Short and Adjustable Signatures

Xiong Fan* Juan Garay† Payman Mohassel‡

Abstract

Motivated by the problem of one-time password generation, we introduce the notion of *adjustable signature schemes* that allow the length of a signature to be adjusted—at the setup, signing or verification stages, depending on the application. We provide security definitions that precisely capture the trade-off between signature length and security for such schemes. We then provide both concrete and general feasibility results.

As a feasibility result, we provide the first instantiation of all variants of adjustable signatures based on indistinguishability obfuscation. Our starting point is the state-of-the-art construction by Ramchen and Waters [CCS 2014]. We observe that their scheme fails to meet our requirements for an adjustable signatures scheme, and enhance it to obtain *shorter* (and adjustable) signatures, *faster* signing and strong unforgeability.

For the simpler case of setup-adjustable signatures, we also provide a concrete construction based on the BLS signature scheme, by instantiating it using smaller group sizes that yield shorter signature lengths while providing reasonable security. We implement this scheme for various signature sizes an report on its efficiency.

1 Introduction

One-Time Passwords (OTPs) are widely used in practice to either replace or strengthen traditional password-based authentication mechanisms. Examples include Google Authenticator [Goo] and RSA's SecureID tokens [RSA78]. In most use cases, a user holds a trusted device (smartphone, security token, etc.) that generates a short, human-readable string (the OTP) that the user presents to a server for authentication. The server generates the OTP on its own and compares it with the string provided by the user. The authentication succeeds if the two match and fails otherwise. OTP-based solutions have two main advantages over traditional passwords: (i) OTPs have higher entropy than human-chosen passwords as they are often generated using cryptographic primitives, and (ii) each OTP is only used once, keeping future sessions protected even if a current OTP is compromised.

To the best of our knowlege, all existing OTP-based authentication solutions are based on a *symmetric-key setup* where the user and the server agree on a long-term secret key k that is used to generate future OTPs for that user. A common implementation follows the "Time-based One-Time Password" (TOTP) algorithm (IETF RFC 6238 [For]), where HMAC [BCK96] is used as a pseudorandom function to generate time-based OTPs. In particular, the OTP is generated by first computing $\mathsf{HMAC}_{\mathsf{k}}(\mathsf{t})$ for a synchronized time value t (see the RFC for details on how t is chosen) and then *truncating* the output of HMAC to an appropriate length that is short enough for humans to type, but still reasonably hard for an adversary to guess. Note that there is a natural trade-off between the size of the OTP and the guessing probability of the adversary, which can be easily adjusted by truncating less/more bits of the HMAC output.

*Cornell University, xfan@cs.cornell.edu. Work partly done while interning at Yahoo Labs.

†Yahoo Labs, garay@yahoo-inc.com.

‡Visa Research, pmohasse@visa.com. Work done while at Yahoo Labs.

A major drawback of any symmetric-key based OTP solution is its susceptibility to a user-data breach which nowadays is an all-too-common ocurrence. In particular, the server needs to store in a user database, a unique secret key for each user of its service, and use that key to generate OTPs for each authentication session. Furthermore, the server cannot just store a hash of the user's secret key (as done with passwords, for example [PM99]), as the full key is needed in order to generate the OTPs. As a consequence, a breach of the user database implies a breach of *all* secret keys, which in turn renders the OTP-based solution ineffective. This motivates OTP generation using public-key primitives where the server only needs to store a per-user public-key, and thus security is not compromised in case of a server breach.

OTP based on short signatures. A natural candidate is to generate OTPs using a digital signatures scheme, where the server stores the verification key vk and the user stores the associated signing key sk. The client generates an OTP by computing OTP$= \mathsf{Sign}(\mathsf{sk}, t)$ for a synchronized time t, and the server verifies it by computing $\mathsf{Verify}(\mathsf{vk}, t,$OTP). The authentication passes if and only if the verification does, with the hardness of guessing the OTP being implied by the unforgeability of the signature.

The problem with this approach is that signatures are too long for humans to type. The state-of-the-art short-signature scheme based on well-studied number-theoretic assumptions is that of [BLS01], generating signatures that are at least 160-bits (20 characters) long. More importantly, and unlike the symmetric-key scenario, the work-around of truncating the signature for a length/security trade-off fails, since a truncated signature cannot be verified by the verification algorithm.

Alternatively, based on the stronger *indistinguishability obfuscation* assumption ($i\mathcal{O}$) [BGI$^+$01, GGH$^+$13], Sahai and Waters [SW14] and Ramchen and Waters [RW14] show how to design even shorter signatures. For the case of selective security, the scheme of [SW14] lends itself to truncation since the signature only consists of a PRF output. For a fully secure signature scheme, however, even the $i\mathcal{O}$-based schemes fall short of achieving an acceptable solution. In particular, the signature scheme in [RW14] contains a tag t that is as long as the security parameter, and a string s of PRF-output length. This almost doubles the signatures length, and it is not clear how to adjust the length by truncating both t and s without breaking the functionality of the scheme. This motivates the formulation and study of the notion of *adjustability* of signature schemes.

Our contributions. We introduce the notion of *adjustable signature schemes* that allow the signature length to be adjusted at setup, signing or verification stage, depending on the application scenario. In case of *setup-adjustable* signatures, the key generation algorithm takes a length parameter ℓ as input, and generates the key-pair $(\mathsf{sk}, \mathsf{vk})$ accordingly. Any signature generated using sk will be ℓ-bits long and the scheme provides a level security that is correlated with ℓ. A drawback of this variant is that one needs to re-run the setup algorithm in order to increase/decrease the signature length, as mandated for example by the current security needs. Nevertheless, setup-adjustable signatures are the simplest notion of adjustability we consider, and the one with the most efficient instantiation.

In the case of *signing-adjustable* and *verification-adjustable* signatures, the length of the signature is decided at signing/verification time and can change from one invocation to the next. These notions provide much higher flexibity as one can decide the level of security on-the-fly, without re-running the setup phase or prior coordination. In the OTP generation scenario, a signing-adjustable scheme allows the user device to change the OTP lenght spontaneously, with the server still being able to verify its authenticity using the same verification key; a verification-adjustable scheme, on the other hand, allows the user to precompute full length signatures (OTPs) beforehand and only adjust their size during the actual verification (authentication) phase.

We provide the necessary syntax and security definitions that concretely capture the trade-off between signature length and security level for all three variants of adjustable signature schemes. We then provide

both concrete and general feasibility results.

It turns out that achieving the new notions—in particular of signing- and verification-adjustable signatures—proves to be a challenging problem, as it is not obvious how to adjust group sizes/security level of a standard signature scheme at the signing or verification stage without making changes to the setup. Nevertheless, we propose the first feasibility results for all variants based on indistinguishability obfuscation. Our starting point is the state-of-the-art $i\mathcal{O}$-based short signature scheme by Ramchen and Waters [RW14], which itself is an improved variant of the signature construction proposed by Sahai and Waters in [SW14]. In order to a achieve a fully secure signature scheme, Ramchen and Waters associate a tag with each signature, which plays an essential role in the security proof. The addition of tags to each signature makes the signatures longer (almost twice as long), and harder to truncate, violating the main goal of adjustable signatures and our motivating application.

We enhance the scheme of [RW14] in order to design a new short and adjustable signature that improves theirs in several ways: (1) our signatures are shorter as they do not include the tag, and can easily be made adjustable since they only contain a PRF output that can be truncated, (2) our signature schemes achieve *strong unforgeability*, and (3) signing is noticeably *faster*.

At a high level, in our signature schemes, instead of generating the tag at random and including it with the signatures, we generate the tag both at signing and verification, "on the fly" and as a deterministic function of the message being signed and the verification key. We achieve this by including a set of random strings (one pair of strings for each bit of the message) in the verification key, and compute the temporary tag by XOR-ing one random string from each pair based on each bit value of the message. As a result, the tag is no longer sent with the signature. Furthermore, in our scheme each tag value is, with all but negligible probability, unique to a message and as a result the generated signatures is also unique, hence achieving strong unforgeability for free. Finally, we optimize the scheme of [RW14], by reducing the signing cost by almost a factor of two.

We also consider the design of concretely efficient adjustable signature schemes. We show how to instantiate a setup-adjustable scheme concretely on specific curves, based on the BLS signature scheme [BLS01]. Specifically, we explore how to instantiate BLS using smaller-size groups that provide reasonable security given known attacks against discrete log, while yielding short signature lengths. We implement our construction for various signature lengths and report on its efficiency.

Related work. The design of schemes supporting short signatures has always been an important efficiency goal. In [BLS01], Boneh, Lynn and Shacham proposed a signature scheme based on bilinear maps in the random oracle model whose signature length is half the size of DSA signatures [oST] for a similar level of security. Later, Boneh and Boyen [BB04] described a signature scheme where signatures are almost as short as the BLS signatures, but whose security holds in the standard model.

In these pairing-based constructions, verification is the most time-consuming algorithm, since it is at this stage where the pairing operations are performed. With that motivation, Camenisch *et al.* [CHP07] proposed the first batch verifier for messages from many signers in the standard model and with a verification complexity where the dominant operation is independent of the number of signatures that are to be verified. Extensions of this batch verifier approach (to identity-based signatures, group signatures, etc.) were presented by Ferrara *et al.* [FGHP09].

Bellare and Rogaway [BR96] initiated the study of the exact security of digital signatures (RSA and Rabin), making precise reductions from a forger to the algorithm that solves the underlying hard problem. Micali and Reyzin [MR02] adapted the concrete security paradigm to Fiat-Shamir-like signature schemes that yield better concrete security than those based on the original Fiat-Shamir method.

As mentioned above, some of our adjustable schemes are based on indistinguishability obfuscation ($i\mathcal{O}$), the first realization of which was proposed by Garg *et al.* [GGH$^+$13] based on multilinear maps. Since its

introduction, indistinguihsability obfuscation has enabled the design of numerous cryptographic primitives and protocols. One such primitive is short signatures. Sahai and Waters in [SW14] gave the first construction of a selectively secure short signature scheme based on $i\mathcal{O}$, with follow-up work by Ramchen and Waters [RW14] showing how to extend it to full security.

Organization of the paper. The balance of the paper is organized as follows. We present notation and the basic cryptographic notions used in the paper in Section 2. The syntax and security definition of all variants of adjustable signatures are presented in Section 3. The $i\mathcal{O}$-based constructions are given in Section 4, while Section 5 is dedicated to a concrete proposal for setup-adjustable signatures based on bilinear groups. For ease of readability, complementary material and some of the proofs are presented in the appendix.

