Compressing Cryptographic Resources

(Extended Abstract)

Niv Gilboa and Yuval Ishai

Computer Science Department
Technion, Haifa 32000, Israel
{gilboa,yuvali}@cs.technion.ac.il

Abstract. A private-key cryptosystem may be viewed as a means by which a trusted dealer privately conveys a large, shared pseudo-random object to a pair of players, using little communication. Alternatively, the messages distributed by the dealer may be viewed as a secure *compression* of a pair of large identical random pads (or random functions) into a shorter shared "key" or "seed".
We pose the question of extending this compression problem to more general correlation patterns among *several* players. Unlike the simple case of identical pads, where the main security concern is with respect to external eavesdroppers, in the case of general correlations participants also have to be protected from each other. That is, collusions of computationally-bounded players should gain no additional knowledge about the joint pads of the remaining players from the compressed messages they receive, other than what follows from the pads they generate and from knowing the joint distribution of all pads. While this ideal requirement is inherently impossible to meet using little communication, it turns out that it can be approximated to a satisfactory level, allowing to securely use such compressed correlated pads in a wide class of protocols. We propose a simple and modular replication-based approach for securely compressing any *linear* correlation pattern, using pseudo-random generators or pseudo-random functions in a black-box manner. Applications include amortizing the communication costs of private multi-party computation and proactive secret-sharing of large secrets.

1 Introduction

This work introduces and studies a natural generalization of the fundamental notions of pseudo-randomness (cf. [28,6,12,18]) to a useful and natural notion of *privately correlated pseudo-randomness*. Alternatively, our generalized notion may be viewed as providing a secure mechanism for *compressing* large correlated pads used as resources in cryptographic protocols.

1.1 Motivation

We consider a scenario in which a trusted dealer wishes to aid a group of players in performing some cryptographic task by supplying them with a resource of correlated messages. For instance, the dealer may allow a pair of players to

Michael Wiener (Ed.): CRYPTO'99, LNCS 1666, pp. 591–608, 1999.

communicate with unconditional secrecy by sending them a sufficiently long shared random one-time pad, or allow all players to form a secure multicast group by sending to all of them the same random pad.

As a second and slightly more sophisticated example, the dealer may aid the players to privately compute the modular sum of their inputs by granting them otherwise-random pads that add up to 0. Then, if each player adds his pad to his input and broadcasts the result, the broadcasted messages reveal no information about individual inputs other than their sum. Moreover, even collusions of players cannot learn from their view additional information about inputs of other players other than what follows from their inputs and the outcome of the computation (namely, the sum of all inputs).

The above solutions to the problems of secure communication and private addition possess several nice properties. First, they provide information theoretic security. Second, they are conceptually simple and computationally super-efficient. Finally, they induce no on-line communication overhead: all extra communication is concentrated in an off-line distribution stage. Then, why aren't these solutions practical?

An obvious and prohibitive disadvantage of these solutions is their off-line communication cost.[1] Indeed, to maintain information theoretic security, each pad distributed by the dealer can only be used to mask a message of the same size; thus, if the players wish to encrypt long messages or perform many private additions, the dealer has to communicate sufficiently long pads. Even if we assume totally secure communication during the off-line distribution stage, the burden on the dealer would become unmanageable.

For the first example above, namely distributing a long shared random pad to two or more players, there is a standard way of reducing the off-line communication. The long shared pad may be replaced by a "compressed capsule" in the form of a seed to a pseudo-random generator or a key of a pseudo-random function. Such a seed or key may be "uncompressed" by the players holding it to obtain a large shared pseudo-random object which cannot be distinguished from a truly random object by external observers. However, for the second example of correlated pads which sum up to zero there seems to be no standard compression method.

Let us see why some simple approaches towards compressing the second kind of correlation fail. First, observe that if the same seed is sent to all players, it may be uncompressed to yield a distribution which is indistinguishable (by an external observers) from the one to be generated. However, such a solution totally fails to protect the privacy of players from each other: using this seed, each player will be able to recover the inputs of the other players from the broadcasted messages. A second attempt is to try to construct a pseudo-random generator such that given independently chosen seeds $s_1, \ldots, s_{n-1}$, it is always possible to find a last seed s_n such that the expansions of all seeds add up to

[1] This is aside from other important issues such as authentication and trust management, which should be handled by any private-key infrastructure and are independent of the concerns of this work.

(a long) zero. However, no such pseudo-random generator exists, as this would imply that its image forms a linear space.

In this work we pose the question of compressing more general correlation patterns and study the special case of *linear* correlations, where the pads form a random vector in a linear space.

How to Share a Zero. As paradoxical as this notion may sound, "secretly sharing a zero" among several players (i.e., using a secret-sharing scheme with the secret set to zero) is a common ingredient in cryptographic protocols. One example is the abovementioned application of privately computing the sum of n inputs. More generally, privately computing linear combinations of inputs is an essential subprotocol in general private multi-party protocols (e.g., [5,9]). But perhaps a better example is that of *proactive secret-sharing* and proactive multi-party computation (cf. [24,13,26]), where security has to be preserved over time against a *mobile* attack. In such applications, a shared representation of zero is periodically generated, either by an external dealer or by the players themselves, for the purpose of refreshing the distributed representation of some shared secret without changing its value. When the secret is very long, as may be the case when proactively sharing a large file, an expensive sharing of a "long zero" is required. One approach towards solving this problem (in the spirit of [17]) is to disperse an encryption of the large file among the players (without secrecy), and proactively share the short encryption key. However, a straightforward implementation of such a solution does not allow to retrieve or update a portion of the shared file without compromising the secrecy of the key. Moreover, some applications (e.g., protocols from [23]) crucially rely on the assumption that a non-encrypted representation of the entire file is shared using some simple linear secret-sharing scheme.

The above discussion motivates the problem of secretly sharing a "long zero" in a communication efficient way. While secret-sharing an *arbitrary* secret cannot use less communication than the length of the secret, it turns out that a shared long zero can be compressed.

1.2 Results

We focus on the case of *linear* correlations, where the pads form a random vector in a linear space. Moreover, while some of our results can be easily pushed to more adversarial scenarios, we adhere to a minimalistic model of an honest dealer and a passive adversary, keeping the problem as clean as possible.

