Fully Secure and Fast Signing from Obfuscation

Kim Ramchen
University of Texas at Austin
kramchen@cs.utexas.edu

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

Abstract

In this work we explore new techniques for building short signatures from obfuscation. Our goals are twofold. First, we would like to achieve short signatures with adaptive security proofs. Second, we would like to build signatures with fast signing, ideally significantly faster than comparable signatures that are not based on obfuscation. The goal here is to create an "imbalanced" scheme where signing is fast at the expense of slower verification.

We develop new methods for achieving short and fully secure obfuscation-derived signatures. Our base signature scheme is built from punctured programming and makes a novel use of the "prefix technique" to guess a signature. We find that our initial scheme has slower performance than comparable algorithms (e.g. EC-DSA). We find that the underlying reason is that the underlying PRG is called $\approx \ell^2$ times for security parameter ℓ.

To address this issue we construct a more efficient scheme by adapting the Goldreich-Goldwasser-Micali [GGM86] construction to form the basis for a new puncturable PRF. This puncturable PRF accepts variable-length inputs and has the property that evaluations on all prefixes of a message can be efficiently pipelined. Calls to the puncturable PRF by the signing algorithm therefore make fewer invocations of the underlying PRG, resulting in reduced signing costs.

We evaluate our puncturable PRF based signature schemes using a variety of cryptographic candidates for the underlying PRG. We show that the resulting performance on message signing is competitive with that of widely deployed signature schemes.

1 Introduction

Obfuscation deals with the problem of how to protect a program from reverse engineering while preserving functionality. Traditionally constructing secure obfuscation in a mathematically sound way has been a very challenging problem. While there have been numerous ad hoc approaches to obfuscation, in practice these have nearly all broken, pointing to the need for a cryptographically grounded solution. This state of affairs changed dramatically with the introduction of candidate indistinguishability obfuscation by Garg, Gentry, Halevi, Raykova, Sahai and Waters [GGH$^+$13a].

Recently, starting with [SW14] there has been much interest in investigating what can be built from indistinguishability obfuscation, since this model leads to poly-time obfuscation of unrestricted program classes, circumventing the known impossibility results of [BGI$^+$01]. Roughly, this body of work can be divided into two classes. The first seeks to discover new applications that were not achievable prior to introduction of secure obfuscation. The second seeks to re-explore the construction of existing cryptographic primitives, but through the lens of obfuscation.

The latter direction is important for several reasons. First, re-exploration leads to qualitatively different ways of approaching cryptographic problems. For example, the Sahai-Waters [SW14] public key scheme was obtained by application of obfuscation to symmetric key encryption, thereby matching Diffie and Hellman's original vision that public key encryption be obtainable by scrambling a private key enciphering program [DH76]. Moreover the techniques that were used to build cryptographic primitives for which their were already candidates, led to new and unexpected results. For example the technique of "punctured programming", which was used to construct public key encryption in turn, led to the first construction of deniable encryption. Second, such schemes have unique and interesting properties in their own right. For example, decryption in the SW public key cryptosystem involves only a symmetric key operations and is therefore quite fast. Likewise the signing algorithm from their signature scheme is also fast due to only applying symmetric key primitives.

This work In this paper we explore building new signature systems from obfuscation. Our goals are twofold. First, we would like to achieve short signatures with adaptive security proofs (matching the GMR definition [GMR88]).

Second, we would like to build signatures with fast signing— ideally significantly faster than comparable signatures that are not based on obfuscation. The goal here is to create an "imbalanced" scheme where signing is fast at the expense of longer verification. Such imbalance could be useful in applications where signing must be done by low power devices such as sensors, which verification can be done by well equipped machines. We note that employing imbalanced schemes has a long history. For example, an earlier feature of low-exponent RSA [Knu81] (e.g. $e = 3$) was that verification was very fast. In addition, recent work on delegation of computation (e.g., [GGP10, LW12, GGH$^+$13b, GVW13, PHGR13]) works on a similar principle of saving resources of a weaker client.

Although current obfuscation candidates will admit prohibitively slow verification, the work of obfuscation is in its nascent stages and it is plausible that systems with reasonable performance will be realized in the not too distant future. This future seems even more possible if one considers that obfuscation candidates might be designed and optimized for implementing particular functionalities.

We begin by overviewing the techniques of the SW signature system [SW14] which builds signatures from puncturable pseudorandom functions (PRFs). We briefly recall that a puncturable PRF is a type of constrained pseudo random function [BW13, BGI14, KPTZ13] where a key $K\{x^*\}$ can be given out that allows one to evaluate the (keyed) function $F_K(\cdot)$ at all inputs x *except* when $x = x^*$. In this scheme the Setup algorithm chooses a puncturable PRF key at random. A message is signed by evaluating the puncturable PRF on it. The verification key is an indistinguishability obfuscation that on input a message and signature pair, verifies that the signature is the correct output of the PRF on the message. One significant limitation of this scheme is that it only satisfies unforgettability against a *selective* attacker. In this notion of security, the attack algorithm is forced to select the message M^* it will attempt to forge on at the beginning of the security game, before seeing the verification key and before he gets to query for signatures on other messages.

We are therefore interested in designing signature systems to accommodate stronger attacks on security - in particular the standard notion of security where that attacker can adaptively choose which message it will forge on. To construct a signature scheme satisfying this notion of security, we employ the prefix-guessing technique of Hohenberger-Waters [HW09]. Here the signature scheme challenger uses the list of pre-committed message queries to guess the shortest differing prefix between these messages and the forgery. This prefix is used to embed a challenge in the verification key, which a successful forger must answer with noticeable probability. We note that this "prefix technique" has been successfully employed in other contexts [CHKP10, BSW11, FS12, MP12, CK12, BHJ$^+$14, Seo14].

The Hohenberger-Waters technique gives a technique to build a scheme against an attacker that adaptively chooses the forged message, but where the signature queries must be declared before seeing the verification key. In [HW09] they transform this notion into a fully adaptive secure scheme by using the well-known technique of applying a chameleon hash [KR00] to the message before running the base signature scheme. Here we wish to avoid this transformation firstly to meet the goal of constructing fully secure signatures using purely obfuscation-based techniques and secondly to keep signing costs low compared to deployed signature schemes such as EC-DSA, necessitates avoiding discrete-log based systems.

To achieve adaptive security from the prefix-embedded scheme, we do something akin[1] to publishing a signature tag t and building a one-time signature using the tag t as the verifcation key for the message. This part of the signature is secure provided that no tag t is ever re-used more than once. We can then sign

[1]Our construction uses the primitive a little differently. In a standard one-time signature scheme, anyone can generate a verification key and sign with the corresponding private key. In our scheme some secret information is needed even to sign with tag t.

the tag with the prefix-guessing scheme, generating a second signature part. The structure of our signature scheme is such that we can xor these parts together, with a suitable modification to the verificaion circuit.

Our construction in a nutshell We now describe the main two pieces of our construction. Firstly we construct a one-time like signature scheme as follows. We generate a tag t of λ bits. Our first "signature piece" is $s_1 = \oplus_{i=1}^{l} F_1(K_1, t\|i\|M(i))$, where $F_1(K_1, \cdot)$ is a puncturable PRF with appropriate input length. Our verification key is an obfuscated circuit that on input $(M, (t, s_1))$ checks that s_1 is of the above form. The security property is that an adversary on seeing a signature for a message M that uses tag t, cannot construct a signature on $M^* \neq M$, that uses the same tag t. To argue security, we use a Lamport-like proof. If $M^* \neq M$ then there exists some index $\hat{i}$ where $M^*(\hat{i}) \neq M(\hat{i})$. Let bit $\hat{b} = M^*(\hat{i})$. The reduction algorithm can guess the position $\hat{i}$ and bit $\hat{b} = M^*(\hat{i})$ with noticeable probability. The reduction first punctures K_1 on $\tau = t\|\hat{i}\|\hat{b}$. Then it evaluates an injective one way function evaluated on the punctured value $F_1(K_1, \tau)$, yielding an image z. This image is embedded inside the verification key, VK, and is used to test validity of message M^*, while the punctured key can still be used to verify all other messages. VK is sent to the adversary. Now suppose an adversary produces a valid forgery (t, s_1^*). The reduction can extract the punctured value as $s_1^* \oplus_{i \neq \hat{i}} F_1(K_1\{\tau\}, t\|i\|M^*(i))$, yielding a pre-image of z. This breaks the security guarantee of the one way function.

Our second piece is the ability to sign the tag t according to the [HW09] prefix-guessing technique. To sign a tag t, a puncturable PRF $F_{2,i}(K_{2,i}, \cdot)$ is evaluated on every prefix. Here $F_{2,i}$ for $i = 1, \ldots, \ell$ takes in inputs of i bits. The signature piece is thus $s_2 = \oplus_{i=1}^{\lambda} F_{2,i}(K_{2,i}, t^{(i)})$, where the length-$i$ prefix of t is denoted $t^{(i)}$. A verification key is an obfuscated circuit that on input (t, s_2), checks that s_2 is of the above form. The security property is as follows. The attacker commits to a list of tags $(t_j)_{j=1}^{q}$. The challenger sends verification key VK, as well as signatures $(\sigma_j)_{j=1}^{q}$ corresponding to the above tags. The attacker should not be able to construct a signature on a tag t^* not contained in the queried list. We argue security as follows. Since t^* is distinct from all queried tags, there exists some tag $t_{j'}$ and index i' such that t^* and $t_{j'}$ agree on the first $i' - 1$ bits, differing at the i'^{th} bit. The reduction algorithm guesses the tag $t_{j'}$ and index i'. The reduction punctures $K_{2,i'}$ on the differing prefix p and generates an equivalent verification circuit, using an injective one way function is to hide the punctured value $F_{2,i'}(K_{2,i'}, p)$. This image is embedded inside the verification key, VK, which is sent to the adversary. Suppose the adversary now submits a valid forgery s_2^*. The reduction extracts the punctured value as $s_2^* \oplus_{i \neq i'} F_{2,i}(K_{2,i}, t^{(i)})$, again breaking the one way function.

Our complete scheme merges these two ideas to generate concise signatures. The signatures s_1 and s_2 are xor-ed together yielding a single signature s. The complete signature is thus (t, s). The verification circuit on input $(M, (t, s))$ computes $s_1 = \oplus_{i=1}^{l} F_1(K_1, t\|i\|M(i))$ and $s_2 = \oplus_{i=1}^{\lambda} F_{2,i}(K_{2,i}, t^{(i)})$ and checks that $s = s_1 \oplus s_2$. In the proof of security, the reduction will deal with the case that a forgery tag t is repeated, or not, separately.

Fast Signing While the scheme above achieves our goal of getting short signatures that are fully secure, it does not meet our goal of getting fast signing. The primarily culprit is that in the generation of the second signature piece, ℓ different punctured PRF systems must be evaluated leading to a $\mathcal{O}(\ell^2)$ calls to the underlying psuedorandom generator when using current constructions based on GGM [GGM86] trees.

We address this problem by giving a slightly modified second construction. The primary change is that instead of using ℓ different punctured PRF systems, each with a different domain size, we will use one punctured PRF with a *variable* length domain $\{0,1\}^{1 \leq i \leq \ell}$. That is the input to the function can be a string of any length up to ℓ. We can then plug this into our main construction.

At first glance it might seem that this modification brings us nothing since the construction still needs to XOR together ℓ different PRF values. However, as we will show that it is possible to create a variable length punctured PRF where the cost of evaluating the PRF on *all* prefixes of an ℓ bit message M is the same as computing the GGM tree once on M. The main modification is that, following Goldreich [Gol06], we now need a length tripling PRG $G : \{0,1\}^{\lambda} \to \{0,1\}^{3 \cdot \lambda}$ that goes from λ bits to $3 \cdot \lambda$ bits. In practice, this will likely consume more computation per pseudo random generator invocation call more than using a length doubling one, but in total should result in significantly faster signatures than the prior approach.

Evaluation We evaluate the cost of the selectively secure Sahai-Waters scheme and our adaptively secure scheme in terms of the cost of the underlying PRGs used by the puncturable PRFs. We compute concrete signing costs at the 128-bit security level using a several cryptographic hash functions and ciphers to instantiate the PRGs. These costs are compared to the RSA and EC-DSA signature schemes at the same security level.

For appropriate choices of the underlying PRG, our adaptively secure construction is significantly faster than EC-DSA. For example, at the 128-bit level, the EC-DSA algorithm takes 348 microseconds. When the ChaCha stream cipher is used to instantiate the PRGs, the selectively secure Sahai-Waters scheme takes 25 microseconds and the adaptively secure scheme takes 81 microseconds. Timings were performed on a quad-core Intel Xeon E3-1270 v2 workstation with 16Gb RAM, clocked @3.50GHz.

2 Signature Scheme Preliminaries

A signature scheme is a tuple of PPT algorithms:

Setup(1^λ) The setup algorithm outputs a pair (VK, SK) where VK is the verification key and SK is the secret key.

