Chosen-prefix Collisions on AES-like Hashing

Shiyao Chen[1], Xiaoyang Dong[2], Jian Guo[1], **Tianyu Zhang**[1]

[1]Nanyang Technological University, Singapore
[2]Tsinghua University, China

March 19, 2025 @ Rome, Italy

Outline

Hash functions

A hash function maps an arbitrary-length message a to fixed-length hash value.

Hash functions need to be **resistant to collision attacks**.

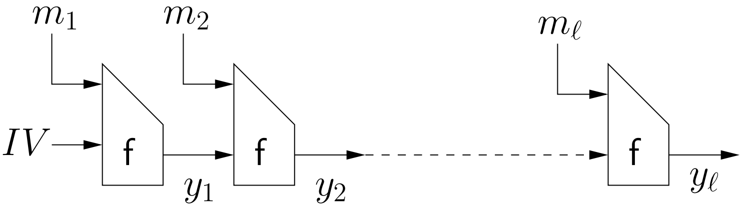


Figure: Merkle-Damgård hash function

Collision attack and variants

Given a hash function H, we have the following **variants** of collision attacks:

- Collision attack: for a chosen IV, find M_0, M_1 $s.t.$ $H(IV||M_0) = H(IV||M_1)$

Collision attack and variants

Given a hash function H, we have the following **variants** of collision attacks:

- Collision attack: for a chosen IV, find M_0, M_1 *s.t.* $H(IV||M_0) = H(IV||M_1)$
- Semi-free-start collision attack: find M_0, M_1 and IV' *s.t.* $H(IV'||M_0) = H(IV'||M_1)$
- Free-start collision attack: find M_0, M_1 and IV'_0, IV'_1 *s.t.* $H(IV'_0||M_0) = H(IV'_1||M_1)$

Collision attack and variants

Given a hash function H, we have the following **variants** of collision attacks:

- Collision attack: for a chosen IV, find M_0, M_1 $s.t.$ $H(IV||M_0) = H(IV||M_1)$
- Semi-free-start collision attack: find M_0, M_1 and IV' $s.t.$ $H(IV'||M_0) = H(IV'||M_1)$
- Free-start collision attack: find M_0, M_1 and IV_0', IV_1' $s.t.$ $H(IV_0'||M_0) = H(IV_1'||M_1)$

At Eurocrypt 2007, Stevens, Lenstra, and de Weger introduced:

- **Chosen-prefix collision (CPC) attack**: for any chosen IV_0, IV_1, find M_0, M_1 $s.t.$ $H(IV_0||M_0) = H(IV_1||M_1)$

Collision attack and variants

Given a hash function H, we have the following **variants** of collision attacks:

- Collision attack: for a chosen IV, find M_0, M_1 $s.t.$ $H(IV||M_0) = H(IV||M_1)$
- Semi-free-start collision attack: find M_0, M_1 and IV' $s.t.$ $H(IV'||M_0) = H(IV'||M_1)$
- Free-start collision attack: find M_0, M_1 and IV'_0, IV'_1 $s.t.$ $H(IV'_0||M_0) = H(IV'_1||M_1)$

At Eurocrypt 2007, Stevens, Lenstra, and de Weger introduced:

- **Chosen-prefix collision (CPC) attack**: for any chosen IV_0, IV_1, find M_0, M_1 $s.t.$ $H(IV_0||M_0) = H(IV_1||M_1)$

In terms of difficulty: CPC $>$ collision $>$ semi-free-start collision $>$ free-start collision

The practical impact of CPC attacks

There are many abuse scenarios of CPC attacks in real word applications, to list a few:

- Generation of colliding X.509 certificates for different identities [SLW07]
- Creation of rogue Certificate Authorities [SSALMOW09]
- Transcript collision attacks and SLOTH attacks on TLS, IKE, and SSH [BL16]
- PGP/GnuPG key-certification forgery [LP20]

The practical impact of CPC attacks

There are many abuse scenarios of CPC attacks in real word applications, to list a few:

- Generation of colliding X.509 certificates for different identities [SLW07]
- Creation of rogue Certificate Authorities [SSALMOW09]
- Transcript collision attacks and SLOTH attacks on TLS, IKE, and SSH [BL16]
- PGP/GnuPG key-certification forgery [LP20]

An efficient CPC attack directly marks the **retirement** of a hash function!

