Algebraic Pseudorandom Functions with Improved Efficiency from Augmented Cascade

Dan Boneh
Hart Montgomery
Ananth Raghunathan

Stanford University
Pseudorandom Functions

• Definition
Pseudorandom Functions

• Definition

\[ s \rightarrow f_s \rightarrow f_s(x) \]
Pseudorandom Functions

- Definition

\[ s \rightarrow f_s \rightarrow f_s(x) \]

The function $f$ looks like a random function.
Pseudorandom Functions

- Definition

\[ s \rightarrow f_s \rightarrow f_s(x) \]

The function \( f \) looks like a random function

- Security Game
Pseudorandom Functions

• Definition

\[ f_s(x) \]

\[ s \rightarrow f_s \rightarrow f_s(x) \]

The function \( f \) looks like a random function

• Security Game

\[
\begin{align*}
\text{Exp-PRF}: & \text{ choose a random key } k \text{ and set } f(x) = f_k(x) \\
\text{Exp-rand}: & \text{ choose a random function } f(x)
\end{align*}
\]
Pseudorandom Functions

- Definition

\[ f_s(x) \]

\( s \rightarrow f_s \rightarrow f_s(x) \)

The function \( f \) looks like a random function

- Security Game

**Exp-PRF**: choose a random key \( k \) and set \( f(x) = f_k(x) \)

**Exp-rand**: choose a random function \( f(x) \)
Pseudorandom Functions

• Definition

The function \( f \) looks like a random function

\[ s \rightarrow f_s \rightarrow f_s(x) \]

• Security Game

\textbf{Exp-PRF}: choose a random key \( k \) and set \( f(x) = f_k(x) \)

\textbf{Exp-rand}: choose a random function \( f(x) \)

PRF? or rand?
Pseudorandom Functions

• Definition

\[ s \rightarrow f_s \rightarrow f_s(x) \]

The function \( f \) looks like a random function

• Security Game

**Exp-PRF:** choose a random key \( k \) and set \( f(x) = f_k(x) \)

**Exp-rand:** choose a random function \( f(x) \)

**Secure if** cannot guess “PRF” or “rand” with probability better than \( 1/2 \)

**PRF? or rand?**
Applications of PRFs

- Workhorse of cryptography. Lots of applications!
  - Private Key Crypto; parties share PRF secret
  - Message Integrity and User Authentication
  - Key Derivation schemes
  - Stateless Signature Schemes
  - Defend against denial-of-service attacks [Ber ’96, CW03]
  - Prove lower bounds in learning theory; impossibility results in complexity theory
Heuristic PRFs
# Heuristic PRFs

<table>
<thead>
<tr>
<th>DES</th>
<th>IDEA</th>
<th>AES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Triple-DES</td>
<td>Serpent</td>
<td></td>
</tr>
</tbody>
</table>
Heuristic PRFs

<table>
<thead>
<tr>
<th>DES</th>
<th>IDEA</th>
<th>AES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Triple-DES</td>
<td>Serpent</td>
<td></td>
</tr>
</tbody>
</table>

Very fast but unfortunately rely on interactive assumptions
### Heuristic PRFs

<table>
<thead>
<tr>
<th>DES</th>
<th>IDEA</th>
<th>AES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Triple-DES</td>
<td>Serpent</td>
<td></td>
</tr>
</tbody>
</table>

Very fast but unfortunately rely on interactive assumptions

AES assumption: AES is a secure PRF!
Heuristic PRFs

<table>
<thead>
<tr>
<th>DES</th>
<th>IDEA</th>
<th>AES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Triple-DES</td>
<td>Serpent</td>
<td></td>
</tr>
</tbody>
</table>

Very fast but unfortunately rely on interactive assumptions

AES assumption: AES is a secure PRF!

Requires interactions between Challenger and Adversary
Heuristic PRFs

Very fast but unfortunately rely on interactive assumptions

AES assumption: AES is a secure PRF!
Requires interactions between Challenger and Adversary

Algebraic Constructions
Heuristic PRFs

<table>
<thead>
<tr>
<th>DES</th>
<th>IDEA</th>
<th>AES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Triple-DES</td>
<td>Serpent</td>
<td></td>
</tr>
</tbody>
</table>

Very fast but unfortunately rely on interactive assumptions

AES assumption: AES is a secure PRF!
Requires interactions between Challenger and Adversary

**Algebraic Constructions**

Non-interactive assumptions: Challenges posted. Require no interaction.
Heuristic PRFs

Very fast but unfortunately rely on interactive assumptions

AES assumption: AES is a secure PRF!
Requires interactions between Challenger and Adversary

Algebraic Constructions

Non-interactive assumptions: Challenges posted. Require no interaction.

