Versatile Padding Schemes for Joint Signature and Encryption

Yevgeniy Dodis
New York University
dodis@cs.nyu.edu

Michael J. Freedman
New York University
mfreed@cs.nyu.edu

Stanislaw Jarecki
University of California, Irvine
stasio@ics.uci.edu

Shabsi Walfish
New York University
walfish@cs.nyu.edu

ABSTRACT

We propose several highly-practical and optimized constructions for joint signature and encryption primitives often referred to as *signcryption*. All our signcryption schemes, built directly from trapdoor permutations such as RSA, 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 and, finally, complete compatibility with the PKCS#1 infrastructure.

Similar to the design of plain RSA-based signature and encryption schemes, such as RSA-FDH and RSA-OAEP, our signcryption schemes are constructed by designing appropriate *padding schemes* suitable for use with trapdoor permutations. We build a general and flexible *framework* for the design and analysis of secure *Feistel-based* padding schemes, as well as three composition paradigms for using such paddings to build optimized signcryption schemes. To unify many secure padding options offered as special cases of our framework, we construct a single *versatile* padding scheme PSEP which, by simply adjusting the parameters, can work optimally with any of the three composition paradigms for either signature, encryption, or signcryption.

We illustrate the utility of our signcryption schemes by applying them to build a secure key-exchange protocol, with performance results showing 3x–5x speed-up compared to standard protocols.

Categories and Subject Descriptors

E.3 [**Data Encryption**]: [Public key cryptosystems]

General Terms

Security, Theory

Keywords

Signcryption, joint signature and encryption, universal padding schemes, Feistel Transform, extractable commitments

CCS'04, October 25-29, 2004, Washington, DC, USA.

1. INTRODUCTION

Until recently, the two main building-blocks of modern public-key cryptography — encryption and signature schemes — have always been considered as *separate* entities. From a design and analysis standpoint, this evolution makes sense as encryption and signatures serve fundamentally different purposes. However, many centrally-important applications use both primitives to ensure message privacy *and* authentication at the same time. Secure email, one of earliest applications of public-key cryptography, requires the application of both primitives. Encryption-based key exchange [25] does so as well. The security requirements of ever-greater numbers of distributed applications, services, and devices place increasing importance on both primitives. Yet, the task of secure and efficient composition of the two primitives is typically left to the application programmers, which has often led to increased risk of security breaches at the application layer [7].

Rather than leaving this composition task to ad-hoc attempts, investing effort into designing a "joint signature and encryption" *primitive* has the potential benefit of creating a cryptographic tool that optimizes the efficiency of applications that use it, yet still provides strong and well-understood security properties.

Our Motivation. To provide such a "joint signature and encryption" tool, Zheng [27] introduced *signcryption* as a primitive, and several subsequent papers provided alternate constructions and improvements [28, 21, 14, 2, 1, 20]. However, these existing signcryption proposals all leave something to be desired in the security properties they achieve. All proposed schemes, with the exception of the generic composition method of [1], are not known to offer *insider* security for both sender and recipient [1, 2], which means that an attacker compromising the sender/recipient can violate the privacy/authenticity of the recipient/sender. They also are either not known to be secure in the multi-user setting [20], or are based on non-standard assumptions [27]. Finally, all have suboptimal message bandwidth or poor exact security bounds.

Moreover, the existing signcryption proposals do not adequately address several important practical concerns that emerge in applications of joint signature and encryption. For example, one might want to re-use the same public key for signing, encryption, *and* for the new signcryption operation in practice, to simplify key management. Additionally, practitioners often need the flexibility to encrypt only portions of a message, yet still sign the entire message (such signed plaintext is commonly known as *associated data*). It is often unclear whether previous schemes support such properties efficiently and securely. Finally, some schemes [27] require all parties to agree on the same public parameters, such as the common discrete log group, which makes any changes to the security parameter or signcryption scheme quite difficult.

On the other hand, we observe that practical signature and encryption schemes such as OAEP [4], OAEP+ [26], and PSS-R [5], are built from trapdoor permutations (TDPs) such as RSA, and are analyzed in the random oracle (RO) model. We call such schemes TDP-based.[1] Although some TDP-based signcryption schemes are known [20] or implied [1], it is natural to ask whether we can build *optimized* signcryption constructions from TDPs.

Overview of Our Results. 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" 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 [24]. 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 (see Table 1 and Section 8).

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 (as opposed to separate signature and encryption keys, as in [1]). Similar to TDP-based signature and encryption schemes, our schemes use some *padding scheme* Pad on message m before passing the result through the corresponding TDPs. However, our schemes use only a *single*, general purpose padding scheme, rather than two independent padding schemes [1]. This design (1) results in noticeable practical savings in both quantitative and qualitative security, (2) improves the message bandwidth and randomness utilization, and (3) simplifies protocol design and implementation.

Table 1 compares our signcryption construction against several earlier proposals, with regards to several properties. Section 8 describes these alternate schemes in more depth, and we include the comparison here for easy reference. Note that we do not claim any improvement in the *computational* efficiency of signcryption based on TDPs (*e.g.*, compared to [1, 20]), since the computational overhead is dominated by the time required to compute and invert TDPs. However, we improve upon existing TDP based signcryption schemes in other ways, as shown in Table 1.

