Security of Hedged Fiat–Shamir Signatures under Fault Attacks

Eurocrypt 2020
ePrint 2019/956

Diego F. Aranha[1] Claudio Orlandi[1]
Akira Takahashi[1] Greg Zaverucha[2]

May 14, 2020

[1]Aarhus University, Denmark

[2]Microsoft Research, United States

This Talk in a Nutshell...

- **Goal**
    - Formally analyze the fault-resilience of existing Fiat–Shamir signatures, motivated by actual attacks.
- **Outline**
    1. Brief history of the fault attacks on FS signatures and randomness hedging.
    2. Fault attacker model.
    3. Overview of our provable security analysis.

Fiat–Shamir-type Signatures and Attacks

Signature from Canonical ID Protocol

Prover$(sk; r)$		Verifier(pk)
$(a, St) \leftarrow \mathsf{Com}(sk; r)$	$\xrightarrow{\quad a \quad}$	
	$\xleftarrow{\quad e \quad}$	$e \leftarrow_\$ C_H$
$z \leftarrow \mathsf{Resp}(sk, e, St)$	$\xrightarrow{\quad z \quad}$	$0/1 \leftarrow \mathsf{V}(a, e, z, pk)$

- If ID is special HVZK and special sound (=Σ-protocol), then SIG $:= \mathbf{FS}$[ID] is UF-CMA secure.

Signature from Canonical ID Protocol

Sign$(sk, m; r)$		Verifier(pk, m)
$(a, St) \leftarrow \mathsf{Com}(sk; r)$		
$e \leftarrow \mathsf{H}(a, m)$		
$z \leftarrow \mathsf{Resp}(sk, e, St)$	$\xrightarrow{a,\, e,\, z}$	$0/1 \leftarrow \mathsf{V}(a, e, z, pk)$
		$\mathsf{H}(a, m) \stackrel{?}{=} e$

- If ID is special HVZK and special sound (=Σ-protocol), then SIG $:= \mathbf{FS}[\text{ID}]$ is UF-CMA secure.
- e.g., Schnorr, Guillou–Quisquater, etc.

Signature from Canonical ID Protocol

Sign$(sk, m; r)$		Verifier(pk, m)
$(a, St) \leftarrow \mathsf{Com}(sk; r)$		
$e \leftarrow \mathsf{H}(a, m)$		
$z \leftarrow \mathsf{Resp}(sk, e, St)$	$\xrightarrow{a, e, z}$	$0/1 \leftarrow \mathsf{V}(a, e, z, pk)$
		$\mathsf{H}(a, m) \stackrel{?}{=} e$

- If ID is special HVZK and special sound (=Σ-protocol), then SIG $:= \mathbf{FS}$[ID] is UF-CMA secure.
- e.g., Schnorr, Guillou–Quisquater, etc.

Sensitivity of Per-signature Randomness

$$\boxed{\mathcal{A}} \qquad \xrightarrow{\quad m \quad} \qquad \boxed{\mathsf{RSign}(sk, m)}$$

$$r \leftarrow \mathsf{RNG}(\cdot)$$

$$(a, St) \leftarrow \mathsf{Com}(sk; r)$$

$$e \leftarrow \mathsf{H}(a, m)$$

$$\xleftarrow{\quad a,\ e,\ z \quad} \qquad z \leftarrow \mathsf{Resp}(sk, e, St)$$

- r must follow the uniform distribution.
- Otherwise there is an attack!

Randomness Failure in Practice

- Poorly designed RNGs.
- VM resets $\rightsquigarrow$ same snapshot will end up with the same seed.
- Side-channel leakage.
- and more…

BBC | Sign in | News | Sport | Weather | Shop | Earth | Travel | Mo

NEWS

Home | UK | World | Business | Politics | Tech | Science | Health | Family & Education | E

Technology

iPhone hacker publishes secret Sony PlayStation 3 key

By Jonathan Fildes
Technology reporter, BBC News

6 January 2011

Share

The PlayStation 3's security has been broken by hackers, potentially allowing anyone to run any software - including pirated games - on the console.

A collective of hackers recently showed off a method that could force the system to reveal secret keys used to load

BBC news. 2011. `https://www.bbc.com/news/technology-12116051`

Popular Solution: Deterministic Randomness Generation

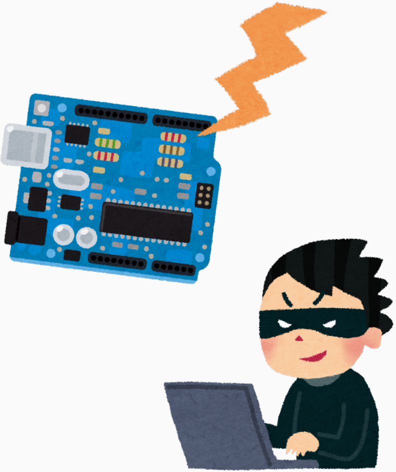
$$\cancel{r \leftarrow \mathsf{RNG}(\cdot)}$$

$$r \leftarrow \mathsf{H}'(sk, m)$$

- Hash each message keyed with sk.
- Widely implemented, e.g., in EdDSA, ECDSA, Dilithium, etc.
- However, another practical issue arises...

