Detering Dishonest Behaviour in Private Information Retrieval

Casey Devet

University ofWaterloo
cjdevet@cs.uwaterloo.ca

Abstract

It is useful to support Byzantine robustness in a Private Information Retrieval (PIR) protocol so that any database operators that are being used cannot maliciously affect the records that a client is looking for. We analyze the problem of Byzantine servers in Goldberg's IT-PIR protocol and relate the problem to the well known stag hunt game. To make this game more applicable, we expand it to include more than two agents, we define a multi-round game and incorporate the idea of punishing dishonest behaviour. Using the relationship between this problem and the that of collusion between PIR servers, we design a mechansim to deter servers from Byzantine behaviour.

1 Introduction

The goal of *Privacy Enhancing Technologies* (PETs) is to give control of the dissemination of personal information to the user(s) that the information pertains to. PETs rely on underlying cryptographic primitives to provide a guarantee of privacy to users. These primitives are generally facts from the field of *information theory* or *computational complexity theory.* Many cryptographic protocols use the latter, relying on assumptions about the infeasibility of solving a specific problem with a limited amount of computing resources. The advantage of the former (information theory) approach over the latter (computational complexity theory) is that it provides the guarantee that no amount of computing resources will allow an adversary to discover the user's private information. However, using information-theoretic primitives instead of those from computational complexity theory requires some equally strong assumption to prove the protocol's privacy guarantees. An assumption used in many PETs, including mix networks [5], secret sharing [27], onion routing [8] and some voting protocols [3, 25], is that no more than some threshold of agents are colluding against the user to discover the private information.

Noncooperative game theory studies the dynamics of situations in which there are multiple self-interested individuals whose actions affect the amount of utility gained (or lost) of all agents. Game theory can be used to evaluate the possible actions of agents in a PET protocol that relied on a non-collusion assumption. This information can be used to design a mechanism that disincentivizes agents from colluding or performing other malicious actions.

1.1 Private Information Retrieval

Private Information Retrieval (PIR) is a PET that allows a user to query a database for some record(s) without letting the database server operator learn anything about the query or the retrieved record(s). The most simple form of PIR is for the client to download the entire database and do the query herself, ensuring the privacy of her query. In a 2007 study, Sion and Carbunar concluded that no PIR protocol would likely outperform this trivial download PIR protocol [28]. However, more recent work has shown that there are indeed non-trivial PIR protocols that perform better than downloading the entire database [23]. PIR has applications in many privacy-sensitive applications, including patent databases [1], domain name registration [22], anonymous email [26], and anonymous communication networks [20].

As a PET, a PIR protocol gets its privacy guarantees from its underlying cryptographic primitives.

One class of PIR protocols, called *computationally secure PIR* (CPIR) uses public-key cryptography to encode the query in such a way that the database server can respond correctly, while learning nothing about the query or retrieved record(s). There is at least one CPIR protocol that performs better than the trivial download protocol [23]. The other class of PIR protocols, called *information-theoretic PIR* (IT-PIR) do not rely on the assumption that a cryptographic primitive is possibly hard to solve with limited computing resources. In 1998, Chor et al. showed that non-trivial IT-PIR is impossible when there is only a single database server [4]. They further designed a multi-server IT-PIR protocol that guaranteed privacy as long as not all of the servers are colluding together against the user. Several IT-PIR protocols have since been proposed [2,9,10,15], all using the same non-collusion assumption. In 2001, Olumofin and Goldberg showed that a number of these multi-server IT-PIR protocols perform better than the trival download PIR. This paper is primarily concerned with IT-PIR protocols and the dynamics between self-interested database servers.

1.2 Related Work

Recently there has been much research in the intersection of the fields of cryptography and game theory. There are some works that use game theory to develop realistic adversarial models for cryptographic protocols. Other works use cryptography to design mechanisms that acheive certain wanted equilibria. As we are particularly interested in secret sharing, we focus on these.

