# Optimal Signcryption from Any Trapdoor Permutation

Yevgeniy Dodis[†] Michael J. Freedman[†] Stanislaw Jarecki[‡] Shabsi Walfish[†]

January 28, 2004

**Abstract**

We build several highly-practical and optimized signcryption constructions directly from trapdoor permutations, in the random oracle model. All our constructions share features such as simplicity, efficiency, generality, near-optimal exact security, flexible and ad-hoc key management, key reuse for sending/receiving data, optimally-low message expansion, "backward" use for plain signature/encryption, long message and associated data support, the strongest-known qualitative security (so-called IND-CCA and sUF-CMA) and, finally, complete compatibility with the PKCS#1 infrastructure. While some of these features are present in previous works to various extents, we believe that our schemes improve on earlier proposals in at least several dimensions, making the overall difference quite noticeable in practice.

Concretely, we present three methods generally based on what we call Parallel, Sequential, and eXtended sequential Padding schemes (**P**-Pad, **S**-Pad, **X**-Pad). **P**-Pad offers parallel "signing" and "encrypting", optimal exact security, and minimum ciphertext length twice as long as the length of a TDP, while still maintaining optimal bandwidth. **S**-Pad loses parallelism and some exact security, but has minimal ciphertext length equal to that of a TDP. Any **S**-Pad can also be used as a "universal padding" scheme. **X**-Pad is similar to **S**-Pad, but regains optimal exact security at the expense of a marginally-longer minimum ciphertext length. Moreover, to unify various padding options, we construct a single *versatile* padding scheme PSEP which, by simply adjusting the lengths of the parameters, can work optimally as either a **P**-Pad, **S**-Pad or **X**-Pad.

**Keywords:** Signcryption, universal padding schemes, Feistel Transform, extractable commitments.

[†]{dodis,mfreed,walfish}@cs.nyu.edu. Department of Computer Science, New York University.

[‡]stasio@ics.uci.edu. Department of Computer Science, University of California, Irvine.

# 1 Introduction

**Signcryption.** Until recently, the two main building-blocks of modern public-key cryptography — encryption and signature schemes — have been considered as *distinct* entities that may be *composed* in various ways to ensure message privacy and authentication. From a design and analysis standpoint, this evolution makes sense, as encryption and signatures serve fundamentally different purposes. In practice, however, many centrally-important applications use both primitives, for example, secure electronic mail. Thus, it seems justified to invest special effort into designing a tailored and efficient solution to implement a "joint signature and encryption" *primitive*. Additionally, such a primitive may simplify and improve the design of complex protocols which require both privacy and authenticity. For instance, the following is a very simple signcryption-based authenticated key-exchange (AKE) protocol: sender Alice sends a fresh nonce $N$ to the receiver Bob, who chooses a session key $K$ and signcrypts $K$ to Alice using $N$ as a label (see Section 3).[1] This protocol both simplifies and generalizes a similar protocol from [21, §8.1], which used a specific "encrypt-then-sign" implementation of (labeled) signcryption [1], and also had to explicitly worry about low-level details such as user identities (which get abstracted away by the powerful signcryption primitive).

Motivated by these reasons, Zheng [23] introduced a new primitive called *signcryption*. While several papers [23, 24, 19, 12] offered security arguments about various signcryption schemes, the first formal investigations appeared only recently [2, 1]. Both works define signcryption as a multi-user primitive which simultaneously satisfies chosen ciphertext security for privacy and existential unforgeability for authenticity. In terms of constructions, Baek *et al.* [2] showed that the original "discrete log-based" proposal of Zheng [23] indeed can be proven secure in the random oracle (RO) model under the so-called Gap Diffie-Hellman assumption. Zheng's signcryption scheme is quite elegant and achieves very good efficiency in the discrete-log setting, but has the disadvantage that all parties must agree on the same public parameters, such as the common discrete log group. Thus, for example, any changes to the security parameter or signcryption scheme requires consensus. Additionally, the security of [2] is based on a specific and somewhat non-standard assumption. On the other hand, An *et al.* [1] examined generic composition methods of building signcryption from *any* secure signature and encryption scheme. These composition paradigms are general and give rise to a variety of signcryption schemes. Additionally, users can easily change their public keys or their favorite signature/encryption scheme, and still be able to seamlessly communicate with other users. While these generic schemes validate that signcryption can be built from ordinary signature and encryption, they are inefficient unless the latter are efficiently implemented.

**Our Motivation.** In practice, truly-efficient signature and encryption schemes, such as OAEP [4], OAEP+ [22], PSS-R [5], are typically built from trapdoor permutations (TDPs), such as RSA, and are analyzed in the RO model. We call such schemes TDP-based. Even with these efficient implementations, however, the generic schemes have several drawbacks. For example, users have to maintain independent keys for signature and for encryption, the message bandwidth is suboptimal (due to two independent "paddings" and additional overhead for identity fraud protection), and the scheme's "exact security" is not tight. Thus, given that practical schemes are already built from trapdoor permutations, it is natural to ask whether we can build optimized *direct* signcryption constructions from trapdoor permutations (in the RO model)[2] that resolve the inefficiencies of the "black-box" composition.

We resolve this question in the affirmative: This paper presents several optimized signcryption constructions, all of which share features such as simplicity, efficiency, generality, near-optimal exact security, flexible and ad-hoc key management, key reuse for sending/receiving data, optimally-low message expansion, "backward"

[1]Unlike Diffie-Hellman AKE, this protocol does not enjoy forward secrecy, but allows to build efficient AKE protocols under different assumptions than Diffie-Hellman, such as RSA.

[2]As we stated, *all* truly efficient plain signature and encryption schemes are analyzed in the RO model, so there seems to be little hope to avoid it for a more powerful signcryption primitive.

use for plain signature/encryption, long message and associated data support, the strongest-known qualitative security (so called IND-CCA and sUF-CMA) and, finally, complete compatibility with the PKCS#1 infrastructure [20]. While some of these attractive features are already present in several previous works to various extents, we believe that our schemes improve on earlier proposals in at least several dimensions, making the overall difference quite noticeable in practice. Note that we do not claim any improvement in the *computational* efficiency of signcryption based on TDPs, since in practice the computational overhead is completely dominated by the time required to compute and invert TDPs. In particular, our signcryption schemes will require exactly one computation of the "receiver's TDP" (for "encryption") and one inverse computation of the "sender's TDP" (for "authentication"), which is clearly optimal for TDP based signcryption (and can already be achieved with generic compositions in the RO model). Nevertheless, we will see in Table 1 that our schemes have many other advantages.

**Overview of Our Results.** Unlike generic TDP-based schemes, in our model each user $U$ independently picks a *single* trapdoor permutation $f_U$ (together with its trapdoor, denoted $f_U^{-1}$) and publishes $f_U$ as its public signcryption key. Similar to TDP-based signature and encryption schemes, our schemes use some *padding scheme* Pad on message $m$ before passing the result through corresponding TDPs. However, our schemes use only a *single*, specialized padding scheme, rather than two independent padding schemes. This design results in noticeable practical savings in both quantitative and qualitative security, as well as improves the message bandwidth and randomness utilization.

Specifically, to send a short message[3] $m$ from $S$ to $R$, we offer three options to $S$, depending on what padding scheme is more convenient in a given application:

- **P**-Pad (Parallel Padding) will produce $\mathsf{Pad}(m) = w\|s$, and $S$ will output $f_R(w)\|f_S^{-1}(s)$.
- **S**-Pad (Sequential Padding) will produce $\mathsf{Pad}(m) = w\|s$, and $S$ will output $f_R(f_S^{-1}(w\|s))$.
- **X**-Pad (eXtended sequential Padding) will produce $\mathsf{Pad}(m) = w\|s$, and $S$ will output $f_R(f_S^{-1}(w))\|s$.

As we can see, **P**-Pad provides parallel application of "signing" $f_S^{-1}$ and "encrypting" $f_R$, which can result in efficiency improvements on parallel machines. However, the ciphertext length is twice as large as compared to **S**-Pad, although the message bandwidth remains as good as **S**-Pad by using messages twice as long. On the other hand, the exact security offered by **S**-Pad is not as tight as that of **P**-Pad. Finally, **X**-Pad regains the optimal exact security of **P**-Pad, while maintaining ciphertext length nearly equal to the length of the TDP (by achieving quite short $s$).

Furthermore, we construct a single *versatile* padding scheme (called PSEP2) which, by simply adjusting some length parameters, can work optimally as either **P**-Pad, **S**-Pad or **X**-Pad! In addition, since each paradigm can naturally yield a regular signature/encryption by setting $f_S/f_R$ equal to the identity permutation (*i.e.* it is a "universal padding" [6]), this versatile padding scheme is truly applicable for any TDP-based public-key usage.

**Our Padding Constructions.** We observe that all popular padding schemes with message recovery currently used for ordinary signature or encryption, such as OAEP [3], OAEP+ [22], OAEP++ [15], PSS-R [5], and "scramble all, encrypt small" [13] (in the future denoted SAP), actually consist of two natural components $w$ and $s$. Moreover, these $w$ and $s$ are always obtained through an application of the Feistel Transform [17] — using a random oracle as the round function — to some more "basic" pair $\langle d, c\rangle$. Thus, rather than defining such specific paddings for our new application, we follow a more general approach. We define some simple, easily verified properties of $\langle d, c\rangle$, such that applying one or two rounds of the Feistel Transform to *any* such $\langle d, c\rangle$, we obtain the desired padding scheme. We show that the needed conditions on $\langle d, c\rangle$ are that they form an *extractable commitment scheme*, which is indeed a trivial condition to check and satisfy in the RO model.

[3] Later, we easily extend our signcryption scheme to support long messages, per [7].

For example, setting $c = H(m\|r)$, $d = (m\|r)$, we get a commitment scheme which defines PSS-R, while setting $c = (H(r) \oplus m)\|H'(r)$, $d = r$, we get a commitment scheme which leads to OAEP.

As special cases of our *one general theorem*, we not only obtain that analogs of OAEP, PSS-R, SAP, etc. are good for our new signcryption application, but: (1) get many of the previous results about signature and encryption as one special case of our general framework; (2) isolate and abstract the usefulness of the Feistel Transform in constructing TDP-based schemes; (3) design *new* padding schemes (without needing new proofs!) which may be specially tailored for particular situations. As an example of the last benefit, we introduce two new padding schemes that we call *Probabilistic Signature-Encryption Paddings* (PSEP). PSEP1 is a **P**-Pad scheme that is a "hybrid" of the standard PSS-R and OAEP paddings that also offers optimal message bandwidth in our setting. PSEP2 is a versatile padding scheme based on PSEP1 capable of achieving optimal bandwidth in *all* of our constructions.

Finally, our **S**-Pad schemes imply a general construction of *universal padding schemes* [6] from any trapdoor permutation. In particular, they generalize two of the three specific constructions in [16], which used a special case of PSS-R-based padding. Our PSEP2 scheme is also a universal padding with optimal bandwidth.

**Organization.** The rest of this paper is structured as follows. Section 2 reviews the literature on padding schemes, Section 3 introduces our notation and security definitions, Section 4 introduces our padding constructions, Section 5 uses these construction to build the new PSEP padding schemes, and Section 6 uses our padding schemes to build a full-fledged signcryption scheme that supports associated data and long messages.

# 2 Related Work

While padding schemes are very popular in the design of ordinary encryption and signature schemes (*e.g.*, [3, 5, 22, 9]), the most relevant previous works are those of [1, 18, 6, 16].

**Comparing with [1].** We already mentioned that our main improvement over the generic methods from [1] come in much improved message bandwidth, key reuse, better exact security, and better qualitative security ([1] cannot achieve both sUF-CMA- and IND-CCA-security, but achieves slightly weaker notions; see [1]). To best illustrate it, we consider the TDP-based implementation of the "commit-then-encrypt-and-sign" ($\mathcal{C}t\mathcal{E}\&\mathcal{S}$) from [1] and compare it to our parallel **P**-Pad approach. In $\mathcal{C}t\mathcal{E}\&\mathcal{S}$, one first applies any commitment scheme to transform a modified message $m'$, consisting of $m$ and hashes of two public keys, into a pair $\langle d, c\rangle$, and then encrypts $d$ and signs $c$. For the encryption and signature one applies two new, independent padding schemes to $d$ and $c$ to obtain $w$ and $s$, and only then applies a corresponding TDP to $w$ and $s$. Thus, the message is padded *four* times (hash of keys, commitment, signature and encryption). In fact, for currently best-known TDP-based encryption methods, one either has to lose exact security [22] or has to pad the message to be longer than the length of the TDP [15]. In contrast, we commit to $m$ once, directly getting $\langle d, c\rangle$, and then apply a deterministic, length-preserving Feistel Transform (where the public keys are only hashed into the round function, and do not affect bandwidth) to obtain the required $w$ and $s$. Moreover, we are guaranteed to always obtain tight exact security.

**Comparing with [18].** This work uses PSS-R padding for sequential signcryption with RSA. Namely, to transmit $RSA_R(RSA_S^{-1}(w\|s))$, where $w\|s$ is the result of PSS-R applied to the message $m$, and $RSA_U$ is the RSA key of user $U$. Thus, it is similar to our **S**-Pad paradigm, albeit restricted to RSA and PSS-R. Unfortunately, PSS-R does not happen to be a good **S**-Pad for general TDPs, and even with RSA the authors obtain very poor exact-security guarantees. For example, their results do not imply practical security guarantees even when using a 2048-bit RSA modulus. In addition, the work of [18] considers a much weaker notion of security for signcryption than we use. Interestingly, our work implies that applying one more Feistel round to PSS-R yields an optimal, secure **S**-Pad that works for any TDP.

**Comparing with [6, 16].** Our **S**-Pads are similar in spirit to the "universal padding" schemes defined by Coron *et al.* [6]. However, in their application, one applies such a padding to *either* a plain $\mathsf{TDP}$-based signature *or* a plain $\mathsf{TDP}$-based encryption, but not to *simultaneous* signature and encryption (*i.e.*, signcryption). While [6] constructed one concrete universal padding scheme ($\mathsf{PSS\text{-}R}$), with poor exact security and only specific to RSA, [16] gave three concrete padding schemes with nearly-optimal exact security for any $\mathsf{TDP}$.

Our work shows that universal paddings schemes are special cases of our **S**-Pad/**X**-Pad schemes. In fact, as we mentioned before, two special cases of our **S**-Pad/**X**-Pad constructions yield two constructions from [16]. However, some extra care needs to be taken to build **S**-Pads/**X**-Pads (for signcryption) from mere universal padding schemes (*e.g.*, to prevent "identity fraud" attacks [1]).

We note that both [6, 16] explicitly considered the question of key reuse for their plain "signature-encryption" application (as did the earlier work of [11]). However, their results do not imply similar results in our more complicated *signcryption* setting (and, anyway, we consider generalized padding schemes instead of proving key reuse on a case-by-case basis). For similar reason, they cannot be *directly applied* to conclude key reuse in the $\mathsf{TDP}$-based generic composition methods of [1].