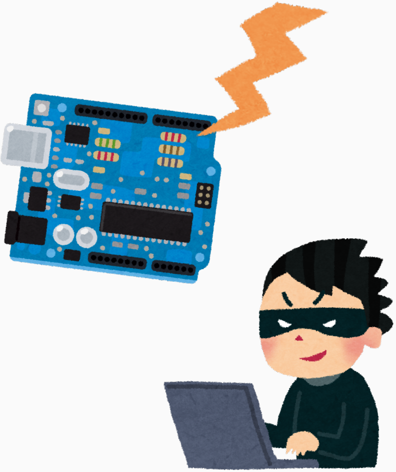
Deterministic FS is Vulnerable to Faults!

- Fault attack
  - Modifies the internal state of the device.
  - Can be performed remotely (e.g., Rowhammer)
- Many recent fault attacks on FS! [BP16, ABF+18, RP17, PSS+18, SB18, BP18, RJH+19]
- Idea: exploit determinism to rewind the prover (= signer).



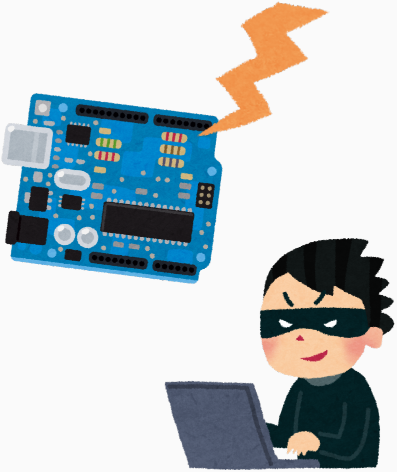
Deterministic FS is Vulnerable to Faults!

- Fault attack
  - Modifies the internal state of the device.
  - Can be performed remotely (e.g., Rowhammer)
- Many recent fault attacks on FS! [BP16, ABF+18, RP17, PSS+18, SB18, BP18, RJH+19]
- Idea: exploit determinism to rewind the prover (= signer).



Deterministic FS is Vulnerable to Faults!

- Fault attack
  - Modifies the internal state of the device.
  - Can be performed remotely (e.g., Rowhammer)
- Many recent fault attacks on FS! [BP16, ABF$^+$18, RP17, PSS$^+$18, SB18, BP18, RJH$^+$19]
- Idea: exploit determinism to rewind the prover (= signer).



Fault Adversary Type I: Special Soundness Attack

$\boxed{\mathcal{A}}$ $\xrightarrow{\quad m \quad}$ $\boxed{\mathsf{DSign}(sk, m)}$

$r \leftarrow \mathsf{H}'(sk, m)$

$(a, St) \leftarrow \mathsf{Com}(sk; r)$

$e \leftarrow \mathsf{H}(a, m)$

$\xleftarrow{\quad a, e, z \quad}$ $z \leftarrow \mathsf{Resp}(sk, e, St)$

- Query 1: get the legitimate signature (a, e, z) on m.
- Query 2: get a faulty signature $(a, \tilde{e}, \tilde{z})$ on the same m, by injecting fault on hash I/O or commitment output.
- Special soundness allows $\mathcal{A}$ to recover sk!

Fault Adversary Type I: Special Soundness Attack

$$\boxed{\mathcal{A}} \xrightarrow{\quad m,\ \lightning \quad} \boxed{\mathsf{DSign}(sk, m)}$$

$$r \leftarrow \mathsf{H}'(sk, m)$$

$$(a, St) \leftarrow \mathsf{Com}(sk; r)$$

$$\tilde{e} \leftarrow \mathsf{H}(a, m)\lightning$$

$$\xleftarrow{\quad a,\ \tilde{e},\ \tilde{z} \quad} \qquad \tilde{z} \leftarrow \mathsf{Resp}(sk, \tilde{e}, St)$$

- Query 1: get the legitimate signature (a, e, z) on m.
- Query 2: get a faulty signature $(a, \tilde{e}, \tilde{z})$ on the same m, by injecting fault on hash I/O or commitment output.
- Special soundness allows $\mathcal{A}$ to recover sk!

Fault Adversary Type I: Special Soundness Attack

$\boxed{\mathcal{A}}$ $\xrightarrow{m,\ \lightning}$ $\boxed{\mathsf{DSign}(sk, m)}$

$$r \leftarrow \mathsf{H}'(sk, m)$$

$$(a, St) \leftarrow \mathsf{Com}(sk; r)$$

$$\tilde{e} \leftarrow \mathsf{H}(a, m)\lightning$$

$$\tilde{z} \leftarrow \mathsf{Resp}(sk, \tilde{e}, St)$$

$\xleftarrow{a,\ \tilde{e},\ \tilde{z}}$

- Query 1: get the legitimate signature (a, e, z) on m.
- Query 2: get a faulty signature $(a, \tilde{e}, \tilde{z})$ on the same m, by injecting fault on hash I/O or commitment output.
- Special soundness allows $\mathcal{A}$ to recover sk!

