Exploring Integrity of AEADs with Faults: Definitions and Constructions

Sayandeep Saha[†,⋆], Mustafa Khairallah[†,‡] and Thomas Peyrin[†]

March 21, 2023

[†] Nanyang Technological University, Singapore
[⋆] ICTEAM/ELEN,Université catholique de Louvain (UClouvain), Belgium
[‡] Seagate Research Group, Singapore

Outline

1. Fault Attacks (FA)
2. Attacks on Classical AEAD Schemes without Key Recovery
3. Levelled Implementations
4. Fault Resilient PRF
5. Fault Resilient MAC
6. Fault Resilient AEAD

Fault Attacks (FA)

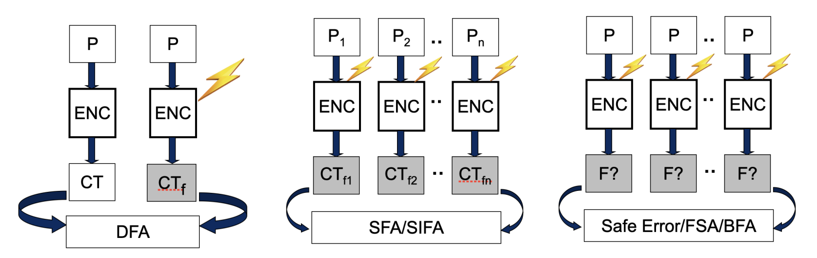
Fault Attacks (FA)

- Fault Attacks (FA) have been introduced in 1997 [BDL97, BS97].
- Over the years, both analysis and fault injection techniques have improved significantly [TMA11, FJLT13, SBHS15, SH07, SBR+20, DEK+18, PCNM15, ZLZ+18, MOG+20, DEK+18, DEG+18, SBR+20, SBJ+21].
- Most fault attacks and fault countermeasures in symmetric key cryptography target key/state recovery.

Fault-Resiliency

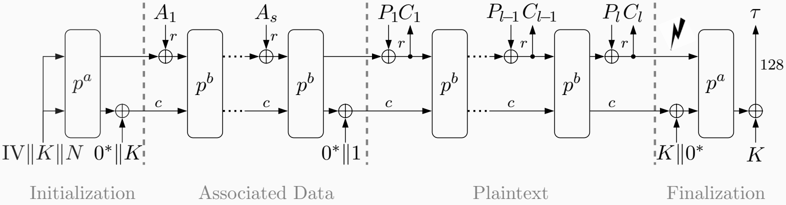
- Dobraunig, Mennink and Primas. [DMP20] discussed the security of sponge-like constructions where the amount of information leaked using faults is limited.
- Some papers discuss building primitives that protect against certain types of fault attacks [MSGR10, SBD+20, BBB+21].
- Fischlin and Günther [FG20] discussed the concept of fault-resilient AE and gave one construction.
- Saha, Khairallah and Peyrin (this work) discussed the definitions of the fault model and how to define different fault-resilient primitives to be able to use in AE scheme. We also show that the construction from [FG20] does not achieve frAE.
- In parallel to this work, Berti, Guo, Peters, Shen and Standaert [BGP+22] showed that it is possible to have frMAC with resiliency against verification faults. The final construction in their paper can be seen as an instantiation of our frMAC.

Examples of Fault Attacks



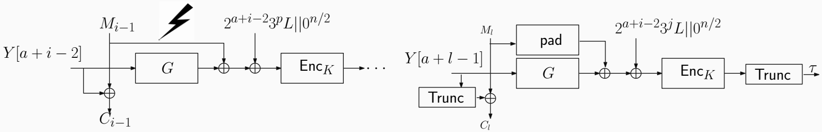
Attacks on Classical AEAD Schemes without Key Recovery

Ascon



A1
As
P1 C1
Pl−1 Cl−1
Pl Cl
τ
r
c
128
pa
pb
IV‖K‖N
0*‖K
0*‖1
K‖0*
K
Initialization
Associated Data
Plaintext
Finalization

