Improved Security for Linearly Homomorphic Signatures: A Generic Framework

David Mandell Freeman

Stanford University, USA

PKC 2012
Darmstadt, Germany
23 May 2012

Problem: Computing on Authenticated Data

Q: How do we *delegate computation* while *ensuring integrity*?

Problem: Computing on Authenticated Data

Q: How do we *delegate computation* while *ensuring integrity*?

Problem: Computing on Authenticated Data

Q: How do we *delegate computation* while *ensuring integrity*?

sk

Untrusted DB

Problem: Computing on Authenticated Data

Q: How do we *delegate computation* while *ensuring integrity*?

sk

$\xrightarrow[\text{grades}]{\text{signed}}$

Untrusted DB

Student	Score	Sig
Adam	91	σ_1
Becky	73	σ_2
$\vdots$	$\vdots$	$\vdots$
Kevin	84	σ_k

σ_1 = signature on
(“grades”, 91, “Adam”)

Problem: Computing on Authenticated Data

Q: How do we *delegate computation* while *ensuring integrity*?

sk

$\xrightarrow[\text{grades}]{\text{signed}}$

Untrusted DB

Student	Score	Sig
Adam	91	σ_1
Becky	73	σ_2
⋮	⋮	⋮
Kevin	84	σ_k

pk

σ_1 = signature on
(“grades”, 91, “Adam”)

Q: How do we *delegate computation* while *ensuring integrity*?

sk

$\xrightarrow[\text{grades}]{\text{signed}}$

Untrusted DB

Student	Score	Sig
Adam	91	σ_1
Becky	73	σ_2
$\vdots$	$\vdots$	$\vdots$
Kevin	84	σ_k

$\xleftarrow{\text{mean?}}$

pk

$\sigma_1 =$ signature on
(“grades”, 91, “Adam”)

Problem: Computing on Authenticated Data

Q: How do we *delegate computation* while *ensuring integrity*?

sk

signed $\longrightarrow$ grades

Untrusted DB

Student	Score	Sig
Adam	91	σ_1
Becky	73	σ_2
⋮	⋮	⋮
Kevin	84	σ_k

mean? $\longleftarrow$

$\longrightarrow$ 87.3, σ

pk

$\sigma_1 =$ signature on ("grades", 91, "Adam")

$\sigma =$ signature on ("grades", 87.3, "mean")

Problem: Computing on Authenticated Data

Q: How do we *delegate computation* while *ensuring integrity*?

sk

signed $\longrightarrow$ grades

Untrusted DB

Student	Score	Sig
Adam	91	σ_1
Becky	73	σ_2
$\vdots$	$\vdots$	$\vdots$
Kevin	84	σ_k

mean? $\longleftarrow$

$\longrightarrow$ 87.3, σ

pk

σ_1 = signature on ("grades", 91, "Adam")

σ = signature on ("grades", 87.3, "mean")

- Validity: σ authenticates 87.3 as the correct mean.

Problem: Computing on Authenticated Data

Q: How do we *delegate computation* while *ensuring integrity*?

sk

signed $\longrightarrow$ grades

Untrusted DB

Student	Score	Sig
Adam	91	σ_1
Becky	73	σ_2
$\vdots$	$\vdots$	$\vdots$
Kevin	84	σ_k

mean? $\longleftarrow$

$\longrightarrow$ 87.3, σ

pk

$\sigma_1 =$ signature on (“grades”, 91, “Adam”)

$\sigma =$ signature on (“grades”, 87.3, “mean”)

- Validity: σ authenticates 87.3 as the correct mean.
- Security: no adversary can authenticate a different mean.

Problem: Computing on Authenticated Data

Q: How do we *delegate computation* while *ensuring integrity*?


sk

signed grades $\longrightarrow$

Untrusted DB

Student	Score	Sig
Adam	91	σ_1
Becky	73	σ_2
$\vdots$	$\vdots$	$\vdots$
Kevin	84	σ_k

$\longleftarrow$ mean?

