Efficient Verifiable Encryption (and Fair Exchange) of Digital Signatures

Giuseppe Ateniese
IBM Zurich Research Laboratory and
Department of Computer Science (DISI), University of Genoa.

Abstract

A fair exchange protocol allows two users to exchange items so that either each user gets the other's item or neither user does. In [2], *verifiable encryption* is introduced as a primitive that can be used to build extremely efficient fair exchange protocols where the items exchanged represent digital signatures. Such protocols may be used to digitally sign contracts.

This paper presents new simple schemes for verifiable encryption of digital signatures. We make use of a trusted third party (TTP) but in an *optimistic* sense, i.e., the TTP takes part in the protocol only if one user cheats or simply crashes. The performance of our schemes significantly surpasses that of prior art.

Keywords: Fair Exchange, Verifiable Encryption, Contract Signing Problem, Public-key Cryptography, Digital Signatures, Proof of Knowledge.

1 Introduction

Exchanging items over the Internet is becoming a major business opportunity. Electronic commerce usually involves two distrusted parties exchanging one item for another, for instance an electronic check for an electronic ticket. Specialized applications may include *contract signing*, *electronic purchase*, and *certified electronic mail* delivery. In simultaneous contract signing, Alice and Bob have agreed on a contract but neither wishes to sign unless the other signs as well. Face to face, this is easily solved: both simultaneously sign the contract. Unfortunately, simultaneity cannot be met in the discrete world.

There have been several approaches to solve the fair exchange problem depending on the definition of fairness on which they are based. In [16], fairness is interpreted as *equal computational effort*. That is, both Alice and Bob generate a signature of the contract and then they communicate by taking turns and sending bit-by-bit their signatures to each other. It is assumed that, at any stage, the computational effort required from the parties to obtain each other's secret is approximately equal. This approach does not require the intervention of a trusted third party; however it requires many rounds of interactions and, more importantly, that the two parties have equal computing power, an often unrealistic and undesirable assumption.

In [6], a probabilistic approach is adopted, i.e., the probability of correctness is gradually increased over several rounds of communication. Such an approach requires the existence and eventually the intervention of a trusted third party invoked only in case of dispute. However, the major drawback of the resulting protocol is its impracticality.

In [1], the authors introduce the *optimistic* approach. It also relies on the existence of a trusted third party but only invoked in the case of an exception. The protocol is *optimistic* since one party (the originator) takes the risk of sending its item first, optimistically hoping that the other party will respond by sending its item. If the other party does not reply as expected, the originator asks the third party to resolve the dispute (this implies that sufficient evidence must be accumulated during the protocol to support the resolution of the dispute). This approach results in particularly efficient fair exchange protocols for generic items ([1, 3]), although their correctness remains unproved.

We focus our attention on the optimistic fair exchange of digital signatures. Recently, papers [2] and [4] have presented protocols for optimistically exchange commonly used digital signature schemes. Both show that it is possible to build fair exchange protocols by means of what the authors in [2] have called *verifiable encryption* of digital signatures (i.e., a way to encrypt a signature under a designated public-key and subsequently prove that the resulting ciphertext indeed contains such a signature). The authors in [8] show how to generalize the schemes in [2] achieving more efficient schemes that can be proved secure without relying on random oracles.

In this paper we present new protocols for verifiable encryption of digital signatures with improved efficiency, thus providing a valid primitive that is of interest in designing fair exchange protocols (although there might be other applications to consider). We provide a security analysis in Appendix A, however we do not provide any formal security proofs for the fair exchange protocols that might be built by means of our verifiable encryption protocols (as rigorously done in [2]).

The rest of the paper is organized as follows. In Section 2, we briefly describe some types of building blocks necessary in the subsequent design of our schemes. In Section 4, we present verifiable encryption protocols for several digital signature schemes. In Section 5, we analyze our protocols in terms of both the number of modular exponentiations and

the amount of data transmitted.

2 Preliminaries

We assume that each communication party has the ability to generate and verify digital signatures. We say that the trusted third party $\mathcal{T}$ is *visible* if the end result of the fair exchange protocol makes it obvious that $\mathcal{T}$ participated during the protocol. The involvement of $\mathcal{T}$ may happen when either of the party cheats or simply crashes. A fair exchange protocol is *invasive* if one can tell that it was used to exchange signatures just by looking at such signatures. Typically, if the trusted third party $\mathcal{T}$ is visible then the fair exchange protocol is invasive (notice that the reverse is not true).

Given an instance s of a digital signature scheme on an arbitrary message, we make a $\mathcal{T}$-verifiable encryption $c(s)$ of s if such an encryption can be verified to contain s in a way that no useful information is revealed about s itself. Only $\mathcal{T}$ is able to recover s from $c(s)$ (during the *recovery* phase).

Finally, a fair exchange protocol provides *perfect fairness* if, when both parties follow the protocol properly, the protocol terminates with both parties having either the each other's item or nothing useful.

2.1 Cryptographic Tools

In this section we present signature schemes allowing a prover to convince a verifier of the equality of discrete logarithms (even when working in different groups). In short, the problem is, given g_1^x, g_2^x and a message m, generating a signature on m and, at the same time, showing that $\mathrm{Dlog}_{g_1} g_1^x = \mathrm{Dlog}_{g_2} g_2^x$ without revealing any useful information about x itself. We will denote an instance of this signature technique by $EQ\_DLOG(m; g_1^x, g_2^x; g_1, g_2)$.

We make use of so-called "proof-of-knowledge" systems that allow demonstrating knowledge of a secret such that no useful information is revealed in the process. Namely, we define Schnorr-like signature schemes [28] in order to show knowledge of relations among secrets. Substantially, these are signature schemes based on proofs of knowledge performed non-interactively making use of an ideal hash function $\mathcal{H}(\cdot)$ (*à la* Fiat-Shamir [17]).

Let G_q denote the unique subgroup of $\mathbb{Z}_p^*$ of order q. The parameters p, q are primes such that q divides $p-1$, for instance $p = 2q+1$.

Let $g, h \in G_q$ be publicly known bases. The prover selects a secret x mod q and computes $y_1 = g^x$ and $y_2 = h^x$. The prover must convince the verifier that:

$$\mathrm{Dlog}_g y_1 = \mathrm{Dlog}_h y_2.$$

The protocol, described in [14] by Chaum and Pedersen, is run as follows:

1. The prover randomly chooses $t \in \mathbb{Z}_q$ and sends $(a, b) = (g^t, h^t)$ to the verifier.
2. The verifier chooses a random challenge $c \in \mathbb{Z}_q$ and sends it to the prover.
3. The prover, then, sends $s = t - cx \bmod q$ to the verifier.
4. The verifier accepts the proof if:

$$a = g^s y_1^c \quad \text{and} \quad b = h^s y_2^c.$$

To turn the protocol above into a signature on an arbitrary message m, the signer can compute the pair (c, s) as:

$$c = \mathcal{H}(m \| y_1 \| y_2 \| g \| h \| g^t \| h^t), \; s = t - cx.$$

where $\mathcal{H}(\cdot)$ is a suitable hash function. To verify the signature (c, s) on m, it is sufficient to check whether $c' = c$, where

$$c' = \mathcal{H}(m \| y_1 \| y_2 \| g \| h \| g^s y_1^c \| h^s y_2^c).$$

This signature scheme works properly also into accurately chosen subgroup of $\mathbb{Z}_n^*$, where n is an RSA-like composite. In particular, in this paper we work into the subgroup of all quadratic residues modulo n, denoted by Q_n. Explicitly, $Q_n \subset \mathbb{Z}_n^*$ is the set of elements $a \in \mathbb{Z}_n^*$ such that there exists an $x \in \mathbb{Z}_n^*$ with $x^2 \equiv a \bmod n$. We select n as product of two *safe* primes p and q, i.e., such that $p = 2p'+1$ and $q = 2q'+1$ with p', q' primes. Thus, notice that Q_n is a cyclic group of order $p'q'$.