We start by studying the information theoretic problem of *private correlation from replication*. This problem is motivated by its suitability for replacing perfect randomness with pseudo-randomness in a black-box manner. The setting is as follows. A dealer generates several *independent* random messages X_j, $1 \leq j \leq m$, and replicates them among n players (i.e., sends each message to a prescribed subset of the players). Then, each player P_i performs some local computation on the messages he received, resulting in some output value Y_i. We say that such a protocol privately generates a distribution D from replication, if: (1) the outputs $(Y_1, \ldots, Y_n)$ are distributed according to D; and (2) each collusion of players

cannot learn from the messages X_j they receive any additional information about $(Y_1, \ldots, Y_n)$ other than what follows from their outputs.

We obtain the following results:

- For the case of additively sharing a zero, we obtain an optimal[2] scheme with $\binom{n}{2}$ messages, each sent to 2 players. This result is refined to privacy with respect to generalized adversary structures[3] (in which privacy should only be maintained with respect to specified collusions), by reducing the problem to an interesting extremal graph-theoretic question. This refined result, in turn, has application to the construction of communication- and round-efficient (information theoretically) private protocols computing modular sum relative to general adversary structures.
- For any linear correlation, uniform over some linear space V, we obtain an optimal scheme whose complexity is proportional to the number of vectors in V with minimal support sets. For some linear spaces (such as the one corresponding to sharing a zero using Shamir's $(n/2, n)$-threshold secret-sharing scheme [27]), the complexity of this solution will be exponential in the number of players.

We then address the possibility of replacing the random independent sources X_j by random independent seeds to a pseudo-random generator, thereby allowing to privately generate large correlated *pseudo-pads* using little communication. We show that while such a substitution cannot *always* preserve the security of an underlying information theoretically secure protocol, for a wide class of protocols meeting some mild technical requirement security is guaranteed to be preserved.

A major disadvantage of most of our solutions is their poor scalability; in some applications their complexity grows exponentially with the number of players.[4] However, when the number of players is reasonably small, the advantages of our approach arguably outweigh this unfortunate byproduct of the replication paradigm. One such advantage is the ability to use any block- or stream-cipher in a black-box manner, which may result in orders of magnitudes of efficiency improvement over potential solutions based on specific number-theoretic constructions. Finally, we stress that the task of uncompressing correlated pads may be done off-line. This means that the on-line efficiency of protocols using our correlated pseudo-pads can be the same as the efficiency of protocols using perfect pads. For instance, suppose that a proactive secret-sharing protocol requires a large file to be shared using Shamir's secret-sharing scheme (for efficient on-line retrieval or updates of selected information). In each refreshment period correlated pseudo-pads are uncompressed and "added" to the current shares, and their seeds are erased. At the end of such a refreshment process, portions of

[2] Here and in the following, "optimal" should be interpreted as: "optimal under the replication paradigm".

[3] The notion of private computation with respect to general adversary structures was first introduced and studied in [14].

[4] Exceptions are resources used for private modular addition (relative to any adversary structure), or (t, n)-threshold secret-sharing of zero when either t or $n - t$ are constant.

the shared file may be freely reconstructed or updated without any additional communication or computation overhead.

We finally note that while the main setting we consider when describing applications involves a trusted dealer, this is not an essential assumption. In fact, assuming that the players are honest but curious, the replication paradigm allows simulating the dealer by the players with no extra cost.

1.3 Related Work

The idea of treating correlated pads as useful cryptographic resources has been recently put forward in [3] under a specialized "commodity-based" model, and is implicit in many other works. However, to our knowledge the idea of *compressing* such general resources has never been treated.

Our paradigm of realizing private correlation via replication is related to the notion of replication-based secret-sharing schemes, which were introduced in [16] as a means of implementing secret-sharing relative to general access structures. Such schemes proceed by first *additively* sharing a secret, writing it as the sum (over a finite group) of otherwise random shares, and then distributing each additive share to some set of players associated with it. Replication-based secret-sharing schemes are implicitly used in [21,22] in the related context of sharing pseudo-random functions, and are used in [4] to realize private computation relative to general adversary structures and in [15] for obtaining efficient private information retrieval schemes. Some special cases of our constructions resemble (and in a sense generalize) the secret-sharing schemes used in these works; in general, however, both the semantics of the problem studied in this work and the replication-based combinatorial structures which underly our solutions are quite different. Our approach for compressing linear correlations may be viewed as a further, differently motivated, application of the replication technique.

Organization. The rest of this paper is organized as follows. Section 2 introduces some general notation. In Section 3 we define the information theoretic notion of private correlation from replication and study it for linear correlation patterns. In Section 4 we use the results of Section 3 to obtain privately correlated pseudo-random generators. Section 5 contains some concluding remarks. Finally, the appendices contain some definitions and a proof deferred from the main body of text.

2 Notation

By $[n]$ we denote the set $\{1, 2, \ldots, n\}$. We use capital letters to denote random variables or probability distributions, and small letters to denote their instances. By $x \leftarrow X$ we denote a random choice of an instance x from the distribution X. All such random choices are made independently of each other. Additional context-specific notation will be defined in subsequent sections.

3 The Information Theoretic Setting: Private Correlation from Replication

In this section we study the information theoretic question of privately generating a joint distribution of correlated outputs from replicated *independent* random sources. This question is motivated by the possibility of replacing the independent random sources by seeds to a pseudo-random generator or a pseudo-random function, which will be explored in the next section.

A formal definition of this notion of *private correlation from replication* is given in the following subsection.

3.1 Definitions

Let $X_1, X_2, \ldots, X_m$ be independent random variables, where each X_j is uniformly distributed over a finite domain $\mathcal{X}_j$, let $X = (X_1, X_2, \ldots, X_m)$ and $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_m$. Let $\mathcal{Y}_1, \ldots, \mathcal{Y}_n$ be some finite sets, and $\mathcal{Y} = \mathcal{Y}_1 \times \cdots \times \mathcal{Y}_n$. For $1 \leq i \leq n$, let f_i be some mapping from $\mathcal{X}$ to $\mathcal{Y}_i$, and $S_i \subseteq [m]$ denote the set of arguments on which f_i depends (e.g., if $f_3(x_1, \ldots, x_m)$ depends on x_2 and x_5 and *only* on these arguments, then $S_3 = \{2, 5\}$). Finally, we let $f = (f_1, \ldots, f_n)$ (so that f maps from $\mathcal{X}$ to $\mathcal{Y}$), and for $1 \leq i \leq n$ we define the random variable $Y_i = f_i(X)$, and $Y = (Y_1, \ldots, Y_n) = f(X)$. Notice that Y is a random variable distributed over $\mathcal{Y}$.

