An Exploration of Group and Ring Signatures

Sarah Meiklejohn

UC San Diego Research Exam
4 February 2011
A real-world problem
A real-world problem
A real-world problem

I need to tell the others!
A real-world problem

1. How can we communicate with the other cars?

I need to tell the others!
A real-world problem

1. How can we communicate with the other cars?
2. Can we make sure that some malicious outsider can’t use the system to create traffic mayhem?
Cryptographic background

Group signatures
Outline

- Cryptographic background
- Group signatures
- Ring signatures
Outline

- Cryptographic background
- Group signatures
- Ring signatures
- Open problems
Digital signatures

• Signatures: Signer wants to send a message to Recipient, but wants to make sure she knows the message really came from him
Digital signatures

• Signatures: Signer wants to send a message to Recipient, but wants to make sure she knows the message really came from him
Digital signatures

• Signatures: Signer wants to send a message to Recipient, but wants to make sure she knows the message really came from him

  Signer

• Signer first runs an algorithm $\text{KeyGen}$ to get signing keypair $(pk,sk)$, ...

  Recipient
Digital signatures

• Signatures: Signer wants to send a message to Recipient, but wants to make sure she knows the message really came from him

Signer

\[ \text{sk} \]

Recipient

\[ \text{pk} \]

• Signer first runs an algorithm \text{KeyGen} to get signing keypair \((\text{pk}, \text{sk})\), ...
Digital signatures

• Signatures: Signer wants to send a message to Recipient, but wants to make sure she knows the message really came from him

  \[ \text{Signer} \quad \text{sk} \]

  \[ \text{Recipient} \quad \text{pk} \]

• Signer first runs an algorithm \text{KeyGen} to get signing keypair \((pk,sk)\), ...

• ...then he can compute \( \sigma = \text{Sign}(sk,m) \) for the desired message \( m \), and ...
Digital signatures

• Signatures: Signer wants to send a message to Recipient, but wants to make sure she knows the message really came from him

• Signer first runs an algorithm $\text{KeyGen}$ to get signing keypair $(pk,sk)$, ...

• ...then he can compute $\sigma = \text{Sign}(sk,m)$ for the desired message $m$, and ...
Digital signatures

• Signatures: Signer wants to send a message to Recipient, but wants to make sure she knows the message really came from him

• Signer first runs an algorithm $\text{KeyGen}$ to get signing keypair $(pk, sk)$, ...

• ...then he can compute $\sigma = \text{Sign}(sk, m)$ for the desired message $m$, and ...
Digital signatures

- Signatures: Signer wants to send a message to Recipient, but wants to make sure she knows the message really came from him

- Signer first runs an algorithm $\text{KeyGen}$ to get signing keypair $(pk, sk)$, ...

- ...then he can compute $\sigma = \text{Sign}(sk, m)$ for the desired message $m$, and ...

- Recipient can run $\text{Verify}(pk, \sigma, m)$ to be sure $\sigma$ was created by Signer
Digital signatures

- Signatures: Signer wants to send a message to Recipient, but wants to make sure she knows the message really came from him.

- We need signatures to be \textit{unforgeable}, which means an adversary cannot successfully pretend to be the Signer (without knowing \textit{sk}).
Outline

Cryptographic background

Group signatures
- Intuition and motivation
- Formal definitions
- Extensions and variants
- Comparison of existing schemes

Ring signatures

Open problems
Group signatures: why do we want them?
Group signatures: why do we want them?
Group signatures: why do we want them?

\[ \text{pk}_A, \text{sk}_A \quad \text{pk}_B, \text{sk}_B \]

\[ \text{pk}_C, \text{sk}_C \quad \text{pk}_D, \text{sk}_D \]

Group 1
Group signatures: why do we want them?

pk_A, sk_A  pk_B, sk_B
pk_C, sk_C  pk_D, sk_D

Group 1

Alice: pk_A
Bob: pk_B
Charlie: pk_C
Dora: pk_D
Group signatures: why do we want them?

$m, \sigma = \text{Sign}(sk_B, m)$

Alice: $pk_A$
Bob: $pk_B$
Charlie: $pk_C$
Dora: $pk_D$
Group signatures: why do we want them?

\[ m, \sigma = \text{Sign}(sk_B, m) \]
Group signatures: why do we want them?

\[ m, \sigma = \text{Sign}(sk_B, m) \]

Verify(pk_B, \sigma, m) = 1...

Group 1
Alice: pk_A
Bob: pk_B
Charlie: pk_C
Dora: pk_D
Group signatures: why do we want them?

\[ m, \sigma = \text{Sign}(sk_B, m) \]

Verify(pk_B, \sigma, m) = 1... so Bob wrote the message!

Group 1
Alice: pk_A
Bob: pk_B
Charlie: pk_C
Dora: pk_D
Group signatures: why do we want them?

Verifying the message:

$$m, \sigma = \text{Sign}(sk_B, m)$$

Then:

$$\text{Verify}(pk_B, \sigma, m) = 1$$

so Bob wrote the message! And he works for the CIA!

