Matching radar signals and fingerprints with MPC

Benjamin Hansen Mortensen[1,2], Mathias Karsrud Nordal[1,3] and
Martin Strand[1]

[1] Norwegian Defence Research Establishment (FFI), Norway
[2] University of Oslo (UiO), Norway
[3] Norwegian University of Science and Technology (NTNU), Norway

Abstract. Vessels can be recognised by their navigation radar due to the characteristics of the emitted radar signal. This is particularly useful if one wants to build situational awareness without revealing one's own presence. Most countries maintain databases of radar fingerprints but will not readily share these due to national security regulations. Sharing of such information will generally require some form of information exchange agreement.

However, all parties in a coalition benefit from correct identification. We use secure multiparty computation to match a radar signal measurement against secret databases and output plausible matches with their likelihoods. We also provide a demonstrator using MP-SPDZ.

Keywords: multiparty computation · radar identification · demonstration

1 Introduction

Distinguishing your friends from your foes is a crucial prerequisite for any activity, and for military operations in particular. Not letting your foe know they have been identified is an additional bonus. To this end, surveillance systems may for instance listen for signals from onboard navigation radars and compare these to pre-collected fingerprints of known radar systems. A process often referred to as fingerprinting, with the end goal to later compare new radar signals against the fingerprints to hopefully identify the emitted source.

The radar fingerprints and how these are gathered, and the actual fingerprint database is usually highly classified, and thus difficult to share with partners. At the same time the mission may benefit from a successful identification. Hence, we explore the applicability of multiparty computation (MPC) for this scenario, and report on the results.

MPC enables parties to jointly compute a function on each party's private input. This is useful as instead of issuing the need for a centralised coalition database for radar fingerprints, MPC can be applied, leveraging the pre-existing databases of each country to attain the equivalent capability. We use the MP-SPDZ framework to prototype our experiment.

Concretely, we model a scenario where vessels enter an area surrounding an offshore installation. On-land antennas detect the signal, and (secret) share the signal with a number of confidential fingerprint databases. These players are then tasked with finding the best match against the union of these databases, hopefully before the vessel reaches the installation.

We implement the following ideal functionality.

Mortensen and Nordal contributed to this work both as summer interns at FFI and as students at their respective universities afterwards.

E-mail: benjamin.hansen.mortensen@tutanota.com (Benjamin Hansen Mortensen), mathias.nordals@gmail.com (Mathias Karsrud Nordal), martin.strand@ffi.no (Martin Strand)

Private input from player 1 A description of an unknown, detected radar fingerprint, characterised by two integers; *radio frequency* (RF) and *pulse repetition interval* (PRI), see Section 2.2.

Private input from players $2..n$ Tables containing vessel ID, average RF and average PRI, and a covariance matrix describing the spread of RF and PRI see Section 4.1 for details and how we preprocess the covariance matrix.

Output to all players A list of *plausible* matches and for each a number corresponding to the likelihood of this being a correct match.

Influence and leakage We follow the convention described by Cramer, Damgård and Nielsen [CDN15, p. 63], which allows the adversary to corrupt and control player i.

The concrete computations are described in detail in Section 4.1. Notice that any adversary that corrupts both player 1 (which supplies the current measurement) and any other player (which holds a partial database) will be able to use the output to create or improve a database entry. This limitation is inherent to the physical setup, which is outside our control.

The motivation for this work is two-fold: Primarily, we want to demonstrate that a real-life problem has a solution. Secondly, we use it as an experiment to investigate the maturity of MPC; can a readily available implementation provide all functionality for a viable demonstration? This work is strictly scoped as an application of MPC. It would be interesting as a follow-up work to investigate if one could optimise protocols to solve the problem more efficiently. The answer to both of these objectives is still positive, which has two implications. Firstly, the above functionality can be implemented and run in a reasonable time, even before specific optimisations have been considered. Secondly, it verifies that current implementations of MPC may be ready for a wider audience in much the same way as Zama [Zam22] has enabled any hobby programmer to experiment with FHE. We believe this is a necessary checkpoint on the path to wider adoption. An improved variant of this work may eventually see use in the Norwegian Armed Forces and its allies.

1.1 Related applications

The first publicly known application of MPC was Denmark's sugar beet auctions [BCD+09]. Kavani and Popa[1] maintain a hub for real- world MPC deployments. There are currently 19 entries in the catalogue. Nine of these are related to cryptocurrencies. There does not seem to be any examples involving government parties, and we are not aware of any other such examples having been described publicly.

The closest example in the Kavani and Popa's list comes from the healthcare sector. By the author's own description, VaultDB is "is a framework for securely computing SQL queries over private data from two or more sources" [RAB+22], and the authors have successfully analysed more health records for more than 13 million people without compromising privacy.

1.2 Our contribution

We primarily describe our attempt to solve a real-world coalition problem using MPC. This provides a rare glimpse of a world that is often kept secret from the public and cryptographers alike. We do not claim to have implemented an optimal solution, nor was this a result of a comprehensive threat analysis. It serves as a demonstration to

---

[1]"MPC Deployments", https://mpc.cs.berkeley.edu

communities still unaware of the possibilities for secure, private computations, and it also provides a real-world scenario to the cryptography community.

Additionally, we present a method to avoid redundant computation on sparse data by introducing a sorting algorithm *Sparse Oblivious Keyword Sort* (SOKS) that extracts k number of rows from a 2D array. The algorithm's communication complexity depends on k therefore it offers an alternative way of sorting that, depending on the sparsity of the data, could yield better performance, cf. Section 5.2.

1.3   Outline

We provide a brief introduction to MPC, radar fingerprints and the necessary statistics in Section 2. In Section 3, we report on some of the insights we gathered on sorting with many parties. We then go on to describe the threat model and the experiment setup in Section 4, before providing our results in Section 5

2   Background

2.1   Secure multiparty computations

MPC [Yao86, GMW87] is a cryptographic technique allowing multiple parties to jointly compute a function over their inputs while keeping them confidential. Unlike traditional computation methods that rely on a central authority, MPC ensures that no single party learns more than what they can deduce from the output and their own input. For a more in-depth introduction to MPC, we refer to Lindell [Lin21] and Cramer, Damgård and Nielsen [CDN15].

