Security/Efficiency Tradeoffs for Permutation-Based Hashing

Phillip Rogaway, John P. Steinberger

April 14, 2008
Motivation

- Most hash functions are built from blockciphers (SHA-1, MD4, MD5, MDC-2, ...)

Motivation

- Most hash functions are built from blockciphers (SHA-1, MD4, MD5, MDC-2, ...)
- Keying costs
Motivation

- Most hash functions are built from blockciphers (SHA-1, MD4, MD5, MDC-2, ...)
- Keying costs
- Use fixed keys $\rightarrow$ random permutations
Motivation

• Most hash functions are built from blockciphers (SHA-1, MD4, MD5, MDC-2, ...)
• Keying costs
• Use fixed keys → random permutations
• Advantages: speed + minimalism + assurance
Difficulties

- Permutations afford no compression
Permutations afford no compression

Black-Cochran-Shrimpton ’05: No rate-1 iterated construction making a single call to a permutation can be secure
Difficulties

- Permutations afford no compression

- Black-Cochran-Shrimpton '05: No rate-1 iterated construction making a single call to a permutation can be secure

- Large number of permutations necessary to achieve reasonable rate of security
Prior Constructions

- Govaerts-Preneel-Vandewalle '93: variety of permutation-based constructions of rates $1/4$–$1/8$; no security proofs
- Shrimpton-Stam '07: A $2n$-to-$n$ bit compression function using 3 calls to a random function, of collision security $2^{n/2}$

Bertoni-Daemens-Peeters-Assche '07: sponge construction
Our results

- A “good” $2n$-to-$n$ bit compression function needs 3 permutations to get collision security $2^{n/2}$
- A good $3n$-to-$2n$ bit compression function needs 5 permutations to get collision security above $2^{n/2}$
Our results

- A “good” $2n$-to-$n$ bit compression function needs 3 permutations to get collision security $2^{n/2}$
- A good $3n$-to-$2n$ bit compression function needs 5 permutations to get collision security above $2^{n/2}$
- A good $mn$-to-$rn$ bit compression function making $k$ calls to a random permutation has collision security at most

$$\sim 2^n(1-(m-0.5r)/k)$$
Our results

- A “good” $2n$-to-$n$ bit compression function needs 3 permutations to get collision security $2^{n/2}$.
- A good $3n$-to-$2n$ bit compression function needs 5 permutations to get collision security above $2^{n/2}$.
- A good $mn$-to-$rn$ bit compression function making $k$ calls to a random permutation has collision security at most

\[ \sim 2^n(1 - (m - 0.5r)/k) \]

- A permutation-based rate $\alpha$ hash function has collision and preimage security at most $\sim 2^n(1 - \alpha)$.
The Model
Distinct-permutation setting: $\pi_i$'s are all different
The Model

- **Distinct-permutation setting**: $\pi_i$'s are all different
- **Single-permutation setting**: $\pi_1 = \pi_2 = \pi_3$
The Model

- Distinct-permutation setting: $\pi_i$'s are all different
- Single-permutation setting: $\pi_1 = \pi_2 = \pi_3$
- Order of permutations is fixed
The Pigeonhole Attack

Have $q$ queries to make.

$V \xrightarrow{f_1} \pi_1 \xrightarrow{f_2} \pi_2 \xrightarrow{f_3} g$
The Pigeonhole Attack

Have $q$ queries to make. Make queries greedily to hash maximum number of inputs.
The Pigeonhole Attack

Have $q$ queries to make.
Make queries greedily to hash maximum number of inputs.

Make $p = \frac{q}{k}$ queries to each permutation.
The Pigeonhole Attack

Have $q$ queries to make. Make queries greedily to hash maximum number of inputs.

Choose $p$ queries to start hashing maximum number of inputs.

Make $p = \frac{q}{k}$ queries to each permutation.
Choose $p$ queries to start hashing maximum number of inputs. Can average $\frac{V}{2^n}$ inputs per query.
The Pigeonhole Attack

Choose $p$ queries to start hashing. Maximum number of inputs. Can average $\frac{V}{2^n}$ inputs per query.

