Defending Against the Unknown Enemy: Applying FLIPIT to System Security

Kevin D. Bowers[1], Marten van Dijk[1], Robert Griffin[2], Ari Juels[1], Alina Oprea[1], Ronald L. Rivest[3], and Nikos Triandopoulos[1]

[1] RSA Laboratories, Cambridge, MA, USA
[2] RSA, The Security Division of EMC, Zurich, Switzerland
[3] MIT, Cambridge, MA, USA

Abstract. Most cryptographic systems carry the basic assumption that entities are able to preserve the secrecy of their keys. With attacks today showing ever increasing sophistication, however, this tenet is eroding. "Advanced Persistent Threats" (APTs), for instance, leverage zero-day exploits and extensive system knowledge to achieve *full* compromise of cryptographic keys and other secrets. Such compromise is often silent, with defenders failing to detect the loss of private keys critical to protection of their systems. The growing virulence of today's threats clearly calls for new models of defenders' goals and abilities.
In this paper, we explore applications of FLIPIT, a novel game-theoretic model of system defense introduced in [17]. In FLIPIT, an *attacker* periodically gains *complete* control of a system, with the unique feature that system compromises are *stealthy*, i.e., not immediately detected by the system owner, called the *defender*. We distill out several lessons from our study of FLIPIT and demonstrate their application to several real-world problems, including password reset policies, key rotation, VM refresh and cloud auditing.

1 Introduction

Targeted attacks against computing systems have recently become significantly more sophisticated. One major consequence is erosion of the main principle on which most cryptographic systems rely for security: That "secret" keys remain strictly secret. Attacks known as Advanced Persistent Threats (APTs), for instance, exploit deep, system-specific knowledge and zero-day vulnerabilities to compromise a system completely, revealing sensitive information that can include *full* cryptographic keys. Moreover, this compromise is *stealthy*, meaning that it's not immediately detected by the system owner or defender. We have previously introduced a game-theoretic model for this volatile new security world called FLIPIT [17], The Game of "Stealthy Takeover."

FLIPIT is a game between two players, known as the *attacker* and *defender*. Players compete to control a shared sensitive resource (e.g., a secret key, a password, or an entire infrastructure, depending on the setting being modeled). A player may take control of the resource at any time by executing a *move*; the player pays a certain (fixed) cost to do so. The fact that moves are *stealthy* in FLIPIT distinguishes it from other games in the literature. A player in FLIPIT doesn't immediately know when her opponent has made a move, but discovers it only when she subsequently moves herself. Each player's objective is to maximize the fraction of time she controls the resource, while minimizing her cumulative move cost.

The goal of this paper is twofold: (1) To present some general principles of effective play in FLIPIT and (2) To demonstrate application of these principles to defensive strategy design in real-world cyberdefense settings. Thus our contributions are:

Principles of effective FLIPIT play: We introduce general principles for effective FLIPIT play by a defender facing a more powerful attacker. These principles fit into three categories: (A) Principles governing defender strategy selection based on knowledge about the class of strategies employed by the attacker; (B) Principles regarding game setup, specifically, effective cost-structure design choices for the defender; and (C) Principles regarding gameplay feedback, namely how the defender can best maximize feedback via system design for effective gameplay. All principles have solid theoretical underpinnings in our analysis in [17].

Application of FLIPIT to real-world security problems: We explore the application of FLIPIT to the problem of managing *credentials* used for user authentication. In particular, we're interested in enabling system owners (defenders) to schedule the expiration or refresh of their credentials most effectively. We focus primarily on *passwords* (namely, password-reset policies) and *cryptographic keys* (key refresh, also known as *key rotation*). Specifically, we show the benefits of *randomizing* password reset intervals (in sharp distinction to the widespread 90-day password reset policy). We also quantify the importance of frequent rotation of keys protecting critical defender assets.

We briefly touch on other applications of the FLIPIT framework, including virtual machine refresh and cloud auditing for service-level-agreement (SLA) enforcement. Our FLIPIT design principles bring to light defensive strategies that improve on current practices in these settings. Study of these applications also introduces new and interesting variants of the basic FLIPIT game whose analysis provides interesting open questions for the community.

Organization. In Section 2, we present the FLIPIT framework and detail on the lessons learned from [17] in the form of set of principles. In Section 3, we apply these principles to password reset and key management, and in Section 4 to virtualization and cloud auditing. We review related work in Section 6 and conclude in Section 7.

2 Framework and Principles

We start this section by introducing the FLIPIT framework. We then introduce several principles for designing defensive strategies in various computer security scenarios derived from our theoretical analysis presented in [17]. In the following sections, we present several applications of the framework and show how the principles introduced here result in effective defensive strategies.

2.1 FLIPIT Framework

We present FLIPIT by the example of "host takeover" where the target resource is a computing device. The goal of the attacker is to compromise the device by exploiting a software vulnerability or credential compromise. The goal of the defender is to keep the device clean through software reinstallation, patching, or other defensive steps.

An action/move by either side carries a cost. For the attacker, the cost of host compromise may be that of, e.g., mounting a social-engineering attack that causes a user to

open an infected attachment. For the defender, cleaning a host may carry labor and lost-productivity costs. The resource can be controlled (or "owned") by either of two *players* (attacker / defender). When a player moves he takes control of the resource; ownership will change back and forth as the players make moves. A distinctive feature of FLIPIT is its *stealthy* aspect, that is the players *don't know* when the other player has taken over. Nor do they know the current ownership of the resource unless they perform a move. For instance, the defender does not find out about the machine compromise immediately, but potentially only after he moves himself; or, the attacker might find out about the host cleanup at a later time, not immediately when the defender moves.

The goal of each player is to maximize the time that he or she controls the resource, while minimizing their move costs; players thus have a disincentive against moving too frequently. A move results in a "takeover" when ownership of the resource changes hands. If the player who moves already had ownership of the resource, then the move was wasted (since it did not result in a takeover). The only way a player can determine the state of the game is to move. Thus a move by either player has two consequences: it acquires control of the resource (if not already controlled by the mover), but at the same time, it reveals information about the state of the resource prior to the player taking control. This knowledge may be used to determine information about the opponent's moves and assist in scheduling future moves.

FLIPIT provides guidance to both players on how to implement a cost-effective move schedule. For instance, it helps the defender answer the question: "How regularly should I clean my system?" and the attacker: "When should I launch my next attack?".

We show a graphical representation of the game in Figure 1. The control of the resource is graphically depicted through shaded rectangles, a blue rectangle (dark gray in grayscale) representing a period of defender control, a red rectangle (light gray in grayscale) one of attacker control. Players' moves are graphically depicted with shaded circles. A player obtains no new information once he makes his n-th move, until he makes his $(n+1)$st move. A vertical arrow denotes a takeover, when a player (re)takes control of the resource upon moving. In this example, a move costs the equivalent of one second of ownership. Thus, at any time t, each player's net score is the number of seconds he has had ownership of the resource, minus his number of moves up to time t.

