Practical Verifiable Encryption and Decryption of Discrete Logarithms

Jan Camenisch* Victor Shoup†

October 31, 2002

Abstract

This paper presents a variant of the new public key encryption of Cramer and Shoup based on Paillier's decision composite residuosity assumption, along with an efficient protocol for verifiable encryption of discrete logarithms. This is the first verifiable encryption system that provides chosen ciphertext security and avoids inefficient cut-and-choose proofs. This has numerous applications, including fair exchange and key escrow. We also present efficient protocols for verifiable decryption, which has applications to, e.g., confirmer signatures. The latter protocols build on a new protocol for proving whether or not two discrete logarithms are equal that is of independent interest. Prior such protocols were either inefficient or not zero-knowledge.
Keywords. Verifiable encryption, verifiable decryption, adaptive chosen ciphertext security, public key encryption.

1 Introduction

A public key encryption scheme is a primitive that allows a sender to secretly transmit a message to a receiver. However, in some applications, parties may be required to prove that a ciphertext encrypts a plaintext that satisfies a particular, application-dependent "validity" property, but without revealing any additional information about the plaintext. The proving party may be the encryptor, in which case we call this *verifiable encryption*, or she may be the decryptor, in which case we call this *verifiable decryption*.

A typical application is one where a designated trusted third party makes available a public encryption key but is otherwise only occasionally involved in the protocol. Using verifiable encryption, the encryptor of a plaintext can assure the recipient of the ciphertext that the plaintext is "valid," so that should the trusted third party be invoked to decrypt the ciphertext, a "valid" plaintext will be obtained. Using verifiable decryption, we can minimize the trust that must be placed in the third party, and we can also use the third party to prove the "validity" of a plaintext, without revealing the plaintext itself.

Example applications are digital payment systems with revocable anonymity (e.g., [12, 26]), verifiable signature sharing (e.g., [27]), (publicly) verifiable secret sharing (e.g., [40]), escrow schemes [35, 41], confirmer signatures (e.g., [13]), anonymous voting protocols (e.g., [7, 29]), group signature and identity escrow schemes (e.g., [4, 31]), and fair exchange of signatures (e.g., [1, 3, 5]).

*IBM Research, Zurich Research Laboratory, CH–8803 Rüschlikon, jca@zurich.ibm.com
†Courant Institute, New York University, shoup@cs.nyu.edu

In this paper we consider the problem of verifiable encryption of discrete logarithms. This is basically a protocol that allows one player to encrypt a discrete logarithm of some group element under the public key of a third party and prove to a second party that the ciphertext will indeed decrypt as advertised. In most applications it is important that the encryption scheme provides chosen ciphertext security [37], as the third party itself is basically a decryption oracle. This requirement is often overlooked in the literature.

Up until now, the only known way to build a verifiable encryption protocol for discrete logarithms in conjunction with any known chosen ciphertext secure encryption scheme was via the relatively inefficient cut-and-choose paradigm (e.g., [1, 10]). However, it was known [11, 32] how to avoid the cut-and-choose paradigm if one was willing to accept a weaker form of security, namely semantic security [28].

In this paper, we present a variant of the new public key encryption of Cramer and Shoup [22] based on Paillier's decision composite residuosity assumption [34], along with an efficient protocol for verifiable encryption of discrete logarithms. This is the first such verifiable encryption system that provides chosen ciphertext security and avoids inefficient cut-and-choose proofs. We note that many of the example applications mentioned above rely on verifiable encryption of discrete logarithms, and hence our construction almost immediately yields significantly better efficiency and/or security properties for these applications.

In addition to verifiable encryption schemes, we consider verifiable *decryption* schemes. It appears that such schemes have not been formally studied before. We present two types of schemes. In the first, the decryptor is presented with a ciphertext and a plaintext, and states and proves whether the ciphertext decrypts to the given plaintext. In the second, the decryptor is presented with a ciphertext and a group element, and states and proves whether the ciphertext decrypts to the discrete logarithm of the given group element.

Besides providing a very efficient mechanism for reducing the trust placed in and increasing the accountability of the decryptor, these protocols can also be used to implement confirmer signatures [17], as we shall discuss briefly. In fact, the resulting confirmer signature scheme is the most efficient known scheme providing perfect convertibility of confirmer signatures into ordinary signatures such as DSS or Schnorr.

We note that we do not rely on the random oracle model [6] in any of our proofs of security. However, we also note that there are no previously known protocols which achieve our goals of efficiency and security even using the random oracle model.

Technical contributions. One technical challenge in developing our schemes was to modify the Cramer-Shoup encryption scheme so as to make it amenable to verifiable encryption and decryption. Dealing properly with elements of order two in the group $\mathbb{Z}_{n^2}^*$ turns out to be a rather subtle problem. In designing our verifiable decryption schemes, we have developed some protocols of independent interest. One is a zero-knowledge proof system for proving that that two discrete logarithms are not equal, where the prover only needs to know one of the discrete logarithms. The simplest form of the protocol works in a group of prime order group. These protocols have application in confirmer signature schemes and in undeniable signature scheme to disavow/confirm signatures. We also generalize our proof system to work in subgroups of $\mathbb{Z}_{n^2}^*$ — maintaining zero-knowledge in this setting turns out to be rather challenging.

2 Preliminaries

2.1 Notation

Let a be a real number. We denote by $\lfloor a \rfloor$ the largest integer $b \leq a$, by $\lceil a \rceil$ the smallest integer $b \geq a$, and by $\lceil a \rfloor$ the largest integer $b \leq a + 1/2$. For positive real numbers a and b, let $[a]$ denote the set $\{0, \ldots, \lfloor a \rfloor - 1\}$ and $[a, b]$ denote the set $\{\lfloor a \rfloor, \ldots, \lfloor b \rfloor\}$ and $[-a, b]$ denote the set $\{-\lfloor a \rfloor, \ldots, \lfloor b \rfloor\}$.

Let a, b, and c be integers, with $b > 0$. Most of the time, we use least non-negative remainders, i.e., $c = a \bmod b$ is $a - \lfloor a/b \rfloor b$ and we have $0 \leq c < b$. Sometimes, we have to compute balanced remainders, i.e., $c = a \operatorname{rem} b$ is $a - \lceil a/b \rfloor b$ and we have $-b/2 \leq c < b/2$. Moreover, if b is odd, then $-(b-1)/2 \leq a \operatorname{rem} b \leq (b-1)/2$ for all a.

By $\mathrm{neg}(\lambda)$ we denote a function for which $\mathrm{neg}(\lambda) < 1/p(\lambda)$ holds for all polynomials $p(\lambda)$ and all sufficiently large λ.

We use notation introduced by Camenisch and Stadler [14] for the various zero-knowledge proofs of knowledge of discrete logarithms and proofs of the validity of statements about discrete logarithms. For instance,

$$PK\{(a, b, c) : y = g^a h^b \ \wedge \ \mathfrak{y} = \mathfrak{g}^a \mathfrak{h}^c \ \wedge \ (u \leq a \leq v)\}$$

denotes a "*zero-knowledge* P*roof of* K*nowledge of integers* a*,* b*, and* g *such that* $y = g^a h^b$ *and* $\mathfrak{y} = \mathfrak{g}^a \mathfrak{h}^c$ *holds, where* $v < a < u$," where $y, g, h, \mathfrak{y}, \mathfrak{g}$, and $\mathfrak{h}$ are elements of some groups $G = \langle g \rangle = \langle h \rangle$ and $\mathfrak{G} = \langle \mathfrak{g} \rangle = \langle \mathfrak{h} \rangle$. The convention is that the elements listed in the round brackets denote quantities the knowledge of which is being proved (and are in general not known to the verifier), while all other parameters are known to the verifier. Using this notation, a proof-protocol can be described by just pointing out its aim while hiding all details.

2.2 Special honest-verifier Zero-Knowledge Protocols

A special honest-verifier zero-knowledge protocol is a protocol between a prover and a verifier, where y is their common input and x is the prover's additional input. The protocol is restricted to three moves: in the first move the prover sends the verifier a "commitment" message t, in the second move the verifier sends the prover a "challenge" message c, and in the third move the prover sends the verifier a "response" message s. Finally there must exist a simulator that, on input y, any "challenge" message $\tilde{c}$, outputs a "commitment" and "response" messages $\tilde{t}$ and $\tilde{s}$ such that the distribution of the triple $(\tilde{t}, \tilde{c}, \tilde{s})$ is (statistically) indistinguishable from the one of triples (t, c, s) stemming from real conversations of the prover and the verifier for which $c = \tilde{c}$. Note that the existence of such a simulator implies that the protocol is (ordinary) honest-verifier zero-knowledge.

For particular types of proof systems, we shall give explicit, detailed definitions of special honest-verifier zero knowledge, as appropriate.

While this notion of zero-knowledge is not sufficient for most applications, there exist a number of generic constructions to turn a special honest-verifier zero-knowledge protocol into one that satisfies stronger notions of zero-knowledge. The most important examples are probably the constructions to obtain concurrent zero-knowledge protocols [23, 25, 15] or witness-hiding protocols [19]. In particular, the construction due to Damgård achieves (concurrent) zero-knowledge virtually for free [23].

2.3 Secure Public-Key Encryption

Here, we recall the notion of a public-key encryption scheme. Actually, we need the notion of a public-key encryption scheme that supports *labels*. A label is an arbitrary bit string that is input to the encryption and decryption algorithms, specifying the "context" in which the encryption or decryption operation is to take place.

A public key encryption scheme provides three algorithms:

- a probabilistic, polynomial-time *key generation* algorithm $\mathcal{G}$ that on input 1^λ — where $\lambda \geq 0$ is a security parameter — outputs a public-key/private-key pair $(\mathsf{PK}, \mathsf{SK})$. A public key PK specifies a finite, easy-to-recognize *message space* M_{PK}.
- a probabilistic, polynomial-time *encryption* algorithm $\mathcal{E}$ that takes as input a public key PK, a message $m \in M_{\mathsf{PK}}$, and a label L, and outputs a ciphertext ψ.
- a deterministic, polynomial-time *decryption* algorithm $\mathcal{D}$ that takes as input a private key SK, a ciphertext ψ, a label L, and outputs either a message $m \in M_{\mathsf{PK}}$, where PK is the public-key corresponding to SK, or a special symbol reject.

Any public-key encryption scheme should satisfy a "correctness" or "soundness" property, which loosely speaking means that the decryption operation "undoes" the encryption operation. For our purposes, we can formulate this as follows. We call a public-key encryption scheme *sound* if for all $(\mathsf{PK}, \mathsf{SK}) \in \mathbf{G}(1^\lambda)$, for all $m \in M_{\mathsf{PK}}$, for all $L \in \{0,1\}^*$, and for all $\psi \in \mathcal{E}(\mathsf{PK}, m, L)$, we have $\mathcal{D}(\mathsf{SK}, \psi, L) = m$.

This definition can easily be relaxed to allow for an incorrect decryption with negligible probability, but we do not pursue this matter here. For all encryption schemes presented in this paper, it is trivial to verify this soundness property, and so we will not explicitly deal with this issue again.

We say that a ciphertext is *valid* w.r.t. a label L (and a key pair $(\mathsf{PK}, \mathsf{SK})$) if the decryption algorithm does not reject it and is *invalid* w.r.t. L otherwise.

Note that in this paper, we only work with finite message spaces.

2.4 Adaptive Chosen Ciphertext Security

Consider a public-key encryption scheme, and consider the following game, played against an arbitrary probabilistic, polynomial-time adversary.

1. *Key-Generation Phase.* Let $\lambda \geq 0$ be the security parameter. We run the key-generation algorithm of the public-key encryption scheme on input 1^λ, and get a key pair $(\mathsf{PK}, \mathsf{SK})$. We equip an *encryption oracle* with the public key PK, and a *decryption oracle* with the secret key SK. The public-key PK is presented to the adversary.
2. *Probing Phase I.* In this phase, the attacker gets to interact with the decryption oracle in an arbitrary, adaptive fashion. This phase goes on for a polynomial amount of time, specified by the adversary. More precisely, in each round of this interaction, the adversary sends a *query* (ψ, L) to the decryption oracle. A query is a pair of bit strings chosen in an arbitrary way by the adversary. The decryption oracle in turn decrypts ψ with label L under the secret key SK, and responds to the query by returning the decryption to the adversary.
3. *Target-Selection Phase.* The adversary selects two messages m_0 and m_1 from the message space, along with a label L^*, and presents (m_0, m_1, L^*) to the encryption oracle. The encryption oracle selects a random $\sigma \in \{0,1\}$, and encrypts m_σ with label L^* under PK. The resulting encryption ψ^*, the *target ciphertext*, is presented to the adversary.

4. *Probing Phase II.* This phase is as Probing Phase I, the only difference being that the decryption oracle only responds to queries (ψ, L) with $(\psi, L) \neq (\psi^*, L^*)$.

5. *Guessing-Phase.* The adversary outputs a bit $\hat{\sigma}$.

The adversary is said to *win* the game if $\hat{\sigma} = \sigma$. We define the *advantage* (over random guessing) of the adversary as the absolute value of the difference of the probability that he wins and 1/2.

A public key encryption scheme is said to be *secure against adaptive chosen ciphertext attack* if for all polynomial time, probabilistic adversaries, the advantage in this guessing game is negligible as a function of the security parameter.

3 The Encryption Scheme

3.1 Background

Let p, q, p', q' be distinct odd primes with $p = 2p' + 1$ and $q = 2q' + 1$, and where p' and q' are both ℓ bits in length. Let $n = pq$ and $n' = p'q'$. Consider the group $\mathbb{Z}_{n^2}^*$ and the subgroup $\mathbf{P}$ of $\mathbb{Z}_{n^2}^*$ consisting of all nth powers of elements in $\mathbb{Z}_{n^2}^*$.

Paillier's Decision Composite Residuosity (DCR) assumption [34] is that given only n, it is hard to distinguish random elements of $\mathbb{Z}_{n^2}^*$ from random elements of $\mathbf{P}$.

To be completely formal, one should specify specify a sequence of bit lengths $\ell(\lambda)$, parameterized by a security parameter $\lambda \geq 0$, and to generate an instance of the problem for security parameter λ, the primes p' and q' should be distinct, random primes of length $\ell = \ell(\lambda)$, such that $p = 2p' + 1$ and $q = 2q' + 1$ are also primes.

The primes p' and q' are called Sophie Germain primes and the primes p and q are called safe primes. It has never been proven that there are infinitely many Sophie Germain primes. Nevertheless, it is widely conjectured, and amply supported by empirical evidence, that the probability that a random ℓ-bit number is Sophie Germain prime is $\Omega(1/\ell^2)$. We shall assume that this conjecture holds, so that we can assume that problem instances can be efficiently generated.

Note that Paillier did not make the restriction to strong primes in originally formulating the DCR assumption. As will become evident, we need to restrict ourselves to strong primes for technical reasons. However, it is easy to see that the DCR assumption without this restriction implies the DCR assumption with this restriction, assuming that strong primes are sufficiently dense, as we are here.

We can decompose $\mathbb{Z}_{n^2}^*$ as an internal direct product

$$\mathbb{Z}_{n^2}^* = \mathbf{G}_n \cdot \mathbf{G}_{n'} \cdot \mathbf{G}_2 \cdot \mathbf{T},$$

where each group $\mathbf{G}_\tau$ is a cyclic group of order τ, and $\mathbf{T}$ is the subgroup of $\mathbb{Z}_{n^2}^*$ generated by $(-1 \bmod n^2)$. This decomposition is unique, except for the choice of $\mathbf{G}_2$ (there are two possible choices). For any $x \in \mathbb{Z}_{n^2}^*$, we can express x uniquely as $x = x(\mathbf{G}_n)x(\mathbf{G}_{n'})x(\mathbf{G}_2)x(\mathbf{T})$, where for each $\mathbf{G}_\tau$, $x(\mathbf{G}_\tau) \in \mathbf{G}_\tau$, and $x(\mathbf{T}) \in \mathbf{T}$.

Note that the element $h = (1 + n \bmod n^2) \in \mathbb{Z}_{n^2}^*$ has order n, i.e., it generates $\mathbf{G}_n$, and that $h^a = (1 + an \bmod n^2)$ for $0 \leq a < n$. Observe that $\mathbf{P} = \mathbf{G}_{n'}\mathbf{G}_2\mathbf{T}$.

3.2 The Scheme

For a security parameter $\lambda \geq 0$, $\ell = \ell(\lambda)$ is an auxiliary parameter.

The scheme makes use of a keyed hash scheme $\mathcal{H}$ that uses a key hk, chosen at random from an appropriate key space associated with the security parameter λ; the resulting hash function $\mathcal{H}_{\mathsf{hk}}(\cdot)$ maps a triple (u, e, L) to a number in the set $[2^\ell]$. We shall assume that $\mathcal{H}$ is collision resistant, i.e., given a randomly chosen hash key hk, it is computationally infeasible to find two triples $(u, e, L) \neq (u', e', L')$ such that $\mathcal{H}_{\mathsf{hk}}(u, e, L) = \mathcal{H}_{\mathsf{hk}}(u', e', L')$.

Let $\mathrm{abs} : \mathbb{Z}_{n^2}^* \to \mathbb{Z}_{n^2}^*$ map $(a \bmod n^2)$, where $0 < a < n^2$, to $(n^2 - a \bmod n^2)$ if $a > n^2/2$, and to $(a \bmod n^2)$, otherwise. Note that $v^2 = (\mathrm{abs}(v))^2$ holds for all $v \in \mathbb{Z}_{n^2}^*$.

We now describe the key generation, encryption, and decryption algorithms of the encryption scheme, as they behave for a given value of the security parameter λ.

Key Generation. Select two random ℓ-bit Sophie Germain primes p' and q', with $p' \neq q'$, and compute $p := (2p' + 1)$, $q := (2q' + 1)$, $n := pq$, and $n' := p'q'$, where $\ell = \ell(\lambda)$ is an auxiliary security parameter. Choose random x_1, x_2, $x_3 \in_R [n^2/4]$, choose a random $g' \in_R \mathbb{Z}_{n^2}^*$, compute $g := (g')^{2n}$, $y_1 := g^{x_1}$, $y_2 := g^{x_2}$, and $y_3 := g^{x_3}$. Also, generate a hash key hk from the key space of the hash scheme $\mathcal{H}$ associated with the security parameter λ. The public key is $(\mathsf{hk}, n, g, y_1, y_2, y_3)$. The secret key is $(\mathsf{hk}, n, x_1, x_2, x_3)$.