Group 1:

- Alice: $pk_A$
- Bob: $pk_B$
- Charlie: $pk_C$
- Dora: $pk_D$
Properties of group signatures: anonymity
Properties of group signatures: anonymity
Properties of group signatures: anonymity
Properties of group signatures: anonymity

\[ \text{pk}_{\text{CIA}} \]

\[ \text{sk}_A, \text{sk}_B, \text{sk}_C, \text{sk}_D \]
Properties of group signatures: anonymity

\[ \text{pk}_{\text{CIA}} \]

\[ \text{sk}_A, \text{sk}_B, \text{sk}_C, \text{sk}_D \]
Properties of group signatures: anonymity

\[ m, \sigma = \text{Sign}(sk_B, m) \]
Properties of group signatures: anonymity

\[ m, \sigma = \text{Sign}(sk_B, m) \]

\[ \text{Verify}(pk_{\text{CIA}}, \sigma, m) = 1 \ldots \text{so someone from the CIA wrote the message.} \]
Properties of group signatures: anonymity

Given $\text{Sign}(sk_B, m)$ and $\text{Sign}(sk_D, m)$, recipient can’t tell the difference.

This should be true even if he knows who has signed previous messages.
Properties of group signatures: **traceability**

\[ m, \sigma = \text{Sign}(sk_B, m) \]

Verify(\( pk_{\text{CIA}} _, \sigma, m \)) = 1... so someone from the CIA wrote the message.
Properties of group signatures: **traceability**

\[
m, \sigma = \text{Sign}(sk_B, m)
\]

Verify(pk_{CIA}, \sigma, m) = 1...
so someone from the CIA wrote the message.
Properties of group signatures: traceability

\[ m, \sigma = \text{Sign}(sk_B, m) \]

\[ \text{Verify}(pk_{\text{CIA}}, \sigma, m) = 1 \ldots \]

so someone from the CIA wrote the message.
Properties of group signatures: traceability

$p_{\text{CIA}}$

$t_{\text{CIA}}$

$m = \text{"The CIA is the worst!"}$

$m, \sigma = \text{Sign}(sk_B, m)$

Verify($p_{\text{CIA}}, \sigma, m) = 1...$

so someone from the CIA wrote the message.
Properties of group signatures: traceability

\[ m, \sigma = \text{Sign}(sk_B, m) \]

- Want new algorithm \( \text{Trace} \) s.t. \( \text{Trace}(tk, \sigma) = \text{Bob} \)
- Whoever has access to \( tk \) breaks anonymity

\[ \text{Verify}(pk_{\text{CIA}}, \sigma, m) = 1 \ldots \text{so someone from the CIA wrote the message.} \]

\[ m = \text{“The CIA is the worst!”} \]
Using group signatures with our real-world problem
Using group signatures with our real-world problem

1. How can we communicate with the other cars?
1. How can we communicate with the other cars?
   - Use dedicated short-range transmitters, send the message and a group signature (group = “all cars”)
Using group signatures with our real-world problem

1. How can we communicate with the other cars?
   • Use dedicated short-range transmitters, send the message and a group signature (group = “all cars”)
2. Can we make sure that some malicious outsider can’t use the system to create traffic mayhem?
Using group signatures with our real-world problem

1. How can we communicate with the other cars?
   • Use dedicated short-range transmitters, send the message and a group signature (group = “all cars”)

2. Can we make sure that some malicious outsider can’t use the system to create traffic mayhem?
   • Yes, because group signatures are traceable
Using group signatures with our real-world problem

1. How can we communicate with the other cars?
   - Use dedicated short-range transmitters, send the message and a group signature (group = “all cars”)

2. Can we make sure that some malicious outsider can’t use the system to create traffic mayhem?
   - Yes, because group signatures are traceable

3. Can we do so without revealing private information?
Using group signatures with our real-world problem

1. How can we communicate with the other cars?
   • Use dedicated short-range transmitters, send the message and a group signature (group = “all cars”)

2. Can we make sure that some malicious outsider can’t use the system to create traffic mayhem?
   • Yes, because group signatures are traceable

3. Can we do so without revealing private information?
   • Yes, because group signatures are anonymous
Group signatures: a formal characterization

- A group signature is a tuple of algorithms \((\text{KeyGen}, \text{Sign}, \text{Verify}, \text{Trace})\)
Group signatures: a formal characterization

- A group signature is a tuple of algorithms \((\text{KeyGen,Sign,Verify,Trace})\)

- \(\text{KeyGen}(1^k,1^n)\): outputs group public key \(pk\), master secret key \(msk\), and signing keys for each user in the group \(\{sk_i\}_i\)
Group signatures: a formal characterization

• A group signature is a tuple of algorithms \((\text{KeyGen,Sign,Verify,Trace})\)

• \(\text{KeyGen}(1^k,1^n)\): outputs group public key \(pk\), master secret key \(msk\), and signing keys for each user in the group \(\{sk_i\}_i\)

