SHA-1 : Beating a Dead Horse

Thomas Peyrin

(joint work with Gaëtan Leurent)

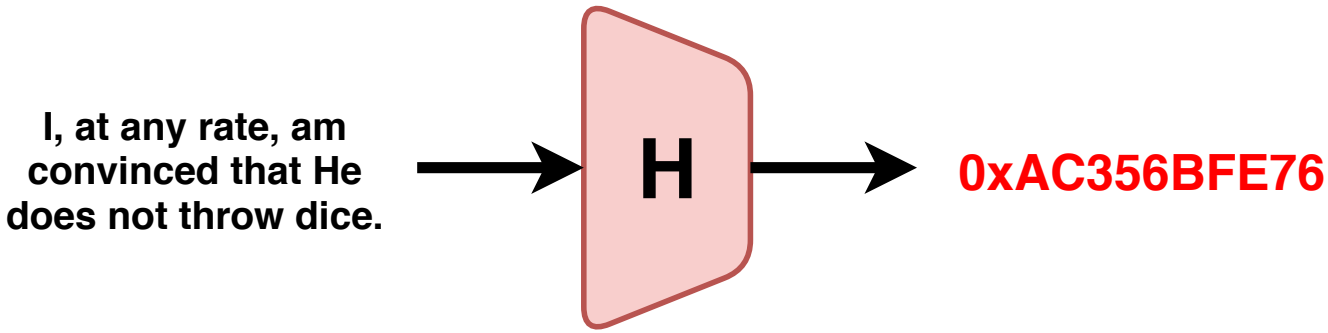
NTU - Singapore

ASK 2019

Kobe (Japan) - December 13, 2019

Outline

What is a Hash Function ?



- H maps an **arbitrary length input** (the message M) to a **fixed length output**. Typically $n = 128$ (MD5), $n = 160$ (SHA-1) or $n = 256$ bits (SHA-256).
- no secret parameter.
- H must be easy to compute.

The security goals

pre-image resistance :

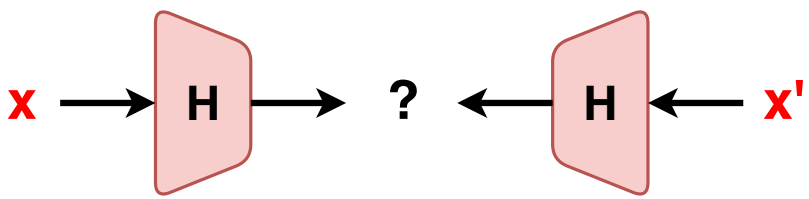
given an output challenge y, the attacker can not find a message x such that $H(x) = y$, in less than $\theta(2^n)$ operations.

2nd pre-image resistance :

given a challenge (x, y) so that $H(x) = y$, the attacker can not find a message $x' \neq x$ such that $H(x') = y$, in less than $\theta(2^n)$ operations.

collision resistance :

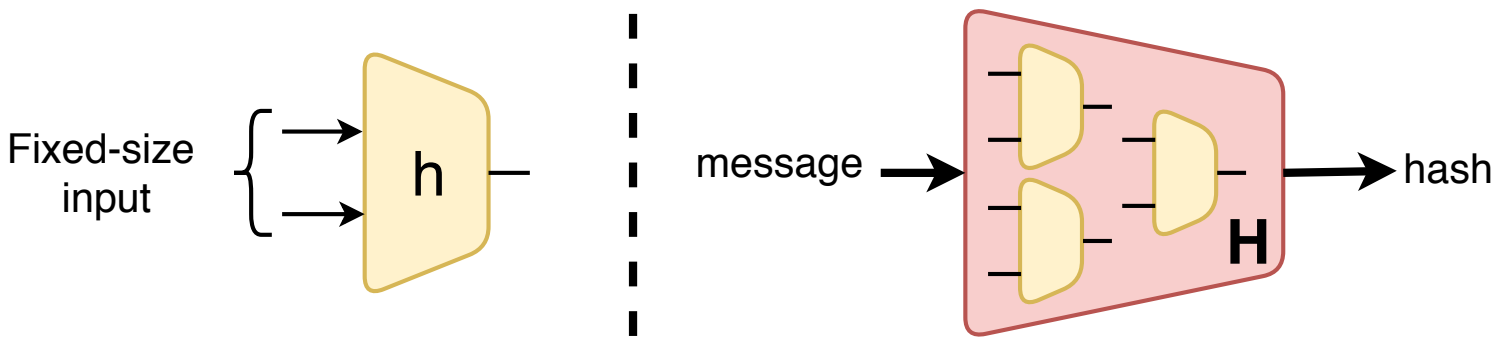
the attacker can not find two messages (x, x') such that $H(x) = H(x')$, in less than $\theta(2^{n/2})$ operations (a generic attack with the birthday paradox exists [Yuval-79]).



General hash construction

For historical reasons, most hash functions are composed of two elements :

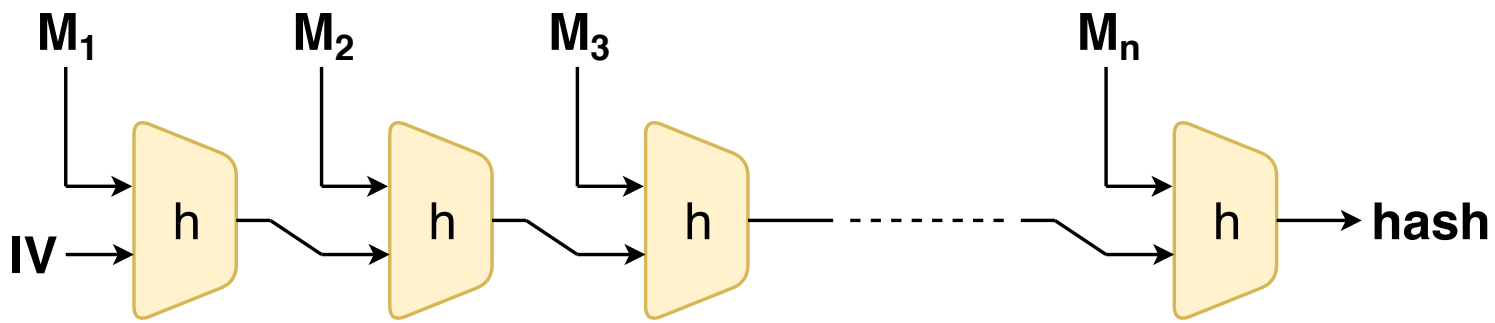
- **a compression function** h **:** a function for which the input and output size is fixed.
- **a domain extension algorithm :** an iterative process that uses the compression function h so that the hash function H can handle inputs of arbitrary length.



The Merkle-Damgård domain extension algorithm

The most famous domain extension algorithm used is called the **Merkle-Damgård** [Merkle Damgård-89] iterative algorithm.

$$\mathbf{pad(M) = M_1 \,||\, M_2 \,||\, M_3 \,||\, \ldots \,||\, M_n}$$



The compression function *h* now takes two fixed-size inputs, the incoming chaining variable *cv* and the message block *m*, and outputs a new chaining variable.

The sad story of MD5

The cryptanalysis history of MD5 is a good example of why **(semi)-free-start collisions are a serious warning**.

- 1992 MD5 RFC published
- 1993 pseudo-collision on the compression function [BB93]
- 1994 semi-free-start collision on the compression function [Dob96]
- 2004 practical collisions on the hash function [WFL04]
- 2007 chosen-prefix collisions and colliding X.509 certificates [SLW07]
- 2009 rogue CA certificates for RapidSSL [S+09] (used chosen-prefix collision)
- 2010-2012 Flame malware (used chosen-prefix collision)

What about SHA-1 ?

The current situation of SHA-1 :

1995	SHA-1 NIST FIPS 180-1 published : basically SHA-0 (1993) with a very small twist
2005	theoretical collision attack on the full hash function [WYY05] - 2^{69}
2006-2011	lots of works computing actual collisions for reduced-round versions of SHA-1
2015	free-start collision on the full compression computed [SKP15] - 2^{57}
2017	full collision on the full hash function computed [SBKAM17] - $2^{64.7}$
2019	chosen-prefix collision attack [LP19] - $2^{67.2}$
New	chosen-prefix collision attack - $2^{63.7}$ + PGP/GnuPG key-certification forgery

Motivations to study SHA-1

Why still studying SHA-1 ?

Design from NSA, **Standard** from NIST, **Still used worldwide** despite deprecation efforts (major browsers now refusing to connect to servers still using SHA-1- based certificates) :

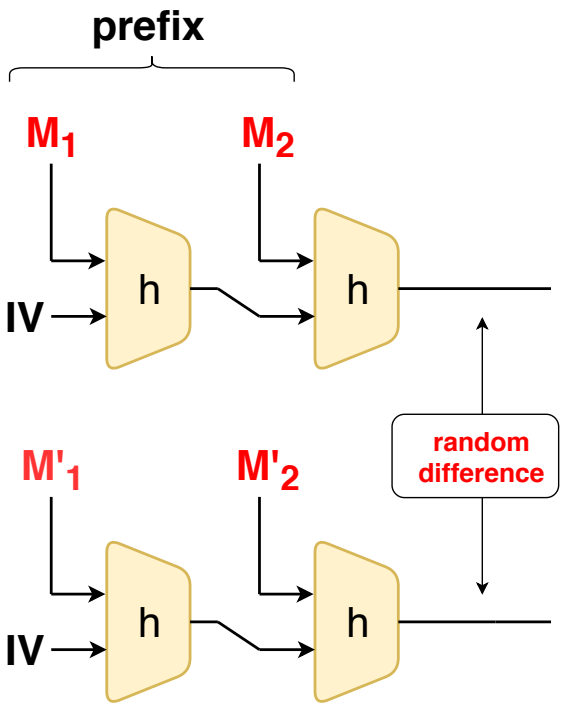
- more than 5% of Alexa's top 1 million **prefer** to use SHA-1 to authenticate TLS handshake messages (including www.skype.com)
- about 30000 servers with SHA-1 certificates (out of 720000 servers with HTTPS support)
- other protocols : about 1 million out of 4.55 millions mail servers (with IMAPS) use a SHA-1 certificate
- it is still possible to buy a SHA-1 certificate from a trusted root, and some can be found in the wild
- the "Mail" application included in Windows 10 still accepts SHA-1 certificates without warnings when opening an IMAPS connection

Yet another push is perhaps needed to accelerate the retirement of SHA-1

What are chosen-prefix collisions ?

Chosen-prefix collision attack

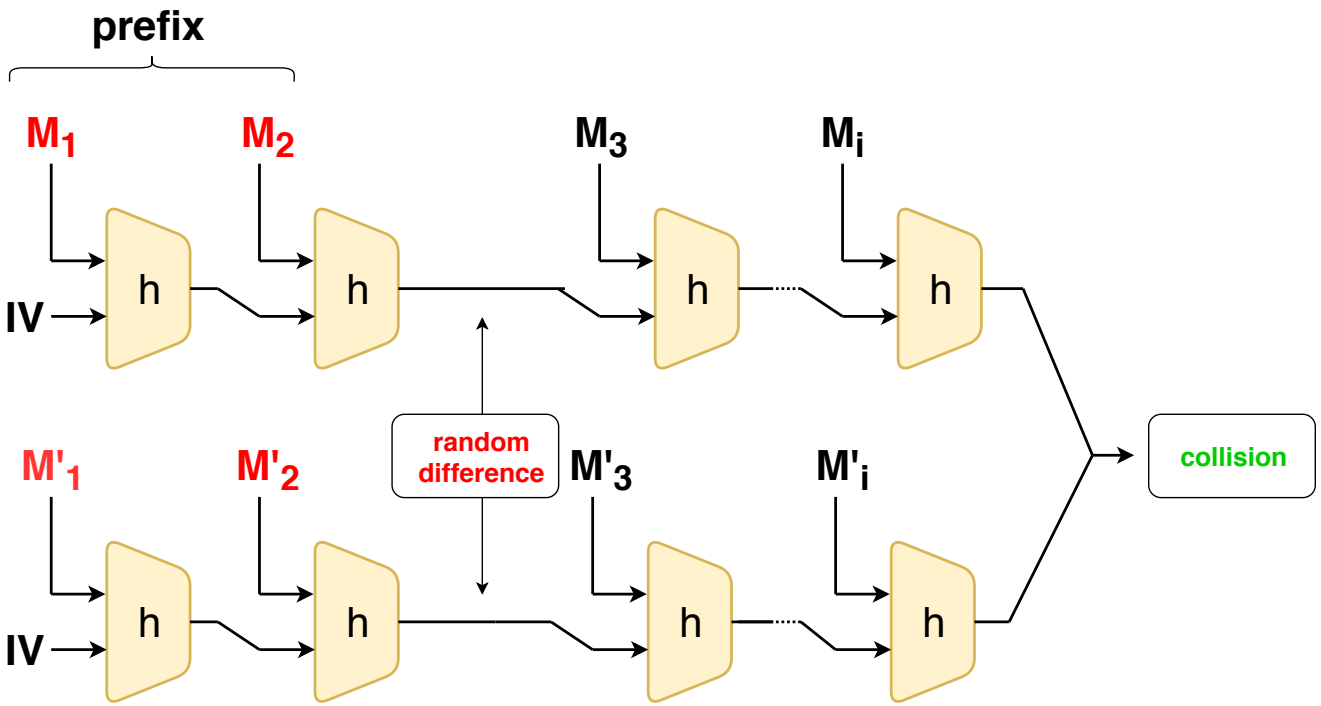
The attacker is first challenged with **two message prefixes** P and P', and its goal is to compute two messages M and M' to create the **collision** $H(P||M) = H(P'||M')$, where $||$ denotes concatenation



What are chosen-prefix collisions ?