Sign(SK, M) The signing algorithm takes in secret key SK and message $M \in \mathcal{M}$ and outputs a signature σ.

Verify(VK, M, σ) The verification algorithm takes in a verification key VK, a message M and a claimed signature σ. The algorithm returns 1 if the signature is valid and $\perp$ otherwise.

Correctness $\forall M \in \mathcal{M}\, \Pr[\mathsf{Verify}(\mathrm{VK}, M, \mathsf{Sign}(\mathrm{SK}, M)) = 1 : (\mathrm{VK}, \mathrm{SK}) \leftarrow \mathsf{Setup}(1^\lambda)] = 1 - \mathrm{negl}(\lambda)$

In what follows will assume $\mathcal{M}$ is an l-bit message space.

2.1 Unforgeability against Adaptive Attacks

We extend the above notion of a secure signature scheme to accommodate adaptive attacks, according to the formalization by Goldwasser, Micali and Rivest [GMR88]. Here an attacker may adaptively make an arbitrary (polynomial) number of signature queries on messages, even after it has received the verification key. The attacker must then output a message on which it did not receive a signature, and a valid signature corresponding to that message.

Setup The challenger runs the algorithm $\mathsf{Setup}(1^\lambda)$ to obtain (VK, SK). The challenger sends VK to the adversary.

Queries Proceeding adaptively, the adversary requests a signature on any message $M \in \mathcal{M}$ and the challenger responds with $\sigma \leftarrow \mathsf{Sign}(\mathrm{SK}, M)$. Let $\mathcal{Q}$ be the set of messages queried by the adversary.

Output Eventually the adversary outputs a pair (M^*, σ^*) and is said to win the game if $M \notin \mathcal{Q}$ and $\mathsf{Verify}(\mathrm{VK}, M^*, \sigma^*) = 1$.

We define $\mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathcal{A},\Pi}$ to be the probability that adversary $\mathcal{A}$ wins in the above game.

Definition 1. *A signature scheme* (Setup, Sign, Verify) *is existentially unforgeable with respect to adaptive chosen message attacks if for all PPT algorithms $\mathcal{A}$*

$$\mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathcal{A},\Pi} \leq \mathrm{negl}(\lambda)$$

3 Obfuscation Preliminaries

Definition 2 (Indistinguishability Obfuscation)**.** *A uniform PPT machine $i\mathcal{O}$ is called an indistinguishability obfuscator for a circuit class $\{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ if it satisfies the following conditions:*

- **Functionality preserving** *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 $C' \leftarrow i\mathcal{O}(\lambda, C)$.*
- **Indistinguishability of obfuscation** *For any not necessarily uniform PPT distinguisher $(Samp, D)$, there exists a negligible function* $\mathrm{negl}(\cdot)$ *such that the following holds: if for all security parameters $\lambda \in \mathbb{N}$*

$$\Pr[\forall x, C_0(x) = C_1(x) : (C_0; C_1; \tau) \leftarrow Samp(1^\lambda)] > 1 - \mathrm{negl}(\lambda),$$

then

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

3.1 Puncturable PRFs

A pseudorandom function (PRF) is a function $F : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ such that the function $F(K, \cdot)$ is indistinguishable from random when $K \xleftarrow{\$} \mathcal{K}$.

A constrained PRF [BW13] is a PRF$F(K, \cdot)$ with the functionality to enable evaluation of the PRF at certain portions of the input space and nowhere else. A *puncturable PRF* [BW13, SW14] is a type of contrained PRF that enables evaluation at the complement of a single arbitrary polynomial-sized subset of the input space. That is, PRF $F(K, \cdot)$ is equipped with additional PPT algorithms $(\mathsf{Eval}_F, \mathsf{Puncture}_F)$ such that the following properties hold

- **Functionality preserved under puncturing** For every PPT algorithm $\mathcal{A}$ which on input 1^λ outputs a set $S \subseteq \{0,1\}^n$, for all $x \in \{0,1\}^n \backslash S$, we have

$$\Pr[\mathsf{Eval}_F(K\{S\}, x) = F(K, x) : K \xleftarrow{\$} \mathcal{K}, K\{S\} \leftarrow \mathsf{Puncture}_F(K, S)] = 1$$

- **Pseudorandom at punctured points** For every pair of PPT algorithms $(\mathcal{A}_1, \mathcal{A}_2)$ and polynomial $m(\lambda)$ such that $\mathcal{A}_1(1^\lambda)$ outputs a set $S \subseteq \{0,1\}^n$ of cardinality $m(\lambda)$ and a state σ it holds that

$$|\Pr[\mathcal{A}_2(\sigma, K\{S\}, F(K, S)) = 1 : (S, \sigma) \leftarrow \mathcal{A}_1(1^\lambda), K\{S\} \leftarrow \mathsf{Puncture}_F(K, S)] - \Pr[\mathcal{A}_2(\sigma, K\{S\}, Y) = 1 : (S, \sigma) \leftarrow \mathcal{A}_1(1^\lambda), K\{S\} \leftarrow \mathsf{Puncture}_F(K, S), Y \xleftarrow{\$} \{0,1\}^{m \cdot |S|}]| \leq \mathrm{negl}(\lambda)$$

For notational convenience we will denote the output of $\mathsf{Puncture}_F(K, S)$ by $K\{S\}$, with the former occasionally suppressed.

4 Our Adaptively Secure Signature Scheme

In this section we describe our adaptively secure signature scheme. Our signature scheme consists of two main pieces. Our first piece is a one-time like signature scheme. We generate a tag t of λ bits. The security property is that an adversary on seeing a signature for a message M that uses tag t, cannot construct a signature on $M^* \neq M$, that uses the same tag t. Our signature piece is $s_1 = \oplus_{i=1}^{l} F_1(K_1, t\|i\|M(i))$, where $F_1(K_1, \cdot)$ is a puncturable PRF with appropriate input length. Our verification key is an obfuscated circuit that on input $(M, (t, s_1))$ checks that s_1 is of the correct form. To argue security, the simulator guesses the position $\hat{i}$ where $M^*(\hat{i}) \neq M(\hat{i})$ and the bit $\hat{b} = M^*(\hat{i})$. The simulator punctures K_1 on $\tau = t\|\hat{i}\|\hat{b}$, and generates an equivalent one-time verification circuit using the punctured key, making use of an (injective) one-way function to hide the punctured value. Now suppose an adversary produces a valid forgery (t, s_1^*). Then the simulator can extract the punctured value as s_1^* xor-ed with $\{F_1(K_1\{\tau\}, t\|i\|M^*(i)) : i \neq \hat{i}\}$. This can be shown to break the security of the one-way function.

Our second piece is the ability to sign the tag t according to the [HW09] prefix-guessing technique. The security property is as follows. The attacker commits to a list of tags $(t_j)_{j=1}^{q}$. The challenger sends verification key VK, as well as signatures $(\sigma_j)_{j=1}^{q}$ corresponding to the above tags. The attacker should not be able to construct a signature on a tag t^* not contained in the queried list. Let $F_{2,i}$ for $i = 1, \ldots, \ell$ be puncturable PRFs taking in inputs of i bits. The second signature piece is $s_2 = \oplus_{i=1}^{\lambda} F_{2,i}(K_{2,i}, t^{(i)})$, where the length-$i$ prefix of t is denoted $t^{(i)}$. A verification key is an obfuscated circuit that on input (t, s_2), checks that s_2 is of the above form. To argue security, the simulator guesses the tag $t_{j'}$ and position i' such that t^* and $t_{j'}$ have common prefix of length $i'-1$. The simulator punctures $K_{2,i'}$ on differing prefix p and generates an equivalent verfication circuit using the punctured key, the one way function is used to hide the punctured value. Suppose the adversary now submits a valid forgery s_2^*. Then the simulator can extract the punctured value as s_2^* xor-ed with $\{F_{2,i}(K_{2,i}, t^{(i)}) : i \neq i'\}$, again breaking the one-way function.

To combines these pieces, the signatures s_1 and s_2 are xor-ed together. A signature is simply (t, s). The verfication circuit on input $(M, (t, s))$ computes s_1 and s_2 as above and checks that $s = s_1 \oplus s_2$. In the proof of security, the simulator will deal with the case that a forgery tag t is repeated, or not, separately. In the former case, the simulator will extract a punctured value as $s^* \oplus s_2^* \oplus_{i \neq \hat{i}} F_1(K_1\{\tau\}, t\|i\|M^*(i))$, where s_2^* is computed using non-punctured PRF key K_2. In the case of the no repeat, the simulator will extract a punctured value as $s^* \oplus s_1^* \oplus_{i \neq i'} F_{2,i}(K_{2,i}, t^{(i)})$, where s_1^* is computed using non-punctured PRF key K_1. To complete the proof of security, the simulator guesses ahead of time which case it will have to deal with it, reducing its success probability by at most one half, hence remaining non-negligible. Our complete scheme follows.

4.1 The Scheme

The message space of the signature scheme is $\{0,1\}^l$. For l-bit message M, let $M(i)$ denote the i-th bit of M. For λ-bit string t, let $t^{(i)}$ denote the first i bits of t. Let $F_1(K_1, \cdot)$ be a puncturable PRF mapping l_t-bit inputs to λ-bit outputs. Here $l_t = \lambda + \lceil \lg l \rceil + 1$. Let $F_{2,i}(K_{2,i}, \cdot)$ be a puncturable PRF mapping i-bit inputs to λ-bit outputs, for each $i \in [1, \lambda]$. Let f be an injective one way function mapping λ-bit inputs to w-bit outputs. Our signature scheme is as follows.

Setup(1^λ) : Pick puncturable PRF keys $K_1 \xleftarrow{\$} \mathcal{K}_1$ and $K_{2,i} \xleftarrow{\$} \mathcal{K}_{2,i} : i \in [1, \lambda]$. The secret key is $(K_1, (K_{2,i})_{i=1}^{\lambda})$. Let the verification key VK be an indistinguishability obfuscation of the program SigCheck defined below.

Sign(SK, M) : Choose $t \xleftarrow{\$} \{0,1\}^\lambda$. Let $s_1 = \oplus_{i=1}^{l} F_1(K_1, t\|i\|M(i))$. Let $s_2 = \oplus_{i=1}^{\lambda} F_{2,i}(K_{2,i}, t^{(i)})$. Compute $s = s_1 \oplus s_2$. Output $\sigma = (t, s)$.

Verify$(\mathbf{VK}, M, \sigma)$: Output VK(M, σ).

```
SigCheck :
Inputs : $M, \sigma$
Constants : PRF keys $K_1$ and $(K_{2,i})_{i=1}^{\lambda}$
  $(t, s) \leftarrow \sigma$
  $s_1 \leftarrow \oplus_{i=1}^{l} F_1(K_1, t\|i\|M(i))$
  $s_2 \leftarrow \oplus_{i=1}^{\lambda} F_{2,i}(K_{2,i}, t^{(i)})$
  if $s = s_1 \oplus s_2$ then output 1 else output $\perp$
```

Theorem 1. *The above signature scheme is existentially unforgeable with respect to chosen message attacks as specified in Definition 1, assuming the existence of secure indistinguishability obfuscators and secure puncturable PRFs.*

Proof. Suppose that $\mathcal{A}$ is a PPT adversary that outputs a valid forgery (M^*, σ^*) with non-negligible probability ϵ. We will construct an adversary $\mathcal{B}$ that inverts the one way function f with non-neligible probability. We may split the forgery submitted by $\mathcal{A}$ into two cases. Let $\sigma^* = (t^*, s^*)$. The first case is that $t^* = t_j$ for some signature σ_j returned by the challenger in response to the j^{th} message query $M_j \in \mathcal{Q}$. The other case is that $t^* \neq t_j$ for all signatures σ_j returned by the challenger on $M_j \in \mathcal{Q}$, where j ranges from 1 to n. We will call the first case a type I forgery and the second case, where $t^* \neq t_j$ for all t_j, a type II forgery. We prove that in both cases, there exists PPT $\mathcal{B}$ which uses $\mathcal{A}$ to invert f. In practice simulator $\mathcal{B}$ guesses ahead of time which type of forgery $\mathcal{A}$ will make and errs with probability at most one half. In what follows we will let $|\mathcal{Q}| = n$. Also define $e_i = 0^{i-1}1$.

Lemma 1. *Suppose that adversary $\mathcal{A}$ in the adaptive security game makes a type I forgery with probability ϵ_I. Then we can construct $\mathcal{B}$ that inverts the one way function f with probability $\epsilon_I/(2nl) - \text{negl}(\lambda)$.*

Proof. To prove this lemma, we define the following sequence of hybrids.

Game 1 This is the original security game in which the attacker receives the verification key, and then queries for signatures on messages adaptively. Let $\mathcal{Q}$ be the set of queried messages. In the final step an attacker outputs (M^*, σ^*). Here $\sigma^* = (t^*, s^*)$ and $t^* = t_j$ for some signature σ_j on $M_j \in \mathcal{Q}$.

