# Signcryption (Short Survey)

Yevgeniy Dodis*

March 22, 2005

Encryption and signature schemes are fundamental cryptographic tools for providing privacy and authenticity, respectively, in the public-key setting. Traditionally, these two important building-blocks of public-key cryptography have been considered as *distinct* entities that may be *composed* in various ways to ensure *simultaneous* message privacy and authentication. However, in the last few years a new, separate primitive — called *signcryption* [14] — has emerged to model a process simultaneously achieving privacy and authenticity. This emergence was caused by many related reasons. The obvious one is the fact that given that *both* privacy and authenticity are simultaneously needed in so many applications, it makes a lot of sense to invest special effort into designing a tailored, more efficient solution than a mere composition of signature and encryption. Another reason is that viewing authenticated encryption as a separate *primitive* may conceptually simplify the design of complex protocols which require both privacy and authenticity, as signcryption could now be viewed as an "indivisible" atomic operation. Perhaps most importantly, it was noticed by [3, 2] (following some previous work in the symmetric-key setting [4, 10]) that proper modeling of signcryption is not so obvious. For example, a straightforward composition of signature and encryption might not always work; at least, unless some special care is applied [2]. The main reason for such difficulties is the fact that signcryption is a complex *multi-user* primitive, which opens a possibility for some subtle attacks (discussed below), not present in the settings of stand-alone signature and encryption.

Defining Signcryption. Syntactically, a signcryption scheme consists of the three efficient algorithms $(\mathsf{Gen}, \mathsf{SC}, \mathsf{DSC})$. The key generation algorithm $\mathsf{Gen}(1^\lambda)$ generates the key-pair $(\mathsf{SDK}_U, \mathsf{VEK}_U)$ for user $U$, 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 that is made public. The randomized signcryption algorithm $\mathsf{SC}$ for user $U$ implicitly takes as input the user's secret key $\mathsf{SDK}_U$, and explicitly takes as input the message $m$ and the identity of the recipient $\mathsf{ID}_R$, in order to compute and output the *signcryptext* on $\Pi$. For simplicity, we consider this identity $\mathsf{ID}_R$ to be a public key $\mathsf{VEK}_R$ of the recipient $R$, although $\mathsf{ID}$'s could generally include more convoluted information (as long as users can easily obtain $\mathsf{VEK}$ from $\mathsf{ID}$). Thus, we write $\mathsf{SC}_{\mathsf{SDK}_U}(m, \mathsf{ID}_R)$ as $\mathsf{SC}_{\mathsf{SDK}_U}(m, \mathsf{VEK}_R)$, or simply $\mathsf{SC}_U(m, \mathsf{VEK}_R)$.

*Department of Computer Science, New York University, 251 Mercer Street, New York, NY 10012, USA. Email: `dodis@cs.nyu.edu`

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

We also remark that it is often useful to add another optional parameter to both $\mathsf{SC}$ and $\mathsf{DSC}$ algorithms: a *label* $L$ (also termed *associated data* [11]). This label can be viewed as a public identifier which is "inseparably bound" to the message $m$ inside the signcryptext. Intuitively, de-signcrypting the signcryptext $\Pi$ of $m$ with the wrong label should be impossible, as well as changing $\Pi$ into a valid signcryptext $\tilde{\Pi}$ of the same $m$ under a different label.

Security of Signcryption. Security of signcryption consists of two distinct components: one ensuring privacy, and the other — authenticity. On a high level, privacy is defined somewhat analogously to the privacy of an ordinary encryption, while authenticity — to that of an ordinary digital signature. For example, one can talk about indistinguishability of signcryptexts under chosen ciphertext attack, or existential unforgeability of signcryptexts under chosen message attack, among others. For concreteness, we concentrate on the above two forms of security too, since they are the strongest.

However, several new issues come up due to the fact that signcryption / de-signcryption take as an extra argument the identity of the sender / recipient. Below, we semi-formally introduce some of those issues (see [2] for in-depth technical discussion, as well as formal definitions of signcryption).

- *Simultaneous Attacks.* Since the user $U$ utilizes its secret key $\mathsf{SDK}_U$ to both send and receive the data, it is reasonable to allow the adversary $\mathcal{A}$ oracle access to both the signcryption and the de-signcryption oracle for user $U$, irrespective of whether $\mathcal{A}$ is attacking privacy or authenticity of $U$.