2 Preliminaries

In this section we present the notation, cryptographic notions and building blocks used throughout the paper. We use λ to denote the security parameter and PPT to denote a probabilistic polynomial-time function of the λ. Furthermore, since the nature of our work requires discussing concrete efficiency and security (cf. [BR96]), we use the function $t : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ (taking security parameter and a length parameter as input) to denote a concrete bound on the running time of an algorithm in a fixed computational model; we call such an algorithm a t-time algorithm. We use a similar concrete function $\epsilon : \mathbb{N} \times \mathbb{N} \to \mathbb{R}$ to denote a bound on the probability of an algorithm (Adversary)'s success. We will use $\sigma|_\ell$ to denote the bit string σ truncated to its least significant ℓ bits, and $\boldsymbol{m}[i]$ to denote the i-th bit of vector $\boldsymbol{m}$. We will be using ℓ to denote the length (in number of bits) of the (adjustable) signatures, and assume it polynomially related to the security parameter, i.e., $\ell = \mathrm{poly}(\lambda)$.

Indistinguishability obfuscation. We first present the definition of indistinguishability obfuscation ($i\mathcal{O}$) as it appeared in [GGH+13].

Definition 2.1. *A uniform* PPT *machine $i\mathcal{O}$ is called an* indistinguishability obfuscator *for a circuit class $\{\mathcal{C}_\lambda\}$ if the following conditions are satisfied:*

- *For all security parameters λ, all circuits $C \in \mathcal{C}_\lambda$, and all inputs x, we have that*
$$\mathbf{Pr}[C'(x) = C(x) | C' \leftarrow i\mathcal{O}(\lambda, C)] = 1.$$
- *For any* PPT *adversaries* $\mathsf{Samp}, \mathsf{D}$*, there exists a negligible function* $\mathsf{negl}(\cdot)$*, such that the following holds: if*
$$\mathbf{Pr}[\forall x, C_0(x) = C_1(x) | (C_0, C_1, \sigma) \leftarrow \mathsf{Samp}(1^\lambda)] > 1 - \mathsf{negl}(\lambda),$$
then we have
$$\begin{aligned} &|\mathbf{Pr}[\mathsf{D}(\sigma, i\mathcal{O}(\lambda, C_0)) = 1 | (C_0, C_1, \sigma) \leftarrow \mathsf{Samp}(1^\lambda)] \\ &- \mathbf{Pr}[\mathsf{D}(\sigma, i\mathcal{O}(\lambda, C_1)) = 1 | (C_0, C_1, \sigma) \leftarrow \mathsf{Samp}(1^\lambda)]| \leq \mathsf{negl}(\lambda). \end{aligned}$$

In this paper we will make use of such indistinguishability obfuscators for all polynomial-size circuits.

Definition 2.2 (Indistinguishability obfuscator for P/poly)**.** *A uniform* PPT *machine $i\mathcal{O}$ is called an* indistinguishability obfuscator for P/poly *if the following holds: Let $\mathcal{C}_\lambda$ be the class of circuits of size at most λ. Then $i\mathcal{O}$ is an indistinguishability obfuscator for the class $\{\mathcal{C}_\lambda\}$.*

Indistinguishability obfuscators for all polynomial-size circuits have been constructed under novel hardness assumptions in [GGH+13, CLT15].

Puncturable PRFs. We now recall the notion of *puncturable* PRFs, a variant of "constrained" PRFs introduced in [BW13, BGI14, KPTZ13]. Roughly speaking, puncturable PRFs are PRFs that can be defined on all bit strings of a certain length, except for any polynomial-size set of inputs.

Definition 2.3 (Puncturable PRFs)**.** *A family of* puncturable PRFs F *is given by a triple of algorithms* Key_F, $\mathsf{Puncture}_F$, Eval_F*, and a pair of computable functions* $n(\cdot), m(\cdot)$*, satisfying the following conditions:*

- *For every* PPT *adversary* $\mathcal{A}$*, such that* $\mathcal{A}(1^\lambda)$ *outputs a set* $T \subset \{0,1\}^{n(\lambda)}$*, then for all* $x \in \{0,1\}^{n(\lambda)}$ *where* $x \notin T$*, we have that*

$$\mathbf{Pr}[\mathsf{Eval}_F(K,x) = \mathsf{Eval}_F(K_T,x) | K \leftarrow \mathsf{Key}_F(1^\lambda), K_T \leftarrow \mathsf{Puncture}_F(K,T)] = 1.$$

- *For every* PPT *adversary* $(\mathcal{A}_1, \mathcal{A}_2)$ *such that* $\mathcal{A}_1(1^\lambda)$ *outputs a set* $S \subset \{0,1\}^{n(\lambda)}$ *and state* σ*, consider an experiment where* $K \leftarrow \mathsf{Key}_F(1^\lambda)$ *and* $K_S \leftarrow \mathsf{Punture}_F(K,S)$*; then we have*

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

where $\mathsf{Eval}_F(K,S)$ *denotes the concatenation of* $\mathsf{Eval}_F(K,x_1), ..., \mathsf{Eval}_F(K,x_k)$*, where* $S = \{x_1, ..., x_k\}$ *is the enumeration of the elements of* S *in lexicographic order , and* U_ℓ *denotes the uniform distribution over* ℓ *bits.*

For ease of notation, we will use $F(K,x)$ to represent $\mathsf{Eval}_F(K,x)$ and $K(T)$ to represent $\mathsf{Puncture}_F(K,T)$. As recently shown in [BW13, BGI14, KPTZ13], puncturable PRFs can be built from one-way functions using the classical GGM tree-based construction of PRFs [GGM84]. Specifically:

Theorem 2.4 ([BW13])**.** *If one-way functions exist, then for all efficiently computable functions* $n(\lambda), m(\lambda)$*, there exists a puncturable PRF family that maps* $n(\lambda)$ *bits to* $m(\lambda)$ *bits.*

Finite collection of one-way permutations. Our constructions require the existence of a concretely secure finite collection of one-way permutations, as defined below.

Definition 2.5. *Let* $\mathcal{L}$ *be a finite set of positive integers. We say a finite collection of permutations* $\Pi_\mathcal{L} = \{\pi_\ell : \{0,1\}^\ell \to \{0,1\}^\ell | \ell \in \mathcal{L}\}$ *is* (t,ϵ)-one-way secure*, if for a fixed* $\ell \in \mathcal{L}$*,* π_ℓ *can be computed in time polynomial in* ℓ*, and for any* $t(\ell)$*-time adversary* $\mathcal{A}$*, it holds that*

$$\mathbf{Pr}[\mathcal{A}(\pi_\ell(x)) = x | x \xleftarrow{\$} \{0,1\}^\ell] < \epsilon(\ell).$$

We note that we only require the existence of such collection, as it is only used in the proofs (cf. Section 4). We present a DL-based instantiation of Definition 2.5 for a specific set $\mathcal{L}$ in Appendix A.

3 Adjustable Signatures

In this section, we present our new notion—syntax and security model—of adjustable signature schemes. Roughly speaking, in such a scheme, the actual length of the signature can be specified, sometimes "on the fly," according to a parameter ℓ. Further, in an adjustable signature scheme $\Sigma = (\mathsf{Setup}, \mathsf{Sign}, \mathsf{Verify})$, the adjustment can happen in one of three phases, corresponding to the three algorithms. If the adjustment is specified in the setup phase, the setup algorithm would take the length parameter as input, and the signature algoritm would generate ℓ-bits long signatures. If the adjustment is specified in the signing phase, the length of each signature is given as input to the signing algorithm and can vary from one invocation to the next.

Finally, when the adjustment is specified in the verification phase, the signing algorithm outputs a standard signature but the verification algorithm will only process the signature's ℓ least significant bits.[1] Next, we discuss each variant in more detail in turn.

Setup-adjustable signatures. In the first variant of adjustable signature schemes, the length parameter ℓ is an input to the Setup algorithm, thus fixing the length of all signatures that are to be produced and verified. Sign and Verify remain as in a standard signature definition. The ability to decide the signature length during the setup allows an application to weigh the security/efficiency trade-offs before generating the keys. A drawback of this variant is that one needs to re-run the setup algorithm in order to increase/decrease the signature length, as mandated for example by the current security needs. Nevertheless, setup-adjustable signatures are the simplest notion of adjustability we consider, and the one with the most efficient instantiations.

$\mathsf{Setup}(1^\lambda, \ell)$: On input the security parameter λ and the length parameter ℓ, the Setup algorithm outputs a signing key sk and a verification key vk.

$\mathsf{Sign}(\mathsf{sk}, m)$: On input the signing key sk and a message m, the signing algorithm outputs a signature σ where $|\sigma| = \ell$.

$\mathsf{Verify}(\mathsf{vk}, \sigma, m)$: On input the verification key vk and a signature/message pair (σ, m), the verification algorithm outputs 1 (accept) or 0 (reject).

Signing-adjustable signatures. Here, how long a signature is going to be is specified at signing, and this can change from one invocation to the next.

$\mathsf{Setup}(1^\lambda)$: On input the security parameter λ, the setup algorithm outputs a signing key sk and a verification key vk.

$\mathsf{Sign}(\mathsf{sk}, m, \ell)$: On input a secret key sk, a message m and the length parameter ℓ, the signing algorithm outputs a signature σ, where $|\sigma| = \ell$. We assume the length parameter ℓ is included in signature.

$\mathsf{Verify}(\mathsf{vk}, \sigma, m)$: On input the verification key vk, a signature/message pair (σ, m), the verification algorithm outputs 1 (accept) or 0 (reject).

Verification-adjustable signatures. In the third variant of adjustable signature schemes, the verification algorithm takes the length parameter as input.

$\mathsf{Setup}(1^\lambda)$: On input the security parameter λ, the setup algorithm outputs a signing key sk and a verification key vk.

$\mathsf{Sign}(\mathsf{sk}, m)$: On input the signing key sk and a message m, the signing algorithm outputs a signature σ.

$\mathsf{Verify}(\mathsf{vk}, \ell, \sigma, m)$: On input the verification key vk, a length parameter ℓ, the signature σ and the message m, the verification algorithm only reads an adjusted ℓ-bit function of σ (i.e., $f(\sigma) = \sigma|_\ell$, and outputs 1 (accept) or 0 (reject).

Definition 3.1. *Let $\mathcal{L}$ be a finite set of positive integers. We call a signature scheme $\mathcal{L}$-adjustable, if the length parameter can be chosen to be any $\ell \in \mathcal{L}$.*

[1] One can envision more general formulations where the verification algorithm can take the ℓ-bit output of an arbitrary function of the signature; for simplicity, here we just focus on the truncation function.

