Policy-based signatures

Mihir Bellare
UCSD

Georg Fuchsbauer
IST Austria

PKC 2014, 28 March 2014

(Full version: eprint 2013/413)
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

\(sk\) \rightarrow \(vk\)
Signatures

Signer

\(vk\)

\(sk\)

\((m, \sigma)\)

Verifier
Policy-based signatures

Authority $\rightarrow pp$

$sk_p \downarrow$

Signer
Policy-based signatures

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

\[ \text{Signer} \quad \downarrow \quad sk_p \quad \rightarrow \quad (m, \sigma) \quad \rightarrow \quad \text{Verifier} \]

\[ \text{only if } (p,m) \in L \subseteq \{0,1\}^* \times \{0,1\}^* \]

\[ L \ldots \text{policy language} \]
Policy-based signatures

Authority \rightarrow pp

\text{Signer} \downarrow \text{sk}_p

only if \((p, m) \in L \subseteq \{0,1\}^* \times \{0,1\}^*\)

L can be any language in \(P\)

Verifier

(m, \sigma)
Policy-based signatures

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

\[ \text{Signer} \rightarrow \text{Verifier} \]

\[ L \text{ can be any language in NP} \]

only if \((p, m) \in L \subseteq \{0,1\}^* \times \{0,1\}^*\)
Security

- **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 hides the policy
  - Signatures under same key are unlinkable
Related work

• **Functional signatures** (Boyle, Goldwasser, Ivan [BGI13]):
  - Key $sk_f$ allows signing messages in range of $f$
  - Interpret $f$ as policy: $(f,m) \in L \iff \exists w : f(w) = m$
Related work

- **Functional signatures** (Boyle, Goldwasser, Ivan [BGI13]):
  - Key $sk_f$ allows signing messages in range of $f$
  - Interpret $f$ as policy: $(f,m) \in L \iff \exists w : f(w) = m$
  - Policy languages in $\mathcal{P}$, succinctness condition
Related work

• **Functional signatures** (Boyle, Goldwasser, Ivan [BGI13]):
  – Key $sk_f$ allows signing messages in range of $f$
  – Interpret $f$ as policy: $(f,m) \in L \iff \exists w : f(w) = m$
  – Policy languages in $\mathcal{P}$, succinctness condition

• **Delegatable functional signatures**
  (Backes, Meiser, Schröder [BMS13]):
  – Signatures verified w.r.t. signer's public key
Related work

• Constrained/delegatable/functional PRFs

  [BW13, BGI13, KPTZ13]:
  – Keys enable evaluation of PRF on parts of domain
Related work

- **Constrained/delegatable/functional PRFs** [BW13, BGI13, KPTZ13]:
  - Keys enable evaluation of PRF on parts of domain

- **Attribute-based signatures** [MPR11]:
  - Keys issued for set of attributes \( \{a_1,a_2,...,a_n\} \)
  - Signing w.r.t. predicate \( \varphi \), possible iff \( \varphi(a_1,a_2,...,a_n) = 1 \)
Motivation for PBS
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 [Cv91]:
- 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** [Cv91]:
  - Anonymous signing, no control of what can be signed

- **Attribute-based signatures** [MPR11]:
  - Verification w.r.t. policies
Practical motivation

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

- **Group signatures** [Cv91]:
  - Anonymous signing, no control of what can be signed

- **Attribute-based signatures** [MPR11]:
  - Verification w.r.t. policies CEO ∨ (board member ∧ 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
Can we do better?

general policies...
  - Does verifier need to know?

general Verification w.r.t. policies...
  - Verifier must judge if message OK under policy, e.g.

CEO ∨ Intern
Can we do better?

⇒ Public policies...
   • Does verifier need to

⇒ Verification w.r.t. policies
   • Verifier must judge if message OK under policy, e.g.

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

CEO ∨ Intern
Can we do better?

Policy-based signatures:

- No public policies
- Verification w.r.t. \( vk \) only
Can we do better?

Example:

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

“sign contract with $C_1, C_2, ..., C_n$”
Can we do better?

Example:

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

Contract with $C_1$
Theoretical motivation

- **Signature analog to functional encryption**  [BSW11]
  - FE: Simply encrypt message, let keys handle access
  - PBS: Simply verify signature; keys handle authorization
Theoretical motivation

- **Signature analog to functional encryption** [BSW11]
  - FE: Simply encrypt message, let keys handle access
  - PBS: Simply verify signature; keys handle authorization

- **Unification of existing notions for signatures with privacy:**
  - (Anonymous) proxy signatures [MUO96, FP08]
  - Ring signatures, mesh signatures [RST01, Boy07]
  - Attribute-based signatures [MPR11]
  - Anonymous credentials [CL01, BCKL08]
  - Group signatures [Cv91]
Definition of PBS
Definition

- **Policy languages:**
  
  We allow any language in $\textbf{NP}$, defined by policy checker $(p, m) \in L(\text{PC}) \iff \exists w : \text{PC}((p, m), w) = 1$
Definition

- **Policy languages:**
  
  We allow any language in \( \mathbf{NP} \), defined by

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

  \( m \) conforms to policy \( p \)

  policy checker
Definition

- **Policy languages:**
  
  We allow any language in \( \mathbf{NP} \), defined by

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

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

• **Indistinguishability**

  An adversary, given $msk$, outputs $sk_0, sk_1$
  and cannot tell with which key a signature was created
Security