COFB



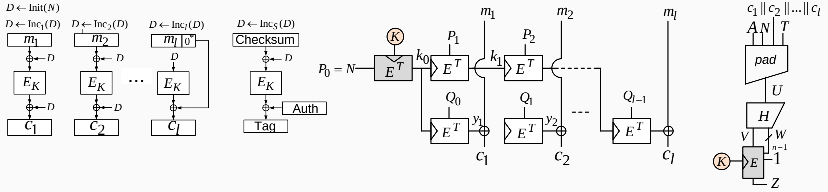
M_{i-1}
$2^{a+i-2}3^pL||0^{n/2}$
$Y[a+i-2]$
G
Enc_K
C_{i-1}
M_l
pad
$2^{a+i-2}3^jL||0^{n/2}$
$Y[a+l-1]$
G
Enc_K
Trunc
Trunc
τ
C_l

Applies to more schemes

- SIV.
- Enc-then-MAC.

Levelled Implementations

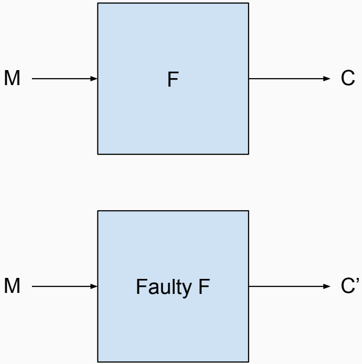
OCB vs. TEDT: Protecting long term secrets at a cheaper cost



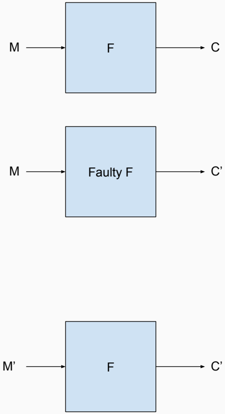
(a) (b)

Fault Resilient PRF

What happens when we inject fault?



WHAT HAPPENS WHEN WE INJECTS FAULT?



Resistance vs. Resilience

- We allow more trivial forgeries/distinguishers than allowed in a classical security notion.
- We allow a phase of the attack where we do not claim security for any message in that phase.

ADVERSARIAL SET UP

1. Training phase: the attacker gets description of the implementation with the ability to inject faults anywhere, but no direct access to the secret key. In this phase the implementation is always real.

Adversarial Set Up

1. Training phase: the attacker gets description of the implementation with the ability to inject faults anywhere, but no direct access to the secret key. In this phase the implementation is always real.
2. Attack phase: the attacker cannot inject faults any more. In this phase the oracle can be real or ideal.

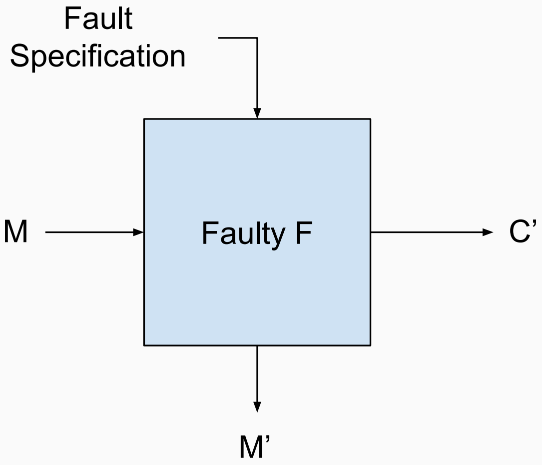
Fault Resilient PRF Game

<table>
<tr><td colspan="2">Challenge: Faulty queries may (in theory) leak information about more than one evaluation of the function at a time.</td></tr>
<tr><td colspan="2">frPRF</td></tr>
<tr><td>Real World</td><td>Ideal World</td></tr>
<tr><td>$\text{PRF}^f_K(M, \mathcal{F})$
. . .
faulty implementation.
. . .

$\text{PRF}_K(M)$
. . .
Real implementation with fresh inputs.
. . .</td><td>$\text{PRF}^f_K(M, \mathcal{F})$
. . .
faulty implementation, but terminates if a faulty query leaks more than one point of the function.
. . .

