Share conversion, pseudorandom secret-sharing and applications to secure distributed computing

Ronald Cramer[1], Ivan Damgård[2⋆], and Yuval Ishai[3⋆⋆]

[1] CWI, Amsterdam and Mathematical Institute, Leiden University (cramer@cwi.nl)
[2] Aarhus University (ivan@daimi.au.dk)
[3] Technion, Haifa (yuvali@cs.technion.ac.il)

Abstract. We present a method for converting shares of a secret into shares of the same secret in a different secret-sharing scheme using only local computation and no communication between players. In particular, shares in a replicated scheme based on a CNF representation of the access structure can be converted into shares from any linear scheme for the same structure.
We show how this can be combined with any pseudorandom function to create, from initially distributed randomness, any number of Shamir secret-sharings of (pseudo)random values without communication. We apply this technique to obtain efficient non-interactive protocols for secure computation of low-degree polynomials, which in turn give rise to other applications in secure computation and threshold cryptography. For instance, we can make the Cramer-Shoup threshold cryptosystem by Canetti and Goldwasser fully non-interactive, or construct non-interactive threshold signature schemes secure without random oracles.
The latter solutions are practical only for a relatively small number of players. However, in our main applications the number of players is typically small, and furthermore it can be argued that no solution that makes a black-box use of a pseudorandom function can be more efficient.

1 Introduction

A secret-sharing scheme enables a dealer to distribute a secret among n players, such that only some predefined qualified subsets of the players can recover the secret from their joint shares and others learn nothing about it. The collection of qualified sets that can reconstruct the secret is called an *access structure*. One useful type of secret-sharing schemes are *threshold schemes*, in which the access structure includes all sets of more than t players, for some threshold t.

Secret-sharing schemes have found numerous applications in cryptography. In most of these applications, one tries to use the "best" known scheme for the access structure at hand. Indeed, in the most popular threshold case, applications

⋆ Supported by BRICS, Basic research in Computer Science, Center of the Danish National Research Foundation and FICS, Foundations in Cryptography and Security, funded by the Danish Natural Sciences Research Council
⋆⋆ Research supported by Israel Science Foundation grant 36/03.

typically rely on Shamir's scheme [28], which is optimal with respect to its share size. It turns out, however, that there are contexts where it is not desirable, or even not at all possible, to use the most succinct available secret-sharing scheme. (Some examples will be provided below.) In such contexts it may be beneficial to share a secret using one secret-sharing scheme and later convert its shares to a different representation, corresponding to another secret-sharing scheme, enjoying the relative advantages of both schemes.

Non-interactive share conversion. Motivated by this general scenario, as well as by the more concrete applications discussed below, we introduce and study the following notion of local conversion between secret-sharing schemes. For secret sharing schemes $\mathcal{S}, \mathcal{S}'$, we say that $\mathcal{S}$ is *locally convertible* to $\mathcal{S}'$, if any valid $\mathcal{S}$-sharing of a secret s may be converted by means of local transformations (performed by each player separately) to valid, though not necessarily random, $\mathcal{S}'$-sharing of the same secret s. Before describing our results on share conversion and their applications, we turn to describe a special class of secret-sharing schemes that play a central role in these results.

Replicated secret-sharing. A very useful type of "inefficient" secret-sharing scheme is the so-called *replicated scheme* [23].[1] The replicated scheme for an access structure Γ proceeds as follows. First, the dealer splits the secret s into additive shares, where each additive share corresponds to some maximal *unqualified* set $T \notin \Gamma$. That is, we view s as an element of some finite field K, and write $s = \sum_{T \in \mathcal{T}} r_T$, where $\mathcal{T}$ is the collection of all maximal unqualified sets, and where the additive shares r_T are random subject to the restriction that they add up to s. Then, the dealer distributes to each player P_j all additive shares r_T such that $j \notin T$.

Pseudorandom secret-sharing. In the threshold case, the replicated scheme involves $\binom{n}{t}$ additive shares and is thus far worse than Shamir's scheme in terms of share size. However, it enjoys the following key property: shares of a *random* secret $s \in K$ consist of replicated instances of random and *independent* elements from K. This property comes handy in applications which require a large number of (pseudo-)random secret-shared values: viewing each replicated share r_T as an independent key to a pseudorandom function, we may get a virtually unlimited supply of independent pseudorandom secrets, each shared using the replicated scheme. Thus, we may use this method to obtain replication-shared secrets at a very low amortized cost. The main difficulty is that these shared secrets cannot be securely used in a higher level application without paying the $\binom{n}{t}$ communication overhead for each secret being used. The goal of share conversion, in this case, would be to locally convert replicated shares of each shared secret used by the application into an equivalent Shamir representation. This would allow to enjoy the best of both worlds, combining the share independence advantage of the replicated scheme with the succinctness advantage of Shamir's scheme.

[1] This scheme can also be obtained from the formula-based construction of [6] by using a CNF representation of the access structure. Hence, it is sometimes referred to in the literature as a CNF-based scheme.

1.1 Our Results

Our contribution goes in two directions. First, we put forward the notion of share conversion and obtain some results on the possibility and efficiency of share conversion. Second, we present various applications of our share conversion results, mainly within the domains of multiparty computation and threshold cryptography. We now provide a more detailed account of these results.

Results on share conversion. Our main result is that shares from the replicated scheme described above can be locally converted into shares of any *linear* scheme[2] for the same (or smaller) access structure. In particular, shares from the replicated scheme for a threshold structure can be converted into Shamir-shares for the same structure. We start by describing a simple conversion procedure for the latter special case, and then generalize it to arbitrary linear schemes and access structures. The general conversion result relies on a representation of the access structure by a *canonical span program* [24].

The share convertibility relation induces a partial order on secret-sharing schemes. Under this order, the replicated scheme is *maximal* in the class of all linear schemes realizing a given access structure. We also identify a minimal scheme in this class, and prove some negative results regarding the possibility and efficiency of share conversion. In particular, we show that the $\binom{n}{t}$ overhead cannot be avoided when converting replicated shares to Shamir shares.

Applications. As discussed above, share conversion can be combined with any pseudorandom function to securely create, from initially distributed randomness, a virtually unlimited supply of Shamir secret sharings of (pseudo)random values without further interaction. We present several applications of this idea in a setting where the cost of pre-distributing $\binom{n}{t}$ keys can be afforded.

Distributed PRFs. We obtain a communication-efficient variant of a distributed PRF construction of Naor, Pinkas, and Reingold [27] by converting its replicated shares to Shamir shares. A natural application of distributed PRFs is to distributing a key-distribution center.

Secure multiparty computation. We present efficient protocols for securely evaluating low-degree polynomials, requiring only two rounds of interaction: a round of broadcast messages followed by a round of point-to-point messages. (If no setup is allowed, then this is provably impossible [19].) Using known techniques, these results for low-degree polynomials can be extended to general functions. In the case of functions which only take a random input (e.g., a function dealing cards to poker players) the first round of broadcasts can be eliminated. Thus, we get efficient and fully non-interactive protocols for distributing a trusted dealer in a wide array of applications.

[2] In a linear secret-sharing scheme the secret is taken from a finite field, and each player's share is obtained by computing some linear function of the secret and the dealer's randomness. Essentially all known useful schemes are linear.

Threshold cryptography. The above results on multiparty computation can be specialized to obtain non-interactive implementations of threshold cryptosystems, taking advantage of their simple algebraic structure. For instance, we show how to make the Cramer-Shoup threshold cryptosystem by Canetti and Goldwasser [11] fully non-interactive and how to construct non-interactive threshold signature schemes secure without random oracles.

