Extending the analysis of the `FlipIt` game

by

Michalis Rossides

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2013

Author .....
Department of Electrical Engineering and Computer Science
May 20, 2013

Certified by .....
Ronald L. Rivest
Viterbi Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by .....
Professor Dennis M. Freeman
Chairman, Department Committee on Graduate Theses

Extending the analysis of the `FlipIt` game

by

Michalis Rossides

Submitted to the Department of Electrical Engineering and Computer Science
on May 20, 2013, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Computer Science and Engineering

Abstract

In this thesis, we extend the game theoretical analysis of the `FlipIt` game[1]. The game was first articulated and analyzed by Marten van Dijk, Ari Juels, Alina Oprea and Ronald L. Rivest. `FlipIt` (or otherwise The Game of "Stealthy Takeover") is a game-theoretic framework for modeling computer security scenarios. Such scenarios include targeted attacks, cryptographic key rotation, password changing policies and cloud auditing.

What we are particularly interested in are situations in which an attacker repeatedly takes control over a system or critical resource completely, learns all its secret information and this is not immediately detected by the system owner. As mentioned in the original paper[1], `FlipIt` is a two-player game between an *attacker* and a *defender*, in which both players try to control a shared resource. The players do not know who is currently in charge of the resource until they move; this is why it is called "Stealthy Takeover." Moreover, each player pays a specific cost every time he moves. The objective of each player is to maximize his benefit; this involves controlling the resource for as large fraction of time as possible and, at the same time, minimizing the total move cost.

Here, we are only considering scenarios in which the defender plays with a renewal strategy; in particular, the intervals between his consecutive moves are given by a fixed probability distribution. However, the attacker can be more sophisticated than that and deploy adaptive strategies; i.e. he moves based on feedback received during the game. The main strategy we are considering for the defender is the *biperiodic* as stated in Section 4. For the attacker, we are considering various non-adaptive and adaptive strategies. Finally, we compare it to the relevant strategies already analyzed in the original FlipIt paper[1].

Thesis Supervisor: Ronald L. Rivest

Title: Viterbi Professor of Electrical Engineering and Computer Science

Acknowledgments

In this section, I want to take the time and thank my thesis advisor, professor Ronald L. Rivest, for all his guidance from the inception of the thesis until the end. He has been an excellent mentor and a great motivator. In addition, I want to thank Ari Juels and Alina Oprea from the RSA Labs for sharing their code for all the work done in the original FlipIt paper[1].

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The primary application of the `FlipIt` game is within the field of cybersecurity. In fact, the game was motivated by *Advanced Persistent Threats* (APTs). These attacks last for long periods of time in a system and they advance stealthily and slowly to avoid detection. Thus, we need a game theoretical framework to analyze situations in which an attacker periodically compromises a system (i.e. he learns its entire state, including its secret keys). We are especially interested in those scenarios where theft is *stealthy* and not immediately noticed by the defender.

`FlipIt` is a two-player game with a shared resource that both players wish to control. Example of resources are a secret key, a password or an entire infrastructure. The two players are called *defender* and an *attacker*. The defender could be a large organization that has its data stored in their network system. The resource is the network system and the attacker could be a hacker, who is trying to access the data.

- Players take control of the resource by "moving" and paying a certain cost per move.
- Players can move at any given time and they do not necessarily alternate turns.
- A player does not immediately know when the other player moves; he only finds out who was in control once he moves.

- Each player is trying to maximize his *benefit*, defined as the fraction of time the player controls the resource minus the average move cost. Thus, it is optimal to maximize the fraction of time the player controls the source and minimize the number of moves.

Game definition by example This example is taken from the original paper[1]. In an example implementation of a basic version of `FlipIt`, each player has a control panel with a single button and a light. The player may push his button at any time; in the most general form of `FlipIt`, we consider time to be continuous, but we support discrete variants as well. Pushing the button always causes the button-pusher to take ownership of the resource. We assume that players don't push their buttons at exactly the same time (or, if they do, then ownership doesn't change hands).

If pushing the button causes ownership to change hands, then the light flashes when the button is pushed. We call this a "takeover." If the button-pusher already had ownership of the resource, then the light doesn't flash and the button-push was wasted.

The players can't see each other's control panels, and thus the defender doesn't know when the attacker takes control, and vice versa. The only way a player can determine the state of the game is to push his button. Thus, a move by either player has two consequences: it acquires the control of the resource (if not already controlled by the mover), but at the same time, it reveals the pre-move state of the resource to the player taking control.

There is always a cost to pushing the button. In this example, pushing the button 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 the number of times he has pushed his button.

We mostly care about analyzing "optimal play" for both attacker and defender in this simple game. We wish to explore this question under various assumptions about the details of the rules or about the strategies the players employ.

Strategies What kind of strategies should the players deploy? We summarize in a hierarchical, graphical representation the various classes of strategies for any player in Table 1.1. The strategies are ordered (from top to bottom) by increasing amount of feedback received by a player during the game. In addition, as we move (from top to bottom) we encompass broader classes of strategies. There are two main categories of strategies for a player: *non-adaptive strategies* (NA) in which the player does not receive any feedback during the game and *adaptive strategies* (AD) in which the player receives certain types of feedback when moving.

Categories	Classes of Strategies		
Non-adaptive (NA)	Exponential	Biperiodic	Periodic
	Renewal		
	General non-adaptive		
Adaptive (AD)	Last move (LM)		
	Full history (FH)		

Table 1.1: Hierarchy of strategies in `FlipIt`.

In the class of non-adaptive strategies, one subclass of particular interest is that of *Renewal strategies*. For a player employing a renewal strategy, the intervals between the player's consecutive moves are independent and identically distributed random variables generated by a *renewal process*. Examples of renewal strategies include: the *periodic* strategy in which the interval between two consecutive moves is fixed at a constant value, the *exponential* strategy in which the intervals between consecutive moves are exponentially distributed and the *biperiodic* strategy in which the interval between two consecutive moves is either τ_a with probability p or τ_b with probability $1 - p$.

In the class of adaptive strategies, we distinguish a subclass called *last-move (LM)*, in which the player finds out upon moving the exact time when his opponent moved last. In a general adaptive strategy, a player receives complete information about his opponent's moves (this is called *full-history (FH)*. When the opponent is playing with a renewal strategy, then *LM* and *FH* are equivalent.

A further dimension to a player's strategy is the amount of information the

player receives before the game starts. Besides the case in which the player receives no information about his opponent before the game, other interesting cases are: (1) *rate-of-play (RP)*, in which the player finds out the exact rate of play of his opponent and (2) *knowledge-of-strategy (KS)*, in which the player finds out full information about the strategy of the opponent (but not the opponent's randomness).

Results from the original paper The results are summarized in Table 1.2. The defender plays with a renewal strategy and we only present the cases where the attacker is more powerful than the defender and organize these cases according to the strength of the attacker. Since the game is symmetric, analogous results apply when the defender is more powerful than the attacker.

Attacker category	Strategy classes		Results
	Attacker	Defender	
NA	Periodic	Periodic	Nash equilibrium (Thm 1)
	Renewal	Renewal	Periodic strategies dominate (Thm 4)
NA+ Additional information	Renewal RP	Renewal	Periodic strategies dominate (Thm 4) Optimal parameters (Thm 5)
	Renewal (RP)	Renewal (RP)	Periodic strategies dominate (Thm 4)
AD	LM	Periodic	Periodic: dominant for attacker
		Exponential	Periodic: dominant for attacker (Thm 6) Optimal parameters (Thms 7 and 8)
		Delayed exponential	Dominant attacker strategy (Thm 9) Defender's benefit increases compared to exponential (Experimental)
	Greedy	Renewal	Analyze several defender distributions Exponential defender: dominant strategy

Table 1.2: Summarized results from the original `FlipIt` paper.

The rest of this thesis is divided as follows: Chapter 2, gives the formal definition of the game. Chapter 3, contains the complete analysis of a defender playing with Biperiodic strategy and the attacker increasing his strategy's sophistication as the chapter proceeds. Chapter 4, summarizes the main results in the case where the defender deploys an Exponential strategy and the attacker deploys an adaptive strategy. Chapter 5, is a synopsis of the work done, including the important contributions.

Chapter 2

Formal Definition and Notation

This section gives a formal definition of the stealthy takeover game, exactly as it is formulated in the FlipIt paper.[1] Additionally, it introduces various pieces of useful notation.

Players There are two players: the defender is the "good" player, identified with 0 (or Alice). The attacker is the "bad" player, identified with 1 (or Bob). It is convenient in our development to treat the game as symmetric between the two players.

Time The game begins at time $t = 0$ and continues indefinitely as $t \to \infty$. In the general form of the game, time is viewed as being *continuous*, but we also support a version of the game with *discrete* time.

Game state The time-dependent variable $C = C(t)$ denotes the current player controlling the resource at time t; $C(t)$ is either 0 or 1 at any time t. We say that the game is in a "good state" if $C(t) = 0$, and in a "bad state" if $C(t) = 1$.

For $i = 0, 1$ we also let

$$C_i(t) = I(C(t) = i)$$

denote whether the game is in a good state for player i at time t. Here I is an "indicator function": $I(\cdot) = 1$ if its argument is true, and 0 otherwise. Thus, $C_1(t) =$

$C(t)$ and $C_0(t) = 1 - C_1(t)$. The use of C_0 and C_1 allows us to present the game in a symmetric manner.

The game begins in a good state: $C(0) = 0$.

Moves A player may "move" (push his / her button) at any time, but is only allowed to push the button a finite number of times in any finite time interval. The player may not, for example, push the button at times $1/2$, $2/3$, $3/4$, ..., as this means pushing the button an infinite number of times in the time interval $[0, 1]$. (One could even impose an explicit lower bound on the time allowed between two button pushes by the same player.)

A player cannot move more than once at a given time. We allow different players to play at the same time, although with typical strategies this happens with probability 0. If it does happen, then the moves "cancel" and no change of state happens. (This tie-breaking rule makes the game fully symmetric, which we prefer to alternative approaches such as giving a preference to one of the players when breaking a tie.) It is convenient to have a framework that handles ties smoothly, since discrete versions of the game, wherein all moves happen at integer times, might also be of interest. In such variants ties may be relatively common.

We denote the sequence of move times, for moves by both players, as an infinite nondecreasing sequence

$$\mathbf{t} = t_1, t_2, t_3, \ldots .$$

The sequence might be nondecreasing, rather than strictly increasing, since we allow the two players move at the same time.

We let p_k denote the player who made the k-th move, so that $p_k \in \{0, 1\}$. We let $\mathbf{p}$ denote the sequence of player identities:

$$\mathbf{p} = p_1, p_2, p_3, \ldots .$$

We assume that $t_1 = 0$ and $p_1 = 0$; the good player (the defender) moves first at time $t = 0$ to start the game.

For $i = 0, 1$ we let

$$\mathbf{t}_i = t_{i,1}, t_{i,2}, t_{i,3}, \ldots$$

denote the infinite increasing sequence of times when player i moves.

The sequences $\mathbf{t}_0$ and $\mathbf{t}_1$ are disjoint subsequences of the sequence $\mathbf{t}$. Every element t_k of $\mathbf{t}$ is either an element $t_{0,j}$ of $\mathbf{t}_0$ or an element $t_{1,l}$ of $\mathbf{t}_1$.

The game's state variable $C(t)$ denotes the player who has moved most recently (not including the current instant t), so that

$$C(t) = p_k \text{ for } t_k < t \leq t_{k+1} \text{ and for all } k \geq 1 .$$

When $C(t) = i$ then player i has moved most recently and is "in control of the game", or "in possession of the resource." We assume, again, that $C(0) = 0$.

Note that $C(t_k) = p_{k-1}$; this is convenient for our development, since if a player moves at time t_k then $C(t_k)$ denotes the player who was previously in control of the game (which could be either player).

For compatibility with our tie-breaking rule, we assume that if $t_k = t_{k+1}$ (a tie has occurred), then the two moves at times t_k and t_{k+1} have subscripts ordered so that no net change of state occurs. That is, we assume that $p_k = 1 - C(t_k)$ and $p_{k+1} = 1 - p_k$. Thus, each button push causes a change of state, but no net change of state occurs.

We let $n_i(t)$ denote the *number of moves made by player* i up to and including time t, and let

$$n(t) = n_0(t) + n_1(t)$$

denote the *total number of moves made by both players* up to and including time t.

For $t > 0$ and $i = 0, 1$, we let

$$\alpha_i(t) = n_i(t)/t$$

denote the *average move rate by player* i up to time t.

We let $r_i(t)$ denote the time of the most recent move by player i; this is the

largest value of $t_{i,k}$ that is less than t (if player i hasn't moved since the beginning of the game, then we define $r_i(t) = -1$). Player 0 always moves at time 0, and therefore $r_0(t) \geq 0$. We let $r(t) = \max(r_0(t), r_1(t)) \geq 0$ denote the time of the most recent move by either player.

Feedback during the game We distinguish various types of feedback that a player may obtain during the game (specifically upon moving).

It is one of the most interesting aspects of this game that the players do *not* automatically find out when the other player has last moved; moves are *stealthy*. A player must move himself to find out (and reassert control).

We let $\phi_i(t_k)$ denote the feedback player i obtains when the player moves at time t_k. This feedback may depend on which variant of the game is being played.

- **Nonadaptive [NA].** In this case, a player does not receive any useful feedback whatsoever when he moves; the feedback function is constant.

$$\phi_i(t_k) = 0$$

- **Last move [LM].** The player moving at time $t_k > 0$ finds out the exact time when the opponent played last before time t_k. That is, player i learns the value:

$$\phi_i(t_k) = r_{1-i}(t_k)$$

- **Full history [FH].** The mover finds out the complete history of moves made by both players so far:

$$\phi_i(t_k) = ((t_1, t_2, \ldots, t_k), (p_1, p_2, \ldots, p_k)) \,.$$

We abbreviate these forms of feedback as NA, LM, and FH, and we define other types of feedback in Section **??**. We consider "non-adaptive" (NA) feedback to be the default (standard) version of the game. When there is feedback and the game

is adaptive, then players interact in a meaningful way and therefore cooperative (e.g.,"tit-for-tat") strategies may become relevant.