In 2004, Halpern and Teague study a particular secret sharing problem using game theory [12]. They consider a situation where there are ℓ agents, each with a share of a secret and that $t+1$ of these shares are needed to recover the secret. They assume that an agent wants to uncover the secret, but strictly prefers to minimize the number of agents that also learn the secret. Halpern and Teague propose a novel solution to the problem and many other works have extended this result [6,11,17,21]. Others have studied the effect of this problem on secure multi-party computation [16,18,19], however in these works the goal is to incentivize cooperation between agent, where we are interested in disincentivizing this behaviour. Game theory has also been used to study the dynamics of privacy and anonymity including Acquisti et al.'s study of the economics of anonymity and other works that attempt to incentivize behaviour of agents in some PETs to behave in a way that most benefits the PETs' users [8].

Henry et al. study the problem of collusion in a specific IT-PIR protocol [13]. They show that if the database servers are all self-interested, revenue-maximizing agents, then the grand coalition (all servers colluding) is a weakly dominant Nash equilibrium and a strictly dominant Bayesian Nash equilibrium. That is, collusion is very likely to occur and the privacy guarantee is not very strong. Henry et al. also propose a mechanism that shifts the equilibrium to non-collusion and show that, with some assumptions about the number of servers that are collude without discovering the query, the non-collusion outcome is the only Nash equilibrium. This means that we can reasonably incentivize non-collusion in this setting.

2 Goldberg's IT-PIR

For the purposes of this paper, we will consider a database in the form of an $r \times s$ matrix D whose elements are of some field $\mathbb{F}$. Each row of the matrix is a record that is indexed by the row index of the matrix. We can get the β-th record of such a database by multiplying an elementary vector $\mathbf{e}_\beta$, containing all zeros except for a one in the β-th position, by the database matrix D (see Figure 1). This simple form of a database can be extended to support more advanced queries, including keyword searches and SQL queries [22].

In [4], Chor et al. propose an IT-PIR protocol that requires there to be $\ell \geq 2$ servers, each with a copy of the database and with the database field $\mathbb{F}$ the binary field. To hide the query, the client chooses $\ell - 1$ vectors $\mathbf{v}_1, \dots, \mathbf{v}_{\ell-1}$ uniformly at random from $\mathbb{F}^r$ and computes $\mathbf{v}_\ell = \mathbf{e}_\beta \oplus (\mathbf{v}_1 \oplus \cdots \oplus \mathbf{v}_{\ell-1})$. The client sends $\mathbf{v}_i$ to server i for each $1 \leq i \leq \ell$, the server computes $\mathbf{r}_i = \mathbf{v}_i \cdot D$ and sends it back to the client. The client then computes

$$\begin{aligned} \mathbf{r} &= \mathbf{r}_1 \oplus \cdots \oplus \mathbf{r}_\ell \\ &= (\mathbf{v}_1 \oplus \cdots \oplus \mathbf{v}_\ell) \cdot D \\ &= \mathbf{e}_\beta \cdot D, \end{aligned}$$

which is the requested database record. This protocol is $(\ell-1)$-*private*, which means no combination of $\ell - 1$ or fewer servers can collude to discover the query. So this protocol guarantees privacy of the query so long as not all ℓ servers are colluding.

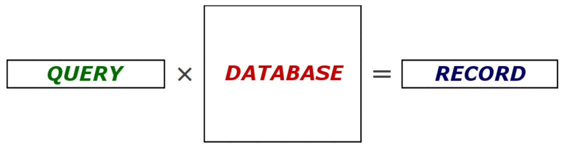

Figure 1: Querying a matrix in the form of an $r \times s$ matrix.