Eg., DDH, Discrete Log, etc.
Cascade Construction
Cascade Construction

- Introduced by Bellare-Canetti-Krawczyk [Crypto ’96]
Cascade Construction

- Introduced by Bellare-Canetti-Krawczyk [Crypto '96]

\[ f : K \times X \rightarrow K \]
• Introduced by Bellare-Canetti-Krawczyk [Crypto ’96]

\[ f : K \times X \rightarrow K \]

**Cascade:**

\[ F(k; x_1 \ldots x_n) \]

The cascade construction is shown in Figure 1a6. More precisely, for an algorithm \( F \), let \( x \in X \) and \( k \in K \) do:

- \( x \leftarrow f(k; x) \)
- Output \( F(k; x_1 \ldots x_n) \)

The security of the cascade construction is stated as above. Note that if for all efficient algorithms \( A \) with \( \text{adv}((F \circ F), A) = 1 \) exists a group \( G \) where \( \text{adv}(F, A) = 1 \), let \( g \in G \) be the vector \( \gamma \) as above. Then, for all algorithms \( A \), let \( k \in K \) and \( x \in X \) do:

- \( g, u \in G \) define \( e \) such that

\[ e(g, u) = 1 \]

where \( e \) is a bilinear group if the group action in \( G \) and an efficiently computable bilinear map \( e : G \times G \rightarrow Z_p \). The cascade pseudorandom function, defined in [ACM CCS 2010], is a generalization of the GGM PRF [3].
Cascade Construction

- Introduced by Bellare-Canetti-Krawczyk [Crypto ’96]

\[ f : K \times X \rightarrow K \]

**Cascade:**

Key from \( K \), inputs from \( X^n \)

\[ F(k; x_1 \ldots x_n) \]
Cascade Construction

- Introduced by Bellare-Canetti-Krawczyk [Crypto ’96]

\[ f : K \times X \rightarrow K \]

Cascade:

Key from \( K \), inputs from \( X^n \)

**Theorem** \( f \) is a secure PRF \( \iff \) \( F \) is a secure PRF
Cascade Construction

- Introduced by Bellare-Canetti-Krawczyk [Crypto ’96]

\[ f : K \times X \rightarrow K \]

**Cascade:**

Key from \( K \), inputs from \( X^n \)

\[ f \text{ is a secure PRF } \iff F \text{ is a secure PRF} \]

Generalizes first algebraic construction of PRF by Goldreich-Goldwasser-Micali [GGM ’86] from Pseudorandom Generators
Augmented Cascade

Range of the function smaller than key domain.
Augmented Cascade

Range of the function smaller than key domain.

Can extend output using pseudorandom generators; but breaks the *algebraic structure*
Augmented Cascade

Range of the function smaller than key domain.

Can extend output using pseudorandom generators; but breaks the *algebraic structure*

\[ f : (S \times K) \times X \rightarrow K \]

Theorem 2.

For every \( q \)-query PRF adversary \( A \) attacking \( F^*n \) there exists a \( q \)-query PRF adversary \( B \) attacking \( F^*n \) such that

\[ \text{PRF}_{\text{adv}}[A, F^*n] \leq nq \cdot \text{PRF}_{\text{adv}}[B, F] \]

where \( B \) runs in about the same time as \( A \).

3.2 Augmented Cascade PRF

The cascade construction works with a PRF \( F \) whose output is as long as the PRF key. When constructing algebraic PRFs, the starting point is often a PRF \( F \) whose output is shorter than required for cascade. We therefore need to augment the output of \( F \) so that its output is a valid key for \( F \).

Consider a PRF \( F \) operating on the following spaces:

\[ f : (S \times K) \times X \rightarrow K \]

Notice that the key for \( F \) is a pair in \( (S, K) \) while the output is in \( K \) and therefore not a complete key. In the augmented cascade we append a fresh random string to the output to make it into a valid key.

We define the augmented cascade, denoted \( \hat{F}^*n \), as a function

\[ F : (S^n \times K) \times X^n \rightarrow K \]

The function's domain is \( X^n \) and its keys are tuples of the form \( (s_1, \ldots, s_n, k_0) \in S^n \times K \). The augmented cascade is shown in Figure 1(b) and is defined as follows:

**input:** key \((s_1, \ldots, s_n, k_0) \in S^n \times K\), and value \((x_1, \ldots, x_n) \in X^n\) for \( i = 1, \ldots, n \)

\[ k_i \leftarrow F((s_i, k_{i-1}), x_i) \]

**output** \( k_n \)

Security.

Unfortunately, the augmented cascade can be insecure even if the underlying function \( F \) is a secure PRF. For example, \( F \) can be a secure PRF even if it ignores the part of the key in \( K \) (i.e. \( F \) only uses the part of the key in \( S \)). In this case, since we ignore \( k_i \) (for all \( i \)), the last block of the augmented cascade construction \( \hat{F}^*n \) ignores the first \( n-1 \) input blocks and hence cannot be a secure PRF. In the next two sections we establish sufficient conditions for security of the augmented cascade.
Theorem 2. For every \( q \)-query PRF adversary \( A \) attacking \( F^* \), there exists a \( q \)-query PRF adversary \( B \) attacking \( F \) such that

\[
\text{PRF}_{\text{adv}}[A,F^*] \leq nq \cdot \text{PRF}_{\text{adv}}[B,F]
\]

where \( B \) runs in about the same time as \( A \).

3.2 Augmented Cascade PRF

The cascade construction works with a PRF \( F \) whose output is as long as the PRF key. When constructing algebraic PRFs, the starting point is often a PRF \( F \) whose output is shorter than required for cascade. We therefore need to augment the output of \( F \) so that its output is a valid key for \( F \).