EQUALITY OF DISCRETE LOGARITHMS IN A GROUP OF UNKNOWN ORDER. Actually, the same signature scheme works properly even when the signer is working over a cyclic subgroup of $\mathbb{Z}_n^*$, $G = \langle g \rangle$, whose order $\#G = p'q'$ is unknown but its bit-length ℓ_G (i.e., the integer ℓ_G s.t. $2^{\ell_G - 1} \leq \#G < 2^{\ell_G}$) is publicly known.

We, now, show how the signer can generate a signature on a message $m \in \{0,1\}^*$ working with elements in G and, at the same time, showing knowledge of the discrete logarithm w.r.t. bases g and h satisfying $y_1 = g^x$ and $y_2 = h^x$. We make use of a hash function $\mathcal{H} : \{0,1\}^* \to \{0,1\}^k$, which maps a binary string of arbitrary length to a k-bit hash value. We also assume a security parameter $\epsilon > 1$. The signer computes a pair $(c, s) \in \{0,1\}^k \times \pm\{0,1\}^{\epsilon(\ell_G + k)+1}$ such that $c = \mathcal{H}(m \| y_1 \| y_2 \| g \| h \| g^s y_1^c \| h^s y_2^c)$. [1] This shows that the discrete logarithms of $y_1 = g^x$, w.r.t. base g, and $y_2 = h^x$, w.r.t. base h, are equal.

The signer, in possession of the secret x, is able to compute the signature (c, s), provided that $x = \mathrm{Dlog}_g y_1 = \mathrm{Dlog}_h y_2$, by choosing a random $t \in \pm\{0,1\}^{\epsilon(\ell_G + k)}$ and then computing c and s as:

$$c = \mathcal{H}(m \| y_1 \| y_2 \| g \| h \| g^t \| h^t), \; s = t - cx \quad (\text{in } \mathbb{Z}) \; .$$

A way of proving the security of the signature scheme above is via the oracle replay technique formalized in [25] by Pointcheval and Stern. In particular, the Schnorr signature with composite modulus has been proved secure in the random oracle model [5] by Poupard and Stern [26]. They showed that if an adversary is able to forge a signature under an adaptively chosen message attack (i.e., the strongest attack), then she is able to compute discrete logarithms in G (although, it may be possible to mount a chosen-message attack for a non-negligible proportion of public-keys).

EQUALITY OF DISCRETE LOGARITHMS FROM DIFFERENT GROUPS. Suppose now that g and h have different orders, q_1 and q_2,respectively. Thus, given two elements $y_1 = g^x$ and $y_2 = h^x$ of *different* groups $G_1 = \langle g \rangle$, $G_2 = \langle h \rangle$, the verifier can only conclude that the signer knows a value x such that $x \bmod q_1 = \mathrm{Dlog}_g y_1$ and $x \bmod q_2 = \mathrm{Dlog}_h y_2$. However, it is possible to prove that a secret x lies in a specific interval, more precisely given g^x with $-2^\ell < x < 2^\ell$ for an integer ℓ, it is possible to prove that x lies in the extended interval $]-2^{\epsilon(\ell+k)}, 2^{\epsilon(\ell+k)}[$. Hence, we might build

[1] The (abused) notation $s \in \pm\{0,1\}^{\epsilon(\ell_G+k)+1}$ denotes $|s| < 2^{\epsilon(\ell_G+k)+1}$

a signature scheme for showing that $\text{Dlog}_g y_1 = \text{Dlog}_h y_2$ in $\mathbb{Z}$ by combining the scheme for showing knowledge of a value x with $x \bmod q_1 = \text{Dlog}_g y_1$ and $x \bmod q_2 = \text{Dlog}_h y_2$, and the scheme for showing that $-2^{\epsilon(\ell+k)} < x < 2^{\epsilon(\ell+k)}$. Clearly, this can be done only if the length ℓ can be chosen such that $2^{\epsilon(\ell+k)+1} < \min\{q_1, q_2\}$, where q_1, q_2 are the orders of g and h, respectively.

This idea is formalized in [12] by Camenisch and Michels. They present, in [12], a concrete protocol for proving equality of discrete logarithms from different groups. Their protocol is mostly based on a technique developed by Fujisaki and Okamoto [18].

To provide a viable example of how it is possible to show that x lies in the extended interval $]-2^{\epsilon(\ell+k)}, 2^{\epsilon(\ell+k)}[$, we present a signature scheme derived from a protocol due to Chan, Frankel and Tsiounis [13], and Camenisch and Michels [9]. The scheme can trivially be extended to the more general interval $]X - 2^{\epsilon(\ell+k)}, X + 2^{\epsilon(\ell+k)}[$, for a given integer X [9]. The signature on a message $m \in \{0,1\}^*$, is the pair $(c, s) \in \{0,1\}^k \times \pm\{0,1\}^{\epsilon(\ell+k)+1}$ such that $c = \mathcal{H}(m\|y\|g\|g^s y^c)$. This shows knowledge of the discrete logarithm of $y = g^x$ w.r.t. base g and that this logarithm lies in $]-2^{\epsilon(\ell+k)}, 2^{\epsilon(\ell+k)}[$.

To produce (c, s), the signer in possession of the secret $x = \text{Dlog}_g y \in]-2^\ell, 2^\ell[$ chooses a random $t \in \pm\{0,1\}^{\epsilon(\ell+k)}$ and then computes c and s as:

$$c = \mathcal{H}(m\|y\|g\|g^t),\ s = t - cx \quad (\text{in } \mathbb{Z}) \ .$$

The underlying interactive protocol is proved to be a proof of knowledge (statistical honest-verifier zero knowledge),under the strong RSA assumption, in [10].

3 A Fair Exchange Protocol

To better clarify how a fair exchange may be built via verifiable encryptions, in this section we present an optimistic fair exchange protocol of digital signatures.

The protocol is essentially what is described in [4], it is non-invasive and provides perfect fairness. It may be used for signing contracts over a reliable communication network. Let Alice and Bob be two users willing to exchange digital signatures on a message m. Let $\mathcal{T}$ be a trusted third party and let $P_U(m)$ denote the encryption of the message m with U's public key, whereas $S_U(m)$ denotes the signature, generated by U, on the message m.