1. Let $t_j \xleftarrow{\$} \{0,1\}^\lambda$ for all $j \in [1, n]$.
2. Pick $K_1 \xleftarrow{\$} \mathcal{K}_1$ and $K_2 \xleftarrow{\$} \mathcal{K}_2$.
3. Let $\text{VK} = i\mathcal{O}(\lambda, \mathsf{SigCheck})$. Here the circuit $\mathsf{SigCheck}$ is padded if necessary, such that its size is equal to that of later inputs to the obfuscator.
4. Output VK.
5. While $M_j \in \mathcal{Q}$ is received:
   (a) Let $s_{1j} = \oplus_{i=1}^{l} F_1(K_1, t_j \| i \| M(i))$ and $s_{2j} = \oplus_{i=1}^{\lambda} F_{2,i}(K_{2,i}, t_j^{(i)})$.
   (b) Compute $s_j = s_{1j} \oplus s_{2j}$. Let $\sigma_j = (t_j, s_j)$.
   (c) Output σ_j.
6. Receive (M^*, σ^*).

$\mathcal{A}$ succeeds if $M^* \notin \mathcal{Q}$ and $\text{VK}(M^*, \sigma^*) = 1$.

Game 2 In this hybrid we change the winning condition. First the challenger choose indices $(\hat{i}, \hat{j})$ in $[1, l] \times [1, n]$ and a bit $\hat{b}$ in $\{0, 1\}$ at random. Suppose an attacker in the final step outputs (M^*, σ^*). The winning condition enforces an additional check that $t^* = t_{\hat{j}}$ and $M^*(\hat{i}) = \hat{b}$ and $\hat{b} \neq M_{\hat{j}}(\hat{i})$.

1. Let $t_j \xleftarrow{\$} \{0,1\}^\lambda$ for all $j \in [1, n]$.
2. Choose $(\hat{i}, \hat{j})$ in $[1, l] \times [1, n]$ and $\hat{b}$ in $\{0, 1\}$ at random. $\underline{\text{Let } \tau = t_{\hat{j}} \| \hat{i} \| \hat{b})}$.
3. Pick $K_1 \xleftarrow{\$} \mathcal{K}_1$ and $K_{2,i} \xleftarrow{\$} \mathcal{K}_{2,i} : i \in [1, \lambda]$.
4. Let $\text{VK} = i\mathcal{O}(\lambda, \mathsf{SigCheck})$. Here the circuit $\mathsf{SigCheck}$ is padded if necessary, such that its size is equal to that of later inputs to the obfuscator.
5. Output VK.
6. While $M_j \in \mathcal{Q}$ is received:
   (a) Let $s_{1j} = \oplus_{i=1}^{l} F_1(K_1, t_j \| i \| M(i))$ and $s_{2j} = \oplus_{i=1}^{\lambda} F_{2,i}(K_{2,i}, t_j^{(i)})$.
   (b) Compute $s_j = s_{1j} \oplus s_{2j}$. Let $\sigma_j = (t_j, s_j)$.
   (c) Output σ_j.
7. Receive (M^*, σ^*).

$\mathcal{A}$ succeeds if $M^* \notin \mathcal{Q}$ and $\text{VK}(M^*, \sigma^*) = 1$ $\underline{\text{ and if } t^* = t_{\hat{j}} \text{ and } M^*(\hat{i}) = \hat{b} \text{ and } \hat{b} \neq M_{\hat{j}}(\hat{i})}$.

Game 3 In this game the challenger creates the verification key as an obfuscation of an alternate verification circuit $\mathsf{SigCheckA}$. First the challenger computes a puncturing of the secret key K_1 at string τ. Let $y = F_1(K_1, \tau)$. The challenger uses the punctured key $K_1\{\tau\}$, punctured value y and the injective OWF f to generate $\mathsf{SigCheckA}$.

1. Let $t_j \xleftarrow{\$} \{0,1\}^\lambda$ for all $j \in [1, n]$.
2. Choose $(\hat{i}, \hat{j})$ in $[1, l] \times [1, n]$ and $\hat{b}$ in $\{0, 1\}$ at random. Let $\tau = (t_{\hat{j}}, \hat{i}, \hat{b})$.
3. Pick $K_1 \xleftarrow{\$} \mathcal{K}_1$ and $K_{2,i} \xleftarrow{\$} \mathcal{K}_{2,i} : i \in [1, \lambda]$. Let $K_1\{\tau\} \leftarrow \mathsf{Puncture}_{F_1}(K_1, \tau)$. Let $y = F_1(K_1, \tau)$. Let $z = f(y)$.
4. Let $\mathrm{VK} = i\mathcal{O}(\lambda, \mathsf{SigCheckA})$.
5. Output VK.
6. While $M_j \in \mathcal{Q}$ is received:
   (a) Let $s_{1j} = \oplus_{i=1}^{l} F_1(K_1, t_j \| i \| M(i))$ and $s_{2j} = \oplus_{i=1}^{\lambda} F_{2,i}(K_{2,i}, t_j^{(i)})$.
   (b) Compute $s_j = s_{1j} \oplus s_{2j}$. Let $\sigma_j = (t_j, s_j)$.
   (c) Output σ_j.
7. Receive (M^*, σ^*).

```
SigCheckA :
Inputs : $M, \sigma$
Constants : Punctured PRF key $K_1\{\tau\}$, keys $(K_{2,i})_{i=1}^{\lambda}$. Strings $\hat{i}, \tau, z$.
  $(t, s) \leftarrow \sigma$
  $s_2 \leftarrow \oplus_{i=1}^{\lambda} F_{2,i}(K_{2,i}, t^{(i)})$
  if $t \| \hat{i} \| M(\hat{i}) = \tau$ then
      if $f(s \oplus s_2 \oplus_{i \neq \hat{i}} F_1(K_1\{\tau\}, t \| i \| M(i))) = z$ then output 1 else output $\perp$
  else
      if $s \oplus s_2 = \oplus_{i=1}^{l} F_1(K_1\{\tau\}, t \| i \| M(i))$ then output 1 else output $\perp$
  end if
```

$\mathcal{A}$ succeeds if $M^* \notin \mathcal{Q}$ and $\mathrm{VK}(M^*, \sigma^*) = 1$ and if $t^* = t_{\hat{j}}$ and $M^*(\hat{i}) = \hat{b}$ and $\hat{b} \neq M_{\hat{j}}(\hat{i})$.

Game 4 In this game the constant y, used to create z in $\mathsf{SigCheckA}$, is replaced with a random λ-bit string. The other parts of the game do not change.

1. Let $t_j \xleftarrow{\$} \{0,1\}^\lambda$ for all $j \in [1, n]$.
2. Choose $(\hat{i}, \hat{j})$ in $[1, l] \times [1, n]$ and $\hat{b}$ in $\{0, 1\}$ at random. Let $\tau = t_{\hat{j}} \| \hat{i} \| \hat{b}$.
3. Pick $K_1 \xleftarrow{\$} \mathcal{K}_1$ and $K_{2,i} \xleftarrow{\$} \mathcal{K}_{2,i} : i \in [1, \lambda]$. Let $K_1\{\tau\} \leftarrow \mathsf{Puncture}_{F_1}(K_1, \tau)$. Choose y at random in $\{0,1\}^\lambda$. Let $z = f(y)$.
4. Let $\mathrm{VK} = i\mathcal{O}(\lambda, \mathsf{SigCheckA})$.
5. Output VK.
6. While $M_j \in \mathcal{Q}$ is received:
   (a) Let $s_{1j} = \oplus_{i=1}^{l} F_1(K_1, t_j \| i \| M(i))$ and $s_{2j} = \oplus_{i=1}^{\lambda} F_{2,i}(K_{2,i}, t_j^{(i)})$.
   (b) Compute $s_j = s_{1j} \oplus s_{2j}$. Let $\sigma_j = (t_j, s_j)$.
   (c) Output σ_j.
7. Receive (M^*, σ^*).

$\mathcal{A}$ succeeds if $M^* \notin \mathcal{Q}$ and $\mathrm{VK}(M^*, \sigma^*) = 1$ and if $t^* = t_{\hat{j}}$ and $M^*(\hat{i}) = \hat{b}$ and $\hat{b} \neq M_{\hat{j}}(\hat{i})$.

Claim 1. *Suppose there exists a PPT adversary $\mathcal{A}$ making a type I forgery such that $\mathsf{Adv}_{\mathcal{A},\Pi}^{\mathsf{Game1}} = \epsilon$. Then the advantage of $\mathcal{A}$ in Game 2, i.e. $\mathsf{Adv}_{\mathcal{A},\Pi}^{\mathsf{Game2}}$, is bounded below by $\epsilon/(2nl)$.*

Proof. For any message M^* submitted by $\mathcal{A}$, since σ^* is a type I forgery, there exists $M_j \in \mathcal{Q}$ such that the signature σ_j satisfies $t_j = t^*$. On the other hand, since $M^* \neq M_j$, there exists some bit position $i \in [1, l]$ for which $M^*(i) \neq M_j(i)$. Since the challenger chooses $(\hat{i}, \hat{j}, \hat{b})$ randomly in $[1, l] \times [1, n] \times \{0, 1\}$, the event $(\hat{i}, \hat{j}, \hat{b}) \overset{?}{=} (i, j, M^*(i))$ occurs with probability $1/(2nl)$. The claim then follows from the fact the view of $\mathcal{A}$ in Game 1 is identical to its view in Game 2. □

Claim 2. *Suppose there exists a PPT adversary $\mathcal{A}$ for which $\mathsf{Adv}^{\mathsf{Game2}}_{\mathcal{A},\Pi} - \mathsf{Adv}^{\mathsf{Game3}}_{\mathcal{A},\Pi} = \epsilon$ is non-negligible. Then we can construct an attacker $\mathcal{B}$ with advantage ϵ in distinguishing the output of the indistinguishability obfuscator.*

Proof. We first demonstrate the functional equivalence of circuits $\mathsf{SigCheck}$ and $\mathsf{SigCheckA}$. Consider a claimed signature (M, σ) which is input to the latter circuit. If it holds that $t\|\hat{i}\|M(\hat{i}) \neq \tau$, then since the 2^{nd} component of τ is $\hat{i}$, $t\|i\|M(i) \neq \tau$ for all $i \in [1, l]$. Therefore $F_1(K_1\{\tau\}, t\|i\|M(i)) = F_1(K_1, t\|i\|M(i))$ for all $i \in [1, l]$, since the punctured PRF key preserves functionality outside the punctured point. Thus $s = \oplus_{i=1}^{l} F_1(K_1, t\|i\|M(i)) \oplus s_2 \Leftrightarrow s \oplus s_2 = \oplus_{i=1}^{l} F_1(K_1\{\tau\}, t\|i\|M(i))$. On the other hand, if $t\|\hat{i}\|M(\hat{i}) = \tau$, then $s = \oplus_{i=1}^{l} F_1(K_1, t\|i\|M(i)) \Leftrightarrow s \oplus s_2 \oplus_{i \neq \hat{i}} F_1(K_1\{\tau\}, t\|i\|M(i)) = F_1(K_1, \tau) \Leftrightarrow f(s \oplus s_2 \oplus_{i \neq \hat{i}} F_1(K_1\{\tau\}, t\|i\|M(i))) = f(y)$, since f is injective. Consider the adversary $\mathcal{B} = (Samp, D)$. The algorithm $Samp$ on 1^λ chooses $t_j \xleftarrow{\$} \{0, 1\}^\lambda$ for $j \in [1, n]$. Next it picks random $(\hat{i}, \hat{j}, \hat{b})$ in $[1, l] \times [1, n] \times \{0, 1\}$, computes string τ and outputs $C_0 = \mathsf{SigCheck}$ and $C_1 = \mathsf{SigCheckA}$. It sets state $\delta = ((t_j)_{i=1}^{n}, K_1, (K_{2,i})_{i=1}^{\lambda}, \hat{i}, \hat{j}, b)$. Now the distingisher D on input $(\delta,\ i\mathcal{O}(\lambda, C_z))$ sends $\mathrm{VK} = i\mathcal{O}(\lambda, C_z)$ as well as a signature σ_j on each message M_j received from $\mathcal{A}$. If $\mathcal{A}$ outputs a pair (M^*, σ^*) for which $M^* \notin \mathcal{Q}$, $\mathrm{VK}(M^*, \sigma^*) = 1$ and $M^*(\hat{i}) = \hat{b}$ and $\hat{b} \neq M_{\hat{j}}(\hat{i})$ then D outputs 1, otherwise $\perp$. If $z = 0$ then $\mathcal{B}$ simulates Game 2. Otherwise $\mathcal{B}$ simulates Game 3. The claim follows. □

Claim 3. *Suppose there exists a PPT adversary $\mathcal{A}$ for which $\mathsf{Adv}^{\mathsf{Game3}}_{\mathcal{A},\Pi} - \mathsf{Adv}^{\mathsf{Game4}}_{\mathcal{A},\Pi} = \epsilon$ is non-negligible. Then we can construct an attacker $\mathcal{B}$ with advatange ϵ in distinguishing the output of the puncturable PRF $F_1(K_1, \cdot)$.*