The practical impact of CPC attacks

There are many abuse scenarios of CPC attacks in real word applications, to list a few:

- Generation of colliding X.509 certificates for different identities [SLW07]
- Creation of rogue Certificate Authorities [SSALMOW09]
- Transcript collision attacks and SLOTH attacks on TLS, IKE, and SSH [BL16]
- PGP/GnuPG key-certification forgery [LP20]

An efficient CPC attack directly marks the **retirement** of a hash function!

Two notable series of works:

- On MD5, by Stevens *et al.* [SLW07; SSALMOW09; SLW12]
- On SHA-1, by Lurent and Peyrin [LP19; LP20]

Generic bound of collision attack (and variants)

On a hash function with n-bit output, we have generic attacks listed as follows:

	Time	Memory	Generic attack
Classical	$O(2^{n/2})$	$O(1)$ cMem	Parallel rho [OW99]

Generic bound of collision attack (and variants)

On a hash function with n-bit output, we have generic attacks listed as follows:

	Time	Memory	Generic attack
Classical	$O(2^{n/2})$	$O(1)$ cMem	Parallel rho [OW99]
Arbitrary qRAM	$O(2^{n/3})$	$O(2^{n/3})$ qRAM	BHT algorithm [BHT98]

Generic bound of collision attack (and variants)

On a hash function with n-bit output, we have generic attacks listed as follows:

	Time	Memory	Generic attack
Classical	$O(2^{n/2})$	$O(1)$ cMem	Parallel rho [OW99]
Arbitrary qRAM	$O(2^{n/3})$	$O(2^{n/3})$ qRAM	BHT algorithm [BHT98]
Without qRAM	$O(2^{2n/5})$	$O(2^{n/5})$ cMem	CNS algorithm [CNS17]

Generic bound of collision attack (and variants)

On a hash function with n-bit output, we have generic attacks listed as follows:

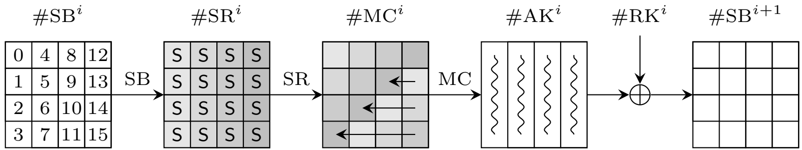
	Time	**Memory**	**Generic attack**
Classical	$O(2^{n/2})$	$O(1)$ cMem	Parallel rho [OW99]
Arbitrary qRAM	$O(2^{n/3})$	$O(2^{n/3})$ qRAM	BHT algorithm [BHT98]
Without qRAM	$O(2^{2n/5})$	$O(2^{n/5})$ cMem	CNS algorithm [CNS17]
Time-space Tradeoff	$O(2^{n/2})$	$O(1)$ cMem	Quantum parallel rho [Ber09]

We focus in the **classical** and **quantum time-space tradeoff (TSTO)** setting.

Outline

The AES round function

AES is selected by NIST in 2001 from the Rijndael block cipher family.



An encryption state of AES is organized as a $4 * 4$ grid of bytes. An AES round consists of the following operations:

- `SubBytes` (`SB`): a non-linear byte-wise substitution (S-box)
- `ShiftRows` (`SR`): a cyclic left shift on the *i*-th row by *i* bytes
- `MixColumns` (`MC`): a column-wise left multiplication of an MDS matrix
- `AddRoundKey` (`AK`): a bitwise XOR of the round key to the state

AES-like Hashing

Description

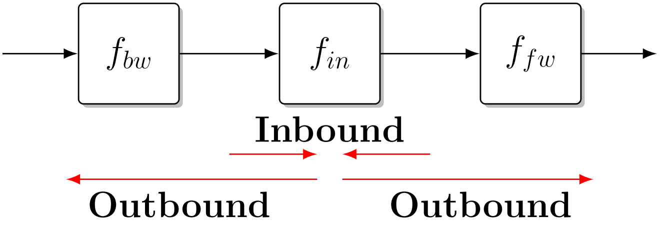
Hash functions built on an AES-like compression function are conventionally referred to as AES-like hashing

Examples include:

