

# Creating transformations for matrix obfuscation

Stephen Drape and Irina Voiculescu

Oxford University Computing Laboratory

August 2009

# What is Obfuscation?

An obfuscation is:

- a program transformation
- used to make programs 'harder to understand'
- a technique for protecting intellectual property
- **not** encryption

The area of obfuscation needs more research

## Two Important Papers

-  Christian Collberg, Clark Thomborson, and Douglas Low.  
A taxonomy of obfuscating transformations.  
Technical Report 148, Department of Computer Science, University of Auckland, July 1997.
-  Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang.  
On the (im)possibility of obfuscating programs.  
In Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology, pages 1–18. Springer-Verlag, 2001.

## Attack models and static analysis

- It is impossible to create obfuscations that could withstand **all** possible attacks (such as static analyses).
- Should aim to produce obfuscations designed to withstand certain attacks — this is the **attack model**.
- For example, restricting the usefulness of program slicing.
- Our attack model assumes that the attacker is trying to prove various program assertions.

# Obfuscation as data refinement

- **Localised obfuscation:**
  - obfuscate certain parts of the program, such as particular methods or variables.
- **Data refinement obfuscation:**
  - obfuscate an abstract data-type
  - (implicitly) obfuscate all the operations of the data-type
  - specify a set of assertions that the data-type operations satisfy (as part of the attack model)
  - obfuscation makes these assertions 'harder to prove'

## Obfuscation Equations

To obfuscate a data-type  $D$  into  $E$  using an obfuscation  $\mathcal{O}$ , we need an abstraction function  $af$ . For  $x :: D$  and  $y :: E$ :

$$x \rightsquigarrow y \iff (x = af(y)) \wedge dti(y)$$

An obfuscated function  $f^{\mathcal{O}}$  is correct with respect to  $f$  if it satisfies:

$$(\forall x :: D; y :: E) x \rightsquigarrow y \Rightarrow f(x) \rightsquigarrow f^{\mathcal{O}}(y)$$

We can rewrite this as

$$f \cdot af = af \cdot f^{\mathcal{O}}$$

# Obfuscating matrices

Matrices are essential in solving systems of equations, wavelets, graph theory, graphics.

The focus of this work is on obfuscating matrices through

- changing the matrix elements
- changing the pattern of the elements

Our work

- studies one established obfuscation
- develops a new obfuscation
- shows how matrices can be used to obfuscate numbers

## Matrix Data-Type

Matrix ( $\alpha$ )
$\text{scale} :: \alpha \rightarrow \text{Matrix } \alpha \rightarrow \text{Matrix } \alpha$ $\text{add} :: \text{Matrix } \alpha \times \text{Matrix } \alpha \rightarrow \text{Matrix } \alpha$ $\text{transpose} :: \text{Matrix } \alpha \rightarrow \text{Matrix } \alpha$ $\text{mult} :: \text{Matrix } \alpha \times \text{Matrix } \alpha \rightarrow \text{Matrix } \alpha$
$\text{transpose} \cdot \text{transpose} = \text{id}$ $\text{transpose} \cdot (\text{scale } \lambda) = (\text{scale } \lambda) \cdot \text{transpose}$ $\text{transpose} (\text{mult}(\mathbf{M}, \mathbf{N})) = \text{mult} (\text{transpose } \mathbf{N}, \text{transpose } \mathbf{M})$ $\text{add}(\mathbf{M}, \mathbf{N}) = \text{add}(\mathbf{N}, \mathbf{M})$

We will take Matrix ( $\alpha$ ) to be  $\mathbb{Q}^{r \times c}$

# Splitting Matrices

Suppose that we want to split a matrix  $\mathbf{M}^{r \times c}$  into  $n$  matrices, called the **split components**,

$$\mathbf{M} \rightsquigarrow \langle \mathbf{M}_0, \dots, \mathbf{M}_{n-1} \rangle_{sp}$$

where  $\mathbf{M}_i$  has size  $r_i \times c_i$  for  $i : [0..n)$ .

We can characterise a split  $sp$  by defining:

- a choice function  $ch$
- a family  $\mathcal{F}$  of injective functions where  $\mathcal{F} = \{f_t\}_{t:[0..n)}$

## Defining the splitting relationship

We define the relationship between  $\mathbf{M}$  and the split components element-wise:

$$\mathbf{M}_t(f_t(i, j)) = \mathbf{M}(i, j) \text{ where } t = ch(i, j)$$

The abstraction function for some split component  $\mathbf{M}_t$  is

$$af(\mathbf{M}_t(i, j)) = \mathbf{M}(f_t^{-1}(i, j))$$

where  $f_t^{-1} \cdot f_t = id$  (which is valid as  $f_t$  is injective).