Get $p \frac{V}{2^n}$ inputs we can start hashing.
The Pigeonhole Attack

Get \( \frac{V}{2^n} \) inputs we can start hashing.

Choose best \( p \) queries again.

Can average \( V \frac{p}{2^n} \frac{1}{2^n} \) inputs per query.
The Pigeonhole Attack

Can continue hashing $V \left( \frac{p}{2^n} \right)^2$ inputs.

Choose best $p$ queries again.

Can average $V \frac{p}{2^n} \frac{1}{2^n}$ inputs per query.
The Pigeonhole Attack

\[ V \left( \frac{p}{2^n} \right)^k > \text{#outputs} \]
The Pigeonhole Attack

Sufficient that $V \left( \frac{p}{2^n} \right)^k > \#\text{outputs}$

Solving, get $q = k2^{n(1-(m-r)/k)}$
Theorem

Let $H : \{0, 1\}^{mn} \rightarrow \{0, 1\}^n$ be a $k$-call permutation-based compression function. Then with

$$q = k2^n(\frac{1-(m-r)/k}{k}) + k$$

queries an adversary can find a collision in $H$. 
The Pigeonhole-Birthday Attack

If outputs are random, sufficient that $V\left(\frac{p}{2^n}\right)^k > (\#\text{outputs})^{\frac{1}{2}}$
The Pigeonhole-Birthday Attack

If outputs are random, sufficient that $V \left( \frac{p}{2^n} \right)^k > \left( \#\text{outputs} \right)^{\frac{1}{2}}$

Solving, get $q = k2^n(1-(m-0.5r)/k)$
Uniformity assumption:
The outputs produced by the pigeonhole-birthday attack behave randomly with respect to collisions.
Uniformity assumption:
The outputs produced by the pigeonhole-birthday attack behave randomly with respect to collisions.

Theorem
Let $H : \{0, 1\}^{mn} \rightarrow \{0, 1\}^r$ be a $k$-call permutation-based compression function. Then, under the uniformity assumption,

$$q \approx k2^{n(1−(m−0.5r)/k)}$$

queries suffice to find a collision with probability $1/2$. 
Uniformity assumption:
The outputs produced by the pigeonhole-birthday attack behave randomly with respect to collisions.

Theorem
Let $H : \{0, 1\}^{mn} \rightarrow \{0, 1\}^r$ be a $k$-call permutation-based compression function. Then, under the uniformity assumption,

$$q \approx k2^n(1 - (m - 0.5r)/k)$$

queries suffice to find a collision with probability $1/2$.

Sufficient condition for uniformity assumption:
When an adversary learns the output values for $K$ inputs, the expected number of collisions is $\sim K^2 / (#\text{outputs})$. 
Attacking a Rate $\alpha$ Hash Function

$m$ blocks of input

$\pi_1$  $\pi_2$  $\pi_3$  $\cdots$  $m/\alpha$ permutations  $\cdots$  $\pi_k$
Attacking a Rate $\alpha$ Hash Function

$m$ blocks of input

$\pi_1, \pi_2, \pi_3, \ldots, m/\alpha$ permutations, $\ldots, \pi_k$

Pigeonhole attack: $q = k2^{n(1-(m-r)/k)} = (m/\alpha)2^{n(1-\alpha+\alpha r/m)}$
Attacking a Rate $\alpha$ Hash Function

$m$ blocks of input

$\pi_1 \quad \pi_2 \quad \pi_3 \quad \cdots \cdots \quad m/\alpha$ permutations \quad \cdots \quad \pi_k$

Pigeonhole attack: $q = k2^{n(1-(m-r)/k)} = (m/\alpha)2^{n(1-\alpha+\alpha r/m)}$

