilde{K}}), \mathcal{O}(\mathsf{Transform\text{-}Image}_{N,v,\mathbf{c},e}), N, e)$. Send (PP, VK) to Att.
2. For each signature query x_i, choose $\rho_i \leftarrow \{0,1\}^{\ell_{\text{sig}}}$, compute $r_i \leftarrow \tilde{F}(\tilde{K}, \rho_i)$ and $\sigma_i = \mathcal{S}.\mathsf{Sign}(\text{SK}, x_i; r_i)$. Send σ_i to Att.
3. Att sends a forgery σ_{agg} along with n tuples $\{\mathsf{Verify}_i, \text{VK}_i, m_i\}_{i=1}^n$. Att wins if
   (a) $\exists i^* \in [n]$ such that $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}$, $\text{VK}_{i^*} = \text{VK}$ and m_{i^*} was not queried during the signature phase
   (b) $\mathsf{UniversalVerify}(\text{PP}, \{\mathsf{Verify}_i, \text{VK}_i, m_i\}, \sigma_{\text{agg}}) = 1$

Game 4 In this experiment, the challenger defines a random challenge subspace of the message space. The adversary wins if all signature queries lie outside the challenge space and the message m_{i^*} corresponding to $\mathcal{S}.\mathsf{Verify}, \text{VK}$ lies in the challenge space.

1. Compute $(\text{SK}, \text{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$. Choose an RSA modulus N, $e \in \mathbb{Z}_{\phi(N)}^*$, $v \in \mathbb{Z}_N^*$ and $c_{i,b} \in \mathbb{Z}_{\phi(N)}$ for all $i \leq d_2, b \in \{0,1\}$.
   Choose $\tilde{K} \leftarrow \tilde{F}.\mathsf{setup}(1^\lambda)$, $a \leftarrow \{0,1\}^\ell$ and set $\alpha = \text{PRG}(a)$. Choose $u \leftarrow \mathsf{AdmSample}(1^\lambda, q_1 + n)$.
   Set $\text{PP} = (\mathcal{O}(\mathsf{Transform\text{-}1}_{N,v,\mathbf{c},\alpha,\text{SK},\tilde{K}}), \mathcal{O}(\mathsf{Transform\text{-}Image}_{N,v,\mathbf{c},e}), N, e)$. Send (PP, VK) to Att.

2. For each signature query x_i,
   (a) If $P_u(x_i) = 0$, flip a coin $\gamma_i \in \{0,1\}$ and abort. Att wins if $\gamma_i = 1$.
   (b) Else, choose $\rho_i \leftarrow \{0,1\}^{\ell_{\mathrm{sig}}}$, compute $r_i \leftarrow \tilde{F}(\tilde{K}, \rho_i)$ and $\sigma_i = \mathcal{S}.\mathsf{Sign}(\mathrm{SK}, x_i; r_i)$. Send σ_i to Att.
3. Att sends a forgery σ_{agg} along with n tuples $\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}_{i=1}^n$. Att wins if
   (a) $\exists i^* \in [n]$ such that $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}$, $\mathrm{VK}_{i^*} = \mathrm{VK}$ and m_{i^*} was not queried during the signature phase. Let i^* be the smallest such index. Then,
   (b) $(\forall i \neq i^*$ such that $\mathsf{Verify}_i = \mathcal{S}.\mathsf{Verify}, \mathrm{VK}_i = \mathrm{VK}, P_u(m_i) = 1)$ **and** $P_u(m_{i^*}) = 0$
   (c) $\mathsf{UniversalVerify}(\mathrm{PP}, \{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}, \sigma_{\mathrm{agg}}) = 1$

$\mathsf{Game}\ 5$ In this experiment, the challenger uses program $\mathsf{Transform\text{-}2}$ instead of $\mathsf{Transform\text{-}1}$. The only difference between $\mathsf{Transform\text{-}1}$ and $\mathsf{Transform\text{-}2}$ is that $\mathsf{Transform\text{-}2}$ rejects inputs of the form $(1, a, \mathcal{S}.\mathsf{Verify}, \mathrm{VK}, m', \sigma')$ such that $\mathcal{S}.\mathsf{Verify}(\mathrm{VK}, m', \sigma') = 1$ and m' lies in the challenge space.

1. Compute $(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$. Choose an RSA modulus N, $e \in \mathbb{Z}^*_{\phi(N)}$, $v \in \mathbb{Z}^*_N$ and $c_{i,b} \in \mathbb{Z}_{\phi(N)}$ for all $i \leq d_2, b \in \{0,1\}$.
   Choose $\tilde{K} \leftarrow \tilde{F}.\mathsf{setup}(1^\lambda)$, $a \leftarrow \{0,1\}^\ell$ and set $\alpha = \mathrm{PRG}(a)$. Choose $u \leftarrow \mathsf{AdmSample}(1^\lambda, q_1 + n)$.
   Let $\mathsf{Transform\text{-}2}$[29] be the circuit defined below.
   Set $\mathrm{PP} = (\mathcal{O}(\mathsf{Transform\text{-}2}_{u,N,v,\mathbf{c},\alpha,\mathrm{SK},\tilde{K}}), \mathcal{O}(\mathsf{Transform\text{-}Image}_{N,v,\mathbf{c},e}), N, e)$. Send $(\mathrm{PP}, \mathrm{VK})$ to Att.
2. For each signature query x_i,
   (a) If $P_u(x_i) = 0$, flip a coin $\gamma_i \in \{0,1\}$ and abort. Att wins if $\gamma_i = 1$.
   (b) Else, choose $\rho_i \leftarrow \{0,1\}^{\ell_{\mathrm{sig}}}$, compute $r_i \leftarrow \tilde{F}(\tilde{K}, \rho_i)$ and $\sigma_i = \mathcal{S}.\mathsf{Sign}(\mathrm{SK}, x_i; r_i)$. Send σ_i to Att.
3. Att sends a forgery σ_{agg} along with n tuples $\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}_{i=1}^n$. Att wins if
   (a) $\exists i^* \in [n]$ such that $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}$, $\mathrm{VK}_{i^*} = \mathrm{VK}$ and m_{i^*} was not queried during the signature phase. Let i^* be the smallest such index. Then,
   (b) $(\forall i \neq i^*$ such that $\mathsf{Verify}_i = \mathcal{S}.\mathsf{Verify}, \mathrm{VK}_i = \mathrm{VK}, P_u(m_i) = 1)$ **and** $P_u(m_{i^*}) = 0$
   (c) $\mathsf{UniversalVerify}(\mathrm{PP}, \{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}, \sigma_{\mathrm{agg}}) = 1$

[29] Padded appropriately to be of the same size as $\mathsf{Transform}$, $\mathsf{Transform\text{-}1}$, $\mathsf{Transform\text{-}3}$ and $\mathsf{Transform\text{-}4}$.