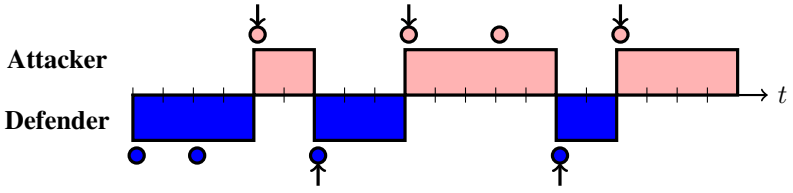

Fig. 1. The FLIPIT game. Blue and red circles represent defender and attacker moves, respectively. Takeovers are represented by arrows. Shaded rectangles show the control of the resource—blue (dark gray in grayscale) for the defender and red (light gray in grayscale) for the attacker. We assume that upon initialization at time 0, the defender has control.

So far, the example introduced the concepts of players (defender and attacker), game state (who is in control at time t), moves, and benefit up to some time t (defined as the amount of time in control minus the moving costs). Since a player's strategy in the game

naturally depends on the player's knowledge achieved either before the game starts or during the game, we need to describe the knowledge that each player has as a function of time before we are able to define player strategies.

The most interesting aspect of this game is that the players do *not* automatically find out when the other player has moved in the past; moves are *stealthy*. A player must move himself to find out (and reassert control). We distinguish various types of feedback that a player may obtain upon moving:

- **Nonadaptive [NA].** No feedback is given to the player upon moving.
- **Last move [LM].** The player moving at time $t > 0$ finds out the exact time when the opponent played last before time t.
- **Full history [FH].** The mover finds out the complete history of moves made by both players so far.

Since feedback is the means by which a player acquires more knowledge in FLIPIT, we are now ready to define the view or knowledge of a player. The *view* of a player after playing his nth move is the history of the game from this player's viewpoint from the beginning of the game up to and including his nth move. It lists every time that player moved, and the feedback received for that move, up to and including his nth move.

We can now define a player's strategy in the game. Informally, a strategy for a player defines how moves in the game are chosen as a function of time, the knowledge about the opponent acquired before the game starts and the amount of feedback received by a player during the game. More formally, a *strategy* for playing FLIPIT is a (possibly randomized) mapping S from views to positive real numbers. If S is a strategy and v a view up to and including the player's nth move, then $S(v)$ denotes the time the player waits before making his $(n+1)$st move.

Strategies can be grouped into several classes. For instance, the class of *non-adaptive strategies* includes all strategies for which players do not receive any feedback during the game. The class of *renewal strategies* is a subset of non-adaptive strategies in which the intervals between a player's consecutive moves are generated by a renewal process. As such, the inter-arrival times between moves in a renewal strategy are independent and identical distributed random variables chosen from the same probability density function. The class of *adaptive strategies* encompasses strategies in which players receive feedback during the game according to either LM or FH notions defined above.

Pre-game information about the opponent's class of strategies from which the opponent selects his strategy can be used in conjunction with the feedback received while moving to determine the strategy of playing the game. More formally, a strategy of a player defines how he chooses the time to play his next move based on the view seen so far. A *strategy* for playing FLIPIT is a (possibly randomized) mapping S from views to positive real numbers. If S is a strategy and v a view up and including the player's jth move, then $S(v)$ denotes the time the player waits before making his $(j+1)$st move.

A game instance in FLIPIT is given by two classes of strategies, one for the attacker and one for the defender, from which the players can select their strategies before the game starts. The strategy can be randomized and adapted according to the feedback received during the game. We denote by $\mathsf{FlipIt}(\mathcal{C}_0, \mathcal{C}_1)$ the FLIPIT game in which player i chooses a strategy from class $\mathcal{C}_i$, for $i \in \{0, 1\}$. Here, we identify the defender with 0 and the attacker with 1.

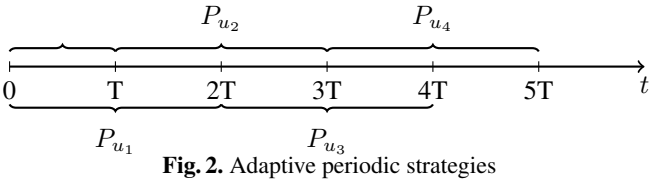

Fig. 2. Adaptive periodic strategies

For a particular choice of strategies $S_0 \in \mathcal{C}_0$ and $S_1 \in \mathcal{C}_1$, the benefit $\beta_i(S_0, S_1)$ of player i is defined in the following way:

- We define k_0 and k_1 as the cost of the defender's and attacker's moves, respectively.
- By $\alpha_i(t)$ we denote the *average move rate* by player i up to time t. In other words, $\alpha_i(t)$ is equal to the total number of moves by player i up to time t divided by t.
- By $\gamma_i(t)$ we denote the *average gain rate* for player i defined as the fraction of time that player i has been in control of the game up to time t.
- Now we are ready to define player i's *average benefit rate* up to time t as $\beta_i(t) = \gamma_i(t) - k_i\alpha_i(t)$. This is equal to the fraction of time the resource has been owned by player i, minus the cost rate for moving.
- The *benefit* β_i of player i is defined as the liminf of player i's benefit rate up to time t as t tends to infinity; $\liminf_{t\to\infty} \beta_i(t)$.

The average move, gain and benefit rates all depend on the exact strategies S_0 and S_1 played by defender and attacker. (Here, benefits represent the notion of *utility*.)

In [17], we have presented a detailed definition of the FLIPIT game and a rigorous analysis of several aspects of the game, including Nash equilibria for certain FlipIt instances and an analysis of dominated strategies within certain classes of strategies.

In the following, we show the connection of FlipIt with existing games studied in the game theory literature. In particular, we demonstrate that FlipIt with adaptive strategies can be reduced to repeated Prisoner's Dilemma.

Connection to Prisoner's Dilemma. FlipIt with adaptive strategies can be complicated and we demonstrate here an example in which players face a choice similar to that in the well-known repeated prisoner's dilemma game [9]. In our example we assume a defender and an attacker who both play an "adaptive periodic strategy," that we explain in Figure 2. We divide time in intervals $[iT, (i+1)T]$ for $i \geq 0$. During each interval the attacker and defender play a periodic strategy with random phase: i.e., the first move in an interval defines the phase and subsequent moves are periodic. By P_u we denote the periodic strategy that starts with a first move after ϕ seconds for some random $\phi \in [0, 1/u]$, all subsequent moves are separated by exactly $1/u$ seconds; $1/u$ is called the period and u is the rate at which the periodic strategy plays moves.

