Practical Verifiable Encryption and Decryption of Discrete Logarithms*

Jan Camenisch† Victor Shoup‡

August 22, 2003

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.
Keywords. Verifiable encryption, verifiable decryption, adaptive chosen ciphertext security, public key encryption.

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 decryptor is the prover 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 key 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 the protocols should work with. In this paper, we consider

*An extended abstract of this paper appears in the proceedings of *Crypto 2003*.

†IBM Research, Zurich Research Laboratory, CH–8803 Rüschlikon, `jca@zurich.ibm.com`

‡Courant Institute, New York University, `shoup@cs.nyu.edu`

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]); combining these protocols with our scheme for verifiable encryption of a representation (i.e., an opening of a commitment), we immediately get corresponding protocols for proving such properties about encrypted values.

Our contribution. The contribution of this paper is to present and analyze a practical public-key encryption scheme, together with a suite of practical 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 scheme of Cramer and Shoup [CS02] based on Paillier's decision composite residuosity assumption [Pai99], suitably modified so as to support our proof protocols. The proof protocols are all of the usual, three move "Σ-protocol" type [CDS94], 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 scheme for verifiable encryption of discrete logarithms is the first one that provides chosen ciphertext security and avoids inefficient cut-and-choose proofs. Our scheme for verifiable decryption of discrete logarithms is the first practical protocol of its kind. Our system is very flexible, in that a single public key for the encryption scheme can be used with many different groups; that is, users can choose their own (arbitrary and varied) groups for discrete logarithms, subject only to some (reasonable) size constraints. As pointed out in [KP98, CM99b, ASW00] such *separability* in system design is highly desirable in practice. 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, either yield more efficient solutions or add security against chosen ciphertext attacks, which is often crucial.

1.1.1 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 an 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 protocols for proving that a committed value is of such a form [FO97, CM99a, DF02, PS00, Bou00].

1.1.2 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 to 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.

1.1.3 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 the third party that the ciphertexts encrypt openings of the commitments to the shares. As 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].

1.1.4 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 act 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.

1.1.5 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 can not 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.

1.1.6 Group Signatures and Anonymous Credentials

In a group signature scheme (see [ACJT00, KP98, CD00]), when a user joined 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 [KP98]).

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 (this is the separability issue). 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* [CL01, 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 those 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.

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.

Our protocols are based on a number of techniques. The key ingredients that make our verifiable encryption protocol possible are:

- Fujisaki and Okamoto's method for proving relations on committed values [FO97] (with some refinements, as in [CS00, DF02]),
- the related interval proofs [CM98, CFT98],
- Paillier encryption [Pai99], and
- Cramer and Shoup's universal hash proof encryption technique [CS02].

The additional ingredients needed to make our verifiable decryption protocols work are:

- Cramer, Damgård, and Schoenmakers' proofs of partial knowledge [CDS94],
- Boudot's exact interval proofs [Bou00], and
- new protocols for proving the inequality of discrete logarithms.

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 about 5 to 6 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.

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 negligible function, i.e., a function f such that $f(\lambda) < 1/p(\lambda)$ holds for all polynomials $p(\lambda)$ and all sufficiently large λ.

Let (P, V) be a pair of interactive Turing machines. By $V(x)_{P(y)}$ we denote the output of V upon interacting with P, where V's input is x and P's input is y.

We use notation introduced by Camenisch and Stadler [CS97] 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 and 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 [Dam00, DNS98, CGGM00] or witness-hiding protocols [CDS94]. In particular, the construction due to Damgård achieves (concurrent) zero-knowledge virtually for free [Dam00].

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 $(\mathsf{PK}, \mathsf{SK}) \in \mathcal{G}(1^\lambda)$, 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 n-th 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 $\mathbf{P}$.

To be completely formal, one should 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 safe primes in originally formulating the DCR assumption. As will become evident, we need to restrict ourselves to safe 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 safe 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 $0 \le a < n$. Observe that $\mathbf{P} = \mathbf{G}_{n'}\mathbf{G}_2\mathbf{T}$.

3.2 The Scheme

For a security parameter $\lambda \ge 0$, $\ell = \ell(\lambda)$ 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 $g' \in_R \mathbb{Z}_{n^2}^*$, and 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 [CS02], 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 [CS02], 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 [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. We also mention that in independent work, Gennaro and Lindell [GL03] devise a similar (but not quite identical) scheme, but for completely different purposes: their goal is to construct efficient password-based key exchange protocols.

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 (c.f. §2.4), 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, as 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 $[nn']$.

Because 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 $\bar{u}$. As 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 $v \neq v^*$, as $(u, e, v, L) \neq (u^*, e^*, v^*, L)$. Because 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: $(u, e, L) \neq (u^*, e^*, L^*)$. As 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 $\bar{u}$. Because 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 decryptor when used as a trusted third party. 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 The Strong RSA and Factoring Assumptions

This strong RSA assumption is the following: given a composite modulus n and a random element $g \in \mathbb{Z}_n^*$, it is hard to compute $h \in \mathbb{Z}_n^*$ and integer $e > 1$ such that $h^e = g$. To be complete, one needs to specify more precisely the distribution from which n is drawn. As in §3, we shall specify that n is of the form pq, where $p = 2p' + 1$, $q = 2q' + 1$, and p' and q' are uniformly distributed over all ℓ-bit numbers such that p, q, p', q' are prime and $p' \neq q'$. We also set $n' = p'q'$. As usual, $\ell = \ell(\lambda)$, where λ is a security parameter.

We will make use of both the strong RSA assumption, as well as the assumption that factoring integers n as above is hard. Of course, the strong RSA assumption implies that factoring is hard.

We will make use of these two assumptions as follows. First, we shall make extensive use of the well known fact that if factoring is hard, then it is hard to compute a non-zero multiple of n'. We shall also make use of the fact that if factoring is hard, then it is hard to compute a non-zero multiple of either p' or q'. To see this, suppose that m is a non-zero multiple of p' or q'. If m is a multiple of n', then the above mentioned result applies. Otherwise, with overwhelming probability, for random $z \in \mathbb{Z}_n^*$, $\gcd(z^{2m} - 1, n)$ will be either p or q, as the reader may easily check using the Chinese Remainder Theorem. Thus, assuming factoring is hard, we may assume that it is hard to compute a non-zero integer m such that $\gcd(m, n') \neq 1$.