- AES-MMO (ISO/IEC standard and standard in the Zigbee protocol suite)
- Whirlpool (ISO/IEC standard)
- Streebog (ISO/IEC standard)
- Grøstl (NIST SHA-3 competition finalists)
- Saturnin (NIST LWC 2nd candidates)

Outline

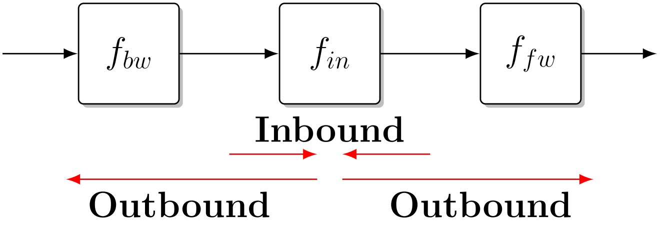
Rebound Attacks



Introduced by Mendel *et al.* at FSE 2009 [MRST09][1], the technique is a variant of differential attacks with two phases:

[1] `https://tosc.iacr.org/index.php/ToSC/ToT_Award`

Rebound Attacks

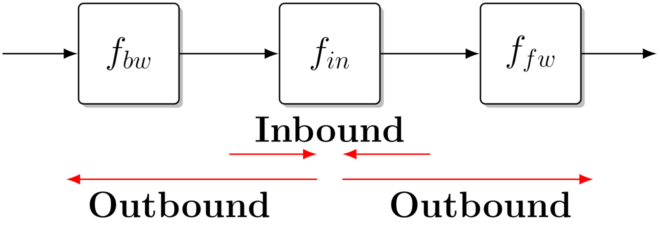


Introduced by Mendel *et al.* at FSE 2009 [MRST09][1], the technique is a variant of differential attacks with two phases:

- **Inbound phase**: allows efficient generation of **starting points** (*i.e.*, input pairs conforming with differential characteristic)

[1] `https://tosc.iacr.org/index.php/ToSC/ToT_Award`

Rebound Attacks



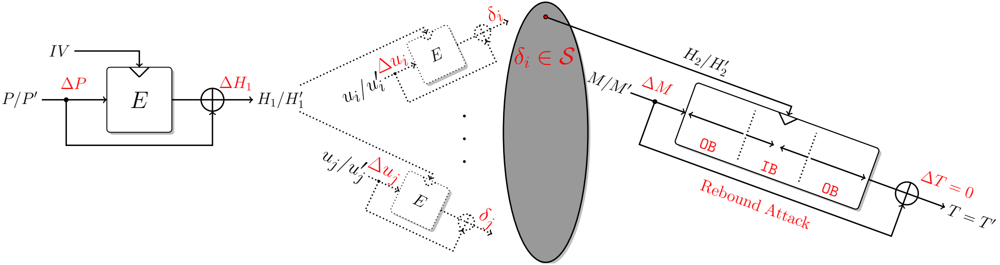
Introduced by Mendel *et al.* at FSE 2009 [MRST09][1], the technique is a variant of differential attacks with two phases:

- **Inbound phase**: allows efficient generation of **starting points** (*i.e.*, input pairs conforming with differential characteristic) → "solving the inbound"
- **Outbound phase**: probabilistically fulfills the rest constraints for collision

[1] `https://tosc.iacr.org/index.php/ToSC/ToT_Award`

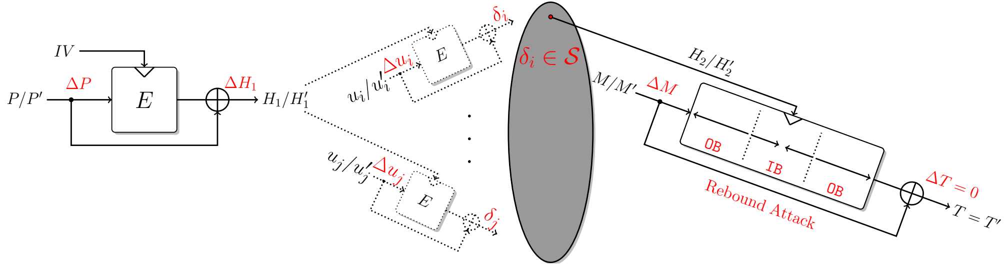
Outline

CPC attack framework based on rebound attacks



Find a class of rebound attacks and a set $\mathcal{S}$, such that for any difference in the chaining value (*i.e.*, key in MMO mode) $\delta \in \mathcal{S}$, we are able to construct a free-start collision.

