Policy-based signatures

Georg Fuchsbauer
IST Austria

joint work with Mihir Bellare
available on eprint 2013/413

University of Bristol, 29 Oct 2013
Overview

- New signature primitive
- Signer can only sign messages conforming to policy
Overview

- New signature primitive
- Signer can only sign messages conforming to policy

- **Practical** applications: use for corporations
- **Theoretical**: unification of existing work
Signatures

Signer

$vk$
Signatures

Verifier

Signer

(m, σ)

vk

sk
Policy-based signatures
Policy-based signatures

\[ (p, m) \in L \subseteq \{0,1\}^* \times \{0,1\}^* \]
Unforgeability:

You can only sign a message $m$ if you have a key for a policy $p$ satisfied by $m$
Security

- **Unforgeability:**
  You can only sign a message $m$ if you have a key for a policy $p$ satisfied by $m$

- **Privacy:**
  The signature does not reveal the policy
Theoretical Motivation

- Signature analog to functional encryption  [BSW11]
Theoretical Motivation

- **Signature analog to functional encryption** \[BSW11\]
- **Unification of existing notions for signatures with privacy:**
  - Group signatures \[Cv91\]
  - (Anonymous) proxy signatures \[MUO96,FP08\]
  - Ring signatures, mesh signatures \[RST01,Boy07\]
  - Attribute-based signatures \[MPR11\]
  - Anonymous credentials \[CL01,BCKL08\]
Group signatures

Manager → $gvk$

Member $i$ → $(m, \sigma)$

Verifier
Group signatures

Manager $\rightarrow gvk$

Member $i$

$sk_i$

Verifier

$(m, \sigma)$

$\sigma$ anonymous, but can be opened:

Manager $\rightarrow i$
Attribute-based signatures

Authority → pp

{sk_{a_1,...,a_n}}

Signer

{a_1,a_2,...,a_n} → Attributes

φ → Predicate
Attribute-based signatures

\[ \text{Authority} \rightarrow pp \]

\[ \text{Signer} \rightarrow (\varphi, m, \sigma) \rightarrow \text{Verifier} \]

\[ \{a_1, a_2, \ldots, a_n\} \rightarrow \{a_1, a_2, \ldots, a_n\} \]

only if \( \varphi(a_1, a_2, \ldots, a_n) = 1 \)

\( \sigma \) does not reveal \( \{a_1, a_2, \ldots, a_n\} \)
Practical Motivation

- Company with public key $vk$
- Employees get signing keys enabling signing **anonymously** on behalf of company
Practical Motivation

- Company with public key $vk$
- Employees get signing keys enabling signing anonymously on behalf of company

- **Group signatures:**
  - Anonymous signing, no control of what can be signed
Practical Motivation

- Company with public key \( vk \)
- Employees get signing keys enabling signing 
  anonymously on behalf of company

- Group signatures:
  - Anonymous signing, no control of what can be signed

- Attribute-based signatures:
  - Signing w.r.t policies like
    CEO \( \lor \) (board member \( \land \) manager)
Can we do better?

⇒ Public policies...

• Does verifier need to know?
Can we do better?

⇒ Public policies…
  • Does verifier need to know?

⇒ Verification w.r.t. policies…
  • Verifier must judge if message ok under policy:

CEO ∨ Intern
Can we do better?

⇒ Public policies...
  • Does verifier need to know?

⇒ Verification w.r.t. policies...
  • Verifier must judge if message ok under policy:

    CEO ∨ Intern

Policy-based signatures:
  • No public policies
  • Verification w.r.t. vk only
Can we do better?

⇒ Public policies...
  • Does verifier need to know?

⇒ Verification w.r.t. policies...
  • Verifier must judge if message ok under policy:

  CEO ∨ Intern

Policy-based signatures:
  • No public policies
  • Verification w.r.t. vk only

Example:
Employee gets key with which she can sign contracts only with company B.
Definition of PBS
Definition

- Policy languages:
  We allow any language in \( \textbf{NP} \), defined by policy checker
    \[ PC : ((p, m), w) \rightarrow \{0, 1\} \]
    \[ (p, m) \in L(PC) \iff \exists w \in \{0, 1\}^* : PC((p, m), w) = 1 \]
    ... iff signing of \( m \) is permitted under \( p \)
Definition

● Policy languages:

We allow any language in \( \textbf{NP} \), defined by policy checker

\[
PC : ((p,m),w) \rightarrow \{0,1\}
\]

\[
(p,m) \in L(\text{PC}) \iff \exists \ w \in \{0,1\}^* : PC((p,m),w) = 1
\]

... iff signing of \( m \) is permitted under \( p \)

● Algorithms:

\[
\begin{align*}
\text{Setup}(1^\lambda) & \quad \rightarrow \quad (pp,msk) \\
\text{KeyGen}(msk,p) & \quad \rightarrow \quad sk_p \\
\text{Sign}(sk_p,m,w) & \quad \rightarrow \quad \sigma \\
\text{Verify}(pp,m,\sigma) & \quad \rightarrow \quad b
\end{align*}
\]
Unforgeability