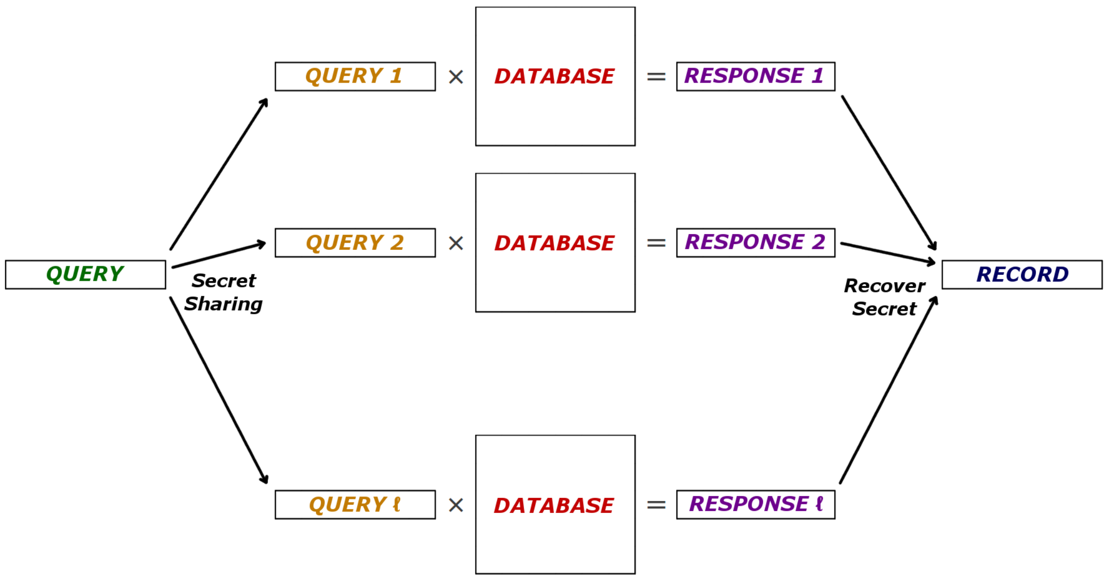

Figure 2: Goldberg's IT-PIR scheme illustrated.

In 2007, Goldberg introduced a generalization of Chor et al.'s protocol [10]. Instead of using $\mathbb{F} = GF(2)$ as in Chor et al., Goldberg let's the field $\mathbb{F}$ be any finite field. So the database is an $r \times s$ matrix whose elements are of this finite field $\mathbb{F}$. In this protocol, the client must choose a value $t < \ell$, called the *privacy level*, that represents the number of database servers that will be able to collude but not recover any information about the query. The client then uses Shamir secret sharing [27] to create ℓ shares $\mathbf{v}_1, \ldots, \mathbf{v}_\ell$ of the query vector $\mathbf{e}_\beta$, one share for each server. Each server i then computes a response $\mathbf{r}_i = \mathbf{v}_i \cdot D$, which it send back to the client. By this linearity of Shamir secret sharing, the responses $\mathbf{r}_1, \ldots, \mathbf{r}_\ell$ are secret shares for the vector D_β, the record the client queries and can be found using any $t + 1$ of the response vectors.

2.1 Robustness

One of the advantages of Goldberg's protocol is that it has built-in robustness. That is, not all of the servers need to reply for the client to be able to recover the database record. This is because we can set t to be any value less than ℓ and only $t+1$ of the servers need to respond. So we can trade off how strong the collusion assumption is to allow for more servers to not respond. This is an advantage because naturally, some of the database servers may be down or have communication errors and the robustness of Goldberg's protocol allows us to still have private database querying even when this happens. For the duration of this paper, $k \leq \ell$ will denote the number of servers that send a query response to the client.