Both players have a choice of playing either $P_{1/8}$ or $P_{1/4}$ in each interval. We assume that the cost of a move for both players is equal to 1 second value of being in control of the resource; $k_0 = k_1 = 1$. The defender starts playing P_{u_1} in interval $[0, 2T]$ for some $u_1 \in \{1/8, 1/4\}$. During the first half of $[0, 2T]$ the attacker adaptively learns rate u_1 at which the defender plays P_{u_1} in $[0, 2T]$. Based on u_1, the attacker chooses a rate $u_2 \in \{1/8, 1/4\}$ at which to play P_{u_2} in $[T, 3T]$. In interval $[T, 2T]$ the defender learns rate u_2 and based on u_1 and u_2 the defender adaptively chooses a rate u_3 at which to play P_{u_3} in $[2T, 4T]$, etc. This results in the attacker playing $P_{u_{2i}}$ in $[(2i-1)T, (2i+1)T]$ and the defender playing $P_{u_{2i-1}}$ in $[(2i-2)T, 2iT]$ for $i \geq 1$. The choice of u_i depends on all previously chosen u_j, $1 \leq j < i$.

Suppose the defender starts playing periodically at rate $u_1 = 1/8$. The attacker may decide to cooperate and play at the same rate $1/8$ or defect by playing at rate $1/4$. As shown below, his choice is the prisoner's dilemma, which proves that FLIPIT with adaptive strategies is as complex as repeated prisoner's dilemma.

	$1/8$	$1/4$
$1/8$	$(\beta_0, \beta_1) = (3/8, 3/8)$	$(\beta_0, \beta_1) = (1/8, 4/8)$
$1/4$	$(\beta_0, \beta_1) = (4/8, 1/8)$	$(\beta_0, \beta_1) = (2/8, 2/8)$

2.2 Principles for Designing Defensive Strategies

FLIPIT was motivated by the observation that systems should nowadays be designed to be resilient to very powerful adversaries that can eventually fully compromise the system. Defenders protecting sensitive resources (including sensitive personal information, cryptographic keys, national secrets) face increasingly sophisticated attackers and traditional defensive techniques are no longer effective. The framework provided by FLIPIT provides a model of continuous interaction between a defender and attacker in controlling a resource, which can be used to study this new reality. Based on our theoretical analysis in [17] we outline in this section several principles for designing effective defensive strategies when dealing with various security situations.

There are three main categories of principles, detailed in the rest of the section:

(A) Principles about selecting a defensive strategy given some knowledge about the class of strategies employed by the attacker;
(B) Principles about the setup of the FLIPIT game resulting in various system design choices for the defender;
(C) Principles about the amount of feedback received by the defender and made available to the attacker during the game.

(A) Principles related to strategy selection. The first type of principles are as follows.

Residual Game [RG]. There is an assumption in game theory that a rational player does not choose to play a strategy that is strongly dominated by other strategies. Therefore, iterative elimination of strongly dominated strategies for both players is a standard technique used to reduce the space of strategies available to each player (see, for instance, the book by Myerson [9]). For a game instance $\mathsf{FlipIt}(\mathcal{C}_0, \mathcal{C}_1)$, we denote by $\mathsf{FlipIt}^*(\mathcal{C}_0, \mathcal{C}_1)$ the *residual* FLIPIT game consisting of surviving strategies after elimination of strongly dominated strategies from classes $\mathcal{C}_0$ and $\mathcal{C}_1$. A rational player will always choose a strategy from the residual game resulting in the following principle:

RG Principle: Given a particular game instance, a defensive strategy should be selected from the residual game.

For instance, one set of results in [17] analyzes the game instance in which both players can select strategies from the class of renewal strategies $\mathcal{R}$ (i.e., to set the time between moves according to a fixed probability distribution) or that of periodic strategies $\mathcal{P}$ with random phase (i.e., to set the first move uniformly at random, while all next moves are chosen according to a fixed period). For this game instance, the periodic strategy with random phase strongly dominates the renewal strategies of similar play rate, i.e., the residual game $\mathsf{FlipIt}^*(\mathcal{R} \cup \mathcal{P}, \mathcal{R} \cup \mathcal{P})$ turns out to be equal to $\mathsf{FlipIt}(\mathcal{P}, \mathcal{P})$.

A second example is a scenario in which an LM adaptive defender plays against an attacker employing an exponential strategy (i.e., the intervals between moves are selected according to an exponential distribution). Then the defender's strongly dominant strategy among all adaptive strategies is periodic play.

Randomized Strategy [RS]. In a FLIPIT game in which an NA defender plays against an adaptive LM attacker (i.e., the attacker acquires additional knowledge through last move feedback), the defender should either introduce randomness when selecting her moves or not play at all (assuming that the attacker plays with some positive rate).

A deterministic, predictable strategy for the defender (e.g., periodic play) results in total loss of control: an adaptive attacker finding out the exact last move time of the defender can predict the time of the defender's next move and move right after the defender. With this strategy, the attacker controls the resource virtually at all times. Therefore, adding randomization to the intervals between defender's moves has the advantage of increasing the attacker's uncertainty about the defender's strategy. This results in the following principle:

RS Principle: The defender should use randomization in her strategy (or not play at all) when confronted with an adaptive attacker moving with positive rate.

While introducing randomization when selecting defensive moves against an adaptive attacker has a clear benefit in increasing the defender's benefit, the amount of variability in the defender's strategy has to be carefully calibrated to not deviate too much from the optimal strategy. We'd like to highlight here that finding the strongly dominant non-adaptive (randomized) defensive strategy against an adaptive attacker is an open problem (see [17]).

Drop Out Principle [DOP]. For some applications the resource is so valuable that the loss of control (even for small fraction of time) results in highly negative benefit for the defender. For such scenarios, the strongly dominant strategy for the defender is to play fast enough in order to force a rational attacker to drop out of the game.

In [17] we showed two results:

- If the defender plays periodic with rate $\alpha > 1/k_1$, then the attacker's strongly dominant adaptive strategy is to drop out of the game.

- If the defender plays periodic with rate $\alpha > 1/(2k_1)$, then the attacker's strongly dominant non-adaptive strategy is to drop out of the game.

These findings result in the following principle:

DOP Principle: For valuable resources, the defender should play fast enough to force the attacker to drop out of the game.

To force the attacker out of the game, the move rate of the defender is dependent on the attacker's move cost. In order for the defender's benefit to be positive, her move cost should be lower than the attacker's ($k_0 < k_1$). As the ratio between the attacker's and defender's move costs increases, the defender improves his benefit. Achieving such conditions is discussed below.

(B) Principles related to game setup. In the security situations that we model, typically the defender has the advantage that she is responsible for setting up the game. Typically, the resource is initially controlled by the defender, and she can make various

design choices that can result in different game parameters. Below we highlight two principles related to controlling the attacker and defender move costs.

Move Cost Principles [MCP]. The defender's benefit increases if she arranges the game so that her moves cost much less than the attacker's moves. Lower move cost for the defender implies that the defender can play more frequently, and control the resource more. For some situations, a reduction in defender's move cost results in the ability of the defender to play with sufficiently high rate that it eventually forces the attacker to drop out of the game (as illustrated in the DOP principle). This observation leads to the following principle:

MCP Principle 1: The defender should arrange the game so that her moves cost much less than the attacker's.

