Robust and Efficient Sharing of RSA Functions

Rosario Gennaro[*], Stanisław Jarecki[*], Hugo Krawczyk[**] and Tal Rabin[*]

Abstract. We present two efficient protocols which implement robust threshold RSA signature schemes, where the power to sign is shared by N players such that any subset of T or more signers can collaborate to produce a valid RSA signature on any given message, but no subset of fewer than T corrupted players can forge a signature. Our protocols are robust in the sense that the correct signature is computed even if up to $T-1$ players behave in arbitrarily malicious way during the signature protocol. This in particular includes the cases of players that refuse to participate or that generate incorrect partial signatures. Our robust protocols achieve optimal resiliency as they can tolerate up to $(N-1)/2$ faults, and their efficiency is comparable to the efficiency of the underlying threshold RSA signature scheme.

Robust threshold signature schemes have very important applications, since they provide increased security and availability for a signing server (e.g. a certification authority or an electronic cash provider). Solutions for the case of the RSA signature scheme are especially important because of its widespread use. In addition, these techniques apply to shared RSA decryption as well, thus leading to efficient key escrow schemes for RSA.

Our schemes are based on some interesting extensions that we devised for the *information checking protocol* of T. Rabin and Ben-Or [Rab94, RB89], and the *undeniable signature* work initiated by Chaum and van Antwerpen [CA90]. These extensions have some attractive properties, and hence are of independent interest.

1 Introduction

The idea of distributed signature schemes is to depart from the single-signer approach in which one person is the sole holder of a secret key, and to allow a group of people to "hold" the key in such a manner that they can, as a group, produce signatures, yet no person on his own can generate a signature. The signature which is generated by the group is the same as if it were generated by a single signer.

We say that a distributed signature scheme is a (T, N)*-threshold signature scheme*, if given a message m, and a group of N players, where each one holds a part of a secret key, any subset of T or more players can generate the signature for m. We say that the scheme is *secure or unforgeable* if no coalition of fewer than T players can produce a valid signature on a new message, even after the system has produced many signatures for different messages. Furthermore, a (T, N)-threshold signature scheme is *robust or fault-tolerant* if it can correctly compute signatures, even in the presence of up to $T-1$ arbitrarily malicious players.

Note that a simple reconstruction of the key in the hands of a single player at signing time would not satisfy our requirement, as it allows future signatures to be generated by this single player (i.e., such reconstruction would create a single point of failure).

A complete version of the paper is available from http://www.research.ibm.com/security/.

[*] MIT Laboratory for Computer Science, 545 Tech Square, Cambridge, MA 02139, USA. Email: {rosario,stasio,talr}@theory.lcs.mit.edu

[**] IBM T.J.Watson Research Center, PO Box 704, Yorktown Heights, New York 10598, USA. Email: hugo@watson.ibm.com

Robust threshold signature schemes have very important applications, and we shall exemplify this briefly. The secret key of a global certification authority will be an attractive target for attacks. If the key is held in one site then once this site is broken into, the key is exposed. Yet, if the key is distributed among N sites, using a (T, N)-threshold signature scheme, then one needs to break into T sites in order to learn the key and forge signatures. In addition, once a site is broken into, it might exhibit arbitrary performance, yet the system should still be able to generate signatures. Hence, we increase the security and availability of a system by distributing the secret key. Furthermore, if the key is held on a single site, then signatures cannot be generated if this site crashes. Note that the trivial solution of key replication solves the availability problem, yet generates more sites that hold the full key. A robust threshold signature protocol can in particular tolerate up to $T - 1$ such crashes (and even arbitrary malicious actions), thus increasing the availability of the signature operation, without decreasing its security.

Threshold signatures are part of the general approach known as *threshold cryptography* introduced through the works of Boyd [Boy86], Desmedt [Des94], and Desmedt and Frankel [DF90]. Solutions for the case of the RSA signature scheme are especially important because of its widespread use (a de-facto standard). In addition, since in the RSA cryptosystem the signing algorithm coincides with the decryption algorithm, solutions to shared RSA signatures usually lead to shared RSA decryption procedures which have applications to key escrow (cf. [Mic92]). Desmedt and Frankel initiated the study of threshold-RSA [DF92], which was followed by De Santis, Desmedt, Frankel, and Yung [DDFY94]. These papers provide solutions for the problem of threshold RSA signatures, however, they lack the robustness property.

In this paper, we present an efficient solution for robust (T, N)-threshold RSA signatures, for any threshold value $T \leq \lceil N/2 \rceil$. We use the solution of [DDFY94] as the basic threshold RSA scheme. We achieve the additional property of fault-tolerance by employing extensions which we developed of the *Information Checking Protocol* of T. Rabin and Ben-Or [Rab94, RB89], and the *undeniable signature* work initiated by Chaum and van Antwerpen [CA90]. These extensions have desirable properties, and hence can be used in other applications as well.

In a recent and independent work, Frankel, Gemmel, and Yung [FGY96], have extended the notion of *result-checking* introduced by Blum [BK89], to the setting of *witness-based cryptographic checking*. Among the main motivations for that work is the generation of a robust threshold RSA signature scheme. While [FGY96] provides a more general theoretical framework, our techniques, specifically designed for RSA, result in much more efficient and practical solutions. In particular, our basic protocols involve just a small constant number of modular exponentiations while in [FGY96] a very large number of such costly exponentiations is required.

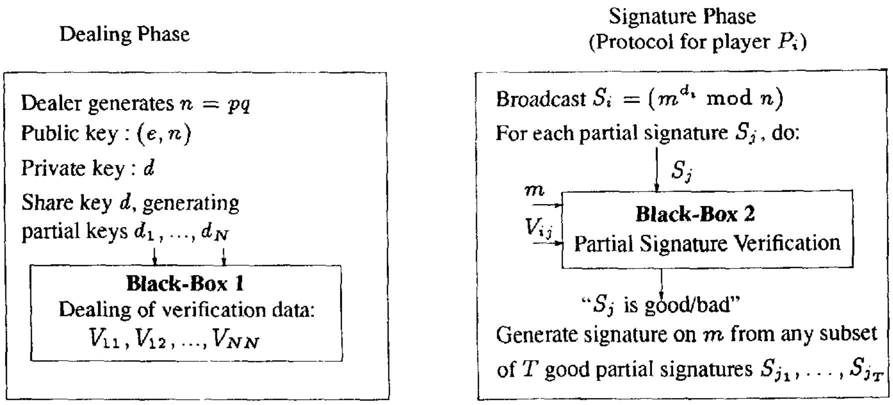
2 Our results

The basic construction underlying the existing threshold RSA schemes can be roughly described as follows: given $n = pq$, where p and q are big primes and $\phi(n) = (p-1)(q-1)$, the public RSA key is a pair (n, e), where $\gcd(e, \phi(n)) = 1$, the secret key is a number