**Summary of comparisons.** Table 1 summarizes the features of our constructions as compared to some of the previous work. Overall, we believe that our methods noticeably improve all previously-proposed signcryption schemes, both from practical and theoretical perspectives. We plan to propose our schemes as a basis for a new standard for public-key signcryption.

| | ZSCR [2] | TBOS [18] | $\mathcal{C}t\mathcal{E}\&\mathcal{S}$ / $\mathcal{S}t\mathcal{E}$ / $\mathcal{E}t\mathcal{S}$ [1] | **P**-Pad / **S**-Pad / **X**-Pad |
|---|---|---|---|---|
| Standard Assumption? | no | **yes** | **yes** | **yes** |
| Exact Security? | poor | very poor | **good** | **excellent / good / excellent** |
| Insider Security? | no | no | **yes** | **yes** |
| Multi-User Setting? | **yes** | no | **yes** | **yes** |
| $\mathsf{CCA}$ security? | **yes** | **yes** | no / **yes** / no | **yes** |
| Strong Unforgeability? | no* | no* | no / no / **yes** | **yes** |
| General Construction? | no | no | **yes** | **yes** |
| Key Flexibility? | no | no | **yes** | **yes** |
| Key Reuse (Short Key)? | **yes** | no* | no* | **yes** |
| Avoid Special Set-up? | no | **yes** | **yes** | **yes** |
| Extract Plain Sig / Enc? | no | only Sig | **yes** / Sig / Enc | **yes** |
| Associated Data? | no | no | no | **yes** |
| Compatible to PKCS#1? | no | maybe | maybe | **yes** |
| Parallel Operations? | n / a | no | **yes** / no / no | **yes** / no / no |
| Message Bandwidth | moderate | very poor | moderate | **optimal** |
| Minimal Ciphertext | $2k + \|m\|$ | $k$ | $2k$ / $k$ / $k$ | $2k$ / $k$ / $k + a$ |

Table 1: Comparison to prior schemes. A star * signifies that the question was not explicitly considered. For min ciphertext, $k, |m|, a$ are the lengths of the public-key domain, the message, and the security parameter.

## 3 Preliminaries

In this section, we formally define multi-party signcryption, some cryptographic primitives, and the extractable commitments that form the basis for our padding constructions.

## 3.1 Signcryption

Our modeling of signcryption is based on [1], except we generalize the latter definitions to include support for associated data (intuitively, a public label which is bound to the ciphertext), in order to provide more useful functionality and more general results.

**Syntax.** A signcryption scheme with associated data $\ell$ consists of the algorithms $(\mathsf{Gen}, \mathsf{SigEnc}, \mathsf{VerDec})$. In the multi-party setting, the $\mathsf{Gen}(1^\lambda)$ algorithm for user $U$ generates the key-pair $(\mathsf{SDK}_U, \mathsf{VEK}_U)$, where $\lambda$ is the security parameter, $\mathsf{SDK}_U$ is the signing/decryption key that is kept private, and $\mathsf{VEK}_U$ is the verification/encryption key made public. Without loss of generality, we assume that $\mathsf{VEK}_U$ is determined from $\mathsf{SDK}_U$.

The randomized signcryption algorithm $\mathsf{SigEnc}$ for user $U$ implicitly takes as input the user's secret key $\mathsf{SDK}_U$ and explicitly takes as input the message $m \in \mathcal{M}$, the label $\ell$ and the identity of the recipient, in order to compute and output the signcryption $\Pi$. For simplicity, we consider this identity $\mathsf{ID}$ to be a public key $\mathsf{VEK}$. Thus, we write this algorithm as $\mathsf{SigEnc}^{\ell}_{\mathsf{SDK}_U}(m, \mathsf{VEK}_R)$, or simply $\mathsf{SigEnc}^{\ell}_{U}(m, \mathsf{VEK}_R)$.

Similarly, user $U$'s deterministic de-signcryption algorithm $\mathsf{VerDec}$ implicitly takes the user's private $\mathsf{SDK}_U$ and explicitly takes as input the signcryption $\tilde{\Pi}$, the label $\ell$ and the senders' identity. We write $\mathsf{VerDec}^{\ell}_{\mathsf{SDK}_U}(\Pi, \mathsf{VEK}_S)$, or simply $\mathsf{VerDec}^{\ell}_{U}(\Pi, \mathsf{VEK}_S)$. The algorithm outputs some message $\tilde{m}$, or $\perp$ if the signcryption does not verify or decrypt successfully. Correctness ensures that for any users $S$ and $R$, $\mathsf{VerDec}^{\ell}_{R}(\mathsf{SigEnc}^{\ell}_{S}(m, \mathsf{VEK}_R), \mathsf{VEK}_S) = m$, for any $m$ and $\ell$.

**Security.** In this paper, we only use the strongest possible notion of *Insider* security for multi-user signcryption [1]. The security notions for our labelled algorithms are similar to those of standard signcryption [1], with the added requirement that $\ell$ is considered part of the ciphertext (for the purposes of $\mathsf{CCA}$ decryption oracle queries), and must be authenticated. However, there is no hiding requirement for $\ell$.

As expected, the security for signcryption consists on $\mathsf{IND\text{-}CCA}$ and $\mathsf{sUF\text{-}CMA}$ components when attacking some user $U$. Both games with the adversary, however, share the following common component. After $(\mathsf{SDK}_U, \mathsf{VEK}_U) \leftarrow \mathsf{Gen}(1^\lambda)$ is run and $\mathcal{A}$ gets $\mathsf{VEK}_U$, $\mathcal{A}$ can make up to $q_S$ adaptive signcryption queries $\mathsf{SigEnc}^{\ell}_{U}(m, \mathsf{VEK}_R)$ for *arbitrary* $\mathsf{VEK}_R$, as well as up to $q_D$ de-signcryption queries $\mathsf{VerDec}^{\ell}_{U}(\Pi, \mathsf{VEK}_S)$, again for arbitrary $\mathsf{VEK}_S$. (Of course, $m, \Pi, \ell$ can be arbitrary too).

The $\mathsf{IND\text{-}CCA}$ security of signcryption requires that no $\mathsf{PPT}$ adversary $\mathcal{A}$ can find some pair $m_0, m_1$ and a label $\ell$ for which he can distinguish $\mathsf{SigEnc}^{\ell}_{S}(m_0, \mathsf{VEK}_U)$ from $\mathsf{SigEnc}^{\ell}_{S}(m_1, \mathsf{VEK}_U)$. Notice, to create "valid" signcryptions that $\mathcal{A}$ must differentiate between, $\mathcal{A}$ must output the *secret key* $\mathsf{SDK}_S$ of the party $S$ sending messages to $U$. While seemingly restrictive, this is a *much stronger* guarantee than if $\mathcal{A}$ *did not know* the key of the sender. A good way to interpret this requirement is to say that even when *compromising* $S$, $\mathcal{A}$ still cannot "understand" messages that $S$ sent to $U$. In fact, we allow $\mathcal{A}$ to even *cook up* the secret key $\mathsf{SDK}_S$ without necessarily generating it via $\mathsf{Gen}$! Formally, for any $\mathsf{PPT}$ $\mathcal{A}$ running in time $t$,

$$\Pr\left[ b = \tilde{b} \;\middle|\; \begin{array}{c} (m_0, m_1, \ell, \mathsf{SDK}_S, \alpha) \leftarrow \mathcal{A}^{\mathsf{SigEnc}^{(\cdot)}_{U}(\cdot,\cdot), \mathsf{VerDec}^{(\cdot)}_{U}(\cdot,\cdot)}(\mathsf{VEK}_U, \mathsf{find}),\ b \xleftarrow{R} \{0,1\}, \\ \Pi \leftarrow \mathsf{SigEnc}^{\ell}_{S}(m_b, \mathsf{VEK}_U),\ \tilde{b} \leftarrow \mathcal{A}^{\mathsf{SigEnc}^{(\cdot)}_{U}(\cdot,\cdot), \mathsf{VerDec}^{(\cdot)}_{U}(\cdot,\cdot)}(\Pi, \ell;\ \alpha, \mathsf{guess}) \end{array} \right] \leq \frac{1}{2} + \varepsilon_{\mathsf{CCA}}$$

where $\varepsilon_{\mathsf{CCA}}$ is negligible in the security parameter $\lambda$, and $(\mathsf{SDK}_U, \mathsf{VEK}_U) \leftarrow \mathsf{Gen}(1^\lambda)$ is implicitly called at the beginning. In the guess stage, $\mathcal{A}$ only has the natural restriction of not querying $\mathsf{VerDec}_U$ with $(\Pi, \mathsf{VEK}_S, \ell)$, but can still use, for example, $(\Pi, \mathsf{VEK}_{S'}, \ell)$ for $\mathsf{VEK}_{S'} \neq \mathsf{VEK}_S$ or $(\Pi, \mathsf{VEK}_S, \ell')$ for $\ell \neq \ell'$.

For $\mathsf{sUF\text{-}CMA}$ security, no $\mathsf{PPT}$ $\mathcal{A}$ can forge a "valid" pair $(\Pi, \ell)$ (of some message $m$) from $U$ to *any* user $R$, provided that $\Pi$ was not previously returned from a query to $\mathsf{SigEnc}^{\ell}_{U}$. Again, in order to define "valid", we strengthen the definition by allowing $\mathcal{A}$ to come up with the presumed secret key $\mathsf{SDK}_R$ as part of his forgery. Formally, for any $\mathsf{PPT}$ $\mathcal{A}$ running in time $t$,

$$\Pr\left[ \mathsf{VerDec}^{\ell}_{R}(\Pi, \mathsf{VEK}_U) \neq \perp \;\middle|\; (\Pi, \ell, \mathsf{SDK}_R) \leftarrow \mathcal{A}^{\mathsf{SigEnc}^{(\cdot)}_{U}(\cdot,\cdot), \mathsf{VerDec}^{(\cdot)}_{U}(\cdot,\cdot)}(\mathsf{VEK}_U) \right] \leq \varepsilon_{\mathsf{CMA}}$$

where $\varepsilon_{\mathsf{CMA}}$ is negligible in the security parameter $\lambda$, $\mathsf{Gen}(1^\lambda)$ is implicit, and $\mathcal{A}$ did not obtain $(\Pi, \ell)$ in response to any $\mathsf{SigEnc}_U^\ell(m, \mathsf{VEK}_R, \ell)$ query. We call any scheme satisfying these properties a $(t, \varepsilon_{\mathsf{CCA}}, \varepsilon_{\mathsf{CMA}}, q_D, q_S)$-secure signcryption scheme.

### 3.2 TDPs and claw-free permutations

**Trapdoor permutations.** A family of trapdoor permutations (TDPs) is a family of permutations such that it is easy to randomly select a permutation $f$ and some "trapdoor" associated with $f$. Furthermore, $f$ is easy to compute and, given the trapdoor information, so is its inverse $f^{-1}$. However, without the trapdoor, $f$ is "hard" to invert on random inputs: No PPT adversary $\mathcal{A}$, given $y = f(x)$ for random $x$, can find $x$ with probability greater than $\varepsilon_{\mathsf{TDP}}$, which is negligible in the security parameter $\lambda$ of the generation algorithm.

**Claw-free permutations.** To improve the exact security of authentication in our constructions, we introduce a general class of TDPs — those induced by a family of *claw-free permutation* pairs [10]. In this context, the generation algorithm outputs $(f, f^{-1}, g)$, where $g$ is another efficient permutation over the same domain as $f$. The task of the PPT adversary $\mathcal{B}$ now is to find a "claw" $(x, z)$ such that $f(x) = g(z)$, which it succeeds at with probability at most $\varepsilon_{\mathsf{claw}}$, which negligible in $\lambda$. It is trivial to see that omitting $g$ from the generation algorithm induces a TDP family with $\varepsilon_{\mathsf{TDP}} \le \varepsilon_{\mathsf{claw}}$ (the reduction invokes $\mathcal{A}$ on a random $g(z)$). All known TDP families, such as RSA, Rabin, and Paillier, are easily seen to be induced by some claw-free permutation families with $\varepsilon_{\mathsf{claw}} = \varepsilon_{\mathsf{TDP}}$. Thus, a tight reduction to "claw-freeness" of such families implies a tight reduction to inverting them. On the other hand, it was shown by [8] that our restriction to claw-free permutations is necessary for tight signature reductions which we will achieve in this paper. We also remark that claw-free permutations are more general than "homomorphic TDPs" used by [16] for a similar reason.

### 3.3 Extractable Commitments

Our constructions for padding schemes all make use of *extractable commitment schemes*. Such commitments have usual properties of standard commitments, but with the additional twist that there exists an extraction algorithm which can extracts a unique decommitment from any valid commitment with high probability, by using some "trapdoor information". In the random oracle model,which is the model we consider here, such information consists of all the random oracle queries made to produce this valid commitment string.

**Syntax.** An extractable commitment scheme $\mathcal{C}$ consists of a triple of algorithms $(\mathsf{Commit}, \mathsf{Open}, \mathsf{Extract})$. Given a message $m \in \mathcal{M}$ and some random coins $r$, $\mathsf{Commit}(m; r)$ outputs a pair $(c, d)$, both $k$ bits long, where $c$ representing the commitment to $m$ and $d$ is a corresponding decommitment. As a shorthand, we write $(c, d) \leftarrow \mathsf{Commit}(m)$. $\mathsf{Open}(c, d)$ outputs $m$ if $(c, d)$ is a valid commitment/decommitment pair for $m$, or $\perp$ otherwise. Correctness requires $\mathsf{Open}(\mathsf{Commit}(m)) = m$ for all $m \in \mathcal{M}$.

We require this commitment scheme to satisfy two security properties:

**Hiding.** No PPT adversary can distinguish the commitment of any messages of its choice from a $k$-bit random string $R$. More formally, no PPT adversary $\mathcal{A}$ running in time $t$ can distinguish between the following two games with probability greater than $\varepsilon_{\mathsf{hide}}$, which is negligible in the secuirty parameter $\lambda$. In both games $\mathcal{A}$ chooses some message $m$, but gets either a properly generated commitment $c(m)$, or a random string $R$.

**Extractability.** There exists a deterministic poly-time algorithm Extract which can extract the "correct" decommitment from any valid commitment, given access to a transcript $\mathcal{T}$ of all RO queries previously issued by the adversary. Formally, for any PPT $\mathcal{A}$ running in time at most $t$,

$$\Pr[\mathsf{Extract}(c, \mathcal{T}) \neq d \wedge \mathsf{Open}(c, d) \neq \perp \mid (c, d) \leftarrow \mathcal{A}(1^\lambda)] \le \varepsilon_{\mathsf{extract}}$$

where $\mathcal{T}$ is a complete transcript of the RO queries made by $\mathcal{A}$ and $\varepsilon_{\mathsf{extract}}$ is negligible in $\lambda$. For syntactic convenience, we define Extract to output a random value in the event that the extraction algorithm "fails".

This completes the definition. A commitment scheme $\mathcal{C}$ is a $(t, \varepsilon_{\mathsf{hide}}, \varepsilon_{\mathsf{extract}})$-secure extractable commitment if it satisfies the above properties. We note that the "standard" notion of a commitment requires a binding property, instead of extractability. However, it is easy to see that a very strong form of binding is implied by extractability, as stated below (see Appendix A for the proof).

**Lemma 1 (Binding property of extractable commitments)** *It is computationally hard to produce $(c, d, d')$ such that $(c, d)$ and $(c, d')$ are valid commitment pairs and $d \neq d'$. Specifically, calling $\varepsilon_{\mathsf{bind}}$ the maximum probability of the adversary to come up with such $(c, d, d')$ in time $t$, we have $\varepsilon_{\mathsf{bind}} \leq 2\varepsilon_{\mathsf{extract}}$.*

When appropriate, we directly use $\varepsilon_{\mathsf{bind}}$ for conceptual clarity and because $\varepsilon_{\mathsf{bind}}$ may in fact be tighter than $2\varepsilon_{\mathsf{extract}}$. Notice, in the above Lemma the adversary cannot even come up with alternative decommitments to the same message $m$.