An interesting research challenge for system designers is how to design an infrastructure in which refresh/clean costs are very low. We believe that virtualization has huge potential in this respect. For instance, refreshing a virtual machine has much lower cost than refreshing a physical machine. For cleaning a physical machine, full system wiping and reinstallation of all software is needed, while a virtual machine imagine can be simply restored from a clean-state version in a couple of minutes.

Moreover, the defender should make design choices that increase the attacker's move costs. This will result in the attacker playing less frequently, which in turn also implies higher control for the defender. Thus, the following principle can be derived:

MCP Principle 2: The defender should arrange the game so that she increases the move cost of the attacker.

Another interesting research problem for system designers is how to setup an infrastructure that increases the attacker's move costs in practice. For instance, sensitive data or cryptographic keys can be split (shared) over multiple storage servers such that only by accessing all servers can the sensitive data can be reconstructed while no information is obtained if at least one of the servers is not accessed/controlled. This reduces the attack surface: in order to compromise the sensitive data, the attacker needs to obtain control of all servers, effectively multiplying his move cost by the number of servers.

In our analysis from [17], we showed, for instance, that when playing with an exponential distribution against an LM-adaptive attacker, the defender can achieve benefits ranging from 0.1262 to 0.75 as the move cost ratio k_1/k_0 varies from 1 to 4. Similarly, when playing with a delayed exponential distribution the benefit achieved by the defender varies between 0.15 and 0.85 as the move cost ratio k_1/k_0 changes from 1 to 4. These examples clearly illustrate the MCP principles.

(C) Principles related to feedback received during the game. Our theoretical analysis in [17] demonstrates that any amount of feedback (even limited) received during the game about the opponent benefits a player in FLIPIT. Both players can control to some extent the amount of feedback received by the opponent, but again the defender has some advantage in setting up and knowing all the details of the internal infrastructure of the resource that she protects.

Feedback Principles [FP]. The defender's benefit increases if the amount of feedback received during the game about the attacker's moves is increased. Defenders, therefore, should monitor their systems frequently to gain information about the attacker's strategy

and detect potential attacks quickly after take over. Both monitoring and fast detection help a defender to more effectively schedule moves, which results in more control of the resource and less budget spent on moves, increasing the defender's benefit. As a consequence, the following principle follows naturally:

FP Principle 1: Defenders should monitor their resources to increase the amount of feedback received during the game.

Moreover, limiting the amount of feedback available to the attacker upon moving can also contribute to an increased benefit for the defender. The defender can employ various techniques to hide information about the exact time when she performed a move. The defender may, e.g., decide not to log timing information about when a system was cleaned. Accordingly, the following principle can be derived:

FP Principle 2: Defenders should limit the amount of feedback available to the attacker during the game.

3 Applications to Credential Expiration

In this section we highlight credential expiration as an application of particular practical interest. Credentials confirm the identity of a party. We focus on the two most common forms: *passwords* and *cryptographic keys*. The most common practice for managing credentials is to let credentials expire after a certain period. As we show the FLIPIT defending principles offer some simple, easy-to-deploy improvements to this practice.

We first discuss password reset and show the benefit of the randomized strategy principle. Next we discuss a storage service managed by a single enterprise that maintains directories with documents for its employees. Employees may update their documents, create new and remove old documents. We assume that access control to employee specific directories is managed by authentication keys. We discuss the well-established practice of key management by means of key-rotation and illustrate the DOP principle by a parameterization in which rational adversaries are forced to drop out. We extend our example by showing a reduction in defensive move costs when the storage service is outsourced to the cloud.

3.1 Password Reset

Knowledge of a password usually equates with control of a resource, such as an account. Thus we may view an adversary's attempt to compromise a password as a game of control. On learning a password, the adversary seizes control of an account. By resetting the password, the account owner regains control.

FLIPIT for password reset. When resetting a password, a user typically obtains no feedback on whether it's been compromised. Conversely, though, on (re-)compromising a password, an attacker learns whether it has been reset, simply by observing whether the password has changed since the last compromise. Where fixed-period password-reset policies are in force, an attacker will also generally know the period, as it's a matter of organization-wide policy. Worse still, in many situations, the adversary may also know or learn over time the *phase* of a user's password reset schedule. Password reset schedules are often determined by employee hire dates or exceptional system-wide password resets, neither of which is treated as secret (at least within an organization).

Password reset thus involves asymmetric knowledge. The defender receives no feedback, while the attacker learns whether a password is still valid. So a FLIPIT game for password reset is similar to the basic FLIPIT model with a non-adaptive defender and LM-adaptive attacker. The move cost for a defender is essentially the human overhead of creating and memorizing a new password. The cost of password compromise by the attacker depends on the environment: There are many vectors for password compromise, e.g., database breaches, password-stealing Trojans, etc.

Resetting passwords at fixed 90-day intervals, as commonly employed by most organizations today, is a poor defensive strategy. The Randomized Strategy (RS) principle offers a key insight into password refreshing:

> *To minimize adversarial control of a password-protected account, password resets should take place at* randomly *determined intervals.*

Case study. Consider the application of FLIPIT to the problem of password reset for corporate e-mail accounts. The cost to an attacker of compromising a password is perhaps most meaningfully reflected in the price of account passwords in underground markets. In a 2008 report on the underground economy, Symantec reported a price range of \$4-\$30 for a compromised e-mail password.[4] Quantifying the defender's cost in this setting is harder, as the overhead of password reset includes a substantial intangible burden on the user. Enterprise help-desk calls for password reset offer an indirect estimate of the human cost. A 2004 Gartner case study [18] documented an average cost of \$17.23 (here, rounded to \$17.00) per password reset call at a large beverage company.

Quantifying the benefit over time of account control is an even greater challenge, and depends largely on the attacker's control objective and its strategy for monetizing or otherwise exploiting a compromised account. We might notionally assume that the benefit of account control is equal for attacker and defender and also that the value of an account is much larger than the cost of password resets. With the cost of password reset at \$17 every 90 days, we assume that the value of the account is 10 times larger, resulting in approximately a value of \$2.00 per-day benefit. We'd like to highlight that these numbers are for illustration purposes only, the analysis can be easily adapted if some of the parameters change their values.

With these parameter settings, we can set $k_0 = 17/2 = 8.5$ and $k_1 \in [4/2, 30/2]$. For a defender playing with a 90-day period strategy against an adaptive attacker, the amount of control is 0 and her benefit is always negative at $0 - 8.5/90 = -0.09$. We quantify now the exact benefits for the attacker and defender in case the defender employs an exponential strategy and the attacker is LM adaptive. From our analysis in [17] (see Theorem 8), we distinguish two cases:

1. If $k_1 \geq k_0/0.854 = 10$, then the defender's optimal play is exponential with rate $\lambda = 1/k_1$ (and mean k_1), and the defender's benefit is $\beta_0 = 1 - k_0/k_1$. The attacker's best response is not playing at all and his benefit β_1 is zero.
2. If $k_1 < k_0/0.854 = 10$, the defender's maximum benefit is achieved by playing at rate $\lambda = (1 - (1 + z)e^{-z})/k_1$, where z is such that $(e^z - 1 - z)/z^3 = k_0/k_1$. The attacker's maximum benefit is achieved for playing periodically with period $\delta = z/\lambda$.

