Leakage Resilient Signatures with Graceful Degradation

J. B. Nielsen, D. Venturi, A. Zottarel

Aarhus University,

Sapienza University of Rome

March 27th, 2014

Authentication

How do we guarantee authenticity of our message?

Authentication

How do we guarantee authenticity of our message?

$\Downarrow$

Message is sent along with a signature

Authentication

How do we guarantee authenticity of our message?

$\Downarrow$

Message is sent along with a signature

Signature Schemes

- Signature scheme = (Gen, Sig, Ver)
- $Gen(1^k)$: generate a signing/verification key tuple
- $Sign(sk, m)$: generate a signature on a message
- $Ver(m, \sigma)$: outputs 0 or 1.

Authentication

How do we guarantee authenticity of our message?

$\Downarrow$

Message is sent along with a signature

Signature Schemes

- Signature scheme $= (Gen, Sig, Ver)$
- $Gen(1^k)$: generate a signing/verification key tuple
- $Sign(sk, m)$: generate a signature on a message
- $Ver(m, \sigma)$: outputs 0 or 1.

Existential Unforgeability

- Adversary has access to signing oracle for messages of his choice.
- Adversary outputs forgery $Sig_{sk}(m^*)$ for m^* of his choice
  m^* not asked to the signing oracle.

Signatures in a leaky world

- extend this model capturing leakage

Signatures in a leaky world

- extend this model capturing leakage
- modeling adversary extracting bits from the secret key:

Signatures in a leaky world

- extend this model capturing leakage
- modeling adversary extracting bits from the secret key:
  - leakage function $h : sk \mapsto \{0,1\}^*$.

Signatures in a leaky world

- extend this model capturing leakage
- modeling adversary extracting bits from the secret key:
  - leakage function $h : sk \mapsto \{0, 1\}^*$.
  - bounded model: $|h(sk)| < \lambda < |sk|$

Signatures in a leaky world

- extend this model capturing leakage
- modeling adversary extracting bits from the secret key:
  - leakage function $h : sk \mapsto \{0,1\}^*$.
  - bounded model: $|h(sk)| < \lambda < |sk|$

Signatures in the Bounded Model

- Adversary has access to signing oracle
  and
  oracle $\mathcal{O}^{(sk)}(h)$ returning $h(sk)$
- Adversary outputs forgery $Sign_{sk}(m^*)$ for m^* of his choice
  m^* not asked to the signing oracle.

If signatures are short..

- if leakage bigger then $|\sigma|$ -> adversary can leak a signature!

If signatures are short..

- if leakage bigger then $|\sigma|$ -> adversary can leak a signature!

Entropic Unforgeability [ADW09]

- Adversary does not choose the challenge message.
- Challenge message is drawn from a high min-entropy distribution

If signatures are short..

- if leakage bigger then $|\sigma|$ -> adversary can leak a signature!

Entropic Unforgeability [ADW09]

- Adversary does not choose the challenge message.
- Challenge message is drawn from a high min-entropy distribution

This Work

New model for signatures in the bounded model:
Number of forgeries depends on the amount of leakage

One-more Unforgeability

- signature scheme $S = (Gen, Sign, Ver)$, adversary A

One-more Unforgeability

- signature scheme $S = (Gen, Sign, Ver)$, adversary A

$Exp_{S,A}^{one-more}(k, \lambda, \gamma)$

1. A is given verification key vk

One-more Unforgeability

- signature scheme $S = (Gen, Sign, Ver)$, adversary A

$Exp_{S,A}^{one-more}(k, \lambda, \gamma)$

1. A is given verification key vk
2. A can query Sign(sk, $\cdot$) and $\mathcal{O}^{(sk)}$

One-more Unforgeability

- signature scheme $S = (Gen, Sign, Ver)$, adversary A

$Exp^{one-more}_{S,A}(k, \lambda, \gamma)$

1. A is given verification key vk
2. A can query Sign(sk, $\cdot$) and $\mathcal{O}^{(sk)}$
3. A outputs $(m_1, \sigma_1), \ldots, (m_n, \sigma_n)$

One-more Unforgeability

- signature scheme $S = (Gen, Sign, Ver)$, adversary A

$Exp^{one-more}_{S,A}(k, \lambda, \gamma)$

1. A is given verification key vk
2. A can query Sign(sk, $\cdot$) and $\mathcal{O}^{(sk)}$
3. A outputs $(m_1, \sigma_1), \ldots, (m_n, \sigma_n)$