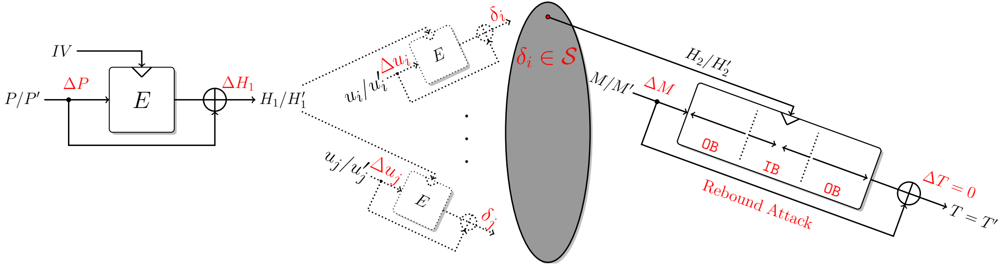
CPC attack framework based on rebound attacks



Find a class of rebound attacks and a set $\mathcal{S}$, such that for any difference in the chaining value (*i.e.*, key in MMO mode) $\delta \in \mathcal{S}$, we are able to construct a free-start collision.

1. Birthday phase: find (u_i, u_i') such that $\delta_i = CF(IV_0, u_i) \oplus CF(IV_1, u_i') \in \mathcal{S}$.

CPC attack framework based on rebound attacks



Find a class of rebound attacks and a set $\mathcal{S}$, such that for any difference in the chaining value (*i.e.*, key in MMO mode) $\delta \in \mathcal{S}$, we are able to construct a free-start collision.

1. Birthday phase: find (u_i, u_i') such that $\delta_i = CF(IV_0, u_i) \oplus CF(IV_1, u_i') \in \mathcal{S}$.
2. Rebound phase: perform the related-key rebound attack and according to δ_i.

Complexity analysis

Birthday phase: the time complexity to find proper (u_i, u_i') is $\sqrt{2^n/|\mathcal{S}|}$ in quantum TSTO and classical setting.

Rebound phase: assuming the probability of the outbound phase as p,

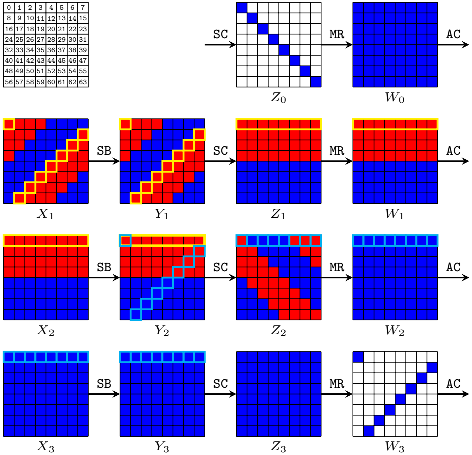
- in classical setting, assuming the time complexity to find one starting point is $\mathcal{T}^c_{\mathtt{IB}}$, the time complexity is $\mathcal{T}^c_{\mathtt{IB}}/p$.
- in quantum TSTO, assuming the time complexity to find one starting point is $\mathcal{T}^q_{\mathtt{IB}}$, the time complexity is $\mathcal{T}^q_{\mathtt{IB}}/\sqrt{p}$.

Remark:

- CPC attacks are backward compatible to collision attacks
- The framework is also a **conversion** from (a particular type of) free-start collision attacks to two-block collision attacks

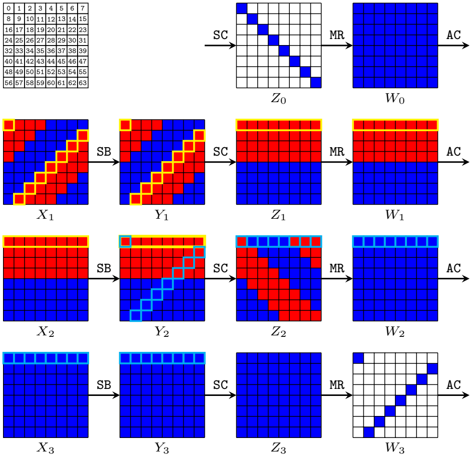
Outline

Hosoyamada and Sasaki's memoryless technique [HS20]



"Solving the inbound": given any $\Delta Z_0, \Delta W_3$ (equiv. $\Delta X_1, \Delta Y_3$), generate starting point Z_0, Z_0'.