Before proceeding with the main definition, we briefly describe the correspondence between the above notation and the correlation from replication paradigm that motivates it. The random variables X_j are the independent random sources to be replicated among the n players. The function f (and its associated sets S_i) induces the following pad distribution protocol:

- The dealer generates $x \leftarrow X$ and sends to each player P_i, $1 \leq i \leq n$, all components x_j with $j \in S_i$.
- Each player P_i locally computes $y_i = f_i(x)$ based on the components of x he received,[5] and outputs y_i as his random pad.

Notice that the function f concisely defines the entire protocol, including both the identity of messages sent to each player and the local computations of all players.

For any set $T \subseteq [n]$, we let $Y_T \stackrel{\text{def}}{=} (Y_i)_{i \in T}$ and $X_T \stackrel{\text{def}}{=} (X_j)_{j \in \cup_{i \in T} S_i}$. The random variable Y_T includes the joint outputs of the players P_i, $i \in T$, and X_T includes all messages they receive from the dealer.

We are now ready to define the notion of privately generating a given probability distribution from replicated independent sources.

Definition 1. *Let D be a probability distribution over $\mathcal{Y}$, and $\mathcal{T} \subseteq 2^{[n]}$ be an* adversary structure *(specifying collusions of players to protect against). We say that a mapping f as above $\mathcal{T}$-*privately generates *D from replication of X (or $\mathcal{T}$-*privately generates *D for short) if the following two conditions hold:*

[5] Note that by the definition of the sets S_i, the x_j's sent by the dealer are necessary and sufficient for this local computation to be possible.

Correctness. *The output n-tuple $Y(= f(X))$ is distributed according to D.*
Privacy. *For any collusion $T \in \mathcal{T}$, the random variable Y is independent of X_T given Y_T; that is, for any $x \in \mathcal{X}$,*

$$\Pr[Y = y \mid X_T = x_T, Y_T = y_T] = \Pr[Y = y \mid Y_T = y_T],$$

where $y = f(x)$. Intuitively, the above privacy condition asserts that the messages X_T which members of T receive from the dealer give them no additional information about the joint output Y other than what follows from their outputs Y_T.

Finally, we say that f t-privately generates *D if it $\mathcal{T}$-privately generates D for the threshold structure $\mathcal{T} = \{T \subseteq [n] : |T| \leq t\}$, and that it* privately generates *D (or* generates *D* with full privacy*) if it n-privately generates D.*

3.2 Additively Sharing a Zero

Before proceeding to study the general case of linear correlations, we start with the important special case of *additively sharing a zero*. Since the basic solution we provide for this special case will be generalized in the next subsection, we allow ourselves to omit some formal proofs and focus on the high-level intuition. Moreover, while in the general case we mainly address the question of constructing fully-private correlation generators, for the special case of additively sharing a zero we pay more attention to realizing privacy with respect to threshold and non-threshold adversary structures, and reduce the latter question to an interesting extremal graph-theoretic problem.

Fix a finite field F,[6] and define the distribution D_s as the uniform distribution over all n-tuples $(d_1, \ldots, d_n) \in F^n$ such that $\sum_{i=1}^{n} d_i = 0$. We first address the case of generating D_s with *full* privacy, and then proceed to study more general adversary structures.

To privately generate D_s from replication, consider the clique K_n whose n vertices are labeled by player indices, and assign an independent random source X_e to each edge $e = \{u, v\}$ in the graph. That is, let $m = \binom{n}{2}$, and label the m entries of X by edge names. Moreover, assume that the edges of the graph are arbitrarily directed (e.g., from the smaller numbered vertex to the larger one). Such a directed version of K_n, and similarly of any other graph, induces the following scheme for generating D_s: Each player P_i receives all sources X_e such that e is incident to its corresponding vertex, and outputs the difference

$$f_i(X) \stackrel{\text{def}}{=} \sum_{e \text{ enters vertex } i} X_e - \sum_{e \text{ exits vertex } i} X_e.$$

It can be easily verified that any such graph-based scheme satisfies the correctness requirement, since the contribution of each edge to the sum $\sum_{i=1}^{n} f_i$ is 0. An obvious *necessary* condition for the privacy requirement is that each

[6] Results here and in the following section carry on to more general algebraic structures. For instance, here F may be replaced by any finite Abelian group.

collusion T of at most $n-2$ players misses at least one source X_e; otherwise, the variables X_e held by such a collusion would completely determine the joint output Y, contradicting the privacy requirement. Indeed, this condition is met by the above clique-based scheme since each such player set T misses at least one edge. This necessary condition also shows that the complexity of the clique-based solution is optimal under our replication paradigm. This is argued as follows. For each collusion T containing *exactly* $n-2$ players, there must exist a source X_j (on which f depends) which T misses; moreover, X_j must be distributed to *both* players outside T, for otherwise the correctness requirement prevents f from depending on X_j. It follows that for each set T of size $n-2$ there exists a unique source X_j it misses, from which it follows that $m \geq \binom{n}{2}$.

In the case of full privacy, it turns out that the above necessary condition for privacy is also sufficient; however, as the following example shows, this is not the case for other adversary structures.

Example 1. Consider the replication scheme induced by a cycle of length n (whose edges are arbitrarily directed, as in the above case of a clique). Such a scheme satisfies the correctness requirement, and also satisfies the above necessary condition for privacy for $|T| < n/2$. Indeed, every set of less than $n/2$ vertices cannot cover all edges of the cycle. However, it is easy to see that there exist collusions of two players which can totally determine the output of a third player, thereby violating the privacy condition of Definition 1. For instance, in the scheme induced by the cycle $(P_1, P_2, P_3, \ldots, P_1)$, P_1 and P_3 can determine the output of P_2 since they cover all edges incident to him.

A *necessary and sufficient* condition for privacy should assert that for any collusion $T \in \mathcal{T}$, the *dimension* of the space of possible Y values given X_T is exactly $n - |T| - 1$, the dimension of Y given Y_T. Intuitively, each of these two dimensions corresponds to the number of "degrees of freedom" of the outputs of the remaining players given a view of members of T. The analysis in the next subsection (or the proof sketch of the following theorem) will show that the above condition is indeed satisfied by the clique-based scheme.

We now reduce the problem of $\mathcal{T}$-privately generating additive shares of zero to an extremal graph theoretic problem.