The fair exchange protocol is run as follows:

1. Alice sends Bob the message $P_{\mathcal{T}}(S_{\text{Alice}}(m))$ along with an *evidence* V stating that she has correctly encrypted her signature on m, i.e., that she has made a $\mathcal{T}$-verifiable encryption of $S_{\text{Alice}}(m)$.

2. Bob verifies V and, if valid, sends $S_{\text{Bob}}(m)$ to Alice, otherwise does nothing.

3. Alice verifies Bob's signature and, if valid, sends $S_{\text{Alice}}(m)$ to Bob.

4. If Bob does not receive anything or if Alice's signature is invalid, then he sends $P_{\mathcal{T}}(S_{\text{Alice}}(m))$ and $S_{\text{Bob}}(m)$ to $\mathcal{T}$. This provides a vehicle for $\mathcal{T}$ to understand whether the protocol was correctly carried out. If this is the case, $\mathcal{T}$ sends $S_{\text{Alice}}(m)$ to Bob and forwards $S_{\text{Bob}}(m)$ to Alice.

One drawback of this three-pass protocol is that Bob can ask $\mathcal{T}$ to reveal Alice's signature at any time after the step 1 above. Needless to say, this is undesirable for some (but not all!) applications. For instance, Alice may receive Bob's signature after a certain amount of time (at Bob's convenience), making the signature itself completely useless.

However, the protocol above is simple and suitable for our discussions. A better model for exchanging digital signatures is defined in [2].

Whatever protocol one might use, the building block remains the verifiable encryption of a digital signature. In the next sections we provide efficient constructions for verifiable encryption of commonly used signature schemes as well as newly proposed signature schemes provably secure against the most powerful attack.

4 Efficient Verifiable Encryptions

Suppose that Alice and Bob have agreed on a common message m. Alice generates a signature $S_A(m)$ and sends it "encrypted" to Bob by computing $C(S_A(m)) = P_{\mathcal{T}}(S_A(m))$. The problem, now, is that Alice must prove that the signature is valid and that $\mathcal{T}$ is able to get $S_A(m)$ from $C(S_A(m))$. In most of our protocols, $P_{\mathcal{T}}(\cdot)$ is the ElGamal encryption scheme. That is, given a secret key x and a corresponding public key g^x, a message m is encrypted by generating a random r and computing $K_1 = mg^{xr}$, $K_2 = g^r$. To get m from K_1, it is sufficient to compute $m = K_1/(K_2)^x$. A security analysis of our protocols is provided in Appendix A.

4.1 RSA Signatures

Let $n = pq$, with $p = 2p' + 1$ and $q = 2q' + 1$ where p', q' are primes . Let (e, n) be Alice's public key with e prime and d the corresponding secret key, i.e., $ed \equiv 1 \bmod p'q'$. To sign a message m, it is sufficient to compute $C = \mathcal{H}(m)^d \bmod n$ where $\mathcal{H}(\cdot)$ is a hash function defined as $\mathcal{H} : \{0,1\}^* \longrightarrow \mathbb{Z}_n$ [27]. [2] The signature is accepted only if $C^e \bmod n$ matches $\mathcal{H}(m)$. In order to make efficient the verifiable encryption of RSA signatures, we will make use of an *initialization phase* by which the user and the trusted third party $\mathcal{T}$ agree on common parameters.

Let Q_n be the subgroup of squares in $\mathbb{Z}_n^*$. During the initialization phase, Alice sends (e, n) to $\mathcal{T}$ (along with a certificate $CERT_A$). $\mathcal{T}$ verifies that (e, n) is the public key of Alice and randomly selects a $\bar{g} \in \mathbb{Z}_n^*$. Then, $\mathcal{T}$ sets $g = \bar{g}^2 \bmod n$, signs and sends back ($g \bmod n$, $y = g^x \bmod n$), where x is a secret random element [3]. The details are shown in Figure 1, where all the operations are taken modulo n and "A" is a string of data identifying Alice. This phase is done only once and $\mathcal{T}$ does not need to know the factors of n. [4]

Given the message m, Alice computes $\mathcal{H}(m)$ and then signs it by computing $\mathcal{H}(m)^d \bmod n$. Then, Alice encrypts the RSA signature via the ElGamal encryption scheme with the public-key $y = g^x$, i.e., by selecting a random r and computing $K_1 = \mathcal{H}(m)^d y^r$ and $K_2 = g^r$. To prove that

[2] For the sake of simplicity, we employ the hash-and-sign paradigm but, in practice, $\mathcal{H}(\ )$ is rather a redundancy function as defined in PKCS#1, ISO/IEC 9796, etc (see [22] p 442)

[3] By definition $g \in Q_n$ and with overwhelming probability the order of g is $p'q'$ (nevertheless paranoids may test whether $\gcd(g \pm 1, n) = 1$)

[4] However, we assume that Alice provides a proof of n being a product of safe primes during either the public-key certification process (executed with a Certification Authority) or the initialization phase (with $\mathcal{T}$ itself) See [11]

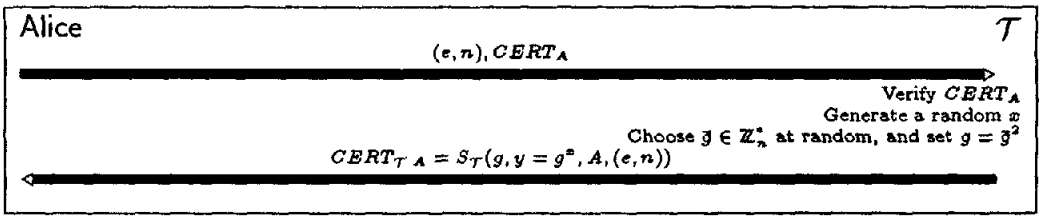

Figure 1: Initialization phase

she has correctly generated a $\mathcal{T}$-verifiable encryption, Alice releases an evidence showing that $\text{Dlog}_{y^e}(y^{er}) = \text{Dlog}_g(g^r)$ (via $EQ\_DLOG(\cdot)$, see section 2). The resulting message is composed of four elements as shown in Figure 2.

After receiving the message from Alice, Bob must verify that it is a $\mathcal{T}$-verifiable encryption of Alice's signature on m. Hence, he computes $\mathcal{H}(m)$ and then verifies $EQ\_DLOG(m; K_1^e / \mathcal{H}(m), K_2; y^e, g)$, where the bases $y = g^x$ and g are taken from $CERT_{\mathcal{T}\,A}$. Since n is product of *safe* primes, with overwhelming probability g, g^x and g^{xe} generate the same group, i.e., Q_n.

Embedding our scheme in a fair exchange protocol needs more careful thoughts which also apply to other schemes in this paper. First of all, $\mathcal{T}$ may avoid to store secret values per capita. In our example, $\mathcal{T}$ has to store x for Alice. This can be avoided by simply inserting a symmetric encryption of x into $CERT_{\mathcal{T}\,A}$ during the initialization phase. Thus, $\mathcal{T}$ needs to store only the symmetric encryption key.

