Homomorphic Signatures for Polynomial Functions

Dan Boneh and David Mandell Freeman

Stanford University, USA

Eurocrypt 2011
Tallinn, Estonia

16 May 2011

Homomorphic Signatures

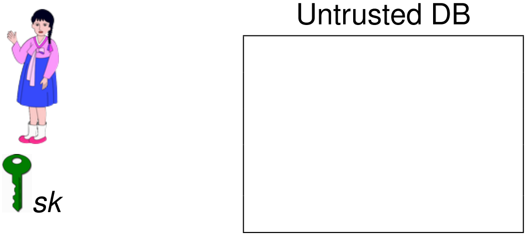
Homomorphic encryption allows users to delegate computation while ensuring *secrecy*.

Homomorphic Signatures

Homomorphic signatures allow users to delegate computation while ensuring *integrity*.

Homomorphic Signatures

Homomorphic signatures allow users to delegate computation while ensuring *integrity*.



Homomorphic Signatures

Homomorphic signatures allow users to delegate computation while ensuring *integrity*.

sk

signed → grades

Untrusted DB

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

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

Homomorphic Signatures

Homomorphic signatures allow users to delegate computation while ensuring *integrity*.

sk

signed → grades

Untrusted DB

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

pk

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

Homomorphic Signatures

Homomorphic signatures allow users to delegate computation while ensuring *integrity*.

sk

signed grades →

Untrusted DB

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

← mean?

pk

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

Homomorphic Signatures

Homomorphic signatures allow users to delegate computation while ensuring *integrity*.

sk

signed → grades

Untrusted DB

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

← mean?

→ 87.3, σ

pk

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

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

Properties of homomorphic signatures

What properties do we want the derived signature σ to have?

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

Properties of homomorphic signatures

What properties do we want the derived signature σ to have?

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

1. Validity: σ authenticates 87.3 as the mean, **and** that the mean was computed correctly.

Properties of homomorphic signatures

What properties do we want the derived signature σ to have?

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

1. Validity: σ authenticates 87.3 as the mean, **and** that the mean was computed correctly.
2. Unforgeability: no adversary can produce a σ^* that authenticates a different mean for the "grades" data.

Properties of homomorphic signatures

What properties do we want the derived signature σ to have?

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

1. Validity: σ authenticates 87.3 as the mean, **and** that the mean was computed correctly.
2. Unforgeability: no adversary can produce a σ^* that authenticates a different mean for the “grades” data.
3. Length efficiency: σ is short.

Properties of homomorphic signatures

What properties do we want the derived signature σ to have?

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

1. Validity: σ authenticates 87.3 as the mean, **and** that the mean was computed correctly.
2. Unforgeability: no adversary can produce a σ^* that authenticates a different mean for the "grades" data.
3. Length efficiency: σ is short.
4. Privacy: σ reveals nothing about data other than the mean.

More generally: $\mathcal{F}$-homomorphic signatures

As introduced by [JMSW02]:

- $\mathcal{F}$ is a set of “admissible” functions on messages.
- τ is the name of the file or data set
  (prevents mixing of data from different sets)
- Given *pk*, admissible function $f \in \mathcal{F}$, and signatures on

  $$(\tau, m_1, 1), \ \ldots, \ (\tau, m_k, k)$$

  anyone can compute a valid signature on

  $$(\tau, \ f(m_1, \ldots, m_k)),$$

More generally: $\mathcal{F}$-homomorphic signatures

As introduced by [JMSW02]:

- $\mathcal{F}$ is a set of "admissible" functions on messages.
- τ is the name of the file or data set
  (prevents mixing of data from different sets)
- Given *pk*, admissible function $f \in \mathcal{F}$, and signatures on

$$(\tau, m_1, 1), \ \ldots, \ (\tau, m_k, k)$$

  anyone can compute a valid signature on

$$(\tau, \ f(m_1, \ldots, m_k)),$$

Observation [JMSW02]

Secure homomorphic signatures for $\mathcal{F} = \{\text{linear functions}\}$ cannot exist.

More generally: $\mathcal{F}$-homomorphic signatures

Our modification: authenticate the function.

- $\mathcal{F}$ is a set of "admissible" functions on messages.
- τ is the name of the file or data set
  (prevents mixing of data from different sets)
- Given pk, admissible function $f \in \mathcal{F}$, and signatures on

$$(\tau, m_1, 1), \ \ldots, \ (\tau, m_k, k)$$

anyone can compute a valid signature on

$$(\tau, \ f(m_1, \ldots, m_k), \ \omega(f)),$$

where $\omega(f)$ is an "encoding" or "digest" of the function f.

More generally: $\mathcal{F}$-homomorphic signatures

Our modification: authenticate the function.

- $\mathcal{F}$ is a set of "admissible" functions on messages.
- τ is the name of the file or data set
  (prevents mixing of data from different sets)
- Given pk, admissible function $f \in \mathcal{F}$, and signatures on

$$(\tau, m_1, 1), \ \ldots, \ (\tau, m_k, k)$$

anyone can compute a valid signature on

$$(\tau, \ f(m_1, \ldots, m_k), \ \omega(f)),$$

where $\omega(f)$ is an "encoding" or "digest" of the function f.

Theorem [BFKW09,GKKR10,BF11]

Secure homomorphic signatures for $\mathcal{F} = \{\text{linear functions}\}$ do exist (under certain assumptions).

Applications

What are homomorphic signatures good for?

$\mathcal{F}$	Application
Linear functions	Mean Fourier transform Network coding

Applications

What are homomorphic signatures good for?