• \(\text{Sign}(sk_i,m)\): outputs signature \(\sigma\) on message \(m\)
Group signatures: a formal characterization

- A group signature is a tuple of algorithms $(\text{KeyGen}, \text{Sign}, \text{Verify}, \text{Trace})$

- $\text{KeyGen}(1^k, 1^n)$: outputs group public key $pk$, master secret key $msk$, and signing keys for each user in the group $\{sk_i\}_i$

- $\text{Sign}(sk_i, m)$: outputs signature $\sigma$ on message $m$

- $\text{Verify}(pk, \sigma, m)$: checks that $\sigma$ is a valid signature on $m$ formed by some member of the group defined by $pk$ (and outputs 1 if yes and 0 if no)
Group signatures: a formal characterization

• A group signature is a tuple of algorithms \((\text{KeyGen}, \text{Sign}, \text{Verify}, \text{Trace})\)

• \(\text{KeyGen}(1^k, 1^n)\): outputs group public key \(pk\), master secret key \(msk\), and signing keys for each user in the group \(\{sk_i\}_i\)

• \(\text{Sign}(sk_i, m)\): outputs signature \(\sigma\) on message \(m\)

• \(\text{Verify}(pk, \sigma, m)\): checks that \(\sigma\) is a valid signature on \(m\) formed by some member of the group defined by \(pk\) (and outputs 1 if yes and 0 if no)

• \(\text{Trace}(msk, \sigma, m)\): outputs either index \(i\) such that \(\sigma = \text{Sign}(sk_i, m)\) or \(\bot\) to indicate failure (or that \(\text{Verify}(pk, \sigma, m) = 0\))
Group signatures: a formal characterization

- A group signature is a tuple of algorithms \((\text{KeyGen}, \text{Sign}, \text{Verify}, \text{Trace})\)

- \text{KeyGen}(1^k, 1^n): outputs group public key \(pk\), master secret key \(msk\), and signing keys for each user in the group \(\{sk_i\}_i\)

- \text{Sign}(sk_i, m): outputs signature \(\sigma\) on message \(m\)

- \text{Verify}(pk, \sigma, m): checks that \(\sigma\) is a valid signature on \(m\) formed by some member of the group defined by \(pk\) (and outputs 1 if yes and 0 if no)

- \text{Trace}(msk, \sigma, m): outputs either index \(i\) such that \(\sigma = \text{Sign}(sk_i, m)\) or \(\bot\) to indicate failure (or that \(\text{Verify}(pk, \sigma, m) = 0\))
Anonymity: a more formal definition

Game $G$
Anonymity: a more formal definition

Game $G$

Adversary $A$
Anonymity: a more formal definition

Game $G$

Adversary $A$
Anonymity: a more formal definition

Game $G$

Adversary $A$

Phase 1: getting to see who signed which messages
Anonymity: a more formal definition

Game $G$

$pk,msk,\{sk_i\} \leftarrow \text{KeyGen}(1^k,1^n)$

Adversary $A$

Phase 1: getting to see who signed which messages
Anonymity: a more formal definition

Game $G$

$pk, msk, \{sk_i\} \leftarrow \text{KeyGen}(1^k, 1^n)$

Phase 1: getting to see who signed which messages
Anonymity: a more formal definition

Game $G$

Adversary $A$

Phase 1: getting to see who signed which messages
Anonymity: a more formal definition

Game $G$:

Phase 1: getting to see who signed which messages
Anonymity: a more formal definition

Game $G$

Phase 1: getting to see who signed which messages
Anonymity: a more formal definition

Game $G$

Adversary $A$

Phase 1: getting to see who signed which messages
Anonymity: a more formal definition

Game $G$

Phase 1: getting to see who signed which messages
Anonymity: a more formal definition

Phase 2: picking identities and receiving a challenge
Anonymity: a more formal definition

Phase 2: picking identities and receiving a challenge
Anonymity: a more formal definition

Phase 2: picking identities and receiving a challenge

\[ b \leftarrow \{0, 1\} \]

\[ m, i_0, i_1 \]

\[ pk, \{sk_i\} \]
Anonymity: a more formal definition

Phase 2: picking identities and receiving a challenge

\[ \sigma = \text{Sign}(sk_{ib}, m) \]

\[ b \leftarrow \{0, 1\} \]
Anonymity: a more formal definition

\[ b \leftarrow \{0,1\} \]

\[ \text{pk, } \{\text{sk}_i\}, \sigma = \text{Sign}(\text{sk}_{ib}, m) \]
Anonymity: a more formal definition

Phase 3: getting to see who signed which messages (again)

\[ b \leftarrow \{0,1\} \]

\[ \text{pk}, \{\text{sk}_i\}, \sigma = \text{Sign}(\text{sk}_{ib}, m) \]
Anonymity: a more formal definition

Phase 3: getting to see who signed which messages (again)

\[ b \leftarrow \{0, 1\} \]

\[ \text{pk}, \{\text{sk}_i\}, \sigma = \text{Sign}\left(\text{sk}_{ib}, m\right) \]
Anonymity: a more formal definition

