A Model for Secure and Mutually Beneficial Software Vulnerability Sharing

Alex Davidson
Royal Holloway
University of London
alex.davidson.2014@rhul.ac.uk

Gregory Fenn
Royal Holloway
University of London
gregory.fenn.2014@rhul.ac.uk

Carlos Cid
Royal Holloway
University of London
carlos.cid@rhul.ac.uk

ABSTRACT

In this work we propose a model for conducting efficient and mutually beneficial information sharing between two competing entities, focusing specifically on software vulnerability sharing. We extend the two-stage game-theoretic model proposed by Khouzani *et al.* [18] for bug sharing, addressing two key features: we allow security information to be associated with different categories and severities, but also remove a large proportion of player homogeneity assumptions the previous work makes. We then analyse how these added degrees of realism affect the trading dynamics of the game. Secondly, we develop a new private set operation (PSO) protocol that enables the removal of the trusted mediation requirement. The PSO functionality allows for bilateral trading between the two entities up to a mutually agreed threshold on the value of information shared, keeping all other input information secret. The protocol scales linearly with set sizes and we give an implementation that establishes the practicality of the design for varying input parameters. The resulting model and protocol provide a framework for practical and secure information sharing between competing entities.[1]

1. INTRODUCTION

Cybersecurity is of crucial importance in today's economy, affecting businesses from a range of sectors, from telecommunications and finance to energy, healthcare and transportation. However, it is clear that businesses cannot work alone in protecting their digital assets from cyber threats, even in highly competitive sectors. It is therefore widely acknowledged that the gathering and exchange of security intelligence are key factors in enhancing the effectiveness of cybersecurity measures, and pivotal to the protection of the modern economy. This recognition has given rise to a range of initiatives – both in the public and the private sectors – to facilitate, galvanise and coordinate the exchange of cybersecurity information between businesses. In the UK, the Cyber-security Information Sharing Partnership (CiSP), which is part of CERT-UK, is a "joint industry government initiative to share cyber threat and vulnerability information in order to increase overall situational awareness of the cyber threat and therefore reduce the impact on UK business"[2]. In the USA, the Cybersecurity Information Sharing Act was signed in December 2015 with the goal of improving "cybersecurity in the United States through enhanced sharing of information about cybersecurity threats, and for other purposes"[3]. In the private sector, we have witnessed the launch of a number of businesses and schemes offering platforms for the exchange of cybersecurity intelligence between parties in a secure and trusted manner (e.g. ThreatStream[4] and Facebook's ThreatExchange[5]). In addition, there has been much recent work in the development of standards for representing and exchanging *threat* information, namely the OASIS standards TAXII, STIX and CybOX.[6]

However, while these tools and initiatives can provide useful and efficient *platforms* for exchange of cybersecurity information, the role of *incentives* must not be ignored. In fact, despite a choice of trusted exchange platforms, one could argue that in many cases there may be incentives for *not* sharing some types of cybersecurity information. For instance, "public disclosure" of security breach incidents can harm consumer and investor confidence and lead to significant decreases in the market value of firms [2, 10].

In this paper, we build from an initial proposal by Khouzani et al. [18], and further develop the game theoretic modelling for the exchange of bug-knowledge between competing organisations that use a common platform. Our modelling is motivated by incidents such as the discovery of the **Heartbleed** bug in the popular **OpenSSL** cryptographic library, which was exposed in April 2014, and by the observation that cybersecurity information sharing appears to be more developed and prevalent in some particular business sectors, e.g. the financial sector.

[1] The authors were supported by the EPSRC and the UK government as part of the Centre for Doctoral Training in Cyber Security at Royal Holloway, University of London (EP/K035584/1).

WISCS'16, October 24 2016, Vienna, Austria

© 2016 ACM. ISBN 978-1-4503-4565-1/16/10…$15.00

DOI: http://dx.doi.org/10.1145/2994539.2994547

[2] https://www.cert.gov.uk/cisp/

[3] https://www.congress.gov/bill/114th-congress/senate-bill/754

[4] https://www.threatstream.com/

[5] https://www.facebook.com/threatexchange/

[6] https://www.oasis-open.org/

1.1 Our contributions and paper layout

In Section 3 we extend the model by Khouzani et al. [18] to significantly weaken numerous assumptions in developing a more complete and realistic interpretation of the information sharing dynamics between two competitive entities. Specifically, we examine a scenario where bugs can be taken from different categories and where the effect of exploitation is defined independently for the two parties. Our analysis focuses on a bilateral sharing scenario where we make use of a generalised mediator who facilitates fair trades during the sharing phase of the game. We also describe numerical results that corroborate and illustrate the results. The weakened assumptions on the economic model means that it applies more readily to real-life situations where competitive organisations evaluate the viability of information sharing.

In Section 4 we establish the exact functionality provided by the mediator for the generalised case where bugs take on a given severity level associated with the harm they cause to the owner of the bug. In Section 5 we provide a novel private set operation (PSO) protocol that enables mutually beneficial trades prescribed by our economic model without the need for third-party mediation, and which is secure in the presence of semi-honest adversaries. The protocol is asymptotically optimal as the computation and communication complexities are both linear in the size of the input sets. This result allows each of the players to learn new bugs up to the point where a mutually agreed threshold is reached; bugs already known to both players, or falling beyond the threshold, are kept private to the input player. We see this as a big advantage in a landscape where finding trusted third parties for multiple parties is a challenge, especially since specific services provide focal points for attackers who may benefit from learning sensitive information about such trades. Moreover, we acknowledge that in the post-Snowden era it is important to limit the amount of knowledge being passed to outsiders regardless of trust assumptions.

Finally in Section 6 we show that our protocol is practical regarding both running times and communication overheads in a proof-of-concept `Go` implementation that is run on commodity hardware. The entire contribution serves as a foundation for initiating and performing information sharing between competitive entities where private knowledge is regarded as too valuable to reveal without receiving intelligence of a suitable value in return.

For all experiments described in this paper, the source code and data is available on request.

2. PRELIMINARIES

2.1 Notation

The paper splits naturally into two parts, with Section 3 concerning the game theoretic findings of our work, while Sections 4 and 5 analyse the possibility of instantiating the mediator via PSOs. As such our notation will differ slightly between these sections to maintain consistency with the notation used in related literature.

In Section 3, we refer to the two economic agents involved in security investment and information trading as i and j. The sets of bugs that is then offered in a trade between them will be denoted by $\boldsymbol{s_i}$ and $\boldsymbol{s_j}$.

In Sections 4 and 5 we will refer to protocol participants by P_1 and P_2 (for the two-party protocol case) with sets S_1 and S_2 where $n = |S_1|$ and $m = |S_2|$. The severity of bugs across the whole paper will be considered in a levelled structure where there are W levels and w refers to a specific level in $[W] = \{1, 2 \ldots, W\}$.

For an array-like structure A, we let $A[i]$ refer to the i-th element in A. Finally, let Γ be an additively homomorphic encryption (AHE) scheme and c_1, c_2 be encryptions of m_1 and m_2. Then $+_H$ denotes the additive operation such that $D_{\mathsf{sk}}(c_1 +_H c_2) = m_1 + m_2$ For a constant κ, $c_1 \cdot \kappa$ denotes a scalar multiplication of the underlying ciphertext.

2.2 Information sharing economics

The need for studying the economics of sharing information derives from the conflicting incentives to maintain an advantage over rivals, while supporting allies. Typically information has a high cost to generate, for example in R&D, but is cheap to distribute, such as over a secure internet connection. Sometimes information is given away for free, such as in a health awareness campaign. But usually, information is traded at a price, paid-for with cash or other information in return. Traditional forms of information of economic interest include market data, newspaper and magazine access, or consumer data for advertising purposes. For a guide on information economics, see [29]. Or for a more a theoretical survey on information sharing games, see [30].

Information sharing in the context of cybersecurity is investigated in papers like [11, 13, 24, 31]. For example, Gordon *et al.* present a model for evaluating the effects of US policy on computer systems security, showing that when economic or legal mechanisms incentivise information sharing between private firms, firms invest less in security but the overall network is made more secure [11]. Further research into the effects of US policy, specifically the Sarbanes-Oxley Act, on security information sharing can be found in [13]. Phillips, Ting and Demurjian take a geopolitical approach to the information economics of extreme security situations, such as international disasters or military action [27].

More recently, Laube and Böhme [23] investigate how legally mandatory sharing of security breaches affects the balance of private security investment from detective to preventative measures. They show that when law enforcement for reporting of security breaches is too strong, firms may over-invest in detection and underinvest in prevention. This socially suboptimal situation may also arise when information sharing is too effective.

Continuing in this strain of research, in this paper we specifically examine the economics of security vulnerabilities in tools or products used or sold by rival firms.

2.3 Security investment games

Cyber security, such as network, application, web or hardware security, is a private good. This means that a user derives benefits from using secure products, and conversely firms that offer more secure products can command a higher price. To improve a product or tool, firms invest in security goods such as penetration testing, formal analysis, network redundancy and antivirus software. Security games examine various economic scenarios relevant to cybersecurity, with each game analysing a different behavioural dynamic [12].