MPC has seen tremendous development since the introduction of Shamir's secret sharing scheme [Sha79] and Beaver's multiplication triples [Bea92]. It is outside the scope of this work to provide a complete survey of MPC. Instead, we briefly recall some of the most important concepts and tools required for this work.

The security of MPC guarantees that any adversary is unable to tell the difference between the real protocol, and an ideal simulator. It is impossible to learn anything from the latter, so any leakage would also provide a distinguishing feature. The adversary may be *malicious* or *semi-honest*: a semi-honest adversary will follow the protocol but will try to learn information from allowed data. Conversely, a malicious adversary will – in plain words – cheat in any way computationally possible.

The adversary is controlling one or more parties and may do so in a *static* or *adaptive* way. Static adversaries must choose the parties to corrupt before the protocol starts, whereas adaptive adversaries may choose to corrupt parties as they see fit during the protocol execution.

Finally, we also limit the number of parties the adversary may control. Common limits are less than a third, less than half, and sometimes even a dishonest majority. Protocol designers must choose the adversarial capabilities they want to defend against. A protocol with security against a malicious dishonest majority will require more computations than one with security against no more than $1/3$ semi-honest parties.

These protocols are building blocks for applications such as ours. For this work, we have chosen to protect against a malicious majority. We therefore use the protocol MASCOT by Keller, Orsini and Scholl [KOS16], as implemented in MP-SPDZ [Kel20]. We refer to the original publication for details of its operation. Notice that the consequence of this choice is that the runtime will provide an upper bound over all security model choices.

We can compute any circuit using MPC. However, programmers might be stumped as they realise that common tools such as branching, looping and random access generally

are unavailable in MPC, as these control operations will leak a secret state[2].

Consequently, efficiently implementing suitable selective computation for MPC protocols becomes crucial, especially for complex calculations, as it substantially reduces runtime.

Assume we have an algorithm to perform heavy calculations on non-zero entries of a secret array x. Unfortunately, random access is not possible when doing secure multiparty computation, leading to the necessity of performing calculations on all entries, unless we structure the data. Let k be the maximum number of non-zero elements. We can then sort and access the k last elements of the sorted array, thereby avoiding redundant computation.

In MPC we require oblivious sorting algorithms, meaning that the access pattern does not reveal information about the underlying data. Bitonic sort [Bat68] is one such classical algorithm. In comparison with regular computing MPC also introduce additional costs with respect to the number of parties m, meaning that depending on the number of items n the dominant complexity could be the one with respects to m, cf. Figure 2. Further in this work, we introduce SOKS which transform the quadratic complexity on the number of items into being dependent on k. Meaning that depending on the value of k it could yield better performance, cf. Section 5.2.

2.1.1 MP-SPDZ

MP-SPDZ is a high-level cryptographic framework developed for secure multiparty computation. It provides a developer-friendly interface that allows us to design and implement privacy-preserving protocols without having to implement with the low-level details [Kel20]. This abstraction simplifies the development process and makes it easier to create complex privacy-preserving applications using MPC.

The framework is based on a line of research starting from 2012 [DPSZ12, DKL+13, KPR18].

2.2 Radar fingerprints

A radar emits pulses of signals, and then waits for reflections. The main characteristics of a radar system is the PRI and the RF. Based on these, we can create a fingerprint of the particular equipment. Radars of the same model will typically have similar characteristics. However, due to imperfections in clocks and other components, even individual products may be identifiable from small variations in these, and other, parameters.

The radars may operate in a number of modes, for instance depending on whether the vessel is in the open sea, or perhaps navigating near a busy harbour. One may also shape the waveform to optimise the radar, but this will also make it even more recognisable.

For the purpose of this work, we only consider PRI and RF. Based on measurements from a number of vessels[3], we create fingerprints based on the average PRI and the average RF recorded for a pulse of signals, but we allow a variation of up to ± 3 MHz in the RF and 10 % in the PRI, due to clock drift.

Furthermore, we assume that the signals follow a two-dimensional standard distribution and store the covariance matrix of the signals. Hence, if a new signal falls within two potential fingerprint ranges, we can perform meaningful computations on which vessel is more likely. The details of this computation are described below.

We can triangulate the position by using two or more antennas. Hence, we can get both the identity and the position of a vessel by passively listening for any emitted signal.

---

[2]Branching can be emulated by computing the condition b and each branch α_1, α_2 separately, and finally computing $b\alpha_1 + (1 - b)\alpha_2$, ensuring the maximum amount of work.

[3]Unfortunately, these measurements are exempted from public disclosure. The provided example fingerprints are therefore synthetic.
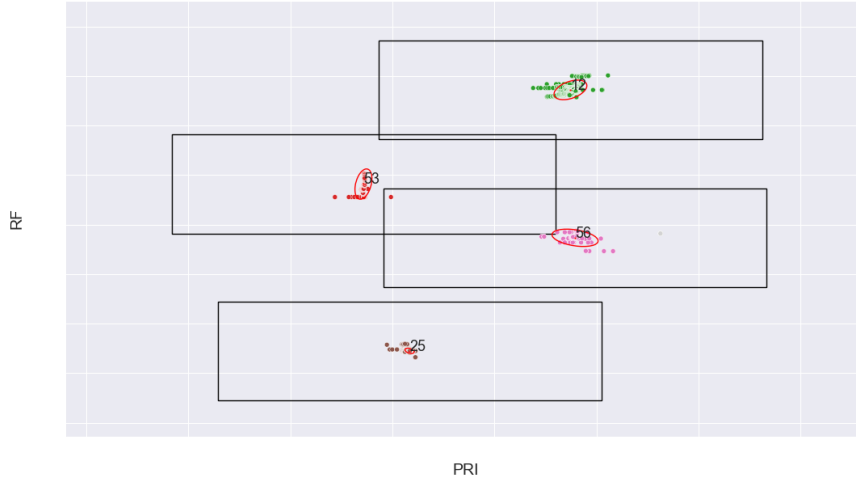
Figure 1: Clusters with overlapping tolerance boxes. The axes have been redacted.

2.3 Mathematical background

Multivariate statistics and numerical linear algebra provide the essential tools to construct our classification functionality for radar signals.