Proof. Simulator $\mathcal{B}$ interacts with the puncturable PRF challenger while acting as a challenger to $\mathcal{A}$. First $\mathcal{B}$ chooses $t_j \xleftarrow{\$} \{0, 1\}^\lambda$ for $j \in [1, n]$. Next it chooses $(\hat{i}, \hat{j}, \hat{b})$ at random from $[1, l] \times [1, n] \times \{0, 1\}$ and computes string τ. $\mathcal{B}$ submits point τ to the PRF challenger and receives punctured key $K_1\{\tau\}$ and PRF challenge y' in return. $\mathcal{B}$ then computes $z = f(y')$ and computes an obfuscation of $\mathsf{SigCheckA}$. It sends $\mathrm{VK} = i\mathcal{O}(\lambda, \mathsf{SigCheckA})$ and a signature σ on every received message M_j. If $\mathcal{A}$ submits (M^*, σ^*) which meets the winning condition, then $\mathcal{B}$ outputs 1, otherwise $\perp$. If $y' = F_1(K_1, \tau)$ then $\mathcal{B}$ simulates Game 3. Otherwise y' is a random w-bit string and $\mathcal{B}$ simulates Game 4. The claim follows. □

Claim 4. *Suppose there exists a PPT adversary $\mathcal{A}$ such that $\mathsf{Adv}^{\mathsf{Game4}}_{\mathcal{A},\Pi} = \epsilon$ is non-negligible. Then we can construct an adversary $\mathcal{B}$ with advantage ϵ in inverting the one way function.*

Proof. Simulator $\mathcal{B}$ interacts with the one way function challenger while acting as a challenger to $\mathcal{A}$. First $\mathcal{B}$ receives the challenge $z' = f(a)$ as an input, where a is a w-bit random string. Then it chooses $t_j \xleftarrow{\$} \{0, 1\}^\lambda$ for $j \in [1, n]$. Next it chooses $(\hat{i}, \hat{j}, \hat{b})$ at random from $[1, l] \times [1, n] \times \{0, 1\}$ and computes string τ. It computes punctured key $K_1\{\tau\}$ and an obfuscation of $\mathsf{SigCheckA}$ with z' hardwired in place of z. Since z' is identically distributed to z, the view of $\mathcal{A}$ is identical to its view in Game 4. It sends VK and signatures on every received message M_j. Then with probability ϵ, $\mathcal{A}$ outputs (M^*, σ^*) such that $M^* \notin \mathcal{Q}$, $\mathrm{VK}(M^*, \sigma^*) = 1$ and $t^*\|\hat{i}\|M^*(\hat{i}) = \tau$. Thus $\mathcal{B}$ can compute $a^* = s^* \oplus s_2^* \oplus_{i \neq \hat{i}} F_1(K_1\{\tau\}, t\|i\|M^*(i))$ which satisfies $a^* = f^{-1}(z')$. □

Suppose that there exists a PPT adversary $\mathcal{A}$ with non-negligible advantage ϵ in breaking the adaptive security of the signature scheme, i.e. in winning $\mathsf{Game\ 1}$. Let the maximal advantage of any adversary in distinguishing the output of the indistinguishability obfuscator and in distinguishing the output of the puncturable PRF be $\epsilon_{i\mathcal{O}}$ and ϵ_{PRF} respectively. Claims 1 - 4 then imply that $\mathcal{A}$ has probability atleast

$\epsilon/(nl) - \epsilon_{i\mathcal{O}} - \epsilon_{PRF}$ in inverting the one way function f. Since n and l are polynomially bounded, the first term is non-negligible and since $i\mathcal{O}$ is a secure indistinguishability obfuscator and F_1 is a secure puncturable PRF the second and third terms are negligible. This contradicts the non-invertibility of f. It follows that the advantage of $\mathcal{A}$ in $\mathsf{Game\ 1}$ must be negligible. This concludes the proof of Lemma 1. □

Lemma 2. *Suppose that adversary $\mathcal{A}$ in the adaptive security game makes a type II forgery with probability ϵ_{II}. Then we can construct $\mathcal{B}$ that inverts the one way function f with probability $\epsilon_{II}/(n\lambda) - \mathrm{negl}(\lambda)$.*

Proof. Similar to the proof of Lemma 1, we proves this result by a hybrid argument.

Game 1 This is the original security game in which the attacker receives the verification key, and then queries for signatures on messages adaptively. Let $\mathcal{Q}$ be the set of queried messages. In the final step an attacker outputs (M^*, σ^*). Here $\sigma^* = (t^*, s^*)$ and $t^* \neq t_j$ for all signatures σ_j on $M_j \in \mathcal{Q}$.

1. Let $t_j \xleftarrow{\$} \{0,1\}^\lambda$ for all $j \in [1, n]$.
2. Pick $K_1 \xleftarrow{\$} \mathcal{K}_1$ and $K_2 \xleftarrow{\$} \mathcal{K}_2$.
3. Let $\mathrm{VK} = i\mathcal{O}(\lambda, \mathsf{SigCheck})$. Here the circuit $\mathsf{SigCheck}$ is padded if necessary, such that its size is equal to that of later inputs to the obfuscator.
4. Output VK.
5. While $M_j \in \mathcal{Q}$ is received:
   (a) Let $s_{1j} = \oplus_{i=1}^{l} F_1(K_1, t_j \| i \| M(i))$ and $s_{2j} = \oplus_{i=1}^{\lambda} F_{2,i}(K_{2,i}, t_j^{(i)})$.
   (b) Compute $s_j = s_{1j} \oplus s_{2j}$. Let $\sigma_j = (t_j, s_j)$.
   (c) Output σ_j.
6. Receive (M^*, σ^*).

$\mathcal{A}$ succeeds if $M^* \notin \mathcal{Q}$ and $\mathrm{VK}(M^*, \sigma^*) = 1$.

Game 2 In this hybrid we change the winning condition. First the challenger choose indices (i', j') in $[1, \lambda] \times [1, n]$ at random. Suppose an attacker in the final step outputs (M^*, σ^*). The winning condition enforces an additional check that t^* and $t_{j'}$ have shortest differing prefix of length i'.

1. Let $t_j \xleftarrow{\$} \{0,1\}^\lambda$ for all $j \in [1, n]$.
2. Choose (i', j') in $[1, \lambda] \times [1, n]$ at random. $\underline{\text{Let } p = t_{j'}^{(i')} \oplus e_{i'}}$.
3. Pick $K_1 \xleftarrow{\$} \mathcal{K}_1$ and $K_{2,i} \xleftarrow{\$} \mathcal{K}_{2,i} : i \in [1, \lambda]$.
4. Let $\mathrm{VK} = i\mathcal{O}(\lambda, \mathsf{SigCheck})$. Here the circuit $\mathsf{SigCheck}$ is padded if necessary, such that its size is equal to that of later inputs to the obfuscator.
5. Output VK.
6. While $M_j \in \mathcal{Q}$ is received:
   (a) Let $s_{1j} = \oplus_{i=1}^{l} F_1(K_1, t_j \| i \| M(i))$ and $s_{2j} = \oplus_{i=1}^{\lambda} F_{2,i}(K_{2,i}, t_j^{(i)})$.
   (b) Compute $s_j = s_{1j} \oplus s_{2j}$. Let $\sigma_j = (t_j, s_j)$.
   (c) Output σ_j.
7. Receive (M^*, σ^*).

$\mathcal{A}$ succeeds if $M^* \notin \mathcal{Q}$ and $\mathrm{VK}(M^*, \sigma^*) = 1$ $\underline{\text{and if } t^{*(i')} = p}$.

Game 3 In this game the challenger creates the verification key as an obfuscation of an alternate verification circuit SigCheckB. First the challenger computes a puncturing of the secret key $K_{2,i'}$ at string p. Let $y = F_{2,i}(K_{2,i}, p)$. The challenger uses the punctured key $K_{2,i'}\{p\}$, punctured value y and the injective OWF f to generate SigCheckB.

1. Let $t_j \xleftarrow{\$} \{0,1\}^\lambda$ for all $j \in [1,n]$.
2. Choose (i', j') in $[1,\lambda] \times [1,n]$ at random. Let $p = t_{j'}^{(i')} \oplus e_{i'}$.
3. Pick $K_1 \xleftarrow{\$} \mathcal{K}_1$ and $K_{2,i} \xleftarrow{\$} \mathcal{K}_{2,i} : i \in [1,\lambda]$. Let $K_{2,i'}\{p\} \leftarrow \mathsf{Puncture}_{F_{2,i'}}(K_{2,i'}, p)$. $\underline{\text{Let } y = F_{2,i'}(K_{2,i'}, p).}$ $\underline{\text{Let } z = f(y).}$
4. Let $\text{VK} = i\mathcal{O}(\lambda, \mathsf{SigCheckB})$.
5. Output VK.
6. While $M_j \in \mathcal{Q}$ is received:
   (a) Let $s_{1j} = \oplus_{i=1}^{l} F_1(K_1, t_j \| i \| M(i))$ and $s_{2j} = \oplus_{i=1}^{\lambda} F_{2,i}(K_{2,i}, t_j^{(i)})$.
   (b) Compute $s_j = s_{1j} \oplus s_{2j}$. Let $\sigma_j = (t_j, s_j)$.
   (c) Output σ_j.
7. Receive (M^*, σ^*).

```
SigCheckB :
Inputs : $M, \sigma$
Constants : PRF keys $K_1$, $(K_{2,i})_{i \neq i'}$, punctured key $K_{2,i'}\{p\}$. Strings $i', p, z$.
  $(t, s) \leftarrow \sigma$
  $s_1 \leftarrow \oplus_{i=1}^{\lambda} F_1(K_1, t \| i \| M(i))$
  if $\underline{t^{(i')} = p}$ then
     if $\underline{f(s \oplus s_1 \oplus_{i \neq i'} F_{2,i}(K_{2,i}, t^{(i)})) = z}$ then output 1 else output $\perp$
  else
     if $s \oplus s_1 = \oplus_{i \neq i'} F_{2,i}(K_{2,i}, t^{(i)}) \oplus F_{2,i'}(K_{2,i'}\{p\}, t^{(i')})$ then output 1 else output $\perp$
  end if
```

$\mathcal{A}$ succeeds if $M^* \notin \mathcal{Q}$ and $\text{VK}(M^*, \sigma^*) = 1$ $\underline{\text{and if } t^{*(i')} = p}$.

Game 4 In this game the constant y, used to create z in SigCheckB, is replaced with a random λ-bit string. The other parts of the game do not change.

1. Let $t_j \xleftarrow{\$} \{0,1\}^\lambda$ for all $j \in [1,n]$.
2. Choose (i', j') in $[1,\lambda] \times [1,n]$ at random. Let $p = t_{j'}^{i'} \oplus e_{i'}$.
3. Pick $K_1 \xleftarrow{\$} \mathcal{K}_1$ and $K_{2,i} \xleftarrow{\$} \mathcal{K}_{2,i} : i \in [1,\lambda]$. Let $K_{2,i'}\{p\} \leftarrow \mathsf{Puncture}_{F_{2,i'}}(K_{2,i'}, p)$. $\underline{\text{Choose } y \text{ at random in } \{0,1\}^\lambda}$. Let $z = f(y)$.
4. Let $\text{VK} = i\mathcal{O}(\lambda, \mathsf{SigCheckB})$.
5. Output VK.
6. While $M_j \in \mathcal{Q}$ is received:
   (a) Let $s_{1j} = \oplus_{i=1}^{l} F_1(K_1, t_j \| i \| M(i))$ and $s_{2j} = \oplus_{i=1}^{\lambda} F_2(K_2, t_j^{(i)})$.
   (b) Compute $s_j = s_{1j} \oplus s_{2j}$. Let $\sigma_j = (t_j, s_j)$.
   (c) Output σ_j.
7. Receive (M^*, σ^*).

$\mathcal{A}$ succeeds if $M^* \notin \mathcal{Q}$ and $\text{VK}(M^*, \sigma^*) = 1$ and if $t^{*(i')} = p$.

Claim 5. *Suppose there exists a PPT adversary $\mathcal{A}$ making a type II forgery such that $\mathsf{Adv}^{\mathsf{Game1}}_{\mathcal{A},\Pi} = \epsilon$. Then the advantage of $\mathcal{A}$ in Game 2, i.e. $\mathsf{Adv}^{\mathsf{Game2}}_{\mathcal{A},\Pi}$, is bounded below by $\epsilon/(n\lambda)$.*

Proof. For any message t^* submitted by $\mathcal{A}$ in Game 1, there exists a shortest common prefix of t^* with the $t_j : j \in [1, n]$. Also, since σ^* is a type II forgery, $t^* \neq t_j$, for all t_j. Thereore the length of this prefix is at most $\lambda - 1$. In particular there exists some string t_j and a prefix of length i, at most λ, for which $t^{*(i)} = t_j^{(i)} \oplus e_i$. Since the challenger chooses (i', j') uniformly at random from $[1, \lambda] \times [1, n]$, the event $(i', j') \overset{?}{=} (i, j)$ occurs with probability $1/(n\lambda)$. The claim then follows from the fact the view of $\mathcal{A}$ in Game 1 is identical to its view in Game 2.

