APPLICATIONS OF SOFTWARE OBFUSCATION

Mark Zhandry – Stanford University

*Joint work with Dan Boneh
Program Obfuscation

Intuition: Scramble a program
- Preserve functionality
- Hide implementation details

Applications:
- IP Protection
- Software Watermarking
- Crypto
Indist. Obfuscation (iO) \([BGI^{+01}, GR'07]\)

If two programs have same functionality, obfuscations are indistinguishable

\[ P_1(x) = P_2(x) \quad \forall x \]

Big questions: How to build? How to use?
Indistinguishability Obfuscation (iO)

An exploding field:

- **[GGH⁺’13] First candidate iO construction**
  - Built from multilinear maps
  - First application: functional encryption
- **[BR’13, BGK⁺’13, …] Additional constructions**
- **[SW’13, GGHR’13, BZ’13, ABGSZ’13, …] Uses**
  - Public key encryption, signatures, deniable encryption, multiparty key exchange, MPC, …
- **[BCPR’13, MR’13, BCP’13, …] Further Investigation**
Our Results

Non-interactive multiparty key exchange
• First scheme without trusted setup

Efficient broadcast encryption
• Constant size ciphertext and secret keys
• First distributed system: users generate keys themselves

Efficient traitor tracing
• Shortest secret keys, ciphertexts, known
• Resolves open problem in Differential Privacy [DNR⁺09]
MULTIPARTY KEY EXCHANGE
(Non-Interactive) Multiparty Key Exchange

Public bulletin board

$K_{ABCD}$ $K_{ABCD}$ $K_{ABCD}$ $K_{ABCD}$

$S_A$ $S_B$ $S_C$ $S_D$
History

2 parties: Diffie Hellman Protocol [DH’76]

3 parties: Bilinear maps [Joux’2000]

n>3 parties: Multilinear maps [BS’03, GGH’13, CLT’13]
- Requires trusted setup phase

Our work: n parties, no trusted setup
Prior Constructions for $n>3$

First achieved using multilinear maps [GGH'13, CLT'13]

- These constructions all require trusted setup before protocol is run
- Trusted authority can also learn group key
Starting point for our construction

Building blocks:
- One-way function $G:S \rightarrow X$
- Pseudorandom function (PRF) $F$

Shared key: $F_k(x_1, x_2, x_3, x_4) \leftarrow$ how to compute securely?
Introduce Trusted Authority (for now)

\[ k \] (\( y_1, ..., y_n, s, i \)) \{
    If G(s) \neq y_i, output \( \bot \)
    Otherwise, output \( F_k(y_1, ..., y_n) \)
\}
First attempt

\[ K_{ABCD} = P_{iO}(x_1, x_2, x_3, x_4, s_1, 1) \]

Problems:
- \( k \) not guaranteed to be hidden using iO
- Still have trusted authority
Removing Trusted Setup

As described, our scheme needs trusted setup

Observation: Obfuscated program can be generated independently of publishing step

Untrusted setup: designate user 1 as “master party”
- generates $P_{iO}$, sends with $x_1$
Multiparty Key Exchange Without Trusted Setup

Security equivalent to security of previous scheme
Hiding $k$

Follow “punctured program” paradigm of SW’13

- Use pseudorandom generator for $G$

  $G: S \rightarrow X \quad |X| \gg |S|$

  $G(s), \; s \leftarrow S \; \text{indist. from } \; x \leftarrow X$

- Use special “punctured PRF” for $F$  [BW’13, KPTZ’13, BGI’13, SW’13]

  Punctured key $k^z \Rightarrow \text{compute } F_{k}(\cdot \cdot) \text{ everywhere but } z$

  $k^z$

  $x \xrightarrow{F} F(k,x)$ if $x \neq z$

  $\perp$ if $x = z$

Security: given $k^z$, cannot compute $t = F_{k}(z)$

Construction: GGM’84
Security of Our Construction

\[ k \quad P( y_1, ..., y_n, s, i ) \{ \]
\[ \quad \text{If } G(s) \neq y_i, \]
\[ \quad \quad \text{output } \bot \]
\[ \quad \text{Otherwise,} \]
\[ \quad \quad \text{output } F_k(y_1, ..., y_n ) \]
\[ \} \]

Adversary’s goal:
Learn \( F_k(x_1, ..., x_n) \)
Step 1: Replace $x_i$

Real World

$$P( y_1, ..., y_n, s, i ) \{$$
$$\text{If } G(s) \neq y_i, \quad \text{output } \bot$$
$$\text{Otherwise,} \quad \text{output } F_k(y_1, ..., y_n)$$
$$\}$$

Alternate World 1

$$P( y_1, ..., y_n, s, i ) \{$$
$$\text{If } G(s) \neq y_i, \quad \text{output } \bot$$
$$\text{Otherwise,} \quad \text{output } F_k(y_1, ..., y_n)$$
$$\}$$

Security of $G \Rightarrow$ words indistinguishable
Step 1: Replace $x_i$

Observation:
Since $|X| \gg |S|$, w.h.p. no $s, i$ s.t. $G(s) = x_i$

Never pass check when $y_1, ..., y_n = x_1, ..., x_n$
Step 2: Puncture

Alternate World 2

\[ k^z \]

\[ P( y_1, \ldots, y_n, s, i ) \{ \]
\[ \text{If } G(s) \neq y_i, \]
\[ \text{output } \bot \]
\[ \text{If } (y_1, \ldots, y_n) = z, \]
\[ \text{output } \bot \]
\[ \text{Otherwise,} \]
\[ \text{output } F_k(y_1, \ldots, y_n) \]
\[ } \]

Alternate World 1

\[ k \]

\[ P( y_1, \ldots, y_n, s, i ) \{ \]
\[ \text{If } G(s) \neq y_i, \]
\[ \text{output } \bot \]
\[ \text{Otherwise,} \]
\[ \text{output } F_k(y_1, \ldots, y_n) \]
\[ } \]

Let \( z = (x_1, \ldots, x_n) \)

W.h.p. programs identical + iO ⇒ Worlds indistinguishable
Security

Alternate World 2

\[ P( y_1, \ldots, y_n, s, i ) \{ \]
\[ \text{If } G(s) \neq y_i, \]
\[ \text{output } \bot \]
\[ \text{If } (y_1, \ldots, y_n) = z, \]
\[ \text{output } \bot \]
\[ \text{Otherwise,} \]
\[ \text{output } F_k(y_1, \ldots, y_n) \]
\[ \} \]

Adversary’s goal: learn \( F_k(z) \)

Success in Real World \( \Rightarrow \) success in World 2

In World 2:

Adversary only sees \( k^z \)
\( \Rightarrow \) cannot learn \( F_k(z) \)

Let \( z=(x_1, \ldots, x_n) \)
Conclusion

Exciting time to study crypto

Future work:
• What else can we do with obfuscation?
• Bring obfuscation closer to practice

Thanks!