We will also use the following property of $(t, \varepsilon_{\mathsf{hide}}, \varepsilon_{\mathsf{extract}})$-secure extractable commitments: It is hard to find a commitment $c$ for which a random decommitment $d$ will be valid with non-negligible probability:

**Lemma 2** *$\forall \mathcal{A}$ running in time $t$,* $\Pr[\mathsf{Open}(c, d) \neq \perp \mid c \leftarrow \mathcal{A}(1^k); d \stackrel{R}{\leftarrow} \{0,1\}^k] \stackrel{\text{def}}{\leq} \varepsilon_{\mathsf{rand}} \leq \varepsilon_{\mathsf{extract}} + 2^{-k}$.

We make use of $\varepsilon_{\mathsf{rand}}$ for conceptual clarity and as, in fact, $\varepsilon_{\mathsf{rand}}$ may be tighter. See Appendix A for the proof.

## 4 Our Constructions

In this section, we construct tailored padding schemes with which one can apply both $\mathsf{TDP}$s $f_R$ and $f_S^{-1}$ *directly* to a *single* padded message to signcrypt it, *i.e.*, to simultaneously "encrypt" and "sign" the message by operating on a single padding. As explained in the Introduction, our direct methods considerably reduce the overhead introduced by the padding as compared to previous generic approaches in [1].

**Our Padding Schemes.** We consider three general paradigms for signcryption paddings. The first padding scheme, which we call **P**-Pads (short for Parallel Paddings), produces a pair $(w, s)$, such that a signcryption is computed as $f_R(w) \| f_S^{-1}(s)$. Note that the expensive $\mathsf{TDP}$ operations $f_R(w)$ and $f_S^{-1}(s)$ may be computed in parallel. The second type of padding scheme, which we call **S**-Pads (Sequential Paddings), outputs a single string $w \| s$, such that a signcryption is computed as $f_R(f_S^{-1}(w \| s))$. Although these **S**-Pads lose parallelism, the minimum length of a signcryptext is much shorter, and the structure conforms more closely to classical padding schemes for signature and encryption. Finally, the third type of padding scheme, which we call **X**-Pads (eXtended sequential Paddings), is a variant of the **S**-Pad and outputs a pair of strings $(w, s)$ such that signcryption is computed as $(f_R(f_S^{-1}(w)), s)$. It also loses parallelism, and its minimum length of a signcryptext is slightly longer than that of **S**-Pad, but it will allow us to achieve somewhat better exact security than that of **S**-Pad.

**Framework Based on Feistel Transforms.** We base the structure of our padding schemes on the well-known Feistel Transform. A Feistel Transform is an operation on a pair of left and right inputs $(L, R)$ which makes use of a "round function" $F$. Applying a single round of the Feistel Transform on a pair $(L, R)$ gives a new pair $(L', R')$ such that $L' = R$ and $R' = F(R) \oplus L$. The transform is very efficient in practice, and is invertible even if $F$ is not (in particular, we can invert by computing $L = F(L') \oplus R'$ and $R = L'$). Feistel Transforms are often used in multiple rounds with different *keyed* round functions, and have been especially useful in the design of block ciphers. In our application, the round function will be public, and will be modeled by the random oracle.

All our padding schemes will produce the pair $(w, s)$ by applying one (**P**-Pad) or two (**S**-Pad/**X**-Pad) rounds of Feistel transform to some "more basic" pair $(d, c)$. In fact, in all our constructions we will use any extractable commitment pair (per Section 3) as the input to the first round: the decommitment $d$ as the left hand input and the commitment $c$ as the right hand input. This will allow us to achieve a very high level of generality, and will also abstract away and emphasize the usefulness of the Feistel Transform in our constructions. Additionally, it will show that applying two rounds of the Feistel Transform results in what we call *versatile padding*: by simply varying the lengths of $c$ and $d$, the same padding can serve as **P**-Pad, **S**-Pad, **X**-Pad, and even as the padding for plain signature or encryption!

For technical reasons — notably, the possibility of "identity fraud" attacks — we specially format all inputs to the random oracle $G$ that serves as the first Feistel round function. We do this by prepending a meta-data string $\mathcal{L}$ to the oracle input, where $\mathcal{L}$ contains the public keys of the intended sender and recipient ($\mathsf{VEK}_S, \mathsf{VEK}_R$, respectively), as well as any desired associated data $\ell$. For simplicity, we use $\hat{G}(\cdot)$ to denote $G(\mathcal{L}, \cdot)$, where one can view $\hat{G}$ as an RO that is uniquely determined by $\mathcal{L}$. Using $\hat{G}$ as our round function rather than $G$ 'binds" the padded message to the meta-data, ensuring that no identity fraud has occurred and that the associated data $\ell$ has not been altered. Note that this technique represents a further optimization over the generic composition paradigm [1], which requires that a collision-resistant hash of the meta-data be included along with the message as input to the padding schemes.

We now describe our constructions and the corresponding security claims. In all definitions, we assume that the signcryption $\mathsf{Gen}$ algorithm generates a $(t, \varepsilon_{\mathsf{TDP}})$-secure $\mathsf{TDP}$ pair $\langle f_U, f_U^{-1} \rangle$ given to the honest user $U$. For security intuition, it is also instructive to think of the sender $S$ and the recipient $R$ as acting maliciously (since we are in the Insider-security model [1]).

### 4.1 P-Pad Schemes

We now describe a generic construction for a class of provably secure **P**-Pad schemes in the RO model, based on a single round of the Feistel Transform applied to any extractable commitment.

**Definition 1 (Feistel P-Pad)** *Let* $\mathcal{C} = (\mathsf{Commit}, \mathsf{Open}, \mathsf{Extract})$ *be any secure Extractable Commitment scheme. Furthermore, let* $G : \{0,1\}^* \to \{0,1\}^{|d|}$ *be a RO. The* Feistel **P**-Pad $\mathsf{Pad}^{\mathcal{L}}(m) \to (w, s)$ *(the padding of message* $m$ *using meta-data* $\mathcal{L}$*) induced by* $\mathcal{C}$ *is given by:*

$$\begin{aligned} (c, d) &\leftarrow \mathsf{Commit}(m) \\ w &\leftarrow c \\ s &\leftarrow \hat{G}(c) \oplus d \end{aligned}$$

*where* $\hat{G}(\cdot) \stackrel{def}{=} G(\mathcal{L}, \cdot)$*. The corresponding decoding operation* $\mathsf{DePad}^{\mathcal{L}}(w, s)$ *can be computed by first obtaining* $d = \hat{G}(w) \oplus s$ *and* $c = w$*, and then returning* $\mathsf{Open}(c, d)$*.*

Note that $(w, s)$ represents a Feistel Transform on input $\langle d, c \rangle$ using $\hat{G}$ as the round function. The following theorem states our main security claim about Feistel **P**-Pads, namely that $f_R(w) \| f_S^{-1}(s)$ is a secure signcryption with support for associated data, provided that properly-formed meta-data $\mathcal{L}$ is used in the padding.

**Theorem 1 (Signcryption from Feistel P-Pads)** *Let* $\mathcal{C}$ *be any* $(t, \varepsilon_{\mathsf{hide}}, \varepsilon_{\mathsf{extract}})$*-secure extractable commitment scheme, and* $\mathsf{Pad}$ *(and the corresponding* $\mathsf{DePad}$*) be the Feistel* **P**-*Pad induced by* $\mathcal{C}$*. Define the* $\mathsf{SigEnc}$ *and* $\mathsf{VerDec}$ *algorithms as follows:*

$$\begin{aligned} \mathsf{SigEnc}^{\ell}(m, \mathsf{VEK}_R = f_R) &\to (\psi = f_R(w) \| \sigma = f_U^{-1}(s)) && \text{where } (w, s) \leftarrow \mathsf{Pad}^{\mathcal{L}}(m) \\ \mathsf{VerDec}^{\ell}(\psi \| \sigma, \mathsf{VEK}_S = f_S) &\to \mathsf{DePad}^{\mathcal{L}}(w \| s) && \text{where } w \| s = f_U^{-1}(\psi) \| f_S(\sigma) \end{aligned}$$

*We require that, at a minimum, the meta-data* $\mathcal{L}$ *must contain the associated data* $\ell$*, as well as the published* $\mathsf{TDP}$*s of the sender and intended recipient of the message (*$f_S$ *and* $f_R$ *respectively).*

*Against any adversary allowed at most $q_G$ queries to the $G$ oracle, this signcryption scheme is a $(t', \varepsilon_{\mathsf{CCA}}, \varepsilon_{\mathsf{CMA}}, q_D, q_S)$-secure signcryption, where*

$$\begin{aligned}
t' &= t - \mathrm{O}((q_G + q_S) \cdot T_f) \\
\varepsilon_{\mathsf{CCA}} &\leq \varepsilon_{\mathsf{TDP}} + (q_S + 2) \cdot ((q_S + q_G) \cdot 2^{-|c|} + \varepsilon_{\mathsf{hide}}) + q_D \cdot \varepsilon_{\mathsf{rand}} + \varepsilon_{\mathsf{bind}} \\
\varepsilon_{\mathsf{CMA}} &\leq q_G \cdot \varepsilon_{\mathsf{TDP}} + q_S \cdot ((q_S + q_G) \cdot 2^{-|c|} + \varepsilon_{\mathsf{hide}}) + (q_D + 2) \cdot \varepsilon_{\mathsf{rand}} + 3\varepsilon_{\mathsf{extract}}
\end{aligned}$$

*If $f_U$ is taken from a family of $(t, \varepsilon_{\mathsf{claw}})$-secure claw-free permutations, we can improve the bound on $\varepsilon_{\mathsf{CMA}}$:*

$$\varepsilon_{\mathsf{CMA}} \leq \varepsilon_{\mathsf{claw}} + q_S \cdot ((q_S + q_G) \cdot 2^{-|c|} + \varepsilon_{\mathsf{hide}}) + (q_D + 2) \cdot \varepsilon_{\mathsf{rand}} + (q_G + 2) \cdot \varepsilon_{\mathsf{extract}}$$

The proof of Theorem 1 is given in Appendix B.

We note that, by a simple argument, the above theorem also implies that a modified syntax $f_R(w)\|s$ can be used as a secure encryption, provided the meta-data somehow indicates that plain encryption is the desired mode of operation. In a similar manner, $w\|f_S^{-1}(s)$ can be used as a secure signature. If $S$ and $R$ have TDPs with different input lengths, it is generally a simple matter to adjust the sizes of the $(c, d)$ pairs and the output length of the $G$ oracle, to accommodate the mismatch without any significant loss of exact security.

### 4.2 S-Pad and X-Pad Schemes

Unfortunately, our previous construction for Feistel **P**-Pads does not suffice to produce an **S**-Pad (which is strictly harder to achieve than a **P**-Pad). For example, we will see in Section 5 that the OAEP padding is a special case of our **P**-Pad construction, and yet it was shown to be potentially insecure when used as a single padding, by the result of [22]. On the positive side, we now show that it is easy (and efficient) to convert any Feistel **P**-Pad into an **S**-Pad by merely adding a second round of the Feistel Transform applied to the $\langle d, c\rangle$ pair.

**Definition 2 (Feistel S-Pad)** *Let $\mathcal{C} = (\mathsf{Commit}, \mathsf{Open}, \mathsf{Extract})$ be any secure Extractable Commitment scheme. Furthermore, let $G : \{0,1\}^* \to \{0,1\}^{|d|}$ and $H : \{0,1\}^* \to \{0,1\}^{|c|}$ be ROs. The* Feistel **S**-Pad $\mathsf{Pad}^{\mathcal{L}}(m) \to w\|s$ *(the padding of message $m$ using meta-data $\mathcal{L}$) induced by $\mathcal{C}$ is given by:*

$$\begin{aligned}
(c, d) &\leftarrow \mathsf{Commit}(m) \\
w &\leftarrow \hat{G}(c) \oplus d \\
s &\leftarrow H(w) \oplus c
\end{aligned}$$

*where $\hat{G}(\cdot) \stackrel{def}{=} G(\mathcal{L}, \cdot)$. The corresponding decoding operation $\mathsf{DePad}^{\mathcal{L}}(w\|s)$ can be computed by first obtaining $c = H(w) \oplus s$ and $d = \hat{G}(c) \oplus w$, and then returning $\mathsf{Open}(c, d)$.*

Note that $(w, s)$ represents a two round Feistel Transform on input $\langle d, c\rangle$ using $\hat{G}$ as the first round function and $H$ as the second round function. The following theorem states our main security claim about Feistel **S**-Pads, namely that $f_R(f_S^{-1}(w\|s))$ is a secure signcryption with support for associated data, provided that properly-formed meta-data $\mathcal{L}$ is used in the padding.

**Theorem 2 (Signcryption from Feistel S-Pads)** *Let $\mathcal{C}$ be any $(t, \varepsilon_{\mathsf{hide}}, \varepsilon_{\mathsf{extract}})$-secure extractable commitment scheme, and* Pad *(and the corresponding* DePad*) be the Feistel **S**-Pad induced by $\mathcal{C}$. Define the* SigEnc *and* VerDec *algorithms as follows:*

$$\begin{aligned}
\mathsf{SigEnc}^{\ell}(m, \mathsf{VEK}_R = f_R) &\to \Pi = f_R(f_U^{-1}((w\|s)) \qquad \text{where } w\|s \leftarrow \mathsf{Pad}^{\mathcal{L}}(m) \\
\mathsf{VerDec}^{\ell}(\Pi, \mathsf{VEK}_S = f_S) &\to \mathsf{DePad}^{\mathcal{L}}(w\|s) \qquad \text{where } w\|s = f_U^{-1}(f_S(\Pi))
\end{aligned}$$

*We require that, at a minimum, the meta-data $\mathcal{L}$ must contain the associated data $\ell$, as well as the published* $\mathsf{TDP}$*s of the sender and intended recipient of the message ($f_S$ and $f_R$ respectively).*

*Against any adversary allowed at most $q_G$ and $q_H$ queries to $G$ and $H$ oracles (respectively), this signcryption scheme is a $(t', \varepsilon_{\mathsf{CCA}}, \varepsilon_{\mathsf{CMA}}, q_D, q_S)$-secure signcryption, where*

$$\begin{aligned} t' &= t - \mathrm{O}((q_G + q_S + \underline{q_H \cdot q_G}) \cdot (T_f + T_{extract})) \\ \varepsilon_{\mathsf{CCA}} &\leq \varepsilon_{\mathsf{TDP}} + (q_H + q_G + q_S)^2 \cdot 2^{-|d|} + (q_S + q_D) \cdot ((2q_G + q_S) \cdot 2^{-|c|} + \varepsilon_{\mathsf{hide}} + \varepsilon_{\mathsf{extract}}) + 3q_G \cdot \varepsilon_{\mathsf{hide}} \\ \varepsilon_{\mathsf{CMA}} &\leq q_G \cdot \varepsilon_{\mathsf{TDP}} + (q_H + q_G + q_S)^2 \cdot 2^{-|d|} + (q_S + q_D) \cdot ((q_G + q_S) \cdot 2^{-|c|} + \varepsilon_{\mathsf{hide}} + 4\varepsilon_{\mathsf{extract}}) \end{aligned}$$

*If $f_U$ is taken from a family of $(t, \varepsilon_{\mathsf{claw}})$-secure claw-free permutations, we can improve the bound on $\varepsilon_{\mathsf{CMA}}$:*

$$\varepsilon_{\mathsf{CMA}} \leq \varepsilon_{\mathsf{claw}} + (q_H + q_G + q_S)^2 \cdot 2^{-|d|} + (q_S + q_D) \cdot ((q_G + q_S) \cdot 2^{-|c|} + \varepsilon_{\mathsf{hide}} + 3\varepsilon_{\mathsf{extract}}) + q_G \cdot \varepsilon_{\mathsf{extract}}$$

The proof of Theorem 2 is given in Appendix C. Interestingly, the proof uses a novel "trick" involving the meta-data input to the $G$ oracle (beyond its usage for identity fraud protection) which does not work for the seemingly symmetric case $f_S^{-1}(f_R(w\|s))$, and thus the order in which the $\mathsf{TDP}$s are applied is significant.[4] We note that, by the same argument used for **P**-Pads, the above theorem also implies that $f_R(w\|s)$ (resp. $f_S^{-1}(w\|s)$) can be used as a secure encryption (resp. signature). That is, a Feistel **S**-Pad can also be used as a standard universal padding scheme as described in [6, 16]. Furthermore, it is easy to show that any Feistel **S**-Pad can be used also as a Feistel **P**-Pad— *i.e.*, by computing $f_R(w)\|f_S^{-1}(s)$ — and thus can achieve the same exact security as **P**-Pads. As the cost of an additional Feistel round is minimal, we recommend the **S**-Pad construction for implementations, since they can be used in either paradigm, as the situation demands.

