Practical Verifiable Encryption and Decryption of Discrete Logarithms

Jan Camenisch[1] and Victor Shoup[2]

[1] IBM Zürich Research Lab <jca@zurich.ibm.com>
[2] New York University <shoup@cs.nyu.edu>

Abstract. This paper addresses the problem of designing practical protocols for proving properties about encrypted data. To this end, it presents a variant of the new public key encryption of Cramer and Shoup based on Paillier's decision composite residuosity assumption, along with efficient protocols for verifiable encryption and decryption of discrete logarithms (and more generally, of representations with respect to multiple bases). This is the first verifiable encryption system that provides chosen ciphertext security and avoids inefficient cut-and-choose proofs. The presented protocols have numerous applications, including key escrow, optimistic fair exchange, publicly verifiable secret and signature sharing, universally composable commitments, group signatures, and confirmer signatures.

1 Introduction

This paper concerns itself with the general problem of *proving properties about encrypted data*. In the case of public-key encryption, which is the setting in which we are interested here, there are two parties who are in a position to prove some property to another party about an encrypted message — namely, the party who created the ciphertext, and the party who holds the secret key. A protocol in which the encryptor is the prover is a *verifiable encryption* protocol, while a protocol in which the prover is the decryptor is a *verifiable decryption* protocol.

For example, suppose a party T has a public key/secret key pair $(\mathsf{PK}, \mathsf{SK})$ for a public key encryption scheme. Party A might encrypt, using T's public PK, a secret message m that satisfies a publicly-defined property θ, and give the resulting ciphertext ψ to another party B. The latter party might demand that A prove that ψ is an encryption of a message satisfying property θ. Ideally, the proof should be "zero knowledge," so that no unnecessary information about m is leaked to B as part of the proof. Another party B' might obtain the ciphertext ψ, and may request that T prove or disprove that ψ decrypts under SK to a message m satisfying a publicly-defined property θ'; a special case of this would be the situation where T simply gives m to B, and proves to B that the decryption was performed correctly. Again, ideally, the proof should be "zero knowledge."

Now, if one expects to obtain reasonably practical protocols for this problem, it seems necessary to restrict the type of properties that protocols should work with. In this paper, we consider only properties related to the discrete logarithm problem. The message m encrypted by A above is the discrete logarithm of an element δ with respect to a base γ, and A proves to B that ψ is an encryption $\log_\gamma \delta$ under T's public key PK. Here, the common inputs to A and B in the proof protocol are PK, ψ, δ, and γ. Similarly, when a party B' presents ψ to T for decryption, T may state and prove whether or nor ψ decrypts to $\log_\gamma \delta$, or alternatively, T may give the decryption of ψ to B', and simply prove that the decryption was performed correctly. We also consider the obvious generalizations from discrete logarithms to representations with respect to several bases — i.e., proving that a ciphertext is an encryption of $(m_1, \ldots, m_k)$ such that $\delta = \gamma_1^{m_1} \cdots \gamma_k^{m_k}$.

Although the restriction to properties related to the discrete logarithm problem may seem excessive, it turns out (as we discuss in some detail below) that protocols for proving such properties have many useful applications in cryptography, including key escrow, optimistic fair exchange, publicly verifiable secret and signature sharing, universally composable commitments, group signatures, and confirmer signatures. One reason why this restriction is not really so excessive is because in the past few years, efficient protocols for proving numerous properties about committed values — using Pedersen's commitment scheme [Ped92] and generalizations to groups of unknown order — have been developed (c.f., [FO97,DF02,Bou00]); by using our scheme for verifiable encryption of a representation (i.e., an opening of a commitment), we immediately get corresponding protocols for proving properties about encrypted values.

The contribution of this paper is to present and analyze an efficient public-key encryption scheme, together with a suite of proof protocols for the properties related to the discrete logarithm problem outlined above. The encryption scheme is a variant of the new public key encryption of Cramer and Shoup based on Paillier's decision composite residuosity assumption, suitably modified so as to support our proof protocols. The proof protocols are all of the usual, three move "Σ-protocol" type, satisfying the usual, and very strong conditions of special honest verifier zero knowledge and special soundness. We note that any such protocol can be easily and efficiently converted into a "real" zero knowledge protocol using well known techniques, e.g., [Dam00]. Our system for verifiable encryption of discrete logarithms is the first one that provides chosen ciphertext security and avoids inefficient cut-and-choose proofs. It is also the first practical system for verifiable decryption of discrete logarithms.

Although our protocols do not rely on the random oracle heuristic, we hasten to point out that even allowing this heuristic, our protocols are much more efficient than previously known protocols for these problems.