$\mathcal{F}$	Application
Linear functions	Mean Fourier transform Network coding
Subsets	Message redaction

Applications

What are homomorphic signatures good for?

$\mathcal{F}$	Application
Linear functions	Mean Fourier transform Network coding
Subsets	Message redaction
Polynomials (bounded degree)	Standard deviation & higher moments Linear least-squares fit

Applications

What are homomorphic signatures good for?

$\mathcal{F}$	Application
Linear functions	Mean Fourier transform Network coding
Subsets	Message redaction
Polynomials (bounded degree)	Standard deviation & higher moments Linear least-squares fit
Arbitrary circuits	Non-linear estimators and regression Data mining (decision trees, SVM, etc.)

State of the art

How can we compute on encrypted or authenticated data?

$\mathcal{F}$	Hom. encryption	Hom. signatures
Linear functions		
Subsets		
Polynomials (bounded degree)		
Arbitrary circuits		

State of the art

How can we compute on encrypted or authenticated data?

$\mathcal{F}$	Hom. encryption	Hom. signatures
Linear functions	[GM82], [B88], [P99], others	
Subsets		
Polynomials (bounded degree)	[BGN05], [GHV10] (quadratic)	
Arbitrary circuits	[G09], [DGHV10], [BV11]	

State of the art

How can we compute on encrypted or authenticated data?

$\mathcal{F}$	Hom. encryption	Hom. signatures
Linear functions	[GM82], [B88], [P99], others	[KFM04], [CJL06], [ZKMH07], [BFKW09], [GKKR10], [BF11]
Subsets		[JMSW02], others
Polynomials (bounded degree)	[BGN05], [GHV10] (quadratic)	
Arbitrary circuits	[G09], [DGHV10], [BV11]	

State of the art

How can we compute on encrypted or authenticated data?

$\mathcal{F}$	Hom. encryption	Hom. signatures
Linear functions	[GM82], [B88], [P99], others	[KFM04], [CJL06], [ZKMH07], [BFKW09], [GKKR10], [BF11]
Subsets		[JMSW02], others
Polynomials (bounded degree)	[BGN05], [GHV10] (quadratic)	This work
Arbitrary circuits	[G09], [DGHV10], [BV11]	

State of the art

How can we compute on encrypted or authenticated data?

$\mathcal{F}$	Hom. encryption	Hom. signatures
Linear functions	[GM82], [B88], [P99], others	[KFM04], [CJL06], [ZKMH07], [BFKW09], [GKKR10], [BF11]
Subsets		[JMSW02], others
Polynomials (bounded degree)	[BGN05], [GHV10] (quadratic)	This work
Arbitrary circuits	[G09], [DGHV10], [BV11]	

Specifically, we construct secure, length-efficient, $\mathcal{F}$-homomorphic signatures for

$$\mathcal{F} = \{\text{polynomials of bounded degree with small coefficients}\}$$

Related Concepts

Computationally Sound Proofs [M00]:

Server computes a short proof of knowledge that for given (f, y)

$$\exists\ (\vec{m}, \sigma) \text{ s.t. } \begin{cases} y = f(\vec{m}) \quad \text{and} \\ \text{Verify}(pk, \vec{m}, \sigma) = 1. \end{cases}$$

Related Concepts

Computationally Sound Proofs [M00]:

Server computes a short proof of knowledge that for given (f, y)

$$\exists\ (\vec{m}, \sigma) \text{ s.t. } \begin{cases} y = f(\vec{m}) \quad \text{and} \\ \text{Verify}(pk, \vec{m}, \sigma) = 1. \end{cases}$$

- Inefficient (requires PCP theorem).
- Hard to compose functions [V07].
- Need random oracle or non-falsifiable assumption [GW11].

Related Concepts

Computationally Sound Proofs [M00]:

Server computes a short proof of knowledge that for given (f, y)

$$\exists\ (\vec{m}, \sigma) \text{ s.t. } \begin{cases} y = f(\vec{m}) \quad \text{and} \\ \text{Verify}(pk, \vec{m}, \sigma) = 1. \end{cases}$$

- Inefficient (requires PCP theorem).
- Hard to compose functions [V07].
- Need random oracle or non-falsifiable assumption [GW11].

Verifiable computation [GKR08,GGP10,CKV10,AIK10]:

Alice outsources computation to server, uses secret key to verify certificate that computation was done correctly.

Related Concepts

Computationally Sound Proofs [M00]:

Server computes a short proof of knowledge that for given (f, y)

$$\exists\ (\vec{m}, \sigma) \text{ s.t. } \begin{cases} y = f(\vec{m}) \quad \text{and} \\ \text{Verify}(pk, \vec{m}, \sigma) = 1. \end{cases}$$

- Inefficient (requires PCP theorem).
- Hard to compose functions [V07].
- Need random oracle or non-falsifiable assumption [GW11].

Verifiable computation [GKR08,GGP10,CKV10,AIK10]:

Alice outsources computation to server, uses secret key to verify certificate that computation was done correctly.

- Homomorphic signatures allow third party verification.

Application: Least Squares Fits

Least squares fits — the basics

For a data set $\{(x_i, y_i)\}_{i=1}^{k}$, the degree d least squares fit is a polynomial

$$f(x) = c_0 + c_1 x + \cdots + c_d x^d$$

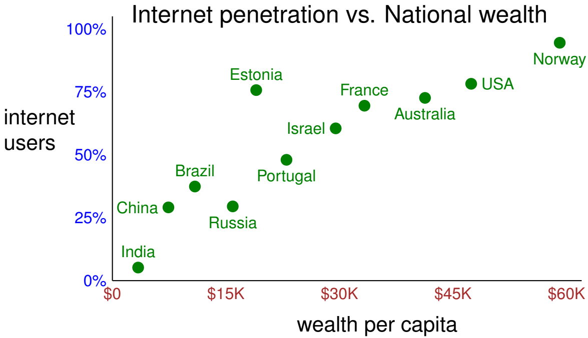
that "best" approximates the y values.