Chosen-prefix collision attack

The attacker is first challenged with **two message prefixes** P and P', and its goal is to compute two messages M and M' to create the **collision** $H(P||M) = H(P'||M')$, where $||$ denotes concatenation



What are chosen-prefix collisions ?

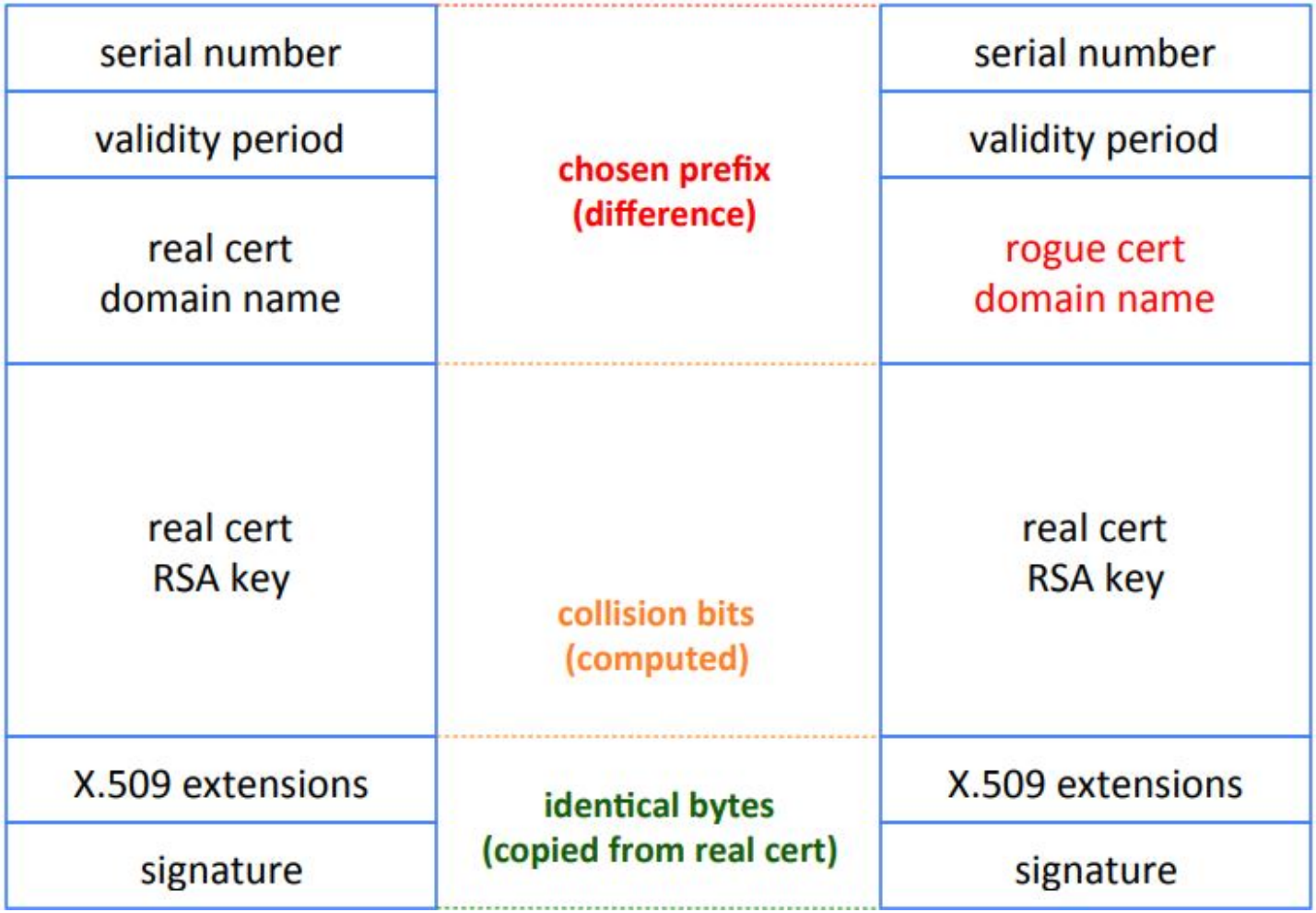
Chosen-prefix collision attack

The attacker is first challenged with **two message prefixes** P and P', and its goal is to compute two messages M and M' to create the **collision** $H(P||M) = H(P'||M')$, where $||$ denotes concatenation

- **much more powerful** than a simple collision attack (i.e. rogue CA certificate)
- supposedly much harder than a simple collision attack (currently for MD5, from 2^{16} to 2^{39})
- birthday attack can apply, thus generic cost remains $2^{n/2}$ (i.e. 2^{80} in the case of SHA-1)

Why chosen-prefix collisions ?

Colliding SSL certificates [S+09] :



Alex Sotirov https://trailofbits.files.wordpress.com/2012/06/flame-md5.pdf

Chosen-prefix collisions for SHA−1

Previous status of chosen-prefix collisions on SHA−1 :

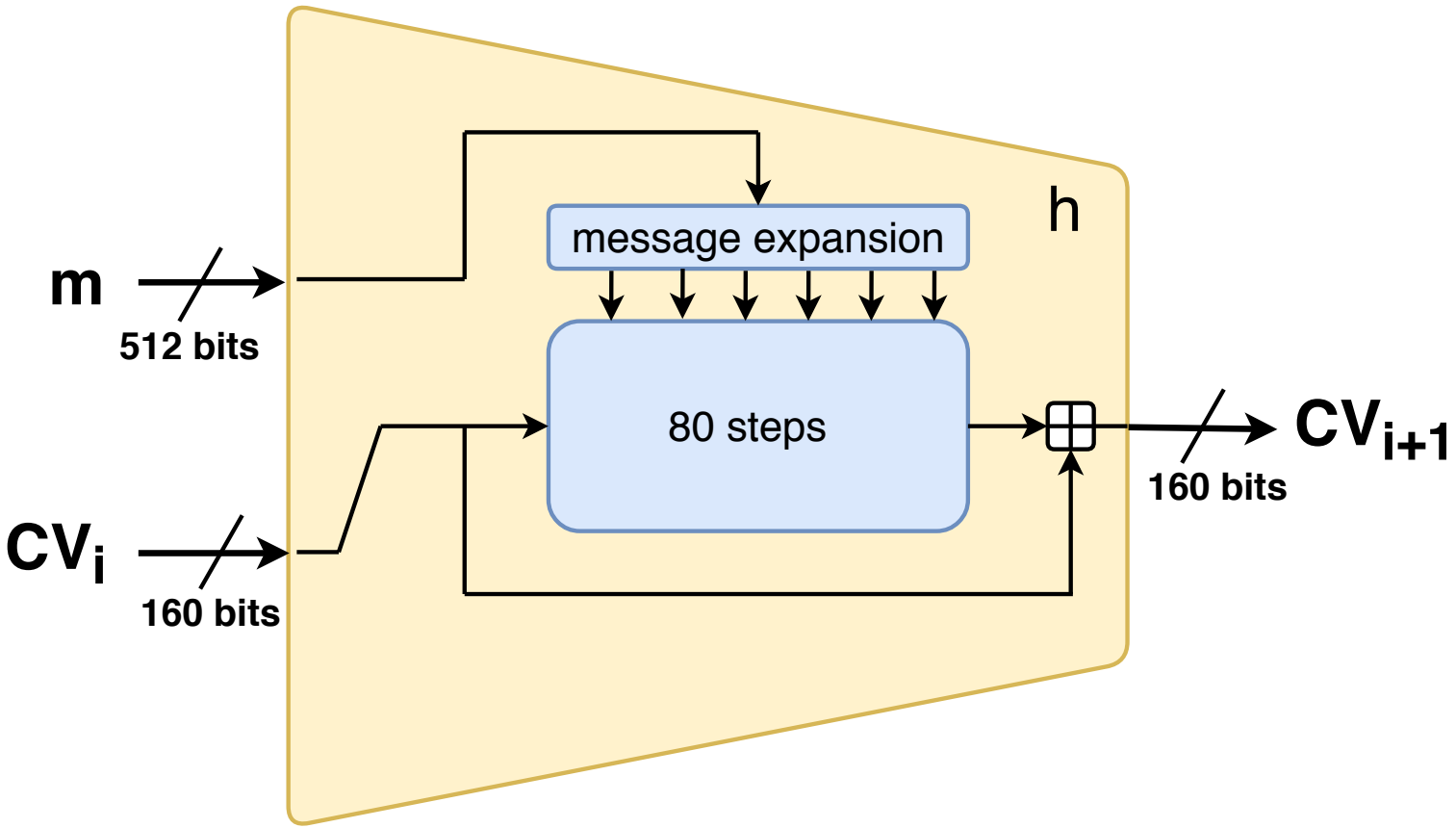
- **previous best known chosen-prefix attack against SHA−1 required $2^{77.1}$ computations** [S13]
  (a factor 8 better than generic attack).
- ... while classical collision could be found with $2^{64.7}$ computations.
- one can't apply directly the SHA−1 collision attack, because of the random state difference due to the challenge prefix.

Can we reduce the gap and make chosen-prefix collisions practical for SHA−1 ?