Correctness. The correctness of an $\mathcal{L}$-adjustable signature scheme Σ is defined similarly to standard signature schemes. A setup-adjustable signature scheme is correct if

$$\forall \lambda, m, \ell \in \mathcal{L}, \Pr[\mathsf{Verify}(\mathsf{vk}, \sigma, m) = 1 | (\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\lambda, \ell), \sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)] = 1.$$

Similarly, a signing-adjustable signature scheme is correct if

$$\forall \lambda, m, \ell \in \mathcal{L}, \Pr[\mathsf{Verify}(\mathsf{vk}, \sigma, m) = 1 | (\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\lambda), \sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m, \ell)] = 1.$$

Finally, a verification-adjustable signature scheme is correct if

$$\forall \lambda, m, \ell \in \mathcal{L}, \Pr[\mathsf{Verify}(\mathsf{vk}, \ell, \sigma, m) = 1 | (\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\lambda), \sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)] = 1.$$

Security. We now define the notion of *strong existential unforgeability under chosen-message attacks* for a $\mathcal{L}$-adjustable signature scheme Σ. For every $\ell \in \mathcal{L}$, we use the experiments $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{setup\text{-}adj}}(\ell, 1^\lambda)$, $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{sign\text{-}adj}}(\ell, 1^\lambda)$ and $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{vfy\text{-}adj}}(\ell, 1^\lambda)$ to describe the interaction between a challenger and an adversary $\mathcal{A}$ in the three scenarios, respectively. (See Figure 1.) In the experiments, $\mathcal{O}_1(\mathsf{sk}, \cdot)$ returns a signature $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$; $\mathcal{O}_2(\mathsf{sk}, \cdot, \cdot)$ returns a signature $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m, \ell)$ if $\ell \in \mathcal{L}$ and $\perp$ otherwise; $\mathcal{O}_3(\mathsf{sk}, \cdot)$ returns a signature $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$.

1. $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(\ell, 1^\lambda)$
2. $(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}_1(\mathsf{sk}, \cdot)}(\mathsf{vk}, \ell)$
3. output $\mathsf{Verify}(\mathsf{vk}, \sigma^*, m^*)$

(a) $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{setup\text{-}adj}}(\ell, 1^\lambda)$

1. $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\lambda)$
2. $(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}_2(\mathsf{sk}, \cdot, \cdot)}(\mathsf{vk}, \ell)$
3. **if** $|\sigma^*| \neq \ell$ output 0;
4. **else** output $\mathsf{Verify}(\mathsf{vk}, \sigma^*, m^*)$

(b) $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{sign\text{-}adj}}(\ell, 1^\lambda)$

1. $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\lambda)$
2. $(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}_3(\mathsf{sk}, \cdot)}(\mathsf{vk}, \ell)$
3. output $\mathsf{Verify}(\mathsf{vk}, \ell, \sigma^*, m^*)$

(c) $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{vfy\text{-}adj}}(1^\lambda)$

Figure 1: *Security experiments for adjustable signature schemes*

We say the adversary $\mathcal{A}$ wins in experiment $\mathsf{Expt}^{\mathsf{setup\text{-}adj}}$ (resp. $\mathsf{Expt}^{\mathsf{sign\text{-}adj}}, \mathsf{Expt}^{\mathsf{vfy\text{-}adj}}$) if the output of $\mathsf{Expt}^{\mathsf{setup\text{-}adj}}$ (resp. $\mathsf{Expt}^{\mathsf{sign\text{-}adj}}, \mathsf{Expt}^{\mathsf{vfy\text{-}adj}}$) is 1.

Definition 3.2. *We say a $\mathcal{L}$-setup-adjustable (respectively, $\mathcal{L}$-signing-adjustable, $\mathcal{L}$-verification-adjustable) signature scheme Σ is* $(t, q_{\mathsf{sign}}, \epsilon)$-strongly existentially unforgeable against adaptive message queries*, if for all $\ell \in \mathcal{L}$, and any $t(\ell)$-time adversary $\mathcal{A}$ that makes at most $q_{\mathsf{sign}}(\ell)$ queries to the signing oracle, the probability of $\mathcal{A}$ wining in* $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{setup\text{-}adj}}(\ell, 1^\lambda)$ *(*$\mathsf{Expt}_{\mathcal{A}}^{\mathsf{sign\text{-}adj}}(\ell, 1^\lambda)$*,* $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{vfy\text{-}adj}}(\ell, 1^\lambda)$*, resp.) is less than $\epsilon(\ell)$.*

Remark 3.3. Since we consider the strong unforgeability notion for signature schemes, we require that if the challenge message m^* is the same as one of the queried messages, then the output signature σ^* in the forgery tuple (m^*, σ^*) must be different from the signature obtained from the query.

We finally remark that in our analyses there will be asymptotic terms of the form $\mathsf{negl}(\lambda)$ (from the security of the obfuscation and the puncturable PRF family, for example) and concrete terms (from the concrete security of the one-way permutation). Throughout the paper, we will assume that λ is large enough to render the asymptotic terms insignificant compared to the concrete terms.

4 $i\mathcal{O}$-based Adjustable Signatures

In this section, we present our constructions for adjustable signatures in all three scenarios based on indistinguishability obfuscation. In the following section, we also show how to instantiate a setup-adjustable signature scheme using BLS signatures [BLS01].

Achieving our notions of adjustable signatures appears to be a challenging problem. For example, it is not obvious how to adjust group sizes/security level of a DL-based signature scheme at the signing or verification stages without making any changes to the keys which are decided at setup. Nevertheless, we propose the first feasibility results for signing/verification-adjustable signatures based on $i\mathcal{O}$, and leave open the question of a concrete instantiation or constructions based on weaker general assumptions.

Our starting point is the short signature scheme by Ramchen and Waters [RW14] which we now briefly review. There are two main components in that construction. The first signature "piece" is a one-time-like signature scheme, as follows: one generates a tag $\boldsymbol{t}$ of λ bits and lets $s_1 = \oplus_{i=1}^{\ell} F_1(K_1, \boldsymbol{t}||i||M(i))$, where $F_1(K_1, \cdot)$ is a puncturable PRF with appropriate input length. The verification key is an obfuscated circuit that on input $(M, (\boldsymbol{t}, s_1))$ 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 $\boldsymbol{t}$, cannot construct a signature on $M^* \neq M$, that uses the same tag $\boldsymbol{t}$.

The second piece is the ability to sign the tag t according to the prefix-guessing technique [HW09]. To sign a tag $\boldsymbol{t}$, a puncturable PRF $F_{2,i}(K_2, i, \cdot)$ is evaluated on every prefix. Here $F_{2,i}$, $i = 1, \ldots, \ell$, takes inputs of i bits. The signature piece is thus $s_2 = \oplus_{i=1}^{\lambda} F_{2,i}(K_2, i, \boldsymbol{t}|_i)$, where the length-$i$ prefix of $\boldsymbol{t}$ is denoted $\boldsymbol{t}|_i$. A verification key is an obfuscated circuit that on input $(\boldsymbol{t}, s_2)$, checks that s_2 is of the above form. The security property is that an adversary, on seeing a signature that uses tags $\boldsymbol{t}_1, \ldots, \boldsymbol{t}_q$, cannot produce a signature with a tag $\boldsymbol{t}^* \neq \boldsymbol{t}_i$ for some i. The complete scheme merges these two ideas to generate a concise signature. 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 $(M, (\boldsymbol{t}, s))$ computes $s_1 = \oplus_{i=1}^{\ell} F_1(K_1, \boldsymbol{t}||i||M(i))$ and $s_2 = \oplus_{i=1}^{\lambda} F_{2,i}(K_2, i, \boldsymbol{t}|_i)$ and checks that $s = s_1 \oplus s_2$. We now turn to our constructions.

4.1 An $i\mathcal{O}$-based setup-adjustable signature scheme

We let $F_i(K_i, \cdot)$ be a puncturable PRF mapping i-bit inputs to λ-bit outputs, for $i \in [\lambda]$. We assume our message space is $\mathcal{M} = \{0,1\}^{\lambda}$. (Longer messages can be hashed into this space.) Then the $i\mathcal{O}$-based setup-adjustable signature scheme $(\mathsf{Setup}, \mathsf{Sign}, \mathsf{Verify})$ can be described as follows:

- $\mathsf{Setup}(1^{\lambda}, \ell)$: On input the security parameter λ and a length parameter ℓ, the setup algorithm first randomly chooses puncturable PRF keys $\{K_i\}_{i=1}^{\lambda}$. Then it selects 2λ random bit-strings $k_{i,b} \in \{0,1\}^{3\lambda}$, for $i \in [\lambda], b \in \{0,1\}$. The signing key is $\mathsf{sk} = \{K_i\}_{i=1}^{\lambda}$, and the verification key vk is an obfuscation of the program described in Figure 2, plus random bit-strings $\{k_{i,b}\}_{i\in[\lambda],b\in\{0,1\}}$. Note that the $k_{i,b}$'s are outside of the obfuscated program and publicly known.
- $\mathsf{Sign}(\mathsf{sk}, \boldsymbol{m})$: On input the signing key $\mathsf{sk} = \{K_i\}_{i=1}^{\lambda}$ and a message $\boldsymbol{m} \in \{0,1\}^{\lambda}$, the signing algorithm first computes a temporary tag $\boldsymbol{t} = \oplus_{i=1}^{\lambda} k_{i,\boldsymbol{m}[i]}$, and then computes

$$\boldsymbol{s} = \oplus_{i=1}^{\lambda} F_i(K_i, \boldsymbol{t}|_i).$$

  It outputs signature $\sigma = \boldsymbol{s}|_{\ell}$.
- $\mathsf{Verify}(\mathsf{vk}, \sigma, \boldsymbol{m})$: The verification algorithm first computes the temporary tag $\boldsymbol{t} = \oplus_{i=1}^{\lambda} k_{i,\boldsymbol{m}[i]}$ for message $\boldsymbol{m}$ and then runs the obfuscated program in Figure 2 on the message/signature pair $(\sigma, \boldsymbol{m})$ and temporary tag $\boldsymbol{t}$, and outputs the result.

Hardcoded: PRF keys $\{K_i\}_{i=1}^{\lambda}$ and a length parameter $\ell \in \mathcal{L}$.
Input: A message/signature pair (m, σ) and a temporary tag $\boldsymbol{t}$.

1. Compute $\boldsymbol{s} = \oplus_{i=1}^{\lambda} F_i(K_i, \boldsymbol{t}|_i)$.
2. Output 1 if $\boldsymbol{s}|_\ell = \sigma$, otherwise output 0.