Towards assessing the practicality of these solutions, we note that many of them, in particular the threshold cryptography applications, are designed for a client-server model, where we would typically have a small number of quite powerful servers and some number of (possibly less powerful) clients. Even the application to general multiparty computation can be naturally set in a client-server model, for instance when a large number of players provide input to a computation, but where it is not practical to have them all participate in the computation – this is then left to a small number of servers. Our solutions fit nicely into such a scenario, since they only require the servers to handle the $\binom{n}{t}$ overhead locally, and since here n is the number of servers which is typically small.

It is important to understand that all our applications could have been realized with the same functionality without using share conversion. Instead, the players would work directly on the replicated shares. For instance, this is exactly what was done for distributed PRF's in [27] and could be done for threshold cryptography and multiparty computation by adapting Maurer's techniques [25] in a straightforward way. However, such solutions would be much less practical. First, the $\binom{n}{t}$ overhead would now also apply to the communication, and hence also to the local computation of the clients. Second, various possibilities for optimization would be lost. For instance, the threshold cryptography applications typically require servers to use shares of fresh (pseudo)random values every time they are called by a client. Using share conversion, many sets of such (Shamir) shares can be generated off-line and stored compactly, making the on-line work and storage of servers efficient in n and t. Without share conversion, the $\binom{n}{t}$ overhead would apply to the entire storage generated off-line, destroying the on-line advantage. Finally, share conversion also yields significant savings in the local computation performed by the *servers*. Without share conversion, the servers' computation in our applications would increase (roughly) by a factor of either $\binom{n}{t}$ or $\binom{n}{t}^2$.

Conceptual contribution. All of the above applications are related to the use of replicated secret-sharing in conjunction with pseudorandomness. But there are also other types of applications which seem to benefit from the use of replicated secret-sharing in different, and sometimes unexpected, ways. For instance, replicated shares yield the most efficient information-theoretic private information retrieval protocols and locally decodable codes [5], the best round complexity for verifiable secret-sharing [18], and the simplest protocols for secure multiparty computation with respect to generalized adversaries [3, 25]. Our results on share conversion provide an explanation for the usefulness of the replicated scheme, suggesting that anything that can be achieved using linear secret-sharing can also be achieved (up to an $\binom{n}{t}$ overhead) using this specific scheme. This may also serve as a useful guideline for the design of cryptographic protocols (e.g., when attempting to improve [5]).

Related Work. The idea of distributing pseudorandom functions by replicating independent keys has been previously used by Micali and Sidney [26] and by Naor et al. [27]. However, without the tool of share conversion, their protocols are either very expensive in communication or lose some of their appealing features. We note that an alternative number-theoretic construction of distributed PRFs, suggested in [27], is not suitable for our applications due to the "multiplicative" representation of the output.

Most relevant to the current work is the work on compressing cryptographic resources by Gilboa and Ishai [20]. The problem considered there is that of using replicated pseudorandom sources to securely "compress" useful correlation patterns. In particular, a conversion from replicated shares of a random secret to Shamir shares of a random secret (though not necessarily the same secret) is implicit in their results. The results of [20] do not explicitly refer to the access structure associated with a given correlation pattern, and do not imply our general results on share conversion.

2 Preliminaries

We define an n-player secret sharing scheme by a tuple $\mathcal{S} = (K, (S_1, \ldots, S_n), R, D)$, where K is a finite *secret-domain* (typically a finite field), each S_j is a finite *share domain* from which P_j's *share* is picked (typically $S_j = K^{a_j}$ for some a_j), R is a probability distribution from which the dealer's random input is picked, and D is a share distribution function mapping a secret s and a random input r to an n-tuple of shares from $S_1 \times \cdots \times S_n$. We say that $\mathcal{S}$ *realizes* an access structure $\Gamma \subseteq 2^{[n]}$ if it satisfies the following.

- Correctness: For any qualified set $Q = \{j_1, \ldots, j_m\} \in \Gamma$ there exists a *reconstruction function* $\mathsf{rec}_Q : S_{j_1} \times \cdots \times S_{j_m} \to K$ such that for every secret $s \in K$, $\Pr[\mathsf{rec}_Q(D(s, R)_Q) = s] = 1$, where $D(s, R)_Q$ denotes a restriction of $D(s, R)$ to its Q-entries.
- Privacy: for any unqualified set $U \notin \Gamma$ and secrets $s, s' \in K$ the random variables $D(s, R)_U$ and $D(s', R)_U$ are identically distributed.

In a *linear* secret-sharing scheme (LSSS) the secret-domain K is a finite field, and the randomness R is a uniformly random m-tuple $(r_1, \ldots, r_m) \in K^m$. The share distribution function D is a linear function of $s, r_1, \ldots, r_m$.

In this work we will refer to the following specific LSSS:

1. Shamir's secret-sharing [28]. Let K be a finite field such that $|K| > n$. Each player P_j is assigned a unique non-zero element from K, which we denote j (by abuse of notation if K is not a prime field). In the *t-private Shamir scheme*, the dealer picks t random and independent field elements $r_1, \ldots, r_t$, which define the univariate polynomial $f(y) = s + r_1 y + r_2 y^2 + \ldots + r_t y^t$, and distributes to each player P_j the share $s_j = f(j)$.

2. Replicated secret-sharing [23]. Let $\Gamma \subseteq 2^{[n]}$ be a (monotone) access structure, and let $\mathcal{T}$ include all *maximal* unqualified sets of Γ. The *replicated scheme* for Γ, denoted $\mathcal{R}_\Gamma$, proceeds as follows. To share a secret $s \in K$ the dealer first *additively* share s into $|\mathcal{T}|$ shares, each labelled by a different set from $\mathcal{T}$; that is, it lets $s = \sum_{T \in \mathcal{T}} r_T$ where the shares r_T are otherwise-random field elements. Then, the dealer distributes to each player P_j all shares r_T such that $j \notin T$; that is, P_j's share vector is $(r_T)_{T \not\ni j}$. Privacy follows from the fact that members of every maximal unqualified set $T \in \mathcal{T}$ jointly miss exactly one additive share, namely the share r_T (hence members of any unqualified set miss *at least* one share). On the other hand, since Γ is monotone, a qualified set $Q \in \Gamma$ cannot be contained in any unqualified set; hence, members of Q jointly view all shares r_T and can thus reconstruct s.
3. DNF-based secret-sharing [23]. In the *DNF-based scheme*, the secret is additively shared between the members of each minimal qualified set, where each additive sharing uses independent randomness. This scheme can be obtained by applying the construction of [6] to the monotone DNF representation of Γ.

In the case of threshold access structures, the latter two schemes may be practical only in contexts where $\binom{n}{t}$ is not too large. Their asymptotic complexity is polynomial in n when $t = O(1)$ or $n - t = O(1)$.

3 Share Conversion

In this section we present our main results on local share conversion. We start by defining this notion, which induces a partial order on secret-sharing schemes.

Definition 1 (Share conversion). *Let $\mathcal{S}, \mathcal{S}'$ be two secret-sharing scheme over the same secret-domain K. We say that $\mathcal{S}$ is* locally convertible *to $\mathcal{S}'$ if there exist local conversion functions $g_1, \ldots, g_n$ such that the following holds. If $(s_1, \ldots, s_n)$ are valid shares of a secret s in $\mathcal{S}$ (i.e., $\Pr[D(s,R) = (s_1, \ldots, s_n)] > 0$), then $(g_1(s_1), \ldots, g_n(s_n))$ are valid shares of the same secret s in $\mathcal{S}'$. We denote by g the concatenation of all g_i, namely $g(s_1, \ldots, s_n) = (g_1(s_1), \ldots, g_n(s_n))$, and refer to g as a* share conversion function.

Note that the above definition does not require that random shares of a secret s in $\mathcal{S}$ will be converted into random shares of s in $\mathcal{S}'$. However, due to the locality feature of the conversion, converted shares cannot reveal more information about s than the original shares. Moreover, in typical applications of our technique the converted shares $\mathcal{S}'$ will indeed be random.