2.3.1 Multivariate statistics

Recall that 99.7 % of values drawn from a one-dimensional normal distribution lies within three standard deviations of the distribution mean. Hence, given normal distribution $Q \sim \mathcal{N}\left(\mu, \sigma^2\right)$ and observation x, calculating how many standard deviations x is away from μ indicates how likely it is that x can haven been drawn from Q. Let d denote the number of standard deviations that an observation x deviates from the mean μ, calculated as $d = \left|\frac{x-\mu}{\sigma}\right| = \sqrt{\frac{(x-\mu)^2}{\sigma^2}}$. To extend this concept to multidimensional distributions, we can analogously consider the covariance matrix as the multidimensional counterpart of σ^2, allowing us to construct an equivalent expression in the multidimensional context. This is the Mahalanobis distance.

Definition 1 (Mahalanobis distance)**.** Given a probability distribution Q on $\mathbb{R}^n$ with mean $\mu = (\mu_1, \ldots, \mu_n)^\top$ and positive definite covariance matrix Σ, the Mahalanobis distance of a point $x \in \mathbb{R}^n$ from Q is

$$D(x, Q) = \sqrt{(x - \mu)^\top \Sigma^{-1} (x - \mu)}.$$

Since $\sqrt{\cdot}$ is a monotonically increasing function, it will often suffice to consider the more convenient quadratic form,

$$D^2(x, Q) = (x - \mu)^\top \Sigma^{-1} (x - \mu).$$

Theorem 1 (Distribution of $D^2(x, Q)$)**.** *Suppose that Q is p-variate normal distribution with mean μ and a positive definite covariance matrix Σ. Then,*

$$D^2(x, Q) = (x - \mu)^\top \Sigma^{-1} (x - \mu)$$

follows a χ^2 distribution with p degrees of freedom.

2.3.2 Numerical linear algebra

The following theorem will provide useful when calculating the squared Mahalanobis distance of a point x from a distribution Q.

Theorem 2 (Cholesky decomposition [Cho24]). *Let A be a real symmetric positive definite matrix. Then A admits a unique decomposition of the form*

$$A = CC^\top,$$

for a lower triangular matrix C with real and positive diagonal entries.

Let $x \in \mathbb{R}^n$ be a point and $Q \sim \mathcal{N}_n (\mu, \Sigma)$. Then consider the squared Mahalanobis distance $D_x^2 = D(x,Q)^2 = (x - \mu)^\top \Sigma^{-1} (x - \mu)$. Recall that for D_x^2 to be defined Σ must be non-singular. Hence, it follows that it must be symmetric positive definite. Thus, by Theorem 2, Σ^{-1} admits a Cholesky decomposition. This yields

$$\begin{aligned} D_x^2 &= (x - \mu)^\top \Sigma^{-1} (x - \mu) \\ &= (x - \mu)^\top (CC^\top)^{-1} (x - \mu) \\ &= (x - \mu)^\top C^{-\top} C^{-1} (x - \mu) \\ &= \left(C^{-1} (x - \mu)\right)^\top \left(C^{-1} (x - \mu)\right). \end{aligned}$$

Every equality above follows from elementary properties of matrix inversion and transposition, and so

$$\eta = C^{-1} (x - \mu) \implies D_x^2 = \|\eta\|_2^2. \tag{1}$$

Hence, calculating D_x^2 follows from solving $C\eta = x - \mu$. In particular since C is a lower triangular matrix, this can be done efficiently through forward substitution.

3 Sorting in MPC

Sorting is an intensively researched problem in computer science and has also received some attention for MPC. The lack of random access is a significant limitation. Oblivious Radix Sort (ORS) [HICT14] is an accessible algorithm for addressing this objective. Nonetheless, it exhibits an exponential round and communication complexity in the number of parties involved in the protocol. ORS is the inbuilt method for sorting 2D arrays[4] in MP-SPDZ, making multidimensional sorting in MP-SPDZ impractical for protocols with a substantial number of participating parties.

3.1 Oblivious Radix Sort and and Oblivious Keyword Sort

We briefly present the central ideas of Oblivious Radix Sort (ORS) and Oblivious Keyword Sort (OKS) for the benefit of the interested reader. Both ORS and OKS take as input a vector $\vec{\kappa}$ of keys a matrix D with the data. The reader may also skip this section with no loss of continuity.

At the core of ORS we have the *Reveal-Sort* algorithm, which sorts partially known keys $\vec{\kappa}$ and secret-shared data $\bar{D}$, both of which may be different from the actual data and keys we want to sort. The parties produce random shuffles of the key, and that can be revealed in each execution. Hence, one can apply the random permutation to both $\vec{\kappa}$ and $\bar{D}$, and then reveal the permuted $\vec{\kappa}'$. Sorting $\vec{\kappa}'$ is essentially to create a permutation,

---

[4]By sorting an 2D array, we essentially mean sorting a table by one of its columns

---

Algorithm 1: Sparse Oblivious Keyword Sort (SOKS)

Data: Sparse 2D array: $A = [\![\alpha_1]\!], \ldots, [\![\alpha_n]\!]$
Result: Sorted array $\Upsilon = [\![v_1]\!], \ldots, [\![v_k]\!]$
$B \leftarrow \left(A^\top\right)_1$;   /* B set to be the first column of A */
$\tilde{B} \leftarrow \mathrm{Sort}\, B$;
for $i \in \{1, \ldots, k\}$ **do**
$\quad \big| \quad [\![v_i]\!] \leftarrow \sum_{j=1}^n \left([\![\tilde{\beta}_i]\!] == j\right) \cdot [\![\alpha_j]\!]$;
end
return $\Upsilon = [\![v_1]\!], \ldots, [\![v_k]\!]$

---

and this permutation can also be applied to the corresponding $\bar{D}'$. The output will be a correctly sorted version of $\bar{D}$ [HICT14].

The authors then apply a binary decomposition to the original keys κ, agree on a shared permutation h on $\{1, \ldots n\}$, and then apply Reveal-Sort repeatedly, starting with h and the most significant bit of $\vec{\kappa}$, giving a radix sort algorithm. In the final iteration, the original data D is introduced and sorted.

When the number of parties and the size of the underlying field is fixed, this results in an $O(1)$ round complexity and an $O(n \log n)$ communication complexity, where n is the number of rows to be sorted.