Figure 2: *Program Verify Signature for the $\mathcal{L}$-setup-adjustable signature scheme*

Theorem 4.1. *For $\ell \in \mathcal{L}$, let $\pi_\ell : \{0,1\}^\ell \to \{0,1\}^\ell$ be a (t, ϵ)-one-way permutation (Definition 2.5)[2]. If the obfuscation scheme used in Figure 2 is indistinguishably secure and $\{F_i\}_{i=1}^{\lambda}$ are secure puncturable PRFs, then the above $\mathcal{L}$-setup-adjustable signature scheme is $(t, q_{\mathsf{sign}}, \epsilon)$-strongly existentially unforgeable (Definition 3.2).*

Proof. The proof consists of a sequence of hybrid experiments, where the first hybrid corresponds to the experiment $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{setup\text{-}adj}}(\ell, 1^\lambda)$. The hybrids can be described as follows:

- Hybrid H_0: In the first hybrid, the following experiment is played between challenger and adversary $\mathcal{A}$:
  1. $\{K_i\}_{i=1}^{\lambda}$ are chosen as keys for the puncturable PRFs $\{F_i\}_{i=1}^{\lambda}$, as well as 2λ random bit strings $\{k_{i,b}\} \in \{0,1\}^{3\lambda}$, for $i \in [\lambda], b \in \{0,1\}$.
  2. The verification key vk is given out as an obfuscation of the program described in Figure 2, plus the 2λ random bit strings $\{k_{i,b}\}_{i\in[\lambda],b\in\{0,1\}}$.
  3. The adversary makes at most q_{sign} queries to the signing oracle on messages $\boldsymbol{m}_j$, and obtains $\boldsymbol{s}|_\ell$ as an answer, where the signing oracle first computes a temporary tag $\boldsymbol{t} = \oplus_{i=1}^{\lambda} k_{i,\boldsymbol{m}[i]}$, and then computes $\boldsymbol{s} = \oplus_{i=1}^{\lambda} F_i(K_i, \boldsymbol{t}|_i)$
  4. The adversary outputs a forgery σ^* for challenge message $\boldsymbol{m}^*$, and wins if $\mathsf{Verify}(\mathsf{vk}, m^*, \sigma^*) = 1$ holds.
- Hybrid H_1: In this hybrid, the challenger requires that the message $\boldsymbol{m}^*$ in the forgery pair $(\boldsymbol{m}^*, \sigma^*)$ has not been queried before. The rest of the hybrid remains unchanged.
- Hybrid H_2: In this hybrid, the challenger changes the wining condition. First, the challenger selects random indices (i', j') from $[\lambda] \times [q_{\mathsf{sign}}]$, and sets $\boldsymbol{p} = \boldsymbol{t}_{j'}|_{i'} \oplus \boldsymbol{e}_{i'}$, where vector $\boldsymbol{e}_{i'}$ is the i'-th unit vector of length λ. For the forgery pair $(\boldsymbol{m}^*, \sigma)$ output by adversary $\mathcal{A}$, the challenger computes the temporary tag $\boldsymbol{t}^* = \oplus_{i=1}^{\lambda} k_{i,\boldsymbol{m}^*[i]}$ and changes the wining condition by enforcing the additional check $\boldsymbol{t}^*|_{i'} = \boldsymbol{p}$. The rest of the hybrid remains unchanged.
- Hybrid H_3: Same as hybrid H_1, except that the challenger first sets $\boldsymbol{z}^* = F_{i'}(K_{i'}, \boldsymbol{p})$ and then punctures the PRF $F_{i'}$ on $\boldsymbol{p} = \boldsymbol{t}_{j'}|_{i'} \oplus \boldsymbol{e}_{i'}$, i.e., $K_{i',\boldsymbol{p}} = \mathsf{Puncture}_{F_{i'}}(K_{i'}, \boldsymbol{p})$. Then challenger sets vk to be the obfuscation of the program described in Figure 3, plus $2n$ random bit strings $\{k_{i,b}\}_{i\in[n],b\in\{0,1\}}$. The rest of the hybrid remains unchanged.

[2] π_ℓ will be used in the proof.

Hardcoded: PRF keys $\{K_i\}_{i\neq i'}$), punctured key $K_{i',\boldsymbol{p}}$, a length parameter $\ell \in \mathcal{L}$. and strings $\boldsymbol{p}, \boldsymbol{z}^*$.
Input: A message/signature pair $(\boldsymbol{m}, \sigma)$ and temporary tag $\boldsymbol{t}$.

1. If $\boldsymbol{t}_{i'} = \boldsymbol{p}$, then if
$$\sigma \oplus_{i\neq\hat{i}} F_i(K_i, \boldsymbol{t}|_i)|_\ell = z^*|_\ell$$
output 1, otherwise output 0.
2. Else if
$$\sigma = \oplus_{i\neq i'} F_i(K_i, \boldsymbol{t}|_i)|_\ell \oplus F_{i',\boldsymbol{p}}(K_{i',\boldsymbol{p}}, \boldsymbol{t}|_{i'})|_\ell$$
output 1, otherwise output 0.

Figure 3: *Program Verify Signature* for the $\mathcal{L}$-setup-adjustable signature scheme*

- Hybrid H_4: The same as hybrid H_3 except that we set $\boldsymbol{z}^* = \boldsymbol{u}|_\ell$, where $\boldsymbol{u}$ is chosen uniformly at random from the range of the puncturable PRF $F_{i'}$.
- Hybrid H_5: The same as hybrid H_4 except that we set $\boldsymbol{z}^* = \pi_\ell(\boldsymbol{\alpha})$, where $\pi_\ell(\cdot)$ is a (t, ϵ)-one-way permutation[3] and $\boldsymbol{\alpha}$ is a random ℓ-bit string.

Claim 4.2. *Suppose the winning probability of adversary $\mathcal{A}$ in $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{setup\text{-}adj}}(\ell, 1^\lambda)$ is ϵ, then the wining probability of adversary $\mathcal{A}$ in hybrid H_1 is also ϵ.*

Proof. We show that the requirement enforced in this hybrid is a benign one, which dose not affect the advantage of adversary in the forgery experiment. For a message $\boldsymbol{m}$, the signature σ for $\boldsymbol{m}$ is deterministically generated by first computing the tag $\boldsymbol{t} = \oplus_{i=1}^{\lambda} k_{i,\boldsymbol{m}[i]}$, and then evaluating the PRFs based on the tag $\boldsymbol{t}$, i.e. $\boldsymbol{s} = \oplus_{i=1}^{\lambda} F_i(K_i, \boldsymbol{t}|_i)$. Therefore, for the forgery pair $(\boldsymbol{m}^*, \sigma)$, where $\boldsymbol{m}^*$ is queried before, in order to pass the verification algorithm, the forged signature σ must be the same as the signature obtained as the response for query $\boldsymbol{m}^*$, which means the advantage of adversary in hybrids H_0 and H_1 are the same. □

Claim 4.3. *Suppose the winning probability of adversary $\mathcal{A}$ in hybrid H_1 is ϵ, then the wining probability of adversary $\mathcal{A}$ in hybrid H_2 is bounded by $\frac{\epsilon(2^\lambda-1)}{2^\lambda \lambda q_{\mathsf{sign}}}$.*

Proof. First, the forgery message $\boldsymbol{m}^*$ is different from all queried messages $\boldsymbol{m}_i$ for $i \in [q_{\mathsf{sign}}]$ and the temporary tag is generated as $\boldsymbol{t} = \oplus_{i=1}^{n} k_{i,\boldsymbol{m}[i]}$, where $k_{i,b}$ are random strings over $\{0,1\}^{3\lambda}$, thus the probability that two messages have the same temporary tag is $\frac{1}{2^{3\lambda}}$. By applying a union bound on all possible pairs of messages $(\boldsymbol{m}_i, \boldsymbol{m}_j)$, we obtain that the probability that any two messages have the same temporary tag is $\frac{2^{2\lambda}-2^\lambda}{2^{3\lambda}}$, therefore we have
$$\mathbf{Pr}[\exists i \in [q_{\mathsf{sign}}] : \boldsymbol{t}^* = \boldsymbol{t}_i] = \frac{2^{2\lambda} - 2^\lambda}{2^{3\lambda}}.$$
Also, since $\boldsymbol{m}^* \neq \boldsymbol{m}_i$ for $i \in [q_{\mathsf{sign}}]$, there exists a shortest common prefix of $\boldsymbol{t}^*$ with $\boldsymbol{t}_i$, $i \in [q_{\mathsf{sign}}]$, and the length of the prefix is at most $\lambda - 1$. In particular, there exists some string $\boldsymbol{t}_j$ and a prefix of length i, such that $\boldsymbol{t}^*|_i = \boldsymbol{t}_j \oplus \boldsymbol{e}_i$. Since the challenger selects the index (i', j') uniformly at random from $[\lambda] \times [q_{\mathsf{sign}}]$, the event $(i', j') = (i, j)$ happens with probability $1/(\lambda q_{\mathsf{sign}})$. □

[3] Alternatively, one can replace the one-way permutation with a puncturable PRF F with truncation to ℓ bits: we first compute $F(\boldsymbol{\alpha})|_\ell$ and then puncture F on input $\boldsymbol{\alpha}$. We discuss this variant in detail in the full version of the paper.

Claim 4.4. *If the obfuscation scheme used in Figure 3 is indistinguishably secure, then the probability of adversary $\mathcal{A}$ distinguishing between hybrids H_2 and H_3 is negligible, i.e.,*

$$|\mathbf{Pr}[\mathcal{A}(\mathsf{H}_2) = 1] - \mathbf{Pr}[\mathcal{A}(\mathsf{H}_3) = 1]| \leq \mathsf{negl}(\lambda).$$

Proof. We argue the computational indistinguishability of these two hybrids based on the security of indistinguishability obfuscation, by demonstrating the functional equivalence of the programs described in Figure 2 and Figure 3. Consider a message/signature pair $(\boldsymbol{m}, \sigma)$ and a temporary tag $\boldsymbol{t}$ for message $\boldsymbol{m}$, which is an input to the verification program in Figure 3. The program first checks the temporary $\boldsymbol{t}$, if it holds that $\boldsymbol{t}|_{i'} \neq \boldsymbol{p}$, then by the functionality-preserving property of puncturable PRF $F_{i'}$, we have $F_{i'}(K_{i'}, \boldsymbol{t}|_{i'}) = F_{i',\boldsymbol{p}}(K_{i',\boldsymbol{p}}, \boldsymbol{t}|_{i'})$. Therefore, by the generation of signature $\sigma = \boldsymbol{s}|_\ell$, we have

$$\sigma = \oplus_{i \neq i'} F_i(K_i, \boldsymbol{t}|_i)|_\ell \oplus F_{i',\boldsymbol{p}}(K_{i',\boldsymbol{p}}, \boldsymbol{t}|_{i'})|_\ell = \oplus_{i \neq i'} F_i(K_i, \boldsymbol{t}|_i)|_\ell \oplus F_{i'}(K_{i'}, \boldsymbol{t}|_{i'})|_\ell.$$

Else, if $\boldsymbol{t}|_{i'} = \boldsymbol{p}$, we have

$$\sigma \oplus_{i \neq \hat{i}} F_i(K_i, \boldsymbol{t}|_i)|_\ell = z^*|_\ell.$$

