A Taxonomy of Rational Attacks

Seth James Nielson
sethn@cs.rice.edu

Scott A. Crosby
scrosby@cs.rice.edu

Dan S. Wallach
dwallach@cs.rice.edu

Department of Computer Science, Rice University

***Abstract*— For peer-to-peer services to be effective, participating nodes must cooperate, but in most scenarios a node represents a self-interested party and cooperation can neither be expected nor enforced. A reasonable assumption is that a large fraction of p2p nodes are *rational* and will attempt to maximize their consumption of system resources while minimizing the use of their own. If such behavior violates system policy then it constitutes an attack. In this paper we identify and create a taxonomy for *rational attacks* and then identify corresponding solutions if they exist. The most effective solutions directly incentivize cooperative behavior, but when this is not feasible the common alternative is to incentivize evidence of cooperation instead.**

1 Introduction

A significant challenge in peer-to-peer (p2p) computing is the problem of cooperation. Unlike client-server systems, a p2p network's effectiveness in meeting design goals is directly correlated to the cooperation of the member nodes. For example, a p2p system might be designed for content distribution. To decrease the upload bandwidth burden on the original content server, only a small number of nodes directly contact it. The content is then propagated from these nodes to additional peers. This system can only scale if nodes are willing to pass on content to downstream peers. Unfortunately, a self-interested node may realize that it can save expensive upload bandwidth if it chooses not to share. If a large number of nodes are self-interested and refuse to contribute, the system may destabilize.

In most p2p systems, self-interested behavior at the expense of the system can be classified as a *rational manipulation* failure [18] or, from a different perspective, a *rational attack*[1]. Successful p2p systems must be designed to be robust against this class of failure. Ideally, a p2p system should be perfectly *faithful* to the designer's specification. In such a system, a self-interested, utility-maximizing node "will follow the default strategy because... there is no other strategy that yields a higher utility for this node" [19]. To achieve faithfulness, a system may employ various measures such as *problem partitioning*, *catch-and-punish*, and *incentives* [18]. Even when these techniques cannot make a system perfectly faithful, they may be enough to prevent destabilization.

An example of a viable p2p technology designed to be robust against rational manipulation failures is BitTorrent [4]. This technology first breaks large files into chunks that are downloaded individually and reassembled by the receiver. The receiving nodes contact one another and trade for chunks they do not yet possess. Each node employs an incremental exchange algorithm that leads it to upload chunks to cooperating nodes and not to share with selfish ones. These incentives encourage cooperative behavior in participating nodes [4]. While BitTorrent is not completely immune to rational manipulation, it is viable in practice [19].

In this paper, we identify, analyze, and create a taxonomy of rational attacks in p2p systems. We then examine this taxonomy to identify corresponding solutions. In the next two sections, we first provide a short background on the economics principles applicable to p2p systems and then specify our system model. The following two sections define our taxonomy of rational attacks and discuss solutions. The final section presents our conclusions.

2 Economics Background

Much of our analysis of p2p cooperation is based on economic models of game theory and mechanism design [17]. In this section, we briefly review some critical terms and concepts as they relate to p2p systems.

An economic *game* is a model of interaction between *players* in which the actions of any player influence the outcome of all other players. The *mechanism* in a game defines what legitimate actions the players can perform and the outcome of their behavior. These outcomes are assigned a numeric value called *utility*. Players that use an algorithm to determine behavior are said to follow a *strategy*

Players in the p2p world represent the nodes participating in the system. There are two types of nodes that do *not* strategize.

- *Altruistic* or *obedient* nodes cooperate with the system irrespective of any other considerations.

[1]Our definition for rational follows the narrow definition provided by Shneidman et al [18]. For the purposes of our paper, rational participants are only interested in exploiting the resources and benefits of the system.

- *Faulty* nodes stop responding, drop messages, or act arbitrarily.

There are two types of nodes that do strategize.

- *Rational* nodes strategize to achieve maximal utility and their actions are based on their current knowledge and understanding of the p2p system. Rational nodes will not attempt to disrupt routing, censor data, or otherwise corrupt the system unless such behavior increases the node's access to shared resources. These nodes are also described as *self-interested*.
- *Irrational* nodes also strategize, but their strategies are either incomplete because they cannot understand the mechanism or they lie outside the economic mechanisms of the system. Denial of service or censorship attacks are examples of this second form of economically irrational behavior[2].

