Constraining Pseudorandom Functions Privately

David Wu
Stanford University

Joint work with Dan Boneh and Kevin Lewi
Pseudorandom Functions (PRFs) [GGM84]

\[ F : \mathcal{K} \times \mathcal{X} \rightarrow \mathcal{Y} \]
Constrained PRFs [BW13, BGI13, KPTZ13]

Constrained PRF: PRF with additional “constrain” functionality

\[ F: \mathcal{K} \times \mathcal{X} \to \mathcal{Y} \]

- PRF key
- Constrain \(_C\)
- Constrained key

Can be used to evaluate at all points \(x \in \mathcal{X}\) where \(C(x) = 1\)
Constrained PRFs [BW13, BGI13, KPTZ13]

**Correctness:** constrained evaluation at $x \in \mathcal{X}$ where $C(x) = 1$ yields PRF value at $x$

**Security:** PRF value at points $x \in \mathcal{X}$ where $C(x) = 0$ are indistinguishable from random
Constrained PRFs [BW13, BGI13, KPTZ13]

Many applications:
- Identity-Based Key Exchange, Optimal Broadcast Encryption [BW13]
- Punctured Programming Paradigm [SW14]
- Multiparty Key Exchange, Traitor Tracing [BZ14]
Puncturable PRFs from GGM

- Puncturable PRF: constrained keys allow evaluation at *all* but a single point
- Easily constructed from GGM:

\[ G(s) = s_0 \parallel s_1 \]

\[ G(s_0) = s_{00} \parallel s_{01} \]

\[ G(s_1) = s_{10} \parallel s_{11} \]
Puncturable PRFs from GGM

\[ G(s) = s_0 \parallel s_1 \]

\[ G(s_0) = s_{00} \parallel s_{01} \]

\[ G(s_1) = s_{10} \parallel s_{11} \]

Given root key \( s \), can evaluate PRF everywhere.
Puncturable PRFs from GGM

\[
G(s) = s_0 \parallel s_1
\]

\[
G(s_0) = s_{00} \parallel s_{01}
\]

\[
G(s_1) = s_{10} \parallel s_{11}
\]

puncture at \( x = 01 \)
Puncturable PRFs from GGM

\[ G(s) = s_0 \parallel s_1 \]

These two values suffice to evaluate at all other points.
Puncturable PRFs from GGM

\[ G(s) = s_0 \parallel s_1 \]

\[ G(s_0) = s_{00} \parallel s_{01} \]

\[ G(s_1) = s_{10} \parallel s_{11} \]

in general, punctured key consists of \( n \) nodes if domain of PRF is \( \{0,1\}^n \).
Puncturable PRFs from GGM

\[ G(s) = s_0 \parallel s_1 \]

\[ G(s_0) = s_{00} \parallel s_{01} \]

\[ G(s_1) = s_{10} \parallel s_{11} \]

given \( s_1 \) and \( s_{00} \), easy to tell that 01 is the punctured point
Constraining PRFs Privately

Can we build a constrained PRF where the constrained key for a circuit $C$ hides $C$?
Constraining PRFs Privately

World 0

msk ← Setup(1^λ)

Constrain(msk, C₀) → C₀, C₁

World 1

msk ← Setup(1^λ)

Constrain(msk, C₁) → C₀, C₁

≈_C

Single-key privacy

Definitions generalize to multi-key privacy. See paper for details.
Private Puncturing

- **Correctness:** constrained evaluation at $x \neq z$ yields $F(k, x)$
- **Security:** $F(k, z)$ is indistinguishable from random
- **Privacy:** constrained key hides $z$
Implications of Privacy

Consider value of $\text{ConstrainEval}(sk_z, z)$:

- **Security**: Independent of $\text{Eval}(msk, z)$
- **Privacy**: Unguessable by the adversary
Using Privacy: Restricted Keyword Search

\[
\text{PRF}_k(\text{Honeycomb}) \rightarrow \{5, 8, 13\} \\
\text{PRF}_k(\text{KitKat}) \rightarrow \{18, 21\} \\
\text{PRF}_k(\text{Lollipop}) \rightarrow \{3, 10, 11\} \\
\text{PRF}_k(\text{Marshmallow}) \rightarrow \{1, 9, 22\}
\]

server with encrypted index

ConstrainEval(sk, Honeycomb)

\{5, 8, 13\}
Using Privacy: Restricted Keyword Search

ConstrainEval(sk, Jelly Bean)