3.1 From Replicated Shares to Shamir

We first address the important special case of threshold structures. Suppose that a secret s has been shared according to the t-private replicated scheme. Thus, we may write:

$$s = \sum_{A \subseteq [n] \,:\, |A| = n-t} r_A$$

where r_A has been given to all players in A.

To locally convert these shares into shares of s according to the t-private Shamir scheme, we assign to player P_i the point i in the field. Now, for each set $A \subseteq [n]$ of cardinality $n-t$, let f_A be the (unique) degree-t polynomial such that:

1. $f_A(0) = 1$ and
2. $f_A(i) = 0$ for all $i \in [n] \setminus A$.

Each player P_j can compute a share s_j as follows:

$$s_j = \sum_{A \subseteq [n] \,:\, |A|=n-t, j \in A} r_A \cdot f_A(j).$$

We claim that this results in a set of shares from Shamir's scheme, consistent with the original secret s. To see this, define a polynomial

$$f = \sum_{A \subseteq [n] \,:\, |A|=n-t} r_A \cdot f_A.$$

Clearly, f has degree (at most) t, and it is straightforward to verify that condition 1 above on the f_A's implies $f(0) = s$ and condition 2 implies $f(j) = s_j$.

3.2 Conversion in General

We now generalize the previous conversion result to non-threshold structures. Specifically, we show that shares of the replicated scheme for an arbitrary access structure Γ can be locally converted into shares of any other LSSS for Γ (in fact, even for any $\Gamma' \subset \Gamma$).

To this end, it will be useful to rely on a representation of LSSS via *span programs*, a linear algebra based model of computation introduced by Karchmer and Wigderson [24]. A span program over the variables $\{x_1, \ldots, x_n\}$ assigns to each literal x_i or $\bar{x}_i$ some subspace of a linear space V. The span program *accepts* an assignment $z \in \{0,1\}^n$ if the n subspaces assigned to the satisfied literals span some fixed nonzero vector in V, referred to as the *target vector*. We will be interested in the monotone version of this model, formalized below.

Definition 2 (MSP). *A* monotone span program *(MSP) is a triple $\mathcal{M} = (K, M, \rho)$, where K is a finite field, M is an $a \times b$ matrix over K, and $\rho : [a] \to [n]$ labels the rows of M by player indices. The* size *of $\mathcal{M}$ is the number of rows a. For any set $A \subseteq [n]$ let M_A denote the submatrix obtained by restricting M to its rows with labels from A (and similarly for any other matrix with a rows). We say that $\mathcal{M}$* accepts *A if the rows of M_A* span *the all-ones vector* $\mathbf{1}$. *We denote by $\Gamma_{\mathcal{M}}$ the collection of all sets in $2^{[n]}$ that are accepted by $\mathcal{M}$, and by $\mathcal{T}_{\mathcal{M}}$ the collection of* maximal *sets* not *accepted by $\mathcal{M}$.*

Note that for any MSP $\mathcal{M}$, the structure $\Gamma_{\mathcal{M}}$ is monotone. We also note that any nonzero vector could have been used as a target vector; however, the specific choice of $\mathbf{1}$ will be convenient in what follows. We now associate with any MSP $\mathcal{M}$ a corresponding LSSS in which the total number of field elements distributed by the dealer is equal to the size of $\mathcal{M}$.

Definition 3 (LSSS induced by MSP). *Let $\mathcal{M} = (K, M, \rho)$ be an MSP, where M is an $a \times b$ matrix. The* LSSS induced by $\mathcal{M}$, *denoted by $\mathcal{S}_{\mathcal{M}}$, proceeds as follows. To share a secret $s \in K$:*

- *Additively share s into $\mathbf{r} = (r_1, \ldots, r_b)$.*
- *Evaluate $\mathbf{s} = M\mathbf{r}$, and distribute to each player P_j the entries $\mathbf{s}_{\{j\}}$ (i.e., those corresponding to rows labelled by j).*

It is easy to verify that the induced scheme $\mathcal{S}_{\mathcal{M}}$ is indeed linear and, in fact, that any LSSS is induced by some corresponding MSP $\mathcal{M}$. Finally, the following claim from [24] establishes the expected link between the MSP semantics and the secret-sharing semantics.

Claim. [24] The scheme $\mathcal{S}_{\mathcal{M}}$ realizes the access structure $\Gamma_{\mathcal{M}}$.

Towards proving the main result of this section, it will be convenient to use the notion of *canonic span programs*, introduced in [24]. We use the following monotone version of their construction.

Definition 4 (Canonic MSP). *Let $\mathcal{M} = (K, M, \rho)$ be an MSP, where M is an $a \times b$ matrix. We define a* canonic MSP $\hat{\mathcal{M}} = (K, \hat{M}, \rho)$ *as follows. $\hat{\mathcal{M}}$ has the same size and row labeling as $\mathcal{M}$, but possibly a different number of columns. Let $\mathcal{T} = \mathcal{T}_{\mathcal{M}}$ be the collection of maximal unqualified sets of $\Gamma_{\mathcal{M}}$. For every $T \in \mathcal{T}$, let $\mathbf{w}^T$ be a length-b column vector satisfying $M_T \cdot \mathbf{w}^T = 0$ and $\mathbf{1} \cdot \mathbf{w}^T = 1$.[3] For each maximal unqualified set $T \in \mathcal{T}$, the matrix $\hat{M}$ will include a corresponding column $\mathbf{c}^T \stackrel{def}{=} M \cdot \mathbf{w}^T$ (so that altogether $\hat{M}$ has as many columns as sets in $\mathcal{T}_{\mathcal{M}}$).*

It can be shown that $\Gamma_{\hat{\mathcal{M}}} = \Gamma_{\mathcal{M}}$ [24]. (This can also be derived as a corollary of the next two lemmas.) The scheme $\mathcal{S}_{\hat{\mathcal{M}}}$ induced by the canonic program $\hat{\mathcal{M}}$ may be viewed as a randomness-inefficient implementation of $\mathcal{S}_{\mathcal{M}}$. We will use $\mathcal{S}_{\hat{\mathcal{M}}}$ as an intermediate scheme in the process of converting shares of the replicated scheme for $\Gamma_{\mathcal{M}}$ into shares of $\mathcal{S}_{\mathcal{M}}$.

Lemma 1. *The scheme $\mathcal{S}_{\hat{\mathcal{M}}}$ is locally convertible to $\mathcal{S}_{\mathcal{M}}$ via the identity function $g(\mathbf{s}) = \mathbf{s}$.*

Proof. We need to show that any valid shares in $\mathcal{S}_{\hat{\mathcal{M}}}$ could have also been obtained in $\mathcal{S}_{\mathcal{M}}$ under the same secret s. Let $\hat{\mathbf{r}} \in K^b$ be some additive sharing of $s = \mathbf{1} \cdot \hat{\mathbf{r}}$ induced by the dealer's randomness in $\mathcal{S}_{\hat{\mathcal{M}}}$. Let $\mathbf{r} = W\hat{\mathbf{r}}$ where W is a concatenation of all column vectors $\mathbf{w}^T$ in the order used for constructing $\hat{M}$. By the construction of $\hat{M}$ we have $\hat{M} = MW$ and so $\hat{M}\hat{\mathbf{r}} = MW\hat{\mathbf{r}} = M\mathbf{r}$. Thus, $\mathbf{r}$ produces the same shares in $\mathcal{S}_{\mathcal{M}}$ as $\hat{\mathbf{r}}$ produces in $\mathcal{S}_{\hat{\mathcal{M}}}$. Finally, since every $\mathbf{w}^T$ must satisfy $\mathbf{1} \cdot \mathbf{w}^T = 1$, we have $\mathbf{1} \cdot \mathbf{r} = \mathbf{1} \cdot W\hat{\mathbf{r}} = \mathbf{1} \cdot \hat{\mathbf{r}}$, and thus $\mathbf{r}$ is consistent with the same secret s. □

[3] The existence of such $\mathbf{w}^T$ may be argued as follows: Since $\mathcal{M}$ does not accept T, the linear system $(M_T)^{\mathrm{T}} \cdot \mathbf{x} = \mathbf{1}$ has no solution (where $(M_T)^{\mathrm{T}}$ is the transpose of M_T); hence there must be a way to linearly combine the equations so that a contradiction of the form $\mathbf{0} \cdot \mathbf{x} = 1$ is obtained.