d, s.t. $ed \equiv 1 \bmod \phi(n)$, and the signature on a message m is $S = m^d \bmod n$ [1]. A distributed (T, N)-threshold signature scheme has two phases. In the first one, called the Dealing Phase, a dealer shares the secret key d, among the N players, such that each player P_i has a "share" d_i of d. These shares are created in such a manner, that in the second phase, which is called the Signature Phase, when a message m is given, any subset of T partial signatures $S_i = m^{d_i} \bmod n$ suffices to generate the signature for m. In reality, the actual techniques used in the known solutions to the threshold RSA problem are more involved. We further elaborate on the techniques of [DDFY94] in Section 3.1.

All of the existing schemes *require*, in order for the generated signature to be correct, that all partial signatures S_i be correct as well. Consequently, these schemes cannot tolerate faults. If we are to confront failures, we must be able to detect which of the partial signatures provided by the players are improper. Once the incorrect partial signatures are sieved out, the computation can be carried out the same way as in the case where there are no faults.

In this paper we concentrate on providing solutions for the problem of detecting an improper partial signature. To achieve this goal, additional *verification data*, denoted $V_{11}, V_{12}, \ldots, V_{NN}$ is generated in the Dealing Phase, where V_{ij} is a piece of information used by player P_i during the signature phase to check the partial signature S_j provided by another player P_j. Following we present a schema that represents in a generic way the phases and components of a robust, threshold RSA signature scheme:


Dealing Phase
Dealer generates $n = pq$
Public key : (e, n)
Private key : d
Share key d, generating partial keys $d_1, \ldots, d_N$
Black-Box 1
Dealing of verification data: $V_{11}, V_{12}, \ldots, V_{NN}$
Signature Phase (Protocol for player P_i)
Broadcast $S_i = (m^{d_i} \bmod n)$
For each partial signature S_j, do:
S_j
m
V_{ij}
Black-Box 2
Partial Signature Verification
"S_j is good/bad"
Generate signature on m from any subset of T good partial signatures $S_{j_1}, \ldots, S_{j_T}$

The protocols presented in this paper correspond to the "black-boxes" shown in the schema. Using these protocols we can prove the following theorem:

Theorem 1. *There exist efficient, robust (T, N)-threshold RSA signature schemes for any value $T \leq \lceil N/2 \rceil$.*

We observe that the performance of these verification protocols adds complexity to the threshold cryptosystem. This added complexity can be saved under proper operation of the system: each player can try to generate a signature from any subset of T broadcasted partial

[1] For simplicity we assume m to be the hash or other proper encoding of the original message to be signed.

signatures, and then check the obtained signature using the known public exponent e. If the result is a proper signature, there is no need for the verification of the partial signatures. Only if the combined signature fails should the verification protocols that we provide be triggered.

In the known threshold RSA solutions (in particular, [DDFY94]), the shares $d_1, \ldots, d_N$ of the secret key d can be viewed themselves as RSA secret keys. Yet, for each of these keys the corresponding *public exponent*, e_i, is *unknown* even to the signer, P_i. Furthermore, the public exponent e_i *must not* be known, as it would expose $\phi(n)$ (and hence would allow P_i to produce full signatures by itself without collaboration of a threshold of players). *Hence, we are faced with the problem of checking a partial signature for which we do not have a public exponent.* Under the circumstances where the public exponent is not known, there must be some other information that allows the verification of such partial signatures. This information is the data $V_{11}, V_{12}, \ldots, V_{NN}$ which is generated in the Dealing Phase.

The trade-offs between different properties of a fault-tolerant threshold RSA depend on the particular application of these techniques. For example, some applications may require minimal communication between the players during the Signature Phase. Other might need that partial signatures will be "publicly verifiable", namely, that any person, even outside the group of N players, can verify every partial signature. The degree to which we trust the dealer is another variable requirement.

We have devised two different protocols for verification of partial signatures to address these different requirements. The choice of which protocol to use depends on the needs of a specific application. The first protocol has a *non-interactive* Signature Phase, and requires each player to hold local secret verification data. The second protocol requires *interaction* in order to verify partial signatures, yet all the verification data is public (in particular, even parties not present at the dealing phase can later act as verifiers of partial signatures). In the Dealing Phase of the latter protocol a publicly known *sample* message and its corresponding partial signatures are generated. In the Signature Phase of this protocol anyone can verify any player's partial signature by interacting with the signer, and using the sample signature as the basis for the proof. Both solutions are efficient computation-wise (a verification involves only a few RSA exponentiations). Surprisingly, our non-interactive verification is somewhat more efficient in computation than the interactive one. Communication-wise the non-interactive solution is clearly optimal, while the interactive solution requires the exchange of four messages between every two players. Both protocols leak no information that can be used by the players (even malicious ones) to forge signatures. The latter is a fundamental property of our protocols which we prove rigorously using zero-knowledge techniques [GMR89].

A final comment on the level of trust needed from the dealer, i.e. the entity that shares the secret key and the verification data $V_{11}, V_{12}, \ldots, V_{NN}$ among the players. Unfortunately it is not known how to efficiently generate in a distributed manner a shared RSA key without the help of a trusted party[2] (in principle, one could use generic results for secure multiparty computation [GMW87], but those are outside the realm of practicality). Hence, at this stage of knowledge, we need to assume that a single dealer generates the public/private RSA keys, and then this dealer needs to be trusted for the secrecy of these keys. For the non-interactive solution we always assume that the dealer (which is active in the dealing

[2] Such results are known for discrete log based signature schemes [Fel87, Ped91, GJKR96].

phase only) is honest. On the other hand, the interactive solution allows for verification of the actions of the dealer during the dealing phase, including the proper generation of the key shares and the sample signatures. Threshold-based key-escrow system is a good example of an application where the ability to verify the dealer's actions is important, because in key-escrow the trustees need to verify whether a player gives them valid shares of the decryption key she or he uses for secret communication.