**X-Pads: Improving the exact security of Feistel S-Pads.** Unfortunately, in the sequential paradigm, **S**-Pads lose a potentially-significant amount of exact security (for the $\mathsf{IND\text{-}CCA}$ security guarantee only) when compared to **P**-Pads. This is due to the substantial increase in the $\mathsf{IND\text{-}CCA}$ reduction's running time, which requires time proportional to $q_H \cdot q_G$ (underlined in the statement of Theorem 2). We notice that the same loss of exact security (or worse) occurs in all known padding schemes for regular encryption, which place the entire padding inside the input of a $\mathsf{TDP}$ (as in [22]). However, if we are willing to place a small portion of the padding outside the $\mathsf{TDP}$ (as was done by [15] for $\mathsf{OAEP{+}{+}}$ encryption) — which slightly increases the minimum ciphertext length — we can avoid this loss of security. Conveniently, we can merely reuse our existing Feistel **S**-Pad construction as an **X**-Pad, for which we have a signcryption of the form $f_R(f_S^{-1}(w))\|s$, where $s$ is short. In particular, define a Feistel **X**-Pad to be a Feistel **S**-Pad with length parameters chosen appropriately for **X**-Pads.

**Theorem 3 (Signcryption from Feistel X-Pads)** *Let $\mathcal{C} = (\mathsf{Commit}, \mathsf{Open}, \mathsf{Extract})$ be a $(t, \varepsilon_{\mathsf{hide}}, \varepsilon_{\mathsf{extract}})$-secure extractable commitment scheme, and* $\mathsf{Pad}$ *(and the corresponding* $\mathsf{DePad}$*) be the Feistel **X**-Pad induced by $\mathcal{C}$. Define the* $\mathsf{SigEnc}$ *and* $\mathsf{VerDec}$ *algorithms as follows:*

$$\begin{aligned} \mathsf{SigEnc}^{\ell}(m, \mathsf{VEK}_R = f_R) &\rightarrow \Pi = f_R(f_U^{-1}((w))\|s \quad \text{where } (w,s) \leftarrow \mathsf{Pad}^{\mathcal{L}}(m) \\ \mathsf{VerDec}^{\ell}(\Pi = \psi\|s, \mathsf{VEK}_S = f_S) &\rightarrow \mathsf{DePad}^{\mathcal{L}}(w,s) \qquad\qquad \text{where } w = f_U^{-1}(f_S(\psi)) \end{aligned}$$

*We require that, at a minimum, the meta-data $\mathcal{L}$ must contain the associated data $\ell$, as well as the published* $\mathsf{TDP}$*s of the sender and intended recipient of the message ($f_S$ and $f_R$ respectively). This signcryption scheme*

[4] For technical reasons, it seems unlikely that this "symmetric" case can be proven secure, but there seems to be no advantage to using it in any case. This should be contrasted with the generic $\mathcal{E}t\mathcal{S}/\mathcal{S}t\mathcal{E}$ compositions, where both orders where equally effective [1].

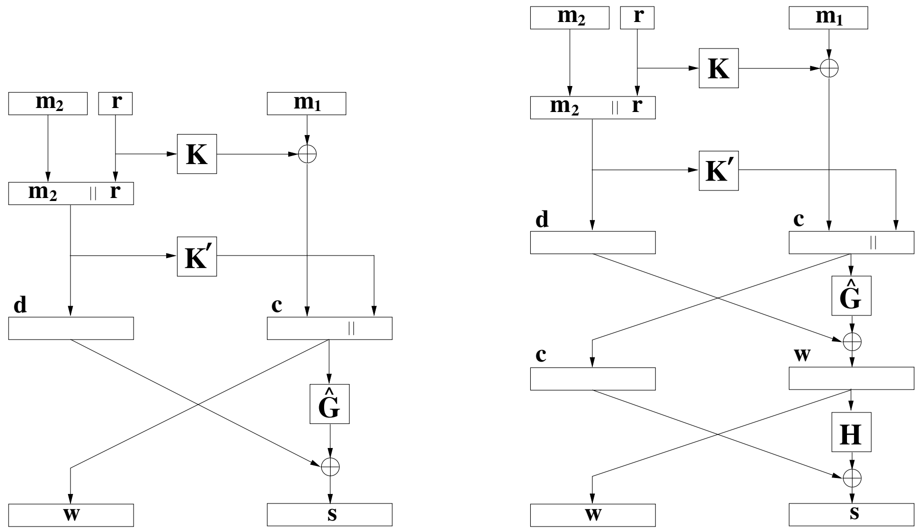


Figure 1: Schema for PSEP1 and PSEP2 on input $m = m_1 \| m_2$

*has the same exact security bounds as those of the Feistel* **S**-*Pads of Theorem 2 for both* TDP*s and claw-free permutations, but with an improvement in the running time of the reduction such that*

$$t' = t - \mathrm{O}((q_G + q_S + q_H) \cdot (T_f + T_{extract}))$$

The proof of Theorem 3 is given in Appendix C. The practical costs of this small increase in the minimum ciphertext length for **X**-Pads are generally not significant, but the resulting increase in exact security is substantial enough to warrant the application of Feistel **X**-Pad instead of Feistel **S**-Pad in most situations.

# 5 Probabilistic Signature and Encryption Padding (PSEP)

In this section, we instantiate our constructions with two new padding schemes we call *Probabilistic Signature and Encryption Paddings* (PSEP) which are designed to provide optimal bandwidth and flexibility. These two paddings, PSEP1 and PSEP2, are constructed by applying the **P**-Pad and **S**-Pad constructions (respectively) to the following extractable commitment scheme, using random oracles $K : \{0,1\}^* \to \{0,1\}^{|m_1|}$ and $K' : \{0,1\}^* \to \{0,1\}^{|c|-|m_1|}$,

$$\begin{aligned} c &\leftarrow \Big(m_1 \oplus K(r)\Big) \parallel K'(m_2 \| r) \\ d &\leftarrow (m_2 \| r) \end{aligned}$$

The scheme is parameterized by the selection of the lengths of $c$,$d$,$m_1$,$m_2$ (denoted $|c|$,$|d|$,$|m_1|$,$|m_2|$,resp.).

The following Lemma gives exact security for the commitment scheme used in PSEP, in terms of the relevant selectable parameters. See Appendix D for the proof.

**Lemma 3** *The commitment scheme* $\langle d = m_2 \| r,\ c = (m_1 \oplus K(r)) \| K'(m_2 \| r)\rangle$ *defining* PSEP *satisfies:*

$$\varepsilon_{\mathsf{hide}} \le (q_K + q_{K'}) \cdot 2^{-(|d|-|m_2|)} \ ; \ \varepsilon_{\mathsf{extract}} \le (q_{K'}^2 + 1) \cdot 2^{-(|c|-|m_1|)} \ ; \ \varepsilon_{\mathsf{bind}} \le 2 \cdot \varepsilon_{\mathsf{extract}} \ ; \ \varepsilon_{\mathsf{rand}} \le 2^{-(|c|-|m_1|)}$$

*where* $q_K$ *and* $q_{K'}$ *are the number of oracle queries to* $K$ *and* $K'$ *made by the adversary.*

Using this commitment pair $\langle d, c\rangle$, we can apply a single round of the Feistel Transform to yield $\mathsf{PSEP1}$: $\Big\langle w \leftarrow c\ ;\ s \leftarrow \hat{G}(w) \oplus d \Big\rangle$, as shown in Figure 1. $\mathsf{PSEP1}$ is sufficient for use as a Feistel **P**-Pad for signcryption. Interestingly, it can be seen that both $\mathsf{OAEP}$ [3] and $\mathsf{PSS\text{-}R}$ [5] are special cases of $\mathsf{PSEP1}$ for appropriate selections of the commitment scheme parameters. The parameters corresponding to $\mathsf{OAEP}$ ($|m_1| = 0$) and $\mathsf{PSS\text{-}R}$ ($|m_2| = 0$), however, are not bandwidth-optimal for **P**-Pads (where one wants to "balance" $|c|$ and $|d|$). For example, if $|c| = |d| = k$, both $\mathsf{OAEP}$ and $\mathsf{PSS\text{-}R}$ would require to set $|m| \leq k$, while the total length $2k$ of $\mathsf{PSEP1}$ potentially allows one to fit $|m| \approx 2k$, which we can indeed do by splitting $m$ almost evenly.

Rather than instantiating $\mathsf{PSEP1}$, we recommend applying a second round of Feistel (a very inexpensive operation). This yields the scheme $\mathsf{PSEP2}$, also shown in Figure 1. $\mathsf{PSEP2}$ can be used in any of the three modes discussed in Section 4, *i.e.*, it can be used as a **P**-Pad, **S**-Pad, or **X**-Pad. Appropriate selection of the commitment scheme parameters can be used to achieve optimal bandwidth in any of these modes — for any desired level of exact security for the extractable commitment.

Note that although the $\mathsf{PSEP2}$ scheme would be rather difficult to analyze directly, in our general framework the proof of simple Lemma 3 is all one needs to obtain many useful results. Namely, by leveraging the Theorems in Section 4 we get tight exact security bounds for $\mathsf{PSEP2}$ showing that it can be used as a **P**-Pad, **S**-Pad, or **X**-Pad for signcryption. Moreover, it is also a secure universal padding scheme (for either plain signature or encryption), and it is safe to reuse public keys with any combination of these primitives for both sending and receiving.

# 6 Signcrypting Long Messages

Using the "concealment" approach described in [7], we can extend any short-message signcryption scheme with support for associated data to include support for long messages. Although arbitrary concealment schemes will suffice, for efficiency purposes we consider concealments utilizing any one-time $(t, \varepsilon_{\mathsf{OTE}})$-secure symmetric encryption scheme $(E, D)$[5] as advocated in [7]. There are many *very efficient* such symmetric encryptions, *i.e.*, $M \oplus F(\tau)$ works when $F$ is a RO (but there are many RO-free encryptions to choose from as well; see [7]).

Specifically, let $\mathcal{SC} = (\mathsf{Gen}, \mathsf{SigEnc}, \mathsf{VerDec})$ be any signcryption scheme on $\hat{n}$-bit messages or longer, with support for associated data, and $(E, D)$ be any one-time encryption scheme with keysize $\hat{n}$ (thus, $\hat{n} \approx 128$ suffices). We define a signcryption scheme $\mathcal{SC}' = (\mathsf{Gen}, \mathsf{SigEnc}', \mathsf{VerDec}')$ on long messages with support for associated data as follows. Let $\mathsf{SigEnc}'^{\ell}(M) = \pi \| \mathsf{SigEnc}^{L}(\tau)$, where $\pi = E_\tau(M)$, $L = \ell \| \pi$, and $\tau$ is a random $\hat{n}$-bit string. Similarly, $\mathsf{VerDec}'^{\ell}(\pi \| \Pi) = D_\tau(\pi)$, where $\tau = \mathsf{VerDec}^{L}(\Pi)$ and $L = \ell \| \pi$.

**Theorem 4** *If $\mathcal{SC}$ is $(t, \varepsilon_{\mathsf{CCA}}, \varepsilon_{\mathsf{CMA}}, q_D, q_S)$-secure and $(E, D)$ is $(t, \varepsilon_{\mathsf{OTE}})$-secure (with encryption/decryption time $T_{\mathsf{OTE}}$), then $\mathcal{SC}'$ is $(t - \mathrm{O}((q_D + q_S) \cdot T_{\mathsf{OTE}}), \varepsilon_{\mathsf{CCA}} + \varepsilon_{\mathsf{OTE}}, \varepsilon_{\mathsf{CMA}}, q_D, q_S)$-secure.*

The proof of this theorem (adapted from [7] for signcryption, with exact security) is given in Appendix E. This result implies that our signcryption constructions — and indeed the separate signature and encryption constructions that they induce — can easily support long messages: Simply apply any symmetric key encryption to the message, and signcrypt the symmetric key while including the encrypted message inside the meta-data $\mathcal{L}$. Additionally, it is possible to move a portion of the message into the padding alongside the encryption key to save otherwise wasted space. The final result is that the overhead for long messages is the same as that for short messages plus the length of the symmetric key, which will typically be approximately 128-bits.

[5] *I.e.*, no distinguisher in time $t$ can tell $E_\tau(M_0)$ from $E_\tau(M_1)$ for any two messages $(M_0, M_1)$ with probability greater than $\varepsilon_{\mathsf{OTE}}$. Notice, the distinguisher is not given either the encryption or the decryption oracles.

# References


[1] Jee Hea An, Yevgeniy Dodis, and Tal Rabin. On the security of joint signature and encrytion. In Lars Knudsen, editor, *Advances in Cryptology—EUROCRYPT 2002*, Lecture Notes in Computer Science. Springer-Verlag, 28 April–2 May 2002. Available from `http://eprint.iacr.org/2002/046/`.

[2] Joonsang Baek, Ron Steinfeld, and Yuliang Zheng. Formal proofs for the security of signcryption. In David Naccache and Pascal Pailler, editors, *5th International Workshop on Practice and Theory in Public Key Cryptosystems — PKC 2002*, volume 2274 of *Lecture Notes in Computer Science*. Springer-Verlag, February 2002.

[3] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communication Security*, pages 62–73, November 1993. Revised version appears in `http://www-cse.ucsd.edu/users/mihir/papers/crypto-papers.html`.

[4] Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In Alfredo De Santis, editor, *Advances in Cryptology—EUROCRYPT 94*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111. Springer-Verlag, 1995, 9–12 May 1994. Revised version available from `http://www-cse.ucsd.edu/users/mihir/`.

[5] Mihir Bellare and Phillip Rogaway. The exact security of digital signatures: How to sign with RSA and Rabin. In Ueli Maurer, editor, *Advances in Cryptology—EUROCRYPT 96*, volume 1070 of *Lecture Notes in Computer Science*, pages 399–416. Springer-Verlag, 12–16 May 1996. Revised version appears in `http://www-cse.ucsd.edu/users/mihir/papers/crypto-papers.html`.

[6] Jean-Sébastian Coron, Marc Joye, David Naccache, and Pascal Pailler. Universal padding schemes for RSA. In Moti Yung, editor, *Advances in Cryptology—CRYPTO 2002*, Lecture Notes in Computer Science. Springer-Verlag, 18–22 August 2002. Available from `http://eprint.iacr.org/2002/115/`.

[7] Yevgeniy Dodis and Jee Hea An. Concealment and its applications to authenticated encryption. In Eli Biham, editor, *Advances in Cryptology—EUROCRYPT 2003*, Lecture Notes in Computer Science. Springer-Verlag, 4 May–8 May 2003.

[8] Yevgeniy Dodis and Leonid Reyzin. On the power of claw-free permutations. In *Conference on Security in Communication Networks*, 2002.

