Obfuscating simple functionalities from knowledge assumptions

Ward Beullens Hoeteck Wee

June 4, 2019

Simple VBB obfuscation with the KOALA

Ward Beullens

Hoeteck Wee

Simple schemes. We only use cyclic groups, no lattices, multi-linear maps,...

Simple functionalities. Point functions, pattern matching (i.e. conjunctions)

Simple proofs from KOALA, a new knowledge assumption.

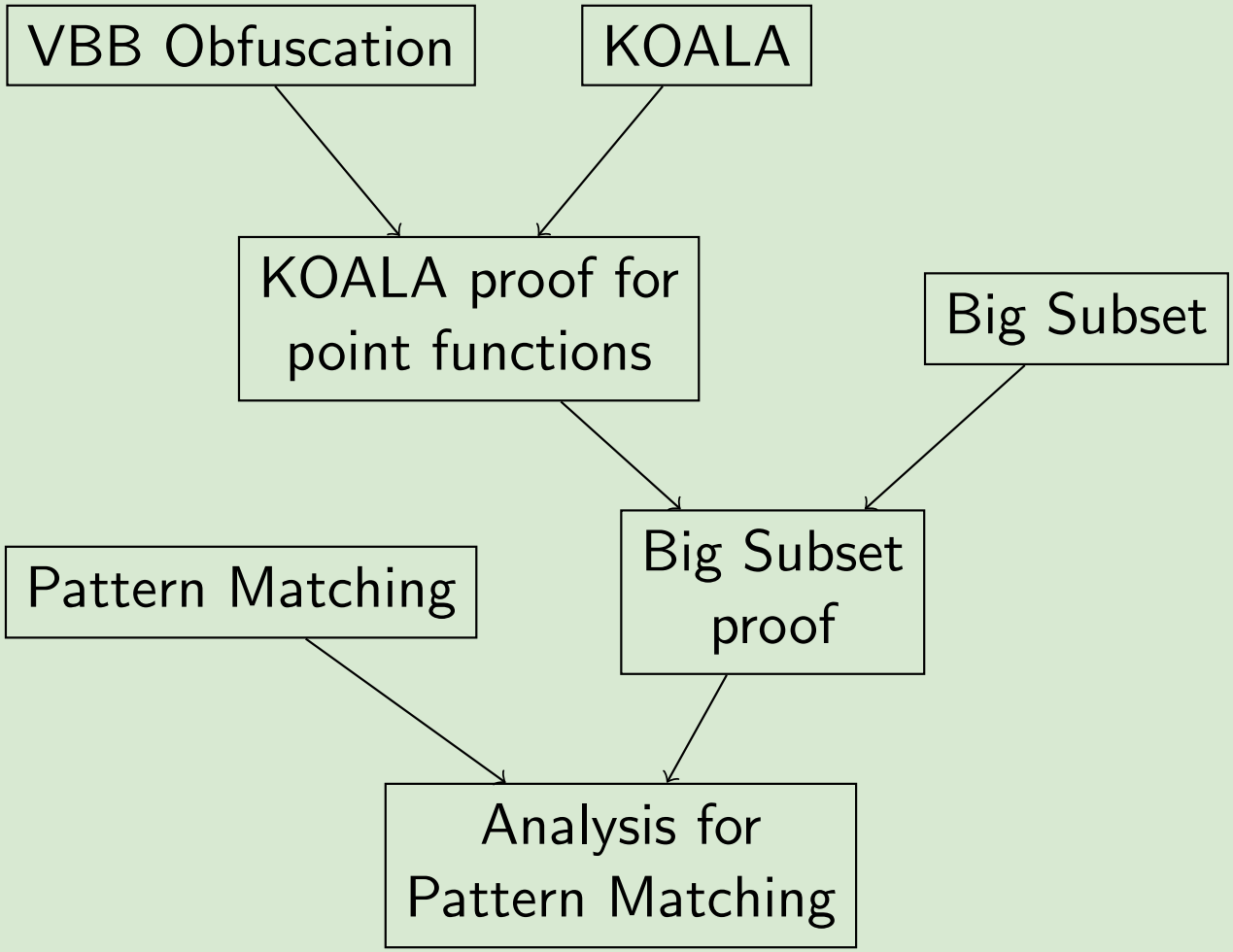
New knowledge assumption KOALA

- Implied by GGM
- Simplifies a lot of VBB proofs

Analysis of an obfuscator for pattern matching (aka conjunctions)