$\mathsf{Transform\text{-}2}_{u,N,v,\mathbf{c},\alpha,\mathrm{SK},\tilde{K}}$:

Inputs: $b \in \{0,1\}, a' \in \{0,1\}^{\ell}, \mathsf{Verify}' \in \{0,1\}^{\ell_{\mathrm{ver}}}$, $\mathrm{VK}' \in \{0,1\}^{\ell_{\mathrm{vk}}}$, $m' \in \{0,1\}^{\ell_{\mathrm{msg}}}$, $\sigma' \in \{0,1\}^{\ell_{\mathrm{sig}}}$.

Constants : $u \in \{0,1,\perp\}^{d_2}$, RSA modulus $N \in \mathbb{N}$, $v \in \mathbb{Z}_N^*$, $\mathbf{c} = \{c_{i,b}\} \in \mathbb{Z}_{\phi(N)}^{2d_2}$, $\alpha \in \{0,1\}^{2\ell}$, $\mathrm{SK} \in \mathcal{SK}$, $\tilde{K} \in \tilde{\mathcal{K}}$.

if $b = 0$ **then**
  if $\mathrm{PRG}(a') \neq \alpha$ **then**
    Output $\perp$.
  else
    Output $(\mathrm{VK}, \mathcal{S}.\mathsf{Sign}(\mathrm{SK}, m'; \tilde{F}(\tilde{K}, \sigma')))$.
  end if
else if $\mathsf{Verify}'(\mathrm{VK}', m', \sigma') = 0$ **then**
  Output $\perp$.
else
  Compute $h(m') = x$ and let $z = \mathsf{Verify}'||\mathrm{VK}'||x$.
  if $\mathsf{Verify}' = \mathcal{S}.\mathsf{Verify}$, $\mathrm{VK}' = \mathrm{VK}$ and $P_u(m') = 0$ **then**
    Output $\perp$.
  end if
  Output $v^{\prod c_{i,z_i}} \pmod N$.
end if

Game 6 This game is similar to the previous one, except for the manner in which the constants $c_{1,b}$ are chosen.

1. Compute $(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^{\lambda})$. Choose an RSA modulus N, $e \leftarrow \mathbb{Z}_{\phi(N)}^*$, $v \in \mathbb{Z}_N^*$.
   Choose $c'_{i,b} \in \mathbb{Z}_{\phi(N)}$ for all $i \leq d_2, b \in \{0,1\}$. Set $c_{1,b} = c'_{1,b} \cdot e^{-1}$ and $c_{i,b} = c'_{i,b}$ for all $i > 1$.
   Choose $\tilde{K} \leftarrow \tilde{F}.\mathsf{setup}(1^{\lambda})$, $a \leftarrow \{0,1\}^{\ell}$ and set $\alpha = \mathrm{PRG}(a)$. Choose $u \leftarrow \mathsf{AdmSample}(1^{\lambda}, q_1 + n)$.
   Set $\mathrm{PP} = (\mathcal{O}(\mathsf{Transform\text{-}2}_{u,N,v,\mathbf{c},\alpha,\mathrm{SK},\tilde{K}}), \mathcal{O}(\mathsf{Transform\text{-}Image}_{N,v,\mathbf{c},e}), N, e)$. Send $(\mathrm{PP}, \mathrm{VK})$ to Att.
2. For each signature query x_i,
   (a) If $P_u(x_i) = 0$, flip a coin $\gamma_i \in \{0,1\}$ and abort. Att wins if $\gamma_i = 1$.
   (b) Else, choose $\rho_i \leftarrow \{0,1\}^{\ell_{\mathrm{sig}}}$, compute $r_i \leftarrow \tilde{F}(\tilde{K}, \rho_i)$ and $\sigma_i = \mathcal{S}.\mathsf{Sign}(\mathrm{SK}, x_i; r_i)$. Send σ_i to Att.
3. Att sends a forgery σ_{agg} along with n tuples $\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}_{i=1}^n$. Att wins if
   (a) $\exists i^* \in [n]$ such that $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}$, $\mathrm{VK}_{i^*} = \mathrm{VK}$ and m_{i^*} was not queried during the signature phase. Let i^* be the smallest such index. Then,
   (b) $(\forall i \neq i^*$ such that $\mathsf{Verify}_i = \mathcal{S}.\mathsf{Verify}, \mathrm{VK}_i = \mathrm{VK}, P_u(m_i) = 1)$ **and** $P_u(m_{i^*}) = 0$
   (c) $\mathsf{UniversalVerify}(\mathrm{PP}, \{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}, \sigma_{\mathrm{agg}}) = 1$

Game 7 In this game, the challenger uses programs $\mathsf{Transform\text{-}3}$ and $\mathsf{Transform\text{-}Image\text{-}1}$.

1. Compute $(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^{\lambda})$. Choose an RSA modulus N, $e \leftarrow \mathbb{Z}_{\phi(N)}^*$, $v \in \mathbb{Z}_N^*$.
   Choose $c_{i,b} \in \mathbb{Z}_{\phi(N)}$ for all $i \leq d_2, b \in \{0,1\}$.
   Choose $\tilde{K} \leftarrow \tilde{F}.\mathsf{setup}(1^{\lambda})$, $a \leftarrow \{0,1\}^{\ell}$ and set $\alpha = \mathrm{PRG}(a)$. Choose $u \leftarrow \mathsf{AdmSample}(1^{\lambda}, q_1 + n)$.
   Let $\mathsf{Transform\text{-}3}$[30] and $\mathsf{Transform\text{-}Image\text{-}1}$[31] be the circuits defined below.