## Example Split

We can define a split, called the  $(k \times k)$ -**square split** (denoted by  $s_k$ ) by ensuring that the first component is a  $k \times k$  matrix.

$$\left( \begin{array}{ccc|ccc} a_{(0,0)} & \dots & a_{(0,k-1)} & a_{(0,k)} & \dots & a_{(0,n-1)} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{(k-1,0)} & \dots & a_{(k-1,k-1)} & a_{(k-1,k)} & \dots & a_{(k-1,n-1)} \\ \hline a_{(k,0)} & \dots & a_{(k,k-1)} & a_{(k,k)} & \dots & a_{(k,n-1)} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{(n-1,0)} & \dots & a_{(n-1,k-1)} & a_{(n-1,k)} & \dots & a_{(n-1,n-1)} \end{array} \right)$$

We will show the definitions of our matrix operations for this split.

## Operations for split matrices

If  $\mathbf{M} \rightsquigarrow \langle \mathbf{M}_0, \dots, \mathbf{M}_3 \rangle_{s_k}$  and  $\mathbf{N} \rightsquigarrow \langle \mathbf{N}_0, \dots, \mathbf{N}_3 \rangle_{s_k}$  then

$$\text{scale } \lambda \mathbf{M} \rightsquigarrow \langle \text{scale } \lambda \mathbf{M}_0, \dots, \text{scale } \lambda \mathbf{M}_3 \rangle_{s_k}$$

$$\text{add } (\mathbf{M}, \mathbf{N}) \rightsquigarrow \langle \text{add } (\mathbf{M}_0, \mathbf{N}_0), \dots, \text{add } (\mathbf{M}_3, \mathbf{N}_3) \rangle_{s_k}$$

$$\mathbf{M}^T \rightsquigarrow \langle \mathbf{M}_0^T, \mathbf{M}_2^T, \mathbf{M}_1^T, \mathbf{M}_3^T \rangle_{s_k}$$

$$\mathbf{M} \times \mathbf{N} \rightsquigarrow \langle (\mathbf{M}_0 \times \mathbf{N}_0) + (\mathbf{M}_1 \times \mathbf{N}_2), \\ (\mathbf{M}_0 \times \mathbf{N}_1) + (\mathbf{M}_1 \times \mathbf{N}_3), \\ (\mathbf{M}_2 \times \mathbf{N}_0) + (\mathbf{M}_3 \times \mathbf{N}_2), \\ (\mathbf{M}_2 \times \mathbf{N}_1) + (\mathbf{M}_3 \times \mathbf{N}_3) \rangle_{s_k}$$

## Review of Splitting

- The obfuscated operations have a similar complexity to the original versions.
- We can use matrix splitting as an array obfuscation.
- It is hard to define **determinants** and **inverses** for split matrices.
- The obfuscation changes the shape of the matrices but not the actual matrix values.

# Polynomial bases

Matrix **A**

$$\begin{array}{c|ccc} & 1 & y & y^2 \\ \hline 1 & 1 & 0 & 2 \\ x & 0 & 1 & 3 \\ x^2 & 0 & 0 & 1 \end{array}$$

stores  $p(x, y) = x^2y^2 + 3xy^2 + 2y^2 + xy + 1$

$$p(x, y) = (1 \quad x \quad x^2) \begin{pmatrix} 1 & 0 & 2 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ y \\ y^2 \end{pmatrix} = \mathbf{XAY}$$

## Bernstein bases

Can change the basis of the linear transformation using Bernstein polynomials  $B_k^n(x) = \binom{n}{k} x^k (1-x)^{n-k}$ ,  $\forall x \in [0, 1]$ ,  $k = 0, \dots, n$ .

$$\begin{aligned}
 p_{\mathcal{B}}(x, y) &= (B_0^2(x) \quad B_1^2(x) \quad B_2^2(x)) \quad \mathbf{C} \quad \begin{pmatrix} B_0^2(y) \\ B_1^2(y) \\ B_2^2(y) \end{pmatrix} \\
 &= (1 \quad x \quad x^2) \begin{pmatrix} 1 & 0 & 0 \\ -2 & 2 & 0 \\ 1 & -2 & 1 \end{pmatrix} \quad \mathbf{C} \quad \begin{pmatrix} 1 & -2 & 1 \\ 0 & 2 & -2 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ y \\ y^2 \end{pmatrix} \\
 &= \quad \mathbf{X} \quad \quad \mathbf{U}_2 \quad \quad \mathbf{C} \quad \quad \mathbf{V}_2 \quad \quad \mathbf{Y}
 \end{aligned}$$

## Bernstein obfuscation of matrices

Since

$$\mathbf{XAY} = p(x, y) = p_{\mathcal{B}}(x, y) = \mathbf{XU}_2\mathbf{CV}_2\mathbf{Y}$$

the Bernstein-form matrix corresponding to  $\mathbf{A} = \begin{pmatrix} 1 & 0 & 2 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{pmatrix}$  is

$$\mathbf{C} = \mathbf{U}_2^{-1}\mathbf{AV}_2^{-1} = \begin{pmatrix} 1 & 1 & 3 \\ 1 & \frac{5}{4} & 5 \\ 1 & \frac{3}{2} & 8 \end{pmatrix}$$

and can be used as an obfuscation of  $\mathbf{A}$ .