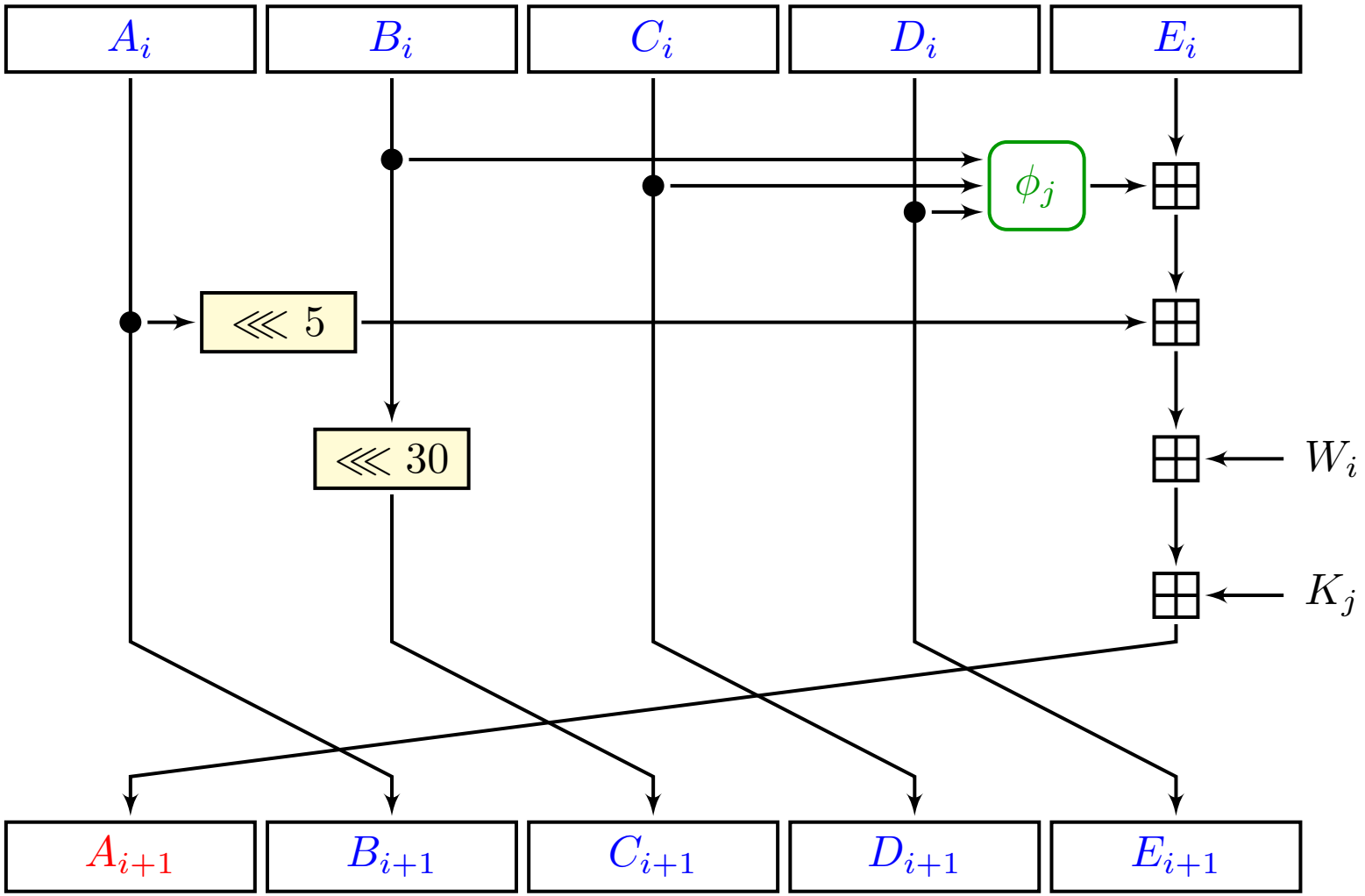
Outline

The SHA-1 compression function

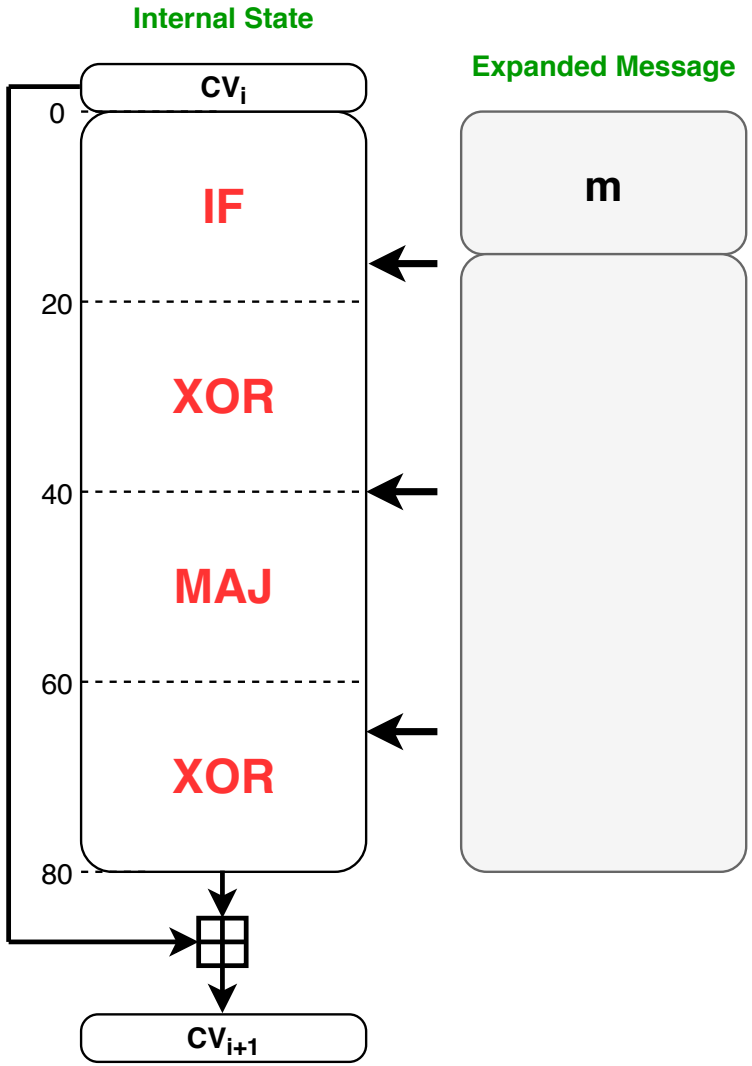
$$m = M_0 || M_1 || \cdots || M_{15}$$

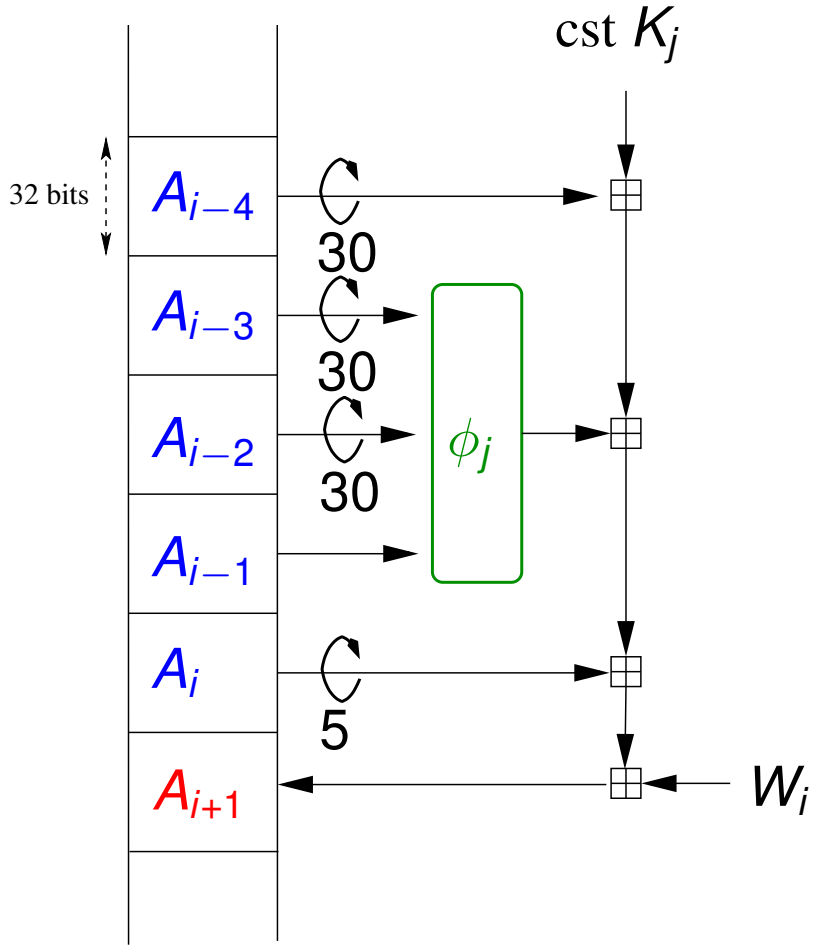


The SHA-1 step function



The SHA−1 step function : alternative representation



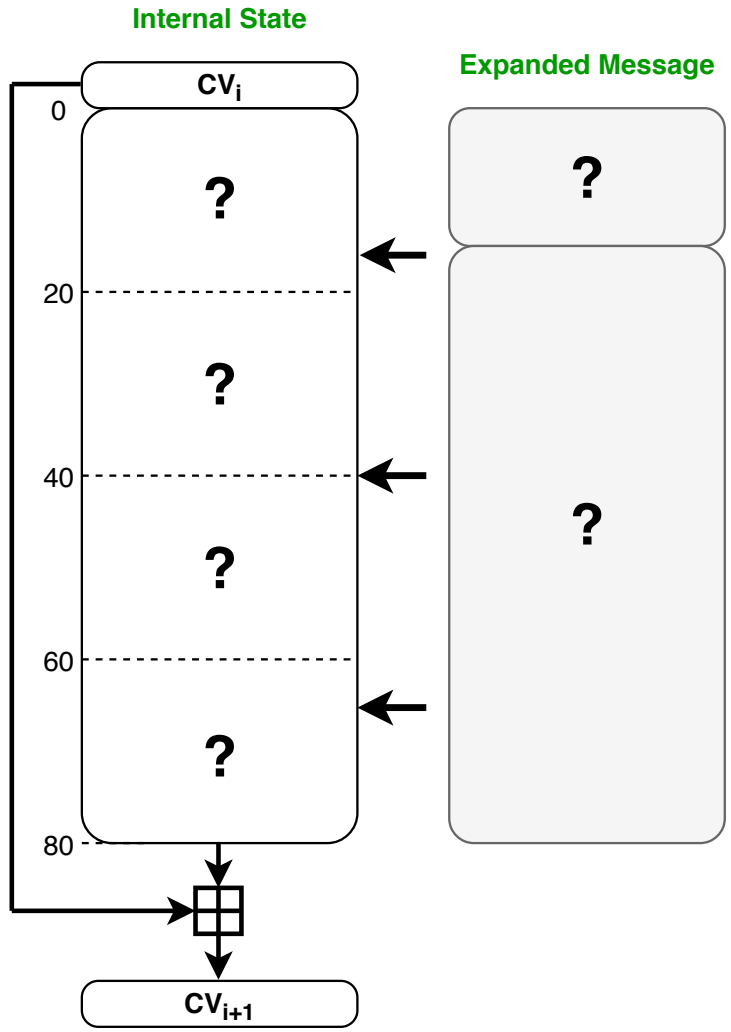


step i, round j

Outline

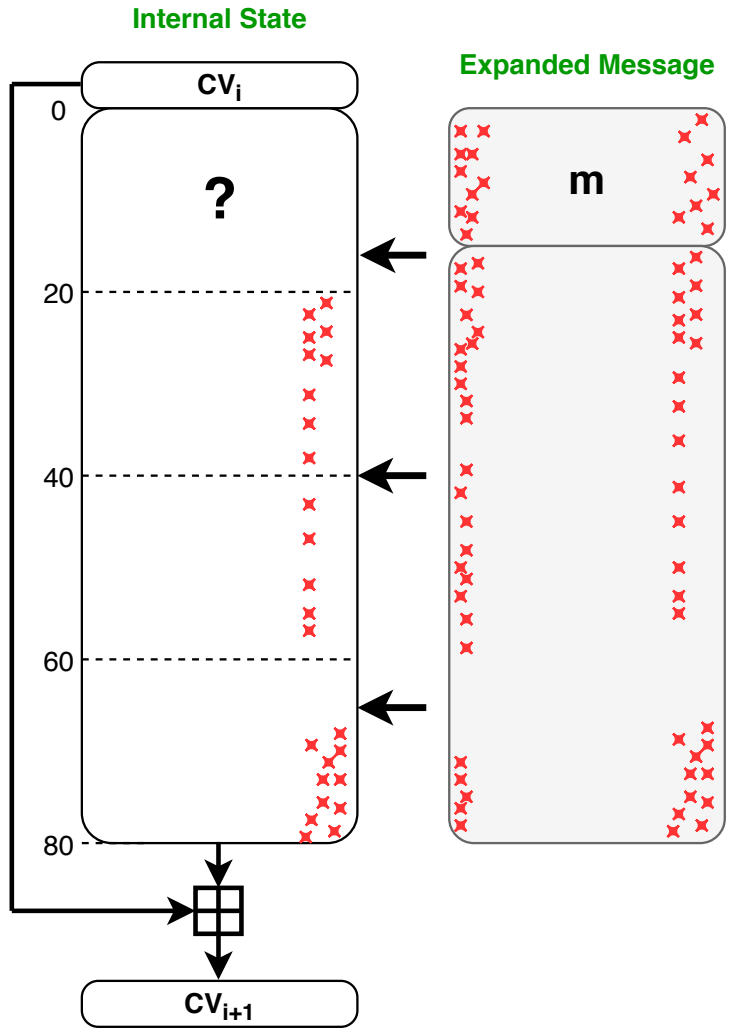
Goal : Find a collision for SHA-1

- **Find a linear path for rounds 16-80, using local collisions** this will fix the entire message difference, and the internal state difference for steps 16-80
- **Find a non-linear path for steps 1-15, using heuristic algorithm** this will fix the internal state difference for rounds 1-15
- **Prepare the collision search speed-up techniques** by using the freedom degrees available
- **Launch the collision search !**



Goal : Find a collision for SHA-1

- **Find a linear path for rounds 16-80, using local collisions** this will fix the entire message difference, and the internal state difference for steps 16-80
- Find a **non-linear path** for steps 1-15, using heuristic algorithm this will fix the internal state difference for rounds 1-15
- **Prepare the collision search speed-up techniques** by using the freedom degrees available
- **Launch the collision search !**



Structure of the differential path

Why did we remove the 16 first steps ?

- it allows to **avoid impossibilities due to the IF function**
- impossible to get a one-block collision with good probability : necessary to use several blocks
- A non-linear part allows to **start from any incoming difference in the chaining variable** : 2-block collision

Structure of the differential path

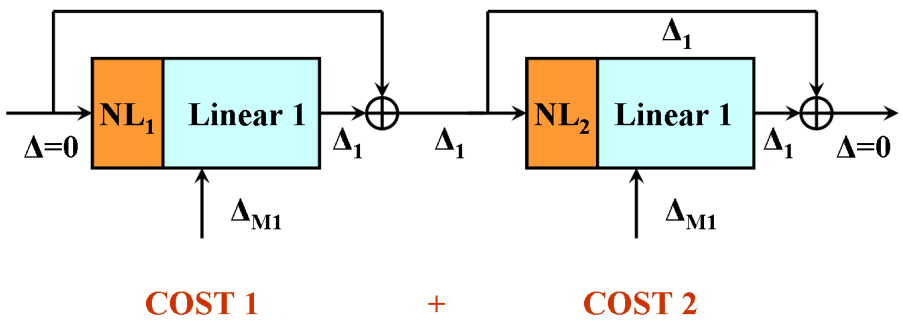
Why did we remove the 16 first steps ?