We also shall use the following facts:

Theorem 2. *Under the assumption that factoring is hard, given a modulus n (distributed as above), along with random elements $g, h \in (\mathbb{Z}_n^*)^2$, it is hard to compute integers a, b, such that*

$$1 = g^a h^b \quad \textit{and} \quad (a \neq 0 \textit{ or } b \neq 0). \tag{13}$$

Proof. Suppose there is an algorithm A that takes as input n, g, h as above, and outputs a, b satisfying (13) with non-negligible probability. We can use use A to factor a given n, as follows: generate $g \in (\mathbb{Z}_n^*)^2$ at random — with overwhelming probability, g has order n'; choose $r \in [1, n^2]$ at random, and set $h = g^r$ — the distribution of h is statistically close to the uniform distribution on $(\mathbb{Z}_n^*)^2$; feed n, g, h to A, obtaining a, b. With non-negligible probability, we have

$$g \text{ has order } n' \ , \qquad 1 = g^{a+rb} \ , \quad \text{and} \quad (a \neq 0 \text{ or } b \neq 0) \ . \tag{14}$$

Claim: with non-negligible probability, not only does (14) hold, but also $a + rb \neq 0$. To prove this claim, let us condition on fixed values of n, g, h, a, b, and coins of A, such that A outputs a, b on inputs n, g, h, and such that the conditions in (14) are satisfied. Let us write $r = r_2 n' + r_1$, where $0 \leq r_1 < n'$. In this conditional probability space, the value r_1 is also fixed, but the distribution of r_2 is statistically close to the uniform distribution on $[4n]$. We can write the equation $a + rb = 0$ as $a + r_2 n' b + r_1 b = 0$, and in this equation all terms are fixed except for r_2. We may as well assume that $b \neq 0$, as otherwise, $a \neq 0$ and the equation never holds. There is at most one solution in r_2 to the equation (as the coefficient $n'b$ is non-zero), and so it holds with only negligible probability. That proves the claim.

The identity g^{a+rb} implies that $a + rb$ is a multiple of n', and if $a + rb \neq 0$, we have a non-zero multiple of n'. □

Theorem 3. *Under the strong RSA assumption, given a modulus n (distributed as above), along with random elements $g, h \in (\mathbb{Z}_n^*)^2$, it is hard to compute $w \in \mathbb{Z}_n^*$ and integers a, b, c such that*

$$w^c = g^a h^b \quad \textit{and} \quad (c \nmid a \textit{ or } c \nmid b). \tag{15}$$

Proof. Suppose we have an algorithm A that given n, g, h as above, computes w, a, b, c satisfying (15) with non-negligible probability.

Case 1. Let us first consider the case where $c = 0$ with non-negligible probability. Then the condition that $c \nmid a$ or $c \nmid b$ simply means that $a \neq 0$ or $b \neq 0$, and the result is implied by Theorem 2.

Case 2. Let us next consider the remaining, and more interesting, case where $c \neq 0$ with non-negligible probability. We may as well assume that $\gcd(c, n') = 1$, since as was discussed at the beginning of the proof, under the assumption that factoring is hard, it is difficult to compute non-zero c such that $\gcd(c, n') \neq 1$. We now show how we can use A to either factor a given n or find a non-trivial root of a given g, thus contradicting the strong RSA assumption (since a random element of $\mathbb{Z}_n^*$ is a square with probability 1/4).

Given n and g, we proceed as follows. First, note that with overwhelming probability, g has order n'. Let us compute $h = g^r$, for r randomly chosen from $[1, n^2]$, so that the distribution of h is statistically close to the uniform distribution on $(\mathbb{Z}_n^*)^2$. Now we feed n, g, h to A, obtaining w, a, b, c. With non-negligible probability, we have

$$g \text{ has order } n', \quad w^c = g^{a+rb}, \quad c \neq 0, \quad \gcd(c, n') = 1, \quad \text{and} \quad c \nmid a \ . \tag{16}$$

Claim: with non-negligible probability, not only does (16) hold, but also $c \nmid (a + rb)$. To prove this claim, let now condition on fixed values of n, g, h, w, a, b, c, and coins of A, such that A outputs w, a, b, c on inputs n, g, h, and such that the conditions in (16) are satisfied. Let us write $r = r_2 n' + r_1$, where $0 \leq r_1 < n'$. In this conditional probability space, the value r_1 is also fixed, but the distribution of r_2 is statistically close to the uniform distribution on $[4n]$.

Now, consider the congruence

$$a + rb \equiv 0 \pmod{c} \ .$$

This congruence holds if and only if

$$a + r_1 b + r_2 n' b \equiv 0 \pmod{c} \ .$$

Now, in the conditional probability space, all terms in the above congruence are fixed, except for r_2. Let us bound from above the probability that this congruence holds. We may as well assume that $c \nmid b$, because if $c \mid b$, then $c \nmid a$, and the congruence will never hold. As $\gcd(c, n') = 1$, it follows that the solutions r_2 to the above congruence are uniquely determined modulo c/d', where $d' = \gcd(c, b)$. Since $c \nmid b$, it follows that d' is a proper divisor of c, and hence $c/d' \geq 2$. Because the distribution of r_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. This proves the claim.

It is left to show that if $c \nmid (a + rb)$, then we can either factor n, or just compute a non-trivial root of g. Let $d = \gcd(c, a + rb)$. Since we are assuming that $c \nmid (a + rb)$, it follows that $c/d \geq 2$. There are integers α and β such that $d = \alpha c + \beta(a + rb)$, and using the identity $w^c = g^a h^b$, we have

$$g^d = (w^\beta g^\alpha)^c,$$

and so $g = \zeta (w^\beta g^\alpha)^{c/d}$ for some $\zeta \mathbb{Z}_n^*$ with $\zeta^d = 1$. Thus, the order of ζ divides d, and of course, since $2n'$ is the exponent of $\mathbb{Z}_n^*$, it follows that the order of ζ divides $\gcd(d, 2n')$. Now, since $d \mid c$ and $\gcd(c, n') = 1$, we have $\gcd(d, n') = 1$, from which it follows that ζ has order dividing 2.