Exp outputs 1 $\iff$

- $Ver(m_i, \sigma_i) = 1$ for every i
- $m_1, \ldots, m_n$ are pairwise distinct
- m_i were not asked to $Sign_{sk}$
- $n \geq \lfloor \lambda / (\gamma |\sigma|) \rfloor + 1$

Remark on Parameters

$$n \geq \lfloor \lambda / (\gamma |\sigma|) \rfloor + 1$$

γ = 1 implies optimal security

λ = 0 implies n = 1 -> standard unforgeability without leakage

$\lambda < |\sigma|$ implies n = 1 -> standard leakage resilience

$\lambda > |\sigma|$ "graceful" degradation

Remark on Parameters

$$n \geq \lfloor \lambda / (\gamma |\sigma|) \rfloor + 1$$

- γ = 1 implies optimal security
- λ = 0 implies n = 1 -> standard unforgeability without leakage
- $\lambda < |\sigma|$ implies n = 1 -> standard leakage resilience
- $\lambda > |\sigma|$ "graceful" degradation

best we can ask for:

Remark on Parameters

$$n \geq \lfloor \lambda / (\gamma |\sigma|) \rfloor + 1$$

γ $= 1$ implies optimal security

λ $= 0$ implies n $= 1$ -> standard unforgeability without leakage

λ $< |\sigma|$ implies n $= 1$ -> standard leakage resilience

λ $> |\sigma|$ "graceful" degradation

best we can ask for:

A can always leak at least $\lfloor \lambda / |\sigma| \rfloor$ signatures

Remark on Parameters

$$n \geq \lfloor \lambda / (\gamma |\sigma|) \rfloor + 1$$

γ = 1 implies optimal security

λ = 0 implies n = 1 -> standard unforgeability without leakage

$\lambda < |\sigma|$ implies n = 1 -> standard leakage resilience

$\lambda > |\sigma|$ "graceful" degradation

best we can ask for:

A can always leak at least $\lfloor \lambda / |\sigma| \rfloor$ signatures

$\Downarrow$

security implies A cannot forger even a signature more than that

Is One More Unforgeability Too Weak?

- adversary can produce n forgeries

Is One More Unforgeability Too Weak?

- adversary can produce n forgeries
- bad if forgeries can be chosen at will

Is One More Unforgeability Too Weak?

- adversary can produce n forgeries
- bad if forgeries can be chosen at will
- good if forgeries are decided after the leakage phase

Is One More Unforgeability Too Weak?

- adversary can produce n forgeries
- bad if forgeries can be chosen at will
- good if forgeries are decided after the leakage phase

assume there exists a simulator

- looking at the state of A after leakage phase and producing a set Q^*

Is One More Unforgeability Too Weak?

- adversary can produce n forgeries
- bad if forgeries can be chosen at will
- good if forgeries are decided after the leakage phase

assume there exists a simulator

- looking at the state of A after leakage phase and producing a set Q^*
- Q^* contains all the messages A can forge

Is One More Unforgeability Too Weak?

- adversary can produce n forgeries
- bad if forgeries can be chosen at will
- good if forgeries are decided after the leakage phase

assume there exists a simulator

- looking at the state of A after leakage phase and producing a set Q^*
- Q^* contains all the messages A can forge

$$\Downarrow$$

then all forgeries are determined by leakage

A Simulation-based Security Notion

Costrained One-more Unforgeability

A Simulation-based Security Notion

Costrained One-more Unforgeability

$Exp^{one-more}_{S,A_1,A_2}(k, \lambda, \gamma)$

1. A_1 is given verification key vk

A Simulation-based Security Notion

Costrained One-more Unforgeability

$Exp^{one-more}_{S,A_1,A_2}(k, \lambda, \gamma)$

1. A_1 is given verification key vk
2. A_1 can query Sign(sk, $\cdot$) and $\mathcal{O}^{(sk)}$

A Simulation-based Security Notion

Costrained One-more Unforgeability

$Exp^{one-more}_{S,A_1,A_2}(k, \lambda, \gamma)$

1. A_1 is given verification key vk
2. A_1 can query Sign(sk, $\cdot$) and $\mathcal{O}^{(sk)}$
3. S is given A_1's state st and outputs Q^* st $|Q^*| \leq \lfloor \lambda / (\gamma |\sigma|) \rfloor$