In the rest of the paper, let $h = (1 + n \bmod n^2) \in \mathbb{Z}_{n^2}^*$, which as discussed above, is an element of order n.

Encryption. To encrypt a message $m \in [n]$ with label $L \in \{0,1\}^*$ under a public key as above, choose a random $r \in_R [n/4]$ and compute

$$u := g^r \ , \qquad e := y_1^r h^m \ , \quad \text{and} \qquad v := \mathrm{abs}\left((y_2 y_3^{\mathcal{H}_{\mathsf{hk}}(u,e,L)})^r \right) \ .$$

The ciphertext is (u, e, v).

Decryption. To decrypt a ciphertext $(u, e, v) \in \mathbb{Z}_{n^2}^* \times \mathbb{Z}_{n^2}^* \times \mathbb{Z}_{n^2}^*$ with label L under a secret key as above, first check that $\mathrm{abs}(v) = v$ and $u^{2(x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)x_3)} = v^2$. If this does not hold, then output reject and halt. Next, let $t = 2^{-1} \bmod n$, and compute $\hat{m} := (e/u^{x_1})^{2t}$. If $\hat{m}$ is of the form h^m for some $m \in [n]$, then output m; otherwise, output reject.

This scheme differs from the DCR-based schemes presented in [22], because in our situation, special attention must be paid to the treatment of elements of order 2 in the $\mathbb{Z}_{n^2}^*$, as these can cause some trouble for the proof systems we discuss in the next sections. Because of these differences, the above encryption scheme does not exactly fit into the general framework of [22], even though the basic ideas are the same. We therefore analyze the security of the scheme starting from first principles, rather than trying to modify their framework.

Before presenting the security analysis, we remark on one of the more peculiar aspects of the scheme, namely, the role of the $\mathrm{abs}(\cdot)$ function in the encryption and decryption algorithms. If one left this out, i.e., replaced $\mathrm{abs}(\cdot)$ by the identity function, then the scheme would be malleable, as (u, e, v) is an encryption of some message m with label L, then so is $(u, e, -v)$. This particular type of malleability [30, 38] is in fact rather "benign," and would be acceptable in most applications. However, we prefer to achieve non-malleability in the strictest sense, and because this comes at a marginal cost, we do so.

Theorem 1. *The above scheme is secure against adaptive chosen ciphertext attack provided the DCR assumption holds, and provided $\mathcal{H}$ is collision resistant.*

The rest of this section is devoted to the proof of Theorem 1.

Let us fix a value of the security parameter λ, which fixes $\ell = \ell(\lambda)$, and let us fix an adversary A. Let $\psi^* = (u^*, e^*, v^*)$ denote the target ciphertext, and L^* the associated label.

We prove this theorem by analyzing a sequence of modifications to the environment in which the adversary runs. We refer to the attack game run with the original environment as Game 0, and to the attack game run with subsequent modifications to the environment as Games 1, 2, etc. Each of these games are best viewed as operating on the same underlying probability space. The value of the random variable σ is identical in each game, but the output $\hat{\sigma}$ of the adversary may vary among games. We define the event T_i, for $i \geq 0$, as the event that the $\sigma = \hat{\sigma}$ in Game i.

Game 1. This is the same as Game 0, except for the following modification to the decryption oracle. If the decryption oracle is invoked in Probing Phase II with a ciphertext/label pair $((u, e, v), L)$ such that $(u, e, L) \neq (u^*, e^*, L^*)$ but $\mathcal{H}_{\mathsf{hk}}(u, e, L) = \mathcal{H}_{\mathsf{hk}}(u^*, e^*, L^*)$, then the decryption oracle *rejects* the ciphertext.

Let F_1 be the event that a ciphertext is rejected in Game 1 using the above rejection rule. It is clear that Games 0 and 1 proceed identically until F_1 occurs; more precisely, the events $T_1 \wedge \neg F_1$ and $T_0 \wedge \neg F_1$ are identical. Therefore,

$$|\Pr[T_1] - \Pr[T_0]| \leq \Pr[F_1]. \tag{1}$$

Moreover, we have

$$\Pr[F_1] \leq \mathsf{AdvCRHF}_{A'}(\lambda), \tag{2}$$

where $\mathsf{AdvCRHF}_{A'}(\lambda)$ denotes the success probability that a particular adversary A' has in finding a collision in $\mathcal{H}$ for the given value of the security parameter λ. The running time of A' is about the same as that of A. Indeed, given a hash key hk, adversary A' simply runs Game 1, using the given value of hk in the key generation algorithm, and when F_1 occurs, A' outputs (u, e, L) and (u^*, e^*, L^*).

Game 2. This game is the same as Game 1, except for the following modification to the decryption oracle. If the decryption oracle is invoked in Probing Phase II with a ciphertext (u, e, v) such that $v^2 = (v^*)^2$ and $v \neq v^*$, then the decryption oracle *rejects* the ciphertext.

Let F_2 be the event that a ciphertext is rejected in Game 2 using the above rejection rule, but would not have been rejected for any other reason. It is clear that Games 1 and 2 proceed identically until F_2 occurs; more precisely, the events $T_2 \wedge \neg F_2$ and $T_1 \wedge \neg F_2$ are identical. Therefore,

$$|\Pr[T_2] - \Pr[T_1]| \leq \Pr[F_2]. \tag{3}$$

Moreover, we have

$$\Pr[F_2] \leq \mathsf{AdvFactor}_{A''}(\lambda), \tag{4}$$

where $\mathsf{AdvFactor}_{A''}(\lambda)$ denotes the success probability that a particular algorithm A'' has in factoring a number n as generated by the encryption algorithm for the given value of the security parameter λ. The running time of A'' is about the same as that of A. Algorithm A'' takes the given number n, constructs the remaining components of the public key, and then lets adversary A run in Game 2. If and when event F_2 occurs, we have $v^2 = (v^*)^2$, $v \neq v^*$, $\mathrm{abs}(v) = v$, and $\mathrm{abs}(v^*) = v^*$. This implies that $v \neq \pm v^*$. It follows that if $v/v^* = (a \bmod n^2)$, then $\gcd(a, n)$ splits n.

Game 3. This game is the same as Game 2, except for the following modification to the encryption oracle. Instead of computing e^* and v^* as in the encryption algorithm, we compute them using the

secret key, as follows:

$$\begin{aligned} e^* &:= (u^*)^{x_1} h^{m_\sigma} \\ v^* &:= \mathrm{abs}\left((u^*)^{x_2 + \mathcal{H}_{\mathsf{hk}}(u^*, e^*, L^*) x_3}\right) \end{aligned}$$

This modification is purely conceptual, since the values of e^* and v^* computed by the encryption oracle in Game 3 are identical to those computed in Game 2. Therefore,

$$\Pr[T_3] = \Pr[T_2]. \tag{5}$$

Game 4**.** Now we further modify the encryption oracle. Let r^* denote the value of r generated by the encryption oracle. Then, instead of computing u^* as g^{r^*}, the encryption oracle in this game chooses a random $\bar{u} \in \mathbf{P}$, and sets $u^* := \bar{u}^2$.

We claim that

$$|\Pr[T_4] - \Pr[T_3]| = O(2^{-\ell}). \tag{6}$$

To see this, observe that $\bar{u}^2$ is uniformly distributed over $\mathbf{G}_{n'}$. Also, observe that with probability $1 - O(2^{-\ell})$, g is a generator for $\mathbf{G}_{n'}$, and that the distribution of r^* is $O(2^{-\ell})$-close to the uniform distribution on $[n']$. It is an easy exercise to show that the bound (6) follows from these observations.

Game 5**.** We again modify the encryption oracle. Instead of choosing $\bar{u}$ at random from $\mathbf{P}$, the encryption oracle chooses $\bar{u}$ at random from $\mathbb{Z}^*_{n^2}$; otherwise, the computation is identical to that of Game 4.

It is clear that any significant difference between $\Pr[T_5]$ and $\Pr[T_4]$ leads immediately to an effective statistical test for distinguishing $\mathbf{P}$ from $\mathbb{Z}^*_{n^2}$. More precisely, there exists an adversary A''', whose running time is roughly the same as that of A, such that

$$|\Pr[T_5] - \Pr[T_4]| \leq \mathsf{AdvDCR}_{A'''}(\lambda), \tag{7}$$

where $\mathsf{AdvDCR}_{A'''}(\lambda)$ denotes the advantage that A''' has in distinguishing $\mathbf{P}$ from $\mathbb{Z}^*_{n^2}$ for the given value of the security parameter λ.

Game 6**.** We again modify the encryption oracle. This time, we replace u^* by a random element of $\mathbf{G}_n \mathbf{G}_{n'}$ such that $u^*(\mathbf{G}_n)$ has order n.

We claim that

$$|\Pr[T_6] - \Pr[T_5]| = O(2^{-\ell}). \tag{8}$$

To see this, note that in Game 5, u^* is uniformly distributed over $\mathbf{G}_n \mathbf{G}_{n'}$, and so $u^*(\mathbf{G}_n)$ has order n with probability $1 - O(2^{-\ell})$. The bound (8) follows immediately.

Game 7**.** Now we modify the key generation algorithm. Instead of choosing x_1, x_2, x_3 at random from $[n^2/4]$, we choose them at random from $[nn']$.

Since the uniform distribution on $[n^2/4]$ is $O(2^{-\ell})$-close to the uniform distribution on $[nn']$, it follows immediately that

$$|\Pr[T_7] - \Pr[T_6]| = O(2^{-\ell}). \tag{9}$$

Game 8**.** Now we modify the decryption oracle. In this game, in addition to rejecting a ciphertext $(u, e, v) \in \mathbb{Z}^*_{n^2} \times \mathbb{Z}^*_{n^2} \times \mathbb{Z}^*_{n^2}$ with label L if $u^{2(x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L) x_3)} \neq v^2$, the decryption oracle also rejects this ciphertext if $u \notin \mathbf{G}_{n'} \mathbf{G}_2 \mathbf{T}$.

In this game, the decryption oracle leaks no information about the value of x_1 modulo n. ¿From this, and the fact that $u^*(\mathbf{G}_n)$ has order n and $e^* = (u^*)^{x_1} h^{m_\sigma}$, it follows that A's output $\hat{\sigma}$ is independent of σ. Therefore,

$$\Pr[T_8] = 1/2. \tag{10}$$

Let F_8 be the event that in Game 8, some ciphertext (u, e, v) with label L is rejected using the special rejection rule introduced in Game 8, but would not have been rejected for any other reason, i.e., the special rejection rules introduced in Games 1 and 2 do not apply, and $u^{2(x_2+\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3)} = v^2$.

It is clear that Games 7 and 8 proceed identically until F_8 occurs. More precisely, the events $T_8 \wedge \neg F_8$ and $T_7 \wedge \neg F_8$ are identical. Therefore,

$$|\Pr[T_8] - \Pr[T_7]| \leq \Pr[F_8]. \tag{11}$$

Let $\kappa = \kappa(\lambda)$ denote an upper bound on the number of decryption oracle queries made by A for the given value of the security parameter λ. We assume this bound holds, regardless of the environment in which A runs. We claim that

$$\Pr[F_8] \leq \kappa \cdot 2^{-\ell}. \tag{12}$$

To prove (12), we argue as follows. Let $\bar{x}_2$ and $\bar{x}_3$ denote the values of x_2 and x_3, respectively, modulo n. Similarly, let $\bar{x}'_2$ and $\bar{x}'_3$ denote the values of x_2 and x_3, respectively, modulo n'.

Let us condition on fixed values of

$$n, g, x_1, \bar{x}'_2, \bar{x}'_3, \mathsf{hk},$$

as well as fixed values of the coin tosses of A. In this conditional probability space, the public key is fixed, A's queries to the decryption oracle in Probing Phase I, as well as the responses of the decryption oracle. To see why responses of the decryption oracle are fully determined, observe that all ciphertexts (u, e, v) with $u \notin \mathbf{G}_{n'}\mathbf{G}_2\mathbf{T}$ are rejected, and that the decryption oracle squares u in all computations involving u; thus, the response of the decryption oracle is determined by $\bar{x}'_2$ and $\bar{x}'_3$, which are fixed. Also, in this conditional probability space, it is determined whether or not A invokes the encryption oracle, and if so, A's inputs to the encryption oracle. However, by the Chinese Remainder Theorem, the values of $\bar{x}_2$ and $\bar{x}_3$ in this conditional probability space are still uniformly and independently distributed over $[n]$.

In this conditional probability space, consider a particular invocation of the decryption oracle in Probing Phase I with a ciphertext (u, e, v) and label L. Suppose that $u \notin \mathbf{G}_{n'}\mathbf{G}_2\mathbf{T}$. Let $\bar{u} = u(\mathbf{G}'_n)^2$, $\bar{u}' = u(\mathbf{G}_n)^2$, and $H = \mathcal{H}_{\mathsf{hk}}(u, e, L)$. Note that $\bar{u} \neq 1$, and so $\bar{u}$ has order p, q, or n. Now, we have

$$u^{2(x_2+\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3)} = (\bar{u})^{\bar{x}_2+H\bar{x}_3}(\bar{u}')^{\bar{x}'_2+H\bar{x}'_3}.$$

It follows that $u^{2(x_2+\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3)}$ is uniformly distributed over a particular coset in $\mathbf{G}_{n'}\mathbf{G}_n$ of the subgroup generated by $\bar{u}$. Since v^2 is fixed in this conditional probability space, it follows that $u^{2(x_2+\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3)} = v^2$ with probability at most $2^{-\ell}$.

Now suppose that in this conditional probability space A invokes the encryption oracle with particular messages m_0 and m_1, and a label L^*. Let us further condition on fixed values of σ and u^*. This determines the value of e^*, and also the value of $H^* = \mathcal{H}_{\mathsf{hk}}(u^*, e^*, L^*)$. Let us also further condition a fixed value of $\bar{x}_2 + H^*\bar{x}_3$ modulo n. This determines the value v^*. In the resulting conditional probability space, the output of the encryption oracle, as well as all queries and responses of decryption oracle queries in Probing Phase II are completely determined.

In this conditional probability space, consider a particular invocation of the decryption oracle in Probing Phase II with a ciphertext (u, e, v) and label L, such that $(u, e, v, L) \neq (u^*, e^*, v^*, L)$. Suppose that $u \notin \mathbf{G}_{n'}\mathbf{G}_2\mathbf{T}$, and that the special rejection rules introduced in Games 1 and 2 do not apply. We consider two cases.

Case 1: $(u, e, L) = (u^*, e^*, L^*)$. We must have $v \neq v^*$, since $(u, e, v, L) \neq (u^*, e^*, v^*, L)$. Since the special rejection rule in Game 2 does not apply, we must have $v^2 \neq (v^*)^2$, which implies that $u^{2(x_2+\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3)} \neq v^2$.

Case 2: $(u, e, L) = (u^*, e^*, L^*)$. Since the special rejection rule in Game 1 does not apply, we must have $H \neq H^*$. By the definition of $\mathcal{H}$, this implies that $H \not\equiv H^* \pmod{p}$ and $H \not\equiv H^* \pmod{q}$. This in turn implies that in this conditional probability space, the distribution of $\bar{x}_2 + H\bar{x}_3$ modulo n is uniform. It follows that $u^{2(x_2+\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3)}$ is uniformly distributed over a particular coset in $\mathbf{G}_{n'}\mathbf{G}_n$ of the subgroup generated by $\bar{u}$. Since v^2 is fixed in this conditional probability space, it follows that $u^{2(x_2+\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3)} = v^2$ with probability at most $2^{-\ell}$.

The above arguments show that the event F_8 occurs for a particular decryption query with probability at most $2^{-\ell}$. The bound (12) now follows.

Putting together (1)-(12), we have

$$|\Pr[T_0] - 1/2| \leq \mathsf{AdvCRHF}_{A'}(\lambda) + \mathsf{AdvFactor}_{A''}(\lambda) + \mathsf{AdvDCR}_{A'''}(\lambda) + \kappa \cdot 2^{-\ell} + O(2^{-\ell}).$$

Theorem 1 now follows immediately.

3.3 Extensions to Threshold Decryption

Our scheme can easily be transformed to provide threshold decryption, where it comes in handy that the knowledge of the factorization of n is not required for decryption. This allows one to reduce the trust assumption for the TTP. This can be done either along the lines in [39], which requires a random oracle security argument, or along the lines in [16], which does not require that argument, but for which the decryption protocol is less efficient.

4 Verifiable Encryption

Loosely speaking, verifiable encryption for a relation $\mathcal{R}$ is a protocol that allows a prover to convince a verifier that the ciphertext is an encryption under a given public key of a value w such that $(\delta, w) \in \mathcal{R}$ for a given δ.

Asokan et al. [1, 2] present a protocol for verifiable encryption for the case where w is a homomorphic pre-image of δ and Camenisch and Damgård [10] present a protocol that works for any relation $\mathcal{R}$ that has a three-move honest-verifier zero-knowledge proof of knowledge where the verifier sends as a second message a random challenge. Both these protocols work for any secure public key encryption scheme. However, they are based on the cut-and-choose paradigm and hence are rather impractical.

In this section we present an efficient verifiable encryption protocol for discrete logarithms in conjunction with the encryption scheme presented in the previous section. We then discuss extensions of this protocol.

4.1 Definition of Verifiable Encryption

Previous definitions [1, 2, 10] for verifiable encryption did not distinguish between the processes of encryption and and of proving that the "right thing" was encrypted. In fact, the protocols proposed in these papers generate the ciphertext as "by-product" of the proof. Thus, with these protocols it is for instance not possible to later on prove statements about the generated ciphertext to other verifiers without resorting to the inefficient general zero-knowledge proof techniques. Similarly, efficient verifiable decryption seems not to be achievable for these schemes.

We are interested in a two-stage method where the encryptor first generates a ciphertext and then proves properties about the ciphertext. This is more modular and seems to be an attractive feature for a cryptographic primitive. Before stating the formal definition of verifiable encryption, we begin with a high level discussion of what we are after, along with some auxiliary definitions.

Let $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ be a public key encryption scheme, and suppose we have generated a key pair $(\mathsf{PK}, \mathsf{SK})$.

