Obfuscating Point Functions with Multibit Output

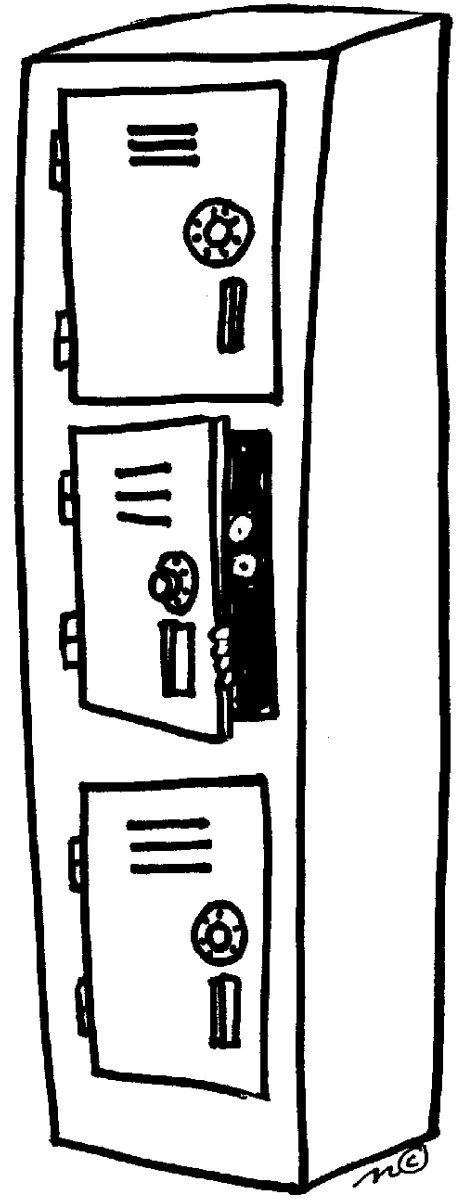
Ran Canetti, Ronny R. Dakdouk

Outline

- Application: Digital Lockers
- Background
- Construction
- Analysis
- (Re)evaluating security

Application: Digital Lockers

- An object with a combination lock
- Correctness:
  - Store content
  - Recover content (with password)
- Secrecy:
  - Content is as secure as the password.
  - The only way of opening a DL is by guessing the password



Encryption is not enough because we have weak passwords.

Digital Lockers

Insecure

Secure

Digital Locker - Correctness

- A DL is a couple of algorithms ***lock*** and ***unlock.***
- Correctness: ***unlock***(pass,***lock***(pass,content))=content
- Probabilistic version allow for negligible correctness error.

Digital Locker - Secrecy

- Recovering content is as hard as guessing the password.
- Simulation-based definition

Lock(pass,content)