[9] Eiichiro Fujisaki, Tatsuaki Okamotoi, David Pointcheval, and Jacques Stern. RSA-OAEP is secure under the RSA assumption. In Kilian [14], pages 260–274.

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

[11] Stuart Haber and Benny Pinkas. Combining public key cryptosystems. In Pierangela Samarati, editor, *Eighth ACM Conference on Computer and Communication Security*. ACM, November 5–8 2001.

[12] W. He and T. Wu. Cryptanalysis and improvement of petersen-michels signcryption schemes. *IEEE Computers and Digital Communications*, 146(2):123–124, 1999.

[13] Markus Jakobsson, Julien P. Stern, and Moti Yung. Scramble all, encrypt small. In Lars Knudsen, editor, *Fast Software Encryption: 6th Internation Workshop, FSE1999*, volume 1636 of *Lecture Notes in Computer Science*, pages 95–111. Springer-Verlag, 1999.

[14] Joe Kilian, editor. *Advances in Cryptology—CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*. Springer-Verlag, 19–23 August 2001.

[15] Kazukuni Kobara and Hideki Imai. Oaep++ : A very simple way to apply oaep to deterministic ow-cpa primitives. Cryptology ePrint Archive, Report 2002/130, 2002.

[16] Yuichi Komano and Kazuo Ohta. Efficient universal padding techniques for multiplicative trapdoor one-way permutation. In Dan Boneh, editor, *Advances in Cryptology—CRYPTO 2003*, Lecture Notes in Computer Science. Springer-Verlag, 17–21 August 2002.

[17] Michael Luby and Charles Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM Journal on Computing*, 17(2):373–386, 1988.

[18] Wenbo Mao and John Malone-Lee. Two birds one stone: Signcryption using RSA. In Marc Joye, editor, *Progress in Cryptology — CT-RSA 2003*, Lecture Notes in Computer Science. Springer-Verlag, 13–17 April 2003.

[19] H. Petersen and M. Michels. Cryptanalysis and improvement of signcryption schemes. *IEEE Computers and Digital Communications*, 145(2):140–151, 1998.

[20] *PKCS #1: RSA Encryption Standard. Version 1.5.* RSA Laboratories, November 1993. Available from `http://www.rsa.com/rsalabs/pubs/PKCS/`.

[21] Victor Shoup. On formal models for secure key exchange. Cryptology ePrint Archive, Report 1999/012, 1999.

[22] Victor Shoup. OAEP reconsidered. In Kilian [14], pages 240–259.

[23] Y. Zheng. Digital signcryption or how to achieve cost(signature & encryption) $\ll$ cost(signature) + cost(encryption). In Burton S. Kaliski Jr., editor, *Advances in Cryptology—CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 165–179. Springer-Verlag, 17–21 August 1997.

[24] Y. Zheng and H. Imai. Efficient signcryption schemes on elliptic curves. *Information Processing Letters*, 68(6):227–233, December 1998.

# A Properties of Extractable Commitments

We now offer proofs for the binding and randomness properties of extractable commitments.

**Proof of binding property (Lemma 1).**

**Proof:** Consider a reduction $\mathcal{B}$ against the extractability property of the commitment scheme as follows. $\mathcal{B}$ runs $\mathcal{A}$ and obtains $(c, d, d')$ if $\mathcal{A}$ succeeds. $\mathcal{B}$ then randomly outputs $(c, d)$ or $(c, d')$ with equal probability. Since $\mathsf{Extract}(c, \mathcal{T})$ is a deterministic value, it matches the output of $\mathcal{B}$ with probability at most $1/2$. In the event that it does not match, $\mathcal{B}$ has broken the extractability property. Since this must happens with probability at most $\varepsilon_{\mathsf{extract}}$, we find that $\mathcal{A}$ succeeds with probability at most $2\varepsilon_{\mathsf{extract}}$. □

**Proof of random decommitment property (Lemma 2).**

**Proof:** Consider a reduction $\mathcal{B}$ against the extractability property of the commitment scheme as follows. $\mathcal{B}$ runs $\mathcal{A}$ and obtains $c \leftarrow \mathcal{A}(1^k)$, chooses a $d$ uniformly at random, and returns $(c, d)$. The probability that $\mathcal{B}$ succeeds is at least the probability that $\mathcal{A}$ succeeds minus the probability that $d = \mathsf{Extract}(c, \mathcal{T})$. Since $d$ is chosen randomly, the probability that $d = \mathsf{Extract}(c, \mathcal{T})$ is $2^{-k}$. The lemma follows. □

# B Feistel P-Pad Proof

The proof of security requires us to establish bounds for both $\varepsilon_{\mathsf{CCA}}$ and $\varepsilon_{\mathsf{CMA}}$ (the maximum success probabilities for PPT adversaries in the IND-CCA and sUF-CMA games, respectively). We first prove the bound on $\varepsilon_{\mathsf{CCA}}$. Recall that $f_U$ will represent the key of the honest party (*i.e.*, it is the only key not controlled by the adversary). In particular, $f_U$ is the key of recipient in the IND-CCA game, and the key of the sender in the sUF-CMA game.

## B.1 Bounding $\varepsilon_{\mathsf{CCA}}$

Consider an adversary $\mathcal{A}_{\mathsf{CCA}}$ against the IND-CCA security of the scheme. We show the bound on $\mathcal{A}_{\mathsf{CCA}}$'s success probability by proceeding through a sequence of games $\mathbf{G}_0, \ldots, \mathbf{G}_4$, where $\mathbf{G}_0$ is the unmodified IND-CCA attack game. We define the event $S_i$ to be the probability that $\mathcal{A}_{\mathsf{CCA}}$ is successful in $\mathbf{G}_i$. Each new game $\mathbf{G}_i$ will have a corresponding analysis bounding the difference between $\Pr[S_i]$ and $\Pr[S_{i-1}]$. The final bound on $\mathcal{A}_{\mathsf{CCA}}$'s advantage in $\mathbf{G}_0$ will follow from the success probability $\Pr[S_4]$, and the bound on $\Pr[S_4] - \Pr[S_0]$ which can be established from the relative bounds on the intermediate games. Throughout the following analysis we will use $\Pi^* = \psi^* \| \sigma^* = f_U(w^*) \| f_S^{-1}(s^*)$ to denote the challenge ciphertext corresponding to $m_b$ (where $b$ is the challenge bit) that is issued to $\mathcal{A}_{\mathsf{CCA}}$ in the IND-CCA attack game in response the challenge oracle query ($\mathcal{L}^*$ will be used to denote the corresponding meta-data, which depends on the adversary's choice of sender $f_S$ and associated data $\ell^*$ used for the challenge query).

**Game $\mathbf{G}_1$:**

Let $\mathbf{G}_1$ be the same as $\mathbf{G}_0$ (the unmodified IND-CCA attack game), except we replace the random oracles and the SigEnc and VerDec oracles with the following simulations:

**Random oracle simulation for IND-CCA proof.** For each query $\mathcal{A}_{\mathsf{CCA}}$ makes to $G$, of the form $G(\mathcal{L}, c)$, the simulation first checks to see if $G(\mathcal{L}, c)$ was previously defined. If so, the simulation replies with the previously defined value. Otherwise, it examines $\mathcal{L}$ to determine whether $f_U$ is specified as the recipient (if not, the oracle simulation proceeds in the standard fashion).

If $f_U$ is the recipient, the simulation chooses a random $x \in \{0,1\}^{|d|}$ and defines $G(\mathcal{L}, c)$ to be $x$. The simulation then records the pair $(f_U(c), c)$ in a lookup table for future reference (used in the VerDec oracle simulation), and returns $x$.

Note that the distribution of values returned by this random oracle simulation is still uniformly random, so this simulation is perfectly "honest". However, it now takes time approximately $T_f$ to compute before responding to each query.

VerDec **oracle simulation for IND-CCA proof.** When a query $\mathsf{VerDec}^{\ell}(\Pi, f_S)$ is issued, the simulation parses $\Pi = \psi \| \sigma$. If the query was of the form $\psi^* \| \sigma$, the oracle immediately returns $\perp$ (*i.e.*, when adversary attempts to reuse the first portion of its challenge ciphertext in a query with an altered second portion). If the query was not of this form, the simulation then searches the lookup table to find a matching $\psi$ in a pair $(\psi, c)$. If no such pair exists, the simulation returns $\perp$, otherwise it properly forms $\mathcal{L}$ and returns $\mathsf{DePad}^{\mathcal{L}}(c, f_S(\sigma))$. We note that if the lookup succeeds, we indeed have that $c = w = f_U^{-1}(\psi)$, so that the DePad will return the correct output.

SigEnc **oracle simulation for IND-CCA proof.** When a query $\mathsf{SigEnc}^{\ell}(m, f_R)$ is issued, the simulation properly forms $\mathcal{L}$ and computes the response as follows:

- Choose a random $x \in \{0,1\}^{|d|}$, and compute $y = f_U(x)$.
- Compute $(c, d) \leftarrow \mathsf{Commit}(m)$.
- Define $G(\mathcal{L}, c)$ to be $y \oplus d$. Fail if $G(\mathcal{L}, c)$ was already defined.
- Return $f_R(c) \| x$.

Observe, if the procedure does not fail, it returns a valid response to the SigEnc query, as $x = f_U^{-1}(G(\mathcal{L}, c) \oplus d)$.

This completes the definitions of the oracle simulations which $\mathbf{G}_1$ uses to replace the functionality of the corresponding oracles from $\mathbf{G}_0$. Provided that these oracle simulations do not fail, $\mathbf{G}_0$ is identical to $\mathbf{G}_1$, so $\Pr[S_1] - \Pr[S_0]$ is bounded by the probability that a failure occurs.

The $\mathsf{VerDec}$ oracle simulation can only fail by incorrectly returning $\perp$ when queried on a valid ciphertext. There are two possible scenarios where this occurs: either $\mathcal{A}_{\mathsf{CCA}}$ queried a valid ciphertext containing a $w$ for which it did not issue a corresponding $G(\mathcal{L}, c)$ oracle query (and thus $\mathcal{A}_{\mathsf{CCA}}$ could not have any information about the corresponding $s$), or $\mathcal{A}_{\mathsf{CCA}}$ queried a valid ciphertext that reused the $\psi^* = f_U(w^*)$ portion from the $\mathsf{IND\text{-}CCA}$ game challenge ciphertext with some $\sigma \neq \sigma^*$.

In the first case, since $G(\mathcal{L}, c)$ was never queried, it is random and independent of the queried ciphertext, which means that the corresponding $d = G(\mathcal{L}, c) \oplus s$ will be random. By definition, the probability that such a random $d$ represents a valid decommitment for $c$ is bounded by $\varepsilon_{\mathsf{rand}}$. Thus the total probability that a failure of this type occurs after $q_D$ queries to the $\mathsf{VerDec}$ oracle simulation is at most $q_D \cdot \varepsilon_{\mathsf{rand}}$. The second kind of failure can only occur if $\mathcal{A}_{\mathsf{CCA}}$ can find $s \neq s^*$ such that $\mathsf{DePad}(w^*, s) \neq \perp$. However, since $s \neq s^*$, this implies that $\mathcal{A}_{\mathsf{CCA}}$ found some $d \neq d^*$ that is a valid decommitment for $c^* = w^*$. Since both $(c^*, d^*)$ and $(c^*, d)$ are valid commitment/decommitment pairs, $\mathcal{A}_{\mathsf{CCA}}$ would have to break the binding property of the commitment scheme to cause this kind of failure. Thus the total probability that the $\mathsf{VerDec}$ simulation fails is bounded by $q_D \cdot \varepsilon_{\mathsf{rand}} + \varepsilon_{\mathsf{bind}}$.

The $\mathsf{SigEnc}$ oracle simulation fails only if there is a "collision" between a freshly generated commitment $c$ and one of the inputs $c'$ for which $G(\mathcal{L}, c')$ has already been defined. We note there are at most $q_S + q_G$ of such previously defined inputs. If $c$ were generated randomly, the probability that it would collide with a previously defined value during one of the queries to the $\mathsf{SigEnc}$ simulation would be at most $q_S \cdot (q_S + q_G) \cdot 2^{-|c|}$. Since $c$ is indistinguishable from random with all but probability $\varepsilon_{\mathsf{hide}}$ (by definition), we find the total probability that the $\mathsf{SigEnc}$ oracle simulation fails after at most $q_S$ queries to be $q_S \cdot ((q_S + q_G) \cdot 2^{-|c|} + \varepsilon_{\mathsf{hide}})$. Thus, we have now established the bound:

$$\Pr[S_1] - \Pr[S_0] \leq q_D \cdot \varepsilon_{\mathsf{rand}} + \varepsilon_{\mathsf{bind}} + q_S \cdot ((q_S + q_G) \cdot 2^{-|c|} + \varepsilon_{\mathsf{hide}}) \tag{1}$$

**Game $\mathbf{G}_2$:**

Game $\mathbf{G}_2$ is $\mathbf{G}_1$ modified to halt in the event that $\mathcal{A}_{\mathsf{CCA}}$ queries $G(\mathcal{L}, w^*)$ for any meta-data $\mathcal{L}$. We denote the event that game $i$ halts prematurely in this fashion by $\mathsf{Halt}_i$. Since $\mathbf{G}_2$ is identical to $\mathbf{G}_1$ unless $\mathsf{Halt}_2$ occurs, we have:

$$\Pr[S_2] - \Pr[S_1] \leq \Pr[\mathsf{Halt}_2] \tag{2}$$

**Game $\mathbf{G}_3$:**

Game $\mathbf{G}_3$ is the same as $\mathbf{G}_2$, but modified so that the second component of the challenge ciphertext returned to $\mathcal{A}_{\mathsf{CCA}}$, (*i.e.*, $\sigma^*$) is replaced by a random string. We note that the game halts if $\mathcal{A}_{\mathsf{CCA}}$ queries $G(\mathcal{L}, w^*)$ for any $\mathcal{L}$, so that while the game is being played $\mathcal{A}_{\mathsf{CCA}}$ learns nothing about the value of $G(\mathcal{L}^*, w^*)$. In particular, this means that if we replace $s^*$ by a random $s$, we can imagine that $G(\mathcal{L}^*, w^*)$ was defined "correctly" to be $s \oplus d^*$ and thus $\mathbf{G}_3$ has the same exact distribution as $\mathbf{G}_2$.

$$\Pr[S_3] = \Pr[S_2] \leq \Pr[\mathsf{Halt}_2] \tag{3}$$

**Game $\mathbf{G}_4$:**

Game $\mathbf{G}_4$ is $\mathbf{G}_3$ modified so that the entire challenge ciphertext $\Pi^*$ is replaced by a random string. That is, we replace $\psi^*$ by a random string, implying that $c^* = w^* = f_R^{-1}(\psi^*)$ is now a random string as well. We note that the challenge ciphertext in $\mathbf{G}_3$ does not provide $\mathcal{A}_{\mathsf{CCA}}$ with any information about $d^*$, since $\sigma^*$ was replaced by a random string. Thus, by the hiding property of commitments, the probability of distinguishing $\mathbf{G}_4$ from $\mathbf{G}_3$ is bounded by $\varepsilon_{\mathsf{hide}}$. In particular, this means that:

$$\Pr[S_4] - \Pr[S_3] \leq \varepsilon_{\mathsf{hide}} \tag{4}$$

and

$$\Pr[\mathsf{Halt}_4] - \Pr[\mathsf{Halt}_2] \leq \varepsilon_{\mathsf{hide}} \tag{5}$$