$\text{RF}_K(M)$
. . .
Random function with fresh inputs.
. . .</td></tr>
</table>

Faulty PRF Oracle

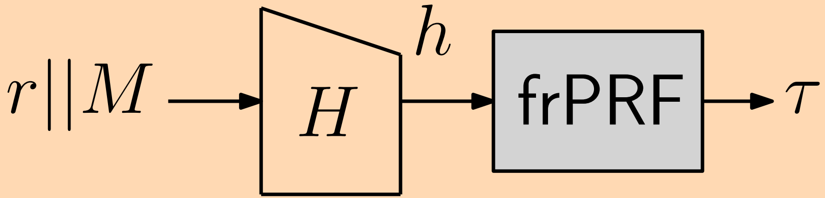


How to build an frPRF

1. We can construct such primitives using a tweakable block cipher protected against fault-attacks.
2. If the cipher does not allow key recovery through fault attacks, it should be possible to use as an frPRF.
3. It may be possible to show that ISAP finalization is an frPRF.
4. In practice, we may not know if the preimage is easy or not, but what this model says is that a small amount of trivial forgeries/distinguishers using faults is unavoidable.

Fault Resilient MAC

frMAC



Security Arguments

- No collision on the random salt, or the output of the hash (frRO).
- Only trivial preimages are prossible.
- frMAC has security similar to frPRF, only need to worry about tag verification.

Fault Resilient AEAD

The frAE Game

- Similar to the frPRF game but taking privacy and decryption into account.
- A variation of the game proposed in [FG20].

SIV$ [FG20]

SIV$

KGen:

1: $(K_1, K_2) \xleftarrow{\$} \mathcal{K}$
2: **return** (K_1, K_2)

$\mathsf{Enc}_K^{\mathsf{SIV\$}}(N, A, M; r)$:

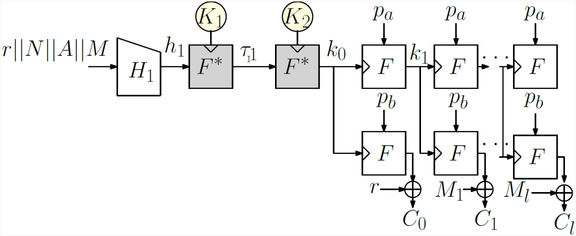
1: $(K_1, K_2) \leftarrow K$
2: $IV \leftarrow \mathsf{PRF}(K_1, N, A, M; r)$
3: $C \leftarrow \mathsf{Enc}(K_2, r||M, IV)$
4: **return** (IV, C)

$\mathsf{Dec}_K^{\mathsf{SIV\$}}(N, A, C')$:

1: $(K_1, K_2) \leftarrow K$
2: $(IV, C) \leftarrow C'$
3: $r||M \leftarrow \mathsf{Dec}(K_2, C, IV)$
4: $IV' \leftarrow \mathsf{PRF}(K_1, N, A, M, r)$
5: **if** $IV = IV'$
6: **return** M
7: **else**:
8: **return** $\perp$

DECOUPLING ATTACK ON ALL SIV-LIKE SCHEMES

1. Fault the MAC to make it give a tag for M'.
2. Encrypt M using the IV corresponding to M'.
3. (N, A, C, IV) is not a valid ciphertext.
4. C can be changed to C' corresponding to M'.

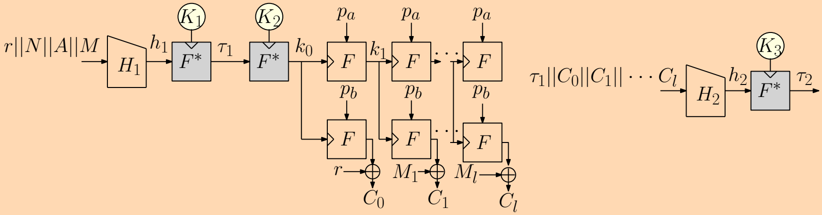


If MAC-then-Enc does not work, what does?