Mechanism design (MD) is the process of creating games where rational behavior by players leads to outcomes desired by the designer. Of course, such systems only affect the behavior of rational nodes. Mechanism design has no impact on faulty or irrational nodes and we exclude them from further discussion, though we recognize that any practical p2p system deployed "in the wild" must be resistant to their behavior. Of course, most p2p systems are robust against failure. The impact of irrational and malicious nodes is an open research problem that is discussed in Castro et al [3].

Distributed algorithmic mechanism design (DAMD) is a subclass of MD that is computationally tractable and operates without centralization. For this reason DAMD is well suited to systems like p2p networks [17]. DAMD assumes each node can independently reward the cooperation of other nodes or penalize their misbehavior but that each node has only limited information on the global state of the system.

3 Model

3.1 Incentives Capabilities

Incentives in p2p systems have some limitations. First, incentives are limited in the guarantees they can provide. While the use of incentives strengthens the p2p system against rational attacks, by themselves they do not guarantee that the system is faithful. To be guaranteed faithful, a mechanism must be validated by a formal proof, the construction of which is not trivial.

The second limitation is that they must be DAMD compatible. DAMD is limited to creating mechanisms that are are computationally tractable across distributed computing resources. Nodes are expected to reward cooperation and penalize misbehavior, but doing so is difficult when trusted global knowledge is unavailable.

With these two limitations in mind, we identify two types of incentives that may be used to create a faithful p2p system. The first type is *genuine incentives* and is characterized by directly incentivizing cooperation. A genuine incentive ties current behavior and future payoff together in some inseparable way. Genuine incentives are inherently robust against rational attacks and limit the strategies available to adversaries.

One example of genuine incentives is incremental exchanges as used in BitTorrent. Money could also be an effective genuine incentive but it would require very efficient micropayment schemes, where potentially every network packet transmission would require an associated payment. Unfortunately, the current generation of such systems (e.g., Millicent [9]) were never intended for such fine-grained commerce.

The second type of incentive is *artificial incentives*[3] which incentivize evidence of cooperation. Such incentives are weaker than their genuine counterparts because, to be rewarded, a node only has to *appear* to cooperate. Nevertheless, artificial incentives are generally easier to create and deploy and may be necessary under circumstances where genuine incentives are not feasible.

Artificial incentives are often designed around an *auditing* process on top of which an enforcement mechanism is layered. In a decentralized system, auditing cannot be globally managed. Each node is aware of the system's policies, but is independently responsible for determining whether peers are in compliance. This can be done by requiring each node to publish assertions about its state which are audited by other nodes. An auditing policy of this type is consistent with DAMD; each node is capable of determining its behavior within the system. An auditing system, however, is subject to the vulnerabilities that we describe in Section 4.1.2.

3.2 Service Maturation

A p2p service provides some tangible benefit to participating nodes. New participants may obtain their payout spread over time, or they can obtain maximal

[2]Our goal is to design systems which are immune to manipulation by nodes seeking increased shared resources. Our definition of rational only includes nodes whose utility function is independent of utility payout to other nodes. Strategies, such as censorship strategies, that obtain benefit by denying utility to other nodes are considered irrational.

[3]Roussopoulos et al. suggests that highly valuable shared resources have inherent incentives while less valuable ones require an extrinsic or artificial incentives for cooperation [16]. Our concept of genuine and artificial incentives is similar, but focuses only on the mechanism and not the value of the resources or social network in which the resources are exchanged.

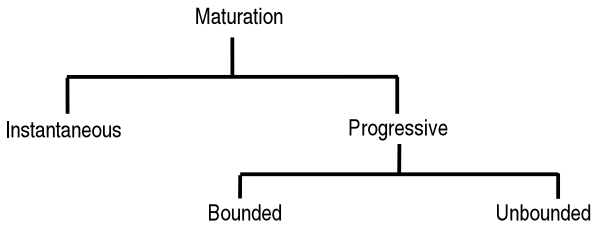

Fig. 1. Service Maturation Taxonomy

benefit immediately in a lump sum. We have termed this service characteristic as *service maturation*. A service is mature when a node has obtained all of the benefit that the service can provide. Services that give out all possible benefit immediately have *instantaneous maturation* while services that distribute benefit over time have *progressive maturation*. Progressive maturation can be further classified as *bounded* or *unbounded* based on whether or not the service has a known, fixed termination of benefit pay-out. The relationship between the different classes of maturation is illustrated in Figure 1.