Thus, the output of program Verify Signature* in Figure 3 is the same as program Verify Signature in Figure 2. Therefore, if there exists an advantage difference, we can create an algorithm $\mathcal{B}$ that breaks the indistinguishability of obfuscation, by submitting Verify Signature and Verify Signature* to the obfuscation challenger. □

Claim 4.5. *Based on the security of puncturable PRFs, the probability of adversary $\mathcal{A}$ distinguishing between hybrids H_3 and H_4 is negligible, i.e.,*

$$|\mathbf{Pr}[\mathcal{A}(\mathsf{H}_3) = 1] - \mathbf{Pr}[\mathcal{A}(\mathsf{H}_4) = 1]| \leq \mathsf{negl}(\lambda).$$

Proof. We now show that for any polynomial-time adversary, the advantage in forging a signature must be negligibly close in hybrids H_3 and H_4. Otherwise, we can construct a reduction algorithm $\mathcal{B}$ that breaks the selective security of the puncturable PRF at the punctured points, as follows. $\mathcal{B}$ first selects random indices (i', j') from $[\lambda] \times [q_{\mathsf{sign}}]$, sets $\boldsymbol{p} = \boldsymbol{t}_{j'}|_{i'} \oplus \boldsymbol{e}_{i'}$ and submits $\boldsymbol{p}$ to the PRF challenger and receives punctured keys $K_{i',\boldsymbol{p}} \leftarrow \mathsf{Puncture}_{F_{i'}}(K_{i'}, \boldsymbol{p})$ and PRF challenge $\boldsymbol{z}$. $\mathcal{B}$ then runs experiment H_2 except that it sets $\boldsymbol{z}^* = \boldsymbol{z}_\ell$. If $\boldsymbol{z}$ is the output of the PRF $F_{i'}$ at point $\boldsymbol{p}$, then we are in hybrid H_3; if $\boldsymbol{z}$ is chosen randomly, then we are in hybrid H_4. $\mathcal{B}$ will output 1 if the adversary wins. Thus, an adversary with different advantages in the two hybrids leads to an algorithm $\mathcal{B}$ that breaks the security of the puncturable PRF. □

Claim 4.6. *Based on the security of indistinguishability obfuscation, the probability of adversary $\mathcal{A}$ distinguishing between hybrids H_4 and H_5 is negligible, i.e.,*

$$|\mathbf{Pr}[\mathcal{A}(\mathsf{H}_4) = 1] - \mathbf{Pr}[\mathcal{A}(\mathsf{H}_5) = 1]| \leq \mathsf{negl}(\lambda).$$

Proof. We argue the computational indistinguishability of these two hybrids based on the security of indistinguishability obfuscation. We observe that the input/output behavior of these two programs are identical. The only difference is that in the first verification program, we set $\boldsymbol{z}^* = \boldsymbol{u}|_\ell$, where $\boldsymbol{u}$ is chosen uniformly from the range of a puncturable PRF $F_{i'}$, and in the current verification program in hybrid H_5, we set $\boldsymbol{z}^* = \pi_\ell(\boldsymbol{u}|_\ell)$, where $\pi_\ell(\cdot)$ is a (t, ϵ)-one-way permutation. Therefore, if there exists a difference in the advantages, we can create an algorithm $\mathcal{B}$ that breaks the indistinguishability of the obfuscation by submitting both programs to the challenger. □

Claim 4.7. *Given a (t, ϵ)-one-way permutation π_ℓ, the probability of a t-time adversary $\mathcal{A}$ winning in hybrid H_5 is less than ϵ.*

Proof. If there exists a successful attacker in hybrid H_5, we can use it to break the (t, ϵ)-security of the one-way permutation π_ℓ. The algorithm $\mathcal{B}$ works as follows: it first receives $\boldsymbol{z}' = \pi_\ell(\boldsymbol{a})$ as an input, where $\boldsymbol{a}$ is a random bit string. Then it randomly picks indices (i', j') from $[\lambda] \times [q_{\mathsf{sign}}]$, sets $\boldsymbol{p} = \boldsymbol{t}_{j'}|_{i'} \oplus \boldsymbol{e}_{i'}$. $\mathcal{B}$ computes the punctured key $K_{i',\boldsymbol{p}}$ as before, and set $\boldsymbol{z}^* = \boldsymbol{z}'$. Since $\boldsymbol{z}'$ is identically distributed to $\boldsymbol{z}$ in hybrid H_5, the view of an attacker $\mathcal{A}$ is identical to the view in hybrid H_5. If an attacker successfully outputs a forgery $(\boldsymbol{m}^*, \sigma^*)$, where $\sigma^* = \boldsymbol{s}^*|_\ell$, then by definition $\mathcal{B}$ can compute the random string $\boldsymbol{z}'$ as

$$\boldsymbol{z}' = \boldsymbol{s}^*|_\ell \oplus_{i \neq i'} F_i(K_i, \boldsymbol{t}^*|_i)|_\ell.$$

Therefore, if the one-way permutation is (t, ϵ)-secure, then no t-time attacker can forge with probability larger then ϵ. □

Combining the description of hybrids $\mathsf{H}_0, \mathsf{H}_1, \mathsf{H}_2, \mathsf{H}_3, \mathsf{H}_4, \mathsf{H}_5$ and the above claims, we conclude that the advantage of a t-time adversary in the existential unforgeability experiment is less than ϵ. □

4.2 An $i\mathcal{O}$-based signing-adjustable signature scheme

Next, we present our construction of the signing-adjustable signature scheme based on $i\mathcal{O}$. We still let $F_i(K_i, \cdot)$ be a puncturable PRF mapping i-bit inputs to λ-bit outputs, for $i \in [\lambda]$. We assume our message space $\mathcal{M} = \{0,1\}^\lambda$. The description of algorithms $\Sigma = (\mathsf{Setup}, \mathsf{Sign}, \mathsf{Verify})$ for this variant are given below. In a nutshell, the length of signatures are now decided on-the-fly at signing time, and so the length parameter is taken as an input by the Sign algorithm; in addition, the obfuscated program (verification key) also takes it as an input (it can actually be derived from the signature itself), in contrast to the setup-adjustable case, where it was hard-coded.

$\mathsf{Setup}(1^\lambda)$: On input the security parameter λ, the setup algorithm first randomly chooses puncturable PRF keys $\{K_i\}_{i=1}^\lambda$. Then select $2n$ random bit strings $k_{i,b} \in \{0,1\}^{3\lambda}$, for $i \in [\lambda], b \in \{0,1\}$. The signing key is $\mathsf{sk} = \{K_i\}_{i=1}^\lambda$, and the verification key vk is an obfuscation of the program described in Figure 4, plus random bit strings $\{k_{i,b}\}_{i \in [\lambda], b \in \{0,1\}}$.

$\mathsf{Sign}(\mathsf{sk}, \boldsymbol{m}, \ell)$: On input the signing key sk, a message $\boldsymbol{m}$ and a length parameter $\ell \in \mathcal{L}$, the signing algorithm first computes the temporary tag $\boldsymbol{t} = \oplus_{i=1}^\lambda k_{i,\boldsymbol{m}[i]}$, and then computes

$$\boldsymbol{s} = \oplus_{i=1}^\lambda F_i(K_i, \boldsymbol{t}|_i).$$

It outputs signature $\sigma = \boldsymbol{s}|_\ell$.

$\mathsf{Verify}(\mathsf{vk}, \boldsymbol{m}, \sigma)$: On input the verification key vk, a message $\boldsymbol{m}$, and a signature σ, the verification algorithm first computes the temporary tag $\boldsymbol{t} = \oplus_{i=1}^\lambda k_{i,\boldsymbol{m}[i]}$ for the message $\boldsymbol{m}$, and runs the obfuscated program in Figure 4 on input $(\boldsymbol{m}, \sigma, \ell, \boldsymbol{t})$, where $\ell = |\sigma|$.

Hardcoded: PRF key $\{K_i\}_{i=1}^\lambda$.
Input: A message $\boldsymbol{m}$, a signature σ, a length parameter $\ell \in \mathcal{L}$, and a temporary tag $\boldsymbol{t}$.

1. Output 1 if $\sigma = \oplus_{i=1}^\lambda F_i(K_i, \boldsymbol{t}|_i)|_\ell$, otherwise output 0.

Figure 4: *Program Verify Signature for the $\mathcal{L}$-signing-adjustable signature scheme*

Theorem 4.8. *For $\ell \in \mathcal{L}$, let $\pi_\ell : \{0,1\}^\ell \to \{0,1\}^\ell$ be a (t,ϵ)-one-way permutation (Definition 2.5). If the obfuscation scheme used in Figure 4 is indistinguishably secure and $\{F_i\}_{i=1}^\lambda$ are secure puncturable PRFs, then the above $\mathcal{L}$-signing-adjustable signature scheme is $(t, q_{\mathsf{sign}}, \epsilon)$-strongly existentially unforgeable (Definition 3.2).*

The proof is similar in spirit to the proof for setup-adjustable signatures above; a proof sketch is presented in Appendix B.1.

4.3 An $i\mathcal{O}$-based verification-adjustable signature scheme

We conclude this section by presenting our $i\mathcal{O}$-based construction of signatures whose length can be fixed at the verification stage. As such, algorithms Setup and Sign are oblivious to the length parameter, while Verify takes it as an additional argument.

$\mathsf{Setup}(1^\lambda)$: On input the security parameter λ, the setup algorithm first randomly chooses puncturable PRF keys $\{K_i\}_{i=1}^\lambda$. Then select $2n$ random bit strings $k_{i,b} \in \{0,1\}^{3\lambda}$, for $i \in [\lambda], b \in \{0,1\}$. The signing key is $\mathsf{sk} = \{K_i\}_{i=1}^\lambda$, and the verification key vk is an obfuscation of the program described in following Figure 5, plus random bit strings $\{k_{i,b}\}_{i\in[\lambda],b\in\{0,1\}}$.

$\mathsf{Sign}(\mathsf{sk}, \boldsymbol{m})$: On input the signing key sk and a message $\boldsymbol{m}$, the signing algorithm first computes the temporary tag $\boldsymbol{t} = \oplus_{i=1}^\lambda k_{i,\boldsymbol{m}[i]}$, and then outputs signature σ

$$\sigma = \boldsymbol{s} = \oplus_{i=1}^\lambda F_i(K_i, \boldsymbol{t}|_i)$$

$\mathsf{Verify}(\mathsf{vk}, \ell, \boldsymbol{m}, \sigma)$: On input the verification key vk, a length parameter ℓ, a message $\boldsymbol{m}$ and a signature σ, the verification algorithm first computes the temporary tag $\boldsymbol{t} = \oplus_{i=1}^\lambda k_{i,\boldsymbol{m}[i]}$ for the message $\boldsymbol{m}$, and runs the obfuscated program in Figure 5 on input $(\boldsymbol{m}, \sigma, \ell, \boldsymbol{t})$.