□

Claim 6. *Suppose there exists a PPT adversary $\mathcal{A}$ for which $\mathsf{Adv}_{\mathcal{A},\Pi}^{\mathsf{Game2}} - \mathsf{Adv}_{\mathcal{A},\Pi}^{\mathsf{Game3}} = \epsilon$ is non-negligible. Then we can construct an attacker $\mathcal{B}$ with advantage ϵ in distinguishing the output of the indistinguishability obfuscator.*

Proof. We demonstrate the functional equivalence of circuits $\mathsf{SigCheck}$ and $\mathsf{SigCheckB}$ as follows. Consider a claimed signature (M, σ) which is input to the latter circuit. If it holds that $t^{(i')} \neq p$, then $F_{2,i'}(K_{2,i'}\{p\}, t^{(i')}) = F_{2,i'}(K_{2,i'}, t^{(i')})$, since the punctured PRF key preserves functionality outside the punctured point. Thus $s = s_1 \oplus_{i \neq i'} F_{2,i}(K_{2,i}, t^{(i)}) \oplus F_{2,i'}(K_{2,i'}, t^{(i')}) \Leftrightarrow s \oplus s_1 = \oplus_{i \neq i'} F_{2,i}(K_{2,i}, t^{(i)}) \oplus F_{2,i'}(K_{2,i'}\{p\}, t^{(i')})$. On the other hand, if $t^{(i')} = p$, then $s = s_1 \oplus_{i=1}^{\lambda} F_{2,i}(K_{2,i}, t^{(i)}) \Leftrightarrow s \oplus s_1 \oplus_{i \neq i'} F_{2,i}(K_{2,i}, t^{(i)}) = F_{2,i'}(K_{2,i'}, p) \Leftrightarrow f\left(s \oplus s_1 \oplus_{i \neq i'} F_{2,i}(K_{2,i}, t^{(i)})\right) = f(y)$, since f is injective. Consider the adversary $\mathcal{B} = (Samp, D)$. The algorithm $Samp$ on 1^λ chooses $t_j \overset{\$}{\leftarrow} \{0,1\}^\lambda$ for $j \in [1, n]$. Next it picks random (i', j') in $[1, \lambda] \times [1, n]$, computes string p and outputs $C_0 = \mathsf{SigCheck}$ and $C_1 = \mathsf{SigCheckB}$. It sets state $\delta = ((t_j)_{i=1}^n, K_1, (K_{2,i})_{i=1}^{\lambda}, i', j')$. Now the distingisher D on input $(\delta,\ i\mathcal{O}(\lambda, C_z))$ sends $\mathrm{VK} = i\mathcal{O}(\lambda, C_z)$ as well as a signature σ_j on each message M_j received from $\mathcal{A}$. If $\mathcal{A}$ outputs a pair (M^*, σ^*) for which $M^* \notin \mathcal{Q}$, $\mathrm{VK}(M^*, \sigma^*) = 1$ and $t^{*(i')} = p$ then D outputs 1, otherwise $\perp$. If $z = 0$ then $\mathcal{B}$ simulates Game 2. Otherwise $\mathcal{B}$ simulates Game 3. The claim follows. □

Claim 7. *Suppose there exists a PPT adversary $\mathcal{A}$ for which $\mathsf{Adv}_{\mathcal{A},\Pi}^{\mathsf{Game3}} - \mathsf{Adv}_{\mathcal{A},\Pi}^{\mathsf{Game4}} = \epsilon$ is non-negligible. Then we can construct an attacker $\mathcal{B}$ with advatange ϵ in distinguishing the output of the puncturable PRF $F_2(K_2, \cdot)$.*

Proof. Simulator $\mathcal{B}$ interacts with the puncturable PRF challenger while acting as a challenger to $\mathcal{A}$. First $\mathcal{B}$ chooses $t_j \overset{\$}{\leftarrow} \{0,1\}^\lambda$ for $j \in [1, n]$. Next it chooses (i', j') at random from $[1, \lambda] \times [1, n]$ and computes string p. $\mathcal{B}$ submits point p to the PRF challenger and receives punctured key $K_{2,i'}\{p\}$ and PRF challenge y' in return. $\mathcal{B}$ then computes $z = f(y')$ and computes an obfuscation of $\mathsf{SigCheckB}$. It sends $\mathrm{VK} = i\mathcal{O}(\lambda, \mathsf{SigCheckB})$ and a signature σ on every received message M_j. If $\mathcal{A}$ submits (M^*, σ^*) which meets the winning condition, then $\mathcal{B}$ outputs 1, otherwise $\perp$. If $y' = F_{2,i'}(K_{2,i'}, p)$ then $\mathcal{B}$ simulates Game 3. Otherwise y' is a random w-bit string and $\mathcal{B}$ simulates Game 4. The claim follows. □

Claim 8. *Suppose there exists a PPT adversary $\mathcal{A}$ such that $\mathsf{Adv}_{\mathcal{A},\Pi}^{\mathsf{Game4}} = \epsilon$ is non-negligible. Then we can construct an adversary $\mathcal{B}$ with advantage ϵ in inverting the one way function.*

Proof. Simulator $\mathcal{B}$ interacts with the one way function challenger while acting as a challenger to $\mathcal{A}$. First $\mathcal{B}$ receives the challenge $z' = f(a)$ as an input, where a is a λ-bit random string. Then it chooses $t_j \overset{\$}{\leftarrow} \{0,1\}^\lambda$ for $j \in [1, n]$. Next it chooses (i', j') at random from $[1, \lambda] \times [1, n]$ and computes string p. It computes punctured key $K_{2,i'}\{p\}$ and an obfuscation of $\mathsf{SigCheckB}$ with z' hardwired in place of z. Since z' is identically distributed to z, the view of $\mathcal{A}$ is identical to its view in Game 4. It sends VK and signatures on every received message M_j. Then with probability ϵ, $\mathcal{A}$ outputs (M^*, σ^*) such that $M^* \notin \mathcal{Q}$, $\mathrm{VK}(M^*, \sigma^*) = 1$ and $t^{*(i')} = p$. Thus $\mathcal{B}$ can compute $a^* = s^* \oplus s_1^* \oplus_{i \neq i'} F_{2,i}(K_{2,i}, t^{(i)})$ which satisfies $a^* = f^{-1}(z')$. □

Suppose that there exists a PPT adversary $\mathcal{A}$ with non-negligible advantage ϵ in breaking the adaptive security of the signature scheme. Similar to the proof of Lemma 1, Claims 5-8 imply that $\mathcal{A}$ has probability $\epsilon/(n\lambda) - \mathrm{negl}(\lambda)$ in inverting the one way function. As before, since n is polynomially bounded, this contradicts its non-invertibility. It follows that the advantage of $\mathcal{A}$ is negligible. □

Given Lemmas 1 and 2 we can conclude Theorem 1 as follows. Simulator $\mathcal{B}$ guesses ahead of time which forgery $\mathcal{A}$ will make. Thus $\mathcal{B}$ has advantage at least $(1/2) \cdot (\epsilon_I/(2nl) + \epsilon_{II}/(n\lambda) - \text{negl}(\lambda)) \geq Adv_{\mathcal{A},\Pi}/(2n \cdot \max\{2l, \lambda\}) - \text{negl}(\lambda)$. Since $\mathcal{B}$ has negligible advantage in inverting f, it follows that $\mathcal{A}$ has negligible advantage in breaking the signature scheme. □

5 Improving Efficiency

One drawback of our previous construction is that it does not meet our goal of achieving fast signing. (We will see some concrete end-to-end comparisons in Section 6.) Relative to the selectively secure Sahai-Waters scheme the primary drawback is that a punctured PRF must be evaluated ℓ different times. If we use the GGM implementation each call results is around ℓ applications of the underlying pseudo random generator,[2] resulting in an $\mathcal{O}(\ell^2)$ applications.

In this section we demonstrate an idea to lower the cost of our fully secure scheme. The primary change is that instead of using ℓ different punctured PRF systems, each with a different domain size, we will use one punctured PRF with a *variable* length domain $\{0,1\}^{1 \leq i \leq \ell}$. That is the input to the function can be a string of any length up to ℓ. We can then plug this into our main construction.

At first glance it might seem that this modification brings us nothing since the construction still needs to XOR together ℓ different PRF values. However, as we will show that it is possible to create a variable length punctured PRF where the cost of evaluating the PRF on *all* prefixes of an ℓ bit message M is the same as computing the GGM tree once on M. The main modification is that, following Goldreich [Gol06], we now need a length tripling PRG $G : \{0,1\}^\lambda \to \{0,1\}^{3 \cdot \lambda}$ that goes from λ bits to $3 \cdot \lambda$ bits. In practice, this could be more than using a length doubling one, but should result in significantly faster signatures than the prior approach.

Below we first give the modified punctured PRF construction. Then we show the changes to our scheme.

5.1 Puncturable PRFs for Variable Length Domain

Let the puncturable PRF have variable message size v bits and output size λ bits. Here $v \leq l$. Let G be a PRG with input size λ bits and output size 3λ bits. Define auxiliary functions G_0, G_1 and $G_\perp$ mapping λ bits to λ bits as follows: $G(x) = G_0(x) \| G_1(x) \| G_\perp(x)$.

Construction

```
• Setup(1^λ) : Let $K$ be a random $\lambda$-bit string. Output key $K$.
• Puncture($K, x$) :
      Write $x$ as $b_1 \ldots b_v$, where $v \leq l$.
      for  $i = 1$ to $v - 1$  do
          $F_{K,i} \leftarrow G_{\overline{b}_i}(G_{b_{i-1}}(\ldots(G_{b_1}(K))\ldots))$
          $F_{K,i,\perp} \leftarrow G_\perp(G_{b_i} \ldots (G_{b_1}(K))\ldots)$
      end for
      $F_{K,v} \leftarrow G_{\overline{b}_v}(G_{b_{v-1}}(\ldots(G_{b_1}(K))\ldots))$
      $F_{K,v+1,0} \leftarrow G_0(G_{b_v}(\ldots(G_{b_1}(K))\ldots))$
      $F_{K,v+1,1} \leftarrow G_1(G_{b_v}(\ldots(G_{b_1}(K))\ldots))$
   Let $K\{x\} = (F_{K,i}, F_{K,i,\perp})_{i\in[1,v-1]}, F_{K,v}, (F_{K,v+1,b})_{b\in[0,1]}$. Output key $K\{x\}$.
• Eval($K\{x\}, y$) :
      if $y = x$ then
          output $\perp$
      else
          Let $x = b_1 \ldots b_v$.
```

[2]The exact number of applications will depend on the length of the input. In Section 6 we work out the details.

```
        Write $y = b_1 \dots b_i b'_{i+1} \dots b'_u$, where $u \le l$.
        if $i < v$ and $u = i$ then
            Output $F_{K,i,\perp}$.
        else if $i < v$ and $u > i$ then
            Output $G_\perp(G_{b'_u}(\dots G_{b'_{i+2}}(F_{K,i+1}) \dots))$.
        else
            Output $G_\perp(G_{b'_u}(\dots G_{b'_{v+2}}(F_{K,v+1,b'_{v+1}}) \dots))$.
        end if
    end if
```

Cost To evaluate this puncturable PRF on a u-bit message requires $u-i$ calls to G, where i is the length of the longest common prefix with the punctured point. Even more significantly, to evaluate the PRF on every prefix of a u-bit message still only requires $u-i$ calls, because evaluations on successive prefixes corresponds to pipelined calls to G, with each evaluation terminated by a single application of $G_\perp$. We will see that this leads to significant savings in our signature scheme.

Security We defer the proof of security of this puncturable PRF to the Appendix.

5.2 Modified Scheme

The message space of the signature scheme is $\{0,1\}^l$. For l-bit message M, let $M(i)$ denote the i-th bit of M. For λ-bit string t, let $t^{(i)}$ denote the first i bits of t. Let $F_1(K_1, \cdot)$ be a puncturable PRF mapping l_t-bit inputs to λ-bit outputs. Here $l_t = \lambda + \lceil \lg l \rceil + 1$. Let $F_2(K_2, \cdot)$ be a puncturable PRF mapping $\{0,1\}^{1 \le i \le \ell}$ to λ-bit outputs. Let f be an injective one way function mapping λ-bit inputs to w-bit outputs. Our signature scheme is as follows.

Setup(1^λ) : Pick puncturable PRF keys $K_1 \xleftarrow{\$} \mathcal{K}_1$ and $K_2 \xleftarrow{\$} \mathcal{K}_2$. The secret key is (K_1, K_2). Let the verification key VK be an indistinguishability obfuscation of the program SigCheck defined below.

Sign(SK, M) : Choose $t \xleftarrow{\$} \{0,1\}^\lambda$. Let $s_1 = \oplus_{i=1}^{l} F_1(K_1, t \| i \| M(i))$. Let $s_2 = \oplus_{i=1}^{\lambda} F_2(K_2, t^{(i)})$. Compute $s = s_1 \oplus s_2$. Output $\sigma = (t, s)$.