For instance, when devices or applications are connected with each other, the security of the whole network depends on the security of each of the individual objects. This means that security is also a public good, as improved security of one firm's products and systems can indirectly improve the security of other firms. The trade-off between investing in private security, and 'free-riding' off the security of others gives rise to interdependent security investment games [16]. For example, Kunreuther and Heal examine how the network effects on security can disincentivise security investment, as players (nodes) can decide to accept the risk from not investing on the hope of their neighbours being secure enough to protect them for free [20]. For a detailed survey of classic research into interdependent security investment games, see [22]. For a discussion on network games in general, see [15].

2.4 Private set operation protocols

Private set operations (PSOs) are an important cryptographic tool that can help multiple participating organisations to strategically collate their private data for further processing (e.g. advanced data mining procedures) without giving their entire input data away. The need for privacy stems from the fact that parties regard their input data secret to maintain client privacy or competitive advantage for example, while learning the output of a given set operation is advantageous for all parties. The applications for private set operations have been established across wide-ranging topics, including location sharing [25] and performing computations over genetic data [14].

The research into PSO protocols stands largely apart from the generic constructions for multi-party computation since custom protocols can be optimised to perform the desired operations much more efficiently. The most researched operations are intersection (PSI) with notable practical designs such as [4, 7, 28], union (PSU) [1, 9], intersection/union cardinality (PSI/PSU-CA) [8], and designs allowing for the computation of multiple operations such as [5, 19].

In this work we describe a new PSO variant that allows for learning new elements from another set up until a threshold value that depends on the value of the elements that are being learnt. The technique that we use in our construction is related to the works of [5, 6, 17] where we use Bloom filters to represent sets and encrypt the entries with a partially homomorphic encryption (PHE) scheme. Our secure information sharing protocol will make use of encrypted Bloom filters along with constructions of oblivious transfer. Both these tools are defined fully in Appendix A.

2.4.1 PSO security model

To provide cryptographic guarantees on the privacy of input sets in a PSO computation we prove security with respect to predefined security models, we detail these models here and use them in the security proof for our protocol in Section 5. Protocols for computing PSOs can be proven secure with respect to either semi-honest or malicious adversaries. Before we show what this means we first define 'computational indistinguishability' for probability distributions:

Definition 2.1. *Let $\mathcal{X} = \{X_\lambda\}_{\lambda \in S}$ and $\mathcal{Y} = \{Y_\lambda\}_{\lambda \in S}$ be probability ensembles indexed by S. We say that these ensembles are computationally indistinguishable if for all probabilistic polynomial time (PPT) distinguishers, $\{\mathcal{D}_n\}_{n \in \mathbb{N}}$, there exists a negligible function $\mathsf{negl} : \mathbb{N} \mapsto [0,1]$ where*

$$|\Pr[\mathcal{D}_n(X_\lambda) = 1] - \Pr[\mathcal{D}_n(Y_\lambda) = 1]| < \mathsf{negl}(n)$$

and we write $\mathcal{X} \simeq \mathcal{Y}$ to denote this.

Now assume that we have a protocol π that is required to securely represent a specific polynomial-time functionality f. Let S_i be the input set for participant P_i for $i \in \{1,2\}$ and let aux_i be a set of auxiliary information that P_i holds (we specifically consider the case where f computes an operation over input sets). For each P_i, define the *view* of the protocol for P_i to be

$$\mathsf{view}_i^\pi(S_1, S_2) = (\mathsf{Inp}_i, r_i, msg_i, \pi(S_1, S_2)_i)$$

where $\mathsf{Inp}_i = (S_i, aux_i)$ is the combined input of P_i to π, r_i are the internal coin tosses of P_i, msg_i is the messages viewed by P_i in the protocol, and $\pi(S_1, S_2)_i$ is the output witnessed by P_i. Then we can formulate the following definition for semi-honest adversaries.

Definition 2.2. *Protocol π securely computes the functionality f in the presence of static semi-honest adversaries if there exist polynomial-time simulators Sim_1, Sim_2 where*

$$\{Sim_1(\mathsf{Inp}_1, f(S_1, S_2))\} \simeq \{\mathsf{view}_1^\pi(S_1, S_2)\}$$

$$\{Sim_2(\mathsf{Inp}_2, f(S_1, S_2))\} \simeq \{\mathsf{view}_2^\pi(S_1, S_2)\}$$

Intuitively, this states that each party's view of the protocol can be simulated using only the input they hold and the output that they receive from the protocol. Notice that we specifically consider the case where one of the players in the computation is corrupted by a semi-honest adversary.

2.5 Canonicalisation of bugs

As part of Section 5 we require that bugs are viewed in unique formats that are commonly known to both participants, allowing for the efficient swapping of these bugs using foundational PSO primitives. In particular we uniquely encode the bugs as elements of $\mathbb{Z}_N$ for a composite integer N so that we can "encrypt the bugs" with Paillier encryption. We justify this assumption based on recent threat intelligence standardisation efforts, such as STIX and CybOX, which aim to define canonical formats for representing different types of security events.

3. THE GAME-THEORETIC MODEL

In the information sharing model developed in [18], it is established that the amount of bug-sharing between two firms using a common platform depends on whether or not the firms believe they gain more of a competitive edge from using bug-free products than the overall harm done to the whole sector following successful attacks. In this case, firms would lose more per bug shared than they would gain from improving the overall market, and so they patch bugs in their own platforms but do not inform their rival firms about it.

In this paper, we build on the base game, but focus instead on further developing the notion of the mediator who can ensure that players set the terms of a trade of bugs between each other, rather than merely unilaterally sending bugs to their competitor. This has two general advantages. The first is that each player can expect a certain return from the other player when they trade their resources, ensuring that they are not disincentivised from sharing security information out of fear of losing their competitive advantage. Without such a mediator, a trade is effectively a unilateral donation of resources from one player to another. In our game, the players are modelled to agree on a pair of packages of bug to trade before sharing information. The second advantage is that the mediator can determine which bugs each player already knows about, so that useless information is not sent. This is useful for modelling trading behaviour straightforwardly in terms of the raw number of bugs sent between participants.

Furthermore, our work also addresses two limitations in the economic model from [18]. Firstly, we consider the case in which bugs to come in a range distinct kinds, such as buffer overflow, weak crypto, unreliable connection, etc – the exact taxonomy of kinds of bugs would depend on the exact situation. Secondly, we reduce the player homogeneity assumptions, and consider the game when bugs are allowed to affect each player with different severities.

3.1 Bug-trading economic model

We consider two firms i and j in a competitive environment, representing therefore an oligopoly market structure. Their competing products rely on the same software, run on machinery with similar configurations, or operate on the same network or server; we simply refer to this as the 'common platform'. The vulnerabilities on the common platform therefore affect both firms. In short, both i and j's products are susceptible to similar security vulnerabilities.

The full economic model is split into two games, with the second stage embedded within the first as a subgame.

1. Players invest in bug-discovery research, at a cost $c_i \geq 0$ for player i, and $c_j \geq 0$ for player j. This will be understood equivalently as choosing a 'security level' $p_i \in [0, 1)$, representing the probability of i discovering an arbitrary bug in the platform. We assume that $p_i(c_i)$ is a strictly increasing, concave function of c_i.
2. The players then trade bug knowledge with each other.

Stage 2 is a well-defined game independent of stage 1, and simply models the decision problem of what trade deals between i and j's resources are mutually acceptable. The rational behaviour of players in stage 1, however, is dependent on stage 2 as players choose their investment strategies (p_i, p_j) with the anticipation of forming trade deals later. In this paper, we only investigate the bug-trading game, stage 2. During this stage, players are modelled to own sets $\mathcal{B}_{iw}$, referring to the set of bugs that i knows of category w. Bug-discovery investment will be left to future research, as our contribution here is focused on the role of the mediator in bug-knowledge sharing.

We will suppose that i and j can agree on the kind of any given bug, and that bugs can be categorised with labels $w = 1, 2, \dots, W$. We assume that any two bugs of a given kind affect players in the same way, so that i is indifferent between any two kind-w bugs. Note that this is not saying that a kind-w bug affects i and j in the same way, only that two kind-w bugs affect i in the same way. Therefore, these labels w should be understood as merely categories of distinct kinds of bugs.

The random variable $B_w \in \mathbb{N}$ is the number of security issues of level w affecting their products. Player i has the belief that there are around λ_{iw} bugs of level w to worry about, and likewise for j. To be precise, the number of bugs of kind w, B_w, are believed by i to follow a Poisson distribution for each $w \in [W]$:

$$B_w \sim Poi(\lambda_{iw}) \tag{1}$$

where the actual realisation of the random variable is b_w. The specific distribution will be unimportant, it is the mean λ_{iw} that turns out to be most relevant to the game. A Poisson distribution is a standard modelling choice for a discrete random variable over $\{0, 1, 2, 3, \dots\}$ where the only other information given is of the expectation (e.g as in [18]).