≈

If pass is correct, reveal content

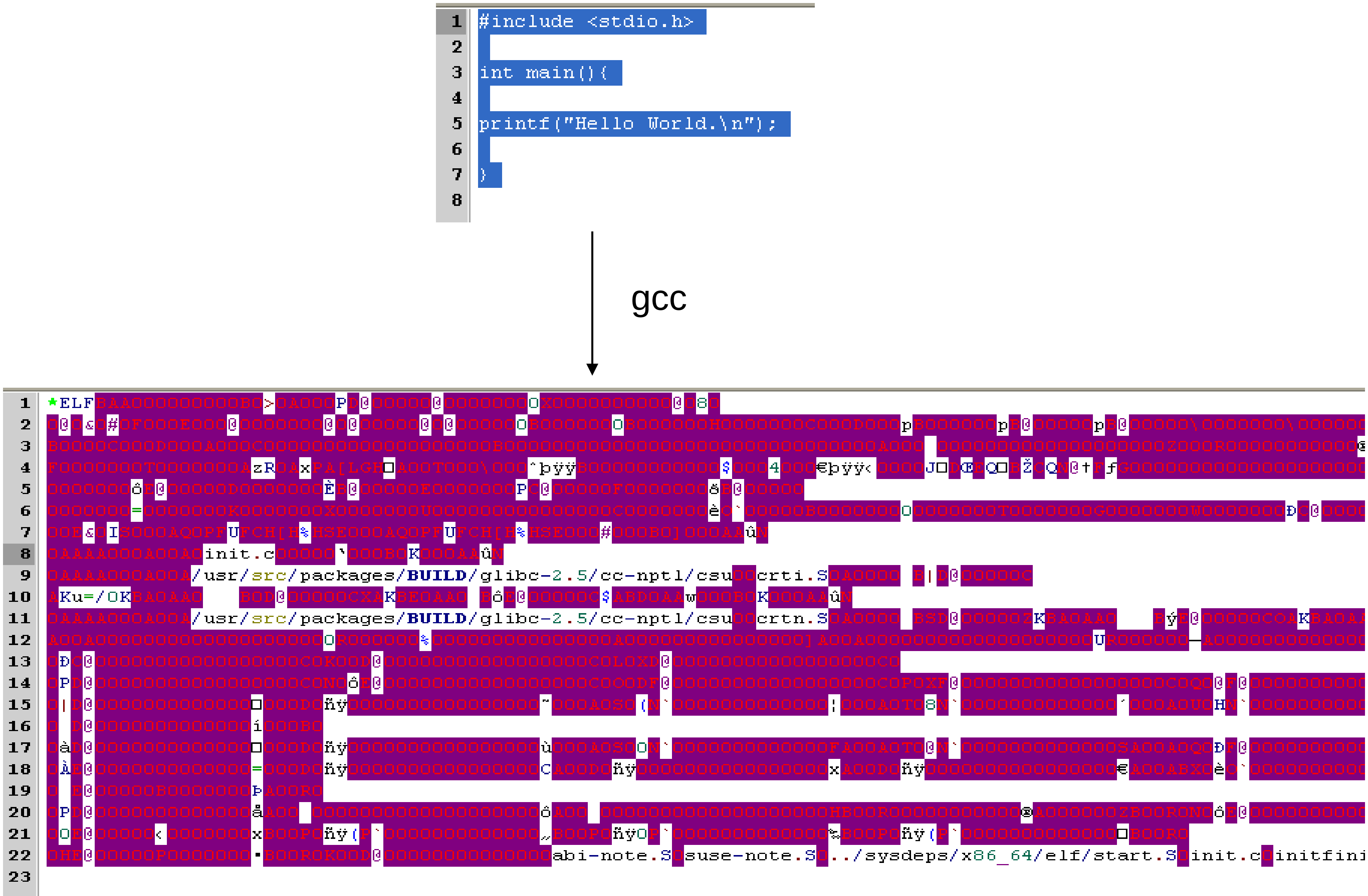
Digital Lockers vs Encryption Schemes

- Encryption guarantees no security unless key is uniform.
- Password-based encryption assume minimum entropy on the key space.
- DL do not assume anything about the password.
- DL protects the password.
- DL does not protect against dictionary attacks. It ensures that such attacks are the only ones possible.

Outline

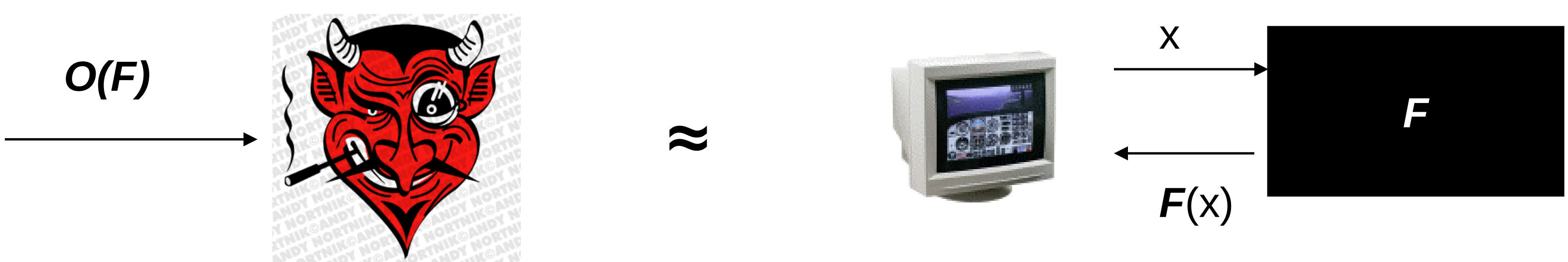
- Application: Digital Lockers
- Background
- Construction
- Analysis
- (Re)evaluating security