$\longrightarrow$ 87.3, σ


pk

σ_1 = signature on ("grades", 91, "Adam")

σ = signature on ("grades", 87.3, "mean")

- Validity: σ authenticates 87.3 as the correct mean.
- Security: no adversary can authenticate a different mean.
- Length efficiency: σ is short.

Solution: Homomorphic Signatures

Messages $(m_1, \ldots, m_k)$ grouped together into *files*, identified by a randomly chosen *tag* τ.

- $\mathsf{KeyGen}(n) \to \mathsf{pk}, \mathsf{sk}$
- $\mathsf{Sign}_{\mathsf{sk}}(\tau, m_i, i) \to$ signature σ_i on ith message
- $\mathsf{Eval}_{\mathsf{pk}}(\tau, (\sigma_1, \ldots, \sigma_k), f) \to$ signature σ on $f(m_1, \ldots, m_k)$
- $\mathsf{Verify}_{\mathsf{pk}}(\tau, m, \sigma, f) \to 1$ iff $m = f(m_1, \ldots, m_k)$

Solution: Homomorphic Signatures

Messages $(m_1, \ldots, m_k)$ grouped together into *files*, identified by a randomly chosen *tag* τ.

- KeyGen$(n) \rightarrow \mathrm{pk}, \mathrm{sk}$
- $\mathrm{Sign}_{\mathrm{sk}}(\tau, m_i, i) \rightarrow$ signature σ_i on ith message
- $\mathrm{Eval}_{\mathrm{pk}}(\tau, (\sigma_1, \ldots, \sigma_k), f) \rightarrow$ signature σ on $f(m_1, \ldots, m_k)$
- $\mathrm{Verify}_{\mathrm{pk}}(\tau, m, \sigma, f) \rightarrow 1$ iff $m = f(m_1, \ldots, m_k)$

Security goal: no adversary can authenticate (m', f) for $m' \neq f(m_1, \ldots, m_k)$.

Solution: Homomorphic Signatures

Messages $(m_1, \ldots, m_k)$ grouped together into *files*, identified by a randomly chosen *tag* τ.

- $\text{KeyGen}(n) \to \text{pk}, \text{sk}$
- $\text{Sign}_{\text{sk}}(\tau, m_i, i) \to$ signature σ_i on ith message
- $\text{Eval}_{\text{pk}}(\tau, (\sigma_1, \ldots, \sigma_k), f) \to$ signature σ on $f(m_1, \ldots, m_k)$
- $\text{Verify}_{\text{pk}}(\tau, m, \sigma, f) \to 1$ iff $m = f(m_1, \ldots, m_k)$

Security goal: no adversary can authenticate (m', f) for
$m' \neq f(m_1, \ldots, m_k)$.

Linearly homomorphic signatures:

- messages m_i are vectors in $\mathbb{Z}^n$ or $\mathbb{F}_q^n$
- functions f are linear combinations.
- applications: mean, Fourier transform, regression models, network coding.

Linearly Homomorphic Signatures: State of the Art

Scheme	Built on	Assumption	Vectors in
[BFKW09]	BLS signatures	CDH in bilinear groups	$\mathbb{F}_p^n$ (large p)
[GKKR10]	RSA signatures	RSA	$\mathbb{Z}^n$
[BF11a,b]	GPV signatures	worst-case lattice problems	$\mathbb{F}_p^n$ (small p)

(Orange = random oracle model)

Linearly Homomorphic Signatures: State of the Art

Scheme	Built on	Assumption	Vectors in
[BFKW09]	BLS signatures	CDH in bilinear groups	$\mathbb{F}_p^n$ (large p)
[GKKR10]	RSA signatures	RSA	$\mathbb{Z}^n$
[BF11a,b]	GPV signatures	worst-case lattice problems	$\mathbb{F}_p^n$ (small p)
[AL11]	Lewko-Waters IBE	nonstandard, decisional assumptions in bilinear groups	$\mathbb{Z}_N^n$
[CFW11]	"adaptive pseudo-free groups"	strong RSA	$\mathbb{Z}^n$
[CFW12]		q-SDH in bilinear groups; strong RSA	$\mathbb{F}_p^n$; $\mathbb{F}_e^n$

(Orange = random oracle model)