Lemma 2. *Let $\mathcal{R}_\Gamma$ be the replicated scheme realizing Γ over a finite field K, $\mathcal{M}' = (K, M', \rho')$ an MSP such that $\Gamma' \stackrel{\text{def}}{=} \Gamma_{\mathcal{M}'}$ satisfies $\Gamma' \subseteq \Gamma$, and $\hat{\mathcal{M}}' = (K, \hat{M}', \rho')$ a canonic MSP of $\mathcal{M}'$. Then, $\mathcal{R}_\Gamma$ is locally convertible to $\mathcal{S}_{\hat{\mathcal{M}}'}$.*

Proof. Suppose first that $\Gamma' = \Gamma$. Let $\mathcal{T}$ be the collection of maximal unqualified sets of Γ. The $\mathcal{R}_\Gamma$-shares viewed by a player P_i are $s_i \stackrel{\text{def}}{=} (r_T)_{T \not\ni i}$, where $\mathbf{r}$ is an additive sharing of the secret s. Define the i-th local conversion function to be

$$g_i(s_i) = \sum_{T \not\ni i} r_T \cdot \mathbf{c}^T_{\{i\}}.$$

Since each column $\mathbf{c}^T$ of $\hat{M}'$ has only zeros in its T-entries, the above functions g_i jointly define the conversion function g which maps $\mathcal{R}_\Gamma$-shares $\mathbf{s}$, obtained by replicating additive shares $\mathbf{r} = (r_T)_{T \in \mathcal{T}}$, into the $\mathcal{S}_{\hat{\mathcal{M}}'}$-shares $\mathbf{s}' = \hat{M}'\mathbf{r}$. The correctness of this conversion is witnessed by letting $\mathbf{r}' = \mathbf{r}$, namely the same additive sharing of s producing $\mathbf{s}$ in $\mathcal{R}_\Gamma$ will also produce $g(\mathbf{s})$ in $\mathcal{S}_{\hat{\mathcal{M}}'}$.

The general case, where Γ' may be a proper subset of Γ, is only slightly more involved. Let $\mathcal{T}'$ denote the maximal unqualified sets in Γ', and assign to each $T \in \mathcal{T}$ some set $T' \in \mathcal{T}'$ containing it. For each $T' \in \mathcal{T}'$, define $r_{T'}$ to be the sum of all r_T such that T is assigned to T' (or 0 if there is no T assigned to T'). Then, the local conversion functions may be defined by $g_i(s_i) = \sum_{T' \not\ni i} r_{T'} \mathbf{c}^{T'}_{\{i\}}$, and the correctness of the induced conversion g is witnessed by letting $\mathbf{r}' = (r_{T'})_{T' \in \mathcal{T}'}$. □

As a direct corollary of the last two lemmas (and using the transitivity of local conversions) we get the main result of this section:

Theorem 1. *The replicated scheme $\mathcal{R}_\Gamma$, realizing Γ over a field K, is locally convertible to any LSSS over K realizing an access structure $\Gamma' \subseteq \Gamma$.*

The above proof in fact provides a *constructive* way for defining the local conversion function from $\mathcal{R}_\Gamma$ to any LSSS $\mathcal{S}$ realizing Γ (or a subset of Γ), given an MSP for $\mathcal{S}$.

Theorem 1 shows that the (CNF-based) replicated scheme $\mathcal{R}_\Gamma$ is maximal with respect to the convertibility relation among all LSSS realizing Γ. Turning to the other extreme, we now argue that the DNF-based scheme (defined in Section 2) is minimal with respect to convertibility.

Theorem 2. *Any LSSS realizing Γ is convertible to the DNF-based scheme for Γ.*

Proof sketch. Suppose that s has been shared according to some LSSS $\mathcal{S}$ for Γ. We need to show that each minimal qualified set $Q \in \Gamma$ can locally compute an additive sharing of s. This easily follows from the linearity of the reconstruction function rec_Q. □

3.3 Negative Results for Share Conversion

We now show some negative results related to the possibility and efficiency of share conversion. We start by showing that the convertibility relation is non-trivial, in the sense that not all schemes realizing the same access structure are convertible to each other. In fact, we show that Shamir shares cannot be generally converted to replicated shares.

Claim. Let $\mathcal{S}$ be the 1-private 3-player Shamir scheme over a field K ($|K| > 3$) and $\mathcal{S}'$ be the replicated scheme with the same parameters. Then $\mathcal{S}$ is not locally convertible to $\mathcal{S}'$.

Proof. By the correctness requirement, the value of g on any valid 3-tuple (s_1, s_2, s_3) of $\mathcal{S}$-shares must take the form $g(s_1, s_2, s_3) = ((r'_2, r'_3), (r'_1, r'_3), (r'_1, r'_2))$. We now use the locality requirement to show that g must be a constant function, contradicting the correctness requirement. Suppose that one of the local functions g_i is non-constant. Assume wlog that $g_1(0) \neq g_1(1)$ and that they differ in their first output r'_2. Then, either $g(0, s_2, 0)$ outputs illegal $\mathcal{S}'$-shares for all $s_2 \in K$ or $g(1, s_2, 0)$ outputs illegal $\mathcal{S}'$-shares for all $s_2 \in K$ (since in either of these cases the first share of P_1 is different from the second share of P_3). Since there exist both valid $\mathcal{S}$-shares of the form $(0, s_2, 0)$ and of the form $(1, s_2, 0)$, we obtain the desired contradiction. □

Motivated by the following applications, it is natural to ask whether one can reduce the amount of replication in the replicated scheme $\mathcal{R}_\Gamma$ and still allow to convert its shares to other useful LSSS for Γ. Specifically, let $\mathcal{S}$ be a secret-sharing scheme for Γ with the property that a qualified set of players can reconstruct not only the secret, but also the shares of all players. Note that Shamir's scheme enjoys this property. We show that the scheme $\mathcal{R}_\Gamma$ cannot be replaced by a more efficient replicated scheme which is still convertible to $\mathcal{S}$ and at the same time is private with respect to all unqualified sets of $\mathcal{S}$.

Definition 5. *A generic conversion scheme from replicated shares to $\mathcal{S}$ consists of a set of independently distributed random variables $R_1, \ldots, R_m$, an assignment of a subset B_j of these to each player P_j, and local conversion functions g_j such that if each P_j applies g_j to the variables in B_j, we obtain values $(s_1, ..., s_n)$ forming consistent $\mathcal{S}$-shares of some secret s. Furthermore, given the information accessible to any unqualified set of Γ, the uncertainty of s is non-zero.*

Note that neither $\mathcal{S}$ nor the conversion functions g_j are assumed to be linear. Also note that the convertibility requirement formulated above is weaker than our default requirement. However, we are about to show a negative result which is only made stronger this way.