So either $\zeta = \pm 1$ or $\gcd(\zeta - 1, n)$ splits n. In the latter case we have factored n. In the former case we can compute such a root of g as follows. If c/d is even, then $(w^\beta g^\alpha)^{c/d} \in (\mathbb{Z}_n^*)^2$ and so (because $g \in (\mathbb{Z}_n^*)^2$), we must have $\zeta = 1$ (as $-1 \notin (\mathbb{Z}_n^*)^2$). If c/d is odd then $g = (\zeta w^\beta g^\alpha)^{c/d}$. In either case, we have computed a (c/d)th root of g.

□

Discussion. The strong RSA assumption was introduced independently in [BP97] and [FO97]. Since then, it has been found to be useful in the analysis of many cryptographic schemes (e.g., [CM98, GHR99, CS00, ACJT00, CL01]). We do not claim that Theorem 3 is new: it has appeared implicitly and in more restricted form in previous papers: the essential idea in the proof of Theorem 3 already appears in [CS00], although that paper deals with a more restricted, and somewhat simpler, setting; also, the paper [DF02] implicitly contains a proof of a statement that is very similar to that of Theorem 3. The paper [FO97] also makes some similar claims (implicitly), but some of their proofs are flawed. Theorem 3 is actually a bit more general than we actually need for our paper, but as it is actually a quite useful theorem in several contexts, we prefer to state it in a very general form.

5 Verifiable Encryption

Loosely speaking, verifiable encryption for a relation $\mathcal{R}$ is a protocol that allows a prover to convince a verifier that a given 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. [ASW98, ASW00] present a protocol for verifiable encryption for the case where w is a homomorphic pre-image of δ and Camenisch and Damgård [CD00] 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.

5.1 Definition of Verifiable Encryption

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 by $\mathcal{E}$ 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 $\Psi[\mathcal{R}, W, \delta]$, 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 (together with the correctness property for the encryption scheme, this ensures that an encryption of a witness decrypts and reconstructs to the same witness).

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 $(\psi, \mathit{coins}) \in \mathcal{E}'(\mathsf{PK}, w, L)$,*

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

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

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

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

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

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. While 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 would require only slight modification to remain secure in this setting.

5.2 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 described 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 $\mathfrak{l} = \mathfrak{l}(\lambda)$.

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 $\mathfrak{n} := n$. 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 $2^k < \min\{p', q', \mathfrak{p}', \mathfrak{q}', \rho\}$ 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

$$u' := g^{2r'}, \ \ e' := y_1^{2r'} h^{2m'}, \ \ v' := (y_2 y_3^{\mathcal{H}_{\mathsf{hk}}(u,e,L)})^{2r'}, \ \ \delta' := \gamma^{m'}, \text{ and } \mathfrak{k}' := \mathfrak{g}^{m'} \mathfrak{h}^{s'}.$$

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

$$\begin{aligned} &u' = u^{2c} g^{2\tilde{r}}, && e' = e^{2c} y_1^{2\tilde{r}} h^{2\tilde{m}}, && v' = v^{2c} (y_2 y_3^{\mathcal{H}_{\mathsf{hk}}(u,e,L)})^{2\tilde{r}}, \\ &\delta' = \delta^c \gamma^{\tilde{m}}, && \mathfrak{k}' = \mathfrak{k}^c \mathfrak{g}^{\tilde{m}} \mathfrak{h}^{\tilde{s}}, \quad \text{and} && -n/4 < \tilde{m} < n/4 \end{aligned}$$

hold. If any of them does not hold, the verifier stops and outputs 0.

3. If $v = \text{abs}(v)$ the verifier outputs 1; otherwise she outputs 0.

Using notation from [CS97] we denote the sub-protocol of Step 2 as

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

Proof of Security. We prove the following theorem about the above system. Given this theorem, one can apply the standard constructions (e.g., [Dam00]) to turn the sub-protocol used in Step 2 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 4. *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:

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

Now we use the strong RSA assumption. By Theorem 3, since we have computed $\mathfrak{k}, \Delta m, \Delta s$, and Δc such that $\mathfrak{k}^{\Delta c} = \mathfrak{g}^{\Delta m} \mathfrak{h}^{\Delta s}$, we may assume that $\Delta c \mid \Delta m$ and $\Delta c \mid \Delta s$. Also, 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.,

$$u = w_1 g^{\Delta r \hat{c}} \tag{17}$$

for some w_1 of order 2. Similarly, we get

$$e = w_2 y_1^{\Delta r \hat{c}} h^{\Delta m/\Delta c} \tag{18}$$

$$v = w_3 (y_2 y_3^{\mathcal{H}_{\mathsf{hk}}(u,e,L)})^{\Delta r \hat{c}} \tag{19}$$

$$\delta = \gamma^{\Delta m/\Delta c} \tag{20}$$

for some w_2 and w_3 of order 2. It is not hard to see that from $v = \mathrm{abs}(v)$ and from Eqns. (17)-(19) 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's 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 $\delta = \gamma^{\Delta m/\Delta c} = \gamma^{\check{m}}$.

$\square$

5.3 Extensions

Our encryption scheme can be extended as follows to encrypt l messages at once. The idea is to use several y_1's to compute several e's. That is, the secret key becomes $(\mathsf{hk}, x_1^{(1)}, \ldots, x_1^{(l)}, x_2, x_3)$ with $x_1^{(1)}, \ldots, x_1^{(l)}, x_2, x_3 \in_R [n^2/4]$, and the public key becomes $(\mathsf{hk}, n, g, y_1^{(1)}, \ldots, y_1^{(l)}, y_2, y_3)$ with $y_1^{(i)} := g^{x_1^{(i)}}$. To encrypt a messages $m^{(i)} \in [n]$ with label $L \in \{0,1\}^*$ under a public key as above, choose a random $r \in_R [n/4]$ and compute

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

To decrypt a ciphertext $(u, e^{(1)}, \ldots, e^{(l)}, v)$ 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 $\hat{m}^{(i)} := (e^{(i)}/u^{x_1^{(i)}})^{2t}$. If all $\hat{m}^{(i)}$'s are of the form $h^{m^{(i)}}$ for some $m^{(i)} \in [n]$, then output the $m^{(i)}$'s; otherwise, output reject. It is easy to prove this encryption scheme secure.