- it allows to **avoid impossibilities due to the IF function**
- impossible to get a one-block collision with good probability : necessary to use several blocks
- A non-linear part allows to **start from any incoming difference in the chaining variable** : 2-block collision

Structure of the differential path

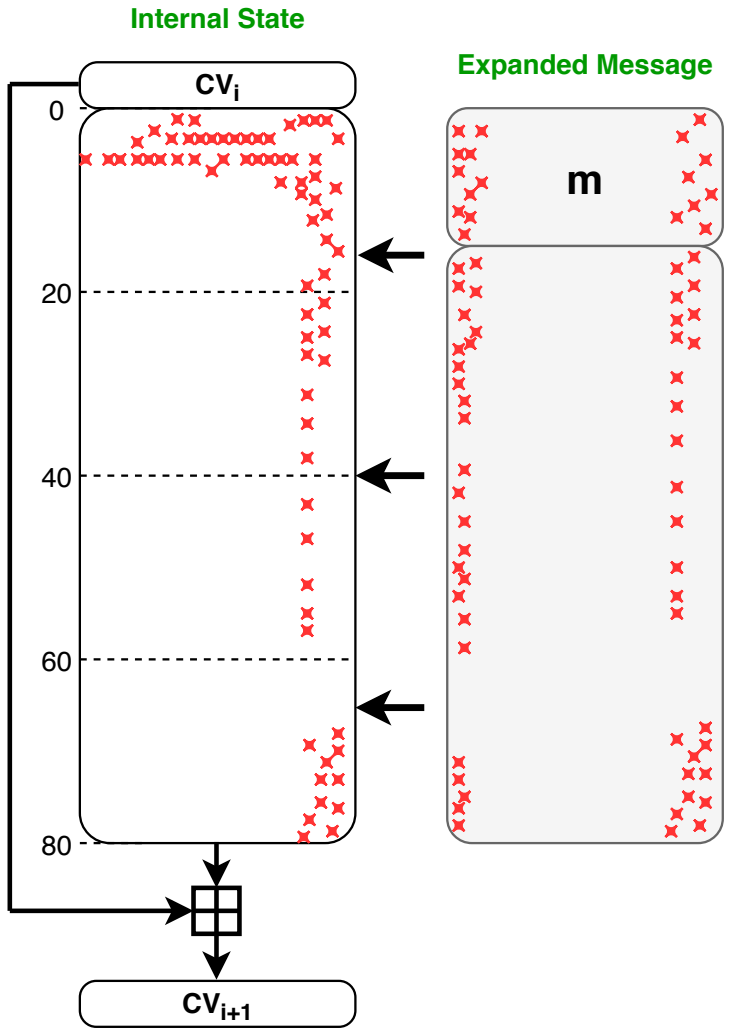
Why did we remove the 16 first steps ?

- it allows to **avoid impossibilities due to the IF function**
- impossible to get a one-block collision with good probability : necessary to use several blocks
- A non-linear part allows to **start from any incoming difference in the chaining variable** : 2-block collision

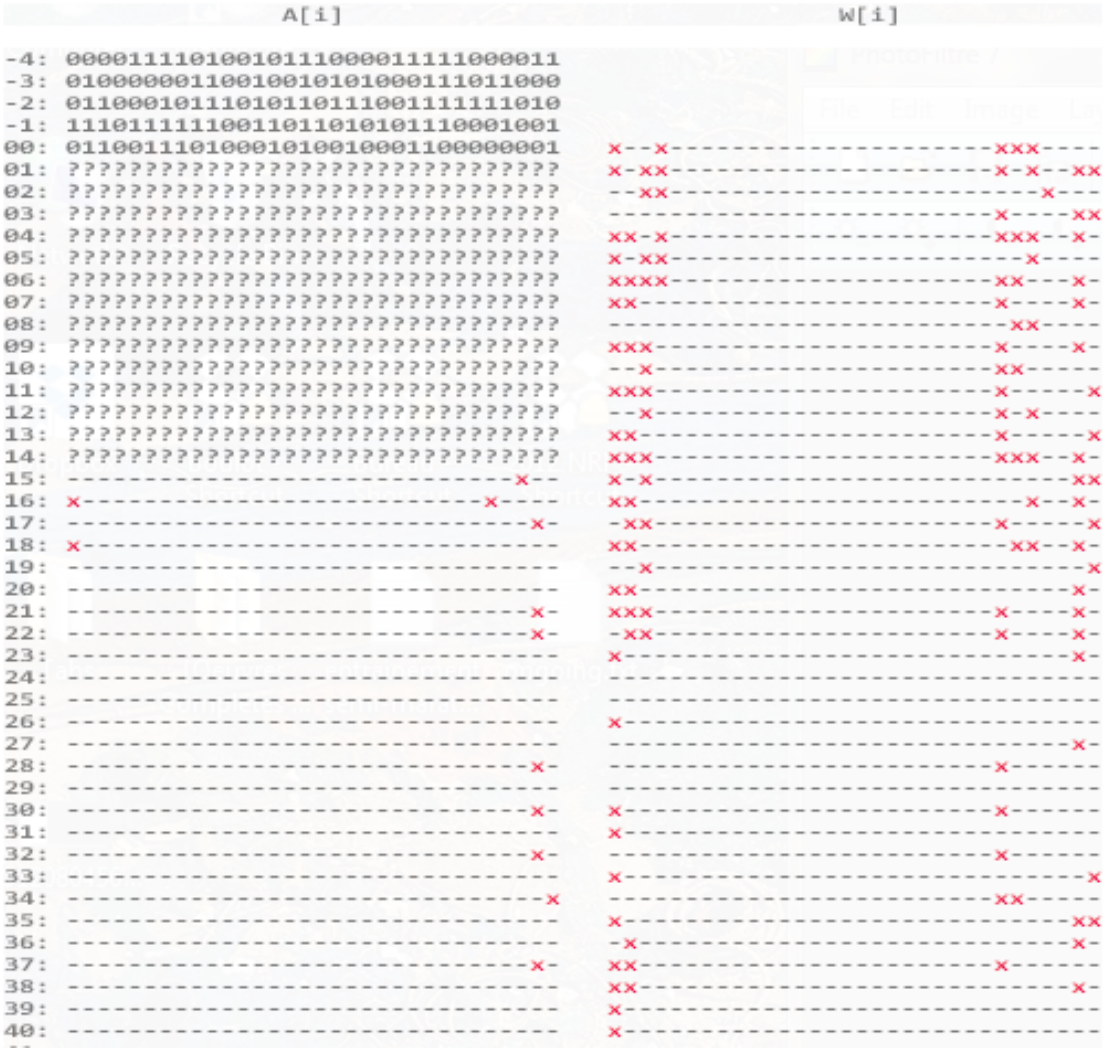


Goal : Find a collision for SHA-1

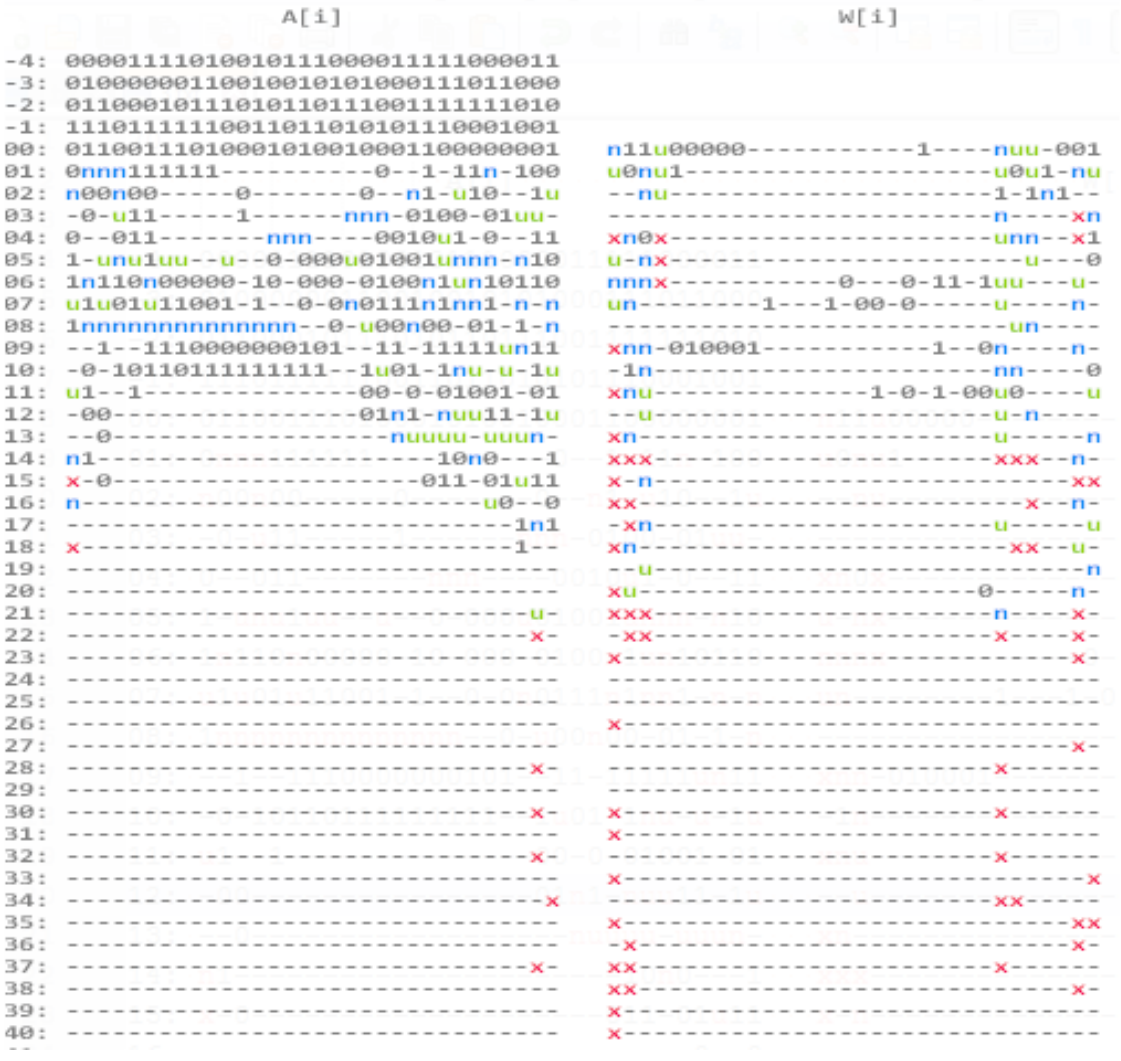
- **Find a linear path for rounds 16-80, using local collisions** this will fix the entire message difference, and the internal state difference for steps 16-80
- **Find a non-linear path for steps 1-15, using heuristic algorithm this will fix the internal state difference for rounds 1-15**
- **Prepare the collision search speed-up techniques** by using the freedom degrees available
- **Launch the collision search !**



Example of a non-linear path



Example of a non-linear path



Generating a non-linear path

The non-linear path search algorithm

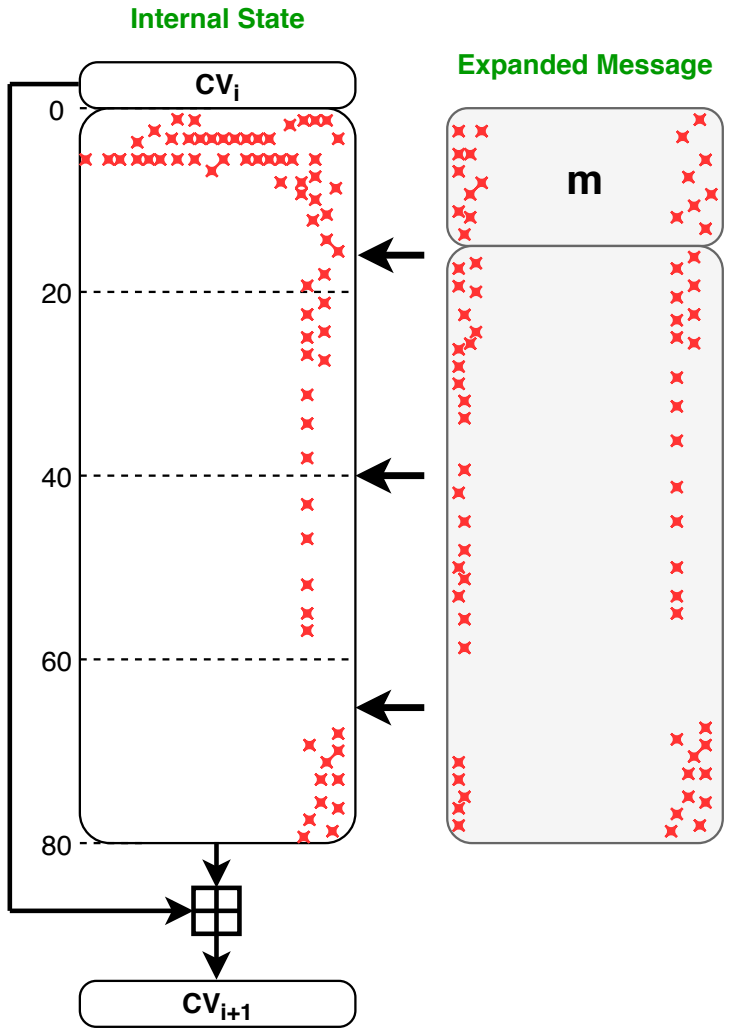
Input : a differential path with '?' only in the internal state in steps 1-14, with steps 14-20 being determined pseudo-linearly

Output : a differential path where no '?' nor 'x' exist anymore in the internal state : the path is fully determined and signed

- **very sensitive to many parameters**, **quite technical** and **hard to make it right**
- heuristic strategy [CR06] or using SAT solvers [SBKAM17]

Goal : Find a collision for SHA-1

- **Find a linear path for rounds 16-80, using local collisions** this will fix the entire message difference, and the internal state difference for steps 16-80
- **Find a non-linear path for steps 1-15, using heuristic algorithm** this will fix the internal state difference for rounds 1-15
- **Prepare the collision search speed-up techniques** by using the freedom degrees available
- **Launch the collision search !**