- **Indistinguishability**
  An adversary, given $msk$, outputs $sk_0$, $sk_1$
  and cannot tell with which key a signature was created

- **Unforgeability**
  An adversary, after querying:
  - keys for policies $p_1$, ... , $p_n$
  - signatures on messages
  should not be able to create signature on new $m^*$
  with $(p_1, m^*)$, ... , $(p_n, m^*) \notin L$
Security

- **Indistinguishability**
  
  An adversary, given $msk$, outputs $sk_0, sk_1$
  and cannot tell with which key a signature was created

- **Unforgeability**
  
  An adversary, after querying:
  - keys for policies $p_1, \ldots, p_n$
  - signatures on messages

  should not be able to create signature on new $m^*$
  with $(p_1, m^*), \ldots, (p_n, m^*) \notin L$
Sim/ext security

- **Simulatability** $\Rightarrow$ indistinguishability
Sim/ext security

- Simulatability $\iff$ indistinguishability
Sim/ext security

• Simulatability $\iff$ indistinguishability

• Extractability $\Rightarrow$ unforgeability

  is efficiently decidable
Constructions of PBS
Construction I

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

- Concrete construction
  - structure-preserving signatures [AFG+10]
  - Groth-Sahai proofs [GS08]
  
  for policy languages over **pairing groups**
  (policies define pairing-product equations)
Primitives from PBS
CCA-secure group signatures

Manager \rightarrow gvk

Member \textit{i} \rightarrow sk_i

Verifier

\((m, \Sigma)\)
CCA-secure group signatures

Manager $\rightarrow gvk$

$sk_i \downarrow$

Member $i$

$(m, \Sigma) \rightarrow$ Verifier

$\Sigma$ can be opened:

$\Sigma \rightarrow i$
CCA-secure group signatures

Manager $\rightarrow gvk$

Manager $\rightarrow sk_i$

Member $i$ $\rightarrow (m, \Sigma)$

Verifier

Construction from PBS and CCA-encryption

$\Sigma$ can be opened: $\Sigma$ $\rightarrow$ $i$

Manager $\rightarrow i$
CCA-secure group signatures

Manager

\( pp_{PBS}, pk \)

Member \( i \)

\( s_{ki} \)

Verifier

\( (m, \Sigma) \)

\( \Sigma \) can be opened:

Manager

\( \Sigma \)

\( i \)
CCA-secure group signatures

Manager \rightarrow (pp_{PBS}, pk)

\[ m, \Sigma = (Enc(pk,i), \sigma) \]

\[ \Sigma \text{ can be opened: } \Sigma \rightarrow i \]
CCA-secure group signatures

\[
(i, (c, m)) \in L \iff \exists r : c = \text{Enc}(pk, i; r)
\]

Manager

\(pp_{PBS}, pk\)

\(sk_i\)

Member \(i\)

Verifier

\(m, \Sigma = (\text{Enc}(pk, i), \sigma)\)

\(\Sigma\) can be opened:

\(\Sigma\) \rightarrow \text{Manager} \rightarrow \text{Verifi}er

\(\Sigma\) \rightarrow \text{Manager} \rightarrow \text{Member} \rightarrow i
CCA-secure group signatures

Manager

\( (pp_{PBS}, pk) \)

\( sk_i \)

Member \( i \)

\( m, \Sigma = (Enc(pk,i), \sigma) \)

Verifier

\( \Sigma \) can be opened: \( \Sigma = (c, \sigma) \)

\( i = Dec(dk,c) \)
Other primitives from PBS

- **Attribute-based signatures** [MPR11]
Other primitives from PBS

- Attribute-based signatures \[\text{[MPR11]}\]
- Simulation-sound extractable NIZK proofs \[\text{[Gro06]}\]
Other primitives from PBS

- Attribute-based signatures [MPR11]

- Simulation-sound extractable NIZK proofs [Gro06]

- CPA-secure public-key encryption
Other primitives from PBS

- Attribute-based signatures [MPR11]
- Simulation-sound extractable NIZK proofs [Gro06]
- CPA-secure public-key encryption
- combining the above [Sah99]: CCA-secure encryption
  thus PBS $\Rightarrow$ group signatures
Delegatable PBS
Re-delegation

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

- Delegatable PBS
  - holding $sk_p$, derive $sk_{p'}$ for subpolicy $p'$

Diagram:
- Hugo
  - "sign contract with $C_1, C_2, ..., C_n$"
- Diego
Re-delegation

- **Delegatable PBS**
  - holding $sk_p$, derive $sk_{p'}$ for subpolicy $p'$

```
   Hugo
   "sign contract with C_1, C_2, ..., C_n"
   "sign contract with C_k"
   Diego
   "sign contract with C_k"
   Lionel
```
Conclusion

- New primitive, practically motivated
- Umbrella notion for previous primitives
Conclusion

- New primitive, practically motivated
- Umbrella notion for previous primitives
  - Definition
  - Constructions
  - Applications

[O'Neill 14]
Conclusion

- New primitive, practically motivated
- Umbrella notion for previous primitives

Open problems / future work

- Practical schemes for specific policy languages
Thank you