It is now straightforward to extend our verifiable encryption protocol to the above encryption scheme to obtain 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.

6 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 which we use it 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
$$PK\{(\alpha, \beta) : \ C = h^\alpha \big(\frac{1}{z}\big)^\beta \ \wedge \ 1 = g^\alpha \big(\frac{1}{y}\big)^\beta\}$$
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 5. *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

$$C = h^\alpha \big(\frac{1}{z}\big)^\beta \qquad\qquad 1 = g^\alpha \big(\frac{1}{y}\big)^\beta \tag{21}$$

hold. From the second equation of (21) one can conclude that

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

Substituting $\beta \log_g y$ for α in the first equation of (21), we get $C = (h^{\log_g y}/z)^\beta$. As 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 briefly 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. Also, we note that the 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 (which the simulator can not compute, but a (dishonest) verifier can if he chooses h such the he knows $\log_g h$).

7 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.

7.1 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 §5.1. 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 $\delta \in \Delta$

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

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 ElGamal based Cramer-Shoup one [CS98]), 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, our protocols would 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* $(\mathcal{A}^*, \mathcal{P}^*)$*,*

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

Special honest-verifier zero knowledge: *There exists a simulator Sim such that for all adver-*

saries ($\mathcal{A}^*$, $\mathcal{B}^*$), *we have*

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

7.2 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 under 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

$$u^{2(x_2+\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3)}/v^2 = 1 \qquad \text{and} \qquad (e/u^{x_1})^2/h^{2m} = 1 \tag{22}$$

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 statement 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 §6. 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 §6 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 §5. 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 (22), 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 not know the values x_1, x_2, and 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 (22) does not hold, we prove that either

$$\Big(\frac{u^{2(x_2+\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3)}}{v^2}\Big)^n \neq 1 \tag{23}$$

or

$$\Big(\frac{u^{2(x_2+\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3)}}{v^2}\Big)^n = 1 \qquad \text{and} \qquad \frac{u^{2(x_2+\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3)}}{v^2} \neq 1 \tag{24}$$

or

$$\left(\frac{(e/u^{x_1})^2}{h^{2m}}\right)^n = (e/u^{x_1})^{2n} \neq 1 \tag{25}$$

or

$$\left(\frac{(e/u^{x_1})^2}{h^{2m}}\right)^n = 1 \qquad \text{and} \qquad \frac{(e/u^{x_1})^2}{h^{2m}} \neq 1 \tag{26}$$

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 §6 to prove that at least one of these four inequalities applies. We remark that the case where the statements (23-25) are false but the statement (26) is true corresponds to the case where the ciphertext is a valid encryption of a message different from m. If any of the statements (23-25) is true corresponds to the cases where the ciphertext is invalid.

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, and 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; the actual protocols are easily derived from it (cf. also the the verifiable encryption protocol presented in §5 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 and decrypts to m under label L, 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 under L, 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

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

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

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

(computed in $\mathbb{Z}$)). (To derive the actual protocol one may 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 the previous section (depending on whether Step 2 or Step 3 gets carried out).

Theorem 6. *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 the one 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. Let us generate a public keys and secret keys according to the usual algorithms, obtaining

$$n, g, y_1, y_2, y_3, x_1, x_2, x_3, \mathfrak{n}, \mathfrak{g}, \mathfrak{h}.$$

All of this information is available to the adversary, who produces m, ψ, L, and is able to make the verifier accept on these inputs with non-negligible probability. Using 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 $m = \mathcal{D}(1^\lambda, \mathsf{SK}, \psi, L)$. We consider these 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:

$$y_1{}^{\Delta c} = g^{\Delta x_1} \ , \qquad y_2{}^{\Delta c} = g^{\Delta x_2} \ , \qquad y_3{}^{\Delta c} = g^{\Delta x_3} \ , \tag{27}$$

$$v^{2\Delta c} = u^{2\Delta x_2} u^{2\mathcal{H}_{\mathsf{hk}}(u,e,L)\Delta x_3} \ , \quad \text{and} \tag{28}$$

$$(\frac{e^2}{h^{2m}})^{\Delta c} = u^{2\Delta x_1} \ . \tag{29}$$

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 (27) that

$$\Delta c\, x_i \equiv \Delta x_i \pmod{n'} \quad \text{for} \quad i = 1, \ldots, 3 \ . \tag{30}$$

Now, $\mathcal{D}(1^\lambda, \mathsf{SK}, \psi, L) \neq m$ means that least one of the four statements (23-26) must be true and therefore at least one of the two statements

$$u^{2(x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)x_3)} \neq v^2 \qquad \text{or} \qquad (e/u^{x_1})^2 \neq h^{2m} \tag{31}$$

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 (28) 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 (30) 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', which would allow us to factor n, which is impossible.

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 (29) and that $\Delta c x_1 - \Delta x_1$ is a non-zero multiple of n', which would again allow us to factor n, which is impossible.