- Modular: through Big Subset functionality.
- Better guarantees: VBB instead of d-VBB
- Weaker assumption: KOALA instead of GGM

Outline of the talk



Outline of the talk

VBB Obfuscation

VBB Obfuscation

Virtual Black Box (VBB) security definitions:

- VBB : Everything that you can learn from an obfuscation you can learn from black box access.

  Simulator based definition
  Very strong (impossible for general circuits)

- Distributional VBB: Like VBB, but functions are drawn from some fixed distribution.

- T-VBB: Everything that you can learn from an obfuscation you can learn from black box access in time $O(T)$+poly.

Outline of the talk

VBB Obfuscation

KOALA

KOALA

For a subspace $V \subset \mathbb{F}_p^n$, when can we distinguish $[\mathbf{v}] = (g^{v_1}, \cdots, g^{v_n})$ for $\mathbf{v} \xleftarrow{\$} V$ from n random group elements?

If we know $\mathbf{u} \in V^\perp$, with $\mathbf{u} \neq 0$, then distinguishing is easy: just check if $\mathbf{u} \cdot [\mathbf{v}] := \prod_i [v_i]^{u_i} = [\mathbf{u} \cdot \mathbf{v}] = [0]$.

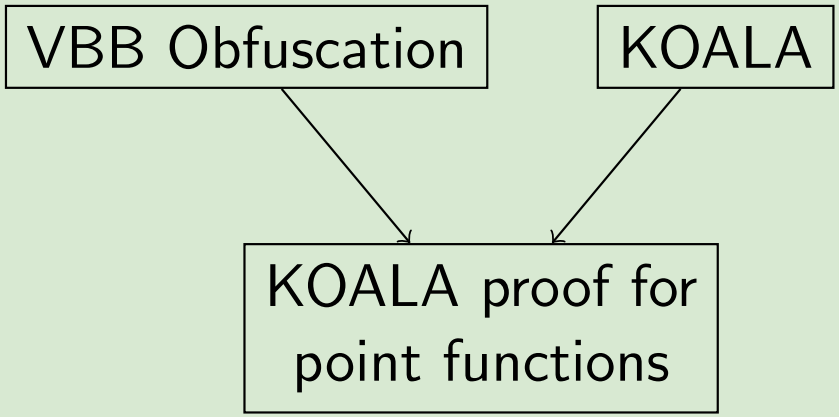
Knowledge of OrthogonALity Assumption (KOALA)

A cyclic group G satisfies KOALA if an adversary can only distinguish $[\mathbf{v}]$ for $\mathbf{v} \xleftarrow{\$} V$ from random if he can also produce $\mathbf{u} \in V^\perp \setminus \{0\}$.

Theorem (GGM $\Rightarrow$ KOALA)

In the generic group model, cyclic groups satisfy KOALA

Outline of the talk



Showing off the KOALA

Point function $p_x(y) = \begin{cases} 1 \text{ if } y = x \\ 0 \text{ otherwise} \end{cases}$

Point function obfuscator [C98]:

Obfuscator: $\mathcal{O}(f_x) = [r, -xr]$ for random $r \in \mathbb{F}_p$
Evaluation at y: check if $(y, 1) \cdot [r, -xr] = [0]$.

Sketch of VBB security proof assuming KOALA: Given $\mathcal{A}$, we need to simulate $\mathcal{A}(\mathcal{O}(f_x))$, with only black box access to f_x.

