# On Two Round Rerunnable MPC Protocols

Paul Laird

Dublin Institute of Technology, Dublin, Ireland
email: {paul.laird}@dit.ie

**Abstract.** Two-rounds are minimal for all MPC protocols in the absence of a trusted PKI, however certain protocols allow the reuse of inputs for different functions, or the re-evaluation of the same function on different inputs without the re-distribution of public key information. These can achieve an amortised round complexity of below two rounds per computation. Function rerunnable MPC has been achieved using FHE, while additive homomorphic properties of DH-based cryptosystems have been used to allow input rerunnable protocols. These differ in properties such as computational cost per execution, collusion tolerance and number of rounds supported. We discuss the characteristics of some rerunnable protocols, and present a proof of the rerunnable aggregation protocol of Kursawe, Danezis and Katz from the Decisional Bilinear Diffie Hellman Assumption [3].

**Key words:** multiparty computation, private aggregation, cryptographic protocols

## 1 Introduction

Minimising the round complexity of multiparty computations has long been a goal of research into MPC[1], with recent developments including optimal 2 round general MPC under the LWE assumption [6]. This enables arbitrary function rerunnable MPC, as arbitrary computations may be carried out on ciphertexts encrypted under a MKFHE scheme [5]. Input rerunnable MPC remains somewhat less researched, with protocols existing for aggregation and certain functions such as inner products. Kursawe, Danezis and Katz presented a protocol for rerunnable privacy-preserving summation based on pairings, and a two round protocol based on standard Diffie-Hellman [4], which were used as the basis of a limited rerunnable DH-based protocol and a rerunnable hybrid protocol using pairings less frequently for efficiency reasons, both of which trade collusion tolerance for efficiency [7]. The security of the pairings-based protocol of Kursawe, Danezis and Katz, and the related hybrid protocol has not previously been demonstrated. We present some notation, following [7], introduce the two-round protocol of Kursawe, Danezis and Katz, followed by its rerunnable extension using pairings, and comment on its extension to the malicious setting, which incurs an overhead only in the first run, and is thus amortised over multiple runs. Performance data for the pairings-based protocol of Kursawe, Danezis and

Katz, as well as the limited rerunnable DH-based protocol and hybrid protocol are available in [7].

## 2 Preliminaries

### 2.1 Notation and definitions

Let $k$ be an integer. We denote the contiguous set of integers $\{1, \ldots, k\}$ by $[k]$. Let $X$ and $Y$ be distributions. The notation $X \underset{C}{\approx} Y$ denotes the fact that both distributions are computationally indistinguishable to any probabilistic polynomial time (PPT) algorithm.

In order to show that the proposed protocol provides the necessary privacy to the participants, we have to show that it provides privacy against collusions of up to $t$ users. Intuitively, suppose $n-2$ users collude, then it should not be possible for the colluding users to learn anything about the 2 honest users' inputs beyond their sum. If $n-1$ users collude, then we expect them to learn the honest party's input. So for the case of $n-1 \leq t \leq n$, there is no privacy requirement, and thus these trivial cases are easily handled in meeting our security definition below.

We adopt the standard simulation-based definition of security in the semi-honest model with static adversaries. We base our definition below on Definition 2.1 in [2]. Here we consider only computational security, and relax the more standard definition to deterministic functionalities with a single output, since this paper is concerned with aggregation. Note that this definition is general enough to accommodate multi-aggregation aggregation as provided by our protocol.

Let $\boldsymbol{m} \in (\{0,1\}^*)^n$ be a vector of the inputs from each party and let $\pi$ be a protocol. We define $\mathsf{OUTPUT}^{\pi}(m_1, \ldots, m_n)$ as the final aggregated result computed with protocol $\pi$ from the input vector $\boldsymbol{m}$. Furthermore, we define the view of a party $P_i$ in the execution of protocol $\pi$ with input vector $\boldsymbol{m}$ as

$$\mathsf{VIEW}_i^{\pi}(\boldsymbol{m}) = (m_i, r_i, \mu_i^{(1)}, \ldots, \mu_i^{(\ell)})$$

where $m_i$ is party $P_i's$ input, $r_i$ is its random coins and $\mu_i^{(1)}, \ldots, \mu_i^{(\ell)}$ are the $\ell$ protocol messages it received during the protocol execution. Similarly, the combined view of a set of $I \subseteq \{1, \ldots, n\}$ parties is denoted by $\mathsf{VIEW}_I^{\pi}(\boldsymbol{x})$.