[4] http://eval.symantec.com/mktginfo/enterprise/white_papers/b-whitepaper_underground_economy_report_11-2008-14525717.en-us.pdf

We present in Table 1 the defender's optimal average inter-move delay (in days), the attacker's period of play (in days), and the defender's and attacker's optimal benefits (expressed in dollars) for different values of k_1. As observed, the defender always achieves positive benefit when employing an exponential strategy. As expected, the defender's benefit increases with higher attacker cost, validating the Move Cost Principle (MCP). The Drop Out Principle (DOP) is also demonstrated as the attacker's optimal strategy is not playing at all once his move cost exceeds a certain threshold.

Table 1. Parameters and benefits for exponential defender strategy. The defender's average inter-move delay and attacker's period are given in days and the defender's and attacker's benefit in $.

k_1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Def. average	289	193	145	116	97	84	74	66	10	11	12	13	14	15
Att. period	35	36	37	38	39	40	41	42	∞	∞	∞	∞	∞	∞
Def. benefit	0.058	0.088	0.116	0.14	0.16	0.206	0.236	0.266	0.3	0.45	0.582	0.692	0.784	0.866
Att. benefit	1.768	1.64	1.548	1.46	1.34	1.24	1.144	1.05	0	0	0	0	0	0

This case study illustrates the principle that the defender should use randomization when facing an adaptive attacker. At the same time, it shows the limitations of employing a simple non-adaptive strategy. Clearly, the defender achieves much lower benefit than the attacker when the attacker's move cost are lower or equal to the defender's. According to our Feedback Principles (FP), the defender would improve his benefit if she is able to obtain more feedback during the game. For instance, certain defensive techniques such as monitoring the infrastructure to detect password compromises or requiring multi-factor authentication will enhance the defender's benefit.

Variant. We might consider an enhanced password-reset model with asymmetric play as well. The attacker then has a second action type available to it, a *check* that determines whether it still has control, and is distinct from a *move*. This move corresponds to an attacker attempt to use a password in order to check its validity. A check move, like an ordinary reset move, carries some cost: It increases the detection risk for the attacker. (In a system in which unsuccessful login events are logged, for instance, every check will potentially trigger an investigation by the defender.)

3.2 Key Management

We now explore the application of FLIPIT in the area of *key management*. We examine the use case of key management for authenticating employee directories, by considering two concrete, contrasting deployment models:

- **Deployment of key management within a single enterprise:** This deployment model is very commonly used in the industry today. Widely adopted key management products, e.g., from IBM, HP, EMC, Thales, Symantec/PGP and many other vendors, provide solutions for managing cryptographic keys at an enterprise level. Architectural and security considerations for this model have been discussed in standards such as NIST SP 800-57, NIST SP 800-130, ANSI 9.26 and ISO 11770. Threat models have been discussed in NIST SP 800-30.

- **Deployment of key management within a cloud infrastructure**. This deployment model relies on the shared infrastructure of a cloud service provider on behalf of multiple tenant enterprises, and is emerging as a significant alternative to enterprise-

based key-management infrastructures. Architectural and security considerations for this model have been discussed in the Cloud Security Alliance Security Guidance for Critical Areas of Focus in Cloud Computing 3.0.[5] We focus on the deployment of enterprise-specific key managers within a dedicated and isolated segment of a Cloud Service Provider infrastructure, a model already exemplified in commercial products, e.g., Microsoft Azure Trust Services.

The FLIPIT game offers an alternative way to look at the question of whether and when to use key rotation. As NIST SP 800-57, Part 2 [2] (pp. 45), suggests, evaluation of key rotation policy should take into account "the threat to the information (e.g., who the information is protected from, and what are their perceived technical capabilities and financial resources to mount an attack)." We achieve this with FLIPIT, by exploring whether there are ways to take advantage of key rotation that might invoke the Drop Out Principle, so that the best strategy for the attacker is to defect from the game.

FLIPIT for key rotation. In this game, the defender's moves implement key rotation in order to refresh keys. We will assume that the defender plays a non-adaptive periodic strategy because she does not see the attacker's moves until the point at which the compromise is exposed. We assume the following parameters for the defender:

- Refreshing a single authentication key costs about \$1 (this estimate seems to be well supported due to the cost of interaction with the parties who need the authentication key for accessing their directory). Let u be the period (measured as a fraction of a year) at which the defender rotates each key. Then $1/u$ equals the defender's move cost in \$ per key per year.

- If the attacker gets hold of an authentication key, then the loss to the defender due to the leakage of the protected documents (which are updated, created and removed continuously) is assumed to be about \$200 /year (this estimate comes from Ponemon estimating in their 2012 report[6] that the costs for responding to a data breach incident typically equals $\$204$ per stolen credit card record; here we assume a continued loss of $\$200$ /year due to illegitimate access to protected documents). So, being in control of an authentication key means that the defender does not incur a loss at rate $\$200$ /year. We may model this as a gain of $\$200$ /year being in control. So, for γ_0 denoting the fraction of time the defender is in control of an authentication key, the defender's gain is equal to $\$200\gamma_0$ per key per year.

In FLIPIT notation, where benefit is normalized with respect to the value of being in control, the defender's benefit is equal to $\gamma_0 - k_0/u$, where $k_0 = 1/200$.

We assume a typical scenario of an enterprise with $10,000$ authentication keys (this estimate comes from [2]). For the attacker we assume the following parameters:

- To exfiltrate keys an attacker needs to launch an attack of $\$10,000$ (move) cost.

- Let n be the total number of exfiltrated keys per attack. As n increases, the defender detects the attack with higher probability. We model this by introducing a parameter r, the probability that the defender detects the compromise of a single key. Then, assuming that the detection of different key compromises are independent events, the

[5] https://cloudsecurityalliance.org/guidance/csaguide.v3.0.pdf.

[6] Ponemon Institute, 2009 Annual Study, http://www.ponemon.org, "Cost of a Data Breach: Understanding Financial impact, Customer Turnover and Preventive Solutions."

probability of detecting the compromise of n keys is $1-(1-r)^n$. In the first example of deployment within the enterprise, we consider the worst case for the defender, in which the attacker is detected with probability 0. When deploying key management to the cloud, we assume a better detection mechanism ($r > 0$) as the cloud provider handles keys of multiple tenants.

- If the attacker gets an authentication key without being detected, then access to the protected documents leads to a gain of about $4 /year (monetizing leaked data is orders of magnitude less than the loss to the defender caused by their leakage).