Consider a PRF \( F \) operating on the following spaces:

\[
f : (S \times K) \times X \rightarrow K
\]

Notice that the key for \( F \) is a pair in \((S, K)\) while the output is in \( K \) and therefore not a complete key. In the augmented cascade we append a fresh random string to the output to make it into a valid key.

We define the augmented cascade, denoted \( \hat{F}^* \), as a function

\[
\hat{F}^* : (S^n \times K) \times X^n \rightarrow K
\]

The function's domain is \( X^n \) and its keys are tuples of the form \((s_1, \ldots, s_n, k_0) \in S^n \times K\). The augmented cascade is shown in Figure 1(b) and is defined as follows:

input: key \((s_1, \ldots, s_n, k_0) \in S^n \times K\), and value \((x_1, \ldots, x_n) \in X^n\) for \( i = 1, \ldots, n \) do:

\[
k_i \leftarrow F((s_i, k_{i-1}), x_i)
\]

output \( k_n \)

Security.

Unfortunately, the augmented cascade can be insecure even if the underlying function \( F \) is a secure PRF. For example, \( F \) can be a secure PRF even if it ignores the part of the key in \( K \) (i.e. \( F \) only uses the part of the key in \( S \)). In this case, since we ignore \( k_i \) (for all \( i \)), the last block of the augmented cascade construction is evaluated independently of the first \( n - 1 \) blocks. Thus, the resulting augmented cascade construction \( \hat{F}^* \) ignores the first \( n - 1 \) input blocks and hence cannot be a secure PRF. In the next two sections we establish sufficient conditions for security of the augmented cascade.
Augmented Cascade

Range of the function smaller than key domain.

Can extend output using pseudorandom generators; but breaks the \textit{algebraic structure}

\[ f : (S \times K) \times X \rightarrow K \]

\textbf{Augmented Cascade:}

\[ F(s_1 \ldots s_n,k;x_1 \ldots x_n) \]
Theorem 2. For every $q$-query PRF adversary $A$ attacking $F^*$ there exists a $q$-query PRF adversary $B$ attacking $F$ such that

$$\text{PRF}_{\text{adv}}[A, F^*] \leq nq \cdot \text{PRF}_{\text{adv}}[B, F]$$

where $B$ runs in about the same time as $A$.

3.2 Augmented Cascade PRF

The cascade construction works with a PRF $F$ whose output is as long as the PRF key. When constructing algebraic PRFs, the starting point is often a PRF $F$ whose output is shorter than required for cascade. We therefore need to augment the output of $F$ so that its output is a valid key for $F$. Consider a PRF $F$ operating on the following spaces:

$$f : (S \times K) \times X \rightarrow K$$

Notice that the key for $F$ is a pair in $(S, K)$ while the output is in $K$ and therefore not a complete key. In the augmented cascade we append a fresh random string to the output to make it into a valid key.

We define the augmented cascade, denoted $\hat{F}^*$, as a function $F : (S^n \times K) \rightarrow K$.

The function's domain is $X^n$ and its keys are tuples of the form $(s_1, s_2, ..., s_n, k) \in S^n \times K$. The augmented cascade is shown in Figure 1(b) and is defined as follows:

- **Input:** key $(s_1, s_2, ..., s_n, k_0) \in S^n \times K$, and value $(x_1, x_2, ..., x_n) \in X^n$ for $i = 1, ..., n$.
- **Process:**
  - $k_i \leftarrow F((s_i, k_{i-1}), x_i)$ for $i = 1, ..., n$.
- **Output:** $k_n$.

Security.

Unfortunately, the augmented cascade can be insecure even if the underlying function $F$ is a secure PRF. For example, $F$ can be a secure PRF even if it ignores the part of the key in $K$ (i.e. $F$ only uses the part of the key in $S$). In this case, since we ignore $k_i$ for all $i$, the last block of the augmented cascade construction is evaluated independently of the first $n-1$ blocks. Thus, the resulting augmented cascade construction $\hat{F}^*$ ignores the first $n-1$ input blocks and hence cannot be a secure PRF. In the next two sections we establish sufficient conditions for security of the augmented cascade.
Is Augmented Cascade secure?
Is Augmented Cascade secure?

f is a secure PRF
Is Augmented Cascade secure?

\[ f \text{ is a } \text{secure PRF} \implies F \text{ is a } \text{secure PRF} \]
Is Augmented Cascade secure?

\[ f \text{ is a } \text{secure PRF} \quad \not\iff \quad F \text{ is a } \text{secure PRF} \]
Is Augmented Cascade secure?

\[ f \text{ is a secure PRF } \quad \not\implies \quad F \text{ is a secure PRF?} \]

Eg.:

\[ \begin{array}{c}
\text{f} \\
\text{k} \\
\text{x}
\end{array} \quad \rightarrow \quad \begin{array}{c}
\text{x} \\
\text{g} \\
\text{s}
\end{array} \]
Is Augmented Cascade secure?

\[ f \text{ is a secure PRF} \quad \not\Rightarrow \quad F \text{ is a secure PRF?} \]

Eg.:

![Diagram of f and g functions with k and x inputs and g output with s input ignored.](image)
Is Augmented Cascade secure?