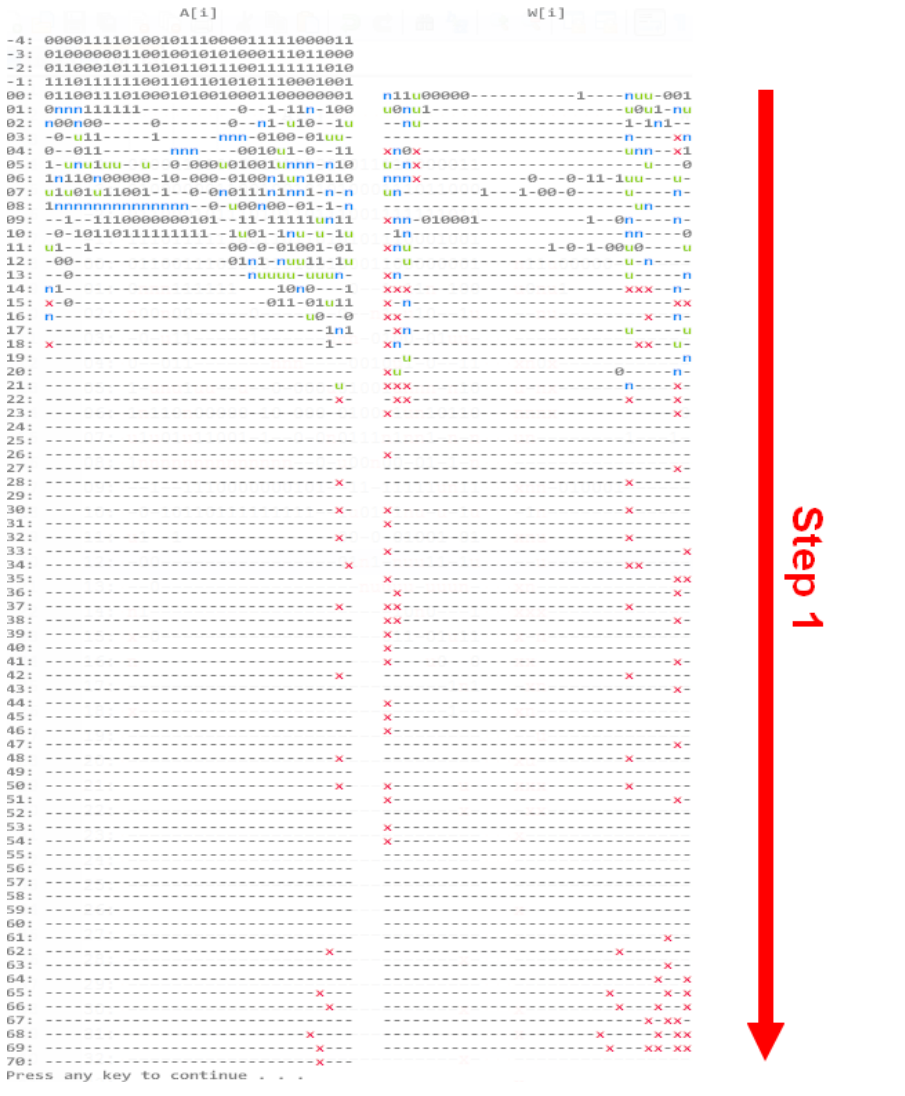
A simple collision search algorithm

A naive collision search algorithm

Repeat until a collision is found :

- pick a random message
- test if it follows the entire differential path

The **very costly** non-linear part has to be paid ☹ ☹ ☹



A better collision search algorithm

A less-naive search algorithm :

$i = 1$

repeat until a collision is found :

- pick a random message word M_i (backtrack sometimes)
- test if it follows the differential path for step i
  - if it does, then $i = i + 1$
- when $i = 16$, test if it follows the entire differential path

The very costly non-linear part is avoided ☺ ☺ ☺, **only the linear probabilistic part remains to be paid**

Final collision search algorithm

Final collision search in 3 phases :

- **Step 1 :** handling the low-probability non-linear parts using the message block freedom
- **Step 2 :** apply the collision search speed-up techniques
- **Step 3 :** the remaining steps are verified probabilistically

Computation cost further reduced 😄 😄 😄 only steps ˜22-80 have to be considered

Collision search speed-up techniques

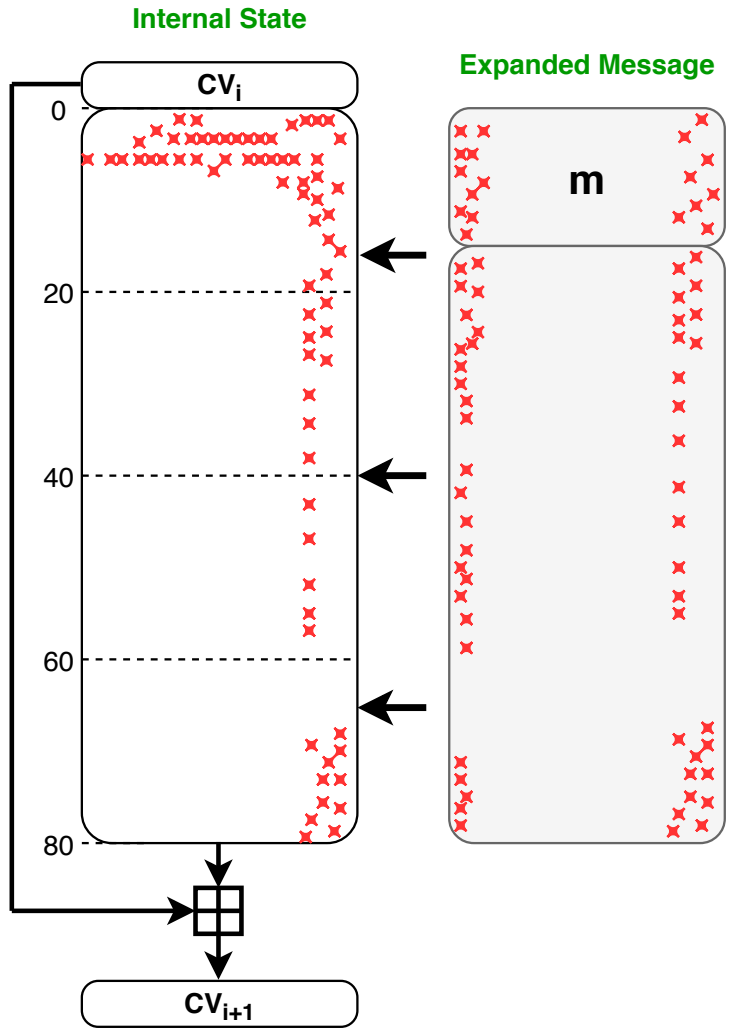
Several techniques to speed-up the collision search :

- message modifications [WYY05]
- neutral bits [BC04]
- boomerangs (or tunnels) [K06,JP07]

All these techniques trade message freedom degrees for a speed-up factor

Goal : Find a collision for SHA-1

- **Find a linear path for rounds 16-80, using local collisions** this will fix the entire message difference, and the internal state difference for steps 16-80
- **Find a non-linear path for steps 1-15, using heuristic algorithm** this will fix the internal state difference for rounds 1-15
- **Prepare the collision search speed-up techniques** by using the freedom degrees available
- **Launch the collision search !**

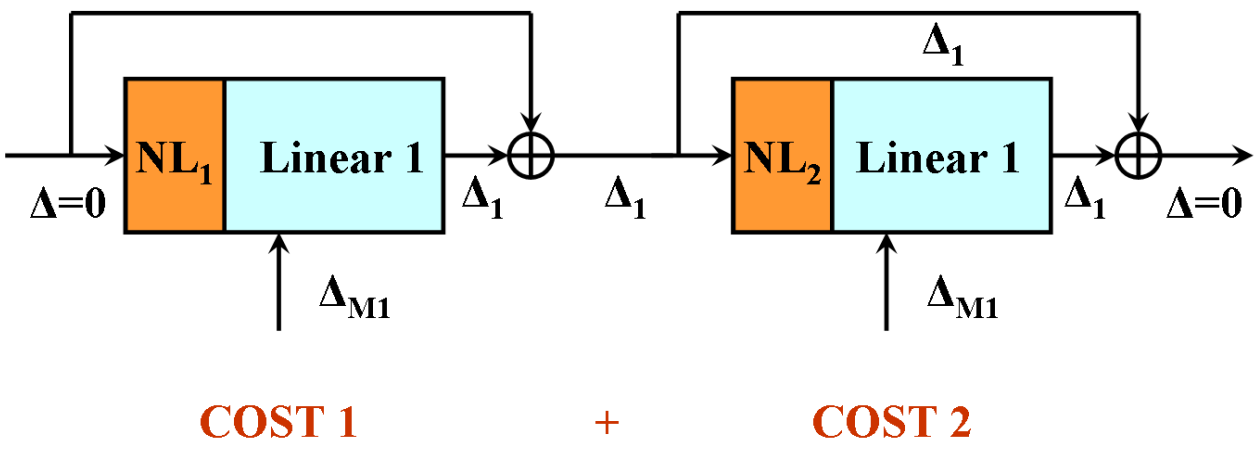


Outline

Why chosen-prefix collision is hard for SHA-1

Can the SHA-1 collision attack be directly adapted for chosen-prefix scenario ?

No : we can't remove the random difference on the chaining variable with the very small number of possible output differences of the linear path



Why chosen-prefix collision is hard for SHA-1

Can the SHA-1 collision attack be directly adapted for chosen-prefix scenario ?

No : we can't remove the random difference on the chaining variable with the very small number of possible output differences of the linear path

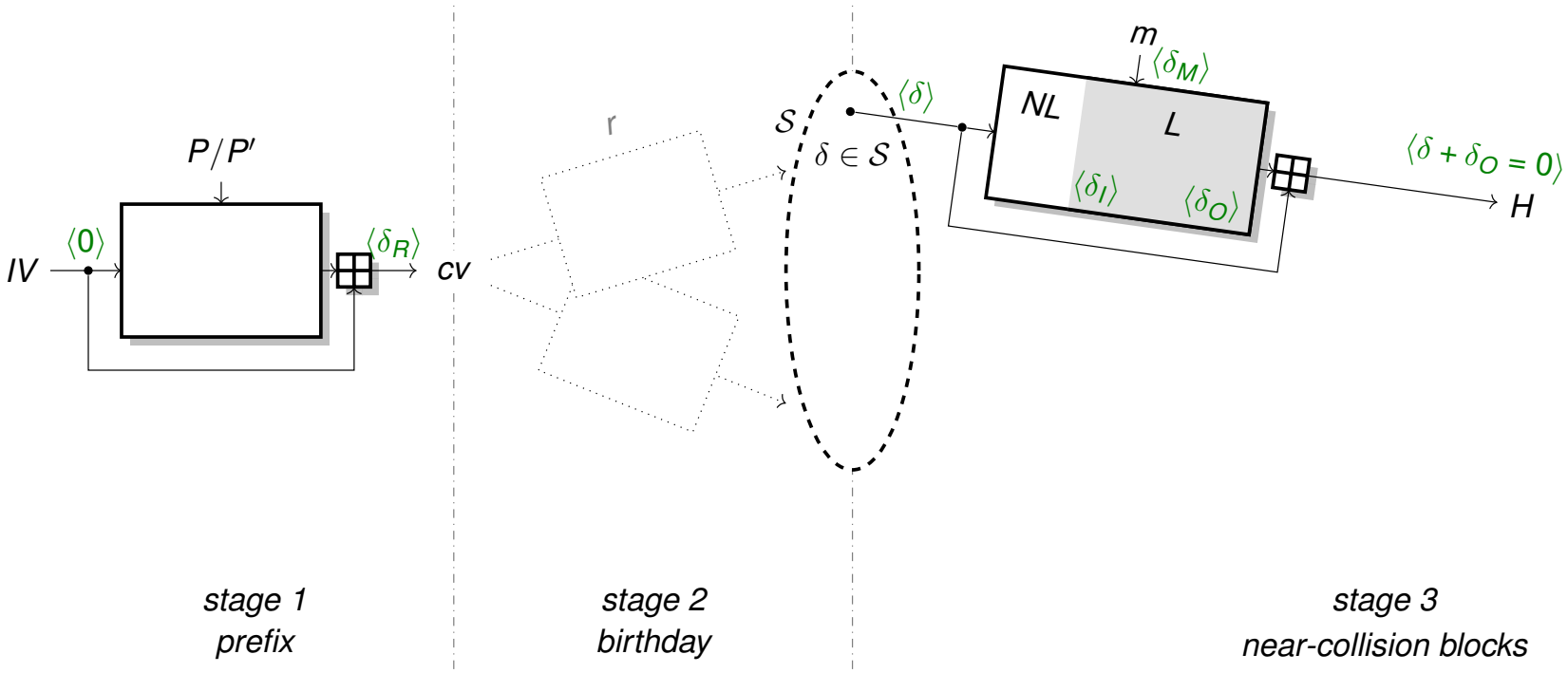
We will use the recent SHA-1 collision attack [SBKAM17] as a black box. Denote C ($\simeq 2^{64.7}$) the computational cost for the last block. (**New** : we recently reduced this cost to $2^{61.4}$)

Assume that we can use the same attack, for the same cost C, whatever is the input difference (this is possible thanks to the non-linear search algorithm). We validated this assumption in practice for several randomly chosen input differences.

Birthday to the rescue !

Trick 1 : birthday search

Use **birthday search** to reduce the entropy of possible chaining variable differences



Birthday to the rescue !