No results

server with encrypted index

server with encrypted index

search for non-existent keyword

PRF_k(Honeycomb) → \{5, 8, 13\}
PRF_k(KitKat) → \{18, 21\}
PRF_k(Lollipop) → \{3, 10, 11\}
PRF_k(Marshmallow) → \{1, 9, 22\}
Using Privacy: Restricted Keyword Search

- $\text{PRF}_k(\text{Honeycomb}) \rightarrow \{5, 8, 13\}$
- $\text{PRF}_k(\text{KitKat}) \rightarrow \{18, 21\}$
- $\text{PRF}_k(\text{Lollipop}) \rightarrow \{3, 10, 11\}$
- $\text{PRF}_k(\text{Marshmallow}) \rightarrow \{1, 9, 22\}$

Server with encrypted index

ConstrinEval($sk, \text{Marshmallow}$)

Search for “restricted” keyword

No results
Using Privacy: Restricted Keyword Search

- **Security**: $\text{ConstrainEval}(sk, \text{Marshmallow}) \neq \text{Eval}(msk, \text{Marshmallow})$
- **Privacy**: Does not learn that no results were returned because no matches for keyword or if the keyword was restricted

- $\text{PRF}_k(\text{Honeycomb}) \rightarrow \{5,8,13\}$
- $\text{PRF}_k(\text{KitKat}) \rightarrow \{18,21\}$
- $\text{PRF}_k(\text{Lollipop}) \rightarrow \{3,10,11\}$
- $\text{PRF}_k(\text{Marshmallow}) \rightarrow \{1,9,22\}$

server with encrypted index

ConstrainEval$(sk, \text{Marshmallow})$

No results
The Many Applications of Privacy

• Private constrained MACs
  • Parties can only sign messages satisfying certain policy (e.g., enforce a spending limit), but policies are hidden

• Symmetric Deniable Encryption [CDNO97]
  • Two parties can communicate using a symmetric encryption scheme
  • If an adversary has intercepted a sequence of messages and coerces one of the parties to produce a decryption key for the messages, they can produce a “fake” key that decrypts all but a subset of the messages

• Constructing a family of watermarkable PRFs
  • Can be used to embed a secret message within a PRF that is “unremovable” – useful for authentication [CHNVW15]

See paper for details!
Summary of our Constructions

• From indistinguishability obfuscation (iO):
  • Private puncturable PRFs from iO + one-way functions
  • Private circuit constrained PRFs from sub-exponentially hard iO + one-way functions

• From concrete assumptions on multilinear maps:
  • Private puncturable PRFs from subgroup hiding assumptions
  • Private bit-fixing PRF from multilinear Diffie-Hellman assumption

This talk

See paper
Constructing Private Constrained PRFs

Tool: indistinguishability obfuscation [BGI^+01, GGH^+13]

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

\[ \text{iO}(P_1) \approx_c \text{iO}(P_2) \]
Indistinguishability Obfuscation (iO)

• First introduced by Barak et al. [BGI⁺01]
• First construction from multilinear maps [GGH⁺13]
  • Subsequent constructions from multilinear maps [BR13, BGK⁺14, AGIS14, Zim14, AB15, ...]
  • Constructions also from (compact) functional encryption [AJ15, AJS15]
Indistinguishability Obfuscation (iO)

Many applications – “crypto complete”

• Functional encryption [GGH\textsuperscript{+}13]
• Deniable encryption [SW13]
• Witness encryption [GGSW13]
• Private broadcast encryption [BZ14]
• Traitor tracing [BZ14]
• Multiparty key exchange [BZ14]
• Multiparty computation [GGHR14]
• and more...
Private Puncturing from iO

• Starting point: puncturable PRFs (e.g. GGM)

• Need a way to hide the point that is punctured
  • Intuition: obfuscate the puncturable PRF

• Question: what value to output at the punctured point?
Private Puncturing from iO

Use iO to hide the punctured point and output uniformly random value at punctured point

\[ P_z(x): \]
- If \( x = z \), output \( r \)
- Else, output \( \text{PRF}(k, x) \)