Least squares fits — the basics

For a data set $\{(x_i, y_i)\}_{i=1}^{k}$, the degree d least squares fit is a polynomial

$$f(x) = c_0 + c_1 x + \cdots + c_d x^d$$

that "best" approximates the y values.

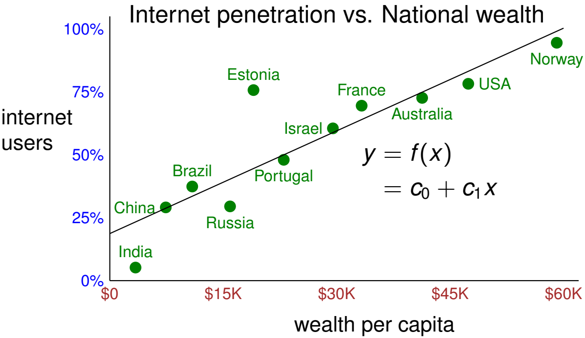


Least squares fits — the basics

For a data set $\{(x_i, y_i)\}_{i=1}^{k}$, the degree d least squares fit is a polynomial

$$f(x) = c_0 + c_1 x + \cdots + c_d x^d$$

that "best" approximates the y values.

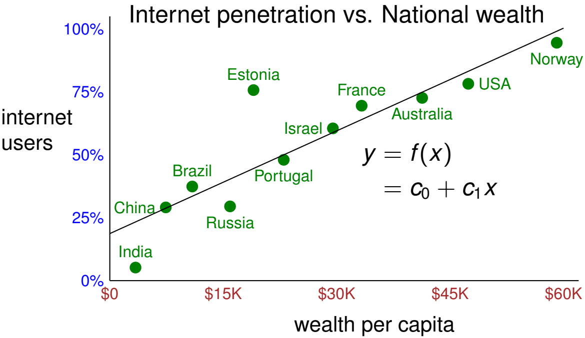


Least squares fits — the basics

For a data set $\{(x_i, y_i)\}_{i=1}^k$, the **degree d least squares fit** is a polynomial

$$f(x) = c_0 + c_1 x + \cdots + c_d x^d$$

that "best" approximates the y values.

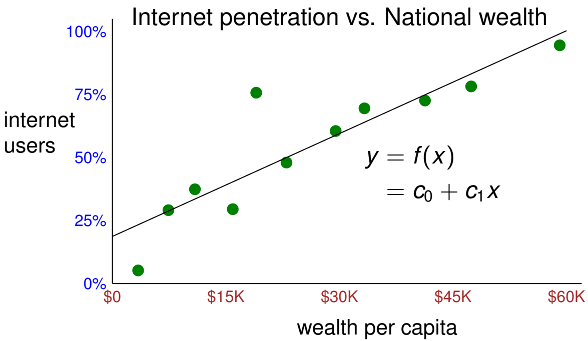


Formula:

$$\vec{c} = (X^t X)^{-1} X^t \vec{y}$$

$\vec{c}$ = vector of coefficients of $f(x)$,
X = Vandermonde matrix of x values,
$\vec{y}$ = vector of y values.

Authenticating a least-squares fit (x-values only)

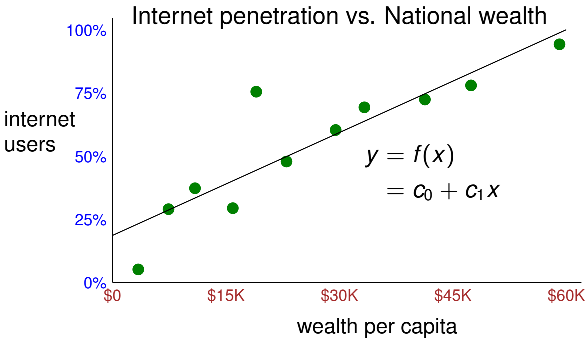


Formula:

$$\vec{c} = (X^t X)^{-1} X^t \vec{y}$$

$\vec{c}$ = vector of coefficients of $f(x)$,
X = Vandermonde matrix of x values,
$\vec{y}$ = vector of y values.

Authenticating a least-squares fit (x-values only)



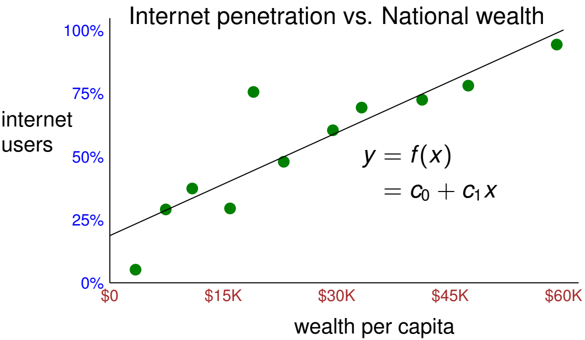
Formula:

$$\vec{c} = (X^t X)^{-1} X^t \vec{y}$$

$\vec{c}$ = vector of coefficients of $f(x)$,
X = Vandermonde matrix of x values,
$\vec{y}$ = vector of y values.

- Coefficients c_j are rational functions of sampled x and y values.