Trick 1 : birthday search

Use **birthday search** to reduce the entropy of possible chaining variable differences

Assume that you have a set $\mathcal{S}$ of differences that you can reach on the output of the internal cipher, for a computation cost C.

phase 1 : compute the (random) state difference δ_R induced by the challenge prefix

phase 2 : apply birthday strategy to map difference δ_R to a difference δ that belongs to a certain set $\mathcal{S}$ (requires $\sqrt{\pi \cdot 2^n/|\mathcal{S}|}$ computations)

phase 3 : apply the collision attack as explained previously (with a cost C) to map difference δ to a pair of colliding states

Birthday to the rescue !

Trick 1 : birthday search

Use **birthday search** to reduce the entropy of possible chaining variable differences

Stevens [S13] identified 192 possible output differences that can be reached for the same minimal cost C

phase 1 : $O(1)$

phase 2 : a birthday phase of $2^{77.1}$ computations

phase 3 : a collision phase of $C = 2^{64.7}$ computations

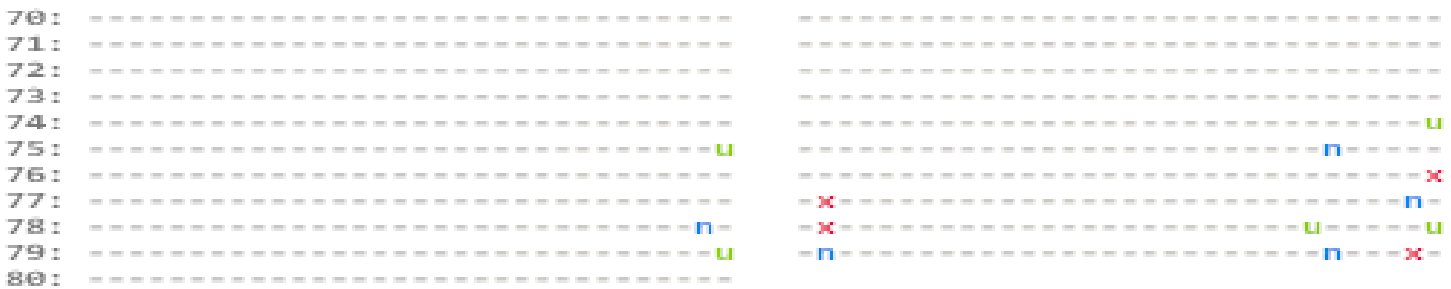
Total is $2^{77.1}$ computations (birthday phase is dominating)

Relaxing the output differences constrains

Trick 2a : generalized output differences

Using heuristics, we found **more allowable output differences** than previously known (576 instead of 192) for a cost C. This will increase $\mathcal{S}$.

For a maximal computational cost of C per block, we found a set $\mathcal{S}$ of 576 elements (instead of 192)

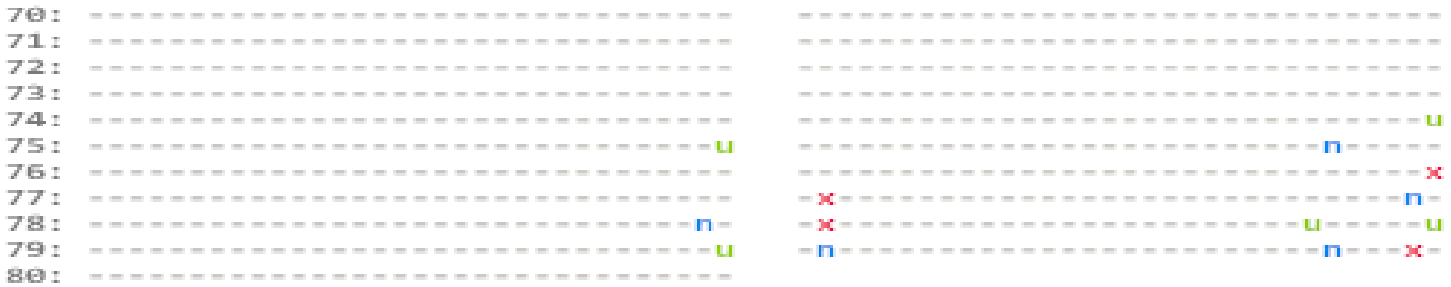


Relaxing the output differences constrains

Trick 2b : generalized output differences

Accept **more costly differential paths** to further increase $\mathcal{S}$ (so that the birthday phase and collision phase have about the same computational cost).

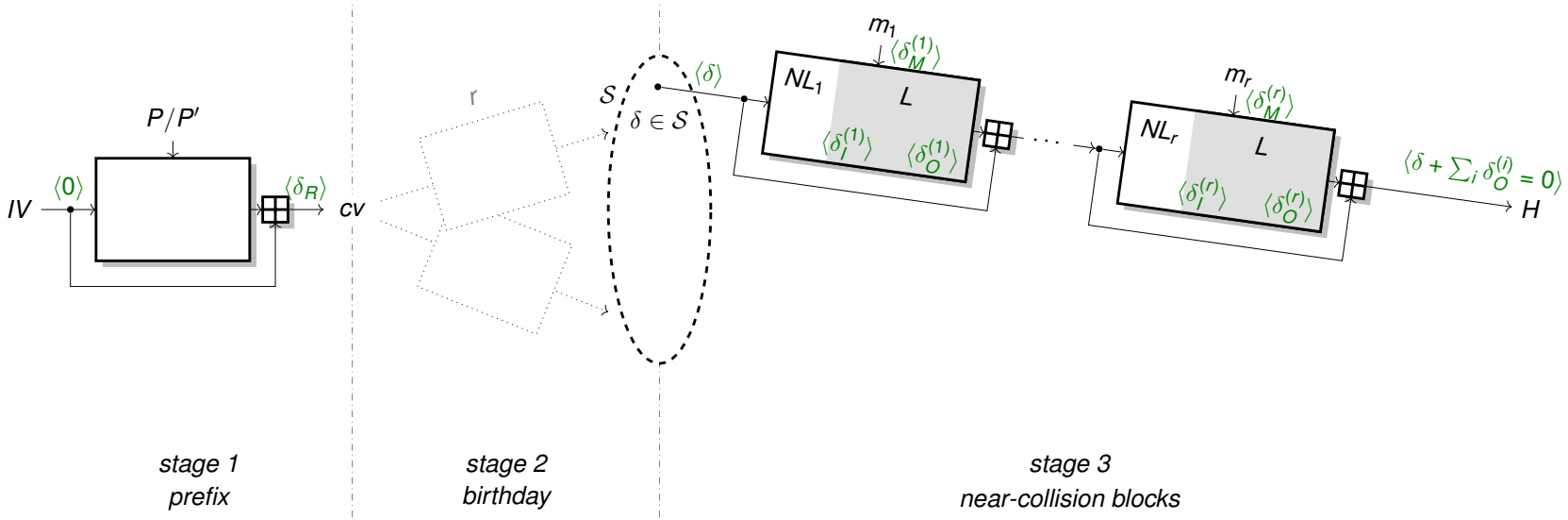
For a maximal computational cost of $8 \cdot C$ per block, we found a set $\mathcal{S}$ of 8768 elements.



Multiblocks : the more the merrier !

Trick 3 : multiblocks

Use **several blocks** to reach the collision after the birthday phase. It will increase the size of $\mathcal{S}$ and thus reduce the birthday phase cost.



Multiblocks : the more the merrier !

Trick 3 : multiblocks

Use **several blocks** to reach the collision after the birthday phase. It will increase the size of $\mathcal{S}$ and thus reduce the birthday phase cost.

With a maximum cost of $3 \cdot C$, one obtains a set $\mathcal{S}$ containing about 2^{30} elements !

The maximum length of the chain of blocks is 54 and the average is 17.

Multiblocks : the more the merrier !

Trick 3 : multiblocks

Use **several blocks** to reach the collision after the birthday phase. It will increase the size of $\mathcal{S}$ and thus reduce the birthday phase cost.

A three-phase attack :

phase 1 : compute the (random) state difference δ_R induced by the challenge prefix

phase 2 : apply birthday strategy to map difference δ_R to a difference δ that belongs to a certain set $\mathcal{S}$

phase 3 : apply a few consecutive SHA-1 block attacks to slowly map difference δ to a pair of colliding states

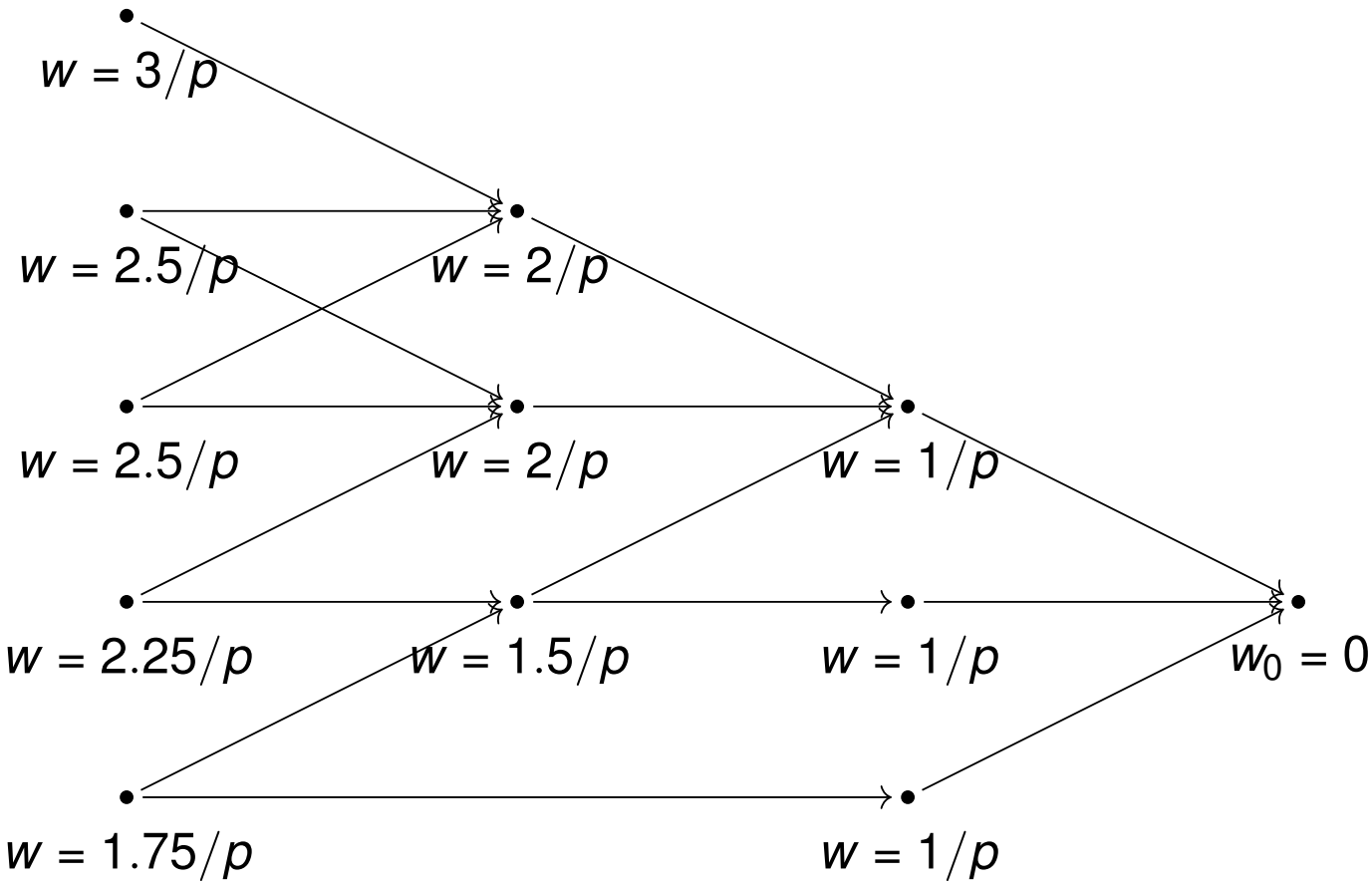
The clustering effect

Trick 4 : use the **clustering effect**

The attacker can sometimes **target several nodes simultaneously** to reduce the cost (because it is easier to hit one node out of many than a fixed one).

He will select **dynamically** the allowable differences at the output of each successive blocks. For that, we need a “map” of the whole situation, so we can decide which output difference I should be targeting at each new block. We will build a graph $\mathcal{G}$ for that.