Hardcoded: PRF key $\{K_i\}_{i=1}^\lambda$.
Input: A message $\boldsymbol{m}$, an adjusted signature σ, a length parameter $\ell \in \mathcal{L}$ and a temporary tag $\boldsymbol{t}$.

1. Output 1 if $\sigma = \oplus_{i=1}^\lambda F_i(K_i, \boldsymbol{t}|_i)|_\ell$, otherwise output 0.

Figure 5: *Program Verify for the $\mathcal{L}$-verification-adjustable signature scheme*

Theorem 4.9. *For $\ell \in \mathcal{L}$, let $\pi_\ell : \{0,1\}^\ell \to \{0,1\}^\ell$ be a (t,ϵ)-one-way permutation (Definition 2.5). If the obfuscation scheme used in Figure 5 is indistinguishably secure and $\{F_i\}_{i=1}^\lambda$ are secure puncturable PRFs, then the above $\mathcal{L}$-verification-adjustable signature scheme is $(t, q_{\mathsf{sign}}, \epsilon)$-strongly existentially unforgeable (Definition 3.2).*

A proof sketch is presented in Appendix B.2.

In the above constructions the adjustment to the signatures was specified using a length parameter ℓ, and the output of the PRF was simply truncated to ℓ-bits. We note that if we replace truncation by any other "uniformness-preserving" function, the correctness and security proofs would still hold with minor modifications.

5 A Concrete Construction and Implementation

We instantiate setup-adjustable signature schemes using the BLS scheme [BLS01]. BLS provides the shortest existing signatures for the same level of security as other schemes, and hence provides a good starting point for achieving smaller size signatures, particularly for our OTP application. We review bilinear maps and BLS signatures in Appendix C.

The basic idea is to instantiate the BLS scheme using smaller size groups. This would yield shorter signatures, but also weaker security guarantees. We show how careful choices of the elliptic curve groups can provide a spectrum of trade-off between the signature lengths and security. We also report on our prototype implementation for different length parameter sizes.

Our setup-adjustable variant. In order to instantiate BLS using smaller group sizes, we first need to understand how it effects the security of the resulting signature scheme. In particular, we first map the hardness of CDH in the original group to hardness of DL in a related finite field. To do so, we need to recall the notion of *security multiplier* introduced in [BLS01]:

Definition 5.1 (Security multiplier)**.** *Let p be a prime number, k be a positive exponent, and E be an elliptic curve over $\mathbb{F}_{p^\ell}$ with m points. Let P on curve E be a point of prime order q where $q^2 \nmid m$. We say that the subgroup $\langle P\rangle$ has a security multiplier α, for some integer $\alpha > 0$, if the order of p^k in $\mathbb{F}_q^*$ is α. In other words,*

$$q \mid p^{k\alpha} - 1, q \nmid p^{kj} - 1, \forall j = 1, ..., \alpha - 1$$

As we will show below, for the CDH problem to be hard in the subgroup $\langle P\rangle$, the security multiplier for this subgroup cannot be too small. On the other hand, in order for the group operations in $\langle P\rangle$ to be efficient, the security multiplier can not be too large. Therefore, the challenge in constructing setup-adjustable signatures based on BLS is to find curves with security multipliers that are large enough for security, but small enough for efficiency. We adapt the analysis of [BLS01] to find the right curves for our setup-adjustable signature.

Let $\langle P\rangle$ be a subgroup of $E/\mathbb{F}_{p^k}$ of order q with the security multiplier α. There are two standard ways of solving the discrete-log problem in $\langle P\rangle$: the MOV reduction of [MOV$^+$93], and the generic reduction algorithms, such as Baby-Step-Giant-Step and Pollard's Rho method [MVOV96]. The running time of the generic methods is proportional to $\sqrt{q}$. Thus, we must ensure that q is sufficiently large to resist these attacks. The MOV reduction, on the other hand, maps the discrete log problem in $\langle P\rangle$ to a discrete log problem in the extension of $\mathbb{F}_{p^k}$, say $\mathbb{F}_{p^{ki}}$ for some i, and requires that the image of $\langle P\rangle$ under this mapping be a subgroup of $\mathbb{F}_{p^{k}i}$ of order q. This requires that $q|(p^{ki} - 1)$, which implies $i \geq \alpha$ by the definition of security multiplier. Hence, the MOV reduction can at best reduce the discrete-log problem in $\langle P\rangle$ to a discrete-log problem in a subgroup of $\mathbb{F}_{p^{\ell\alpha}}$. Therefore, to ensure that the discrete-log problem in $\langle P\rangle$ is hard in our setting, we need to choose curves with large security multipliers.

Supersingular curves usually have small representations, which in our setting will result in small signature sizes. As we will show below, the supersingular curves given by $E_{3,2} : y^2 = x^3 + 2x + 2$,and $E_{3,1} : y^2 = x^3 + 2x + 1$ have the security multiplier 6, which is sufficient for generating short signatures. The MOV reduction described above reduces the DL for these curves to the discrete-log problem in $E_{3,2}/\mathbb{F}_{3^{6k}}$, and ($E_{3,1}/\mathbb{F}_{3^{6k}}$). This means that we can use relatively small values of k to obtain short signatures, but the security depends on a discrete-log problem in a large finite field. We use the following two lemmas to describe the behavior of these two curves.

Lemma 5.2 ([Kob98])**.** *The curve $E_{3,2} : y^2 = x^3 + 2x + 2$ defined over $\mathbb{F}_{3^k}$ satisfies:*

$$\#(E_{3,2}/\mathbb{F}_{3^k}) = \begin{cases} 3^k + 1 + \sqrt{3 \cdot 3^k} & \text{when } k = \pm 1 \bmod 12 \\ 3^k + 1 - \sqrt{3 \cdot 3^k} & \text{when } k = \pm 5 \bmod 12 \end{cases}$$

The curve $E_{3,1} : y^2 = x^3 + 2x + 1$ *defined over* $\mathbb{F}_{3^k}$ *satisfies:*

$$\#(E_{3,1}/\mathbb{F}_{3^k}) = \begin{cases} 3^k + 1 - \sqrt{3 \cdot 3^k} & \text{when } k = \pm 1 \bmod 12 \\ 3^k + 1 + \sqrt{3 \cdot 3^k} & \text{when } k = \pm 5 \bmod 12 \end{cases}$$

Lemma 5.3 ([BLS01])**.** *Let curves* $E_{3,1}, E_{3,2}$ *be defined as above over* $\mathbb{F}_{3^k}$*, where* $k \bmod 12 = \pm 1$ *or* ± 5*, then we have* $\#(E/\mathbb{F}_{3^k})|3^{6k} - 1$.

Combining the above two lemmas, we obtain that for relevant values of k, the curves $E_{3,1}, E_{3,2}$ over finite field $\mathbb{F}_{3^k}$ will have security multiplier at most 6.

In addition to supersingular curves described above, we also use MNT curve proposed in [MNT01] for implementation. Let α be an even number and we choose a polynomial basis for $\mathbb{F}_{p^\alpha}$ over $\mathbb{F}_p$ by a irreducible polynomial with no odd terms, i.e.

$$F_{p^\alpha} = \mathbb{F}_p[x]/(f(x)), \qquad f(x) = x^\alpha + b_{\alpha-2}x^{\alpha-2} + \cdots b_2 x^2 + b_0$$

The curve is set to be $E_{\mathsf{MNT}} : y^2 = x^3 - 3x + B$, where $B \in \mathbb{F}_p$, and the group order divisible by a large prime N and such that N divides $p^\alpha - 1$ and N dose not divide $p^\beta - 1$ for $\beta < \alpha$.

Finally, for hashing onto the elliptic curves in the BLS signature scheme, we follow the same approach as [BLS01] to construct the hash function. Suppose we are given a hash function $H' : \{0,1\}^* \to \mathbb{F}_{p^k} \times \{0,1\}$, and let curves be denoted by $y^2 = f(x)$. For self-containness, we recall the MaptoGroup hash function $H : \{0,1\}^* \to G$ is as follows:

Input: $m \in \{0,1\}^*$

1. Set $i = 0$ and $(x, b) \leftarrow H'(i|m) \in \mathbb{F}_{p^k} \times \{0,1\}$.
2. If $f(x)$ is a quadratic residue in $\mathbb{F}_{p^k}$, then let $y_0, y_1 \in \mathbb{F}_{p^k}$ be the two square roots of $f(x)$. First set $\tilde{P}_m \in E/\mathbb{F}_{p^k}$ be the point (x, y_b), where $b \in \{0,1\}$. Then compute $P_m = (m/q)\tilde{P}_m$. Output P_m as the result if $P_m \in G^*$.
3. Otherwise, increment i, and continue from step 1.

Figure 6: Description of hash function MaptoGroup

For simplicity, we assume that the length parameter is chosen among one of those in Table 1. Then, the setup-adjustable signature scheme based on the BLS signature can be described as follows:

- $\mathsf{Setup}(1^\lambda, \ell)$: Given one of the values ℓ in Table 1, let $E/\mathbb{F}_{3^k}$ be the corresponding curve and q be the largest prime factor of the curve. We set $P \in E/\mathbb{F}_{3^k}$ be a point of order q. Select a random $x \in \mathbb{Z}_q^*$ and set $R = x \cdot P$. Set the verification key $\mathsf{vk} = (k, q, P, R)$ and secret key $\mathsf{sk} = x$.
- $\mathsf{Sign}(\mathsf{sk}, m)$: On input a message $m \in \{0,1\}^*$, first use hash function MaptoGroup to map m to a point $P_m \in \langle P \rangle$. Then set $S_m = x \cdot P_m$. The signature σ of message m is the x-coordinate of S_m.
- $\mathsf{Verify}(\mathsf{vk}, m, \sigma)$: On input a message/signature pair (m, σ) and verification key $\mathsf{vk} = (k, q, P, R)$, do the following:
  1. Plug in the signature σ into elliptic curve equation to get a valid y-coordinate y. Set point $S = (\sigma, y)$. If no valid y could be found, output $\perp$.
  2. If $e(P, S) = e(R, H(m))$, accept the signature. Otherwise, reject.

We present the security proof for our setup-adjustable signature scheme below. The proof structure follows the proof paradigm of Lemma 5 in [BLS01], thus we only provide a proof sketch below.