A content distribution service might have instantaneous or progressive maturation depending on policy. If a newly joined node can completely download its desired content before redistributing that content to peers, the service has instantaneous maturation. Conversely, BitTorrent has progressive maturation because it only allows nodes to obtain the full content through repeated interaction with the system. Because BitTorrent's payout of benefit ends when the file download is complete, its progressive maturation is bounded.

An example of a service with unbounded progressive maturation is a remote back-up service. In such a system, the benefit payout is distributed over time without a fixed point of termination.

There is a correlation between instantaneous maturation to the Prisoner's Dilemma (PD) and progressive maturation to the Iterated Prisoner's Dilemma (IPD). In the single round PD, all of the utility that the game can pay out is disbursed in a single interaction. In IPD, the total utility is paid out to participants over some arbitrary number of interactions.

IPD also has an analog to the concept of bounded maturation. The game can be played with the players either aware or ignorant of the number of rounds that they will play. From the players' perspective, the game is bounded only if they know the number of rounds. An IPD game degenerates into a PD game if the number of rounds are known.

Game theoretic analysis has proven that it is not rational to cooperate in single round PD but that it is rational to cooperate in IPD [2]. Services with instantaneous maturation are extremely susceptable to the attacks described in Section 4.2.

3.3 System Model

For convenience, we define a constrained environment suitable to explore rational attacks. The p2p model characterized in this section has many features that are common to most p2p networks. In Section 5 we break some of these assumptions as possible solutions to rational attacks.

Our model is described by the following assumptions and limitations.

Assumption: *Secure node ID's*. Douceur [6] observes that if identity within the p2p system is not centrally controlled, any participant can simultaneously assume a plethora of electronic personae. With many identities at its disposal, a participant can subvert the entire network by subverting the routing primitive. We assume that the node ID's in our model are made secure in one of three ways:

- **Trust** - Node ID creation and distribution is done through a centralized and mutually trusted agent.
- **Expense** - Node ID creation has some arbitrary cost attached. A participant can replace its node ID infrequently and with some difficulty.
- **Relevance** - Node ID creation is unrestricted because having multiple ID's cannot aid the rational attacker.

Assumption: *There is no "trusted" software*. A p2p system cannot guarantee that their members are using conforming software. Trusted computing technologies allow a node to attest that it is running a conforming application [15], [20]. Enforcing a trusted software policy is not only technically challenging, but developing and deploying such a policy is undesirable to many groups for ethical or practical reasons [21].

Assumption: *Nodes are computationally limited*. We assume that any given node may have the same resources as the typical desktop PC. Nodes may subvert their machine to behave in arbitrary ways. However nodes are assumed to be incapable of breaking cryptographic primitives or taking global control of the underlying network.

Due to the potential size of p2p systems and because nodes are in mutually untrusting domains, we apply the following limitations to our model.

Limitation: *Each node maintains minimal state*. A node can only have firsthand observations about a small fraction of the nodes in the system. Similarly

a node can only maintain state about a small number of the nodes in the system.

Limitation: *No second-hand information*. Nodes can only trust what they directly observe because there is no inherent reason to trust an assertion by any node about a third party. An accusation can only be trusted if the evidence is independently believable regardless of trust in the accuser. Such proofs usually require the cooperation of the accused to create.

4 Taxonomy of Rational Attacks

The motive for the attacks we consider are unfairly increased access to p2p shared resources. We identify two general classes of attack:

1) Unrecorded Misuse of Resources
2) Unpunished Misuse of Resources

Attacks can be made by a single node, or by several nodes colluding together for an advantage.

4.1 Unrecorded Misuse of Resources

If an attacker can obtain resources without producing a record of the misuse, the attacker is safe from any sanctions. Attacks of this kind exploit "holes" in auditing policies (*policy attacks*), or actively disrupt the auditing mechanism (*auditing attack*).