Fault Adversary Type II: Large Randomness Bias Attack

$$
\begin{array}{lcl}
\boxed{\mathcal{A}} & \xrightarrow{\quad m \quad} & \boxed{\mathsf{DSign}(sk, m)} \\
 & & r \leftarrow \mathsf{H}'(sk, m) \\
 & & (a, St) \leftarrow \mathsf{Com}(sk; r) \\
 & & e \leftarrow \mathsf{H}(a, m) \\
 & \xleftarrow{\quad a, e, z \quad} & z \leftarrow \mathsf{Resp}(sk, e, St)
\end{array}
$$

- Query 1: get the legitimate signature (a, e, z) on m.
- Query 2: get a faulty signature $(\tilde{a}, \tilde{e}, \tilde{z})$ on the same m, by injecting fault on r or Resp input.
- Second signature relies on correlated randomness $\tilde{r} = r + \Delta$!

$\boxed{\mathcal{A}}$ $\xrightarrow{\quad m,\ \lightning \quad}$ $\boxed{\mathsf{DSign}(sk, m)}$

$$\tilde{r} \leftarrow \mathsf{H}'(sk, m)\lightning$$

$$(\tilde{a}, \tilde{St}) \leftarrow \mathsf{Com}(sk; \tilde{r})$$

$$\tilde{e} \leftarrow \mathsf{H}(\tilde{a}, m)$$

$\xleftarrow{\quad \tilde{a}, \tilde{e}, \tilde{z} \quad}$ $\tilde{z} \leftarrow \mathsf{Resp}(sk, \tilde{e}, \tilde{St})$

- Query 1: get the legitimate signature (a, e, z) on m.
- Query 2: get a faulty signature $(\tilde{a}, \tilde{e}, \tilde{z})$ on the same m, by injecting fault on r or Resp input.
- Second signature relies on correlated randomness $\tilde{r} = r + \Delta$!

Fault Adversary Type II: Large Randomness Bias Attack

$\boxed{\mathcal{A}}$		$\boxed{\mathsf{DSign}(sk, m)}$
	$\xrightarrow{m,\ \lightning}$	
		$\tilde{r} \leftarrow \mathsf{H}'(sk, m)\lightning$
		$(\tilde{a}, \tilde{St}) \leftarrow \mathsf{Com}(sk; \tilde{r})$
		$\tilde{e} \leftarrow \mathsf{H}(\tilde{a}, m)$
	$\xleftarrow{\tilde{a},\ \tilde{e},\ \tilde{z}}$	$\tilde{z} \leftarrow \mathsf{Resp}(sk, \tilde{e}, \tilde{St})$

- Query 1: get the legitimate signature (a, e, z) on m.
- Query 2: get a faulty signature $(\tilde{a}, \tilde{e}, \tilde{z})$ on the same m, by injecting fault on r or Resp input.
- Second signature relies on correlated randomness $\tilde{r} = r + \Delta$!

Better Countermeasure? – Randomness Hedging

$$\cancel{r \leftarrow \mathsf{RNG}(\cdot)}$$

$$\cancel{r \leftarrow \mathsf{H}'(sk, m)}$$

$$r \leftarrow \mathsf{H}'(sk, m, nonce)$$

- Nonces could be from low-quality PRNG, or just a counter.
- Randomness r doesn't repeat on the same message.
- Seems secure, but no formal analysis so far.

To what extent are hedged FS signatures secure against fault attacks?

Better Countermeasure? – Randomness Hedging

$$\cancel{r \leftarrow \mathsf{RNG}(\cdot)}$$

$$\cancel{r \leftarrow \mathsf{H}'(sk, m)}$$

$$r \leftarrow \mathsf{H}'(sk, m, nonce)$$

- Nonces could be from low-quality PRNG, or just a counter.
- Randomness r doesn't repeat on the same message.
- Seems secure, but no formal analysis so far.

To what extent are hedged FS signatures secure against fault attacks?

Better Countermeasure? – Randomness Hedging

$$\cancel{r \leftarrow \mathsf{RNG}(\cdot)}$$

$$\cancel{r \leftarrow \mathsf{H}'(sk, m)}$$

$$r \leftarrow \mathsf{H}'(sk, m, nonce)$$

- Nonces could be from low-quality PRNG, or just a counter.
- Randomness r doesn't repeat on the same message.
- Seems secure, but no formal analysis so far.

To what extent are hedged FS signatures secure against fault attacks?

Better Countermeasure? – Randomness Hedging

$$\cancel{r \leftarrow \mathsf{RNG}(\cdot)}$$

$$\cancel{r \leftarrow \mathsf{H}'(sk, m)}$$

$$r \leftarrow \mathsf{H}'(sk, m, nonce)$$

- Nonces could be from low-quality PRNG, or just a counter.
- Randomness r doesn't repeat on the same message.
- Seems secure, but no formal analysis so far.

To what extent are hedged FS signatures secure against fault attacks?

Contributions

- Formal attacker model and security notions to capture the corrupted nonces and previous fault attacks.
- Proved that hedged FS schemes in general are (in)secure against certain class of fault attacks.
- Application to concrete instantiations.
  - XEdDSA: Variant of EdDSA used in Signal
  - Picnic2: NIST PQC competition round 2 candidate

Attacker Model and Security Notions

Approach

- UF-fCMNA Security
  - UnForgeability against Faults, Chosen Message and Nonce Attacks
  - Models hedged construction and corrupted nonces (inspired by [BPS16, BT16]).
  - Equips the adversary with bit-tampering fault attacks.
  - Tailored to Fiat–Shamir.

Approach

- UF-fCMNA Security
    - UnForgeability against Faults, Chosen Message and Nonce Attacks
    - Models hedged construction and corrupted nonces (inspired by [BPS16, BT16]).
    - Equips the adversary with bit-tampering fault attacks.
    - Tailored to Fiat–Shamir.