Theorem 1. *Let $\mathcal{T} \subseteq 2^{[n]}$ be an adversary structure, and let $G(V, E)$ be a graph with $V = [n]$. Then, G induces a $\mathcal{T}$-private generator of D_s if and only if for every $T \in \mathcal{T}$ the subgraph $G - T$, induced by removing all T-vertices from G, is connected.*[7]

Proof sketch. The proof follows from the fact that the rank of the incidence matrix[8] of a graph equals its number of vertices minus its number of connected components (cf. [7, page 38]). This implies that if T does not disconnect G, then

[7] Note that the above requirement on the subgraphs $G - T$ is not monotone with respect to T (i.e., it is possible that $T' \subseteq T$ yet only the removal of T' disconnects G).

[8] The incidence matrix of a graph $G(V, E)$ is a $|V| \times |E|$ matrix M, such that for any edge $e = \{u, v\} \in E$ the e-th column of M contains 1 in its u-th entry, -1 in its v-th entry (or vice versa) and 0 elsewhere.

the dimension of the space of possible Y values given X_T is exactly $n - |T| - 1$ as required, and otherwise it is smaller. □

The above connectivity requirement illuminates a qualitative difference between the type of combinatorial structures considered in our context and the ones required in the related context of replication-based secret-sharing [16,21]. Crudely speaking, the latter context only induces covering-type requirements on restricted unions of replication sets, which in our context are generally not sufficient (as demonstrated by Example 1).

For the special case of threshold structures, Theorem 1 suggests that a t-private generator of D_s can be based on any $(t+1)$-connected graph. For instance, when t is odd, a corresponding optimal graph may be obtained by connecting each vertex i to the $(t+1)/2$ vertices with (cyclically) consecutive indices.

We end this subsection by mentioning a further application of the above graph-theoretic characterization. By letting the players simulate the dealer (e.g., having each replicated source be picked and multicast by one player to which it is assigned), the above results can be used to obtain communication-efficient, perfectly-private two-round protocols for modular addition relative to general adversary structures. For the case of threshold structures, this can be compared with the (communication-optimal) private protocol of [10], which is only slightly more communication-efficient but requires many rounds of interaction.

3.3 Optimal Construction for Linear Correlations

We now move to the general case of *linear* correlations, where the joint output distribution D forms a uniformly random vector in some linear space over a finite field F. In the following we will stick to coding theory terminology: we will refer to the linear space as a *linear code*, and to a vector in this space as a *codeword*. To simplify notation, we only address the case where D is uniformly distributed over some *length-n* linear code $C \subseteq F^n$; that is, we focus on generating probability distributions in which each player P_i outputs the i-th coordinate of a uniformly random codeword $c \in C$. More general linear correlations, in which each player outputs several coordinates of a random codeword c, can be handled in a similar fashion.

Let D_C denote the uniform distribution over C. We will privately generate the distribution D_C using a linear generator of the form $f(x) = Mx$, where M is an n-row matrix over F. Note that in order to satisfy the correctness requirement alone, it suffices for the columns of M to generate the code C. This is not sufficient, however, for satisfying the privacy requirement. For instance, if M contains only nonzero entries, then each player views all of X and hence obtains full information on Y.

Our construction of a private generator for D_C will use a notion of *minimal support codewords*. For any codeword $c \in C$ we define the support set of c, denoted $s(c)$, as the set of coordinates i such that $c_i \neq 0$. We say that a codeword c is of *minimal support*, if $c \neq 0$ and there is no $c' \in C$, $c' \neq 0$, such that $s(c') \subset s(c)$. It is straightforward to observe that if two codewords c, c' are both of minimal support and $s(c) = s(c')$, then one must be a multiple of the other (otherwise there exists a linear combination of c, c' whose support contradicts

their minimality). Hence the set of minimal support codewords in C can be naturally partitioned into equivalence classes, where two minimal support codewords are equivalent if one is a multiple of the other. Finally, we define $\mathrm{Min}(C)$ as a set of representatives of the above equivalence classes (e.g., $\mathrm{Min}(C)$ may include all minimal support codewords whose first nonzero coordinate is 1).

The following theorem gives our construction of a private linear correlation generator.

Theorem 2. *Let C be a length-n linear code over F, and M be an $n \times m$ matrix whose m columns consist of all codewords in $\mathrm{Min}(C)$ (i.e., all representatives of minimal-support codewords in C). Then, the linear function $f_M : F^m \to F^n$ defined by $f_M(x) = Mx$ privately generates D_C from replication.*

The proof of Theorem 2 uses the following sequence of lemmas. The (straightforward) proofs of the first two lemmas are omitted from this version.

Lemma 1. *For any linear code C, $\mathrm{span}(\mathrm{Min}(C)) = C$.* □

Lemma 2. *Let U^b denote the uniform distribution over F^b. Then, for any $a \times b$ matrix A over F, the random variable AU^b is uniformly distributed over the column-span of A.* □

It follows from Lemmas 1, 2 that $Y = MX$ is uniformly distributed over C. Hence, the function f_M defined in Theorem 2 *correctly* generates D_C. It remains to show that it also satisfies the privacy requirement, with respect to any collusion $T \subseteq [n]$.

We Let x_T, y_T denote restrictions of instances x, y of the random variables X, Y, similarly to the notation X_T, Y_T defined in Subsection 3.1. (Note that this restriction has a different meaning when applied to an m-tuple x than when applied to an n-tuple y). We let $Y|x_T$ denote the conditional space induced by conditioning the random variable Y by the event $X_T = x_T$. Similarly, by $Y|y_t$ we denote the conditional space induced by conditioning the random variable Y by the event $Y_T = y_T$. Intuitively, $Y|x_T$ models the information learned by members of T from their view of X, and $Y|y_T$ models what they must infer from their outputs.

Lemma 3. *For any $x \in F^m$, the conditional spaces $Y|x_T$ and $Y|y_T$ are identically distributed, where $y = f_M(x)$.*

Proof sketch. Let $[Y|x_T]$ and $[Y|y_T]$ denote the supports of the corresponding spaces (i.e., the sets of $y \in F^n$ that have nonzero probability). First, writing the corresponding systems of linear equations, it is easy to observe that both spaces are *uniformly* distributed over some affine subspace of C. Hence, it suffices to show that the supports of the two spaces are equal, i.e. that $[Y|x_T] = [Y|y_T]$.