- Game

Adversary

Challenger

Query key for $p$

Query signature on $m$

$m^*, \sigma^*$
Unforgeability

- Game

Adversary

Challenger

$pp$ 

Query key for $p$

Query signature on $m$

$m^*, \sigma^*$

A wins if
- $\text{Verify}(pp, m^*, \sigma^*) = 1$,
- no signature query for $m^*$,
- for all key queries for $p$: $(p, m^*) \notin L(\text{PC})$
Indistinguishability

- Game

Adversary

Challenger

\[
\begin{align*}
\text{pp,msk} & \quad \rightarrow \quad \text{pp,msk} \\
p_0, p_1, m, w_0, w_1 & \quad \rightarrow \quad \text{pp,msk} \\
sk_0 & \leftarrow \text{KeyGen}(p_0) \\
sk_1 & \leftarrow \text{KeyGen}(p_1) \\
\sigma & \leftarrow \text{Sign}(sk_b,m,w_b) \\
b' & \rightarrow \text{pp,msk} \\
b & \leftarrow \{0,1\}
\end{align*}
\]
Indistinguishability

- **Game**

  \[ pp,msk \]

  \[ p_0, p_1, m, w_0, w_1 \]

  \[ sk_0 \leftarrow \text{KeyGen}(p_0) \]
  \[ sk_1 \leftarrow \text{KeyGen}(p_1) \]
  \[ \sigma \leftarrow \text{Sign}(sk_b, m, w_b) \]

  \[ b' \]

  A wins if
  - \( \text{PC}((p_0,m),w_0) = \text{PC}((p_1,m),w_1) = 1 \),
  - \( b' = b \)
Indistinguishability

- Game

\[ \text{Adversary} \]

\[ \text{Challenger} \]

\[ pp,msk \]

\[ p_0, p_1, m, w_0, w_1 \]

\[ sk_0 \leftarrow \text{KeyGen}(p_0) \]

\[ sk_1 \leftarrow \text{KeyGen}(p_1) \]

\[ \sigma \leftarrow \text{Sign}(sk_b, m, w_b) \]

\[ b' \]

\[ A \text{ wins if } \]

- \( \text{PC}((p_0,m),w_0) = \text{PC}((p_1,m),w_1) = 1 \),
- \( b' = b \)
Security notions

- Indistinguishability
  - Adversary gets $sk_0$ and $sk_1$ $\Rightarrow$ unlinkability
Security notions

- **Indistinguishability**
  - Adversary gets $sk_0$ and $sk_1$ $\Rightarrow$ unlinkability
  - Policy-revealing schemes still secure! (e.g. if only one policy per message)
Security notions

- **Indistinguishability**
  - Adversary gets $sk_0$ and $sk_1$ $\Rightarrow$ unlinkability
  - Policy-revealing schemes still secure! (e.g. if only one policy per message)

$\Rightarrow$ Simulation-based definition
Security notions

- **Indistinguishability**
  - Adversary gets $sk_0$ and $sk_1$ $\Rightarrow$ unlinkability
  - Policy-revealing schemes still secure!
    (e.g. if only one policy per message)
    $\Rightarrow$ Simulation-based definition

- **Unforgeability**
  - Not efficiently verifiable if game was won
    (have to check whether $(m^*, p) \notin L(\text{PC}))$
Security notions

- **Indistinguishability**
  - Adversary gets $sk_0$ and $sk_1$ $\Rightarrow$ unlinkability
  - Policy-revealing schemes still secure!
    (e.g. if only one policy per message)
  $\Rightarrow$ Simulation-based definition

- **Unforgeability**
  - Not efficiently verifiable if game was won
    (have to check whether $(m^*, p) \not\in L(PC)$)
  $\Rightarrow$ Extraction-based definition

Extr+Sim $\Rightarrow$ UF+IND
Constructions of PBS
Constructions

- **Generic construction** (à la [BMW03])
  - signatures
    - IND-CPA encryption
    - NIZK proofs
  for any policy language in **NP**
Constructions

• **Generic construction** (à la [BMW03])
  
  based on - signatures or based on
  
  - IND-CPA encryption - SSE-NIZK
  
  - NIZK proofs

  for any policy language in **NP**
Constructions

- **Generic construction** (à la [BMW03])
  - based on signatures or based on
    - IND-CPA encryption - SSE-NIZK
    - NIZK proofs
  for any policy language in \( \text{NP} \)

- **Efficient construction**
  - based on structure-preserving signatures [AFG+10]
    - Groth-Sahai proofs [GS08]
  for policy languages over **pairing groups**
  (policies define pairing-product equations)
Primitives from PBS
Attribute-based signatures from PBS

- **Authority**
  - \( pp \)
  - \( sk_{k(a_1,...,a_n)} \)
  - \( \{a_1,a_2,...,a_n\} \)