Secondly, Alice should sign the message sent to Bob. This message should include the verifiable encryption of the Alice's signature and a *label* describing what Alice expects from Bob. This would assure the correctness of the recovery phase performed by $\mathcal{T}$ in case of dispute.

Thirdly, it helps to have semantic security [5]. If the message space is the same as the group generated by g, i.e., $Q_n = \langle g \rangle$, then it is well-known that DDH [6] implies the semantic security of the ElGamal encryption scheme modulo a composite. In our protocol, it is sufficient to square $\mathcal{H}(m)^d$ then encrypt $\alpha = \mathcal{H}(m)^{2d}$ (d is odd since e is prime). At Bob's side, he can compute W as $(K_1)^e / \mathcal{H}(m)^2$. Notice that, in case of dispute, $\mathcal{T}$ may get $\mathcal{H}(m)^d$ from $\alpha = \mathcal{H}(m)^{2d}$ by simply using the Euclidean algorithm, i.e., the mapping into Q_n does not affect the recovery algorithm. Namely, since $\alpha^e = \mathcal{H}(m)^2$ and $\gcd(e, 2) = 1$, there exist two integers a_1 and a_2 such that $\mathcal{H}(m)^d = \alpha^{a_1}\,\mathcal{H}(m)^{a_2}$. If Bob needs to know that he is working with quadratic residues modulo n (as presumably required by a proof of soundness), he can just square all the parameters (even the bases) before performing any modular operations with them.

Remark 1. Adding semantic security by squaring the digital signature suggests a simple and effective way to encrypt via the ElGamal encryption over a composite modulus. The ElGamal encryption scheme working in $\mathbb{Z}_n^*$, where n is a composite integer, was proposed by McCurley [21]. Given the encryption of a message m, i.e., my^w, g^w, where $y = g^x$ is a designated public-key, McCurley proved that learning m is at least as difficult as factoring the modulus n. However, the scheme *may* not be semantically secure since the Jacobi-symbol of g^x and g^w *may* leak information about y^w (although this is not always the case).

The standard solution is to use (under some conditions) a suitable hash function, then hashing y^w and xoring the result with m. Alternatively, it is possible to rely only on modular operations by selecting g of order $p'q'$, generator of Q_n (n is product of safe primes). Encryption of m can be done by computing $[R(m)]^2 y^w$ and g^w, where $R(\cdot)$ is a redundancy function. Therefore, the semantic security derives from the DDH assumption. The encryption function costs only 1.2 exponentiations via the exponent array algorithm (see Section 5). Decryption can be done by recovering $c = [R(m)]^2$ and computing the four square roots [7] of c. It is then sufficient to select the square root (the plaintext) that possesses the proper redundancy.

4.2 Gennaro-Halevi-Rabin Signatures

Let n be the product of two safe primes $p = 2p' + 1$ and $q = 2q' + 1$. Alice's certified public-key is (n, s), where s is randomly chosen in $\mathbb{Z}_n^*$. In order to sign a message m, Alice computes $e = \mathcal{H}(m)$ and $\sigma = s^{1/e} \bmod n$. To verify the signature, Bob computes $e = \mathcal{H}(m)$ and checks whether $\sigma^e = s \bmod n$. The Gennaro-Halevi-Rabin signature scheme [19] has been proved to be resistant against adaptive chosen message attack (i.e., where the attacker can dynamically ask the signer to sign any message, using him as an oracle), in the random oracle model [5], under the strong RSA assumption [8]. The authors in [19] provide other constructions eliminating the need for the random oracle model.

To make a $\mathcal{T}$-verifiable encryption of the signature σ, Alice performs an initialization phase, the same as in the RSA scheme. Thus, Alice gets the certificate $CERT_{\mathcal{T}\,A} = S_{\mathcal{T}}(g, y = g^x, A, (n, s))$ from $\mathcal{T}$, with g of order $p'q'$, generator of Q_n. The resulting protocol is shown in Figure 3 (all the operations are modulo n).

The output length of the hash function $\mathcal{H}(\cdot)$ should be sufficiently large, it is suggested that it should be $|n|$-bit long. Furthermore, $\mathcal{H}(\cdot)$ outputs primes and must satisfy some additional properties rigorously described in the original paper [19]. Finally, note that finding an odd integer that is not co-prime with $\phi(n)$ is as hard as factoring n (this assures the existence of the multiplicative inverse of

[5] Intuitively, a cryptosystem is semantically secure if, a passive attacker, who knows that one of just two possible messages has been encrypted, cannot yield any information about which of the two was actually encrypted by simply analyzing the ciphertext.

[6] Roughly said, given $G = \langle g \rangle$, the *Decisional Diffie-Hellman assumption* (DDH) states that no efficient algorithm can distinguish between the two distributions $\langle g^a, g^b, g^{ab} \rangle$ and $\langle g^a, g^b, g^c \rangle$ where a, b, c are random elements in $[1, |G|]$. The DDH is believed to be intractable in Q_n (see [7])

[7] If $\gcd(R(m), n) \neq 1$ then c has only one or two distinct square roots, but this happens very unlikely

[8] The strong RSA assumption states that given a random $z \in \mathbb{Z}_n^*$ with n an RSA modulus, finding a pair $(u, e) \in \mathbb{Z}_n^* \times \mathbb{Z}$ such that $e > 1$ and $u^e = z$ is hard to solve

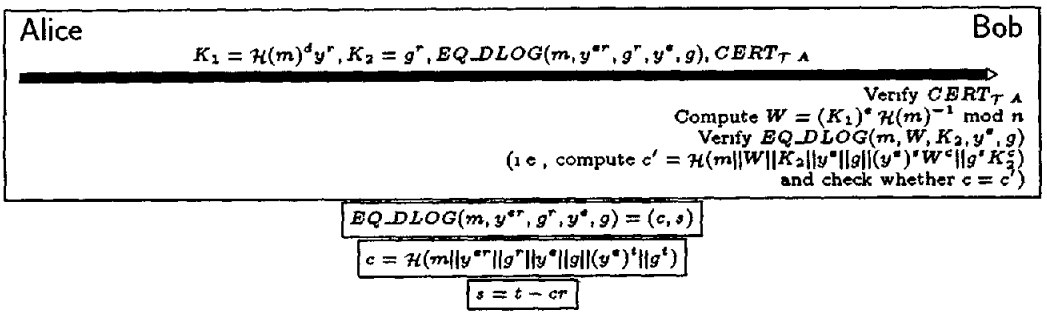

Figure 2: Verifiable encryption of an RSA signature

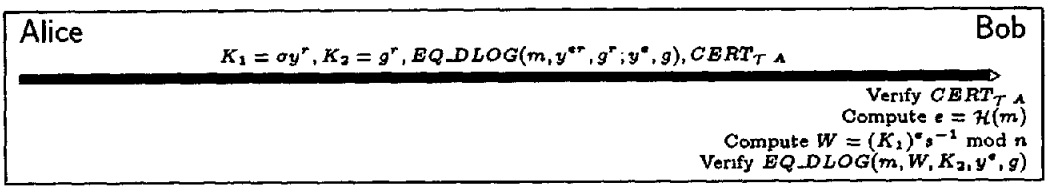

Figure 3: Verifiable encryption of a Gennaro-Halevi-Rabin signature

$e = \mathcal{H}(m)$ for any m in the message space). To get semantic security, it is sufficient to square σ (since $(\sigma^2)^{\mathcal{H}(m)} = s^2$ and $\gcd(\mathcal{H}(m), 2) = 1$, it is easy to recover σ) or, alternatively, to select a priori the parameter s as a quadratic residue, consequently $\sigma \in Q_n$.