Views and History A *view* is the history of the game from one player's viewpoint, from the beginning of the game up to time t. It lists every time that player moved, and the feedback received for that move.

For example, the view for player i at time t is the list:

$$\mathbf{v}_i(t) = ((t_{i,1}, \phi(t_{i,1})), (t_{i,2}, \phi(t_{i,2})), \ldots, (t_{i,j}, \phi(t_{i,j})))$$

where $t_{i,j}$ is the time of player i's j-th move, her last move up to time t, and $\phi(t_{i,j})$ is the feedback player i obtains when making her j-th move.

A *history* of a game is the pair of the players' views.

Strategies A *strategy* for playing this game is a (possibly randomized) mapping S from views to positive real numbers. If S is a strategy and v a view of length j, then $S(v)$ denotes the time for the player to wait before making move $j+1$, so that $t_{i,j+1} = t_{i,j} + S(v)$.

The next view for the player following strategy S will thus be

$$((t_{i,1}, \phi(t_{i,1})), (t_{i,2}, \phi(t_{i,2})), \ldots, (t_{i,j+1}, \phi(t_{i,j+1}))) \ .$$

We define now several classes of strategies, and we refer the reader to Figure **??** for a hierarchy of these classes.

Nonadaptive strategies: We say that a strategy is *nonadaptive* if it does not require feedback received during the game, and we denote by $\mathcal{N}$ the class of all nonadaptive strategies. A player with a nonadaptive strategy plays in the same manner against every opponent. A player with a nonadaptive strategy can in principle generate the time sequence for all of his moves in advance, since they don't depend on what the other player does. They may, however, depend on some independent source of randomness; nonadaptive strategies may be randomized.

Renewal strategies: Renewal strategies are non-adaptive strategies for which the intervals between consecutive moves are generated by a renewal process (as defined, for instance, by Feller [?]). Therefore, the inter-arrival times between moves are independent and identical distributed random variables chosen from a probability density function f. As the name suggests, these strategies are "renewed" after each move: the interval until the next move only depends on the current move time and not on previous history.

Periodic strategies: An example of a simple renewal strategy is a *periodic* strategy. We call a strategy *periodic* if there is a δ such that the player always presses his button again once exactly δ seconds have elapsed since his last button-push. We assume that the periodic strategy has a *random phase*, i.e., the first move is selected uniformly at random from interval $[0, \delta]$ (if the strategy is completely deterministic an adaptive opponent can find out the exact move times and schedule his moves accordingly).

Exponential strategies: We call a strategy *exponential* or *Poisson* if the player pushes his button in a Poisson manner: there is some rate λ such that in any short time increment Δ the probability of a button-push is approximately $\lambda \cdot \Delta$. In this case, $S(v)$ has an exponential distribution and this results in a particular instance of a renewal strategy.

Adaptive strategies: The class of adaptive strategies encompasses strategies in which players receive feedback during the game. In the LM class of strategies, denoted A_{LM}, a player receives last-move feedback, while in the FH-adaptive class, denoted A_{FH}, a player receives full history feedback, as previously defined.

No-play strategies: We denote by Φ the strategy of not playing at all (effectively dropping out of the game). We will show that this strategy is sometimes the best response for a player against an opponent playing extremely fast.

Information received before the game starts Besides receiving feedback during the game, sometimes players receive additional information about the opponent before the game starts. We capture this with $\phi_i(0)$, which denotes the information

received by player i before the game starts. There are several cases we consider:

- **Rate of Play [RP].** In this version of the game, player i finds out the limit of the rate of play $\alpha_{1-i}(t)$ of its opponent at the beginning of the game (assuming that the rate of play converges to a finite value):

$$\phi_i(0) = \lim_{t \to \infty} \alpha_{1-i}(t).$$

  No additional information, however, is revealed to the player about his opponent's moves during the game.

- **Knowledge of Strategy [KS].** Player i might find additional information about the opponent's strategy. For instance, if the opponent (player $1-i$) employs a renewal strategy generated by probability density function f, then KS information for player i is the exact distribution f:

$$\phi_i(0) = f.$$

  Knowledge of the renewal distribution in this case does not uniquely determine the moves of player $1-i$, as the randomness used by player $1-i$ is not divulged to player i. In this paper, we only use KS in conjunction with renewal strategies, but the concept can be generalized to other classes of strategies.

RP and KS are meaningfully applicable only to a non-adaptive attacker. An adaptive attacker from class LM or FH can estimate the rate of play of the defender, as well as the defender's strategy during the game from the information received when moving. But a non-adaptive attacker receiving RP or KS information before the game starts can adapt his strategy and base moves on pre-game knowledge about the opponent's strategy.

Pre-game information received at the beginning of the game, hence, could be used, in conjunction with the feedback received while moving, to determine the

strategy of playing the game. We can more formally extend the definition of views to encompass this additional amount of information as:

$$\mathbf{v}_i(t) = (\phi_i(0), (t_{i,1}, \phi(t_{i,1})), (t_{i,2}, \phi(t_{i,2})), \ldots, (t_{i,j}, \phi(t_{i,j})))$$

Gains and Benefits Players receive benefit equal to the number of time units for which they are the most recent mover, minus the cost of making their moves. We denote the cost of a move for player i by k_i; it is important for our modeling goals that these costs could be quite different.

Player i's total *gain* G_i in a given game (before subtracting off the cost of moves) is just the integral of C_i:

$$G_i(t) = \int_0^t C_i(x)\, dx$$

Thus $G_i(t)$ denotes the total amount of time that player i has owned the resource (controlled the game) from the start of the game up to time t, and

$$G_0(t) + G_1(t) = t\ .$$

The *average gain rate* for player i is defined as:

$$\gamma_i(t) = G_i(t)/t\ ;$$

so that $\gamma_i(t)$ is the fraction of time that player i has been in control of the game up to time t. Thus, for all $t > 0$:

$$\gamma_0(t) + \gamma_1(t) = 1\ .$$

We let $B_i(t)$ denote *player i's net benefit* up to time t; this is the gain (total possession time) minus the cost of player i's moves so far:

$$B_i(t) \ = \ G_i(t) - k_i n_i(t)\ .$$

We also call $B_i(t)$ the *score* of player i at time t. The maximum benefit or score

$B_0(t) = t - k_0$ for player 0 would be obtained if neither player moved again after player 0 took control at time $t = 0$.

We let $\beta_i(t)$ denote *player i's average benefit rate* up to time t:

$$\beta_i(t) = B_i(t) = \gamma_i(t) - k_i\alpha_i(t)$$

this is equal to the fraction of time the resource has been owned by player i, minus the cost rate for moving.

In a given game, we define player i's *asymptotic benefit rate* or simply *benefit* as

$$\beta_i = \liminf_{t\to\infty} \beta_i(t) \ .$$

We use $\liminf$ since $\beta_i(t)$ may not have limiting values as t increases to infinity.

While we have defined the benefit for a given game instance, it is useful to extend the notion over strategies. Let S_i be the strategy of player i, for $i \in \{0, 1\}$. Then the benefit of player i for the `FlipIt` game given by strategies (S_0, S_1) (denoted $\beta_i(S_0, S_1)$) is defined as the expectation of the benefit achieved in a game instance given by strategies (S_0, S_1) (the expectation is taken over the coin flips of S_0 and S_1).

Game-theoretic definitions We denote by $FlipIt(\mathcal{C}_0, \mathcal{C}_1)$ the `FlipIt` game in which player i chooses a strategy from class $\mathcal{C}_i$, for $i \in \{0, 1\}$. For a particular choice of strategies $S_0 \in \mathcal{C}_0$ and $S_1 \in \mathcal{C}_1$, the benefit of player i is defined as above. In our game, benefits are equivalent to the notion of *utility* used in game theory.

Using terminology from the game theory literature:

- A strategy $S_0 \in \mathcal{C}_0$ is *strongly dominated* for player 0 in game $FlipIt(\mathcal{C}_0, \mathcal{C}_1)$ if there exists another strategy $S_0' \in \mathcal{C}_0$ such that:

$$\beta_0(S_0, S_1) < \beta_0(S_0', S_1), \forall S_1 \in \mathcal{C}_1.$$

- A strategy $S_0 \in \mathcal{C}_0$ is *weakly dominated* for player 0 in game $FlipIt(\mathcal{C}_0, \mathcal{C}_1)$ if

there exists another strategy $S_0' \in \mathcal{C}_0$ such that:

$$\beta_0(S_0, S_1) \leq \beta_0(S_0', S_1), \forall S_1 \in \mathcal{C}_1,$$

with at least one S_1 for which the inequality is strict.

- A strategy S_0 is *strongly dominant* for player 0 in game $FlipIt(\mathcal{C}_0, \mathcal{C}_1)$ if:

$$\beta_0(S_0, S_1) > \beta_0(S_0', S_1), \forall S_0' \in \mathcal{C}_0, \forall S_1 \in \mathcal{C}_1.$$

- A strategy S_0 is *weakly dominant* for player 0 in game $FlipIt(\mathcal{C}_0, \mathcal{C}_1)$ if:

$$\beta_0(S_0, S_1) \geq \beta_0(S_0', S_1), \forall S_0' \in \mathcal{C}_0, \forall S_1 \in \mathcal{C}_1.$$

Similar definitions can be given for player 1 since the game is fully symmetric.

There is an implicit assumption in the game theory literature 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 [**?**]). We denote by $FlipIt^*(\mathcal{C}_0, \mathcal{C}_1)$ 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.

A *Nash equilibrium* for the game $FlipIt(\mathcal{C}_0, \mathcal{C}_1)$ is a pair of strategies $(S_0, S_1) \in \mathcal{C}_0 \times \mathcal{C}_1$ such that:

$$\beta_0(S_0, S_1) \geq \beta_0(S_0', S_1), \forall S_0' \in \mathcal{C}_0;$$
$$\beta_1(S_0, S_1) \geq \beta_1(S_0, S_1'), \forall S_1' \in \mathcal{C}_1.$$

Chapter 3

Biperiodic Defender

In this case, there are two pre-specified periods for the defender τ_a and τ_b and he chooses to play τ_a with probability p and τ_b with probability $1-p$. We call this strategy BP, for Biperiodic. When the attacker is non-adaptive, the analysis of this game is very similar to the one for simple periodic strategies. In fact, the results we get for the Biperiodic defender are similar to a periodic defender, who is playing with period $\delta_0 = p\tau_a + (1-p)\tau_b$, since we are analyzing average (i.e. expected) performance. Since biperiodic strategy is a renewal strategy, then by Theorem 4 of the original `FlipIt` paper, we should expect that the periodic strategy performs better. This is indeed the conclusion from sections 3.1, 3.2 and 3.3. However, we still went on to carry out some experiments in order to get a better intuition of the strategy. Thus, sections 3.1, 3.2 and 3.3 are preliminary results, before we address the main question: what should a defender do against an adaptive attacker?

3.1 Non-adaptive, no information attacker

This section is based on the analysis described in section 4.1 of the original FlipIt paper[1] (a non-adaptive continuous game, in which both players employ a periodic strategy with a random phase). The parameters in that case are the defender's period $\delta_0 = \frac{1}{\alpha_0}$ and cost k_0 and the attacker plays with period δ_1 and cost k_1. For the case where the defender plays at least as fast as the attacker ($\delta_0 \leq \delta_1$), the

benefits are:

$$\beta_0(\delta_0, \delta_1) = 1 - \frac{\delta_0}{2\delta_1} - \frac{k_0}{\delta_0}$$

$$\beta_1(\delta_0, \delta_1) = \frac{\delta_0}{2\delta_1} - \frac{k_1}{\delta_1}$$

The symmetric case where the defender plays no faster than the attacker has a similar analysis.

The difference now is that the defender does not play with a specific period δ_0, but he is probabilistic – biperiodic. All other parameters are the same. We notice that the expected cost, C, of the defender in this case is given by $k_0 \cdot \alpha$, where $\alpha = \frac{1}{p\tau_a + (1-p)\tau_b}$. Thus, as mentioned before the most relevant case – to compare against – is a simple periodic strategy, where the defender's period is $\delta_0 = p\tau_a + (1-p)\tau_b$. In both cases, the expected cost is the same. What is, thus, left to compare is the gain of the defender in each situation and the one with the larger gain will have a larger benefit.

In this section, we assume that no one has information about the other player. We merely want to analyze what the expected benefits of this strategy are. This gives us a better intuition to address the following sections. Most of the work done in this section is experimental. Before we dive into simulations and experimental results, we analyze the situation and come up with the hypothesis that the defender is better off playing a simple periodic strategy with period $\delta_0 = p\tau_a + (1-p)\tau_b$ instead of playing τ_a with probability p and τ_b with probability $1-p$, given that the attacker is non-adaptive. There could be some quantization (or edge) effects due to the short time allowed, but it is enough to show the intuition behind our experimental analysis that follows.

Consider the following example for a total time of 200 time units, depicted in Figure 3-1, in which the attacker plays with period $\delta_1 = 100$:

Strategy 1: Defender plays with period $\delta_0 = 70$.

Strategy 2: Defender plays $\tau_a = 49$ with probability $p = 0.5$ and $\tau_b = 91$ with probability $1 - p = 0.5$. Notice that $p\tau_a + (1-p)\tau_b = 70$.

We calculate for both strategies the total time that the attacker has control over the

source and the defender should try and minimize that number.

Strategy 1: The defender starts at time $t = 0$ and takes over at time $t = 70$, which is before the attacker takes over and nothing changes, at $t = 140$ takes the source back from the attacker who took over at $t = 100$, and at $t = 210$, which is outside of the range we are considering. Thus, deterministically, the attacker controls the source for only **40** time units out of a total of 200.