[30] Padded appropriately to be of the same size as $\mathsf{Transform}$, $\mathsf{Transform\text{-}1}$, $\mathsf{Transform\text{-}2}$ and $\mathsf{Transform\text{-}4}$.
[31] Padded appropriately to be of the same size as $\mathsf{Transform\text{-}Image}$ and $\mathsf{Transform\text{-}Image\text{-}2}$.

Set $\mathrm{PP} = (\mathcal{O}(\mathsf{Transform\text{-}3}_{u,N,v,\mathbf{c},\alpha,\mathrm{SK},\tilde{K},e^{-1}}), \mathcal{O}(\mathsf{Transform\text{-}Image\text{-}1}_{N,v,\mathbf{c}}), N, e)$ where the circuits Transform-3 and Transform-Image-1 are defined below. Send $(\mathrm{PP}, \mathrm{VK})$ to Att.

2. For each signature query x_i,
   (a) If $P_u(x_i) = 0$, flip a coin $\gamma_i \in \{0,1\}$ and abort. Att wins if $\gamma_i = 1$.
   (b) Else, choose $\rho_i \leftarrow \{0,1\}^{\ell_{\mathrm{sig}}}$, compute $r_i \leftarrow \tilde{F}(\tilde{K}, \rho_i)$ and $\sigma_i = \mathcal{S}.\mathsf{Sign}(\mathrm{SK}, x_i; r_i)$. Send σ_i to Att.
3. Att sends a forgery σ_{agg} along with n tuples $\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}_{i=1}^n$. Att wins if
   (a) $\exists i^* \in [n]$ such that $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}$, $\mathrm{VK}_{i^*} = \mathrm{VK}$ and m_{i^*} was not queried during the signature phase. Let i^* be the smallest such index. Then,
   (b) $(\forall i \neq i^*$ such that $\mathsf{Verify}_i = \mathcal{S}.\mathsf{Verify}, \mathrm{VK}_i = \mathrm{VK}, P_u(m_i) = 1)$ **and** $P_u(m_{i^*}) = 0$
   (c) $\mathsf{UniversalVerify}(\mathrm{PP}, \{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}, \sigma_{\mathrm{agg}}) = 1$

$\mathsf{Transform\text{-}3}_{u,N,v,\mathbf{a},\alpha,\mathrm{SK},\tilde{K},e^{-1}}$:

Inputs: $b \in \{0,1\}, a \in \{0,1\}^\ell, \mathsf{Verify}' \in \{0,1\}^{\ell_{\mathrm{ver}}}$, $\mathrm{VK}' \in \{0,1\}^{\ell_{\mathrm{vk}}}$, $m' \in \{0,1\}^{\ell_{\mathrm{msg}}}$, $\sigma' \in \{0,1\}^{\ell_{\mathrm{sig}}}$.

Constants : $u \in \{0,1,\perp\}^{d_2}$, RSA modulus $N \in \mathbb{N}$, $v \in \mathbb{Z}_N^*$, $\mathbf{c} = \{a_{i,b}\} \in \mathbb{Z}_N^{2d_2}$, $\alpha \in \{0,1\}^{2\ell}$, $\mathrm{SK} \in \mathcal{SK}$, $\tilde{K} \in \tilde{\mathcal{K}}$, $e^{-1} \in \mathbb{Z}_{\phi(N)}^*$.

if $b = 0$ **then**
  if $\mathrm{PRG}(a) \neq \alpha$ **then**
    Output $\perp$.
  else
    Output $(\mathrm{VK}, \mathcal{S}.\mathsf{Sign}(\mathrm{SK}, m'; \tilde{F}(\tilde{K}, \sigma')))$.
  end if
else if $\mathsf{Verify}'(\mathrm{VK}', m', \sigma') = 0$ **then**
  Output $\perp$.
else
  Compute $h(m') = x$ and let $z = x||\mathsf{Verify}'||\mathrm{VK}'$.
  if $\mathsf{Verify}' = \mathcal{S}.\mathsf{Verify}$ and $\mathrm{VK}' = \mathrm{VK}$ and $P_u(m') = 0$ **then**
    Output $\perp$.
  end if
  Output $v^{(\prod_i c_{i,z_i}) \cdot e^{-1}} \pmod N$.
end if

$\mathsf{Transform\text{-}Image\text{-}1}_{N,v,\mathbf{c}}$:

Inputs: $\mathsf{Verify}' \in \{0,1\}^{\ell_{\mathrm{ver}}}$, $\mathrm{VK}' \in \{0,1\}^{\ell_{\mathrm{vk}}}$, $m' \in \{0,1\}^{\ell_{\mathrm{msg}}}$.
Constants : RSA modulus $N \in \mathbb{N}$, $v \in \mathbb{Z}_N^*$, $\mathbf{c} = \{c_{i,b}\} \in \mathbb{Z}_N^{2d_2}$.

Compute $h(m') = x$ and let $z = x||\mathsf{Verify}'||\mathrm{VK}'$.
Output $v^{\prod_i c_{i,z_i}} \pmod N$.

Game 8 In this game, the challenger modifies the manner in which constants $c_{i,b}$ are chosen. Instead of choosing them uniformly at random from $\mathbb{Z}_{\phi(N)}$, the challenger now chooses $a_{i,b} \leftarrow \mathbb{Z}_{\phi(N)}$ and sets $c_{i,b}$ appropriately, depending on u and $y = \mathcal{S}.\mathsf{Verify}||\mathrm{VK}$.