More specifically, we offer three options for a sender S to transmit a message m to receiver R: **P**-Pad (Parallel Padding), **S**-Pad (Sequential Padding), and **X**-Pad (eXtended sequential Padding). The convenience of each padding scheme depends on the application for which it is used. For example, **P**-Pad provides parallel application of "signing" f_S^{-1} and "encrypting" f_R (with optimal exact security), while **S**-Pad permits a shorter minimal ciphertext length by losing parallelism (and some exact security), and **X**-Pad regains optimal exact security by slightly increasing the minimum ciphertext length. We note that the minor trade-offs represented by each of these three options appear to be necessary.

Our Generalized Padding Constructions. We observe that all popular padding schemes with message recovery currently used for ordinary signature or encryption, such as OAEP [3], OAEP+ [26], OAEP++ [17], PSS-R [5], and "scramble all, encrypt small" [15] (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 [19]—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 we obtain the desired padding scheme by applying one or two rounds of the Feistel Transform to *any* such $\langle d, c\rangle$.

We show that the needed conditions on $\langle d, c\rangle$ are that they form an *extractable commitment scheme* (see Section 2), which is a trivial condition to check and satisfy in the RO model. For example, setting $c = H(m\|r)$, $d = (m\|r)$, we get a commitment scheme which defines PSS-R, while setting $c = H(r) \oplus (m\| 0^\lambda)$, $d = r$, 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.* enable signcryption constructions, but: (1) we get several of the previous results about signature and encryption as one special case of our general framework; (2) we isolate and abstract the usefulness of the Feistel Transform in constructing TDP-based schemes; and (3) we derive *new* padding schemes (without needing new proofs!) which can be specially tailored for particular situations.

As an example of the last benefit, we construct a single, versatile padding scheme, called *Probabilistic Signature-Encryption Padding* (PSEP). PSEP with a single Feistel round is a "hybrid" of the standard PSS-R and OAEP paddings, yet also offers optimal message bandwidth in our setting. With a second Feistel round, PSEP is a versatile padding scheme capable of achieving optimal bandwidth in *all* of our constructions, even when used for plain signature or encryption applications (*i.e.*, it is also a "universal padding" [6]). Thus, our suggested padding scheme is truly applicable for any TDP-based public-key usage.

Extensions and Applications. Our signcryption scheme with support for associated data can be easily extended to support long messages using the technique described in Section 6 (see also [23]), while retaining all the benefits of the original scheme. In Section 7, we use our signcryption scheme to build a simple two-round authenticated key-exchange (AKE) protocol, based on any TDP such as RSA. In addition to reducing round complexity, our implementation results suggest a 3x–5x speed-up when compared to standard key-exchange protocols offering comparable security guarantees.

Organization. The rest of this paper is structured as follows. Section 2 introduces our three padding constructions, Section 3 applies these constructions to build efficient signcryption scheme, and Section 4 shows their usefulness to plain encryption and signature. Section 5 uses these constructions to build the new PSEP padding scheme. In Section 6, we show how these schemes can easily be extended to support long messages, while Section 7 presents the signcryption-based key-exchange protocol. Section 8 discusses related work, and Section 9 concludes.

2. PADDING CONSTRUCTIONS

In this section, we construct tailored padding schemes with which one can apply a combination of TDPs to a *single* padded message to achieve encryption, signature, or both in the form of a signcryption primitive. 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 key. Similar to TDP-based signature and encryption schemes, our schemes use some *padding scheme* Pad on mes-

[1] We only consider protocols in the RO model. However, as 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.

	ZSCR [2]	TBOS [20]	CtE&S / $\mathcal{S}$tE / EtS [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**
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	$2k + \lvert m\rvert$	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.

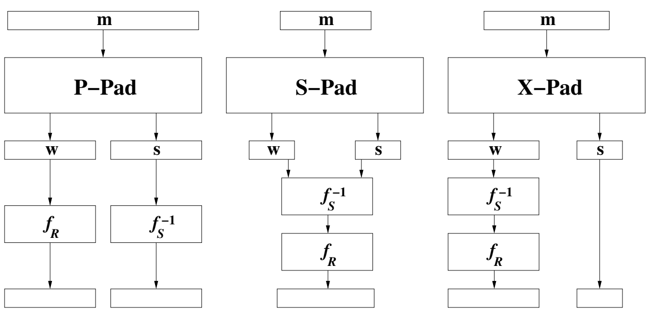

Figure 1: Generalized paddings as used by signcryption

Padding Type	Encryption	Signature	Signcryption
Parallel	$f_R(w)\|s$	$w\|f_S^{-1}(s)$	$f_R(w)\|f_S^{-1}(s)$
Sequential	$f_R(w\|s)$	$f_S^{-1}(w\|s)$	$f_R(f_S^{-1}(w\|s))$
e**X**tended sequential	$f_R(w)\|s$	$f_S^{-1}(w)\|s$	$f_R(f_S^{-1}(w))\|s$

Table 2: Proposed TDP-based padding schemes

sage m before passing the result through corresponding TDP(s). However, our scheme allows one to use only a *single*, specialized padding scheme for any of the three primitives. This design (1) unifies the design of cryptographic padding schemes into a single general construction, (2) simplifies protocol design and implementation, including the ability to trivially prove tight security bounds on new or existing schemes, and (3) in the case of signcryption, results in noticeable improvements in both quantitative and qualitative security, as well as optimizes message bandwidth and randomness utilization.