Approach

- UF-fCMNA Security
  - UnForgeability against Faults, Chosen Message and Nonce Attacks
  - Models hedged construction and corrupted nonces (inspired by [BPS16, BT16]).
  - Equips the adversary with bit-tampering fault attacks.
  - Tailored to Fiat–Shamir.

Approach

- UF-fCMNA Security
  - UnForgeability against Faults, Chosen Message and Nonce Attacks
  - Models hedged construction and corrupted nonces (inspired by [BPS16, BT16]).
  - Equips the adversary with bit-tampering fault attacks.
  - Tailored to Fiat–Shamir.

Modeling Fault Attackers

- $\texttt{flip\_bit}_i(x)$ does a logical negation of the i-th bit of x.

$$\texttt{flip\_bit}_2(\texttt{0110}\ldots) \to \texttt{0010}\ldots$$

- $\texttt{set\_bit}_{i,b}(x)$ sets the i-th bit of x to b.

$$\texttt{set\_bit}_{4,1}(\texttt{0110}\ldots) \to \texttt{0111}\ldots$$

- Focuses on the single-bit faults, characterizing recent attacks on FS.
- Models most basic transient fault attackers on data flow, e.g.,
  - CPU register values
  - Data buses
  - Memory cells

Modeling Fault Attackers

- $\texttt{flip\_bit}_i(x)$ does a logical negation of the i-th bit of x.

$$\texttt{flip\_bit}_2(\texttt{0110}\ldots) \rightarrow \texttt{0010}\ldots$$

- $\texttt{set\_bit}_{i,b}(x)$ sets the i-th bit of x to b.

$$\texttt{set\_bit}_{4,1}(\texttt{0110}\ldots) \rightarrow \texttt{0111}\ldots$$

- Focuses on the single-bit faults, characterizing recent attacks on FS.
- Models most basic transient fault attackers on data flow, e.g.,
  - CPU register values
  - Data buses
  - Memory cells

Modeling Fault Attackers

- $\texttt{flip\_bit}_i(x)$ does a logical negation of the i-th bit of x.

$$\texttt{flip\_bit}_2(\texttt{0110}\ldots) \to \texttt{0010}\ldots$$

- $\texttt{set\_bit}_{i,b}(x)$ sets the i-th bit of x to b.

$$\texttt{set\_bit}_{4,1}(\texttt{0110}\ldots) \to \texttt{0111}\ldots$$

- Focuses on the single-bit faults, characterizing recent attacks on FS.
- Models most basic transient fault attackers on data flow, e.g.,
  - CPU register values
  - Data buses
  - Memory cells

Modeling Fault Attackers

- $\texttt{flip\_bit}_i(x)$ does a logical negation of the i-th bit of x.

$$\texttt{flip\_bit}_2(\texttt{0110}\ldots) \to \texttt{0010}\ldots$$

- $\texttt{set\_bit}_{i,b}(x)$ sets the i-th bit of x to b.

$$\texttt{set\_bit}_{4,1}(\texttt{0110}\ldots) \to \texttt{0111}\ldots$$

- Focuses on the single-bit faults, characterizing recent attacks on FS.
- Models most basic transient fault attackers on data flow, e.g.,
  - CPU register values
  - Data buses
  - Memory cells

UF-fCMNA Security

$\mathsf{Exp}^{\mathsf{UF\text{-}fCMNA}}_{\mathsf{HSIG},\mathsf{H},\mathsf{H}'}(\mathcal{A})$: UF-fCMNA experiment

$\mathcal{A}^{\mathsf{H},\mathsf{H}'}(pk)$		$\mathsf{OFaultHSign}_{sk}(\cdot,\cdot)$
	$\xrightarrow[f_j \in \{\texttt{set\_bit}_{i,b}, \texttt{flip\_bit}_i\}]{m_i,\ n}$	
		$r \leftarrow f_2(\mathsf{H}'(f_1(sk), f_0(m_i, n)))$
		$(a, St) \leftarrow f_4(\mathsf{Com}(f_3(sk; r)))$
		$\hat{a}, \hat{m}_i, \hat{pk} \leftarrow f_5(a, m_i, pk)$
		$e \leftarrow f_6(\mathsf{H}(\hat{a}, \hat{m}_i, \hat{pk}))$
	$\xleftarrow{a,\ e,\ z}$	$z \leftarrow f_8(\mathsf{Resp}(f_7(sk, e, St)))$
	$\xrightarrow{m^*,\ \sigma^*}$	Out $\mathsf{Verify}(pk, m^*, \sigma^*) \wedge m^* \neq \hat{m}_i$

- H and H′ are modeled as RO.
- HSIG is UF-fCMNA secure if $\Pr[\mathsf{Exp}^{\mathsf{UF\text{-}fCMNA}}_{\mathsf{HSIG},\mathsf{H},\mathsf{H}'}(\mathcal{A}) \rightarrow 1]$ is negligible.