OKS was introduced by Zhang [Zha11] as a protocol for sorting D based on comparison. OKS can be divided into two parts, the transformation part and the sorting part.

In the transformation part, the keys $\vec{\kappa}$ are transformed to a new set of keys $\vec{\kappa}'$, such that if $\kappa_i \leq \kappa_j$ for some indices i and j, the transformed keys would satisfy $\kappa_i' < \kappa_j'$. This steps then stores a new array C, where each entry C_i holds the position of κ_i' if $\vec{\kappa}'$ were to be sorted. In the sorting part D gets sorted obliviously according to the entries of C.

The resulting protocol achieves $O(1)$ round complexity and $O(n^2)$ communication complexity.

3.2 Modified Oblivious Keyword Sort for sparse two-dimensional arrays

Zhang [Zha11] has proposed a significant improvement to ORS regarding round and communication complexity, a constant round sorting algorithm for arbitrary keZhang11ay indexed data structures called oblivious keyword sort (OKS). Unlike ORS, OKS exhibits quadratic communication complexity in the number of parties. This significantly reduces the complexity associated with a larger number of participating parties, m.

However, further improvements can be made to the OKS algorithm under additional assumptions. Suppose the goal of the protocol is to perform a large computation on a small subsample of rows, given a specific condition. The output of this computation is of primary interest, and the goal is to obtain an array where the $k \ll n$ relevant rows are grouped together. Since we are primarily interested in these relevant rows, and they will all be treated the same, we may assume that each row has a unique identifier, such as a row number. If not, a new column can be added to provide this information.

The original OKS algorithm includes a transformation step that modifies the first column of the input data to ensure that all entries are unique while preserving their original order. Under the assumptions outlined above, this transformation step is no longer necessary and can be omitted. Instead, the first column can be sorted directly using an efficient one-dimensional sorting algorithm. For convenience, this can be done using Batcher's Merge Sort [JKU11], although any efficient oblivious one-dimensional sorting algorithm that does not exhibit exponential round and communication complexity in m
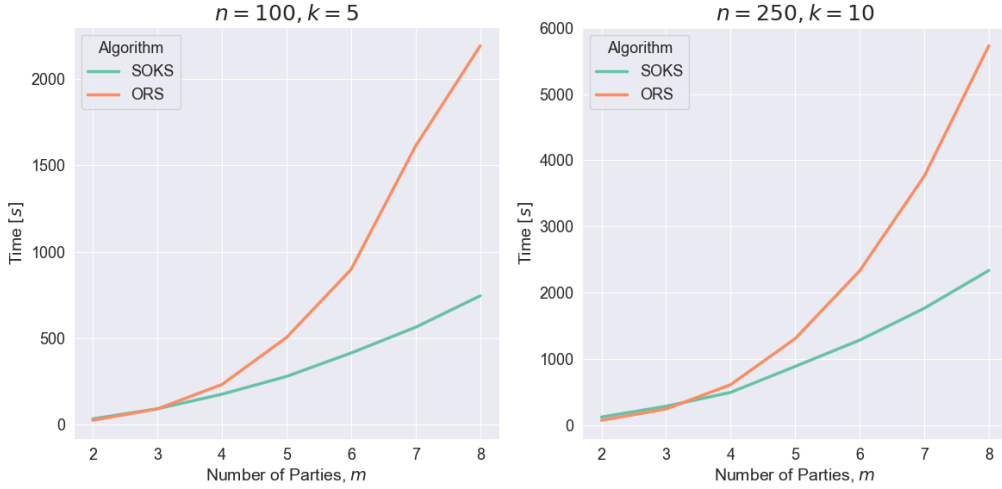
Figure 2: Runtime comparison of ORS and SOKS for 2D arrays of length n and number of non-zero entries k.

would suffice.

A more detailed explanation of the idea is as follows: Let $A = (r_1, \ldots r_n)$ be some 2D array, i.e. $r_i = (r_{i,1}, \ldots r_{i,m})$ Assume that we want to sort by the first column, and that it consists of the row numbers, i.e. $r_{i,1} = i$. Use an indicator function on any of the data of the row, and multiply the result with all fields, yielding some $A' = (r'_1, \ldots r'_n)$. This will change the rows we want to ignore to $(0, \ldots, 0)$, leaving k non-zero rows. Make a copy $c = (r'_{1,1}, \ldots r'_{n,1})$ of only the first column, and sort it. Naturally, the non-zero elements will gather at one end. Moreover, the non-zero entries in c holds the original index of the corresponding row in A.

Given the sorted 1D array c, we can "sort" the remainder of A. Let $c = (c_1, \ldots c_n)$ be the sorted 1D array and let $\chi_{c_i}(r'_j) = 1$ if $c_i = r'_{j,1}$ and 0 otherwise. Set $\hat{A} = (\hat{r}_1, \ldots \hat{r}_n)$ where $\hat{r}_i = \sum_{j=1}^n \chi_{c_i}(r'_j) \cdot r'_j$. Observe that assuming $r_{i,1} = i$ evaluating $\chi_{c_i}(r'_j)$ is equivalent to checking whether or not c_i equals j. To see this, note that $r'_{j,1} = r_{j,1} = j$ if the j'th row is not ignored. Hence the binary output of the comparison $c_i == j$ equals the value of $\chi_{c_i}(r'_j)$.

We are then left with a 2D array with k of the original rows, correctly sorted by the first column.

The advantage of SOKS diminishes as the size of the array and the treshold k increases. This outcome is expected, as ORS has better asymptotic complexity in the size of the array. We have compared SOKS and ORS using the MASCOT protocol with an integer size of 18. The sorting algorithms were applied to arrange an array of size $n \times 3$ with k rows containing non-zero values. A visual comparison of SOKS and ORS can be seen in Figure 2.

In summary, SOKS (Algorithm 1), presents a viable and efficient alternative to ORS for sorting 2D spares arrays in MPC under the assumptions presented in this section. We believe these assumptions are likely to be commonly encountered in MPC applications, making SOKS a viable solution for addressing them. It has constant round and quadratic communication complexity in terms of the number of parties involved, making it a practical choice for protocols with a large number of participants. Further, it offers an alternative way of extracting k number of rows from a 2D array, which later, cf. Section 5.2, is shown to be more efficient than ORS when k is of certain values.