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

$$v^2 = u^{2(x_2+\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3)} \qquad \text{and} \qquad u^{2x_1} = (\frac{e}{h^m})^2 \ . \tag{32}$$

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. [CDS94])). 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

$$y_1{}^{\Delta c} = g^{\Delta x_1} \ , \qquad y_2{}^{\Delta c} = g^{\Delta x_2} \ , \quad \text{and} \qquad y_3{}^{\Delta c} = g^{\Delta x_3} \ , \tag{33}$$

hold and either

$$C_1^{\Delta c} = u^{2n\Delta r_1}(\frac{1}{v})^{2n\Delta a_1} \ , \quad \mathfrak{C}_1^{\Delta c} = \mathfrak{g}^{\Delta a_1}\mathfrak{h}^{\Delta b_1} \ , \quad \text{and} \quad 1 = (\frac{1}{\mathfrak{C}_1})^{\Delta x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)\Delta x_3}\mathfrak{g}^{\Delta r_1}\mathfrak{h}^{\Delta s_1} \tag{34}$$

or

$$C_2^{\Delta c} = u^{2\Delta r_2}(\frac{1}{v})^{2\Delta a_2} \ , \quad \mathfrak{C}_2^{\Delta c} = \mathfrak{g}^{\Delta a_2}\mathfrak{h}^{\Delta b_2} \ , \quad \text{and} \quad 1 = (\frac{1}{\mathfrak{C}_2})^{\Delta x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)\Delta x_3}\mathfrak{g}^{\Delta r_2}\mathfrak{h}^{\Delta s_2} \tag{35}$$

or

$$C_3^{\Delta c} = u^{2n\Delta r_3}(\frac{1}{e})^{2n\Delta a_3} \ , \quad \mathfrak{C}_3^{\Delta c} = \mathfrak{g}^{\Delta a_3}\mathfrak{h}^{\Delta b_3} \ , \quad \text{and} \quad 1 = (\frac{1}{\mathfrak{C}_3})^{\Delta x_1}\mathfrak{g}^{\Delta r_3}\mathfrak{h}^{\Delta s_3} \tag{36}$$

or

$$C_4^{\Delta c} = u^{2\Delta r_4}(\frac{1}{e})^{n\Delta a_4} \ , \quad \mathfrak{C}_4^{\Delta c} = \mathfrak{g}^{\Delta a_4}\mathfrak{h}^{\Delta b_4} \ , \quad \text{and} \quad 1 = (\frac{1}{\mathfrak{C}_4})^{\Delta x_1}\mathfrak{g}^{\Delta r_4}\mathfrak{h}^{\Delta s_4} \tag{37}$$

hold. We know x_i such that $y_i = g^{x_i}$ and therefore it follows from (33) that

$$\Delta c\, x_i \equiv \Delta x_i \pmod{n'} \quad \text{for} \quad i = 1, \ldots, 3 \ . \tag{38}$$

We next consider the implications of the cases when the equations (34), the equations (35), the equations (36), or the equations (37) hold in conjunction with (33).

Case 1. Consider the case where Equations (33) and (34) hold. From the last two equations of (34) we get

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

Under the assumption that factoring $\mathfrak{n}$ is hard, and applying Theorem 2, we may assume that

$$\Delta a_1(\Delta x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)\Delta x_3) = \Delta c \Delta r_1 \ . \tag{39}$$

Because n is the product of two safe primes and we have $|\Delta c| < \min\{p, q, p'q'\}$, it follows from $C_1^2 \neq 1$ (which is checked by the verifier in Step 3c) that $C_1^{\Delta c} \neq 1$. From the first equation of (34) it follows that $u^{2n\Delta r_1} \neq v^{2n\Delta a_1}$. By Eq. (39) 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

$$u^{2n(\Delta x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)\Delta x_3)} \neq v^{2n\Delta c} \ . \tag{40}$$

From (40) and the first equation of (32) 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 (38) 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 can not occur.

Case 2. We consider the case where Equations (33) and (35) 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 (assuming $\mathfrak{n}$ is hard to factor). 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 (38) 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', which would allow us to factor n, which is a contradiction.

Case 3. Similarly as in case 1, from the Equations (33) and (36), one can derive that

$$u^{2n\Delta x_1} \neq e^{2n\Delta c} \tag{41}$$

holds (or we factor $\mathfrak{n}$ with non-negligible probability). From the second equation of (32) 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 (41), 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. (38) and thus this case can not occur.

Case 4. Similarly as before, from the Equations (33) and (37) one can show that

$$u^{2\Delta x_1} \neq (\frac{e}{h^m})^{2\Delta c} \tag{42}$$

holds (or we factor $\mathfrak{n}$ with non-negligible probability). From the second equation of (32) and from (42) 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.

□

7.3 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 §5.2; 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

$$u^{2(x_2+\mathcal{H}_{\mathsf{hk}}(u,e,L)x_3)}/v^2 = 1 \qquad \text{or} \qquad (e/u^{x_1})^{2n} = 1 \qquad \text{or} \qquad \delta = \gamma^{(\log_{h^2}(e/u^{x_1})^2 \operatorname{rem} n)} \tag{43}$$

hold. Note that $\log_{h^2}(e/u^{x_1})^2$ exists if and only if $(e/u^{x_1})^{2n} = 1$. The first two statements of (43) 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 [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 that an integer can always be represented as four squares and Rabin and Shallit [RS86] provide an efficient algorithm for finding such 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 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 obtains 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. [RS86]).

2. If (u, e, v) indeed decrypts to $\log_\gamma \delta$ under 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,

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

   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

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

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.[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 $C_4 \neq 1$.

Theorem 7. *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 $f'(\delta, \psi, L, \mathsf{SK})$ 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 6.

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 and stops. Otherwise, the simulator chooses random integers $m, w_1, \ldots, w_4 \in_R [-n/2, n/2]$.

If $f'(\delta, \psi, L, \mathsf{SK}) = 1$, it simulates step 2 as follows. It 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 it sends the values $\mathfrak{W}_1, \ldots, \mathfrak{W}_4$, and $\mathfrak{M}$ to the verifier and finally invokes the simulator for the PK-protocol of step 2.

If $f'(\delta, \psi, L, \mathsf{SK}) = 1$, 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 $\mathfrak{C}_1 := \mathfrak{h}^{b_1}$, $\mathfrak{C}_2 := \mathfrak{h}^{b_2}$, $\mathfrak{C}_3 := \mathfrak{h}^{b_3}$, $\mathfrak{C}_4 := \mathfrak{h}^{b_4}$, $\mathfrak{W}_1 := \mathfrak{h}^{t_1}$, $\mathfrak{W}_2 := \mathfrak{h}^{t_2}$, $\mathfrak{W}_3 := \mathfrak{h}^{t_3}$, $\mathfrak{W}_4 := \mathfrak{h}^{t_4}$, 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 $C_4 \in_R \Gamma$. It finally 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 6.