Strategy 2: This strategy has multiple cases and, thus, we can only report the expected time that the attacker has control over the source. We consider the following cases of Defender moves, because they are the only ones that can happen in 200 time units. Moreover, since $p = 0.5$, all cases are equally likely:

1. τ_a, τ_a, τ_a

   At $t = 49$ and $t = 98$, nothing changes because the defender was in control already. Then, the attacker takes over at $t = 100$ and the defender takes it back at $t = 147$. Now, it does not matter what the defender does anymore since he will have control until $t = 200$. Thus, the attacker controls the source for **47** time units.

2. τ_a, τ_a, τ_b

   At $t = 49$ and $t = 98$, nothing changes because the defender was in control already. Then, the attacker takes over at $t = 100$ and the defender takes it back at $t = 189$. Now, it does not matter what the defender does anymore since he will have control until $t = 200$. Thus, the attacker controls the source for **89** time units.

3. τ_a, τ_b, τ_a

   Notice that $\tau_a + \tau_b = 2 \cdot 70$ and, thus, this one and the following three cases give us the same number as *Strategy 1*, which is **40** time units.

4. τ_a, τ_b, τ_b

   40 time units

5. τ_b, τ_a, τ_a

   40 time units

6. τ_b, τ_a, τ_b

   40 time units

7. τ_b, τ_b, τ_a

   At $t = 91$, nothing changes because the defender was in control already. Then, the attacker takes over at $t = 100$ and the defender takes it back at $t = 182$. Now, it does not matter what the defender does anymore since he will have control until $t = 200$. Thus, the attacker controls the source for **82** time units.

8. τ_b, τ_b, τ_b

   Same case as the one above and, thus, **82** time units.

All of the aforementioned cases are equally likely and, thus, we get an expected number of $\frac{1}{8}(47 + 89 + 42 + 42 + 42 + 42 + 82 + 82) = 58.5$ time units, which is more than the simple periodic strategy.

We analyzed various similar biperiodic strategies by varying times τ_a and τ_b and varying the probability, p. In most cases, the periodic strategy of the defender is preferred over the biperiodic one. The results are summarized in Tables 3.1, 3.2 and 3.3. In the case, where both τ_a and τ_b are less than δ_1 (Table 3.1), which also means that the expected period of the defender is less than the attacker's, the periodic strategy is always better. Moreover, the closer both τ_a and τ_b are to the period of the periodic strategy the better the gain for the defender. Therefore, it is better to use probabilities that are closer to $p = 0.5$.

In the case, where $\tau_a < \delta_1$ and $\tau_b > \delta_1$ (Table 3.2), in such a way that the expected period of the defender is less than the attacker's, we observe two things. The first one is that if the expected period of the defender is significantly smaller than the attacker's then, we still prefer the periodic strategy. The second one is that as τ_a and

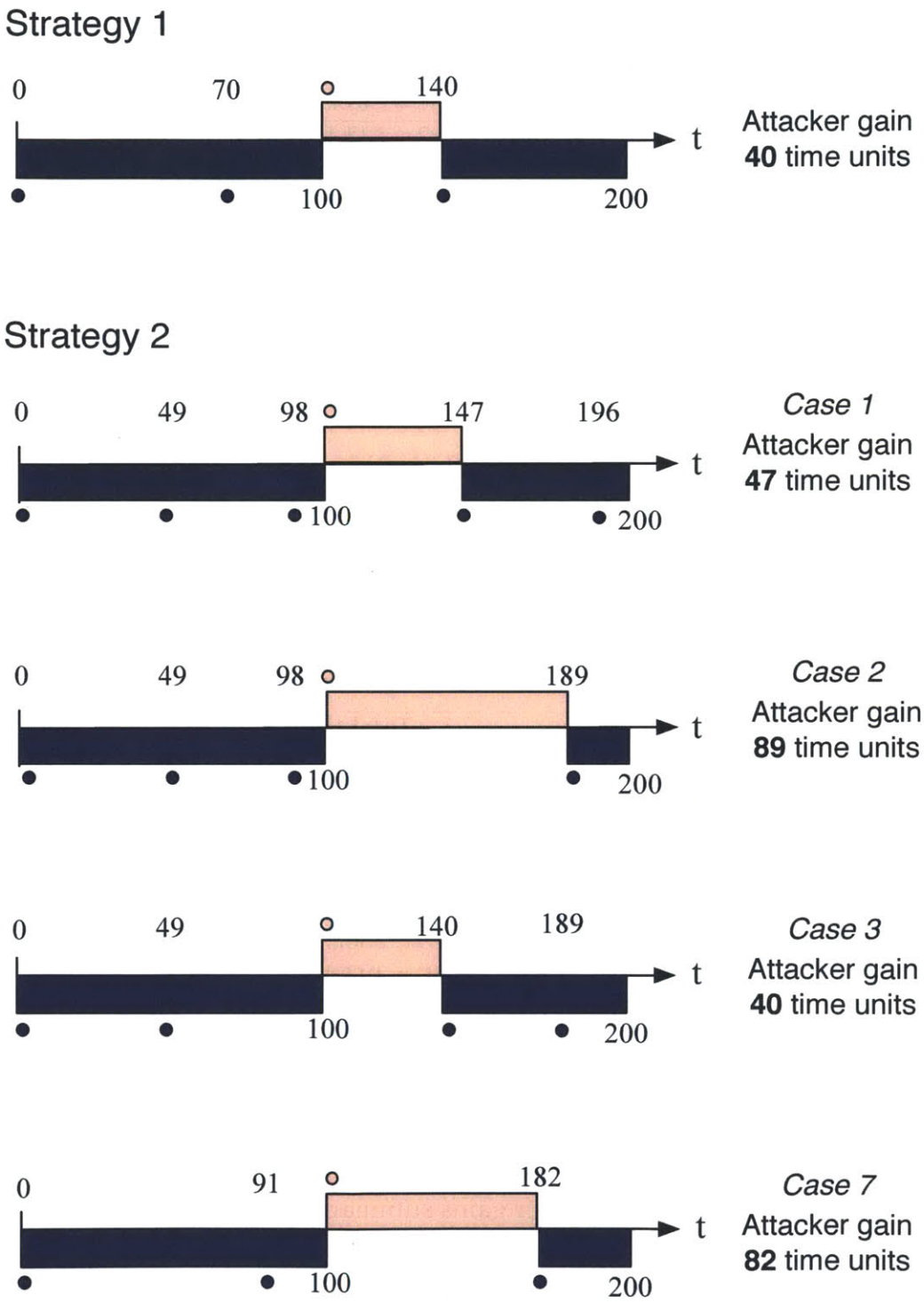

Figure 3-1: Comparing periodic and biperiodic strategies.

Parameters			Total time				
τ_a	τ_b	p	300	500	700	900	1100
Fixed: 70			50	120	210	250	310
65	77.5	0.6	65.7	142.1	215.2	274.6	348.5
81	59	0.5	85	157.2	222.4	284.1	358.23
80	55	0.6	78.4	150.7	224.3	285.5	360.5
80	35	0.7	92.7	156	234.6	302.4	376.9
91	49	0.5	106.9	184.8	239.8	316.6	402.6

Table 3.1: Attacker plays with $\delta_1 = 100$. Both $\tau_a, \tau_b < \delta_1$ Attacker gains summarized.

τ_b are symmetrically moving away from the optimal period of the periodic strategy, call it δ_0^*, then the biperiodic strategy performs better up to a certain point and the benefit of the defender is comparable to the benefit of the periodic defender. In fact, for specific values of τ_a and τ_b, the biperiodic strategy is preferred, however if τ_a and τ_b are too far away from δ_0^*, then the simple periodic strategy is preferred.

Parameters			Total time				
τ_a	τ_b	p	300	500	700	900	1100
Fixed: 70			50	120	210	250	310
30	110	0.5	54.1	138.8	221	300.4	378.6
Fixed: 90			150	260	330	360	440
79	101	0.5	73.7	173.8	271.7	367.3	438
70	110	0.5	77.4	144.9	236.5	314	400.6
60	120	0.5	62	139	197.4	277	359.1
55	125	0.5	66	172	274	370	470
Fixed: 95			175	330	465	580	675
85	105	0.5	94	185	312.6	402	512
75	115	0.5	67.7	177.8	280	355.6	455.2

Table 3.2: Attacker plays with $\delta_1 = 100$. $\tau_a < \delta_1$ and $\tau_b > \delta_1$ Attacker gains summarized.

Finally, in the case where the expected period based on τ_a and τ_b is greater than the period of the attacker (Table 3.3), then the periodic strategy performs better. Both when $\tau_a, \tau_b > \delta_1$ and when τ_a or τ_b is less than δ_1. If both $\tau_a, \tau_b > \delta_1$, then the closer the τ_a and τ_b are to their expected period or if they are too far away, the worse the biperiodic performance, but still worse than the periodic.

Parameters			Total time				
τ_a	τ_b	p	300	500	700	900	1100
Fixed: 105			15	50	105	180	275
95	115	0.5	71.7	134.5	220	306	412
90	120	0.5	74.1	150	240	342	437.8
Fixed: 115			45	150	315	320	375
95	135	0.5	103.7	207	313	435.8	543
85	145	0.5	108.3	210	328.7	442	555.4
110	120	0.5	45	150	273.6	357	461
Fixed: 120			60	200	200	260	400
110	130	0.5	60	168	281.5	390.1	489
116	124	0.5	60	200	301.2	411.6	542.3
103	138	0.5	61.5	171	295	422	536
Fixed: 140			120	140	200	240	320
135	145	0.5	120	240	336.5	490	574

Table 3.3: Attacker plays with $\delta_1 = 100$. Both $\tau_a, \tau_b > \delta_1$ or expected period $> \delta_1$ Attacker gains summarized.

With these observations in mind, we proceed in large scale simulations of the biperiodic strategy. The code for the simulations is found in Appendix A along with the relevant, summarized data. We arbitrarily choose the costs of the defender and the attacker (k_0 and k_1, respectively) to be 10. We only care about relative performance in the simulations of this section and, thus, the actual numbers used are not affecting anything. Also, after a lot of tests we chose the number of attacker moves to be 5000. This number determines the total time that the simulation will run (i.e. if $\delta_1 = 10$ then the total simulation will run for 50000 time units). It is long enough to give us a good understanding of the various strategies without being unreasonably long. The simulation happens in the function `pr_periodic()` and the rest of the code in the appendix is simply preparing the input and using the output of that function.

First, we consider a pair of times τ_a and τ_b one at a time and choose them with varying probabilities. For every probability, we calculate the expected period and compare the theoretical value of the periodic strategy with the experimental value of the biperiodic strategy. Notice that the experimental value of the periodic case usually varies from the theoretical, but their difference is insignificant for most

cases. These experiments give us intuition on how different probabilities affect the strategy. In addition, they tell us how the values τ_a and τ_b affect the strategy relative to the attacker's period (*Table A1*) .

The second set of experiments is based on the same code found in Appendix A, but now we fix the expected period and symmetrically broaden the difference between τ_a and τ_b to observe how these values affect the biperiodic strategy relative to the periodic strategy (*Table A2*).

The main observations are:

- The performance of the biperiodic strategy with parameters τ_a, τ_b and p is similar to the periodic strategy with $\delta_0 = p\tau_a + (1-p)\tau_b$.

- If τ_a and τ_b are such that the the expected period δ_0 is significantly lower than the attacker's period δ_1, then the Defender usually prefers to play the periodic strategy with the expected period δ_0 (See *Table A1*, entry $\delta_1 = 50, \tau_a = 16$ and $\tau_b = 29$). Even though playing τ_a and τ_b with values very close to δ_0 is also possible. It does not matter whether both $\tau_a, \tau_b < \delta_1$ or $\tau_a < \delta_1$ and $\tau_b > \delta_1$. Moreover, it is observed (*Table A2*) that the farther away τ_a is from τ_b the worse the biperiodic strategy performs.

- If the Defender affords to play faster than the Attacker, then that is what he should do. In the case where this is not possible and the expected δ_0 – given from τ_a and τ_b – is significantly greater than δ_1, then we still observe (*Table A2*) that the farther away τ_a is from τ_b the worse the biperiodic strategy performs. Also, the periodic strategy is again preferred by the defender.

- Even if the expected period δ_0 is close to δ_1, if τ_a and τ_b are far away from each other, the Defender prefers the periodic strategy. (See *Table A1*, entry $\delta_1 = 50, \tau_a = 11$ and $\tau_b = 51$ and the entry $\delta_1 = 50, \tau_a = 11$ and $\tau_b = 71$)

- We notice in *Table A2* that as we symmetrically expand τ_a and τ_b about δ_0, we get an improvement in the Defender's performance, but after a certain

point, when τ_a and τ_b are far away, the biperiodic strategy is worse than the periodic.

The main conclusion is, then, that except some very specific cases, in which we can obtain a small improvement in the Defender's gain by deploying a biperiodic strategy, we are better off deploying a periodic one, with random phase always.

3.2 Non-adaptive, Known costs attacker

So far, we have examined the case in which no player has information about the other player. We just observed how the gains and the benefits of each player are depended on the variables δ_1, τ_a, τ_b and p.

In this section, we deal with the case, in which both players have information about each other's costs. In particular, the defender knows that the attacker deploys a periodic strategy and his cost per move is k_1. The attacker knows that the defender deploys a biperiodic strategy and his cost per move is k_0. They do not receive any updates during the game, therefore it is still non-adaptive.

The relative costs need to be taken into account, when choosing a strategy. In fact, we need to consider game theoretical notions like Nash equilibria and iterative rationalizability; given what the other player wants to do, what is best for me? Moreover, the other player knows what is best for me given what he is doing and he needs to incorporate this in his decision making process. We use the results derived in the FlipIt[1] paper and see how they apply in this specific case. The Nash equilibria points for which neither player will increase his benefit by changing his rate of play, in the case where both players play with a periodic strategy, are given by **Theorem 1**[1]:

$$k_0 < k_1 \quad \delta_0^* = 2k_1, \delta_1^* = \frac{2k_1^2}{k_0}$$

$$k_0 = k_1 \quad \delta_0^* = \delta_1^* = 2k_0$$

$$k_0 > k_1 \quad \delta_0^* = \frac{2k_0^2}{k_1}, \delta_1^* = 2k_0$$

The main result is that if you have low costs relative to your opponent you can have a significant gain, since you can afford to try and take over the source frequently. Thus, the opponent will end up having a very small gain and probably negative benefit. In fact, if the defender plays with period $\delta_0 < 2k_1$, the attacker should drop out of the game to avoid negative benefits. However, playing too fast leads to decreased benefits; therefore, even if you have low costs you should not play arbitrarily fast. Thus, both players should play with the periods mentioned above in order to achieve the maximum possible benefit.