4 Threat model and experiment setup

Let $\mathcal{P} = \{P_2, \ldots, P_n\}$ be the countries in a coalition, each having a fingerprint database. P_1 (which may belong to the same country as one of the above) conducts a radar signal measurement $x = (x_{\mathrm{PRI}}, x_{\mathrm{RF}})$ of an unknown vessel and seeks to identify its ship type. The goal is to correctly match x to the possible radar fingerprints across the databases.

As we are not developing new MPC protocols, the question of malicious or semi-honest security is here a question of performance, not mathematics. For this work, we have assumed malicious adversaries – after all, the databases are usually highly classified. Any production choice will be a trade-off between availability and risk willingness. There might also be operational considerations outside the realm of cryptography. For now, we use MASCOT [KOS16].

We have been provided with real radar signal observations, from which we have generated fingerprints. Unfortunately, the observations are exempt from public disclosure. As such, the accompanying sample fingerprints are synthetic, sampled at random from the same RF and PRI ranges as the real fingerprints.

The RF range is between 9.3 and 9.5 GHz. Hence, we can reduce the magnitude of our data by a linear translation. Meanwhile, the PRI measurements, which vary significantly in configuration, find a reasonable representation in microseconds.

In pursuit of improved computational efficiency, one can use these observations to further simplify the data. Floating-point operations, particularly division, demand substantial resources due to their bit-intensive nature for larger values. Our primary focus is on harnessing addition and multiplication operations to expedite computations. However, it's crucial to work with as small numbers as possible. To achieve this, two key adjustments have been implemented: firstly, subtracting 9.2 GHz from RF, and secondly, truncating PRI values to microseconds and scaling them to integers. These refinements collectively aim to reduce computation time while maintaining sufficient precision.

Each database entry consists of a cluster ID, average RF, average PRI, and a covariance matrix. Recall that the fingerprint by definition is the midpoint surrounded by a ± 3 MHz and ± 10 % *tolerance box*.

4.1 Fingerprint and signal identification

The box surrounding the average represents all points that could thinkably belong to the radar. However, quite a few of these boxes could overlap. A measurement in this overlap results in a number of plausible matches. This is illustrated in Figure 1. To effectively distinguish these fingerprints, we need a more advanced approach involving a maximum likelihood classifier. We have established a constant k, representing the maximum number of intersecting tolerance boxes for any point on the plane.

The protocol receives a radar measurement, denoted as x, from party P_1, and a multidimensional array database from each of the remaining $n - 1$ parties as input.

A measurement falling outside of a tolerance box should be assumed to be coming from an unknown vessel. Hence, the first part of the classification protocol examines if the measurement falls within any of the tolerance boxes. This reduces the number of redundant calculations performed by the algorithm. For the remaining fingerprints we conduct the following hypothesis test.

Hypothesis 0 (H0). *x comes from $Q \sim \mathcal{N}_2 (\mu, \Sigma)$*

Hypothesis 1 (H1). *x does not come from Q*

Recall from Theorem 1 that under the assumption of H0 the Mahalanobis distance, $(x - \mu)^\top \Sigma^{-1} (x - \mu)$, is χ^2 distributed with two degrees of freedom. In particular, one can calculate the probability of a given observation $y \sim \mathcal{N}_2 (\mu, \Sigma)$ falling outside the ellipse

defined by $(x - \mu)^\top \Sigma^{-1} (x - \mu) = D_x^2$. This probability is given by the complimentary cumulative density function (CCDF) of χ_2^2.

$$\bar{F}_{\chi_2^2} \left(D_x^2 \right) = \mathrm{P} \left(\chi_2^2 > D_x^2 \right) = 1 - F_{\chi_2^2} \left(D_x^2 \right) = 1 - \left(1 - e^{-\frac{D_x^2}{2}} \right) = e^{-\frac{D_x^2}{2}}.$$

Specifically, this is the p-value of the above test. Furthermore, since the CCDF of χ_2^2 is strictly decreasing it suffices to consider the squared Mahalanobis distance directly. That is, the larger the distance, the smaller the p-value, indicating that there is evidence for rejecting H0. We can now apply the notions outlined in Section 2.3.2 to calculate D_x^2 efficiently. Notably, not calculating the exact p-value reduces the number of fixed-point operations done by the protocol, resulting in faster computation.

Let us revisit Equation 1. To perform this calculation effectively, having the Cholesky decomposition of matrix Σ greatly assists us.

$$\Sigma = \begin{bmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{12} & \sigma_{22} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 \\ l_{21} & l_{22} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} \\ 0 & l_{22} \end{bmatrix} \implies \sigma_{11} = l_{11}^2, \quad \sigma_{12} = l_{11} l_{21}, \quad \sigma_{22} = l_{22}^2 + l_{21}^2.$$

This yields the following form of Eq. 1:

$$\begin{bmatrix} l_{11} & 0 \\ l_{21} & l_{22} \end{bmatrix} \begin{bmatrix} \eta_1 \\ \eta_2 \end{bmatrix} = \begin{bmatrix} x_1 - \mu_1 \\ x_2 - \mu_2 \end{bmatrix} \overset{\text{fwd.sub}}{\implies} \begin{cases} \eta_1 = \frac{x_1 - \mu_1}{l_{11}} \\ \eta_2 = \frac{(x_2 - \mu_2) - l_{21} \eta_1}{l_{22}}. \end{cases}$$

Lastly, we compute the squared L_2 norm of η,

$$\begin{aligned} \|\eta\|_2^2 = \eta_1^2 + \eta_2^2 &= \frac{(x_1 - \mu_1)^2}{l_{11}^2} + \frac{(x_2 - \mu_2)^2 - 2 (x_2 - \mu_2) l_{21} \eta_1 + l_{21}^2 \eta_1^2}{l_{22}^2} \\ &= \frac{(x_1 - \mu_1)^2 \sigma_{22} + (x_2 - \mu_2)^2 \sigma_{11} - 2 (x_1 - \mu_1) (x_2 - \mu_2) \sigma_{21}}{\sigma_{22} \sigma_{11} - \sigma_{21}^2}. \end{aligned}$$

Due to the slow nature of fixed-point division, one should strive to avoid divisions if possible. Notice that for $\mathsf{lbl} \in \{11, 22, 21\}$, $c_{\{2,1,3\}} = \sigma_{\mathsf{lbl}} / \left(\sigma_{22} \sigma_{11} - \sigma_{21}^2 \right)$ (resp.) depends only on values from a single party, and can therefore be precomputed. Let $y_1 = x_1 - \mu_1$ and $y_2 = x_2 - \mu_2$. Then, $\|\eta\|_2^2$ can be computed as a linear combination

$$c_1 y_1^2 + c_2 y_2^2 - 2 c_3 y_1 y_2,$$

where c_1, c_2 and c_3 are fixed-point values.