1.1 Applications

In this section, we outline some of the numerous applications of verifiable encryption and decryption of discrete logarithms and representations. For all of them our protocols, used together with the existing solutions, yields more efficient solutions or adds security to chosen ciphertext attacks.

Key escrow. Party A may encrypt its own secret key for an asymmetric cryptographic primitive under the public key of a trusted third party T, and present to a second party B the ciphertext ψ and a proof that ψ is indeed an encryption of it's secret key. This problem area has attracted a good deal of attention, with specific schemes being proposed in [Sta96,BG96,YY98,ASW00,PS00].

Now, if A's secret key is, say, a key for a discrete log based scheme, such as Schnorr or DSS signatures or ElGamal encryption, we can use our verifiable encryption protocol directly. We note that for this and other applications, it is important to be able to bind some public data, called a *label*, to the ciphertext at both encryption and decryption time. In this application, user A would attach a label to ψ that indicates the conditions under which ψ should be decrypted, e.g., A's identity and perhaps and expiration date. The definition of chosen ciphertext security ensures that decrypting a ciphertext under any label different from the label used to create the ciphertext reveals no information about the original encrypted message.

Even though T is "trusted," it might be nice to minimize the trust we need to place in T. To this end, verifiable decryption comes in handy — we can force T to prove that it performed the decryption operation correctly. Of course, this does not prevent T from misbehaving in other ways, such as divulging a secret key to an unauthorized party.

If A's secret key is for a factoring based scheme, one can still use our protocol for verifiable encryption of a representation. One can use Pedersen's commitment scheme to commit to some quantity related to the secret key, and then use an appropriate protocol to prove that the committed value is indeed the right one, together with our protocol to prove that the encryption contains an opening of the commitment. The quantity committed to could be the factorization of an RSA modulus, the decryption exponent of an RSA scheme, or an appropriate root in a Guillou-Quisquater scheme — there are (not too terribly inefficient) protocols for proving that a committed value is of such a form [FO97,CM99a,DF02,PS00,Bou00].

Optimistic fair exchange. Two parties A and B want to exchange some valuable digital data (e.g., signatures on a contract, e-cash), but in a fair way: either each party obtains the other's data, or neither party does. One way to do this is by employing a trusted third party T, but, for the sake of efficiency, with T only involved in crisis situations. One approach to this problem is to have both parties verifiably encrypt to each other their data under T's public key, and only then do they reveal their data to each other — if one party backs out unexpectedly, the other can go to T to obtain the required data. The general problem of optimistic fair exchange has been extensively studied, c.f., [ASW97,BDM98,BP90,Mic,ASW00], while the solution using verifiable encryption was studied in detail in [ASW00].

Our scheme for verifiable encryption may be used directly to efficiently implement the fair exchange of Schnorr or DSS signatures. As outlined in [ASW00], if the public key of the Schnorr signature scheme consists of the base γ and the group element $\alpha = \gamma^x$, and A has a signature on a message m of the form (β, c, s), where $\beta = \gamma^r$, $c = H(\beta, m)$, $s = r + xc \bmod \rho$, and ρ is the group size, then A gives to B the triple (β, c, δ), where $\delta = \gamma^s$, along with an encryption ψ of s under T's public key, and proves to B that ψ is an encryption of $\log_\gamma \delta$. In addition to checking the proof that ψ is a correct encryption of $\log_\gamma \delta$, B also checks that $\delta = \beta\gamma^c$; with these checks, B can be sure that if the need arises, ψ can be decrypted so as to obtain a signature on m. As argued in [ASW00], this technique of reducing a signature to a discrete logarithm does not make it any easier for anyone to forge a signature. Moreover, as discussed in [ASW00], similar techniques can be used to facilitate the fair exchange of other items, such as electronic cash.

As in the escrow application, the label mechanism plays a crucial role here, helping to enforce the logic of the exchange protocol, and a verifiable decryption protocol may be used to hold T's feet to the fire.

Publicly verifiable secret sharing and signature sharing. Stadler [Sta96] introduced the notion of *publicly verifiable secret sharing*. Here, one party, the dealer, shares a secret with several proxies $P_1, \ldots, P_n$, in such a way that a third party (other than the dealer and the proxies) can verify that the sharing was done correctly. This can be done quite simply by sharing the secret using Shamir's secret sharing scheme: the dealer encrypts P_i's share under P_i's public key, and gives to the third party commitments to these shares, along with commitments to the coefficients of the blinding polynomial, and all of the ciphertexts, and proves to to the third party that the ciphertexts encrypt openings of the commitments to the shares. Since the openings to the commitments are just discrete logarithms, verifiable encryption of discrete logarithms is just the right tool.