Let $C_{\overline{T}}$ denote the subspace of C consisting of all codewords c such that $c_T = 0$, and $M_{\overline{T}}$ denote the submatrix of M containing the rows with indices from $\overline{T} = [n] \setminus T$, restricted to columns from $C_{\overline{T}}$ (whose T-rows are all zero). That is, write M as

$$M = \begin{pmatrix} M_1 & 0 \\ M_2 & M_{\overline{T}} \end{pmatrix},$$

where the rows of M_1 are those labeled by members of T, and every column of M_1 includes at least one nonzero entry.

We now turn to show the equality $[Y|x_T] = [Y|y_T]$. The "$\subseteq$" inclusion follows immediately from the fact that x_T *determines* y_T (since y_T is the result of local deterministic computations performed on x_T). For the other inclusion, it suffices to show that $|[Y|y_T]| = |[Y|x_T]|$, or equivalently that $\dim[Y|y_T] = \dim[Y|x_T]$ (where $\dim V$ denotes the dimension of an affine space V). From the corresponding systems of linear equations one can deduce that:

$$\dim[Y|y_T] = \dim C_{\overline{T}} \tag{1}$$

$$\dim[Y|x_T] = \operatorname{rank} M_{\overline{T}}. \tag{2}$$

It follows from the definitions of M and $M_{\overline{T}}$ that the columns of $M_{\overline{T}}$ (augmented with zeros as their T-entries) include all codewords in $\mathrm{Min}(C_{\overline{T}})$. By Lemma 1 these columns span $C_{\overline{T}}$, hence $\dim C_{\overline{T}} = \operatorname{rank} M_{\overline{T}}$, and using equations (1),(2) above we obtain that $\dim[Y|y_T] = \dim[Y|x_T]$ as required. □

It follows from Lemma 3 that the function f_M defined in Theorem 2 also satisfies the *privacy* requirement with respect to any collusion $T \subseteq [n]$, concluding the proof of Theorem 2.

The (fully-private) solution for additively sharing a zero, as described in the previous subsection, may be derived as a special case of Theorem 2. Another useful special case is a secret-sharing of zero using Shamir's scheme.

Corollary 1. *Let D be the distribution induced by sharing 0 according to Shamir's secret-sharing scheme with parameters $(t+1, n)$ over a finite field F ($|F| > n$). That is, D is generated by picking a random polynomial p over F of degree$\leq t$ and free coefficient 0, and outputting the n-tuple $(p(\alpha_1), \ldots, p(\alpha_n))$ (where $\alpha_1, \ldots, \alpha_n$ are some fixed distinct nonzero field elements). Then, D can be privately generated from replication with $m = \binom{n}{t-1}$, where each of the m independent field elements is sent to $n - t + 1$ players.*

Proof. It is not hard to observe that a distribution D as above is uniform over a linear code C such that $\mathrm{Min}(C)$ includes $\binom{n}{t-1}$ codewords, each of Hamming weight $n - t + 1$. More generally, the same is true for any MDS code (cf. [19]) of length n and dimension t. □

We end this section with a couple of remarks about the above construction.

Remarks:

1. **On optimality.** The construction of the correlation generator f_M in Theorem 2 can be shown to be optimal (under the correlation from replication paradigm), with respect to the communication complexity of the induced protocol. The proof of this fact may proceed similarly to the arguments made in the previous subsection for the special case of additively sharing a zero, and is omitted from this version.
2. **On generalization to $\mathcal{T}$-privacy.** The above solution can be made more efficient if the full privacy requirement is relaxed to $\mathcal{T}$-privacy. In fact, it is possible to derive from the proof of Theorem 2 a generalization of

Theorem 1, providing a necessary and sufficient condition on a subset of the columns of M so that the corresponding submatrix M' will $\mathcal{T}$-privately generate the distribution D_C. Specifically, the relaxed privacy requirement should assert that for every $T \in \mathcal{T}$, $\dim C_{\overline{T}} = \operatorname{rank} M'_{\overline{T}}$, where $C_{\overline{T}}$ and $M'_{\overline{T}}$ are as defined in the proof of Lemma 3.

4 The Computational Setting: Privately Correlated Pseudo-Random Generators

In this section we exploit private correlation generators constructed in the previous section for implementing "privately-correlated pseudo-random generators". Intuitively, such generators should allow to *securely substitute in applications* long correlated perfectly-random pads by correlated pseudo-random pads, where these "pseudo-pads" are generated by locally uncompressing short seeds sent by the dealer. While the results shown in this section are not particularly strong, clean, nor elegant, we view them as an important demonstration of plausibility. We start by addressing the delicate issue of formally defining the desired computational notion.

A first (and technical) difficulty with incorporating the results of the previous section in a computational setting is the fact that for certain linear correlation patterns, the complexity of their generation from replication is exponential in n. (We stress though that for some useful correlations, such as additive shares of zero, this is not the case.) For simplicity, throughout the remainder of this section we fix a linear distribution D (which in particular determines the number of players n and the size of the field F to be constants), and treat the number of generated instances of D (corresponding to the length of the generated pads) as the major complexity parameter. While this "resolves" the above technical difficulty, a more inherent difficulty is discussed below.

The previous section described a procedure for privately generating a joint distribution D of n correlated pads from m independent and replicated random sources. Such a procedure may be defined using a more general terminology of private multi-party computation (cf. [8,1,20]) as follows. We consider a simple model consisting of n parties, connected to a trusted dealer via secure channels, and a passive, static $\mathcal{T}$-adversary. The function f is a $\mathcal{T}$-private generator for the distribution D (as defined in Definition 1) if it induces a *perfectly* $\mathcal{T}$-private protocol computing the *probabilistic* n-output function that takes no inputs and outputs an n-tuple y distributed according to D.[9] Now, a natural way to define the notion of privately-correlated pseudo-random generators using general terminology of private multi-party computation is the following. Let ℓ denote the number of *repetitions* of the distribution D to be generated. The parameter ℓ, as well as a security parameter κ, will be given as inputs to all players. Moreover, to

[9] To be consistent with the usual computationally-oriented definitions of private multi-party computation found in the literature, one also needs to add an extra requirement of efficient simulatability to Definition 1. However, for constant n (and arbitrarily large F) our solutions for linear correlations are easily seen to satisfy this extra requirement.

simplify definitions we assume that $\ell = p(\kappa)$ for some polynomial $p(\kappa) = \omega(k)$; that is, the length of the generated pads is some fixed superlinear polynomial of the security parameter. The distribution D on F^n induces a distribution D^ℓ on $(F^\ell)^n$, generated by ℓ independent repetitions of D. Notice that the information theoretic constructions of the previous section easily generalize from privately generating a distribution D to privately generating D^ℓ. Using the above notation, we would like to define a privately-correlated pseudo-random generator as a *computationally private* protocol[10] computing the probabilistic function outputting D^ℓ (with no input), or some distribution which is indistinguishable from D^ℓ, using *little communication*. Specifically, we restrict the length of messages sent from the dealer to the players to be $O(\kappa)$.