Phase 3: getting to see who signed which messages (again)
Anonymity: a more formal definition

Phase 3: getting to see who signed which messages (again)
Anonymity: a more formal definition

\[ b \leftarrow \{0, 1\} \]

\[ \text{pk, } \{\text{sk}_i\}, \sigma = \text{Sign}(\text{sk}_{ib}, m) \]
Anonymity: a more formal definition

Phase 4: guessing the bit $b$

\[ b \leftarrow \{0, 1\} \]

\[ (pk, \{sk_i\}, \sigma = \text{Sign}(sk_{ib}, m)) \]

**Phase 4:** guessing the bit $b$
Anonymity: a more formal definition

Phase 4: guessing the bit $b$

\[ b \leftarrow \{0, 1\} \]

\[ \text{pk, \{sk}_i\text{\}, } \sigma = \text{Sign}(\text{sk}_{ib}, m) \]

Phase 4: guessing the bit $b$
Anonymity: a more formal definition

We say that A wins at G if $b = b'$

$pk, \{sk_i\}, \sigma = \text{Sign}(sk_{ib}, m)$

Phase 4: guessing the bit $b$
Anonymity: a more formal definition

Phase 4: guessing the bit $b$

$b \leftarrow \{0,1\}$

Say that scheme is anonymous if the probability that $A$ wins at $G$ is very small (negligible)

We say that $A$ wins at $G$ if $b = b'$

$pk, \{sk_i\}, \sigma = \text{Sign}(sk_{ib}, m)$

Phase 4: guessing the bit $b$
Traceability: a more formal definition
Traceability: a more formal definition
Traceability: a more formal definition

Phase 1: getting to pick a corrupt coalition
Traceability: a more formal definition

Phase 1: getting to pick a corrupt coalition
Traceability: a more formal definition

Phase 1: getting to pick a corrupt coalition
Traceability: a more formal definition

Phase 1: getting to pick a corrupt coalition
Traceability: a more formal definition

\[ \text{pk, msk, } \{\text{sk}_i\} \leftarrow \text{KeyGen}(1^k, 1^n) \]

*Phase 1*: getting to pick a corrupt coalition
Traceability: a more formal definition

\[ \text{pk, msk, } \{\text{sk}_i\} \leftarrow \text{KeyGen}(1^k, 1^n) \]

Phase 1: getting to pick a corrupt coalition
Traceability: a more formal definition

Phase 1: getting to pick a corrupt coalition
Traceability: a more formal definition

Phase 1: getting to pick a corrupt coalition
Traceability: a more formal definition

Phase 1: getting to pick a corrupt coalition
Traceability: a more formal definition

Phase 1: getting to pick a corrupt coalition
Traceability: a more formal definition

Phase 1: getting to pick a corrupt coalition
Traceability: a more formal definition

Phase 2: outputting a forgery
Traceability: a more formal definition

Phase 2: outputting a forgery
Traceability: a more formal definition

Phase 2: outputting a forgery
Traceability: a more formal definition

Phase 2: outputting a forgery

\[ \text{Sign}(\text{sk}_i, m, \sigma) \]

\[ m, \sigma \]

\[ \text{pk}, \text{msk} \]

\[ C \]
We say that A wins at G if \(\text{Verify}(pk,\sigma,m) = 1\) and:

1. \(\exists i \text{ s.t. } \text{Trace}(msk,\sigma,m) = i\),
2. \(i \notin C\), and
3. A did not query oracle on \((i,m)\)
Traceability: a more formal definition

We say that A wins at G if Verify(pk, σ, m) = 1 and:
(1) \( \exists i \) s.t. \( \text{Trace}(msk, σ, m) = i \), (2) \( i \not\in C \), and (3) A did not query oracle on \((i, m)\)

Say that scheme is traceable if the probability that A wins at G is very small (i.e., negligible)
Supporting dynamic groups

Back in real-world application: what if someone buys a car?
Supporting dynamic groups

Back in real-world application: what if someone buys a car?

So we can also support dynamic groups in which users join over time

- Replace $\text{KeyGen}(1^k,1^n)$ with $\text{Setup}(1^k)$ (just outputs msk and pk)

- Add $\text{Join()} \leftrightarrow \text{Enroll}(\text{msk})$ protocol for group master to hand out keys as members join
Supporting dynamic groups

Back in real-world application: what if someone buys a car?