Hosoyamada and Sasaki's memoryless technique [HS20]

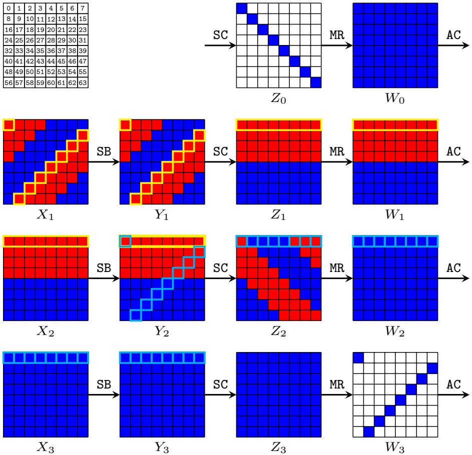


"Solving the inbound": given any $\Delta Z_0, \Delta W_3$ (equiv. $\Delta X_1, \Delta Y_3$), generate starting point Z_0, Z_0'.

Steps:

1. Enumerate X_1[■], compute Z_2[■], Z_2'[■]
2. For row i, enumerate Z_2[■], Z_2'[■], compute full row i of Y_3, Y_3', check if they comply with ΔY_3
3. After all rows of Y_3, Y_3' are recovered, compute backward to check if they comply with ΔX_1[■]

Hosoyamada and Sasaki's memoryless technique [HS20]



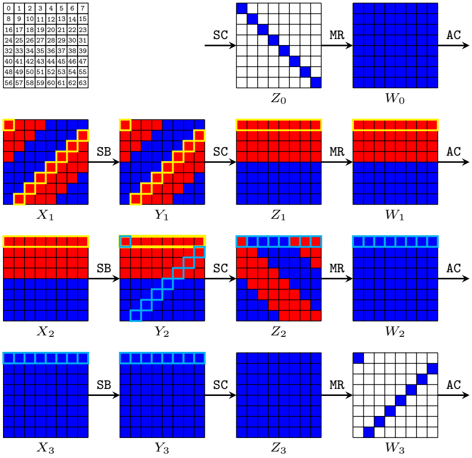
"Solving the inbound": given any $\Delta Z_0, \Delta W_3$ (equiv. $\Delta X_1, \Delta Y_3$), generate starting point Z_0, Z_0'.

Steps:

1. Enumerate X_1[■], compute Z_2[■], Z_2'[■]
2. For row i, enumerate Z_2[■], Z_2'[■], compute full row i of Y_3, Y_3', check if they comply with ΔY_3
3. After all rows of Y_3, Y_3' are recovered, compute backward to check if they comply with ΔX_1[■]

Time: $t_C = 2^{8\cdot(d^2/2+d/2+d/2)}$, $t_Q = \sqrt{t_C}$
Memory: $O(1)$

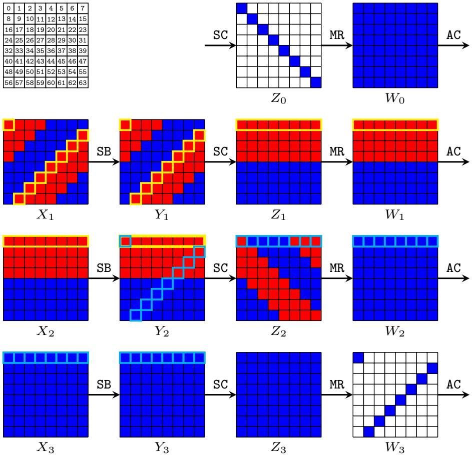
Improved memoryless algorithm to solve 3-round inbound



We improve **Step II**, now for each row:

- Instead of enumerating $Z_2[\blacksquare]$, $Z'_2[\blacksquare]$, and check with ΔY_3