Authenticating a least-squares fit (x-values only)



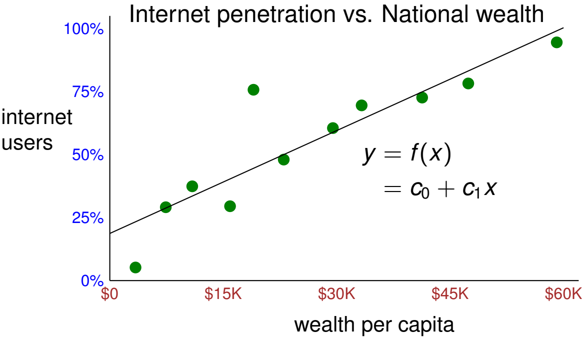
Formula:

$$\vec{c} = (X^t X)^{-1} X^t \vec{y}$$

$\vec{c}$ = vector of coefficients of $f(x)$,
X = Vandermonde matrix of x values,
$\vec{y}$ = vector of y values.

- Coefficients c_j are rational functions of sampled x and y values.
- However: $\det(X^t X) \cdot c_j$ are polynomial functions of x and y.

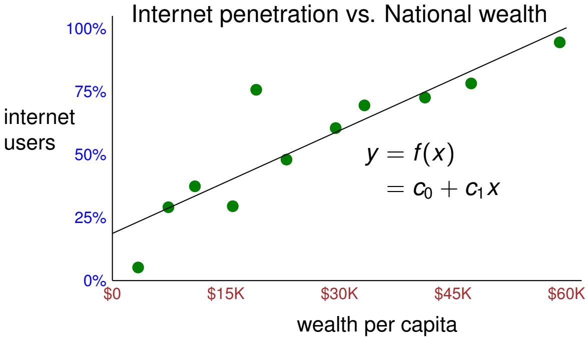
Authenticating a least-squares fit (x-values only)



Formula: $\det(X^t X) \cdot c_j =$ polynomial in $\{x_i, y_i\}$

- United Nations stores signed data on server using polynomially homomorphic signature.

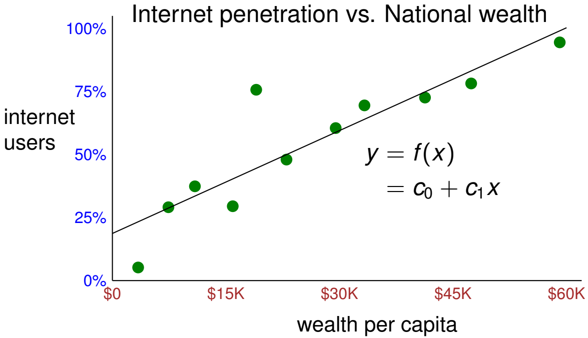
Authenticating a least-squares fit (x-values only)



Formula: $\det(X^t X) \cdot c_j =$ polynomial in $\{x_i, y_i\}$

- United Nations stores signed data on server using polynomially homomorphic signature.
- Server can authenticate $\det(X^t X)$ and $\det(X^t X) \cdot \vec{c}$.

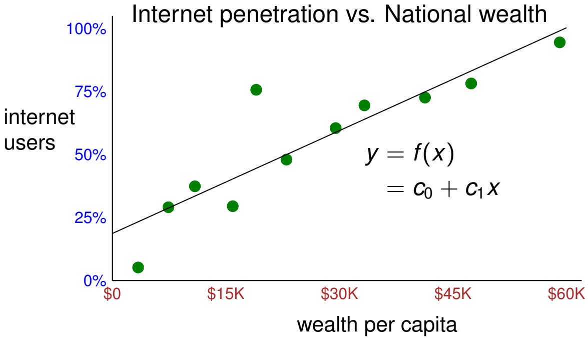
Authenticating a least-squares fit (x-values only)



Formula: $\det(X^t X) \cdot c_j =$ polynomial in $\{x_i, y_i\}$

- United Nations stores signed data on server using polynomially homomorphic signature.
- Server can authenticate $\det(X^t X)$ and $\det(X^t X) \cdot \vec{c}$.
- User can compute least-squares fit from server's values.

Authenticating a least-squares fit (x-values only)



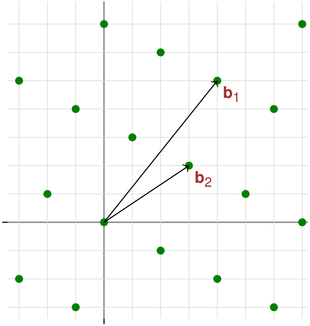
Formula: $\det(X^t X) \cdot c_j =$ polynomial in $\{x_i, y_i\}$

- United Nations stores signed data on server using polynomially homomorphic signature.
- Server can authenticate $\det(X^t X)$ and $\det(X^t X) \cdot \vec{c}$.
- User can compute least-squares fit from server's values.
- Linear fit can be computed using degree 3 polynomials.

Homomorphic Signatures:
Our Construction

Building block: GPV Signatures

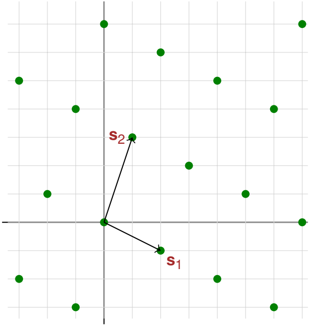
pk: $\Lambda \subset \mathbb{Z}^n$ a lattice (full-rank additive subgroup), defined by "bad" basis.



Building block: GPV Signatures

pk: $\Lambda \subset \mathbb{Z}^n$ a lattice (full-rank additive subgroup), defined by "bad" basis.

sk: "good" basis of Λ.