MAC-then-Enc

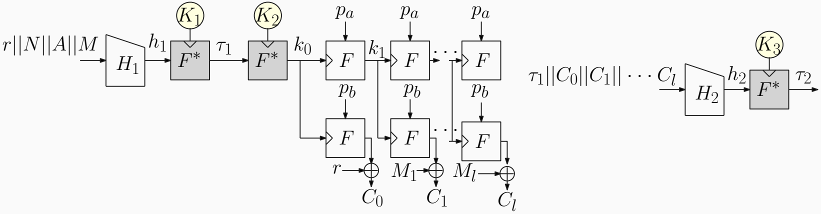
then MAC Again

MAC-ENC-MAC (MEM)



Security Arguments

- No collision on the random salt, or the output of the first MAC.
- The security then reduces to the frMAC security of the two MACs and the frPRF security of the key derivation function in the encryption layer.



Conclusion

- It is possible to protect certain classes of fault attacks using levelled implementations.
- Randomness is critical to prevent differential fault attacks in unprotected primitives.
- It is possible to prevent single differential fault attacks with less cost and more effectively than dummy duplication.

Future Work

- Indifferentiability of randomized hash functions from frRO.
- Show frAE is secure against combined attacks (combined fault and leakage resilience).
- Protecting against multiple faults.
  - A solution to prevent a d-fault version of the decoupling attack may be to keep interleaving Enc and MAC (MEMEM...).
  - Is there a solution a solution that protects against arbitrary number of faults?
- Are there efficient solutions for the security of MAC against differential faults without randomness?
  - In parallel work, Berti *et al.* [BGP+22] showed an example of a MAC that does not need randomness and protects against a single differential fault. It requires two MAC invocations.
  - A more efficient solution would need less than $i + 1$ invocations to protect against i differential faults.
- Relate the security of different fault countermeasures to the frPRF assumption (e.g. is ISAP's PRF an frPRF?).

Thank you!

More details in eprint 2022/1055

Anubhab Baksi, Shivam Bhasin, Jakub Breier, Mustafa Khairallah, Thomas Peyrin, Sumanta Sarkar, and Siang Meng Sim.
Default: Cipher level resistance against differential fault attack.
In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 124–156. Springer, 2021.

Dan Boneh, Richard A DeMillo, and Richard J Lipton.
On the importance of checking cryptographic protocols for faults.
In *Proceedings of 15th International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pages 37–51, Konstanz, Germany, May 1997. Springer.

References II

Francesco Berti, Chun Guo, Thomas Peters, Yaobin Shen, and François-Xavier Standaert.
Secure message authentication in the presence of leakage and faults.
Cryptology ePrint Archive, Paper 2022/1142, 2022.
`https://eprint.iacr.org/2022/1142`.

Eli Biham and Adi Shamir.
Differential fault analysis of secret key cryptosystems.
In *Proceedings of 17th Annual International Cryptology Conference (CRYPTO)*, pages 513–525, Santa Barbara, USA, Aug 1997. Springer.

- Christoph Dobraunig, Maria Eichlseder, Hannes Gross, Stefan Mangard, Florian Mendel, and Robert Primas.
  Statistical ineffective fault attacks on masked AES with fault countermeasures.
  In *Proceedings of 24th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*, pages 315–342, Brisbane, QLD, Australia, Dec 2018. Springer.

- Christoph Dobraunig, Maria Eichlseder, Thomas Korak, Stefan Mangard, Florian Mendel, and Robert Primas.
  SIFA: exploiting ineffective fault inductions on symmetric cryptography.
  IACR Transactions on Cryptographic Hardware and Embedded Systems, 2018(3):547–572, 2018.

References IV

Christoph Dobraunig, Bart Mennink, and Robert Primas.
Leakage and tamper resilient permutation-based cryptography.
Cryptology ePrint Archive, 2020.