A verifiable encryption scheme proves that a ciphertext encrypts a plaintext satisfying a certain relation $\mathcal{R}$. The relation $\mathcal{R}$ is defined by a *generator* algorithm, $\mathcal{G}'$, which on input 1^λ outputs a *description* $\Psi = \Psi[\mathcal{R}, W, \Delta]$ of a binary relation $\mathcal{R}$ on $W \times \Delta$. We require that the sets $\mathcal{R}$,W, and Δ are easy to recognize (given Ψ). For $\delta \in \Delta$, an element $w \in W$ such that $(w, \delta) \in \mathcal{R}$ is called a *witness* for δ. The idea is that the encryptor will be given a value δ, a witness w for δ, and a label L, and then encrypts w under L, yielding a ciphertext ψ. After this, the encryptor may prove to another party that ψ decrypts under L to a witness for δ. In carrying out the proof, the encryptor will of course need to make use of the random coins that were used by the encryption algorithm: we denote by $\mathcal{E}'(\mathsf{PK}, m, L)$ the pair (ψ, coins), where ψ is the output of $\mathcal{E}(\mathsf{PK}, m, L)$ and *coins* are the random coins used to compute ψ.

In such a proof system, the (honest) verifier will output 0 or 1, with 1 signifying "accept." We of course shall require that the proof system is sound, in the sense that if a verifier accepts a proof, then with overwhelming probability, ψ indeed decrypts under L to a witness for δ. However, it is convenient, and adequate for many applications, to take a more relaxed approach: instead of requiring that ψ decrypts under L to a witness, we only require that a witness can be easily reconstructed from the plaintext using some efficient *reconstruction* algorithm. Such an algorithm *recon* takes as input a public key PK, a relation description $\Psi[R, W, \delta]$, an element $\delta \in \Delta$, and a message $m \in M_{\mathsf{PK}} \cup \{\mathsf{reject}\}$, and outputs $w \in W \cup \{\mathsf{reject}\}$.

We need to make some technical "compatibility" requirements: we say that an encryption scheme, a relation generator, and a reconstruction algorithm as above are *mutually compatible* if for all $\lambda \geq 0$, all $(\mathsf{PK}, \mathsf{SK}) \in \mathcal{G}(1^\lambda)$, and all $\Psi[\mathcal{R}, W, \Delta] \in \mathcal{G}'(1^\lambda)$, we have

- $W \subset M_{\mathsf{PK}}$, and
- for all $(w, \delta) \in \mathcal{R}$, we have $\mathit{recon}(\mathsf{PK}, \Psi, \delta, w) = w$.

The first requirement simply says that witness "fit" into the message space, and the second requirement simply says that the reconstruction routine does not modify valid witnesses.

We shall also require that the proof system is special honest-verifier zero knowledge. To formulate this more precisely below, we let $\mathit{Trans}(\mathsf{PK}, \Psi, \delta, \psi, L, c, w, \mathit{coins})$ denote the transcript seen by a verifier that uses a *fixed* challenge c.

Definition 1. *A proof system $(\mathcal{P}, \mathcal{V})$, together with mutually compatible encryption scheme $(\mathcal{G}, \mathcal{E}, \mathcal{D})$, relation generator $\mathcal{G}'$, and reconstruction algorithm recon, form a* verifiable encryption scheme*, if the following properties hold.*

Correctness: *for all $(\mathsf{PK}, \mathsf{SK}) \in \mathcal{G}(1^\lambda)$, for all $\Psi[\mathcal{R}, W, \Delta] \in \mathcal{G}'(1^\lambda)$, for all $(w, \delta) \in \mathcal{R}$, for all $L \in \{0,1\}^*$, for all $(\psi, \mathit{coins}) \in \mathcal{E}'(\mathsf{PK}, w, L)$,*

$$\Pr[x \leftarrow \mathcal{V}(\mathsf{PK}, \Psi, \delta, \psi, L)_{\mathcal{P}(\mathsf{PK}, \Psi, \delta, \psi, L, w, \mathit{coins})} : x = 1] = 1 - \mathrm{neg}(\lambda).$$

Soundness: *for all adversaries* $(\mathcal{A}^*, \mathcal{P}^*)$*,*

$$\begin{array}{lll}\Pr[& (\mathsf{PK},\mathsf{SK}) \leftarrow \mathcal{G}(1^\lambda); \Psi[\mathcal{R}, W, \Delta] \leftarrow \mathcal{G}'(1^\lambda); & \\ & (\delta, \psi, L, aux) \leftarrow \mathcal{A}^*(\mathsf{PK},\mathsf{SK},\Psi); & \\ & x \leftarrow \mathcal{V}(\mathsf{PK},\Psi,\delta,\psi,L)_{\mathcal{P}^*(aux)}; & \\ & m \leftarrow \mathcal{D}(\mathsf{SK},\psi,L); & \\ & w \leftarrow \mathit{recon}(\mathsf{PK},\Psi,\delta,m): & \\ & x = 1 \ \wedge \ (w,\delta) \notin \mathcal{R} &] \ = \ \mathrm{neg}(\lambda).\end{array}$$

Special honest-verifier zero knowledge: *There exists a simulator Sim such that for all adversaries* $(\mathcal{A}^*, \mathcal{B}^*, \mathcal{C}^*)$*, we have*

$$\begin{array}{lll}\Pr[& (\mathsf{PK},\mathsf{SK}) \leftarrow \mathcal{G}(1^\lambda); \Psi[\mathcal{R}, W, \Delta] \leftarrow \mathcal{G}'(1^\lambda); & \\ & (w, \delta, L, aux) \leftarrow \mathcal{A}^*(\mathsf{PK},\mathsf{SK},\Psi), \ \textit{where } (w,\delta) \in \mathcal{R}; & \\ & (\psi, coins) \leftarrow \mathcal{E}'(\mathsf{PK}, w, L); & \\ & c \leftarrow \mathcal{B}^*(aux, \psi); & \\ & b \leftarrow \{0,1\}; & \\ & \textit{if } b = 0 & \\ & \quad \textit{then } \alpha \leftarrow \mathit{Trans}(\mathsf{PK},\Psi,\delta,\psi,L,c,w,coins) & \\ & \quad \textit{else } \ \alpha \leftarrow \mathit{Sim}(\mathsf{PK},\Psi,\delta,\psi,L,c); & \\ & \hat{b} \leftarrow \mathcal{C}^*(aux,\psi,\alpha): & \\ & b = \hat{b} &] \ = 1/2 + \mathrm{neg}(\lambda).\end{array}$$

The above definitions are fairly traditional. Our formulations of soundness and special honest-verifier zero knowledge are basically of the "computational" variety, but where we have taken the notion of "computational" one step further: instead of universally quantifying over the inputs to the verifier (respectively, simulator), we quantify "computationally." This is technically convenient, and is adequate for most applications.

Also, the above definitions assume that the key for the encryption scheme are generated by a trusted party. However, it is possible to define verifiable encryption in a setting where the keys are not generated by a trusted party; the definitions in this case are a bit more complicated and subtle, and we do not present them here. Nevertheless, our protocols require only slight modification to remain secure in this setting.

4.2 Verifiable Encryption of a Discrete Logarithm

Let $(\mathsf{hk}, n, g, y_1, y_2, y_3)$ be a public key of the encryption scheme provided in §3. Recall that the message space associated with this public key is $[n]$.

Let Γ be a cyclic group of order ρ generated by γ. We assume that γ and ρ are publicly known, and that ρ is prime. Let $W = [\rho]$ and $\Delta = \Gamma$, and let $\mathcal{R} = \{(w,\delta) \in W \times \Delta : \gamma^w = \delta\}$. The "discrete logarithm" relation $\mathcal{R}$ is the relation with respect to which we want to verifiably encrypt.

We shall of course require that $n > \rho$ (in fact, we will make a stronger requirement). The reconstruction routine *recon* will map a plaintext $m \in [n]$ to the integer $(m \operatorname{rem} n) \mod \rho$, i.e., it computes the balanced remainder of m modulo n, and then computes the least non-negative remainder of this modulo ρ.

Setup. Our protocol requires the auxiliary parameters $\mathfrak{n}$, which must the product of two safe $(\mathfrak{l}+1)$-bit primes $\mathfrak{p} = 2\mathfrak{p}' + 1$ and $\mathfrak{q} = 2\mathfrak{q}' + 1$, and $\mathfrak{g}$ and $\mathfrak{h}$, which are two generators of $\mathfrak{G}_{\mathfrak{n}'} \subset \mathbb{Z}^*_{\mathfrak{n}}$,

where $\mathfrak{n}' = \mathfrak{p}'\mathfrak{q}'$; $\mathfrak{G}_{\mathfrak{n}'}$ is the subgroup of $\mathbb{Z}_{\mathfrak{n}}^*$ of order $\mathfrak{n}'$, and $\mathfrak{l} = \mathfrak{l}(\lambda)$. The requirement that $\mathfrak{p}$ and $\mathfrak{q}$ be safe primes is not essential [24], however, it simplifies our presentation.

One may view $\mathfrak{n}$, $\mathfrak{g}$, and $\mathfrak{h}$ as additional components of the public key of the encryption scheme, or as system parameters generated by a trusted party. Depending on the setting, we may simply put $\mathfrak{n} := n$ and $\mathfrak{g} := g$. In any event, the prover should not be privy to the factorization of $\mathfrak{n}$.

Let $k = k(\lambda)$ and $k' = k'(\lambda)$ be further security parameters, where $2^{-k(\lambda)}$ and $2^{-k'(\lambda)}$ are negligible functions ($\{0,1\}^k$ is the "challenge space" of the verifier and k' controls the quality of the zero-knowledge property). We require that $2^k < \min\{p, q, p'q', \mathfrak{p}, \mathfrak{q}, \mathfrak{p}', \mathfrak{q}', \rho\}$ holds. Finally, we require that $\rho < n2^{-k-k'-3}$ holds, i.e., that $\log_\gamma \delta$ "fits into an encryption". (If this condition is not meet, the value $\log_\gamma \delta$ could be split into smaller pieces, each of which would then be verifiably encrypted. However, we do not address this here.)

The protocol. The common input of the prover and verifier is: the public key $(\mathsf{hk}, n, g, y_1, y_2, y_3)$, the augmented public key $(\mathfrak{n}, \mathfrak{g}, \mathfrak{h})$, a group element (δ), a ciphertext (u, e, v), and a label L. The prover has additional inputs $m = \log_\gamma \delta$ and $r \in_R [n/4]$ such that

$$u = g^r, \qquad e = y_1^r h^m, \quad \text{and} \qquad v = \operatorname{abs}\left((y_2 y_3^{\mathcal{H}_{\mathsf{hk}}(u,e,L)})^r\right) .$$

1. The prover chooses a random $s \in_R [\mathfrak{n}/4]$ and computes $\mathfrak{k} := \mathfrak{g}^m \mathfrak{h}^s$. The prover sends $\mathfrak{k}$ to the verifier.
2. Then the prover and verifier engage in the following protocol.
   (a) The prover chooses random
   $$r' \in_R [-n2^{k+k'-2}, n2^{k+k'-2}],\ \ s' \in_R [-\mathfrak{n}2^{k+k'-2}, \mathfrak{n}2^{k+k'-2}],\ \ m' \in_R [-\rho 2^{k+k'}, \rho 2^{k+k'}].$$
   The prover computes
   $$u' := g^{r'},\ \ e' := y_1^{r'} h^{m'},\ \ v' := (y_2 y_3^{\mathcal{H}_{\mathsf{hk}}(u,e,L)})^{r'},\ \ \delta' := \gamma^{m'},\ \text{and}\ \mathfrak{k}' := \mathfrak{g}^{m'} \mathfrak{h}^{s'}.$$
   The prover sends u', e', v', δ', and $\mathfrak{k}'$ to the verifier.
   (b) The verifier chooses a random challenge $c \in_R \{0,1\}^k$ and sends c to the prover.
   (c) The prover replies with $\tilde{r} := r' - cr$, $\tilde{s} := s' - cs$, and $\tilde{m} := m' - cm$ (computed in $\mathbb{Z}$).
   (d) The verifier checks whether the relations
   $$u'^2 = u^{2c} g^{2\tilde{r}}, \qquad e'^2 = e^{2c} y_1^{2\tilde{r}} h^{2\tilde{m}}, \qquad v'^2 = v^{2c} (y_2 y_3^{\mathcal{H}_{\mathsf{hk}}(u,e,L)})^{2\tilde{r}},$$
   $$\delta' = \delta^c \gamma^{\tilde{m}}, \qquad \mathfrak{k}' = \mathfrak{k}^c \mathfrak{g}^{\tilde{m}} \mathfrak{h}^{\tilde{s}}, \quad \text{and} \qquad -n/4 < \tilde{m} < n/4$$
   hold. If any of them does not hold, the verifier stops and outputs 0.
3. If $v = \operatorname{abs} v$ the verifier outputs 1; otherwise she outputs 0.

Using notation from [14] we denote the sub-protocol of step 2 as

$$PK\{(r,m,s) : u^2 = g^{2r} \ \wedge\ e^2 = y_1^{2r} h^{2m} \ \wedge\ v^2 = (y_2 y_3^{\mathcal{H}_{\mathsf{hk}}(u,e,L)})^{2r} \ \wedge\ \delta = \gamma^m \ \wedge\ \mathfrak{k} = \mathfrak{g}^m \mathfrak{h}^s \ \wedge\ -n/2 < m < n/2\} .$$

Proof of Security. We prove the following theorem about the above system. Given this theorem, one can apply the standard constructions (e.g., [23]) to turn the sub-protocol used in Step step:subpk into an efficient one that is zero-knowledge w.r.t. any verifier, and can thus obtain a verifiable encryption system that satisfies computational zero-knowledge.

Theorem 2. *Under the strong RSA assumption, the above system is a verifiable encryption scheme.*

Proof. The correctness and special honest-verifier zero knowledge properties are easy to verify, and we leave this to the reader.

It remains to consider soundness.

If the success-probability of the prover is non-negligible, then there is a knowledge extractor that produces (in time polynomial in λ and with non-negligible probability) two answers $(\tilde{r}^{(1)}, \tilde{s}^{(1)}, \tilde{m}^{(1)})$ $(\tilde{r}^{(2)}, \tilde{s}^{(2)}, \tilde{m}^{(2)})$ from the prover on two different challenges $c^{(1)}$ and $c^{(2)}$ w.r.t. the same u', e', v', δ', and $\mathfrak{k}'$. W.l.o.g., suppose that $c^{(2)} > c^{(1)}$. Let $\Delta r = \tilde{r}^{(1)} - \tilde{r}^{(2)}$, $\Delta s = \tilde{s}^{(1)} - \tilde{s}^{(2)}$, $\Delta m = \tilde{m}^{(1)} - \tilde{m}^{(2)}$, and $\Delta c = c^{(2)} - c^{(1)} > 0$. From the verification equations one can derive the following equations:

$$u^{2\Delta c} = g^{2\Delta r} \qquad e^{2\Delta c} = y_1^{2\Delta r} h^{2\Delta m} \qquad v^{2\Delta c} = (y_2 y_3^{\mathcal{H}_{\mathsf{hk}}(u,e,L)})^{2\Delta r}$$
$$\delta^{\Delta c} = \gamma^{\Delta m} \qquad \mathfrak{k}^{\Delta c} = \mathfrak{g}^{\Delta m} \mathfrak{h}^{\Delta s}$$

Under the strong RSA assumption we can assume that $\Delta c \mid \Delta s$ and $\Delta c \mid \Delta m$ (we will investigate this claim later). By construction we have $|\Delta c| < \min\{p, q, p'q', \mathfrak{p}, \mathfrak{q}, \mathfrak{p}', \mathfrak{q}', \rho\}$ and hence Δc is invertible modulo any of those primes. Let $\hat{c} = \Delta c^{-1} \bmod nn'$. As u^2 has order dividing nn', we get $u^2 = g^{2\Delta r \hat{c}}$, i.e.,

$$u = w_1 g^{\Delta r \hat{c}} \tag{13}$$

for some w_1 of order 2. Similarly, we get

$$e = w_2 y_1^{\Delta r \hat{c}} h^{\Delta m / \Delta c} \tag{14}$$
$$v = w_3 (y_2 y_3^{\mathcal{H}_{\mathsf{hk}}(u,e,L)})^{\Delta r \hat{c}} \tag{15}$$
$$\delta = \gamma^{\Delta m / \Delta c} \tag{16}$$

for some w_2 and w_3 of order 2. It is not hard to see that from $v = \operatorname{abs} v$ and from Eqns. (13)-(15) it follows that decryption of the triple (u, e, v) will provide the integer $\bar{m} := \Delta m / \Delta c \bmod n$ modulo n (note that due to the squarings in the decryption algorithm, all the w_i disappear).

We claim that for $\check{m} = (\bar{m} \operatorname{rem} n) \bmod \rho$ we have $\delta = \gamma^{\check{m}}$, i.e., that (u, e, v) is an encryption of $\log_\gamma \delta$. As $|\tilde{m}_1|, |\tilde{m}_2| < n/4$ and $\Delta c \mid \Delta m$, we must have $|\Delta m / \Delta c| < n/2$. Hence $\Delta m / \Delta c = ((\Delta m / \Delta c \bmod n) \operatorname{rem} n) = \bar{m} \operatorname{rem} n$ and therefore $\delta = \gamma^{\Delta m / \Delta c} = \gamma^{\check{m}}$.

It remains to prove that $\Delta c \mid \Delta m$ holds under the strong RSA assumption. The following arguments are along the lines those found in [21, 24]. We show that if $\Delta c \nmid \Delta m$, then with probability at least about $1/2$, we can compute a non-trivial root of $\mathfrak{g}$. Now, $\mathfrak{g}$ is a random element of order $\mathfrak{n}'$, and as a random number mod $\mathfrak{n}$ is of this form with probability about $1/4$, we can use the prover to compute a non-trivial root of a random number mod $\mathfrak{n}$ with non-negligible probability.

Let us modify the encryption scheme slightly. Let $\mathfrak{g}$ be element of order $\mathfrak{n}'$ as usual, but let us compute $\mathfrak{h} = \mathfrak{g}^a$, for a randomly chosen from a sufficiently large range, e.g., from $[\mathfrak{n}^2]$. This is a negligible change in the distribution of $\mathfrak{h}$. We have

$$\mathfrak{k}^{\Delta c} = \mathfrak{g}^{\Delta m + a \Delta s} \quad .$$

Let us now condition on fixed values of the prover's view and fixed values of the two challenges used by the knowledge extractor. So in particular, we now view Δc, Δm, and Δs as fixed, and we assume that $\Delta c \nmid \Delta m$. Let us write $a = a_2\mathfrak{n}' + a_1$, where $0 \leq a_1 < \mathfrak{n}'$. In this conditional probability space, the value a_1 is also fixed, but the distribution of a_2 is statistically close to the uniform distribution on $[4\mathfrak{n}]$.