A Simulation-based Security Notion

Costrained One-more Unforgeability

$Exp^{one-more}_{S,A_1,A_2}(k, \lambda, \gamma)$

1. A_1 is given verification key vk
2. A_1 can query Sign(sk, $\cdot$) and $\mathcal{O}^{(sk)}$
3. S is given A_1's state st and outputs Q^* st $|Q^*| \leq \lfloor \lambda / (\gamma |\sigma|) \rfloor$
4. A_2 is given st and Q^* and outputs (m, σ)

A Simulation-based Security Notion

Costrained One-more Unforgeability

$Exp^{one-more}_{S,A_1,A_2}(k, \lambda, \gamma)$

1. A_1 is given verification key vk
2. A_1 can query Sign(sk, $\cdot$) and $\mathcal{O}^{(sk)}$
3. S is given A_1's state st and outputs Q^* st $|Q^*| \leq \lfloor \lambda/(\gamma|\sigma|) \rfloor$
4. A_2 is given st and Q^* and outputs (m, σ)

Exp outputs 1 $\iff$

- $Ver(m, \sigma) = 1$
- $m \notin Q \cup Q^*$

A Simulation-based Security Notion

Costrained One-more Unforgeability

$Exp^{one-more}_{S,A_1,A_2}(k, \lambda, \gamma)$

1. A_1 is given verification key vk
2. A_1 can query Sign(sk, $\cdot$) and $\mathcal{O}^{(sk)}$
3. S is given A_1's state st and outputs Q^* st $|Q^*| \leq \lfloor \lambda/(\gamma|\sigma|) \rfloor$
4. A_2 is given st and Q^* and outputs (m, σ)

Exp outputs 1 $\iff$

- $Ver(m, \sigma) = 1$
- $m \notin Q \cup Q^*$

- Simulator determines signatures obtained through leakage

Equivalence

Costrained one-more unforgeability is equivalent to One-more unforgeability

Equivalence

Costrained one-more unforgeability is equivalent to One-more unforgeability

Consequences

- forgeries are determined after leakage phase
- A cannot choose to forge on messages at its will
- similar to standard unforgeability with more signing queries from leakage oracle

Tools

Commitment Scheme

commit to a message m using randomness r

Tools

Commitment Scheme

commit to a message m using randomness r

- computationally binding: hard to open a commitment to two different messages

Tools

Commitment Scheme

commit to a message m using randomness r

- computationally binding: hard to open a commitment to two different messages
- statistically hiding: commitment reveals no info about the message

Tools

Commitment Scheme

commit to a message m using randomness r

- computationally binding: hard to open a commitment to two different messages
- statistically hiding: commitment reveals no info about the message
- homomorphic:
  $Commit(m_0, r_0) + Commit(m_1, r_1) = Commit(m_0 + m_1, r_0 + r_1)$
  $Commit(m, r)^c = Commit(c \cdot m, c \cdot r)$

Tools

Commitment Scheme

commit to a message m using randomness r

- computationally binding: hard to open a commitment to two different messages
- statistically hiding: commitment reveals no info about the message
- homomorphic:
  $Commit(m_0, r_0) + Commit(m_1, r_1) = Commit(m_0 + m_1, r_0 + r_1)$
  $Commit(m, r)^c = Commit(c \cdot m, c \cdot r)$

Non Interactive Argument System

proving that x is in a language L using a witness w

Tools

Commitment Scheme

commit to a message m using randomness r

- computationally binding: hard to open a commitment to two different messages
- statistically hiding: commitment reveals no info about the message
- homomorphic:
  $Commit(m_0, r_0) + Commit(m_1, r_1) = Commit(m_0 + m_1, r_0 + r_1)$
  $Commit(m, r)^c = Commit(c \cdot m, c \cdot r)$

Non Interactive Argument System

proving that x is in a language L using a witness w

- zero-knowledge: a simulator with trapdoor can simulate valid proofs

Tools

Commitment Scheme

commit to a message m using randomness r

- computationally binding: hard to open a commitment to two different messages
- statistically hiding: commitment reveals no info about the message
- homomorphic:
  $Commit(m_0, r_0) + Commit(m_1, r_1) = Commit(m_0 + m_1, r_0 + r_1)$
  $Commit(m, r)^c = Commit(c \cdot m, c \cdot r)$

Non Interactive Argument System

proving that x is in a language L using a witness w

- zero-knowledge: a simulator with trapdoor can simulate valid proofs
- extractability: can extract a witness from a valid proof

Generic Construction

Signature Scheme

Gen : sample pk and crs. Choose $a_0, \ldots, a_d, r_0, \ldots, r_d$ in F.