Improved memoryless algorithm to solve 3-round inbound

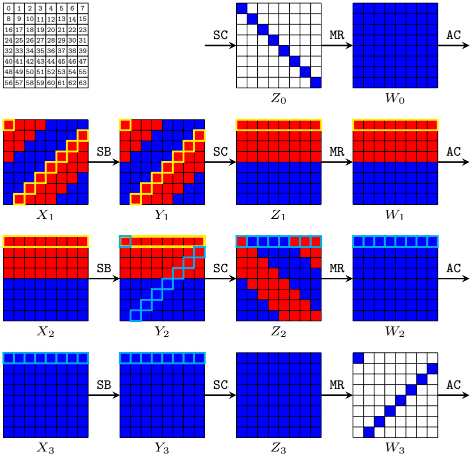


We improve **Step II**, now for each row:

- Instead of enumerating $Z_2[\blacksquare], Z_2'[\blacksquare]$, and check with ΔY_3
- We enumerate $Z_2[\blacksquare]$, compute Y_3, and obtain $Y_3' = \Delta Y_3 \oplus Y_3$

This replaces the filtering on ΔY_3

Improved memoryless algorithm to solve 3-round inbound



We improve **Step II**, now for each row:

- Instead of enumerating $Z_2[\blacksquare]$, $Z_2'[\blacksquare]$, and check with ΔY_3
- We enumerate $Z_2[\blacksquare]$, compute Y_3, and obtain $Y_3' = \Delta Y_3 \oplus Y_3$

This replaces the filtering on ΔY_3

Time: $t_C = 2^{8\cdot(d^2/2+d/2)}$, $t_Q = \sqrt{t_C}$
Memory: $O(1)$

Outline

Target	Type of Attack	Rounds	Time	C-Mem	qRAM	Setting	Source
Hash function	Collision	4/10	2^{120}	2^{16}	-	Classic	[MRST09]
	Collision	5/10	2^{120}	2^{64}	-	Classic	[LMRRS09; GP10]
	Collision	6/10	2^{248}	2^{248}	-	Classic	[DHSLWH21]
	Collision	6/10	2^{240}	2^{240}	-	Classic	[CGLSZ24]
	Collision	6/10	2^{228}	-	-	Quantum	[HS20]
	Collision	6/10	$2^{201.4}$	-	-	Quantum	This work
	Collision/CPC	6/10	$2^{205.4}$	-	-	Quantum	This work
Compression function	Semi-free	5/10	2^{120}	2^{16}	-	Classic	[MRST09]
	Semi-free	7/10	2^{184}	2^{8}	-	Classic	[LMRRS09]
	Free-start	8/10	2^{120}	2^{8}	-	Classic	[SWWW12]
	Free-start	9/10	$2^{220.5}$	-	-	Quantum	[DZSWWH21]
	Free-start	9/10	$2^{204.53}$	-	-	Quantum	This work

Outline

Target	Type of Attack	Rounds	Time	C-Mem	qRAM	Setting	Source
Hash function	Collision	5/16	2^{64}	2^{64}	-	Classic	[DZSWWH21]
	Collision	7/16	$2^{113.5}$	-	-	Quantum	[DZSWWH21]
	Collision/CPC	6/16	2^{112}	2^{64}	-	Classic	This work
	Collision/CPC	7/16	2^{113}	-	-	Quantum	This work

Outline

Target	Type of Attack	Rounds	Time	C-Mem	qRAM	Setting	Source
Hash function	Collision	5/10	2^{56}	2^{16}	-	Classic	[MRST09]
	Collision	6/10	2^{56}	2^{32}	-	Classic	[LMRRS09; GP10]
	Collision	7/10	$2^{42.5}$	-	2^{48}	Quantum	[HS20]
	Collision	7/10	$2^{59.5}$	-	-	Quantum	[HS20]
	Collision	7/10	$2^{45.8}$	-	-	Quantum	[DSSGWH20]
	Collision	8/10	$2^{55.53}$	-	-	Quantum	[DGLP22]
	Collision/CPC	5/10	2^{57}	2^{32}	-	Classic, MMO	This work
	Collision/CPC	5/10	2^{52}	2^{32}	-	Classic, MP	This work
	Collision/CPC	6/10	2^{61}	-	-	Quantum	This work

Thank you for listening :)