Besides λ_{iw}, the expected number of bugs of type w, and $\mathcal{B}_{iw}$, the set of such bugs that i has privately discovered, there are $3W$ other parameters that characterises the type of player i. For each category w, i has a vector $(\delta_{iw}, \tau_{iw}, l_{iw}) \in (\mathbb{R}_0^+)^3$ that characterises the severity of a bug of level w as far as i is concerned. They have the following intuitive meanings in our game:

- Bugs that still exist in j's product after the game, but are known (and fixed) in i's product will give a competitive gain to i of δ_{iw} and a competitive loss to j worth δ_{jw}. So δ represents the betterness of a product, modelling the customers who choose i's product over j's because of some bug.
- A w-bug that exists in any firm's product, either i or j's, will harm both firms by some value $\tau_w > 0$. The number τ_{iw} is what i takes as the expected value of τ_w, noting that i and j may disagree about the market harm severity of a kind-w bug. When i and j use the same platform, or produce similar products, τ_{iw} (or τ_{jw}) represents the perceived harm done to the market as a whole when bugs of level w are unpatched, modelling the loss in consumer confidence in the sector that i and j are engaged in.
- Finally, any bug of level w that is unpatched in i's final product will individually harm firm i by some amount modelled as l_{iw}. This could represent the compensation that firms pay to their customers because of defects, or the reduced confidence in that firm to produce high-quality products.

So player i's *type* is $((\delta_{iw}, \tau_{iw}, l_{iw}, \lambda_{iw}, \mathcal{B}_{iw})$ for $w \in [W])$.

3.1.1 Player strategies

In our bug-sharing game, players conduct a trading strategy $(\boldsymbol{s_i}, \boldsymbol{s_j})$ where we have

$$\boldsymbol{s_i} = (s_{i1}, s_{i2}, \dots, s_{iW}) \in \mathbb{N} \times \mathbb{N} \times \cdots \times \mathbb{N}, \tag{2}$$

and s_{iw} represents the number of bugs of kind w that i sends to j. Without loss of generality, we assume that every bug that j sends to i is useful information to i, because giving information to i that it already owns is no different from not sending the information at all. In practice, the ability to tell whether or not j already knows some information that i knows will be actualised with our cryptographic mediator in Sections 4 and 5. In particular, we have that

$$\forall w \in [W],\ 0 \leq s_{iw} \leq b_w - s_{jw} \leq b_w. \tag{3}$$

Players are modelled to work together to choose a strategy pair $(\boldsymbol{s_i}, \boldsymbol{s_j})$ that is maximally beneficial to both players, negotiating a mutually agreeable trade of security information. In this sense, our game can be understood as a coalition game, as the action of player i will be a function of the action of j, made possible with the mediator. Note that real-life firms need not literally negotiate each individual information-sharing arrangement, the game models any situation whereby the set of mutually-agreeable trade deals can be codified, with the optimal arrangement automatically selected. Here, optimal arrangements refer to maximal 'Pareto dominance', whereby $(\boldsymbol{s_i}, \boldsymbol{s_j})$ Pareto dominates $(\boldsymbol{s_i'}, \boldsymbol{s_j'})$ iff

$(\boldsymbol{s_i}, \boldsymbol{s_j})$ results in a greater utility for both i and j than $(\boldsymbol{s'_i}, \boldsymbol{s'_j})$ would. In practice, we expect machines i and j to compute their type and bug-knowledge, and run the cryptographic protocol described later to automate this process. The mediator will also enforce a property that we will call 'fair trading', described in Definition 3.2. This restricts the space of possible action-pairs $(\boldsymbol{s_i}, \boldsymbol{s_j})$.

3.1.2 Utility function

From a competitive firm's selfish point of view, a bug from the category labeled 'w' takes one of four forms.

- $B_{\neg i \neg j:w}$: bugs known to neither player.
- $B_{\neg i j:w}$: bugs only known to j.
- $B_{i \neg j:w}$: bugs only known to i.
- $B_{ij:w}$: bugs known to both players.

These are random variables over $\mathbb{N}$, counting the number of bugs to be discovered after firms have engaged in their private bug-discovery research.

In this paper, the most general utility function for player i, over both stages of the game, is as follows.

Definition 3.1. *In a 2-player oligopoly, the bug information sharing game is characterised by*

$$\begin{aligned} &\mathcal{U}_i(\boldsymbol{s_i}, \boldsymbol{s_j}) \\ &= \sum_{w=1}^{W} \big[-(\tau_{iw} + l_{iw}) \cdot b_{\neg i \neg j:w} + \ (\delta_{iw} - \tau_{iw}) \cdot (b_{i \neg j:w} - s_{iw}) \\ &\qquad - \ (\delta_{iw} + \tau_{iw} + l_{iw}) \cdot (b_{\neg i j:w} - s_{jw}) \ + \ 0 \cdot b_{ij:w} \big] \end{aligned} \tag{4}$$

where s_{jw} is the number of (useful) bugs of level w that j sends to i, and vice versa for s_{iw}, and $b_{i \neg j:w}$ is the realised value of $B_{i \neg j:w}$.

Strictly speaking, $\mathcal{U}_i$ is a discrete function of

$$(s_{iw}, s_{jw}) \in ([0, b_{i \neg j:w}] \cap \mathbb{Z}) \times ([0, b_{\neg i j:w}] \cap \mathbb{Z})$$

for each bug-kind w. But we will idealise the utility function and suppose s_{iw} is a continuum over $[0, b_{i \neg j:w}]$ (and likewise for s_{jw}). This will be useful for framing the model as a calculus problem later. Such an idealisation is justified from experiments described in Section 3.3. Player i is assumed to choose $\boldsymbol{s_i}$ to maximise $\mathcal{U}_i$, given a belief about j's $\boldsymbol{s_j}$.

3.2 Bilateral trading with a mediator

To determine what sets of bugs i would require from j in order to agree to a trade $(\boldsymbol{s_i}, \boldsymbol{s_j})$, we develop here a working definition for a 'fair trade' between i and j, such that both players benefit from the trade in equal (positive) measure. It will be seen that this is a helpful way to determine rational coalition strategies.

In a bug-trade $(\boldsymbol{s_i}, \boldsymbol{s_j})$, we refer to Definition 3.1 and note that player i believes that their utility increases by the amount

$$\mathcal{U}_i^+ = \sum_{w=1}^{W} \left[s_{iw} \cdot (\tau_{iw} - \delta_{iw}) + s_{jw} \cdot (\delta_{iw} + \tau_{iw} + l_{iw}) \right] \tag{5}$$

and likewise for j, flipping the i and j subscripts as usual. Notice we could equivalently define the marginal utility to i, denoted $\mathcal{U}_i^+$, as a function given by:

$$\mathcal{U}_i^+(\boldsymbol{s_i}, \boldsymbol{s_j}) = \sum_{w=1}^{W} \left[s_{iw} \cdot \frac{\partial \mathcal{U}_i}{\partial s_{iw}} \right] + \sum_{w=1}^{W} \left[s_{jw} \cdot \frac{\partial \mathcal{U}_i}{\partial s_{jw}} \right] \tag{6}$$

with $\mathcal{U}_i$ from Definition 3.1. We can now formally define a fair trade.

Definition 3.2. *Let $(\boldsymbol{s_i}, \boldsymbol{s_j})$ be a pair of packages of tradable bugs (bugs not already known to the other party). We say that $(\boldsymbol{s_i}, \boldsymbol{s_j})$ is a "fair trade" iff $\mathcal{U}_i^+(\boldsymbol{s_i}, \boldsymbol{s_j}) = \mathcal{U}_j^+(\boldsymbol{s_i}, \boldsymbol{s_j})$.*

That is, the marginal, or incremental, utility is equal for both players in any fair trade. From now on, we suppose a theoretical mediator limits all possible trading agreements between i and j to fair trades only. We can characterise the effect of a fair trade with the following Lemma, of which we omit the full proof.

Lemma 3.2.1. *In the bug-trading game, let $(\boldsymbol{s_i}, \boldsymbol{s_j})$ be a fair trade. The marginal utility for both players is*

$$\mathcal{U}^+ = \frac{1}{2} \sum_{k \in \{i,j\}} \sum_{w=1}^{W} [s_{kw} \cdot (\tau_{kw} + \tau_{k'w} + l_{k'w} + \delta_{k'w} - \delta_{kw})]$$

where $\{k, k'\} = \{i, j\}$. Conversely, $\mathcal{U}_i^+ = \mathcal{U}^+ \Longrightarrow \mathcal{U}_i^+ = \mathcal{U}_j^+$.