Generic Construction

Signature Scheme

Gen : sample pk and crs. Choose $a_0, \ldots, a_d, r_0, \ldots, r_d$ in F.
Compute $com_i = Commit_{pk}(a_i, r_i)$.

Generic Construction

Signature Scheme

Gen : sample pk and crs. Choose $a_0, \ldots, a_d, r_0, \ldots, r_d$ in F.
Compute $com_i = Commit_{pk}(a_i, r_i)$.
Output $sk = (\mathbf{a}, \mathbf{r})$ and $vk = (crs, pk, \{com_i\}_i)$

Generic Construction

Signature Scheme

Gen : sample pk and crs. Choose $a_0, \ldots, a_d, r_0, \ldots, r_d$ in F.
Compute $com_i = Commit_{pk}(a_i, r_i)$.
Output $sk = (\mathbf{a}, \mathbf{r})$ and $vk = (crs, pk, \{com_i\}_i)$

Sign : let $f(X) = \sum a_i X^i$ and compute $f(m)$.

Generic Construction

Signature Scheme

Gen : sample pk and crs. Choose $a_0, \ldots, a_d, r_0, \ldots, r_d$ in F.
Compute $com_i = Commit_{pk}(a_i, r_i)$.
Output $sk = (\mathbf{a}, \mathbf{r})$ and $vk = (crs, pk, \{com_i\}_i)$

Sign : let $f(X) = \sum a_i X^i$ and compute $f(m)$.
Output a proof π

$$\prod_i com_i^{m^i} = Commit(f(m), \sum_i r_i \cdot m^i)$$

Generic Construction

Signature Scheme

Gen : sample pk and crs. Choose $a_0, \ldots, a_d, r_0, \ldots, r_d$ in F.
Compute $com_i = Commit_{pk}(a_i, r_i)$.
Output $sk = (\mathbf{a}, \mathbf{r})$ and $vk = (crs, pk, \{com_i\}_i)$

Sign : let $f(X) = \sum a_i X^i$ and compute $f(m)$.
Output a proof π

$$\prod_i com_i^{m^i} = Commit(f(m), \sum_i r_i \cdot m^i)$$

Ver : verify proof π

Theorem

Assumptions

- (Setup, Commit) is statistically hiding, computationally binding and homomorphic
- (Init, Prov, Ver) is NI zero-knowledge argument of knowledge

Given the assumptions above, the scheme is one-more unforgeable for

$$\lambda = d \cdot log|F| \text{ and } \gamma = log|F|/|\sigma|$$

Proof Sketch

Lemma

- Consider a challenger committing to $a_0, \ldots, a_d$ using randomness $r_0, \ldots, r_d$

Proof Sketch

Lemma

- Consider a challenger committing to $a_0, \ldots, a_d$ using randomness $r_0, \ldots, r_d$
- Adversary sees the commitment and can leak λ bits from $\mathbf{a}, \mathbf{r}$

Proof Sketch

Lemma

- Consider a challenger committing to $a_0, \ldots, a_d$ using randomness $r_0, \ldots, r_d$
- Adversary sees the commitment and can leak λ bits from $\mathbf{a}, \mathbf{r}$
- Adversary can open t commitments

Proof Sketch

Lemma

- Consider a challenger committing to $a_0, \ldots, a_d$ using randomness $r_0, \ldots, r_d$
- Adversary sees the commitment and can leak λ bits from $\mathbf{a}, \mathbf{r}$
- Adversary can open t commitments
- Adversary wins if it outputs remaining d-t values of $\mathbf{a}$