3 Preliminaries

MODEL. We assume that our computation model is composed of a set of N *players* $\{P_1, \ldots, P_N\}$. They are connected by a complete network of private (i.e. untappable) point-to-point channels. In addition, they have access to a dedicated broadcast channel; by dedicated we mean that if P_i broadcasts a message, it will be recognized by the other players as coming from P_i.

These assumptions (privacy of the communication channels and dedication of the broadcast channel) allow us to focus on a high-level description of the protocols. However, it is worth noticing that these abstractions can be substituted with standard cryptographic techniques for privacy and authentication.

THE ADVERSARY. We assume that an adversary, $\mathcal{A}$, can corrupt up to $T-1$ of the N players in the network. We consider the worse possible kind of adversary, i.e. a *malicious* adversary that learns all the information held by the corrupted players and hears the broadcasted messages. He may cause corrupted players to behave in *any* (possibly malicious) way. We assume a computationally bounded adversary which is unable to forge (regular) RSA signatures. We omit from these proceedings any formal definitions. See [GJKR96] for the definition of secure threshold signature schemes (based on [GMR88]).

3.1 Threshold Sharing of RSA functions

In [DDFY94] the authors show how to perform threshold sharing of RSA functions. The main problem is that it is not possible to use Shamir's (T, N)-threshold secret sharing [Sha79] in a straightforward way, as the secret key d belongs to the ring $Z_{\phi(n)}$ where polynomial interpolation is not always possible.

De Santis et al. [DDFY94] solve this problem by extending the ring of integers modulo $\phi(n)$ to a structure (a module) where interpolation is always possible. A sketch of their work is presented here, but readers are referred to [DDFY94] for complete details. Let G be $Z_{\phi(n)}$ and H be Z_n^*, where n is an RSA modulus. Let $\pi > N$ be a small prime and let u be a root of the cyclotomic polynomial $p(x) = \sum_{j=0}^{\pi-1} x^j$. Consider the ring $G[u] \equiv G[x]/(p(x)) \equiv G^{\pi-1}$. Elements in ring $G^{\pi-1}$ are $(\pi-1)$-dimensional vectors of elements in G. Addition and multiplication are defined using the isomorphism to $G[x]/(p(x))$, where to every element $(a_1, \ldots, a_{\pi-1}) \in G^{\pi-1}$ there corresponds a polynomial $\sum_{j=1}^{\pi-1} a_j x^{(j-1)}$ in $G[x]/(p(x))$, or element $\sum_{j=1}^{\pi-1} a_j u^{(j-1)}$ in $G[u]$. If one defines $x_i \overset{\text{def}}{=} \sum_{j=0}^{i-1} u^j$, it turns out that all x_i's and $(x_i - x_j)$'s have (multiplicative) inverses in $G[u]$.

The extended Shamir scheme is then as follows: the dealer chooses a random polynomial R over the ring $G^{\pi-1}$ of degree $T-1$ such that $R(0) = [d, 0, \ldots, 0]$, and gives $d_i = R(x_i)$

to player P_i. Because of the properties of the x_i's it is then possible to perform polynomial interpolation from any T shares to reconstruct $[d, 0, \ldots, 0] = \sum_j \lambda_j d_j$ where λ_j are the appropriate Lagrange coefficients.

However, in our case we need to recover m^d mod n (the signature on m) rather than d. If $d_i = [d_{i,1}, \ldots, d_{i,(\pi-1)}] \in G^{\pi-1}$ then P_i broadcasts $S_i = [m^{d_{i,1}}, \ldots, m^{d_{i,(\pi-1)}}] \in H^{\pi-1}$. Given T of the shares S_j, one can compute $S = \prod_j S_j^{\lambda_j}$, where multiplication and exponentiation in $H^{\pi-1}$ are defined as the natural extensions of addition and multiplication in $G^{\pi-1}$, respectively. The signature m^d mod n will be the first component of the vector S.

From the above brief description it is apparent that the correctness of the (interpolated) signature computed in this way heavily relies on the correctness of each partial signature.

Remark 1: The partial signature of player P_i in this scheme is just a vector of $\pi - 1$ "regular" RSA signatures. Hence, it will suffice to check each of these component signatures on its own. This will allow us in the following to perform operations simply modulo n and not in the algebraic structure described above. Thus resulting in a simpler, yet complete, exposition of our protocols.
Remark 2: Throughout this paper we need the following technical assumption. P_i's partial signature S_i will be accepted as valid if $S_i = m^{d_i}$ mod n or if $S_i = -m^{d_i}$ mod n. This might result in the final interpolated signature being "correct up to its sign", but this can be easily checked (and corrected) using the public exponent e.

3.2 Notation

For a positive integer k we denote $[k] \overset{\text{def}}{=} \{1, \cdots, k\}$. The public modulus is denoted by n. We assume $n = pq$, and $p = 2p' + 1$, $q = 2q' + 1$, where $p < q$ and p, q, p', q' are all prime numbers. Z_n^* denotes the multiplicative group of integers modulo n, and $\phi(n) = (p-1)(q-1)$ the order of this group. For an element $w \in Z_n^*$ we denote by $ord(w)$ the order of w in Z_n^* and by $ind(w)$ the index of w in this group (it holds that $ind(w) = \phi(n)/ord(w)$). The subgroup generated by an element $w \in Z_n^*$ is denoted by $<w>$. The number $d \in [\phi(n)]$ denotes the (private) signature exponent. For any message $m \in Z_n^*$ we denote by S_m the corresponding signature on m, namely, $S_m = m^d$ mod n.

4 Non-Interactive Robust Threshold RSA

Here we present our non-interactive solution to the robustness problem of threshold RSA signatures. Section §4.2 contains the protocol for dealing verification information during the Dealing Phase, while in section §4.3 we describe the protocol for verification of partial signatures during the Signature Phase. (See the schema in Section 2 for a schematic representation of the role of these components in the full threshold signature protocol.) Our solution is based on the Information Checking Protocol (ICP) from [Rab94, RB89]. The original ICP technique is intended for *one-time* verification of information provided by an untrusted party. In our case we extend this technique to verification of *multiple* partial signatures; in particular, we extend ICP to work over the integers rather than over a prime field as originally designed.