Using the notion discussed above above for reducing a signature to a discrete logarithm, one can easily implement a (publicly) verifiable *signature* sharing scheme [FR95,CG98] for Schnorr and DSS signatures.

These two applications of verifiable encryption were discussed in [CD00].

Universally composable commitments. The notion of *universally composable (UC) commitments*, introduced by Canetti and Fischlin [CF01], is a very strong notion of security for a commitment scheme. It basically says that commitments in the real world acts like commitments in an ideal world in which, when a party A commits to a value x to a party B, A presents x to an idealized trusted party T (that does not exist in the real world), and when A opens the commitment, T gives x to B. In the ideal world, no information about x is revealed to B prior to opening, and A is forced to fix the value committed to when the commitment protocol runs.

This notion of security is so strong, in fact, that it can only be realized in the *common reference string (CRS)* model, where all parties have access to a string that was generated by a trusted party according to some prescribed distribution. In the CRS model, the simulator S in the ideal world is given the privilege of generating the common reference string, and so S may know some "side information" related to the common reference string that is not available to anyone in the real world.

Verifiable encryption of a representation may be used to implement UC commitments in the CRS model, as follows. The CRS consists of a public key for the encryption scheme, along with bases γ_1 and γ_2 for some suitable group. When A commits a value x to B, he creates a Pedersen commitment $C = \gamma_1^x \gamma_2^r$, and an encryption ψ of the representation (x, r) of C with respect to (γ_1, γ_2). A then gives (C, ψ) to B, and proves to B that ψ indeed decrypts to a representation of C. In order to satisfy the definition of security for UC commitments, and in particular, to prevent "man in the middle attacks," a label containing A's identity should be attached to ψ.

The reason this is secure is that the simulator S in the CRS model knows the secret key to the encryption scheme, which allows him to "extract" values committed by corrupted parties, and S knows the discrete logarithm of γ_2 with respect to γ_1, which allows him to "equivocate" values committed by honest parties. The proof that ψ is an encryption of a representation C ensures that the value extracted by the simulator at commitment time agrees with the value revealed at opening time.

The details of this construction and security proof are the subject of a forthcoming paper.

Confirmer signatures. In a confirmer signature scheme, a notion introduced in [Cha94], a party A creates an "opaque signature" ψ on a message m, which cannot be verified by any other party except a designated trusted third party T, who may either confirm or deny the validity of the signature to another party B. Under appropriate circumstances, T may also *convert* ψ into an ordinary signature, which may then be verified by anybody. Additionally, the party A may prove the validity of an opaque signature ψ to a party B, at the time that A creates and gives ψ to B. As described in [CM00], one may implement confirmer signatures as follows: A creates an ordinary signature σ on m, and encrypts σ under T's public key. Using verifiable encryption, A may prove to B that the resulting ciphertext ψ indeed encrypts a valid signature on m, and using verifiable decryption, T may confirm or deny the validity of ψ, or alternatively, just decrypt ψ, thus converting it to the ordinary signature σ. To implement this idea for Schnorr signatures, one again uses the idea outlined in above for reducing signatures to discrete logarithms. The details of all this are the subject of a forthcoming paper.

Group signatures and anonymous credentials. In a group signature scheme (see [ACJT00,KP98,CD00]), when a user joins a group (whose membership is controlled by a special party, called the *group manager*), the user may sign messages on behalf of the group, without revealing his individual identity; however, under appropriate circumstances, the identity of the individual who actually signed a particular message may be revealed (using a special party, called the *anonymity revocation manager*, which may be distinct from the group manager).

Without going into too many details, verifiable encryption may be used in the following way as a component in such a system. When a group member signs a message, he encrypts enough information under the public key of the anonymity revocation manager, so that later, if the identity of the signer needs to be revealed, this information can be decrypted. To prove that this information correctly identifies the signer, he makes a Pedersen commitment to this information, proves that the committed value identifies the user, encrypts the opening of the commitment, and proves that the ciphertext decrypts to an opening of the commitment. To turn this into a signature scheme, one must use the Fiat-Shamir heuristic [FS87] to make it non-interactive (the interactive version is called an *identity escrow* scheme).

Although one can implement group signatures without it, by using verifiable encryption, one can build a more modular system, in which the group manager and anonymity manager are separate entities with independently generated public keys. As pointed out in [KP97,CM99b,ASW00] such *separability* in system design is highly desirable in practice. Verifiable decryption can be used both to ensure the correct behavior of the anonymity revocation manager (preventing it from "framing" innocent users), and to allow even more fine-grained control of anonymity revocation: instead of simply revealing the identity of a particular signer, the anonymity revocation manager can state (and prove) whether or not a particular signature was generated by a particular user.