Proof Sketch

Lemma

- Consider a challenger committing to $a_0, \ldots, a_d$ using randomness $r_0, \ldots, r_d$
- Adversary sees the commitment and can leak λ bits from $\mathbf{a}, \mathbf{r}$
- Adversary can open t commitments
- Adversary wins if it outputs remaining d-t values of $\mathbf{a}$

$$\text{P[A wins]} \leq 2^{\lambda}/|F|^{d-t}$$

Proof Sketch

Lemma

- Consider a challenger committing to $a_0, \ldots, a_d$ using randomness $r_0, \ldots, r_d$
- Adversary sees the commitment and can leak λ bits from $\mathbf{a}, \mathbf{r}$
- Adversary can open t commitments
- Adversary wins if it outputs remaining d-t values of $\mathbf{a}$

$$\text{P[A wins]} \leq 2^{\lambda}/|F|^{d-t}$$

- statistically hiding -> commitments reveals no information about $\mathbf{a}$

Proof Sketch

Lemma

- Consider a challenger committing to $a_0, \ldots, a_d$ using randomness $r_0, \ldots, r_d$
- Adversary sees the commitment and can leak λ bits from $\mathbf{a}, \mathbf{r}$
- Adversary can open t commitments
- Adversary wins if it outputs remaining d-t values of **a**

$$\text{P[A wins]} \leq 2^{\lambda}/|F|^{d-t}$$

- statistically hiding -> commitments reveals no information about **a**
- leaking λ bits decrease min-entropy of λ

Proof Sketch Continued

- $\mathcal{A}$ playing one-more unforgeability experiment

Proof Sketch Continued

- A playing one-more unforgeability experiment
- use simulation trapdoor to simulate proof (zero-knowledge)

Proof Sketch Continued

- A playing one-more unforgeability experiment
- use simulation trapdoor to simulate proof (zero-knowledge)
- if A outputs n valid forgeries (m_i, σ_i) extract witnesses with extraction trapdoor

Proof Sketch Continued

- A playing one-more unforgeability experiment
- use simulation trapdoor to simulate proof (zero-knowledge)
- if A outputs n valid forgeries (m_i, σ_i) extract witnesses with extraction trapdoor
- we have n tuples $(\tilde{m}_j, \tilde{r}_j)$ of openings for $\prod_i com_i^{m_j^i}$

Proof Sketch Continued

- A playing one-more unforgeability experiment
- use simulation trapdoor to simulate proof (zero-knowledge)
- if A outputs n valid forgeries (m_i, σ_i) extract witnesses with extraction trapdoor
- we have n tuples $(\tilde{m}_j, \tilde{r}_j)$ of openings for $\prod_i com_i^{m_j^i}$
- event Bad: for some j we have $\tilde{m}_j \neq f(m_j)$

Proof Sketch Continued

- A playing one-more unforgeability experiment
- use simulation trapdoor to simulate proof (zero-knowledge)
- if A outputs n valid forgeries (m_i, σ_i) extract witnesses with extraction trapdoor
- we have n tuples $(\tilde{m}_j, \tilde{r}_j)$ of openings for $\prod_i com_i^{m_j^i}$
- event Bad: for some j we have $\tilde{m}_j \neq f(m_j)$
- in case Bad we have adversary breaking binding property

Proof Sketch Continued

- A playing one-more unforgeability experiment
- use simulation trapdoor to simulate proof (zero-knowledge)
- if A outputs n valid forgeries (m_i, σ_i) extract witnesses with extraction trapdoor
- we have n tuples $(\tilde{m}_j, \tilde{r}_j)$ of openings for $\prod_i com_i^{m_j^i}$
- event Bad: for some j we have $\tilde{m}_j \neq f(m_j)$
- in case Bad we have adversary breaking binding property
- in case $\overline{Bad}$ we break property from Lemma

Proof Sketch Continued

- A playing one-more unforgeability experiment
- use simulation trapdoor to simulate proof (zero-knowledge)
- if A outputs n valid forgeries (m_i, σ_i) extract witnesses with extraction trapdoor
- we have n tuples $(\tilde{m}_j, \tilde{r}_j)$ of openings for $\prod_i com_i^{m_j^i}$
- event Bad: for some j we have $\tilde{m}_j \neq f(m_j)$
- in case Bad we have adversary breaking binding property
- in case $\overline{Bad}$ we break property from Lemma
- A wins with negligible probability

A Concrete Instantiation

Linear Assumption

for $g, g_1, g_2 \leftarrow G$ and $a, b, c \leftarrow F$

$$\{g, g_1, g_2, g_1^a, g_2^b, g^{a+b}\} \approx \{g, g_1, g_2, g_1^a, g_2^b, g^c\}$$