Specifically, to send a short message[2] m from S to R, we offer three options to S. The convenience of each padding scheme depends on the application for which it is used. Each padding scheme produces $\mathsf{Pad}(m) = w\|s$, used as shown in Table 2.

Figure 1 illustrates the use of such padding schemes in for signcryption. As we can see, **P**-Pad signcryption provides parallel application of "signing" f_S^{-1} and "encrypting" f_R, which can result in efficiency improvements on parallel machines. However, the minimum ciphertext length is twice as large as compared to **S**-Pad, yet 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).

In this section, we first discuss our construction framework and its use of Feistel Transforms and *extractable commitments*, before presenting the above padding schemes.

2.1 Cryptographic Components

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 $\langle d, c\rangle$. In fact, in all our constructions we will use any extractable commitment pair (described next) 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

[2] Section 6 easily extends our scheme to support long messages.

($\mathsf{VEK}_S, \mathsf{VEK}_R$, respectively), as well as any desired associated data ℓ. This use of meta-data will become more apparent when we introduce signcryption in Section 3. For simplicity, we use $\hat{G}(\cdot)$ to denote $G(\mathcal{L}, \cdot)$, where one can view $\hat{G}$ as an RO uniquely determined by $\mathcal{L}$. Using $\hat{G}$ as our round function, rather than G, "binds" the padded message to the meta-data, preventing identity fraud and ensuring the integrity of the associated data ℓ.

Extractable Commitments. Our constructions for padding schemes all make use of *extractable commitment schemes*. Such commitments have the usual properties of standard commitments, but with the additional twist that there exists an extraction algorithm which can extract a unique decommitment from any valid commitment with high probability, by using some "trapdoor information". In the random oracle model, such information is provided by a transcript of all random oracle queries.

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, which we informally describe here. See Appendix A for formal definitions and proofs.

- **Hiding.** No PPT adversary running in time t can distinguish the commitment of any messages of its choice from a k-bit random string R with probability greater than $\varepsilon_{\mathsf{hide}}$.

- **Extractability.** There exists a deterministic poly-time algorithm Extract which can extract the "correct" decommitment from any valid commitment in time t—failing with probability at most $\varepsilon_{\mathsf{extract}}$—given access to a transcript $\mathcal{T}$ of all RO queries previously issued by the adversary.

A commitment scheme $\mathcal{C}$ is a $(t, \varepsilon_{\mathsf{hide}}, \varepsilon_{\mathsf{extract}})$-secure extractable commitment if it satisfies the above properties. Extractability implies two other strong computational infeasibility lemmas:

- It is hard to produce (c, d, d') such that (c, d) and (c, d') are valid commitment pairs and $d \neq d'$.
- It is hard to find a c for which a random decommitment d will be valid with non-negligible probability.

We refer to the probability that a PPT adversary in time t can break such properties as $\varepsilon_{\mathsf{bind}}$ and $\varepsilon_{\mathsf{rand}}$, respectively. When appropriate, we directly specify security in terms of $\varepsilon_{\mathsf{bind}}$ and $\varepsilon_{\mathsf{rand}}$ for conceptual clarity and because it is generally simple to prove tight bounds on these properties directly (rather than relying on reductions to breaking extractability). Again, see Appendix A for the proofs.

Trapdoor Permutations (TDPs) 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 λ of the generation algorithm.

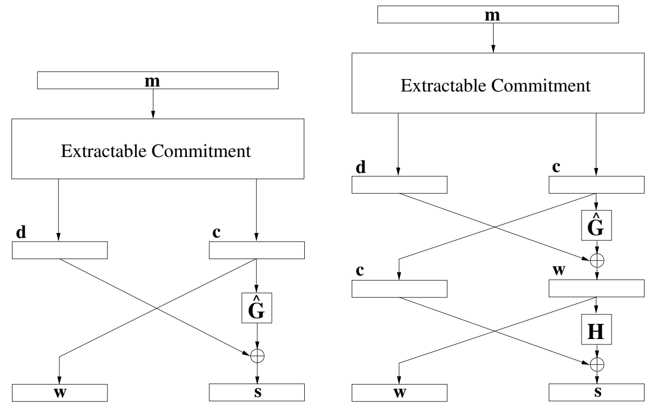

Figure 2: Schema for a Feistel P-Pad (left) and a Feistel S-Pad/X-Pad (right) on input message m.

2.2 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.

2.3 S-Pad and X-Pad Schemes

Our previous construction for Feistel **P**-Pads does not suffice to produce a secure **S**-Pad (which is strictly harder to achieve than a **P**-Pad). For example, we will see in Section 5 that OAEP is a special case of our **P**-Pad construction. Yet, it was shown to be potentially insecure when used as a single padding, by the result of [26]. 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.

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 IND-CCA security guarantee only) when compared to **P**-Pads. This is due to the substantial increase in the IND-CCA reduction's running time, which requires time proportional to $q_H \cdot q_G$ (underlined in the statement of Theorem 2 in the following section). 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 TDP (as in [26]). However, if we are willing to place a small portion of the padding outside the TDP (as was done by [17] for 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.

3. SIGNCRYPTION

We now see how these Feistel constructions allow one to build simple, efficient, and secure signcryption primitives from any TDP.