Proposition 1. *For any generic conversion scheme for $\mathcal{S}$ as defined above, it holds that m is at least the number of maximal unqualified sets.*

Proof. Fix any maximal unqualified set T, and let B_T be the set of R_i's known to T. We may assume that for each $R_i \in B_j$, it is the case that $H(s_j|B_j \setminus R_i) > 0$, i.e., R_i is necessary for s_j. If there was not the case, we could remove R_i from B_j and get a more efficient scheme. For each player $P_j \not{i}nT$, we let $C_{j,T} = B_j \setminus T$, thus representing the information available to P_j but not to T. Each such set must be non-empty, otherwise T could determine the value of s.

Now, the set $T \cup P_j$ is qualified, and hence, for any other $P_i \notin T$, it is the case that the share $s_i = g_i(B_i)$ is uniquely determined from $B_T \cup B_j$ – by assumption on $\mathcal{S}$. It follows that $B_i \subset B_T \cup B_j$ and therefore that $C_{i,T} \subset C_{j,T}$. If this was not the case, then by independence of the R_i's, s_i would not be uniquely determined from $B_T \cup B_j$. Since this argument works for any $P_j \notin T$, it follows that in fact $C_{i,T} = C_{j,T}$, so we call this set C_T for short.

Now, consider a different maximal unqualified set T'. We will be done if we show that $C_T \cap C_{T'} = \emptyset$, since this and each C_T being non-empty means that there must be as many R_i's as there are sets T.

So assume some $R_i \in C_T \cap C_{T'}$, and consider a player P_j who is in $T' \setminus T$. This means that P_j knows all variables in C_T, in particular also R_i, but this is a contradiction since R_i is also in $C_{T'}$ and $B_{T'} \cap C_{T'} = \emptyset$ by construction. □

4 Applications

The ability to convert replicated shares to Shamir shares allows to create, from initially distributed randomness, any number of Shamir secret sharings of (pseudo) random values without communication.[4] In this section we present several applications of this idea.

We begin by describing some useful sub-protocols that are common to most of these applications. The first protocol provides precisely the functionality described above: secure generation of (pseudo)random Shamir-shared secrets without communication. Recall the share conversion procedure described in Section 3.1. A secret s has been shared according to the t-private replicated scheme, namely $s = \sum_{A \subseteq [n]\,:\,|A|=n-t} r_A$ where r_A has been given to all players in A. To locally convert these shares into Shamir shares, each player P_j computes its share as $s_j = \sum_{|A|=n-t, j \in A} r_A \cdot f_A(j)$, where f_A is a degree-t polynomial determined by A.

The main observation is that when the secret s is random, all replicated shares r_A will be random and independent. Hence we may use the initially distributed r_A as keys to a PRF $\psi.(\cdot)$, and as long as players agree on a common input a to the function, all players in A can compute $\psi_{r_A}(a)$ and use it in the above construction in place of r_A. Concretely, we get the following.

Protocol Pseudorandom Secret-Sharing (PRSS)
Common inputs: a value a and independent keys $\{r_A\}$ that have been predis-

[4] While we focus the attention on Shamir-based schemes for threshold access structures, the results of this section can be extended to linear schemes realizing general access structures.

tributed as above. Each player P_j computes his share s_j as:

$$s_j = \sum_{A \subseteq [n] \,:\, |A|=n-t, j \in A} \psi_{r_A}(a) \cdot f_A(j) \qquad (1)$$

Note that if we choose K to be of characteristic 2, we can modify the PRSS protocol so that the shared value is guaranteed to be 0 or 1 by simply using a PRF that always outputs 0 or 1. We call this Binary Pseudorandom Secret-Sharing (BPRSS). Assuming that $t < n/3$ it is easy to turn this into a non-interactive verifiable secret-sharing scheme, in a model where a broadcast primitive is available: we simply arrange it such that a Dealer knows all the involved keys. This allows him to compute the pseudorandom shared value and correct it into the value he wants to share:

Protocol Non-Interactive Verifiable Secret-Sharing (NIVSS)
Common inputs: a value a and keys $\{r_A\}$ as above. A dealer D holds all keys as well as an input value $v \in K$. Each player P_j computes a preliminary share $\tilde{s}_i$ as in Eq. 1. Using his knowledge of the keys, D computes the secret s determined by the preliminary shares. D then broadcasts $(v - s)$. Each P_j computes his share as $\tilde{s}_j + (v - s)$.

It is straightforward to verify that this creates a valid Shamir sharing of v if D is honest, and will create a valid sharing of some value no matter how D acts. Furthermore, since $t < n/3$, this value can be reconstructed using standard error correction techniques as long as at most t of the shares are wrong. Finally, for the privacy, we have the following.

Lemma 3. *Consider an adversary Adv that corrupts up to t of the players, but not D. Adv may invoke the protocol NIVSS multiple times, (adaptively) choosing a secret v_j and a distinct evaluation point a_j at each invocation. The adversary gets to see the executions of NIVSS where in the j-th invocation a_j is used as the common input and either (case 0) v_j or (case 1) a random independent value is given as input to D. Assuming the underlying PRF is secure, cases 0 and 1 are computationally indistinguishable.*

Proof. Assume that some Adv can distinguish case 0 and 1, and make the (worst case) assumption that Adv corrupts t players. This means that only one key r_A is unknown to Adv, where A consists of the $n - t$ uncorrupted players. We build an algorithm Alg that breaks the PRF. It gets oracle access to either $\psi_{r_A}()$ or a random oracle and must tell the two apart. Alg gets the inputs a_j, v_j from Adv and at each invocation simply invokes the NIVSS protocol on these inputs, except that it calls the oracle whenever it needs to compute $\psi_{r_A}()$. It is now straightforward to verify that if Alg's oracle is random, Adv will see an exact emulation of case 1: in this case the value of s computed at each invocation will be uniformly random and independent of previous values (by uniqueness of a_j) and so will $v_j - s_j$. On the other hand, if Alg talks to $\psi_{r_A}()$ we emulate exactly case 0. Thus *Adv*'s ability to distinguish case 0 and 1 translates to breaking the PRF with the same advantage. □

It is easy to adapt the NIVSS protocol such that we are guaranteed that the shared value is 0 or 1, along the same lines as the BPRSS protocol. We will refer to this as BNIVSS. Note also that if we just want to create a shared random value known to the dealer, we can simply omit the broadcast step and use the preliminary shares as final shares. We will refer to this as *NIVSS without broadcast*.

The technique for pseudo-random secret-sharing can be generalized to create sharings of a particular value, such as zero. We explain how this is done for the same threshold t access structure as before, but for polynomials of degree $2t$, since this is what we need in the following. Generalizations to other degrees follow easily. Consider a set A of size $n-t$ and consider the set of polynomials

$$F_A = \{f |\ deg(f) \leq 2t, f(0) = 0, j \notin A \Rightarrow f(j) = 0\}.$$

If we think of the set of all degree-$(2t)$ polynomials as a vector space over K, it is easy to see that F_A is a subspace of dimension $2t+1-t-1=t$. So we choose for each A, once and for all, a basis for F_A consisting of t polynomials $f_A^1, ..., f_A^t$. Finally, we distribute initially t keys $r_A^1, ..., r_A^t$ to every player in A. This leads to the following protocol:

Protocol Pseudorandom Zero-Sharing (PRZS)
Common input: a value a, keys $\{r_A^i | i = 1..t, |A| = n-1\}$ that have been predistributed as above. Each player P_j computes his share s_j as:

$$s_j = \sum_{A \subseteq [n]\ :\ |A|=n-t, j \in A} \sum_{i=1}^{t} \psi_{r_A^i}(a) \cdot f_A^i(j)$$

It is straightforward to verify that this results in shares consistent with the polynomial $f_0 = \sum_{A,|A|=n-t,P_j \in A} \sum_{i=1}^{t} \psi_{s_A^i}(a) \cdot f_A^i$, that $deg(f_0) \leq 2t$ and that $f_0(0) = 0$.