**Definition 1 (privacy of $n$-party protocols for deterministic aggregation functionalities).** *Let $f : (\{0,1\}^*)^n \to (\{0,1\}^*)$ be a deterministic $n$-ary functionality and let $\pi$ be a protocol. We say that $\pi$ privately computes $f$ if for every $\boldsymbol{m} \in (\{0,1\}^*)^n$ where $|m_1| = \ldots = |m_n|$,*

$$\mathsf{OUTPUT}^{\pi}(m_1, \ldots, m_n) = f(m_1, \ldots, m_n) \tag{1}$$

*and there exists a PPT algorithm $\mathcal{S}$ such that for every $I \subset [n]$ with $|I| \leq t$, and every $\boldsymbol{m} \in (\{0,1\})^n$ where $|m_1| = \ldots = |m_n|$, it holds that:*

$$\{\mathsf{VIEW}_I^{\pi}(\boldsymbol{m})\} \underset{C}{\approx} \{\mathcal{S}(I, \boldsymbol{m}_I, f(\boldsymbol{m}))\}. \tag{2}$$

### 2.2 Single-Aggregation KDK

Kursawe, Danezis and Kohlweiss (KDK) [4] present a specialized multiparty computation (MPC) protocol for private summation, along with a single-run 2-round concrete instantiation which is shown to be secure in the semi-honest model under the Decisional Diffie-Hellman (DDH) assumption. We refer to this protocol as KDK. In their protocol, $n$ parties $P_1, \ldots, P_n$ can compute a joint sum of their inputs $m_1, \ldots, m_n \in \{0, \ldots, \beta\}$ for some positive integer $\beta$. An overview of their protocol follows.

Let $p$ be a prime. The "public parameters" used in the protocol consist of a description of a cyclic group $\mathbb{G}$ of order $p$ together with a generator $g$ of $\mathbb{G}$. It is assumed that DDH is intractable in $\mathbb{G}$. These public parameters $\mathsf{PP} = (\mathbb{G}, g, p)$ are known to all parties $P_i$. The group operation of $\mathbb{G}$ is written multiplicatively.

1. **Setup:** Party $P_i$ generates a secret key $x_i \in \mathbb{Z}_p$ and computes her public key $u_i = g^{x_i} \in \mathbb{G}$. She broadcasts $u_i$.
2. **Main Round:**
   - Party $P_i$ chooses her input $m_i \in \{0, \ldots, \beta\}$.
   - Compute $w \leftarrow \prod_{j \in 1}^{i-1} u_j^{-1} \cdot \prod_{j \in i+1}^{n} u_j \in \mathbb{G}$.
   - Broadcast $v_i \leftarrow w^{x_i} \cdot g^{m_i} \in \mathbb{G}$.
3. **Output:** The protocol produces an output in $\{0, \ldots, n\beta\}$, namely the sum of the user inputs. To compute the sum $\sigma$:
   - Compute $z \leftarrow \prod_{j=1}^{n} v_j$.
   - Use Pollard's Lambda algorithm to compute the discrete log $\sigma \in \{0, \ldots, n\beta\}$ of $z$ with respect to $g$ in $G$.
     The time complexity of Pollard's lambda algorithm is $\sqrt{n\beta}$, or lower if there is a region of the solution space where the result is known to lie with high probability (such as within the error bounds of a substation power consumption reading for the area containing smart meters, whose actual readings are being aggregated.
   - Output $\sigma$.

It can be easily observed that $\prod_{j=1}^{n} v_j = g^{\sum_{j=1}^{n} m_j}$

### 2.3 Multi-Aggregation KDK (MA-KDK)

If the protocol must be run a number of times, it would be desirable to avoid re-running the "Setup" phase above which involves each party generating and broadcasting a new public key; in practice, a verification step for these keys may also be needed, further increasing the round complexity. To re-use the published keys $u_i, \ldots, u_n$ for more than a single round of aggregation, Kursawe et al. propose a multi-aggregation protocol accommodates an unbounded number of aggregations, using bilinear pairings to achieve this.

Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be cyclic groups of prime order $p$. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a cryptographic bilinear pairing. Furthermore, the Bilinear Decisional Diffie Hellman (BDDH) assumption is expected to hold with respect to $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ and $e$. Let $H : \mathbb{Z} \to \mathbb{G}_2$ be a hash function. The main changes to KDK to support multiple rounds are as follows (optimizations are discussed later):

– The public parameters include generators $P \in \mathbb{G}_1, Q \in \mathbb{G}_2$ and $g = e(P, Q) \in \mathbb{G}_T$.
– The public keys are generated as $U_i \leftarrow x_i P \in \mathbb{G}_1$ for all $1 \leq i \leq n$.
– In aggregation $k$, party $P_i$ computes
  • $Q_k \leftarrow H(k) \in \mathbb{G}_2$ (i.e. for a good choice of $H$, we have $Q_k = rQ$ for some uniformly random $r$, which is intractable to find).
  • $w \leftarrow \prod_{j=1}^{i-1} e(U_j, Q_k)^{-1} \cdot \prod_{j=i+1}^{n} e(U_j, Q_k) \in \mathbb{G}_T$.