Credential systems [Cha85,CL01] are a generalization of group signatures that allow users to show credentials to various organizations, and obtain new credentials, without revealing their identity, except through the use of an anonymity revocation manager. Verifiable encryption can be used as a component in such systems in a manner similar to that described above for group signatures. In fact, our verifiable encryption scheme is used in a prototype credential system developed at IBM called *IDEMIX* [CVH02].

1.2 Previous work and further discussion

In all applications mentioned in §1.1, it is essential that the underlying encryption scheme provide security against chosen ciphertext attacks. As pointed out in [ASW00], the earlier work on verifiable encryption in [Sta96,BG96,YY98] overlooked this fact, as does [PS00].

Our encryption scheme and proof protocols are quite efficient. In particular, the proof protocols are conventional "Σ-protocols," rather than the generally more expensive "cut and choose" protocols, such as in [Sta96,BG96,YY98,ASW00], that have been previously designed for the problem of verifiable encryption. Moreover, our verifiable encryption scheme actually produces a proof that a given ciphertext is correct, as opposed to the paradigm followed in [Sta96,BG96,YY98,ASW00], which intertwines the process of encrypting and proving, so that the entire transcript of the proof must be retained by the verifier in lieu of a (short) ciphertext. Additionally, the combined encrypting/proving paradigm makes it much harder to incorporate any type of verifiable decryption protocol.

Our verifiable decryption protocols are the first practical schemes of their kind.

Unlike, e.g., the schemes in [Sta96,YY98], we do not require that all users of the system work with the same algebraic group — in our system, there are no "double decker" discrete logarithms, and the encryption keys may be used with any group or groups, provided certain reasonable size restrictions are met.

To give the reader a rough idea of the complexity of of our protocols, consider a setting in which the discrete logarithms being encrypted are with respect to an element of order ρ, where ρ is, say, around $\ell' \approx 160$ bits. For such a ρ, it suffices to work with a modulus n of around $\ell \approx 1024$ bits for the Paillier encryption scheme. Counting just squarings, which are all that matter asymptotically, and ignoring lower order terms, the encryption algorithm takes 3ℓ squarings mod n^2, and the decryption algorithm takes 5ℓ squarings mod n^2. For the verifiable encryption protocol, the prover performs 2ℓ squarings mod n, 3ℓ squarings mod n^2, and ℓ' squarings in the underlying group; the verifier performs 3ℓ squarings mod n^2, ℓ squarings mod n, and ℓ' squarings in the group. The verifiable decryption protocols are several times slower than this. For representations with respect to several bases, the complexity of the encryption and decryption algorithms, and the corresponding proof protocols, grows linearly in the number of bases, as one would expect.

Our decryption procedure can be implemented as a *threshold decryption protocol*. This allows one to minimize the trust placed in the decryptor, and in some applications this may be a preferable alternative to verifiable decryption.

2 Preliminaries

2.1 Notation

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

Let a, b, and c be integers, with $b > 0$. Then $c = a \bmod b$ denotes $a - \lfloor a/b \rfloor b$ (and we have $0 \leq c < b$), and $c = a \operatorname{rem} b$ denotes $a - \lceil a/b \rfloor b$ (and we have $-b/2 \leq c < b/2$).

2.2 Σ-protocols

A Σ-protocol [Cra96] is a protocol between a prover and a verifier, where y is their common input and x is the prover's additional input, which consists of 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 random "challenge" message c, and in the third move the prover sends the verifier a "response" message s.

Such a protocol is *special honest verifier zero knowledge* if there exists a simulator that, on input (y, c), outputs (t, s) such that the distribution of the triple (t, c, s) is is indistinguishable from that of an actual conversation, conditioned on the event that the verifier's challenge is c. This property implies (ordinary) honest verifier zero knowledge, and also allows the protocol to be easily and efficiently transformed into one that satisfies much stronger notions of zero knowledge.

Such a protocol is said to satisfy the *special soundness condition with respect to a property* θ if it is computationally infeasible to find two valid conversations (t, c, s) and (t, c', s'), with $c \neq c'$, unless the input y satisfies θ. Via standard rewinding arguments, this notion of soundness implies the more general notion of computational soundness.

We use notation introduced by Camenisch and Stadler [CS97] for the various proofs of relations among 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* c *such that* $y = g^a h^b$*,* $\mathfrak{y} = \mathfrak{g}^a \mathfrak{h}^c$*, and* $u \leq a \leq v$ holds," 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.

2.3 Secure Public-Key Encryption

We need the notion of a public-key encryption scheme secure against chosen ciphertext attacks [RS92] that supports *labels* [Sho01]. 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. The definition of security for such a scheme is the same as the one without labels except that now the adversary is given a target ciphertext ψ^* and a target label L^* and is then allowed to submit any queries (ψ, L) subject to $(\psi, L) \neq (\psi^*, L^*)$.