3.1 Definitions

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 ℓ 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 λ is the security parameter, SDK_U is the signing/decryption key that is kept private, and VEK_U is the verification/encryption key made public. Without loss of generality, we assume that VEK_U is determined from SDK_U.

The randomized signcryption algorithm SigEnc for user U implicitly takes as input the user's secret key SDK_U and explicitly takes as input the message $m \in \mathcal{M}$, the label ℓ, and the identity of the recipient, in order to compute and output the signcryption Π. For simplicity, we consider this identity ID to be a public key 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 de-signcryption algorithm VerDec implicitly takes the user's private SDK_U and explicitly takes as input the signcryption $\tilde{\Pi}$, the label ℓ, 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 ℓ.

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, with the added requirement that ℓ is considered part of the ciphertext (for the purposes of CCA decryption oracle queries), and must be authenticated. However, there is no hiding requirement for the label ℓ.

As expected, the security for signcryption consists of IND-CCA and sUF-CMA components when attacking some user U. We provide formal definitions of security in Appendix A; here, we only informally describe the necessary properties. For IND-CCA security, let $\varepsilon_{\mathsf{CCA}}$ be the probability that any PPT adversary can distinguish between $\mathsf{SigEnc}^\ell_S(m_0, \mathsf{VEK}_U)$ from $\mathsf{SigEnc}^\ell_S(m_1, \mathsf{VEK}_U)$ for some message pair m_0, m_1 and a label ℓ. For sUF-CMA security, let $\varepsilon_{\mathsf{CMA}}$ be the probability that any PPT adversary can forge a "valid" signature pair (Π, ℓ) of some message m from U to *any* user R. Both $\varepsilon_{\mathsf{CCA}}$ and $\varepsilon_{\mathsf{CMA}}$ must be negligible in the security parameter λ. We call any scheme satisfying these properties a $(t, \varepsilon_{\mathsf{CCA}}, \varepsilon_{\mathsf{CMA}}, q_D, q_S)$-secure signcryption scheme.

3.2 Signcryption from Feistel Paddings

We now construct signcryption primitives with support for associated data using the three padding paradigms described in Section 2. 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 provided that properly-formed meta-data $\mathcal{L}$ is used in the padding.

Due to space limitations, the proofs of the following theorems can be found in the full version of this paper [9].

Theorem 1 (Signcryption from Feistel P-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* **P***-Pad induced by $\mathcal{C}$. Define the* SigEnc *and* VerDec *algorithms as follows:*

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

We require that, at a minimum, the meta-data $\mathcal{L}$ must contain the associated data ℓ, as well as the published TDPs 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

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

To improve the exact security of authentication in our constructions, we consider a general class of TDPs: those induced by a family of *claw-free permutation* pairs [12]. See Appendix A for a formal definition of claw-free permutations. (Note that all known TDP families, such as RSA, Rabin, and Paillier are induced by a claw-free permutation family with $\varepsilon_{\mathsf{claw}} = \varepsilon_{\mathsf{TDP}}$.)

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

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

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.

The following theorem states our main 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. Again, see [9] for the proofs.

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:*

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

We require that, at a minimum, the meta-data $\mathcal{L}$ must contain the associated data ℓ, as well as the published TDPs 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

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

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

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

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 TDPs are applied is significant.[3]

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 safely used in any of our paradigms, as the situation demands.

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

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

Again, we require that, at a minimum, the meta-data $\mathcal{L}$ must contain the associated data ℓ, as well as the published TDPs of the message's sender and intended recipient. This signcryption scheme has the same exact security bounds as those of the Feistel **S**-Pads of Theorem 2 for both TDPs and claw-free permutations, but with an improvement in the running time of the reduction such that

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

Recall that **S**-Pad and **X**-Pad are constructed identically, the only difference being the selection of length parameters (and the manner in which the TDPs are applied to the output). In particular, since s is concatenated alongside the output of the TDPs, length parameters should be chosen to minimize the length of s. 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 use of the **X**-Pad paradigm instead of the **S**-Pad paradigm in most situations.

4. PLAIN ENCRYPTION & SIGNATURE

Although not all signcryption schemes imply natural stand-alone encryption and/or signature schemes (*e.g.*, [27] does not), our constructions lead to both natural and *optimal* signature and encryption schemes. By a simple argument, our proofs of security for signcryption will also imply tight reductions for the IND-CCA security of encryptions and the sUF-CMA security of signatures when the TDPs are applied appropriately, shown in Table 2. In particular, this implies that our padding constructions can be used as "universal paddings", as in [6, 18]. Furthermore, the encryption and signature schemes induced by our suggested construction (see Section 5) in the **S**-Pad paradigm are optimal (as compared to the best-known TDP-based schemes of equivalent ciphertext length). The same construction used in the **X**-Pad paradigm achieves tighter security for encryption at the expense of a slightly increased minimum ciphertext length, as was done in [17].

Additional useful consequences of our security proofs include the ability to reuse the same public key for encryption, signature, *and* signcryption (both as sender and recipient), the capability to extract a "signature"[4] from a signcryption (a *non-repudiation* guarantee), and support for non-malleability of encryption with respect to public associated data (often referred to as a "tag"). Signatures on long messages can be easily achieved by including messages in the metadata field. Similarly, the efficient extension of our signcryption scheme to long messages described in Section 6 also applies to the induced encryption schemes. Once again, all of these features can be provided by a *single*, elegant implementation using our proposed scheme.