Linearly Homomorphic Signatures: State of the Art

Scheme	Built on	Assumption	Vectors in
[BFKW09]	BLS signatures	CDH in bilinear groups	$\mathbb{F}_p^n$ (large p)
[GKKR10]	RSA signatures	RSA	$\mathbb{Z}^n$
[BF11a,b]	GPV signatures	worst-case lattice problems	$\mathbb{F}_p^n$ (small p)
[AL11]	Lewko-Waters IBE	nonstandard, decisional assumptions in bilinear groups	$\mathbb{Z}_N^n$
[CFW11]	"adaptive pseudo-free groups"	strong RSA	$\mathbb{Z}^n$
[CFW12]		q-SDH in bilinear groups; strong RSA	$\mathbb{F}_p^n$; $\mathbb{F}_e^n$

(Orange = random oracle model)

Missing: Weak assumptions in the standard model!

Our Contribution (1)

Generic framework for converting (ordinary) signatures to linearly homomorphic signatures.

- Applies to signature schemes with certain "pre-homomorphic" properties.
- Security based on same assumption as underlying scheme.
- Efficiency comparable to previous constructions.

Our Contribution (1)

Generic framework for converting (ordinary) signatures to linearly homomorphic signatures.

- Applies to signature schemes with certain "pre-homomorphic" properties.
- Security based on same assumption as underlying scheme.
- Efficiency comparable to previous constructions.

Instantiations:

Scheme	Assumption (in standard model)
[W05]	CDH in bilinear groups
[BB04b]	q-SDH in bilinear groups
[GHR99]	strong RSA
[HW09b]	RSA

Our Contrubution (2)

Stronger security model for homomorphic signatures.

Our Contrubution (2)

Stronger security model for homomorphic signatures.

Chall.

sk

Adversary

Our Contrubution (2)

Stronger security model for homomorphic signatures.

Chall. $\xrightarrow{\quad pk \quad}$ Adversary

 *pk*

 sk

Our Contrubution (2)

Stronger security model for homomorphic signatures.

Chall.

pk $\longrightarrow$

Adversary

$\longleftarrow$ file $F = \{m_1, \ldots, m_k\}$

sk

Our Contrubution (2)

Stronger security model for homomorphic signatures.

Chall.

sk

pk

$\longrightarrow$

file $F = \{m_1, \ldots, m_k\}$

$\longleftarrow$

tag τ_F, sigs $\sigma_1^F, \ldots, \sigma_k^F$

$\longrightarrow$

Adversary

Our Contrubution (2)

Stronger security model for homomorphic signatures.

pk

Chall.

sk

Adversary

$\xrightarrow{\hspace{6em}}$

$\xleftarrow{\text{file } F = \{m_1, \ldots, m_k\}}$

$\xrightarrow{\text{tag } \tau_F, \text{ sigs } \sigma_1^F, \ldots, \sigma_k^F}$

$\Big\}$ repeat

Our Contrubution (2)

Stronger security model for homomorphic signatures.

Chall.  pk $\xrightarrow{\hspace{6em}}$ Adversary

$\xleftarrow{\text{file } F = \{m_1, \ldots, m_k\}}$

$\xrightarrow{\text{tag } \tau_F, \text{ sigs } \sigma_1^F, \ldots, \sigma_k^F}$

$\Bigg\}$ repeat

 sk

$\xleftarrow{\text{forgery } \tau^*, m^*, \sigma^*, f}$

Our Contrubution (2)

Stronger security model for homomorphic signatures.