Optimize for $m \rightarrow q \approx nr2^{n(1-\alpha)}$
Theorem

Let $H : \{0, 1\}^* \to \{0, 1\}^r$ be a permutation-based hash function with rate $\alpha = 1/\beta$. Then with

$$q = [\beta \lceil \ln(2) \alpha n r + \alpha \rceil ] (e^{2n(1-\alpha)} + 1) \approx 1.89nr2^{n(1-\alpha)}$$

queries an adversary can find a collision in $H$. 
Preimage Resistance

- The pigeonhole attack yields the hash of more inputs than there are outputs, which suggests a preimage attack.
Preimage Resistance

- The pigeonhole attack yields the hash of more inputs than there are outputs, which suggests a preimage attack.
- But...

\[ n \]

\[ \pi \]

\[ n \]
The pigeonhole attack yields the hash of more inputs than there are outputs, which suggests a preimage attack.

But...

Uniformity assumption for preimage resistance (UAPR)

When an adversary learns the output values for $K$ inputs, the chance of finding any particular output is $\sim K/($#outputs$)$. 
Theorem

Let $H : \{0, 1\}^{mn} \rightarrow \{0, 1\}^r$ be a $k$-call permutation-based compression function. Then, if $H$ obeys the UAPR, with

$$q \approx k2^n(1-(m-r)/k)$$

queries an adversary can find a preimage in $H$ with probability $1/2$.

Theorem

Let $H : \{0, 1\}^* \rightarrow \{0, 1\}^r$ be a permutation-based hash function with rate $\alpha$. Then, if $H$ obeys the UAPR, with

$$q \approx 1.89nr2^n(1-\alpha)$$

queries an adversary can find a preimage in $H$ with probability $1/2$. 
“Too-Few-Wires Attack”

An $mn$-bit to $rn$-bit compression function which keeps at most $mn$ bits in memory at all times is insecure.
Recent Progress on Constructions

- Have had good progress constructing compression functions that meet the bound of the pigeonhole-birthday attack
Recent Progress on Constructions

- Have had good progress constructing compression functions that meet the bound of the pigeonhole-birthday attack.
- A $2n$-bit to $n$-bit compression function using 3 calls to a random permutation, of collision resistance $2^{n/2}$ and preimage resistance $2^{2n/3}$.
Recent Progress on Constructions

- Have had good progress constructing compression functions that meet the bound of the pigeonhole-birthday attack.

- A $2n$-bit to $n$-bit compression function using 3 calls to a random permutation, of collision resistance $2^{n/2}$ and preimage resistance $2^{2n/3}$.

- A $3n$-bit to $2n$-bit compression function using 5 calls to a random permutation, of collision resistance $2^{0.54n}$ and preimage resistance $2^{0.8n}$. 
Recent Progress on Constructions

- Have had good progress constructing compression functions that meet the bound of the pigeonhole-birthday attack.

- A $2n$-bit to $n$-bit compression function using 3 calls to a random permutation, of collision resistance $2^{n/2}$ and preimage resistance $2^{2n/3}$.

- A $3n$-bit to $2n$-bit compression function using 5 calls to a random permutation, of collision resistance $2^{0.54n}$ and preimage resistance $2^{0.8n}$.

- A $3n$-bit to $2n$-bit compression function using 6 calls to a random permutation, of collision resistance $2^{0.6n}$ and preimage resistance $2^{0.8n}$. 
Recent Progress on Constructions

- Have had good progress constructing compression functions that meet the bound of the pigeonhole-birthday attack.

- A $2n$-bit to $n$-bit compression function using 3 calls to a random permutation, of collision resistance $2^{n/2}$ and preimage resistance $2^{2n/3}$.

- A $3n$-bit to $2n$-bit compression function using 5 calls to a random permutation, of collision resistance $2^{0.54n}$ and preimage resistance $2^{0.8n}$.

- A $3n$-bit to $2n$-bit compression function using 6 calls to a random permutation, of collision resistance $2^{0.6n}$ and preimage resistance $2^{0.8n}$.

- The Shrimpton-Stam construction can be implemented with feed-forward random permutations and maintain collision resistance of $2^{n/2}$.