Now, consider the congruence

$$\Delta m + a\Delta s \equiv 0 \pmod{\Delta c} \ .$$

This congruence holds if and only if

$$\Delta m + a_1\Delta s + a_2\mathfrak{n}'\Delta s \equiv 0 \pmod{\Delta c} \ .$$

Now, in the conditional probability space, all terms in the above congruence are fixed, except for a_2. We want to bound from above the probability that this equation holds. We may as well assume that $\Delta c \nmid \Delta s$, because if $\Delta c \mid \Delta s$, then because we are assuming that $\Delta c \nmid \Delta m$, the congruence will never hold. Since $\gcd(\Delta c, \mathfrak{n}') = 1$, it follows that the solutions a_2 to the above congruence are uniquely determined modulo $\Delta c/d'$, where $d' = \gcd(\Delta c, \Delta s) \leq \Delta c/2$. Since the distribution of a_2 is statistically close to the uniform distribution on a very large range, it follows that the congruence holds with probability at most about $1/2$.

It is left to show that if $\Delta c \nmid (\Delta m + a\Delta s)$, then a non-trivial root of $\mathfrak{g}$ can be computed. Let $d = \gcd(\Delta c, (\Delta m + a\Delta s)) < 2^k < \min\{p, q, p'q', \mathfrak{p}, \mathfrak{q}, \mathfrak{p}', \mathfrak{q}', \rho\}$. There are integers α and β such that $d = \alpha\Delta c + \beta(\Delta m + a2\Delta s)$ and we have

$$\mathfrak{g}^d = (\mathfrak{k}^\beta \mathfrak{g}^\alpha)^{\Delta c}$$

and so $\mathfrak{g} = \mathfrak{w}(\mathfrak{k}^\beta \mathfrak{g}^\alpha)^{\Delta c/d}$ for some $\mathfrak{w}$ with $\mathfrak{w}^d = 1$. As $d < \mathfrak{p}', \mathfrak{q}'$, the order of $\mathfrak{w}$ is either 1 or 2. So either $\mathfrak{w} = \pm 1$ or $\gcd(\mathfrak{w} - 1, \mathfrak{n})$ splits $\mathfrak{n}$. In the latter case we have factored $\mathfrak{n}$ and can therefore also compute a root of $\mathfrak{g}$. In the former case we can compute such a root as follows. If $\Delta c/d$ is even, then $(\mathfrak{k}^\beta \mathfrak{g}^\alpha)^{\Delta c/d} \in \mathfrak{G}_{\mathfrak{n}'}$ and, because $\mathfrak{g} \in \mathfrak{G}_{\mathfrak{n}'}$, we must have $\mathfrak{w} = 1$ (since $-1 \notin \mathfrak{G}_{\mathfrak{n}'}$). If $\Delta c/d$ is odd then $\mathfrak{g} = (\mathfrak{w}\mathfrak{k}^\beta \mathfrak{g}^\alpha)^{\Delta c/d}$. In both cases we have found a root of $\mathfrak{g}$.

□

4.3 Extensions

It is straightforward to extend the above verifiable encryption scheme to a verifiable encryption scheme that encrypts a (subset of a) representation of a group element with respect to several bases. Further, all of these protocols can be easily adapted to the case where the order of the group Γ is not known, i.e., a subgroup of of $\mathbb{Z}_N^*$ for an RSA-modulus N, provided the order is not divisible by any small primes.

5 Proving the Inequality of Discrete Logarithms

Our protocol for verifiable decryption (below) requires that one party proves to another party whether or not two discrete logarithms are equal, where one of the discrete logarithms might *not be known* to the prover (that is, in the case the discrete logarihtms are not equal). There are well-known, efficient, special honest-verifier zero-knowledge proof systems for proving that two discrete logarithms are equal (see [18]), so we focus on the problem of proving that two discrete logarithms are unequal. We discuss an efficient protocol for this problem separately as it is of independent

interest and as the algebraic setting here is simpler than the one in which we use it on the next section.

Let $G = \langle g \rangle$ be a group of prime order q. The prover and verifier have common inputs $g, h, y, z \in G$, where g and h are generators for G, and $\log_g y \neq \log_h z$. The prover has the additional input $x = \log_g y$. The prover and verifier then engage in the following protocol.

1. The prover chooses $r \in_R \mathbb{Z}_q$, computes the auxiliary commitment $C = (h^x/z)^r$, and sends C to the verifier.

2. The prover executes the protocol denoted

$$PK\{(\alpha, \beta) : \ C = h^\alpha (\frac{1}{z})^\beta \ \wedge \ 1 = g^\alpha (\frac{1}{y})^\beta\}$$

   with the verifier.

3. The verifier accepts if it accepts in step 2, and if $C \neq 1$; otherwise, the verifier rejects.

Note that in an actual implementation, the value C may be sent to the verifier as part of the first message in the sub-protocol in step 2.

Theorem 3. *The above protocol is a special honest-verifier proof system for proving that* $\log_g y \neq \log_h z$.

Proof. Correctness of the protocol is by inspection.

Consider the protocol's soundness. If a prover can make an honest verifier accept with non-negligible probability, then using standard rewinding arguments, there exist values α and β such that the equations

$$C = h^\alpha (\frac{1}{z})^\beta \qquad\qquad 1 = g^\alpha (\frac{1}{y})^\beta \tag{17}$$

hold. From the second equation of (17) one can conclude that

$$\alpha \equiv \beta \log_g y \pmod{q} \ .$$

Substituting $\beta \log_g y$ for α in the first equation of (17), we get $C = (h^{\log_g y}/z)^\beta$. Since the verifier accepts only if $C \neq 1$, this implies that $h^{\log_g y}/z \neq 1$, i.e., that $\log_g y \neq \log_h z$.

To see that the protocol is special honest-verifier zero knowledge, note that in an actual run of the protocol with an honest prover, C is a random element of G. Thus, the simulator can simply generate C at random, and then use the simulator for the proof in step 2. □

Let us discuss related work. Independently of our work, Bresson and Stern [9] provide a protocol to prove that two discrete logartithms are not equal that is similar to ours. However, their protocol is about a factor of two less efficient than ours and is only computationally sound. Let us finally note that the (efficient) protocol proposed by Michels and Stadler [33] to prove whether or not two discrete logarithms are equal is *not* zero-knowledge because it reveals the value h^x (which the simulator cannot compute, but a (dishonest) verifier can if he chooses h such the he knows $\log_g h$).

6 Verifiable Decryption

In this section we provide a protocol that allows the decryptor to prove that she decrypted correctly. In particular, we provide a protocol that allows the decryptor to prove whether or not a given ciphertext decrypts to a given plaintext. We then extend the protocol to one for proving whether or not a given ciphertext decrypts to the discrete logarithm of a given group element. These protocols allow one, for instance, to lessen the trust placed in a third party in scenarios such as fair exchange. The second protocol enables, for instance, confirmer signatures scheme with perfect signature conversion (c.f. [13]), as we will show as well. In fact, the resulting confirmer signature scheme is an order of magnitude more efficient than the ones with the same properties known to date.

6.1 Definition of Verifiable Decryption

Verifiable decryption is a protocol between a prover, knowing the decryption key, and a verifier, who as the result of the protocol either rejects or learns whether or not a given ciphertext decrypts under a given label to a plaintext that satisfies a given relation.

We adopt the notation and terminology in §4.1. In addition, for mutually compatible encryption scheme encryption scheme $(\mathcal{G}, \mathcal{E}, \mathcal{D})$, relation generator $\mathcal{G}'$, and reconstruction algorithm *recon*, we define the function f that for all $(\mathsf{PK}, \mathsf{SK}) \in \mathcal{G}(1^\lambda)$, all $\Psi[\mathcal{R}, W, \Delta] \in \mathcal{G}'$, all $\psi, L \in \{0,1\}^*$, and all $\delta \in \Delta$

$$f(\Psi, \delta, \psi, L, \mathsf{SK}) = \begin{cases} 1 & (\mathit{recon}(\mathsf{PK}, \Psi, \delta, \mathcal{D}(\mathsf{SK}, \psi, L)), \delta) \in \mathcal{R} \\ -1 & \text{otherwise.} \end{cases}$$

The (honest) verifier in a verifiable decryption protocol will output either a value ± 1, indicating that this is the value of f, or the value 0, indicating that the proof is invalid.

A difficulty in defining soundness for verifiable decryption is that for many public key encryption schemes (including ours and, e.g., the El-Gamal based Cramer-Shoup one [20]), it is not well defined whether or not a ciphertext is valid given only the public key. More precisely, there are ciphertexts that can be both valid and invalid, depending on the actual value of the secret key. Hence, it is in principle possible that the decryptor/prover could change her mind about such ciphertexts, which seems inappropriate. In the following definition, we assume that the public and secret key are generated by a trusted party which allows us to define soundness in terms of the secret key and public key rather than only the public key. As for verifiable encryption, the definitions for the setting where the keys are not generated by a trusted party are a bit more complicated and subtle, and we do not present them here. However, also in here our protocols require only slight modification to remain secure in this setting.

Definition 2. *A proof system $(\mathcal{P}, \mathcal{V})$, together with mutually compatible encryption scheme $(\mathcal{G}, \mathcal{E}, \mathcal{D})$, relation generator $\mathcal{G}'$, and reconstruction algorithm recon, form a* verifiable decryption scheme*, if the following properties hold.*

Correctness: *For all $(\mathsf{PK}, \mathsf{SK}) \in \mathcal{G}(1^\lambda)$, for all $\Psi[\mathcal{R}, W, \Delta] \in \mathcal{G}'(1^\lambda)$, for all $\delta \in \Delta$, for all $\psi, L \in \{0,1\}^*$,*

$$\Pr[x \leftarrow \mathcal{V}(\mathsf{PK}, \Psi, \delta, \psi, L)_{\mathcal{P}(\mathsf{PK}, \Psi, \delta, \psi, L, \mathsf{SK})} : x = f(\Psi, \delta, \psi, L, SK)] = 1 - \mathrm{neg}(\lambda) \ .$$

Soundness: *For all adversaries* $(\mathcal{A}^*, \mathcal{P}^*)$*,*

$$\Pr[\ \begin{array}{l}(\mathsf{PK},\mathsf{SK}) \leftarrow \mathcal{G}(1^\lambda); \Psi[\mathcal{R}, W, \Delta] \leftarrow \mathcal{G}'(1^\lambda);\\ (\delta, \psi, L, aux) \leftarrow \mathcal{A}^*(\mathsf{PK},\mathsf{SK},\Psi);\\ x \leftarrow \mathcal{V}(\mathsf{PK},\Psi,\delta,\psi,L)_{\mathcal{P}^*(aux)} :\\ x = -f(\Psi,\delta,\psi,L,SK)\end{array} \ ] \ = \ \mathrm{neg}(\lambda)\ .$$

Special honest-verifier zero knowledge: *There exists a simulator Sim such that for all adversaries* $(\mathcal{A}^*, \mathcal{B}^*)$*, we have*

$$\Pr[\ \begin{array}{l}(\mathsf{PK},\mathsf{SK}) \leftarrow \mathcal{G}(1^\lambda); \Psi[\mathcal{R}, W, \Delta] \leftarrow \mathcal{G}'(1^\lambda);\\ (\delta, \psi, L, c, aux) \leftarrow \mathcal{A}^*(\mathsf{PK},\mathsf{SK},\Psi);\\ b \leftarrow \{0,1\};\\ \mathit{if}\ b = 0\\ \quad \mathit{then}\ \alpha \leftarrow \mathit{Trans}(\mathsf{PK},\Psi,\delta,\psi,L,c,\mathsf{SK})\\ \quad \mathit{else}\ \ \alpha \leftarrow \mathit{Sim}(\mathsf{PK},\Psi,\delta,\psi,L,c,f(\Psi,\delta,\psi,L,\mathsf{SK}));\\ \hat{b} \leftarrow \mathcal{B}^*(aux,\alpha) :\\ b = \hat{b}\end{array} \ ] \ = 1/2 + \mathrm{neg}(\lambda)\ .$$

6.2 Verifiable Decryption of a Matching Plaintext

We give a protocol for the decryptor to prove whether or not a ciphertext (u, e, v) decrypts to a message m with label L, i.e., using this protocol she can show that she did correctly decrypt. This is a special case of verifiable decryption in which the relation $\mathcal{R}$ is equality, and the reconstruction routine returns its last input as its output.

For our encryption scheme in §3, this proof corresponds to proving whether or not the two equations

$$u^{2(x_2+\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3)}/v^2 = 1 \qquad \text{and} \qquad (e/u^{x_1})^2/h^{2m} = 1 \tag{18}$$

hold (assuming that the public test $\mathrm{abs}(v) = v$ is satisfied). If the ciphertext is invalid, one or both of the two statements do not hold. If the ciphertext is valid but decrypts to another message, the first statements holds but the second one does not.

Proving that both of these equations hold is a fairly straightforward application of known techniques.

To prove that at least one of the equations does not hold, we can use the "proof of partial knowledge" technique of [19], combined with the technique developed in §5. However, because in the present setting the group has non-prime order we can not prove the relationship among the secrets in the same way as in §5 and, more importantly, the resulting protocol would not be zero-knowledge. The former problem can be solved using an auxiliary group $\mathfrak{G}_{\mathfrak{n}'} \subset \mathbb{Z}^*_{\mathfrak{n}}$ as we did in Section 4. We consider the latter problem. Depending on the values of the secret keys x_1, x_2, and x_3, the left hand sides of the equations (18), and thus the auxiliary commitments to be provided in the protocol, lie in different (sub-)groups, i.e., in $\mathbf{G}_n$, $\mathbf{G}_{n'}$, or $\mathbf{G}_n\mathbf{G}_{n'}$. As the simulator does to know the values of $x_1, \ldots, x_3$, it can not simulate these auxiliary commitments. We solve this problem using the fact that for all elements $a \in \mathbf{G}_n\mathbf{G}_{n'}$ we have

$$a \neq 1 \qquad \Leftrightarrow \qquad (a^n \in \mathbf{G}_{n'} \wedge a^n \neq 1) \quad \vee \quad (a \in \mathbf{G}_n \wedge a \neq 1)\ .$$

Thus, to prove that (at least) one of the equations (18) does not hold, we prove that either

$$\Big(\frac{u^{2(x_2+\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3)}}{v^2}\Big)^n \neq 1 \tag{19}$$

or

$$\Big(\frac{u^{2(x_2+\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3)}}{v^2}\Big)^n = 1 \qquad \text{and} \qquad \frac{u^{2(x_2+\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3)}}{v^2} \neq 1 \tag{20}$$

or

$$\Big(\frac{(e/u^{x_1})^2}{h^{2m}}\Big)^n = (e/u^{x_1})^{2n} \neq 1 \tag{21}$$

or

$$\Big(\frac{(e/u^{x_1})^2}{h^{2m}}\Big)^n = 1 \qquad \text{and} \qquad \frac{(e/u^{x_1})^2}{h^{2m}} \neq 1 \tag{22}$$

holds. Now, whenever one of the four cases applies it is always well defined in which group the left-hand sides of the inequalities lie and we can apply the ideas underlying the protocol in Section 5. We remark that the case where the statements (19-21) are false but the statement (22) is true corresponds to the case, where the ciphertexts is a valid encryption of a message different from m.

We are now ready to describe the protocol between the decryptor and a verifier. Their common input is $(\mathsf{hk}, n, g, y_1, y_2, y_3), (\mathfrak{n}, \mathfrak{g}, \mathfrak{h}), (u, e, v), m, L$ and the additional input to the decryptor is (x_1, x_2, x_3). The triple $(\mathfrak{n}, \mathfrak{g}, \mathfrak{h})$ is an auxiliary parameter as in the one previous section. (As we assume here that n is generated by a trusted party as well, i.e., that the decryptor is not provided with n's factorization; also, n and $\mathfrak{n}$ could be identical.) In the following description we assume that all the messages the prover sends to the verifier prior to the execution of one of the possible *PK* protocols will in fact be bundled with the first message of that *PK* protocol. Here we provide the proof-protocols only by high-level notation; deriving the actual protocols is easily derived from it (cf. also the the verifiable encryption protocol presented in §4 and its high-level notation).

1. If $m \notin [n]$ or the ciphertext is malformed, (e.g., if $v \neq \mathrm{abs}(v)$), the verifier outputs -1, and the protocol stops.
2. If (u, e, v) is a valid ciphertext with label L and decrypts to m, the decryptor sends 1 to the verifier, and then engages in the protocol denoted

   $$PK\{(x_1, x_2, x_3) : y_1 = g^{x_1} \ \wedge \ y_2 = g^{x_2} \ \wedge \ y_3 = g^{x_3} \ \wedge v^2 = u^{2x_2} u^{2\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3} \ \wedge \ \frac{e^2}{h^{2m}} = u^{2x_1}\}$$

   with the verifier.
3. If (u, e, v) is an invalid ciphertext w.r.t. the label L or decrypts to some message different from m, then the decryptor sends -1 to the verifier. They proceed as follows.

   (a) The decryptor chooses $a_1 \in_R [n/4]$, $a_2 \in_R [n^2/4]$, $a_3 \in_R [n/4]$, and $a_4 \in_R [n^2/4]$, along with $b_1, b_2, b_3, b_3 \in_R [\mathfrak{n}/4]$.
   She then computes $\mathfrak{C}_1 := \mathfrak{g}^{a_1}\mathfrak{h}^{b_1}$, $\mathfrak{C}_2 := \mathfrak{g}^{a_2}\mathfrak{h}^{b_2}$, $\mathfrak{C}_3 := \mathfrak{g}^{a_3}\mathfrak{h}^{b_3}$, and $\mathfrak{C}_4 := \mathfrak{g}^{a_4}\mathfrak{h}^{b_4}$.
   She chooses $C_1 \in_R \mathbf{G}_{n'}$, $C_2 \in_R \mathbf{G}_n$, $C_3 \in_R \mathbf{G}_{n'}$, and $C_4 \in_R \mathbf{G}_n$.
   Furthermore,

   (Case 1) if $u^{2n(x_2+\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3)} \neq v^{2n}$, she sets $C_1 := (u^{x_2+\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3}/v)^{2na_1}$,
   (Case 2) else if $u^{2(x_2+\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3)} \neq v^2$, she sets $C_2 := (u^{x_2+\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3}/v)^{2a_2}$,
   (Case 3) else if $(u^{x_1}/e)^2 \notin \langle h \rangle$, she sets $C_3 := (u^{x_1}/e)^{2na_3}$,
   (Case 4) else $(u^{x_1}/e)^2 \neq h^{2m}$, and she sets $C_4 := (u^{x_1}h^m/e)^{2a_4}$.