So we can also support dynamic groups in which users join over time

- Replace $\text{KeyGen}(1^k, 1^n)$ with $\text{Setup}(1^k)$ (just outputs msk and pk)

- Add $\text{Join}() \leftrightarrow \text{Enroll}(\text{msk})$ protocol for group master to hand out keys as members join

In practice, this approach could be emulated by a group master who simply runs $\text{KeyGen}(1^k, 1^N)$ for some $N >> n$, stockpiles extra keys for later
Using group managers instead of masters

Now, we have group manager who doesn’t know your secret key

So $\text{Join()} \leftrightarrow \text{Enroll(msk)}$ is a secure two-party computation at the end of which the member learns their secret key and nothing else, and the group manager learns nothing (except that the member successfully enrolled)
Using group managers instead of masters

Now, we have group manager who **doesn’t know your secret key**

So $\text{Join()} \leftrightarrow \text{Enroll}(msk)$ is a secure two-party computation at the end of which the member learns their secret key and nothing else, and the group manager learns nothing (except that the member successfully enrolled)

Now it makes sense to split tracing capability, $\text{Setup}(1^k)$ will output $msk$ used for enrollment, $pk$ used as group public key, and $tk$ used as tracing key
Using group managers instead of masters

Now, we have group manager who doesn’t know your secret key

So $\text{Join}() \leftrightarrow \text{Enroll}(\text{msk})$ is a secure two-party computation at the end of which the member learns their secret key and nothing else, and the group manager learns nothing (except that the member successfully enrolled)

Now it makes sense to split tracing capability, $\text{Setup}(1^k)$ will output $\text{msk}$ used for enrollment, $\text{pk}$ used as group public key, and $\text{tk}$ used as tracing key

We can further talk about notions of non-frameability, in which corrupt coalition might also involve the group manager
Supporting revocation

What if someone publishes my secret key on the internet?
Supporting revocation

What if someone publishes my secret key on the internet?

We need a method to revoke member privileges; allow certain members to continue signing on behalf of the group but block others from doing so.
Supporting revocation

What if someone publishes my secret key on the internet?

We need a method to revoke member privileges; allow certain members to continue signing on behalf of the group but block others from doing so.

This is often accomplished using a revocation list (RL)

- In verifier-local revocation, RL is sent to all verifiers, who then perform some additional checks using $\text{Verify}(pk, RL, \sigma, m)$

- We could also have remaining signers update their keys to match some updated public key using $\text{KeyUpdate}(pk', pk, RL, sk_i) \rightarrow sk_i'$
How do we evaluate group signature schemes?

- **Efficiency**: want really fast Sign and Verify

- **Size of the signatures**: want them to be independent of the group size

- **Security**: want highest level of security (CCA-style anonymity, full traceability)

- **Flexibility**: group manager? dynamic addition? revocation?

- **Uses reasonable assumptions**: random oracles? crazy weird-looking assumptions?
### Comparison of group signature schemes

<table>
<thead>
<tr>
<th></th>
<th>Efficiency</th>
<th>Size</th>
<th>Security</th>
<th>Flexibility</th>
<th>Assumptions</th>
<th>R.O.?</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS’97</td>
<td></td>
<td></td>
<td>CPA-A, PT</td>
<td>manager, +</td>
<td>DLP + strong RSA</td>
<td></td>
</tr>
<tr>
<td>BMW’03</td>
<td></td>
<td>C*</td>
<td>CCA-A, FT</td>
<td>master</td>
<td>TDP</td>
<td></td>
</tr>
<tr>
<td>DKNS’04</td>
<td></td>
<td></td>
<td>CPA-A, FT</td>
<td>manager, +</td>
<td>Strong RSA</td>
<td></td>
</tr>
<tr>
<td>BBS’04</td>
<td></td>
<td></td>
<td>CPA-A, FT</td>
<td>master, -</td>
<td>q-SDH + DLIN</td>
<td></td>
</tr>
<tr>
<td>BSZ’05</td>
<td></td>
<td>C*</td>
<td>CCA-A, FT</td>
<td>master, +</td>
<td>TDP</td>
<td></td>
</tr>
<tr>
<td>BW’06</td>
<td></td>
<td>Ig(N)</td>
<td>CPA-A, FT</td>
<td>master, +/-</td>
<td>CDH + SGH</td>
<td></td>
</tr>
<tr>
<td>Groth’06</td>
<td></td>
<td></td>
<td>CCA-A, FT</td>
<td>manager, +</td>
<td>DLIN</td>
<td></td>
</tr>
<tr>
<td>BW’07</td>
<td></td>
<td></td>
<td>CPA-A, FT</td>
<td>master, +/-</td>
<td>CDH + SGH + HSDH</td>
<td></td>
</tr>
</tbody>
</table>
## Comparison of group signature schemes