Proof. The result follows from algebraic manipulation. We can use equation (5) to define $\mathcal{U}_i^+$ and $\mathcal{U}_j^+$ (by flipping the i and j subscripts). We then solve $\frac{1}{2} \cdot (\mathcal{U}_i^+ + \mathcal{U}_j^+)$ and let $\mathcal{U}_i^+ = \mathcal{U}_j^+$ to derive the formula. □

We will refer to the value of $\mathcal{U}^+$ as the 'trade-value' of the exchange. Clearly, whenever $\mathcal{U}^+(\boldsymbol{s_i}, \boldsymbol{s_j}) > 0$, both players would prefer to trade their bugs packages $(\boldsymbol{s_i}, \boldsymbol{s_j})$ than to trade nothing at all $(\boldsymbol{0}, \boldsymbol{0})$. But the optimal action is to agree on a package $(\boldsymbol{s_i}, \boldsymbol{s_j})$ to maximise $\mathcal{U}^+(\boldsymbol{s_i}, \boldsymbol{s_j})$. Note that $\mathcal{U}^+$ is maximised iff $\mathcal{U}_i$ is maximised subject to the restriction of trade deals to fair trades. To be more precise, when i and j are forced to engage in a fair trade with each other, i will choose $(\boldsymbol{s_i}, \boldsymbol{s_j})$ to maximise $\mathcal{U}_i$ subject to $\mathcal{U}_i^+ = \mathcal{U}_j^+$.

Proposition 3.1. *In a fair-trade bug-sharing game, the players select $(\boldsymbol{s_i}, \boldsymbol{s_j})$ to solve the following problem:*

$$\texttt{Maximise} \ \sum_{w=1}^{W} [s_{iw} \cdot (\tau_{iw} - \delta_{iw}) + s_{jw} \cdot (\delta_{iw} + \tau_{iw} + l_{iw})]$$

$$\begin{aligned} \textit{such that} \ \sum_{w=1}^{W} [&s_{jw} \cdot (\delta_{jw} + \delta_{iw} + \tau_{iw} + l_{iw} - \tau_{jw}) \\ &- s_{iw} \cdot (\delta_{iw} + \delta_{jw} + \tau_{jw} + l_{jw} - \tau_{iw})] = 0 \end{aligned}$$

$$\textit{and} \ \ \forall w, \ s_{iw} \in [0, \ b_{i \neg j:w}], \ s_{jw} \in [0, \ b_{\neg i j:w}]$$

The following important corollary establishes plausible conditions under which trading behaviour reduces to one player sharing *all* their bug knowledge in exchange for a subset of their rival's.

Corollary 3.1. *Suppose that*

$$|\delta_{iw} - \delta_{jw}| < \tau_{iw} + \tau_{jw} + Min\{l_{iw}, l_{jw}\} \tag{7}$$

and

$$|\tau_{iw} - \tau_{jw}| < \delta_{iw} + \delta_{jw} + Min\{l_{iw}, l_{jw}\} \tag{8}$$

for every bug kind w. Then, in any optimal fair trade arrangement, one (or both) players will trade all of their bugs for a subset of their opponents, i.e.

$$\forall w, \ s_{iw} = b_{i \neg j:w} \ \texttt{or} \ \forall w, \ s_{jw} = b_{\neg i j:w}$$

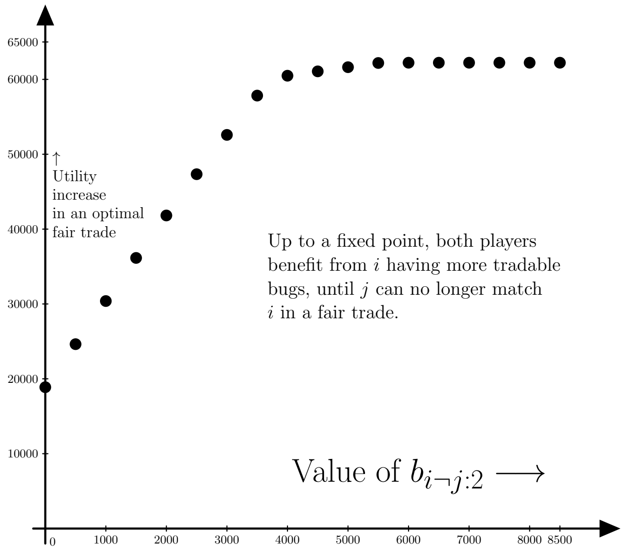

Figure 1: Utility improvement for both players when trading fairly as $b_{i\neg j:2}$ varies. Before $b_{i\neg j:2}$ reaches around 4000, player i trades all $\boldsymbol{B_{i\neg j}}$ in exchange for a subset of j's bugs. After this point is reach, j trades everything for a subset of i's information.

Proof. Let's start with some trade package $(\boldsymbol{s_i}, \boldsymbol{s_j})$ that is fair. Note that such a package must exist since $(\boldsymbol{0}, \boldsymbol{0})$ is trivially fair. Now, suppose there exist $x, y \in \{1, 2, \ldots, W\}$ such that $s_{ix} < b_{i\neg j:x}$ and $s_{iy} < b_{\neg ij:y}$.

We claim that we can increase s_{ix} or s_{jy} to $b_{i\neg j:x}$ or $b_{\neg ij:y}$ whilst maintaining fairness. Note that this scenario would be desirable to both players, because from Lemma 3.2.1, and the assumption (7), we have that $\mathcal{U}^+$ is strictly increasing in every s_{iw}, s_{jw}.

Let

$$\mu = \frac{\delta_{ix} + \delta_{jx} + \tau_{jx} + l_{jx} - \tau_{ix}}{\delta_{jy} + \delta_{iy} + \tau_{iy} + l_{iy} - \tau_{jy}}$$

and so by assumption (8), we have $\mu > 0$. Now define

$$\epsilon = min\{b_{i\neg j:x} - s_{ix},\ \frac{1}{\mu} \cdot (b_{\neg ij:y} - s_{jy})\}.$$

It follows that $(s_{ix} + \epsilon,\ s_{jy} + \lambda \cdot \epsilon)$ is a pair of tradable assets that is still fair, as can be checked against Proposition 3.1, and induces a greater utility than $(s_{ix},\ s_{jy})$. By construction of ϵ, at least one of these is maximal. We can iterate this process for any s_{ix}, s_{jy} in a fair trade such that both are non-maximal, until there are no such pairs left. □

3.3 Numerical evaluation

We computed numerical solutions to Proposition 3.1 using the `Python3` module `pulp` to check that the linear program can be solved quickly. We ran the experiments using both the continuum idealisation and the more realistic discrete spaces for $(\boldsymbol{s_i}, \boldsymbol{s_j})$. When the number of tradable bugs is fairly large, such as over 100, the proportional difference in optimal fair trade utility in the continuous and discrete cases becomes very small. We consider this a justification for the idealisation when analysing the game mathematically. The experiments are described fully in Appendix B.

	$i1$	$i2$	$i3$	$j1$	$j2$	$j3$
δ	3	3.5	5	3	4	5
τ	1	2	1.5	1	2	2.4
l	4	2	7	6	9	10
b	1000	2500	800	1111	2222	3333

Table 1: Default game settings to experimentally test Proposition 3.1 ($W = 3$). The table rows indicate the parameter type and the columns indicate the relevant subscript (b_{iw} should be read as $b_{i\neg j:w}$). See Appendix B for full results.

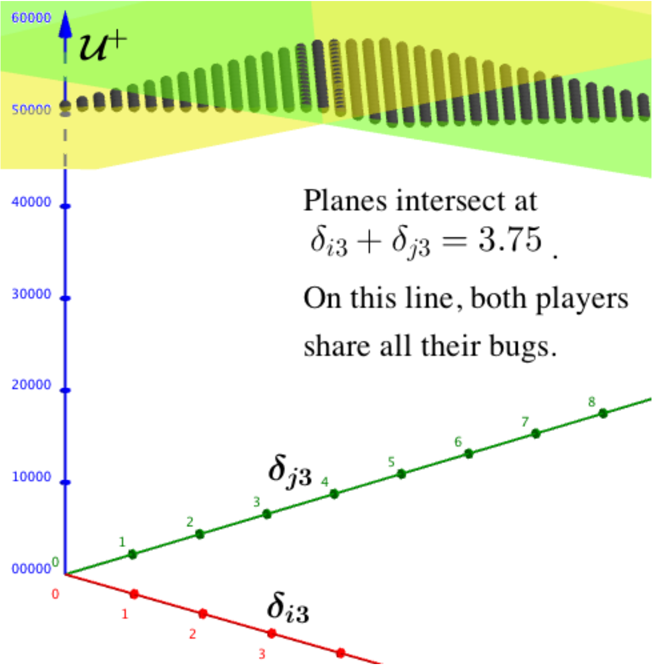

Figure 2: A 3D plot of $\mathcal{U}^+$ as a function of $(\delta_{i3}, \delta_{j3})$. All other parameters as in Table 1. The plot shows that the mutual value of a fair trade is maximised when players have a balanced view on the competitive value of each bit of information.

3.4 Bug discovery with anticipated fair trades

A general analysis of the security investment bug-discovery game is not covered this paper, as we intend to focus on the role of a trading mediator between firms. However, preliminary investigations show that both players do invest in bug-discovery research, avoiding the free-riding problem discovered in [18]. The intuition behind this result is that when a mechanism exists to restrict bug trading to what we have defined as 'fair' trades, both players try to discover bugs, so that they are in a stronger negotiating position to learn even more bugs from their competitor. On the other hand, players will not over-invest in bug-discovery, because their opponent should be expected to provide a significant proportion of information too. We hope to investigate this dynamic in more depth in future research.