Another similar problem occurs when some of the database servers send corrupt or maliciously altered responses. Goldberg's protocol can also overcome this problem of Byzantine servers. It does this by treating the server responses $\mathbf{r}_1, \ldots, \mathbf{r}_k$ as Reed-Solomon codewords for the database record D_β. Reed-Solomon error correction algorithms can be used to recover the database record even if some of the servers are Byzantine. Devet et al. [7] show that this error correction can be successfully carried out in polynomial time as long as the number of Byzantine servers v is at most $v^* = k - t - 2$. They also show that this is the optimal bound on the number of Byzantine servers allowed to successfully recover the database record. An advantage of using Reed-Solomon error correction is this way, is that as long as the number of Byzantine servers is below the tolerable bound ($v \leq v^*$), on top of being able to recover the database record, we will be able to discover which servers sent incorrect responses.

2.2 Multi-Block Queries

Henry et al. leverage the Reed-Solomon decoding algorithms used by Devet et al. to allow for a client to request multiple database records at once, using the same amount of communication as for one query [14]. This ability greatly improves the throughput of Goldberg's protocol, but it comes at a cost. Henry et al. show that if we attempt to query for q blocks of the database at once, the bound on the number of Byzantine servers v is reduced from $v^* = k-t-2$ to $v^* = k-t-q-1$. This shows that we can trade off some Byzantine robustness for an improvement in the throughput of Goldberg's protocol.

3 Model

In this paper, we are interested in whether or not servers are more inclined to be Byzantine or if they prefer to be honest and provide correct responses. We use a game-theoretic model to analyze this problem.

We assume that there are ℓ database servers that are queried by a client using Goldberg's IT-PIR protocol outlined in Section 2. As we are not concerned with the robustness problem (servers not responding), we will assume that all ℓ servers respond, so $k = \ell$. The privacy level for the protocol is set at some constant $t < \ell$. We will first assume that multi-block queries (outlined in Section 2.2) are not being used. This means that the bound on the number of Byzantine servers v for successful record recovery is $v^* = k - t - 2$.

We are interested in the decisions of the servers, so we must outline the utility functions applicable for each server to understand their actions. We will assume that each server is somehow rewarded for responding the a query and that each server i gets positive utility $p_i > 0$ for participating in a PIR query. We also assume that any server caught sending Byzantine responses will not receive her reward. Finally, servers get some type of satisfaction out of being able to send a Byzantine response that is not caught and hence makes the client unable to recover the database record from the set of responses. We will represent this utility for server i as $r_i > 0$.

The actions available to each server are $A_i = \{\mathbf{B}, \mathbf{H}\}$ where **B** represents the server choosing to send a Byzantine response and **H** represents the server choosing to be honest.

The game has the following outcomes:

If server i chooses action **B**, then

- If the number of Byzantine servers v is greater than v^*, then server i has utility $p_i + r_i$.
- Otherwise, server i has utility 0.

If server i chooses action **H**, then server i has utility p_i.

For example, consider a game with 2 servers and $v^* = 1$. Then the game can be described as follows:

	B	**H**
B	$p_1 + r_1, p_2 + r_2$	$0, p2$
H	$p_1, 0$	p_1, p_2

3.1 The Stag Hunt Game

We can easily see that this game is an example of a stag hunt game. The stag hunt is a game where two people both have a choice between hunting a stag or hunting a hare. Hunting the stag will give the hunters more meat than hunting a hare, however both hunters must hunt the stag together to be successful, while only one hunter needs to hunt a hare to be successful. In general, a stag hunt has the form

	S	**H**
S	a_1, a_2	c_1, b_2
H	b_1, c_2	d_1, d_2

where $a_i > b_i \geq d_i > c_i$.

The stag hunt is very similar to the Prisoner's Dilemma in that there is one strategy that is best for the collective and another that is safe. However, unlike the Prisoner's Dilemma, the strategy that is best for the collective is not strictly dominated by the other strategy. Some researchers consider the stag hunt to be just as useful in representing social cooperation as the Prisoner's Dilemma [29]. The stag hunt is commonly used in research on evolutionary game theory [24].