Example of a graph $\mathcal{G}$



(we assume that all the edges have probability p)

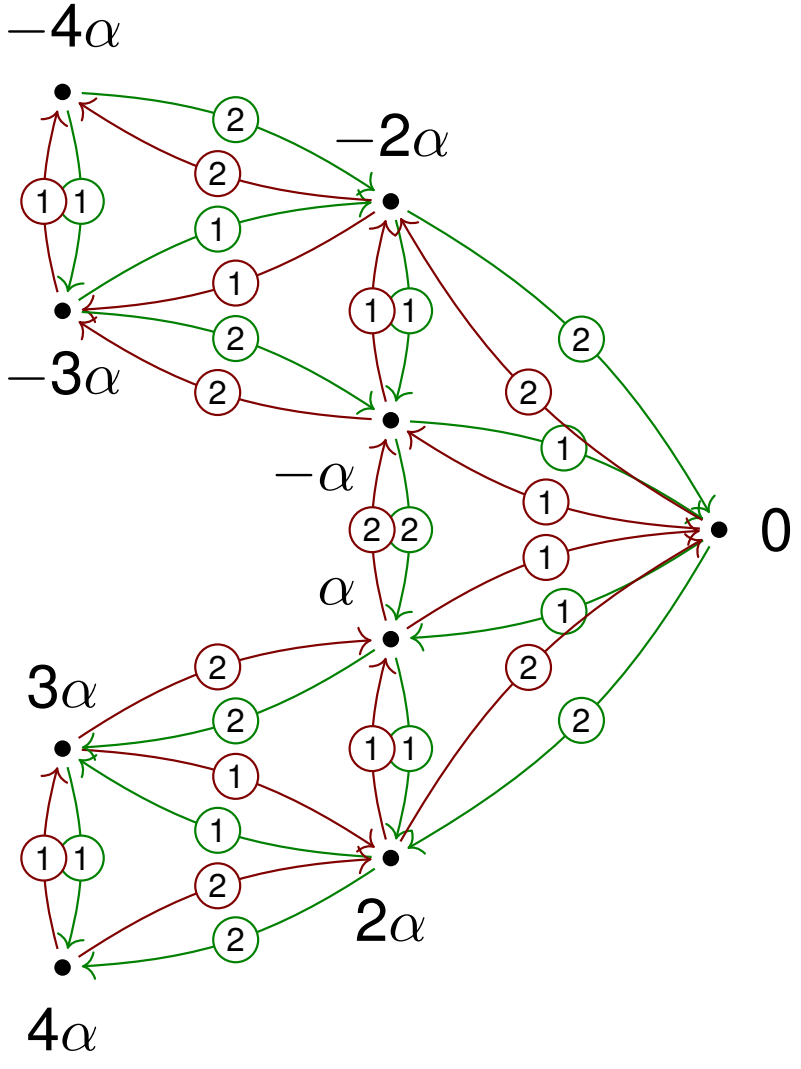
Building the graph $\mathcal{G}$ and the set $\mathcal{S}$

We call **bundle** a set of output differences that can be tried at the same time. We can build a graph $\mathcal{G}'$ with the bundles.

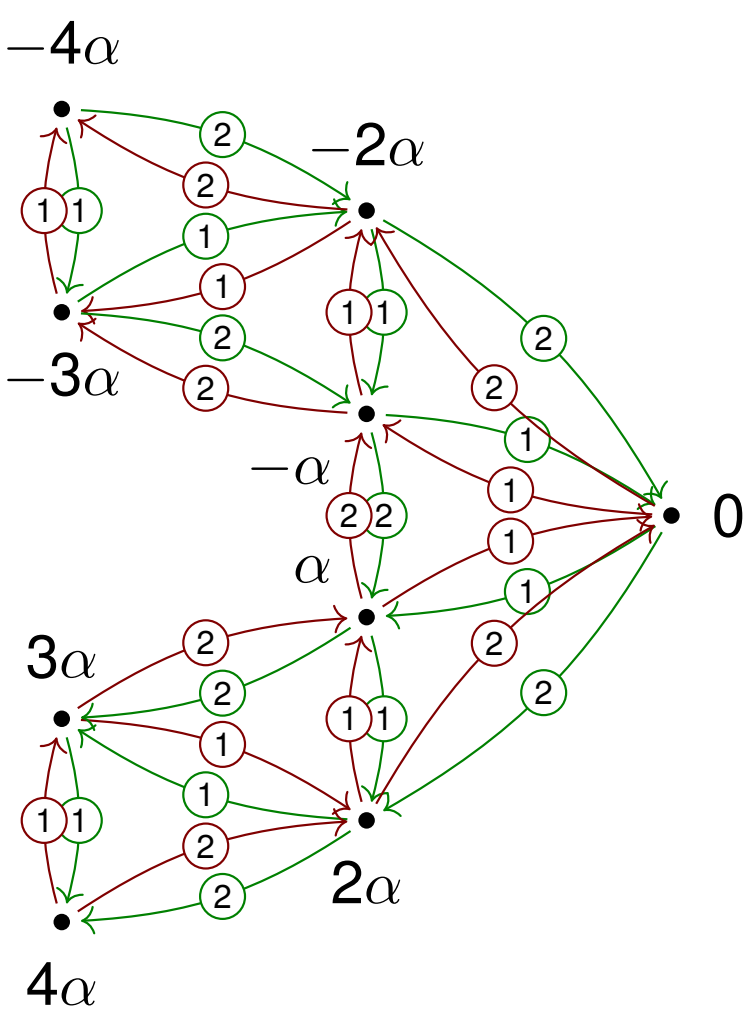
Consider that you have :

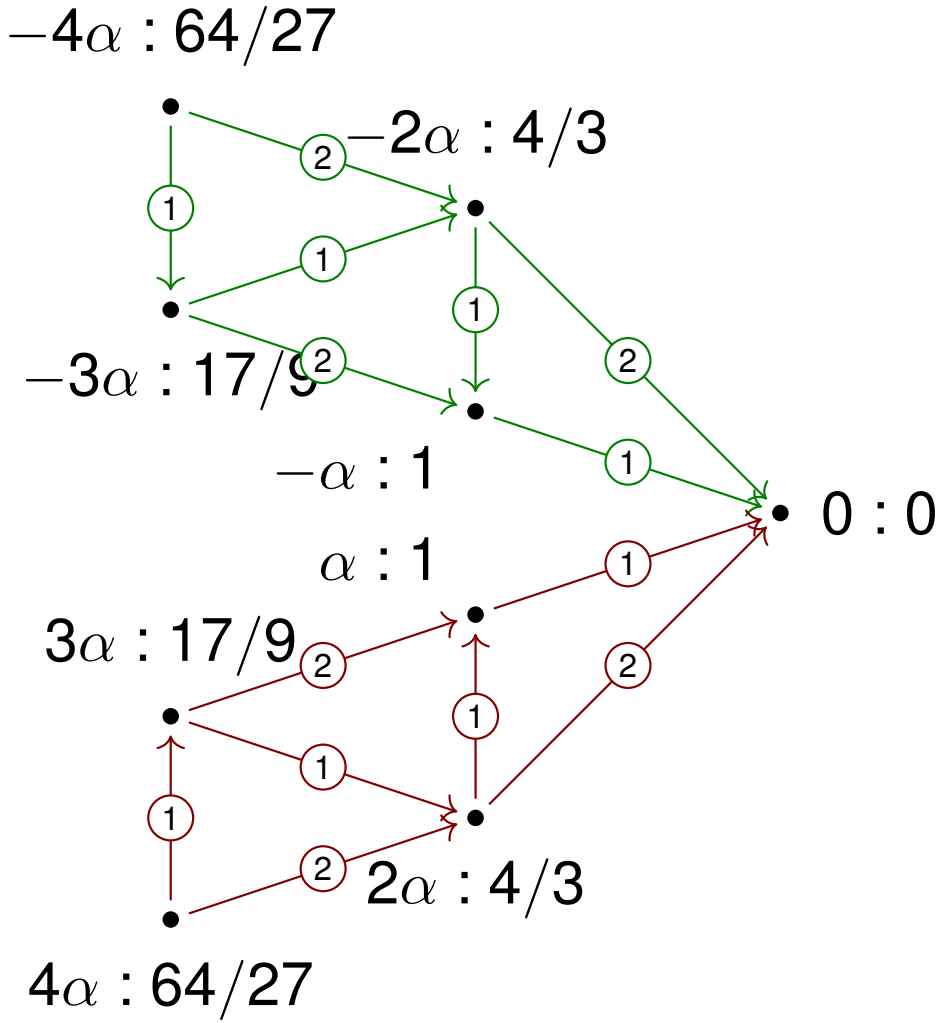
- a bundle $\{\alpha, 2\alpha\}$ with costs 1 and 2 (green lines)
- a bundle $\{-\alpha, -2\alpha\}$ with costs 1 and 2 (red lines)

The corresponding set $\mathcal{S}$ is $\{-4\alpha, -3\alpha, -2\alpha, -\alpha, 0, \alpha, 2\alpha, 3\alpha, 4\alpha\}$



Building the graph $\mathcal{G}$ and the set $\mathcal{S}$





Outline

Our results : SHA-1

We obtain a **practical chosen-prefix coll. attack on** SHA-1 :

- with only $2^{63.7}$ computations
- **only a small factor more costly than a classical collision attack** (that we improved from $2^{64.7}$ to $2^{61.4}$)
- We computed an actual chosen-prefix collision

Function	Collision type	Complexity	Reference
SHA-1	free-start collision	$2^{57.5}$	[SKP16]
	collision	2^{69}	[WYY05]
		$2^{64.7}$	[SBKAM17]
		$2^{61.4}$	our result NEW
	chosen-prefix collision	$2^{77.1}$	[S13]
		$2^{67.2}$	our result EC19
		$2^{63.7}$	our result NEW

GPU renting service

Bitcoin rise/fall created cheap GPU renting services :-)

We found very cheap GPU renting service, much cheaper than Amazon services : https ://www.gpuserversrental.com/
(we rented about 900 GTX-1060 for 2 months)

GPU renting service

Bitcoin rise/fall created cheap GPU renting services :-)

We found very cheap GPU renting service, much cheaper than Amazon services : https ://www.gpuserversrental.com/
(we rented about 900 GTX-1060 for 2 months)

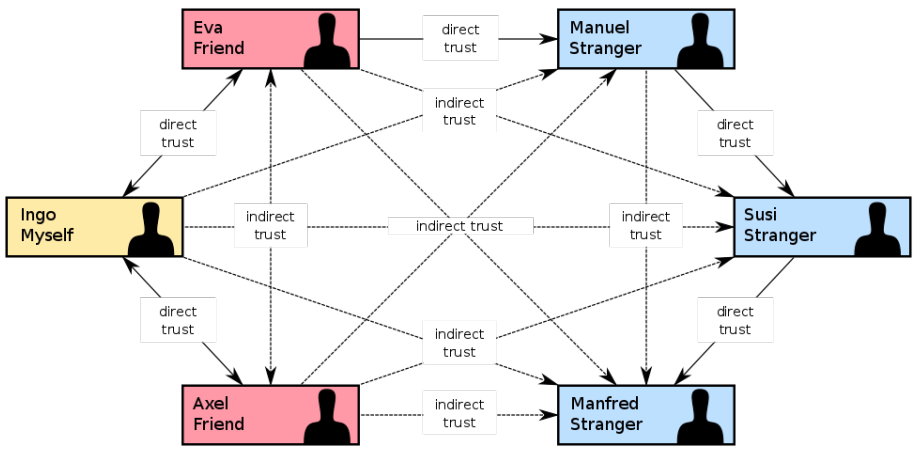
Finding the chosen-prefix collision costed us ”only” 75k USD (should be now 50k USD), classical collision costs now 10k.
In a few years, this will cost only a few thousands of USD.
Note : it was already practical since many years ago.

Breaking PGP Web-of-Trust (1)

Are we creating a simple chosen-prefix collision for SHA-1 ? No ! We have several possible proof-of-concept we could try (obvious one : creation of a rogue X.509 certificate).

CONFIDENTIAL :

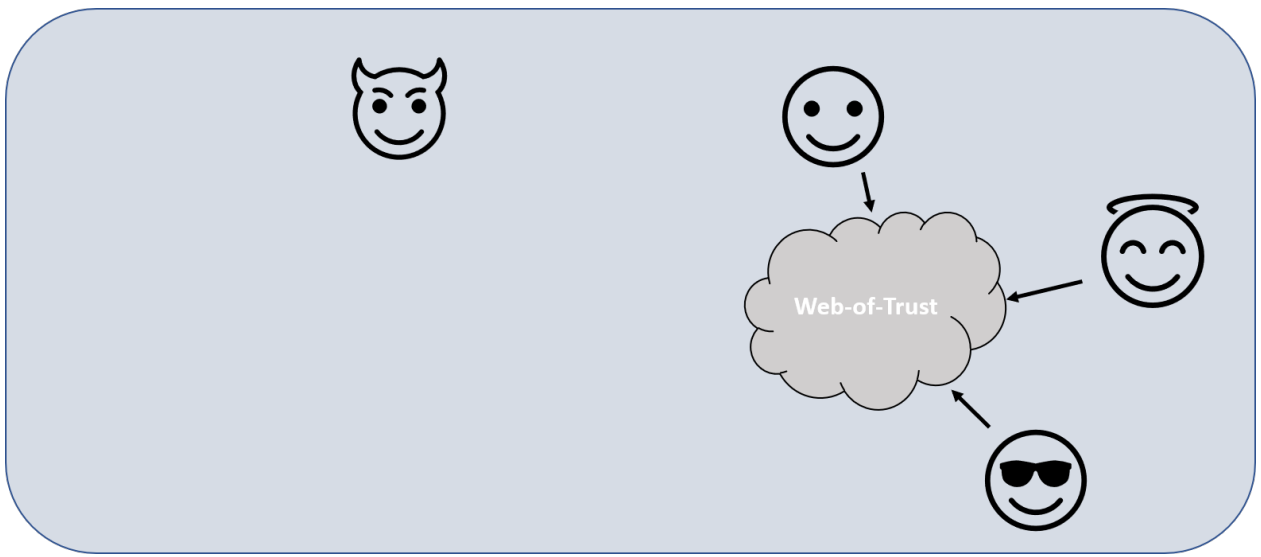
We chose to use the chosen-prefix collision to break OpenPGP key certification signatures, undermining the PGP Web-of-Trust.