In summary, the main results of our model is that when the two players can access a mediator that ensures bugs are traded "fairly", according to an agreed pricing scheme, they are incentivised to trade as many of their bugs as they can. Further, players will use this expectation to help them decide on their security investment levels (i.e. into bug-discovery) in order to strike a balance between not investing so much that the potential benefits of trading later are wasted, and not so little that they do not have enough to trade. In effect, the bugs act as valuable resources, and this game describes a special case of optimal resource acquisition and trading that is of interest to information security.

4. FORMALISATION OF THE MEDIATOR

We showed in Section 3 that it is in the best interest of both players to share information whenever a fair trade can be established. Using the same concept of a theoretical mediator that was first inferred in [18], it is possible to construct an environment where fair trades are then assumed to occur. Unfortunately, a third party who carries out this role may be hard to find and requires strong assumptions of trust from both parties. With this in mind, the second contribution of this paper is the proposal of a novel cryptographic protocol where the trusted party can be replaced with a mechanism with privacy guarantees, ensuring that fair trades are enforced. While syntaxes and services exist for sharing threat intelligence, none provide a bilateral trading mechanism that enables a party to receive new information for any information they share themselves. In this section we categorically define the functionality of the third party mediator so that we can later provide a cryptographic protocol that implements this functionality.

As noted earlier, we will adopt a slightly different notation in the next sections, to maintain consistency with other literature in private set operations. Let $\mathcal{M}$ be a mediator as referred to in Section 3.2, and let P_1 and P_2 be the participants from the game in Section 3 with corresponding sets S_1 and S_2 of vulnerabilities discovered in the early phase of the game. Here we will assume that bugs $b_1, \ldots, b_{|E|}$, where E is the universe of bugs, can be assigned any category from the set $[W]$ with the corresponding economic value of a bug b denoted $\mathbf{val}(b)$. Let $b_{1,j}, \ldots, b_{n,j}$ be the set of bugs belonging to the set S_j for $j = 1, 2$. The mediator takes T corresponding to a threshold total value of bugs to be traded and then computes an approximately optimal sharing of bugs to return to the two players. We define the formal functionality for the mediator in Figure 3.

```
Mediator $\mathcal{M}$
1 : Take as input the sets $S_1, S_2$ from $P_1, P_2$
2 : Compute $S_j^* = S_j \setminus (S_1 \cap S_2)$ for $j = 1, 2$
3 : Compute $\mathbf{val}(S_j^*) := \sum_{i=1}^{n} \mathbf{val}(b_{i,j})$ for $j = 1, 2$
4 : Compute $T = \min(\mathbf{val}(S_1^*), \mathbf{val}(S_2^*))$
5 : Take $S_1' \subseteq S_1^*, S_2' \subseteq S_2^*$ where:
      • $\mathbf{val}(S_1') = \mathbf{val}(S_2') = V' \leq T$
      • $V'$ is maximal.
6 : Send $S_1'$ to $P_2$ and send $S_2'$ to $P_1$
```

Figure 3: Generalised mediator for bilateral trades.

Note that we could define a mediator that satisfies fair trading of bugs of equal categories, and just define this mediator for each level. However, this would imply that bugs of different kinds could not be traded; our generalised mediator above allows for these richer trades.

Remark 4.1. *We compute T to be the maximum value possible as set out by the economic model. This can be generalised to allow both players to input a maximum amount to share from which the mediator takes the minimum amount.*

Remark 4.2. *As discussed in Section 3.3, some inexactitude in the threshold T is tolerable. As such, the reason that the mediator in step* 5 *of Figure 3 allows for swapping some value of bugs $V' \leq T$ is because the combination of bugs in both sets may not allow for the swapping of the full amount T. The mediator here computes the answer to a form of the 'knapsack problem'. These are generally NP-hard problems, but since the difference between discrete and continuous information sharing is negligible (Section 3.3), we can use polynomial-time algorithms such as those shown by [21] to approximate T.*

5. MEDIATOR PROTOCOL

By generalising the mediator we can achieve our goal of ensuring that sharing takes place in all cases. In this section we analyse the orthogonal question of "can we instantiate such a mediator in a real-world scenario?". As discussed previously, relying on a third-party trust assumption is far from ideal and so we develop a cryptographic protocol that can be carried out between the players that replicates the functionality from Section 4.

We devise a Private Set Operation (PSO) protocol that allows the particular set operation over input sets from the two players to be computed without revealing other information of relevance. We point the reader to the security model in Section 2.4.1 for a formal treatment of the security guarantees. Notice that the set operation we require is similar to a union operation in that the output reveals elements contained in the opposing set, though in our case a threshold limits how many of the elements are learnt. While more complex set operations have emerged regarding limiting thresholds (e.g. [19]), no current protocols provide the functionality we require for our bug-sharing scheme. In this section we develop a novel solution, based on a recent proposal of a construction for a family of PSO protocols with linear complexities [5]. Similar to most PSO proposals, we will consider security against semi-honest adversaries in our construction as opposed to the strongest 'malicious' adversary' model which is often considered in PSO proposals.

The main body of our design focuses on the use of encrypted Bloom filters (as defined in Section A.1). Our protocol trades a set of bugs corresponding to an agreed threshold T. The threshold T represents the maximum value of bugs that can be traded in any given interaction. We require that the protocol satisfies three security properties: (i) hides the intersection of both sets, (ii) does not reveal to the sending party (P_2) which bugs have been sent, only their security levels, and (iii) can only reveal bugs to the receiving party (P_1) that are shared from the sending party's set. We formally articulate these requirements next.

5.1 Protocol construction

Let P_1 and P_2 be participants in the protocol with input sets S_1 and S_2 respectively, where $S_1 = \{x_i\}_{i \in \{1,\ldots,n\}}$ and

$S_2 = \{y_j\}_{j \in \{1,\dots,m\}}$. For the purposes of this construction x_i, y_j are bugs that can be encoded uniquely in $\mathbb{Z}_N$ for some integer $N = pq$ for primes p, q. These bugs are marked with category w for each $w \in [W]$. During the protocol we will assume that P_1 will receive a subset of value T in terms of the bug values from S_2, P_2 will only receive output T as an output. For bilateral trading we simply require that the protocol is run twice in parallel with the players playing opposing roles. We denote the set operation that we realise in the later sections by $\cup|_T$ and the corresponding protocol by $\pi_{\cup|_T}$ to reflect the fact that we are effectively restricting a union computation with the threshold T. During the computation P_1 is the receiver and P_2 is the sender.

Let $\mathbf{BF}_1$ denote the Bloom filter of length B, $\mathbf{IBF}_1$ the inverted Bloom filter, $\mathbf{EBF}_1$ the encrypted Bloom filter, and $\mathbf{EIBF}_1$ the encrypted, inverted Bloom filter, all representing the set S_1. Let $\Gamma = (\mathcal{K}, E, D)$ be a public-key, additively homomorphic, probabilistic encryption scheme satisfying IND-CPA security. Let $\mathsf{pk}, \mathsf{sk} \leftarrow\!\$\, \mathcal{K}$, then $E_{\mathsf{pk}}(x)$ is the encryption algorithm that takes a public key pk and a message x and generates a ciphertext c, and $D_{\mathsf{sk}}(c)$ is the decryption algorithm taking the secret key sk and a ciphertext c and returning a message x. We denote the additive operation that we obtain over the ciphertexts by $+_H$ where $D_{\mathsf{sk}}(c +_H c') = m + m'$, with $c = E_{\mathsf{pk}}(m)$ and $c' = E_{\mathsf{pk}}(m')$. An encryption scheme satisfying all these properties is the Paillier scheme shown in [26]. We assume that both players have access to pk and that only P_1 has access to sk. We will also have an alternate public key-pair $\mathsf{pk}', \mathsf{sk}' \leftarrow\!\$\, \mathcal{K}$ where pk' is known to both P_1, P_2 while sk' is known only to P_2.

We assume that the maximum value of bugs to be swapped, $T_{\max} = \min(\mathbf{val}(S_1), \mathbf{val}(S_2))$, is agreed by the players in advance via prior interaction; it is possible to do this using garbled circuits to calculate the minimum of two values (the work of Brickell et al. [1] demonstrates this). Finally, our participants are aware of the cardinalities of the two input sets. We can write the inputs of P_1 as $(\mathsf{Inp}_1, \mathsf{aux}_1) = (S_1, (|S_2|, \mathsf{pk}, \mathsf{sk}, \mathsf{pk}', T_{\max}))$ with output $S_1 \cup \mathcal{O}$, and the inputs of P_2 to be $(\mathsf{Inp}_2, \mathsf{aux}_2) = (S_2, (|S_1|, \mathsf{pk}, \mathsf{pk}', \mathsf{sk}', T_{\max}))$ with output T where $T = T_{\max} - \mathbf{val}(S_1 \cap S_2)$ (see step 8). The steps of the protocol follow below.