UF-fCMNA Security

$\mathsf{Exp}^{\mathsf{UF\text{-}fCMNA}}_{\mathsf{HSIG},\mathsf{H},\mathsf{H}'}(\mathcal{A})$: UF-fCMNA experiment

$\mathcal{A}^{\mathsf{H},\mathsf{H}'}(pk)$		$\mathsf{OFaultHSign}_{sk}(\cdot,\cdot)$
	$\xrightarrow[f_j \in \{\texttt{set\_bit}_{i,b},\, \texttt{flip\_bit}_i\}]{m_i,\, n}$	
		$r \leftarrow f_2(\mathsf{H}'(f_1(sk), f_0(m_i, n)))$
		$(a, St) \leftarrow f_4(\mathsf{Com}(f_3(sk; r)))$
		$\hat{a}, \hat{m}_i, \hat{pk} \leftarrow f_5(a, m_i, pk)$
		$e \leftarrow f_6(\mathsf{H}(\hat{a}, \hat{m}_i, \hat{pk}))$
	$\xleftarrow{a,\, e,\, z}$	$z \leftarrow f_8(\mathsf{Resp}(f_7(sk, e, St)))$
	$\xrightarrow{m^*,\, \sigma^*}$	Out $\mathsf{Verify}(pk, m^*, \sigma^*) \wedge m^* \neq \hat{m}_i$

- H and H′ are modeled as RO.
- HSIG is UF-fCMNA secure if $\Pr[\mathsf{Exp}^{\mathsf{UF\text{-}fCMNA}}_{\mathsf{HSIG},\mathsf{H},\mathsf{H}'}(\mathcal{A}) \to 1]$ is negligible.

UF-fCMNA Security

$\mathsf{Exp}^{\mathsf{UF\text{-}fCMNA}}_{\mathsf{HSIG},\mathsf{H},\mathsf{H}'}(\mathcal{A})$: UF-fCMNA experiment

$\mathcal{A}^{\mathsf{H},\mathsf{H}'}(pk)$		$\mathsf{OFaultHSign}_{sk}(\cdot,\cdot)$
	$\xrightarrow[f_j \in \{\texttt{set\_bit}_{i,b}, \texttt{flip\_bit}_i\}]{m_i,\, n}$	
		$r \leftarrow f_2(\mathsf{H}'(f_1(sk), f_0(m_i, n)))$
		$(a, St) \leftarrow f_4(\mathsf{Com}(f_3(sk; r)))$
		$\hat{a}, \hat{m}_i, \hat{pk} \leftarrow f_5(a, m_i, pk)$
		$e \leftarrow f_6(\mathsf{H}(\hat{a}, \hat{m}_i, \hat{pk}))$
	$\xleftarrow{a,\, e,\, z}$	$z \leftarrow f_8(\mathsf{Resp}(f_7(sk, e, St)))$
	$\xrightarrow{m^*,\, \sigma^*}$	Out $\mathsf{Verify}(pk, m^*, \sigma^*) \wedge m^* \neq \hat{m}_i$

- H and H′ are modeled as RO.
- HSIG is UF-fCMNA secure if $\Pr[\mathsf{Exp}^{\mathsf{UF\text{-}fCMNA}}_{\mathsf{HSIG},\mathsf{H},\mathsf{H}'}(\mathcal{A}) \to 1]$ is negligible.

UF-fCMNA Security

$\mathsf{Exp}^{\mathsf{UF\text{-}fCMNA}}_{\mathsf{HSIG},\mathsf{H},\mathsf{H}'}(\mathcal{A})$: UF-fCMNA experiment

$\mathcal{A}^{\mathsf{H},\mathsf{H}'}(pk)$		$\mathsf{OFaultHSign}_{sk}(\cdot,\cdot)$
	$\xrightarrow[f_j \in \{\texttt{set\_bit}_{i,b},\, \texttt{flip\_bit}_i\}]{m_i,\, n}$	
		$r \leftarrow f_2(\mathsf{H}'(f_1(sk), f_0(m_i, n)))$
		$(a, St) \leftarrow f_4(\mathsf{Com}(f_3(sk; r)))$
		$\hat{a}, \hat{m}_i, \hat{pk} \leftarrow f_5(a, m_i, pk)$
		$e \leftarrow f_6(\mathsf{H}(\hat{a}, \hat{m}_i, \hat{pk}))$
	$\xleftarrow{a,\, e,\, z}$	$z \leftarrow f_8(\mathsf{Resp}(f_7(sk, e, St)))$
	$\xrightarrow{m^*,\, \sigma^*}$	Out $\mathsf{Verify}(pk, m^*, \sigma^*) \wedge m^* \neq \hat{m}_i$

- H and H′ are modeled as RO.
- HSIG is UF-fCMNA secure if $\Pr[\mathsf{Exp}^{\mathsf{UF\text{-}fCMNA}}_{\mathsf{HSIG},\mathsf{H},\mathsf{H}'}(\mathcal{A}) \to 1]$ is negligible.