4.1.1 Policy Attacks: A rational node may exploit an auditing policy. We identify two examples.

Excuses Any legitimate "excuse" for being unable to perform a service may be exploited. Such excuses may be needed to deal with edge conditions including crash recovery, network interruption, packet loss, etc. Consider a remote backup system like Samsara that requires every node to contribute as much space as it consumes [5]. If the system policy is overly generous to recovering nodes that recently crashed by not requiring them to prove they are maintaining their quota, a malicious node may exploit this by repeatedly claiming to have crashed.

Picking on the newbie Some systems require that new nodes "pay their dues" by requiring them to give resources to the system for some period of time before they can consume any shared resources [22], [7]. If this policy is not carefully designed, a veteran node could move from one newbie node to another, leeching resources without being required to give any resources back.

4.1.2 Auditing Attacks: Auditing attacks are designed to prevent the auditing system from identifying misbehavior. These attacks only apply to designs based around auditing using artificial incentives. Here are a number of examples of this type of attack:

Fudged books Auditing relies on the accounting records being tamper-resistant and difficult to forge.

Manufactured evidence In this scenario, an attacker who is in a state of non-compliance manages to produce "proof" of compliance deceptively.

Accounting interruption (kill the auditor) A node being audited can attempt to interfere with the auditing node. This might be accomplished by a denial-of-service attack, a worm, a virus, etc.

Group deception, local honesty This attack is a type of manufactured evidence attack through collusion. Ngan, et al describes an accounting system where nodes publishing their debits and credits publicly in logs which are later audited by nodes' peers [8]. Debts on one node must match credits on another node, making it more difficult for a node to cook its books. However, it is possible for single node in debt to become locally honest for an audit by pushing its debt to a co-conspirator. As a group, the conspiring nodes' books are not balanced and they are in debt jointly. All colluding nodes reciprocate in sharing (or hiding) the debt.

4.2 Unpunished Misuse of Resources

An identified misbehaving node may attempt to avoid or mitigate punishment. Two such attacks are:

Elusion The attacker leaves the system permanently before they can be sanctioned by the p2p system. This attack generally exploits short-maturation and high-value resources. In such a scenario, the attacker obtains the resources and leaves (e.g., join a content distribution service long enough to obtain an object and then disappear forever).

Reincarnation Reincarnation is repeated elusion. The attacker avoids punishment for misbehavior by assuming a new node ID thus releasing them from any penalties associated with its old reputation. We note that this attack is a limited form of the Sybil attack [6] where multiple ID's are acquired and discarded over time rather than all at once.

This class of attacks operates almost entirely against p2p services with instantaneous maturation.

5 Solutions

As stated previously, an ideal p2p system is perfectly faithful, but creating such a mechanism and proving its validity is difficult. In some cases a perfectly faithful design may be impossible, but a p2p system need not be perfectly faithful to be viable. In this section, we describe defenses against rational attacks by self-interested nodes in descending order of theoretical effectiveness.

5.1 Eliminate rationality as a concern

Under certain circumstances, forcing all nodes to be obedient may be practical and desirable. We identify three options for coercing obedience.

Out-of-band trust Obedience is enforced external to the p2p system. Such a scenario might be viable for a group of friends, or centrally administered machines within corporations, academic institutions, and government agencies.

Partial centralization It may be possible to introduce some aspect of centralization that induces nodes to be obedient. For instance a central authority can be used to require secure node ID creation. BitTorrent uses a central authority to act as a rendezvous point where nodes can determine the addresses of their peers.

Trusted software - If a user is prevented from modifying their software, they must behave obediently. Many software applications are closed-source and difficult to modify. This may also be done with "trusted computing" technologies [1], [20].

5.2 Design genuine incentives

Genuine incentives are always preferred to artificial incentives. Because they are often difficult to implement in a DAMD context, it may be tempting for a designer to overlook them. Not only do genuine incentives eliminate many of the attacks described in Section 4.1.2, but they are also simpler than artificial incentives because they require no additional enforcement mechanisms.

For example, consider a back-up system with a storage policy similar to Samsara where each node must provide as much disk-space as it consumes in backups. One artificial incentives approach proposed by Fuqua, et al is to require that all nodes publish what data they are storing locally and to prove that they actually have that data in their possession on audit [8]. The auditing mechanism may be vulnerable to one or more of the auditing attacks described in Section 4.1.