<table>
<thead>
<tr>
<th></th>
<th>Efficiency</th>
<th>Size</th>
<th>Security</th>
<th>Flexibility</th>
<th>Assumptions</th>
<th>R.O.?</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS’97</td>
<td></td>
<td></td>
<td>CPA-A, PT</td>
<td>manager, +</td>
<td>DLP + strong RSA</td>
<td></td>
</tr>
<tr>
<td>BMW’03</td>
<td></td>
<td>C*</td>
<td>CCA-A, FT</td>
<td>master</td>
<td>TDP</td>
<td></td>
</tr>
<tr>
<td>DKNS’04</td>
<td></td>
<td></td>
<td>CPA-A, FT</td>
<td>manager, +</td>
<td>Strong RSA</td>
<td></td>
</tr>
<tr>
<td>BBS’04</td>
<td></td>
<td></td>
<td>CPA-A, FT</td>
<td>master, -</td>
<td>q-SDH + DLIN</td>
<td></td>
</tr>
<tr>
<td>BSZ’05</td>
<td></td>
<td>C*</td>
<td>CCA-A, FT</td>
<td>master, +</td>
<td>TDP</td>
<td></td>
</tr>
<tr>
<td>BW’06</td>
<td></td>
<td>lg(N)</td>
<td>CPA-A, FT</td>
<td>master, +/-</td>
<td>CDH + SGH</td>
<td></td>
</tr>
<tr>
<td>Groth’06</td>
<td></td>
<td></td>
<td>C*</td>
<td>manager, +</td>
<td>DLIN</td>
<td></td>
</tr>
<tr>
<td>BW’07</td>
<td></td>
<td></td>
<td>CPA-A, FT</td>
<td>master, +/-</td>
<td>CDH + SGH + HSDH</td>
<td></td>
</tr>
</tbody>
</table>

- **Holy grail**: Efficient, CCA-A and FT secure, fully dynamic but short signatures, secure under mild assumptions and without random oracles

- There’s no clear winner here!
Outline

Cryptographic background

Group signatures

Ring signatures
- Intuition and motivation
- Formal definitions
- Comparison of existing schemes

Open problems
Ring signatures: why do we want them?
Ring signatures: why do we want them?

1. Bob contacts the Senate staff, requests that a group be made (for all the senators)
Ring signatures: why do we want them?

1. Bob contacts the Senate staff, requests that a group be made (for all the senators)
Ring signatures: why do we want them?

1. Bob contacts the Senate staff, requests that a group be made (for all the senators)

2. Government picks a group master/manager
Ring signatures: why do we want them?

1. Bob contacts the Senate staff, requests that a group be made (for all the senators)

2. Government picks a group master/manager
Ring signatures: why do we want them?

1. Bob contacts the Senate staff, requests that a group be made (for all the senators)

2. Government picks a group master/manager
Ring signatures: why do we want them?

1. Bob contacts the Senate staff, requests that a group be made (for all the senators)

2. Government picks a group master/manager
Ring signatures: why do we want them?

1. Bob contacts the Senate staff, requests that a group be made (for all the senators)

2. Government picks a group master/manager
Ring signatures: why do we want them?

1. Bob contacts the Senate staff, requests that a group be made (for all the senators)

2. Government picks a group master/manager

3. Government picks a tracer
Ring signatures: why do we want them?

1. Bob contacts the Senate staff, requests that a group be made (for all the senators)

2. Government picks a group master/manager

3. Government picks a tracer

4. Boss issues key for Senator #1
Ring signatures: why do we want them?

1. Bob contacts the Senate staff, requests that a group be made (for all the senators)
2. Government picks a group master/manager
3. Government picks a tracer
4. Boss issues key for Senator #1

What if Bob wants to protect his privacy unconditionally?
Ring signatures: why do we want them?
Ring signatures: why do we want them?

\[ \text{pk}_A, \text{sk}_A \quad \text{pk}_B, \text{sk}_B \]

\[ \text{pk}_C, \text{sk}_C \quad \text{pk}_D, \text{sk}_D \]
Ring signatures: why do we want them?

pk_A, sk_A  pk_B, sk_B

pk_C, sk_C  pk_D, sk_D
Ring signatures: why do we want them?

\( m, \sigma = \text{Sign}(sk_B, R, m) \)
Ring signatures: why do we want them?

\[ m, \sigma = \text{Sign}(\text{sk}_B, R, m) \]

\[ R = \text{“US senators”} \]
\[ \{\text{pk}_A, \text{pk}_B, \text{pk}_C, \text{pk}_D, \ldots\} \]
Ring signatures: why do we want them?

$m, \sigma = \text{Sign}(sk_B, R, m)$

$R = \text{“US senators”}$

$\{pk_A, pk_B, pk_C, pk_D, \ldots\}$
Ring signatures: why do we want them?

\[ m, \sigma = \text{Sign}(sk_B, R, m) \]

\[ \text{Verify}(R, \sigma, m) = 1 \ldots \text{so a senator wrote the message... but I don’t know if the Senate sanctioned it.} \]

\[ R = “\text{US senators}” \{pk_A, pk_B, pk_C, pk_D, \ldots\} \]
Ring signatures: a formal characterization

• A ring signature is a tuple of algorithms \((\text{KeyGen,Sign,Verify})\)
Ring signatures: a formal characterization

• A ring signature is a tuple of algorithms $(\text{KeyGen}, \text{Sign}, \text{Verify})$

• $\text{KeyGen}(1^k)$: outputs public key $pk$ and secret key $sk$
Ring signatures: a formal characterization

- A ring signature is a tuple of algorithms \((\text{KeyGen},\text{Sign},\text{Verify})\)

- \(\text{KeyGen}(1^k)\): outputs public key \(pk\) and secret key \(sk\)