   The decryptor sends C_1, C_2, C_3, C_4, $\mathfrak{C}_1$, $\mathfrak{C}_2$, $\mathfrak{C}_3$, and $\mathfrak{C}_4$ to the verifier.

(b) The decryptor and the verifier carry out the protocol denoted

$$
\begin{aligned}
PK\Big\{ & (x_1, x_2, x_3,\, a_1, \ldots, a_4,\, b_1, \ldots, b_4,\, r_1, \ldots, r_4\, s_1, \ldots, s_4) : \\
& \Big[y_1 = g^{x_1} \ \wedge\ y_2 = g^{x_2} \ \wedge\ y_3 = g^{x_3} \ \wedge \\
& \quad C_1 = u^{2nr_1}(\frac{1}{v})^{2na_1} \ \wedge\ \mathfrak{C}_1 = \mathfrak{g}^{a_1}\mathfrak{h}^{b_1} \ \wedge\ 1 = (\frac{1}{\mathfrak{C}_1})^{x_2}(\frac{1}{\mathfrak{C}_1})^{\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3}\mathfrak{g}^{r_1}\mathfrak{h}^{s_1} \Big] \\
\vee\ & \Big[y_1 = g^{x_1} \ \wedge\ y_2 = g^{x_2} \ \wedge\ y_3 = g^{x_3} \ \wedge \\
& \quad C_2 = u^{2r_2}(\frac{1}{v})^{a_2} \ \wedge\ \mathfrak{C}_2 = \mathfrak{g}^{a_2}\mathfrak{h}^{b_2} \ \wedge\ 1 = (\frac{1}{\mathfrak{C}_2})^{x_2}(\frac{1}{\mathfrak{C}_2})^{\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3}\mathfrak{g}^{r_2}\mathfrak{h}^{s_2} \Big] \\
\vee\ & \Big[y_1 = g^{x_1} \ \wedge\ y_2 = g^{x_2} \ \wedge\ y_3 = g^{x_3} \ \wedge \\
& \quad C_3 = u^{2nr_3}(\frac{1}{e})^{2na_3} \ \wedge\ \mathfrak{C}_3 = \mathfrak{g}^{a_3}\mathfrak{h}^{b_3} \ \wedge\ 1 = (\frac{1}{\mathfrak{C}_3})^{x_1}\mathfrak{g}^{r_3}\mathfrak{h}^{s_3} \Big] \\
\vee\ & \Big[y_1 = g^{x_1} \ \wedge\ y_2 = g^{x_2} \ \wedge\ y_3 = g^{x_3} \ \wedge \\
& \quad C_4 = u^{2r_4}(\frac{h^m}{e})^{2a_4} \ \wedge\ \mathfrak{C}_4 = \mathfrak{g}^{a_4}\mathfrak{h}^{b_4} \ \wedge\ 1 = (\frac{1}{\mathfrak{C}_4})^{x_1}\mathfrak{g}^{r_4}\mathfrak{h}^{s_4} \Big] \Big\} \ ,
\end{aligned}
$$

where $r_1, \ldots, r_4, s_1, \ldots, s_4$ are temporary secrets (i.e.,

$$
\begin{aligned}
r_1 &= a_1(x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)x_3), & s_1 &= b_1(x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)x_3), \\
r_2 &= a_2(x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)x_3), & s_2 &= b_2(x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)x_3), \\
r_3 &= x_1 a_3, & s_3 &= x_1 b_3, \\
r_4 &= x_1 a_4, & s_4 &= x_1 b_4,
\end{aligned}
$$

(computed in $\mathbb{Z}$)). (To derive the actual protocol one has to apply the techniques by Cramer et al.[19] for realizing the $\vee$'s.)

(c) The verifier checks that $C_1^2 \neq 1$, $C_2^2 \neq 1$, $C_3^2 \neq 1$, and $C_4^2 \neq 1$.

The computational load of the prover and the verifier is about one to four times the load in the protocol for verifiable encryption described in the previous section (depending on whether step 2 or step 3 gets carried out).

Theorem 4. *Assuming factoring is hard, the above scheme is a verifiable decryption scheme (for matching plaintexts).*

Proof. Correctness is trivial, and we leave this to the reader.

We now show that the protocol is special honest-verifier computational zero-knowledge by providing a simulator.

First the simulator executes step 1 of the protocol as the decryptor would, that is, if $m \notin [n]$ or if the ciphertext is malformed the simulator stops. The simulator queries an oracle to determine whether or not ψ decrypts to m. If it does, it sends the verifier 1 it simulates step 2 by the simulator for the PK-protocol of step 2. If does not, it simulates step 3 as follows. First the simulator sends the verifier -1. Then it chooses $b_1, b_2, b_3, b_3 \in_R [\mathfrak{n}/4]$. It then computes $\mathfrak{C}_1 := \mathfrak{h}^{b_1}$, $\mathfrak{C}_2 := \mathfrak{h}^{b_2}$, $\mathfrak{C}_3 := \mathfrak{h}^{b_3}$, and $\mathfrak{C}_4 := \mathfrak{h}^{b_4}$. It chooses $C_1 \in_R \mathbf{G}_{n'}$, $C_2 \in_R \mathbf{G}_n$, $C_3 \in_R \mathbf{G}_{n'}$, and $C_4 \in_R \mathbf{G}_n$. Next it invokes the simulator for the PK-protocol of step 3. This concludes the simulator.

It remains to show that the simulator indeed works. It is clear that the simulation of steps 1 and 2 works. Consider step 3.

Note that in the real run as well as in the simulation the pairs $(\mathfrak{C}_1, C_1), \ldots, (\mathfrak{C}_4, C_4)$ are independently distributed. Moreover they obviously have the same distribution in the simulation as in the real run except for one the pair for which the prover replaces the C_i.

We consider the cases where the prover replaces C_1 and C_2, respectively. The remaining two cases are analogous.

Case 1. Here $u^{2n(x_2+\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3)} \neq v^{2n}$ holds and the prover replaces C_1. Note that $(u^{(x_2+\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3)}/v)^{2n} \in \mathbf{G}_{n'}$ and $(u^{(x_2+\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3)}/v)^{2n} \neq 1$. Thus $(u^{(x_2+\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3)}/v)$ generates $\mathbf{G}_{n'}$ (or we could factor n) and $C_1 = (u^{x_2+\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3}/v)^{2na_1}$ is a random element of $\mathbf{G}_{n'}$ as a_1 is chosen at random from the appropriate interval. Also, as b_1 is chosen independently of a_1, $\mathfrak{C}_1$ is a random element from $\mathfrak{G}_{n'}$. Hence $\mathfrak{C}_1$ and C_1 have the same distribution in the run with the real prover as in the simulation.

Case 2. As the above case does not apply, i.e., $(u^{(x_2+\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3)}/v)^{2n} = 1$ we have that $(u^{x_2+\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3)}/v)^2 \in \mathbf{G}_n$. Again, $(u^{x_2+\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3)}/v)^2$ generates $\mathbf{G}_n$ (or we could factor n) and $C_2 = (u^{x_2+\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3}/v)^{2a_2}$ as a_1 is chosen at random. For the same reason as in Case 1, $\mathfrak{C}_2$ is a random element from $\mathfrak{G}_{n'}$ and $\mathfrak{C}_2$ and C_2 have the same distribution in the run with the real prover as in the simulation.

These facts together with the fact that all the *PK*-protocols used as sub-protocols are special honest-verifier zero-knowledge (showing the latter is standard and left to the reader), imply that the verifiable decryption protocol is special honest-verifier zero-knowledge. Note that we have used in an essential way the fact that we quantify "computationally" over the inputs to the simulator: the inputs that cause the simulator to fail are assumed to be hard to find.

In the remainder we prove soundness. To this end, we present an algorithm that uses algorithms $\mathcal{A}^*$ and $\mathcal{P}^*$ and takes as input n and $\mathfrak{n}$. We then show that if $\mathcal{A}^*$ and $\mathcal{P}^*$ violate the soundness property with non-negligible probability then the algorithm outputs a factorization of either n or $\mathfrak{n}$ with non-negligible probability. The first probability is w.r.t. the coin tosses of $\mathcal{G}$, $\mathcal{A}^*$, $\mathcal{P}^*$, and $\mathcal{V}$ while the second probability is w.r.t. the random choice of n and $\mathfrak{n}$ (drawn from the same distribution as induced by $\mathcal{G}$) and the coin tosses of the factoring algorithm.

We choose a random $\mathfrak{g}' \in_R \mathbb{Z}^*_{\mathfrak{n}}$ and we set $\mathfrak{g} := \tilde{\mathfrak{g}}^2$, choose a random $r \in_R [\mathfrak{n}^2]$ and set $\mathfrak{h} := \mathfrak{g}^r$. We choose random x_1, x_2, $x_3 \in_R [n^2/4]$, choose a random $g' \in_R \mathbb{Z}^*_{n^2}$, and compute $g := (g')^{2n}$, $y_1 := g^{x_1}$, $y_2 := g^{x_2}$, and $y_3 := g^{x_3}$.

Now we run $\mathcal{A}^*$ and $\mathcal{P}^*$ on input $\mathsf{PK} = ((n, g, y_1, y_2, y_3), (\mathfrak{n}, \mathfrak{g}, \mathfrak{h}))$ and $\mathsf{SK} = (x_1, x_2, x_3)$. By standard rewinding techniques we can produce two accepting conversations for either the *PK* protocol in step 2 or the one in step 3 (for different challenges but the same first message), depending on whether $m = \mathcal{D}(1^\lambda, \mathsf{SK}, \psi, L)$ for (m, ψ, L) provided by $\mathcal{A}^*$. We consider the two cases.

Case I. First assume that $m \neq \mathcal{D}(1^\lambda, \mathsf{SK}, \psi, L)$ but that V's output is 1. Let $(u, e, v) := \psi$. In this case we get two accepting conversations of the *PK* protocol in step 2 and hence two answers

$$(\tilde{x}_1^{(1)}, \tilde{x}_2^{(1)}, \tilde{x}_3^{(1)}) \quad \text{and} \quad (\tilde{x}_1^{(2)}, \tilde{x}_2^{(2)}, \tilde{x}_3^{(2)})$$

for the two different challenges $c^{(1)}$ and $c^{(2)}$ but with the same first message (here we use the same notation for the protocol variables as for the *PK* protocol in the previous section). W.l.o.g., suppose that $c^{(2)} > c^{(1)}$. Let $\Delta x_1 = \tilde{x}_1^{(1)} - \tilde{x}_1^{(2)}$, $\Delta x_2 = \tilde{x}_2^{(1)} - \tilde{x}_2^{(2)}$, $\Delta x_3 = \tilde{x}_3^{(1)} - \tilde{x}_3^{(2)}$, and $\Delta c = c^{(2)} - c^{(1)}$.

From the verification equation of the *PK* protocol one can derive the following equations:

$$y_1{}^{\Delta c} = g^{\Delta x_1} \ , \qquad y_2{}^{\Delta c} = g^{\Delta x_2} \ , \qquad y_3{}^{\Delta c} = g^{\Delta x_3} \ , \tag{23}$$

$$v^{2\Delta c} = u^{2\Delta x_2} u^{2\mathcal{H}_{\mathsf{hk}}(u,e,L)\Delta x_3} \ , \quad \text{and} \tag{24}$$

$$(\frac{e^2}{h^{2m}})^{\Delta c} = u^{2\Delta x_1} \ . \tag{25}$$

As n is the product of two safe primes p and q, we have $|\Delta c| < \min\{p, q, p'q'\}$ and hence Δc is invertible modulo $n'n$. We know x_i such that $y_i = g^{x_i}$ and therefore it follows from (23) that

$$\Delta c \, x_i \equiv \Delta x_i \pmod{n'} \quad \text{for} \quad i = 1, \ldots, 3 \ . \tag{26}$$

Now, $\mathcal{D}(1^\lambda, \mathsf{SK}, \psi, L) \neq m$ means that least one of the four statements (19-22) must be true and therefore at least one of the two statements

$$u^{2(x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)x_3)} \neq v^2 \qquad \text{or} \qquad (e/u^{x_1})^2 \neq h^{2m} \tag{27}$$

holds. We consider these two cases:

Case 1. If $u^{2(x_2+\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3)} \neq v^2$ we must have that $u^{2\Delta c(x_2+\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3)} \neq v^{2\Delta c} = u^{2\Delta x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)\Delta x_3}$ (from Equation (24) and because Δc is invertible modulo nn') and therefore also

$$\Delta c(x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)x_3) \not\equiv \Delta x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)\Delta x_3 \pmod{n'n} \ ,$$

as the order of u^2 divides $n'n$. From (26) if follows that

$$\Delta c(x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)x_3) \equiv \Delta x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)\Delta x_3 \pmod{n'} \ .$$

Therefore $\Delta c x_2 - \Delta x_2 + (\Delta c x_3 - \Delta x_3)\mathcal{H}_{\mathsf{hk}}(u,e,L)$ must be a non-zero multiple of n' and we can factor n.

Case 2. If $u^{2x_1} \neq (\frac{e}{h^m})^2$ we can, similarly as in case 1, conclude that $u^{2\Delta c x_1} \neq u^{2\Delta x_1}$ from Equation (25) and and that $\Delta c x_1 - \Delta x_1$ is a non-zero multiple of n' which again allows us to factor n.

Case II. It remains to consider the case when V's output is -1 but $m = \mathcal{D}(1^\lambda, \mathsf{SK}, \psi, L)$ holds. Let $(u, e, v) := \psi$. Thus we have

$$v^2 = u^{2(x_2+\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3)} \qquad \text{and} \qquad u^{2x_1} = (\frac{e}{h^m})^2 \ . \tag{28}$$

As usual we obtain two accepting conversation of the *PK* protocol in step 3 and thus two answers

$$(\tilde{x}_1^{(1)}, \tilde{x}_2^{(1)}, \tilde{x}_3^{(1)}, \tilde{a}_1^{(1)}, \ldots, \tilde{a}_6^{(1)}, \tilde{b}_1^{(1)}, \ldots, \tilde{b}_4^{(1)}, \tilde{r}_1^{(1)}, \ldots, \tilde{r}_4^{(1)}, \tilde{s}_1^{(1)}, \ldots, \tilde{s}_4^{(1)})$$

and

$$(\tilde{x}_1^{(2)}, \tilde{x}_2^{(2)}, \tilde{x}_3^{(2)}, \tilde{a}_1^{(2)}, \ldots, \tilde{a}_4^{(2)}, \tilde{b}_1^{(2)}, \ldots, \tilde{b}_4^{(2)}, \tilde{r}_1^{(2)}, \ldots, \tilde{r}_4^{(2)}, \tilde{s}_1^{(2)}, \ldots, \tilde{s}_4^{(2)})$$

for the two different challenges $c^{(1)}$ and $c^{(2)}$ but with the same first message (here we use the same notation for the protocol variables as for the *PK* protocol in the previous section and left out an intermediate step that deals with the $\vee$'s (c.f. [19])). W.l.o.g., suppose that $c^{(2)} > c^{(1)}$. Let

$$\begin{aligned}
&\Delta x_i = \tilde{x}_i^{(1)} - \tilde{x}_i^{(2)} \ (i = 1, \ldots, 3); && \Delta a_i = \tilde{a}_i^{(1)} - \tilde{a}_i^{(2)} \ (i = 1, \ldots, 4);\\
&\Delta b_i = \tilde{b}_i^{(1)} - \tilde{b}_i^{(2)} \ (i = 1, \ldots, 4); && \Delta s_i = \tilde{s}_i^{(1)} - \tilde{s}_i^{(2)} \ (i = 1, \ldots, 4);\\
&\Delta r_i = \tilde{r}_i^{(1)} - \tilde{r}_i^{(2)} \ (i = 1, \ldots, 4); && \Delta c = c^{(2)} - c^{(1)} \ .
\end{aligned}$$

From the verification equation of the *PK* protocol one can derive that

$$y_1{}^{\Delta c} = g^{\Delta x_1} \ , \qquad y_2{}^{\Delta c} = g^{\Delta x_2} \ , \quad \text{and} \qquad y_3{}^{\Delta c} = g^{\Delta x_3} \ , \tag{29}$$

hold and either

$$C_1^{\Delta c} = u^{2n\Delta r_1}(\frac{1}{v})^{2n\Delta a_1} \ , \quad \mathfrak{C}_1^{\Delta c} = \mathfrak{g}^{\Delta a_1}\mathfrak{h}^{\Delta b_1} \ , \quad \text{and} \quad 1 = (\frac{1}{\mathfrak{C}_1})^{\Delta x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)\Delta x_3}\mathfrak{g}^{\Delta r_1}\mathfrak{h}^{\Delta s_1} \tag{30}$$

or

$$C_2^{\Delta c} = u^{2\Delta r_2}(\frac{1}{v})^{2\Delta a_2} \ , \qquad \mathfrak{C}_2^{\Delta c} = \mathfrak{g}^{\Delta a_2}\mathfrak{h}^{\Delta b_2} \ , \quad \text{and} \quad 1 = (\frac{1}{\mathfrak{C}_2})^{\Delta x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)\Delta x_3}\mathfrak{g}^{\Delta r_2}\mathfrak{h}^{\Delta s_2} \tag{31}$$

or

$$C_3^{\Delta c} = u^{2n\Delta r_3}(\frac{1}{e})^{2n\Delta a_3} \ , \quad \mathfrak{C}_3^{\Delta c} = \mathfrak{g}^{\Delta a_3}\mathfrak{h}^{\Delta b_3} \ , \quad \text{and} \quad 1 = (\frac{1}{\mathfrak{C}_3})^{\Delta x_1}\mathfrak{g}^{\Delta r_3}\mathfrak{h}^{\Delta s_3} \tag{32}$$

or

$$C_4^{\Delta c} = u^{2\Delta r_4}(\frac{1}{e})^{n\Delta a_4} \ , \qquad \mathfrak{C}_4^{\Delta c} = \mathfrak{g}^{\Delta a_4}\mathfrak{h}^{\Delta b_4} \ , \quad \text{and} \quad 1 = (\frac{1}{\mathfrak{C}_4})^{\Delta x_1}\mathfrak{g}^{\Delta r_4}\mathfrak{h}^{\Delta s_4} \tag{33}$$

hold. We know x_i such that $y_i = g^{x_i}$ and therefore it follows from (29) that

$$\Delta c\, x_i \equiv \Delta x_i \pmod{n'} \quad \text{for} \quad i = 1, \ldots, 3 \ . \tag{34}$$

We next consider the implications of the cases when the equations (30), the equations (31), the equations (32), or the equations (33) hold in conjunction with (29).