3 The Encryption Scheme

3.1 Background

Let p, q, p', and 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 [Pai99] is that given only n, it is hard to distinguish random elements of $\mathbb{Z}_{n^2}^*$ from random elements of $\mathbf{P}$.

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 $0 \le a < n$. Observe that $\mathbf{P} = \mathbf{G}_{n'}\mathbf{G}_2\mathbf{T}$.

3.2 The Scheme

Let ℓ be a system parameter. The scheme makes use of a keyed hash scheme $\mathcal{H}$ that uses a key hk, chosen at random from some key space; 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 $(u, e, L) \ne (u', e', L')$ 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 $a \in [n^2]$, 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.

Key Generation. Select two random ℓ-bit Sophie Germain primes p' and q', with $p' \ne q'$, and compute $p := (2p'+1)$, $q := (2q'+1)$, $n := pq$, and $n' := p'q'$. 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 $\mathcal{H}$. 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 computes the ciphertext (u, e, v) as follows.

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

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 [CS01], 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 [CS01], 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.

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 [ADR02,Sho01] 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.*

We refer to the full version of the paper [CS02] for the proof of Theorem 1.

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 [SG98], which requires a random oracle security argument, or along the lines in [CG99], which does not require that argument, but for which the decryption protocol is less efficient.

4 Verifiable Encryption

4.1 Definitions

At a high level, a verifiable encryption scheme for a binary relation $\mathcal{R}$ is a protocol that allows a prover to convince a verifier that a ciphertext ψ is an encryption under a given public key PK and label L of a value w such that $(\delta, w) \in \mathcal{R}$ for a given δ. Here, the common input to the prover and the verifier consists of PK, L, ψ, and δ, and the prover has as additional input the "witness" w and the random bit string that was used to create ψ. We shall require that the protocol is a Σ-protocol that is special honest verifier zero knowledge, and that satisfies the special soundness condition for the property described above.

We refer the reader to the full version of the paper [CS02] for a more detailed definition, but we briefly mention a few subtle points that apply here, as well as in other definitions in this paper: (1) our notion of security is computational, even to the extent that the we quantify "computationally" (rather than universally) over the common input to the prover and verifier in the definition of honest verifier zero knowledge and special soundness; (2) we assume that the public key/secret key pair for the encryption scheme is generated by a trusted party using the appropriate key generation algorithm; (3) in defining soundness, we only require that the proof convinces the verifier that plaintext can be easily transformed into a witness using some scheme-specific *reconstruction routine*.

4.2 The protocol

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 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 $\mathfrak{l}$ is an additional system parameter.

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 and k' be further system parameters, where 2^{-k} and $2^{-k'}$ are negligible ($\{0,1\}^k$ is the "challenge space" of the verifier and k' controls the quality of the zero-knowledge property). We require that $2^k < \min\{p', q', \mathfrak{p}', \mathfrak{q}', \rho\}$ holds. Finally, we require that $\rho < n2^{-k-k'-3}$ holds, i.e., that $\log_\gamma \delta$ "comfortably fits into an encryption."

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

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 $r' \in_R [-n2^{k+k'-2}, n2^{k+k'-2}]$, $s' \in_R [-\mathfrak{n}2^{k+k'-2}, \mathfrak{n}2^{k+k'-2}]$, and $m' \in_R [-\rho 2^{k+k'}, \rho 2^{k+k'}]$.
   The prover computes $u' := g^{r'}$, $e' := y_1^{r'} h^{m'}$, $v' := (y_2 y_3^{\mathcal{H}_{\mathsf{hk}}(u,e,L)})^{r'}$, $\delta' := \gamma^{m'}$, 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 $u'^2 = u^{2c} g^{2\tilde{r}}$, $e'^2 = e^{2c} y_1^{2\tilde{r}} h^{2\tilde{m}}$, $v'^2 = v^{2c} (y_2 y_3^{\mathcal{H}_{\mathsf{hk}}(u,e,L)})^{2\tilde{r}}$, $\delta' = \delta^c \gamma^{\tilde{m}}$, $\mathfrak{k}' = \mathfrak{k}^c \mathfrak{g}^{\tilde{m}} \mathfrak{h}^{\tilde{s}}$, and $-n/4 < \tilde{m} < n/4$ 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 [CS97] we denote the sub-protocol of step 2 as

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

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

We refer to the full version of the paper [CS02] for the proof of Theorem 2.