- \(\text{Sign}(sk_i,R,m)\): outputs signature \(\sigma\) on message \(m\)
Ring signatures: a formal characterization

- A ring signature is a tuple of algorithms \((\text{KeyGen}, \text{Sign}, \text{Verify})\)
  
  - \(\text{KeyGen}(1^k)\): outputs public key \(pk\) and secret key \(sk\)
  
  - \(\text{Sign}(sk_i, R, m)\): outputs signature \(\sigma\) on message \(m\)
  
  - \(\text{Verify}(R, \sigma, m)\): checks that \(\sigma\) is a valid signature on \(m\) formed by some member of the ring defined by \(R\) (and outputs 1 if yes and 0 if no)
Ring signature anonymity

Anonymity against full key exposure:
Ring signature anonymity

Anonymity against full key exposure:

- Phase 1: $\text{KeyGen}(1^k)$ is run $m$ times to get $\{\text{pk}_i, \text{sk}_i\}$.
Ring signature anonymity

Anonymity against full key exposure:

• Phase 1: $\text{KeyGen}(1^k)$ is run $m$ times to get $\{pk_i, sk_i\}$

• Phase 2: A gets to see $S=\{pk_i\}$, access signing oracle $\text{Sign}(.,.,.)$ that on input $(i,R,m)$ will output $\text{Sign}(ski,R,m)$ (we could have $R \not\subset S$)
Anonymity against full key exposure:

- **Phase 1**: $\text{KeyGen}(1^k)$ is run $m$ times to get $\{\text{pk}_i,\text{sk}_i\}$

- **Phase 2**: A gets to see $S=\{\text{pk}_i\}$, access signing oracle $\text{Sign}(.,.,.)$ that on input $(i,R,m)$ will output $\text{Sign}(\text{sk}_i,R,m)$ (we could have $R \not\subset S$)

- **Phase 3**: A outputs challenge $(i_0,i_1,R,m)$ (again could have $R \not\subset S$) and gets back $\text{Sign}(\text{sk}_{i_b},R,m)$ for some bit $b$ it doesn’t know
Anonymity against full key exposure:

- Phase 1: `KeyGen(1^k)` is run \( m \) times to get \( \{pk_i, sk_i\} \)

- Phase 2: A gets to see \( S=\{pk_i\} \), access signing oracle `Sign(.,.,.)` that on input \( (i,R,m) \) will output `Sign(ski,R,m)` (we could have \( R \not\subseteq S \))

- Phase 3: A outputs challenge \( (i_0,i_1,R,m) \) (again could have \( R \not\subseteq S \)) and gets back `Sign(ski_b,R,m)` for some bit \( b \) it doesn’t know

- Phase 4: A now gets to see all \( \{sk_i\} \), eventually outputs a guess bit \( b' \)
We obviously can’t consider traceability, since there is no tracer! So we instead define unforgeability against coalitions and chosen-ring attacks:
Ring signature unforgeability

We obviously can’t consider traceability, since there is no tracer! So we instead define unforgeability against coalitions and chosen-ring attacks:

- Phase 1: \( \text{KeyGen}(1^k) \) is run \( m \) times to get \( \{pk_i, sk_i\} \).
Ring signature unforgeability

We obviously can’t consider traceability, since there is no tracer! So we instead define unforgeability against coalitions and chosen-ring attacks:

• Phase 1: $\text{KeyGen}(1^k)$ is run $m$ times to get $\{pk_i, sk_i\}$

• Phase 2: A gets to see $S=\{pk_i\}$ and has access to two oracles: one that, on input $(i, R, m)$ will output $\text{Sign}(sk_i, R, m)$ (we could have $R \subset S$), and the other that, on input $i$, will give A $sk_i$ and consider User i “corrupted”
Ring signature unforgeability

We obviously can’t consider traceability, since there is no tracer! So we instead define unforgeability against coalitions and chosen-ring attacks:

• Phase 1: $\text{KeyGen}(1^k)$ is run $m$ times to get $\{\text{pk}_i, \text{sk}_i\}$.

• Phase 2: A gets to see $S=\{\text{pk}_i\}$ and has access to two oracles: one that, on input $(i, R, m)$ will output $\text{Sign}(\text{sk}_i, R, m)$ (we could have $R \not\subset S$), and the other that, on input $i$, will give A $\text{sk}_i$ and consider User $i$ “corrupted”.

• Phase 3: A at some point has to output a successful forgery $(R^*, \sigma^*, m^*)$ (i.e., such that $\text{Verify}(R^*, \sigma^*, m^*) = 1$)
How do we evaluate ring signature schemes?

• **Efficiency:** want really fast Sign and Verify

• **Size of the signatures:** want them to be independent of the ring size

• **Security:** want highest level of security (full anonymity, full unforgeability)

• **Flexibility:** can users pick their own signature schemes?