The stag hunt has two pure strategy Nash equilibria. One of these is when both hunters hunt the stag and the other is when both hunters hunt for hare. The stag equilibrium Pareto dominates the hare equilibrium, so it is better collectively for all hunters to hunt the stag. However, the hare equilibrium is less risky because a hunter hunting a hare has a higher level of guaranteed utility than if he hunts the stag. The stag hunt also has a mixed strategy Nash equilibrium.

The stag hunt can easily be generalized to the N-person stag hunt in which N hunters each have to choose between hunting stag or hare and unless some threshold m of the hunters choose to hunt the stag, all of those hunting stag will not be successful. This very closely resembles the problem that we are interested in.

3.2 Understanding the Equilibria

We now consider our game again. As with the stag hunt, there are two pure strategy equilibria. The first is when all servers choose to be Byzantine (**B**) and the other is when all servers choose to be honest (**H**). The second is only a Nash equilibrium when $v^* \geq 1$, however, if $v^* = 0$, we have no Byzantine tolerance and our problem becomes meaningless and so we choose $v^* \geq 1$.

There is also a mixed strategy Nash equilibrium when, for all servers i, the probability that $v > v^*$, given that server i is Byzantine, is $\frac{p_i}{p_i + r_i}$.

The "all Byzantine" equilibrium is the only strong Nash equilibrium (equivalently, it Pareto dominates the others), and so is the most appealling to the servers. However, the "all honest" equilibrium is less risky in case some of the servers decide to be honest.

As these equilibria make clear, a server is more likely to be Byzantine if she is confident that other servers will be Byzantine, but will be more likely to be honest if she cannot be sure that enough servers will be Byzantine to not get caught. Because of this, this problem is very much related to the problem of collusion between servers. If servers are colluding, they may be able to form a large enough coalition to be Byzantine and not get caught, and in doing so make the client unsuccessful in receiving the correct database record. Even if the coalition is not large enough for that, it will improve a server's knowledge of what others are doing and help to inform them what the better option is.

4 Multi-Round Game

Although the above game gives great insight into this problem, it is more realistic to consider this problem as a multi-round game. This is because a database server is not being utilized very well if it is only going to be used once. Generally, there would be multiple queries and multiple rounds of the game are needed.

If we consider a finite-length multi-round game, we see can easily show through backwards induction that the "always be Byzantine" and "always be honest" strategies are still Nash equilibria. However, in general, a server will not know how many rounds will be played, so this model is not ideal. For this reason, we will consider infinite multi-round games.

4.1 Punishing Byzantine Servers

As mentioned in Section 2.1, if the number of Byzantine servers v is not greater than the bound v^*, then the client will know which servers have not been honest. Because of this, the client can change the game to punish these servers.

One way of punishing those Byzantine servers that are caught is to reduce the payment that they get for responding to a query, p_i. A disadvantage of this approach is that as p_i decreases, it may become more enticing for a server to behave dishonestly. Because of this, we will choose to punish the dishonest servers by not querying them again.

We represent this using the following stochastic game:

- The set of servers $N = \{1, 2, \ldots, k = \ell\}$.
- The set of games $Q = \{G_X : X \subseteq N\}$.

  For a game G_X, we call X the set of honest servers.

  If $|X| \geq t + 3$, then we define G_X in the same way as our single round game in Section 3, except that the utility of any server not in X will be 0 for all actions.

If $|X| < t+3$, then we do not have enough honest servers to make a query, so the utility of all servers will be 0 for all actions.

- The set of action profiles $A = A_1 \times \cdots A_k$, where $A_i = \{\mathbf{B}, \mathbf{H}\}$.
- The transition function $P : Q \times A \to Q$ which returns the next game state based on the current game state and action profile. For $G_X \in Q$ and $a = (a_1, \ldots, a_k) \in A$ we define P as:

$$P(G_X, a) = \begin{cases} G_X & : |B(X,a)| \leq v^*(X) \\ G_{X \setminus B(X,a)} & : otherwise \end{cases}$$

where $B(X,a) = \{i \in X : a_i = \mathbf{B}\}$ is the set of Byzantine servers in X for action profile a and $v^*(X) = |X| - t - 2$ is the bound on the number of Byzantine servers when we are querying only the servers X.