5.1.1 Protocol steps

1. P_1 creates $\mathbf{BF}_1$ to represent their set S_1, and computes $\mathbf{EIBF}_1$ by inverting and encrypting each element before sending to P_2.
2. P_2 computes
$$\mathbf{EIBF}_1[h_l(y_j)] = C_l^{(j)}$$
for each $y_j \in S_2$ and for $l \in \{1, \dots, k\}$. Let $c_j = C_1^{(j)} +_H \dots +_H C_k^{(j)}$.
3. P_2 samples random values $r_{j,1}, r_{j,2} \leftarrow\!\$\, \mathbb{Z}_N$ for each y_j and computes $\tilde{r}_{j,1} \leftarrow E_{\mathsf{pk}}(r_{j,1})$.
4. P_2 now computes $((c_j \cdot y_j) +_H \tilde{r}_{j,1})$ and the messages
$$msg_j = ((c_j \cdot y_j) +_H \tilde{r}_{j,1},\ c_j \cdot r_{j,2},\ E_{\mathsf{pk}'}(w_j))$$
for each y_j where $E_{\mathsf{pk}'}(w_j)$ is a probabilistic encryption of the severity level of y_j.
5. P_2 sends $(msg_j, \mathsf{ind}(msg_j))$ in a randomly permuted order to P_1 where $\mathsf{ind}(msg_j)$ denotes the place of msg_j in the order of messages sent.
6. P_1 parses each message into the form shown above and computes $q_j := D_{\mathsf{sk}}(msg_j[1])$.
   - If $y_j \in S_1 \cap S_2$, then $q_j = 0$; and therefore y_j cannot be learnt from $msg_j[0]$.
   - Else, q_j is some random value.
7. Let $I = \{j : q_j = 0\}$, P_1 computes $\tilde{w}_j := \underset{j \in I}{+_H} msg_j[2]$ sends $\tilde{w}_j' := \tilde{w}_j +_H E_{\mathsf{pk}'}(0)$ to P_2. The homomorphic addition of zero re-randomises the ciphertext.
8. P_2 computes $D_{\mathsf{sk}'}(\tilde{w}_j') = \sum_{j \in I}(w_j)$ and performs $T = T - \sum_{j \in I} w_j = T - T_{int}$. Set $T' = T$. The value T is kept static while T' is used later.
9. P_1 picks a random msg_u where $u \notin I$ and re-randomises $msg_u[2]$ as above to compute $\hat{w}_u'$. Send $\hat{w}_u'$ to P_2.
10. P_2 decrypts $E_{\mathsf{pk}'}(w_u)'$ using sk' to learn w_u for the underlying element y_u being queried. P_2 then computes $T' = T' - w_u$.
11. If $T' < 0$, P_2 sends a 'reject' message to P_1. P_1 returns to step 9 to pick another message. If no other choice is possible then P_1 goes to step 16.
12. If $T' > 0$, then both players now participate in a $\binom{m}{1}$ OT. P_1 is the receiver in the OT and P_2 is the sender with $(r_{j,1}, r_{j,2})$ for each $y_j \in S_2$.
    - P_1 submits $\mathsf{ind}(msg_u)$ to the OT
    - P_1 receives the corresponding $(r_{u,1}, r_{u,2})$ for msg_u
13. P_1 computes $q_u \cdot r_{u,2}^{-1}$ to obtain $z_u \in \mathbb{N}$ (note that $z_u := D_{\mathsf{sk}}(c_u)$).
14. P_1 then computes $\tilde{p}_u := msg_u[0] +_H (-\tilde{r}_{u,1})$; and then $D_{\mathsf{sk}}(\tilde{p}_u) = z_u \cdot y_u$.
15. Finally, P_1 computes $y_u := z_u \cdot y_u \cdot z_u^{-1}$ and adds y_u to the output set $\mathcal{O}$ and then returns to step 9.
16. P_1 outputs $S_1 \cup \mathcal{O}$ and P_2 outputs T from step 8.

5.1.2 Protocol correctness

Elements $y_j \in S_1 \cap S_2$ lead to $D_{\mathsf{sk}}(c_j) = 0$ due to the characteristics of the inverted Bloom filter. Therefore, it is impossible to add y_j to $\mathcal{O}$ since $(c_j \cdot y_j) +_H \tilde{r}_{j,1}$ will just decrypt to the random value $r_{j,1}$. Further, elements $y_j \notin S_1$ are only learnt until the threshold value T' is reduced to a point where the remaining bugs will cause too much to be shared, contravening the fairness of the trade. Specifically, if $y_j \notin S_1$, then via the OT P_1 can learn the randomness values $r_{j,1}, r_{j,2}$ decrypt and multiply by the inverses to learn y_j. Player P_2 monitors T', so that elements are only learnt by P_1 while T' remains positive. Therefore, only elements y_j that P_1 do not know such that the severity level w_j does not reduce T' below 0, are learnt. We finally require both players to receive new elements. This is achieved straightforwardly with two rounds of the protocol in reversed roles.

5.1.3 Protocol security

We can formalise the security requirement by constructing an ideal functionality for the operation we want to perform. For input sets S_1, S_2 the ideal functionality can be denoted by $\mathcal{F}_{\cup|_T} = (\{S_2\}_T, T)$ and is defined as:

$$\mathcal{F}_{\cup|_T} = (\{S_2\}_T, T) \qquad (9)$$

where T is the recalculated value of elements to be shared and the notation $\{X\}_T$ indicates a random selection of elements that have value equal to T from the set X. The ideal functionality is defined commutatively so that both players receive both outputs over two separate executions. We now state the following theorem:

THEOREM 1. *Suppose that the protocol $\pi_{\cup|_T}$ is instantiated with an OT that is semi-honest secure, and an IND-CPA secure, additively homomorphic encryption scheme. The protocol $\pi_{\cup|_T}$ securely realises the ideal functionality $\mathcal{F}_{\cup|_T}$ shown in equation* (9) *w.r.t semi-honest adversaries.*

PROOF. For brevity we only provide an intuitive proof. Firstly, the IND-CPA encryption of the Bloom filter hides the elements in $\mathbf{EIBF}_1$ from P_2, and the IND-CPA encryption of the severities w_j hides these values from P_1. Furthermore, the unrevealed values are hidden by virtue of them being masked by random values. The provision of the output value T (after subtraction of the intersection value) to P_2 allows the enforcing of the threshold in the simulation and similarly for the output that P_1 receives. The OT can be simulated in a sub-routine since no other messages are sent in this period. Finally, messages that are sent during the protocol are randomised, preventing either player from recalculating the message values to infer which set elements are being sent. Otherwise the ciphertexts c_j are created deterministically and so they could also be calculated by P_1 to uncover elements in the intersection of the two sets. $\square$

5.1.4 Protocol efficiency

Assuming that $|S_1| = |S_2| = n$, both the computation and communication (in bits) is linear in n, with only five communication rounds needed (assuming that the OTs run in parallel as a sub-routine in one of the rounds; we use the OT as shown in [3]). These complexities imply practical scalability for our design as set sizes increases; we reinforce this with running times and communication loads in Section 6.

5.1.5 Comparison with ideal mediator functionality

Comparing the protocol with the mediator of Section 4 we see that some extra information is leaked by our protocol. For example, the players explicitly learn the cardinality of the intersection of the two sets (this was deducible from the mediator's actions but not explicitly given away). Also, P_2 learns the levels of the bugs that P_1 is learning but not the explicit bugs; this is so that P_2 can enforce the value of the threshold T'. The minor information leakage will have no significant effect on the rational trading behaviour of the players, as the benefit of eschewing the need for a trusted third party outweighs the negligible leakage that may occur.

It is possible to reduce the leakage profile of the protocol by adapting the design so that both players swap a random number of all bugs (regardless of the value of the bugs swapped). This would mean that the kind of bugs sent would not have to be revealed to P_2 throughout the protocol. Additionally would allow a hard threshold to be enforced, and the participants would no longer be required to communicate the category levels of bugs and thus this reduces the number of communication rounds to three (assuming the OT is the third round). However such an adaption does not guarantee fair trade, as players swap random bugs and thus could receive many high severity level bugs and vice-versa. This random sharing adaptation would represent a trade-off between higher security in the random case, and the fair trade property in the original case.

6. PROTOCOL IMPLEMENTATION

In Appendix C we detail an implementation of the mediator protocol from Section 5.1 written in the language `Go`. We find that it can be run successfully and efficiently using commodity computing hardware (e.g. 8GB RAM, 3.4GHz processor) for data sets of up to at least 2^{12} items. The results are encouraging and emphasise the applicability of our protocol to real-world information sharing scenarios, especially given the likelihood that more powerful dedicated hardware could lead to further performance gains.

7. CONCLUSION

In this work we have developed a game-theoretic model for undertaking cyber security information sharing and provided a method by which fair, bilateral trades can be performed based on a mediating party. Moreover, we propose a novel cryptographic protocol, which implements the mediator role, removing the need for any trusted third-party presence. The trading model and cryptographic protocol can be integrated alongside tools such as STIX and TAXII, giving potential users a choice between unilateral sharing, and secure bilateral threat-intelligence sharing.