The above ideas for "non-interactive random secret-sharing" are less efficient if the number of sets A is large.[5] On the other hand, the pseudo-random function is used as a black-box and hence any pseudo-random function can be used. By Proposition 1, our solution is optimal among a class of generic schemes making a black-box use of a PRF.

[5] Note that when t is constant, the number of sets A is polynomial in n. Thus, share conversion allows to efficiently achieve a constant level of privacy with an arbitrarily high level of robustness.

Distributed PRFs An immediate application of the basic PRSS protocol described above is to the problem of distributing pseudorandom functions (or KDCs), studied by Naor, Pinkas and Reingold [27]. A distributed PRF should allow a client to query its value on any chosen input a by contacting n servers, where the output should remain private from any collusion of t servers. Moreover, even if t servers are actively corrupted, the client should still learn the right output. A simple solution to this problem is to use the PRSS protocol, where the client sends a to each server P_j and receives s_j, the corresponding Shamir-share of the output, in return. A similar scheme that was suggested in [27] relies on replicated PRFs but does not use share conversion. Thus, the communication complexity of the scheme from [27] is very high, as servers are required to send all replicated shares to the client.

4.1 Applications to Secure Multiparty Computation

We show how the pseudorandom secret-sharing approach can be used to securely compute low-degree polynomials via an efficient two-round protocol. We then discuss an extension of this result to general functions.

It will be convenient to use the following model for secure computation: the input will be supplied by m *input clients* $I_1, ..., I_m$. The computation will be performed by n *servers* $P_1, .., P_n$. The outputs are to be distributed to v *output clients* $O_1, ..., O_v$. Since one player can play several of these roles, this is a generalization of the standard model, which fits well with the applications we give later. We will assume that input clients can broadcast information to the servers and we also assume secure point to point channels between servers and output clients. A typical protocol will have the following two-round structure: in Round 1 each input client broadcasts values to the servers (one value for each input) and in Round 2 each server sends (over a secure channel) a message to each output client. For some applications Round 1 will not be necessary, in which case we get fully non-interactive protocols.

We assume an adversary that can corrupt any number of clients and up to t servers. We consider both the case of a passive and an active adversary. The adversary is for now assumed to be static (non-adaptive).

We will make the following set-up assumptions: sets of keys for a pseudorandom function have been distributed to input clients and servers, such that each input client can act as the dealer in the NIVSS protocol and the servers can execute the PRSS and PRZS protocols.

Secure computation of low-degree polynomials. We show how to securely compute the value of a degree d multivariate polynomial $Q()$ in m variables, where I_j supplies the j-th variable x_j. We assume that the output value $Q(x)$ is to become known to all output clients. Generalizations to more input variables, more polynomials and different polynomials for different output clients follow easily. For a passive adversary, we assume $dt < n$, while for an active adversary we assume $(d + 2)t < n$. The protocol proceeds as follows:

1. In round 1, each input client I_j acts as the dealer in the NIVSS protocol using x_j as his private input. (If the inputs x_i should be restricted to take binary values, BNIVSS is used instead of NIVSS.) Let $x_{j,i}$ be the share obtained by server P_i. We execute the PRZS protocol adapted such that we create shares of a degree dt polynomial that evaluates to 0 on 0. Let z_i be the share obtained by P_i. Each P_i now computes $Q(x_{1,i}, ..., x_{m,i}) + z_i$.
2. In round 2, each server sends $Q(x_{1,i}, ..., x_{m,i}) + z_i$ to all output clients.
3. Each output client considers the values it receives as values of a degree dt polynomial f - where up to t values may be wrong if the adversary is active. He reconstructs the value $f(0)$ (using standard error correction in the active adversary case) and defines this to be his output.

Note that if we only wanted to compute shares of the value $Q(x_1, ..., x_m)$ we could do this by simply omitting steps 2 and 3. Using Canetti's definition of secure function evaluation from [9], we get:

Theorem 3. *The above protocol computes the function $Q(x_1, ..., x_m)$ securely against a passive, static adversary if $dt < n$ and against an active, static adversary if $(d + 2)t < n$.*

Proof sketch. For some adversary Adv, the required Ideal model adversary, or simulator, works as follows: it simulates the broadcast of honest input clients by broadcasting random values. Since it knows the keys that corrupt input clients use in the NIVSS protocols, it can compute the values that these clients are sharing, and send them to the ideal functionality. Note that it also knows the keys held by corrupt servers so it can compute the share $sh_j = Q(x_{1,i}, ..., x_{m,i}) + z_i$ that each corrupt P_j holds of the result. When given the output value y, it therefore chooses a random polynomial f such that $f(0) = y$ and $f(j) = s_j$ for each corrupt P_j. And for each honest P_i, it sends $f(i)$ to corrupt output clients in round 2, to simulate the contributions of honest servers to the result.

To argue that the simulation works, we can argue along the same lines as for Lemma 3. If for some adversary Adv and some set of inputs $x_1, ..., x_m$, the output from the real process could be distinguished from that of the ideal process, we could build an algorithm Alg for breaking the pseudorandom function. Alg will have oracle access to $\psi_s()$ for all keys s not known to Adv, or to a set of random oracles. Alg will now use the given inputs $x_1, ..., x_m$ and keys for the corrupt players that it chooses itself, to execute the protocol with Adv. It emulates the honest players according to the protocol, except that it calls its oracles whenever an honest player would have used a key not known to Adv. One can now verify that if the oracles contain pseudorandom functions, we produce output distributed exactly as in the real process, whereas of they are random, we produce output according to the ideal process. It is in this last part that we need $dt < n$ ($(d + 2)t < n$) since this ensures that the polynomial f reconstructed by the honest output clients will determine the correct output value y, regardless of whether f was constructed by the simulator or by Alg.

Thus the ability to distinguish between the real and the ideal process translates to breaking the pseudorandom function with the same advantage. □

Note that this result extends easily to computing polynomials where some of the inputs are to be chosen at random: we just use the PRSS protocol to create shares of these inputs.

General MPC in 2 rounds. Using known techniques, one can reduce the secure computation of general functions to that of degree-3 polynomials. In case of functions that can be efficiently represented by (boolean or arithmetic) branching programs, such a reduction is given by constructions of randomizing polynomials from [21, 22]. A similar reduction for arbitrary (polynomial-time computable) functions is possible under standard intractability assumptions [1]. Alternatively, it is possible to modify the garbled circuit technique of Yao [31] and Beaver, Micali and Rogaway [2] to obtain 2-round protocols for arbitrary functions using our protocol for degree-3 polynomials. Using either approach, one can get 2-round general MPC protocols with security threshold $t < n/3$ ($t < n/5$) in the passive (active) case.

Distributing a trusted dealer. An important class of multi-party functionalities are those that distribute correlated random resources to output clients (without taking any inputs). For such functionalities, we can distribute a trusted dealer via a totally non-interactive protocol in which each server send a single message to each output client. Applications of such functionalities range from emulating a trusted poker dealer to providing players with correlated resources for general MPC protocols (e.g., [14, 16]). For the applications to threshold cryptography, discussed next, we use a similar approach but rely on the special structure of the relevant functionalities to gain better efficiency.

4.2 Applications to Threshold Cryptography

As mentioned above, low-degree polynomials that take only random inputs, possibly along with other inputs that have been pre-shared, can be securely computed in only one round using our techniques. In this section, we show that this efficiently extends to functions defined over finite groups of prime order, involving exponents that are low degree polynomials. It will later become clear how such functions can be used to handle problems in threshold cryptography without interaction.