Summarizing, a single attack in which n keys are extracted costs $\$10,000$ and leads to an expected value of being in control of $[(1-(1-r)^n)\cdot 0+(1-r)^n\cdot 4]\cdot n$ $/year. In FLIPIT notation: $k_1 = 10000/[4(1-r)^n n]$.

An adaptive attacker drops out if he can get no benefit at all: this happens if $u < k_1$. This shows that if the defender chooses u slightly less than $k_1 = 2500/[(1-r)^n n]$, then the attacker must drop out in order to avoid a negative benefit. In the enterprise deployment example, in the worst case for the defender, $r = 0$ (no detection) and $n = 10000$ (all keys within the enterprise are stolen). For these parameters the defender should choose $u < 0.25$, i.e., the defender's period is at most 3 months. The defender's benefit per key per year is equal to $\$(200 - 1/u) = \$196$ which is very close to $200. Hence, for a small cost of $4 per key per year, no documents will be stolen by a rational adversary playing a periodic strategy. Overall, the Drop Out Principle offers a key insight for effective key rotation:

> *To minimize the possibility of exfiltration of documents protected using authentication keys, key rotation should be applied sufficiently often; at least every 3 months in our concrete setting above.*

For a non-adaptive attacker the drop out condition is $u < 2k_1$, and key rotation must occur at least every 6 months in which case for a small cost of $2 per key per year for the defender no documents will be stolen by a rational adaptive attacker. So, by reducing the feedback available to the attacker, the defender halves his cost per key per year. This demonstrates the Feedback Principle:

> *To minimize the possibility of exfiltration of documents protected using authentication keys, key rotation should be applied sufficiently often; for a non-adaptive attacker at least every 6 months in our concrete setting above.*

As an extension to this case study we now assume that the enterprise outsources its document storage and key management to a cloud provider. Since the cloud provider manages the keys of several enterprises/tenants, we may assume that more detection mechanisms are available to detect stolen keys, e.g., let us assume that the probability of detecting the event of stealing one key is equal to $r = 1/10000$.

The optimal defender's period u against an adaptive attacker is now equal to $k_1 = 2500/[(1-r)^n n]$, which is in the worst case minimized for n equal to the minimum of $-1/\ln(1-r)$ and 10,000, the total number of keys of a single enterprise. This results in an optimal defender's period of $u = 0.68$ or 248 days. This shows that the increased risk to the attacker allows the defender/cloud to choose a period which is 2.72 times larger than the 3 months key rotation period for enterprise key management. This leads to a reduction from $4 cost per key to $1.47 cost per key. (A second benefit of having a

cloud provider manage enterprise keys is a reduction in the initial start up costs which is now shared among all the tenants of the cloud provider.)

The main goal in both the enterprise and the cloud service provider game is to invoke the Drop Out Principle, creating such a significant advantage for the defender that the rational strategy for the attacker is to quit the game. A key factor in achieving this result is the risk to the attacker. We have formulated this risk in terms of the possibility that the value of the stolen information will be negated if and when the attack is exposed. Such a result can be demonstrated in a number of real-life situations in which the rapid discovery of an attack prevented the attacker from deriving value from their theft, such as in the case of the 2011 attack on Lockheed-Martin that attempted to use information stolen from RSA, as a vector in the attack.[7]

4 Other Applications

We next consider two more applications of FLIPIT, emphasizing its breadth of application, rather than detailed analysis. We first examine defensive virtual-machine refresh. While less mature a practice than password reset and key rotation, it's an emerging approach that fits well within the basic FLIPIT framework. Secondly, we consider FLIPIT as a model for automated (cryptographic) cloud service auditing.

4.1 Virtual-Machine Refresh

Virtualization is seeing heavy use today in the deployment of servers in data centers. As individual servers may experience periods of idleness, consolidating multiple servers as VMs on a single physical host often results in greater hardware utilization. Similarly, Virtual Desktop Infrastructure (VDI) is an emerging workplace technology that provisions users with VMs (desktops) maintained in centrally managed servers. In this model, users are not bound to particular physical machines. They can access their virtual desktops from any endpoint device available to them, even smart phones.

While virtualization exhibits many usability challenges, one key advantage is a security feature: VMs can be periodically refreshed (or built from scratch) from "clean" images.

Takeover of a VM results in a game very similar to that for a physical host. Virtualization is of particular interest in the context of FLIPIT, though, because FLIPIT offers a means of measuring its security benefits. Refreshing a VM is much less cumbersome than rebuilding the software stack in a physical host. In other words, virtualization lowers the move cost for the defender illustrating the Move Cost Principle (MCP):

> *When designing system infrastructures, virtualization is a key technique useful in reducing the defender's move cost.*

4.2 Cloud Service Auditing

When a cloud service provider furnishes a resource to a client, it's desirable for the client, or an auditor acting on its behalf, to *audit* the provider. A provider generally furnishes resources to clients under a Service-Level Agreement (SLA), a contractual

[7] See article in Infosecurity Magazine at http://www.infosecurity-magazine.com/view/18299.

specification of configuration options and minimum service levels. Compliance or deviation from an SLA, however, isn't always readily apparent to clients—particularly for security or reliability objectives.

Case study. A cloud provider stores every file F under an SLA stipulating retention of two redundant copies to protect against file loss.[8] But given the extra hardware costs of compliance, the cloud provider has a financial incentive not to store extra file copies. At the same time, a client / tenant will access only the primary copy of F in the course of ordinary use. So it won't detect non-compliance by the provider, i.e., dropped copies.

A client thus has an interest in challenging the cloud from time to time to prove storage of extra copies of F.[9] For each challenge, however, a client will be billed for the associated resource costs. So, how can a client most cost-effectively schedule its challenges? Of course, there are many other service-level and security provisions that a client may wish to audit, e.g., computing-resource isolation [19], retrievability of archived files (e.g., [1,4]), database integrity (e.g.,[11]). Data-center audits, as outlined, for instance, in the popular SAS70 standard[10] also include inspections of much more than computing resources—such as workforce screening, physical maintenance, and so forth. These types of audit may be modeled in FLIPIT as well. Modeling the audit process in FLIPIT, as we now show, offers insight into how a client can best schedule challenges to detect and rectify service-provider compliance failures.

Although real-time cloud-service auditing, applied as remote spot-checks, isn't common today, it will inevitably become a regular practice, as recognized by the growth of supporting standards such as SCAP and CloudTrust. The growing literature on remote testing of cloud security properties [1,4] largely neglects the question of how challenges should be scheduled, or assumes a simplistic partitioning of time into epochs. Overall, FLIPIT offers a more refined temporal framework for these protocols.

FLIPIT for cloud service audit. For brevity, we use the terms *up* and *down* to denote compliant and non-compliant states respectively for a target resource. In our case study, F is up if the provider is storing at least two extra copies of F; otherwise, it's down.