Each game starts in game state $q_0 = G_N$ and in each round $j \geq 1$, an action profile $a^{(j)} \in A$ is chosen by the servers. The game then changes to game state $q_j = P(q_{j-1}, a^{(j)})$. This continues for infinitely many rounds.

Note that if $|X| < t+3$ in some round, we enter the zero utility game and never transition back to a game state $G_{X'}$ with $|X'| > t$. This represents the end of the game as no servers will gain or lose any more utility. This is done so that no round has has a Byzantive server bound of $v^* < 1$.

4.2 Evaluating the Multi-Round Game

When considering strategies of this game, we will use the future discounted rewards method. That is, for the j-th round in the future (the 0-th round being the current round), we will discount the utility by multiplying by δ^j where $0 < \delta < 1$. Suppose server i has rewards $\mu_i^{(1)}, \mu_i^{(2)}, \ldots$ in rounds $1, 2, \ldots$, respectively, then server i's *future discount reward* is

$$R_i = \sum_{j=1}^{\infty} \delta^j r_i^{(j)}.$$

The discount represents the servers' uncertainty about whether the game will continue another round.

It can be shown that the strategy profiles where everyone is Byzantine every round or everyone is honest every round are Nash equilibria for this game. However, there are likely many more Nash equilibria, as there generally are in infinite games.

Let us consider any strategy profile that is not one of the two above and that is made up of each server's strategy being a single-round mixed-strategy that is chosen every round. Suppose that more than v^* of the servers are using the "always Byzantine" strategy. Then one of the servers i that is not using this strategy would be better off using it, as no other single-round mixed-strategy will be able to have utility as high as $p_i + r_i$. It is unclear whether or not the strategy profile where not greater than v^* of the servers are always Byzantine contains any Nash equilibria. We propose that future work look more closely into this game to classify the Nash equilibria or some other solution concept.

5 Trading Robustness for Throughput

As mentioned in Section 2.2, Henry et al. designed a method for querying for multiple records of a database at once [14]. This is done in such a way that the servers can not distinguish whether the query is for one record or many. This provides the ability to fetch more records with the same amount of communication. However, this increase in throughput comes at the cost of Byzantine robustness.

Devet et al. showed that the client can successfully recover the database record from a set of server responses if at most $v^* = k - t - 2$ of the responses are maliciously changed. This is the optimal Byzantine robustness for polynomial-time Reed-Solomon list decoding [7]. Henry et al. show that if we want to query the servers for q queries at once, the Byzantine robustness is $v^* = k - t - q - 1$ [14].

We can use this trade off between throughput and Byzantine robustness in our game. Henry et al. suggest that it may be worthwhile to decrease the robustness to almost nothing for most queries, but once in while increase the robustness to its maximum to check for any large-scale Byzantine behaviour.

We can change our game to incorporate this idea:

- Change the set of states to $Q' = \{G_X, G'_X : X \subseteq N\}$. The state G_X is as defined above. The state G'_X will have no Byzantine robustness.

- Change transition function to $P' : Q' \times A \times Q' \to [0,1]$. $P'(p, a, \hat{p})$ is the probability that $\hat{p}$ will be the next state given that we are in state p with action profile a. We define the probability that we are in the maximum Byzantine robust round as γ. We define P' as follows in terms of P (defined above):

  If $P(G_X, a) = G_Y$, then

$$P'(G_X, a, G_Y) = \gamma$$
$$P'(G'_X, a, G_Y) = \gamma$$
$$P'(G_X, a, G'_Y) = 1 - \gamma$$
$$P'(G'_X, a, G'_Y) = 1 - \gamma$$

- With probability γ we start in state G_N and with probability $1 - \gamma$ we start in state G'_N.