4.2   Implementation

Recall that we assume that P_1 provides the radar signal, while the other parties provide (partial) databases. For the 2-party variant, the latter is the complete database. For the general case, all of the databases are consolidated into a single tensor prior to the rest of the classification process. Consequently, broadening the 2-party setting to an arbitrary m-party setup is a straightforward extension. For the sake of brevity and without loss of generality, this section focuses exclusively on the 2-party protocol.

We must iterate over all fingerprints to identify potential matches. Any fingerprint entry that does not contain x within its specified tolerance box is systematically nullified. This process yields a sparse tensor only containing information about the potential matches.

Recall that the process of performing maximum likelihood classification requires some extra computation per item in the database. Therefore, an optimal approach would involve applying the classifier solely to the non-zero entries. However, this strategy is unfeasible due to the absence of random access. We solve this issue by sorting the list, and only

---

Algorithm 2: Implementation

Data: Measurement: $x = (\llbracket x_{\text{pri}} \rrbracket, \llbracket x_{\text{rf}} \rrbracket)$, Database: $\Delta = \llbracket \delta_1 \rrbracket, \ldots, \llbracket \delta_n \rrbracket$
Result: Array of probable fingerprints: $\Psi = \llbracket \psi_1 \rrbracket, \ldots, \llbracket \psi_k \rrbracket$
$\tau \leftarrow$ Empty array of length n;
$\Psi \leftarrow$ Empty array of shape $k \times 2$;
for $i \in \{1, \ldots, n\}$ **do**
  if contained($\llbracket x_{pri} \rrbracket, \llbracket x_{rf} \rrbracket, \llbracket \delta_{i,2} \rrbracket, \llbracket \delta_{i,3} \rrbracket$) **then**
   | $\llbracket \tau_i \rrbracket \leftarrow \llbracket \delta_{i,1} \rrbracket$
  else
   | $\llbracket \tau_i \rrbracket \leftarrow \llbracket \mathbf{0} \rrbracket$
  end
end
τ.sort() ;   /* To hide the origin of each label */
Reveal τ;
for $j \in \{1, \ldots, k\}$ **do**
  | $\llbracket \psi_{j,1} \rrbracket \leftarrow \llbracket \tau_j \rrbracket$;
  | $\llbracket \psi_{j,2} \rrbracket \leftarrow$ mahalanobis($\llbracket K \rrbracket, \llbracket \delta_{\tau_j} \rrbracket, \llbracket x_{\text{pri}} \rrbracket, \llbracket x_{\text{rf}} \rrbracket$);
end
return $\Psi = \llbracket \psi_1 \rrbracket, \ldots, \llbracket \psi_k \rrbracket$

Algorithm 3: Contained

Data: Measurement: $x = (\llbracket x_{\text{pri}} \rrbracket, \llbracket x_{\text{rf}} \rrbracket)$, Mean of RF values in cluster: $\llbracket \delta_2 \rrbracket$, Mean of PRI values in cluster: $\llbracket \delta_3 \rrbracket$
Result: 1 if measurement x is contained in tolerance box. 0 otherwise.
$c \leftarrow 0$;
$c \leftarrow c + (\llbracket x_{\text{rf}} \rrbracket <= \llbracket \delta_2 \rrbracket + 30000)$;
$c \leftarrow c + (\llbracket x_{\text{rf}} \rrbracket >= \llbracket \delta_2 \rrbracket - 30000)$;
$c \leftarrow c + (\llbracket x_{\text{pri}} \rrbracket <= \llbracket \delta_3 \rrbracket + 9400)$;
$c \leftarrow c + (\llbracket x_{\text{pri}} \rrbracket >= \llbracket \delta_3 \rrbracket - 9400)$;
return ($c == 4$)

---

applying the classification on the k last entries within this array. This gives accurate classification of fingerprints that closely resemble the new measurement. Implementing this protocol was conveniently done in MP-SPDZ. We outline this procedure in Algorithm 2, our code is available on Github[5].

The subprotocol contained outputs true if the measurement is contained in the tolerance box corresponding to cluster i. The mahalanobis procedure performs the calculation derived in Section 4.1. The pseudocode of the subprocedures is listed in Algorithm 3 and 4.

All data points are in the same order of magnitude. Instead of computing ± 10 % for each fingerprint, we use the fixed value 9400 to describe all tolerance boxes, which saves a number of multiplications.

5   Results

We executed our experiment on a commercial laptop (16 GB RAM, Intel i5 8265U, 1.6 GHz CPU) running Ubuntu 22.04 LTS and MP- SPDZ v0.3.7. To eliminate variation in network speed all parties were run on the same machine locally.

---

[5] https://github.com/FFI-no/Paper-radar-signature-matching

Algorithm 4: Mahalanobis

Data: Measurement: $x = (\llbracket x_{\text{pri}} \rrbracket, \llbracket x_{\text{rf}} \rrbracket)$, Database entry: $\llbracket \delta \rrbracket$, Array of
       precomputed coefficients: $\llbracket K \rrbracket$

Result: Mahalanobis distance from x to distribution corresponding to $\llbracket \delta \rrbracket$

$y_1 \leftarrow \llbracket x_{\text{pri}} \rrbracket - \llbracket \delta_3 \rrbracket$;
$y_2 \leftarrow \llbracket x_{\text{rf}} \rrbracket - \llbracket \delta_2 \rrbracket$;

$s_1 \leftarrow y_1^2 \cdot \llbracket K[1] \rrbracket$;
$s_1 \leftarrow y_2^2 \cdot \llbracket K[2] \rrbracket$;
$s_1 \leftarrow -2y_1 y_2 \cdot \llbracket K[3] \rrbracket$;
return $s_1 + s_2 + s_3$

Let $n = 88$ be the number of fingerprints from our data analysis. The maximum number of tolerance boxes that overlap at a single point is bounded by $k = 10$.