4.3 Cramer-Shoup Signatures

Let $\jmath$ and z be two security parameters such that $\jmath + 1 < z$. Let $n = pq$, where p and q are z-bit safe primes, i.e., $p = 2p' + 1$ and $q = 2q' + 1$ with both p',q' primes. Alice's public-key is (n, b, x, e'), where b, x are randomly chosen in the subgroup of quadratic residues modulo n, Q_n, and e' is a random $(\jmath+1)$-bit prime. To generate a Cramer-Shoup signature [15] on a message m, Alice randomly selects a $(\jmath + 1)$-bit prime $e \neq e'$ and $u' \in Q_n$. Then, she computes $x' = (u')^{e'} b^{-\mathcal{H}(m)}$ and, finally, $u = (xb^{\mathcal{H}(x')})^{1/e}$. [9] The resulting signature on m is (e, u, u'). To verify it, Bob checks that e is a $(\jmath + 1)$-bit number different from e' and computes $x' = (u')^{e'} b^{-\mathcal{H}(m)}$. Then, Bob checks whether $x = u^e b^{-\mathcal{H}(x')}$. The Cramer-Shoup signature is quite efficient but, more importantly, it is provably secure (in the standard model) against adaptive chosen message attack under the strong RSA assumption.

To make a $\mathcal{T}$-verifiable encryption of the signature (e, u, u'), Alice performs an initialization phase, the same as in the RSA scheme. Thus, Alice gets the certificate $CERT_{\mathcal{T}\cdot A} = S_{\mathcal{T}}(g, y = g^x, A, (n, b, x, e'))$ from $\mathcal{T}$, with g of order $p'q'$. Then, Alice releases (e, u') along with the verifiable encryption of u. The details are shown in Figure 4.

In the matter of semantic security, it should be noted that $u \in Q_n$, hence $K_1 = uy^r$ is a quadratic residue.

[9] $\mathcal{H}(\,)$ is a hash function $\mathcal{H}\ \ \{0,1\} \longrightarrow \{0,1\}^{\jmath}$

4.4 Guillou-Quisquater Signatures

A trusted third party generates common parameters v, $n = pq$ and, for each user, a secret key B and an ID-based public-key J such that $B^v J \equiv 1 \bmod n$. Only the trusted third party knows the order of $\mathbb{Z}_n^*$. The Guillou-Quisquater signature [20] on a message m is computed as follows: randomly choose $r \in \mathbb{Z}_n^*$ and compute $T = r^v \bmod n$, $d = \mathcal{H}(m\|T)$ and $D = rB^d \bmod n$, where $\mathcal{H}(\cdot)$ is a suitable hash function. The resulting signature is the pair (d, D). The trusted third party, which has generated the system parameters, may cease to exist after the initialization phase.

To verify the signature, it is sufficient to check whether $d = \mathcal{H}(m\|D^v J^d)$, in fact notice that $D^v J^d \equiv (rB^d)^v J^d \equiv r^v (B^v J)^d \equiv r^v \bmod n$.

To make a verifiable encryption of a Guillou-Quisquater signature (d, D), we slightly modify the idea proposed in [4] since the derivative protocol, although quite elegant, can easily be broken as shown in Remark 2. Basically, the modulus n is generated as product of two safe primes, i.e., $n = pq$ with $p = 2p' + 1$ and $q = 2q' + 1$, and an element $g \in Q_n$ of order $p'q'$ is selected. The public-key of $\mathcal{T}$ is $y = g^x \bmod n$. To generate a $\mathcal{T}$-verifiable encryption of (d, D), Alice sends the ElGamal encryption of D, i.e., $K_1 = Dy^r$, $K_2 = g^r$, along with $V = D^v$, d itself and a signature showing knowledge of two equal discrete logarithms, $EQ\_DLOG(m; y^{vr}, g^r; y^v, g)$ (notice that Alice does not know the order of g). The signature is verified by checking the correctness of $EQ\_DLOG(m; K_1^v/V, K_2; y^v, g)$ and testing whether $d = \mathcal{H}(m\|VJ^d)$.

Remark 2. The protocol described in [4] is the same as above but n is generated such that it is straightforward to select a prime-order subgroup $G = \langle g \rangle$ of $\mathbb{Z}_n^*$. Specifically, $n = (2p'q+1)(2pq+1)$ for primes p', p, q. Then, g is selected as $g = \beta^{2pp'}$, where β is a generator of both $\mathbb{Z}^*_{2p'q+1}$ and $\mathbb{Z}^*_{2pq+1}$. Unfortunately, this implies that the subgroup $G = \langle g \rangle$ is of order the publicly known prime q (see [4]), fact

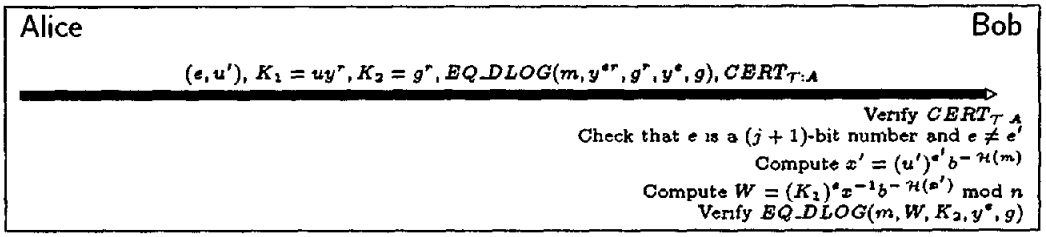

Figure 4: Verifiable encryption of a Cramer-Shoup signature

which can be exploited by a passive attacker in order to get the encrypted value D. In fact, given $K_1 = Dg^{xr} \bmod n$, the attacker may compute K_1^q yielding D^q (because $g^q = 1$). Since $V = D^v$ is known, it is enough to note that $\gcd(v, q) = 1$, then there exist two integers a_1, a_2 such that $a_1 v + a_2 q = 1$. Therefore, $D = (D^v)^{a_1}(D^q)^{a_2}$. Other attacks are possible since the inverse of v mod q always exists (since q is prime).

4.5 Discrete Logarithms

In this section we present methods for making verifiable encryptions of discrete logarithms. These methods allow us to make verifiable encryptions of digital signatures like DSA, ElGamal, and Schnorr.

Given a large prime p, we are actually working in prime-order subgroup of $\mathbb{Z}_p^*$ where finding the discrete logarithm is supposed to be hard. The problem we are facing, now, is the following: Alice and Bob agree on a common value α^x and Alice would like to generate a $\mathcal{T}$-verifiable encryption of x, given α^x. We propose two approaches to solve this problem in the following sections.