- *Two- vs. Multi-user Setting.* In the simplistic two-user setting, where there are only two users $S$ and $R$ in the network, the explicit identities become redundant. This considerably simplifies the design of secure signcryption schemes (see below), while providing a very useful intermediate step towards general, multi-user constructions (which are often obtained by adding a simple twist to the basic two-user construction). Intuitively, the security in the two-user model already ensures that there are no weaknesses in the way the message is encapsulated inside the signcryptext, but does not ensure that the message is bound to the identities of the sender and/or recipient. In particular, it might still allow the adversary a large class of so called *identity fraud* attacks, where the adversary can "mess up" correct user identities without affecting the hidden message.

- *Public Non-Repudiation?* In a regular signature scheme, anybody can verify the validity of the signature, and unforgeability of the signature ensures that a signer $S$

indeed certified the message. Thus, we say that a signcryption scheme provides *non-repudiation* if the recipient can extract a regular (publicly verifiable) digital signature from the corresponding signcryptext. In general, however, it is a-priori only clear that the *recipient* $R$ is sure that $S$ sent the message. Indeed, without $R$'s secret key $\mathsf{SDK}_R$ others might not be able to verify the authenticity of the message, and it might not be possible for $R$ to extract a regular signature of $m$. Thus, signcryption does not necessarily provide non-repudiation. In fact, for some applications we might explicitly want *not* to have non-repudiation. For example, $S$ might be willing to send some confidential information to $R$ only under the condition that $R$ cannot convince others of this fact. To summarize, non-repudiation is an optional feature which some schemes support, others don't, and others explicitly avoid!

- *Insider vs. Outsider Security.* In fact, even with $R$'s secret key $\mathsf{SDK}_R$ it might be unclear to an observer whether $S$ indeed sent the message $m$ to $R$, as opposed to $R$ "making it up" with the help of $\mathsf{SDK}_R$. This forms the main basis for distinction between *insider-* and *outsider-secure* signcryption. Intuitively, in an outsider-secure scheme the adversary must compromise communication between two honest users (whose keys he does not know). Insider-secure signcryption protects a given user $U$ even if his partner might be malicious. For example, without $U$'s key, one cannot forge signcryptext from $U$ to any other user $R$, even with $R$'s secret key. Similarly, if honest $S$ sent $\Pi = \mathsf{SC}_S(m, \mathsf{VEK}_U)$ to $U$ and later exposed his key $\mathsf{SDK}_S$ to the adversary, the latter still cannot decrypt $\Pi$. Clearly, insider-security is stronger than outsider-security, but might not be needed in a given application. In fact, for applications supporting message repudiation, one typically does not want to have insider-security.

Supporting Long Inputs. Sometimes, it is easier to design natural signcryption schemes supporting short inputs. Below we give a general method how to create signcryption $\mathsf{SC}'$ supporting arbitrarily long inputs from $\mathsf{SC}$ which only supports fixed-length (and much shorter) inputs. The method was suggested by [8] and uses a new primitive called *concealment*. A concealment is a publicly known randomized transformation, which, on input $m$, outputs a *hider* $h$ and a *binder* $b$. Together, $h$ and $b$ allow one to recover $m$, but separately, (1) the hider $h$ reveals "no information" about $m$, while (2) the binder $b$ can be "meaningfully opened" by at most one hider $h$. Further, we require $|b| \ll |m|$ (otherwise, one could trivially set $b = m$, $h = \emptyset$). Now, we let $\mathsf{SC}'(m) = \langle \mathsf{SC}(b), h \rangle$ (and $\mathsf{DSC}'$ is similar). It was shown in [8] that the above method yields a secure signcryption $\mathsf{SC}'$. Further, a simple construction of concealment was given: set $h = E_\tau(m)$, $b = \langle \tau, H(h) \rangle$, where $E$ is a symmetric-key one-time secure encryption (with short key $\tau$) and $H$ is a collision-resistant hash function (with short output).

# Current Signcryption Schemes

We now survey several signcryption schemes achieving various levels of provable security.