Obfuscation



Definition [B+01]

- ***O*** is an obfuscator for a family of functions, ***F*** if:
  - **Approximate functionality**: ***O(F)≈F***
  - **Polynomial slowdown:** Running time of ***O(F)*** is comparable to that of ***F*****.**
  - **Virtual Blackbox:** whatever can be computed from ***O(F)*** can be computed from the input/output of ***F.***



Virtual Blackbox

For any e± cient adversary A and any poly-nomial p, there exists an e± cient simulator S such that for any F 2 F and su± ciently large n:

$$jPr[b\,Ã\;A(O(F)) : b = 1]¡\;Pr[b\,Ã\;S^{F}(1^{n}) : b = 1]j$$

$$\cdot\quad \frac{1}{p(n)}:$$

Multibit Point Functions

A ***point function with multibit output*** outputs a long string on a single point and 0 everywhere else.

$$F_{x,y}(z) = \begin{cases} y & \text{if } z = x \\ 0 & \text{if } z \neq x \end{cases}$$

DL vs Point Function Obfuscation

- DL from obfuscation of multibit point functions:

$$lock(pass; content) = O(F_{pass;content})$$

- Next: Obfuscating multibit point functions…

Previous Results on Point Function Obfuscation

- Obfuscation of point functions is known [C97,CMR98,W05].

- [LPS04] has a Random Oracle obfuscation for multibit point functions (where *r* is uniform):

$$O^{R_1;R_2}(F_{x;y}) = r; R_1(x;r); R_2(x;r) \copyright y$$

- [FKSW05] has a multibit point function obfuscation for uniform x (G is a pseudorandom generator) :

$$O(F_{x;y}) = G(x) \copyright (0^n y)$$

- [W05] realizes [LPS04] construction for output with log length.

Outline

- Application: Digital Lockers
- Background
- Construction
- Analysis
- (Re)evaluating security

The Construction

- The class of functions:

$$\mathcal{F} = \{ F_{x,y} : x, y \in \{0,1\}^n \}$$

- The tool: An obfuscator, ***H***, for point functions (more about that later).

A ***point function*** outputs 1 on a single point and 0 everywhere else.

$$F_x(y) = \begin{cases} 1 & \text{if } y = x \\ 0 & \text{if } y \neq x \end{cases}$$

The First Attempt

$$
\begin{array}{lllllll}
y & = & & 1 & 0 & \ldots & 0 & 1 \\
 & & & \downarrow & \downarrow & \ldots & \downarrow & \downarrow \\
O(F_{x;y}) & = & H(F_x); & H(F_x); & H(F_{U_n}); & \ldots & H(F_{U_n}); & H(F_x) \\
O(F_{x;y}) & = & u_1; & u_2; & u_3; & \ldots & u_n; & u_{n+1}
\end{array}
$$

As such, this is just a string. The construction needs some processing code:

```
input    : a
constant : u1; u2; ...; un+1

if u1(a) = 0 then
    return 0;
else
    for i ← 2 to n + 1 do
        if ui(a) = 1 then
            yi ← 1;
        else
            yi ← 0;
        return y = y1; ...; yt;
    end
```

Outline

- Application: Digital Lockers
- Background
- Construction
- Analysis
- (Re)evaluating security

Analysis

- ***H*** has to be a probabilistic obfuscator.
- We would like to prove security based on this assumption only.
- However, this is not sufficient.
- ***H*** has to be secure under "composition" or concatenation.
- Example: $H(x;r_1);H(x;r_2)$ should not reveal x, if x is uniform.

On Noncomposable Obfuscators

- Suppose we have an obfuscator that looks like:

$$H(F_x; r = (r_1; r_2)) = H'(x; r_1); r_2; < x; r_2 > :$$

- Then, it is completely insecure under composition.