Such a definition would not only be natural but also quite useful, since it gives hope to allow automatically plugging pseudo-pads in every application that uses perfect pads. However, even under our relatively benign adversary model, such a definition would be impossible to meet. To illustrate why this is the case, let $n = 2$ and consider the simple distribution outputting a pair of identical random strings. Intuitively, any low-communication protocol for computing this distribution will reveal to P_1 a "short explanation" for the output of P_2 (which is with overwhelming probability identical to his own output), whereas if this information could be simulated based on the output of P_1 alone then one could distinguish the output of P_1 from a truly random output. In the example of additively sharing a zero, members of a collusion T will obtain a short explanation for the sum of the outputs of the remaining players (which again, cannot be simulated by viewing the T-outputs alone).

For the reason discussed above, we avoid specifying an explicit definition of the general computational primitive and replace it with an *ad hoc* definition. Namely, we define a privately-correlated pseudo-random generator with respect to a protocol or a class of protocols.

Definition 2. *Let $\mathcal{P}$ be a (perfectly or computationally) $\mathcal{T}$-private protocol consisting of the following two stages:*

Off-line stage: *The dealer sends (over secure channels) to each player P_i the i-th component of a random instance $y^\ell \leftarrow D^\ell$. After this stage the dealer vanishes.*

On-line stage: *The players interact, using their "long" random pads y^ℓ.*

Now, let $\mathcal{G}$ be a protocol between the dealer and the players (presumably outputting some pseudo-pads approximating D^ℓ), with total communication $O(\kappa)$.

We say that $\mathcal{G}$ is a privately-correlated pseudo-random generator (PCPRG) *for distribution D^ℓ with respect to the protocol $\mathcal{P}$ if the protocol $\tilde{\mathcal{P}}$ obtained by replacing the off-line stage in $\mathcal{P}$ by $\mathcal{G}$ is* computationally $\mathcal{T}$*-private. We say that $\mathcal{G}$ is a PCPRG for D^ℓ with respect to a protocol class $\mathcal{C}$, if it is a PCPRG with respect to every $\mathcal{P} \in \mathcal{C}$.*

Using the above terminology, an ordinary pseudo-random generator may be viewed as a PCPRG in which the dealer sends a κ-bit seed to one player, who locally expands it to an ℓ-bit pseudo-random string.

[10] See Appendix A for a definition of a computationally private protocol in our setting.

We now use the results of the previous section to construct, for any linear correlation D_C, a PCPRG $\mathcal{G}_C$ with respect to a wide class of protocols meeting some mild technical requirement.

Our PCPRG $\mathcal{G}_C$ for a linear correlation D_C is defined in a straightforward manner using the function f_M constructed in Theorem 2. Specifically, the protocol proceeds as follows. Let $G : \{0,1\}^\kappa \to F^\ell$ be any pseudo-random generator (i.e., the output of this generator is indistinguishable from a random ℓ-tuple over F). Let S denote a uniform distribution over $(\{0,1\}^\kappa)^m$ (S represents a uniform distribution of m independent κ-bit seeds), and let $f = f_M$. In the protocol $\mathcal{G}_C$, the dealer first generates a seed vector $s \leftarrow S$, and sends each seed s_j to all players P_i with $j \in S_i$ (where the sets S_i are determined by the information theoretic generator f as in the previous section). Then, each player P_i evaluates $\tilde{y}_i \stackrel{\text{def}}{=} f_i(G(s))$, where $G(s) \stackrel{\text{def}}{=} (G(s_1), \ldots, G(s_m))$, using the seeds it received. The ℓ-tuple $\tilde{y}_i \in F^\ell$ is used as P_i's pseudo-pad.

The next step is to define a general class of protocols $\mathcal{P}$ for which $\mathcal{G}_C$ is a PCPRG generating D_C. The following definition will use the notion of *simulator* as used in definitions of private protocols. We refer the reader to Appendix A for definitions of the relevant notions.

For any perfectly $\mathcal{T}$-private protocol $\mathcal{P}$, such that in the off-line stage of $\mathcal{P}$ the dealer distributes pads generated according to D_C^ℓ, we define a protocol $\mathcal{P}'$ obtained by replacing the original off-line stage by the distribution procedure induced by a corresponding $\mathcal{T}$-private correlation from replication generator f_M. (It can be shown that in our setting such $\mathcal{P}'$ is also perfectly $\mathcal{T}$-private).

Definition 3. *We say that the protocol $\mathcal{P}'$ admits a* strong simulator*, if for every adversary $\mathcal{A}'$ there exists a strong simulator $\mathcal{S}'$ following the following two-step procedure:*

- *Generates,* independently of its view of the inputs and the outputs, *a uniformly random x_T^ℓ, representing pads received from the dealer in $\mathcal{P}'$, and outputs (the appropriate replication of) x_T^ℓ as the corresponding part of the simulated view;*
- *Proceeds arbitrarily, possibly relying on the generated view x_T^ℓ in addition to its view of the inputs and outputs.*

Theorem 3. *For any linear correlation D_C, and any protocol $\mathcal{P}$ using D_C such that $\mathcal{P}'$ admits a strong simulator, $\mathcal{G}_C$ is a PCPRG with respect to $\mathcal{P}$.*

Proof. See Appendix B. □

We stress that the existence of a strong simulator in the sense of the above definition is a very mild requirement on perfectly private protocols. For instance, the simple information theoretic protocol for modular addition described in the introduction, as well as much more complicated protocols such as an implementation of [5] using linearly correlated pads, satisfy this requirement.

To see why a restriction on the protocol $\mathcal{P}$ is at all necessary, consider a simple perfectly-private protocol $\mathcal{P}$ in which the dealer first additively shares a long zero among the players, and then each player outputs his share. This protocols

computes, with perfect and full privacy, the *probabilistic* function outputting some distribution D_C (namely, additive sharing of zero). However, the corresponding protocol $\mathcal{G}_C$ is *not* a PCPRG with respect to $\mathcal{P}$. (Intuitively, members of T can generate a "short explanation", which cannot be efficiently simulated, for the *sum* of the outputs of the other players.) Indeed, the above protocol $\mathcal{P}$ does not admit a strong simulator, since the messages viewed by the adversary during the off-line stage depend on the adversary's view of the output.