In a FLIPIT game for audit, we may view the provider as controlling the resource while it's down. We presume that for resources of interest, a down state is more cost-effective for the provider. Conversely, the client has control while the resource is up, i.e., in the state desired by the client (and perhaps specified in an SLA).

It's convenient to think of the provider as the *attacker* and the client as the *defender*.

In some cases, discovery of a down resource—e.g., a corrupted database, lost archive, or a VM infected through provider negligence—might be grounds for service termination by the client. In most cases, though, SLAs are broad, i.e., encompass a wide variety of service requirements, at the same time that client-provider relationships are sticky because of high switching costs. Provider penalties for down resources are then unlikely to be fatal, but might instead carry a small penalty or degrade the provider's reputation.

[8] This policy is common: Amazon, for instance, claims to store redundant copies of files in its storage services, such as S3. (See at: http://aws.amazon.com/backup-storage.)

[9] The RAFT protocol [3] is an example of how such a test may be performed remotely.

[10] Statement on Auditing Standards No. 70, AICPA, http://sas70.com.

Here, we assume that when a client discovers that a resource is down, the server is immediately forced to bring the resource up. In other words, a challenge transfers control to the defender. Our model can also include a provider penalty.

While the client / defender only learns when a resource is down at the time of a challenge, the provider / attacker, of course, knows the resource state exactly. Thus, the key difference over the standard FLIPIT setup is asymmetric knowledge: The defender learns the game state only when taking control, while *the attacker has full knowledge of the game state, i.e., at all times knows the complete history of defender moves*.

Lessons. Our Randomization Principle (RP) shows that the defender must adopt a randomized strategy to perform well, i.e., audit spot checks must be unpredictable to be effective in an adversarial environment.

> *An optimal cloud service auditing strategy is* adaptive*, i.e., conditions challenge times on the observed compliance or non-compliance of the provider.*

The defender has a disadvantage in the game defined so far as the cloud provider has complete feedback about the defender's moves. The Feedback Principle (FP) teaches us that the defender further benefits in the game if the exact audit times are not divulged to the provider. To implement such a defensive technique, the defender might, for instance, use an auditing technique having the property that audit requests are indistinguishable from normal requests. This allows the defender to spread and hide her audits at slow rate among the normal requests to the cloud. We believe that designing such an auditing techniques is an interesting topic of future work.

5 Applications to Modeling Advanced Persistent Threats

Advanced Persistent Threats (APTs) are sophisticated, well funded, and highly targeted campaigns against a computing, digital, or cyberphysical resource. High-profile examples have first come to light over the past couple of years; it's likely that many have merely gone undetected or unpublicized. In addition to Stuxnet and that against RSA, APT attacks have also been reported against Google[11] and Lockheed Martin,[12] while the recently surfaced Duqu worm[13] appears to be an updated incarnation of Stuxnet.

Conceived in part as a modeling tool for APTs, we view FLIPIT as a useful encapsulation of new set of assumptions and defensive strategies in the cybersecurity community. Traditional perimeter defense is giving way to a view of cyberdefense as an ongoing territorial war. In other words, FLIPIT reflects an essential new reality:

> *Continuous, partial compromise of critical infrastructure and large, high-value computing systems is inescapable.*

As a macro-level game, FlipIt characterizes the uncertain knowledge and dynamics of play of an attacker and defender in an APT scenario. Of interest is a more nuanced view that composes micro-level FlipIt games of the type described in Section 4. We sketch a possible approach to such composition here.

Consider a scenario in which the goal of an attacker is to establish a control path from an external command center to a target sensitive resource internal to a defender's

[11] See at http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2010-0249 for more details.
[12] See at http://www.nytimes.com/2011/05/28/business/28hack.html?_r=3 for more details.
[13] See at http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2011-3402 for more details.

system. The state of the system may be modeled as a directed *attack graph* $\mathcal{G} = (V, E)$ [16], in which the internal nodes V are subsystems, a unique source represents an outward- (Internet-) facing resource, and a unique sink represents the target resource. (Generalizations to multiple sources and sinks are possible, of course.) Edges in the graph represent different attack vectors that the attacker can exploit to compromise new nodes and traverse the graph. A node is considered compromised if the attacker has reached it through a path from the source by a series of exploits. These exploits represent attacker moves. The defender moves by fixing exploits and "cleaning" infected nodes.

In this graph-based FLIPIT extension, the objective of the attacker is to maximize the fraction of time it controls a path from source to sink. The objective of the defender is to maximize her control of the target sensitive resource. Consequently, players are seeking to maximize control, rather than benefit. An appropriate global model, therefore, is one in which players have rate-limited global budgets.

One interesting question is how can the defender allocate the global budget across various resources with the global goal of protecting the sensitive target resource. An effective defensive strategy is to protect the resources along the graph's min-cut. In graph theory, a min-cut is defined as the number of edges of total minimum cost that have to be removed so that there is no path from the source to the target node. Translating to our setting, a cut in the attack graph represents a set of resources that, when controlled by the defender, prevent the attacker from reaching the target node. The min-cut represents the cut that can be controlled by the defender with minimum cost.

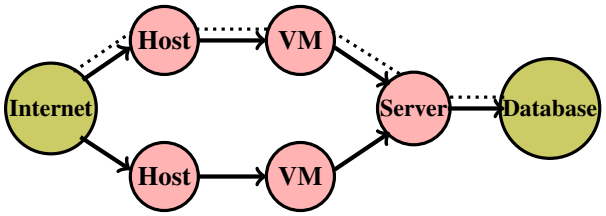

Fig. 3. Micro-level games in an APT scenario and attacker control path (dotted edges)

Case study. Figure 3 depicts an example attack graph that incorporates some of the micro-level games described in Section 4. The target resource is the financial database of a corporation. (Ongoing compromise confers the valuable ability to anticipate and even potentially manipulate the corporation's financial results.) The nodes include:

1. **Hosts:** In this example, hosts are endpoint devices, e.g., PCs, that the attacker subverts by means of social-engineering attacks (e.g., delivery of infected documents).
2. **VMs:** Each user has a virtual desktop (VM in a VDI) for access to sensitive company information. The attacker uses a compromised host as a staging point for compromise of the associated user's VDI instance.
3. **Password-protected server:** The front-end to the target database is a password-protected server. Given control of a VDI instance, and compromise of the server password, the attacker can log into the server and access the database.

On every graph node, the attacker and defender play a distinct game of FLIPIT.

Consider, for example, the following simple setup. Suppose both players play non-adapative strategies. Let $\lambda_i^{(v)}$ denote the rate of play of player i on node $v \in V$ and $k_i^{(v)}$ the move cost of player i for the game on node v. Let Λ_i denote a budget for player i, meaning that the player's strategy must meet the bound $\sum_{v \in V} \lambda_i^{(v)} k_i^{(v)} \leq \Lambda_i$.