Chall.

sk

$\xrightarrow{\quad pk \quad}$

$\xleftarrow{\text{file } F = \{m_1, \ldots, m_k\}}$

$\xrightarrow{\text{tag } \tau_F,\ \text{sigs } \sigma_1^F, \ldots, \sigma_k^F}$

repeat

$\xleftarrow{\text{forgery } \tau^*, m^*, \sigma^*, f}$

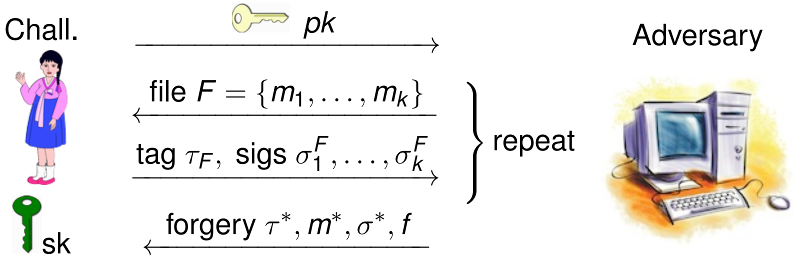
Adversary

Forgery is a valid signature σ^* on (τ^*, m^*, f) with

$$m^* \neq f(\text{messages in file w/ tag } \tau^*).$$

Our Contrubution (2)

Stronger security model for homomorphic signatures.



Forgery is a valid signature σ^* on (τ^*, m^*, f) with

$$m^* \neq f(\text{messages in file w/ tag } \tau^*).$$

- Original adversary: must query entire files at once.

Our Contrubution (2)

Stronger security model for homomorphic signatures.

Chall.

 pk $\longrightarrow$

Adversary

$\xleftarrow{\text{file } F = \{m_1, \ldots, m_k\}}$

$\xrightarrow{\text{tag } \tau_F,\ \text{sigs } \sigma_1^F, \ldots, \sigma_k^F}$

} repeat

 sk

$\xleftarrow{\text{forgery } \tau^*, m^*, \sigma^*, f}$

Forgery is a valid signature σ^* on (τ^*, m^*, f) with

$$m^* \neq f(\text{messages in file w/ tag } \tau^*).$$

- Original adversary: must query entire files at once.
- Stronger adversary: adaptively queries *one message at a time* from any file.

Our Contrubution (2)

Stronger security model for homomorphic signatures.

Chall.

 sk

 pk $\longrightarrow$

tuple (F, i, m_i) $\longleftarrow$

tag τ_F, sig σ_i^F $\longrightarrow$

repeat

forgery τ^*, m^*, σ^*, f $\longleftarrow$

Adversary

Forgery is a valid signature σ^* on (τ^*, m^*, f) with

$$m^* \neq f(\text{messages in file w/ tag } \tau^*).$$

- Original adversary: must query entire files at once.
- Stronger adversary: adaptively queries *one message at a time* from any file.

Our Contrubution (2)

Stronger security model for homomorphic signatures.

Chall.

sk

pk $\longrightarrow$

$\longleftarrow$ tuple (F, i, m_i)

tag τ_F, sig σ_i^F $\longrightarrow$

repeat

$\longleftarrow$ forgery τ^*, m^*, σ^*, f

Adversary

Forgery is a valid signature σ^* on (τ^*, m^*, f) with

$$m^* \neq f(\text{messages in file w/ tag } \tau^*).$$

- Original adversary: must query entire files at once.
- Stronger adversary: adaptively queries *one message at a time* from any file.

Our schemes are secure against the stronger adversary.

Previous RSA Construction [GKKR10]

Homomorphic hash: fix public $h_1, \ldots, h_n \in \mathbb{Z}_N^*$;
for vector $\mathbf{v} \in \mathbb{Z}^n$, define

$$H_{\text{hom}}(\mathbf{v}) = h_1^{v_1} \cdots h_n^{v_n}$$

.

Previous RSA Construction [GKKR10]

Homomorphic hash: fix public $h_1, \ldots, h_n \in \mathbb{Z}_N^*$;
for vector $\mathbf{v} \in \mathbb{Z}^n$, define

$$H_{\text{hom}}(\mathbf{v}) = h_1^{v_1} \cdots h_n^{v_n}$$

Signatures: to sign ith vector $\mathbf{v}_i$, compute:

$$\sigma = (t_i \cdot H_{\text{hom}}(\mathbf{v}_i))^{1/e} \bmod N \qquad (t_i \text{ public}).$$

Previous RSA Construction [GKKR10]