To understand the role of the information checking protocol in our non-interactive verification, we first give a very rough sketch of the non-interactive solution. Consider two players P and V (in our case P is the signer and V is a party that verifies P 's partial signature). The prover P holds values d_P (the secret key) and y. The verifier V holds b and c, such that $y = bd_P + c$. The values d_P, y, b and c are dealt to the corresponding parties during the dealing phase (and kept secret by the parties). Given a message m, the prover generates the partial signature $m^{d_P} \bmod n$, and the additional information $m^y \bmod n$. P gives these values to V, who verifies the partial signature by checking whether $(m^{d_P})^b m^c = m^y (\mathrm{mod} n)$. An important technical aspect of this solution is that the equality $y = bd_P + c$ needs to hold over the integers. The more natural approach of generating this equation modulo $\phi(n)$, would enable P and V to combine their information and compute a multiple of $\phi(n)$ which, in turn, would allow for the efficient factorization of n. In the next subsection we present an extension over the integers of the original ICP protocol.

4.1 Extensions of Information Checking

The following protocol is carried out by three players: a dealer D, who is non-faulty, and two additional players: prover P and verifier V , who can be either faulty or not. In Figure 1 we present the ICP-Generation protocol over the integers, carried out by the dealer.

ICP-Gen-Integers

Input: RSA composite n, secret value $d_P \in [\phi(n)]$ known to D,
security parameters $0 \leq \delta_1, \delta_2 \leq 1$.

1. Choose $b \in [n^{\delta_1}]$ and $c \in [n^{1+\delta_1+\delta_2}]$ with uniform distribution.
2. Compute $y = c + bd_P$ over the integers.
3. Secretly transmit d_P and y to the prover P .
4. Secretly transmit b and c to the verifier V .

Fig. 1. The ICP Generation Protocol

In a generic (one-time) application of ICP the variables y, d_P and b, c are used by players P and V in the following way: When the prover P wants to prove to the verifier V that he holds the value d_P which he received from D, he sends d_P and y to V . Upon receiving values $\hat{d}_P, \hat{y}$ from P , the verifier concludes that $\hat{d}_P = d_P$ only if $\hat{y} = b\hat{d}_P + c$. For the following we denote $\mathcal{Y} \overset{\text{def}}{=} [n^{\delta_1+1}, \ldots, n^{1+\delta_1+\delta_2}]$.

Lemma 2. *Given values $d_P \in [\phi(n)]$ and $y \in \mathcal{Y}$, for every possible value of b there is exactly one possible value for $c \in [n^{1+\delta_1+\delta_2}]$ such that $y = bd_P + c$.*

Proof. Since the computation is over the integers, there is exactly one value of c for each b, d_P and y. Furthermore, if $b \in [n^{\delta_1}]$ and $y \in [n^{\delta_1+1}, \ldots, n^{1+\delta_1+\delta_2}]$ then value $c = y - bd_P$ is contained in $[n^{1+\delta_1+\delta_2}]$ because

$$1 \leq n^{1+\delta_1} - n^{\delta_1}\phi(n) = y_{min} - b_{max}d_{P_{max}} \leq c \leq y_{max} - b_{min}d_{min} = n^{1+\delta_1+\delta_2} - 1$$

which proves the lemma. □

Lemma 3. $Pr(\, y \notin \mathcal{Y}\,) \leq \frac{1}{n^{\delta_2}}$.

Proof. The number of options to choose different pairs of b and c is $n^{1+\delta_1+\delta_2}n^{\delta_1}$. The range $\mathcal{Y}$ is of size $n^{1+\delta_1+\delta_2} - n^{1+\delta_1}$. From Lemma 2 it follows that each value y in this range can be generated by n^{δ_1} pairs (b, c). Consequently, the probability of y falling outside of this range is $1 - \frac{(n^{1+\delta_1+\delta_2} - n^{1+\delta_1})n^{\delta_1}}{n^{1+\delta_1+\delta_2}n^{\delta_1}} = \frac{1}{n^{\delta_2}}$. □

Lemma 4. ICP over the Integers.

Completeness. *If P and V follow the protocol then V always accepts d_P.*
Soundness. *The probability that P generates $\hat{d}_P, \hat{y}$ such that $c + b\hat{d}_P = \hat{y}$ when in fact $\hat{d}_P \neq d_P$ is at most $\frac{1}{n^{\delta_1}} + \frac{1}{n^{\delta_2}}$. We denote this occurrence as OC_1.*
Zero-knowledge. *Given b, c the verifier learns no additional information on the value d_P.*

Proof.
Completeness. Immediate

Soundness. Notice that $\Pr(\, OC_1\,) \leq \Pr(\, OC_1 \mid y \in \mathcal{Y}\,) + \Pr(\, y \notin \mathcal{Y}\,)$. From Lemma 2 it follows that if $y \in \mathcal{Y}$ then P will be able to generate values $\hat{d}_P, \hat{y}$ (where $\hat{d}_P \neq d_P$) which satisfy the equation $\hat{y} = b\hat{d}_P + c$ with probability at most $\frac{1}{n^{\delta_1}}$. Lemma 3 gives us the probability that y falls out of this range. Combining these probabilities we prove our lemma.

Zero-knowledge. Values b and c are uniformly distributed and randomly chosen without any correlation to d_P, and hence reveal no information on its value. □

4.2 Generation of Verification Data (Dealing Phase)

In order to generate the data for verification of partial signatures within the context of the Dealing Phase, the dealer simply runs the ICP-Gen-Integers (Figure 1) for every pair of players P_i and P_j. All these invocations have as input the same RSA composite n and security parameters δ_1, δ_2. For the invocation where P_i is the prover P , and P_j is the verifier V , the secret key input is d_i, namely P_i's secret partial key.[3]

It will be seen later that a single pair of values b, c suffices for V to verify multiple different signatures. This results in an efficient protocol for the dealer, as the number of invocations to the ICP-Gen-Integers protocol during the dealing phase depends only on the number of players but not on the number of signatures that the system will need to generate. As a result of the complete Dealing Phase, player P_i holds the following values:

1. His share d_i.
2. Auxiliary authentication values $y_{i,1}, \ldots, y_{i,N}$ where $y_{i,j} \in \boldsymbol{Z}$ is used to prove his partial signature to P_j.
3. Verification data $V_{1,i}, \ldots, V_{N,i}$, where $V_{j,i} = (b_{j,i}, c_{j,i})$, $b_{j,i} \in [n^{\delta_1}]$ and $c_{j,i} \in [n^{1+\delta_1+\delta_2}]$. For each j, the pair $V_{j,i}$ is used to verify the correctness of P_j's partial signature.