Verify$(\mathbf{VK}, M, \sigma)$: Output VK(M, σ).

```
SigCheck :
Inputs : $M, \sigma$
Constants : PRF keys $K_1$ and $K_2$
  $(t, s) \leftarrow \sigma$
  $s_1 \leftarrow \oplus_{i=1}^{l} F_1(K_1, t \| i \| M(i))$
  $s_2 \leftarrow \oplus_{i=1}^{\lambda} F_2(K_2, t^{(i)})$
  if $s = s_1 \oplus s_2$ then output 1 else output $\perp$
```

We omit the proof of this scheme since it follows extremely close to the base scheme of Section 4.

6 Analysis and Evaluation

In this section we evaluate the cost of the (selectively secure) Sahai-Waters construction and the proposed adaptively secure construction in terms of the cost of the puncturable PRFs. We begin by expressing the cost of each scheme in terms of the underlying length-doubling and length-tripling PRGs. Next we show costs on a common architecture assuming 128-bit security and compare this against the RSA and elliptic curve DSA signature schemes.

6.1 Analysis of Costs

Let g_D be the cost of the length-doubling PRG and g_T be the cost of the length-tripling PRG. We assume the messages to be signed are l-bits.

Sahai-Waters [SW14] This scheme makes a single call to the puncturable PRF on an l-bit message. This call traverses the GGM tree according to the message bits, requring l invocations of the length-doubling PRG. The cost is therefore $g_D \cdot l$.

Adaptively secure scheme Our adaptively secure scheme calls the fixed-length puncturable PRF once on each of l inputs, where each input is $\lambda + \lceil \lg l \rceil + 1$ bits. However each input has the same λ-bit suffix, differing only in the remaining bits. Therefore the GGM tree can be traversed to a depth of λ, before a depth-first search is performed to an additional $\lceil \lg l \rceil + 1$ depth. Thus $\lambda + 2l - 1$ calls are made to the length-doubling PRG. Additionally the scheme evaluates the variable-length puncturable PRF once on a λ-bit input, outputting pipelined evaluations on each prefix. Therefore the modified GGM tree is traversed to a depth of λ, requiring λ calls to the length-tripling PRG. Therefore the total cost is $g_D \cdot (\lambda + 2l - 1) + g_T \cdot \lambda$.

6.2 Comparison of Signature Computation for 128-bit Security

To achieve a security level of 128 bits, we consider signatures on 256-bit messages. In practice such messages are produced by application of a collision-resistant hash function, we disregard this cost here. For our analysis we considered several different candidates for the PRG. These include the SHA-256 cyptographic hash function, the ChaCha stream cipher, the RC5 block cipher and AES-256 (software and hardware). We compare this against the cost of elliptic curve DSA signatures. All primitives are implemented using v1.0.1 of the OpenSSL library, excepting the ChaCha stream cipher which uses a C implementation available here [Cha14]. The AES hardware implementation is based upon the Intel AES-NI instruction set which is available via the EVP wrapper of the OpenSSL library. All timings were performed on a quad-core Intel Xeon E3-1270 v2 workstation with 16Gb RAM, clocked @3.50GHz.

Primitives

SHA-256 : The SHA-256 compression function maps 512-bits to 256-bits.

ChaCha : The ChaCha stream cipher is seeded via a 256-bit key and 64-bit IV. It generates 512-bit psuedorandom bits per update operation of the internal state.

AES (software only) : The AES-256 block cipher is seeded via a 256-bit key and 64-bit IV. It operates on 128-bit blocks.

AES (hardware accelerated) : The AES-256 block cipher is seeded via a 256-bit key and 64-bit IV. It operates on 128-bit blocks.

RC5 : The RC5 block cipher has a variable key/block size. The default implementation uses a 128-bit key and operates on 64-bit blocks.

RSA : The RSA algorithm (PKCS #1 v2.0) is used to generate signatures on a 3072-bit modulus. The cost is 3400µs.

EC-DSA : The elliptic curve DSA algorithm is used to generate signatures on a 256-bit curve. The cost is 348µs.

Length-doubling PRG

SHA-256 : The input is zero-padded to 512 bits. The SHA-256 compression function is then applied. The cost of this routine is 0.52µs.

ChaCha : The input is zero-padded to 256 bits. We then extract the first 256 bits of the 512 pseudorandom bits produced by an update operation and xor these with the input.[3] The cost to seed is 0.03µs. The amortized invocation cost is 0.18µs.

[3]The remaining bits are cached for the next invocation.

AES (software only) : The input is extended to 256 bits. We then apply the AES cipher in CTR mode. The cost to seed is 0.17µs. The invocation cost is 0.28µs.

AES (hardware accelerated) : The input is extended to 256 bits. We then apply the AES cipher in CTR mode. The cost to seed is 0.16µs. The invocation cost is 0.08µs.

RC5 : The input is extended to 256 bits. We then apply the RC5 cipher in CTR mode. The cost to seed is 1.11µs. The invocation cost is 0.36µs.

Length-tripling PRG

SHA-256 : The input is extended to 1024 bits. The SHA-256 compression function is then applied on each 512-bit block and the first 384 bits of the output extracted. The cost of this routine is 1.04µs.

ChaCha : The input is zero-padded to 384 bits. We then extract the first 384 bits of the 512 pseudorandom bits produced by an update operation and xor these with the input.[3] The cost to seed is 0.03µs. The amortized invocation cost is 0.27µs.

AES (software only) : The input is extended to 384 bits. We then apply the AES cipher in CTR mode. The cost to seed is 0.17µs. The invocation cost is 0.42µs.

AES (hardware accelerated) : The input is extended to 384 bits. We then apply the AES cipher in CTR mode. The cost to seed is 0.16µs. The invocation cost is 0.12µs.

RC5 : The input is extended to 384 bits. We then apply the RC5 cipher in CTR mode. The cost to seed is 1.11µs. The invocation cost is 0.54µs.

PRG	SHA-256	ChaCha	AES	AES (hardware)	RC5
[SW14]	133	23	36	10	47
Adapt. scheme	465	81	125	36	162
RSA	3400				
EC-DSA	348				

Table 1: Signature cost on 256-bit messages. By PRG, we mean the appropriate choice of length-doubling PRG or length-tripling PRG. All times given in microseconds.

Optimizations Exploiting the stream property of ChaCha and CTR mode of operation for the above block ciphers allows some efficiency gains. In a call to the fixed domain puncturable PRF, only one output block need be computed at each level of the GGM tree, halving the invocation cost of the length-doubling PRG. Likewise in a call to the variable-domain puncturable PRF, only two out of three output blocks need be computed at each level of the modified GGM tree, reducing the invocation cost of the length-tripling PRG by a factor of 2/3.

Pulling it Together Our measurements show some interesting features. For all pseudo random generator candidates considered, signing in the Sahai-Waters scheme is significantly faster than EC-DSA. Using the AES (software) or ChaCha based solutions it is around ten to fifteen times faster.

Our adaptively secure scheme adds an overhead of about 2.5, relative to the selectively secure Sahai-Waters.

Finally, we note that our scheme is conducive to leveraging parallelism. In Appendix A we describe a slight generalization to larger width GGM trees. For small increases in width it is feasible to utilize certain PRG structures that allow for computing multiple bits of the PRG output in parallel.

Acknowledgements

We thank Amit Sahai for observing that a wider tree structure in tandem with a parallelizable PRG could be conducive to leveraging parallelism.

References

[BGI$^+$01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In *Lecture Notes in Computer Science*, pages 1–18. Springer-Verlag, 2001.

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In Hugo Krawczyk, editor, *Public-Key Cryptography PKC 2014*, volume 8383 of *Lecture Notes in Computer Science*, pages 501–519. Springer Berlin Heidelberg, 2014.

[BHJ$^+$14] Florian Böhl, Dennis Hofheinz, Tibor Jager, Jessica Koch, and Christoph Striecks. Confined guessing: New signatures from standard assumptions. *Journal of Cryptology*, pages 1–33, 2014.

[BSW11] Elette Boyle, Gil Segev, and Daniel Wichs. Fully leakage-resilient signatures. In Kenneth G. Paterson, editor, *Advances in Cryptology EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 89–108. Springer Berlin Heidelberg, 2011.

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. Cryptology ePrint Archive, Report 2013/352, 2013. `http://eprint.iacr.org/`.

[Cha14] ChaCha stream cipher implementation, May 2014. `http://cr.yp.to/streamciphers/timings/estreambench/submissions/salsa20/chacha8/ref/chacha.c`.

[CHKP10] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In Henri Gilbert, editor, *Advances in Cryptology EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 523–552. Springer Berlin Heidelberg, 2010.

[CK12] Melissa Chase and Markulf Kohlweiss. A new hash-and-sign approach and structure-preserving signatures from DLIN. In *Proceedings of the 8th International Conference on Security and Cryptography for Networks*, SCN'12, pages 131–148, Berlin, Heidelberg, 2012. Springer-Verlag.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[DN94] Cynthia Dwork and Moni Naor. An efficient existentially unforgeable signature scheme and its applications. In *Proceedings of the 14th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '94, pages 234–246, London, UK, UK, 1994. Springer-Verlag.

[FS12] Dario Fiore and Dominique Schröder. Uniqueness is a different story: Impossibility of verifiable random functions from trapdoor permutations. In *Proceedings of the 9th International Conference on Theory of Cryptography*, TCC'12, pages 636–653, Berlin, Heidelberg, 2012. Springer-Verlag.

[GGH$^+$13a] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. Cryptology ePrint Archive, Report 2013/451, 2013.

[GGH$^+$13b] Sanjam Garg, Craig Gentry, Shai Halevi, Amit Sahai, and Brent Waters. Attribute-based encryption for circuits from multilinear maps. In Ran Canetti and JuanA. Garay, editors, *Advances in Cryptology CRYPTO 2013*, volume 8043 of *Lecture Notes in Computer Science*, pages 479–499. Springer Berlin Heidelberg, 2013.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, August 1986.

[GGP10] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *Proceedings of the 30th Annual Conference on Advances in Cryptology*, CRYPTO'10, pages 465–482, Berlin, Heidelberg, 2010. Springer-Verlag.

[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, April 1988.

[Gol06] Oded Goldreich. *Foundations of Cryptography: Volume 1.* Cambridge University Press, New York, NY, USA, 2006.

[GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In *Proceedings of the 45th annual ACM symposium on Symposium on theory of computing*, STOC '13, pages 545–554, New York, NY, USA, 2013. ACM.

[HW09] Susan Hohenberger and Brent Waters. Short and stateless signatures from the RSA assumption. In Shai Halevi, editor, *Advances in Cryptology - CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 654–670. Springer Berlin Heidelberg, 2009.

[Knu81] Donald E. Knuth. *The Art of Computer Programming, Volume 2 (2nd Ed.): Seminumerical Algorithms.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1981.

[KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer &#38; Communications Security*, CCS '13, pages 669–684, New York, NY, USA, 2013. ACM.

[KR00] Hugo Krawcyzk and Tal Rabin. Chameleon signatures. In *Network and Distributed Systems Security Symposium*, 2000.

[LW12] Allison Lewko and Brent Waters. New proof methods for attribute-based encryption: Achieving full security through selective techniques. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 180–198. Springer Berlin Heidelberg, 2012.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *Advances in Cryptology–EUROCRYPT 2012*, pages 700–718. Springer, 2012.

[PHGR13] B. Parno, J. Howell, C. Gentry, and M. Raykova. Pinocchio: Nearly practical verifiable computation. In *Security and Privacy (SP), 2013 IEEE Symposium on*, pages 238–252, May 2013.

[Seo14] Jae Hong Seo. Short signatures from diffie-hellman, revisited: Sublinear public key, CMA security, and tighter reduction. Cryptology ePrint Archive, Report 2014/138, 2014. `http://eprint.iacr.org/`.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: Deniable encryption, and more. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, STOC '14, pages 475–484, New York, NY, USA, 2014. ACM.

A Puncturable PRF from Large Width GGM

There have been several prior constructions [BW13, BGI14, KPTZ13] of puncturable PRFs from the tree-based construction of [GGM86]. In this section we show a simple extension to a 2^c-ary tree for some constant c. One can view this as an application of the Dwork-Naor [DN94] put into the [GGM86] context. The primary motivation of this construction is in the construction of PRFs from PRGs which have a large cost to seed, however then transform input relatively quickly. In this case is it reasonable to consider a value $c > 1$.

A.1 Puncturable PRFs from [GGM86] via [DN94]

Let the puncturable PRF have message size l bits and output size λ bits. Consider a construction where an input message is broken into l/c words each of size c-bits. Thus we can write $x = w_1 \dots w_d$ where $w_i \in \{0,1\}^c$ and $d = l/c$.[4]. Let G be a PRG with input size λ bits and output size $\lambda \cdot 2^c$ bits. Clearly c should be a small constant, due to the exponentially dependency in the output size. We define auxiliary functions $G_w : \{0,1\}^\lambda \to \{0,1\}^\lambda$ as follows: $G(x) = G_1(x) \| G_2(x) \| \dots \| G_{2^c}(x)$.