We will assume that we work in a fixed finite group G of prime order q, such as a subgroup of Z_p^*, where q divides $p-1$. Furthermore, we assume we have an ideal implementation of a function that chooses a vector $X = (x_1, ..., x_u) \in Z_q^u$, secret-shares these values according to Shamir's scheme with polynomials of degree $\leq t$, and outputs shares $x_{i,j}$, $j = 1..u$, to each server P_i. Finally, the function chooses and distributes seeds as required for the PRSS and PRZS protocols we described earlier. This corresponds to the key generation phase in a threshold cryptosystem.

We assume as usual an adversary that corrupts at most t servers. For simplicity, we assume first that the adversary is static and passive. Consider a randomized function $\Phi()$, which we will define using a fixed set of multivariate degree 2 polynomials $Q_1(X,R),...,Q_w(X,R)$ (the following extends trivially to small degrees larger than 2, but degree 2 is all we will need in the following). To compute the function, all servers input the shares they received earlier, while an input client broadcasts elements $g_1,...,g_w \in G$ to the servers. We need that the shares supplied by servers uniquely determine X. This is always the case if the adversary is passive, and is also the case for an active adversary, provided $t < n/3$. The function outputs to all players the element

$$\Phi(g_1,...,g_w,\{x_{i,j}|\ i=1..n, j=1..u\}) = g_1^{Q_1(X,R)} \cdots g_w^{Q_w(X,R)}$$

where $R = (r_1,...,r_v) \in Z_q^v$ consists of v uniformly random numbers. We let $(\lambda_1,...,\lambda_n)$ be chosen as Lagrange interpolation coefficients such that $f(0) = \sum_{i=1}^n \lambda_i f(i)$ for any polynomial f with $deg(f) < n$.

To build a protocol for evaluating $\Phi()$, we use a technique similar to what we used before, but we now put everything "in the exponent": assuming players have shares of the x_i's and r_i's, they can compute the "same" expression as in the definition of Φ, using their respective shares in place of the x_i's and r_i's. When each local result is broadcast, one can find the answer using interpolation "in the exponent". And even though the shares of the r_i's are not predistributed, we can create them non-interactively using PRSS. Also as previously, we need to randomize the degree $2t$ polynomial that is (implicitly) revealed, using the PRZS technique.

Protocol Compute $\Phi()$

1. Each server P_i computes a share $r_{j,i}$ of a pseudorandom value r_j, for $j = 1..v$, as well as a share t_i in a degree-$(2t)$ sharing of 0, as described in protocols PRSS and PRZS. Let $R_i = (r_{1,i},...,r_{v,i})$ and $X_i = (x_{1,i},...,x_{u,i})$. He sends to the output client(s) the group element
$$G_i = g_1^{Q_1(X_i,R_i)} \cdots g_w^{Q_w(X_i,R_i)} g_1^{t_i}$$
2. The output client(s) compute the output as $\prod_{i=1}^n G_i^{\lambda_i}$.

For the above protocol, one can prove the following:

Theorem 4. *Assuming a passive, static adversary, $t < n/2$, and that $\psi.(\cdot)$ (used in the pseudo-random secret-sharing) is a pseudo-random function, the above protocol computes $\Phi()$ securely.*

Proof sketch. Rewriting the definition of $\Phi()$ by expressing all group elements as powers of some fixed generator g of G, it is clear that the output value is of form $g^{Q(X,R)}$ for some multivariate degree 2 polynomial. Hence, from an information theoretic point of view, we are in fact computing $Q(X,R)$ using exactly the protocol we saw earlier. Therefore, essentially the same proof as for the earlier protocol applies here. □

Note that since we need to assume that the number of servers is small in order to use PRSS in the first place, the results from [10] imply that this same protocol is also adaptively secure.

For the case of an active, static adversary, we can use the same protocol, provided that $t < n/4$ and that we make a modification in the final step. In the earlier protocol for computing low-degree polynomials, we could use standard error correction, but this will not work here. We are faced with elements $G_1, ..., G_n$ and all we know is that all but t of them are of form $G_i = g^{f(i)}$ for some polynomial f of degree at most $2t$. Since we cannot expect to compute discrete logs, direct error correction does not apply.

Since the number of servers is small, one option is to find the correct subset by exhaustive search. But this may not be entirely satisfactory. We do have to assume that servers have enough memory and computing power to handle such a problem (in order to carry out the PRSS protocol). But in a real application, we may not want to assume this about the clients.

We sketch a solution to this using results from [13]. We will assume two-level sharings, that is the x_j's have been shared and then the shares have themselves been shared using degree t polynomials. If $x_{j,i}$ is P_i's share, P_i also receives as part of his share the polynomial used for sharing $x_{j,i}$. We will use the same type of two-level sharing for the random r_j's and for the degree $2t$ sharing of 0. This can all be done non-interactively by our results on general share conversion, because such a two-level sharing is a linear scheme.

Hence, when P_i claims that the value G_i he contributes really satisfies $G_i = g_1^{Q_1(X_i,R_i)} \cdots g_w^{Q_w(X_i,R_i)} g_1^{t_i}$, we can assume that the exponents are shared among the servers using degree-$(2t)$ polynomials and P_i knows the polynomials that have been used. Therefore the value can non-interactively be shown to be correct using a straightforward generalization of the techniques from [13].

Theorem 5. *Assuming an active, static adversary, $t < n/4$, and that $\psi.(\cdot)$ (used in the pseudo-random secret-sharing) is a pseudo-random function, the above protocol, modified as described for active adversaries, computes $\Phi()$ securely (and non-interactively).*

We expect that this general technique will be useful in many contexts. Below we give a few examples for applications to some concrete threshold cryptosystems.

Threshold Cramer-Shoup. Canetti and Goldwasser [11] proposed a threshold version of the Cramer-Shoup cryptosystem, the first really efficient public-key system that could proved secure under chosen ciphertext attacks, without assuming random oracles.

This scheme works in a group G of order q as we did above. The private key is (x_1, x_2, y_1, y_2, z), all chosen at random in Z_q. The public key consists of a number of elements in G, namely $g_1, g_2, c = g_1^{x_1} g_2^{x_2}, d = g_1^{y_1} g_2^{y_2}, h = g_1^z$.

A ciphertext is a 4-tuple of elements (u_1, u_2, e, v). To decrypt, one computes a value α from the ciphertext using a public hash function. Then we set $v' = u_1^{x_1+y_1\alpha} u_2^{x_2+y_2\alpha}$. We choose $r \in Z_q$ at random and compute $b = u_1^z (v' v^{-1})^r$. Finally, the output (which will be the decrypted message if the ciphertext was valid) is eb^{-1}.

The randomization introduced by r is a modification of the original scheme suggested in [11] to maintain CCA security even if one defines the decryption algorithm to output b always - instead of an error message in case the ciphertext was invalid. Clearly, if one can securely compute b assuming that x_1, x_2, y_1, y_2, z have been pre-shared, a secure threshold version of the scheme follows. In [11] this was done non-interactively, essentially assuming that a number of random values had been pre-shared, so they could play the role of r. This resource runs out quickly since r-values cannot be reused, so this is not a satisfactory solution. Indeed Canetti and Goldwasser asked whether it was possible to create shares of r pseudorandomly without interaction. This is exactly possible using our techniques. Indeed by rewriting, the value we need to compute is

$$b = u_1^{z+x_1 r+\alpha y_1 r} u_2^{x_2 r+\alpha y_2 r} v^{-r}$$

It should be clear that this expression is a special case of the (class of) function(s) $\Phi()$. Hence a protocol for computing b securely follows immediately from the protocols for computing Φ, both in the passive and active adversary case. We also obtain the same bounds on t as in [11] [6].

Threshold Signatures. It is known how to obtain efficient non-interactive threshold signatures in the random oracle model based on RSA, see for instance [29]. If we drop the random oracle assumption, things seem to be much more difficult. We do know efficient secure signature schemes that need no random oracles [15, 17], but it is not at all clear how one could design a non-interactive threshold version of those schemes.