[3] As pointed out in Section 3.1 we can regard d_i as a single element in $[\phi(n)]$, and carry out the computations accordingly.

4.3 Partial Signatures Verification (Signature Phase)

We show the protocol for verification of a partial signature where there are two players, P and V , each holding the data which they received in ICP-Gen-Integers. The protocol appears in Figure 2. In the context of the Signature Phase this protocol will be carried out by every pair of players. After executing all these invocations of the Non-interactive Verification Protocol, player P_i will take a subset of T shares which he has accepted, and will generate the signature for m.

Non-interactive Verification

Input: Player V : $b \in [n^{\delta_1}]$, $c \in [n^{1+\delta_1+\delta_2}]$
Player P : $d_P \in [\phi(n)], y = bd_P + c$
Both players: message $m \in Z_n^*$, RSA composite n

1. P broadcasts the partial signature $S = m^{d_P} \bmod n$ and the auxiliary value $Y = m^y \bmod n$
2. V checks if $S^b m^c = Y$. If yes, he concludes that $S = \pm m^{d_P} \bmod n$ and accepts it.

Fig. 2. The Non-interactive Verification Protocol

Theorem 5. Non-interactive Verification *Assume that a cheating prover P^* cannot break RSA (in particular, he does not know and cannot compute the factorization of n). Let $n = pq$, where $p < q$, $p = 2p' + 1$, $q = 2q' + 1$, and p, q, p', q' are all prime numbers.*

Completeness. *If P and V follow the protocol then V always accepts the partial signature.*
Soundness. *A cheating prover P^* can convince V to accept $S \neq \pm m^{d_P} \bmod n$, with probability at most $\frac{1}{p'} + \frac{1}{n^{\delta_1}} + \frac{1}{n^{\delta_2}}$.*
Zero-knowledge. *Any (possibly cheating) verifier V^* interacting with prover P does not learn any information beyond the signature $S = m^{d_P} \bmod n$.*

Proof.
Completeness. Immediate.

Soundness. First we shall examine the case where $y \in \mathcal{Y} = [n^{\delta_1+1}, \ldots, n^{1+\delta_1+\delta_2}]$. Notice that the verifier uses a deterministic procedure to accept or reject the published pair S, Y. Therefore the probability stated in the theorem is taken over these coin-tosses of the dealer in ICP-Gen-Integers which are consistent with the view of the prover (i.e. the value y). In order for P to convince V to accept S, Y, it must hold that $Y = S^b m^c \bmod n$. We know from the ICP-Gen-Integers protocol that $y = bd_P + c$, and hence $m^y = (m^{d_P})^b m^c \bmod n$. By dividing these two equations we get that:

$$Ym^{-y} = (Sm^{-d_P})^b \bmod n \qquad (1)$$

This means that Ym^{-y} must be in the subgroup $<Sm^{-d_P}>$. Let k be the minimal value such that $Ym^{-y} = (Sm^{-d_P})^k \bmod n$. Consequently, Equation (1) is satisfied only if: $b = k \bmod ord(Sm^{-d_P})$. Since b is chosen at random with uniform distribution from $[n^{\delta_1}]$, the probability that a pair (S, Y) satisfies Equation 1 is

$$\Pr(\, b = k \bmod ord(Sm^{-d_P})\,) \leq \frac{\left\lceil \frac{n^{\delta_1}}{ord(Sm^{-d_P})} \right\rceil}{n^{\delta_1}} \leq \frac{1}{ord(Sm^{-d_P})} + \frac{1}{n^{\delta_1}}.$$

Because of the special form of n, there are only four elements of Z_n^* whose order is smaller than p', namely the four roots of unity. If $Sm^{-d_P} = \pm 1 \bmod n$ then $S = \pm m^{d_P} \bmod n$. If the prover could find S such that Sm^{-d_P} is a non-trivial root of unity, then he could factor n which we assume to be infeasible. For all other choices of S, order $ord(Sm^{-d_P}) \geq p'$. This completes the proof for the case where $y \in \mathcal{Y}$. However, from Lemma 3 we know that the probability that $y \notin \mathcal{Y}$ is at most $\frac{1}{n^{\delta_2}}$, by combining these two probabilities we get the desired probability.

Zero-knowledge. Values b and c are picked independently from d_P, hence they give out no information on d_P. Furthermore, knowing b, c and $S = m^{d_P} \bmod n$, the verifier can compute $Y = m^y = S^b m^c \bmod n$. □

5 Interactive Robust Threshold RSA

The two components of the interactive protocol are the protocol for dealing verification information during the Dealing Phase (§5.1), and the protocol for verification of partial signatures during the Signature Phase (§5.2). (See the schema in Section 2.)

The basic idea underlying the interactive solution is that it suffices to know a single sample message w and its correct partial signature $w^{d_P} \bmod n$ in order to verify the partial signature of any other message m, under the same key d_P. Our solution is based on a protocol due to Chaum and van Antwerpen [CA90], and further developed in [Cha90, BCDP91], designed to prove in zero-knowledge the equality of the discrete logarithms of two elements over a prime field Z_p relative to two different bases. The protocol and the proof presented in the above papers do not work over Z_n for composite n as required here, in particular, since they strongly rely on the existence of a generator for the multiplicative group Z_p^*. However an adaptation of the protocol, and a more involved proof, can be shown to help solving our problem over Z_n.

5.1 Generation of Verification Data (Dealing Phase)

One of the advantages of our interactive solution relative to the non-interactive protocol presented in Section 4 is that it allows to verify the actions of the dealer during the dealing phase (still the dealer is trusted not to reveal the factorization of n or the private key d). We present the details of verification of the dealer's actions in the Appendix. For simplicity, our following presentation assumes a trusted and honest dealer.

The verification information dealt during the initialization protocol consists of a random public *sample* message w and and its corresponding sample partial signatures w^{d_i}, for each one of the partial keys d_i held by the players. The sample signatures are broadcast to all players (no secrecy required). See Figure 3.

Sample-Signature-Generation

Input: Public: RSA modulus n
Dealer $\mathcal{D}$: key-shares $d_i \in [\phi(n)]$, for $i = 1, 2, \ldots, N$

1. $\mathcal{D}$ chooses a random value w in Z_n^* and broadcasts values $w_i = w^{d_i} \bmod n$ for $i = 1, 2, \ldots, N$.