Breaking PGP Web-of-Trust (2)

Idea :

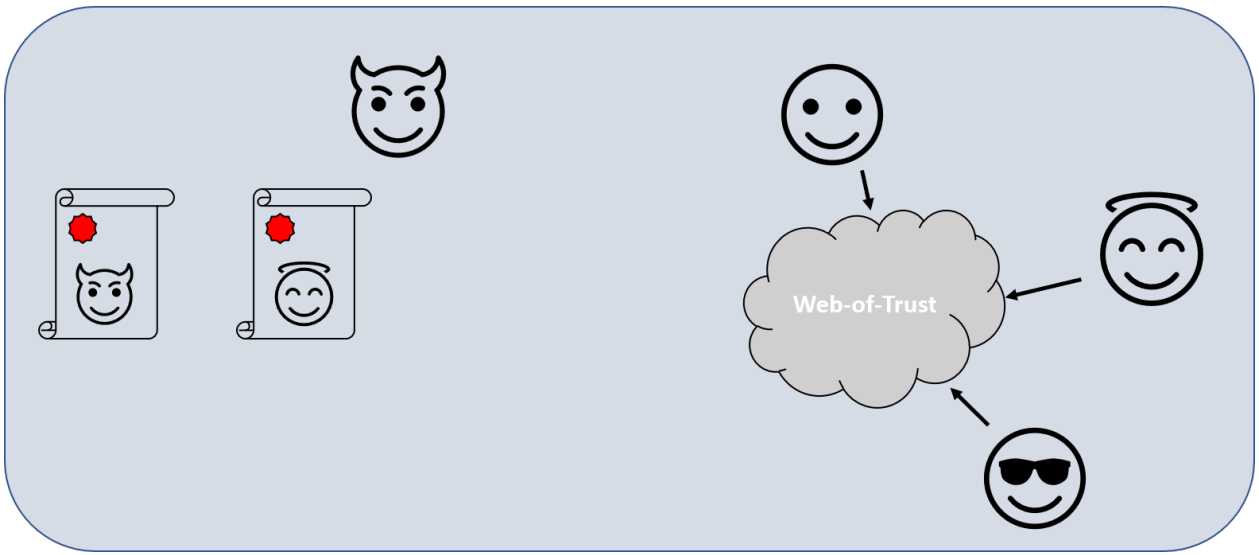
-
-
-
-



Breaking PGP Web-of-Trust (2)

Idea :

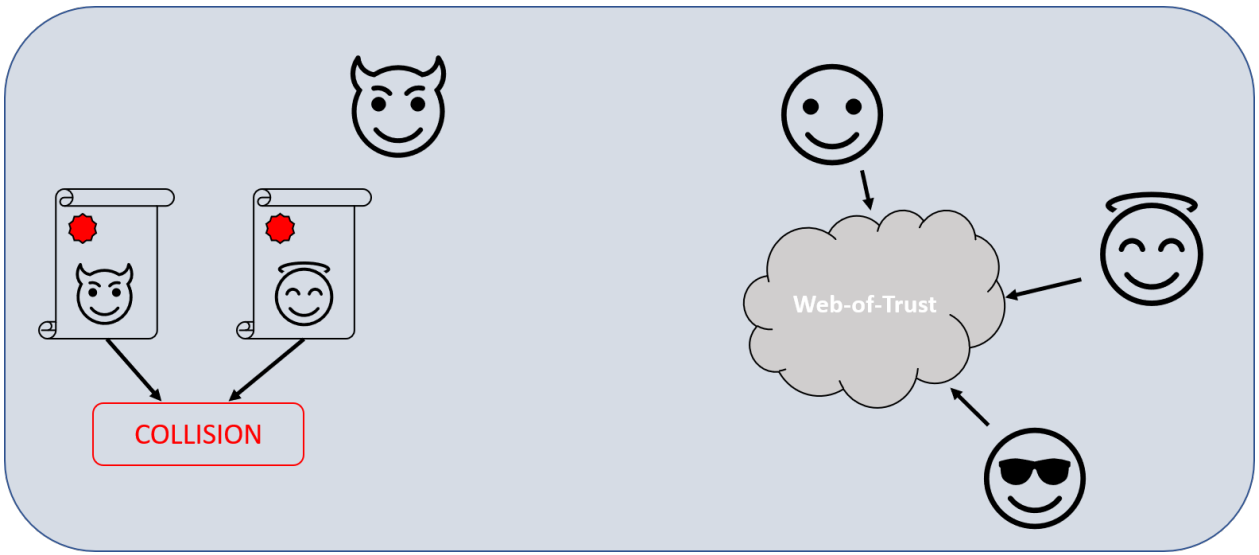
- create a pair of keys with two different UserIDs :
  one with victim name (A), and with real attacker's name (B)
-
-
-



Breaking PGP Web-of-Trust (2)

Idea :

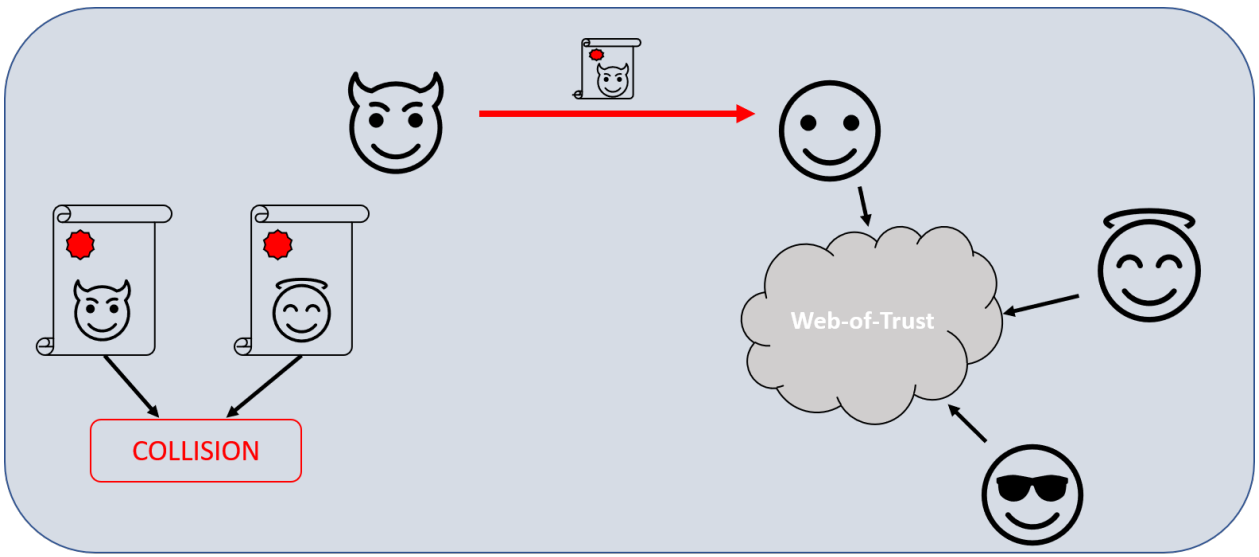
- create a pair of keys with two different UserIDs
- using a chosen-prefix collision, we craft the keys such that the SHA-1 hash that is signed for the key certification is the same for both keys.
-
-



Breaking PGP Web-of-Trust (2)

Idea :

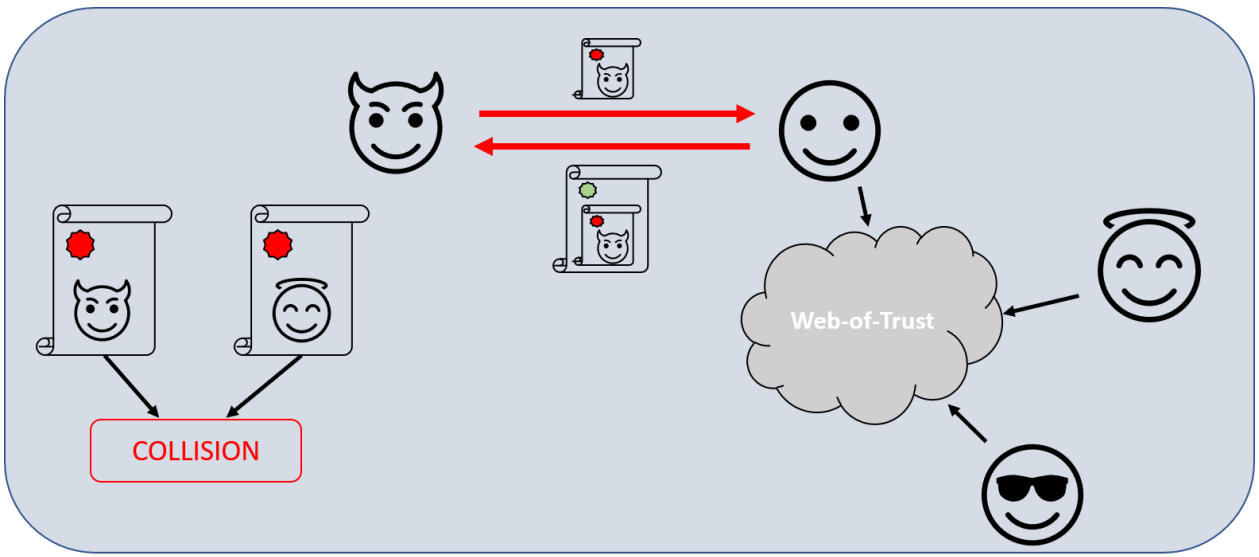
- create a pair of keys with two different UserIDs
- collide key certifications
- the attacker asks for key certifications of key B : since he knows the corresponding secret key, and the UserID matches his official ID, he will collect trust-worthy signatures and integrate the web-of-trust.
-



Breaking PGP Web-of-Trust (2)

Idea :

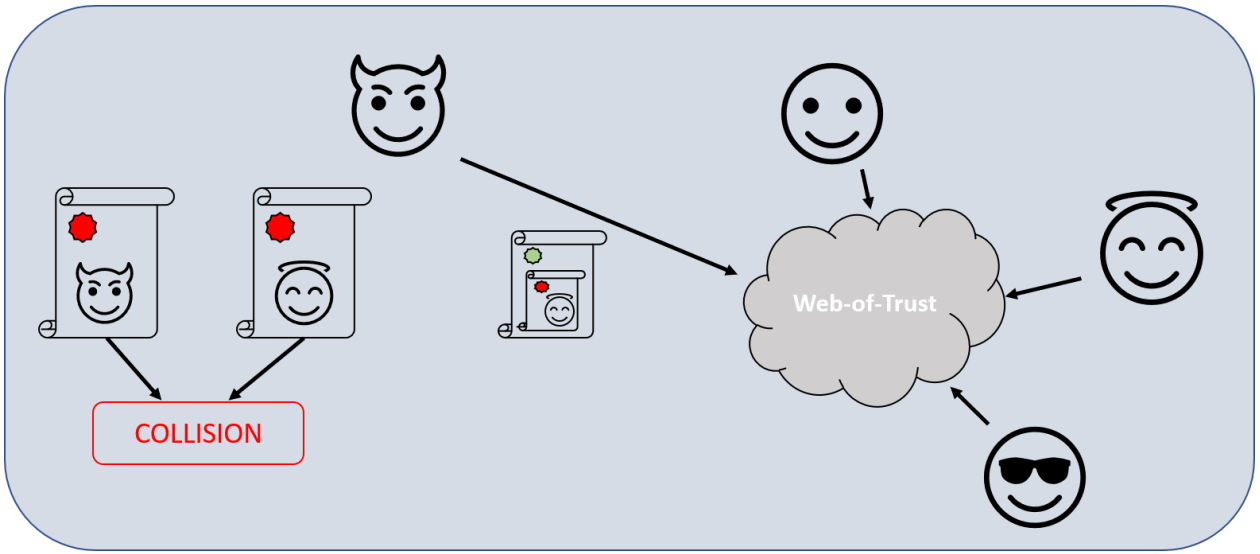
- create a pair of keys with two different UserIDs
- collide key certifications
- the attacker asks for key certifications of key B : since he knows the corresponding secret key, and the UserID matches his official ID, he will collect trust-worthy signatures and integrate the web-of-trust.
-



Breaking PGP Web-of-Trust (2)

Idea :

- create a pair of keys with two different UserIDs
- collide key certifications
- integrate web of trust with UserID B
- since the hash of both keys collide, he can transplant the signatures to key A, creating a key with the UserID of the victim, trusted by the web-of-trust, and for which he controls the secret key. He can then sign messages pretending to be the victim.



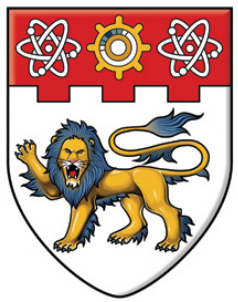
Breaking PGP Web-of-Trust (2)

Idea :

- create a pair of keys with two different UserIDs
- collide key certifications
- integrate web of trust with UserID B
- since the hash of both keys collide, he can transplant the signatures to key A, creating a key with the UserID of the victim, trusted by the web-of-trust, and for which he controls the secret key. He can then sign messages pretending to be the victim.

This result will be announced at RWC 2020

Thank you for your attention !



NANYANG TECHNOLOGICAL UNIVERSITY