In the remainder we prove soundness. Let us generate a public keys and secret keys according to the usual algorithms, obtaining

$$n, g, y_1, y_2, y_3, x_1, x_2, x_3, \mathfrak{n}, \mathfrak{g}, \mathfrak{h}.$$

All of this information is available to the adversary, who produces δ, ψ, L, and is able to make the verifier accept on these inputs with non-negligible probability. 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 these two cases.

Case I. First assume that $\delta \neq \gamma^{\hat{m}}$ or $\mathsf{reject} = \mathcal{D}(\mathsf{SK}, \psi, L)$ 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:

$$y_1{}^{\Delta c} = g^{\Delta x_1} \qquad y_2{}^{\Delta c} = g^{\Delta x_2} \qquad y_3{}^{\Delta c} = g^{\Delta x_3} \tag{44}$$

$$v^{2\Delta c} = u^{2\Delta x_2} u^{2\mathcal{H}_{\mathsf{hk}}(u,e,L)\Delta x_3} \tag{45}$$

$$e^{2\Delta c} = u^{2\Delta x_1} h^{2\Delta m} \tag{46}$$

$$\mathfrak{W}_1^{\Delta c} = \mathfrak{g}^{\Delta w_1}\mathfrak{h}^{\Delta t_1} \qquad \mathfrak{W}_2^{\Delta c} = \mathfrak{g}^{\Delta w_2}\mathfrak{h}^{\Delta t_2} \qquad \mathfrak{W}_3^{\Delta c} = \mathfrak{g}^{\Delta w_3}\mathfrak{h}^{\Delta t_3} \qquad \mathfrak{W}_4^{\Delta c} = \mathfrak{g}^{\Delta w_4}\mathfrak{h}^{\Delta t_4} \tag{47}$$

$$\mathfrak{M}^{\Delta c} = \mathfrak{g}^{\Delta m}\mathfrak{h}^{\Delta t_5} \qquad \mathfrak{g}^{\Delta c (n-1)^2/4} = \mathfrak{M}^{\Delta m}\mathfrak{W}_1^{\Delta w_1}\mathfrak{W}_2^{\Delta w_2}\mathfrak{W}_3^{\Delta w_3}\mathfrak{W}_4^{\Delta w_4}\mathfrak{h}^{\Delta s} \tag{48}$$

$$\delta^{\Delta c} = \gamma^{\Delta m} \tag{49}$$

Consider the equations (47) and (48). Under the strong RSA assumption, and using Theorem 3, we may assume that Δc divides each of Δm, $\Delta w_1, \ldots, \Delta w_4$, $\Delta t_1, \ldots, \Delta t_5$, and Δs. So we compute $\hat{m} = \Delta m / \Delta c$, $\hat{w}_1 = \Delta w_1/\Delta c, \ldots, \hat{w}_4 = \Delta w_4/\Delta c$, $\hat{t}_1 = \Delta t_1/\Delta c, \ldots, \hat{t}_5 = \Delta t_5/\Delta c$, and $\hat{s} = \Delta s/\Delta c$ and we know that

$$\mathfrak{M} = \mathfrak{m}\mathfrak{g}^{\hat{m}}\mathfrak{h}^{\hat{t}_5} \quad \mathfrak{W}_1 = \mathfrak{w}_1\mathfrak{g}^{\hat{w}_1}\mathfrak{h}^{\hat{t}_1} \quad \mathfrak{W}_2 = \mathfrak{w}_2\mathfrak{g}^{\hat{w}_2}\mathfrak{h}^{\hat{t}_2} \quad \mathfrak{W}_3 = \mathfrak{w}_3\mathfrak{g}^{\hat{w}_3}\mathfrak{h}^{\hat{t}_3} \quad \mathfrak{W}_4 = \mathfrak{w}_4\mathfrak{g}^{\hat{w}_4}\mathfrak{h}^{\hat{t}_4} \quad \delta = \gamma^{\hat{m}} \tag{50}$$

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 (48) as follows

$$\mathfrak{g}^{(n-1)^2/4} = \mathfrak{a}\mathfrak{g}^{\hat{m}^2 + \sum \hat{w}_i^2}\mathfrak{h}^{\hat{m}\hat{t}_5 + \sum \hat{w}_i\hat{t}_i + \hat{s}} \tag{51}$$

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 $\mathfrak{n}$. Applying Theorem 2, we may assume that

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

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

$$-(n-1)/2 \leq \hat{m} \leq (n-1)/2 \ . \tag{52}$$

Consider Equations (44-46). 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 (44) that

$$\Delta c\, x_i \equiv \Delta x_i \pmod{n'} \quad \text{for} \quad i = 1, \ldots, 3 \ . \tag{53}$$

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

$$u^{2(x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)x_3)}/v^2 \neq 1 \qquad \text{or} \qquad (e/u^{x_1})^{2n} \neq 1 \qquad \text{or} \qquad (\frac{e}{u^{x_1}})^2 \neq h^{2\hat{m}} \tag{54}$$

must hold (cf. (43)), where the last is equivalent to $\delta \neq \gamma^{(m \operatorname{rem} n)}$ because of Equations (49) and (52) 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 (45) 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 (53) 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, a contradiction.

Case 2. If $u^{2nx_1} \neq e^{2n}$ we have that $u^{2n\Delta cx_1} \neq e^{2n\Delta c}$. Because of (46) and $h^n = 1$, we get

$$u^{2n\Delta cx_1} \neq u^{2n\Delta x_1} \qquad \text{and thus} \qquad \Delta cx_1 \not\equiv \Delta x_1 \pmod{n'} \ ,$$

because u^{2n} has order dividing n'. The latter, however, is a contradiction to Eqn. (53) and thus this case can not occur.