1. Compute $(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$. Choose an RSA modulus N, $e \leftarrow \mathbb{Z}_{\phi(N)}^*$, $v \in \mathbb{Z}_N^*$.
   Choose $\tilde{K} \leftarrow \tilde{F}.\mathsf{setup}(1^\lambda)$, $a \leftarrow \{0,1\}^\ell$ and set $\alpha = \mathrm{PRG}(a)$. Choose $u \leftarrow \mathsf{AdmSample}(1^\lambda, q_1 + n)$.
   Choose $a_{i,b} \in \mathbb{Z}_N$ for all $i \leq d_2, b \in \{0,1\}$. Let $y = \mathcal{S}.\mathsf{Verify}||\mathrm{VK}$.

For $i \leq d_1$, set $c_{i,b} = e \cdot a_{i,b} \mod \phi(N)$ if $u_i = b$, else $c_{i,b} = e \cdot a_{i,b} + 1 \mod \phi(N)$.
For $i > d_1$, set $c_{i,b} = e \cdot a_{i,b} \mod \phi(N)$ if $y_{i-d_1} \neq b$, else $c_{i,b} = e \cdot a_{i,b} + 1 \mod \phi(N)$.
Set $\mathrm{PP} = (\mathcal{O}(\mathsf{Transform\text{-}3}_{u,N,v,\mathbf{c},\alpha,\mathrm{SK},\tilde{K},e^{-1}}), \mathcal{O}(\mathsf{Transform\text{-}Image\text{-}1}_{N,v,\mathbf{c}}), N, e)$. Send $(\mathrm{PP}, \mathrm{VK})$ to Att.

2. For each signature query x_i,
   (a) If $P_u(x_i) = 0$, flip a coin $\gamma_i \in \{0,1\}$ and abort. Att wins if $\gamma_i = 1$.
   (b) Else, choose $\rho_i \leftarrow \{0,1\}^{\ell_{\mathrm{sig}}}$, compute $r_i \leftarrow \tilde{F}(\tilde{K}, \rho_i)$ and $\sigma_i = \mathcal{S}.\mathsf{Sign}(\mathrm{SK}, x_i; r_i)$. Send σ_i to Att.
3. Att sends a forgery σ_{agg} along with n tuples $\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}_{i=1}^n$. Att wins if
   (a) $\exists i^* \in [n]$ such that $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}$, $\mathrm{VK}_{i^*} = \mathrm{VK}$ and m_{i^*} was not queried during the signature phase. Let i^* be the smallest such index. Then,
   (b) $(\forall i \neq i^*$ such that $\mathsf{Verify}_i = \mathcal{S}.\mathsf{Verify}, \mathrm{VK}_i = \mathrm{VK}, P_u(m_i) = 1)$ **and** $P_u(m_{i^*}) = 0$
   (c) $\mathsf{UniversalVerify}(\mathrm{PP}, \{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}, \sigma_{\mathrm{agg}}) = 1$

Game 9 This game is similar to the previous one, except that the constants $c_{i,b}$ are computed within the program $\mathsf{Transform}$-4 (which is used instead of $\mathsf{Transform}$-3). Similarly, program $\mathsf{Transform\text{-}Image}$-2 is used instead of $\mathsf{Transform\text{-}Image}$-1.

1. Compute $(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$. Choose an RSA modulus N, $e \leftarrow \mathbb{Z}^*_{\phi(N)}$, $v \in \mathbb{Z}^*_N$.
Choose $\tilde{K} \leftarrow \tilde{F}.\mathsf{setup}(1^\lambda)$, $a \leftarrow \{0,1\}^\ell$ and set $\alpha = \mathrm{PRG}(a)$. Choose $u \leftarrow \mathsf{AdmSample}(1^\lambda, q_1 + n)$.
Choose $a_{i,b} \in \mathbb{Z}_N$ for all $i \leq d_2, b \in \{0,1\}$. Let $\mathbf{a} = \{a_{i,b}\}$ and $y = \mathcal{S}.\mathsf{Verify}||\mathrm{VK}$.
Let $\mathsf{Transform}$-4[32] and $\mathsf{Transform\text{-}Image}$-2[33] be the circuits defined below.
Set $\mathrm{PP} = (\mathcal{O}(\mathsf{Transform\text{-}4}_{u,N,v,\mathbf{a},\alpha,\mathrm{SK},\tilde{K},e,y}), \mathcal{O}(\mathsf{Transform\text{-}Image\text{-}2}_{N,v,\mathbf{a},e,y}), N, e)$ where $\mathsf{Transform}$-4 and $\mathsf{Transform\text{-}Image}$-2 are defined below. Send $(\mathrm{PP}, \mathrm{VK})$ to Att.
2. For each signature query x_i,
   (a) If $P_u(x_i) = 0$, flip a coin $\gamma_i \in \{0,1\}$ and abort. Att wins if $\gamma_i = 1$.
   (b) Else, choose $\rho_i \leftarrow \{0,1\}^{\ell_{\mathrm{sig}}}$, compute $r_i \leftarrow \tilde{F}(\tilde{K}, \rho_i)$ and $\sigma_i = \mathcal{S}.\mathsf{Sign}(\mathrm{SK}, x_i; r_i)$. Send σ_i to Att.
3. Att sends a forgery σ_{agg} along with n tuples $\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}_{i=1}^n$. Att wins if
   (a) $\exists i^* \in [n]$ such that $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}$, $\mathrm{VK}_{i^*} = \mathrm{VK}$ and m_{i^*} was not queried during the signature phase. Let i^* be the smallest such index. Then,
   (b) $(\forall i \neq i^*$ such that $\mathsf{Verify}_i = \mathcal{S}.\mathsf{Verify}, \mathrm{VK}_i = \mathrm{VK}, P_u(m_i) = 1)$ **and** $P_u(m_{i^*}) = 0$
   (c) $\mathsf{UniversalVerify}(\mathrm{PP}, \{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}, \sigma_{\mathrm{agg}}) = 1$

[32] Padded appropriately to be of the same size as $\mathsf{Transform}$, $\mathsf{Transform}$-1, $\mathsf{Transform}$-2 and $\mathsf{Transform}$-3.
[33] Padded appropriately to be of the same size as $\mathsf{Transform\text{-}Image}$ and $\mathsf{Transform\text{-}Image}$-1.