- **Signer**
  - \( (\varphi,m,\sigma) \)
  - only if \( \varphi(a_1,a_2,...,a_n) = 1 \)
  - \( \sigma \) does not reveal \( \{a_1,a_2,...,a_n\} \)

- **Verifier**
Attribute-based signatures from PBS

- Policies $p$ ... set of attributes $A = \{a_1, a_2, ..., a_n\}$
- PBS messages of form $M = (\varphi, m)$
Attribute-based signatures from PBS

- Policies $p$ ... set of attributes $A = \{a_1, a_2, \ldots, a_n\}$
- PBS messages of form $M = (\phi, m)$
- PC: $(A, (\phi, m))$
Attribute-based signatures from PBS

- Policies $p$ ... set of attributes $A = \{a_1, a_2, ..., a_n\}$
- PBS messages of form $M = (\varphi, m)$
- PC : $(A, (\varphi, m)) \rightarrow \varphi(A)$
Group signatures from PBS

Manager $\rightarrow gvk$

$sk_i$ $

Member i

(m, \sigma)$

Verifier

$\sigma$anonymous, but can be opened:

Manager $\rightarrow i$
Group signatures from PBS

- Use public-key encryption scheme
- Policies $p \ldots$ group-member identity $i$
Group signatures from PBS

- Use public-key encryption scheme
- Policies $p$ ... group-member identity $i$
- PBS messages of form $M = (c,m)$
- PC : $(((ek,i),(c,m)),r) \rightarrow [c = Enc(ek,i;r)]$
Group signatures from PBS

- Use public-key encryption scheme
- Policies $p$ ... group-member identity $i$
- PBS messages of form $M = (c,m)$
- PC : $(((ek,i),(c,m)),r) \rightarrow [c = Enc(ek,i;r)]$

**GroupKeyGen**: - create $(ek,dk)$ for Enc, $(pp,msk)$ for PBS
  - member $i$ gets key for $p = (ek,i)$

**Sign**(sk$_i$,m): encrypt $i$ as $c$, sign $(c,m)$, output $\Sigma = (c,\sigma)$
Group signatures from PBS

- Use public-key encryption scheme
- Policies \( p \) ... group-member identity \( i \)
- PBS messages of form \( M = (c,m) \)
- PC : \( (((ek,i),(c,m)),r) \rightarrow [c = Enc(ek,i;r)] \)

**GroupKeyGen:**
- create \( (ek,dk) \) for Enc, \( (pp,msk) \) for PBS
- member \( i \) gets key for \( p = (ek,i) \)

**Sign\( (sk_i,m) \):** encrypt \( i \) as \( c \), sign \( (c,m) \), output \( \Sigma = (c,\sigma) \)

**Verify:** verify PBS

**Open\( (dk,(c,\sigma)) \):** decrypt \( c \)
Other primitives from PBS

- Simulation-sound extractable NIZK proofs [Gro06]
Other primitives from PBS

- Simulation-sound extractable NIZK proofs \[\text{Gro06}\]
- CPA-secure public-key encryption
Other primitives from PBS

- Simulation-sound extractable NIZK proofs [Gro06]
- CPA-secure public-key encryption
- combining the above [Sah99]: CCA-secure encryption
Other primitives from PBS

- Simulation-sound extractable NIZK proofs [Gro06]
- CPA-secure public-key encryption
- combining the above [Sah99]: CCA-secure encryption
  thus PBS $\Rightarrow$ group signatures
Other primitives from PBS

- Simulation-sound extractable NIZK proofs [Gro06]
- CPA-secure public-key encryption

- combining the above [Sah99]: CCA-secure encryption

  thus PBS ⇒ group signatures

- Signatures of knowledge [CL06]
Delegatable PBS
Re-delegation

- **Delegatable PBS**
  - holding $sk_p$, one can delegate $sk_{p'}$ for subpolicy $p'$
- Reflects hierarchies in organizations
Re-delegation

• Delegatable PBS
  – holding $sk_p$, one can delegate $sk_{p'}$ for subpolicy $p'$
• Reflects hierarchies in organizations

```
Alice

“sign contract with $C_1,C_2,...,C_n$”

Bob
```
Re-delegation

- **Delegatable PBS**
  - holding $sk_p$, one can delegate $sk_{p'}$ for subpolicy $p'$
- Reflects hierarchies in organizations

```
Alice

“sign contract with $C_1, C_2, \ldots, C_n$”

Bob

“sign contract with $C_k$”

Carol
```
Thank you
PBS from SSE-NIZK

- Simulation-sound extractable NIZK:
  - prove membership for NP languages

- Authority has signature key pair $(vk, sk)$
- $sk_p$ is signature on $p$
- PBS-signature on $m$ is SSE-NIZK proof that $(vk, m) \in L$
  defined by

$$((vk, m), (p, sig, w)) \in R_L \iff ((p, m), w) \in PC \land \text{Verify}(vk, p, sig) = 1$$