We would like to point out a few possible extensions of the results of this section. First, pseudo-random generators can be replaced by pseudo-random functions, thereby obtaining a stronger notion of *privately-correlated pseudo-random functions*. Second, the dealer may be simulated by the players, resulting in private protocols under a more standard dealer-less model. In the case of a passive adversary, this can be done by picking for each seed distributed by the dealer a single player from the set of players who should receive it, and letting this player generate this seed and multicast it to the appropriate set of players. Finally, while the framework discussed in this section does not apply *as is* to the proactive secret-sharing application described in the introduction, our compression technique for linear correlations can be securely used in such context.

5 Concluding Remarks

While the correlation from replication paradigm seems to be the most general one could expect when pseudo-randomness is used in a black-box manner, an interesting open question is that of obtaining more efficient *direct* constructions for linear correlations and other useful types of correlations. A particularly interesting open problem is that of compressing (non-linear) correlations of the type used in [3] for allowing efficient 2-party oblivious transfer (OT) [25,11].[11]

Acknowledgments. We wish to thank two anonymous referees for their helpful suggestions and comments. We also thank Hugo Krawczyk, Eyal Kushilevitz, and Moni Naor for related discussions.

References

1. D. Beaver. Foundations of secure interactive computing. In *Proceedings of CRYPTO*, 1991.
2. D. Beaver. Correlated pseudorandomness and the complexity of private computation. In *Proc. of 28th STOC*, pages 479–488, 1996.
3. D. Beaver. Commodity-based cryptography. In *Proc. of 29th STOC*, pages 446–455, 1997.

[11] A seemingly related problem was studied in [2], where it is shown that an initial "seed" of κ OT's suffices to generate $O(\kappa^c)$ OT's using one-way functions alone; however, the main concern of [2] is that of weakening the required cryptographic assumptions rather than reducing the communication complexity of this task.

4. D. Beaver and A. Wool. Quorum-based secure multi-party computation. In *Proc. of EUROCRYPT'98, LNCS 1403, Springer Verlag*, pages 375–390, 1998.
5. M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proc. of 20th STOC*, pages 1–10, 1988.
6. M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal on Computing*, 13(4):850–864, 1984.
7. B. Bollobás. *Graph Theory.* Springer-Verlag, 1979.
8. R. Canetti. Security and composition of multi-party cryptographic protocols. Theory of Cryptography Library, Record 98-18, 1998.
9. D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols (extended abstract). In *Proc. of 20th STOC*, pages 11–19, 1988.
10. B. Chor and E. Kushilevitz. A communication-privacy tradeoff for modular addition. *Information Processing Letters*, 45(4):205–210, March 1993.
11. S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *Comm. of ACM*, 28:637–647, 1985.
12. O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *JACM*, 33(4):792–807, October 1986.
13. A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung. Proactive secret sharing or: How to cope with perpetual leakage. In *Proceedings of CRYPTO*, 1995.
14. M. Hirt and U. Maurer. Complete characterization of adversaries tolerable in secure multi-party computation (extended abstract). In *Proceedings of the Sixteenth Annual ACM Symposium on Principles of Distributed Computing*, pages 25–34, 1997.
15. Y. Ishai and E. Kushilevitz. Improved upper bound on information theoretic private information retrieval. In *Proc. of 31th STOC*, 1999.
16. 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.
17. H. Krawczyk. Secret sharing made short. In *Proceedings of CRYPTO*, 1994.
18. M. Luby. *Pseudorandomness and Cryptographic Applications.* Princeton University Press, 1996.
19. F.J. Macwilliams and N.J. Sloane. *The theory of error correcting codes.* North Holland, 1978.
20. S. Micali and P. Rogaway. Secure computation. In *Proceedings of CRYPTO*, 1991.
21. S. Micali and R. Sidney. A simple method for generating and sharing pseudo-random functions with applications to clipper-like key escrow systems. In *Advances in Cryptology - CRYPTO '95*, pages 185–196, 1995.
22. 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.
23. R. Ostrovsky and V. Shoup. Private information storage. In *Proc. of the 29th Annu. ACM Symp. on the Theory of Computing*, pages 294–303, 1997.
24. R. Ostrovsky and M. Yung. How to withstand mobile virus attacks. In *Proc. 10th ACM Symp. on Principles of Distributed Computation*, pages 51–61, 1991.
25. M. O. Rabin. How to exchange secrets by oblivious transfer. Technical Report TR-81, Harvard Aiken Computation Laboratory, 1981.
26. T. Rabin. A simplified approach to threshold and proactive RSA. In *Proceedings of CRYPTO*, 1998.
27. A. Shamir. How to share a secret. *Commun. ACM*, 22(6):612–613, June 1979.
28. A. C. Yao. Theory and application of trapdoor functions. In *Proc. of the 23th Annu. IEEE Symp. on Foundations of Computer Science*, pages 80–91, 1982.

A Definitions of Private Computation

In the following we outline definitions of perfect (aka information theoretic) and computational privacy of protocols in our minimalistic model. We refer the interested reader to, e.g., [8] for more general and detailed definitions.

We consider a passive, static adversary, corrupting at the beginning of the computation some set of players $T \in \mathcal{T}$ (denoted T-players). The view of such an adversary consists of the entire joint view of the T-players. That is, this view includes the inputs and random tapes of the T-players, the messages exchanged between the dealer and T-players during the off-line distribution stage, and *all* messages exchanged during the on-line stage.[12]

We let a denote an input n-tuple, and g denote a (deterministic or probabilistic, single- or multi-output) function mapping the input into an output n-tuple denoted z. Notice that if g is deterministic, then a determines z. Otherwise, every input a induces some probability distribution on the output z. For every n-tuple v, v_T denotes restriction of v to its T-entries.