Marc Fischlin and Felix Günther.
Modeling memory faults in signature and authenticated encryption schemes.
In *Topics in Cryptology – CT-RSA 2020: The Cryptographers' Track at the RSA Conference 2020*, page 56–84, San Francisco, CA, USA, 2020. Springer.

References V

Thomas Fuhr, Eliane Jaulmes, Victor Lomné, and Adrian Thillard.

Fault attacks on AES with faulty ciphertexts only.
In *Proceedings of 10th IEEE Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, pages 108–118, Santa Barbara, USA, Aug 2013. IEEE.

Kit Murdock, David Oswald, Flavio D Garcia, Jo Van Bulck, Daniel Gruss, and Frank Piessens.
Plundervolt: Software-based fault injection attacks against Intel SGX.
In *Proceedings of 41st IEEE Symposium on Security and Privacy (S&P)*, pages 1466–1482, San Francisco,, USA, May 2020. IEEE.

References VI

Marcel Medwed, François-Xavier Standaert, Johann Großschädl, and Francesco Regazzoni.
Fresh re-keying: Security against side-channel and fault attacks for low-cost devices.
In *International Conference on Cryptology in Africa*, pages 279–296. Springer, 2010.

Sikhar Patranabis, Abhishek Chakraborty, Phuong Ha Nguyen, and Debdeep Mukhopadhyay.
A biased fault attack on the time redundancy countermeasure for AES.
In *Proceedings of 6th International Workshop on Constructive Side-Channel Analysis and Secure Design (COSADE)*, pages 189–203, Berlin, Germany, Apr 2015. Springer.

Bertram Poettering and Paul Rösler.
Combiners for aead.
Cryptology ePrint Archive, 2020.

Thierry Simon, Lejla Batina, Joan Daemen, Vincent Grosso, Pedro Maat Costa Massolino, Kostas Papagiannopoulos, Francesco Regazzoni, and Niels Samwel.
Friet: an authenticated encryption scheme with built-in fault detection.
In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 581–611. Springer, 2020.

References VIII

Bodo Selmke, Stefan Brummer, Johann Heyszl, and Georg Sigl.
Precise laser fault injections into 90nm and 45nm SRAM-cells.
In *Proceedings of 14th International Conference on Smart Card Research and Advanced Applications (CARDIS)*, pages 193–205, Bochum, Germany, Nov 2015. Springer.

Sayandeep Saha, Arnab Bag, Dirmanto Jap, Debdeep Mukhopadhyay, and Shivam Bhasin.
Divided we stand, united we fall: Security analysis of some SCA+SIFA countermeasures against SCA-enhanced fault template attacks.
In *ASIACRYPT 2021*, 2021.

Sayandeep Saha, Arnab Bag, Debapriya Basu Roy, Sikhar Patranabis, and Debdeep Mukhopadhyay.
Fault template attacks on block ciphers exploiting fault propagation.
In *Proceedings of 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pages 612–643, Zagreb, Croatia, May 2020. Springer.

J. M. Schmidt and Michael Hutter.
Optical and EM fault-attacks on CRT-based RSA: Concrete results.
In *Proceedings of 15th Austrian Workhop on Microelectronics (Austrochip)*, pages 61–67, Graz, Austria,, Oct 2007. Verlag der Technischen Universität Graz.

References x

Michael Tunstall, Debdeep Mukhopadhyay, and Subidh Ali.
Differential fault analysis of the advanced encryption standard using a single fault.
In *Proceedings of 5th Information Security Theory and Practice. Security and Privacy of Mobile Devices in Wireless Communication (WISTP)*, pages 224–233, Crete, Greece, June 2011. Springer.

Fan Zhang, Xiaoxuan Lou, Xinjie Zhao, Shivam Bhasin, Wei He, Ruyi Ding, Samiya Qureshi, and Kui Ren.
Persistent fault analysis on block ciphers.
IACR Transactions on Cryptographic Hardware and Embedded Systems, 2018(3):150–172, 2018.

Modelling Faults

Physical to Logical

- A classification is important.
- Physical faults eventually cause some data or control corruption.
- In general, localized corruptions are observed which can be one/multiple-bit flip or set/reset. For software implementations, we also observe instruction modification/skip.
- The precision of faults are dependent on the target device and injection instruments.