$\mathsf{Transform\text{-}4}_{u,N,v,\mathbf{a},\alpha,\mathrm{SK},\tilde{K},e}$:

Inputs: $b \in \{0,1\}, a \in \{0,1\}^{\ell}, \mathsf{Verify}' \in \{0,1\}^{\ell_{\mathrm{ver}}}$, $\mathrm{VK}' \in \{0,1\}^{\ell_{\mathrm{vk}}}$, $m' \in \{0,1\}^{\ell_{\mathrm{msg}}}$, $\sigma' \in \{0,1\}^{\ell_{\mathrm{sig}}}$.

Constants : $u \in \{0,1,\perp\}^{d_2}$, RSA modulus $N \in \mathbb{N}$, $v \in \mathbb{Z}_N^*$, $\mathbf{a} = \{a_{i,b}\} \in \mathbb{Z}_N^{2d_2}$, $\alpha \in \{0,1\}^{2\ell}$, $\mathrm{SK} \in \mathcal{SK}$, $\tilde{K} \in \tilde{\mathcal{K}}$, $e \in \mathbb{Z}_{\phi(N)}^*$, $y \in \{0,1\}^{\ell_{\mathrm{ver}}} \times \{0,1\}^{\ell_{\mathrm{vk}}}$.

Let $y' = \mathsf{Verify}'||\mathrm{VK}'$.
if $b = 0$ **then**
  if $\mathrm{PRG}(a) \neq \alpha$ **then**
    Output $\perp$.
  else
    Output $(\mathrm{VK}, \mathcal{S}.\mathsf{Sign}(\mathrm{SK}, m'; \tilde{F}(\tilde{K}, \sigma')))$.
  end if
else if $\mathsf{Verify}'(\mathrm{VK}', m', \sigma') = 0$ **then**
  Output $\perp$.
else
  Compute $h(m') = x$ and let $z = x||\mathsf{Verify}'||\mathrm{VK}'$.
  if $\mathsf{Verify}' = \mathcal{S}.\mathsf{Verify}$ and $\mathrm{VK}' = \mathrm{VK}$ and $P_u(m') = 0$ **then**
    Output $\perp$.
  end if
  if $y = y'$ **then**
    Let i' be the first index such that $u_{i'} = x_{i'}$. Set $c_{i',x_{i'}} = a_{i',x_{i'}}$.
    $\forall i \leq d_1, i \neq i'$, set $c_{i,b} = e \cdot a_{i,b}$ if $u_i = b$, else $c_{i,b} = e \cdot a_{i,b} + 1$.
    $\forall i > d_1$, set $c_{i,b} = e \cdot a_{i,b}$ if $y_{i-d_1} \neq b$, else $c_{i,b} = e \cdot a_{i,b} + 1$.
  else
    Let i' be the first index such that $y_{i'} \neq y'_{i'}$. Set $c_{d_1+i',y'_{i'}} = a_{d_1+i',y'_{i'}}$.
    $\forall i \leq d_1$, set $c_{i,b} = e \cdot a_{i,b}$ if $u_i = b$, else $c_{i,b} = e \cdot a_{i,b} + 1$.
    $\forall i > d_1, i - d_1 \neq i'$, set $c_{i,b} = e \cdot a_{i,b}$ if $y_{i-d_1} \neq b$, else $c_{i,b} = e \cdot a_{i,b} + 1$.
  end if
  Output $v^{\prod c_{i,z_i}} \pmod N$.
end if

$\mathsf{Transform\text{-}Image\text{-}2}_{N,v,\mathbf{a},e}$:

Inputs: $\mathsf{Verify}' \in \{0,1\}^{\ell_{\mathrm{ver}}}$, $\mathrm{VK}' \in \{0,1\}^{\ell_{\mathrm{vk}}}$, $m' \in \{0,1\}^{\ell_{\mathrm{msg}}}$.
Constants : RSA modulus $N \in \mathbb{N}$, $v \in \mathbb{Z}_N^*$, $\mathbf{a} = \{a_{i,b}\} \in \mathbb{Z}_N^{2d_2}$, $e \in \mathbb{Z}_{\phi(N)}^*$, $y \in \{0,1\}^{\ell_{\mathrm{ver}}} \times \{0,1\}^{\ell_{\mathrm{vk}}}$.

Compute $h(m') = x$ and set $z = x||\mathsf{Verify}'||\mathrm{VK}'$.
$\forall i \leq d_1$, set $c_{i,b} = e \cdot a_{i,b}$ if $u_i = b$, else $c_{i,b} = e \cdot a_{i,b} + 1$.
$\forall i > d_1$, set $c_{i,b} = e \cdot a_{i,b}$ if $y_{i-d_1} \neq b$, else $c_{i,b} = e \cdot a_{i,b} + 1$.
Output $v^{\prod c_{i,x_i}} \pmod N$.

B.1.2 Adversary's Advantage in the Games

Let $\mathsf{Adv}^{j}_{\mathsf{Att}}$ denote the advantage of adversary Att in Game j.