Fig. 3. The Sample Signature Generation Protocol

In terms of our generic schema in Section 2, the verification data is $V_{i,j} = w_j = w^{d_j} \bmod n$, for all i, j. Notice that unlike in the non-interactive protocol, here the $V_{i,j}$'s are public.

5.2 Verification of Partial Signatures (Signature Phase)

Each player P_i checks the partial signatures produced by each other player P_j. For clarity of presentation, we concentrate on two players only, the prover (or signer) P and the verifier V. The player P has his secret signature key $d_P \in [\phi(n)]$ and both players have access to a publicly known sample message w and its partial signature (under P's key) $w^{d_P} \bmod n$. For any $x \in Z_n^*$ we denote by S_x the corresponding signature of P on x, namely, $S_x = x^{d_P} \bmod n$. By $\hat{S}_x$ we denote the "alleged" signature on x, i.e. a string claimed (but not yet verified) to be the signature of x.

Interactive Verification

Input: Prover: secret $d_P \in [\phi(n)]$
Common: RSA composite n, sample message $w \in Z_n^*$,
signature S_w, message $m \in Z_n^*$, claimed $\hat{S}_m$

1. V chooses $i, j \in_R [n]$ and computes $R \overset{\text{def}}{=} m^i w^j \bmod n$
   $V \longrightarrow P : R$
2. P computes $S_R \overset{\text{def}}{=} R^{d_P} \bmod n$
   $P \longrightarrow V : S_R$
3. V verifies that $S_R = \hat{S}_m^i {S_w}^j \bmod n$.
   If equality holds then V accepts $\hat{S}_m$ as the signature on m, otherwise it rejects.

Fig. 4. The Interactive Verification Protocol

Figure 4 presents the basic interactive verification protocol. This description corresponds to an interactive proof between P and V. The completeness and soundness of the protocol are proved in Theorem 6 below. The protocol as presented is *not* zero-knowledge. (For example, a cheating verifier V^* can choose R in a different way than specified and then learn S_R, which V^* could not compute by himself.) However, there are well-known techniques [GMW86, BCC88, Gol95] to add the zero-knowledge property to the above protocol using the notion of a *commitment function*: Instead of P sending S_R in Step 2, he sends a commitment $commit(S_R)$, after which V reveals to P the values of i and j. After checking that $R = m^i w^j \bmod n$, P sends S_R to V . The verifier checks that S_R corresponds to the value committed by P and then performs the test of Step 3 above.

The zero-knowledge condition is achieved through the properties of the commitment function, namely, (I) $commit(x)$ reveals no information on x, and (II) P cannot find x' such that $commit(x) = commit(x')$. Commitment functions can be implemented in many ways. For example, in the above protocol $commit(S_R)$ can be implemented as a probabilistic RSA encryption of S_R (the encryption is required to be semantically secure, see [GM84, BR94]), using a public key for which the private key is not known to V (and possibly, not even known to P). To open the commitment, P reveals both S_R and the string r used for the probabilistic encryption. This implementation of a commitment function is very efficient as it involves no long exponentiations.

In the following theorem we state the security properties of the complete Interactive Verification protocol.

Theorem 6. Interactive Verification. *Assume that a cheating prover P^* cannot break RSA (in particular, does not know and cannot compute the factorization of n), and that w was chosen at random. Let $n = pq$, where $p < q, p = 2p' + 1, q = 2q' + 1$, and p, q, p', q' are all prime numbers.*

Completeness. *If P and V follow the protocol then V always accepts.*
Soundness. *No cheating prover P^* can convince V to accept $\hat{S}_m \neq \pm m^{d_P} \bmod n$, except for a negligible probability $\frac{O(1)}{p}$.*
Zero-knowledge. *Any (possibly cheating) verifier V^* interacting with prover P does not learn any information beyond the signature $S_m = m^{d_P} \bmod n$.*

The proof of completeness of the protocol is immediate and the zero-knowledge property is argued above. Here we prove the soundness property. The following is the core claim behind the proof of soundness.

Lemma 7. *The prover's cheating probability in the Interactive Verification Protocol is at most* $\frac{ind(w)}{ord(\hat{S}_m/m^{d_P})} + 2^{\frac{n-\phi(n)}{n}}$.

For space limitations we omit the proof of Lemma 7 from these proceedings. We stress that the above lemma holds also for a computationally unbounded cheating prover, and that the bound in the lemma is tight for such a prover (up to the term $2^{\frac{n-\phi(n)}{n}}$). Next, we show how to apply the lemma to prove the soundness of the protocol in the case that n is chosen with the particular form stated in Theorem 6, and the (cheating) prover cannot break RSA.

Proof of Theorem 6 (soundness). The bound in Lemma 7 is given in terms of the order of some elements in the group Z_n^*. Thus, in order to establish the exact bound for the above special form of n we need to study the order of elements in this particular group. There is one element of order 1 in Z_n^* (the unit element), 3 of order 2 (-1 and two other non-trivial roots of 1), $4p' + 4q' - 8$ elements of order ranging between p' and $2q'$, and the rest have all order which is at least $p'q'$. (This can be argued based on the special form of p and q and the order of elements modulo these primes, and then using the Chinese Remainder Theorem). In particular, the order of w, which is chosen at random, is at least $p'q'$, with probability $1 - \frac{4(p'+q')}{\phi(n)} \geq 1 - \frac{2}{p'}$, and then $ind(w)$ which equals $\phi(n)/ord(w)$ is at most 4 (notice that $\phi(n) = 4p'q'$).

As in the non-interactive protocol (see soundness part of theorem 5), a successful cheating of P^* happens when it convinces V to accept a value $\hat{S}_m = bm^{d_P} \bmod n$, for $b \not\equiv \pm 1 \bmod n$. Notice that the prover (who knows d_P) can compute b. This excludes the possibility that b would be one of the non-trivial square roots of 1, since knowledge of such an element would allow the prover to factor n. Therefore, $b = \hat{S}_m/m^{d_P}$ must be of order at least p'. Finally, the expression $\frac{n-\phi(n)}{n}$ is at most $1/p'$ in this case. The corollary then follows by replacing these values in the bound expression $\frac{ind(w)}{ord(\hat{S}_m/m^{d_P})} + 2\frac{n-\phi(n)}{n}$ in Lemma 7. □

6 Conclusions and Further Applications

We presented two protocols for verifying partial signatures. The first protocol is a non-interactive one, the second is interactive, yet provides the ability to have public verification of the partial signatures. Both protocols are low on computation and communication. Thus, achieving an efficient, robust, threshold-RSA signature scheme.