Based on the results of the previous chapter, the biperiodic defender should play with τ_a and τ_b such that the expected period of the biperiodic strategy $\delta_0 = p\tau_a + (1-p)\tau_b$ is close to the Nash equilibrium. If $\delta_0^* >> \delta_1^*$ or $\delta_0^* << \delta_1^*$, then the Defender should play with $\tau_a = \tau_b = \delta_0^*$. However, if δ_0^* is close to δ_1^*, then we know that for $p = 0.5$ and for some τ_a and τ_b symmetrically away from δ_0^*, but not too far, the defender could get a slightly improved benefit. Therefore, we should start at $\tau_a = \tau_b = \delta_0^*$ and symmetrically expand τ_a and τ_b (i.e. decrease τ_a by the same amount that we increase τ_b). Eventually we will hit the point for which the defender's gain and benefit will be maximized. After that point the benefit will only decrease. Since that point is different depending on the specific values of δ_1 and δ_0, we should run multiple simulations in order to identify it. The code for the simulations is depicted in Appendix B along with the relevant results that support this statement (see *Table B1*). Again, most of the time, the defender should deploy a periodic strategy.

Additionally, notice that the last observation of the previous section, based on the τ_a and τ_b having a small least common multiple is very unlikely to occur, when the defender starts with random phase uniformly chosen from the interval $[0, \delta_0^*]$. The reason is that in continuous situations the probability of trying to take over the source at the exact same time is zero. This is why, for example, instead of using $\delta_0^* = 20$, we use $\delta_0^* = 19.99$; to avoid cases of exact matching.

3.3 Non-adaptive, RP Attacker

In this case, the attacker has full information of the defender's strategy. He knows the period δ_0 and the cost k_0. The only thing he does not know is the initial random phase chosen by the defender. No additional information is received after the game starts. The defender only knows the attacker's cost, k_1. However, the optimal periods for both players can be calculated based on ***Theorem 5***,[1] as follows:

- $k_1 < (4 - \sqrt{12})k_0$

$$\delta_0^{opt} = \frac{8k_0^2}{k_1}, \quad \delta_1^{opt} = 4k_0$$

- $k_1 \geq (4 - \sqrt{12})k_0$

$$\delta_0^{opt} = 2k_1, \quad \delta_1^{opt} = 0$$

Again, as in the previous section where the costs are known, the periods of the periodic strategies are based on the costs of the players. Therefore, as before, we use the same approach. In particular, the defender chooses τ_a and τ_b with probability $p = 0.5$ symmetrically about the δ_0 (as specified by Theorem 5). The simulation for this is very similar to the one from Section 1.2 and the code can be found in Appendix B. The results are summarized in Appendix C.

From the results shown in Table C1, we observe that most of the time, the optimal benefit is achieved, when τ_a and τ_b are some small distance away from the optimal δ_0, which is found after running the simulation. Moreover, we see that when δ_0 is slightly larger than δ_1, the defender's benefit is significantly higher. Thus, even though it is a good approach to choose the random phase uniformly from the interval $[0, \delta_0]$, it is better if we assign slightly higher probability to smaller values in that interval. In this way, the attacker will still not have enough information and he will not want to change his strategy. Notice that when $\delta_0 < \delta_0^{opt}$, the optimal benefit given by Theorem 5 is an upper bound; whereas, when $\delta_0 > \delta_0^{opt}$ this bound does not hold anymore.

3.4 LM attacker

The periodic strategy of the defender does not make sense anymore, because the attacker receives feedback of the defender's move. In fact, LM means that the attacker knows when the defender made his last move. As observed in the FlipIt paper, when the defender is playing with a renewal strategy, like the one stated in this section, an LM attacker is as powerful as an FH attacker. FH attacker is the attacker that at every move he receives Full History (full knowledge) of what happened so far in the game. Thus, if the LM attacker stores the information he receives at every step, the two attackers are equivalent. Thus, the attacker can always play right after the defender and have complete control of the source. In such a situation, the defender has zero gain and prefers to not play at all. If the defender deploys a biperiodic strategy, he can increase his gain.

It will take a few steps for the attacker to find out what τ_a and τ_b are and a few more to figure out what the probability, p, of selecting τ_a is. But, in the long run, this does not affect the result; thus, we assume that these are known from the beginning for simplicity in our simulations and our analysis.

Let us assume, w.l.o.g, that $\tau_a < \tau_b$, for the rest of this section. There are only three possible strategies for an LM attacker.

1. Move right after τ_a time has passed, since the defender's last move.

2. Move right after τ_b time has passed, since the defender's last move.

3. Move right after τ_a time has passed, since the defender's last move, and if the defender hasn't move yet play again, right after τ_b time has passed.

This is a significant improvement for the defender compared to the periodic strategy. In cases (1) and (2), the defender can have positive gain and even positive benefit depending on τ_a and τ_b. In case (3), the attacker plays more times and, thus, it will be easier for the defender to play fast enough to kick the attacker out of the game.

Given k_0 and k_1, the defender needs to select τ_a, τ_b and p to maximize his benefit. Thus, in our simulations, for a fixed pair of costs, we try a range of periods and evaluate the three attacker strategies. In essence, the defender is trying to minimize the attacker's maximum possible gain.

As we know already, the player with the lower cost has an advantage. In fact, the defender should play only when his cost is significantly lower than the attacker's, otherwise the attacker affords to pay the cost and have complete control over the source. We have not quantified "significantly lower," but, from our experimental observations, it should be an order of magnitude less than the attacker's cost.

3.4.1 Negative Benefit for Attacker

A useful result from the analysis in Section 5.2 of the FlipIt paper[1] is that if the defender plays fast enough, then the attacker ends up with negative benefit and, thus, he would probably drop out. Notice that in the original `FlipIt` paper, no negative benefits are allowed. However, we present it here as a possibility, since this could be a real life scenario. For example, someone may be paying the attacker in order to damage the defender. Observe that the attacker does not choose a period, but he follows the defender. In the simple periodic case, the attacker has full control of the source and, thus, his gain is 1. In addition, his period is exactly the same as the defender and, thus, $\delta_1 = \delta_0$. His benefit is given by:

$$\beta_1 = 1 - \frac{k_1}{\delta_0}$$

which is negative for $\delta_0 < k_1$.

In cases (1) and (2), the period with which the attacker is playing is a linear combination of τ_a and τ_b such that $\tau_a < \delta_1 < \tau_b$. In fact, as expected, $\delta_1 = p\tau_a + (1 - p)\tau_b$. However, now the gain is not always 1; it depends on the choice of τ_a, τ_b and p. It also depends on whether, it is case (1) or (2). We do not attempt to calculate it theoretically, but we see how the gain is affected by those parameters in our

experimental analysis, later on in this section. The important observation, though, is that:

- If $\tau_a, \tau_b < k_1$, then the attacker's benefit is negative.
- If $\tau_a, \tau_b > k_1$, then the attacker's benefit is positive.
- If $\tau_a < k_1$ and $\tau_b > k_1$, then if τ_b is not "too far" away from k_1, then the attacker's benefit is negative.

Finally, for case (3), the attacker has complete control of the source; thus, a gain of 1. However, he plays more frequently now compared to the periodic case and his cost will be higher. In fact, the attacker's period now, becomes:

$$\delta_1 = p\tau_a + (1-p)\frac{\tau_b}{2}$$

which is less than the period in the other two cases. Similar to the periodic case, the attacker's benefit is negative for $\delta_1 < k_1$.

Having analyzed all the three different cases, we now know how to choose τ_a, τ_b and p, to guarantee a negative benefit for the attacker for all three cases. We should choose the aforementioned in such a way to minimize the defender's cost.

(i) If the attacker cannot afford negative benefit, then he simply drops out. The defender's gain will be 1.

(ii) If the attacker can afford negative benefit, he will play if the defender's benefit is even more negative.

3.4.2 Experimental Analysis

In this section, we modify the three variables τ_a, τ_b and p and observe how they affect the attacker's gain and benefit. The defender aims to find the optimal values of these three variables based on a fixed pair of player costs (k_0 and k_1). For this specific case and based on the observations of section 1.4.1, we chose $k_0 = 1$ and

$k_1 = 10$. Notice that these specific values do not affect the analysis in any way, since the results we obtain are relative.

There are four main experiments that we consider here. The first one is fixing the probability to $p = 0.5$ and keeping the times τ_a and τ_b close to each other. Then, we simply try various pairs (τ_a, τ_b) and see how the three attacker strategies behave. The results are summarized in Figure 3-2. "Expected delay" is obtained as follows: If $\tau_a = 9$ and $\tau_b = 11$, then by taking their average, we get the expected delay to be 10. In our experiment the values of τ_a and τ_b differ by exactly 2.

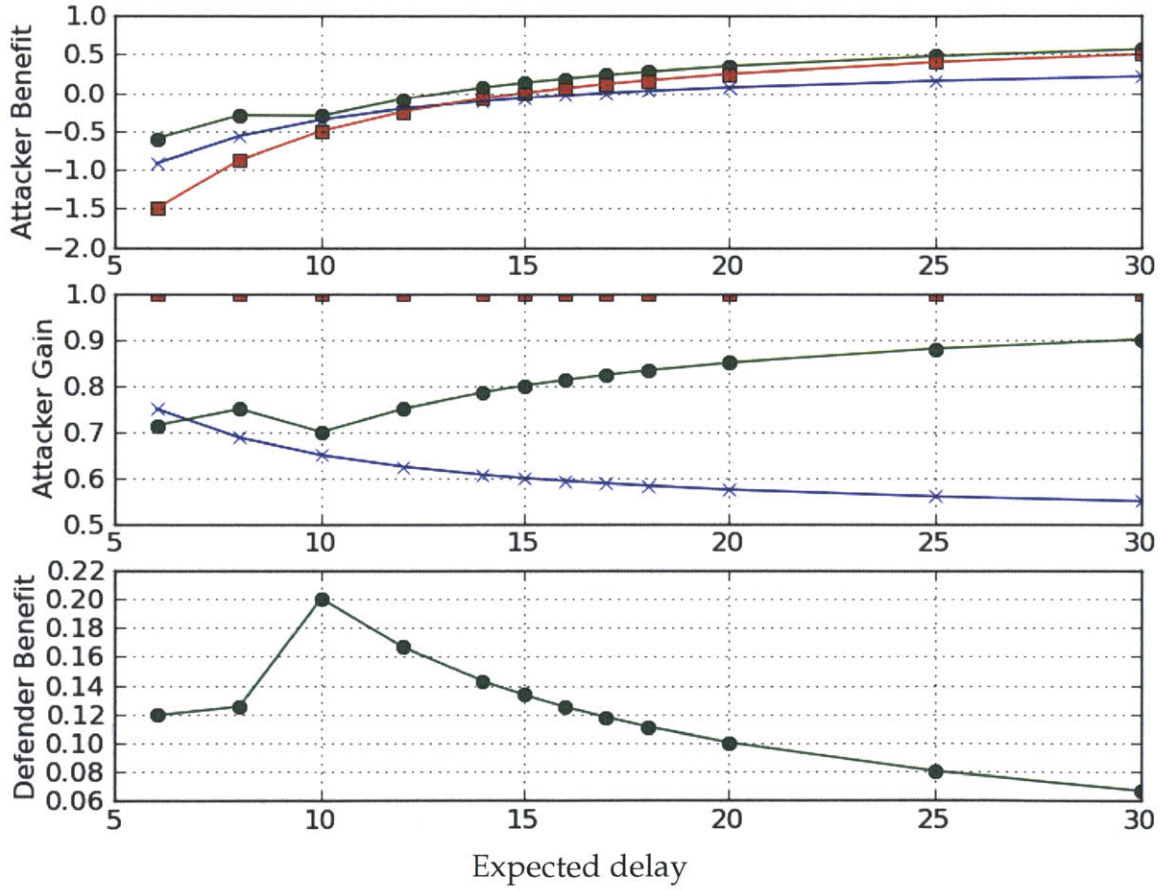

Figure 3-2: Experiment 1: τ_a and τ_b close to each other and fixed $p = 0.5$
Strategy 1: Cross (Blue), Strategy 2: Circle (Green), Strategy 3: Square (Red)

Experiment 1 observations:

1. The main observation is that the best strategy for the attacker is to deploy Strategy 2; i.e. play immediately after τ_b time has passed since the defender's

last move.

Even though, Strategy 3 has a gain of 1, the costs of that strategy are high. In this, case Strategy 2 has a high enough gain and significantly lower costs.

2. As the magnitude of the two periods is increasing Strategy 3 is asymptotically approaching Strategy 2.

   The reason is that the rate of play is decreased and since the gain is unchanged, the attacker obtains a higher benefit.

3. As the magnitude of the two periods is increasing the gain and the benefit of Strategy 2 are increasing.

   The gain is increased, because even though the difference $\tau_b - \tau_a$ is the same, the rate of play is decreased. Thus, the defender is trying to take over the source less frequently. The fact that the attacker is playing immediately after τ_b guarantees the attacker the control of the source for that period of time when nobody is moving.

4. The defender's benefit increases at the beginning and, then, it is consistently decreasing.

Based on these observations the defender should choose small τ_a and τ_b, but not too small because then his costs are higher. They should be small enough. A more quantitative approach is given at the end of this section. This happens only for small values of the defender's cost k_0, otherwise the defender's benefit is negative and, thus, it keeps increasing as we increase τ_a and τ_b and it asymptotically reaches zero.