Theorem 5.4. *Suppose $E/\mathbb{F}_{3^k}$ is one of the curves described above, q is the largest prime dividing $\#E$, P is a point of order q on E, and the CDH problem is (t', ϵ')-hard in group $G = \langle P \rangle$. Let $H' : \{0,1\}^* \to \mathbb{F}_{3^\ell} \times \{0,1\}$ be a random oracle. Then the setup-adjustable scheme described above is $(t, q_H, q_{\mathsf{Sign}}, \epsilon)$-secure against existential forgery on adaptive chosen-message attacks in the random oracle model, where*

$$t \leq t' - 2c(\log q)(q_H + q_{\mathsf{Sign}}) - q_H \log(\#E/\mathbb{F}) - 2\tau, \text{ and } \epsilon \geq 2e \cdot q_{\mathsf{Sign}}\epsilon'$$

where c is a small constant and τ is equal to twice the time necessary to compute pairing on group G.

We omit a full proof of the theorem here, but note that the BLS signature scheme is $(t_1, q_H, q_{\mathsf{Sign}}, \epsilon_1)$-secure against existential forgery on adaptive chosen-message attacks according to Theorem C.3, where

$$t_1 \leq t' - 2c \lg p(q_H + q_S), \qquad \epsilon_1 \geq 2e \cdot q_S\epsilon'$$

By adjusting the concrete terms based on the concrete security of the hash function H and the concrete security over the field $\mathbb{F}_{3^k}$ instead of the group G as in [BLS01], we obtain the new bounds on t and ϵ.

Remark 5.5. *For simplicity, we only provided the analysis and experiments for supersingular curves $E_{3,2} : y^2 = x^3 + 2x + 2$, $E_{3,1} : y^2 = x^3 + 2x + 1$ over $\mathbb{F}_{3^k}$, hence focusing on the length parameters depcited in Table 1. It is possible to accomodate other choices of ℓ by searching for appropriate curves with appropriate security guarantees.*

Implementation details. We implement our construction using the elliptic curves discussed above. Our implementation uses the NTL library [Sho]. The most expensive feature of the BLS signature is the verification algorithm as it computes two pairing. The initialization of the bilinear group, the setup, signing and the verification algorithms are all timed separately in Table 1 using a 1.6GHz dual-core processor machine.

Curve	Signature size (bits)	k	Initialize (s)	Setup (s)	Signing (s)	Verify (s)	DLog security
$E_{3,2}(\mathbb{F}_{3^{17}})$	27	17	0.53	0.28	0.07	6.45	23
$E_{3,1}(\mathbb{F}_{3^{53}})$	85	53	11.39	5.64	0.33	152.37	82
$E_{3,2}(\mathbb{F}_{3^{79}})$	126	79	42.27	21.09	1.43	504.12	126
$E_{3,1}(\mathbb{F}_{3^{97}})$	154	97	57.38	39.24	1.86	989.42	154
E_{MNT}	149	N/A	0.03	0.31	0.02	4.29	149

Table 1: *Timing of setup-adjustable signatures for different values of ℓ.*

References

[BB04] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 56–73. Springer, May 2004.

[BCK96] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Message authentication using hash functionsthe hmac construction. *RSA Laboratories CryptoBytes*, 2(1):12–15, 1996.

[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 Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 1–18. Springer, August 2001.

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 501–519. Springer, March 2014.

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer, December 2001.

[BR96] Mihir Bellare and Phillip Rogaway. The exact security of digital signatures: How to sign with RSA and Rabin. In Ueli M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 399–416. Springer, May 1996.

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 280–300. Springer, December 2013.

[CHP07] Jan Camenisch, Susan Hohenberger, and Michael Østergaard Pedersen. Batch verification of short signatures. In Moni Naor, editor, *EUROCRYPT 2007*, volume 4515 of *LNCS*, pages 246–263. Springer, May 2007.

[CLT15] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. New multilinear maps over the integers. In *CRYPTO 2015, Part I*, LNCS, pages 267–286. Springer, August 2015.

[FGHP09] Anna Lisa Ferrara, Matthew Green, Susan Hohenberger, and Michael Østergaard Pedersen. Practical short signature batch verification. In Marc Fischlin, editor, *CT-RSA 2009*, volume 5473 of *LNCS*, pages 309–324. Springer, April 2009.

[For] Internet Engineering Task Force. Time-based one-time password algorithm. `https://tools.ietf.org/html/rfc6238`.

[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 *54th FOCS*, pages 40–49. IEEE Computer Society Press, October 2013.

[GGM84] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions (extended abstract). In *25th FOCS*, pages 464–479. IEEE Computer Society Press, October 1984.

[Goo] Google. Google authenticator. `https://en.wikipedia.org/wiki/Google_Authenticator`.

[HW09] Susan Hohenberger and Brent Waters. Short and stateless signatures from the RSA assumption. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 654–670. Springer, August 2009.

[Kob98] Neal Koblitz. An elliptic curve implementation of the finite field digital signature algorithm. In Hugo Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 327–337. Springer, August 1998.

[KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 13*, pages 669–684. ACM Press, November 2013.

[MNT01] Atsuko Miyaji, Masaki Nakabayashi, and Shunzou Takano. New explicit conditions of elliptic curve traces for fr-reduction. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 84(5):1234–1243, 2001.

[MOV+93] Alfred J Menezes, Tatsuaki Okamoto, Scott Vanstone, et al. Reducing elliptic curve logarithms to logarithms in a finite field. *Information Theory, IEEE Transactions on*, 39(5):1639–1646, 1993.

[MR02] Silvio Micali and Leonid Reyzin. Improving the exact security of digital signature schemes. *Journal of Cryptology*, 15(1):1–18, 2002.

[MVOV96] Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone. *Handbook of applied cryptography*. CRC press, 1996.

[oST] National Institute of Standards and Technology. Digital signature algorithm. `https://en.wikipedia.org/wiki/Digital_Signature_Algorithm`.

[PM99] Niels Provos and David Mazieres. Bcrypt algorithm. USENIX, 1999.

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

[RW14] Kim Ramchen and Brent Waters. Fully secure and fast signing from obfuscation. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 14*, pages 659–673. ACM Press, November 2014.

[Sho] Vitor Shoup. Number theory library. `http://www.shoup.net/ntl/`.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *46th ACM STOC*, pages 475–484. ACM Press, May / June 2014.

A A Concrete DL-based One Way Permutation

In the security proofs of our $i\mathcal{O}$-based constructions that are presented in this paper, we make use of a concrete one-way permutation, which here we show how to instantiate. We recall the definition of a (t, ϵ)-one-way permutation $\pi_\ell : \{0,1\}^\ell \to \{0,1\}^\ell$: the probability of finding the pre-image of $\pi_\ell(x)$ for a random ℓ-bit input x is at most ϵ for a t-time adversary. The construction of a (t, ϵ)-OWP based on the DL-problem in the curves discussed in Section 5 is given in Figure 7.

Input: $x \in \{0,1\}^\ell$

1. Given one of the values in Table 1, let $E/\mathbb{F}_{3^k}$ be the corresponding curve and q be the largest prime factor of the curve. Set $P \in E/\mathbb{F}_{3^k}$ be a point of order q.
2. Denote $x = x_1 x_2 \cdots x_\ell \in \{0,1\}^\ell$, and set $y = \sum_{i=1}^{\ell} 2^i x_i$. Set point $S = y \cdot P$. Output the bit representation of x-coordinate of point S as $\pi(x)$.

Figure 7: Description of (t, ϵ)-one-way-permutation

Based on the CDH hardness on the curve $E/\mathbb{F}_{3^k}$, we then have the following lemma:

Lemma A.1. *Suppose $E/\mathbb{F}_{3^k}$ is one of the curves described in Section 5, q is the largest prime dividing $\#E$, P is a point of order q on E, and CDH problem is (t, ϵ')-hard on group $G = \langle P \rangle$, then the above construction is a (t, ϵ)-one-way permutation.*

B Proofs

B.1 Proof of Theorem 4.8

Proof (sketch): The proof is similar to the proof of Theorem 4.1, consisting of a sequence of hybrid experiments, where the first hybrid corresponds to the original signature security experiment $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{sign\text{-}adj}}(\ell, 1^\lambda)$, and similar argument for indistinguishability between two consecutive hybrids.

- Hybrid H_0: In the first hybrid, the following experiment is played:
  1. $\{K_i\}_{i=1}^{\lambda}$ are chosen as keys for the puncturable PRFs $\{F_i\}_{i=1}^{\lambda}$. Then choose 2λ random bit strings $\{k_{i,b}\} \in \{0,1\}^{3\lambda}$, for $i \in [\lambda], b \in \{0,1\}$.
  2. The verification key vk given out as an obfuscation of the program described in Figure 4, 2λ random bit strings $\{k_{i,b}\}_{i \in [\lambda], b \in \{0,1\}}$ and a length parameter ℓ chosen from set $\mathcal{L}$ are sent to the adversary $\mathcal{A}$.

3. The adversary makes at most q_{sign} queries to the signing oracle on messages $(\boldsymbol{m}_i, \ell)$. For query $(\boldsymbol{m}_i, \ell)$, the signing oracle first computes temporary tag $\boldsymbol{t}_i = \oplus_{j=1}^{\lambda} k_{j,\boldsymbol{m}_i[j]}$, then outputs signature $\sigma = (\oplus_{j=1}^{\lambda} F_j(K_i, \boldsymbol{t}_i)|_\ell$.
4. The adversary outputs a forgery σ^* for challenge message m^*, and wins if $\mathsf{Verify}(\mathsf{vk}, m^*, \sigma^*) = 1$ and $|\sigma^*| = \ell$ hold.

- Hybrid H_1: In this hybrid, the challenger requires the message $\boldsymbol{m}^*$ in the forgery pair $(\boldsymbol{m}^*, \sigma^*)$ is not queried before. The rest of the hybrid remains unchanged.
- Hybrid H_2: In this hybrid, challenger changes the wining condition. First the challenger selects random indices (i', j') from $[\lambda] \times [q_{\mathsf{sign}}]$, and set $\boldsymbol{p} = \boldsymbol{t}_{j'}|_{i'} \oplus \boldsymbol{e}_{i'}$, where vector $\boldsymbol{e}_{i'}$ is the i'-th unit vector of length λ. For the forgery pair $(\boldsymbol{m}^*, \sigma)$ output by adversary $\mathcal{A}$, the challenger computes the temporary tag $\boldsymbol{t}^* = \oplus_{i=1}^{\lambda} k_{i,\boldsymbol{m}^*[i]}$ and changes the wining condition by enforcing an additional check: $\boldsymbol{t}^*|_{i'} = \boldsymbol{p}$. The rest of hybrid remains unchanged.
- Hybrid H_3: The same as hybrid H_2, except that the challenger first sets $\boldsymbol{z}^* = F_{i'}(K_{i'}, \boldsymbol{p})$ and then punctures the PRF $F_{i'}$ on $\boldsymbol{p} = \boldsymbol{t}_{j'}|_{i'} \oplus \boldsymbol{e}_{i'}$, i.e. $K_{i',\boldsymbol{p}} = \mathsf{Puncture}_{F_{i'}}(K_{i'}, \boldsymbol{p})$. Then challenger sets vk to be obfuscation of the program described in Figure 8, plus $2n$ random bit strings $\{k_{i,b}\}_{i\in[n],b\in\{0,1\}}$. The rest of hybrid remains unchanged.