\( f \) is a \textit{secure} PRF \quad \not\iff \quad \text{F is a \textit{secure} PRF?}

\textbf{Eg.:

\[
\begin{align*}
&\text{f} \\
&\text{g}
\end{align*}
\]

(is ignored)

\[
\begin{align*}
&\text{f} \\
&\text{f} \\
&\text{f} \\
&\text{f} \\
&\text{f}
\end{align*}
\]
Is Augmented Cascade secure?

\[ f \text{ is a secure PRF} \quad \not\implies \quad F \text{ is a secure PRF?} \]

Eg.:

Easy to see: \( F(s_1 \ldots s_n, k; x_1 \ldots x_n) = g(s_n, x_n) \)
Is Augmented Cascade secure?

$\begin{align*}
\text{f is a secure PRF} & \quad \neq \quad \text{F is a secure PRF?} \\
\text{Easy to see: } F(s_1...s_n,k;x_1...x_n) = g(s_n,x_n)
\end{align*}$

Eg.

Challenger
Is Augmented Cascade secure?

\[ f \text{ is a secure PRF} \quad \cancel{\Rightarrow} \quad F \text{ is a secure PRF?} \]

**Eg.:**

Easy to see: \[ F(s_1 \ldots s_n, k; x_1 \ldots x_n) = g(s_n, x_n) \]

**Challenger**

00, 10
Is Augmented Cascade secure?

If \( f \) is a secure PRF, does \( F \) is a secure PRF?

Eg.:

\[
\begin{align*}
\text{Easy to see: } & \quad F(s_1 \ldots s_n, k; x_1 \ldots x_n) = g(s_n, x_n) \\
& \quad (\text{is ignored})
\end{align*}
\]

Challenger

\[
\begin{align*}
\text{00, 10} & \quad f(00), f(10)
\end{align*}
\]
Is Augmented Cascade secure?

$f$ is a secure PRF $\iff$ $F$ is a secure PRF?

Eg.:

$$f(x, k, s) = g(s, x)$$

Easy to see: $F(s_1...s_n, k; x_1...x_n) = g(s_n, x_n)$

Challenger:

00, 10

f(00), f(10)

If $f(00) = f(10)$ output “PRF”
Is Augmented Cascade secure?

f is a secure PRF \( \not\Rightarrow \) F is a secure PRF?

Eg.:

\[
\begin{align*}
\text{f} & \quad \text{g} \\
\text{k} & \quad \text{s} \\
\text{x} & \\
\end{align*}
\]

(is ignored)

Easy to see: \( F(s_1\ldots s_n,k;x_1\ldots x_n) = g(s_n,x_n) \)

Challenger

00, 10
\[
\begin{align*}
\text{f(00)}, \text{ f(10)} \\
\end{align*}
\]

If \( f(00) = f(10) \) output “PRF”

\( f(00) = f(10) \) for random \( f \) with very low probability
Therefore, \( F \) is not a secure PRF!
Consider \( q \) related keys \((s, k_1), \ldots, (s, k_q)\)
Consider $q$ related keys $(s, k_1), \ldots, (s, k_q)$
Consider $q$ related keys $(s, k_1), \ldots, (s, k_q)$

These functions look like $q$ random functions
Parallel Composition Security

Consider \( q \) related keys \((s, k_1), \ldots, (s, k_q)\)

These functions look like \( q \) random functions

In other words: “Simultaneously Secure”
Consider $q$ related keys $(s,k_1), \ldots, (s,k_q)$

These functions look like $q$ random functions

In other words:
"Simultaneously Secure"

Theorem

$f$ is a $q$-parallel secure PRF $\iff$ $F$ is a secure PRF
Proof Outline
Proof Outline

\[ x_1 \rightarrow x_2 \rightarrow x_3 \rightarrow \ldots \rightarrow x_i \rightarrow x_{i+1} \rightarrow f \rightarrow \ldots \rightarrow f \rightarrow P_i(s, k; x) \]

\[ s_{i+1} \rightarrow s_n \]
Proof Outline

Initially:

\[ f_{\text{random}} \rightarrow f \rightarrow \ldots \rightarrow f_{i+1} \rightarrow s_{i+1} \rightarrow f_{i} \rightarrow \ldots \rightarrow f_{n} \rightarrow P_{i}(s,k;x) \]
Proof Outline

Initially:

\[ x_1 \quad x_2 \quad x_3 \quad \ldots \quad x_{i+1} \quad x_n \]

\[ f_{\text{random}} \quad f \quad \ldots \quad f \quad P_i(s,k;\mathbf{x}) \]

\[ s_{i+1} \quad s_n \]

\[ f(s,k;\mathbf{x}) \]
Proof Outline

Initially:

Finally:

Initially:

Finally:
Proof Outline

Initially:

Finally:
Proof Outline

\[
\begin{align*}
X_1 &\rightarrow f_{\text{random}} & &\rightarrow f & &\rightarrow P_t(s,k;x) \\
X_2 &\rightarrow & &\rightarrow & &\rightarrow \\
X_3 &\rightarrow & &\rightarrow & &\rightarrow \\
\vdots &\rightarrow & &\rightarrow & &\rightarrow \\
X_t &\rightarrow & &\rightarrow & &\rightarrow \\
X_{t+1} &\rightarrow & &\rightarrow & &\rightarrow \\
X_n &\rightarrow & &\rightarrow & &\rightarrow \\
S_{t+1} &\rightarrow & &\rightarrow & &\rightarrow \\
S_n &\rightarrow & &\rightarrow & &\rightarrow 
\end{align*}
\]
Proof Outline

\[ f_{\text{random}}(x_1, x_2, x_3, \ldots, x_t, x_{t+1}) \rightarrow f \rightarrow P_{t+1}(s, k; x) \]
Proof Outline

\[ X_1 \rightarrow X_2 \rightarrow X_3 \rightarrow \ldots \rightarrow X_t \rightarrow X_{t+1} \downarrow \]

\[ f_{\text{random}} \rightarrow \ldots \rightarrow f \rightarrow P_{t+1}(s,k;x) \uparrow \]

\[ s \rightarrow k_1 \rightarrow f \rightarrow k_2 \rightarrow f \rightarrow \ldots \rightarrow k_Q \rightarrow f \rightarrow f_{\text{random}} \]

or

Challenger
Proof Outline

\[ f_{\text{random}}(x_1, x_2, x_3, \ldots, x_t, x_{t+1}) \]

\[ f(x_1, x_2, x_3, \ldots, x_t) \rightarrow f(x_{t+1}) \]

Challenger

\[ f_{\text{random}}(s, k; x) \rightarrow P_{t+1}(s, k; x) \]
Proof Outline

$s_{t+1}$ is set to $s$

Depending on $x_1...x_t$ we choose the value of $f(x_1...x_t)$ to be some key $k_j$ - maintained in an associative table

**Easy to see**: We simulate either $P_t$ or $P_{t+1}$ depending upon Challenger

**ACM CCS 2010**

Algebraic Pseudorandom Functions with Improved Efficiency from Augmented Cascade
Why Augmented Cascade?
Why Augmented Cascade?

- Generic tool to build efficient algebraic PRFs
Why Augmented Cascade?

- Generic tool to build efficient algebraic PRFs
Why Augmented Cascade?

- Generic tool to build **efficient** algebraic PRFs
Why Augmented Cascade?

- Generic tool to build efficient algebraic PRFs
Why Augmented Cascade?

• Generic tool to build efficient algebraic PRFs
Why Augmented Cascade?

- Generic tool to build efficient algebraic PRFs

\[ f(s, k; x) \]

Why Augmented Cascade?
Why Augmented Cascade?

- Generic tool to build efficient algebraic PRFs

\[ f(s,k;x) \]

• Simple proofs for existing algebraic PRFs. We show q-parallel security of:
  Naor-Reingold [FOCS ’97] & Lewko-Waters [CCS ’09]
Naor-Reingold [FOCS '97]
Naor-Reingold [FOCS ’97]

\[ b \in \{0,1\} \]

\[ h \in G \]

\[ x \in \mathbb{Z}_p \]

\[ nr \]

\[ h \text{ if } b = 0 \]

\[ h^x \text{ if } b = 1 \]
Naor-Reingold [FOCS ’97]

\[ b \in \{0,1\} \]

\[ h \in G \]

\[ x \in \mathbb{Z}_p \]

\[ h \text{ if } b=0 \]

\[ h^x \text{ if } b=1 \]

Aug. Cascade
Naor-Reingold [FOCS '97]

\[ h \in G, \quad b \in \{0,1\} \]

\[ x \in \mathbb{Z}_p \]

\[ h \] if \( b = 0 \)

\[ h^x \] if \( b = 1 \)

Aug. Cascade

\[ h \in G, \quad b \in \{0,1\}^m \]

\[ x_1 \ldots x_m \]

\[ h^{x_1b_1 \ldots x_mb_m} \]
Theorem: \( \text{nr} \) is \( q \)-parallel secure under the DDH assumption.
Theorem: \( nr \) is \( q \)-parallel secure under the DDH assumption.

Group \( G \) of prime order \( p \) with generator \( g \).
\( g^{ab} \) looks \textit{random} in \( G \), given \( g^a, g^b \) for random \( a, b \),
Naor-Reingold [FOCS ’97]

**Theorem:** \( nr \) is \( q \)-parallel secure under the DDH assumption

Group \( G \) of prime order \( p \) with generator \( g \).
\( g^{ab} \) looks random in \( G \), given \( g^a, g^b \) for random \( a, b \),

**Corollary:** \( NR \) is secure PRF under the DDH assumption
Lewko-Waters [CCS ’09]

- Details in the paper
Lewko-Waters [CCS ’09]

- Details in the paper

**Theorem:** lw is q-parallel secure under the k-linear assumption
• Details in the paper

**Theorem:** \( \text{lw} \) is \( q \)-parallel secure under the \( k \)-linear assumption

Follows from the randomized self-reducibility (due to Lewko-Waters) of the \( k \)-linear assumption
Lewko-Waters [CCS ’09]

• Details in the paper

**Theorem:** \( \text{lw} \) is \( q \)-parallel secure under the \( k \)-linear assumption

Follows from the randomized self-reducibility (due to Lewko-Waters) of the \( k \)-linear assumption