A If $\mathcal{A}$ does not distinguish $[\langle (1, -x) \rangle]$ from random, then simulator outputs $\mathcal{A}([r_1, r_2])$.

B Otherwise, use the KOALA to extract nonzero $(a, b) \perp (1, -x)$ and let simulator output $\mathcal{A}(\mathcal{O}(f_{\frac{a}{b}}))$.

Self composeability from KOALA

Definition (informal)

An obfuscator $\mathcal{O}$ for function family $\mathcal{F}$ is k-self composeable if $(\mathcal{O}(f_i), \cdots, \mathcal{O}(f_k))$ reveals nothing more than k black box oracles for $f_1, \cdots, f_k$.

Example

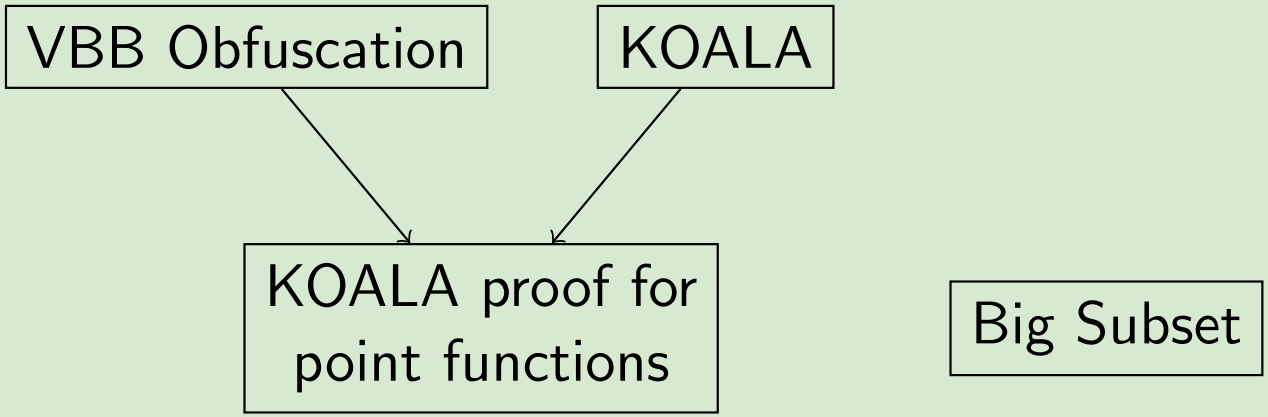
$\mathcal{O}(f_x) = \mathcal{H}(x)$ is VBB secure obfuscator in ROM, but it is not 2-self composeable because $\mathcal{O}(f_{x_1}), \mathcal{O}(f_{x_2})$ leaks whether $x_1 = x_2$.

Theorem

For any $k > 0$, the point function obfuscator of [C98] is k-self composeable.

Proof: "Random or learn" with KOALA

Outline of the talk



Big Subset Functionality

Given a subset $Y \subset \{1, \cdots, n\}$ and a treshold t, define the function

$$f_{Y,t} : P(\{1, \cdots, n\}) \to \{0, 1\} : X \mapsto \begin{cases} 1 \text{ if } |X| \geq t \text{ and } X \subset Y \\ 0 \text{ otherwise} \end{cases}$$

Example

$$t = 4, Y = \blacksquare\,\square\,\square\,\blacksquare\,\blacksquare\,\square\,\blacksquare\blacksquare$$

$$f_{Y,t}(\square\,\square\,\square\,\blacksquare\,\blacksquare\,\square\,\blacksquare\blacksquare) = 1$$

$$f_{Y,t}(\square\,\blacksquare\,\square\,\blacksquare\,\blacksquare\,\square\,\blacksquare\blacksquare) = 0$$

$$f_{Y,t}(\blacksquare\,\square\,\square\,\blacksquare\,\square\,\square\,\blacksquare\square) = 0$$

