A Very Simple $\LaTeX 2_\varepsilon$ Template

Vitaly Surazhsky
Department of Computer Science
Technion—Israel Institute of Technology
Technion City, Haifa 32000, Israel

Yossi Gil
Department of Computer Science
Technion—Israel Institute of Technology
Technion City, Haifa 32000, Israel

April 23, 2012

Abstract

This is the paper's abstract . . .

1 Introduction

This is time for all good men to come to the aid of their party!

1.1 split

```
#splitting by whitespace
"python is kind of fun".split()
```

Regular Expressions

The module to import in python is called re. A regular expression is written

```
import re
re.findall(r"[0-9]","1+2==3")
#--> ['1','2','3']
re.findall(r"[0-9][0-9]","12345")
#--> ['12','34']
re.findall(r"[0-9]+","13 from 1 in 1776")
#--> ['13', '1', '1776'] Maximal Munch. Don't stop early. go all the way
```

where the *r* actually means *raw string* instead of *regular expression.* The + and * operators are called *Kleene operators* after Stephen C. Kleene.

Finite State Machine (FSM)

A visual representation or a pictorial equivalent to regular expressions. A **non-deterministic FSM** includes epsilon transitions or ambiguity. A **deterministic FSM/lock-step FSM** includes **no** epsilon edges or ambiguity. However, every non-deterministic FSM has a corresponding deterministic FSM that accepts exaclty the same strings. Non-deterministic FSMs are not more powerful, they are just more convenient.

Server

A server is a machine optimized for sitting in a closet and hosting files.

HyperText Markup Language (HTML)

Use the tags *strong* and *em* when the contents of your page requires that certain words or phrases be stressed. If you are only highlighting for visual effect use the tags *b* and *i*.

HTTP Request

GET request: GET /foo HTTP/1.1

HTTP Response

A response can be static, which is a pre-written file or dynamic, which is a page made on the fly by programs called web applications.

* Response: HTTP/1.1 200 OK
* Date: Tue Mar 2012 04:33:33 GMT
* Server: Apache /2.2.3 - Similar to User-Agent header on the request. Best to make this up, otherwise you're just giving away free information to a would be hacker that want to know what vulnerability that works against you.
* Content-Type: text/html; charset=utf-8
* Content-Length: 1539

Status Codes

200 OK 302 Found - The document is located somewhere else 404 Not Found 500 Server Error - The server broke trying to handle your request

HTML Header

Valid headers, such as *User-Agent*, *Host*, but really, you can make up all the headers you want.

Use the <*br* /> tag instead of <*br*> for an inline line break, but the *p* tag to make a block.

Use the *span* for an inline container, which content can be styled, and *div* for a block container.

If your browser crashes, you should quit using Internet Explorer.

Cryptography

Shannon's Keyspace Theorem

Monoalphabetic Substitution Cipher

Each letter in the alphabet is mapped to a substitution letter.

CT only attack. "E" is the most common letter in the English language (appears about 12.7% of the time), which will appear as the most frequent coded letter in the cipher. Next is "T" (9.1%), "A" (8.1%), etc. Next step is to study frequency of pairs of letters: "he", "an", "in", etc.

One way to prove that the cipher is imperfect, is to use *Shannon's Keyspace Theorem*1.1. Assume a 26-letter alphabet,

$$|K| < |M|, 26! < 2^{89}, 26! < 26^{19}. \tag{1}$$

Another way to prove the cipher's imperfection is by showing a ciphertext c that could not decrypt to a message m for any key k,

$$c = \mathsf{aa}, m = \mathsf{cs} \tag{2}$$

Randomness

Kolmogorov Complexity. $K(s)$ =length of the shortest possible description of s [Andrey Kolmogorov (1903-1987)]. If there isn't any short program that can describe the sequence, that's an indication that the sequence is random, e.i, s is random if $K(s) = |s| + C$. There's no simplier way to understand the sequence other than to see the whole sequence. However, the Kolmogorov Complexity is **uncomputable**.

Statistical Tests - can only show non-randomness. We can always find some non-random sequence that satisfies all of our statistical tests.

Physically Random Events

* Quantum Mechanics
* Thermal Noise
* User key presses/mouse movements (?)

Pseudo-Random Number Generator (PRNG) takes a small amount of physical randomness and turn them into a long sequence of apparently "random" bits. This can be done by extracting a seed once from a *random pool* and reusing it in every step, encrypting a sequence of values (which can be a counter).

Does this produce a sequence that appears random? No, it repeats values too infrequently, why the key is changed every few million outputs. Another concern is whether the pool of randomness is good enough. On unix machines, this pool is stored in */dev/random* and is collecting events that are believed to be random, like user interactions. A popular PRNG is *Fortuna*.

Cipher Block Chaning (CBC

Assuming E has perfect secrecy (impossible since $|K| \geq |M|$), an attacker can still learn the length of the message and which blocks in m are equal from a captured c, where

$$c_i = E_k(m_i) \tag{3}$$

With CBC,

$$\begin{aligned} c_0 &= E_k(m_0 \oplus IV) \\ c_i &= E_k(m_i \oplus c_{i-1}) \end{aligned} \tag{4}$$

where the initial message block is xor:ed with an initialization vector, IV, which should not be repeated. The point with the initialization vector is just to hide repetition in the first block. Being lost, the whole message can still be recovered, except for the very first block.

$$\begin{aligned} m_0 &= D_k(c_0) \oplus IV \\ m_{n-i} &= D_k(c_{n-1}) \oplus c_{n-2} \end{aligned} \tag{5}$$

Outline The remainder of this article is organized as follows. Section 2 gives account of previous work. Our new and exciting results are described in Section 3. Finally, Section 4 gives the conclusions.

2 Previous work

A much longer LaTeX 2$_\varepsilon$ example was written by Gil [**?**].

3 Results

In this section we describe the results.

4 Conclusions

We worked hard, and achieved very little.

Index