5. PROBABILISTIC SIGNATURE AND ENCRYPTION PADDING

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, which uses a random oracle $K : \{0,1\}^{|d|} \rightarrow \{0,1\}^{|c|}$,

$$\begin{array}{rcl} c & \leftarrow & (m_1 \|\, 0^{(|c|-|m_1|)}) \oplus K(m_2\|r) \\ d & \leftarrow & (m_2\|r) \end{array}$$

The scheme, shown in Figure 3, is parameterized by the selection of the lengths of c, d, m_1, m_2.

[3]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 $\mathsf{EtS}/\mathcal{S}\mathsf{tE}$ compositions, where both orders where equally effective [1].

[4]Technically, the "signature" requires the metadata $\mathcal{L}$ for verification, which includes the identity of the intended message recipient.

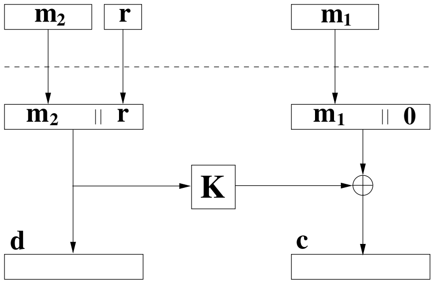

Figure 3: Schema for the PSEP extractable commitment on input $m = m_1 \| m_2$

The following Lemma gives exact security for the commitment scheme used in PSEP, in terms of the relevant selectable parameters. See [9] for the simple proof.

Lemma 1 *The commitment scheme*

$$\left\langle d = m_2 \| r,\ c = (m_1 \|\, 0^{(|c| - |m_1|)}) \oplus K(m_2 \| r) \right\rangle$$

defining PSEP *satisfies the following, where* q_K *is the number of oracle queries to* K *made by the adversary:*

$$\begin{aligned}
\varepsilon_{\mathsf{hide}} &\leq q_K \cdot 2^{-(|d| - |m_2|)} \\
\varepsilon_{\mathsf{extract}} &\leq (q_K^2 + 1) \cdot 2^{-(|c| - |m_1|)} \\
\varepsilon_{\mathsf{bind}} &\leq 2 \cdot \varepsilon_{\mathsf{extract}} \\
\varepsilon_{\mathsf{rand}} &\leq 2^{-(|c| - |m_1|)}
\end{aligned}$$

Using this commitment pair $\langle d, c \rangle$, applying a single round of the Feistel Transform yields PSEP1: $\left\langle w \leftarrow c\ ;\ s \leftarrow \hat{G}(w) \oplus d \right\rangle$. PSEP1 is sufficient for use as a Feistel **P**-Pad for signcryption. Interestingly, it can be seen that both OAEP [3] and PSS-R [5] are special cases of PSEP1 for appropriate selections of the commitment scheme parameters. The parameters corresponding to OAEP ($|m_1| = 0$) and PSS-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 would require $|m| \leq k$, while the total length $2k$ of PSEP1 potentially allows one to fit $|m| \approx 2k$, which we can indeed do by splitting m almost evenly.

Applying a second round of Feistel (a very inexpensive operation) yields the scheme PSEP2. PSEP2 can be used in any of the three modes discussed in Section 2 (*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 PSEP2 scheme would be rather difficult to analyze directly, in our general framework the proof of the simple Lemma 1 is all one needs to obtain many useful results. Namely, by leveraging the Theorems in Section 3, we get tight exact security bounds for PSEP2, showing that it can be used as a **P**-Pad, **S**-Pad, or **X**-Pad. Moreover, per Section 4, 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 [8], 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] There are many *very efficient* such symmetric encryptions, *i.e.*, $M \oplus F(\tau)$ works when F is a RO. (There are many RO-free encryptions as well; see [8].)

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 key-size $\hat{n}$ (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 τ 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_{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 [8] for signcryption, with exact security) is given in [9].

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 metadata $\mathcal{L}$, as shown in Figure 4. Additionally, it is possible to move a portion of the message into the padding alongside the encryption key to save otherwise wasted space. Thus, the overhead for long messages is the same as that for short messages plus the length of the symmetric key, which will typically be 128-bits.

7. SIGNCRYPTION & KEY EXCHANGE

In this section, we evaluate the performance of our signcryption scheme in the context of a simple key-exchange protocol, comparing the result to the SSLv3 (Secure Sockets Layer version 3) protocol. SSLv3 is the dominant protocol for establishing secure connections between client web browsers and web servers. In the common situation, web servers have certified RSA public keys representing their identities, whereas clients do not (and thus are usually not authenticated).

A minimal SSLv3 protocol will proceed roughly with a client (S) sending a *handshake* to the server (R), which acknowledges the handshake and responds with its certified public-key ID_R. In the third round, S sends the encryption of a fresh symmetric key τ, namely, $RSA_R(\tau)$.

To achieve forward security, the server can either generate a signed, one-time RSA key per connection or can employ Diffie-Hellman Key Exchange. However, both schemes are considered too computationally expensive for popular servers. Setting aside the issue of forward security, the computational requirements for the server mainly come from decrypting the fresh key τ, which remains expensive, even for low-exponent RSA.