Case 3. The inequality $(\frac{e}{u^{x_1}})^2 \neq h^{2\hat{m}}$ is equivalent to $(\frac{e}{h^{\hat{m}}})^2 \neq u^{2x_1}$. Recalling that $\hat{m}\Delta c = \Delta m$ we can rewrite (46) as

$$\left(\frac{e^{\Delta c}}{h^{\Delta m}}\right)^2 = \left(\frac{e}{h^{\hat{m}}}\right)^{2\Delta c} = u^{2\Delta x_1} \qquad \text{and conclude that} \qquad u^{2\Delta cx_1} \neq u^{2\Delta x_1} \ .$$

Similarly to case 1, it follows that $\Delta cx_1 - \Delta x_1$ is a multiple of n' and we are again able to factor n, a contradiction.

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

$$u^{2(x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)x_3)}/v^2 = 1 \qquad (e/u^{x_1})^{2n} = 1 \qquad \delta = \gamma^{(\log_{h^2}(e/u^{x_1})^2 \,\mathrm{rem}\, n)} \tag{55}$$

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

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

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. [CDS94])). W.l.o.g., suppose that $c^{(2)} > c^{(1)}$. Let

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

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

$${y_1}^{\Delta c} = g^{\Delta x_1} \ , \qquad {y_2}^{\Delta c} = g^{\Delta x_2} \ , \quad \text{and} \qquad {y_3}^{\Delta c} = g^{\Delta x_3} \ , \tag{56}$$

hold and either

$$C_1^{\Delta c} = u^{2n\Delta r_1}(\frac{1}{v})^{2n\Delta a_1} \ , \quad \mathfrak{C}_1^{\Delta c} = \mathfrak{g}^{\Delta a_1}\mathfrak{h}^{\Delta b_1} \ , \quad \text{and} \quad 1 = (\frac{1}{\mathfrak{C}_1})^{\Delta x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)\Delta x_3}\mathfrak{g}^{\Delta r_1}\mathfrak{h}^{\Delta s_1} \tag{57}$$

or

$$C_2^{\Delta c} = u^{2\Delta r_2}(\frac{1}{v})^{2\Delta a_2} \ , \qquad \mathfrak{C}_2^{\Delta c} = \mathfrak{g}^{\Delta a_2}\mathfrak{h}^{\Delta b_2} \ , \quad \text{and} \quad 1 = (\frac{1}{\mathfrak{C}_2})^{\Delta x_2 + \mathcal{H}_{\mathsf{hk}}(u,e,L)\Delta x_3}\mathfrak{g}^{\Delta r_2}\mathfrak{h}^{\Delta s_2} \tag{58}$$

or

$$C_3^{\Delta c} = u^{2n\Delta r_3}(\frac{1}{e})^{2n\Delta a_3} \ , \quad \mathfrak{C}_3^{\Delta c} = \mathfrak{g}^{\Delta a_3}\mathfrak{h}^{\Delta b_3} \ , \quad \text{and} \quad 1 = (\frac{1}{\mathfrak{C}_3})^{\Delta x_1}\mathfrak{g}^{\Delta r_3}\mathfrak{h}^{\Delta s_3} \tag{59}$$

or

$$C_4^{\Delta c} = \gamma^{\Delta r_4}(\frac{1}{\delta})^{\Delta a_4} \ , \qquad \mathfrak{C}_4^{\Delta c} = \ \mathfrak{g}^{\Delta a_4}\mathfrak{h}^{\Delta b_4} \ , \qquad 1 = (\frac{1}{\mathfrak{C}_4})^{\Delta m}\mathfrak{g}^{\Delta r_4}\mathfrak{h}^{\Delta s_4} \tag{60}$$

$$e^{2\Delta c} = u^{2\Delta x_1}h^{2\Delta m} \ , \quad \mathfrak{M}^{\Delta c} = \mathfrak{g}^{\Delta m}\mathfrak{h}^{\Delta t_5} \ , \quad \mathfrak{g}^{\Delta c(n-1)^2/4} = \mathfrak{M}^{\Delta m}\mathfrak{W}_1^{\Delta w_1}\mathfrak{W}_2^{\Delta w_2}\mathfrak{W}_3^{\Delta w_3}\mathfrak{W}_4^{\Delta w_4}\mathfrak{h}^{\Delta s_5} \tag{61}$$

$$\mathfrak{W}_1^{\Delta c} = \mathfrak{g}^{\Delta w_1}\mathfrak{h}^{\Delta t_1} \ , \quad \mathfrak{W}_2^{\Delta c} = \mathfrak{g}^{\Delta w_2}\mathfrak{h}^{\Delta t_2} \ , \quad \mathfrak{W}_3^{\Delta c} = \mathfrak{g}^{\Delta w_3}\mathfrak{h}^{\Delta t_3} \ , \quad \text{and} \quad \mathfrak{W}_4^{\Delta c} = \mathfrak{g}^{\Delta w_4}\mathfrak{h}^{\Delta t_4} \ . \tag{62}$$

hold. We know x_i such that $y_i = g^{x_i}$ and therefore it follows from (33) that

$$\Delta c\, x_i \equiv \Delta x_i \pmod{n'} \quad \text{for} \quad i = 1, \ldots, 3 \ . \tag{63}$$

We next consider the implications of the cases when the equations (57), the equations (58), the equations (59), or the equations (60-62) hold in conjunction with (56). The first three cases appear also in the proof of Theorem 6, while the last one is different:

Case 4. Similarly as in Case I above, from the Equations (61) and (62) we can derive that

$$e^{2\Delta c} = u^{2\Delta x_1}h^{2\Delta c\hat{m}} \qquad \text{and} \qquad -(n-1)/2 \le \hat{m} \le (n-1)/2 \tag{64}$$

where $\hat{m} = \Delta m/\Delta c$. Using Equations (63) 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 (64),

$$\hat{m} = (\log_{h^2} u^{2x_1}/e^2 \operatorname{rem} n) \tag{65}$$

Similarly as we did in Case II in the proof of Theorem 6, one can derive from the last two equations of (60) that

$$\Delta r_4 = \Delta a_4 \hat{m} \tag{66}$$

holds (using the strong RSA assumption for $\mathfrak{n}$). Now using (66) in the first equation of (60)

$$C_4^{\Delta c} = \gamma^{\Delta a_4 \hat{m}}(\frac{1}{\delta})^{\Delta a_4} \qquad \text{and} \qquad C_4 = (\frac{\gamma^{\hat{m}}}{\delta})^{\hat{a}_4} \ , \tag{67}$$

where $\hat{a}_4 := \Delta a_4/\Delta c \pmod{\rho}$. Because $C_4 \neq 1$ we must have that $\delta \neq \gamma^{\hat{m}}$ and because of (65) that

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

which is a contradiction to the third equation of (55) and hence this case can not occur.