For example, suppose that $\Lambda_0 = \Lambda_1 = 1$ and all move costs are 1. One possible strategy is for each player to play exponentially with rate $1/5$ on each of the five internal nodes. In this case, given that the two players play identical strategies, both the attacker and defender control each internal node half the time. The attacker's control of a specific adjacent host / VM pair at a given time is asymptotically thus $1/4$, while that of *at least one* of the two pairs is $7/16$. The password-protected server is controlled half of the time. Asymptotically, then, the attacker will control a source-sink path for a fraction of time equal to $7/16 \times 1/2 = 7/32$.

Assume that move costs for different resources are different. For instance, refreshing a virtual machine has a much lower cost than cost controlling the password-protected server. More concretely, assume that the defender costs for refreshing a VM, controlling the server, and controlling a host are 1,10, and 3, respectively. Then the min-cut of the graph from the defender's perspective is to play on the two VM nodes. The defender, therefore, should refresh virtual machines frequently and control a cut in the graph.

We leave it as an open research problem to decide whether always playing on a min-cut in an attack graph is optimal for the defender.

6 Related Work

FLIPIT was first presented at an invited talk by Ron Rivest at CRYPTO 2011[14]; [17] introduces FLIPIT and gives a formal treatment with theoretical analysis.

In the game theory literature, FLIPIT is related to "repeated games" (see, for example, the excellent text by Mailath and Samuelson [6]), but it differs from them through its stealthy aspect and continuous time. Nonetheless, at a higher level, FLIPIT does share some qualitative characteristics with repeated games. If both players of FLIPIT play adaptively, then FLIPIT acquires the rich complexity of repeated Prisoner's Dilemma, where players may choose to cooperate for their mutual benefit.

FLIPIT is also related to a game of timing [13] where (1) there is an infinite time interval and a finite amount of resources (moves) within each finite subinterval, and (2) the resources/moves of a player are either silent (i.e., the other player does not learn when the moves take place) or noisy with delay till the other player moves (i.e., the other player learns the full history of moves when he moves himself). As future work, we plan to investigate how the theory of games of timing applies to FLIPIT.

Conventional game theory has a long history of application to and enhancement by cryptography and network security; see [5,7] for two surveys. More pertinent to FLIPIT are games modeling system security. Roy et al. [15] offer a taxonomy and survey of game-theoretic models in network security in particular. They note a preponderance of games differing from FLIPIT in two ways: The games assume perfect information (i.e., players know the full game state) and synchronous moves by players.

Some recent information security modeling has made use of extensive forms, which permit complex modeling, strictly within a framework of synchronous moves. This approach gives rise to security games with imperfect information, as in a game devised

by Moore et al. [8] to model zero-day disclosure by competing entities. Nguyen et al. [10] consider an abstract, repeated security game with imperfect information and also incomplete information, in the sense that players don't know one another's payoffs. Related work also includes a synchronous territorial game of incomplete, perfect information proposed by Pavlovic [12] which models two-player cybersecurity scenarios for information gathering via deception.

7 Conclusion

While its rules are simple, we have shown that FLIPIT is a conceptually rich security model that yields both important general defensive principles and specific guidance in a number of real-world security scenarios. The Randomized Strategy Principle, for instance, yields a beneficial randomization of password-reset policies. The Drop Out Principle highlights the importance of key rotation frequency. The Move Cost Principle underscores one of the benefits of virtualization, namely its reduction in defender's move costs. FLIPIT offers similarly useful insights across a broad range of real-world security applications, of which we've presented only a small set here. It also gives rise to a wealth of variants applicable to diverse and potentially complex security scenarios.

While this paper provides a glimpse into the applications of FLIPIT, the underlying model of *complete* and *silent* compromise has countless uses, especially in a world where no system is safe and the longstanding assumptions of cryptographers and security system designers can no longer be taken for granted.

References

1. G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song. Provable data possession at untrusted stores. In *Proc. 14th ACM Conference on Computer and Communication Security (CCS)*, 2007.
2. E. Barker, W. Barker, W. Polk, and M. Smid. Recommendation for key management II: Best practices for key management organization. *NIST SP*, (2/3):1–79, 2005.
3. K. D. Bowers, M. van Dijk, A. Juels, A Oprea, and R. Rivest. Remote assessment of fault tolerance. In *Proc. 18th ACM Conf. on Computer and Communication Security (CCS)*, 2011.
4. A. Juels and B. Kaliski. PORs: Proofs of retrievability for large files. In *Proc. 14th ACM Conference on Computer and Communication Security (CCS)*, pages 584–597, 2007.
5. J. Katz. Bridging game theory and cryptography: Recent results and future directions. In *Proc. Theory of Cryptography Conference (TCC)*, pages 251–272, 2008.
6. G. J. Mailath and L. Samuelson. *Repeated Games and Reputations: Long-run relationships*. Oxford, 2006.
7. M. Manshaei, Q. Zhu, T. Alpcan, T. Basar, and J.P. Hubaux. Game Theory Meets Network Security and Privacy. Technical report, EPFL, 2010.
8. T. Moore, A. Friedman, and A. Procaccia. Would a "cyber warrior" protect us? Exploring trade-offs between attack and defense of information systems. In *NSPW*, pages 85–94, 2010.
9. R. B. Myerson. *Game Theory—Analysis of Conflict*. Harvard University Press, 1997.
10. K. C. Nguyen, T. Alpcan, and T. Basar. Security games with incomplete information. In *Proc. IEEE International Conference on Communications (ICC)*, 2009.
11. Charalampos Papamanthou, Roberto Tamassia, and Nikos Triandopoulos. Optimal verification of operations on dynamic sets. In *CRYPTO*, pages 91–110, 2011.
12. D. Pavlovic. Gaming security by obscurity, 2011. CoRR abs/1109.5542.

13. T. Radzik. Results and problems in games of timing. *Statistics, Probability and Game Theory*, 30, 1996.
14. R. L. Rivest. Illegitimi non carborundum. Invited keynote talk given at CRYPTO 2011, August 15, 2011. `http://people.csail.mit.edu/rivest/pubs.html#Riv11b`.
15. S. Roy, C. Ellis, S. Shiva, D. Dasgupta, V. Shandilya, and Q. Wu. A survey of game theory as applied to network security. In *Int. Conf. on System Sciences (HICSS)*, pages 1–10, 2010.
16. O. M. Sheyner. Scenario graphs and attack graphs, 14 April 2004. PhD Thesis, Carnegie Mellon University, CMU-CS-04-122.
17. M. van Dijk, A. Juels, A. Oprea, and R. L. Rivest. FlipIt: The game of "stealthy takeover". To appear in Journal of Cryptology, 2012.
18. R. J. Witty, K. Brittain, and A. Allen. Justify identity management investment with metrics, 23 February 2004. Gartner Group report.
19. Y. Zhang, A. Juels, A. Oprea, and M. K. Reiter. HomeAlone: Co-residency detection in the cloud via side-channel analysis. In *IEEE Symp. on Security & Privacy*, pages 313–328, 2011.