Construction

- Setup(1^λ) : Let K be a random λ-bit string. Output key K.
- Puncture(K, x) :

```
Write $x \in \{0,1\}^l$ as $w_1 \dots w_d$.
**for** $i = 1$ to $d$ **do**
    **for** $w \in \{0,1\}^c \backslash \{w_i\}$ **do**
        $F_{K,i,w} \leftarrow G_w(G_{w_{i-1}} \dots G_{w_2}(G_{w_1}(K)) \dots))$
    **end for**
**end for**
Let $K\{x\} = (F_{K,i,w})_{i \in [1,d], w \in \{0,1\}^l \backslash \{w_i\}}$. Output key $K\{x\}$.
```

- Eval$(K\{x\}, y)$:

```
**if** $y = x$ **then**
    output $\perp$
**else**
    Let $x = w_1 \dots w_d$.
    Write $y = w_1 \dots w_{i-1} w w'_{i+1} \dots w'_d$ where $w \neq w_i$.
    Output $G_{w'_d}(G_{w'_{d-1}}(\dots G_{w'_{i+1}}(F_{K,i,w}) \dots))$.
**end if**
```

Security

Theorem 2. *The above scheme is a selectively secure puncturable PRF if G is a secure PRG stretching λ bits to $\lambda \cdot 2^c$ bits.*

Proof. We prove the above theorem by defining the following sequence of hybrids.

Hybrid 1

1. Choose K at random in $\{0,1\}^\lambda$.
2. Receive an l-bit message x, to be punctured on.
3. Puncture K on x as follows

```
Let $x = w_1 \dots w_d$.
**for** $i = 1$ to $d$ **do**
    **for** $w \in \{0,1\}^c \backslash \{w_i\}$ **do**
        $F_{K,i,w} \leftarrow G_w(G_{w_{i-1}} \dots G_{w_2}(G_{w_1}(K)) \dots))$
    **end for**
**end for**
Let $K\{x\} = (F_{K,i,w})_{i \in [1,d], w \in \{0,1\}^l \backslash \{w_i\}}$.
```

4. Send key $K\{x\}$.
5. Send $G_{w_d}(G_{w_{d-1}}(\dots G_{w_1}(K) \dots))$.

[4]Note that inputs can always be zero padded to ensure l is a multiple of c

Hybrid $2[j] : j \in [0, d]$

1. Choose K at random in $\{0,1\}^\lambda$.
2. Receive an l-bit message x, to be punctured on.
3. Puncture K on x as follows

```
Let $x = w_1 \dots w_d$.
for $i = 1$ to $j$ do
    for $w \in \{0,1\}^c \backslash \{w_i\}$ do
        $F_{K,i,w} \overset{\$}{\leftarrow} \{0,1\}^\lambda$
    end for
end for
Let $y \overset{\$}{\leftarrow} \{0,1\}^\lambda$.
for $i = j+1$ to $d$ do
    for $w \in \{0,1\}^c \backslash \{w_i\}$ do
        $F_{K,i,w} \leftarrow G_w(G_{w_{i-1}} \dots G_{w_{j+1}}(y)) \dots))$
    end for
end for
Let $K\{x\} = (F_{K,i,w})_{i \in [1,d], w \in \{0,1\}^l \backslash \{w_i\}}$.
```

4. Send key $K\{x\}$.
5. Send $G_{w_d}(G_{w_{d-1}}(\dots G_{w_{j+1}}(y) \dots))$.

Hybrid 3

1. Choose K at random in $\{0,1\}^\lambda$.
2. Receive an l-bit message x, to be punctured on.
3. Puncture K on x as follows

```
Let $x = w_1 \dots w_d$.
for $i = 1$ to $d$ do
    for $w \in \{0,1\}^c \backslash \{w_i\}$ do
        $F_{K,i,w} \overset{\$}{\leftarrow} \{0,1\}^\lambda$
    end for
end for
Let $K\{x\} = (F_{K,i,w})_{i \in [1,d], w \in \{0,1\}^l \backslash \{w_i\}}$.
```

4. Send key $K\{x\}$.
5. Let $y' \overset{\$}{\leftarrow} \{0,1\}^\lambda$. Send y'.

Hybrid $4[j] : j \in [0, d]$

1. Choose K at random in $\{0,1\}^\lambda$.
2. Receive an l-bit message x, to be punctured on.
3. Puncture K on x as follows

```
Let $x = w_1 \dots w_d$.
for $i = 1$ to $d - j$ do
    for $w \in \{0,1\}^c \backslash \{w_i\}$ do
        $F_{K,i,w} \overset{\$}{\leftarrow} \{0,1\}^\lambda$
    end for
end for
Let $y \overset{\$}{\leftarrow} \{0,1\}^\lambda$.
for $i = d+1-j$ to $d$ do
    for $w \in \{0,1\}^c \backslash \{w_i\}$ do
```

```
        $F_{K,i,w} \leftarrow G_w(G_{w_{i-1}} \dots G_{w_{j+1}}(y)) \dots))$
    **end for**
**end for**
Let $K\{x\} = (F_{K,i,w})_{i \in [1,d], w \in \{0,1\}^l \setminus \{w_i\}}$.
```

4. Send key $K\{x\}$.
5. Let $y' \xleftarrow{\$} \{0,1\}^\lambda$. Send y'.

Hybrid 5

1. Choose K at random in $\{0,1\}^\lambda$.
2. Receive an l-bit message x, to be punctured on.
3. Puncture K on x as follows

```
Let $x = w_1 \dots w_d$.
**for** $i = 1$ to $d$ **do**
    **for** $w \in \{0,1\}^c \setminus \{w_i\}$ **do**
        $F_{K,i,w} \leftarrow G_w(G_{w_{i-1}} \dots G_{w_2}(G_{w_1}(K)) \dots))$
    **end for**
**end for**
Let $K\{x\} = (F_{K,i,w})_{i \in [1,d], w \in \{0,1\}^l \setminus \{w_i\}}$.
```

4. Send key $K\{x\}$.
5. Let $y' \xleftarrow{\$} \{0,1\}^\lambda$. Send y'.

Claim 9. *Suppose there exists a PPT adversary $\mathcal{A}$ such that $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid2[j-1]}} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid2[j]}} = \epsilon$ for $j \in [1,d]$. Then the advantage of $\mathcal{A}$ in distinguishing the output of G is at least ϵ.*

Proof. $\mathcal{B}$ receives point $x = w_1 \dots w_d$ from $\mathcal{A}$. $\mathcal{B}$ receives PRG challenge $z = z_1 \dots z_{2^c}$, where z is either the output of G on string y, or is a truly random string of length $\lambda \cdot 2^c$. $\mathcal{B}$ sets $F_{K,i,w} \xleftarrow{\$} \{0,1\}^\lambda : i < j$. $\mathcal{B}$ sets $F_{K,j,w} = z_w : w \in \{0,1\}^l \setminus \{w_j\}$. $\mathcal{B}$ sets $F_{K,i,w} = G_w(G_{w_{i-1}}(\dots G_{w_{j+1}}(z_w) \dots)) : i > j, w \in \{0,1\}^l \setminus \{w_i\}$. Lastly $\mathcal{B}$ sends $G_{w_d}(G_{w_{d-1}}(\dots G_{w_{j+1}}(z_w) \dots))$. If $z = G(y)$, then $\mathcal{B}$ is in Hybrid $2[j-1]$, otherwise if z is truly random, then $\mathcal{B}$ is in Hybrid $2[j]$. It follows the advantage of $\mathcal{B}$ in distinguishing the output of G is ϵ. □

Claim 10. *Suppose there exists a PPT adversary $\mathcal{A}$ such that $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid4[j-1]}} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid4[j]}} = \epsilon$ for $j \in [1,d]$. Then the advantage of $\mathcal{A}$ in distinguishing the output of G is at least ϵ.*

Proof. $\mathcal{B}$ receives point $x = w_1 \dots w_d$ from $\mathcal{A}$. $\mathcal{B}$ receives PRG challenge $z = z_1 \dots z_{2^c}$, where z is either the output of G on string y, or is a truly random string of length $\lambda \cdot 2^c$. $\mathcal{B}$ sets $F_{K,i,w} \xleftarrow{\$} \{0,1\}^\lambda : i > d-j$. $\mathcal{B}$ sets $F_{K,d-j,w} = z_w : w \in \{0,1\}^l \setminus \{w_{d-j}\}$. $\mathcal{B}$ sets $F_{K,i,w} = G_w(G_{w_{i-1}}(\dots G_{w_{d+1-j}}(z_w) \dots)) : i > d-j, w \in \{0,1\}^l \setminus \{w_i\}$. Lastly $\mathcal{B}$ sends $G_{w_d}(G_{w_{d-1}}(\dots G_{w_{d+1-j}}(z_w) \dots))$. If $z = G(y)$, then $\mathcal{B}$ is in Hybrid $4[j-1]$, otherwise if z is truly random, then $\mathcal{B}$ is in Hybrid $4[j]$. It follows the advantage of $\mathcal{B}$ in distinguishing the output of G is ϵ. □

First observe that the Hybrids in each of the following pairs of are identical: $(1, 2[0]), (2[d], 3), (3, 4[0])$ and $(4[d], 5)$. Thus claims 5 and 6 imply that the advantage of any PPT adversary in distinguishing Hybrids 1 and 5 is at most $2d \cdot \epsilon_{PRG}$, where ϵ_{PRG} is the maximal distinguishing adversary of any adversary in distinguishing the output of the PRG. On the other hand, any adversary distinguishing Hybrids 1 and 5 distinguishes the output of the puncturable PRF. Since ϵ_{PRG} is neglgible by assumption, it follows that no adversary has more than a negligible advantage in distinguishing the output of the puncturable PRF.

□

B Proof of Security of Variable-Length Domain Puncturable PRF

Theorem 3. *The scheme given in Section 5 is a selectively secure puncturable PRF if G is a secure PRG stretching λ bits to 3λ bits.*

Proof. We prove the above theorem by defining the following sequence of hybrids.

Hybrid 1

```
1. Choose $K$ at random in $\{0,1\}^\lambda$.
2. Receive an $v$-bit message $x$, to be punctured on.
3. Puncture $K$ on $x$ as follows
     Write $x$ as $b_1 \dots b_v$, where $v \le l$.
     for  $i = 1$ to $v-1$  do
         $F_{K,i} \leftarrow G_{\overline{b}_i}(G_{b_{i-1}}(\dots(G_{b_1}(K))\dots))$
         $F_{K,i,\perp} \leftarrow G_\perp(G_{b_i}\dots(G_{b_1}(K))\dots)$
     end for
     $F_{K,v} \leftarrow G_{\overline{b}_v}(G_{b_{v-1}}(\dots(G_{b_1}(K))\dots))$
     $F_{K,v+1,0} \leftarrow G_0(G_{b_v}(\dots(G_{b_1}(K))\dots))$
     $F_{K,v+1,1} \leftarrow G_1(G_{b_v}(\dots(G_{b_1}(K))\dots))$
   Let $K\{x\} = (F_{K,i}, F_{K,i,\perp})_{i\in[1,v-1]}, F_{K,v}, (F_{K,v+1,b})_{b\in[0,1]}$.
4. Send key $K\{x\}$.
5. Send $G_\perp(G_{b_v}(\dots(G_{b_1}(K))\dots))$.
```

Hybrid $2[j] : j \in [0, v]$

```
1. Choose $K$ at random in $\{0,1\}^\lambda$.
2. Receive an $v$-bit message $x$, to be punctured on.
3. Puncture $K$ on $x$ as follows
     Write $x$ as $b_1 \dots b_v$, where $v \le l$.
     for  $i = 1$ to $j$  do
         $y_{i,0} \xleftarrow{\$} \{0,1\}^\lambda$
         $y_{i,1} \xleftarrow{\$} \{0,1\}^\lambda$
         $y_{i,\perp} \xleftarrow{\$} \{0,1\}^\lambda$
     end for
     for  $i = 1$ to $j$  do
         $F_{K,i} \leftarrow y_{i,0}$
         $F_{K,i,\perp} \leftarrow y_{i,\perp}$
     end for
     for  $i = j+1$ to $v-1$  do
         $F_{K,i} \leftarrow G_{\overline{b}_i}(G_{b_{i-1}}\dots(y_{j,1})\dots)$
         $F_{K,i,\perp} \leftarrow G_\perp(G_{b_i}(\dots(y_{j,\perp})\dots))$
     end for
     $F_{K,v} \leftarrow G_{\overline{b}_v}(G_{b_{v-1}}(\dots(y_{j,1})\dots))$
     $F_{K,v+1,0} \leftarrow G_0(G_{b_v}(\dots(y_{j,1})\dots))$
     $F_{K,v+1,1} \leftarrow G_1(G_{b_v}(\dots(y_{j,1})\dots))$
   Let $K\{x\} = (F_{K,i}, F_{K,i,\perp})_{i\in[1,v-1]}, F_{K,v}, (F_{K,v+1,b})_{b\in[0,1]}$.
4. Send key $K\{x\}$.
5. Send $G_\perp(G_{b_v}(\dots(y_{j,1})\dots))$.
```

Hybrid 3

1. Choose K at random in $\{0,1\}^\lambda$.
2. Receive an v-bit message x, to be punctured on.
3. Puncture K on x as follows

```
    Write $x$ as $b_1 \dots b_v$, where $v \leq l$.
    for $i = 1$ to $v - 1$ do
        $y_{i,0} \xleftarrow{\$} \{0,1\}^\lambda$
        $y_{i,1} \xleftarrow{\$} \{0,1\}^\lambda$
        $y_{i,\perp} \xleftarrow{\$} \{0,1\}^\lambda$
    end for
    for $i = 1$ to $v - 1$ do
        $F_{K,i} \leftarrow y_{i,0}$
        $F_{K,i,\perp} \leftarrow y_{i,\perp}$
    end for
    $y_{v,0} \xleftarrow{\$} \{0,1\}^\lambda$
    $y_{v+1,0} \xleftarrow{\$} \{0,1\}^\lambda$
    $y_{v+1,1} \xleftarrow{\$} \{0,1\}^\lambda$
    $y_{v+1,\perp} \xleftarrow{\$} \{0,1\}^\lambda$
    $F_{K,v} \leftarrow y_{v,0}$
    $F_{K,v+1,0} \leftarrow y_{v+1,0}$
    $F_{K,v+1,1} \leftarrow y_{v+1,1}$
```

   Let $K\{x\} = (F_{K,i}, F_{K,i,\perp})_{i\in[1,v-1]}, F_{K,v}, (F_{K,v+1,b})_{b\in[0,1]}$.
4. Send key $K\{x\}$.
5. Send $y_{v+1,\perp}$.