□

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 (M. Bellare, ed.), 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 (L. Knudsen, ed.), LNCS, vol. 2332, Springer, 2002, pp. 83–107.

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

[ASW98] N. Asokan, V. Shoup, and M. Waidner, *Optimistic fair exchange of digital signatures*, Advances in Cryptology — EUROCRYPT '98 (K. Nyberg, ed.), LNCS, vol. 1403, Springer Verlag, 1998, pp. 591–606.

[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*, Tech. Report TR 688, MIT Laboratory for Computer Science, April 1996.

[Bou00] F. Boudot, *Efficient proofs that a committed number lies in an interval*, Advances in Cryptology — EUROCRYPT 2000 (B. Preneel, ed.), 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.

[BP97] N. Barić and B. Pfitzmann, *Collision-free accumulators and fail-stop signature schemes without trees*, Advances in Cryptology — EUROCRYPT '97 (W. Fumy, ed.), LNCS, vol. 1233, Springer Verlag, 1997, pp. 480–494.

[BS02] E. Bresson and J. Stern, *Proofs of knowledge for non-monotone discrete-log formulae and applications*, Information Security (ISC 2002) (A. H. Chan and V. Gligor, eds.), 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 (T. Okamoto, ed.), 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 (Y. G. Desmedt, ed.), LNCS, vol. 839, Springer Verlag, 1994, pp. 174–187.

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

[CFT98] A. Chan, Y. Frankel, and Y. Tsiounis, *Easy come – easy go divisible cash*, Advances in Cryptology — EUROCRYPT '98 (K. Nyberg, ed.), LNCS, vol. 1403, Springer Verlag, 1998, pp. 561–575.

[CG98] D. Catalano and R. Gennaro, *New efficient and secure protocols for verifiable signature sharing and other applications*, Advances in Cryptology — CRYPTO '98 (Berlin) (H. Krawczyk, ed.), 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 (J. Stern, ed.), LNCS, vol. 1592, Springer Verlag, 1999, pp. 90–106.

[CGGM00] R. Canetti, O. Goldreich, S. Goldwasser, and S. Micali, *Resettable zero-knowledge*, Proc. 32st Annual ACM Symposium on Theory of Computing (STOC), ACM Press, 2000, pp. 235–244.

[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 (A. De Santis, ed.), 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 (B. Pfitzmann, ed.), LNCS, vol. 2045, Springer Verlag, 2001, pp. 93–118.

[CM98] J. Camenisch and M. Michels, *A group signature scheme with improved efficiency*, Advances in Cryptology — ASIACRYPT '98 (K. Ohta and D. Pei, eds.), LNCS, vol. 1514, Springer Verlag, 1998, pp. 160–174.

[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 (J. Stern, ed.), 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 (M. Wiener, ed.), 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 (B. Preneel, ed.), 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 (E. F. Brickell, ed.), LNCS, vol. 740, Springer-Verlag, 1993, pp. 89–105.

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

[CS98] R. Cramer and V. Shoup, *A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack*, Advances in Cryptology — CRYPTO '98 (Berlin) (H. Krawczyk, ed.), LNCS, vol. 1642, Springer Verlag, 1998, pp. 13–25.

[CS00] R. Cramer and V. Shoup, *Signature schemes based on the strong RSA assumption*, ACM Transactions on Information and System Security **3** (2000), no. 3, 161–185.

[CS02] R. Cramer and V. Shoup, *Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption*, Advances in Cryptology — EUROCRYPT 2002, LNCS, vol. 2332, Springer, 2002, pp. 45–64.

[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, acm press, 2002.

[Dam00] I. Damgård, *Efficient concurrent zero-knowledge in the auxiliary string model*, Advances in Cryptology — EUROCRYPT 2000 (B. Preneel, ed.), 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, Springer, 2002.

[DNS98] C. Dwork, M. Naor, and A. Sahai, *Concurrent zero knowledge*, Proc. 30th Annual ACM Symposium on Theory of Computing (STOC), 1998.

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

[FR95] M. Franklin and M. Reiter, *Verifiable signature sharing*, Advances in Cryptology — EUROCRYPT '95 (L. C. Guillou and J.-J. Quisquater, eds.), LNCS, vol. 921, Springer Verlag, 1995, pp. 50–63.

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

[GHR99] R. Gennaro, S. Halevi, and T. Rabin, *Secure hash-and-sign signatures without the random oracle*, Advances in Cryptology — EUROCRYPT '99 (J. Stern, ed.), LNCS, vol. 1592, Springer Verlag, 1999, pp. 123–139.

[GL03] R. Gennaro and Y. Lindell, *A framework for password-based authenticated key exchange*, Advances in Cryptology — EUROCRYPT 2003 (E. Biham, ed.), LNCS, vol. 2656, Springer Verlag, 2003, pp. 524–543.

[KP98] J. Kilian and E. Petrank, *Identity escrow*, Advances in Cryptology — CRYPTO '98 (Berlin) (H. Krawczyk, ed.), 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 (K. Nyberg, ed.), 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 (J. Stern, ed.), 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 (J. Feigenbaum, ed.), 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 (B. Preneel, ed.), 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.

[SG98] V. Shoup and R. Gennaro, *Securing threshold cryptosystems against chosen ciphertext attack*, Advances in Cryptology — EUROCRYPT '98 (K. Nyberg, ed.), 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 (U. Maurer, ed.), 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 (K. Nyberg, ed.), LNCS, vol. 1403, Springer Verlag, 1998, pp. 17–31.