Generic Composition Schemes. The two natural composition paradigms are "encrypt-then-sign" ($\mathcal{E}t\mathcal{S}$) and "sign-then-encrypt" ($\mathcal{S}t\mathcal{E}$). More specifically, assume $\mathsf{Enc}$ is a semantically secure encryption against chosen ciphertext attack, and $\mathsf{Sig}$ is an existentially unforgeable signature (with message recovery) against chosen message attack. Each user $U$ has a key for for $\mathsf{Sig}$ and $\mathsf{Enc}$. Then the "basic" $\mathcal{E}t\mathcal{S}$ from $S$ to $R$ outputs $\mathsf{Sig}_S(\mathsf{Enc}_R(m))$, while $\mathcal{S}t\mathcal{E}$ — $\mathsf{Enc}_R(\mathsf{Sig}_S(m))$. Additionally, [2] introduced a novel generic composition paradigm for *parallel* signcryption. Namely, assume we have a secure *commitment scheme*, which on input $m$, outputs a commitment $c$ and a decommitment $d$ (where $c$ is both hiding and binding). Then "commit-then-encrypt-and-sign" ($\mathcal{C}t\mathcal{E}\&\mathcal{S}$) outputs a pair $\langle \mathsf{Enc}_R(d), \mathsf{Sig}_S(c)\rangle$. Intuitively, the scheme is private as public $c$ reveals no information about $m$ (while $d$ is encrypted), and authentic since $c$ binds one to $m$. The advantage of the above scheme over the sequential $\mathcal{E}t\mathcal{S}$ and $\mathcal{S}t\mathcal{E}$ variants is the fact that expensive signature and encryption operations are performed in parallel. In fact, by using trapdoor commitments in place or regular commitments, most computation in $\mathcal{C}t\mathcal{E}\&\mathcal{S}$ — including the expensive computation of both public-key signature and encryption — can be done off-line, even before the message $m$ is known!

It was shown by [2] that all three basic composition paradigms yield an insider-secure signcryption in the two-user model. Moreover, $\mathcal{E}t\mathcal{S}$ is outsider-secure even if $\mathsf{Enc}$ is secure only against the chosen plaintext attack, and $\mathcal{S}t\mathcal{E}$ is outsider-secure even if $\mathsf{Sig}$ is only secure against no message attack. Clearly, all three paradigms are insecure in the multi-user model, since no effort is made to bind the message $m$ to the identities of the sender / recipient. For example, intercepting a signcryptext of the form $\mathsf{Sig}_S(e)$ from $S$ to $R$, an adversary $\mathcal{A}$ can produce $\mathsf{Sig}_{\mathcal{A}}(e)$, which is a valid signcryptext from $\mathcal{A}$ to $R$ of the same message $m$, even though $m$ is *unknown* to $\mathcal{A}$. [2] suggest a simple solution: when encrypting, always append the identity of the sender to the message, and when signing — of the recipient. For example, a multi-user secure variant of $\mathcal{E}t\mathcal{S}$ is $\mathsf{Sig}_S(\mathsf{Enc}_R(m, \mathsf{VEK}_S), \mathsf{VEK}_R)$. Notice, if $\mathsf{Enc}$ and/or $\mathsf{Sig}$ support labels, these identities can be part of the label rather than the message.

Finally, we remark that $\mathcal{S}t\mathcal{E}$ and $\mathcal{C}t\mathcal{E}\&\mathcal{S}$ always support non-repudiation, while $\mathcal{S}t\mathcal{E}$ might or might not.

Schemes from Trapdoor Permutations. The generic schemes above validate the fact that signcryption can be built from ordinary signature and encryption, but will be inefficient unless the latter are efficiently implemented. In practice, efficient signature and encryption schemes, such as $\mathsf{OAEP}$ [5], $\mathsf{OAEP}+$ [13], $\mathsf{PSS\text{-}R}$ [6], are built from trapdoor permutations, such as RSA, and are analyzed in the random oracle model. Even with these efficient implementations, however, the generic schemes will have several drawbacks. For example, users have to store two independent keys, the message bandwidth is suboptimal and the "exact security" of the scheme is not as good as one might expect. Thus, given that practical schemes are anyway built from trapdoor permutations, it is natural to have highly optimized *direct* signcryption constructions from trapdoor permutations (in the random oracle model).

This is the approach of [9]. In their model, each user $U$ independently picks a trapdoor permutation $f_U$ (together with its trapdoor, denoted $f_U^{-1}$) and publishes $f_U$ as its public key. (Notice, only a *single* key is chosen, unlike what is needed for the generic schemes.)