Abstract Model of Physical Events

Fault Representation	
Params	Description
v_i	Denote the variables corrupted by faults. $v_i \in$ data $\cup$ control $\cup$ constant.
nf	The number of faults injected throughout the computation (in the same or different clock cycles).
w_i	Denote the width (how many bits within a target variable are corrupted) of a fault ($0 \leq w_i \leq \|v_i\|$).
mod_i	The logical abstraction of physical nature of faults (fault models). $mod_i \in$ fix $\cup$ diff $\cup$ rand $\cup$ nof.
t_i	Denote if the fault is transient/persistent and the temporal fault location.

Variable Classification	
Params	Description
data	Denotes the set of data-flow variables (input, output and intermediate states of the computation).
control	Denotes the set of control-flow variables (branch statements).
constant	Denotes the set of constants, tables, and domain separators of the AEAD algorithm.

Fault Models	
fix	Denote faults where the adversary is allowed to fix w_i bits of the target variable to some desired value.
diff	Denote the differential faults where the adversary is allowed to select a bitwise differential Δ_i for variable v_i (with $HW(\Delta_i) = w_i$) and set $v_i' = v_i \oplus \Delta_i$. Here v_i' is the faulty version of v_i and $HW(\cdot)$ denote the Hamming weight.
rand	Same as diff except the fact that $\Delta_i \xleftarrow{\$} \{0,1\}^{\|v_i\|}$ and $HW(\Delta_i) = w_i$.
nof	Denotes the case when the adversary chooses not to inject a fault in the execution.

COMPARISON TO AN AEAD COMBINER [PR20]

- A different approach to prevent such failures would have been to use an AEAD combiner with two AEAD schemes.
- Assuming both schemes are based on SIV$, this solution needs two MACs and two Encryption schemes.
- In fact, it is impossible to get an AEAD combiner with less than $4A + 4M$ for the encryption and decryption cost ([PR20]).
- The cheapest blackbox combiner for two SIV$ schemes would need $2A + 6M$ for either encryption of decryption[1].
- Our solution only needs $A + 3M$.
- For ciphertext lengths, our scheme is on-par with AEAD combiners $M + 3\tau$.

[1] [PR20] reports $2A + 3M$, but this is not considering that the AEAD scheme itself processes the message twice

THE FRAE GAME: TRAINING[2]

if $d = 1$
  return $\perp$
$r \xleftarrow{\$} \mathcal{R}$
$(\mathcal{N}_{flt}, \mathcal{AD}_{flt}, C) \leftarrow \text{Fault}(\text{MEMEnc}_K(N, A, M; r), \mathcal{F})$
$\mathcal{I}_{vld} \leftarrow \{(N', A') \in \mathcal{N}_{flt} \times \mathcal{AD}_{flt} \mid$
  $\text{MEMDec}_K(N', A', C) \neq \perp \wedge (N', A', C) \notin \mathcal{S}_{flt}\}$
if $b = 1 \wedge |\mathcal{I}_{vld}| > 1$
  return $\perp$
$\mathcal{S}_{flt} \leftarrow \mathcal{S}_{flt} \cup \{(N', A', C) \mid$
  $(N', A') \in \mathcal{I}_{vld}\}$
return C

[2] A variation of the game proposed in [FG20]

The frAE Game: Attack[3]

$\mathsf{Enc}_K(N, A, M)$

1: $r \xleftarrow{\$} \mathcal{R}$
2: $C \leftarrow \mathsf{MEMEnc}_K(N, A, M; r)$
3: $\mathcal{S} \leftarrow \mathcal{S} \cup \{(N, A, C)\}$
4: **if** $b = 1$
5: $\quad C \xleftarrow{\$} \{0, 1\}^{|M|+|\tau_1|+|\tau_2|+|r|}$
6: $d \leftarrow 1$
7: **return** C

$\mathsf{Dec}_K(N, A, C)$

1: **if** $b = 1 \vee (N, A, C) \in \mathcal{S}_{flt} \cup \mathcal{S}$
2: $\quad$ **return** $\perp$
3: **else**
4: $\quad$ **return** $\mathsf{MEMDec}_K(N, A, C)$

[3] A variation of the game proposed in [FG20]

Fault Resilient Random Oracle

Random Oracles

- A random oracle in this work refers to an arbitrary input length and fixed output length random function.
- Unlike a PRF, a random oracle has no meaning implementation that can be faulted.
- We could view the random oracle as a large table that can be faulted, but that is not very useful.