4.5.1 Tokens Approach

During the initialization phase, $\mathcal{T}$ generates and sends Alice *tokens* that can be used to perform one-time verifiable encryptions (a similar approach is also proposed in [2]). More concretely, Alice sends her certified public-key (p, q, α, α^a) to $\mathcal{T}$ who generates tokens of the form α^{t_i} with $1 \leq i \leq k$ and sends back the values $t_1, \ldots t_k$ and the signatures $S_{\mathcal{T}}(A, a_1 = \alpha^{t_1}), \ldots, S_{\mathcal{T}}(A, a_k = \alpha^{t_k})$ i.e., certificates binding the tokens to Alice's identity. Given $y = \alpha^x \bmod p$, to generate a $\mathcal{T}$-verifiable encryption of x, Alice selects $1 \leq j \leq k$ and encrypts x by computing $\ell = t_j - x \bmod q$. Then, she sends this encryption and the corresponding token to Bob. See Figure 5 for details.

Notice that each certificate $S_{\mathcal{T}}(A, a_j = \alpha^{t_j})$ can be used only once. After receiving the message from Alice, Bob needs to verify that Alice has actually encrypted the value $\mathrm{Dlog}_\alpha y$ by checking whether $a_j = \alpha^\ell y$.

The drawback of this approach is that one can make verifiable encryptions of discrete logarithms only for a limited number of times. In the next section we provide a method that overcomes this problem.

4.5.2 Trap-door Functions Approach

Given $\alpha^x \bmod p$, where α is the generator of a prime-order subgroup of $\mathbb{Z}_p^*$, it is hard to compute x. Suppose, now, that $\mathcal{T}$ selects an appropriate group G of order n in which computing the discrete logarithm is an easy task i.e., given an element $g \in G$ and $g^x \bmod n$, getting x is trivial. Therefore, Alice could make a verifiable encryption of x just by sending $\alpha^x \bmod p$ and $g^x \bmod n$ and then proving that $\mathrm{Dlog}_\alpha \alpha^x = \mathrm{Dlog}_g g^x$. Obviously, only $\mathcal{T}$ should be able to compute discrete logarithms w.r.t. base g, i.e., g and a description of the group G have to be public and constitute a trap-door function of some sort. To the best of our knowledge, there are two possible choices for a suitable trap-door function: those defined in the Naccache-Stern [23] and the Okamoto-Uchiyama [24] public-key cryptosystems, respectively.

The Naccache-Stern cryptosystem can shortly be described as follows: let $n = pq$ be an RSA modulus and B a small integer. Compute σ as a square-free odd B-smooth integer such that it divides $\phi(n)$ and is prime to $\phi(n)/\sigma$ (suggested size $\sigma > 2^{160}$). Let g be an element whose multiplicative order modulo n is a large multiple of σ. A message $m < \sigma$ is encrypted by computing $g^m \bmod n$. Decryption is performed using the prime factors of σ, getting m by chinese remaindering (see [23] for details). The resulting scheme is quite efficient (the optimized version costs a couple of RSA operations with a similar modulus) and it has already been implemented on smart-cards. Although a formal proof is not provided, the security of the scheme seems based on the *higher residuosity problem* that is widely believed to be infeasible.

The Okamoto-Uchiyama cryptosystem is a probabilistic encryption scheme provably secure against passive adversaries assuming the intractability of factoring $n = p^2 q$. Let k be a security parameter. The public-key is (n, g, h, k) where $n = p^2 q$ with p, q two large primes ($|p| = |q| = k$), $g \in \mathbb{Z}_n^*$ such that $\bar{g} = g^{p-1} \bmod p^2$ has order p, and $h = g^n \bmod n$. To encrypt a message $0 < m < 2^{k-1}$, select a random $r \in \mathbb{Z}_n$ and compute $C = g^m h^r \bmod n$. To decrypt it is sufficient to compute $C_p = C^{p-1} \bmod p^2$ and then $m = L(C_p)/L(\bar{g}) \bmod p$ where $L(x) = (x-1)/p$ is a function defined on elements in $\mathbb{Z}_{p^2}^*$ congruent to 1 modulo p (see [24] for details). The efficiency of the scheme depends on the size of m and is almost the same as that of the RSA scheme, assuming small size of the messages (e.g., 128 bits).

For the sake of simplicity, we focus on the Naccache-Stern cryptosystem but all the results can easily be adapted to work with the Okamoto-Uchiyama cryptosystem. The verifiable encryption of x given α^x is performed by computing $g^x \bmod n$ and showing that $\mathrm{Dlog}_\alpha \alpha^x = \mathrm{Dlog}_g g^x$ via $EQ\_DLOG(m; \alpha^x, g^x; \alpha, g)$, for an arbitrary message m. Although we still use the same notation (i.e., $EQ\_DLOG(\cdot)$) as in the rest of the paper, proving equality of discrete logarithms from different groups requires an additional step by which the verifier generates an element of secret order that is subsequently used in the protocol. Without this element, the proof of equality may not work properly (see [12]).

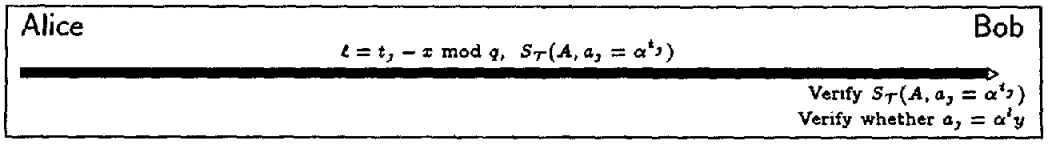

Figure 5: Verifiable encryption of a discrete logarithm

4.6 Schnorr and Poupard-Stern Signatures

Let α be a generator of the unique cyclic group of order a prime q in $\mathbb{Z}_p^*$, where p is some large prime number such that q divides $p-1$. Alice (the signer) selects the private key $1 \leq a \leq q$, computes $y = \alpha^a \bmod p$ and publishes (p, q, α, y). To generate a Schnorr signature on a message m, Alice selects a random secret integer k, with $1 \leq k \leq q-1$, and computes $r = \alpha^k \bmod p$, $e = \mathcal{H}(m \| r)$ [10], and $s = ae + k \bmod q$. Alice's signature on m is the pair (s, e). Bob verifies the signature by checking whether $e = e'$, where $e' = \mathcal{H}(m \| \alpha^s y^{-e})$.

Notice that, when Bob receives Alice's signature (s, e), he exponentiates α with s. Therefore, to make a $\mathcal{T}$-verifiable encryption of (s, e), Alice could send $\{\alpha^s, e\}$ to Bob, prove that she knows $\text{Dlog}_\alpha \alpha^s$ and, finally, make a verifiable encryption of s. The details are shown in Figure 6, where (g, n) denotes the public-key of $\mathcal{T}$ generated according to the Naccache-Stern cryptosystem (in particular, we set $\sigma > q$).