Claim B.1. Assuming $\tilde{F}$ is a secure PRF, for any PPT adversary Att,

$$\mathsf{Adv}^0_{\mathsf{Att}} - \mathsf{Adv}^1_{\mathsf{Att}} \leq \mathrm{negl}(\lambda).$$

Proof. Similar to proof of Claim 5.1. ∎

Claim B.2. Assuming $\mathcal{O}$ is a secure indistinguishability obfuscator, for any PPT adversary Att,

$$\mathsf{Adv}^1_{\mathsf{Att}} - \mathsf{Adv}^2_{\mathsf{Att}} \leq \mathrm{negl}(\lambda).$$

Proof. Similar to the proof of Claim 5.2. ∎

Claim B.3. Assuming PRG is a secure pseudorandom generator, for any PPT adversary Att,

$$\mathsf{Adv}^2_{\mathsf{Att}} - \mathsf{Adv}^3_{\mathsf{Att}} \leq \mathrm{negl}(\lambda).$$

Proof. Similar to proof of Claim 5.3. ∎

Claim B.4. For any adversary Att,

$$\mathsf{Adv}^4_{\mathsf{Att}} \geq \mathsf{Adv}^3_{\mathsf{Att}}/\theta(q_1 + n).$$

Proof. This follows from the θ-admissibility of h. Let $\mathcal{I} = \{i' : \mathsf{Verify}_{i'} = \mathcal{S}.\mathsf{Verify}, \mathrm{VK}_{i'} = \mathrm{VK}\}$. Let $y_i = h(x_i)$ for all $i \leq q_1$, and $y_{q_1+j} = h(m_j)$ for all $j \in \mathcal{I}$. Note that $m_{i^*} \neq x_i$ for all $i \leq q_1$, and $m_{i^*} \neq m_j$ for all $j \in \mathcal{I}, j \neq i^*$. Therefore,

$$\Pr[\forall i \leq q_1, P_u(x_i) = 1 \textbf{ and } \forall j \in \mathcal{I}, j \neq i^* P_u(m_j) = 1 \textbf{ and } P_u(m_{i^*}) = 0] \geq 1/\theta(q_1 + n)$$

where the probability is only over the choice of $u \leftarrow \mathsf{AdmSample}(1^\lambda, q_1 + n)$. ∎

Claim B.5. Assuming $\mathcal{O}$ is a secure virtual black box obfuscator, $\tilde{F}$ is a secure pseudorandom function, PRG is a secure pseudorandom generator and $\mathcal{S}$ is a $(\ell_{\mathrm{ver}}, \ell_{\mathrm{vk}}, \ell_{\mathrm{msg}}, \ell_{\mathrm{sig}})$-length qualified secure signature scheme,

$$\mathsf{Adv}^4_{\mathsf{Att}} - \mathsf{Adv}^5_{\mathsf{Att}} \leq \mathrm{negl}(\lambda).$$

Proof. The proof of this claim is along the lines of the proof of Claim 5.1. Given only oracle access to either $\mathsf{Transform\text{-}1}_{N,v,\mathbf{c},\alpha,\mathrm{SK},\tilde{K}}$ or $\mathsf{Transform\text{-}2}_{u,N,v,\mathbf{c},\alpha,\mathrm{SK},\tilde{K}}$, the only way in which an adversary S can distinguish between the two is by sending a query of the form $(1, a', \mathsf{Verify}', \mathrm{VK}', m', \sigma')$ such that $\mathsf{Verify}' = \mathcal{S}.\mathsf{Verify}$, $\mathrm{VK}' = \mathrm{VK}$, $\mathsf{Verify}'(\mathrm{VK}', m', \sigma') = 1$ and $P_u(m') = 0$. Note that m' was not queried during the signature phase, since $P_u(m') = 0$. This implies (m', σ') is a valid forgery, thereby breaking the signature scheme $\mathcal{S}$. ∎

Claim B.6. For any adversary Att,

$$\mathsf{Adv}^5_{\mathsf{Att}} = \mathsf{Adv}^6_{\mathsf{Att}}.$$

Proof. The only difference between Game 5 and Game 6 is the choice of $c_{1,b}$. In Game 5, $c_{1,b} \leftarrow \mathbb{Z}_{\phi(N)}$, while in Game 6, $c'_{1,b} \leftarrow \mathbb{Z}_{\phi(N)}$ and $c_{1,b} = c'_{1,b} \cdot e^{-1} \mod \phi(N)$. However, the distributions $\mathcal{D}_1 = \{(c, e) | c \leftarrow \mathbb{Z}_{\phi(N)}\}$ and $\mathcal{D}_2 = \{(c \cdot e^{-1} \mod \phi(N), e) | c \leftarrow \mathbb{Z}_{\phi(N)}\}$ are identical, which implies that Game 5 and Game 6 are identical. ∎

Claim B.7. Assuming $\mathcal{O}$ is a secure indistinguishability obfuscator, for any PPT adversary Att,

$$\mathsf{Adv}^6_{\mathsf{Att}} - \mathsf{Adv}^7_{\mathsf{Att}} \leq \mathrm{negl}(\lambda).$$

Proof. Let $(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$, $\tilde{K} \leftarrow \tilde{F}.\mathsf{setup}(1^\lambda)$, $a \leftarrow \{0,1\}^\ell$, $\alpha = \mathrm{PRG}(a)$ and $u \leftarrow \mathsf{AdmSample}(1^\lambda, q_1 + n)$. Choose an RSA modulus N, let $e \leftarrow \mathbb{Z}^*_{\phi(N)}$, $v \leftarrow \mathbb{Z}^*_N$ and $c'_{i,b} \leftarrow \mathbb{Z}_{\phi(N)}$. Let $\mathbf{c} = \{c_{i,b}\}$ where $c_{1,b} = c'_{1,b} \cdot e^{-1}$ and $c_{i,b} = c'_{i,b}$ for all $i > 1$. Let $\tilde{\mathbf{c}} = \{\tilde{c}_{i,b}\}$ where $\tilde{c}_{i,b} = c'_{i,b}$ for all i.