The rest of the protocol remains unchanged except that the computations are performed in $\mathbb{G}_T$, and $P_i$ may choose a different input value in every round. Naturally, the output of the protocol is then $(\sigma_1, \ldots, \sigma_\ell) \in \{0, \ldots, n\beta\}^\ell$ if $\ell$ rounds are executed.

Note that there is no requirement for $P \neq Q$ or $\mathbb{G}_1 \neq \mathbb{G}_2$ hence we provide a proof of security where $P = Q$; $\mathbb{G}_1 = \mathbb{G}_2; g = e(P, P) \in \mathbb{G}_T; Q_k = rP$ for uniformly random r. This allows a more straightforward mapping to the DBDH as described in [3].

**Theorem 1.** *Under the DBDH assumption, the KDK multi-aggregation protocol is computationally private in the random oracle model.*

The proof of security of MA-KDK is not presented in the KDK paper, and is outlined in the following section.

## 3 Proof of Theorem 1

**Theorem 1.** *Under the DBDH assumption, the KDK multi-aggregation protocol is computationally private in the random oracle model.*

*Proof.* Let $h = n - t$ be the number of honest users. If $h \leq 1$, it is trivial to construct a simulator $\mathcal{S}$ since $\mathcal{S}$ can fully learn $\boldsymbol{m}$ and then simulate all parties. Therefore, we assume that $h \geq 2$. Let $w = h(h-1)/2$. Let $q$ be the number of executions of the protocol. For each round $\rho$, the random oracle is invoked so that, for common string $s_\rho$ representing the execution instance (such as the time), $Q_\rho \leftarrow H(s_\rho) \in \mathbb{G}_2$ (i.e. we have $Q_\rho = r_\rho P$ for some uniformly random $r_\rho$, which is intractable to find). Consider the following series of Hybrids.
**Hybrid 0:** This is the same as the real distribution i.e. the LHS of Equation 2 with the exception that we "simulate" each honest party $P_k$ using input $m_k^{(\rho)}$ in round $\rho$; therefore we have access to $x_k$.

For $1 \leq s \leq w$: Hybrid $s$ involves two honest parties which we denote by $P_i$ and $P_j$, in round $\rho$. Their equations share the monomial $x_i x_j r_\rho$. There are $wq$ such monomials and the goal of each Hybrid $s$ is to replace the $s$-th monomial with a uniformly random element.
**Hybrid s:** The changes between Hybrid $s$ and Hybrid $s-1$ involve changing the protocol messages of the honest parties $P_i$ and $P_j$ in round $\rho$. Let $m_i^{(\rho)}$ and $m_j^{(\rho)}$ be the inputs of these honest parties in round $\rho$. Generate a uniformly random

integer $a_s \in \{0, \ldots, p-1\}$ and replace all occurrences of $g^{x_i x_j r_\rho}$ by $g^{a_s}$ in the computation of the second message by $P_i$ and $P_j$ in round $\rho$.

Hybrid $s-1$ and Hybrid $s$ are computationally indistinguishable under the DBDH assumption[3].
Hybrid $s-1$ involves the DBDH instance $(P, x_iP, x_jP, r_\rho P, e(P,P)^{x_i x_j r_\rho})$ and Hybrid $s$ involves the DBDH instance $(P, x_iP, x_jP, r_\rho P, e(P,P)^{a_s})$ where $x_i, x_j, r_\rho$ and $a_s$ are uniformly distributed in $\{0, \ldots, p-1\}$. A non-negligible advantage distinguishing between Hybrid 0 and Hybrid 1 implies a non-negligible advantage against DBDH.

$H$ is modelled as a random oracle and as such $r_\rho$ will be a uniformly random element in $\mathbb{Z}_p$ for each round $\rho$. For each DBDH tuple $(P, x_iP, x_jP, r_\rho P, e(P,P)^{x_i x_j r_\rho})$, there are several tuples which differ in just one of $\{x_iP, x_jP, r_\rho P\}$, and for which $e(P,P)^{x_i x_j r_\rho}$ may be known (such as $e(P,P)^{x_d x_j r_\rho}$ for dishonest party $P_d$'s $x_d$, used in computing $v_d$, however these do not confer any non-negligible advantage in distinguishing $(P, x_iP, x_jP, r_\rho P, e(P,P)^{x_i x_j r_\rho})$ and $(P, x_iP, x_jP, r_\rho P, e(P,P)^{a_s})$ under DBDH as otherwise an adversary could sample some uniformly random $b \in \mathbb{Z}_p$, evaluate $bP$ and $e(x_iP, x_jP)^b$ for a verified tuple $(P, x_iP, x_jP, bP, e(P,P)^{x_i x_j b})$, and similarly replace $x_i$ or $x_j$, and thus gain an advantage in DBDH.