Case 1. Consider the case where Equations (29) and (30) hold. From the last two equations of (30) we get

$$\mathfrak{g}^{\Delta a_1(\Delta x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)\Delta x_3)}\mathfrak{h}^{\Delta b_1(\Delta x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)\Delta x_3)} = \mathfrak{g}^{\Delta c\Delta r_1}\mathfrak{h}^{\Delta c\Delta s_1} \ .$$

Therefore

$$\Delta a_1(\Delta x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)\Delta x_3) = \Delta c\Delta r_1 \tag{35}$$

must hold (in $\mathbb{Z}$); otherwise (with overwhelming probability) either α or $\alpha - r\beta$ is a non-zero multiple of $\mathfrak{n}'$ where

$$\alpha := \Delta a_1(\Delta x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)\Delta x_3) - \Delta c\Delta r_1 \ ,$$

$$\beta := \Delta c\Delta e_1 - \Delta b_1(\Delta x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)\Delta x_3) \ ,$$

and r is the random number we chose at the beginning of the factoring algorithm to compute $\mathfrak{h}$. This allows us to factor $\mathfrak{n}$.

Because n is the product of two safe primes and we have $|\Delta c| < \min\{p, q, p'q'\}$ and it follows from $C_1^2 \neq 1$ (which is checked by the verifier in step 3c) that $C_1^{\Delta c} \neq 1$. Therefore, from the first equation of (30) it follows that $u^{2n\Delta r_1} \neq v^{2n\Delta a_1}$, and by Eq. (35) and the fact that u^{2n} and v^{2n} have order dividing n', we have

$$u^{2n\Delta a_1(\Delta x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)\Delta x_3)} \neq v^{2n\Delta c\Delta a_1} \ ,$$

and hence

$$u^{2n(\Delta x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)\Delta x_3)} \neq v^{2n\Delta c} \ . \tag{36}$$

From (36) and the first equation of (28) we have

$$u^{2n(\Delta x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)\Delta x_3)} \neq v^{2n\Delta c} = u^{2n\Delta c(x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)x_3)} \ .$$

Because the order of u^{2n} divides n' we can further conclude that

$$\Delta x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)\Delta x_3 \not\equiv \Delta c(x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)x_3) \pmod{n'} \ .$$

From (34) if follows that

$$\Delta x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)\Delta x_3 \equiv \Delta c(x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)x_3) \pmod{n'} \ ,$$

which is a contradiction to the previous equation and hence this case cannot occur.

Case 2. We consider the case where Equations (29) and (31) hold. Similarly as in case 1, we can derive that

$$u^{2(\Delta x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)\Delta x_3)} \neq v^{2\Delta c} = u^{2\Delta c(x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)x_3)}$$

holds (or we factor $\mathfrak{n}$ with non-negligible probability). Because the order of u^2 divides $n'n$ we can further conclude that

$$\Delta x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)\Delta x_3 \not\equiv \Delta c(x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)x_3) \pmod{n'n} \ .$$

From (34) if follows that

$$\Delta x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)\Delta x_3 \equiv \Delta c(x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)x_3) \pmod{n'} \ .$$

Therefore $\Delta c x_2 - \Delta x_2 + (\Delta c x_3 - \Delta x_3)\mathcal{H}_{\mathsf{hk}}(u,e,L)$ must be a non-zero multiple of n' and we can factor n.

Case 3. Similarly as in case 1, from the Equations (29) and (32), one can derive that

$$u^{2n\Delta x_1} \neq e^{2n\Delta c} \tag{37}$$

holds (or we factor $\mathfrak{n}$ with non-negligible probability). From the second equation of (28) and $h^n = 1$ if follows that $u^{2nx_1} = e^{2n}$ and $u^{2n\Delta c x_1} = e^{2n\Delta c}$, and from (37), that

$$u^{2n\Delta c x_1} \neq u^{2n\Delta x_1} \qquad \text{and finally that} \qquad \Delta c x_1 \not\equiv \Delta x_1 \pmod{n'}$$

as u^{2n} has order dividing n'. The latter, however, is a contradiction to Eqn. (34) and thus this case cannot occur.

Case 4. Similarly as before, from the Equations (29) and (33) one can show that

$$u^{2\Delta x_1} \neq (\frac{e}{h^m})^{2\Delta c} \tag{38}$$

holds (or we factor $\mathfrak{n}$ with non-negligible probability). From the second equation of (28) and from (38) we get $u^{2\Delta c x_1} \neq u^{2\Delta x_1}$. Similarly as in case 2, it follows that $\Delta c x_1 - \Delta x_1$ is a multiple of n' and we are again able to factor n.

□

6.3 Verifiable Decryption of a Discrete Logarithm

We now describe how the protocol provided in the previous section can be modified to obtain a protocol for verifiable decryption of a discrete logarithm. The setting and notation are as in §4.2; in particular, we make use of the same reconstruction routine.

We need to modify the protocol from the previous section only for the cases where the ciphertext is valid. That is, instead of proving that the ciphertext decrypts (or does not decrypt) to a given message, the decryptor now has to prove that it decrypts (or does not decrypt) to a value m such that $(m \operatorname{rem} n) \equiv \log_\gamma \delta \pmod{\rho}$. This corresponds to proving whether or not the three equations

$$u^{2(x_2+\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3)}/v^2 = 1 \qquad \text{or} \qquad (e/u^{x_1})^{2n} = 1 \qquad \text{or} \qquad \delta = \gamma^{(\log_{h^2}(e/u^{x_1})^2 \operatorname{rem} n)} \tag{39}$$

hold. Note that $\log_{h^2}(e/u^{x_1})^2$ exist if and only if $(e/u^{x_1})^{2n} = 1$. The first two statements of (39) can be handled as in the previous section. The last one can be handled by proving knowledge of a secret, say m, that (1) equals the encrypted message modulo n, (2) equals (or doesn't equal) $\log_\gamma \delta$ modulo q, and (3) lies in the interval $[-(n-1)/2, (n-1)/2]$. The first two properties can be proved under the strong RSA assumption using additional parameters $(\mathfrak{n}, \mathfrak{g}, \mathfrak{h})$ as in the previous section. We discuss proving the last one. Different from the interval-proof used for verifiable encryption, this interval-proof needs to be *exact*, i.e., if we allowed for the same sloppiness, then the prover could for instance add a multiple of n to m and then show that (u, e, v) does not (or does) decrypt to $\log_\gamma \delta$.

Boudot [8] presents several protocols to prove that in integer m lies exactly in an interval $[a, b]$. One protocol uses the fact that $x \in [a, b]$ is equivalent to $b - x \geq 0$ and $x - a \geq 0$ and that one can show that an integer is positive by proving knowledge of four values the squares of which sum up to the considered integer (in $\mathbb{Z}$), again under the strong RSA assumption using additional parameters $(\mathfrak{n}, \mathfrak{g}, \mathfrak{h})$. Lagrange proved the an integer can always be represented as four squares and Rabin and Shallit [36] provide an efficient algorithm for it.

We note that in our case the interval is symmetric and it therefore suffices to prove that $((n-1)/2)^2 - m^2 \geq 0$ holds, which is more efficient.

With these observations one can derive the following protocol for verifiable decryption of a discrete logarithm from the protocol presented in the previous section.

The common input of the decryptor and the verifier is $(\mathsf{hk}, n, g, y_1, y_2, y_3), (\mathfrak{n}, \mathfrak{g}, \mathfrak{h}), (u, e, v), \delta, L$ and the additional input to the decryptor is (x_1, x_2, x_3).

1. If $\delta \notin \Gamma$ or the ciphertext is malformed (e.g., if $v \neq \operatorname{abs}(v)$), the verifier outputs -1, and the protocol stops.

   In case (u, e, v) is a valid ciphertext w.r.t. label L, the prover decrypts it, thereby obtain m, and computes integers $w_1, \ldots, w_4$ such that $\sum_{i=1}^4 w_i = (n-1)^2/4 - m^2$ (c.f. [36]).

2. If (u, e, v) indeed decrypts to $\log_\gamma \delta$ with label L, i.e., if $\delta = \gamma^{m \operatorname{rem} n}$, the decryptor sends 1 to the verifier, chooses $t_1, \ldots, t_5 \in_R [\mathfrak{n}/4]$, computes

$$\mathfrak{W}_1 := \mathfrak{g}^{w_1}\mathfrak{h}^{t_1}, \mathfrak{W}_2 := \mathfrak{g}^{w_2}\mathfrak{h}^{t_2}, \mathfrak{W}_3 := \mathfrak{g}^{w_3}\mathfrak{h}^{t_3}, \mathfrak{W}_4 := \mathfrak{g}^{w_4}\mathfrak{h}^{t_4}, \text{ and } \mathfrak{M} := \mathfrak{g}^{m}\mathfrak{h}^{t_5} \ ,$$

and sends $\mathfrak{W}_1$, $\mathfrak{W}_2$, $\mathfrak{W}_3$, $\mathfrak{W}_4$, and $\mathfrak{M}$ to the verifier.

The prover and the verifier engage in the protocol

$$\begin{aligned} PK\{(x_1, x_2, x_3, m, w_1, \ldots, w_4, t_1, \ldots, t_5, s) : \\ y_1 = g^{x_1} \ \wedge \ y_2 = g^{x_2} \ \wedge \ y_3 = g^{x_3} \ \wedge \\ v^2 = u^{2x_2} u^{2\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3} \ \wedge \ e^2 = u^{2x_1} h^{2m} \ \wedge \\ \mathfrak{W}_1 = \mathfrak{g}^{w_1}\mathfrak{h}^{t_1} \ \wedge \ \mathfrak{W}_2 = \mathfrak{g}^{w_2}\mathfrak{h}^{t_2} \ \wedge \ \mathfrak{W}_3 = \mathfrak{g}^{w_3}\mathfrak{h}^{t_3} \ \wedge \ \mathfrak{W}_4 = \mathfrak{g}^{w_4}\mathfrak{h}^{t_4} \ \wedge \\ \mathfrak{M} = \mathfrak{g}^{m}\mathfrak{h}^{t_5} \ \wedge \ \mathfrak{g}^{(n-1)^2/4} = \mathfrak{M}^m \mathfrak{W}_1^{w_1} \mathfrak{W}_2^{w_2} \mathfrak{W}_3^{w_3} \mathfrak{W}_4^{w_4} \mathfrak{h}^s \ \wedge \\ \delta = \gamma^m \} \ , \end{aligned}$$

where s is a temporary secret (i.e., $s = -t_5 m - \sum_{i=1}^{4} w_i t_i$).

3. If (u, e, v) is an invalid ciphertext w.r.t. the label L or decrypts to some message m such that $\delta \neq \gamma^{m \operatorname{rem} n}$, then the decryptor sends -1 to the verifier. They proceed as follows.

   (a) The decryptor chooses $a_1 \in_R [n/4]$ $a_2 \in_R [n^2/4]$, $a_3 \in_R [n/4]$, and $a_4 \in_R [\rho]$, along with $b_1, \ldots, b_3, t_1, \ldots, t_5 \in_R [\mathfrak{n}/4]$.
   She computes $\mathfrak{C}_1 := \mathfrak{g}^{a_1}\mathfrak{h}^{b_1}$, $\mathfrak{C}_2 := \mathfrak{g}^{a_2}\mathfrak{h}^{b_2}$, $\mathfrak{C}_3 := \mathfrak{g}^{a_3}\mathfrak{h}^{b_3}$, and $\mathfrak{C}_4 := \mathfrak{g}^{a_4}\mathfrak{h}^{b_4}$.
   She computes $\mathfrak{W}_1 := \mathfrak{h}^{t_1}$, $\mathfrak{W}_2 := \mathfrak{h}^{t_2}$, $\mathfrak{W}_3 := \mathfrak{h}^{t_3}$, $\mathfrak{W}_4 := \mathfrak{h}^{t_4}$, and $\mathfrak{M} := \mathfrak{h}^{t_5}$.
   She chooses $C_1 \in_R \mathbf{G}_{n'}$, $C_2 \in_R \mathbf{G}_n$, $C_3 \in_R \mathbf{G}_{n'}$, and $C_4 \in_R \Gamma$.
   Furthermore,

(Case 1) if $u^{2n(x_2+\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3)} \neq v^{2n}$,	she sets	$C_1 := (u^{x_2+\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3}/v)^{2na_1}$,
(Case 2) else if $u^{2(x_2+\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3)} \neq v^2$,	she sets	$C_2 := (u^{x_2+\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3}/v)^{2a_2}$,
(Case 3) else if $(u^{x_1}/e)^2 \notin \langle h \rangle$,	she sets	$C_3 := (u^{x_1}/e)^{2na_3}$,
(Case 4) else $\delta \neq \gamma^{m \operatorname{rem} n}$,	and she sets	$C_4 := (\gamma^m/\delta)^{2a_4}$,
		$\mathfrak{W}_i := \mathfrak{g}^{w_i}\mathfrak{h}^{t_i}$ $(i = 1, \ldots, 4)$, and
		$\mathfrak{M} := \mathfrak{g}^{m}\mathfrak{h}^{t_5}$.

   The decryptor sends C_1, C_2, C_3, C_4, $\mathfrak{C}_1$, $\mathfrak{C}_2$, $\mathfrak{C}_3$, and $\mathfrak{C}_4$ to the verifier.

(b) The decryptor and the verifier carry out the protocol denoted

$$
\begin{aligned}
PK\Big\{ (x_1, x_2, x_3,\, a_1, \ldots, a_4,\, b_1, \ldots, b_4,\, r_1, \ldots, r_4\, s_1, \ldots, s_5, t_1, \ldots, t_5, w_1, \ldots, w_4, m) : \\
\Big[y_1 = g^{x_1} \ \wedge\ y_2 = g^{x_2} \ \wedge\ y_3 = g^{x_3} \ \wedge \\
C_1 = u^{2nr_1}(\frac{1}{v})^{2na_1} \ \wedge\ \mathfrak{C}_1 = \mathfrak{g}^{a_1}\mathfrak{h}^{b_1} \ \wedge\ 1 = (\frac{1}{\mathfrak{C}_1})^{x_2}(\frac{1}{\mathfrak{C}_1})^{\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3}\mathfrak{g}^{r_1}\mathfrak{h}^{s_1} \Big] \\
\vee \Big[y_1 = g^{x_1} \ \wedge\ y_2 = g^{x_2} \ \wedge\ y_3 = g^{x_3} \ \wedge \\
C_2 = u^{2r_2}(\frac{1}{v})^{a_2} \ \wedge\ \mathfrak{C}_2 = \mathfrak{g}^{a_2}\mathfrak{h}^{b_2} \ \wedge\ 1 = (\frac{1}{\mathfrak{C}_2})^{x_2}(\frac{1}{\mathfrak{C}_2})^{\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3}\mathfrak{g}^{r_2}\mathfrak{h}^{s_2} \Big] \\
\vee \Big[y_1 = g^{x_1} \ \wedge\ y_2 = g^{x_2} \ \wedge\ y_3 = g^{x_3} \ \wedge \\
C_3 = u^{2nr_3}(\frac{1}{e})^{2na_3} \ \wedge\ \mathfrak{C}_3 = \mathfrak{g}^{a_3}\mathfrak{h}^{b_3} \ \wedge\ 1 = (\frac{1}{\mathfrak{C}_3})^{x_1}\mathfrak{g}^{r_3}\mathfrak{h}^{s_3} \Big] \\
\vee \Big[y_1 = g^{x_1} \ \wedge\ y_2 = g^{x_2} \ \wedge\ y_3 = g^{x_3} \ \wedge \\
e^2 = u^{2x_1}h^{2m} \ \wedge \\
\mathfrak{W}_1 = \mathfrak{g}^{w_1}\mathfrak{h}^{t_1} \ \wedge\ \mathfrak{W}_2 = \mathfrak{g}^{w_2}\mathfrak{h}^{t_2} \ \wedge\ \mathfrak{W}_3 = \mathfrak{g}^{w_3}\mathfrak{h}^{t_3} \ \wedge\ \mathfrak{W}_4 = \mathfrak{g}^{w_4}\mathfrak{h}^{t_4} \ \wedge \\
\mathfrak{M} = \mathfrak{g}^{m}\mathfrak{h}^{t_5} \ \wedge\ \ \mathfrak{g}^{(n-1)^2/4} = \mathfrak{M}^{m}\mathfrak{W}_1^{w_1}\mathfrak{W}_2^{w_2}\mathfrak{W}_3^{w_3}\mathfrak{W}_4^{w_4}\mathfrak{h}^{s_5} \ \wedge \\
C_4 = \gamma^{r_4}(\frac{1}{\delta})^{a_4} \ \wedge\ \mathfrak{C}_4 = \mathfrak{g}^{a_4}\mathfrak{h}^{b_4} \ \wedge\ 1 = (\frac{1}{\mathfrak{C}_4})^{m}\mathfrak{g}^{r_4}\mathfrak{h}^{s_4} \Big] \Big\} \ ,
\end{aligned}
$$

where $r_1, \ldots, r_4, s_1, \ldots, s_4$ are temporary secrets (i.e.,

$$
\begin{aligned}
r_1 &= a_1(x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)x_3), & s_1 &= b_1(x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)x_3), \\
r_2 &= a_2(x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)x_3), & s_2 &= b_2(x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)x_3), \\
r_3 &= x_1 a_3, & s_3 &= x_1 b_3, \\
r_4 &= m a_4, & s_4 &= m b_4, \\
& & s_5 &= -t_5 m - \sum_{i=1}^{4} w_i t_i .
\end{aligned}
$$

(computed in $\mathbb{Z}$)). (To derive the actual protocol one has to apply the techniques by Cramer et al.[19] for realizing the $\vee$'s.)

