RSAConference2019

San Francisco | March 4 – 8 | Moscone Center

BETTER.

SESSION ID: CRYP-R03

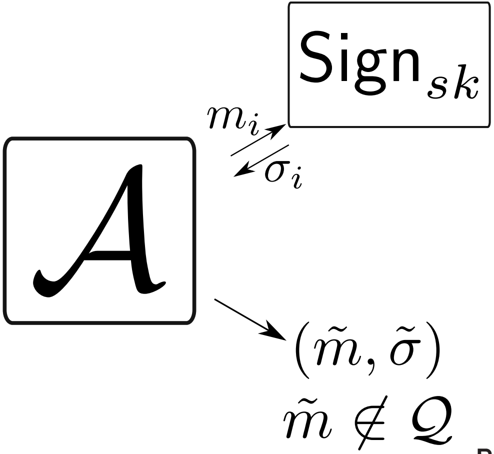
Efficient Fully-Leakage Resilient One-More Signature Schemes

Antonio Faonio

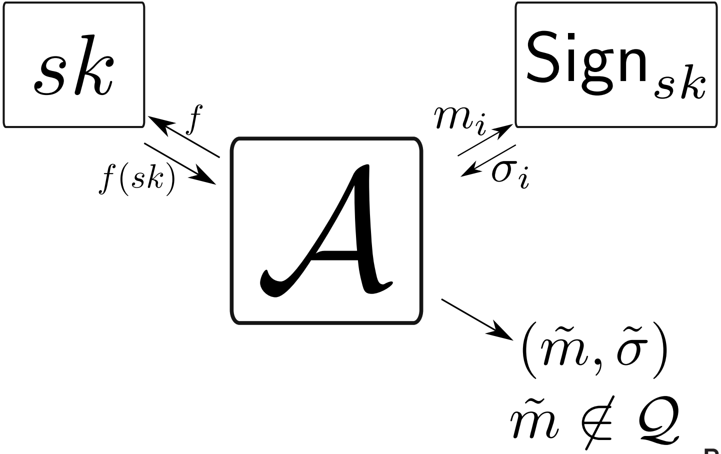
IMDEA Software Institute

SPECTRE

Digital Signature - Existential Unforgeability CMA



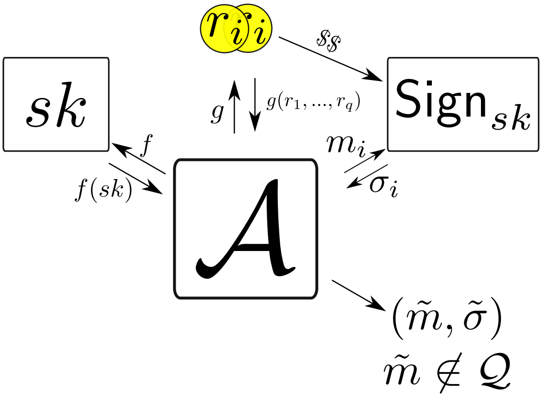
Digital Signature - Existential Unforgeability CMA



Cryptographers seldom sleep well

Silvio Micali

Digital Signature - Existential Unforgeability CMA



Boyle, Segev, Wichs - EC'11 and Malkin et al - TCC'11

Let $f_1, f_2, \ldots$ adaptively chosen leakage functions:

Bounded Leakage Model

$$\sum_i |f_i(SK)| \leqslant \lambda < |SK|$$

Where λ is the leakage parameter.

Our Goal: Small Signatures AND Large Leakage Resilience

$$|\sigma| \ll \lambda < |sk|$$

Our Goal: Small Signatures AND Large Leakage Resilience

$$|\sigma| \ll \lambda < |sk|$$

$\mathcal{A}$ can always leak $f(sk) := \text{Sign}_{sk}(m)$.

Our Goal: Small Signatures AND Large Leakage Resilience

$$|\sigma| \ll \lambda < |sk|$$

$\mathcal{A}$ can always leak $f(sk) := \text{Sign}_{sk}(m)$.

Even worse...