**Hybrid wq+1:** Without loss of generality, assume that parties $P_1, \ldots, P_h$ are the honest parties. For all $1 \leq i < h$ and $1 \leq \rho \leq q$, replace the protocol message $v_{i,(\rho)}$ of party $P_i$ in aggregation $\rho$ with $g^{R_{i,(\rho)}} \cdot g^{m_{i,(\rho)}}$ for uniformly random $R_{i,(\rho)} \in \mathbb{Z}_p$. Furthermore, for every $1 \leq \rho \leq q$, replace the protocol message $v_{h,(\rho)}$ with $g^{-\sum_{j=1}^{h-1} r_{j,(\rho)} + m_{h,(\rho)}}$. As the replacement of each term shared by two honest parties with uniformly random values cannot be distinguished without an advantage in DBDH, nor can an adversary distinguish the product of such terms and a single uniformly random value. Distinguishing Hybrid $wq+1$ and Hybrid $wq$ is therefore impossible.

**Hybrid wq + 2** Finally, in this Hybrid, the inputs $m_{1,(\rho)}, \ldots, m_{h,(\rho)}$ are replaced by a random partition of $\sum_{k=1}^{h} m_{k,(\rho)}$, namely the values $s_{1,(\rho)}, \ldots, s_{h,(\rho)}$ for every $\rho \in \{1, \ldots, q\}$.

An adversary has a zero advantage distinguishing Hybrid $wq+2$ and Hybrid $wq+1$. To see this, suppose the adversary could distinguish the hybrids. Then it can determine that some party's input (say $P_i$) in some aggregation $\rho$ is not $s_{i,(\rho)}$. But $vi,(\rho) = g^{r'}$ for some uniformly random $r'$, which provides no information about the message (whether it is $m_{i,(\rho)}$ or $s_{i,(\rho)}$). Note that $v_{h,(\rho)}$ gives no additional information since it can be derived from the information known to the adversary (recall that the sum in each aggregation is known).

Since Hybrid $wq+2$ no longer relies on the honest parties' messages, and all other information needed to construct the distribution can be derived from the simulators' inputs in Equation 2, it follows that there exists an algorithm $\mathcal{S}$ that can simulate the real distribution. □

## 4 Privacy in the Malicious Setting

We only give a brief overview here of how to prove privacy in the presence of malicious adversaries.The protocol uses a NIZK argument system $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ for statements of the form $S_i = \{(x_i) : u_i = x_i P\}$. The common reference string $\sigma \leftarrow \mathsf{Setup}(1^\kappa)$ is known to all parties and consists of a description of a hash function $H_{\mathsf{NIZK}}$, which is modeled in the proof as a random oracle. A party $P_j$ rejects a public key and proof pair $(u_i, \mathfrak{p}_i)$ if $\mathsf{Verify}(\sigma, S_i, \mathfrak{p}_i) \neq 1$. As a result, we can argue that the $x_i$ for $i \in I$ are independent of $\{x_j\}_{j \in [n] \setminus I}$ with all but negligible probability. The main modification to the proof of Theorem 1 involves the simulation of the NIZK proofs for the honest parties, since we need to embed DBDH challenges and thus do not know the exponents. Before embedding the DBDH challenges, we have a series of $h = n - t$ hybrids, where in the $k$-th such hybrid, we invoke the zero-knowledge property of the NIZK argument system to simulate (which will involve programming the oracle $H_{\mathsf{NIZK}}$) the proof string $\mathfrak{p}_k$ for honest party $P_k$ with a computationally indistinguishable proof string $\mathfrak{p}'_k$. The remainder of the proof proceeds in the same manner as the proof of Theorem 1.

## References


1. Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold fhe. *Advances in Cryptology–EUROCRYPT 2012*, pages 483–501, 2012.
2. Gilad Asharov and Yehuda Lindell. A full proof of the BGW protocol for perfectly-secure multiparty computation. *Electronic Colloquium on Computational Complexity (ECCC)*, 18:36, 2011.
3. Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In *Eurocrypt*, volume 2656, pages 255–271. Springer, 2003.
4. Klaus Kursawe, George Danezis, and Markulf Kohlweiss. Privacy-friendly aggregation for the smart-grid. In *Privacy Enhancing Technologies*, pages 175–191. Springer, 2011.
5. A. López-Alt, E. Tromer, and V. Vaikuntanathan. On-the-fly Multiparty Computation on the Cloud via Multikey Fully Homomorphic Encryption. In *Proceedings of the 44th symposium on Theory of Computing*, pages 1219–1234, 2012.
6. Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key fhe. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 735–763. Springer, 2016.
7. Constantinos Patsakis, Michael Clear, and Paul Laird. Private aggregation with custom collusion tolerance. In *International Conference on Information Security and Cryptology*, pages 72–89. Springer, 2014.