(c) The verifier checks that $C_1^2 \neq 1$, $C_2^2 \neq 1$, $C_3^2 \neq 1$, and $C_4 \neq 1$.

Theorem 5. *Under the strong RSA assumption, the above scheme is a verifiable decryption scheme (for discrete logarithms).*

Proof. One needs to prove soundness, correctness and special honest-verifier zero-knowledge w.r.t. an oracle $f'(\delta, , \mathsf{SK}, \psi, L)$ that replies with 1 if $\delta = \gamma^{\hat{m}}$ where $\hat{m} = \mathcal{D}(\mathsf{SK}, \psi, L) \operatorname{rem} n$, or with -1 otherwise.

The following proof is very similar to the one of Theorem 4.

Correctness is by inspection.

We now show that the whole protocol is special honest-verifier computational zero-knowledge by providing a simulator.

First the simulator executes step 1 of the protocol as the decryptor would, that is, if $\delta \notin \Gamma$ or $v \neq \mathrm{abs}(v)$ it indicate this to the verifier, sends the verifier -1, and stops. Otherwise, the simulator chooses random $m, w_1, \ldots, w_4 \in_R [-n/2, n/2]$.

If $f'(\delta, \mathsf{SK}, \psi, L) = 1$, it simulates step 2 as follows. the simulator chooses $t_1, \ldots, t_5 \in_R [\mathfrak{n}/4]$ and computes $\mathfrak{W}_1 := \mathfrak{h}^{t_1}$, $\mathfrak{W}_2 := \mathfrak{h}^{t_2}$, $\mathfrak{W}_3 := \mathfrak{h}^{t_3}$, $\mathfrak{W}_4 := \mathfrak{h}^{t_4}$, and $\mathfrak{M} := \mathfrak{h}^{t_5}$. Then is sends the $\mathfrak{W}_1, \ldots, \mathfrak{W}_4$, and $\mathfrak{M}$ to the verifier and finally invokes the simulator for the PK-protocol of step 2.

If $f'(\delta, \mathsf{SK}, \psi, L) = 1$, it simulates step 3 as follows. The simulator chooses $b_1, b_2, b_3, b_3 \in_R t_1, \ldots, t_5[\mathfrak{n}/4]$. It then computes $\mathfrak{C}_1 := \mathfrak{h}^{b_1}$, $\mathfrak{C}_2 := \mathfrak{h}^{b_2}$, $\mathfrak{C}_3 := \mathfrak{h}^{b_3}$, and $\mathfrak{C}_4 := \mathfrak{h}^{b_4}$. $\mathfrak{W}_1 := \mathfrak{h}^{t_1}$, $\mathfrak{W}_2 := \mathfrak{h}^{t_2}$, $\mathfrak{W}_3 := \mathfrak{h}^{t_3}$, $\mathfrak{W}_4 := \mathfrak{h}^{t_4}$, and $\mathfrak{M} := \mathfrak{h}^{t_5}$. It chooses $C_1 \in_R \mathbf{G}_{n'}$, $C_2 \in_R \mathbf{G}_n$, $C_3 \in_R \mathbf{G}_{n'}$, and $C_4 \in_R \Gamma$. Next it invokes the simulator for the PK-protocol of step 3. This concludes the simulator.

The argument that this simulation actually works is rather similar to the one given in the proof of Theorem 4.

In the remainder we prove soundness. To this end, we present an algorithm that uses algorithms $\mathcal{A}^*$ and $\mathcal{P}^*$ and takes as input n and $(\mathfrak{n}, \mathfrak{z})$. We show that if $\mathcal{A}^*$ and $\mathcal{P}^*$ violate the soundness property with non-negligible probability then the algorithm outputs either a non-trivial root of $\mathfrak{z}$ or the factorization of n. with non-negligible probability. The first probability is w.r.t. the coin tosses of $\mathcal{G}$, $\mathcal{A}^*$, $\mathcal{P}^*$, and $\mathcal{V}$ while the second probability is w.r.t. the random choice of n and $\mathfrak{n}$ (drawn from the same distribution as induced by $\mathcal{G}$), as well as the random choice of $\mathfrak{z}$, and the coin tosses of the SRSA-breaking algorithm. Assume that $\mathfrak{z}$ is a random element of order $\mathfrak{n}'$. (As a random number mod $\mathfrak{n}$ is of this form with probability about $1/4$, this assumption degrades the success probability of our SRSA-breaking algorithm by this factor.)

We set $\mathfrak{g} := \mathfrak{z}$, choose a random $r \in_R [\mathfrak{n}^2]$ and set $\mathfrak{h} := \mathfrak{g}^r$. We choose random x_1, x_2, $x_3 \in_R [n^2/4]$, choose a random $g' \in_R \mathbb{Z}^*_{n^2}$, and compute $g := (g')^{2n}$, $y_1 := g^{x_1}$, $y_2 := g^{x_2}$, and $y_3 := g^{x_3}$.

Now we run $\mathcal{A}^*$ and $\mathcal{P}^*$ on input $\mathsf{PK} = ((n, g, y_1, y_2, y_3), (\mathfrak{n}, \mathfrak{g}, \mathfrak{h}))$ and $\mathsf{SK} = (x_1, x_2, x_3)$. By standard rewinding techniques we can produce two accepting conversations for either the PK protocol in step 2 or the one in step 3 (for different challenges but the same first message), depending on whether $\delta = \gamma^{\hat{m}}$ where $\hat{m} = \mathcal{D}(\mathsf{SK}, \psi, L) \operatorname{rem} n$, for (δ, ψ, L) provided by $\mathcal{A}^*$. We consider the two cases.

Case I. First assume that $\delta \neq \gamma^{\hat{m}}$ or $\mathsf{reject} = \mathcal{D}(\mathsf{SK}, \psi, L) \operatorname{rem} n$ but that V's output is 1. Let $(u, e, v) := \psi$. We can now get two accepting conversations of the PK protocol in step 2 and hence two answers

$$(\tilde{x}_1^{(1)}, \tilde{x}_2^{(1)}, \tilde{x}_3^{(1)}, \tilde{m}^{(1)}, \tilde{w}_1^{(1)}, \ldots, \tilde{w}_4^{(1)}, \tilde{t}_1^{(1)}, \ldots, \tilde{t}_5^{(1)}, \tilde{s}^{(1)})$$

and

$$(\tilde{x}_1^{(2)}, \tilde{x}_2^{(2)}, \tilde{x}_3^{(2)}, \tilde{m}^{(2)}, \tilde{w}_1^{(2)}, \ldots, \tilde{w}_4^{(2)}, \tilde{t}_1^{(2)}, \ldots, \tilde{t}_5^{(2)}, \tilde{s}^{(2)})$$

for the two different challenges $c^{(1)}$ and $c^{(2)}$ but with the same first message (here we use the same notation for the protocol variables as for the PK protocol in the previous section). W.l.o.g., suppose that $c^{(2)} > c^{(1)}$. Let $\Delta x_1 = \tilde{x}_1^{(1)} - \tilde{x}_1^{(2)}$, $\Delta x_2 = \tilde{x}_2^{(1)} - \tilde{x}_2^{(2)}$, $\Delta x_3 = \tilde{x}_3^{(1)} - \tilde{x}_3^{(2)}$, $\Delta m = \tilde{m}^{(1)} - \tilde{m}^{(2)}$, $\Delta w_1 = \tilde{w}_1^{(1)} - \tilde{w}_1^{(2)}$, …, $\Delta w_4 = \tilde{w}_4^{(1)} - \tilde{w}_4^{(2)}$, $\Delta t_1 = \tilde{t}_1^{(1)} - \tilde{t}_1^{(2)}$, …, $\Delta t_5 = \tilde{t}_5^{(1)} - \tilde{t}_5^{(2)}$, $\Delta s = \tilde{s}^{(1)} - \tilde{s}^{(2)}$, and $\Delta c = c^{(2)} - c^{(1)}$. From the verification equation of the PK protocol one can derive the following

equations:

$$y_1{}^{\Delta c} = g^{\Delta x_1} \qquad y_2{}^{\Delta c} = g^{\Delta x_2} \qquad y_3{}^{\Delta c} = g^{\Delta x_3} \tag{40}$$

$$v^{2\Delta c} = u^{2\Delta x_2} u^{2\mathcal{H}_{\mathsf{hk}}(u,e,L)\Delta x_3} \tag{41}$$

$$e^{2\Delta c} = u^{2\Delta x_1} h^{2\Delta m} \tag{42}$$

$$\mathfrak{W}_1^{\Delta c} = \mathfrak{g}^{\Delta w_1}\mathfrak{h}^{\Delta t_1} \qquad \mathfrak{W}_2^{\Delta c} = \mathfrak{g}^{\Delta w_2}\mathfrak{h}^{\Delta t_2} \qquad \mathfrak{W}_3^{\Delta c} = \mathfrak{g}^{\Delta w_3}\mathfrak{h}^{\Delta t_3} \qquad \mathfrak{W}_4^{\Delta c} = \mathfrak{g}^{\Delta w_4}\mathfrak{h}^{\Delta t_4} \tag{43}$$

$$\mathfrak{M}^{\Delta c} = \mathfrak{g}^{\Delta m}\mathfrak{h}^{\Delta t_5} \qquad \mathfrak{g}^{\Delta c(n-1)^2/4} = \mathfrak{M}^{\Delta m}\mathfrak{W}_1^{\Delta w_1}\mathfrak{W}_2^{\Delta w_2}\mathfrak{W}_3^{\Delta w_3}\mathfrak{W}_4^{\Delta w_4}\mathfrak{h}^{\Delta s} \tag{44}$$

$$\delta^{\Delta c} = \gamma^{\Delta m} \tag{45}$$

Consider the equations (43) and (44). If Δc does not divide all of Δm, Δw_1, …, Δw_4, Δt_1, …, Δt_5, and Δs we can compute a root of $\mathfrak{g} = \mathfrak{z}$ with probability 1/2 (see proof of Theorem 2 for how). If Δc does divide all of them, we compute $\hat{m} = \Delta m/\Delta c$, $\hat{w}_1 = \Delta w_1/\Delta c$, …, $\hat{w}_4 = \Delta w_4/\Delta c$, $\hat{t}_1 = \Delta t_1/\Delta c$, …, $\hat{t}_5 = \Delta t_5/\Delta c$, and $\hat{s} = \Delta s/\Delta c$ and we know that

$$\mathfrak{M} = \mathfrak{m}\mathfrak{g}^{\hat{m}}\mathfrak{h}^{\hat{t_5}} \quad \mathfrak{W}_1 = \mathfrak{w}_1\mathfrak{g}^{\hat{w}_1}\mathfrak{h}^{\hat{t}_1} \quad \mathfrak{W}_2 = \mathfrak{w}_2\mathfrak{g}^{\hat{w}_2}\mathfrak{h}^{\hat{t}_2} \quad \mathfrak{W}_3 = \mathfrak{w}_3\mathfrak{g}^{\hat{w}_3}\mathfrak{h}^{\hat{t}_3} \quad \mathfrak{W}_4 = \mathfrak{w}_4\mathfrak{g}^{\hat{w}_4}\mathfrak{h}^{\hat{t}_4} \quad \delta = \gamma^{\hat{m}} \tag{46}$$

holds for some $\mathfrak{m}$, $\mathfrak{w}_1$, $\mathfrak{w}_2$, $\mathfrak{w}_3$, and $\mathfrak{w}_4$ such that $\mathfrak{m}^2 = 1$ and $\mathfrak{w}_i^2 = 1$. Furthermore, we can rewrite the second equation of (44) as follows

$$\mathfrak{g}^{(n-1)^2/4} = \mathfrak{a}\mathfrak{g}^{\hat{m}^2 + \sum \hat{w}_i^2}\mathfrak{h}^{\hat{m}\hat{t}_5 + \sum \hat{w}_i\hat{t}_i + \hat{s}} \tag{47}$$

for some $\mathfrak{a}$ such that $\mathfrak{a}^2 = 1$. In fact, $\mathfrak{a} = 1$ as, first, $\mathfrak{a}$ must lie in $\langle\mathfrak{g}\rangle$ and, second, if $\mathfrak{a} \neq \pm 1$ then $\gcd(\mathfrak{a} - 1, \mathfrak{n})$ splits $\mathfrak{n}$. Let $\alpha := (n-1)^2/4 - \hat{m}^2 - \sum \hat{w}_i^2$ and $\beta := \hat{m}\hat{t}_5 + \sum \hat{w}_i\hat{t}_i + \hat{s}$. As we set $\mathfrak{h} = \mathfrak{g}^r$, we can rewrite (47) as

$$\mathfrak{g}^{\alpha} = \mathfrak{g}^{r\beta} \tag{48}$$

If $\alpha \neq r\beta$ (in $\mathbb{Z}$) we can factor n. Let $r = r_1 + \mathrm{ord}(\mathfrak{g})r_2$. Note that we choose r random from $[\mathfrak{n}^2]$ and that the adversary has no information about r_2. Therefore $\alpha = (r_1 + \mathrm{ord}(\mathfrak{g})r_2)\beta$ and $\alpha \neq 0$ can only happen with probability at most $1/\mathfrak{n}$, which is negligible. Therefore we must have

$$(n-1)^2/4 - \hat{m}^2 = \hat{w}_1^2 + \hat{w}_2^2 + \hat{w}_3^2 + \hat{w}_4^2 \quad (\text{in } \mathbb{Z})$$

and thus $(n-1)^2/4 - \hat{m}^2 \geq 0$ which is equivalent to

$$-(n-1)/2 \leq \hat{m} \leq (n-1)/2 \ . \tag{49}$$

Consider Equations (40-42). As n is the product of two safe primes p and q, we have $|\Delta c| < \min\{p, q, p'q'\}$ and hence Δc is invertible modulo $n'n$. By construction we know x_i such that $y_i = g^{x_i}$ and therefore it follows from (40) that

$$\Delta c\, x_i \equiv \Delta x_i \pmod{n'} \quad \text{for} \quad i = 1, \ldots, 3 \ . \tag{50}$$

Now we can either have $\mathcal{D}(\mathsf{SK}, \psi, L) = \mathsf{reject}$ or $\delta \neq \gamma^{(m \operatorname{rem} n)}$ where $m = \mathcal{D}(\mathsf{SK}, \psi, L) = \log_{h^2}(e/u^{x_1})^2$, i.e., one of the three statements

$$u^{2(x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)x_3)}/v^2 \neq 1 \qquad \text{or} \qquad (e/u^{x_1})^{2n} \neq 1 \qquad \text{or} \qquad (\frac{e}{u^{x_1}})^2 \neq h^{2\hat{m}} \tag{51}$$

must hold (cf. (39)), where the last is equivalent to $\delta \neq \gamma^{(m \operatorname{rem} n)}$ because Equation (49) and the fact that $-(n-1)/2 \leq (m \operatorname{rem} n) \leq (n-1)/2$.

We consider these three cases:

Case 1. If $u^{2(x_2+\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3)} \neq v^2$ we must have that $u^{2\Delta c(x_2+\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3)} \neq v^{2\Delta c} = u^{2\Delta x_2+\mathcal{H}_{\mathsf{hk}}(u,e,L)\Delta x_3}$ (from Equation (41) and because Δc is invertible modulo nn') and therefore also

$$\Delta c(x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)x_3) \not\equiv \Delta x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)\Delta x_3 \pmod{n'n} \ ,$$

as the order of u^2 divides $n'n$. From (50) it follows that

$$\Delta c(x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)x_3) \equiv \Delta x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)\Delta x_3 \pmod{n'} \ .$$

Therefore $\Delta c x_2 - \Delta x_2 + (\Delta c x_3 - \Delta x_3)\mathcal{H}_{\mathsf{hk}}(u,e,L)$ must be a non-zero multiple of n' and we can factor n.

Case 2. If $u^{2nx_1} \neq e^{2n}$ we have that $u^{2n\Delta c x_1} \neq e^{2n\Delta c}$. Because of (42) and $h^n = 1$, we get

$$u^{2n\Delta c x_1} \neq u^{2n\Delta x_1} \qquad \text{and thus} \qquad \Delta c x_1 \not\equiv \Delta x_1 \pmod{n'} \ ,$$

because u^{2n} has order dividing n'. The latter, however, is a contradiction to Eqn. (50) and thus this case cannot occur.

Case 3. From $(\frac{e}{u^{x_1}})^2 \neq h^{2\hat{m}}$ is equivalent to $\frac{e^2}{h^{2\hat{m}}} \neq u^{2x_1}$. Recalling that $\hat{m}\Delta c = \Delta m$ we can rewrite (42) as

$$\frac{e^{2\Delta c}}{h^{2\Delta m}} = \Big(\frac{e}{h^{\hat{m}}}\Big)^{2\Delta c} = u^{2\Delta x_1} \qquad \text{and conclude that} \qquad u^{2\Delta c x_1} \neq u^{2\Delta x_1} \ .$$

Similarly to case 1, it follows that $\Delta c x_1 - \Delta x_1$ is a multiple of n' and we are again able to factor n.