Provable Security Analysis

Security Proof Overview

$$\text{UF-KOA} \xrightarrow[\text{Non-repeating } (m,n)]{\text{special HVZK}} \text{UF-fCMNA for } \{f_1, f_5, f_6, f_8, f_9, f_{10}\}$$

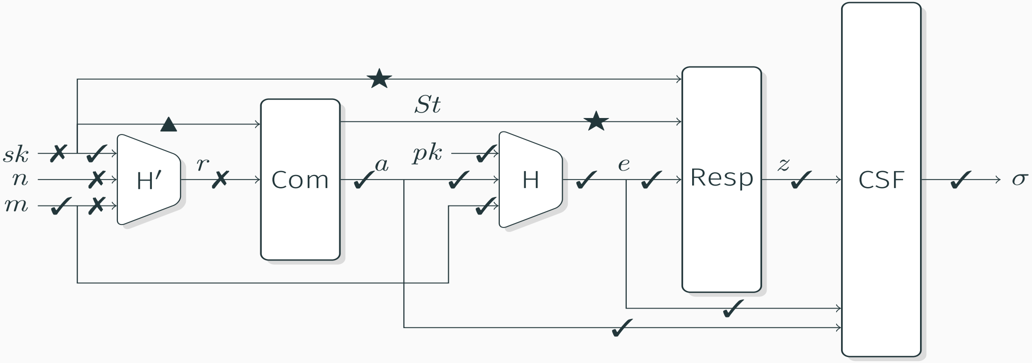
- UF-KOA (Key Only Attack): $\mathcal{A}$ is not given signing oracle.
- UF-KOA $\rightarrow$ UF-fCMNA
  - Simulate the faulty HSign oracle by invoking special HVZK simulator.
  - Non-repeating (message, nonce) is crucial, since otherwise the scheme is deterministic!

Security Proof Overview

$$\text{UF-KOA} \xrightarrow[\text{Non-repeating } (m,n)]{\text{special HVZK}} \text{UF-fCMNA for } \{f_1, f_5, f_6, f_8, f_9, f_{10}\}$$

- UF-KOA (Key Only Attack): $\mathcal{A}$ is not given signing oracle.
- UF-KOA $\rightarrow$ UF-fCMNA
  - Simulate the faulty HSign oracle by invoking special HVZK simulator.
  - Non-repeating (message, nonce) is crucial, since otherwise the scheme is deterministic!

Overview of Our Results



If $\mathcal{A}$ doesn't query the same (m, n) pair more than once

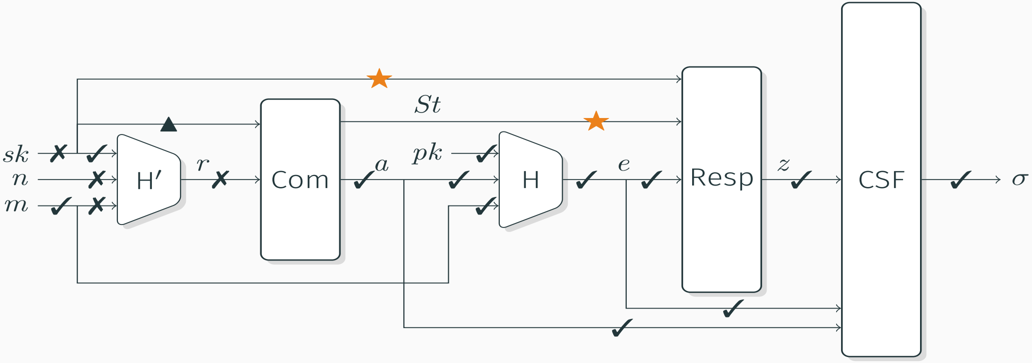
✓ secure against single-bit faults.

✗ insecure against single-bit faults.

★ security only holds for signatures from subset-revealing ID (e.g., Picnic).

▲ security only holds for signatures from input-delayed ID (e.g., XEdDSA).

Overview of Our Results



If $\mathcal{A}$ doesn't query the same (m, n) pair more than once

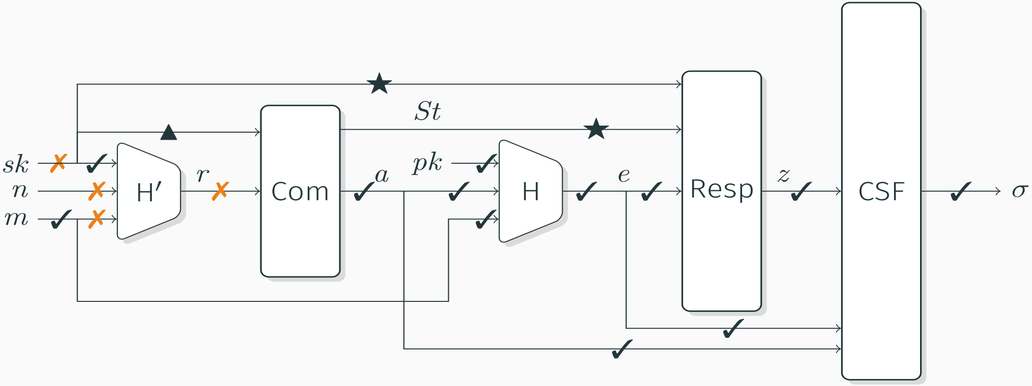
- ✓ secure against single-bit faults.
- ✗ insecure against single-bit faults.
- ★ security only holds for signatures from subset-revealing ID (e.g., Picnic).
- ▲ security only holds for signatures from input-delayed ID (e.g., XEdDSA).