However, we can make use of the fact that Boneh and Boyen in [8] suggested a fully secure ID based encryption scheme without random oracles. A more efficient scheme was suggested by Waters in [30].

Briefly, an ID based encryption scheme has a public key pk, and a master secret key sk. Each user has an identity ID, and using the master key one can derive a secret key sk_{ID} for this user. Knowing only the ID, it is possible to encrypt a message such that only the user who knows sk_{ID} can decrypt.

Our interest in this comes from the fact that any such scheme implies a signature scheme, with public key pk and private key sk. To sign a string, one thinks of it as an identity and uses sk to extract sk_{ID} which now plays the role of a signature. The security properties of the ID based scheme imply security of the signature scheme in a natural way.

The scheme of [30] works in a prime order group G equipped with a bilinear mapping (which we do not have to consider explicitly here). Keys are generated as follows: fix a generator g of G, choose a random $\alpha \in Z_q$ and set $g_1 = g^\alpha$. Also pick random elements $g_2, u', u_1, ..., u_l$ where l is the length of identities. Then the public key is $g, g_1, g_2, u', u_1, ..., u_l$ and the master secret key is g_2^α. The secret

[6] In the active case, it was claimed in [11] that their solutions work for $t < n/3$ non-interactively and for $t < n/2$ with interaction, but this is not correct. The authors of [11] have confirmed that the correct bounds are $t < n/4$ and $t < n/3$, respectively.

key corresponding to identity v where the i'th bit of v is v_i is constructed by choosing r at random and setting sk_{ID} to be the pair

$$sk_{ID} = (g_2^{\alpha}(u' \prod_{i \in V} u_i)^r,\ g^r)$$

where V is the set of indices such that $v_i = 1$. Now, since $u' \prod_{i \in V} u_i$ is an element all players can compute by themselves, also this expression is a special case of our function Φ. It follows that the protocol for computing Φ can be used to compute, non-interactively and securely, a signature in the scheme derived from [7].

This is the first non-interactive threshold signature scheme that can be shown secure without random oracles. We note that concurrently and independently from our work, Boneh and Boyen have recently found a different technique for distributing this signature scheme. This method is tailored to their scheme and does not use share conversion based techniques. It scales better w.r.t. the number of players than our method, but is less general. For instance, our technique also applies directly to distribute non-interactively the recent signature scheme by Camenisch and Lysyanskaya [12]. This leads to a distributed signature scheme with a much smaller public key than starting from Waters' scheme, and it also implies a distributed implementation of the authorities issuing credentials in their anonymous credential scheme.

References

1. B. Applebaum, Y. Ishai, and E. Kushilevitz. Computationally private randomizing polynomials and their applications. Manuscript, 2004.
2. D. Beaver, S. Micali, and P. Rogaway. The round complexity of secure protocols (extended abstract). In *Proc. of 22nd STOC*, pages 503–513, 1990.
3. D. Beaver and A. Wool. Quorum-based secure multi-party computation. In *Proc. of EUROCRYPT '98, LNCS 1403, Springer Verlag*, pages 375–390, 1998.
4. A. Beimel. *Secure schemes for secret sharing and key distribution.* PhD thesis, Technion, 1996.
5. A. Beimel, Y. Ishai, E. Kushilevitz, and J. F. Raymond. Breaking the $O(n^{1/(2k-1)})$ Barrier for Information-Theoretic Private Information Retrieval. In *Proceedings of the 43rd IEEE Conference on the Foundations of Computer Science (FOCS '02)*, pages 261–270, 2002.
6. J. Benaloh and J. Leichter. Generalized secret sharing and monotone functions. In *Proc. of CRYPTO '88, LNCS 403, Springer Verlag*, pages 27–35, 1990.
7. D. Boneh and X. Boyen. Efficient Selective Identity-based Encryption. In *Proc. of Eurocrypt '04*.
8. D. Boneh and X. Boyen. Secure Identity-Based Encryption Without Random Oracles. In *Proc. of Crypto '04*.
9. R. Canetti. Security and composition of multiparty cryptographic protocols. In *J. of Cryptology*, 13(1), 2000.
10. R. Canetti, I. Damgård, S. Dziembowski, Y. Ishai, and T. Malkin. On Adaptive vs. Non-adaptive Security of Multiparty Protocols. In *J. of Cryptology*, 17(3), 2004. Preliminary version in Eurocrypt '01.

11. R. Canetti and S. Goldwasser. An efficient threshold public-key cryptosystem secure against adaptive chosen ciphertext attacks. In *Proc. of Eurocrypt '99.*
12. J. Camenisch and A. Lysyanskaya. Signature Schemes and Anonymous Credentials from Bilinear Maps. In *Proc. of Crypto 2004.*
13. R. Cramer and I. Damgård. Secret-Key Zero-Knowledge and Non-interactive Verifiable Exponentiation. In *Proc. TCC '04.*
14. R. Cramer, I. Damgård, and J. Nielsen. Multiparty computation from threshold homomorphic encryption. In *Proc. of EUROCRYPT '01*, LNCS 2045, pp. 280-299, 2001.
15. R. Cramer and V. Shoup. Signature Schemes Based on the Strong RSA Assumption. In *Proc. ACM Conference on Computer and Communications Security,* 1999.
16. M. Fitzi, S. Wolf and J. Wullschleger. Pseudo-signatures, broadcast, and multiparty computation from correlated randomness. In *Proc. Crypto '04.*
17. R. Gennaro, S. Halevi and T. Rabin. Secure Hash-and-Sign Signatures Without Random Oracles. In *Proc. of Eurocrypt '99.*
18. R. Gennaro, Y. Ishai, E. Kushilevitz and T. Rabin. The Round Complexity of Verifiable Secret Sharing and Secure Multicast. In *Proceedings of the 33rd ACM Symp. on Theory of Computing (STOC '01)*, pages 580-589, 2001.
19. R. Gennaro, Y. Ishai, E. Kushilevitz and T. Rabin. On 2-round secure multiparty computation. In *Proc. Crypto '02.*
20. N. Gilboa and Y. Ishai. Compressing cryptographic resources. In *Proc. of CRYPTO '99.*
21. Y. Ishai and E. Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *Proc. 41st FOCS*, pp. 294–304, 2000.
22. Y. Ishai and E. Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. In *Proc. 29th ICALP*, pp. 244–256, 2002.
23. M. Ito, A. Saito, and T. Nishizeki. Secret sharing schemes realizing general access structures. In *Proc. IEEE Global Telecommunication Conf., Globecom 87*, pages 99–102, 1987.
24. M. Karchmer and A. Wigderson. On span programs. In *Proc. of 8th IEEE Structure in Complexity Theory*, pages 102–111, 1993.
25. U. Maurer. Secure multi-party computation made simple. In *Proc. of SCN '02.*
26. S. Micali and R. Sidney. A simple method for generating and sharing pseudo-random functions with applications to clipper-like key escrow systems. In *Proc. of CRYPTO '95, LNCS 963, Springer Verlag*, pages 185–196, 1995.
27. M. Naor, B. Pinkas, and O. Reingold. Distributed pseudo-random functions and KDCs. In *Proc. of EUROCRYPT '99, LNCS 1592, Springer Verlag*, pages 327–346, 1999.
28. A. Shamir. How to share a secret. *Commun. ACM*, 22(6):612–613, June 1979.
29. V. Shoup. Practical Threshold Signatures. In *Proc. of Eurocrypt '00.*
30. B. R. Waters. Efficient Identity-Based Encryption Without Random Oracles. Eprint report 2004/180.
31. A. C. Yao. How to generate and exchange secrets. In *Proc. 27th FOCS*, pp. 162–167, 1986.