4.3 Extensions

It is straightforward to extend the above verifiable encryption scheme to a verifiable encryption scheme that encrypts 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 logarithms are not equal). There are well-known, efficient, special honest-verifier zero-knowledge proof systems for proving that two discrete logarithms are equal (see [CP93]), 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 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 $PK\{(\alpha, \beta) :\ C = h^{\alpha} \left(\frac{1}{z}\right)^{\beta} \ \wedge \ 1 = g^{\alpha} \left(\frac{1}{y}\right)^{\beta}\}$ with the verifier.
3. The verifier accepts if it accepts in step 2, and if $C \neq 1$; otherwise, the verifier rejects.

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

We refer to the full version of this paper [CS02] for the proof of Theorem 3.

Let us discuss related work. Independently of our work, Bresson and Stern [BS02] provide a protocol to prove that two discrete logarithms 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. We finally note that the (efficient) protocol proposed by Michels and Stadler [MS98] to prove whether or not two discrete logarithms are equal is *not* zero-knowledge because it reveals the value h^x.

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.

6.1 Definition of Verifiable Decryption

At a high level, a verifiable decryption scheme for a binary relation $\mathcal{R}$ is a protocol that allows a prover to convince a verifier *whether or not* a ciphertext ψ is an encryption under a given public key PK and label L of a value w such that $(\delta, w) \in \mathcal{R}$ for a given δ. Here, the common input to the prover and the verifier consists of PK, L, ψ, and δ, and the prover has as additional input the "witness" w and the *secret key* SK *corresponding to* PK. We shall require that the protocol is a Σ-protocol that is special honest verifier zero knowledge, and that satisfies the special soundness condition for the property described above.

We refer the reader to the full version of the paper [CS02] for a more detailed definition, but that as for verifiable encryption, the statement being proved (or disproved) is whether the plaintext reconstructs to a witness using the specified reconstruction routine. We also point out that since the prover tells whether or not the given condition holds, the zero-knowledge simulator must be given this one bit of information as well.

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 is the identity function.

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

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

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 [CDS94], 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 §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 (1), 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 $a \neq 1 \;\Leftrightarrow\; (a^n \in \mathbf{G}_{n'} \wedge a^n \neq 1) \vee (a \in \mathbf{G}_n \wedge a \neq 1)$. Thus, to prove that (at least) one of the equations (1) does not hold, we prove that either

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

or

$$\Big(\frac{u^{2(x_2+\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3)}}{v^2}\Big)^n = 1 \qquad \text{and} \qquad \frac{u^{2(x_2+\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3)}}{v^2} \neq 1 \tag{3}$$

or

$$\Big(\frac{(e/u^{x_1})^2}{h^{2m}}\Big)^n = (e/u^{x_1})^{2n} \neq 1 \tag{4}$$

or

$$\Big(\frac{(e/u^{x_1})^2}{h^{2m}}\Big)^n = 1 \qquad \text{and} \qquad \frac{(e/u^{x_1})^2}{h^{2m}} \neq 1 \tag{5}$$

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 (2-4) are false but the Statement (5) 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 $\big((\mathsf{hk}, n, g, y_1, y_2, y_3), (\mathfrak{n}, \mathfrak{g}, \mathfrak{h}), (u, e, v), m, L\big)$ 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 §4.2. (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.

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

where $r_1, \ldots, r_4, s_1, \ldots, s_4$ are temporary secrets (i.e., $r_1 = a_1(x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)x_3)$, $s_1 = b_1(x_2+\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3)$, $r_2 = a_2(x_2+\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3)$, $s_2 = b_2(x_2+\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3)$, $r_3 = x_1a_3$, $s_3 = x_1b_3$, $r_4 = x_1a_4$, $s_4 = x_1b_4$, (all computed in $\mathbb{Z}$)). (To derive the actual protocol one has to apply the techniques by Cramer et al. [CDS94] 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 §4.2 (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).*

We refer to the full version of this paper [CS02] for the proof of Theorem 4.

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 relation $\mathcal{R}$ and 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

$$u^{2(x_2+\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3)}/v^2 = 1 \ , \quad \left(\frac{e}{u^{x_1}}\right)^{2n} = 1 \ , \text{ and } \quad \delta = \gamma^{(\log_{h^2}(e/u^{x_1})^2 \operatorname{rem} n)} \qquad (6)$$

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 (6) can be handled as in §4.2. 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 §4.2. 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 [Bou00] 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 [RS86] provide an efficient algorithm to find these squares. 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 obtain a protocol for verifiable decryption of a discrete logarithm from the protocol presented in §4.2. For lack of space, we refer the reader to the full version of this paper [CS02] for the details. We also note that it is straightforward to adapt this protocol to verifiably decrypt representations with respect to several bases. One can also "mix and match," proving whether or not ψ decrypts to a representation, one or more components of which match specified values.