The questions that naturally occur after the first experiment are: What if we vary the probability, p? (Experiment 2) What if τ_a and τ_b are far away from each other? (Experiment 3) What if we do both? (Experiment 4)

In the second experiment, we fix the two periods τ_a and τ_b and vary the probability p. The dominant strategy of the attacker in Experiment 1 is strategy 2. We observe that as we decrease $p < 0.5$, we are making the event that the defender

will play τ_b, even more probable and, thus, strategy 2 becomes even better for the attacker. Therefore, in experiment 2 we want to increase p and observe what happens. We get different results based on the τ_a and τ_b that we select at the beginning and we distinguish them in three different categories.

(i) $\tau_a, \tau_b > k_1$

In this case, as we have seen in Experiment 1, the attacker has an advantage and, even if we modify p, the attacker is still better off and the defender's benefit is very low.

(ii) $\tau_a, \tau_b < k_1$

In this case, Strategy 2 of the attacker is the dominant one, for any p. In addition, we already observed that as p increases, Strategy 2 is performing worse. Thus, we can choose a pair τ_a and τ_b and increase p. However, if p is very close to one, then the biperiodic strategy becomes periodic and, thus, we should keep p low enough to avoid this.

Notice that if τ_a and τ_b are too close together, then the results are similar to the next category, category (iii). This observation is what leads to experiments 3 and 4, since the distance between τ_a and τ_b matters.

(iii) $\tau_a < k$ and $\tau_b > k$, with τ_a and τ_b reasonably close.

In this category, Strategy 3 of the attacker is the worst one and the most important observation is that Strategy 1 and Strategy 2 are crossing over at some probability, p.

Experiment 2 observations for category (iii):

1. At lower values of p, Strategy 2 gives the attacker the highest benefit and gain and, therefore, a low benefit for the defender.

2. At high values of p, Strategy 1 gives the attacker a higher benefit and, again a low benefit for the defender.

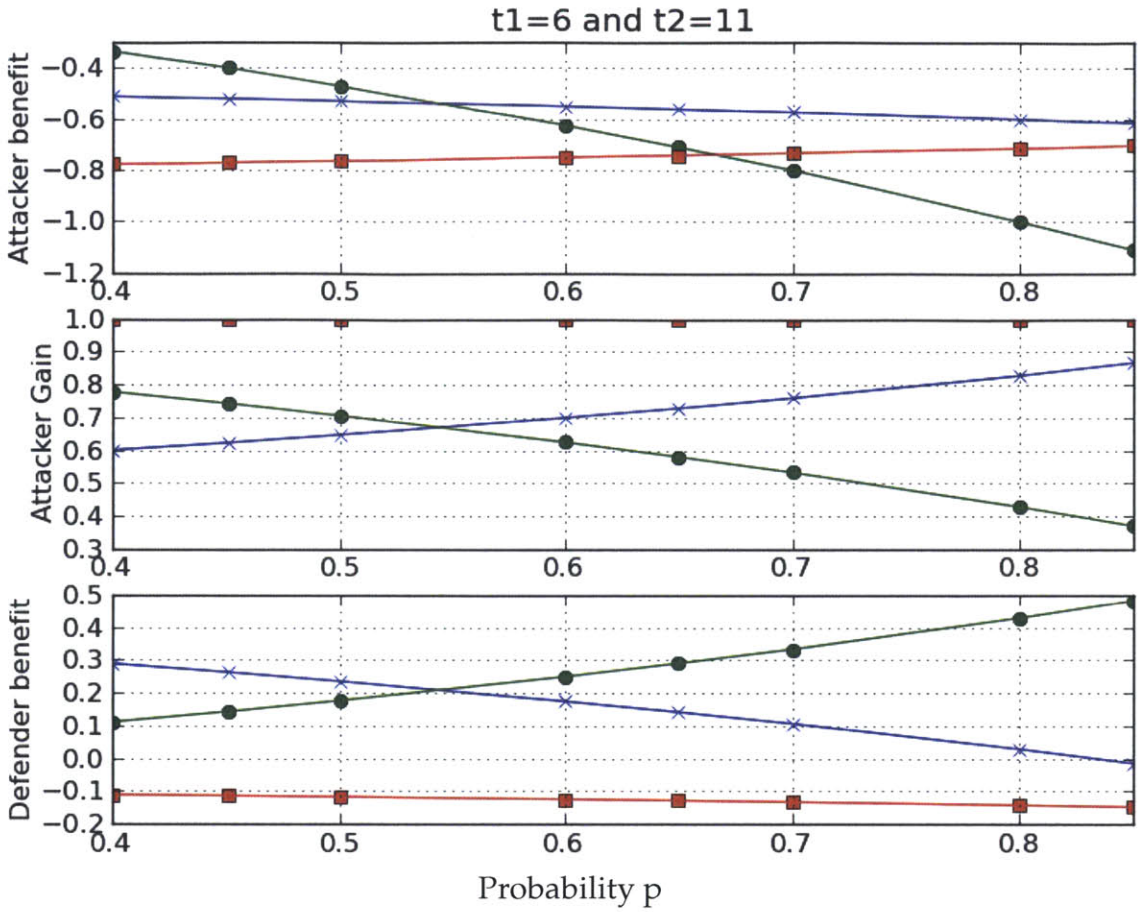

Figure 3-3: Experiment 2: $\tau_a = 6, \tau_b = 11$. The cross-over point is at $p = 0.55$ Strategy 1: Cross (Blue), Strategy 2: Circle (Green), Strategy 3: Square (Red)

3. The point of interest is the exact p for which the two Strategies are crossing over. As you can see in Figure 3-3, that is the point at which the highest defender benefit is achieved. In this figure, this point is at $p = 0.55$. In general, this point is determined from the graph and it depends on the values of τ_a and τ_b.

4. The value of the benefit depends mostly on the difference between the two periods, as shown in the following experiments. Here, we are considering equidistant pairs (τ_a, τ_b).

5. When the equidistant pair is low valued the cross-over probability p_{low} is less than p_{high}, the cross-over probability when the values are high. For example, in Figure 3-3, where $\tau_a = 6$ and $\tau_b = 11$, $p_{low} = 0.55$. When $\tau_a = 9$ and $\tau_b = 14$,

$p_{high} = 0.65$.

In the third experiment, we keep the probability constant again and we vary the distance between τ_a and τ_b. Even though, there are no significant improvements in the maximum defender's benefit we have observed so far (around 0.3), we still have some useful observations that will help us design Experiment 4. This experiment proceeded in two phases: (1) we fix τ_b and vary τ_a and we repeat this for different $\tau_b s$ and (2) we fix τ_a and vary τ_b and, again, we repeat this for different $\tau_a s$.

Experiment 3 observations:

1. As we keep increasing the distance between τ_a and τ_b, we get some improvement, but only up to a specific (and not too large) distance.

2. As we decrease τ_a, we get (see Figure 3-4):

   - Strategy 1: The attacker gain increases and the benefit increases. However, if τ_b is low enough, then the attacker benefit decreases as τ_a decreases.
   - Strategy 2: The attacker gain decreases and the benefit decreases.
   - Strategy 3: The attacker gain is still 1, since the attacker has complete control, and the benefit decreases a lot.
   - Increasing τ_b above a certain value leads to dramatic decrease in the defender's benefit and we stop there.

3. As we increase τ_b, we get (see Figure 3-5):

   - Strategy 1: The attacker gain increases and the benefit increases, as well.
   - Strategy 2: The attacker gain decreases, but the benefit increases.
   - Strategy 3: The attacker gain is still 1 and the benefit increases.
   - We observe a crossover from Strategy 2 being dominant to Strategy 1 being better, while both being better than Strategy 3. When this crossover happens, the defender's benefit will now be significantly smaller.

- As we increase τ_b even further, another crossover happens and Strategy 3, becomes better than Strategy 2. Strategy 1 is still dominant.
- Increasing τ_a above a certain value leads to dramatic decrease in the defender's benefit and we stop there.

4. We obtain similar maximum benefits for different (τ_a, τ_b) pairs, but almost every time they are obtained from Strategy 2.

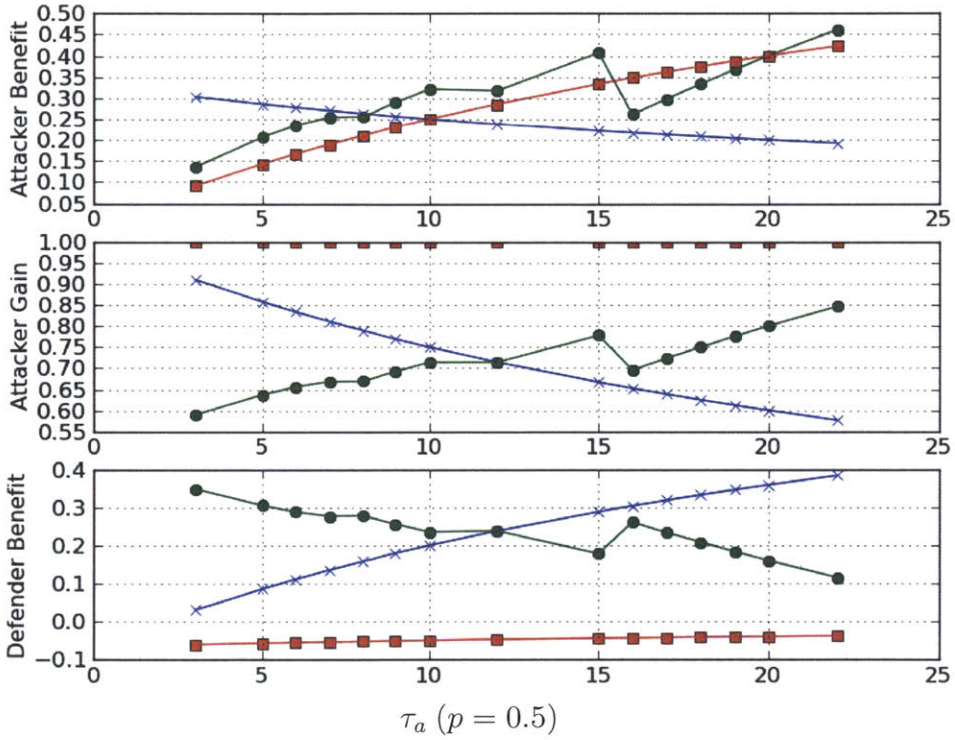

Figure 3-4: Experiment 3a. Fix $\tau_b = 30$ and vary τ_a.
Strategy 1: Cross (Blue), Strategy 2: Circle (Green), Strategy 3: Square (Red)

Finally, in the fourth experiment, we try (τ_a, τ_b) pairs, who are far away and with varying probability. This is very similar to Experiment 2, but for different (τ_a, τ_b) pairs.

Experiment 4 observations:

1. The main observation is that τ_a and τ_b should be chosen in such a way that would make the attacker choose Strategy 2 over some probability range. We

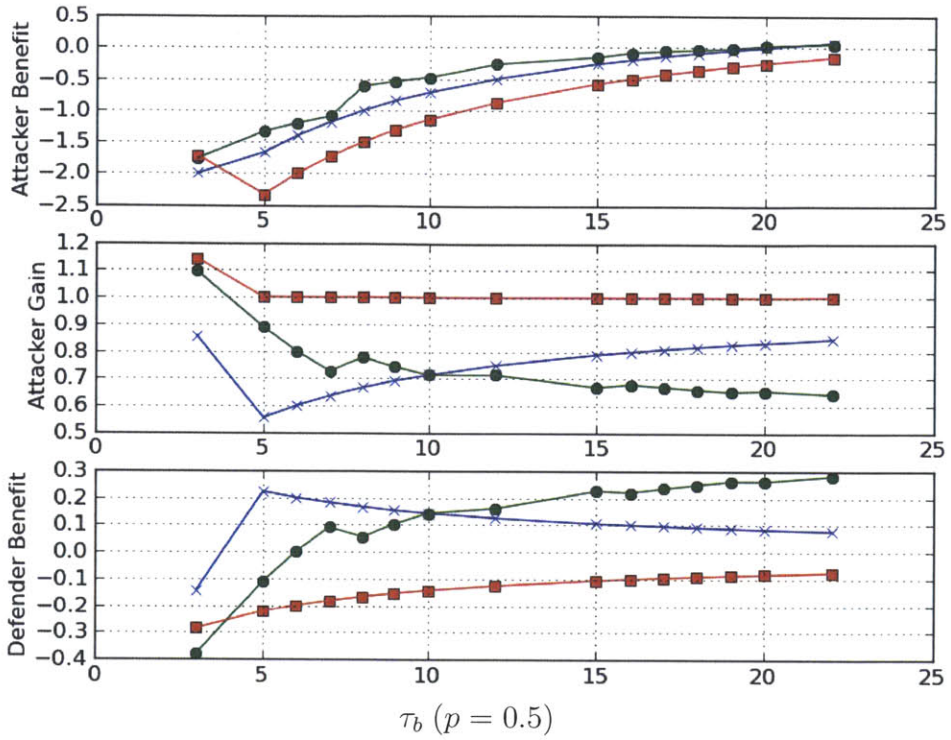

Figure 3-5: Experiment 3b. Fix $\tau_a = 4$ and vary τ_b
Strategy 1: Cross (Blue), Strategy 2: Circle (Green), Strategy 3: Square (Red)

observe that this range is conveniently in the higher probability values. This is exactly what we want, because the higher the probability, the lower the attacker benefit, when deploying Strategy 2. Therefore, we want Strategy 2 to outperform the other two strategies, but at the same time to perform as poorly as possible.

2. The second important observation, which we also mentioned earlier in the "Costs" section, is that the cost of the defender, k_0, should be significantly smaller than the attacker's cost; otherwise, the defender cannot control the source for a significant amount of time. Furthermore, notice that even if the ratio of the costs is the same, we could get different performance. The defender prefers $k_0 = 2$ and $k_1 = 10$ as opposed to $k_0 = 1$ and $k_0 = 5$. This indicates that we want the absolute value of the cost of the attacker to be as high as possible.