By developing information sharing as an area that sits importantly in the intersection between economics and cyber security we also establish many future routes for research. The game-theoretic model can be modified in myriad ways to model richer economic situations; e.g. including risk-averse behaviour or oligopolies of more than 2 firms. Future research could model the sharing of other security information besides bugs; for example network data known to an oligopoly of ISPs or security management policies of interacting firms and governments. Further, the security investment stage of the new gameplay may be further investigated. Preliminary results of our generalised game on security investment suggest that anticipated fair trade mechanisms incentivise both players to invest in bug-discovery research, thus avoiding the free-riding effects founds in [18]. The cryptographic component may be improved by fulfilling stronger security requirements, e.g. preventing some of the leakage that we describe from the protocol. This could include expanding the random sharing model that we highlighted in Section 5.1.5. Finally, the protocol could be modified to cater for computations allowing multiple competing organisations to share their information in a secure environment and allowing greater industry collaboration.

References

[1] Justin Brickell and Vitaly Shmatikov. "Privacy- Preserving Graph Algorithms in the Semi-honest Model". In: *ASIACRYPT*. Vol. 3788. Lecture Notes in Computer Science. Springer, 2005, pp. 236–252.

[2] Huseyin Cavusoglu, Birendra Mishra, and Srinivasan Raghunathan. "The effect of internet security breach announcements on market value". In: *Intl. J. of Electronic Commerce* 9.1 (2004), pp. 70–104.

[3] Tung Chou and Claudio Orlandi. *The Simplest Protocol for Oblivious Transfer*. Cryptology ePrint Archive, Report 2015/267. http://eprint.iacr.org/. 2015.

[4] Emiliano De Cristofaro and Gene Tsudik. “Practical Private Set Intersection Protocols with Linear Complexity”. In: *Financial Cryptography and Data Secu-rity, 14th Int’ Conference, Tenerife, Canary Islands, Jan 25-28, 2010.* 2010, pp. 143–159.

[5] Alex Davidson and Carlos Cid. *Computing Private Set Operations with Linear Complexities.* Cryptology ePrint Archive, Report 2016/108. 2016.

[6] Sumit Kumar Debnath and Ratna Dutta. “Secure and Efficient Private Set Intersection Cardinality Using Bloom Filter”. In: *Information Security - 18th International Conference, ISC 2015, Trondheim, Norway, September 9-11, 2015, Proceedings.* 2015, pp. 209–226.

[7] Changyu Dong, Liqun Chen, and Zikai Wen. “When private set intersection meets big data: an efficient and scalable protocol”. In: *2013 ACM SIGSAC, CCS’13, Berlin, Germany, Nov 4-8, 2013.* 2013, pp. 789–800.

[8] Rolf Egert, Marc Fischlin, David Gens, Sven Jacob, Matthias Senker, and J¨orn Tillmanns. “Privately Com-puting Set-Union and Set-Intersection Cardinality via Bloom Filters”. In: *Information Security and Privacy - 20th Australasian Conference, ACISP 2015, Brisbane, QLD, Australia, June 29 - July 1, 2015, Proceedings.* 2015, pp. 413–430.

[9] Keith B. Frikken. “Privacy-Preserving Set Union”. In: *Applied Cryptography and Network Security, 5th International Conference, ACNS 2007, Zhuhai, China, June 5-8, 2007, Proceedings.* 2007, pp. 237–252.

[10] Sanjay Goel and Hany A Shawky. “Estimating the market impact of security breach announcements on firm values”. In: *Information & Management* 46.7 (2009), pp. 404–410.

[11] Lawrence Gordon, Martin Loeb, and William Lucyshyn. “Sharing information on computer systems security: An economic analysis”. In: *Journal of Accounting and Public Policy* 22.6 (2003), pp. 461–485.

[12] Jens Grossklags, Nicolas Christin, and John Chuang. “Secure or insure?: a game-theoretic analysis of information security games”. In: *Proceedings of the 17th international conference on World Wide Web.* ACM. 2008, pp. 209–218.

[13] Kjell Hausken. “Information sharing among firms and cyber attacks”. In: *Journal of Accounting and Public Policy* 26.6 (2007), pp. 639–688.

[14] Farhad Hormozdiari, Jong Wha J. Joo, Akshay Wadia, Feng Guan, Rafail Ostrovsky, Amit Sahai, and Eleazar Eskin. “Privacy preserving protocol for detecting genetic relatives using rare variants”. In: *Bioin-formatics* 30.12 (2014), pp. 204–211. DOI: 10.1093/bioinformatics/btu294.

[15] Matthew O Jackson and Yves Zenou. “Games on net-works”. In: *Handbook of Game Theory* 4 (2014).

[16] Benjamin Johnson, Jens Grossklags, Nicolas Christin, and John Chuang. “Uncertainty in interdependent se-curity games”. In: *Decision and Game Theory for Se-curity.* Springer, 2010, pp. 234–244.

[17] Florian Kerschbaum. “Outsourced private set intersection using homomorphic encryption”. In: *7th ACM Symposium on Information, Compuer and Communications Security, ASIACCS ’12, Seoul, Korea, May 2- 4, 2012.* 2012, pp. 85–86.

[18] M. H. R. Khouzani, Viet Pham, and Carlos Cid. “Strategic Discovery and Sharing of Vulnerabilities in Com-petitive Environments”. In: *GameSec 2014, Los Angeles, CA. Nov 6-7, 2014. Proceedings.* 2014, pp. 59–78.

[19] Lea Kissner and Dawn Xiaodong Song. “Privacy Preserving Set Operations”. In: *CRYPTO 2005: 25th Annual Int’ Cryptology Conference, Santa Barbara, CA. Aug 14-18, 2005, Proceedings.* 2005, pp. 241–257.

[20] Howard Kunreuther and Geoffrey Heal. “Interdependent security”. In: *Journal of risk and uncertainty* 26.2-3 (2003), pp. 231–249.

[21] K. Lai and M. Goemans. “The knapsack problem and fully polynomial time approximation schemes”. 2006. URL: http : / / math . mit . edu / ˜goemans / 18434S06 /knapsack-katherine.pdf.

[22] Aron Laszka, Mark Felegyhazi, and Levente Buttyan. “Survey of interdependent information security games”. In: *ACM Computing Surveys* 47.2 (2015), p. 23.

[23] Stefan Laube and Rainer Böhme. “Mandatory security information sharing with authorities: Implications on investments in internal controls”. In: *Proceedings of the 2nd ACM Workshop on Information Sharing and Col-laborative Security.* ACM. 2015, pp. 31–42.

[24] Dengpan Liu, Yonghua Ji, and Vijay Mookerjee. “Knowledge sharing and investment decisions in information security”. In: *Decision Support Systems* 52.1 (2011), pp. 95–107.

[25] Arvind Narayanan, Narendran Thiagarajan, Mugdha Lakhani, Michael Hamburg, and Dan Boneh. “Location Privacy via Private Proximity Testing”. In: *NDSS proceedings, San Diego, California, 6th Feb - 9th Feb, 2011.* The Internet Society, 2011.

[26] Pascal Paillier. “Public-Key Cryptosystems Based on Composite Degree Residuosity Classes”. In: *EUROCRYPT ’99, Int’ Conference on the Theory and Application of Cryptographic Techniques, Prague, May 2-6, 1999. Proceeding.* 1999, pp. 223–238.

[27] Charles E Phillips Jr, TC Ting, and Steven A Demurjian. “Information sharing and security in dynamic coalitions”. In: *Proceedings of the seventh ACM sympo-sium on Access control models and technologies.* ACM. 2002, pp. 87–96.

[28] Benny Pinkas, Thomas Schneider, and Michael Zohner. “Faster Private Set Intersection Based on OT Extension”. In: *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014.* 2014, pp. 797–812.

[29] Carl Shapiro and Hal R Varian. *Information rules: a strategic guide to the network economy.* Harvard Business Press, 1999.

[30] Marco Slikker, Henk Norde, and Stef Tijs. “Information sharing games”. In: *International Game Theory Review* 5.01 (2003), pp. 1–12.

[31] Qiang Xiong and Xiaoyan Chen. “Incentive Mechanism Design Based on Repeated Game Theory in Security Information Sharing”. In: *2nd International Conference on Science and Social Research (ICSSR 2013)*. Atlantis Press. 2013.

APPENDIX

A. BLOOM FILTERS AND OBLIVIOUS TRANSFER

A.1 Bloom filters

Bloom filters are lightweight data structures that allows for the representation of data sets and checking of inclusion using only hash function evaluations. A Bloom filter is initially represented by a string of B bits that are all initialised to 0. There are k public hash functions

$$h_l : \{0,1\}^* \mapsto [B]$$

for each $l \in [k]$. We then represent set elements $x \in X$ in the Bloom filter by evaluating $h_1(x), \ldots, h_k(x)$ and changing each index that these hash functions point to from 0 to 1. If a value has already been changed to 1 then it is left alone. The resulting Bloom filter can then be checked against to see if different elements lie in the set by evaluating the k hash functions and checking if all the positions are set to 1.