However, it is clear that since the challenge ciphertext was replaced by a random string, it no longer depends the challenge bit $b$. Thus, $\mathcal{A}_{\mathsf{CCA}}$ must guess the bit correctly with probability $1/2$ in game $\mathbf{G}_4$ when it plays to completion (*i.e.*, when it does not halt):

$$\Pr[S_4] \leq 1/2 + \Pr[\mathsf{Halt}_4] \tag{6}$$

Finally, we bound the probability that $\mathbf{G}_4$ halts by observing that the entire game can be run by a simulator with no knowledge of the secret trapdoor $f_U^{-1}$. Such a simulator can act as a reduction to the security of the underlying $\mathsf{TDP}$. The reduction accepts a $\mathsf{TDP}$ inversion challenge $y^*$ as an input, and then runs $\mathbf{G}_4$ with $\mathcal{A}_{\mathsf{CCA}}$. However, rather than replacing $\psi^*$ by an arbitrary random string, the reduction will supply $y^*$ as the first component of the challenge ciphertext. The game will halt if $\mathcal{A}_{\mathsf{CCA}}$ issues a query of the form $G(\mathcal{L}, x^*)$ such that $f_U(x^*) = y^*$. Thus, if the game halts, the reduction can then simply output the this pre-image $x^*$, a successful inversion of the $\mathsf{TDP}$. This upper bounds $\Pr[\mathsf{Halt}_4]$ by $\varepsilon_{\mathsf{TDP}}$ (the maximum probability of inverting the $\mathsf{TDP}$). Combining this bound on $\Pr[\mathsf{Halt}_4]$ with (1) - (6) gives the desired result:

$$\varepsilon_{\mathsf{CCA}} = \Pr[S_0] - 1/2 \leq \varepsilon_{\mathsf{TDP}} + 2\varepsilon_{\mathsf{hide}} + q_D \cdot \varepsilon_{\mathsf{rand}} + \varepsilon_{\mathsf{bind}} + q_S \cdot ((q_S + q_G) \cdot 2^{-|c|} + \varepsilon_{\mathsf{hide}}) \tag{7}$$

### B.2 Bounding $\varepsilon_{\mathsf{CMA}}$

To establish a bound on $\varepsilon_{\mathsf{CMA}}$, we will construct a reduction $\mathcal{B}$ that uses any $\mathcal{A}_{\mathsf{CMA}}$ breaking $\mathsf{sUF\text{-}CMA}$ security of the signcryption scheme to break the security of the underlying $\mathsf{TDP}$. The reduction will run a modified $\mathsf{sUF\text{-}CMA}$ attack game against $\mathcal{A}_{\mathsf{CMA}}$, simulating the random oracles, a $\mathsf{SigEnc}$ oracle, and a $\mathsf{VerDec}$ oracle, in similar fashion to the simulations introduced in game $\mathbf{G}_1$ used in the $\varepsilon_{\mathsf{CCA}}$ bound analysis. The reduction $\mathcal{B}$ takes an inversion challenge $y^*$ as input and finds a pre-image $x^*$ such that $y^* = f_U(x^*)$.

**RO simulation for $\mathsf{sUF\text{-}CMA}$ proof.** At the start of the simulation, a random integer $i \in \{1, \ldots, q_G\}$ is selected. The $i$-th query to the random oracle simulation is treated as a special case (in particular, we will hope that $\mathcal{A}_{\mathsf{CMA}}$ outputs a forged signcryption corresponding to the $i$-th query). The random oracle simulation handles queries of the form $G(\mathcal{L}, c)$ exactly as in the random oracle simulation for the $\mathsf{IND\text{-}CCA}$ proof, unless it happens to be the $i$-th query to the $G$ oracle. In this case, if the meta-data $\mathcal{L}$ *also* indicates that $f_U$ is the sender, the simulation defines $G(\mathcal{L}, c)$ to be $y^* \oplus \mathsf{Extract}(c, \mathcal{T})$ and returns the defined value. Notice that $y^*$ will thus correspond to $s^* = G(\mathcal{L}, c) \oplus d$ where $d$ is the extracted decommitment, so that the correct corresponding $\sigma^* = f_U^{-1}(s^*) = f_U^{-1}(y^*)$. Furthermore, the distribution of outputs for this special case is still uniformly random and independent provided that the inversion challenge $y^*$ is random. Thus the random oracle simulation remains "honest" in all cases.

$\mathsf{VerDec}$ **oracle simulation for $\mathsf{sUF\text{-}CMA}$ proof.** This oracle simulation is identical to the simulation from the $\mathsf{IND\text{-}CCA}$ proof. Notice that there is no challenge ciphertext in this game, so the oracle only outputs $\perp$ on ciphertexts for which $G(\mathcal{L}, c)$ was not queried, or $\mathsf{Open}(c, d)$ legitimately returns $\perp$.

$\mathsf{SigEnc}$ **oracle simulation for $\mathsf{sUF\text{-}CMA}$ proof.** This oracle simulation is identical to the simulation from the $\mathsf{IND\text{-}CCA}$ proof.

Let $\mathsf{SimBad}$ denote the event that one of the oracle simulations fails to return the correct response. The random oracle simulation is always correct, but the $\mathsf{VerDec}$ oracle may fail exactly as in the first failure case described in the IND-CCA proof. Similarly, the SigEnc oracle fails with the same probability as in the IND-CCA proof. The total probability of SimBad occurring is thus bounded as follows:

$$\Pr[\mathsf{SimBad}] \leq q_D \cdot \varepsilon_{\mathsf{rand}} + q_S \cdot ((q_S + q_G) \cdot 2^{-|c|} + \varepsilon_{\mathsf{hide}}) \tag{8}$$

Provided that none of the simulations fail (*i.e.*, conditioned on $\neg\mathsf{SimBad}$), the game will play out honestly and $\mathcal{A}_{\mathsf{CMA}}$ will output a valid forgery tuple $(\Pi, \ell, f_R^{-1})$ with probability $\varepsilon_{\mathsf{CMA}}$.

Let us write ForgeBad to denote the event $\mathcal{A}_{\mathsf{CMA}}$ outputs a valid $\Pi = \psi \| \sigma$ such that either (1) $\mathcal{A}_{\mathsf{CMA}}$ reused one of the $\psi$ values returned by a SigEnc oracle query, but with a different $\sigma$; or (2), the case (1) did not happen but $G(\mathcal{L}, c)$ was not first queried by $\mathcal{A}_{\mathsf{CMA}}$. In case (1) it is clear that $\mathcal{A}_{\mathsf{CMA}}$ has either broken the binding property of the commitment by reusing $c = f_R^{-1}(w)$ with a different $d = f_U^{-1}(\sigma)$, or $\mathcal{A}_{\mathsf{CMA}}$ has changed one of the public keys involved, causing the meta-data $\mathcal{L}$ to change, which in turn causes $d$ to be randomly defined. Since a randomly chosen $d$ is valid with probability at most $\varepsilon_{\mathsf{rand}}$, the total probability that $\mathcal{A}_{\mathsf{CMA}}$ can construction a such a forgery is at most $\varepsilon_{\mathsf{rand}} + \varepsilon_{\mathsf{bind}}$. For case (2) we note that since $w = c = f_R^{-1}(\psi)$ was never queried along with properly formed meta-data $\mathcal{L}$ to the $G$ oracle, the value of $G(\mathcal{L}, c)$ will be randomly defined which also implies that $d = f_U(\sigma) \oplus G(\mathcal{L}, c)$ will be randomly defined, and thus the probability that $(c, d)$ represents a valid pair is bounded by $\varepsilon_{\mathsf{rand}}$. This gives:

$$\Pr[\mathsf{ForgeBad}] \leq 2\varepsilon_{\mathsf{rand}} + \varepsilon_{\mathsf{bind}} \tag{9}$$

Conditioned on $\neg\mathsf{ForgeBad}$, a $G(\mathcal{L}, c)$ oracle query was made by $\mathcal{A}_{\mathsf{CCA}}$ corresponding to the output forgery. By definition, with probability $1/q_G$, this also happened to be the $i$-th query. In this case, if the Extract algorithm was successful, the oracle simulation responded to query with $y^* \oplus d$, where $d$ is the decommitment $\mathcal{A}_{\mathsf{CCA}}$ used in constructing the forgery. Note, the Extract algorithm fails with probability at most $\varepsilon_{\mathsf{extract}}$, by definition. Thus we have that $\sigma = f_U^{-1}(s) = f_U^{-1}(G(\mathcal{L}, c) \oplus d) = f_U^{-1}(y^*)$. Indeed, $\sigma$ is a valid pre-image for $y^*$, and the reduction $\mathcal{B}$ succeeds by returning $\sigma$. We can now establish a lower bound on $\mathcal{B}$'s success probability in terms of $\mathcal{A}_{\mathsf{CMA}}$'s success probability:

$$\begin{aligned}
\Pr[\mathcal{B} \text{ succeeds}] &\geq (\Pr[\mathcal{A} \text{ succeeds} \wedge \neg\mathsf{ForgeBad}] - \varepsilon_{\mathsf{extract}})/q_G \\
&\geq (\Pr[\mathcal{A} \text{ succeeds}] - \Pr[\mathsf{ForgeBad}] - \varepsilon_{\mathsf{extract}})/q_G \\
&\geq (\Pr[\mathcal{A} \text{ succeeds} \mid \neg\mathsf{SimBad}] - \Pr[\mathsf{SimBad}] - \Pr[\mathsf{ForgeBad}] - \varepsilon_{\mathsf{extract}})/q_G \\
&\geq (\varepsilon_{\mathsf{CMA}} - \Pr[\mathsf{SimBad}] - \Pr[\mathsf{ForgeBad}] - \varepsilon_{\mathsf{extract}})/q_G
\end{aligned} \tag{10}$$

and by definition of $(t, \varepsilon_{\mathsf{TDP}})$-secure TDPs, we also have that

$$\Pr[\mathcal{B} \text{ succeeds}] \leq \varepsilon_{\mathsf{TDP}} \tag{11}$$

Combining (8)-(11), rearranging the terms and upper bounding, we obtain the final bound on $\varepsilon_{\mathsf{CMA}}$:

$$\varepsilon_{\mathsf{CMA}} \leq q_G \cdot \varepsilon_{\mathsf{TDP}} + q_D \cdot \varepsilon_{\mathsf{rand}} + q_S \cdot ((q_S + q_G) \cdot 2^{-|c|} + \varepsilon_{\mathsf{hide}}) + 2\varepsilon_{\mathsf{rand}} + \varepsilon_{\mathsf{bind}} + \varepsilon_{\mathsf{extract}} \tag{12}$$

Observe that the runtime for the simulations (and thus the reductions corresponding to each of the games in IND-CCA proof, as well as the reduction in the sUF-CMA proof) is $\mathrm{O}((q_G + q_S) \cdot T_f)$ where $T_f$ is the time to compute a single TDP operation, and the space required is $\mathrm{O}(k \cdot (q_H + q_G + q_S))$ where $k$ is the input length of the TDPs.

### B.3 Tighter Bound on $\varepsilon_{\mathsf{CMA}}$ for Claw-Free Permutations

If $f_U$ is induced by a $(t, \varepsilon_{\mathsf{claw}})$-secure claw-free permutation. where the claw-free pair is denoted $\langle f_U, g_U \rangle$, we can construction a reduction $\mathcal{B}'$ which uses any $\mathcal{A}_{\mathsf{CMA}}$ breaking the sUF-CMA security of the signcryption to find claws for $\langle f_U, g_U \rangle$. The reduction $\mathcal{B}'$ uses a modified random oracle simulation, but is otherwise identical to the reduction $\mathcal{B}$ until $\mathcal{A}_{\mathsf{CMA}}$ has output a forgery (which will now be used to produce a claw, rather than an inversion).

**RO simulation for $\mathsf{sUF\text{-}CMA}$ proof using claw-free permutations** . The RO simulation handles queries $G(\mathcal{L}, c)$ by examining the meta-data$\mathcal{L}$ to see if $f_U$ is specified as the sender. If not, the standard honest simulation is performed. If so, the simulation performs the following procedure:

- Choose a random $x \in \{0,1\}^{|d|}$ and compute $y = g_U(x)$.
- Define $G(\mathcal{L}, c)$ to be $y \oplus \mathsf{Extract}(c, \mathcal{T})$.
- Record the pair $(y, x)$ in a lookup table.

The simulation then returns the defined value of $G(\mathcal{L}, c)$. The distribution of outputs is still uniformly random since $y$ is random. Notice that if $\mathsf{Extract}$ outputs the correct value $d$, a properly formed ciphertext $\Pi = f_R(c) \| f_U^{-1}(G(\mathcal{L}, c) \oplus d)$ will give $f_U^{-1}(y)$ as the second component. In particular, the pair $\langle f_U^{-1}(y), x \rangle$ is a claw, since $f_U(f_U^{-1}(y)) = y = g_U(x)$.

Most of the analysis proceeds as before. The events $\mathsf{SimBad}$ and $\mathsf{ForgeBad}$ are defined identically, and their analysis is identical. The only significant change is that when $\mathcal{A}_{\mathsf{CCA}}$ outputs a forgery corresponding to one of the $G(\mathcal{L}, c)$ queries, unless $\mathcal{A}_{\mathsf{CCA}}$ has broken the extractability property, it is now *always* possible for the reduction to output a valid claw, since there is no longer a single specially programmed query. In particular, provided that $\mathcal{A}_{\mathsf{CCA}}$ did not break extractability, and that $\mathcal{A}_{\mathsf{CCA}}$ output a forgery tuple $(\Pi = \psi \| \sigma, \ell, f_R^{-1})$ and $\mathsf{ForgeBad}$ did not happen, the reduction always succeeds by computing $y = f_U(\sigma)$ and finding $(y, x)$ in the lookup table, then outputting $\langle \sigma, x \rangle$ as the claw. The only subtlety is that $\mathcal{A}_{\mathsf{CCA}}$ may now break extractability for any one among the $q_G$ queries to $G$, so that we now lose $q_G \cdot \varepsilon_{\mathsf{extract}}$ rather than $\varepsilon_{\mathsf{extract}}$. The final bound for $(t, \varepsilon_{\mathsf{claw}})$-secure $\langle f_U, g_U \rangle$ is now

$$\varepsilon_{\mathsf{CMA}} \leq \varepsilon_{\mathsf{claw}} + q_D \cdot \varepsilon_{\mathsf{rand}} + q_S \cdot ((q_S + q_G) \cdot 2^{-|c|} + \varepsilon_{\mathsf{hide}}) + 2\varepsilon_{\mathsf{rand}} + \varepsilon_{\mathsf{bind}} + q_G \cdot \varepsilon_{\mathsf{extract}} \tag{13}$$

The runtime of this reduction is $\mathrm{O}((q_G + q_S) \cdot (T_f + T_{extract}))$.

# C Feistel S-Pad Proof

This proof is similar to the proof of security for Feistel **P**-Pads, and thus we will follow the syntactical conventions of that proof.

## C.1 Bounding $\varepsilon_{\mathsf{CCA}}$

We use the same approach as in the previous proof, however the precise details of the games will be different (and slightly more complicated). We begin with the new simulations required for $\mathbf{G}_1$.

**RO simulations for $\mathsf{IND\text{-}CCA}$ proof.** Oracle queries to $H$ are simulated honestly in the standard fashion.

For each query $\mathcal{A}_{\mathsf{CCA}}$ makes to $G$, of the form $G(\mathcal{L}, c)$, the simulation first checks to see if $G(\mathcal{L}, c)$ was previously defined. If so, the simulation replies with the previously defined value. Otherwise, it examines $\mathcal{L}$ to determine whether $f_U$ is specified as the recipient (if not, the oracle simulation proceeds in the standard fashion).