Homomorphic hash: fix public $h_1, \ldots, h_n \in \mathbb{Z}_N^*$;
for vector $\mathbf{v} \in \mathbb{Z}^n$, define

$$H_{\text{hom}}(\mathbf{v}) = h_1^{v_1} \cdots h_n^{v_n}$$

Signatures: to sign ith vector $\mathbf{v}_i$, compute:

$$\sigma = (t_i \cdot H_{\text{hom}}(\mathbf{v}_i))^{1/e} \bmod N \qquad (t_i \text{ public}).$$

Homomorphic: If σ_1, σ_2 are signatures on $\mathbf{v}_1, \mathbf{v}_2$, then

$$\begin{aligned} \sigma_1 \cdot \sigma_2 &= \left(t_1 \cdot H_{\text{hom}}(\mathbf{v}_1) \cdot t_2 \cdot H_{\text{hom}}(\mathbf{v}_2)\right)^{1/e} \\ &= (t_1 t_2 \cdot H_{\text{hom}}(\mathbf{v}_1 + \mathbf{v}_2))^{1/e} \end{aligned}$$

authenticates $\mathbf{v}_1 + \mathbf{v}_2$ for the function $f(x, y) = x + y$.

Previous RSA Construction [GKKR10]

Homomorphic hash: fix public $h_1, \ldots, h_n \in \mathbb{Z}_N^*$;
for vector $\mathbf{v} \in \mathbb{Z}^n$, define

$$H_{\text{hom}}(\mathbf{v}) = h_1^{v_1} \cdots h_n^{v_n}$$

Signatures: to sign ith vector $\mathbf{v}_i$, compute:

$$\sigma = (t_i \cdot H_{\text{hom}}(\mathbf{v}_i))^{1/e} \bmod N \qquad (t_i \text{ public}).$$

Homomorphic: If σ_1, σ_2 are signatures on $\mathbf{v}_1, \mathbf{v}_2$, then

$$\begin{aligned} \sigma_1 \cdot \sigma_2 &= \left(t_1 \cdot H_{\text{hom}}(\mathbf{v}_1) \cdot t_2 \cdot H_{\text{hom}}(\mathbf{v}_2)\right)^{1/e} \\ &= (t_1 t_2 \cdot H_{\text{hom}}(\mathbf{v}_1 + \mathbf{v}_2))^{1/e} \end{aligned}$$

authenticates $\mathbf{v}_1 + \mathbf{v}_2$ for the function $f(x, y) = x + y$.

- t_i must be different for each file to prevent mixing.
- Secure if $t_i = R(i, \tau)$ produced by a random oracle.

Removing the Random Oracle

Instead of RSA sigs, use [GHR99]:

$$\text{Sign}(m) = g^{1/H(m)} \bmod N.$$

- g public, H hashes to odd primes.
- secure in standard model under *strong RSA assumption*:
  - Given (g, N), find any $(e, g^{1/e} \bmod N)$.

Removing the Random Oracle

Instead of RSA sigs, use [GHR99]:

$$\text{Sign}(m) = g^{1/H(m)} \bmod N.$$

- g public, H hashes to odd primes.
- secure in standard model under *strong RSA assumption*:
  - Given (g, N), find any $(e, g^{1/e} \bmod N)$.

Our idea: to sign ith vector $\mathbf{v}_i$ for file τ, compute:

$$\sigma = \Big(\underbrace{g^{1/H(\tau)}}_{\sigma_1}, \underbrace{(t_i \cdot H_{\text{hom}}(\mathbf{v}))^{1/H(\tau)}}_{\sigma_2}\Big) \qquad (t_i \text{ public}).$$

Removing the Random Oracle

Instead of RSA sigs, use [GHR99]:

$$\text{Sign}(m) = g^{1/H(m)} \bmod N.$$

- g public, H hashes to odd primes.
- secure in standard model under *strong RSA assumption*:
  - Given (g, N), find any $(e, g^{1/e} \bmod N)$.