One constraint on Bloom filters is that they can lead to false positives when checking membership, i.e. an element $y \notin X$ may appear to be in X after checking all the hash outputs if all the values had already been set to 1. However, as shown in [7], if $p = 1-(1-1/B)^{kn}$ is the probability that a particular bit in the Bloom filter is set to 1, and $n = |X|$ is the size of the input set, then the upper bound of the false-positive probability is given by

$$\epsilon = p^k \times \left(1 + \mathcal{O}\left(\frac{k}{p}\sqrt{\frac{\ln(B) - k \cdot \ln(p)}{B}}\right)\right),$$

which is negligible in k.

In this work we will assume B and k are chosen optimally for efficiency reasons, namely:

$$B = n \log_2 e \cdot \log_2\left(\frac{1}{\epsilon}\right), \quad k = \log_2\left(\frac{1}{\epsilon}\right), \tag{10}$$

where e is the base of the natural logarithm.

A.1.1 Encryption and inversion

In our protocol design we work with Bloom filters that are encrypted and inverted. We define here what we mean by these notions and we note that previous works have used similar conventions (see Section 2).

Definition A.1. *Let $\Gamma = (\mathcal{K}, E, D)$ be a public-key cryptosystem and $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{K}$ be generated at random from the key space $\mathcal{K}$. Let the Bloom filter calculated by P_i for the set S_i be denoted as $\boldsymbol{BF}_i$ and have B entries such that $\forall\ b \in [B]$ we have $\boldsymbol{BF}_i[b] \in \{0,1\}$. The corresponding encrypted Bloom filter is denoted $\boldsymbol{EBF}_i$ and defined to be:*

$$\boldsymbol{EBF}_i[b] = E_{pk}(\boldsymbol{BF}_i[b]).$$

In the following we define $\mathbf{EBF}_i = \{C[1], \ldots, C[B]\}$ and for $y_j \in S_i$, then $\mathbf{EBF}_i[h_u(y_j)] = C_u^{(j)}$ for $u \in [k]$.

Definition A.2. *Let $\boldsymbol{BF}_i$ be a Bloom filter that represents the set of party P_i. We define the corresponding inverted Bloom filter to be $\boldsymbol{IBF}_i$ where*

$$\boldsymbol{IBF}_i[b] = \begin{cases} 1 & \textit{if } \boldsymbol{BF}_i[b] = 0 \\ 0 & \textit{otherwise.} \end{cases}$$

To make the encryption of the Bloom filter well-defined we use 0 and 1 entries (where 1 is the identity element) from the ring $\mathbb{Z}_N$ for some $N \in \mathbb{N}$.[7]

A.2 Oblivious transfer

An Oblivious Transfer (OT) protocol allows a receiving party to secretly learn something from a sending party without learning any other information that the sender holds, and without the sender learning what the receiver has learnt. The basic functionality is usually described as the receiver R having a choice value $b \in [n]$ and the sender S having n different elements $s_1, \ldots, s_n$. The OT results in R receiving the element s_b without learning any of the elements s_i, where $i \neq b$ and without S learning the element that R learns. We usually refer to this OT as 1-out-of-N OT or, more concisely, $\binom{n}{1}$OT. In this work we will use the OT construction shown by Chou and Orlandi [3].

B. NUMERICAL EVALUATION OF THE MAXIMISATION PROBLEM FOR OPTIMAL FAIR TRADING OF BUGS

For our experimental analysis, we fixed $W = 3$ and chose a particular set of 24 default parameters to look at, as listed in Table 1. We chose the defaults fairly arbitrarily, but we ensured that the conditions for Proposition 3.1 hold.

In Figure 1, we modified the parameter $b_{i\neg j:2}$, leaving all others as defaults, and computed the incremental utility (the trade-value) for both players in the optimal fair trade agreement. As can be seen, up to around $b_{i\neg j:2} = 4000$, every increase in $b_{i\neg j:2}$ induces a proportional increase in the trade-value. After around 4000, player j can no longer match i's total set of tradable bugs, and so the optimal fair trade flattens. We ran this experiment for 30 values of $b_{i\neg j:2}$, but only the first 18 are plotted on the graph as it merely levels-off thereafter. These 30 experiments took less than 13 seconds on a 2.5 GHz Intel Core i5 processor.

In Figure 2, we ran 400 experiments with the competitive-edge factors of the third bug kind by varying $(\delta_{i3}, \delta_{j3})$ over $\{0, 0.25, 0.5\ldots, 4.75\}^2$. The 3D plot shows that a greater competitive parameter for i, δ_{i3}, will improve overall utility resulting from fair trades, as long as $\delta_{i3} + \delta_{j3} < 3.75$. This initial improvement in social welfare can be understood as the competitive edge that i can achieve from bug knowledge getting closer to j's. Notice from Lemma 3.2.1 that the social welfare incremental utility, $\mathcal{U}^+$ increases at its fastest rate when the deltas are close. On the other hand, when $\delta_{i3} + \delta_{j3} > 3.75$ each increase in the δ_{i3} parameter will reduce overall social welfare, as the competitive value of bugs becomes more important to i than to j.

In other experiments, we instead modified all 24 parameters, leaving the others as defaults. Our method was to replace each parameter with 33%, 66%, 100%, 133%, and 166% of its default value, and computed the resulting trade-value

[7] This allows us to use an additively homomorphic encryption scheme such like that of Paillier [26].

in an optimal fair exchange in both the continuous and discrete cases (the induced $24 \times 5 \times 2 = 240$ experiments took 155 seconds). The results were identical for both the discrete and continuous case up to three significant figures, for every experiment. Further, in every game, one of the players traded all, or almost all in case of rounding errors, of their bugs in exchange for a subset of their opponent's, which is consistent with the predictions from Corollary 3.1.

C. PSO PROTOCOL IMPLEMENTATION

Here we detail a proof-of-concept implementation of the protocol from Section 5 written in the programming language `Go`. We used `Go` due to the ease that concurrency can be built into the programming and we use this to make use of parallel execution for our PSO design. The implementation was run on a laptop with 8GB of RAM and a 4th Generation Intel Core i7-4700HQ Processor (3.4GHz). To instantiate the protocol we use an implementation of Paillier encryption and we implement the OT proposed by Chou and Orlandi [3] in `Go`. For the experiments we range over set sizes from $n = 2^6$ to $n = 2^{12}$, we choose $k = 60$ and thus the false-positive probability rate is 2^{-60}. The length of the modulus N is 1024 bits, and the threshold T is set equal to n where the artificial limit just guarantees that not all elements are known. Finally the domain of set elements is set to $5 * n$ (to give a reasonable size intersection between sets) and we choose the optimal Bloom filter size for the above choices. The implementation itself makes use of parallelisation in what P_2 computes and during the OT, with rudimentary tests suggesting that we achieve an approximate ×3 speed-up on a pipelined execution and so we do not provide figures for this case. In Table 4 we display the encryption time for a Bloom filter corresponding to each set size, and in Tables 2 and 3 we show the communication and computation costs for each of the cases.

We note that due to the use of public-key operations, the initial encryption of the Bloom filter takes a long time (see Table 4). However, we can amortise this computation over numerous executions of the PSO protocol and so we do not include these times in the running times of the PSO protocol in Table 3. Firstly, after encrypting the Bloom filter we still retain the functionality provided by a normal Bloom filter, e.g. we can still add elements to the set. The AHE scheme allows the holder to change entries in the Bloom filter even while it is encrypted (e.g. by adding ciphertexts, encrypting one to entries that need to be changed after adding an element), thus the Bloom filter can be used for multiple executions with different parties. For running with the same party it is enough to just re-randomise each ciphertext before sending it again.[8] Secondly, sets may not change during multiple executions, especially if carrying out the information sharing protocol concurrently with different entities, and so the same Bloom filter can be used in each one.

The running times and communication costs that we show are encouraging for the model we propose and show that the protocol is viable for a situation where entities want to engage in privacy-preserving cyber security information sharing. It is quite clear that the computation and communication scale linearly as set sizes grow as the asymptotics suggest.

Set sizes	2^6	2^8	2^{10}	2^{12}
Comms (Mb)	1.41967	5.6765	22.7038	90.81278

Table 2: Total communication overhead (Mb).

Set sizes	2^6	2^8	2^{10}	2^{12}
Time (sec)	15.93	69.47	266.668	1024.94

Table 3: Total PSO running times (sec).

Set sizes	2^6	2^8	2^{10}	2^{12}
Time (mins)	3.03	12.85	52.27	~ 180

Table 4: Total encryption times (mins).

If comparing our timings with the work of [28], which categorises current PSI protocols, we do notice that our protocol has much more expensive overheads (e.g. running times are over an order of magnitude greater than the most efficient dedicated PSI designs). On the other hand, the communication costs appear to be in line with some of the showcased practical protocols, notably outperforming generic circuit designs for PSI. It is however important to note that the protocol we have proposed and implemented provides a much more complex structure than a PSI protocol and so comparing these designs may seem inappropriate. Additionally, trading of software vulnerabilities is unlikely to reach set sizes above 2^{12} and so for our use case the running times and communication costs are reasonable, especially given the novel functionality we propose.

[8] Although multiple trades with the same entity are not covered by the economic model, where it is assumed that post-sharing all unknown bugs are exploited in the platform they own.