A Concrete Instantiation

Linear Assumption

for $g, g_1, g_2 \leftarrow G$ and $a, b, c \leftarrow F$

$$\{g, g_1, g_2, g_1^a, g_2^b, g^{a+b}\} \approx \{g, g_1, g_2, g_1^a, g_2^b, g^c\}$$

Pedersen Commitment

- h_1, h_2 in G, $h_2 = h_1^b$

A Concrete Instantiation

Linear Assumption

for $g, g_1, g_2 \leftarrow G$ and $a, b, c \leftarrow F$

$$\{g, g_1, g_2, g_1^a, g_2^b, g^{a+b}\} \approx \{g, g_1, g_2, g_1^a, g_2^b, g^c\}$$

Pedersen Commitment

- h_1, h_2 in G, $h_2 = h_1^b$
- $Commit(x, r) = h_1^x h_2^r = h_1^{x+b \cdot r}$

A Concrete Instantiation

Linear Assumption

for $g, g_1, g_2 \leftarrow G$ and $a, b, c \leftarrow F$

$$\{g, g_1, g_2, g_1^a, g_2^b, g^{a+b}\} \approx \{g, g_1, g_2, g_1^a, g_2^b, g^c\}$$

Pedersen Commitment

- h_1, h_2 in G, $h_2 = h_1^b$
- $Commit(x, r) = h_1^x h_2^r = h_1^{x+b\cdot r}$

Groth Argument of Knowledge

- with Pedersen:
  $\prod_i (com_i^{m^i}) = \prod_i (h_1^{a_i} h_2^{r_i})^{m^i} = \prod_i (h_1^{a_i+b\cdot r_i})^{m^i} = h_1^{f(m)+b\tilde{r}}$
- use Groth NI proof of knowledge for discrete logarithm

A Concrete Instantiation

Linear Assumption

for $g, g_1, g_2 \leftarrow G$ and $a, b, c \leftarrow F$

$$\{g, g_1, g_2, g_1^a, g_2^b, g^{a+b}\} \approx \{g, g_1, g_2, g_1^a, g_2^b, g^c\}$$

Pedersen Commitment

- h_1, h_2 in G, $h_2 = h_1^b$
- $Commit(x, r) = h_1^x h_2^r = h_1^{x+b\cdot r}$

Groth Argument of Knowledge

- with Pedersen:
  $\prod_i (com_i^{m^i}) = \prod_i (h_1^{a_i} h_2^{r_i})^{m^i} = \prod_i (h_1^{a_i+b\cdot r_i})^{m^i} = h_1^{f(m)+b\tilde{r}}$
- use Groth NI proof of knowledge for discrete logarithm

Notice: $|\sigma|$ is independent from $|sk|$

Conclusions

1. two equivalent definitions of unforgeability with leakage

Conclusions

1. two equivalent definitions of unforgeability with leakage
2. capture degradation of security when $\lambda > |\sigma|$

Conclusions

1. two equivalent definitions of unforgeability with leakage
2. capture degradation of security when $\lambda > |\sigma|$
3. show forgeries depends only on the specific leakage

Conclusions

1. two equivalent definitions of unforgeability with leakage
2. capture degradation of security when $\lambda > |\sigma|$
3. show forgeries depends only on the specific leakage
4. general construction for one-more unforgebility

Conclusions

1. two equivalent definitions of unforgeability with leakage
2. capture degradation of security when $\lambda > |\sigma|$
3. show forgeries depends only on the specific leakage
4. general construction for one-more unforgebility
5. application in identification schemes

Conclusions

1. two equivalent definitions of unforgeability with leakage
2. capture degradation of security when $\lambda > |\sigma|$
3. show forgeries depends only on the specific leakage
4. general construction for one-more unforgebility
5. application in identification schemes
6. concrete instantiation under linear assumption where $|\sigma|$ is independent from $|sk|$

Conclusions

1. two equivalent definitions of unforgeability with leakage
2. capture degradation of security when $\lambda > |\sigma|$
3. show forgeries depends only on the specific leakage
4. general construction for one-more unforgebility
5. application in identification schemes
6. concrete instantiation under linear assumption where $|\sigma|$ is independent from $|sk|$