Obfuscating big subset

Obfuscate $f_{Y,t}$: Pick polynomial $f \in \mathbb{F}_q[x]$ of degree $t-1$ with $f(0) = 0$. Obfuscation is $[\mathbf{v}]$ with

$$v_i = \begin{cases} f(i) & \text{if } i \in Y \\ \text{random} & \text{if } i \notin Y \end{cases}$$

Evaluate obfuscated program at $X \subset \{1, \cdots, n\}$**.**
Compute interpolation coefficients:

$$u_i = \begin{cases} \prod_{j \in X \setminus \{i\}} \frac{j}{i-j} & \text{if } i \in X \\ 0 & \text{if } i \notin X \end{cases},$$

and check if $\mathbf{u} \cdot [\mathbf{v}] = [0]$. If $f_{Y,t}(X) = 1$, then $\mathbf{u} \cdot [\mathbf{v}] = [f(0)] = [0]$, otherwise $\mathbf{u} \cdot [\mathbf{v}]$ is uniformly random.

Obfuscating big subset: Example

Let $Y = \square \quad \blacksquare \quad \blacksquare \quad \blacksquare \quad , \quad t = 2$

Obfuscate $f_{Y,t}$: pick f of degree 1 with $f(0) = 0$:

$$f(x) = ax$$

$$\mathcal{O}(f_{Y,t}) = [(r, 2a, 3a, 4a)]$$

Evaluate at $X = \square \quad \blacksquare \quad \blacksquare \quad \square$:

$$\mathbf{u} = (0, \frac{3}{2-3}, \frac{2}{3-2}, 0) = (0, -3, 2, 0)$$

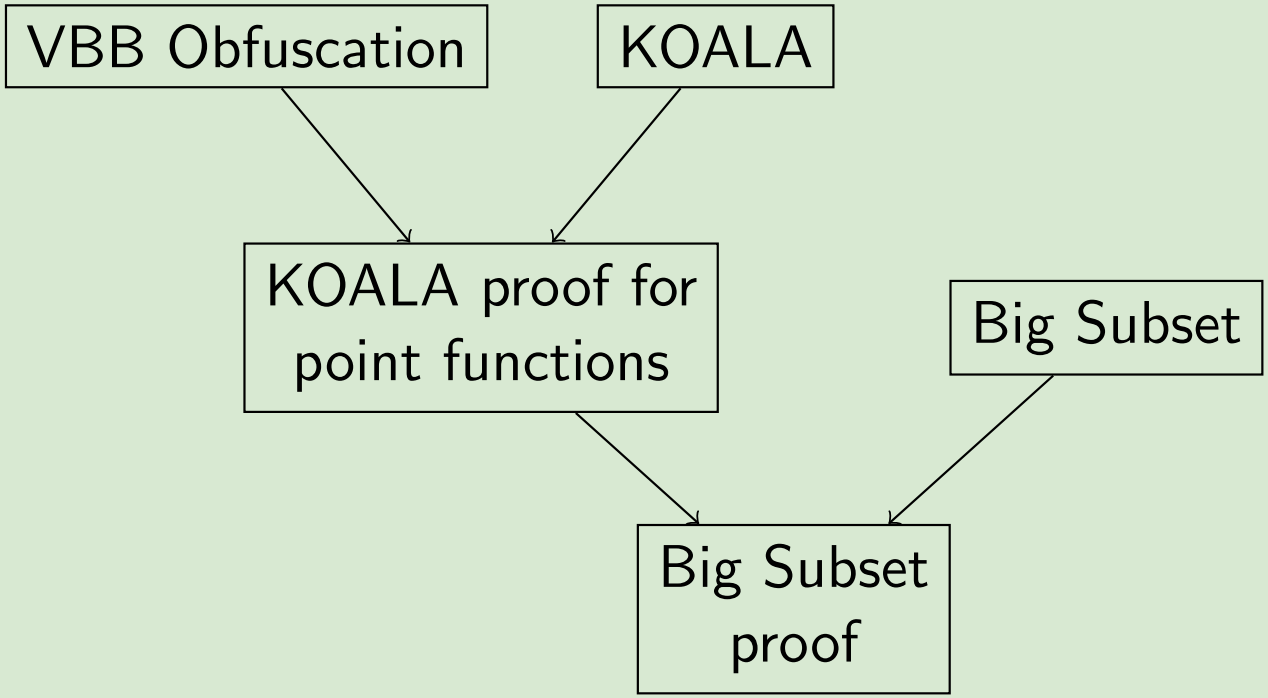
$$\mathbf{u} \cdot [\mathbf{v}] = [-3 * 2a + 3 * 2a] = [0]$$