To prove this claim, it suffices to show that $\mathsf{Transform\text{-}2}_{u,N,v,\mathbf{c},\alpha,\mathrm{SK},\tilde{K}}$ and $\mathsf{Transform\text{-}3}_{u,N,v,\tilde{\mathbf{c}},\alpha,\mathrm{SK},\tilde{K},e^{-1}}$ are functionally identical. Let us consider the behavior of the two programs on input $(b, a', \mathsf{Verify}', \mathrm{VK}', m', \sigma')$. Let $x' = h(m')$. The only case where $\mathsf{Transform\text{-}2}$ and $\mathsf{Transform\text{-}3}$ can possibly differ is when $b = 1$, $\mathsf{Verify}'(\mathrm{VK}', m', \sigma') = 1$, $\mathsf{Verify}' = \mathcal{S}.\mathsf{Verify}$, $\mathrm{VK}' = \mathrm{VK}$ and $P_u(m') = 1$. $\mathsf{Transform\text{-}2}$ outputs $v^{\prod c_{i,z'_i}} \pmod N$ while $\mathsf{Transform\text{-}3}$ outputs $v^{(\prod \tilde{c}_{i,z'_i}) \cdot e^{-1}} \pmod N$. However, note that $\prod_i c_{i,z'_i} = (\prod_i \tilde{c}_{i,z'_i}) \cdot e^{-1}$ since $c_{1,b} = c'_{1,b} \cdot e^{-1} = \tilde{c}_{1,b}$. This concludes our proof. ∎

Claim B.8. For any adversary Att,

$$\mathsf{Adv}^7_{\mathsf{Att}} - \mathsf{Adv}^8_{\mathsf{Att}} \leq \mathrm{negl}(\lambda).$$

Proof. Let us consider the two distributions $\mathcal{D}_1 = \{a | a \leftarrow \mathbb{Z}_{\phi(N)}\}$ and $\mathcal{D}_2 = \{a \mod \phi(N) | a \leftarrow \mathbb{Z}_N\}$. The statistical distance between $\mathcal{D}_1$ and $\mathcal{D}_2$ is $(p + q - 1)/N$, where $N = pq$. Since $p, q \in \Theta(2^\lambda)$, the statistical distance between $\mathcal{D}_1$ and $\mathcal{D}_2$ is negligible in λ.

Next, note that the distributions $\mathcal{D}'_1 = \{(a, e) | a \leftarrow \mathbb{Z}_{\phi(N)}\}$, $\mathcal{D}'_2 = \{(a \cdot e \mod \phi(N), e) | a \leftarrow \mathbb{Z}_{\phi(N)}\}$ and $\mathcal{D}'_3 = \{(a \cdot e + 1 \mod \phi(N), e) | a \leftarrow \mathbb{Z}_{\phi(N)}\}$ are identical. As a result, Game 6 and Game 7 are statistically indistinguishable. ∎

Claim B.9. Assuming $\mathcal{O}$ is a secure indistinguishability obfuscator, for any PPT adversary Att,

$$\mathsf{Adv}^8_{\mathsf{Att}} - \mathsf{Adv}^9_{\mathsf{Att}} \leq \mathrm{negl}(\lambda).$$

Proof. Let N be an RSA modulus, $e \leftarrow \mathbb{Z}^*_{\phi(N)}$, $a_{i,b} \leftarrow \mathbb{Z}_N$ and $u \leftarrow \mathsf{AdmSample}(1^\lambda, q_1 + n)$, $a \leftarrow \{0,1\}^\ell$, $\alpha = \mathrm{PRG}(a)$, $(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$ and $\tilde{K} \leftarrow \tilde{F}.\mathsf{setup}(1^\lambda)$. Let $c_{i,b} = e \cdot a_{i,b} \pmod \phi)(N)$ if $u_i = b$, else $c_{i,b} = e \cdot a_{i,b} + 1 \pmod \phi)(N)$. In order to prove this claim, it suffices to prove that the programs $\mathsf{Transform\text{-}3}_{u,N,v,\mathbf{c},\alpha,\mathrm{SK},\tilde{K},e^{-1}}$ and $\mathsf{Transform\text{-}4}_{u,N,v,\mathbf{a},\alpha,\mathrm{SK},\tilde{K},e}$ are functionally identical, and similarly, the programs $\mathsf{Transform\text{-}Image\text{-}1}_{N,v,\mathbf{c}}$ and $\mathsf{Transform\text{-}Image\text{-}1}_{N,v,\mathbf{a},e}$ are functionally identical. We will use the following observation.

Observation B.1. Let $v \in \mathbb{Z}^*_N$, $w_i \in \mathbb{Z}$ for $i \leq n$. Let $w' = w \mod \phi(N)$ Then $v^{\prod_i w_i} = v^{\prod_i w'_i}$.

This follows from the fact that $v^{\phi(N)} = 1$.

Let us first consider the circuits $\mathsf{Transform\text{-}3}_{u,N,v,\mathbf{c},\alpha,\mathrm{SK},\tilde{K},e^{-1}}$ and $\mathsf{Transform\text{-}4}_{u,N,v,\mathbf{c},\alpha,\mathrm{SK},\tilde{K},e}$. On input $(b, a', \mathsf{Verify}', \mathrm{VK}', m', \sigma')$, the only case $\mathsf{Transform\text{-}3}$ and $\mathsf{Transform\text{-}4}$ can possibly differ is when $b = 1$, $\mathsf{Verify}'(\mathrm{VK}', m', \sigma') = 1$, and either $\mathsf{Verify}'||\mathrm{VK}' \neq \mathcal{S}.\mathsf{Verify}||\mathrm{VK}$ or $P_u(m') = 1$. Let $y = \mathcal{S}.\mathsf{Verify}||\mathrm{VK}$, $y' = \mathsf{Verify}'||\mathrm{VK}'$ and $z' = h(m')||\mathsf{Verify}'||\mathrm{VK}'$. Either there exists an index i' such that $u_{i'} = h(m')_{i'}$, in which case set j' to be the first such index, or there exists an index i' such that $y_{i'} \neq y'_{i'}$, in which case set $j' = d_1 + i'$. Note that $c_{j',z_{j'}} = e \cdot a_{j',z'_{j'}}$.

$$\begin{aligned}
&\mathsf{Transform\text{-}3}_{u,N,v,\mathbf{c},\alpha,\mathrm{SK},\tilde{K},e^{-1}}(\mathsf{Verify}', \mathrm{VK}', m', \sigma') \\
=& v^{(\prod c_{j,z'_j}) \cdot e^{-1}} \\
=& v^{(\prod_{j \neq j'} c_{j,z'_j}) \cdot c_{j',z'_{j'}} \cdot e^{-1}} \\
=& v^{(\prod_{j \neq j'} c_{j,z'_j}) \cdot a_{j',z'_{j'}} \mod \phi(N)} \\
=& \mathsf{Transform\text{-}4}_{u,N,v,\mathbf{a},\alpha,\mathrm{SK},\tilde{K},e}(\mathsf{Verify}', \mathrm{VK}', m', \sigma')
\end{aligned}$$

where the last step follows from Observation B.1.