Main Positive Result + Subset Revealing ID

Prover$(sk; r)$		Verifier(pk)
$(a, \{St_1, \ldots, St_c\}) \leftarrow \mathsf{Com}(sk; r)$	$\xrightarrow{\quad a \quad}$	
	$\xleftarrow{\quad e \subseteq [1, c] \quad}$	$e \leftarrow_{\$} C_H$
$z \leftarrow \{St_i\}_{i \in e}$	$\xrightarrow{\quad z \quad}$	$0/1 \leftarrow \mathsf{V}(a, e, z, pk)$

Intuition: $\{St_i\}$ is resilient to faults since it doesn't rely on sk!

Negative Results



- Fault on H′ input $(m, n) \rightsquigarrow$ degenerates to deterministic signature.
- Fault on H′ output $r \rightsquigarrow$ directly causes randomness bias.
  - Remark: still better than DSign, as large randomness bias doesn't occur.

Application to Concrete Schemes

XEdDSA

- EdDSA is essentially a deterministic Schnorr.
- XEdDSA = hedged Schnorr.
- More fault resilient than EdDSA/Schnorr!
- Already deployed in Signal protocol.

Picnic2

- Derived from ZKP based on MPC-in-the-head by [KKW18].
- Picnic2 follows FS.
- Underlying ZKP is subset-revealing $\rightsquigarrow$ Hedged Picnic2 has more fault resistance!
- Specification recommends randomness hedging.

XEdDSA and Picnic2

XEdDSA

- EdDSA is essentially a deterministic Schnorr.
- XEdDSA = hedged Schnorr.
- More fault resilient than EdDSA/Schnorr!
- Already deployed in Signal protocol.

Picnic2

- Derived from ZKP based on MPC-in-the-head by [KKW18].
- Picnic2 follows FS.
- Underlying ZKP is subset-revealing $\rightsquigarrow$ Hedged Picnic2 has more fault resistance!
- Specification recommends randomness hedging.

XEdDSA and Picnic2

XEdDSA

- EdDSA is essentially a deterministic Schnorr.
- XEdDSA = hedged Schnorr.
- More fault resilient than EdDSA/Schnorr!
- Already deployed in Signal protocol.

Picnic2

- Derived from ZKP based on MPC-in-the-head by [KKW18].
- Picnic2 follows FS.
- Underlying ZKP is subset-revealing $\rightsquigarrow$ Hedged Picnic2 has more fault resistance!
- Specification recommends randomness hedging.

XEdDSA and Picnic2

XEdDSA

- EdDSA is essentially a deterministic Schnorr.
- XEdDSA = hedged Schnorr.
- More fault resilient than EdDSA/Schnorr!
- Already deployed in Signal protocol.

Picnic2

- Derived from ZKP based on MPC-in-the-head by [KKW18].
- Picnic2 follows FS.
- Underlying ZKP is subset-revealing $\rightsquigarrow$ Hedged Picnic2 has more fault resistance!
- Specification recommends randomness hedging.

XEdDSA and Picnic2

XEdDSA

- EdDSA is essentially a deterministic Schnorr.
- XEdDSA = hedged Schnorr.
- More fault resilient than EdDSA/Schnorr!
- Already deployed in Signal protocol.

Picnic2

- Derived from ZKP based on MPC-in-the-head by [KKW18].
- Picnic2 follows FS.
- Underlying ZKP is subset-revealing $\rightsquigarrow$ Hedged Picnic2 has more fault resistance!
- Specification recommends randomness hedging.

XEdDSA and Picnic2

XEdDSA

- EdDSA is essentially a deterministic Schnorr.
- XEdDSA = hedged Schnorr.
- More fault resilient than EdDSA/Schnorr!
- Already deployed in Signal protocol.

Picnic2

- Derived from ZKP based on MPC-in-the-head by [KKW18].
- Picnic2 follows FS.
- Underlying ZKP is subset-revealing $\rightsquigarrow$ Hedged Picnic2 has more fault resistance!
- Specification recommends randomness hedging.

XEdDSA

- EdDSA is essentially a deterministic Schnorr.
- XEdDSA = hedged Schnorr.
- More fault resilient than EdDSA/Schnorr!
- Already deployed in Signal protocol.

Picnic2

- Derived from ZKP based on MPC-in-the-head by [KKW18].
- Picnic2 follows FS.
- Underlying ZKP is subset-revealing $\rightsquigarrow$ Hedged Picnic2 has more fault resistance!
- Specification recommends randomness hedging.

XEdDSA and Picnic2

XEdDSA

- EdDSA is essentially a deterministic Schnorr.
- XEdDSA = hedged Schnorr.
- More fault resilient than EdDSA/Schnorr!
- Already deployed in Signal protocol.

Picnic2

- Derived from ZKP based on MPC-in-the-head by [KKW18].
- Picnic2 follows FS.
- Underlying ZKP is subset-revealing $\rightsquigarrow$ Hedged Picnic2 has more fault resistance!
- Specification recommends randomness hedging.

Conclusion

- Defined formal model and security notions tailored to FS.
- Proved (in)security of hedged FS signatures against basic faults and corrupt nonces.
- Hedging is provably more resilient than the randomized/deterministic FS, but H' input/output should be protected!
- Open questions
  - Extension to more advanced fault attacker model.
    - Multi-bit/position faults. Partially handled by Fischlin and Günther [FG20] (CT-RSA'20) for generic signatures.
    - Fault within Com, Resp or public parameters.
    - Model for instruction skipping faults.
    - Fault + QROM.
  - Lattice signatures from FS with aborts.

Thank you!