Our idea: to sign ith vector $\mathbf{v}_i$ for file τ, compute:

$$\sigma = \Big(\underbrace{g^{1/H(\tau)}}_{\sigma_1}, \underbrace{(t_i \cdot H_{\text{hom}}(\mathbf{v}))^{1/H(\tau)}}_{\sigma_2}\Big) \qquad (t_i \text{ public}).$$

To verify (σ_1, σ_2) on vector $\mathbf{w}$ for function $f(\vec{x}) = \sum c_i x_i$:

1. Check that $\sigma_1^{H(\tau)} = g$.
2. Check that $\sigma_2^{H(\tau)} = \prod t_i^{c_i} \cdot H_{\text{hom}}(\mathbf{w})$

Homomorphic Property

$$\text{Sign}(\tau, \mathbf{v}_i) \to \Big(\underbrace{g^{1/H(\tau)}}_{\sigma_1}, \underbrace{(t_i \cdot H_{\text{hom}}(\mathbf{v}))^{1/H(\tau)}}_{\sigma_2} \Big).$$

$\text{Verify}(\tau, \mathbf{w}, (\sigma_1, \sigma_2), f)$ with $f(\vec{x}) = \sum c_i x_i$:

$$\sigma_1^{H(\tau)} \stackrel{?}{=} g, \qquad \sigma_2^{H(\tau)} \stackrel{?}{=} \prod t_i^{c_i} \cdot H_{\text{hom}}(\mathbf{w}).$$

Homomorphic Property

$$\text{Sign}(\tau, \mathbf{v}_i) \to \Big(\underbrace{g^{1/H(\tau)}}_{\sigma_1}, \underbrace{(t_i \cdot H_{\text{hom}}(\mathbf{v}))^{1/H(\tau)}}_{\sigma_2}\Big).$$

Verify$(\tau, \mathbf{w}, (\sigma_1, \sigma_2), f)$ with $f(\vec{x}) = \sum c_i x_i$:

$$\sigma_1^{H(\tau)} \stackrel{?}{=} g, \qquad \sigma_2^{H(\tau)} \stackrel{?}{=} \prod t_i^{c_i} \cdot H_{\text{hom}}(\mathbf{w}).$$

Homomorphic: If $(\sigma_1, \sigma_2), (\sigma_1, \sigma_2')$ are signatures on $\mathbf{v}_1, \mathbf{v}_2$, then

$$\sigma_2 \cdot \sigma_2' = (t_1 t_2 \cdot H_{\text{hom}}(\mathbf{v}_1 + \mathbf{v}_2))^{1/H(\tau)}$$

so $(\sigma_1, \sigma_2 \cdot \sigma_2')$ authenticates $\mathbf{v}_1 + \mathbf{v}_2$ for $f(x, y) = x + y$.

Security

Forgery is a valid signature σ^* on (τ^*, m^*, f) with

$$m^* \neq f(\text{messages in file w/ tag } \tau^*).$$

Two types:

1. τ^* not obtained in response to a query, or
2. $\tau^* = \tau$ for query $(m_1, \ldots, m_k)$, and $m^* \neq f(m_1, \ldots, m_k)$.

Security

Forgery is a valid signature σ^* on (τ^*, m^*, f) with

$$m^* \neq f(\text{messages in file w/ tag } \tau^*).$$

Two types:

1. τ^* not obtained in response to a query, or
2. $\tau^* = \tau$ for query $(m_1, \ldots, m_k)$, and $m^* \neq f(m_1, \ldots, m_k)$.

Type 1 forgery breaks underlying GHR scheme:

- computes $g^{1/H(\tau^*)}$ for previously unseen τ^*.

Security

Forgery is a valid signature σ^* on (τ^*, m^*, f) with

$$m^* \neq f(\text{messages in file w/ tag } \tau^*).$$

Two types:

1. τ^* not obtained in response to a query, or
2. $\tau^* = \tau$ for query $(m_1, \ldots, m_k)$, and $m^* \neq f(m_1, \ldots, m_k)$.

Type 1 forgery breaks underlying GHR scheme:

- computes $g^{1/H(\tau^*)}$ for previously unseen τ^*.

Type 2 forgery breaks an RSA assumption:

- strong RSA if H is [GHR99] hash function.
- RSA if H is a random oracle.
- RSA if H is [HW09b] hash function.