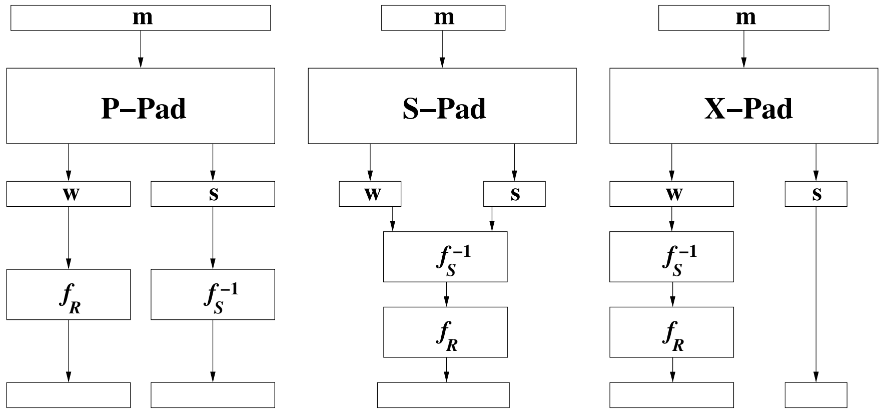


Figure 1: Generalized paddings as used by signcryption

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

Table 1: Signcryption Schemes Based on Trapdoor Permutations.

Then, [9] considers the following three paradigms termed **P**-Pad, **S**-Pad and **X**-Pad. Each paradigm proceeds by constructing a padding scheme produces $\pi(m) = w\|s$, and then these composing it with the corresponding permutations of the sender and the recipient as shown in Figure 1. Table 1 also shows how the corresponding approaches could be used for plain signature and encryption as well.

The convenience of each padding scheme depends on the application for which it is used. As was shown in [9], **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 trapdoor permutation (by achieving quite short $s$).

It remains to describe secure padding schemes $\pi$ for **P**-Pad, **S**-Pad and **X**-Pad. All constructions offered by [9] are quite similar. One starts with any *extractable commitment* $(c, d)$, where $c$ is the commitment and $d$ is the decommitment. Such schemes are very easy to construct in the random oracle model. For example, if $|m| = n$, for any $0 \leq a \leq n$, the following scheme is an extractable commitment: split $m = m_1\|m_2$, where $|m_1| = a$, $|m_2| = n - a$, and set

$$\begin{aligned} c &= G(r) \oplus m_1 \| H(m_2\|r) \\ d &= m_2\|r \end{aligned}$$

where $G$ and $H$ are random oracles (with appropriate input/output lengths) and $r$ is a random salt.

To get a secure padding scheme for the **P**-Pad paradigm, one should then apply the Feistel Transform to the resulting pair $(d, c)$, with yet another random oracle $F$ as the round function. Namely, set $w = c$, $s = F(c) \oplus d$. For example, using the extractable commitment above with $a = n$, we get nothing else but the OAEP padding, while $a = 0$ would give the PSS-R padding! For arbitrary $a$, [9] call the resulting hybrid between PSS-R and OAEP *Probabilistic Signature-Encryption Padding* (PSEP).

To get the padding $\pi$ sufficient for either **S**-Pad or **P**-Pad, one only needs to perform one more Feistel round to the construction above: $w' = s$, $s' = F'(s) \oplus w$, and set $\pi(m) = w' \| s'$. Coincidentally, the resulting $\pi$ also gives a very general construction of the so called *universal padding schemes* [7].

As described, the Feistel-based padding schemes above would only give the insider security in the two-user setting. To get multi-user security, all one needs to do is to prepend the pair $(\mathsf{VEK}_S, \mathsf{VEK}_R)$ to all the inputs to the random oracles $F$ and $F'$: namely, create effectively independent $F$ and $F'$ for every sender-recipient pairing! More generally, the paddings above also provide label support, if one sticks the label $L$ as part of the inputs to $F$ and $F'$.

Finally, we remark that **P**-Pad, **S**-Pad and **X**-Pad always support non-repudiation.

SCHEMES BASED ON GAP DIFFIE-HELLMAN. Finally, we present two very specific, but efficient schemes based on the so called *Gap Diffie-Hellman* assumption. Given a cyclic group $G$ of prime order $q$, and a generator $g$ of $G$, the assumption states that the computational Diffie-Hellman problem (CDH) is computationally hard, even if one is given oracle access to the decisional Diffie-Hellman (DDH) oracle. Specifically, it is hard to compute $g^{ab}$ from $g^a$ and $g^b$, even if one can test whether a tuple $\langle g^x, g^y, g^z \rangle$ satisfies $z = xy \bmod q$.