3. Considering τ_a and τ_b, we should have $\tau_a < k_1$ and $\tau_b > k_1$. τ_a should not

be too low, because then the costs go up for the defender and its benefit decreases significantly. τ_b should not be too far away, because then the expected rate of play of the defender is too low. Also, as τ_b increases Strategy 3 becomes better and we want to avoid that. In our specific specific example, $\tau_a = 4$ and $\tau_b = 50$ is what we want (Figure 3-6a). When $\tau_a = 3$ and $\tau_b = 30$, τ_a is too low (Figure 3-6b) and when $\tau_a = 6$ and $\tau_b = 80$, τ_b is too high (Figure 3-7).

4. In addition, what we observe in these figures is that the higher the p, the higher the benefit for the defender. Therefore, we should choose $p = 1 - \epsilon$, where $\epsilon > 0$ is arbitrarily close to zero, but not equal. However, as we already mentioned in another section, p should not be too close to one, because, then, we get a situation that is very similar to the periodic strategy; a strategy that performs poorly against an LM attacker. Later on in this section, we show that $p = 0.9$ gives satisfactory results.

5. Considering the optimal situation, where $\tau_a < k_1, \tau_b > k_1$ and p is high enough:

   - Strategy 1: Very high gain for the attacker. However, since we play τ_a with very high probability and the average rate of play is very high, the costs of this strategy are extremely high. Thus, the attacker's benefit is rather low.

   - Strategy 2: Relatively low gain for the attacker, but he only plays τ_b. Thus, his costs are low and he prefers this strategy. This is very good for the defender, because the difference $\tau_b - \tau_a$ is high and, thus, he achieves this large gain frequently; the higher the p the more frequently this gain is achieved.

   - Strategy 3: The gain is 1, but the costs are extremely high.

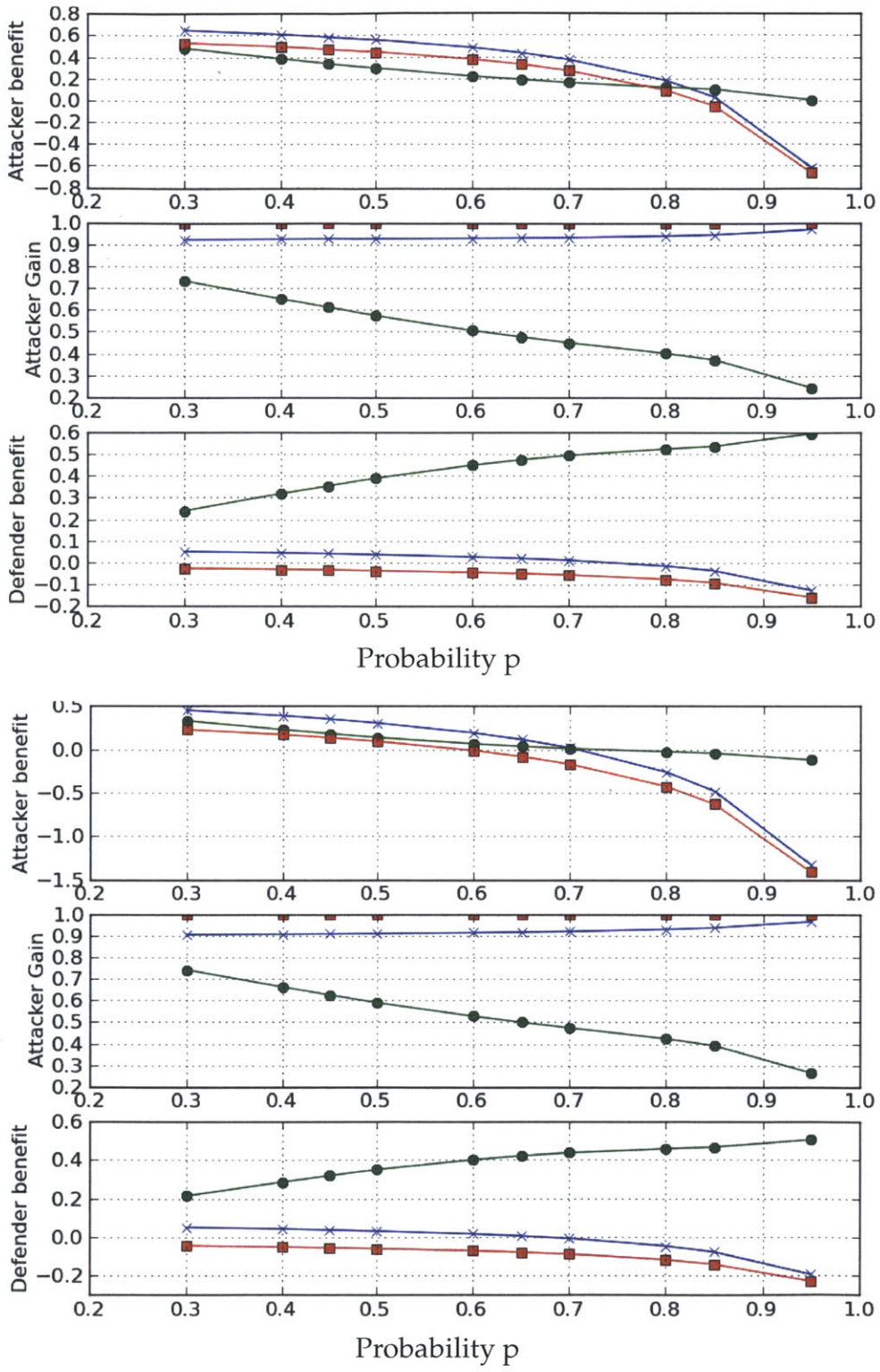

Figure 3-6: Experiment 4a (above): $\tau_a = 4, \tau_b = 50$. 4b (below): $\tau_a = 3, \tau_b = 30$. Strategy 1: Cross (Blue), Strategy 2: Circle (Green), Strategy 3: Square (Red)

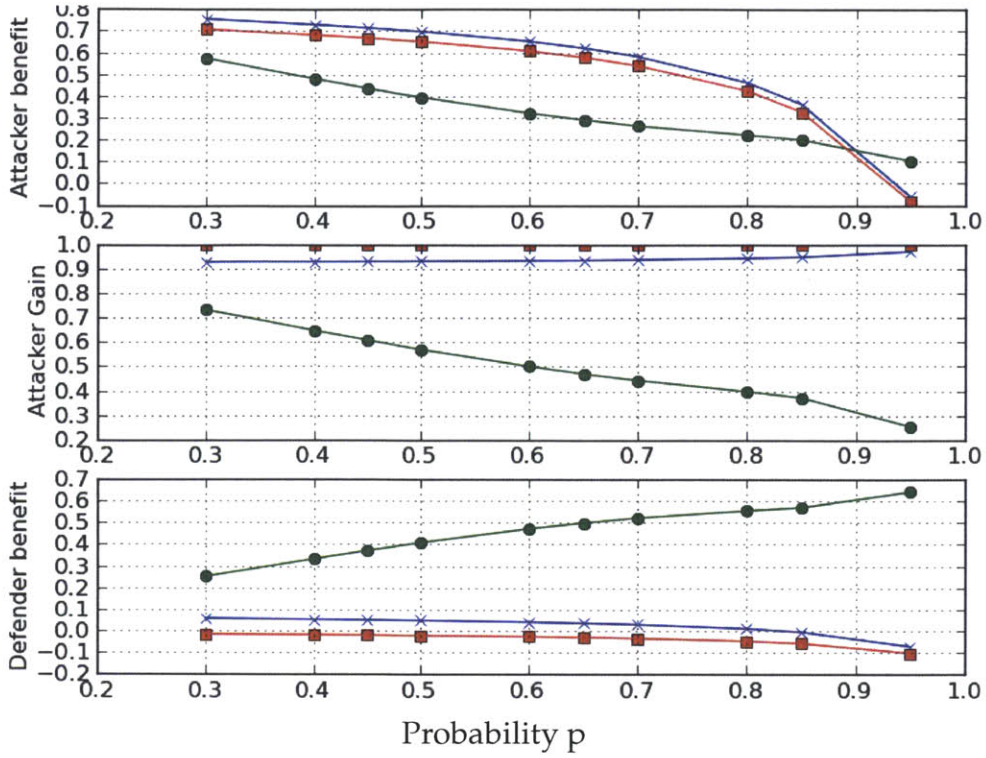

Figure 3-7: Experiment 4c: $\tau_a = 6, \tau_b = 80$.
Strategy 1: Cross (Blue), Strategy 2: Circle (Green), Strategy 3: Square (Red)

So far, we have all the guidelines we need to qualitatively describe the defender's biperiodic strategy. How do we actually find the appropriate quantities for the variables τ_a, τ_b and p?

As we have already observed, the higher the probability the better, but we do not want it to be too high, because then we get an almost periodic strategy. In fact, after running a lot of simulations (which include varying all three variables), we observe that $p = 0.9$ gives satisfactory results for the Defender. Therefore, we fix $p = 0.9$ and, then, according to the players' costs, we choose different τ_a and τ_b every time. The additional information we have is that $\tau_a < k_1$ and $\tau_b > k_1$. Therefore, τ_a is an integer in $(1, k_1)$ and, most of the times, it is not a number near the edges of this range. Now, we want to find a range for τ_b.

It is observed that as we increase τ_a, the optimal τ_b decreases. Thus, we run the code given in Appendix D with $\tau_a = 2$ and we obtain an optimal $\tau_b^{\tau_a=2}$, which we use as an upper bound, since if we use a higher τ_a, the value of the optimal τ_b will

be lower than $\tau_b^{\tau_a=2}$. Thus, the range, in which the optimal τ_b will be, is $(k_1, \tau_b^{\tau_a=2}]$.

The ranges for τ_a and τ_b are usually small enough and, thus, we apply exhaustive search in those ranges and find the optimal pair (τ_a, τ_b). If the ranges are too big, then more sophisticated – divide and conquer or heuristic – search techniques could be applied. However, we did not find it necessary, in any of our simulations.

3.4.3 Conclusion

Given that the attacker is adaptive, the biperiodic strategy is better than the periodic strategy, since the defender can, now, actually have some positive gain; instead of the opponent having full control. And even if the opponent has full control, which is obtained by deploying Strategy 3, then the attacker has to significantly increase his rate of play, since he has to play τ_a and with probability $(1-p)$ he has to play τ_b as well, which leads to higher costs. Thus, it is easier to lead him to a drop out.

Chapter 4

Exponential Defender

We consider an instance of the `FlipIt` game in which the defender is playing with an exponential strategy, but the attacker is more powerful and receives feedback during the game. In particular, the attacker finds out the exact time of the defender's last move every time he moves. It is an immediate observation that for a defender playing with a renewal strategy, an LM attacker is just as powerful as an FH one.

The defender plays with an exponential distribution with rate λ, denoted E_λ. The exponential distribution is *memoryless*: in a renewal process with inter-arrival time between events distributed exponentially, the time of the next event is independent of the time elapsed since the last event. Thus, an LM attacker that knows the time of the defender's last move has no advantage over a non-adaptive attacker in predicting the time of the defender's next move. Accordingly, we can prove that strategies from the LM class are strongly dominated by periodic strategies.

Combining Theorems 6, 7 and 8 from the original `FlipIt` paper, we realize that the choice of parameters λ (the rate of play of an exponential defender) and δ (the period of a periodic LM attacker) are dependent only on the costs of the two players. Thus, if the costs are known, the theoretical analysis is given below.

Theorem 1 *Assume that players' move costs are k_0 and k_1. The defender plays exponentially at rate λ and an LM attacker chooses period δ, based on the choice of λ, as follows:*

$$e^{-\lambda\delta}(1+\lambda\delta) = 1 - \lambda k_1$$

for $\lambda < 1/k_1$. Otherwise, the strongly dominant LM strategy for the attacker is not playing at all. Now, how does the defender chooses λ?

1. *If $k_0 \geq 0.854 \cdot k_1$, the maximum defender's benefit is achieved by playing at rate:*

$$\lambda = \frac{1-(1+z)e^{-z}}{k_1},$$

where z is the unique solution of equation

$$\frac{k_0}{k_1} = \frac{e^z - 1 - z}{z^3}.$$

For this choice of λ, the attacker's maximum benefit is achieved for period $\delta = \frac{z}{\lambda}$.

The benefits achieved by both players are:

$$\begin{aligned}
\beta_0(E_\lambda, P_{1/\delta}) &= 1 - \frac{1-e^{-z}}{z} - \frac{k_0}{k_1}\left[1-(1+z)e^{-z}\right] \geq 1 - k_0/k_1 \\
\beta_1(E_\lambda, P_{1/\delta}) &= \frac{1-e^{-z}}{z} - \frac{1-(1+z)e^{-z}}{z} = e^{-z}
\end{aligned}$$

2. *If $k_0 < 0.854 \cdot k_1$, the maximum defender's benefit is achieved by playing at rate $\lambda = 1/k_1$ and the attacker's best response is not playing at all. The benefits achieved by both players are:*

$$\beta_0(E_\lambda, P_{1/\delta}) = 1 - k_0/k_1; \;\; \beta_1(E_\lambda, P_{1/\delta}) = 0.$$

Chapter 5

Future Guidance & Contributions

As mentioned in the Introduction, we are only considering renewal strategies for the defender, whereas the attacker can be both adaptive and non-adaptive. We separate our observations in those two main categories based on the attacker's strategy. In each category, we are considering three defender strategies: (1) periodic, (2) exponential and (3) biperiodic.

1. Non-adaptive attacker.

   As it was shown in the original paper, the periodic strategy was the strongly dominant strategy. Therefore, the periodic defender has a higher expected benefit than the exponential. How about the biperiodic? The biperiodic defender benefit is at most as good as the periodic one and depending on the choice of τ_a, τ_b and p it could be performing worse than the exponential as well. Therefore, for a non-adaptive attacker, we select a periodic defender. Moreover, the non-adaptive attacker is periodic, as well.