Hybrid $4[j] : j \in [1, v+1]$

1. Choose K at random in $\{0,1\}^\lambda$.
2. Receive an v-bit message x, to be punctured on.
3. Puncture K on x as follows

```
    Write $x$ as $b_1 \dots b_v$, where $v \leq l$.
    for $i = 1$ to $v + 1 - j$ do
        $y_{i,0} \xleftarrow{\$} \{0,1\}^\lambda$
        $y_{i,1} \xleftarrow{\$} \{0,1\}^\lambda$
        $y_{i,\perp} \xleftarrow{\$} \{0,1\}^\lambda$
    end for
    for $i = 1$ to $v + 1 - j$ do
        $F_{K,i} \leftarrow y_{i,0}$
        $F_{K,i,\perp} \leftarrow y_{i,\perp}$
    end for
    for $i = v + 2 - j$ to $v - 1$ do
        $F_{K,i} \leftarrow G_{\overline{b}_i}(G_{b_{i-1}} \dots (y_{v+1-j,1}) \dots)$
        $F_{K,i,\perp} \leftarrow G_\perp(G_{b_i}(\dots (y_{v+1-j,\perp}) \dots))$
    end for
    $F_{K,v} \leftarrow G_{\overline{b}_v}(G_{b_{v-1}}(\dots (y_{v+1-j,1}) \dots))$
    $y_{v+1,\perp} \xleftarrow{\$} \{0,1\}^\lambda$
    $F_{K,v+1,0} \leftarrow G_0(G_{b_v}(\dots (y_{v+1-j,1}) \dots))$
    $F_{K,v+1,1} \leftarrow G_1(G_{b_v}(\dots (y_{v+1-j,1}) \dots))$
    Let $K\{x\} = (F_{K,i}, F_{K,i,\perp})_{i\in[1,v-1]}, F_{K,v},\ (F_{K,v+1,b})_{b\in[0,1]}$.
4. Send key $K\{x\}$.
5. Send $y_{v+1,\perp}$.
```

□

Hybrid 5

```
1. Choose $K$ at random in $\{0,1\}^\lambda$.
2. Receive an $v$-bit message $x$, to be punctured on.
3. Puncture $K$ on $x$ as follows
      Write $x$ as $b_1 \dots b_v$, where $v \le l$.
      **for** $i = 1$ to $v-1$ **do**
          $F_{K,i} \leftarrow G_{\overline{b}_i}(G_{b_{i-1}}(\dots(G_{b_1}(K))\dots))$
          $F_{K,i,\perp} \leftarrow G_\perp(G_{b_i}\dots(G_{b_1}(K))\dots)$
      **end for**
      $F_{K,v} \leftarrow G_{\overline{b}_v}(G_{b_{v-1}}(\dots(G_{b_1}(K))\dots))$
      $y_{v+1,\perp} \overset{\$}{\leftarrow} \{0,1\}^\lambda$
      $F_{K,v+1,0} \leftarrow G_0(G_{b_v}(\dots(G_{b_1}(K))\dots))$
      $F_{K,v+1,1} \leftarrow G_1(G_{b_v}(\dots(G_{b_1}(K))\dots))$
    Let $K\{x\} = (F_{K,i}, F_{K,i,\perp})_{i\in[1,v-1]}, F_{K,v},\ (F_{K,v+1,b})_{b\in[0,1]}$.
4. Send key $K\{x\}$.
5. Send $y_{v+1,\perp}$.
```

Claim 11. *Suppose there exists a PPT adversary $\mathcal{A}$ such that $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid2[j-1]}} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid2[j]}} = \epsilon$ for $j \in [1, v]$. Then the advantage of $\mathcal{A}$ in distinguishing the output of G is at least ϵ.*

Proof. Simulator $\mathcal{B}$ receives point $x = b_1 \dots b_v$ from $\mathcal{A}$. $\mathcal{B}$ sets $y_{i,0} \overset{\$}{\leftarrow} \{0,1\}^\lambda, y_{i,\perp} \overset{\$}{\leftarrow} \{0,1\}^\lambda, F_{K,i} \leftarrow y_{i,0}, F_{K,i,\perp} \leftarrow y_{i,\perp} : i < j$. $\mathcal{B}$ is given $z = y_{j,0}\|y_{j,1}\|y_{j,\perp}$ as input which is either the output of G on a random λ-bit input, or is a truly random 3λ-bit string. $\mathcal{B}$ sets $F_{K,i} \leftarrow G_{\overline{b}_i}(G_{b_{i-1}} \dots (y_{j,1}) \dots), F_{K,i,\perp} \leftarrow G_\perp(G_{b_i}(\dots(y_{j,\perp})\dots)) : i \in [j, v-1]$. $\mathcal{B}$ sets $F_{K,v} \leftarrow G_{\overline{b}_v}(G_{b_{v-1}}(\ \dots(y_{j,1})\dots\ )), F_{K,v+1,0} \leftarrow G_0(G_{b_v}(\dots(y_{j,1})\dots)), F_{K,v+1,1} \leftarrow G_1(G_{b_v}(\dots(y_{j,1})\dots))$. $\mathcal{B}$ sends $K\{x\}$ and $G_\perp(G_{b_v}(\ \dots(y_{j,1})\dots\ ))$. If $\mathcal{B}$ is given $G(y_{j-1,1})$ for random $y_{j-1,1}$ then it is in Hybrid $2[j-1]$, otherwise if z is truly random, it is in Hybrid $2[j]$. The claims follows. □

Claim 12. *Suppose there exists a PPT adversary $\mathcal{A}$ such that $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid2[v]}} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid3}} = \epsilon$. Then the advantage of $\mathcal{A}$ in distinguishing the output of G is at least ϵ.*

Proof. Simulator $\mathcal{B}$ receives point $x = b_1 \dots b_v$ from $\mathcal{A}$. $\mathcal{B}$ sets $y_{i,0} \overset{\$}{\leftarrow} \{0,1\}^\lambda, y_{i,\perp} \overset{\$}{\leftarrow} \{0,1\}^\lambda, F_{K,i} \leftarrow y_{i,0}, F_{K,i,\perp} \leftarrow y_{i,\perp} : i \in [1, v]$. $\mathcal{B}$ is given $z = y_{v+1,0}\|y_{v+1,1}\|y_{v+1,\perp}$ as input which is either the output of G on a random λ-bit input, or is a truly random 3λ-bit string. $\mathcal{B}$ sets $F_{K,v} \leftarrow y_{v,0}, F_{K,v+1,0} \leftarrow y_{v+1,0}, F_{K,v+1,1} \leftarrow y_{v+1,1}$. $\mathcal{B}$ sends $K\{x\}$ and $y_{v+1,\perp}$. If $\mathcal{B}$ is given $G(y_{v,1})$ for random $y_{v,1}$ then it is in Hybrid $2[v]$, otherwise if z is truly random, it is in Hybrid 3. The claims follows. □

Claim 13. *Suppose there exists a PPT adversary $\mathcal{A}$ such that $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid3}} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid4[1]}} = \epsilon$. Then the advantage of $\mathcal{A}$ in distinguishing the output of G is at least ϵ.*

Proof. Simulator $\mathcal{B}$ receives point $x = b_1 \dots b_v$ from $\mathcal{A}$. $\mathcal{B}$ sets $y_{i,0} \overset{\$}{\leftarrow} \{0,1\}^\lambda, y_{i,\perp} \overset{\$}{\leftarrow} \{0,1\}^\lambda, F_{K,i} \leftarrow y_{i,0}, F_{K,i,\perp} \leftarrow y_{i,\perp} : i \in [1, v]$. $\mathcal{B}$ is given $z = y_{v+1,0}\|y'_{v+1,1}\|y_{v+1,\perp}$ as input which is either the output of G on a random λ-bit input, or is a truly random 3λ-bit string. $\mathcal{B}$ sets $F_{K,v} \leftarrow y_{v,0}, F_{K,v+1,0} \leftarrow y_{v+1,0}, F_{K,v+1,1} \leftarrow y_{v+1,1}$. $\mathcal{B}$ sends $K\{x\}$ and $y_{v+1,\perp} \overset{\$}{\leftarrow} \{0,1\}^\lambda$. If $\mathcal{B}$ is given $G(y_{v,1})$ for random $y_{v,1}$ then it is in Hybrid 4[1], otherwise if z is truly random, it is in Hybrid 3. The claims follows. □

Claim 14. *Suppose there exists a PPT adversary $\mathcal{A}$ such that $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid4[j-1]}} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid4[j]}} = \epsilon$ for $j \in [2, v+1]$. Then the advantage of $\mathcal{A}$ in distinguishing the output of G is at least ϵ.*

Proof. Simulator $\mathcal{B}$ receives point $x = b_1 \dots b_v$ from $\mathcal{A}$. $\mathcal{B}$ sets $y_{i,0} \xleftarrow{\$} \{0,1\}^\lambda, y_{i,\perp} \xleftarrow{\$} \{0,1\}^\lambda, F_{K,i} \leftarrow y_{i,0}, F_{K,i,\perp} \leftarrow y_{i,\perp} : i > v+1-j$. $\mathcal{B}$ is given $z = y_{v+1-j,0} \| y_{v+1-j,1} \| y_{v+1-j,\perp}$ as input which is either the output of G on a random λ-bit input, or is a truly random 3λ-bit string. $\mathcal{B}$ sets $F_{K,i} \leftarrow G_{\overline{b}_i}(G_{b_{i-1}} \dots (y_{v+1-j,1}) \dots)$, $F_{K,i,\perp} \leftarrow G_\perp(G_{b_i}(\dots (y_{v+1-j,\perp}) \dots)) : i \in [v+1-j, v-1]$. $\mathcal{B}$ sets $F_{K,v} \leftarrow G_{\overline{b}_v}(G_{b_{v-1}}(\dots (y_{v+1-j,1}) \dots))$, $F_{K,v+1,0} \leftarrow G_0(G_{b_v}(\dots (y_{v+1-j,1}) \dots)), F_{K,v+1,1} \leftarrow G_1(G_{b_v}(\dots (y_{v+1-j,1}) \dots))$. $\mathcal{B}$ sends $K\{x\}$ and $G_\perp(G_{b_v}(\dots (y_{v+1-j,1}) \dots))$. If $\mathcal{B}$ is given $G(y_{v-j,1})$ for random $y_{v-j,1}$ then it is in Hybrid $2[j-1]$, otherwise if z is truly random, it is in Hybrid $2[j]$. The claims follows. $\square$

Observe that Hybrid 1 is identical to Hybrid 2[0]. Similiarly Hybrid $4[v+1]$ is identical to Hybrid 5. Thus claims 8 - 11 imply that the advantage of any PPT adversary in distinguishing Hybrids 1 and 5 is at most $2(v+1) \cdot \epsilon_{PRG}$, where ϵ_{PRG} is the maximal distinguishing adversary of any adversary in distinguishing the output of the PRG. On the other hand, any adversary distinguishing Hybrids 1 and 5 distinguishes the output of the puncturable PRF. Since ϵ_{PRG} is negligible by assumption, it follows that no adversary has more than a negligible advantage in distinguishing the output of the puncturable PRF.