A genuine incentive for the remote back-up service is to require that all of a node's data that is stored on the network be tangled with the data it is supposed to be storing [22]. Nodes can then occasionally broadcast portions of the tangled data they are storing and ask for its owner to claim it or risk its deletion. Now the self-interested node must actually keep the data it claims to be storing or it cannot recognize claim-requests for its own data. However, to be useful, there must be a policy that allows a node to reclaim its data after a crash even if it has lost all local-storage. This policy may expose the mechanism to the excuses attack described in Section 4.1.1. Despite this weakness, however, this mechanism is more robust and significantly simpler than the auditing alternative.

5.3 Improving artificial incentives design

Artificial incentives are a less desirable solution to rational attacks, but they may be the easiest to design into a service and are sometimes the only viable solution. Artificial incentives will generally entail having a well-defined auditing policy. A number of design decisions influence the effetiveness of these incentives.

5.3.1 Eliminating instantaneous maturation: A service which instantaneously matures is difficult to secure against rational attacks. Once a rational node has obtained the maximum benefit for a service, it has no incentive to continue participation. Thus, services that instantly mature are inherently vulnerable to elusion and reincarnation attacks. Also, because a node obtains its desired utility quickly, there is not much time for an auditing scheme to stop an attacker. Several techniques may help convert instantaneous to progressive maturation:

Centralized ID Creation If node ID's are centrally created and distributed, a node will be forced to maintain its identity in all of its future interactions with the p2p system. In this case if a node steals from the system and leaves, it will face punishment when it returns.

Security Deposit A node must contribute resources during a probationary period before it can benefit from the system's shared resources. Tangler is an example of system using this technique [22], [7].

5.3.2 Limited number of peers: Changing a node's ID incurs a cost. If an auditing system can detect and kick out a misbehaving node sufficiently fast, then the cost of changing identity outweighs the benefit. In most p2p systems, a node can only access the network through a limited number of neighbors. Once an attacker has freeloaded on its neighbors, they will refuse to interact with it and it will be effectively removed from the system. This solution has been used for multicast and storage accounting [14], [12], [13].

5.3.3 Reputation: With perfect global knowledge of every peer's behavior, a node would be incentivized to cooperate because any time it cheated, that information would be immediately available to all of its peers. Unfortunately, perfect global knowledge is only possible through an oracle which is not available in a DAMD context such as p2p networks.

Distributed systems may try to recreate the notion of a global, trusted oracle using gossip protocols, rating schemes, or some other from of peer endorsements. Mojo Nation had a global reputation system and EigenTrust describes how such systems might be built [10].

5.3.4 Protecting an auditing infrastructure: Because artificial incentives require building and protecting an auditing infrastructure, these mechanisms have additional complexity that may be prone to design and implementation errors. We suggests three practices for building effective auditing mechanisms:

Force the truth to be told Nodes can usually only believe what they observe for themselves. Secure history techniques [11], however, may be useful to generate authenticated records of misbehavior that are trustable by remote hosts.

Double-entry bookkeeping A double-entry bookkeeping system as described earlier in Section 4.1.2.

Create a global clock When multiple nodes are being audited, they may be able to pass debts around from one node to the next, such that any particular node, while it is being audited, appears to have its books balanced. If several nodes can be simultaneously audited at provably the same time, this may defeat such attacks. Again, secure history techniques may provide an approximate solution to this problem.

6 Conclusions

In this paper we explored a number of rational attacks. While we used a narrow definition of "rational", we feel that this usage is justified by the unique nature of such attacks. From our analysis, we believe that designs that incorporate genuine incentives will generally be simpler and more robust that those with artificial incentives. Artificial incentives often require an auditing mechanism that is complicated and difficult to construct.

Unfortunately, given the difficulty of designing and implementing genuine incentives in a DAMD context such as p2p networks, artificial incentives will often be essential to incentivize cooperation for some parts of the system. When this is the case, avoiding instantaneous maturation eliminates unpunished misuse of resources attacks. A carefully designed policy and a robust auditing scheme are essential to mitigating unrecorded misuse of resources.

References

[1] W. A. Arbaugh, D. J. Farber, and J. M. Smith. A secure and reliable bootstrap architecture. In *Proc. of the 1997 IEEE Symposium on Security and Privacy*, page 65, San Diego, CA, USA, May 1997. IEEE Computer Society.