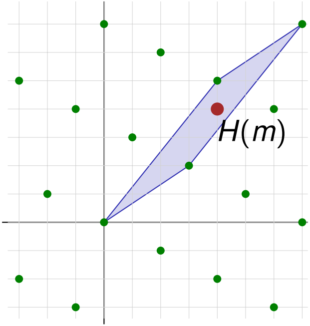


Building block: GPV Signatures

pk: $\Lambda \subset \mathbb{Z}^n$ a lattice (full-rank additive subgroup), defined by "bad" basis.

sk: "good" basis of Λ.

H: $\{0,1\}^* \to \mathbb{Z}^n/\Lambda$
(fix unique representatives).



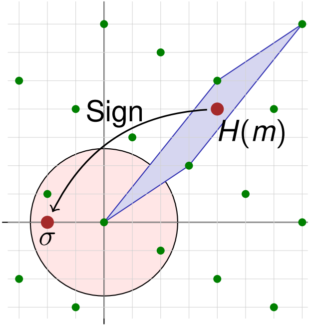
Building block: GPV Signatures

pk: $\Lambda \subset \mathbb{Z}^n$ a lattice (full-rank additive subgroup), defined by "bad" basis.

sk: "good" basis of Λ.

H: $\{0,1\}^* \to \mathbb{Z}^n/\Lambda$ (fix unique representatives).

- $\text{Sign}(sk, m) =$ short vector $\sigma \in \Lambda + H(m)$.



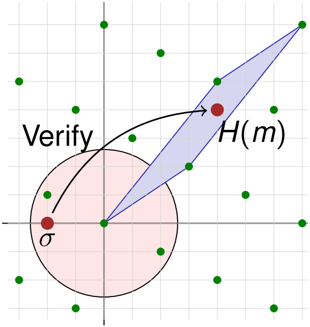
Building block: GPV Signatures

pk: $\Lambda \subset \mathbb{Z}^n$ a lattice (full-rank additive subgroup), defined by "bad" basis.

sk: "good" basis of Λ.

H: $\{0,1\}^* \to \mathbb{Z}^n/\Lambda$ (fix unique representatives).

- $\text{Sign}(sk, m) =$ short vector $\sigma \in \Lambda + H(m)$.
- $\text{Verify}(pk, m, \sigma)$: check that
  1. σ is short,
  2. $\sigma \bmod \Lambda = H(m)$.



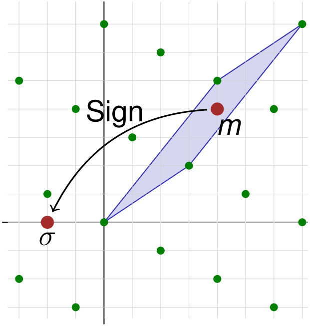
Building block: GPV Signatures

pk: $\Lambda \subset \mathbb{Z}^n$ a lattice (full-rank additive subgroup), defined by "bad" basis.

sk: "good" basis of Λ.

H: $\{0,1\}^* \to \mathbb{Z}^n/\Lambda$ (fix unique representatives).

- $\text{Sign}(sk, m) =$ short vector $\sigma \in \Lambda + m$.
- $\text{Verify}(pk, m, \sigma)$: check that
  1. σ is short,
  2. $\sigma \bmod \Lambda = m$.



What if we encode m in $\mathbb{Z}^n/\Lambda$ directly?

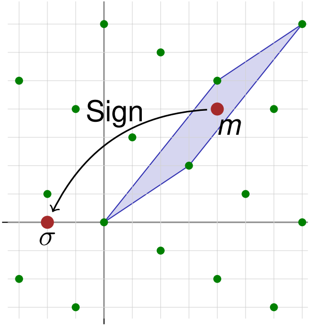
Building block: GPV Signatures

pk: $\Lambda \subset \mathbb{Z}^n$ a lattice (full-rank additive subgroup), defined by "bad" basis.

sk: "good" basis of Λ.

H: $\{0,1\}^* \to \mathbb{Z}^n/\Lambda$
(fix unique representatives).

- $\mathsf{Sign}(sk, m) =$
  short vector $\sigma \in \Lambda + m$.
- $\mathsf{Verify}(pk, m, \sigma)$: check that
  1. σ is short,
  2. $\sigma \bmod \Lambda = m$.



What if we encode m in $\mathbb{Z}^n/\Lambda$ directly?
Then signatures are linearly homomorphic:

$$(\sigma_1 + \sigma_2) \text{ is short}, \quad (\sigma_1 + \sigma_2) \bmod \Lambda = m_1 + m_2$$

so $\sigma_1 + \sigma_2$ authenticates $m_1 + m_2$!

Problem: Removing hash function destroys security

Valid signature doesn't imply function was computed correctly.

Problem: Removing hash function destroys security

Valid signature doesn't imply function was computed correctly.

sk

signed grades $\longrightarrow$

Untrusted DB

Student	Score	Sig
1	91	σ_1
2	73	σ_2
⋮	⋮	⋮
k	84	σ_k

$\longleftarrow$ mean?

pk

Problem: Removing hash function destroys security

Valid signature doesn't imply function was computed correctly.

sk

signed grades →

Untrusted DB

Student	Score	Sig
1	91	σ_1
2	73	σ_2
⋮	⋮	⋮
k	84	σ_k

← mean?

→ 87.3, σ

pk

- Honest DB outputs $87.3 = \frac{1}{k}\sum s_i$ and signature $\sigma = \frac{1}{k}\sum \sigma_i$.

Problem: Removing hash function destroys security

Valid signature doesn't imply function was computed correctly.

sk

signed grades →

Untrusted DB

Student	Score	Sig
1	91	σ_1
2	73	σ_2
⋮	⋮	⋮
k	84	σ_k

← mean?