Proof Sketch for Type 2 Forgery

Consider a *weak* adversary that submits files $F_\ell = \{\mathbf{v}_1^\ell, \ldots, \mathbf{v}_k^\ell\}$ for $\ell = 1, \ldots, q$ and receives pk, tags τ_ℓ, and signatures

$$\sigma = \left(g^{1/H(\tau)}, (t_i \cdot H_{\mathrm{hom}}(\mathbf{v}))^{1/H(\tau)}\right).$$

Proof Sketch for Type 2 Forgery

Consider a *weak* adversary that submits files $F_\ell = \{\mathbf{v}_1^\ell, \ldots, \mathbf{v}_k^\ell\}$ for $\ell = 1, \ldots, q$ and receives pk, tags τ_ℓ, and signatures

$$\sigma = \left(g^{1/H(\tau)}, (t_i \cdot H_{\text{hom}}(\mathbf{v}))^{1/H(\tau)} \right).$$

1. Given RSA challenge g, choose $\ell^* \stackrel{R}{\leftarrow} \{1, \ldots, q\}$ and set

$$x = g^{\prod_\ell H(\tau_\ell)}, \qquad y = g^{\prod_{\ell \neq \ell^*} H(\tau_\ell)}.$$

   - Simulator can compute $x^{1/H(\tau_\ell)}$ for all ℓ.
   - Simulator can compute $y^{1/H(\tau_\ell)}$ for all $\ell \neq \ell^*$
   - $y^{1/H(\tau_{\ell*})}$ can be used to solve RSA problem (w.h.p).

Proof Sketch for Type 2 Forgery

Consider a *weak* adversary that submits files $F_\ell = \{\mathbf{v}_1^\ell, \ldots, \mathbf{v}_k^\ell\}$ for $\ell = 1, \ldots, q$ and receives pk, tags τ_ℓ, and signatures

$$\sigma = \left(g^{1/H(\tau)}, (t_i \cdot H_{\text{hom}}(\mathbf{v}))^{1/H(\tau)}\right).$$

1. Given RSA challenge g, choose $\ell^* \xleftarrow{R} \{1, \ldots, q\}$ and set

$$x = g^{\prod_\ell H(\tau_\ell)}, \qquad y = g^{\prod_{\ell \neq \ell^*} H(\tau_\ell)}.$$

   - Simulator can compute $x^{1/H(\tau_\ell)}$ for all ℓ.
   - Simulator can compute $y^{1/H(\tau_\ell)}$ for all $\ell \neq \ell^*$
   - $y^{1/H(\tau_{\ell^*})}$ can be used to solve RSA problem (w.h.p).

2. Construct public key so that $t_i \cdot H_{\text{hom}}(\mathbf{v}_i^\ell) = x^{a_{\ell,i}} y^{b_{\ell,i}}$ for $a_{\ell,i}, b_{\ell,i}$ known to simulator, and $b_{\ell,i} = 0$ for $\ell = \ell^*$.

   - Can sign all queried vectors $\mathbf{v}$.
   - Forgery on ℓ^*th file contains a y term $\Rightarrow$ solve RSA.

Proof Sketch for Type 2 Forgery

Consider a *weak* adversary that submits files $F_\ell = \{\mathbf{v}_1^\ell, \ldots, \mathbf{v}_k^\ell\}$ for $\ell = 1, \ldots, q$ and receives pk, tags τ_ℓ, and signatures

$$\sigma = \left(g^{1/H(\tau)}, (t_i \cdot H_{\text{hom}}(\mathbf{v}))^{1/H(\tau)} \right).$$

1. Given RSA challenge g, choose $\ell^* \xleftarrow{R} \{1, \ldots, q\}$ and set

$$x = g^{\prod_\ell H(\tau_\ell)}, \qquad y = g^{\prod_{\ell \neq \ell^*} H(\tau_\ell)}.$$

   - Simulator can compute $x^{1/H(\tau_\ell)}$ for all ℓ.
   - Simulator can compute $y^{1/H(\tau_\ell)}$ for all $\ell \neq \ell^*$
   - $y^{1/H(\tau_{\ell^*})}$ can be used to solve RSA problem (w.h.p).