2. Adaptive attacker.

   The periodic strategy of the defender is very weak against an adaptive attacker. The adaptive attacker has a gain of one and, thus, the benefit of the defender can be at most zero. Therefore, we do not consider that strategy. Now, we only have to compare the benefits of an exponential defender (Section 4) with the benefits of a biperiodic defender (Section 3.4). In both cases,

the selection of parameters depends only on the costs k_0 and k_1 (known by both players). We modify the costs and observe the defender's benefits. The results are summarized in Figure 5-1. Here we fix $k_1 = 20$ and we discretely vary $k_0 \in [1, 19]$. As we can see the Exponential defender outperforms the biperiodic defender. Thus, the biperiodic defender benefits are between the periodic and the exponential defender ones.

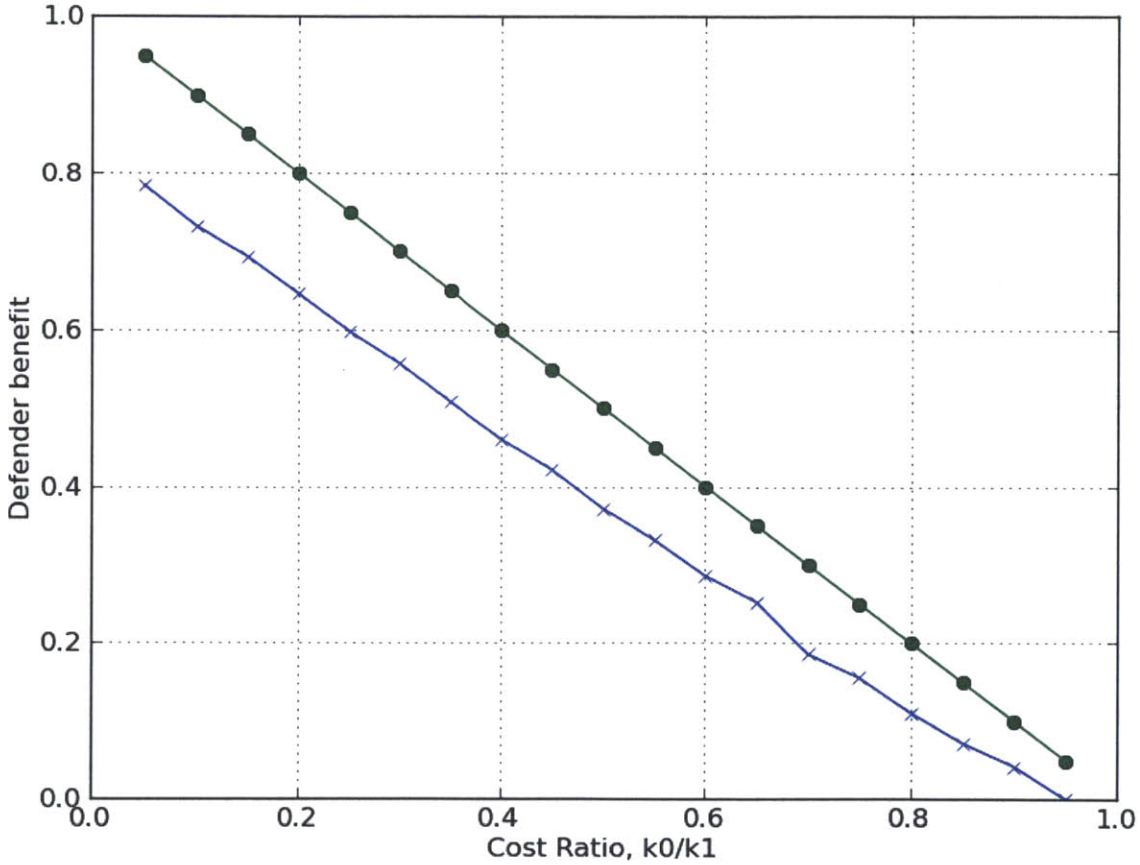

Figure 5-1: Biperiodic and Exponential Defender against LM Attacker
Blue(Cross): Biperiodic, Green(Circle): Exponential

Future Guidance

i. The *delayed exponential* distribution, in which the defender waits for a fixed interval of time Δ before choosing an exponentially distributed interval until his

next move, outperforms the simple exponential strategy, for some parameter choices. This is demonstrated experimentally in the original `FlipIt` paper. Future research can be done on modified exponential strategies in aim for improved defender benefits.

ii. More specifically, in the delayed exponential, which seems to be an interesting direction to follow: if you do not know what Δ is, how can you infer it from the feedback? You will need some statistical interpretation.

iii. What should the period of an LM attacker be when the defender deploys a delayed exponential strategy?

iv. An interesting instance of `FlipIt` is when both players are sophisticated enough; i.e. they both deploy adaptive strategies. Are there any dominant strategies? Can it be reduced to a residual game? What kind of adaptive strategies should the utilize?

v. The biggest open question is whether there exist other strategies (or distributions) that outperform an exponential defender, when playing against an LM attacker?

Contributions

1. Exhaustive analysis of a biperiodic defender. Some theoretical analysis, which was mostly based on results from the original paper. However, most of the analysis was experimental and it reinforced the theoretical results.

2. Provide a framework for the experimental analysis of the FlipIt game. Based on the core algorithms found in the Appendices, you can easily extend or modify them in such a way to evaluate new strategies.

3. Instead of biperiodic defender, you could choose from more than two possible time intervals with some probabilities. This makes the defender less predictable. As the options tend to infinity, you get results closer to the exponential strategy.

4. In the case of an LM attacker, we prefer an exponential defender, because he is less predictable. However, if we want to use a biperiodic defender, we provide a qualitative framework on how to choose the parameters of the strategy. Moreover, we present an algorithm to quantitatively find those parameters.

Bibliography

[1] M. van Dijk, A. Juels, A Oprea, and R. Rivest. `FlipIt:` *The Game of "Stealthy Takeover"*. RSA Laboratories and CSAIL, Cambridge, MA, 2012

[2] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song. *Provable data possessionat untrusted stores.* In Proc. 14th ACM Conference on Computer and Communication Security (CCS), 2007

[3] K. D. Bowers, M. van Dijk, A. Juels, A Oprea, and R. Rivest. *Remote assesment of fault tolerance.* In Proc. 18th ACM Conference on Computer and Communication Security (CCS), 2011

[4] D. Y. Chan and M. A. Vasarhelyi. *Innovation and practice of continuous auditing.* International Journal of Accounting Information Systems, 12(2):152160, 2011

[5] Y. Dodis, J. Katz, S. Xu, and M. Yung. *Key-insulated public key cryptosystems.* In Proc. IACR EUROCRYPT, 2002

[6] W. Feller. *An introduction to probability theory and its applications.* John Wiley and Sons, second edition, 2011

[7] D. Fundenberg and J. Tirole. *Game Theory.* MIT, 1991

[8] R. G. Gallager. *Discrete Stochastic Processes.* Springer, 1996

[9] S. N. Hamilton, W. L. Miller, A. Ott, and O. S. Saydjari. *Challenges in applying game theory to the domain of information warfare.* In Information Survivability Workshop (ISW), 2002

[10] G. Itkis. *Cryptographic tamper-evidence.* In Proc. 10th ACM Conference on Computer and Communication Security (CCS), 2003

[11] G. Itkis. *Handbook of Information Security.* John Wiley and Sons, 2006

[12] J. Katz. *Bridging game theory and cryptography: recent results and future directions.* In Proc. Theory of Cryptography Conference (TCC), pages 251272, 2008

[13] G. J. Mailath and L. Samuelson. *Repeated Games and Reputations: Long-run relationships.* Oxford, 2006

[14] T. Moore, A. Friedman, and A. Procaccia. *Would a cyber warrior protect us? exploring trade-offs between attack and defense of information systems.* In Proc. New Security Paradigms Workshop (NSPW), pages 8594, 2010

[15] R. B. Myerson. *Game Theory – Analysis of Conflict.* Harvard University Press, 1997

[16] K.C.Nguyen, T.Alpcan and T.Basar. *Security games with incomplete information.* In Proc. IEEE International Conference on Communications (ICC), 2009

[17] D. Pavlovic. *Gaming security by obscurity.* CoRR abs/1109.5542, 2011

[18] R. L. Rivest. *Illegitimi non carborundum.* Invited keynote talk given at CRYPTO 2011, 2011

[19] S. Ross. *Stochastic Processes.* John Wiley and Sons, Inc., 1996

[20] S. Roy, C. Ellis, S. Shiva, D. Dasgupta, V. Shandilya, and Q. Wu. *A survey of game theory as applied to network security.* In Proc. Hawaii International Conference on System Sciences (HICSS), pages 110, 2010

[21] N.D. Schwartz and C. Drew. *RSA faces angry users after breach..* New York Times, page B1, 8 June 2011

Appendix A

Biperiodic Defender - Simulation

This is the basic code that we used to run all our simulations for the Biperiodic Strategy of the Defender. Variations of this code were used to generate the results presented in this Appendix.

```
#Fix the intervals of the attacker: 0, d1, 2*d1, 3*d1,...
#and check how the defender interferes.
import random

#choose t_a with probability p and t_b with probability 1-p.
def get_dist(p, t_a, t_b):
    if (random.random() < p):
        return t_a
    else:
        return t_b

#the probabilistic periodic strategy
def pr_periodic(d1, t_a, t_b, def_cost, att_cost, p, repeat):
    def_gain=0 #defender gain
    att_gain=0 #attacker gain
    #the times defender attempts to take over control
    count_def = 0
    count_att = 1 #same for the attacker
    total = repeat*d1 #Total time we run the simulation.
    #Pointers simulating the progress of the two players.
    defender = 0
    attacker = d1

    while (attacker <= total):
        if (defender < attacker):
            while (defender < attacker):
                distance = get_dist(p, t_a, t_b)
```

```
                defender += distance
                count_def += 1
                def_gain += distance
            def_gain -= distance
            def_gain += (attacker-(defender-distance))

        if (defender == attacker):
            attacker += d1
            count_att += 1

        if (defender > attacker):
            att_gain += (defender - attacker)
            while (attacker < defender):
                attacker += d1
                count_att += 1

    def_benefit = float(def_gain-def_cost*count_def)/total
    att_benefit = float(att_gain-att_cost*count_att)/total

    #The following numbers represent the percentage gain.
    d = float(def_gain)/(total)
    a = float(att_gain)/(total)
    return (d, a, def_benefit, att_benefit)

#Parameters
#Try different periods for the attackers
Attacker_list = [20, 50, 100, 150]
#Defender periods, based on attacker periods
Defender_list = {20: [11, 16, 29, 51],
          50: [11, 16, 29, 51, 71, 91, 101, 105, 137],
          100: [16, 51, 91, 101, 105, 137],
          150: [51, 91, 105, 137, 161, 191, 201]
          }

#Try for different probabilities.
probabilities = [0.1, 0.3, 0.5, 0.65, 0.9]

k_def = 10 #defender cost
k_att = 10 #attacker cost

#This implicictly defines the total simulation time.
total_attacker_moves = 5000

for attacker in Attacker_list:
    print "ATTACKER == ", attacker
    defender = Defender_list[attacker]
    for t_a in defender:
```

```
        for t_b in defender:
            print "t_a=", t_a, ", t_b=", t_b
            if (t_a < attacker and t_b < attacker):
                print "both less than attacker"
            elif (t_a > attacker and t_b > attacker):
                print "both greater than attacker"
            else:
                print "1 less and 1 greater than attacker"
            for p in probabilities:
                #d0 is the expected period
                d0 = p*t_a + (1-p)*t_b
              #Calculate theoretical results for comparison
                if (d0 > attacker):
                    r = float(attacker)/d0
                    v = 1 - r/2
                    b1 = r/2 - float(k_def)/d0
                    b2 = v - float(k_att)/attacker
                else:
                    r = float(d0)/attacker
                    v = r/2
                    b1 = 1- v - float(k_def)/d0
                    b2 = v - float(k_att)/attacker

                #Repeat multiple times and report averages.
                #We checked that 2000 is big enough.
                gain_def=0
                gain_att=0
                benefit_def=0
                benefit_att=0
                for i in range(2000):
                    results = pr_periodic(attacker, t_a, t_b,
                    k_def, k_att, p, total_attacker_moves)
                    gain_def += results[0]
                    gain_att += results[1]
                    benefit_def += results[2]
                    benefit_att += results[3]
                aver_def = float(gain_def)/2000
                aver_att = float(gain_att)/2000
                aver_def_benefit = float(benefit_def)/2000
                aver_att_benefit = float(benefit_att)/2000
```

The following results summarized in the table are specific instances of the simulations that we chose to present in this thesis that reflect the main observations of the whole data set.

Attacker $\delta_1 = 20$					
τ_a	τ_b	p	**Attacker Gain**	**Attacker Benefit**	**Defender Benefit**
11	11	0.5	0.25	-0.25	-0.16
Expected δ_0: 11			0.275	-0.225	-0.184
11	16	0.1	0.367	-0.134	-0.0113
		δ_0: 15.5	0.388	-0.113	-0.032
		0.3	0.347	-0.153	-0.036
		δ_0: 14.5	0.363	-0.138	-0.052
		0.9	0.267	-0.233	-0.137
		δ_0: 11.5	0.288	-0.213	-0.157
16	29	0.1	0.623	0.123	0.016
		δ_0: 27.7	0.638	0.139	0
		0.5	0.545	0.044	0.011
		δ_0: 22.5	0.556	0.056	0
		0.9	0.419	-0.081	0.003
		δ_0: 17.3	0.433	-0.068	0
Attacker $\delta_1 = 50$					
29	51	0.5	0.7475	0.24	0.013
		δ_0: 40	0.75	0.25	0
		0.9	0.668	0.168	0.016
		δ_0: 31.2	0.679	0.18	0
11	51	0.3	0.466	0.266	0.278
		δ_0: 39	0.39	0.19	0.354
		0.5	0.43	0.23	0.25
		δ_0: 31	0.31	0.11	0.37
11	71	0.3	0.607	0.407	0.204
		δ_0: 53	0.53	0.33	0.28
		0.5	0.568	0.368	0.188
		δ_0: 41	0.41	0.21	0.336

τ_a	τ_b	p	Attacker Gain	Attacker Benefit	Defender Benefit
16	29	0.1	0.2725	0.0724	0.367
		δ_0: 27.7	0.277	0.077	0.362
		0.5	0.234	0.034	0.322
		δ_0: 22.5	0.225	0.025	0.33
40	60	0.3	0.456	0.256	0.36
		δ_0: 54	0.537	0.337	0.278
		0.65	0.389	0.129	0.43
		δ_0: 47	0.47	0.27	0.217
45	54	0.3	0.504	0.304	0.3
		δ_0: 51.3	0.513	0.313	0.292
		0.5	0.488	0.288	0.31
		δ_0: 49.5	0.495	0.295	0.303
40	71	0.3	0.592	0.392	0.246
		δ_0: 61.7	0.594	0.394	0.243
		0.9	0.431	0.231	0.337
		δ_0: 43.1	0.431	0.231	0.337
Attacker $\delta_1 = 100$					
91	105	0.1	0.512	0.412	0.391
		δ_0: 103.6	0.52	0.417	0.386
		0.5	0.487	0.387	0.411
		δ_0: 98	0.49	0.39	0.408
85	105	0.3	0.474	0.374	0.425
		δ_0: 99	0.495	0.395	0.404
		0.5	0.455	0.355	0.439
		δ_0: 95	0.475	0.375	0.42

Table A1: Summary of experimental results of varying τ_a, τ_b and p and comparison with theoretical values related to the expected period $\delta_0 = p\tau_a + (1-p)\tau_b$.