## Operations for the Bernstein Obfuscation

For a matrix  $\mathbf{S}$  with  $\dim(\mathbf{S}) = (a + 1, b + 1)$ , and a matrix  $\mathbf{T}$ , it can be proved that

$$\text{scale}_{\mathcal{B}} \lambda \mathbf{S} = \text{scale } \lambda \mathbf{S}$$

$$\text{add}_{\mathcal{B}}(\mathbf{S}, \mathbf{T}) = \mathbf{S} + \mathbf{T}$$

$$\text{transpose}_{\mathcal{B}}(\mathbf{S}) = \mathbf{U}_b^{-1} \mathbf{V}_b^T \mathbf{S}^T \mathbf{U}_a^T \mathbf{V}_a^{-1}$$

$$\text{mult}_{\mathcal{B}}(\mathbf{S}, \mathbf{T}) = \mathbf{S} \mathbf{V}_b \mathbf{U}_b \mathbf{T}$$

$$\text{inverse}_{\mathcal{B}}(\mathbf{S}) = \mathbf{U}_a^{-1} \mathbf{V}_a^{-1} \mathbf{S}^{-1} \mathbf{U}_a^{-1} \mathbf{V}_a^{-1}$$

$$\det_{\mathcal{B}}(\mathbf{S}) = \det(\mathbf{U}_a) \times \det(\mathbf{S}) \times \det(\mathbf{V}_b)$$

## Review of the Bernstein obfuscation

- The obfuscated matrices have different structure.
- Determinants and inverses of matrices can be computed easily.
- The scaling and addition operations are **not** obfuscated.
- **Any** change of basis transformation would be suitable.
- We can **generalise** to more than two dimensions, or to arrays.
- There is an extra cost in computational complexity.

## Using matrices to obfuscate numbers

We can use matrices to obfuscate other data-types.

To obfuscate a **rational number data-type** which contains the operations of **addition**, **multiplication** and **reciprocal**.

We need matrix operations plus, times and recip such that, for rational numbers  $n$  and  $p$ , if  $n \rightsquigarrow \mathbf{A}$  and  $p \rightsquigarrow \mathbf{C}$  then

$$n + p \rightsquigarrow \text{plus}(\mathbf{A}, \mathbf{C}) \quad n \times p \rightsquigarrow \text{times}(\mathbf{A}, \mathbf{C}) \quad n^{-1} \rightsquigarrow \text{recip}(\mathbf{A})$$

## Using determinants

- There are many ways in which to use matrices for the obfuscation of rational numbers.
- We show an obfuscation whose abstraction function is the determinant.
- We choose to obfuscate a number  $n$  by a  $2 \times 2$  matrix which has 1 and  $n$  as eigenvalues.

Here is one possible transformation (where  $a \in \mathbb{Q}$ ):

$$n \rightsquigarrow \begin{pmatrix} a & b \\ \frac{(a-1)(n-a)}{b} & n+1-a \end{pmatrix} \text{ where } b \neq 0$$

## Obfuscated number operations

$$\text{plus}\left(\begin{pmatrix} a & b \\ c & d \end{pmatrix}, \begin{pmatrix} e & f \\ g & h \end{pmatrix}\right) = \begin{pmatrix} a + e - 1 & bf \\ \frac{(a+e-2)(d+h-1)}{bf} & d + h \end{pmatrix}$$

$$\text{times}\left(\begin{pmatrix} a & b \\ c & d \end{pmatrix}, \begin{pmatrix} e & f \\ g & h \end{pmatrix}\right) = \begin{pmatrix} (a+d)(e+h) + 1 & bf \\ \frac{-(a+d)(e+h)(a+d+e+h)}{bf} & 1 - a - d - e - h \end{pmatrix}$$

$$\text{recip}\left(\begin{pmatrix} a & b \\ c & d \end{pmatrix}\right) = \begin{pmatrix} \frac{d}{a+d-1} & b \\ \frac{(a-1)(d-1)}{b(a+d-1)^2} & \frac{a}{a+d-1} \end{pmatrix}$$

## Evaluation of techniques

- We have considered obfuscations at an appropriate level of abstraction: we are not concerned with implementation details.
- Our obfuscations can be evaluated against the assertion attack model: our obfuscations make the assertions harder to prove.
- To understand our matrix obfuscations, a human attacker needs, e.g., familiarity with the Bernstein basis and with the relationship between bi-variate polynomials and matrices.
- We can combine obfuscations to create more complicated obfuscations.

## Conclusions

- The Bernstein obfuscation allows us to devise obfuscations for a wider variety of matrix operations than matrix splitting did.
- However the Bernstein obfuscation can adversely affect the complexity of the matrix operations.
- There is usually a trade-off between the quality of the obfuscations and their complexity.
- Considering obfuscations at an abstract level creates the opportunity to devise more general obfuscations.