2. Construct public key so that $t_i \cdot H_{\text{hom}}(\mathbf{v}_i^\ell) = x^{a_{\ell,i}} y^{b_{\ell,i}}$ for $a_{\ell,i}, b_{\ell,i}$ known to simulator, and $b_{\ell,i} = 0$ for $\ell = \ell^*$.
   - Can sign all queried vectors $\mathbf{v}$.
   - Forgery on ℓ^*th file contains a y term $\Rightarrow$ solve RSA.

3. Generalize using *homomorphic chameleon hash*.

Generalization to Other Signatures

Construction works for any signatures of the form

$$\mathsf{Sign}(m) = \left(g^{f(\mathsf{sk}, m, r)}, \sigma_2\right)$$

where g generates some group $\mathbb{G}$ and r is random.

Generalization to Other Signatures

Construction works for any signatures of the form

$$\text{Sign}(m) = \left(g^{f(\text{sk}, m, r)}, \sigma_2\right)$$

where g generates some group $\mathbb{G}$ and r is random. E.g.:

1. [GS02/BB04a/W05]: $\text{sk} = \alpha$, $\text{Sign}(m) = (g^r, g^\alpha H(m)^r)$
   - Secure under CDH assumption in bilinear group $\mathbb{G}$ if H is random oracle or Waters hash function.

Generalization to Other Signatures

Construction works for any signatures of the form

$$\mathsf{Sign}(m) = \left(g^{f(\mathsf{sk},m,r)}, \sigma_2\right)$$

where g generates some group $\mathbb{G}$ and r is random. E.g.:

1. [GS02/BB04a/W05]: $\mathsf{sk} = \alpha$, $\mathsf{Sign}(m) = (g^r, g^\alpha H(m)^r)$
   - Secure under CDH assumption in bilinear group $\mathbb{G}$ if H is random oracle or Waters hash function.
2. [BB04b]: $\mathsf{sk} = \alpha$, $\mathsf{Sign}(m) = g^{1/(m+\alpha)}$.
   - Secure under q-SDH assumption in bilinear group $\mathbb{G}$.

Generalization to Other Signatures

Construction works for any signatures of the form

$$\mathsf{Sign}(m) = \left(g^{f(\mathsf{sk},m,r)}, \sigma_2\right)$$

where g generates some group $\mathbb{G}$ and r is random. E.g.:

1. [GS02/BB04a/W05]: $\mathsf{sk} = \alpha$, $\mathsf{Sign}(m) = (g^r, g^\alpha H(m)^r)$
   - Secure under CDH assumption in bilinear group $\mathbb{G}$ if H is random oracle or Waters hash function.
2. [BB04b]: $\mathsf{sk} = \alpha$, $\mathsf{Sign}(m) = g^{1/(m+\alpha)}$.
   - Secure under q-SDH assumption in bilinear group $\mathbb{G}$.

Generalized homomorphic signature on ith vector $\mathbf{v}_i$ for file τ is

$$\sigma = \left(g^{f(\mathsf{sk},\tau,r)}, \sigma_2, (t_i \cdot H_{\mathsf{hom}}(\mathbf{v}_i))^{f(\mathsf{sk},\tau,r)}\right).$$

For details see full version (IACR eprint 2012/060).

The Big Picture

Comparison with [CFW12] (previous talk):

- Theirs are more efficient (no σ_1 component).
- Ours are more general (can use CDH assumption).

The Big Picture

Comparison with [CFW12] (previous talk):

- Theirs are more efficient (no σ_1 component).
- Ours are more general (can use CDH assumption).

Open questions:

1. Improve efficiency.
2. Strengthen adversary — allow adv. to choose tags?
3. Adapt to lattice setting [BF11a,b] — polynomial functions?

The Big Picture

Comparison with [CFW12] (previous talk):

- Theirs are more efficient (no σ_1 component).
- Ours are more general (can use CDH assumption).

Open questions:

1. Improve efficiency.
2. Strengthen adversary — allow adv. to choose tags?
3. Adapt to lattice setting [BF11a,b] — polynomial functions?

Thank you!