Essentially, we add a new non-robust state for each of the old states and after each round we go to the robust state with probability γ and to the non-robust state with probability $1 - \gamma$.

We leave finding solutions to this game for future work.

6 Connection to Collusion Problem

As mentioned in Section 3.2, this problem is closely related to the problem of collusion between database servers. The reason for this is because a server that has knowledge that other servers are going to be Byzantine is more likely to be Byzantine. This knowledge can only come from collusion of some sort.

Recall that for a PIR query to to be private, there must not be any collusion of $t + 1$ or more servers. Suppose that we have some method of ensuring that no collusions of size greater than t are made. We can use this method to create a mechanism to deter servers from Byzantine behaviour.

To do this, we make one assumption: a database server will not choose to by Byzantine unless it knows of at least v^* other servers that are going to be Byzantine.

We can then use our original multi-round game to model the problem. We will ensure that for every round the number of honest servers $|X|$ is at least $2t + 3$. Then due to our assumption and that $v^* = k - t - 2 \geq t + 1$, no servers will ever choose to be Byzantine.

It is important to note that this assumption that we make is not very strong. It is definitely possible for a server to risk being caught even if she does not know whether or not there are even Byzantine servers to be safe. We leave it for future work to study the relationship between the server collusion problem and our Byzantine server problem.

7 Conclusions

The problem of Byzantine behaviour in multi-server IT-PIR can be modelled using game theory to try to find the behaviour of rational self-interest database server operators. This problem is a form of the stag hunt problem, which has two pure strategy equilibria, one when all of the servers are Byzantine and one when all servers are honest. This reveals the close relationship between this problem and the problem of collusion between PIR servers. The Byzantine server problem becomes more interesting and applicable as a multi-round game. However, because of the complexity of the stochastic game being used, we did not discover much about different solution concepts for this infinite game. Finally, we redefine our game to take into account a trade off between throughput and Byzantine robustness. Using this, we hope to be able to show that we can improve the throughput of our IT-PIR scheme but still support Byzantine robustness to a high enough level.

References

[1] D. Asonov. Private Information Retrieval: An overview and current trends. In *ECDPvA Workshop*, 2001.

[2] A. Beimel, Y. Ishai, and T. Malkin. Reducing the servers' computation in private information retrieval: Pir with preprocessing. *J. Cryptology*, 17(2):125–151, 2004.

[3] D. Chaum, R. Carback, J. Clark, A. Essex, S. Popoveniuc, R. L. Rivest, P. Y. A. Ryan, E. Shen, A. T. Sherman, and P. L. Vora. Scantegrity ii: end-to-end verifiability by voters of optical scan elections through confirmation codes. *IEEE Transactions on Information Forensics and Security*, 4(4):611–627, 2009.

[4] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan. Private information retrieval. *J. ACM*, 45:965–981, November 1998.

[5] G. Danezis, R. Dingledine, and N. Mathewson. Mixminion: Design of a type iii anonymous remailer protocol. In *IEEE Symposium on Security and Privacy*, pages 2–15. IEEE Computer Society, 2003.

[6] V. Dani, M. Movahedi, Y. Rodriguez, and J. Saia. Scalable rational secret sharing. In C. Gavoille and P. Fraigniaud, editors, *PODC*, pages 187–196. ACM, 2011.

[7] C. Devet, I. Goldberg, and N. Heninger. Optimally Robust Private Information Retrieval. In *21st USENIX Security Symposium*, 2012.

[8] R. Dingledine, N. Mathewson, and P. Syverson. Tor: the second-generation onion router. In *13th USENIX Security Symposium*, 2004.

[9] Y. Gertner, S. Goldwasser, and T. Malkin. A Random Server Model for Private Information Retrieval. In *2nd International Workshop on Randomization and Approximation Techniques in Computer Science*, pages 200–217, 1998.

[10] I. Goldberg. Improving the robustness of private information retrieval. In *2007 IEEE Symposium on Security and Privacy*, pages 131–148, 2007.