If $f_U$ is the recipient, the simulation parses the specified sender key $f_S$ from $\mathcal{L}$ and then performs the following steps:

- Compute $d = \mathsf{Extract}(c, \mathcal{T})$
- Choose a random $x = x_1 \| x_2 \in \{0,1\}^k$ and compute $y = y_1 \| y_2 = f_S(x)$
- Define $G(\mathcal{L}, c)$ to be $y_1 \oplus d$.
- Define $H(y_1)$ to be $y_2 \oplus c$. If $H(y_1)$ was already defined, fail.
- Compute $f_U(x)$ and record the pair $(f_U(x), x)$ in a lookup table

The simulation then returns the newly defined $G(\mathcal{L}, c)$ value. Note that if the Extract operation output the correct $d$, a properly formed ciphertext based on this $G$ oracle query would appear as $f_U(f_S^{-1}(y_1 \| y_2)) = f_U(x)$. Furthermore, the output distribution of the oracle is still uniformly random[6] for all inputs.

However, it nows takes approximately $2T_f$ to compute before responding to each query (both $f_S$ and $f_U$ must be computed).

VerDec **oracle simulation for** IND-CCA **proof.** When a query $\mathsf{VerDec}^{\ell}(\Pi, f_S)$ is issued, the simulation searches the lookup table to find a matching $\Pi$ in a pair $(\Pi, x)$. If none is found, the simulation returns $\perp$, otherwise it properly forms the corresponding $\mathcal{L}$ and returns $\mathsf{DePad}^{\mathcal{L}}(x)$. Note that if the lookup succeeds, DePad will return a correct decoding. If the lookup fails, it is possible that $\perp$ was an incorrect response, but as we will see, this occurs with negligible probability.

SigEnc **oracle simulation for** IND-CCA **proof.** When a query $\mathsf{SigEnc}^{\ell}(m, f_R)$ is issued, the simulation properly forms $\mathcal{L}$ and computes the response as follows:

- Choose a random $x = x_1 \| x_2 \in \{0,1\}^k$, and compute $y = y_1 \| y_2 = f_U(x)$.
- Compute $(c, d) \leftarrow \mathsf{Commit}(m)$.
- Define $G(\mathcal{L}, c)$ to be $y_1 \oplus d$. Fail if $G(\mathcal{L}, c)$ was already defined.
- Define $H(y_1)$ to be $y_2 \oplus c$. Fail if $H(y_1)$ was already defined.
- Return $f_R(x_1 \| x_2)$.

It is easy to verify that if the above procedure does not fail, it returns a valid response to the SigEnc query.

This completes the definitions of the oracle simulations which $\mathbf{G}_1$ uses to replace the functionality of the corresponding oracles from $\mathbf{G}_0$. Provided that these oracle simulations do not fail, $\mathbf{G}_0$ and $\mathbf{G}_1$ are identical, so $\Pr[S_1] - \Pr[S_0]$ will be bounded by the probability that a simulation failure occurs.

The $H$ oracle simulation is honest, and so never fails. However, the probability that the $G$ simulation fails is exactly the probability that the random string $y_1$ collides with one of the strings previously defined for the $H$ oracle. Since there are at most $q_H + q_G + q_S$ such strings (a single $H$ oracle query is issued for every $G$ oracle query and every SigEnc oracle query as well), the probability that any particular query fails is at most $(q_H + q_G + q_S) \cdot 2^{-|d|}$. The total probability that one of the $q_G$ oracle queries fails is at most $q_G \cdot (q_H + q_G + q_S) \cdot 2^{-|d|}$.

The VerDec oracle simulation can only fail in one of two ways: (1) the corresponding $G(\mathcal{L}, c)$ was never queried and thus there was no entry in the table; or (2) a corresponding $G(\mathcal{L}, c)$ query was made (either by

[6] Unfortunately, this is only true provided that $f_S$ is indeed a permutation. However, when considering *insider security*, it is possible that since $f_S$ is chosen by the adversary it may not be a permutation at all. In practice it may not be efficient to check this condition, but it seems unlikely that any significant attacks can result from using such a malformed signing key.

$\mathcal{A}_{\mathsf{CMA}}$ or the challenge oracle) but $\mathcal{A}_{\mathsf{CCA}}$ (or perhaps even the challenge oracle) used a $d \neq \mathsf{Extract}(c)$ to form the ciphertext. In case (1), we know that $d = G(\mathcal{L}, c) \oplus w$ where $w$ is fixed to be the first component of the pre-image in the ciphertext will be randomly defined, since $G(\mathcal{L}, c)$ is randomly defined. Thus the probability that such a ciphertext is valid is at most $\varepsilon_{\mathsf{rand}}$, by definition. Case (2) exactly corresponds to breaking the extractability property, and so it occurs with probability at most $\varepsilon_{\mathsf{extract}}$. The total probability of a failure occurring among $q_D$ queries is thus bounded by $q_D \cdot (\varepsilon_{\mathsf{rand}} + \varepsilon_{\mathsf{extract}})$.

The $\mathsf{SigEnc}$ oracle simulation can only fail if either (1) $G(\mathcal{L}, c)$ was already defined for a freshly generated $c$; or (2), the random string $y_1$ collides with one of the strings for which $H$ was previously defined. Since $c$ is indistinguishable from random with all but probability $\varepsilon_{\mathsf{hide}}$, case (1) must occur with probability at most $(q_G + q_S) \cdot 2^{-|c|} + \varepsilon_{\mathsf{hide}}$ on any single query. From the previous analysis for the RO simulation failure, we know that case (2) happens with probability at most $(q_H + q_G + q_S) \cdot 2^{-|d|}$ per query. Thus the total probability of failure after $q_S$ queries is bounded by $q_S \cdot ((q_G + q_S) \cdot 2^{-|c|} + \varepsilon_{\mathsf{hide}} + (q_H + q_G + q_S) \cdot 2^{-|d|})$.

The combined failure probability from the above analysis gives us the bound:

$$\Pr[S_1] - \Pr[S_0] \leq (q_G + q_S) \cdot (q_H + q_G + q_S) \cdot 2^{-|d|} + q_S \cdot ((q_G + q_S) \cdot 2^{-|c|} + \varepsilon_{\mathsf{hide}}) + q_D \cdot (\varepsilon_{\mathsf{hide}} + \varepsilon_{\mathsf{extract}}) \tag{14}$$

**Game $\mathbf{G}_2$:**

Game $\mathbf{G}_2$ is $\mathbf{G}_1$ modified to halt in the event that $\mathcal{A}_{\mathsf{CCA}}$ queries *both* $H(w^*)$ and $G(\mathcal{L}^*, c)$ such that $s^* = H(w^*) \oplus c$. That is, the halting condition occurs when some $w$ is queried to $H$ and some $(\mathcal{L}^*, c)$ is queried to $G$ such that $\Pi^* = f_U(f_S^{-1}(w \| H(w) \oplus c))$ (where $f_S$ is the sender specified in $\mathcal{L}^*$, the meta-data associated with the challenge oracle query). We will denote the event that game $i$ halts prematurely in this fashion by $\mathsf{Halt}_i$. Since $\mathbf{G}_2$ is identical to $\mathbf{G}_1$ unless $\mathsf{Halt}_2$ occurs:

$$\Pr[S_2] - \Pr[S_1] \leq \Pr[\mathsf{Halt}_2] \tag{15}$$

**Game $\mathbf{G}_3$:**

Game $\mathbf{G}_3$ is the same as $\mathbf{G}_2$, but modified so that the $w^*$ component used in producing the challenge ciphertext return $\mathcal{A}_{\mathsf{CCA}}$ is replaced by a random string $\hat{w}$. That is, we now have $s^* = H(\hat{w}) \oplus c^*$ and $\Pi^* = f_U(f_S^{-1}(\hat{w} \| s^*))$. We will show that $\mathbf{G}_3$ cannot be distinguished from $\mathbf{G}_2$ with better than negligible probability by arguing that, with high probability, it must halt before $\mathcal{A}_{\mathsf{CCA}}$ learns sufficient information to distinguish it. In particular, note that unless $G(\mathcal{L}^*, c^*)$ has been queried by $\mathcal{A}_{\mathsf{CCA}}$, from $\mathcal{A}_{\mathsf{CCA}}$'s point of view we may imagine that $G(\mathcal{L}^*, c^*)$ was correctly defined to be $d^* \oplus \hat{w}$. Thus, in order to distinguish, $\mathcal{A}_{\mathsf{CCA}}$ must first query $G(\mathcal{L}^*, c^*)$. However, if $\mathcal{A}_{\mathsf{CCA}}$ has queried $H(\hat{w})$, the game will halt at this point, so we need only consider the probability of $\mathcal{A}_{\mathsf{CCA}}$ querying $G(\mathcal{L}^*, c^*)$ without first querying $H(\hat{w})$. Since $H(\hat{w})$ has not yet been queried, $\mathcal{A}_{\mathsf{CCA}}$ has been given no information regarding $c^* = H(\hat{w}) \oplus s^*$. Furthermore, $\mathcal{A}_{\mathsf{CCA}}$ has no information regarding $d^*$, since $d^*$ no longer appears in the computation of $\Pi^*$. Thus, $c^*$ appears random to $\mathcal{A}_{\mathsf{CCA}}$ with all probability $\varepsilon_{\mathsf{hide}}$, so the probability that $\mathcal{A}_{\mathsf{CCA}}$ queries $G(\mathcal{L}^*, c^*)$ without halting the game is bounded by $q_G \cdot (2^{-|c|} + \varepsilon_{\mathsf{hide}})$. As we have already argued that this game is indistinguishable from $\mathbf{G}_2$ unless $\mathcal{A}_{\mathsf{CCA}}$ queries $G(\mathcal{L}^*, c^*)$, we arrive at the bound:

$$\Pr[S_3] - \Pr[S_2] \leq q_G \cdot (2^{-|c|} + \varepsilon_{\mathsf{hide}}) \tag{16}$$

Furthermore, note that the same bound applies to the difference $\Pr[\mathsf{Halt}_3] - \Pr[\mathsf{Halt}_2]$, since a distinguisher could be built based on the halting event:

$$\Pr[\mathsf{Halt}_3] - \Pr[\mathsf{Halt}_2] \leq q_G \cdot (2^{-|c|} + \varepsilon_{\mathsf{hide}}) \tag{17}$$

**Game $\mathbf{G}_4$:**

Game $\mathbf{G}_4$ is $\mathbf{G}_3$ modified so that the entire challenge ciphertext $\Pi^*$ is replaced by a random string. We may view this as replacing $s^*$ by a random string $\hat{s}$ and computing the ciphertext $\Pi^* = f_U(f_S^{-1}(\hat{w}\|\hat{s}))$. In particular, note that in $\mathbf{G}_3$, $s^* = H(\hat{w}) \oplus c^*$, and that no information about $d^*$ is available to $\mathcal{A}_{\mathsf{CCA}}$. In going from $\mathbf{G}_3$ to $\mathbf{G}_4$ we may simply replace $c^*$ by a random string, giving us the identical distribution (*i.e.*, it is the same as sampling a random $\hat{s}$ directly). Thus, by the hiding property of commitments, the probability of distinguishing $\mathbf{G}_4$ from $\mathbf{G}_3$ is bounded by $\varepsilon_{\mathsf{hide}}$. In particular, we have:

$$\Pr[S_4] - \Pr[S_3] \leq \varepsilon_{\mathsf{hide}} \tag{18}$$

and, similarly for the halting event,

$$\Pr[\mathsf{Halt}_4] - \Pr[\mathsf{Halt}_3] \leq \varepsilon_{\mathsf{hide}} \tag{19}$$

However, it is clear that since the challenge ciphertext was replaced by a random string, it is independent of the challenge bit $b$. Thus $\mathcal{A}_{\mathsf{CCA}}$ must guess the bit correctly with probability $1/2$ in game $\mathbf{G}_4$ when it plays to completion (*i.e.*, when it does not halt):

$$\Pr[S_4] \leq 1/2 + \Pr[\mathsf{Halt}_4] \tag{20}$$

Finally, we bound the probability that $\mathbf{G}_4$ halts by observing that the entire game can be run by a simulator with no knowledge of the secret trapdoor $f_U^{-1}$. Such a simulator can act as a reduction to the security of the underlying $\mathsf{TDP}$. The reduction accepts a $\mathsf{TDP}$ inversion challenge $y^*$ as an input, and then runs $\mathbf{G}_4$ with $\mathcal{A}_{\mathsf{CCA}}$. However, rather than replacing the challenge ciphertext $\Pi^*$ by an arbitrary random string, it replaces the challenge ciphertext with $y^*$. The reduction tests for the halting condition by examining all pairs of queries of the form $H(w)$ and $G(\mathcal{L}^*, c)$ to see when the condition $\Pi^* = f_U(f_S^{-1}(w\|H(w) \oplus c)$ is satisfied. Note that the reduction can indeed compute $f_S^{-1}$ since the trapdoor for $S$ must be provided by $\mathcal{A}_{\mathsf{CCA}}$ when requesting a challenge ciphertext. If the halting condition is satisfied, the reduction simply outputs $f_S^{-1}(w\|H(w) \oplus c)$ as a valid pre-image for $\Pi^* = y^*$, thus successfully inverting the $\mathsf{TDP}$ $f_U$. Since this reduction cannot succeed with probability greater than $\varepsilon_{\mathsf{TDP}}$, we conclude that $\Pr[\mathsf{Halt}_4] \leq \varepsilon_{\mathsf{TDP}}$. Combining this result with (14) - (20) gives the desired result:

$$\begin{aligned}
\varepsilon_{\mathsf{CCA}} &= \Pr[S_0] - 1/2 \\
&\leq \varepsilon_{\mathsf{TDP}} + (q_G + q_S) \cdot (q_H + q_G + q_S) \cdot 2^{-|d|} + q_S \cdot ((q_G + q_S) \cdot 2^{-|c|} + \varepsilon_{\mathsf{hide}}) \\
&\quad + q_D \cdot (\varepsilon_{\mathsf{hide}} + \varepsilon_{\mathsf{extract}}) + 2q_G \cdot (2^{-|c|} + \varepsilon_{\mathsf{hide}}) + 2\varepsilon_{\mathsf{hide}}
\end{aligned} \tag{21}$$

Observe that the worst case runtime above occurs in the reductions for $\mathbf{G}_2$ and above, where we must detect the halting condition. Detecting the halting condition requires computing $\mathsf{TDP}$s on all pairs of queries to $G$ and $H$. Otherwise, the runtime is completely determined by the oracle simulation runtimes (and thus the reductions corresponding to each of the above games). Thus the total runtime is $\mathrm{O}((q_G + q_S + q_H \cdot q_G) \cdot (T_f + T_{extract}))$. The space required is $\mathrm{O}(k \cdot (q_H + q_G + q_S))$.

## C.2 Bounding $\varepsilon_{\mathsf{CMA}}$

To establish a bound $\varepsilon_{\mathsf{CMA}}$, we construct a reduction $\mathcal{B}$ as in the proof of $\mathsf{sUF\text{-}CMA}$ security for **P**-Pads. The reduction operates in completely analogous fashion, and we begin with a description of the oracle simulations. For convenience, we will write the inversion challenge to the reduction as $y^* = y_1^*\|y_2^*$.