More details in ePrint 2019/956

Conclusion

- Defined formal model and security notions tailored to FS.
- Proved (in)security of hedged FS signatures against basic faults and corrupt nonces.
- Hedging is provably more resilient than the randomized/deterministic FS, but H' input/output should be protected!
- Open questions
    - Extension to more advanced fault attacker model.
        - Multi-bit/position faults. Partially handled by Fischlin and Günther [FG20] (CT-RSA'20) for generic signatures.
        - Fault within Com, Resp or public parameters.
        - Model for instruction skipping faults.
        - Fault + QROM.
    - Lattice signatures from FS with aborts.

Thank you!

More details in ePrint 2019/956

Conclusion

- Defined formal model and security notions tailored to FS.
- Proved (in)security of hedged FS signatures against basic faults and corrupt nonces.
- Hedging is provably more resilient than the randomized/deterministic FS, but H' input/output should be protected!
- Open questions
  - Extension to more advanced fault attacker model.
    - Multi-bit/position faults. Partially handled by Fischlin and Günther [FG20] (CT-RSA'20) for generic signatures.
    - Fault within Com, Resp or public parameters.
    - Model for instruction skipping faults.
    - Fault + QROM.
  - Lattice signatures from FS with aborts.

Thank you!

More details in ePrint 2019/956

Conclusion

- Defined formal model and security notions tailored to FS.
- Proved (in)security of hedged FS signatures against basic faults and corrupt nonces.
- Hedging is provably more resilient than the randomized/deterministic FS, but H' input/output should be protected!
- Open questions
  - Extension to more advanced fault attacker model.
    - Multi-bit/position faults. Partially handled by Fischlin and Günther [FG20] (CT-RSA'20) for generic signatures.
    - Fault within Com, Resp or public parameters.
    - Model for instruction skipping faults.
    - Fault + QROM.
  - Lattice signatures from FS with aborts.

Thank you!

More details in ePrint 2019/956

Conclusion

- Defined formal model and security notions tailored to FS.
- Proved (in)security of hedged FS signatures against basic faults and corrupt nonces.
- Hedging is provably more resilient than the randomized/deterministic FS, but H' input/output should be protected!
- Open questions
    - Extension to more advanced fault attacker model.
        - Multi-bit/position faults. Partially handled by Fischlin and Günther [FG20] (CT-RSA'20) for generic signatures.
        - Fault within Com, Resp or public parameters.
        - Model for instruction skipping faults.
        - Fault + QROM.
    - Lattice signatures from FS with aborts.

Thank you!

More details in ePrint 2019/956

References i

- Christopher Ambrose, Joppe W. Bos, Björn Fay, Marc Joye, Manfred Lochter, and Bruce Murray.
  Differential attacks on deterministic signatures.
  In Nigel P. Smart, editor, *CT-RSA 2018*, volume 10808 of *LNCS*, pages 339–353. Springer, Heidelberg, April 2018.
- Alessandro Barenghi and Gerardo Pelosi.
  A note on fault attacks against deterministic signature schemes.
  In Kazuto Ogawa and Katsunari Yoshioka, editors, *IWSEC 16*, volume 9836 of *LNCS*, pages 182–192. Springer, Heidelberg, September 2016.

Leon Groot Bruinderink and Peter Pessl.
Differential fault attacks on deterministic lattice signatures.
IACR TCHES, 2018(3):21–43, 2018.
https://tches.iacr.org/index.php/TCHES/article/view/7267.

Mihir Bellare, Bertram Poettering, and Douglas Stebila.
From identification to signatures, tightly: A framework and generic transforms.
In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 435–464. Springer, Heidelberg, December 2016.

Mihir Bellare and Björn Tackmann.
Nonce-based cryptography: Retaining security when randomness fails.
In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 729–757. Springer, Heidelberg, May 2016.

Marc Fischlin and Felix Günther.
Modeling memory faults in signature and authenticated encryption schemes.
In Stanislaw Jarecki, editor, *CT-RSA 2020*, volume 12006 of *LNCS*, pages 56–84. Springer, 2020.

Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang.
Improved non-interactive zero knowledge with applications to post-quantum signatures.
In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 525–537. ACM Press, October 2018.

Damian Poddebniak, Juraj Somorovsky, Sebastian Schinzel, Manfred Lochter, and Paul Rosler.
Attacking Deterministic Signature Schemes using Fault Attacks.
In *Euro S&P 2018*, pages 338–352. IEEE, 2018.

Prasanna Ravi, Mahabir Prasad Jhanwar, James Howe, Anupam Chattopadhyay, and Shivam Bhasin.
Exploiting Determinism in Lattice-based Signatures: Practical Fault Attacks on Pqm4 Implementations of NIST Candidates.
In *Asia CCS 2019*, Asia CCS '19, pages 427–440. ACM, 2019.

Y. Romailler and S. Pelissier.
Practical Fault Attack against the Ed25519 and EdDSA Signature Schemes.
In *FDTC 2017*, pages 17–24, September 2017.

Niels Samwel and Lejla Batina.
Practical fault injection on deterministic signatures: The case of EdDSA.
In Antoine Joux, Abderrahmane Nitaj, and Tajjeeddine Rachidi, editors, *AFRICACRYPT 18*, volume 10831 of *LNCS*, pages 306–321. Springer, Heidelberg, May 2018.