Furthermore, the program was compiled using an integer length (`-F`) of 22, which is sufficient to accommodate the largest numbers in the dataset. Specifically, the largest integers we are working with have magnitudes less than $2 \cdot 10^6$, which is within the range representable by 22 bits. The integer length is also chosen small to reduce the protocol's resource requirements.

To accurately represent the precomputed coefficients used in the Mahalanobis procedure, the fixed-point parameters $(f, k) = (19, 38)$ were chosen. The value $k = 38$ was selected to accommodate the largest fixed-point values encountered in the protocol. The parameter $f = 19$ represents the number of bits used to denote the decimal part of any fixed-point number. This value of f was determined through testing, providing satisfactory precision for our computations while minimising computational cost.

The number of participating parties is denoted by m, where one party acts as the sensor who only provides a radar signal $x = (x_{\text{PRI}}, x_{\text{RF}})$. The other $m - 1$ parties are databases with a total of n radar fingerprints. Table 1 shows the runtime of the computation for the number of parties involved. Note that for all executions, we used the same x.

Table 1: Runtime for fingerprint matching, with $m = 2, 3, 4, 5$ parties. Two signficant digits.

| m	2	3	4	5
Runtime (sec.)	31	86	200	400

We have also created a user and audience friendly visual demonstrator, which can be run by cloning the repository and following the instructions.

5.1   Practicability

Detecting radar signals depends on the strength of the signal. Consider an antenna mounted 400 m above ground. They will be able to detect a radar signal emitted about 20 m above sea level, up to 100 km away. Define the detection zone as being two of these mounted side by side on the coastline pointing at the sea. Using this setting we establish a baseline for measuring the practicability of our results.

Imagine a vessel moving at a realistic 13 knots (24 km/h, 15 mph). For $m = 2$ we used 31 seconds to identify the vessel. In the same time, the vessel will have travelled approximately 200 m. Compare that to the range of the antennas: it has traversed 0.2 % of the radius of the detection zone.

We may also consider a fast-moving vessel at 60 knots (110 km/h, 70 mph). With four databases (i.e., $m = 5$) we got a result in 400 seconds. This vessel would have moved
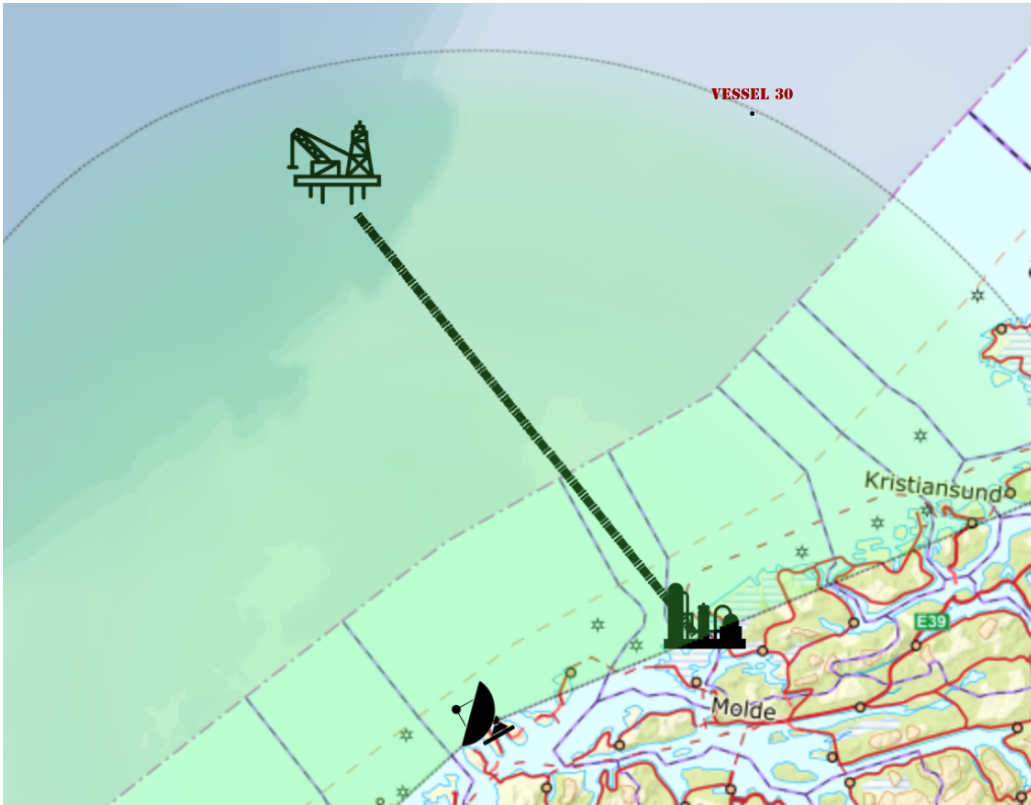
Figure 3: A screenshot from the demonstrator. The map is © norgeskart.no (CC-BY 4.0).

12.3 km before being identified, still well within the 100 km zone.

5.2 Trade-offs between sorting methods

While we ended up not having to use neither ORS nor SOKS, we still present some of the insights we gathered.

From our experiment we found that the most efficient solutions had to be balanced in terms of the complexity of k and m. The array length n was fixed for all experiments. In the case of one database, $m = 2$, ORS sorts faster than SOKS. When using three databases, i.e., $m = 4$, SOKS exhibits greater speed due to its non-exponential complexity in the number of participating parties, while ORS encounters limitations, affecting both the round and communication complexity. For our case with $k = 10$, we find that for $m = 4$, it is more efficient to use SOKS, giving a threshold of $K = 4$.

The threshold K depends on k, which we demonstrate with an example. Fix $m = 4$, then consider $k = 10$. Using SOKS outperformed the ORS by 10.05 % in an experiment. However, with $k = 20$ ORS is faster by 7.61 % in the same experiment.

6   Conclusion

We have provided another specimen to the MPC application zoo, in a category that is – to the best of our knowledge – not previously explored in the open. Even with the most rigid security model, we provide an answer to the matching problem well within the time requirements for the example scenario we present.