**Corollary:** \( \text{LW} \) is secure PRF under the \( k \)-linear assumption
New Algebraic PRF
New Algebraic PRF

\[ x \in \{1, \ldots, L\} \]

\[ h \in G \]

\[ s \in \mathbb{Z}_p \]

\[ h^{\frac{1}{s+x}} \]
New Algebraic PRF

Dodis-Yampolskiy [PKC ’05]
New Algebraic PRF

Aug. Cascade

Dodis-Yampolskiy [PKC ’05]
New Algebraic PRF

\[ x \in \{1, \ldots, L\} \]
\[ h \in G \]
\[ s \in \mathbb{Z}_p \]

\[ h^{1/(s+x)} \]

Aug. Cascade

\[ x \in \{1, \ldots, L\}^n \]
\[ h \]
\[ s_1 \ldots s_n \]

\[ 1/(s_1+x_1) \ldots (s_n+x_n) \]

Dodis-Yampolskiy [PKC ’05]
New Algebraic PRF

$\begin{align*}
\text{Aug. Cascade} & \\
\text{Our PRF} & \\
\end{align*}$

Dodis-Yampolskiy [PKC ’05]

**Theorem:** is $q$-parallel secure under the L-DDH (inversion) assumption
**New Algebraic PRF**

\[ x \in \{1, \ldots, L\} \]
\[ h \in G \]
\[ s \in \mathbb{Z}_p \]

**Aug. Cascade**

\[ x \in \{1, \ldots, L\}^n \]
\[ 1/(s_1+x_1) \ldots (s_n+x_n) \]

**Dodis-Yampolskiy [PKC ’05]**

**Theorem:** \( \text{DY} \) is q-parallel secure under the L-DDH (inversion) assumption

\[ g^{1/x} \text{ looks random in } G \text{ given } (g, g^x, g^{x^2}, \ldots, g^{x^L}) \]
New Algebraic PRF

Dodis-Yampolskiy [PKC '05]

**Theorem**: $DY$ is $q$-parallel secure under the L-DDH (inversion) assumption

$g^{1/x}$ looks random in $G$ given $(g, g^x, g^{x^2}, ..., g^{x^L})$

(Details in the paper)
New Algebraic PRF

Theorem: \textbf{DY} is q-parallel secure under the L-DDH (inversion) assumption

Corollary: is secure PRF under the L-DDH assumption
Comparison to Naor-Reingold
## Comparison to Naor-Reingold

<table>
<thead>
<tr>
<th></th>
<th>Naor-Reingold PRF</th>
<th>Our PRF</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Domain</strong></td>
<td>{0,1}^m</td>
<td>([L]^n) ((L=2^l,\ n=m/l))</td>
</tr>
<tr>
<td><strong>Keys</strong></td>
<td>(h \in G, \ x_1, \ldots, x_m \in \mathbb{Z}_p)</td>
<td>(h \in G, \ s_1, \ldots, s_n \in \mathbb{Z}_p)</td>
</tr>
</tbody>
</table>
| **PRF**          | \(\beta = x_1 \cdot x_2 \cdot x_3 \cdot x_4 \cdot x_5 \cdot x_6 \cdot x_7 \cdot x_8\)  
                      | \(F(x) = h^\beta\)                                                     | \(\beta = (s_1 + \ldots)(s_2 + \ldots)\)                             
                      | \(F(x) = h^{1/\beta}\)                                                  |
## Comparison to Naor-Reingold

<table>
<thead>
<tr>
<th></th>
<th>Naor-Reingold PRF</th>
<th>Our PRF</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Domain</strong></td>
<td>( {0,1}^m )</td>
<td>([L]^n \ (L=2^l, \ n=m/l))</td>
</tr>
<tr>
<td><strong>Keys</strong></td>
<td>( h ) in ( G ), ( x_1, \ldots, x_m ) in ( \mathbb{Z}_p )</td>
<td>( h ) in ( G ), ( s_1, \ldots, s_n ) in ( \mathbb{Z}_p )</td>
</tr>
<tr>
<td><strong>PRF</strong></td>
<td>( \beta = x_1 \cdot x_2 \cdot x_3 \cdot x_4 \cdot x_5 \cdot x_6 \cdot x_7 \cdot x_8 )</td>
<td>( \beta = (s_1 + [\text{white}]) (s_2 + [\text{blue, red, purple, green, yellow, orange, brown, black}]) )</td>
</tr>
<tr>
<td></td>
<td>( F(x) = h^{\beta} )</td>
<td>( F(x) = h^{1/\beta} )</td>
</tr>
</tbody>
</table>

Secret key shorter by a **factor of** \( l \)
## Comparison to Naor-Reingold

<table>
<thead>
<tr>
<th></th>
<th>Naor-Reingold PRF</th>
<th>Our PRF</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Domain</strong></td>
<td>({0,1}^m)</td>
<td>([L]^n\ (L=2^l, \ n=m/l))</td>
</tr>
<tr>
<td><strong>Keys</strong></td>
<td>(h \in G, \ x_1,\ldots, x_m \in \mathbb{Z}_p)</td>
<td>(h \in G, \ s_1,\ldots, s_n \in \mathbb{Z}_p)</td>
</tr>
<tr>
<td><strong>PRF</strong></td>
<td>(\beta = x_1 \cdot x_2 \cdot x_3 \cdot x_4 \cdot x_5 \cdot x_6 \cdot x_7 \cdot x_8)</td>
<td>(\beta = (s_1 + \square)(s_2 + \square))</td>
</tr>
<tr>
<td></td>
<td>(F(x) = h^\beta)</td>
<td>(F(x) = h^{1/\beta})</td>
</tr>
</tbody>
</table>