In both schemes, the user $U$ chooses a random $x_U \in \mathbb{Z}_q$ as its secret key $\mathsf{VEK}_U$, and sets its public key $\mathsf{SDK}_U = y_U = g^{x_U}$. The scheme of [1] is based on the following non-interactive key agreement between users $S$ and $R$. Namely, both $S$ and $R$ can compute the quantity $Q_{SR} = g^{x_R x_S} = y_S^{x_R} = y_R^{x_S}$. They then set the key $K_{SR} = H(Q_{SR})$, where $H$ is a random oracle, and then always use $K_{SR}$ to perform symmetric-key authenticated encryption of the message $m$. For the latter, they can use any secure symmetric-key scheme, like "encrypt-then-mac" [4] or OCB [12]. The resulting signcryption scheme can be shown to be outsider-secure for both privacy and authenticity, in the multi-user setting. Clearly, it is not insider-secure, since both $S$ and $R$ know the key $K_{SR}$. In fact, the scheme is perfectly repudiable, since all the signcryptexts from $S$ could have been easily faked by $R$.

To get insider-security for authenticity under the same assumption, one can instead consider the following scheme, originally due to [14], but formally analyzed by [3]. Below $G$ and $H$ are random oracles with appropriate domains, and $E$ is a one-time secure symmetric-key encryption (e.g., one-time pad will do). To signcrypt a message from $S$ to $R$, $S$ chooses a random $x \in \mathbb{Z}_q$, computes $Q = y_R^x$, makes a symmetric key $K = H(Q)$, sets $c \leftarrow E_K(m)$, computes the "validation tag" $r = G(m, y_A, y_B, Q)$ and finally $t = x(r + x_S)^{-1} \bmod q$. Then $S$ outputs $\langle c, r, t \rangle$ as the signcryption of $m$. To de-signcrypt $\langle c, r, t \rangle$, $R$ first recovers $g^x$ via $w = (y_S g^r)^t$, then recovers the Diffie-Hellman key $Q = w^{x_R}$, the encryption key $K = H(Q)$ and the message $m = D_K(c)$. Before outputting $m$, however, it double checks if $r = G(m, y_A, y_B, Q)$. While this scheme is insider-secure for authenticity, it is still not insider-secure for privacy.

We also mention that the scheme supports public non-repudiation. All that $R$ has to do is to reveal $Q$, $m$ and a proof that $Q = w^{x_R}$ (which can be done non-interactively using the Fiat-Shamir heuristics, applied to the three-move proof that $\langle g, y_R, w, Q\rangle$ form a $\mathsf{DDH}$-tuple).

# References


[1] Jee Hea An. Authenticated encryption in the public-key setting: Security notions and analyses. Cryptology ePrint Archive, Report 2001/079, 2001.

[2] 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/`.

[3] 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.

[4] Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In Tatsuaki Okamoto, editor, *Advances in Cryptology—ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 531–545, Kyoto, Japan, 3–7 December 2000. Springer-Verlag.

[5] 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/`.

[6] 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`.

[7] 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/`.

[8] 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.

[9] Yevgeniy Dodis, Michael J. Freedman, Stanislaw Jarecki, and Shabsi Walfish. Versatile Padding Schemes for Joint Signature and Encryption. In Birgit Pfitzmann, editor, *Eleventh ACM Conference on Computer and Communication Security*, pages 196–205. ACM, Octoberr 25–29 2004.

[10] Hugo Krawczyk. The order of encryption and authentication for protecting communications (or: How secure is ssl?). In Joe Kilian, editor, *Advances in Cryptology—CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 310–331. Springer-Verlag, 19–23 August 2001.

[11] Phillip Rogaway. Authenticated-encryption with associated-data. In Ravi Sandhu, editor, *Ninth ACM Conference on Computer and Communication Security*. ACM, November 17–21 2002.

[12] Phillip Rogaway, Mihir Bellare, John Black, and Ted Krovetz. OCB: A block-cipher mode of operation for efficient authenticated encryption. In Pierangela Samarati, editor, *Eighth ACM Conference on Computer and Communication Security*, pages 196–205. ACM, November 5–8 2001. Full version available from `http://www.cs.ucsdavis.edu/~rogaway`.

[13] Victor Shoup. OAEP reconsidered. In Joe Kilian, editor, *Advances in Cryptology—CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 240–259. Springer-Verlag, 19–23 August 2001.

[14] 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.