On the other hand, a minimal, forward-secure signcryption protocol requires only two rounds: the client S sends a handshake and a one-time public key ID_S; the server R generates a fresh symmetric key τ, and acknowledges the handshake with ID_R and the signcryption of τ from R to S. Provided that the client uses a one-time public key for every key exchange, the protocol provides

[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.

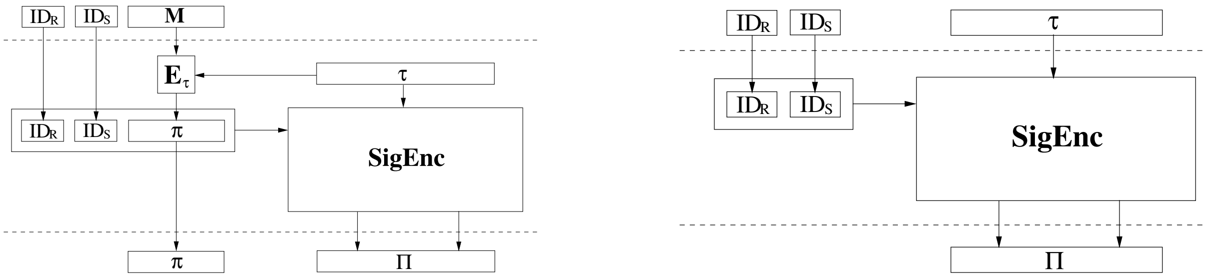

Figure 4: Using the signcryption primitive on long messages (left) and for key-exchange (right)

RSA	PSEP1	PSEP2	**SSLv3** (not)	**SSLv3** (forward)
1024-bit	24	24	29	83
2048-bit	99	96	66	430
4096-bit	565	563	308	2890

Table 3: Latency (in milliseconds) of sequential key exchange for PSEP protocols, compared to SSLv3 without and with forward security (*not* and *forward*, respectively).

full forward security and precludes replay attacks, yet the expensive key-generation protocol[6] is performed by the client offline, as opposed to by the server as before.

As a faster alternative, but at the cost of forward security, the client could include a short nonce N along with a longer-lived ID_S in its handshake, which the server R would additionally use as part of the label in the signcryption of τ. This simple protocol generalizes [25, §8.1]. The trade-off of efficiency against forward security can thus be determined by the client, which is free to generate fresh public keys as often as desired.

We implemented this signcryption-based key-exchange protocol to quantify its performance relative to SSLv3. Our signcryption implementation uses the PSEP1 and PSEP2 constructions described in Section 5 and RSA as the trapdoor permutation. Both our implementation and stunnel 3.26, a popular SSLv3 implementation used for comparison, use the OpenSSL 0.9.7b crypto libraries for all underlying cryptographic operations.

Table 3 summarizes the online performance results from these key-exchange implementations. We ran the client and server applications on two separate Pentium II Xeon 2.0 GHz machines, both running RedHat Linux 7.3 and located on the same 100 Mbps switched Ethernet (with network latency of ˜0.5 ms). The stunnel protocol with no forward security essentially uses the minimal SSL key exchange described above. The stunnel benchmarks with forward security were obtained by enabling its Diffie-Hellman key exchange, with parameters chosen to meet the same security level as the RSA parameters (*i.e.,*, the moduli lengths were the same). Note that the performance numbers for our signcryption protocol do *not* include the overhead of generating one-time RSA keys, as this computation can be performed offline using spare cycles. We report the minimum average latency achieved over ten trials, performing ten key exchanges in each trial.

We see that for 1024-bit keys, our signcryption-based protocol is a little more than three-times faster than the forward-secure SSL protocol; for 4096-bit keys, our protocol enjoys operation that is more than five-times faster. Note that, as SSL has one extra round compared to our signcryption-based key exchange, our protocol would offer even better comparative performance in the wide-area.

[6]We measured a Pentium II Xeon 2.0 GHz machine to take ˜260 ms on average to generate a 1024-bit RSA key.

8. RELATED WORK

While padding schemes are very popular in the design of ordinary encryption and signature schemes (*e.g.*, [3, 5, 26, 11]), the most relevant previous works are those related to signcryption and universal paddings. The comparison of our constructions to previous work is summarized in Table 1 in the Introduction.

Comparing with signcryption schemes [1, 20]. We believe that our methods noticeably improve all previously-proposed signcryption schemes, both from practical and theoretical perspectives.

Our main improvement over the generic methods from [1] come in much improved message bandwidth, key reuse, better exact security, and better qualitative security. To best illustrate it, we consider the TDP-based implementation of the "commit-then-encrypt-and-sign" (CtE&S) and compare it to our parallel **P**-Pad approach. In CtE&S, one first applies any commitment scheme to transform a modified message m', then applies two new, independent padding schemes to the commitment result, and finally applies a corresponding TDP to the padding results. 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 [26] or has to pad the message to be longer than the length of the TDP [17]. In contrast, we commit to m once and then apply a deterministic, length-preserving Feistel Transform to obtain the required w and s. Moreover, we are guaranteed to always obtain tight exact security. Recent work [22] optimizing CtE&S for the RO model still inherits many of its drawbacks, while also limiting the message bandwidth to less than half of the ciphertext length.