[2] R. Axelrod and W. D. Hamilton. The evolution of cooperation. *Science*, 211:1390–1396, 1981.

[3] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach. Secure routing for structured peer-to-peer overlay networks. In *Proceedings of Operating System Design and Implementation*, Boston, MA, Dec. 2002.

[4] B. Cohen. Incentives build robustness in BitTorrent. In *1st Internation Workshop on Economics of P2P Systems*, June 2003.

[5] L. P. Cox and B. D. Noble. Samsara: Honor among thieves in peer-to-peer storage. In *SOSP '03: Proc. of the Nineteenth ACM Symposium on Operating Systems Principles*, pages 120–132. ACM Press, Oct. 2003.

[6] J. R. Douceur. The Sybil attack. In *Proceedings for the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02)*, Cambridge, Massachusetts, Mar. 2002.

[7] E. J. Friedman and P. Resnick. The social cost of cheap pseudonyms. *Journal of Economics & Management Strategy*, 10(2):173–199, June 2001.

[8] A. C. Fuqua, T.-W. J. Ngan, and D. S. Wallach. Economic behavior of peer-to-peer storage networks. In *Workshop on Economics of Peer-to-Peer Systems*, Berkeley, CA, June 2003.

[9] S. Glassman, M. Manasse, M. Abadi, P. Gauthier, and P. Sobalvarro. The millicent protocol for inexpensive electronic commerce. *World Wide Web Journal, Fourth International World Wide Web Conference Proceedings*, 1(1):603–618, 1996.

[10] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *Proc. of the Twelfth International Conference on World Wide Web*, pages 640–651, May 2003.

[11] P. Maniatis and M. Baker. Secure history preservation through timeline entanglement. In *Proc. of the 11th USENIX Security Symposium*, pages 297–312. USENIX Association, 2002.

[12] T.-W. J. Ngan, A. Nandi, A. Singh, D. S. Wallach, and P. Druschel. On designing incentives-compatible peer-to-peer systems. In *2nd Bertinoro Workshop on Future Directions in Distributed Computing (FuDiCo II: S.O.S.)*, Bertinoro, Italy, June 2004.

[13] T.-W. J. Ngan, D. S. Wallach, and P. Druschel. Enforcing fair sharing of peer-to-peer resources. In *2nd International Workshop on Peer-to-Peer Systems (IPTPS), LNCS 2735*, pages 149–159, Berkeley, CA, Feb. 2003.

[14] T.-W. J. Ngan, D. S. Wallach, and P. Druschel. Incentives-compatible peer-to-peer multicast. In *2nd Workshop on the Economics of Peer-to-Peer Systems*, Cambridge, MA, June 2004.

[15] Microsoft "Palladium": A business overview, 2002. `http://www.microsoft.com/presspass/features/2002/jul02/0724palladiumwp%.asp`.

[16] M. Roussopoulos, M. Baker, and D. S. H. Rosenthal. 2 p2p or not 2 p2p? In *IPTPS '04*, Feb. 2004.

[17] J. Shneidman and D. Parkes. Rationality and self-interest in peer to peer networks. In *IPTPS '03*, Berkeley, CA, USA, Feb. 2003.

[18] J. Shneidman and D. C. Parkes. Specification faithfulness in networks with rational nodes. In *Proc. 23rd ACM Symp. on Principles of Distributed Computing (PODC'04)*, St. John's, Canada, July 2004.

[19] J. Shneidman, D. C. Parkes, and L. Massoulie. Faithfulness in internet algorithms. In *Proc. SIGCOMM Workshop on Practice and Theory of Incentives and Game Theory in Networked Systems (PINS'04)*, Portland, OR, USA, Sept. 2004.

[20] TCPA. Building a foundation of trust in the PC. Technical report, Trusted Computing Platform Alliance, 2000.

[21] 'Trusted Computing' frequently asked questions, 2003. `http://www.cl.cam.ac.uk/~rja14/tcpa-faq.html`.

[22] M. Waldman and D. Mazieres. Tangler: a censorship-resistant publishing system based on document entanglements. In *Proc. of the 8th ACM Conference on Computer and Communications Security*, pages 126–135. ACM Press, Nov. 2001.