References

[ACJT00] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik, *A practical and provably secure coalition-resistant group signature scheme*, Advances in Cryptology — CRYPTO 2000, LNCS, vol. 1880, Springer Verlag, 2000, pp. 255–270.

[ADR02] J. H. An, Y. Dodis, and T. Rabin, *On the security of joint signature and encryption*, Advances in Cryptology: EUROCRYPT 2002, LNCS, vol. 2332, Springer, 2002, pp. 83–107.

[ASW97] N. Asokan, M. Schunter, and M. Waidner, *Optimistic protocols for fair exchange*, 4th ACM Conference on Computer and Communication Security, 1997, pp. 6–17.

[ASW00] N. Asokan, V. Shoup, and M. Waidner, *Optimistic fair exchange of digital signatures*, IEEE Journal on Selected Areas in Communications **18** (2000), no. 4, 591–610.

[BDM98] F. Bao, R. Deng, and W. Mao, *Efficient and practical fair exchange protocols with off-line TTP*, IEEE Symposium on Security and Privacy, IEEE Computer Society Press, 1998, pp. 77–85.

[BG96] M. Bellare and S. Goldwasser, *Encapsulated key escrow*, Preprint, 1996.

[Bou00] F. Boudot, *Efficient proofs that a committed number lies in an interval*, Advances in Cryptology — EUROCRYPT 2000, LNCS, vol. 1807, Springer Verlag, 2000, pp. 431–444.

[BP90] H. Bürk and A. Pfitzmann, *Digital payment systems enabling security and unobservability*, Computer & Security **9** (1990), no. 8, 715–721.

[BS02] E. Bresson and J. Stern, *Proofs of knowledge for non-monotone discrete-log formulae and applications*, Information Security (ISC 2002), LNCS, vol. 2433, Springer Verlag, 2002, pp. 272–288.

[CD00] J. Camenisch and I. Damgård, *Verifiable encryption, group encryption, and their applications to group signatures and signature sharing schemes*, Advances in Cryptology — ASIACRYPT 2000, LNCS, vol. 1976, Springer Verlag, 2000, pp. 331–345.

[CDS94] R. Cramer, I. Damgård, and B. Schoenmakers, *Proofs of partial knowledge and simplified design of witness hiding protocols*, Advances in Cryptology — CRYPTO '94, LNCS, vol. 839, Springer Verlag, 1994, pp. 174–187.

[CF01] R. Canetti and M. Fischlin, *Universally composable commitments*, Advances in Cryptology — CRYPTO 2001, LNCS, vol. 2139, Springer Verlag, 2001, pp. 19–40.

[CG98] D. Catalano and R. Gennaro, *New efficient and secure protocols for verifiable signature sharing and other applications*, Advances in Cryptology — CRYPTO '98 (Berlin), LNCS, vol. 1642, Springer Verlag, 1998, pp. 105–120.

[CG99] R. Canetti and S. Goldwasser, *An efficient threshold public key cryptosystem secure against adaptive chosen ciphertext attack*, Advances in Cryptology — EUROCRYPT '99, LNCS, vol. 1592, Springer Verlag, 1999, pp. 90–106.

[Cha85] D. Chaum, *Security without identification: Transaction systems to make big brother obsolete*, Communications of the ACM **28** (1985), no. 10, 1030–1044.

[Cha94] D. Chaum, *Designated confirmer signatures*, Advances in Cryptology — EUROCRYPT '94, LNCS, vol. 950, Springer Verlag Berlin, 1994, pp. 86–91.

[CL01] J. Camenisch and A. Lysyanskaya, *Efficient non-transferable anonymous multi-show credential system with optional anonymity revocation*, Advances in Cryptology — EUROCRYPT 2001, LNCS, vol. 2045, Springer Verlag, 2001, pp. 93–118.

[CM99a] J. Camenisch and M. Michels, *Proving in zero-knowledge that a number n is the product of two safe primes*, Advances in Cryptology — EUROCRYPT '99, LNCS, vol. 1592, Springer Verlag, 1999, pp. 107–122.

[CM99b] J. Camenisch and M. Michels, *Separability and efficiency for generic group signature schemes*, Advances in Cryptology — CRYPTO '99, LNCS, vol. 1666, Springer Verlag, 1999, pp. 413–430.

[CM00] J. Camenisch and M. Michels, *Confirmer signature schemes secure against adaptive adversaries*, Advances in Cryptology — EUROCRYPT 2000, LNCS, vol. 1807, Springer Verlag, 2000, pp. 243–258.