Let $\mathcal{P}$ be an n-party protocol (presumably computing g). We consider two scenarios. In the *real-life scenario*, an adversary $\mathcal{A}$ is exposed to the view described above induced by an execution of $\mathcal{P}$ on some input a. In the *ideal-model scenario*, an output $z \leftarrow g(a)$ is picked, and a *simulator* $\mathcal{S}$, viewing only a_T, z_T (modeling what the adversary *inevitably* learns), has to efficiently "simulate" the real-life view of $\mathcal{A}$ given the unknown input a and output z. We denote by $\mathbf{exec}^{\mathcal{A}}(a)$ the random variable which includes the view of the adversary $\mathcal{A}$ on input a, concatenated with the outputs of the remaining players.[13] Similarly, by $\mathbf{exec}^{\mathcal{S}}(a)$ we denote the random variable including the simulated output of $\mathcal{S}$ (on inputs a_T, z_T) along with the outputs $z_{\overline{T}}$.

We say that $\mathcal{P}$ is a *perfectly (computationally) $\mathcal{T}$-private protocol computing* g if for every efficient adversary $\mathcal{A}$ (corrupting some set of players $T \in \mathcal{T}$), there exists an efficient simulator $\mathcal{S}$ (corrupting the same set of players), such that the probability ensembles $\{\mathbf{exec}^{\mathcal{S}}(a)\}_{a \in A}$ and $\{\mathbf{exec}^{\mathcal{A}}(a)\}_{a \in A}$, are identically distributed (resp., computationally indistinguishable), where A is the (infinite) set of all input n-tuples whose n inputs are of the same length.[14]

B Proof Sketch of Theorem 3

Recall that the protocol $\tilde{\mathcal{P}}$ is the protocol obtained by replacing the off-line stage of $\mathcal{P}$ or $\mathcal{P}'$ by the protocol $\mathcal{G}_C$. We prove that if $\mathcal{P}'$ admits a strong simulator, then $\tilde{\mathcal{P}}$ is computationally private.

[12] The assumption that all on-line messages are broadcasted does not compromise generality because of the existence of private channels in the off-line stage.

[13] Setting $T = \emptyset$, this concatenation will capture the *correctness* requirement. In the case of multi-output functions, it may also capture the adversary's possibility of learning computational information about the outputs of the remaining players.

[14] Note that this definition effectively restricts all inputs to be of the same length and uses the input length as the security parameter.

We define the following random variables for the various protocols involved. By X we denote the independent messages sent by the dealer in $\mathcal{P}'$ (X is uniformly distributed over $(F^\ell)^m$).[15] By X_T^c we denote the complement of X_T, i.e. the restriction of X to the entries *not* viewed by players in T (notice that X_T^c is not the same as $X_{\overline{T}}$), and similarly for other m-tuples. By S we denote the seeds sent in $\tilde{\mathcal{P}}$ (uniformly distributed over $(\{0,1\}^\kappa)^m$) and by $\tilde{X}$ the m-tuple $G(S)$ (where G is the pseudo-random generator used by $\mathcal{G}_C$).

Recall that a strong simulator $\mathcal{S}'$ for an adversary $\mathcal{A}'$ attacking protocol $\mathcal{P}'$ proceeds as follows. On input a_T, z_T it first outputs a uniformly random x_T (independently of its inputs), and then *exactly* generates the distribution of the remaining simulated view conditioned by a, z, x_T (where the probability space of the corresponding conditional view of $\mathcal{A}'$ in the real-life protocol is over X_T^c and the coin-tosses of all players). We now construct a simulator $\tilde{\mathcal{S}}$ for an adversary $\tilde{\mathcal{A}}$ attacking the protocol $\tilde{\mathcal{P}}$. Simulator $\tilde{\mathcal{S}}$, on input a_T, z_T, proceeds as follows:

- generates random seeds s_T;
- evaluates $\tilde{x}_T = G(s_T)$ and outputs an appropriate replication of $\tilde{x}_T$ as the simulation of the off-line stage;
- invokes the second phase of the strong simulator $\mathcal{S}'$ on inputs $a_T, z_T, \tilde{x}_T$, and outputs the remaining simulated view.

To prove the validity of the simulator $\tilde{\mathcal{S}}$ constructed above, it may be conceptually convenient to define an intermediate protocol $\tilde{\mathcal{P}}'$ between $\tilde{\mathcal{P}}$ and $\mathcal{P}'$, obtained by replacing the pseudo-random strings $\tilde{X}_T^c$ with perfectly random strings of the same length, X_T^c. Note that by a standard hybrid argument, X_T^c is computationally indistinguishable from $\tilde{X}_T^c$. We prove that a distinguisher $\mathcal{D}$ between $\{\mathbf{exec}^{\tilde{\mathcal{S}}}(a)\}$ and $\{\mathbf{exec}^{\tilde{\mathcal{A}}}(a)\}$ can be turned into a distinguisher between $\{X_T^c\}$ and $\{\tilde{X}_T^c\}$. For any input a, construct a distinguisher $\hat{\mathcal{D}}_a$ as follows. On input q, sampled either according to X_T^c or $\tilde{X}_T^c$:

- generate random seeds s_T and evaluate $\tilde{x}_T = G(s_T)$;
- invoke the on-line stage of the protocol $\mathcal{P}'$ on input a, using $f(\tilde{x}_T, q)$ as correlated pads, and let v_T denote the entire view of the T-players (including a_T and the seeds s_T) and z denote the protocol's outputs;
- invoke the distinguisher $\mathcal{D}$ on $\mathbf{exec} = (v_T, z_{\overline{T}})$.

It remains to make the following key observation: If q is distributed according to X_T^c, then the random variable $\mathbf{exec}$ is distributed *identically* to the real-life view in the intermediate protocol $\tilde{\mathcal{P}}'$, which in turn is identical to $\mathbf{exec}^{\tilde{\mathcal{S}}}(a)$ (otherwise one could contradict the perfect simulation of $\mathcal{S}'$); on the other hand, if q is distributed according to $\tilde{X}_T^c$, then $\mathbf{exec}$ is distributed identically to $\mathbf{exec}^{\tilde{\mathcal{A}}}(a)$, the real-life view in $\tilde{\mathcal{P}}$. Hence, if there exists an infinite sequence of a's on which $\mathbf{exec}^{\tilde{\mathcal{S}}}(a)$ and $\mathbf{exec}^{\tilde{\mathcal{A}}}(a)$ are distinguished by $\mathcal{D}$, we may use $\hat{\mathcal{D}}_a$ (with the same sequence of a's) to derive the desired contradiction to the indistinguishability of X_T^c and $\tilde{X}_T^c$. □

[15] The notation X^ℓ would be more appropriate than X in terms of consistency with previous notation. We omit the superscripts ℓ to avoid further notation overload.