Mao and Lee [20] use PSS-R padding for sequential signcryption with RSA. Namely, for R to transmit message m to S (where each user U has key RSA_U), R sends $RSA_R(RSA_S^{-1}(w\|s))$, where $w\|s$ is the result of PSS-R applied to m. Thus, it is similar to our **S**-Pad paradigm, albeit restricted to RSA and PSS-R. Unfortunately, PSS-R is not 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. 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 universal padding schemes [6, 18]. 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 TDP-based signature *or* a plain TDP-based encryption, but not to *simultaneous* signature and encryption (*i.e.*, signcryption). While [6] constructed one concrete universal padding scheme (PSS-R), with poor exact security and only specific to RSA, [18] gave three concrete padding schemes with nearly-optimal exact security for any 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 [18]. 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]). Additionally, while both [6, 18] explicitly considered the question of key reuse for their plain "signature-encryption" application (as did the earlier work of [13]), their results do not imply similar results in our more complicated *signcryption* setting.

9. CONCLUSIONS

This paper proposes several highly-practical and optimized constructions for use as signcryption primitives. All our signcryption schemes, built directly from trapdoor permutations such as RSA, 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.

We present three methods for signcryption, based on what we call Parallel, Sequential, and eXtended sequential Padding schemes (**P**-Pad, **S**-Pad, **X**-Pad). All three schemes entail applying one or two rounds of the Feistel Transform to some basic pair $\langle d, c\rangle$ that form an *extractable commitment scheme*, followed by the application of a TDP for signature and encryption. This general framework allows us to design the new PSEP padding scheme (without needing new proofs) which may be specially tailored for particular situations. Finally, our signcryption primitives, having support for associated data (and long messages), can be used to build a simple, efficient, and secure protocol for performing key-exchange.

10. REFERENCES

[1] AN, J. H., DODIS, Y., AND RABIN, T. On the security of joint signature and encrytion. In *Advances in Cryptology—EUROCRYPT 2002* (Amsterdam, The Netherlands, Apr. 2002).

[2] BAEK, J., STEINFELD, R., AND ZHENG, Y. Formal proofs for the security of signcryption. In *Proc. 5th Workshop on Practice and Theory in Public Key Cryptosystems* (Paris, France, Feb. 2002).

[3] BELLARE, M., AND ROGAWAY, P. Random oracles are practical: A paradigm for designing efficient protocols. In *Proc. 1st ACM Conference on Computer and Communication Security* (Fairfax, Virginia, Nov. 1993).

[4] BELLARE, M., AND ROGAWAY, P. Optimal asymmetric encryption. In *Advances in Cryptology—EUROCRYPT 94* (Perugia, Italy, May 1994).

[5] BELLARE, M., AND ROGAWAY, P. The exact security of digital signatures: How to sign with RSA and Rabin. In *Advances in Cryptology—EUROCRYPT 96* (Saragossa, Spain, May 1996).

[6] CORON, J.-S., JOYE, M., NACCACHE, D., AND PAILLER, P. Universal padding schemes for RSA. In *Advances in Cryptology—CRYPTO 2002* (Santa Barbara, California, Aug. 2002).

[7] DAVIS, D. Defective sign & encrypt in s/mime,pkcs#7, moss, pem, pgp, and xml. In *Proc. USENIX Technical Conference* (Boston, Massachusetts, June 2001).

[8] DODIS, Y., AND AN, J. H. Concealment and its applications to authenticated encryption. In *Advances in Cryptology—EUROCRYPT 2003* (Warsaw, Poland, May 2003).

[9] DODIS, Y., FREEDMAN, M. J., JARECKI, S., AND WALFISH, S. Optimal signcryption from any trapdoor permutation. Cryptology ePrint Archive, Report 2004/020, 2004.

[10] DODIS, Y., AND REYZIN, L. On the power of claw-free permutations. In *Proc. 3rd Conference on Security in Communication Networks* (Amalfi, Italy, 2002).

[11] FUJISAKI, E., OKAMOTOI, T., POINTCHEVAL, D., AND STERN, J. RSA-OAEP is secure under the RSA assumption. In *Advances in Cryptology—CRYPTO 2001* (Santa Barbara, California, Aug. 2001).

[12] GOLDWASSER, S., MICALI, S., AND RIVEST, R. L. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing 17*, 2 (Apr. 1988), 281–308.

[13] HABER, S., AND PINKAS, B. Combining public key cryptosystems. In *Proc. 8th ACM Conference on Computer and Communication Security* (Philadelphia, Pennsylvania, Nov. 2001).

[14] HE, W., AND WU, T. Cryptanalysis and improvement of petersen-michels signcryption schemes. *IEEE Computers and Digital Communications 146*, 2 (1999), 123–124.

[15] JAKOBSSON, M., STERN, J. P., AND YUNG, M. Scramble all, encrypt small. In *Proc. 6th Workshop on Fast Software Encryption* (Rome, Italy, Mar. 1999).

[16] KILIAN, J., Ed. *Advances in Cryptology—CRYPTO 2001* (University of California, Santa Barbara, 19–23 Aug. 2001), vol. 2139 of *Lecture Notes in Computer Science*, Springer-Verlag.

[17] KOBARA, K., AND IMAI, H. OAEP++ : A very simple way to apply OAEP to deterministic OW-CPA primitives. Cryptology ePrint Archive, Report 2002/130, 2002.

[18] KOMANO, Y., AND OHTA, K. Efficient universal padding techniques for multiplicative trapdoor one-way permutation. In *Advances in Cryptology—CRYPTO 2003* (Santa Barbara, California, Aug. 2003).