Case II. It remains to consider the case when V's output is -1 but $\delta = \gamma^{(\mathcal{D}(\mathsf{SK},\psi,L) \operatorname{rem} n)}$ holds. Let $(u,e,v) := \psi$. Now all the three equations

$$u^{2(x_2+\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3)}/v^2 = 1 \qquad (e/u^{x_1})^{2n} = 1 \qquad \delta = \gamma^{(\log_{h^2}(e/u^{x_1})^2 \operatorname{rem} n)} \tag{52}$$

must hold. As usual we obtain two accepting conversation of the PK protocol in step 3 and thus two answers

$$(\tilde{x}_1^{(1)}, \tilde{x}_2^{(1)}, \tilde{x}_3^{(1)}, \tilde{a}_1^{(1)}, \ldots, \tilde{a}_6^{(1)}, \tilde{b}_1^{(1)}, \ldots, \tilde{b}_4^{(1)}, \tilde{r}_1^{(1)}, \ldots, \tilde{r}_6^{(1)}, \tilde{s}_1^{(1)}, \ldots, \tilde{s}_5^{(1)}, \tilde{t}_1^{(1)}, \ldots, \tilde{t}_5^{(1)}, \tilde{w}_1^{(1)}, \ldots, \tilde{w}_4^{(1)}, \tilde{m}_1^{(1)})$$

and

$$(\tilde{x}_1^{(2)}, \tilde{x}_2^{(2)}, \tilde{x}_3^{(2)}, \tilde{a}_1^{(2)}, \ldots, \tilde{a}_4^{(2)}, \tilde{b}_1^{(2)}, \ldots, \tilde{b}_4^{(2)}, \tilde{r}_1^{(2)}, \ldots, \tilde{r}_4^{(2)}, \tilde{s}_1^{(2)}, \ldots, \tilde{s}_5^{(2)}, \tilde{t}_1^{(2)}, \ldots, \tilde{t}_5^{(2)}, \tilde{w}_1^{(2)}, \ldots, \tilde{w}_4^{(2)}, \tilde{m}_1^{(2)})$$

for the two different challenges $c^{(1)}$ and $c^{(2)}$ but with the same first message (here we use the same notation for the protocol variables as for the PK protocol in the previous section and left out an

intermediate step that deals with the $\vee$'s (c.f. [19])). W.l.o.g., suppose that $c^{(2)} > c^{(1)}$. Let

$$\begin{aligned}
\Delta x_i &= \tilde{x}_i^{(1)} - \tilde{x}_i^{(2)} \ (i = 1, \ldots, 3); & \Delta a_i &= \tilde{a}_i^{(1)} - \tilde{a}_i^{(2)} \ (i = 1, \ldots, 4); \\
\Delta b_i &= \tilde{b}_i^{(1)} - \tilde{b}_i^{(2)} \ (i = 1, \ldots, 4); & \Delta r_i &= \tilde{s}_i^{(1)} - \tilde{r}_i^{(2)} \ (i = 1, \ldots, 4); \\
\Delta s_i &= \tilde{b}_i^{(1)} - \tilde{s}_i^{(2)} \ (i = 1, \ldots, 5); & \Delta t_i &= \tilde{s}_i^{(1)} - \tilde{t}_i^{(2)} \ (i = 1, \ldots, 5); \\
\Delta w_i &= \tilde{r}_i^{(1)} - \tilde{w}_i^{(2)} \ (i = 1, \ldots, 4); & \Delta m &= m^{(1)} - c^{(2)}; \\
\Delta c &= c^{(2)} - c^{(1)} \ . &&
\end{aligned}$$

From the verification equation of the *PK* protocol one can derive that

$$y_1{}^{\Delta c} = g^{\Delta x_1} \ , \qquad y_2{}^{\Delta c} = g^{\Delta x_2} \ , \quad \text{and} \qquad y_3{}^{\Delta c} = g^{\Delta x_3} \ , \tag{53}$$

hold and either

$$C_1^{\Delta c} = u^{2n\Delta r_1}(\frac{1}{v})^{2n\Delta a_1} \ , \quad \mathfrak{C}_1^{\Delta c} = \mathfrak{g}^{\Delta a_1}\mathfrak{h}^{\Delta b_1} \ , \quad \text{and} \quad 1 = (\frac{1}{\mathfrak{C}_1})^{\Delta x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)\Delta x_3}\mathfrak{g}^{\Delta r_1}\mathfrak{h}^{\Delta s_1} \tag{54}$$

or

$$C_2^{\Delta c} = u^{2\Delta r_2}(\frac{1}{v})^{2\Delta a_2} \ , \qquad \mathfrak{C}_2^{\Delta c} = \mathfrak{g}^{\Delta a_2}\mathfrak{h}^{\Delta b_2} \ , \quad \text{and} \quad 1 = (\frac{1}{\mathfrak{C}_2})^{\Delta x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)\Delta x_3}\mathfrak{g}^{\Delta r_2}\mathfrak{h}^{\Delta s_2} \tag{55}$$

or

$$C_3^{\Delta c} = u^{2n\Delta r_3}(\frac{1}{e})^{2n\Delta a_3} \ , \quad \mathfrak{C}_3^{\Delta c} = \mathfrak{g}^{\Delta a_3}\mathfrak{h}^{\Delta b_3} \ , \quad \text{and} \quad 1 = (\frac{1}{\mathfrak{C}_3})^{\Delta x_1}\mathfrak{g}^{\Delta r_3}\mathfrak{h}^{\Delta s_3} \tag{56}$$

or

$$C_4^{\Delta c} = \gamma^{\Delta r_4}(\frac{1}{\delta})^{\Delta a_4} \ , \qquad \mathfrak{C}_4^{\Delta c} = \ \mathfrak{g}^{\Delta a_4}\mathfrak{h}^{\Delta b_4} \ , \qquad 1 = (\frac{1}{\mathfrak{C}_4})^{\Delta m}\mathfrak{g}^{\Delta r_4}\mathfrak{h}^{\Delta s_4} \tag{57}$$

$$e^{2\Delta c} = u^{2\Delta x_1}h^{2\Delta m} \ , \quad \mathfrak{M}^{\Delta c} = \mathfrak{g}^{\Delta m}\mathfrak{h}^{\Delta t_5} \ , \quad \mathfrak{g}^{\Delta c(n-1)^2/4} = \mathfrak{M}^{\Delta m}\mathfrak{W}_1^{\Delta w_1}\mathfrak{W}_2^{\Delta w_2}\mathfrak{W}_3^{\Delta w_3}\mathfrak{W}_4^{\Delta w_4}\mathfrak{h}^{\Delta s_5} \tag{58}$$

$$\mathfrak{W}_1^{\Delta c} = \mathfrak{g}^{\Delta w_1}\mathfrak{h}^{\Delta t_1} \ , \quad \mathfrak{W}_2^{\Delta c} = \mathfrak{g}^{\Delta w_2}\mathfrak{h}^{\Delta t_2} \ , \quad \mathfrak{W}_3^{\Delta c} = \mathfrak{g}^{\Delta w_3}\mathfrak{h}^{\Delta t_3} \ , \quad \text{and} \quad \mathfrak{W}_4^{\Delta c} = \mathfrak{g}^{\Delta w_4}\mathfrak{h}^{\Delta t_4} \ . \tag{59}$$

hold. We know x_i such that $y_i = g^{x_i}$ and therefore it follows from (29) that

$$\Delta c\, x_i \equiv \Delta x_i \pmod{n'} \quad \text{for} \quad i = 1, \ldots, 3 \ . \tag{60}$$

We next consider the implications of the cases when the equations (54), the equations (55), the equations (56), or the equations (57-59) hold in conjunction with (53). The first three cases appear also in the proof of Theorem 4, while the last one is different:

Case 4. Similarly as in Case I above, from the Equations (58) and (59) we can derive that

$$e^{2\Delta c} = u^{2\Delta x_1}h^{2\Delta c\hat{m}} \qquad \text{and} \qquad -(n-1)/2 \leq \hat{m} \leq (n-1)/2 \tag{61}$$

where $\hat{m} = \Delta m/\Delta c$. Using Equations (60) and the fact that Δc is invertible modulo nn', we get

$$e^2 = u^{2x_1}h^{2\hat{m}} \ ,$$

and, because of the second equation of (61),

$$\hat{m} = (\log_{h^2} u^{2x_1}/e^2 \operatorname{rem} n) \tag{62}$$

Similarly as we did in case II in the proof of Theorem 4, one can derive from the last two equations of (57) that

$$\Delta r_4 = \Delta a_4 \hat{m} \tag{63}$$

holds (or one factors $\mathfrak{n}$ and thus compute a non-trivial root of $\mathfrak{z}$). Now using (63) in the first equation of (57)

$$C_4^{\Delta c} = \gamma^{\Delta a_4 \hat{m}} (\frac{1}{\delta})^{\Delta a_4} \qquad \text{and} \qquad C_4 = (\frac{\gamma^{\hat{m}}}{\delta})^{\hat{a}_4} \ , \tag{64}$$

where $\hat{a}_4 := \Delta a_4 / \Delta c \pmod{\rho}$. Because $C_4 \neq 1$ we must have that and because of (62)

$$\delta \neq \gamma^{(\log_{h^2} u^{2x_1}/e^2 \operatorname{rem} n)} \ ,$$

which is a contradiction to the third equation of (52) and hence this case can not occur.

□

6.4 Application to Confirmer Signature Scheme

We apply the generic construction for a confirmer signature scheme by Camenisch and Michels [13]: The signer chooses a public and secret key of any signature scheme and the confirmer chooses a public and secret key of any encryption scheme. To sign a message, the signer uses the signing algorithm to produce an ordinary signature on the message and then encrypts the signature under the confirmer's public key with the label containing the required additional information such as the confirmation policy. This encryption becomes the confirmer-signature on the message. The confirmer can confirm or disavow a confirmer-signature to a verifier as follows. She first decrypts the confirmer-signature and, if decryption does not fail, checks whether the decrypted value constitutes an (ordinary) signature on the message under the signer's public key. If any of these checks failed, she tells that the verifier that the signature is invalid; otherwise, she tells the verifier that it is valid. Finally she proves to the verifier in zero-knowledge the correctness of her statement. To convert a confirmer-signature into an ordinary one, the confirmer just publishes the decryption of the confirmer-signature. Note that the resulting signature is one of the signature scheme selected by the signer, that is, we have perfect convertibility (c.f. [13]).

Instantiation of this construction with our encryption scheme, our protocol for verifiable decryption of a discrete logarithm (together with a suitable method for converting it from special honest-verifier zero-knowledge into real zero-knowledge), and the signature reduction technique [1, 27] for discrete-logarithm-based signature schemes such as DSS or Schnorr gives us an efficient confirmer signature scheme with perfect conversion w.r.t. these signature schemes. We note that the recovery information produced by the signature reduction algorithm as well as the confirmation policy must be included in label input to the encryption algorithm.

The resulting scheme can be proven secure in the model of Camenisch and Michels [13] and is an order of magnitude more efficient than the previously known ones that provide perfect conversion (the known schemes all apply the cut-and-choose paradigm to realize the confirmer's proof of correctness of her statement).

References

[1] N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. In K. Nyberg, editor, *Advances in Cryptology — EUROCRYPT '98*, volume 1403 of *LNCS*, pages 591–606. Springer Verlag, 1998.

[2] N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. *IEEE Journal on Selected Areas in Communications*, 18(4):591–610, Apr. 2000.

[3] G. Ateniese. Efficient verifiable encryption (and fair exchange) of digital signatures. In *Proc. 6th ACM Conference on Computer and Communications Security*, pages 138–146. ACM press, Nov. 1999.

[4] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In M. Bellare, editor, *Advances in Cryptology — CRYPTO 2000*, volume 1880 of *LNCS*, pages 255–270. Springer Verlag, 2000.

[5] F. Bao. An efficient verifiable encryption scheme for the encryption of discrete logarithms. In J.-J. Quisquater and B. Schneier, editors, *Smart Card Research and Applications (CARDIS '98)*, volume 1820 of *LNCS*. Springer Verlag, 2000.

[6] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *First ACM Conference on Computer and Communication Security*, pages 62–73. Association for Computing Machinery, 1993.

[7] J. C. Benaloh and D. Tuinstra. Receipt-free secret-ballot elections (extended abstract). In *Proc. 26th Annual ACM Symposium on Theory of Computing (STOC)*, pages 544–553. ACM, 1994.

[8] F. Boudot. Efficient proofs that a committed number lies in an interval. In B. Preneel, editor, *Advances in Cryptology — EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 431–444. Springer Verlag, 2000.

[9] E. Bresson and J. Stern. Proofs of knowledge for non-monotone discrete-log formulae and applications. In A. H. Chan and V. Gligor, editors, *Information Security (ISC 2002)*, volume 2433 of *LNCS*, pages 272–288. Springer Verlag, 2002.

[10] J. Camenisch and I. Damgård. Verifiable encryption, group encryption, and their applications to group signatures and signature sharing schemes. In T. Okamoto, editor, *Advances in Cryptology — ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 331–345. Springer Verlag, 2000.

[11] J. Camenisch and A. Lysyanskaya. An identity escrow scheme with appointed verifiers. In J. Kilian, editor, *Advances in Cryptology — CRYPTO 2001*, volume 2139 of *LNCS*, pages 388–407. Springer Verlag, 2001.

[12] J. Camenisch, U. Maurer, and M. Stadler. Digital payment systems with passive anonymity-revoking trustees. In E. Bertino, H. Kurth, G. Martella, and E. Montolivo, editors, *Computer Security — ESORICS 96*, volume 1146 of *LNCS*, pages 33–43. Springer Verlag, 1996.

[13] J. Camenisch and M. Michels. Confirmer signature schemes secure against adaptive adversaries. In B. Preneel, editor, *Advances in Cryptology — EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 243–258. Springer Verlag, 2000.

[14] J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In B. Kaliski, editor, *Advances in Cryptology — CRYPTO '97*, volume 1296 of *LNCS*, pages 410–424. Springer Verlag, 1997.

[15] R. Canetti, O. Goldreich, S. Goldwasser, and S. Micali. Resettable zero-knowledge. In *Proc. 32st Annual ACM Symposium on Theory of Computing (STOC)*, pages 235–244. ACM Press, 2000.

[16] R. Canetti and S. Goldwasser. An efficient threshold public key cryptosystem secure against adaptive chosen ciphertext attack. In J. Stern, editor, *Advances in Cryptology — EUROCRYPT '99*, volume 1592 of *LNCS*, pages 90–106. Springer Verlag, 1999.

[17] D. Chaum. Designated confirmer signatures. In A. De Santis, editor, *Advances in Cryptology — EUROCRYPT '94*, volume 950 of *LNCS*, pages 86–91. Springer Verlag Berlin, 1994.

[18] D. Chaum and T. P. Pedersen. Wallet databases with observers. In E. F. Brickell, editor, *Advances in Cryptology — CRYPTO '92*, volume 740 of *LNCS*, pages 89–105. Springer-Verlag, 1993.

[19] R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Y. G. Desmedt, editor, *Advances in Cryptology — CRYPTO '94*, volume 839 of *LNCS*, pages 174–187. Springer Verlag, 1994.

[20] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In H. Krawczyk, editor, *Advances in Cryptology — CRYPTO '98*, volume 1642 of *LNCS*, pages 13–25, Berlin, 1998. Springer Verlag.

[21] R. Cramer and V. Shoup. Signature schemes based on the strong RSA assumption. *ACM Transactions on Information and System Security*, 3(3):161–185, 2000.

[22] R. Cramer and V. Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. `http://eprint.iacr.org/2001/108`, 2001.

[23] I. Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In B. Preneel, editor, *Advances in Cryptology — EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 431–444. Springer Verlag, 2000.

[24] I. Damgård and E. Fujisaki. An integer commitment scheme based on groups with hidden order. `http://eprint.iacr.org/2001/064`, 2001.

[25] C. Dwork, M. Naor, and A. Sahai. Concurrent zero knowledge. In *Proc. 30th Annual ACM Symposium on Theory of Computing (STOC)*, 1998.

[26] Y. Frankel, Y. Tsiounis, and M. Yung. "Indirect discourse proofs:" Achieving efficient fair off-line e-cash. In K. Kim and T. Matsumoto, editors, *Advances in Cryptology — ASIACRYPT '96*, volume 1163 of *LNCS*, pages 286–300. Springer Verlag, 1996.

[27] M. Franklin and M. Reiter. Verifiable signature sharing. In L. C. Guillou and J.-J. Quisquater, editors, *Advances in Cryptology — EUROCRYPT '95*, volume 921 of *LNCS*, pages 50–63. Springer Verlag, 1995.

[28] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, Apr. 1984.

[29] M. Hirt and K. Sako. Efficient receipt-free voting based on homomorphic encryption. In B. Preneel, editor, *Advances in Cryptology — EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 539–556. Springer Verlag, 2000.

[30] Y. D. Jee Hea An and T. Rabin. On the security of joint signature and encryption. In L. Knudsen, editor, *Advances in Cryptology: EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 83–107. Springer, 2002.

[31] J. Kilian and E. Petrank. Identity escrow. In H. Krawczyk, editor, *Advances in Cryptology — CRYPTO '98*, volume 1642 of *LNCS*, pages 169–185, Berlin, 1998. Springer Verlag.

[32] P. MacKenize and M. K. Reiter. Two-party generation of DSA signatures. In J. Kilian, editor, *Advances in Cryptology — CRYPTO 2001*, volume 2139 of *LNCS*, pages 137–154. Springer Verlag, 2001.

[33] M. Michels and M. Stadler. Generic constructions for secure and efficient confirmer signature schemes. In K. Nyberg, editor, *Advances in Cryptology — EUROCRYPT '98*, volume 1403 of *LNCS*, pages 406–421. Springer Verlag, 1998.

[34] P. Paillier. Public-key cryptosystems based on composite residuosity classes. In J. Stern, editor, *Advances in Cryptology — EUROCRYPT '99*, volume 1592 of *LNCS*, pages 223–239. Springer Verlag, 1999.

[35] G. Poupard and J. Stern. Fair encryption of RSA keys. In B. Preneel, editor, *Advances in Cryptology: EUROCRYPT 2000*, volume 1087 of *LNCS*, pages 173–190. Springer Verlag, 2000.

[36] M. O. Rabin and J. O. Shallit. Randomized algorithms in number theory. *Communications on Pure and Applied Mathematics*, 39:239–256, 1986.

[37] C. Rackoff and D. R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In J. Feigenbaum, editor, *Advances in Cryptology: CRYPTO '91*, volume 576 of *LNCS*, pages 433–444. Springer, 1992.

[38] V. Shoup. A proposal for an iso standard for public key encryption. `http://eprint.iacr.org/2001/112`, 2001.

[39] V. Shoup and R. Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. In K. Nyberg, editor, *Advances in Cryptology: EUROCRYPT '98*, volume 1403 of *LNCS*. Springer, 1998.

[40] M. Stadler. Publicly verifiable secret sharing. In U. Maurer, editor, *Advances in Cryptology — EUROCRYPT '96*, volume 1070 of *LNCS*, pages 191–199. Springer Verlag, 1996.

[41] A. Young and M. Young. Auto-recoverable auto-certifiable cryptosystems. In K. Nyberg, editor, *Advances in Cryptology — EUROCRYPT '98*, volume 1403 of *LNCS*, pages 17–31. Springer Verlag, 1998.