Hardcoded: PRF keys $\{K_i\}_{i\neq i'}$), punctured key $K_{i',\boldsymbol{p}}$, and strings $\boldsymbol{p}, \boldsymbol{z}^*$.
Input: A message $\boldsymbol{m}$, a signature σ, a length parameter ℓ, and a temporary tag $\boldsymbol{t}$.

1. If $\boldsymbol{t}_{i'} = \boldsymbol{p}$, then if
$$\sigma \oplus_{i\neq\hat{i}} F_i(K_i, \boldsymbol{t}|_i)|_\ell = z^*|_\ell$$
output 1, otherwise output 0.
2. Else if
$$\sigma = \oplus_{i\neq i'} F_i(K_i, \boldsymbol{t}|_i)|_\ell \oplus F_{i',\boldsymbol{p}}(K_{i',\boldsymbol{p}}, \boldsymbol{t}|_{i'})|_\ell$$
output 1, otherwise output 0.

Figure 8: *Program Verify Signature* for signing-adjustable signature*

- Hybrid H_4: The same as hybrid H_3 except that we set $\boldsymbol{z}^* = \boldsymbol{u}|_\ell$, where $\boldsymbol{u}$ is chosen uniformly at random from the range of the puncturable PRF $F_{i'}$.
- Hybrid H_5: The same as hybrid H_4 except that we set $z^* = \pi_\ell(\boldsymbol{\alpha})$, where $\pi_\ell(\cdot)$ is a (t, ϵ)-one-way permutation.

The proof now follows through a series of claims, similarly to Theorem 4.1. Combining the description of hybrids $\mathsf{H}_0, \mathsf{H}_1, \mathsf{H}_2, \mathsf{H}_3, \mathsf{H}_4, \mathsf{H}_5$ and the corresponding (omitted) claims, we conclude that the advantage of t-time adversary in the existential unforgeability experiment is less than ϵ. □

B.2 Proof of Theorem 4.9

Proof (sketch): The proof of the theorem is similar to the previous two. Here we describe the sequence of hybrid experiments.

- Hybrid H_0: in the first hybrid, the following experiment is played:
1. $\{K_i\}_{i=1}^{\lambda}$ are chosen as keys for the puncturable PRFs $\{F_i\}_{i=1}^{\lambda}$. Then choose 2λ random bit strings $\{k_{i,b}\} \in \{0,1\}^{3\lambda}$, for $i \in [\lambda], b \in \{0,1\}$.

2. The verification key vk given out as an obfuscation of the program described in Figure 4, 2λ random bit strings $\{k_{i,b}\}_{i\in[\lambda],b\in\{0,1\}}$ and a length parameter ℓ chosen from set $\mathcal{L}$ are sent to the adversary $\mathcal{A}$.
3. The adversary makes at most q_{sign} queries to the signing oracle on messages $\boldsymbol{m}$. For query $\boldsymbol{m}_i$, the signing oracle first computes temporary tag $\boldsymbol{t}_i = \oplus_{j=1}^{\lambda} k_{j,\boldsymbol{m}_i[j]}$, then outputs signature $\sigma = \oplus_{j=1}^{\lambda} F_j(K_i, \boldsymbol{t}_i)$.
4. The adversary outputs a forgery tuple $(\boldsymbol{m}^*, \sigma^*, \ell)$ for challenge message m^*, and wins we have $\mathsf{Verify}(\mathsf{vk}, m^*, \sigma^*, \ell) = 1$.

- Hybrid H_1: In this hybrid, the challenger requires the message $\boldsymbol{m}^*$ in the forgery tuple $(\boldsymbol{m}^*, \sigma^*, \ell)$ is not queried before. The rest of the hybrid remains unchanged.
- Hybrid H_2: In this hybrid, challenger changes the wining condition. First the challenger selects random indices (i', j') from $[\lambda] \times [q_{\mathsf{sign}}]$, and set $\boldsymbol{p} = \boldsymbol{t}_{j'}|_{i'} \oplus \boldsymbol{e}_{i'}$, where vector $\boldsymbol{e}_{i'}$ is the i'-th unit vector of length λ. For the forgery pair $(\boldsymbol{m}^*, \sigma)$ output by adversary $\mathcal{A}$, the challenger computes the temporary tag $\boldsymbol{t}^* = \oplus_{i=1}^{\lambda} k_{i,\boldsymbol{m}^*[i]}$ and changes the wining condition by enforcing an additional check: $\boldsymbol{t}^*|_{i'} = \boldsymbol{p}$. The rest of hybrid remains unchanged.
- Hybrid H_3: The same as hybrid H_2, except that the challenger first sets $\boldsymbol{z}^* = F_{i'}(K_{i'}, \boldsymbol{p})$ and then punctures the PRF $F_{i'}$ on $\boldsymbol{p} = \boldsymbol{t}_{j'}|_{i'} \oplus \boldsymbol{e}_{i'}$, i.e. $K_{i',\boldsymbol{p}} = \mathsf{Puncture}_{F_{i'}}(K_{i'}, \boldsymbol{p})$. Then challenger sets vk to be obfuscation of the program described in Figure 9, plus $2n$ random bit strings $\{k_{i,b}\}_{i\in[n],b\in\{0,1\}}$. The rest of hybrid remains unchanged.

Hardcoded: PRF keys $\{K_i\}_{i\neq i'}$), punctured key $K_{i',\boldsymbol{p}}$, and strings $\boldsymbol{p}, \boldsymbol{z}^*$.
Input: A message $\boldsymbol{m}$, an adjusted signature σ, a length parameter ℓ and a temporary tag $\boldsymbol{t}$.

1. If $\boldsymbol{t}_{i'} = \boldsymbol{p}$, then if
$$\sigma \oplus_{i\neq\hat{i}} F_i(K_i, \boldsymbol{t}|_i)|_\ell = z^*|_\ell$$
output 1, otherwise output 0.
2. Else if
$$\sigma = \oplus_{i\neq i'} F_i(K_i, \boldsymbol{t}|_i)|_\ell \oplus F_{i',\boldsymbol{p}}(K_{i',\boldsymbol{p}}, \boldsymbol{t}|_{i'})|_\ell$$
output 1, otherwise output 0.

Figure 9: *Program Verify Signature* for the verification-adjustable signature scheme*

- Hybrid H_4: The same as hybrid H_3 except that we set $\boldsymbol{z}^* = \boldsymbol{u}_\ell$, where $\boldsymbol{u}$ is chosen uniformly at random from the range of the puncturable PRF $F_{i'}$.
- Hybrid H_5: The same as hybrid H_4 except that we set $z^* = \pi_\ell(\boldsymbol{\alpha})$, where $\pi_\ell(\cdot)$ is a (t, ϵ)-one-way permutation.

Combining the description of hybrids $\mathsf{H}_0, \mathsf{H}_1, \mathsf{H}_2, \mathsf{H}_3, \mathsf{H}_4, \mathsf{H}_5$ and corresponding (omitted) claims about their indistinguishability/security, we obtain that the advantage of a polynomial t-time adversary in the existential unforgeability experiment is less than ϵ. □

C Bilinear Maps and BLS Signatures

C.1 Bilinear maps and CDH

We now present a few facts related to groups with efficiently computable bilinear maps, followed by a specific number-theoretic assumption.

Let G and G_T be two multiplicative cyclic groups of prime order q. Let g be a generator of group G and e be a bilinear map, i.e., $e : G \times G \to G_T$ satisfying the following properties:

1. Bilinearity: for all $u, v \in G$ and $a, b \in \mathbb{Z}_q$, we have $e(u^a, v^b) = e(u, v)^{ab}$.
2. Non-degeneracy: $e(g, g) \neq 1$.

We say that G is a bilinear group if the group operation in G and the bilinear map $e : G \times G \to G_T$ are both efficiently computable. Notice that the map e is symmetric since $e(g^a, g^b) = e(g, g)^{ab} = e(g^b, g^a)$.

We present the *computational Diffie-Hellman* assumption (CDH) as follows. Choose a group G of prime order q according to the security parameter, and let $a, b \in \mathbb{Z}_q$ be chosen at random. The problem is, given (g, g^a, g^b), to compute g^{ab}.

Definition C.1. *We say that the CDH assumption holds if no polynomial-time adversary has a non-negligible advantage in solving the problem above.*

Remark C.2. *In this work, we use BLS signatures [BLS01] (described in Section 5) to construct setup-adjustable signatures. Therefore, we use a concrete version of the CDH assumption in the instantiation, namely, we say the that the CDH assumption is (t, λ, ϵ)-secure, if for any t-time adversary, the advantage of solving the problem above is at most $\epsilon(t, \lambda)$.*

BLS signatures. We described the algorithms $(\mathsf{Setup}, \mathsf{Sign}, \mathsf{Verify})$ of the BLS signature as follows. We let the group G be a cyclic group of size q generated by the generator g, where the CDH assumption is hard in G. We also let $G^* = G - \{1\}$. The signature scheme also makes use of a full-domain hash function $H : \{0,1\}^* \to G^*$.

$\mathsf{Setup}(1^\lambda)$: On input the security parameter λ, the setup algorithm picks a random $x \in \mathbb{Z}_q^*$ and computes $v = g^x$. The verification key is $\mathsf{vk} = v$, and the signing key is x.

$\mathsf{Sign}(\mathsf{sk}, \mu)$: On input the signing key $\mathsf{sk} = x$ and a message $m \in \{0,1\}^*$, the signing algorithm first computes $h = H(m)$ and $\sigma = h^x$, and outputs σ.

$\mathsf{Verify}(\mathsf{vk}, m, \sigma)$: On input the verification key $\mathsf{vk} = v$ and a message/signature pair (m, σ), the verification algorithm first computes $h = H(m)$ and outputs 1 if (g, v, h, σ) is a valid Diffie-Hellman tuple, i.e. $e(g, \sigma) \stackrel{?}{=} e(h, v)$, and otherwise outputs 0.

It is easy to verify the correctness of the BLS scheme. For security, we resort to the following theorem.

Theorem C.3. *Assuming the CDH assumption in group G is (τ, t', ϵ')-hard , the signature scheme is (t, q_H, q_S, ϵ)-hard against existential forgery on adaptive chosen-message attacks, where*

$$t \leq t' - 2c \lg p(q_H + q_S), \qquad \epsilon \geq 2e \cdot q_S \epsilon'$$

where c is a small constant, e is the base of natural logarithm, q_H, q_S denote the number of queries to the random oracle H and the signing oracle, respectively, and τ is equal to twice the time necessary to compute paring on group G.