Our techniques are closely related to the notion of undeniable signatures [CA90], and can form the basis for RSA-based undeniable signatures (known undeniable signature schemes are based on discrete logarithm-based systems, not on RSA). Undeniable signatures are characterized by the fact that public information is not sufficient in order to verify the signature but interaction with the signer is required for such verification. The techniques we present can be further applied to separate between the signing and verification processes in the sense that a signer could delegate the ability to verify signatures to a third party while the latter cannot forge signatures.

We mention again the applicability of our results to key-escrow systems in which a user shares its decryption capability with a set of escrow agents ([Mic92, DDFY94]). The techniques for shared RSA signature generation apply to shared RSA decryption as well. Using these mechanisms, a user (acting as the dealer) shares its private decryption key with a set of agents, such that the cooperation of at least a threshold of these agents is required in order to decrypt messages intended for that user; no coalition of less than T agents can decrypt such messages or learn about the user's decryption key. In this application the verifiability of the dealer's actions is particularly important since the latter may have a strong interest to prevent the eventual decryption by the agents of messages intended for him/her.

References

[BCC88] G. Brassard, D. Chaum, and C. Crépeau. Minimum disclosure proofs of knowledge. *JCSS*, 37(2):156–189, 1988.

[BCDP91] J. Boyar, D. Chaum, I. Damgård, and T. Pedersen. Convertible undeniable signatures. In A.J. Menezes and S. A. Vanstone, editors, *Proc. CRYPTO 90*, pages 189–205. Springer-Verlag, 1991. Lecture Notes in Computer Science No. 537.

[BK89] M. Blum and S. Kannan. Program correctness checking and the design of programs that check their work. In *Proc. of the 21st ACM Symposium on Theory of Computing*, 1989.

[Boy86] C. Boyd. Digital Multisignatures. In H. Baker and F. Piper, editors, *Cryptography and Coding*, pages 241–246. Claredon Press, 1986.

[BR94] M. Bellare and P. Rogaway. Optimal asymmetric encryption. In *Eurocrypt'94*, 1994.

[CA90] David Chaum and Hans Van Antwerpen. Undeniable signatures. In G. Brassard, editor, *Proc. CRYPTO 89*, pages 212–217. Springer-Verlag, 1990. Lecture Notes in Computer Science No. 435.

[Cha90] D. Chaum. Zero–knowledge undeniable signatures. In *Proc. EUROCRYPT 90*, pages 458–464. Springer-Verlag, 1990. Lecture Notes in Computer Science No. 473.

[DDFY94] Alfredo De Santis, Yvo Desmedt, Yair Frankel, and Moti Yung. How to share a function securely. In *Proc. 26th ACM Symp. on Theory of Computing*, pages 522–533, Santa Fe, 1994. IEEE.

[Des94] Yvo G. Desmedt. Threshold cryptography. *European Transactions on Telecommunications*, 5(4):449–457, July 1994.

[DF90] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In G. Brassard, editor, *Proc. CRYPTO 89*, pages 307–315. Springer-Verlag, 1990. Lecture Notes in Computer Science No. 435.

[DF92] Y. Desmedt and Y. Frankel. Shared generation of authenticators and signatures. In J. Feigenbaum, editor, *Proc. CRYPTO 91*, pages 457–469. Springer, 1992. Lecture Notes in Computer Science No. 576.

[Fel87] P. Feldman. A Practical Scheme for Non-Interactive Verifiable Secret Sharing. In *Proceeding 28th Annual Symposium on the Foundations of Computer Science*, pages 427–437. IEEE, 1987.

[FGY96] Y. Frankel, P. Gemmell, and M. Yung. Witness-based Cryptographic Program Checking and Robust Function Sharing. To appear in proceedings of STOC96, 1996.

[GJKR96] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust threshold dss signatures. To appear in Eurocrypt'96, 1996.

[GM84] S. Goldwasser and S. Micali. Probabilistic encryption. *JCSS*, 28(2):270–299, April 1984.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Computing*, 17(2):281–308, April 1988.

[GMR89] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems. *SIAM. J. Computing*, 18(1):186–208, February 1989.

[GMW86] O. Goldreich, S. Micali, and A. Wigderson. Proofs that Yield Nothing but the Validity of the Assertion, and a Methodology of Cryptographic Protocol Design. In *Proceeding 27th Annual Symposium on the Foundations of Computer Science*, pages 174–187. ACM, 1986.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to Play Any Mental Game. In *Proceeding 19th Annual Symposium on the Theory of Computing*, pages 218–229. ACM, 1987.

[Gol95] Oded Goldreich. *Foundation of Cryptography – Fragments of a Book*. Electronic Colloquium on Computational Complexity, February 1995. Available online from *http://www.eccc.uni-trier.de/eccc/*.

[Mic92] Silvio Micali. Fair public-key cryptosystems. In Ernest F. Brickell, editor, *Proc. CRYPTO 92*, pages 113–138. Springer-Verlag, 1992. Lecture Notes in Computer Science No. 740.

[Ped91] T. Pedersen. Distributed provers with applications to undeniable signatures. In *Eurocrypt'91*, 1991.

[Rab94] T. Rabin. Robust Sharing of Secrets When the Dealer is Honest or Faulty. *Journal of the ACM*, 41(6):1089–1109, 1994.

[RB89] T. Rabin and M. Ben-Or. Verifiable Secret Sharing and Multiparty Protocols with Honest Majority. In *Proceeding 21st Annual Symposium on the Theory of Computing*, pages 73–85. ACM, 1989.

[Sha79] A. Shamir. How to Share a Secret. *Communications of the ACM*, 22:612–613, 1979.

A Verifying the dealing phase

In this section we show how to verify that the dealer performs correctly the sharing of the keys and of the verification data in the interactive protocol of Section 5. This will allow us to reduce significantly the level of trust on the server that initializes the system. Still we need to trust the dealer not to communicate the factorization of n (or the secret exponent d) to any players. However, the ability to verify that the dealer deals correct and consistent information is a fundamental aspect to guarantee the successful operation of the system during the signature phase. In what follow we sketch the main aspects of the verification protocol. Full details, including the actions of the players in case of detection of dishonest behavior, will appear in the complete version of the paper.