Key Observations

1. A hash function that is collision-resistant remains collision-resistant with faults. (Maybe not so obvious)
2. Preimage resistance is less clear: the adversary can force a faulty hash value that corresponds to a given input message.
3. Random salts prevent this type of attacks.
4. Randomness needs to be synchronized during verification.
5. Can we do something even stronger?

Random Oracles vs. Hash Function

- Usually we are using the random oracle model to argue about the security of a hash-based scheme.
- It is more meaningful to argue about the security of the random oracle in the relation to the implementation of the actual hash function.

frRO: Idea

- Use the hash function implementation to find out the effect of the fault.
- Use the random oracle to generate the tag.
- We need random salt to prevent certain prefix attacks.

frRO

The frRO Oracle

INIT

for $y \in \{0,1\}^*$

  $\mathsf{RO}(y) \xleftarrow{\$} \bot$

$\mathcal{R}_{\mathit{flt}} \leftarrow \emptyset$

$\mathsf{frRO}^f(x; r, \mathcal{F})$

if $r \in \mathcal{R}_{\mathit{flt}}$ **then** `bad`

$\mathcal{R}_{\mathit{flt}} \leftarrow \mathcal{R}_{\mathit{flt}} \cup \{r\}$

if $\mathcal{F}.\mathit{mod} = \mathsf{nof}$

  if $\mathsf{RO}(r\|x) = \bot$

    $\mathsf{RO}(r\|x) \xleftarrow{\$} \{0,1\}^{|h|}$

  $Z \leftarrow \mathsf{RO}(r\|x)$

else if $\mathcal{F}.v = r$

  $r \leftarrow r \oplus \Delta$

  if $r \in \mathcal{R}_{\mathit{flt}}$ **then** `bad`

  $\mathcal{R}_{\mathit{flt}} \leftarrow \mathcal{R}_{\mathit{flt}} \cup \{r\}$

  if $\mathsf{RO}(r\|x) = \bot$

    $\mathsf{RO}(r\|x) \xleftarrow{\$} \{0,1\}^{|h|}$

  $Z \leftarrow \mathsf{RO}(r\|x)$

else if $\mathcal{F}.v = x$

  $x \leftarrow x \oplus \Delta$

  if $\mathsf{RO}(r\|x) = \bot$

    $\mathsf{RO}(r\|x) \xleftarrow{\$} \{0,1\}^{|h|}$

  $Z \leftarrow \mathsf{RO}(r\|x)$

else

  if $\mathsf{RO}(r\|x) = \bot$

    $\mathsf{RO}(r\|x) \xleftarrow{\$} \{0,1\}^{|h|}$

  $(M^f, h) \leftarrow \mathsf{Fault}(H(r\|x), \mathcal{F})$

  $\Delta \leftarrow H(r\|x) \oplus h$

  $Z \leftarrow \mathsf{RO}(r\|x) \oplus \Delta$

return (r, M^f, Z)

$\mathsf{frRO}(x; r)$

  $\mathcal{F}.\mathit{mod} \leftarrow \mathsf{nof}$

  return $\mathsf{frRO}^f(x; r, \mathcal{F})$

CLAIMS ON THE frRO

Theorem
As long as the `bad` *event is never set, then* frRO *is indistinguishable from a fault-free random oracle.*

Conjecture
If a hash function H *is indifferentiable from a random oracle, then its faulty implementation with differential faults is indifferentiable from an* frRO*.*