τ_a	τ_b	**Attacker Gain**	**Attacker Benefit**	**Defender Benefit**
Attacker $\delta_1 = 100$				
$\delta_0 = 23$		0.115	0.065	0.6676
22	24	0.105	0.055	0.6774
18	28	0.11	0.06	0.6722
$\delta_0 = 53$		0.265	0.215	0.6407
52	54	0.255	0.205	0.651
46	60	0.259	0.21	0.646
43	63	0.269	0.22	0.636
$\delta_0 = 99$		0.495	0.445	0.4545
98	100	0.487	0.437	0.463
84	114	0.492	0.442	0.457
79	119	0.502	0.451	0.448
$\delta_0 = 131$		0.618	0.568	0.344
106	156	0.61	0.56	0.351
81	181	0.622	0.572	0.34
Attacker $\delta_1 = 50$				
$\delta_0 = 35$		0.35	0.15	0.364
35	35	0.3	0.09	0.414
30	40	0.257	0.05	0.4573
21	49	0.396	0.196	0.3185
20	50	0.314	0.11	0.400
$\delta_0 = 50$		0.5	0.3	0.3
45	55	0.455	0.255	0.345
40	60	0.42	0.22	0.38
36	64	0.502	0.302	0.298
25	75	0.375	0.175	0.425

Table A2: We fix δ_0 and symmetrically expand τ_a and τ_b, while $p = 0.5$.

Appendix B

Known costs - Simulation

```
p = 0.5, a_moves = 5000 #attacker_moves
#Try different costs and based on them calculate
#the attacker's and the defender's period.
Costs = [10, 15, 20, 50, 80, 100]
#symmetrically expanding t_a & t_b about d0.
check = [0, 1, 2, 4, 5, 7, 10, 15, 20, 25, 30]
for k0 in Costs:
    for k1 in Costs:
        if (k0 < k1):
            d1 = 2*pow(k1,2)/float(k0), d0 = 2*k1
        elif (k0 > k1):
            d1 = 2*k0, d0 = 2*pow(k0,2)/float(k1)
        else:
            d0 = d1 = 2*k0
        d0 -= 0.01 #simulating random phase
        for i in check:
            if (i >= d0):
                break
            t_a = d0 + i, t_b = d0 - i
            #Repeat it multiple times and report averages.
            gain_def=0, gain_att=0, benefit_def=0, benefit_att=0
            for i in range(2000):
                results = pr_periodic(d1,t_a,t_b,k0,k1,p,a_moves)
                gain_def += results[0]
                gain_att += results[1]
                benefit_def += results[2]
                benefit_att += results[3]
            aver_def = float(gain_def)/2000
            aver_att = float(gain_att)/2000
            aver_def_benefit = float(benefit_def)/2000
            aver_att_benefit = float(benefit_att)/2000
```

Costs	τ_a	τ_b	Defender Gain	Defender Benefit
$k_0 = 10, k_1 = 10$	19.99	19.99	0.45	- 0.0497
$\delta_0 = 20, \delta_1 = 20$	18.99	20.99	0.5001	- 0.0003
	17.99	21.99	0.498	- 0.0023
	20.01	20.01	0.5496	0.0498
	19.01	21.01	0.4995	- 0.0003
$k_0 = 10, k_1 = 20$	39.99	39.99	0.725	0.475
$\delta_0 = 40, \delta_1 = 80$	38.99	40.99	0.75	0.5
	34.99	44.99	0.746	0.496
$k_0 = 10, k_1 = 80$	159.99	159.99	0.931	0.869
$\delta_0 = 160, \delta_1 = 1280$	157.99	161.99	0.938	0.875
	139.99	179.99	0.9365	0.874
$k_0 = 10, k_1 = 100$	199.99	199.99	0.945	0.895
$\delta_0 = 200, \delta_1 = 2000$	194.99	204.99	0.95	0.9
	189.99	209.99	0.949	0.899
$k_0 = 15, k_1 = 20$	39.99	39.99	0.625	0.25
$\delta_0 = 40, \delta_1 = 53.3$	38.99	40.99	0.6246	0.2496
	35.99	43.99	0.621	0.246
$k_0 = 15, k_1 = 100$	199.99	199.99	0.925	0.85
$\delta_0 = 200, \delta_1 = 1333.3$	194.99	204.99	0.925	0.85
	179.99	219.99	0.924	0.849
$k_0 = 100, k_1 = 10$	1999.99	1999.99	0.0015	-0.048
$\delta_0 = 2000, \delta_1 = 200$	1995.99	2003.99	0.0496	-0.0005
	1994.99	2004.99	0.0498	-0.0002
$k_0 = 100, k_1 = 50$	399.99	399.99	0.0315	-0.218
$\delta_0 = 400, \delta_1 = 200$	394.99	404.99	0.25	0
	392.99	406.99	0.249	-0.001

Table B1: Vary the costs of the players and find their Nash equilibria.

Appendix C

RP Attacker - Simulation

Costs	τ_a	τ_b	Defender Gain	Defender Benefit
$k_0 = 20, k_1 = 10$	Theoretical, $\delta_0 = 20$		-	0.0625
$\delta_0 = 320, \delta_1 = 80$	319.99	319.99	0.0197	- 0.0428
	318.99	320.99	0.119	0.0565
	315.99	323.99	0.125	0.0623
	320.01	320.01	0.231	0.168
	319.01	321.01	0.1306	0.068
	300.01	340.01	0.137	0.074
$k_0 = 50, k_1 = 15$	Theoretical, $\delta_0 = 1333$		-	0.0375
$\delta_0 = 1333, \delta_1 = 200$	1331	1335	0.0749	0.0374
	1326	1340	0.075	0.0375
$k_0 = 100, k_1 = 20$	Theoretical, $\delta_0 = 4000$		-	0.025
$\delta_0 = 4000, \delta_1 = 400$	3995.99	4003.99	0.0479	0.0229
	3974.99	4024.99	0.0476	0.0225
	3999.01	4001.01	0.055	0.03
	3980.01	4020.01	0.052	0.027

Table C1: Vary the costs and find the optimal periods based on Theorem 5.

Appendix D

LM Attacker - Simulation

This is the basic code that we used to run all our simulations for the LM attacker. We have three functions and every function represents one of the three different strategies of the attacker.

```
def  attacker_Strategy1(t_a,t_b,p, def_cost, att_cost, repeat,e):
    def_gain=0 #defender gain
    att_gain=0 #attacker gain
    #the times defender attempts to take over control
    count_def = 0
    count_att = 1
    total = repeat*t_b #Total time we run the simulation.
    #Pointers simulating the progress of the two players.
    defender = 0
    attacker = t_a+e

    token = True #defender controls
    current = 0 #most recent acquisition of token

    while (defender < total):
        distance = get_dist(p, t_a, t_b)
        defender += distance
        count_def += 1
        if (defender < attacker):
            if (token):
                def_gain += defender - current
            else:
                att_gain += defender - current
            def_gain += (attacker - defender)
            token = False #attacker controls
```

```
            current = attacker
        else:
            if (token):
                def_gain += attacker - current
            else:
                att_gain += attacker - current
            att_gain += (defender - attacker)
            token = True
            current = defender
        attacker = defender + t_a + e
        count_att += 1
    total = defender
    def_benefit = float(def_gain-def_cost*count_def)/total
    att_benefit = float(att_gain-att_cost*count_att)/total
    #The following numbers represent the percentage gain.
    d = float(def_gain)/(total)
    a = float(att_gain)/(total)
    #print "Defender rate", count_def/float(total)
    #print "Attacker rate", count_att/float(total)
    return (d, a, def_benefit, att_benefit)

def  attacker_Strategy2(t_a,t_b,p, def_cost, att_cost, repeat,e):
    def_gain=0 #defender gain
    att_gain=0 #attacker gain
    #the times defender attempts to take over control
    count_def = 0
    count_att = 1
    total = repeat*t_b #Total time we run the simulation.
    #Pointers simulating the progress of the two players.
    defender = 0
    attacker = t_b+e
    old = defender
    old_token = False
    prev_defender = 0
    token = True #defender controls
    current = 0 #most recent acquisition of token
    update = False

    while (defender < total):
        distance = get_dist(p, t_a, t_b)
        defender += distance
        count_def += 1
        if (prev_defender + e == attacker):
            update = True
        if (attacker - defender > t_a):
            if (token):
                def_gain += defender - current
```

```
            else:
                att_gain += defender - current
            token = True
            current = defender
            old = defender
            old_token = True
            prev_defender = defender
            continue
        if (defender < attacker):
            if (token):
                def_gain += defender - current
            else:
                att_gain += defender - current
            def_gain += (attacker - defender)
            token = False #attacker controls
            current = attacker
            attacker = defender + t_b + e
        else:
            if (token):
                def_gain += attacker - current
            else:
                att_gain += attacker - current
            att_gain += (defender - attacker)
            token = True
            current = defender
            attacker = defender + e
        if (old_token and distance == t_b):
            attacker = old + t_b + e
        if (old_token):
            old_token = False
        if (update):
            update = False
            attacker = prev_defender+t_b+e
        prev_defender = defender
        count_att += 1
    total = defender
    def_benefit = float(def_gain-def_cost*count_def)/total
    att_benefit = float(att_gain-att_cost*count_att)/total
    #The following numbers represent the percentage gain.
    d = float(def_gain)/(total)
    a = float(att_gain)/(total)
    #print "Defender rate", count_def/float(total)
    #print "Attacker rate", count_att/float(total)
    return (d, a, def_benefit, att_benefit)

def attacker_Strategy3(t_a,t_b,p, def_cost, att_cost, repeat,e):
    def_gain=0 #defender gain
```

```
    att_gain=0 #attacker gain
    #the times defender attempts to take over control
    count_def = 0
    count_att = 1
    total = repeat*t_b #Total time we run the simulation.
    #Pointers simulating the progress of the two players.
    defender = 0
    attacker = t_a+e

    token = True #defender controls
    current = 0 #most recent acquisition of token

    while (defender < total):
        distance = get_dist(p, t_a, t_b)
        defender += distance
        count_def += 1
        if (defender < attacker):
            if (token):
                def_gain += defender - current
            else:
                att_gain += t_a-e
            def_gain += e
        else:
            count_att += 1
            att_gain += defender - attacker
            def_gain += 2*e
            if (token):
                def_gain += t_a+e
            else:
                att_gain += t_a-e
        token = False
        current = attacker
        attacker = defender + t_a + e
        count_att += 1
    total = defender
    def_benefit = float(def_gain-def_cost*count_def)/total
    att_benefit = float(att_gain-att_cost*count_att)/total
    d = float(def_gain)/(total)
    a = float(att_gain)/(total)
    return (d, a, def_benefit, att_benefit)
```

The following code is the exhaustive search that finds the optimal pair (τ_a, τ_b), given that $p = 0.9$.

```
def find_optimal_pair(t_a, t_b, p, k0, k1, total_attacker_moves,
                                        e, best_benefit):
    results = att_t_a(t_a,t_b,p, k0, k1, total_attacker_moves,e)
    def_bens1 = results[2]
    att_bens1 = results[3]

    results = att_t_b(t_a,_b,p, k0, k1, total_attacker_moves,e)
    def_bens2 = results[2]
    att_bens2 = results[3]

    results = att_both(t_a,t_b,p, k0, k1, total_attacker_moves,e)
    def_bensb = results[2]
    att_bensb = results[3]
    best_t_a = -1
    best_t_b = -1
    if (att_bens2 > att_bens1 and att_bens2 > att_bensb):
        if (def_bens2 > best_benefit):
            best_benefit = def_bens2
            best_t_a = t_a
            best_t_b = t_b
    return (best_benefit, best_t_a, best_t_b)

#parameters
e = 0.001
k0 = 1
k1 = 10
total_attacker_moves = 5000
p=0.9

#find the range for t_b
find_upper_bound=15*k1
best_t_a = 0
best_t_b = 0
best_benefit=-10
t_a=2
for t_b in range(k1+1, find_upper_bound):
    a, b, c = find_optimal_pair(t_a, t_b, p, k0, k1,
                 total_attacker_moves, e, best_benefit)
    best_benefit = a
    if (b != -1):
        best_t_a = b
        best_t_b = c
```

```
upper_bound = best_t_b
best_t_a = 0
best_t_b = 0
best_benefit=-10
#exhaustive search for t_a and t_b
for t_a in range(3, k1):
    for t_b in range(k1+1, upper_bound):
        a, b, c = find_optimal_pair(t_a, t_b, p, k0, k1,
                total_attacker_moves, e, best_benefit)
        best_benefit = a
        if (b != -1):
            best_t_a = b
            best_t_b = c

print "t_a=", best_t_a, "t_b=", best_t_b, "best_b=", best_benefit
```