→ 18.0, σ

pk

- Honest DB outputs $87.3 = \frac{1}{k}\sum s_i$ and signature $\sigma = \frac{1}{k}\sum \sigma_i$.
- Malicious DB outputs $18.0 = s_1 - s_2$ and signature $\sigma = \sigma_1 - \sigma_2$.

Problem: Removing hash function destroys security

Valid signature doesn't imply function was computed correctly.

sk

signed $\longrightarrow$ grades

Untrusted DB

Student	Score	Sig
1	91	σ_1
2	73	σ_2
⋮	⋮	⋮
k	84	σ_k

mean? $\longleftarrow$

$\longrightarrow$ 18.0, σ

pk

- Honest DB outputs $87.3 = \frac{1}{k}\sum s_i$ and signature $\sigma = \frac{1}{k}\sum \sigma_i$.
- Malicious DB outputs $18.0 = s_1 - s_2$ and signature $\sigma = \sigma_1 - \sigma_2$.
- σ authenticates 18, but 18 is not the mean!

How to recover security

Use a second lattice to authenticate functions:

- $\Lambda_2 \subset \mathbb{Z}^n$ distinct from $\Lambda_1 := \Lambda$.

How to recover security

Use a second lattice to authenticate functions:

- $\Lambda_2 \subset \mathbb{Z}^n$ distinct from $\Lambda_1 := \Lambda$.

"Encode" functions f as elements $\omega(f) \in \mathbb{Z}^n / \Lambda_2$.
Sign functions by computing

$$\text{Sign}(f) := \text{short vector in } (\Lambda_2 + \omega(f)).$$

How to recover security

Use a second lattice to authenticate functions:

- $\Lambda_2 \subset \mathbb{Z}^n$ distinct from $\Lambda_1 := \Lambda$.

"Encode" functions f as elements $\omega(f) \in \mathbb{Z}^n/\Lambda_2$.
Sign functions by computing

$$\text{Sign}(f) := \text{short vector in } (\Lambda_2 + \omega(f)).$$

If "encoding" $\omega(\cdot)$ is linear, (i.e., $\omega(f) + \omega(g) = \omega(f + g)$)
signatures are linear on the space of functions.

"Intersection method" binds messages to functions

Messages $m \in \mathbb{Z}^n/\Lambda_1$, functions f encoded as $\omega(f) \in \mathbb{Z}^n/\Lambda_2$.

“Intersection method” binds messages to functions

Messages $m \in \mathbb{Z}^n/\Lambda_1$, functions f encoded as $\omega(f) \in \mathbb{Z}^n/\Lambda_2$.

Define signature as simultaneous GPV signature on $(m, \omega(f))$.

- $\text{Sign}(m) =$ short vector in $(\Lambda_1 + m) \cap (\Lambda_2 + \omega(f))$.
  $sk =$ “good” basis of $\Lambda_1 \cap \Lambda_2$.

“Intersection method” binds messages to functions

Messages $m \in \mathbb{Z}^n/\Lambda_1$, functions f encoded as $\omega(f) \in \mathbb{Z}^n/\Lambda_2$.

Define signature as simultaneous GPV signature on $(m, \omega(f))$.

- Sign(m) $=$ short vector in $(\Lambda_1 + m) \cap (\Lambda_2 + \omega(f))$.
  sk $=$ “good” basis of $\Lambda_1 \cap \Lambda_2$.
- Verify(σ): check that
  1. σ is short,
  2. σ mod $\Lambda_1 = m$,
  3. σ mod $\Lambda_2 = \omega(f)$.

“Intersection method” binds messages to functions

Messages $m \in \mathbb{Z}^n/\Lambda_1$, functions f encoded as $\omega(f) \in \mathbb{Z}^n/\Lambda_2$.

Define signature as simultaneous GPV signature on $(m, \omega(f))$.

- Sign(m) = short vector in $(\Lambda_1 + m) \cap (\Lambda_2 + \omega(f))$.
  sk = “good” basis of $\Lambda_1 \cap \Lambda_2$.
- Verify(σ): check that
  1. σ is short,
  2. $\sigma \bmod \Lambda_1 = m$,
  3. $\sigma \bmod \Lambda_2 = \omega(f)$.
- Evaluate$(f, (\sigma_1, \ldots, \sigma_k)) = f(\sigma_1, \ldots, \sigma_k)$ for linear f.
  If σ_i = Sign(m_i), output authenticates $f(m_1, \ldots, m_k)$.

Homomorphic signatures for polynomial functions

Linearly homomorphic scheme: messages in $\mathbb{Z}^n/\Lambda_1$, functions encoded in $\mathbb{Z}^n/\Lambda_2$, signatures in $\mathbb{Z}^n$.

Verification computes a linear map
$\Rightarrow$ adding signatures corresponds to adding messages.

Homomorphic signatures for polynomial functions

Linearly homomorphic scheme: messages in $\mathbb{Z}^n/\Lambda_1$, functions encoded in $\mathbb{Z}^n/\Lambda_2$, signatures in $\mathbb{Z}^n$.

Verification computes a linear map
$\Rightarrow$ adding signatures corresponds to adding messages.

What if $\mathbb{Z}^n$ has a ring structure and Λ_1, Λ_2 are ideal lattices?

Homomorphic signatures for polynomial functions

Linearly homomorphic scheme: messages in $\mathbb{Z}^n/\Lambda_1$, functions encoded in $\mathbb{Z}^n/\Lambda_2$, signatures in $\mathbb{Z}^n$.