[11] S. D. Gordon and J. Katz. Rational secret sharing, revisited. In R. D. Prisco and M. Yung, editors, *SCN*, volume 4116 of *Lecture Notes in Computer Science*, pages 229–241. Springer, 2006.

[12] J. Y. Halpern and V. Teague. Rational secret sharing and multiparty computation: extended abstract. In L. Babai, editor, *STOC*, pages 623–632. ACM, 2004.

[13] R. Henry, T. Elahi, and Y. Huang. Trustworthy pir. Technical report, University of Waterloo, 2012.

[14] R. Henry, Y. Huang, and I. Goldberg. One (Block) Size Fits All: PIR and SPIR Over Arbitrary-Length Records via Multi-block PIR Queries. In *20th Network and Distributed System Security Symposium*, 2013.

[15] R. Henry, F. G. Olumofin, and I. Goldberg. Practical pir for electronic commerce. In Y. Chen, G. Danezis, and V. Shmatikov, editors, *ACM Conference on Computer and Communications Security*, pages 677–690. ACM, 2011.

[16] S. Izmalkov, S. Micali, and M. Lepinski. Rational secure computation and ideal mechanism design. In *FOCS*, pages 585–595. IEEE Computer Society, 2005.

[17] G. Kol and M. Naor. Games for exchanging information. In C. Dwork, editor, *STOC*, pages 423–432. ACM, 2008.

[18] M. Lepinski, S. Micali, C. Peikert, and A. Shelat. Completely fair sfe and coalition-safe cheap talk. In S. Chaudhuri and S. Kutten, editors, *PODC*, pages 1–10. ACM, 2004.

[19] A. Lysyanskaya and N. Triandopoulos. Rationality and adversarial behavior in multi-party computation. In C. Dwork, editor, *CRYPTO*, volume 4117 of *Lecture Notes in Computer Science*, pages 180–197. Springer, 2006.

[20] P. Mittal, F. Olumofin, C. Troncoso, N. Borisov, and I. Goldberg. PIR-Tor: Scalable Anonymous Communication Using Private Information Retrieval. In *20th USENIX Security Symposium*, pages 475–490, 2011.

[21] M. Nojoumian. Socio-rational secret sharing as a new direction in both rational cryptography and game theory. *IACR Cryptology ePrint Archive*, 2011:370, 2011.

[22] F. Olumofin and I. Goldberg. Privacy-preserving queries over relational databases. In *10th International Privacy Enhancing Technologies Symposium*, pages 75–92, 2010.

[23] F. Olumofin and I. Goldberg. Revisiting the Computational Practicality of Private Information Retrieval. In *15th International Conference on Financial Cryptography and Data Security*, pages 158–172, 2011.

[24] J. M. Pacheco, F. C. Santos, M. O. Souza, and B. Skyrms. Evolutionary dynamics of collective action in n-person stag hunt dilemmas. *Proceedings of the Royal Society B: Biological Sciences*, 276(1655):315–321, 2009.

[25] P. Y. A. Ryan and S. A. Schneider. Prêt à voter with re-encryption mixes. In D. Gollmann, J. Meier, and A. Sabelfeld, editors, *ESORICS*, volume 4189 of *Lecture Notes in Computer Science*, pages 313–326. Springer, 2006.

[26] L. SDDBLP:conf/podc/DaniMRS11oassaman, B. Cohen, and N. Mathewson. The Pynchon Gate: a Secure Method of Pseudonymous Mail Retrieval. In *Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society (WPES '05)*, pages 1–9, 2005.

[27] A. Shamir. How to share a secret. *Commun. ACM*, 22:612–613, November 1979.

[28] R. Sion and B. Carbunar. On the computational practicality of private information retrieval. In *Proceedings of the Network and Distributed Systems Security Symposium*, 2007.

[29] B. Skyrms. *The Stag Hunt and the Evolution of Social Structure*. Cambridge University Press, 2004.