Showing, via $EQ\_DLOG(\cdot)$, that the discrete logarithm of w_1 equals that of w_2, implicitly proves that the signer (Alice) knows $\text{Dlog}_\alpha w_1$ and $\text{Dlog}_g w_2$, respectively. The exponent s must satisfy some range constraints as described in section 2.1; this might affect the choice of the random element k during the signature process.

The technique above also applies to the Poupard-Stern [26] signature scheme, that is the Schnorr signature modulo a composite. The Poupard-Stern signature has formally been proved to be resistant against chosen message attack in the random oracle model [5], under the discrete logarithm assumption.

4.7 ElGamal and DSA Signatures

Let α be a generator of the unique cyclic group of order a prime q in $\mathbb{Z}_p^*$, where p is some large prime number such that q divides $p-1$. Alice (the signer) selects the private key $1 \leq a \leq q$, computes $y = \alpha^a \bmod p$ and publishes (p, q, α, y). To generate an ElGamal signature on a message m, Alice selects a random $k \in [1, p-2]$ and computes $r = \alpha^k \bmod p$ and $s = ar + k\,\mathcal{H}(m) \bmod q$. [11] The signature is the pair (r, s). To verify it, Bob checks that $1 \leq r \leq p-1$ and tests whether $\alpha^s = y^r r^{\mathcal{H}(m)}$.

To make a $\mathcal{T}$-verifiable encryption of (r, s), Alice releases the values α^s, r along with the verifiable encryption of s. Once again, (g, n) denotes the public-key of $\mathcal{T}$ generated according to the Naccache-Stern cryptosystem. The details are shown in Figure 7.

The technique for ElGamal signatures could clearly be extended to work with DSA signatures, although care must be taken in specifying the size of the secret parameters as well as the output length of the hash function.

[10] $\mathcal{H}(\,)$ is a hash function $\mathcal{H}\ \ \{0,1\}^* \longrightarrow \mathbb{Z}_q$

[11] This is actually a modified version of the ElGamal signature scheme since we are working in prime-order subgroup of $\mathbb{Z}_p^*$ whereas the original scheme works directly in $\mathbb{Z}_p^*$

5 Comparisons

In this section we analyze the proposed verifiable encryption protocols in terms of the most expensive operation, i.e., the modular exponentiation. A modular exponentiation in $\mathbb{Z}_n^*$ requires almost $1.5\log(n)$ modular multiplications, however simultaneous exponentiations can be computed more efficiently by means of an exponent array (see [22] page 618). Namely, exponentiations of the form $a_1^{x_1}\ a_2^{x_2}$ and $a_1^{x_1}\ a_2^{x_2}\ a_3^{x_3}$ are only 16.7%, and 25% more costly than a single exponentiation, respectively.

In Table 1 we evaluate our schemes with respect to both the number of modular exponentiations and the amount (expressed in bytes) of data transmitted. The exponentiations we considered are those performed by *both* Alice and Bob *including the amount needed by each signature algorithm.* Finally, we are assuming a 1200-bit composite modulus n, a 768-bit prime modulus p, a 160-bit prime modulus q and a 128-bit hash function $\mathcal{H}(\cdot)$.

The efficiency of our protocols is remarkably better than that of protocols in related works. For instance, compared to the scheme in [2], our verifiable encryption of RSA signatures is about ten times faster and the amount of data transmitted is about twenty times smaller when choosing the same modulus for both schemes. Analogously, our verifiable encryption of ElGamal/DSA signatures is about twenty times faster and nine times shorter than that in [2]. Other verifiable encryptions, like those of Gennaro-Halevi-Rabin and Cramer-Shoup signatures, do not appear in existing works, consequently we can only evaluate them.

Table 1: Evaluation of our protocols.

Signature	Verifiable Encryption	
	# exponentiations	Size (bytes)
RSA	7.5	400
Gennaro et al	7.5	400
CramerShoup	8.7	544
GQ	10.5	544
Schnorr	8.3	388
ElGamal	8.5	388
DSA	11.6	484

6 Conclusion

This paper presented simple and particularly efficient verifiable encryption protocols for digital signatures. These protocols may be used as building blocks for designing efficient fair exchange of digital signatures. We have slightly modified the model of fair exchange by introducing an initialization phase for some of the digital signature schemes. However,

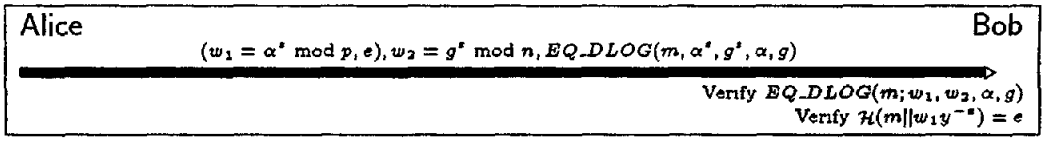

Figure 6: Verifiable encryption of a Schnorr signature

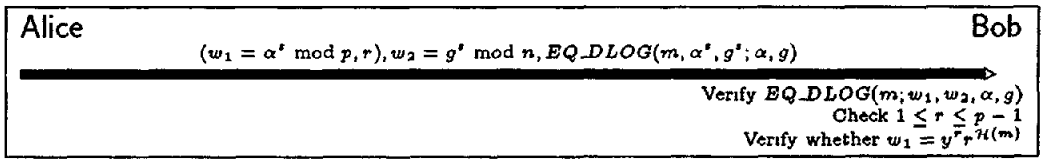

Figure 7: Verifiable encryption of an ElGamal signature

this phase is done only once and the resulting protocols are much more efficient than those of prior art. We have taken two directions for future research. First, we would like to analyze the security of fair exchange protocols built via our verifiable encryption schemes, eventually in an extended version of this paper. Second, as a long-term goal, we would like to develop a prototype implementation, preferably in a smart-card environment.

7 Acknowledgments

We are grateful to Victor Shoup for illuminating discussions concerning the topic of this paper. We would also like to express our thanks to Jan Camenisch, Marc Joye, and the anonymous referees for their insightful comments.

References

[1] N. Asokan, M. Schunter, and M. Waidner. Optimistic protocols for fair exchange. In *4th ACM Conference on Computer and Communication Security*, pages 8–17, ACM Press, 1997

[2] N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. In *Advances in Cryptology – EUROCRYPT '98*, volume 1403 of *Lecture Notes in Computer Science*, pages 591–606, Springer-Verlag, 1998.

[3] N. Asokan, V. Shoup, and M. Waidner. Asynchronous Protocols for Optimistic Fair Exchange. In *IEEE Symposium on Security and Privacy*, Oakland, California, 1998.

[4] F. Bao, R. H. Deng, and W. Mao. Efficient and Practical Fair Exchange Protocols with Off-line TTP. In *IEEE Symposium on Security and Privacy*, Oakland, California, 1998.

[5] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *1st ACM Conference on Computer and Communication Security*, pages 62–73, ACM Press, 1993.

[6] M. Ben-Or, O. Goldreich, S. Micali, and R. Rivest. A fair protocol for signing contracts. In *IEEE Transactions on Information Theory*, IT-36(1), pp. 40-46, 1990

[7] D. Boneh. The decision Diffie-Hellman problem. In *Algorithmic Number Theory (ANTS-III)*, volume 1423 of *Lecture Notes in Computer Science*, pages 48–63, Springer-Verlag, 1998.