[19] LUBY, M., AND RACKOFF, C. How to construct pseudorandom permutations from pseudorandom functions. *SIAM Journal on Computing 17*, 2 (1988), 373–386.

[20] MAO, W., AND MALONE-LEE, J. Two birds one stone: Signcryption using RSA. In *Progress in Cryptology — CT-RSA 2003* (San Francisco, California, Apr. 2003).

[21] PETERSEN, H., AND MICHELS, M. Cryptanalysis and improvement of signcryption schemes. *IEEE Computers and Digital Communications 145*, 2 (1998), 140–151.

[22] PIEPRZYK, J., AND POINTCHEVAL, D. Parallel authentication and public-key encryption. In *Proc. 8th ACISP* (Wollongong, Australia, July 2003).

[23] ROGAWAY, P. Authenticated-encryption with associated-data. In *Ninth ACM Conference on Computer and Communication Security* (Washington, D.C., Nov. 17–21 2002), R. Sandhu, Ed., ACM.

[24] RSA LABORATORIES. *PKCS #1: RSA Encryption Standard. Version 1.5*. Nov. 1993.

[25] SHOUP, V. On formal models for secure key exchange. Cryptology ePrint Archive, Report 1999/012, 1999.

[26] SHOUP, V. OAEP reconsidered. In *Advances in Cryptology—CRYPTO 2001* (Santa Barbara, California, Aug. 2001).

[27] ZHENG, Y. Digital signcryption or how to achieve cost(signature & encryption) $\ll$ cost(signature) + cost(encryption). In *Advances in Cryptology—CRYPTO 97* (Santa Barbara, California, Aug. 1997).

[28] ZHENG, Y., AND IMAI, H. Efficient signcryption schemes on elliptic curves. *Information Processing Letters 68*, 6 (Dec. 1998), 227–233.

APPENDIX

A. SECURITY DEFINITIONS

A.1 Extractable Commitments

An extractable commitment must 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 security parameter λ. 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,

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

where $\mathcal{T}$ is a complete transcript of the RO queries made by $\mathcal{A}$ and $\varepsilon_{\mathsf{extract}}$ is negligible in λ. For syntactic convenience, we define $\mathsf{Extract}$ 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. While the "standard" notion of a commitment requires a binding property, a very strong form of binding is implied by extractability.

Lemma 2 (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.

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 $2\varepsilon_{\mathsf{extract}}$. $\square$

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 3 *$\forall\ \mathcal{A}$ running in time t,*

$$\Pr\left[\ \mathsf{Open}(c,d) \neq \perp \ \middle|\ c \leftarrow \mathcal{A}(1^k); d \xleftarrow{R} \{0,1\}^k \right] \stackrel{\text{def}}{\leq} \varepsilon_{\mathsf{rand}} \leq \varepsilon_{\mathsf{extract}} + 2^{-k}$$

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. $\square$

A.2 Security of Signcryption

The security for signcryption consists of $\mathsf{IND\text{-}CCA}$ 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 VEK_U, $\mathcal{A}$ can make up to q_S adaptive signcryption queries $\mathsf{SigEnc}_U^\ell(m, \mathsf{VEK}_R)$ for *arbitrary* VEK_R, as well as up to q_D de-signcryption queries $\mathsf{VerDec}_U^\ell(\Pi, \mathsf{VEK}_S)$, again for arbitrary VEK_S. (Of course, m, Π, ℓ can be arbitrary too).

The $\mathsf{IND\text{-}CCA}$ security of signcryption requires that no PPT adversary $\mathcal{A}$ can find some pair m_0, m_1 and a label ℓ, for which $\mathcal{A}$ can distinguish $\mathsf{SigEnc}_S^\ell(m_0, \mathsf{VEK}_U)$ from $\mathsf{SigEnc}_S^\ell(m_1, \mathsf{VEK}_U)$. Note, to create "valid" signcryptions that $\mathcal{A}$ must differentiate between, $\mathcal{A}$ must output the *secret key* 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 that even when *compromising* S, $\mathcal{A}$ still cannot "understand" messages that S sent to U. In fact, we even allow $\mathcal{A}$ to create the secret key SDK_S without necessarily generating it via Gen! Formally, for any PPT $\mathcal{A}$ running in time t,

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

where $\varepsilon_{\mathsf{CCA}}$ is negligible in the security parameter λ, and $\mathsf{Gen}(1^\lambda)$ (outputting $(\mathsf{SDK}_U, \mathsf{VEK}_U)$) is implicitly called at the beginning. In the guess stage, $\mathcal{A}$ only has the natural restriction of not querying 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 PPT $\mathcal{A}$ can forge a "valid" pair (Π, ℓ) (of some message m) from U to *any* user R, provided that Π was not previously returned from a query to SigEnc_U^ℓ. Again, in order to define "valid", we strengthen the definition by allowing $\mathcal{A}$ to come up with the presumed secret key SDK_R as part of his forgery. Formally, for any PPT $\mathcal{A}$ running in time t,

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

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

A.3 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 [12]. 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 λ. It is trivial to see that omitting g from the generation algorithm induces a TDP family with $\varepsilon_{\mathsf{TDP}} \leq \varepsilon_{\mathsf{claw}}$ (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 [10] 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 [18] for a similar reason.