GENERATION OF w. In order to make sure that the sample message w is chosen at random, the players collectively generate it by using some protocol for collective coin toss, thus dispensing of the dealer for this choice.

VERIFICATION OF SHARES AND SAMPLE SIGNATURES. The following is a procedure by which each player can verify the correct dealing of d into the shares $d_1, \ldots, d_N$, and the correct value of the sample signatures $w_i = w^{d_i} \bmod n$. (In the sequel we omit the $\bmod n$ notation.) Verifying the correctness of the shares means that the values d_i all lie in a unique polynomial of degree $T-1$ whose free coefficient is d (the secret exponent of the collective signature). The values w_i are correct if they correspond to the partial signatures w^{d_i} for the verified shares d_i. Let $x_1, \ldots, x_N$ be the values used for polynomial evaluation in the share generation procedure of [DDFY94], and let $x_0 = 0$. Let $a_{ij}, i = 1, 2, \ldots, T,\ \ j = 0, 1, \ldots, N$, be *interpolation coefficients* such that for any polynomial f of degree $T-1$, and for $j = 0, 1 \ldots, N$ it holds that $f(x_j) = \sum_{i=1}^{T} a_{ij} f(x_i)$. Since a set of values $d_1, \ldots, d_N$ is a correct sharing of the value d if and only if there exists a polynomial f of degree $T-1$ such that $d = f(0)$, and $d_j = f(x_j)$, for $j = 1, 2, \ldots, N$, then we get that a correct sharing is verified by the equations:

$$\text{(A.1)} \qquad d = \sum_{i=1}^{T} a_{i0} d_i \quad \text{and} \quad d_j = \sum_{i=1}^{T} a_{ij} d_i, \ \text{ for } j = T+1, \ldots, N.$$

In our case the explicit values of all the shares d_i are not available to each player, therefore the checking of correct dealing is done using the equivalent of the above equations "in the exponent", namely, each player verifies that (remember that $w_i = w^{d_i}$):

$$\text{(A.2)} \qquad w = \prod_{i=1}^{T} (w_i^{a_{i0}})^e \quad \text{and} \quad w_j = \prod_{i=1}^{T} w_i^{a_{ij}}, \ \text{ for } j = T+1, \ldots, N.$$

In order to be able to claim that the verification of (A.2) implies the correctness of (A.1) we need to solve two problems. The first is the fact that there may be a value w_i which is not a power of w, i.e. there is no value t for which $w_i = w^t$. The second problem is that even if the values w_i are all exponents of w, the equality in (A.2) only implies that the equality in (A.1) holds modulo $ord(w)$, which may be a problem if w is an element of low order.

To verify that the values w_i are indeed exponents of w we use the following subprotocol. For each $i \in [N]$ the dealer $\mathcal{D}$ chooses a value $r \in_R [\phi(n)]$ and broadcasts $w' = w^r$. The players collectively choose a random bit b. If $b = 0$, $\mathcal{D}$ broadcasts the value r, otherwise it broadcasts the value $d_i + r \bmod \phi(n)$. In the first case, each player can check whether $w^r = w'$, and in the second, whether $w^{(r+d_i)} = w'w_i$. If $w_i \notin <w>$ then the probability that $\mathcal{D}$ passes this test is $1/2$. By repeating this procedure k times the probability that the dealer can cheat goes down to 2^{-k}.

As for the problem that equalities (A.1) are verified only modulo $ord(w)$, we point out that because of the assumed form of p and q (i.e., $(p-1)/2$ and $(q-1)/2$ being prime numbers), the order of a random element w is equal to $\phi(n)/2$ or $\phi(n)/4$ with overwhelming probability. In the former case, the order of w is a multiple of the order of all other elements in Z_n^* and then (A.2) implies (A.1). In the case $ord(w) = \phi(n)/4$, the element $-w$ is of order $\phi(n)/2$. Therefore, one solution to the above problem is to repeat the described process for both w and $-w$. If the above verification procedure is completed successfully for both values, then only the value of w is carried to the signature generation phase.[4]

VERIFICATION OF THE PRIME FACTORS. We need to check that the dealer chooses the modulus n of the right form, i.e. $n = pq$ with $p = 2p' + 1$ and $q = 2q' + 1$. Although this choice can be theoretically checked using the general results of [GMW87] on secure computation, the resultant solution would be hardly practical. To alleviate this problem one could have the dealer generate a large set of moduli $n_1, n_2, \cdots$ from which the players collectively choose a random element, say n_i. Next, $\mathcal{D}$ shows the factorization into primes of all the other moduli in the set. If all are of the right form then n_i is chosen as the modulus n, otherwise $\mathcal{D}$ is disqualified. Considering that the set of moduli that $\mathcal{D}$ can produce can be of only moderate size (given the high cost of producing such special primes) a cheating dealer will still have a small but non-negligible probability to cheat. (On the other hand, given that the dealing phase is done very rarely one may afford having the dealer produce a significantly large number of the above moduli, thus considerably reducing the cheating probability by the dealer.)

We summarize the properties of the dealing phase in the following lemma.

Lemma 8. *Assume that the composite n is chosen as specified and let $T \leq \lceil N/2 \rceil$. If there are at most $T-1$ cheating players during the above dealing phase, and the dealer is not disqualified, then the good players end that phase with correct partial signatures on w for every non-disqualified player, and the corresponding shares d_i interpolate to the correct exponent d, as chosen by $\mathcal{D}$. Moreover, if the dealer is honest nothing is learned by any of the players that can help a coalition of less than T players to forge a signature.*

[4] Another option is to test only w. If $ord(w) = \phi(n)/2$ then no cheating for $\mathcal{D}$ is possible. If $ord(w) = \phi(n)/4 = p'q'$ then the only possible cheating by $\mathcal{D}$ is to deal instead of the right exponent d, the exponent $d' = d + p'q'$ (or $d' = d + 3p'q'$) which satisfies all equations for w but not for values w' of order $2p'q'$. However, even in this case the equations are satisfied up to their sign (since in this case $w'^{d'} = -w'^{d}$), and as stated in Section 3.1 getting a right signature except for the wrong sign is acceptable in our setting.