• **Uses reasonable assumptions:** random oracles? crazy weird-looking assumptions?
### Comparison of ring signature schemes

<table>
<thead>
<tr>
<th></th>
<th>Efficiency</th>
<th>Size</th>
<th>Security</th>
<th>Flexibility</th>
<th>Assumptions</th>
<th>R.O.?</th>
</tr>
</thead>
<tbody>
<tr>
<td>RST’01</td>
<td>linear</td>
<td></td>
<td>UFA</td>
<td></td>
<td>TDP</td>
<td></td>
</tr>
<tr>
<td>DKNS’04</td>
<td></td>
<td>C</td>
<td>CFA</td>
<td></td>
<td>Strong RSA</td>
<td></td>
</tr>
<tr>
<td>BKM’06</td>
<td>linear</td>
<td></td>
<td>CFA, FU</td>
<td></td>
<td>TDP</td>
<td></td>
</tr>
<tr>
<td>SW’07</td>
<td>linear</td>
<td></td>
<td>CFA, FU</td>
<td></td>
<td>CDH + SGH</td>
<td></td>
</tr>
<tr>
<td>Boyen’07</td>
<td>linear</td>
<td></td>
<td>UFA, PU</td>
<td></td>
<td>Poly-SDH</td>
<td></td>
</tr>
</tbody>
</table>
Comparison of ring signature schemes

<table>
<thead>
<tr>
<th></th>
<th>Efficiency</th>
<th>Size</th>
<th>Security</th>
<th>Flexibility</th>
<th>Assumptions</th>
<th>R.O.?</th>
</tr>
</thead>
<tbody>
<tr>
<td>RST’01</td>
<td>linear</td>
<td>UFA</td>
<td></td>
<td></td>
<td>TDP</td>
<td></td>
</tr>
<tr>
<td>DKNS’04</td>
<td>C</td>
<td>CFA</td>
<td></td>
<td></td>
<td>Strong RSA</td>
<td></td>
</tr>
<tr>
<td>BKM’06</td>
<td>linear</td>
<td>CFA, FU</td>
<td></td>
<td></td>
<td>TDP</td>
<td></td>
</tr>
<tr>
<td>SW’07</td>
<td>linear</td>
<td>CFA, FU</td>
<td></td>
<td></td>
<td>CDH + SGH</td>
<td></td>
</tr>
<tr>
<td>Boyen’07</td>
<td>linear</td>
<td>UFA, PU</td>
<td></td>
<td></td>
<td>Poly-SDH</td>
<td></td>
</tr>
</tbody>
</table>

- **Holy grail**: Efficient, CFA and FU secure, flexible but short signatures, secure under mild assumptions and without random oracles

- Again, there’s no clear winner!
Outline

- Cryptographic background
- Group signatures
- Ring signatures
- Open problems
Open problems for group signatures

• We already saw this “holy grail” of a scheme that is efficient, CCA-A and FT secure, fully dynamic but short signatures, secure under mild assumptions and without random oracles

32
Open problems for group signatures

• We already saw this “holy grail” of a scheme that is efficient, CCA-A and FT secure, fully dynamic but short signatures, secure under mild assumptions and without random oracles

• Also would be nice to see more applications in the real world (just DAA and VSC for now)
Open problems for group signatures

• We already saw this “holy grail” of a scheme that is efficient, CCA-A and FT secure, fully dynamic but short signatures, secure under mild assumptions and without random oracles

• Also would be nice to see more applications in the real world (just DAA and VSC for now)

• Generic construction for a fully dynamic scheme (i.e., one that supports revocation)
Open problems for group signatures

- We already saw this “holy grail” of a scheme that is efficient, CCA-A and FT secure, fully dynamic but short signatures, secure under mild assumptions and without random oracles

- Also would be nice to see more applications in the real world (just DAA and VSC for now)

- Generic construction for a fully dynamic scheme (i.e., one that supports revocation)

- Better definitions and formalizations for revocation
Open problems for ring signatures

• Find a real-world application!!
Open problems for ring signatures

• **Find a real-world application!!**

• Again, achieve holy grail of scheme that is efficient, CFA and FU secure, flexible but short signatures, secure under mild assumptions and without random oracles
Open problems for ring signatures

• Find a real-world application!!

• Again, achieve holy grail of scheme that is efficient, CFA and FU secure, flexible but short signatures, secure under mild assumptions and without random oracles

• Figure out way to overcome this linear-sized signature barrier (ideally without random oracles)
Open problems for ring signatures

• **Find a real-world application!!**

• Again, achieve holy grail of scheme that is efficient, CFA and FU secure, flexible but short signatures, secure under mild assumptions and without random oracles

• Figure out way to overcome this linear-sized signature barrier (ideally without random oracles)

• Can we even achieve flexibility using a non-generic construction?
Open problems for ring signatures

• **Find a real-world application!!**

• Again, achieve holy grail of scheme that is efficient, CFA and FU secure, flexible but short signatures secure under mild assumptions and without random oracles

• Figure out way to overcome this linear-sized signature barrier (ideally without random oracles)

• Can we even achieve flexibility using a non-generic construction?

Any questions?