Evaluate at $X = \blacksquare \quad \blacksquare \quad \square \quad \square$:

$$\mathbf{u} = (\frac{2}{1-2}, \frac{1}{2-1}, 0, 0) = (-2, 1, 0, 0)$$

$$\mathbf{u} \cdot [\mathbf{v}] = [-2r + 2a] \neq [0]$$

Outline of the talk



KOALA proof for Big Subset Obfuscator

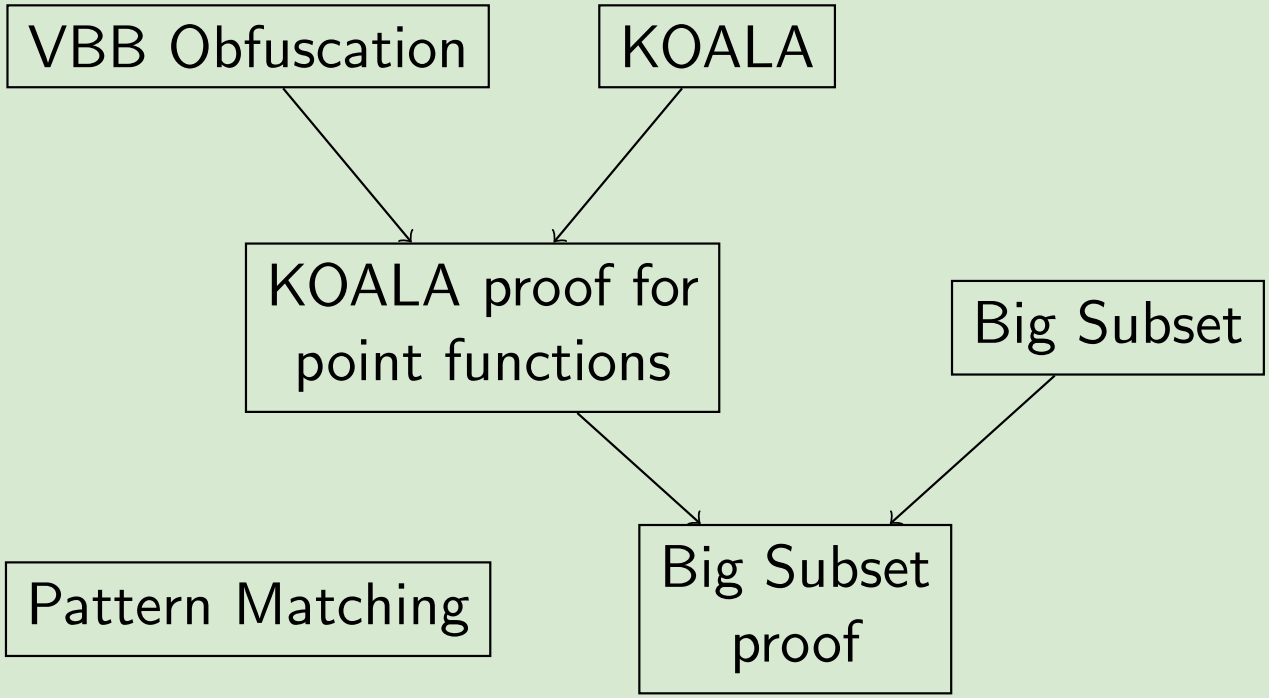
Sketch of VBB security proof assuming KOALA: Given $\mathcal{A}$, we need to simulate $\mathcal{A}(\mathcal{O}(f_{Y,t}))$, with only black box access to $f_{Y,t}$.
Obfuscations of $f_{Y,t}$ are uniformly random in a subspace V, where $V = RS \times \mathbb{F}_q^{n-|Y|}$

A If $\mathcal{A}$ does not distinguish $[V]$ from random, then simulator outputs $\mathcal{A}([\text{random vector}])$.