- ***x*** can be recovered by solving the linear system:

$$\begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_t \end{pmatrix} x = \begin{pmatrix} < x; r_1 > \\ < x; r_2 > \\ \vdots \\ < x; r_t > \end{pmatrix} :$$

- Applies also for obfuscator with auxiliary input [GK05]

A Composable Definition of Obfuscation

- Virtual Blackbox [LPS04]:

Original definition

For any efficient adversary A and any polynomial p, there exists an efficient simulator S such that for any $F \in \mathcal{F}$ and sufficiently large n:

$$\left|\Pr[b \leftarrow A(O(F)) : b = 1] - \Pr[b \leftarrow S^{F}(1^n) : b = 1]\right| \leq \frac{1}{p(n)}.$$

Composable definition

For any efficient adversary A and any polynomial p, there exists an efficient simulator S such that for any $F_1, \ldots, F_{t(n)} \in \mathcal{F}$ and sufficiently large n:

$$\left|\Pr[b \leftarrow A(O(F_1), \ldots, O(F_{t(n)})) : b = 1] - \Pr[b \leftarrow S^{F_1, \ldots, F_{t(n)}}(1^n) : b = 1]\right| \leq \frac{1}{p(n)}.$$

Analysis Based on Perfectly One-way functions

- We do not know if composable obfuscation of point functions exist.
- The closest primitive is ***Perfectly one-way*** (POW) function.
- We use statistical POW function in our construction to get obfuscation.
- We use computational POW function to get a weak version of obfuscation (when x and y are independent).

POW functions

- ***Secrecy***: A sequence of hashes of the *same* input is indistinguishable from a sequence of hashes of *independent and uniform strings*.

$H(x); \ldots; H(x)$

$\approx$

$H(U_n); \ldots; H(U_n)$

From Statistical POW Functions to Multibit Point Function Obfuscation

Theorem. If H is a statistical POW function then the construction is an obfuscation of multibit point functions.

Proof highlights:

- Given: For any high-entropy distribution, X, $H(X), \ldots, H(X)$ is statistically close to $W = (H(U_n), \ldots, H(U_n))$.
- Then, $O(F_{X,Y})$ is close to W.
- Then, for every machine and all but polynomially-many x (call this set L): $O(F_{x,y})$ is indistinguishable from W.
- We construct a simulator, S. S receives the "bad" L as advice string. If the oracle accepts $x \in L$, S runs the adversary, A, on $O(F_{x,y})$. Otherwise, it runs A on W.

From Computational POW Function

- The previous proof does not follow in the computational case.
- Why?
- Because *y* can depend on *x*.
  - Given: For any high-entropy distribution, X, $H(X), \ldots, H(X)$ is statistically close to $W = (H(U_n), \ldots, H(U_n))$.

We use the fact that statistical difference between two distributions does not increase by applying a function on them: $\Delta(f(A), f(B)) \leq \Delta(A, B)$

  - Then, $O(F_{X,Y})$ is close to W.
- The result holds if *y* is independent of *x*.

Summary

Assumption \ Construction	Composable Obfuscation	Obfuscation	Weak Obfuscation
Obfuscation of Point Functions [W05], [C97]	No	No	No
Computational POW functions [C97]	??	??	Yes
Statistical POW Functions [Unknown]	??	Yes	Yes
Composable Obfuscation of Point Functions [Unknown]	Yes	Yes	Yes

Outline

- Application: Digital Lockers
- Background
- Construction
- Analysis
- (Re)evaluating security

The Definition: Is It Sufficient?

- Problem: does not rule out constructions insecure on a small set of input
- Example: DL breaks on all English words!
- This is not surprising:
  - Running time of adversary and simulator are not tightly related.
- A general weakness in the definition of obfuscation
- Suggested Solution: number of queries the simulator makes is proportional to running time of A.
- Ongoing work…

More Formally

Def 3 (t-Virtual Blackbox)
For any efficient adversary A and any polynomial p, there exists an efficient simulator S such that for any $F \in \mathcal{F}$ and sufficiently large n:

$$\left|\Pr[b \leftarrow A(O(F)) : b = 1] - \Pr[b \leftarrow S^F(1^n) : b = 1]\right| \leq \frac{1}{p(n)};$$

where S makes at most $t(R_{A;F}; n; p)$ queries and $R_{A;F}$ is the worst-case running time of A on O(F), taken over the coin tosses of A and O.

t-secure DL → DL doesn't reveal content on, say, more than t(n) + t(n)/(n-1) passwords.

Questions???