[CP93] D. Chaum and T. P. Pedersen, *Wallet databases with observers*, Advances in Cryptology — CRYPTO '92, LNCS, vol. 740, Springer-Verlag, 1993, pp. 89–105.

[Cra96] R. Cramer, *Modular design of secure yet practical cryptographic protocols*, Ph.D. thesis, University of Amsterdam, 1996.

[CS97] J. Camenisch and M. Stadler, *Efficient group signature schemes for large groups*, Advances in Cryptology — CRYPTO '97, LNCS, vol. 1296, Springer Verlag, 1997, pp. 410–424.

[CS01] 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.

[CS02] J. Camenisch and V. Shoup, *Practical verifiable encryption and decryption of discrete logarithms*, `http://eprint.iacr.org/2002/161`, 2002.

[CVH02] J. Camenisch and E. Van Herreweghen, *Design and implementation of the* idemix *anonymous credential system*, Proc. 9th ACM Conference on Computer and Communications Security, 2002.

[Dam00] I. Damgård, *Efficient concurrent zero-knowledge in the auxiliary string model*, Advances in Cryptology — EUROCRYPT 2000, LNCS, vol. 1807, Springer Verlag, 2000, pp. 431–444.

[DF02] I. Damgård and E. Fujisaki, *An integer commitment scheme based on groups with hidden order*, Advances in Cryptology — ASIACRYPT 2002', LNCS, vol. 2501, 2002.

[FO97] E. Fujisaki and T. Okamoto, *Statistical zero knowledge protocols to prove modular polynomial relations*, Advances in Cryptology — CRYPTO '97, LNCS, vol. 1294, Springer Verlag, 1997, pp. 16–30.

[FR95] M. Franklin and M. Reiter, *Verifiable signature sharing*, Advances in Cryptology — EUROCRYPT '95, LNCS, vol. 921, Springer Verlag, 1995, pp. 50–63.

[FS87] A. Fiat and A. Shamir, *How to prove yourself: Practical solution to identification and signature problems*, Advances in Cryptology — CRYPTO '86, LNCS, vol. 263, Springer Verlag, 1987, pp. 186–194.

[KP97] J. Kilian and E. Petrank, *Identity escrow*, Theory of Cryptography Library, Record Nr. 97-11, `http://theory.lcs.mit.edu/~tcryptol`, August 1997.

[KP98] J. Kilian and E. Petrank, *Identity escrow*, Advances in Cryptology — CRYPTO '98 (Berlin), LNCS, vol. 1642, Springer Verlag, 1998, pp. 169–185.

[Mic] S. Micali, *Efficient certificate revocation and certified e-mail with transparent post offices*, Presentation at the 1997 RSA Security Conference.

[MS98] M. Michels and M. Stadler, *Generic constructions for secure and efficient confirmer signature schemes*, Advances in Cryptology — EUROCRYPT '98, LNCS, vol. 1403, Springer Verlag, 1998, pp. 406–421.

[Pai99] P. Paillier, *Public-key cryptosystems based on composite residuosity classes*, Advances in Cryptology — EUROCRYPT '99, LNCS, vol. 1592, Springer Verlag, 1999, pp. 223–239.

[Ped92] T. P. Pedersen, *Non-interactive and information-theoretic secure verifiable secret sharing*, Advances in Cryptology – CRYPTO '91, LNCS, vol. 576, Springer Verlag, 1992, pp. 129–140.

[PS00] G. Poupard and J. Stern, *Fair encryption of RSA keys*, Advances in Cryptology: EUROCRYPT 2000, LNCS, vol. 1087, Springer Verlag, 2000, pp. 173–190.

[RS86] M. O. Rabin and J. O. Shallit, *Randomized algorithms in number theory*, Communications on Pure and Applied Mathematics **39** (1986), 239–256.

[RS92] C. Rackoff and D. R. Simon, *Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack*, Advances in Cryptology: CRYPTO '91, LNCS, vol. 576, Springer, 1992, pp. 433–444.

[SG98] V. Shoup and R. Gennaro, *Securing threshold cryptosystems against chosen ciphertext attack*, Advances in Cryptology: EUROCRYPT '98, LNCS, vol. 1403, Springer, 1998.

[Sho01] V. Shoup, *A proposal for an ISO standard for public key encryption*, `http://eprint.iacr.org/2001/112`, 2001.

[Sta96] M. Stadler, *Publicly verifiable secret sharing*, Advances in Cryptology — EUROCRYPT '96, LNCS, vol. 1070, Springer Verlag, 1996, pp. 191–199.

[YY98] A. Young and M. Young, *Auto-recoverable auto-certifiable cryptosystems.*, Advances in Cryptology — EUROCRYPT '98, LNCS, vol. 1403, Springer Verlag, 1998, pp. 17–31.