[8] J. Camenisch and I. B. Damgard. Verifiable Encryption and Applications to Group Signatures and Signature Sharing. BRICS Technical Report, RS-98-32.

[9] J. Camenisch and M. Michels. A group signature scheme with improved efficiency. In *Advances in Cryptology – ASIACRYPT '98*, volume 1514 of *Lecture Notes in Computer Science*, pages 160–174, Springer-Verlag, 1998.

[10] J. Camenisch and M. Michels. A group signature scheme based on an RSA-variant. Tech. Report RS-98-27, BRICS, Aarhus, November 1998. An earlier version appears in [9].

[11] J. Camenisch and M. Michels. Proving in zero-knowledge that a number is the product of two safe primes. In *Advances in Cryptology – EUROCRYPT '99, Lecture Notes in Computer Science*, Springer-Verlag. To appear, 1999.

[12] J. Camenisch and M. Michels. Separability and Efficiency for Generic Group Signature Schemes. In *Advances in Cryptology – Crypto '99*, to appear, 1999.

[13] A. Chan, Y. Frankel and Y. Tsiounis. Easy come – easy go divisible cash. In *Advances in Cryptology – EUROCRYPT '98*, volume 1403 of *Lecture Notes in Computer Science*, pages 561–575, Springer-Verlag, 1998. Updated and corrected version available as GTE Technical Report.

[14] D. Chaum and T. Pedersen. Wallet databases with observers. In *Advances in Cryptology – Crypto '92*, pages 89-105, 1992.

[15] R. Cramer and V. Shoup. Signature Schemes Based on the Strong RSA Assumption. In *6st ACM Conference on Computer and Communication Security*, ACM Press, 1999.

[16] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *Comm. ACM.* vol. 28, no. 6, pp.637-647,1985.

[17] A. Fiat and A. Shamir. How to prove yourself: practical solutions to identification and signature problems. In *Advances in Cryptology – CRYPTO '86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194, Springer-Verlag, 1987.

[18] E. Fujisaki and T. Okamoto. Statistical Zero Knowledge Protocols to Prove Modular Polynomial Relations. In *Advances in Cryptology – CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 16–30, Springer-Verlag, 1997.

[19] R. Gennaro, S. Halevi, and T. Rabin. Secure signatures, without trees or random oracles. In *Advances in Cryptology – EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 123–139, Springer-Verlag, 1999

[20] L. C. Guillou and J. J. Quisquater. A paradoxical identity-based signature scheme resulting from zero-knowledge. In *Advances in Cryptology - CRYPTO '88*, volume 403 of *Lecture Notes in Computer Science*, pages 216–231, Springer-Verlag, 1988.

[21] K.S. McCurley. A key distribution system equivalent to factoring. In *Journal of Cryptology*, 1, pages 95-105, 1988.

[22] A. J. Menezes, P.C. van Oorschot, and S.A. Vanstone. *Handbook of applied cryptography*. CRC Press, 1996. ISBN 0-8493-8523-7.

[23] D. Naccache and J. Stern. A New Public Key Cryptosystem Based on Higher Residues. In *5th ACM Conference on Computer and Communications Security*, pages 59–66, ACM Press, 1998.

[24] T. Okamoto and S. Uchiyama. A New Public-Key Cryptosystem as Secure as Factoring. In *Advances in Cryptology - EUROCRYPT '98*, volume 1403 of *Lecture Notes in Computer Science*, pages 308–318, Springer-Verlag, 1998.

[25] D. Pointcheval and J. Stern. Security proofs for signature schemes. In *Advances in Cryptology - EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 387–398, Springer-Verlag, 1996.

[26] G. Poupard and J. Stern. Security analysis of a practical "on the fly" authentication and signature generation. In *Advances in Cryptology - EUROCRYPT '98*, volume 1403 of *Lecture Notes in Computer Science*, pages 422–436, Springer-Verlag, 1998.

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

[28] C.P. Schnorr. Efficient signature generation by smart-cards. *Journal of Cryptology*, 4(3):161–174, 1991.

A Security Analysis

We concentrate on the interactive protocol underlying the verifiable encryption of RSA signatures. The result also applies to the other RSA-based schemes.

Let $M = [H(m)]^2$, $y = g^x$ and $h = y^e$. The interactive protocol between Alice (prover) and Bob (verifier) is run as follows:

1. Alice chooses at random r, t and sends Bob:
$$\{K_1 = M^d y^r, K_2 = g^r, a = h^t, b = g^t\}.$$
2. Bob chooses a random challenge $c \in \{0,1\}^{160}$ and sends it to Alice.
3. Alice replies with $s = t + cr \pmod{p'q'}$.
4. Bob computes $W = K_1^e/M$ and checks whether:
$$h^s = aW^c \text{ and } g^s = bK_2^c.$$

Suppose, now, that Alice can successfully answer two distinct challenges. That is, given two challenges c_1, c_2 from Bob, Alice answers s_1, s_2. As noted in [14], we then have:

$$h^{s_1-s_2} = W^{c_1-c_2} \text{ and } g^{s_1-s_2} = K_2^{c_1-c_2}.$$

Thus, a value r exists such that:

$$r = \text{Dlog}_h W = \text{Dlog}_g K_2 = (s_1 - s_2)/(c_1 - c_2) (\text{mod } p'q').$$

Consequently, $W = h^r$, $K_2 = g^r$ and $K_1 = (Mh^r)^d$. This proves that (K_1, K_2) is indeed an encryption of Alice's signature on the message m.

Notice that we do not have to prove that the protocol above is a proof of knowledge, nevertheless it is interesting to analyze this possibility. We must rely on the strong RSA assumption, in fact the knowledge extractor does not know the group order $(p'q')$ and thus cannot compute $(s_1 - s_2)/(c_1 - c_2)$ (modulo $p'q'$). However, this computation can be carried out in $\mathbf{Z}$ since it is easy to see that, under the strong RSA assumption, $(c_1 - c_2)$ divides $(s_1 - s_2)$ (in $\mathbf{Z}$). Furthermore, since Alice (the prover) knows the factorization of n (excluding the protocol for the Guillou-Quisquater signatures), we should slightly modify the protocol as follows:

- During the initialization phase the trusted third party $\mathcal{T}$ selects a composite $\hat{n}$ and $\hat{g} \in \mathbf{Z}_{\hat{n}}^*$, and sends $CERT_{\mathcal{T}\,A} = S_{\mathcal{T}}(g, y = g^x, A, (e,n), \hat{g}, \hat{n})$ to Alice.
- Given $K_1 = M^d y^r$, $K_2 = g^r$, $a = h^t$, $b = g^t$ and $K_3 = \hat{g}^r \bmod \hat{n}$, Alice now convinces Bob that:
$$\text{Dlog}_y(K_1^e/M) = \text{Dlog}_g K_2 = \text{Dlog}_{\hat{g}} K_3.$$

Therefore, we can apply the proof techniques in [18, 10].