Verification computes a linear map
$\Rightarrow$ adding signatures corresponds to adding messages.

What if $\mathbb{Z}^n$ has a ring structure and Λ_1, Λ_2 are ideal lattices?

Then verification computes a ring homomorphism
$\Rightarrow$ adding or multiplying signatures corresponds to adding or multiplying messages.

Homomorphic signatures for polynomial functions

Linearly homomorphic scheme: messages in $\mathbb{Z}^n/\Lambda_1$, functions encoded in $\mathbb{Z}^n/\Lambda_2$, signatures in $\mathbb{Z}^n$.

Verification computes a linear map
$\Rightarrow$ adding signatures corresponds to adding messages.

What if $\mathbb{Z}^n$ has a ring structure and Λ_1, Λ_2 are ideal lattices?

Then verification computes a ring homomorphism
$\Rightarrow$ adding or multiplying signatures corresponds to adding or multiplying messages.

- Same construction now authenticates polynomial functions on messages.

Homomorphic signatures for polynomial functions

Linearly homomorphic scheme: messages in $\mathbb{Z}^n/\Lambda_1$,
functions encoded in $\mathbb{Z}^n/\Lambda_2$, signatures in $\mathbb{Z}^n$.

Verification computes a linear map
$\Rightarrow$ adding signatures corresponds to adding messages.

What if $\mathbb{Z}^n$ has a ring structure and Λ_1, Λ_2 are ideal lattices?

Then verification computes a ring homomorphism
$\Rightarrow$ adding or multiplying signatures corresponds to adding or multiplying messages.

- Same construction now authenticates polynomial functions on messages.
- Length of signature vector grows with polynomial degree
  $\Rightarrow$ degree must be bounded to ensure security.

Security

What does it mean to forge a homomorphic signature?

- Forgery is a valid signature on (τ, m^*, f) with $m^* \neq f(\text{messages in file } \tau)$.

What does it mean to forge a homomorphic signature?

- Forgery is a valid signature on (τ, m^*, f) with $m^* \neq f(\text{messages in file } \tau)$.

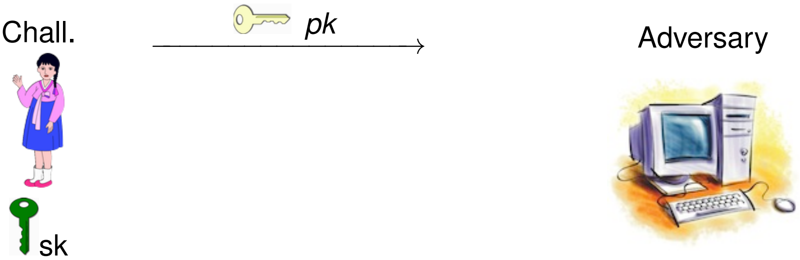
Chall.

sk

Adversary

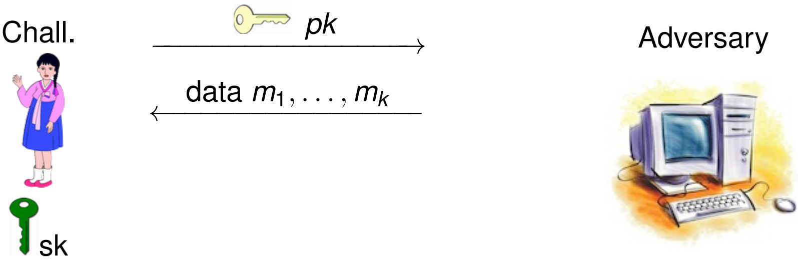
What does it mean to forge a homomorphic signature?

- Forgery is a valid signature on (τ, m^*, f) with $m^* \neq f(\text{messages in file } \tau)$.



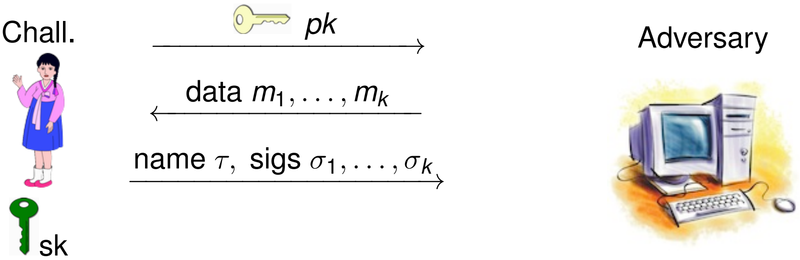
What does it mean to forge a homomorphic signature?

- Forgery is a valid signature on (τ, m^*, f) with $m^* \neq f(\text{messages in file } \tau)$.



What does it mean to forge a homomorphic signature?

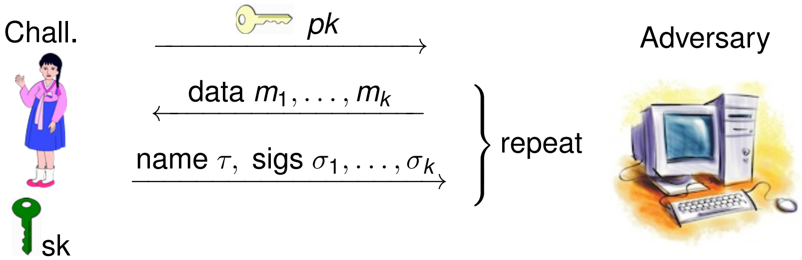
- Forgery is a valid signature on (τ, m^*, f) with $m^* \neq f(\text{messages in file } \tau)$.



Security

What does it mean to forge a homomorphic signature?