Let $n = \lceil \frac{\lambda}{|\sigma|} \rceil$, $\mathcal{A}$ can always leak

$$f(sk) := (\text{Sign}_{sk}(m_1), \text{Sign}_{sk}(m_1), \ldots, \text{Sign}_{sk}(m_n)).$$

One More Unforgeability [NielsenVZ PKC'13, FaonioNV ICALP'15]

$\mathcal{A}$ can forge $n := \lceil \lambda / |\sigma| \rceil$ signature

One More Unforgeability [NielsenVZ PKC'13, FaonioNV ICALP'15]

$\mathcal{A}$ can forge $n := \lceil \lambda / |\sigma| \rceil$ signature

but not $n + 1$.

One More Unforgeability [NielsenVZ PKC'13, FaonioNV ICALP'15]

$\mathcal{A}$ can forge $n := \lceil \lambda / |\sigma| \rceil$ signature

but not $n + 1$.

Graceful degradation:

- If $\lambda = 0$ then standard notion of EUF;
- If $\lambda < |\sigma|$ then standard notion of LR-EUF;
- If $\lambda \geqslant |\sigma|$ then the $\mathcal{A}$ **cannot** forge more signatures than it can leak: the best it can do.

Weird Looking Scheme

- Let Sign be one-more leakage-resilient unforgeable.
- Define $\mathsf{Sign}'(sk, M)$ to output $(\sigma \| \sigma)$ where $\sigma \leftarrow \mathsf{Sign}(sk, M)$.

Weird Looking Scheme

- Let Sign be one-more leakage-resilient unforgeable.
- Define $\mathsf{Sign}'(sk, M)$ to output $(\sigma \| \sigma)$ where $\sigma \leftarrow \mathsf{Sign}(sk, M)$.

Introducing the slack parameter γ:

$$n = \frac{1}{\gamma} \cdot \left\lceil \frac{\lambda}{|\sigma|} \right\rceil$$

Contributions

Scheme	Fully	γ	Assumption
NVZ14	✗	$O(1)$	DLIN
$\mathrm{FNV15}_2$	✓	$O(1/q_{\mathsf{sign}})$	DLIN
$\mathcal{SS}_1$	✓	$O(1/k)$	SXDH
$\mathcal{SS}_2$	✓	1	KEA

Roadmap

The Marvelous Knowledge of The Exponent Assumption

A Simplified Scheme

Ideas behind the Proof

Efficiency

KEA-based Pedersen Commitment

- Let $[\vec{h}, \alpha\vec{h}]_1 \in \mathbb{G}_1^{2\times 2}$ the commitment key [1]
- Let $\mathsf{Commit}(m, r) := (m, r) \cdot [\vec{h}, \alpha\vec{h}]_1$

The commitment scheme is extractable

[1] We use the implicit notation where $[x]_1 := g_1^x \in \mathbb{G}_1$.

KEA-based Pedersen Commitment

- Let $[\vec{h}, \alpha\vec{h}]_1 \in \mathbb{G}_1^{2\times 2}$ the commitment key [1]
- Let $\mathsf{Commit}(m, r) := (m, r) \cdot [\vec{h}, \alpha\vec{h}]_1$

The commitment scheme is extractable and perfectly hiding!

[1]We use the implicit notation where $[x]_1 := g_1^x \in \mathbb{G}_1$.

KEA-based Pedersen Commitment

- Let $[\vec{h}, \alpha\vec{h}]_1 \in \mathbb{G}_1^{2\times 2}$ the commitment key [1]
- Let $\mathsf{Commit}(m, r) := (m, r) \cdot [\vec{h}, \alpha\vec{h}]_1$

The commitment scheme is extractable and perfectly hiding!

KE-Pedersen is linearly homomorphic!

[1] We use the implicit notation where $[x]_1 := g_1^x \in \mathbb{G}_1$.

Perfect Hiding and Leakage from Randomness

Process 1

- $c = \text{Commit}(s, r)$
- Leak $l = f(r)$
- **Output** (c, l, s)

Process 2

- $c = \text{Commit}(0, r')$
- Leak $l = f'(s)$ where:
  1. Find r s.t. $c = \text{Commit}(s, r)$,
  2. return $f(r)$
- **Output** (c, l, s)

Perfect Hiding and Leakage from Randomness

Process 1

- $c = \text{Commit}(s, r)$
- Leak $l = f(r)$
- **Output** (c, l, s)

Process 2

- $c = \text{Commit}(0, r')$
- Leak $l = f'(s)$ where:
  1. Find r s.t. $c = \text{Commit}(s, r)$,
  2. return $f(r)$
- **Output** (c, l, s)

We **reduce** leakage on r to leakage on s

Perfect Indistinghuishability is the **perfect** tool against leakage from the randomness!