**RO simulation for $\mathsf{sUF\text{-}CMA}$ proof.** At the start of the simulation, a random integer $i \in \{1, \ldots, q_G\}$ is selected. The $i$-th query to the $G$ simulation is treated as a special case (in particular, we will hope that $\mathcal{A}_{\mathsf{CMA}}$ outputs a forged signcryption corresponding to the $i$-th query). The random oracle simulation handles queries of the form $G(\mathcal{L}, c)$ and $H(w)$ exactly as in the random oracle simulation for the $\mathsf{IND\text{-}CCA}$ proof, unless it happens to be $i$-th query to the $G$ oracle. In this case, if the meta-data $\mathcal{L}$ *also* indicates that $f_U$ is the sender, the simulation $G(\mathcal{L}, c)$ to be $y_1^* \oplus \mathsf{Extract}(c, \mathcal{T})$. Furthermore, the simulation defines $H(y_1^*)$ to be $y_2^* \oplus c$. If $H(y_1^*)$ was already defined, the simulation fails. If this failure does not occur, the simulation remains perfectly honest since both of these definitions are uniformly random provided $y^*$ is actually a random challenge. Notice that $w\|s = G(\mathcal{L}, c) \oplus d \| H(G(\mathcal{L}, c) \oplus d) \oplus c$ will equal $y^*$ provided that $\mathsf{Extract}(c, \mathcal{T})$ returns the correct value $d$.

$\mathsf{VerDec}$ **oracle simulation for $\mathsf{sUF\text{-}CMA}$ proof.** The $\mathsf{VerDec}$ oracle simulation is identical to the one introduced in game $\mathbf{G}_1$ of the $\mathsf{IND\text{-}CCA}$ analysis.

$\mathsf{SigEnc}$ **oracle simulation for $\mathsf{sUF\text{-}CMA}$ proof.** The $\mathsf{SigEnc}$ oracle simulation is identical to the one introduced in game $\mathbf{G}_1$ of the $\mathsf{IND\text{-}CCA}$ analysis.

Let $\mathsf{SimBad}$ denote the event that one of the oracle simulations fails. The RO simulation fails in the same fashion as in the $\mathsf{IND\text{-}CCA}$ proof, but now may additionally fail when if value of $H(y_1^*)$ was already defined. This clearly happens with probability at most $(q_H + q_G + q_S) \cdot 2^{-|d|}$, since there are at most $q_H + q_G + q_S$ defined inputs to $H$. The $\mathsf{VerDec}$ and $\mathsf{SigEnc}$ simulations are the same as in the $\mathsf{IND\text{-}CCA}$ proof, and thus fail with the same probability. Thus,

$$\Pr[\mathsf{SimBad}] \leq (q_G+q_S+1)\cdot(q_H+q_G+q_S)\cdot 2^{-|d|}+q_S\cdot((q_G+q_S)\cdot 2^{-|c|}+\varepsilon_{\mathsf{hide}})+q_D\cdot(\varepsilon_{\mathsf{hide}}+\varepsilon_{\mathsf{extract}}) \tag{22}$$

Provided that none of the simulations fail (*i.e.*, conditioned on $\neg\mathsf{SimBad}$), the game plays out honestly and $\mathcal{A}_{\mathsf{CMA}}$ will output a valid forgery tuple $(\Pi, \ell, f_R^{-1})$ with probability $\varepsilon_{\mathsf{CMA}}$.

Let us write $\mathsf{ForgeBad}$ to denote the event that $\mathcal{A}_{\mathsf{CMA}}$ outputs a valid $\Pi = f_R(f_U^{-1}(w\|s))$ such that either (1) $\mathcal{A}_{\mathsf{CMA}}$ reused an $(\mathcal{L}, c)$ pair that was queried by the $\mathsf{SigEnc}$ oracle simulation, or (2), the case (1) did not happen but $\mathcal{A}_{\mathsf{CMA}}$ has not queried $G(\mathcal{L}, c)$ for the appropriate $\mathcal{L}$. Clearly, case (1) can only occur if $\mathcal{A}_{\mathsf{CMA}}$ found a $d' \neq d$ such that $(c, d')$ is a valid pair (since otherwise $\mathcal{A}_{\mathsf{CMA}}$ is simply copying the output of a $\mathsf{SigEnc}$ query which is not an allowable forgery). That is, case (1) corresponds to breaking the binding property of the commitment, and therefore cannot occur with more than probability $\varepsilon_{\mathsf{bind}}$. In case (2) note that $G(\mathcal{L}, c)$ is randomly since it was never queried. This implies that $d = w \oplus G(\mathcal{L}, c)$ is randomly defined, and thus $\Pi$ represents a valid forgery with probability at most $\varepsilon_{\mathsf{rand}}$. We can now upper bound $\mathsf{ForgeBad}$:

$$\Pr[\mathsf{ForgeBad}] \leq \varepsilon_{\mathsf{rand}} + \varepsilon_{\mathsf{bind}} \tag{23}$$

Conditioned on $\neg\mathsf{ForgeBad}$, a $G(\mathcal{L}, c)$ oracle query was issued by $\mathcal{A}_{\mathsf{CCA}}$ corresponding to the $(\mathcal{L}, c)$ used in the output forgery. By definition, with probability $1/q_G$, this also happened to be the $i$-th query. In this case, if the $\mathsf{Extract}$ algorithm was successful (i.e. it obtained the correct $d$ used by $\mathcal{A}_{\mathsf{CCA}}$ in the forgery), the oracle simulation will cause $w\|s$ used in the forgery to equal $y^*$, as explained above. Note that the probability that $\mathsf{Extract}$ fails is at most $\varepsilon_{\mathsf{extract}}$, by definition. Furthermore, if it does not fail, given a forgery tuple $(\Pi, \ell, f_R^{-1})$, the reduction can compute $f_R^{-1}(\Pi) = f_R^{-1}(f_R(f_U^{-1}(w\|s)) = f_U^{-1}(w\|s) = f_U^{-1}(y^*)$. Thus if the reduction outputs $f_R^{-1}(\Pi)$ it will successfully invert the $\mathsf{TDP}$, an event whose probability is bounded by $\varepsilon_{\mathsf{TDP}}$. The analysis proceeds completely analogously to that for **P**-Pads, and carrying out that analysis, replacing (8) with (22) and (9) with (23), we obtain:

$$\begin{aligned} \varepsilon_{\mathsf{CMA}} \;\leq\; & q_G \cdot \varepsilon_{\mathsf{TDP}} + (q_G + q_S + 1) \cdot (q_H + q_G + q_S) \cdot 2^{-|d|} + q_S \cdot ((q_G + q_S) \cdot 2^{-|c|} + \varepsilon_{\mathsf{hide}}) \\ & + q_D \cdot (\varepsilon_{\mathsf{hide}} + \varepsilon_{\mathsf{extract}}) + \varepsilon_{\mathsf{rand}} + \varepsilon_{\mathsf{bind}} + \varepsilon_{\mathsf{extract}} \end{aligned} \tag{24}$$

The runtime of this reduction is $\mathrm{O}((q_G + q_S) \cdot (T_f + T_{extract}))$, and the space requirement the same as for the $\mathsf{IND\text{-}CCA}$ proof up to constant factors.

### C.3 Tighter Bound on $\varepsilon_{\mathsf{CMA}}$ for Claw-Free Permutations

This proof is analogous to the proof in Section B.3, and we will use the same notation. We begin by defining the new RO simulation.

**RO simulation for sUF-CMA proof using claw-free permutations.** The RO simulation is similar to the one for the previous sUF-CMA security proof in Section C.2. However, queries of the form $G(\mathcal{L}, c)$ where $f_U$ is specified as the sender will *all* be handled by the following procedure (rather than singling out the $i$-th query):

- Choose a random $x \in \{0,1\}^k$ and compute $y = y_1 \| y_2 = g_U(x)$.
- Define $G(\mathcal{L}, c)$ to be $y_1 \oplus \mathsf{Extract}(c, \mathcal{T})$.
- Define $H(y_1)$ to be $y_2 \oplus c$. If $H(y_1)$ was already defined, fail.
- Record the pair $(y, x)$ in a lookup table.

This output distribution of the oracle simulation is clearly honest, and it fails with at most $q_G$ times the probability that oracle simulation in Section B.3 fails (since it may now fail on every query, rather than just the $i$-th). Notice that if the Extract algorithm correctly computed $d$, and $\mathcal{A}_{\mathsf{CCA}}$ outputs a forgery $\Pi = f_R(f_U^{-1}(w\|s))$ corresponding to this $G$ oracle query, it is easily verified that $y = f_U(w\|s)$, and thus $\langle w\|s, x\rangle$ will form a claw.

The analysis proceeds as in Section B.3, but due to the increased probability that the RO simulation fails, we now have

$$\Pr[\mathsf{SimBad}] \leq (2q_G+q_S)\cdot(q_H+q_G+q_S)\cdot 2^{-|d|} + q_S\cdot((q_G+q_S)\cdot 2^{-|c|}+\varepsilon_{\mathsf{hide}})+q_D\cdot(\varepsilon_{\mathsf{hide}}+\varepsilon_{\mathsf{extract}}) \tag{25}$$

The remainder of the analysis carried out as in Section B.3, and we obtain:

$$\begin{aligned}\varepsilon_{\mathsf{CMA}} \leq\ & \varepsilon_{\mathsf{claw}} + (q_G+q_S+1)\cdot(q_H+q_G+q_S)\cdot 2^{-|d|} + q_S\cdot((q_G+q_S)\cdot 2^{-|c|}+\varepsilon_{\mathsf{hide}}) \\ & + q_D\cdot(\varepsilon_{\mathsf{hide}}+\varepsilon_{\mathsf{extract}}) + \varepsilon_{\mathsf{rand}} + \varepsilon_{\mathsf{bind}} + q_G\cdot\varepsilon_{\mathsf{extract}}\end{aligned} \tag{26}$$

The runtime of this reduction is $\mathrm{O}((q_G+q_S)\cdot(T_f+T_{extract}))$, and the space requirement the same as for the IND-CCA proof up to constant factors.

### C.4 Improved Exact Security Bound for X-Pads

We note that a TDP $f'_U(w\|s)$ may be constructed from any TDP $f_U(w)$ by simply setting $f'_U(w\|s) = f_U(w)\|s$. This fact is sufficient to allow us to apply the proofs of security for **S**-Pads directly to **X**-Pads, since we may view the TDPs as being constructed in this fashion. However, we notice that the $s$ portion of the padding is now outside the TDP, so that it is no longer necessary for the reduction in the IND-CCA proof to compute $f_U$ for *all pairs* of queries to $G$ and $H$, since the query to $G$ will completely determine the $w$ component that is placed inside the actual TDP $f_U$. Thus the halting condition can now be determined in time proportional to $q_G\cdot T_f$, and the running time of the reduction is reduced to $\mathrm{O}((q_G+q_S+q_H)\cdot(T_f+T_{extract}))$ as claimed.

## D Proof of Lemma 3

We briefly argue the scheme's exact security bounds for $\varepsilon_{\mathsf{hide}}$ and $\varepsilon_{\mathsf{extract}}$.To break hiding, an adversary $\mathcal{A}$ must differentiate $c$ from some random value $R \leftarrow \{0,1\}^{|c|}$, given the fixed $m$. It is easy to see that this can happen only if $\mathcal{A}$ queries $K(r)$ or $K'(m_2\|r)$. Since $r$ is a random string of length $|d|-|m_2|$,

$$\varepsilon_{\mathsf{hide}} \leq (q_K+q_{K'})\cdot 2^{-(|d|-|m_2|)}$$

To break extractability, the adversary finds some $\langle d', c\rangle$, where $d' = m_2' \| r'$, and one of two cases occur. In the first case, $m_2' \| r'$ was not queried to $K'$. In the second, the adversary finds some $d' \neq d$ that represents a birthday attack on $K'$, *i.e.*, finds some $K'(m_2' \| r') = K'(m_2 \| r)$ where the output length of the $K'$ oracle is $|c| - |m_1|$. Upper-bounding the probability of both events in the obvious way, we get the following:

$$\varepsilon_{\mathsf{extract}} \;\leq\; 2^{-(|c|-|m_1|)} \;+\; q_{K'}(q_{K'}-1) \cdot 2^{-(|c|-|m_1|+1)} \;<\; (q_{K'}^2+1) \cdot 2^{-(|c|-|m_1|)}$$

To show the bound on $\varepsilon_{\mathsf{rand}}$, consider that (for fixed $c$) a random $d$ will be valid if and only if $K'(m_2 \| r) = K'(m_2' \| r')$ where $m_2'$ and $r_2'$ are randomly defined by $d$. Since $K'$ is a random oracle with output length $|c| - |m_1|$, this happens with probability $2^{-|K'(\cdot)|} = 2^{-(|c|-|m_1|)}$.

# E Proof of Theorem 4 (Signcryption of Long Messages)

**Proof:** The $\mathsf{sUF\text{-}CMA}$ security bound is automatic, since the notion of a forgery for signcryption with associated data encompasses the entire signcryptext, including the label. In other words, consider a reduction $\mathcal{B}$ against the $\mathsf{sUF\text{-}CMA}$ security of $\mathcal{SC}$ that uses any $\mathcal{A}$ that breaks the $\mathsf{sUF\text{-}CMA}$ security of $\mathcal{SC}'$. $\mathcal{B}$ simply answers $\mathcal{A}$'s signcryption queries $\mathsf{SigEnc}'^{\ell}(M, \mathsf{VEK})$ by selecting at random a $\tau$, and then returning $\mathsf{SigEnc}^{\ell \| E_\tau(M)}(\tau, \mathsf{VEK})$. $\mathcal{B}$ uses the obvious corresponding approach for $\mathsf{VerDec}'$ queries. Clearly, any signcryptext $\mathcal{A}$ forges against $\mathcal{SC}'$ is also a valid forgery against $\mathcal{SC}$, and thus the reduction succeeds with the same probability as $\mathcal{A}$ by simply returning $\mathcal{A}$'s forgery.

The $\mathsf{IND\text{-}CCA}$ security reduction is also as described above, and the security bound follows from a simple two-step hybrid argument.

(1) We modify the original $\mathsf{IND\text{-}CCA}$ game by replacing the $E_\tau$ operation during the construction of the challenge ciphertext with $E_{\tilde{\tau}}$, where $\tilde{\tau}$ is a random key independent of the signcrypted key $\tau$. Any adversary capable of telling this game apart from the original game can be used to win the $\mathsf{IND\text{-}CCA}$ game against the underlying signcryption scheme $\mathcal{SC}$ with at least the same advantage. It does this by simply using the label $E_\tau(M_b)$ (where $b \leftarrow \{0,1\}$) and providing $m_0 = \tau$ and $m_1 = \tilde{\tau}$ as the messages it claims to distinguish against $\mathcal{SC}$ in the $\mathsf{IND\text{-}CCA}$ attack (it also uses the same oracle simulations as $\mathcal{B}$). Thus, in this step, the advantage of $\mathcal{B}$ is reduced by at most $\varepsilon_{\mathsf{CCA}}$.

(2) We replace $E_{\tilde{\tau}}(M)$ in the challenge ciphertext by $E_{\tilde{\tau}}(\tilde{M})$, where $\tilde{M}$ is a random message. Any adversary capable of differentiating this game from the game of Step 1 can be used to break the security of the one-time encryption with at least the same advantage. (In a fashion similar to Step 1, we can use $\mathcal{SC}$ to signcrypt a random string with either $E_{\tilde{\tau}}(M)$ or $E_{\tilde{\tau}}(\tilde{M})$ as the label and use $\mathcal{A}$ to distinguish the resulting ciphertexts.) Thus, in going to this final step, the advantage of $\mathcal{B}$ is further reduced by at most $\varepsilon_{\mathsf{OTE}}$.

We note that, in the final step, $\mathcal{B}$ cannot have any advantage over guessing, since the challenge ciphertext is random and independent of the challenge messages. Therefore, by this hybrid argument, $\mathcal{B}$ has a total advantage at most $\varepsilon_{\mathsf{CCA}} + \varepsilon_{\mathsf{OTE}}$ in the original game, and the proof is complete. $\square$