For further development, we must consider the fact that some fingerprints could include description of waveforms, as well as accepting more than a single measurement as input.

While the MPC protocol gives strong security guarantees, the protocol itself may leak information. For instance, it is likely that P_1, the provider of the measurement, also holds one of the fingerprint databases. If the best match is not among their entries, they will now have gathered *some* information for a new entry: They know the measurement, and they know the best answer. The other parties don't know the exact measurement but may know the approximate vicinity if one of their own entries turned up among the plausible matches.

However, this is an underlying issue with the process itself, and not one that any cryptographic technique can compensate for. The alternative is simply not getting a correct identification. Whether or not this is worth the cost is a decision that belongs to the information owners.

Finally, a production-level implementation should also include MPC experts to optimise the underlying protocols.

References

[Bat68]   Kenneth E. Batcher. Sorting networks and their applications. In *American Federation of Information Processing Societies: AFIPS Conference Proceedings*, volume 32 of *AFIPS Conference Proceedings*, pages 307–314. Thomson Book Company, Washington D.C., 1968. doi:10.1145/1468075.1468121.

[BCD+09]   Peter Bogetoft, Dan Lund Christensen, Ivan Damgård, Martin Geisler, Thomas P. Jakobsen, Mikkel Krøigaard, Janus Dam Nielsen, Jesper Buus Nielsen, Kurt Nielsen, Jakob Pagter, Michael I. Schwartzbach, and Tomas Toft. Secure multiparty computation goes live. In Roger Dingledine and Philippe Golle, editors, *Financial Cryptography and Data Security, 13th International Conference, FC 2009*, volume 5628 of *Lecture Notes in Computer Science*, pages 325–343. Springer, 2009. doi:10.1007/978-3-642-03549-4_20.

[Bea92]   Donald Beaver. Efficient multiparty protocols using circuit randomization. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 420–432. Springer, Heidelberg, August 1992. doi:10.1007/3-540-46766-1_34.

[CDN15]   Ronald Cramer, Ivan Damgård, and Jesper Buus Nielsen. *Secure Multiparty Computation and Secret Sharing.* Cambridge University Press, 2015. URL: http://www.cambridge.org/de/academic/subjects/computer-scien ce/cryptography-cryptology-and-coding/secure-multiparty-com putation-and-secret-sharing?format=HB&isbn=9781107043053, doi: 10.1017/CBO9781107337756.

[Cho24]   André-Louis Cholesky. Note sur une méthode de résolution des équations normales provenant de l'application de la méthode des moindres carrésa un syteme d'équations linéaires en nombre inférieura celui des inconnues (Published six years after Cholesky's death by Benoit). *Bull. Géodésique*, 2:67–77, 1924. doi:10.1007/BF03031308.

[DKL+13]   Ivan Damgård, Marcel Keller, Enrique Larraia, Valerio Pastro, Peter Scholl, and Nigel P. Smart. Practical covertly secure MPC for dishonest majority - or: Breaking the SPDZ limits. In Jason Crampton, Sushil Jajodia, and Keith Mayes, editors, *ESORICS 2013*, volume 8134 of *LNCS*, pages 1–18. Springer, Heidelberg, September 2013. `doi:10.1007/978-3-642-40203-6_1`.

[DPSZ12]   Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 643–662. Springer, Heidelberg, August 2012. `doi:10.1007/978-3-642-32009 -5_38`.

[GMW87]   Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987. `doi:10.1145/28395.28420`.

[HICT14]   Koki Hamada, Dai Ikarashi, Koji Chida, and Katsumi Takahashi. Oblivious radix sort: An efficient sorting algorithm for practical secure multi-party computation. Cryptology ePrint Archive, Report 2014/121, 2014. `https: //eprint.iacr.org/2014/121`.

[JKU11]   Kristján Valur Jónsson, Gunnar Kreitz, and Misbah Uddin. Secure multi-party sorting and applications. Cryptology ePrint Archive, Report 2011/122, 2011. `https://eprint.iacr.org/2011/122`.

[Kel20]   Marcel Keller. MP-SPDZ: A versatile framework for multi-party computation. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 1575–1590. ACM Press, November 2020. `doi:10.1145/3372 297.3417872`.

[KOS16]   Marcel Keller, Emmanuela Orsini, and Peter Scholl. MASCOT: Faster malicious arithmetic secure computation with oblivious transfer. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 830–842. ACM Press, October 2016. `doi: 10.1145/2976749.2978357`.

[KPR18]   Marcel Keller, Valerio Pastro, and Dragos Rotaru. Overdrive: Making SPDZ great again. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EURO-CRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 158–189. Springer, Heidelberg, April / May 2018. `doi:10.1007/978-3-319-78372-7_6`.

[Lin21]   Yehuda Lindell. Secure multiparty computation. *Commun. ACM*, 64(1):86–96, 2021. `doi:10.1145/3387108`.

[RAB+22]   Jennie Rogers, Elizabeth Adetoro, Johes Bater, Talia Canter, Dong Fu, Andrew Hamilton, Amro Hassan, Ashley Martinez, Erick Michalski, Vesna Mitrovic, Fred D. Rachman, Raj C. Shah, Matt Sterling, Kyra VanDoren, Theresa L. Walunas, Xiao Wang, and Abel N. Kho. Vaultdb: A real-world pilot of secure multi-party computation within a clinical research network. *CoRR*, abs/2203.00146, 2022. URL: `https://doi.org/10.48550/arXiv.2203.00146`, `arXiv:2203.00146`, `doi:10.48550/ARXIV.2203.00146`.

[Sha79]   Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979. `doi:10.1145/359168.359176`.

[Yao86]   Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986. `doi:10.1109/SFCS.1986.25`.

[Zam22]   Zama. Concrete: TFHE Compiler that converts python programs into FHE equivalent, 2022. `https://github.com/zama-ai/concrete`.

[Zha11]   Bingsheng Zhang. Generic constant-round oblivious sorting algorithm for MPC. In Xavier Boyen and Xiaofeng Chen, editors, *Provable Security - 5th International Conference, ProvSec 2011*, volume 6980 of *Lecture Notes in Computer Science*, pages 240–256. Springer, 2011. `doi:10.1007/978-3-642-2 4316-5\_17`.