- Forgery is a valid signature on (τ, m^*, f) with $m^* \neq f(\text{messages in file } \tau)$.



What does it mean to forge a homomorphic signature?

- Forgery is a valid signature on (τ, m^*, f) with $m^* \neq f(\text{messages in file } \tau)$.

Chall.

sk

pk →

← data $m_1, \ldots, m_k$

name τ, sigs $\sigma_1, \ldots, \sigma_k$ →

} repeat

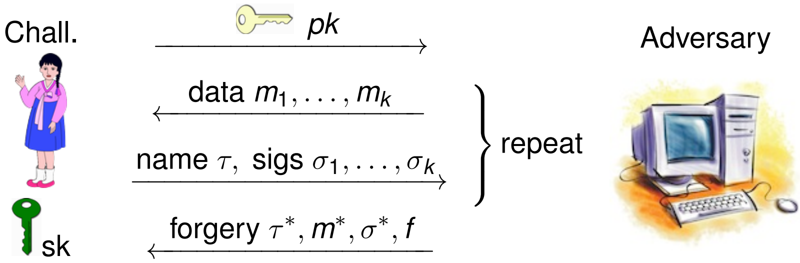
← forgery τ^*, m^*, σ^*, f

Adversary

Security

What does it mean to forge a homomorphic signature?

- Forgery is a valid signature on (τ, m^*, f) with $m^* \neq f(\text{messages in file } \tau)$.



Adversary wins if f admissible, σ^* verifies for (τ^*, m^*, f), and

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 Theorem

Theorem

An adversary that wins the security game (in the random oracle model) can be used to compute a short nonzero vector in Λ_2.

Security Theorem

Theorem

An adversary that wins the security game (in the random oracle model) can be used to compute a short nonzero vector in Λ_2.

To implement system securely:
Choose Λ_2 such that finding short vectors in Λ_2 is hard!

Security Theorem

Theorem

An adversary that wins the security game (in the random oracle model) can be used to compute a short nonzero vector in Λ_2.

To implement system securely:
Choose Λ_2 such that finding short vectors in Λ_2 is hard!

- Linear system: use "q-ary" lattices defined by a random matrix over $\mathbb{F}_q$.

  Finding short vectors is as hard as solving worst-case lattice problems [A96,MR04,GPV08].

Security Theorem

Theorem

An adversary that wins the security game (in the random oracle model) can be used to compute a short nonzero vector in Λ_2.

To implement system securely:
Choose Λ_2 such that finding short vectors in Λ_2 is hard!

- Linear system: use "q-ary" lattices defined by a random matrix over $\mathbb{F}_q$.
  Finding short vectors is as hard as solving worst-case lattice problems [A96,MR04,GPV08].
- Polynomial system: use ideal lattices proposed for homomorphic encryption [SV10].

Privacy

Privacy property: derived signature on $f(m_1, \ldots, m_k)$ reveals nothing about $m_1, \ldots, m_k$ beyond value of f.

Privacy

Privacy property: derived signature on $f(m_1, \ldots, m_k)$ reveals nothing about $m_1, \ldots, m_k$ beyond value of f.

Specifically: given data sets

$$\vec{m} = (m_1, \ldots, m_k), \qquad \vec{m}' = (m'_1, \ldots, m'_k)$$

and admissible function f with

$$f(\vec{m}) = f(\vec{m}'),$$

even unbounded adversary cannot distinguish derived signature on $f(\vec{m})$ from derived signature on $f(\vec{m}')$.

Privacy

Privacy property: derived signature on $f(m_1, \ldots, m_k)$ reveals nothing about $m_1, \ldots, m_k$ beyond value of f.

Specifically: given data sets

$$\vec{m} = (m_1, \ldots, m_k), \qquad \vec{m}' = (m'_1, \ldots, m'_k)$$

and admissible function f with

$$f(\vec{m}) = f(\vec{m}'),$$

even unbounded adversary cannot distinguish derived signature on $f(\vec{m})$ from derived signature on $f(\vec{m}')$.

Theorem

Our linearly homomorphic signatures are private.

Open questions

1. Privacy for polynomially homomorphic signatures.
   - Current polynomial construction is not private.

Open questions

1. Privacy for polynomially homomorphic signatures.
   - Current polynomial construction is not private.

2. Remove random oracle from security proof.
   - Achieved for linear scheme.
   - Work in progress for polynomial scheme.

Open questions

1. Privacy for polynomially homomorphic signatures.
   - Current polynomial construction is not private.
2. Remove random oracle from security proof.
   - Achieved for linear scheme.
   - Work in progress for polynomial scheme.
3. Reduce security to worst-case lattice problems.
   - Achieved for linear scheme.
   - Achieve for polynomial scheme using Gentry's techniques?

Open questions

1. Privacy for polynomially homomorphic signatures.
   - Current polynomial construction is not private.
2. Remove random oracle from security proof.
   - Achieved for linear scheme.
   - Work in progress for polynomial scheme.
3. Reduce security to worst-case lattice problems.
   - Achieved for linear scheme.
   - Achieve for polynomial scheme using Gentry's techniques?
4. Fully homomorphic signatures!
   - Adapt "bootstrapping" approach???

Open questions

1. Privacy for polynomially homomorphic signatures.
    - Current polynomial construction is not private.
2. Remove random oracle from security proof.
    - Achieved for linear scheme.
    - Work in progress for polynomial scheme.
3. Reduce security to worst-case lattice problems.
    - Achieved for linear scheme.
    - Achieve for polynomial scheme using Gentry's techniques?
4. Fully homomorphic signatures!
    - Adapt "bootstrapping" approach???

Thank you!