Secret key shorter by a **factor of** \(l\)

**Factor of** \(l\) fewer multiplications to evaluate \(\beta\)
Comparison to Naor-Reingold

<table>
<thead>
<tr>
<th></th>
<th>Naor-Reingold PRF</th>
<th>Our PRF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain</td>
<td>({0,1}^m)</td>
<td>([L]^n (L=2^l, n=m/l))</td>
</tr>
<tr>
<td>Keys</td>
<td>(h \in G, x_1,...,x_m \in \mathbb{Z}_p)</td>
<td>(h \in G, s_1,...,s_n \in \mathbb{Z}_p)</td>
</tr>
<tr>
<td>PRF</td>
<td>(\beta = x_1 \cdot x_2 \cdot x_3 \cdot x_4 \cdot x_5 \cdot x_6 \cdot x_7 \cdot x_8)</td>
<td>(\beta = (s_1 + \ldots)(s_2 + \ldots))</td>
</tr>
<tr>
<td></td>
<td>(F(x) = h^\beta)</td>
<td>(F(x) = h^{1/\beta})</td>
</tr>
</tbody>
</table>

Secret key shorter by a factor of \(l\)

Factor of \(l\) fewer multiplications to evaluate \(\beta\)

However, security becomes weaker as \(l\) increases
### Comparison to Naor-Reingold

<table>
<thead>
<tr>
<th></th>
<th>Naor-Reingold PRF</th>
<th>Our PRF</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Domain</strong></td>
<td>( {0,1}^m )</td>
<td>([L]^n \ (L=2^l, \ n=m/l ))</td>
</tr>
<tr>
<td><strong>Keys</strong></td>
<td>( h \in G, x_1,\ldots,x_m \in \mathbb{Z}_p )</td>
<td>( h \in G, s_1,\ldots,s_n \in \mathbb{Z}_p )</td>
</tr>
<tr>
<td><strong>PRF</strong></td>
<td>( \beta = x_1 \cdot x_2 \cdot x_3 \cdot x_4 \cdot x_5 \cdot x_6 \cdot x_7 \cdot x_8 )</td>
<td>( \beta = (s_1 + \square)(s_2 + \square) )</td>
</tr>
<tr>
<td>( F(x) = h^\beta )</td>
<td>( F(x) = h^{1/\beta} )</td>
<td></td>
</tr>
</tbody>
</table>

- **Secret key shorter by a factor of** \( l \)
- **Factor of** \( l \) **fewer multiplications to evaluate** \( \beta \)
- **However, security becomes weaker as** \( l \) **increases**
- \( l = 4 \) or 8 is reasonable
Verifiable Random Functions
Verifiable Random Functions

PRF
Verifiable Random Functions

\[ x \leftrightarrow f(k;x) \leftrightarrow y \leftrightarrow f(k;y) \]
Verifiable Random Functions

\[ x \quad \leftrightarrow \quad f(k;x) \quad \leftrightarrow \quad y \quad \leftrightarrow \quad f(k;y) \]
Verifiable Random Functions

PRF

\[ f(k; x) \]

\[ f(k; y) \]

random

\[ x \]

\[ y \]

\[ f(k; y) \]
Verifiable Random Functions

PRF

\[ f(k; x) \]

\[ f(k; y) \]

random

random
Verifiable Random Functions

How to guarantee honest evaluation?

PRF

\[ x \xrightarrow{\text{random}} y \xrightarrow{\text{random}} \]

\[ f(k; x) \xrightarrow{\text{random}} f(k; y) \]
Verifiable Random Functions

- Introduced by Micali-Rabin-Vadhan [FOCS ’99]

How to guarantee honest evaluation?
Verifiable Random Functions

- Introduced by Micali-Rabin-Vadhan [FOCS ’99]
- Produce non-interactive proofs (that can be verified by anyone) that the PRF is evaluated correctly

How to guarantee honest evaluation?
Verifiable Random Functions

- Introduced by Micali-Rabin-Vadhan [FOCS ’99]
- Produce non-interactive proofs (that can be verified by anyone) that the PRF is evaluated correctly
- Not at the cost of pseudorandomness!
Verifiable Random Functions

- Introduced by Micali-Rabin-Vadhan [FOCS ’99]
- Produce non-interactive proofs (that can be verified by anyone) that the PRF is evaluated correctly
- Not at the cost of pseudorandomness!
- Applications: Unique-signature schemes, e-cash schemes [BCKL ’09, ASM ’07], updatable zero-knowledge databases [Liskov ’05] etc.
Verifiable Random Functions

- We construct VRFs using the augmented cascade
Verifiable Random Functions

- We construct VRFs using the augmented cascade
- Security from Bilinear Diffie-Hellman Inversion assumption
Verifiable Random Functions

- We construct VRFs using the augmented cascade
- Security from Bilinear Diffie-Hellman Inversion assumption