Program for punctured PRF (punctured at \( z \))

random value (hard coded)

real value of the PRF
Private Puncturing from iO

Suppose PRF is puncturable (e.g., GGM)
  • Master secret key: PRF key $k$
  • PRF output at $x \in \mathcal{X}$: $\text{PRF}(k, x)$

\[ P_z(x): \]
  • If $x = z$, output $r$
  • Else, output $\text{PRF}(k, x)$

Punctured key for a point $z$ is an obfuscated program
Constrained evaluation corresponds to evaluating obfuscated program
Private Puncturing from iO: Privacy

Recall privacy notion:

\[ msk \leftarrow \text{Setup}(1^\lambda) \]

World 0

\[ x_0, x_1 \in \mathcal{X} \]

\[ \text{Puncture}(k, x_0) \]

\[ \approx_c \]

World 1

\[ x_0, x_1 \in \mathcal{X} \]

\[ \text{Puncture}(k, x_1) \]
Private Puncturing from iO: Privacy

\[ P_{x_0}(x) : \]
- If \( x = x_0 \), output \( r \)
- Else, output \( \text{PRF}(k, x) \)

\[ P'_{x_0}(x) : \]
- If \( x = x_0 \), output \( r \)
- Else, output \( \text{PRF}(k_{x_0}, x) \)

By correctness of puncturing, \( P_{x_0} \) and \( P'_{x_0} \) compute identical functions.
Private Puncturing from iO: Privacy

Hybrid 0: Real game

\[ P_{x_0}(x): \]
- If \( x = x_0 \), output \( r \)
- Else, output \( \text{PRF}(k, x) \)

\[ \approx_c \text{iO} \]

Hybrid 1: Challenger responds to puncture query with iO of this program

\[ P'_{x_0}(x): \]
- If \( x = x_0 \), output \( r \)
- Else, output \( \text{PRF}(k_{x_0}, x) \)
Private Puncturing from iO: Privacy

Invoke puncturing security

\[ P'_{x_0}(x) : \]
\begin{itemize}
\item If \( x = x_0 \), output \( r \)
\item Else, output \( \text{PRF}(k_{x_0}, x) \)
\end{itemize}

\[ P''_{x_0}(x) : \]
\begin{itemize}
\item If \( x = x_0 \), output \( \text{PRF}(k, x_0) \)
\item Else, output \( \text{PRF}(k_{x_0}, x) \)
\end{itemize}
Private Puncturing from iO: Privacy

Invoke iO security

\[ P'''_x(x) : \]
\[ \begin{align*}
&\text{If } x = x_0, \text{ output } \text{PRF}(k, x_0) \\
&\text{Else, output } \text{PRF}(k_{x_0}, x)
\end{align*} \]

\[ \approx_c \]

Similar argument holds starting from \( P_{x_1}(x) \)
Private Puncturing from iO: Summary

Use iO to hide the punctured point and output uniformly random value at punctured point

$P_z(x)$:
- If $x = z$, output $r$
- Else, output $\text{PRF}(k, x)$
Construction generalizes to circuit constraints, except random values now derived from another PRF

\[ P_C(x) : \]
- If \( C(x) = 0 \), output \( \text{PRF}(k', x) \)
- If \( C(x) = 1 \), output \( \text{PRF}(k, x) \)

\( k' \) is independently sampled PRF key

“real” PRF value
Private Circuit Constrained PRF from iO

Recall intuitive requirements for private constrained PRF:

- **Security**: Values at constrained points independent of actual PRF value at those points
- **Privacy**: Values at constrained points are unguessable by the adversary

\[ P_C(x): \]
- If \( C(x) = 0 \), output \( \text{PRF}(k', x) \)
- If \( C(x) = 1 \), output \( \text{PRF}(k, x) \)
Private Circuit Constrained PRF from iO

Security proof similar to that for private puncturable PRF

Requires exponential number of hybrids (one for each input), so require sub-exponential hardness for iO and one-way functions

\[ P_C(x) : \]
- If \( C(x) = 0 \), output \( \text{PRF}(k', x) \)
- If \( C(x) = 1 \), output \( \text{PRF}(k, x) \)
Conclusions

• New notion of private constrained PRFs

• Simple definitions, but require powerful tools to construct: iO / multilinear maps

• Private constrained PRFs immediately provide natural solutions to many problems
Open Questions

• Puncturable PRFs can be constructed from OWFs
  • Can we construct private puncturable PRFs from OWFs?
  • Can we construct private circuit constrained PRFs without requiring sub-exponentially hard iO?

• Most of our candidate applications just require private puncturable PRFs
  • New applications for more expressive families of constraints?
Thanks!