Let us now consider Transform-Image-1 and Transform-Image-2. This case follows directly from Observation B.1, since the only difference between the two programs is that Transform-Image-$1_{N,v,\mathbf{c}}$ has $\mathbf{c}$ hardwired, while in Transform-Image-$2_{N,v,\mathbf{a},e}$, $\mathbf{a}$ is hardwired.

■

Claim B.10. Assuming RSA is secure, for any PPT adversary Att,

$$\mathsf{Adv}^{9}_{\mathsf{Att}} \leq \mathrm{negl}(\lambda).$$

Proof. Suppose there exists a PPT adversary Att such that $\mathsf{Adv}^{9}_{\mathsf{Att}} = \epsilon$. We will construct a PPT algorithm $\mathcal{B}$ that breaks the RSA assumption with advantage ϵ.

$\mathcal{B}$ receives as input N, e, v. It chooses $(\mathrm{SK}, \mathrm{VK}) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\lambda)$. It chooses $\tilde{K} \leftarrow \tilde{F}.\mathsf{setup}(1^\lambda)$, $a \leftarrow \{0,1\}^\ell$, sets $\alpha = \mathrm{PRG}(a)$. It chooses $u \leftarrow \mathsf{AdmSample}(1^\lambda, q_1{+}n)$, $a_{i,b} \leftarrow \mathbb{Z}_N$. It sends $\mathrm{PP} = (\mathcal{O}(\text{Transform-}4_{N,v,\mathbf{a},\alpha,\mathrm{SK},\tilde{K},e}), \mathcal{O}(\text{Transform-Image-}2_{N,v,\mathbf{a},e}), e)$ and VK to Att.

For each signing query x_i, $\mathcal{B}$ checks that $P_u(x_i) = 1$ and sends $\sigma_i = \mathcal{S}.\mathsf{Sign}(\mathrm{SK}, x_i)$ to Att.

Finally, Att outputs a forgery σ_{agg} along with n tuples $\{\mathsf{Verify}_i, \mathrm{VK}_i, m_i\}$. Let $z_i = h(m_i)||\mathcal{S}.\mathsf{Verify}||\mathrm{VK}$, $y = \mathcal{S}.\mathsf{Verify}||\mathrm{VK}$, $y_i = \mathsf{Verify}'||\mathrm{VK}'$ and $\mathcal{I} = \{i | \mathsf{Verify}_i = \mathcal{S}.\mathsf{Verify}, \mathrm{VK}_i = \mathrm{VK}\}$.

If Att wins, then $\exists i^* \in [n]$ such that $\mathsf{Verify}_{i^*} = \mathcal{S}.\mathsf{Verify}$, $\mathrm{VK}_{i^*} = \mathrm{VK}$, m_{i^*} was not queried during the signature phase, $P_u(m_{i^*}) = 0$, for all $i \in \mathcal{I}, i \neq i^*$ $P_u(m_i) = 1$, and $\sigma^e_{\mathrm{agg}} = \prod_i \text{Transform-Image-}2_{N,v,\mathbf{a},e}(\mathsf{Verify}_i, \mathrm{VK}_i, m_i)$.

Let us consider Transform-Image-$2_{N,v,\mathbf{a},e}(\mathsf{Verify}_i, \mathrm{VK}_i, m_i)$ for some $i \neq i^*$. For $j \leq d_1$, let $c_{j,b} = e \cdot a_{j,b}$ if $u_j = b$, else $c_{j,b} = e \cdot a_{j,b} + 1$. For $j > d_1$, let $c_{j,b} = e \cdot a_{j,b}$ if $y_{j-d_1} \neq b$, else $c_{j,b} = e \cdot a_{j,b} + 1$. If $y = y'$, let $j' \in [d_1]$ be the first index such that $u_{j'} = z_{i,j'}$. Else, let j'' be the first index such that $y_{j''} \neq y_{i,j''}$ and set $j' = d_1 + j''$. Then,

$$\begin{aligned}
&\text{Transform-Image-}2_{N,v,\mathbf{a},e}(\mathsf{Verify}_i, \mathrm{VK}_i, m_i)\\
=&v^{\prod c_{j,z_{i,j}}} \pmod N\\
=&v^{(\prod_{j\neq j'} c_{j,z_{i,j}})\cdot e\cdot a_{j',z_{i,j'}}} \pmod N\\
=&v^{e\cdot\tau_i} \pmod N \text{ where } \tau_i = a_{j',z_{i,j'}} \cdot (\prod_{j\neq j'} c_{j,z_{i,j}}).
\end{aligned}$$

On the other hand, if we consider the term corresponding to i^*, then

$$\begin{aligned}
&\text{Transform-Image-}2_{N,v,\mathbf{c},e}\\
&= v^{\prod_j c_{j,z_{i^*,j}}} \pmod N\\
&= v^{\prod_j (e\cdot a_{j,z_{i^*,j}}+1)} \pmod N\\
&= v \cdot v^{e\cdot\tau_{i^*}} \pmod N \text{ for some } t_{i^*} \text{ that can be efficiently computed using } e, a_{i,b}.
\end{aligned}$$

Hence, $\sigma^e_{\mathrm{agg}} = v \cdot (v^{\sum \tau_i})^e \pmod N$. $\mathcal{B}$ finally outputs $x = \sigma_{\mathrm{agg}}/(v^{\sum \tau_i}) \pmod N$, and wins with advantage ϵ.

■

Therefore, assuming $\mathcal{O}$ is a secure VBB obfuscator for class $\mathcal{C}$, F is a selectively secure puncturable PRF, $\tilde{F}$ is a secure PRF, PRG is a secure pseudorandom generator and the RSA assumption holds, the construction described in Section B is adaptive secure with respect to all length-qualified signature schemes.