B Otherwise, use the KOALA to extract nonzero $u \perp V$. Support of $u \subset Y$, and u is a nonzero codeword, so it has at least t nonzero entries.
$\Rightarrow$ We recover a subset of Y of size $\geq t$.
Recover Y, t from black box access.
Output $\mathcal{A}(\mathcal{O}(f_{Y,t}))$.

k-self composeability holds too.

Outline of the talk



Pattern matching

We study [BKMPRS18] obfuscator for pattern matching with wildcards.

Pattern matching with wildcards

Given a pattern $\pi \in \{0,1,\star\}^n$, there is a pattern matching function

$$f_\pi : \{0,1\}^n \to \{0,1\}$$

$$: \quad \mathbf{x} \quad \mapsto \begin{cases} 1 \text{ if } \forall i \quad x_i = \pi_i \text{ or } \pi_i = \star \\ 0 \text{ otherwise} \end{cases}$$

Example

$$\pi = 01 \star 00$$

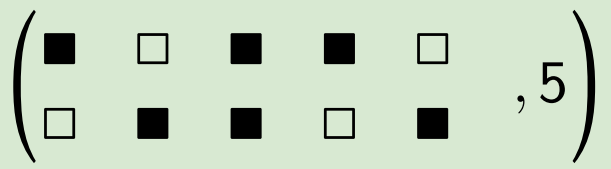
$$f_\pi(01100) = 1$$

$$f_\pi(00000) = 0\,.$$

Pattern Matching embeds into Big Subset

Pattern Matching	$\to$	Big subset
n	$\mapsto$	$2n$
$\pi \in \{0,1,\star\}^n$	$\mapsto$	$(Y_\pi = \{1 + 2 * i + j \mid \pi_i = \star \text{ or } \pi_i = j\}, n)$
$\mathbf{x} \in \{0,1\}^n$	$\mapsto$	$X_\mathbf{x} = \{1 + 2 * i + j \mid x_i = j\}$
$f_\pi(\mathbf{x}) = 1$	$\Leftrightarrow$	$f_{Y_\pi,n}(X_\mathbf{x}) = 1$