Section 2

A Simplified Scheme

Main Ingredients

- **KEA-Pedersen Commitment**.
- **Perfect NIZK** for knowledge of the "opening of a Pedersen".

Signature Scheme

sk= $\delta_0 \; \delta_1 \cdots \delta_n$ Coefficients of a polynomial

pk= $[c_0] \; [c_1] \cdots [c_n]$

Commitments of the secretkey elements

$\delta_i, \delta, m \in \mathbb{F}$

Sign(m)

$\delta = \sum_i \delta_i m^i$

$c = \sum_i [c_i] m^i$

1. $\bar{c}$ =Com(δ)
2. π =Prove(c,$\bar{c}$,δ)

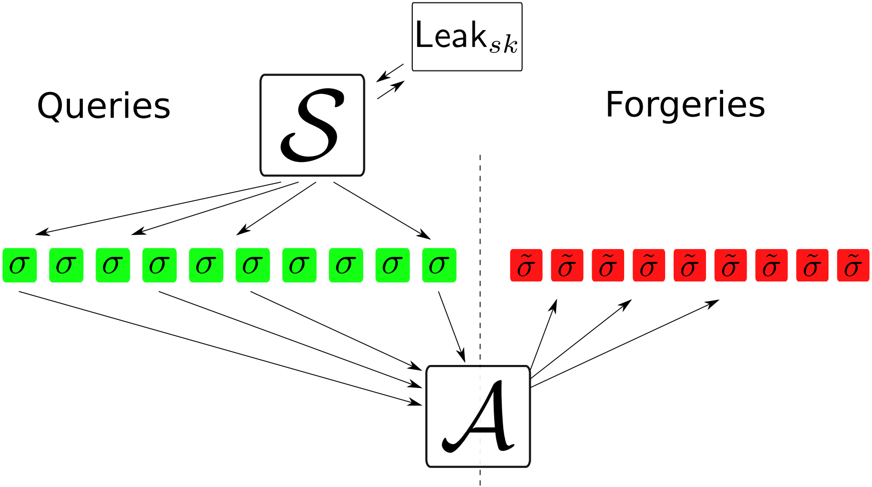
Relation

$$\left\{ (c,\bar{c}),\delta \;\middle|\; \begin{array}{l} c = \mathrm{Com}(\delta) \\ \bar{c} = \mathrm{Com}(\delta) \end{array} \right\}$$

σ= $\bar{c} \; \pi$

Section 3

Ideas behind the Proof


Leak$_{sk}$
Queries
$\mathcal{S}$
Forgeries
σ σ σ σ σ σ σ σ σ σ
$\tilde{\sigma}$ $\tilde{\sigma}$ $\tilde{\sigma}$ $\tilde{\sigma}$ $\tilde{\sigma}$ $\tilde{\sigma}$ $\tilde{\sigma}$ $\tilde{\sigma}$ $\tilde{\sigma}$
$\mathcal{A}$

Extractability of KEA-based Pedersen kicks in!

- From **signature** of m we **extract** $\sum_i \delta_i m^i$.
- With $n+1$ we can **interpolate** the polynomial.

Extractability of KEA-based Pedersen kicks in!

- From **signature** of m we **extract** $\sum_i \delta_i m^i$.
- With $n+1$ we can **interpolate** the polynomial.

The absurd.

- With $\mathbb{P}[\mathcal{A} \text{ wins}]$ the δ is uniquely defined.
- Leakage $\ell = |\delta| - k$ then guess with prob. $1/2^k$

Efficiency

- Kiltz-Wee QA-NIZK for subspace + KEA

Efficiency

- Kiltz-Wee QA-NIZK for subspace + KEA

Signature size : 8 group elements;
Sign : constant number exp;
Verify : constant number of pairing.

Efficient Fully-Leakage-Resilient Signatures with Graceful Degradation

Antonio Faonio

IMDEA Software Institute

Thanks!