<table>
<thead>
<tr>
<th>VRF</th>
<th>Domain Size</th>
<th>Assumption Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>[DY ’05]</td>
<td>$L^n$ (poly in sec. param.)</td>
<td>$L^n$</td>
</tr>
<tr>
<td>our small-domain VRF</td>
<td>$L^n$ (poly in sec. param.)</td>
<td>$nL$</td>
</tr>
</tbody>
</table>
Verifiable Random Functions

• We construct VRFs using the augmented cascade

• Security from Bilinear Diffie-Hellman Inversion assumption

<table>
<thead>
<tr>
<th>VRF</th>
<th>Domain Size</th>
<th>Assumption Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>[DY ’05]</td>
<td>$L^n$ (poly in sec. param.)</td>
<td>$L^n$</td>
</tr>
<tr>
<td>our small-domain VRF</td>
<td>$L^n$ (poly in sec. param.)</td>
<td>$nL$</td>
</tr>
</tbody>
</table>

• Hohenberger-Waters [Eurocrypt ’10] constructed an efficient large-domain VRF
Verifiable Random Functions

- We construct VRFs using the **augmented cascade**
- Security from Bilinear Diffie-Hellman Inversion assumption

<table>
<thead>
<tr>
<th>VRF</th>
<th>Domain Size</th>
<th>Assumption Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>[DY ’05]</td>
<td>$L^n$ (poly in sec. param.)</td>
<td>$L^n$</td>
</tr>
<tr>
<td>our small-domain VRF</td>
<td>$L^n$ (poly in sec. param.)</td>
<td>$nL$</td>
</tr>
</tbody>
</table>

- Hohenberger-Waters [Eurocrypt ’10] constructed an efficient large-domain VRF
- Require **weaker assumption** than [HW ’10] but less efficient
Verifiable Random Functions

- We construct VRFs using the augmented cascade
- Security from Bilinear Diffie-Hellman Inversion assumption

<table>
<thead>
<tr>
<th>VRF</th>
<th>Domain Size</th>
<th>Assumption Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>[DY ’05]</td>
<td>$L^n$ (poly in sec. param.)</td>
<td>$L^n$</td>
</tr>
<tr>
<td>our small-domain VRF</td>
<td>$L^n$ (poly in sec. param.)</td>
<td>$nL$</td>
</tr>
</tbody>
</table>

- Hohenberger-Waters [Eurocrypt ’10] constructed an efficient large-domain VRF
- Require *weaker assumption* than [HW ’10] but less efficient

<table>
<thead>
<tr>
<th>VRF</th>
<th>Domain Size</th>
<th>Assumption Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>[HW ’10]</td>
<td>$2^m$ (arbitrary)</td>
<td>$O(mQ)$</td>
</tr>
<tr>
<td>our large-domain VRF</td>
<td>$2^m$ (arbitrary)</td>
<td>$O(m)$</td>
</tr>
</tbody>
</table>
Conclusions
Conclusions

1) New generic tool to construct PRFs – augmented cascade
Conclusions

1) New generic tool to construct PRFs – **augmented cascade**

$$F(s_1 \ldots s_n, k; x_1 \ldots x_n)$$
Conclusions

1) New generic tool to construct PRFs – augmented cascade

\[
F(s_1\ldots s_n, k; x_1\ldots x_n)
\]

2) Simple proofs of security for existing algebraic PRFs
Conclusions

1) New generic tool to construct PRFs – augmented cascade

\[
F(s_1...s_n,k;x_1...x_n) = \underbrace{f(f(f(...f}_{n \text{ times}}(x_1))...))
\]

2) Simple proofs of security for existing algebraic PRFs
Conclusions

1) New generic tool to construct PRFs – **augmented cascade**

\[
F(s_1...s_n, k; x_1...x_n)
\]

2) Simple proofs of security for existing algebraic PRFs
Conclusions

1) New generic tool to construct PRFs – **augmented cascade**

\[ F(s_1...s_n, k; x_1...x_n) \]

2) Simple proofs of security for existing algebraic PRFs

3) New **efficient large-domain PRF**
Conclusions

1) New generic tool to construct PRFs – augmented cascade

\[ F(s_1...s_n, k; x_1...x_n) \]

2) Simple proofs of security for existing algebraic PRFs

3) New efficient large-domain PRF

\[ h = \frac{1}{s_1 + x_1} \cdots \frac{1}{s_n + x_n} \]
Conclusions

1) New generic tool to construct PRFs – augmented cascade

\[
F(s_1 \ldots s_n, k; x_1 \ldots x_n)
\]

2) Simple proofs of security for existing algebraic PRFs

3) New efficient large-domain PRF

\[
h \left( \frac{1}{(s_1 + x_1) \ldots (s_n + x_n)} \right)
\]

4) VRFs based on better security assumptions
Conclusions

1) New generic tool to construct PRFs – augmented cascade

\[ F(s_1 \ldots s_n, k; x_1 \ldots x_n) \]

2) Simple proofs of security for existing algebraic PRFs

3) New efficient large-domain PRF

\[ x \in \{1, \ldots, L\}^n \]

\[ h \rightarrow \frac{1}{(s_1 + x_1) \ldots (s_n + x_n)} \]

4) VRFs based on better security assumptions

5) (in paper) Simulatable VRFs
Any questions?