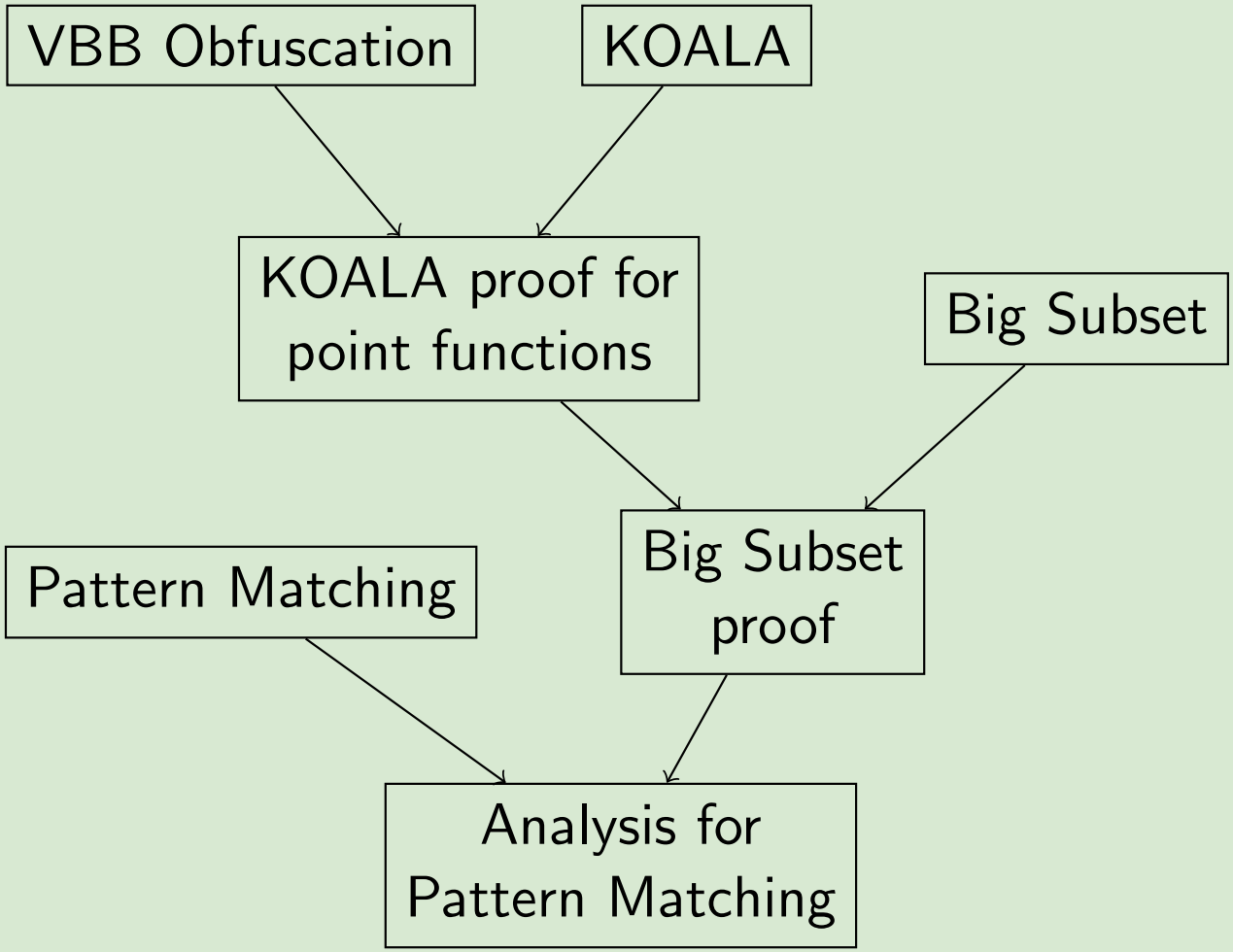
Example

$01 \star 01 \mapsto$ 

$01101 \mapsto$ 

Big subset obfuscator + embedding = [BKMPRS18] obfuscator.

Outline of the talk



Analyzing BKMPRS18 Obfuscator

From KOALA, we were able to prove a lot of security guarantees, and we also gave matching attacks to prove that they are optimal.

Crypto '18 Obfuscator is:	**Crypto '18 Obfuscator is not:**
$2^{n/2}$-VBB secure	$2^{n/2-\epsilon}$-VBB secure
D-VBB secure for any distribution with min entropy $n + \omega(\log n)$ [1]	D-VBB secure for some dist. with min entropy $n - \omega(\log n)$
VBB secure if number of wildcards is $O(\log n)$	VBB secure if number of wildcards is $\omega(\log n)$.
...	...

[1] Also in independent work by J Bartusek, T Lepoint, F Ma, M Zhandry

Conclusion

In our work we:

- Introduce KOALA, which simplifies a lot of GGM proofs,
- prove security guarantees for BKMPRS18 obfuscator,
- give attacks to show our security guarantees are optimal.

Open problems:

- Construct simple (i.e. relying on cyclic groups) obfuscator for pattern matching with better security.
  (e.g. $2^{\epsilon n}$-VBB security)
- Construct simple obfuscators for other functionalities