On the Security of Two-Round Multi-Signatures

Manu Drijvers\textsuperscript{1}, Kasra Edalatnejad\textsuperscript{2}, Bryan Ford\textsuperscript{2}, Eike Kiltz\textsuperscript{3}, Julian Loss\textsuperscript{3}, Gregory Neven\textsuperscript{1}, Igors Stepanovs\textsuperscript{4}

\textsuperscript{1}DFINITY, \textsuperscript{2}EPFL, \textsuperscript{3}Ruhr-University Bochum, \textsuperscript{4}UCSD
Multi-signatures

\[ (pk_1, sk_1) \leftarrow Kg \]
\[ (pk_2, sk_2) \leftarrow Kg \]
\[ (pk_3, sk_3) \leftarrow Kg \]

Sign((pk_1, pk_2, pk_3), sk_1, m) \leftrightarrow \text{Sign}( (pk_1, pk_2, pk_3), sk_2, m) \leftrightarrow \text{Sign}( (pk_1, pk_2, pk_3), sk_3, m)

\[ \rightarrow \sigma \]
\[ \rightarrow \sigma \]
\[ \rightarrow \sigma \]

Verify((pk_1, pk_2, pk_3), m, \sigma) = 1

Every signer must agree to sign m

**Goal:** short signature (preferably \(\approx\) single signature, efficiently verifiable definitely \(<\ll N\) signatures)**
Multi-signatures

$$(pk_1, sk_1) \leftarrow Kg$$

Sign$$( (pk_1, pk_2, pk_3), sk_1, m) \leftarrow \text{Sign}( (pk_1, pk_2, pk_3), sk_2, m) \leftarrow \text{Sign}( (pk_1, pk_2, pk_3), sk_3, m)$$

$$\rightarrow \sigma$$

Key aggregation: $$apk \leftarrow \text{KAgg}(pk_1, pk_2, pk_3)$$

Verify$$(apk, m, \sigma) = 1$$

Every signer must agree to sign m

**Goal:** short signature (preferably $\approx$ single signature, definitely $<< N$ signatures)

efficiently verifiable
Applications of multi-signatures

- Improve Bitcoin throughput / reduce blockchain size
  - "multisig" transactions as small as other transactions
  - Reduce size of multi-input multi-output transactions

- Collective signing by co-thorities (e.g., CoSi [STV+16])

- Distributed random beacons (e.g., RandHound [SJK+17])

- Block certification in proof-of-stake / permissioned blockchains
  - e.g., Dfinity, OmniLedger, Ziliqa, Harmony, Algorand, ...
Existing multi-signatures
Schnorr signatures

\[ \text{pk} = g^{sk} \]
\[ r \leftarrow_R Z_q \]
\[ t \leftarrow g^r \]
\[ c \leftarrow H(t,m) \]
\[ s \leftarrow r + c \cdot sk \mod q \]
\[ \sigma \leftarrow (c, s) \]

Verification:
\[ c = H(g^s \cdot \text{pk}^{-c}, m) \]

Efficient & Provably secure
- under discrete-log assumption
- in the random-oracle model: model hash function as ideal random function
"Plain" Schnorr multi-signatures

\[
\begin{align*}
\text{pk}_1 &= g^{sk_1} \\
\quad r_1 &\gets_R Z_q \\
\quad t_1 &\gets g^{r_1} \\
\quad t &\gets t_1 \cdot t_2 \cdot t_3 \\
\quad c &\gets H(t, m) \\
\quad s_1 &\gets r_1 + c \cdot sk_1 \pmod{q} \\
\quad s &\gets s_1 + s_2 + s_3 \pmod{q} \\
\quad \sigma &\gets (c, s)
\end{align*}
\]

\[
\begin{align*}
\text{pk}_2 &= g^{sk_2} \\
\quad r_2 &\gets_R Z_q \\
\quad t_2 &\gets g^{r_2} \\
\quad t &\gets t_1 \cdot t_2 \cdot t_3 \\
\quad c &\gets H(t, m) \\
\quad s_2 &\gets r_2 + c \cdot sk_2 \pmod{q} \\
\quad s &\gets s_1 + s_2 + s_3 \pmod{q} \\
\quad \sigma &\gets (c, s)
\end{align*}
\]

\[
\begin{align*}
\text{pk}_3 &= g^{sk_3} \\
\quad r_3 &\gets_R Z_q \\
\quad t_3 &\gets g^{r_3} \\
\quad t &\gets t_1 \cdot t_2 \cdot t_3 \\
\quad c &\gets H(t, m) \\
\quad s_3 &\gets r_3 + c \cdot sk_3 \pmod{q} \\
\quad s &\gets s_1 + s_2 + s_3 \pmod{q} \\
\quad \sigma &\gets (c, s)
\end{align*}
\]

\[
\begin{align*}
\text{apk} &\gets \text{pk}_1 \cdot \text{pk}_2 \cdot \text{pk}_3 \\
\text{Check} &\quad c = H(g^s \cdot \text{apk}^{-c}, m)
\end{align*}
\]
Problem 1: Rogue-key attacks

\[
\begin{align*}
\text{pk}_1 &= g^{sk_1} \\
\text{pk}_2 &= g^{sk_2} / \text{pk}_1 \\
\text{apk} &= \text{pk}_1 \cdot \text{pk}_2 = g^{sk_2}
\end{align*}
\]

can compute signatures under apk by himself!

Known remedies:

• Per-signer challenges [BN06]

• Proofs of possession added to pk [RY07,BCJ08]

• MuSig key aggregation: \( \text{apk} \leftarrow \prod \text{pk}_i^{H(pki, \{\text{pk}_1, ..., \text{pk}_N\})} \) [MPSW18]
Problem 2: Signature simulation

\[ \begin{align*}
    c, s_1 & \leftarrow_R \mathbb{Z}_q \\
    t_1 & \leftarrow g^{s_1} \cdot (p^k_1)^c \\
    t & \leftarrow t_1 \cdot t_2 \\
    c & \leftarrow H(t,m)
\end{align*} \]

Standard Schnorr proof technique does not work
(cannot program random oracle, because adversary knows \( t \) before simulator does)
## Multi-signatures from discrete logarithms

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Rounds</th>
<th>Rogue keys</th>
<th>Signature simulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>BN [BN06]</td>
<td>3</td>
<td>per-signer challenges</td>
<td>preliminary round (H(t_i))</td>
</tr>
<tr>
<td>BCJ-1 [BCJ08]</td>
<td>2</td>
<td>per-signer challenges</td>
<td>homomorphic equivocable (HE) commitments</td>
</tr>
<tr>
<td>BCJ-2 [BCJ08]</td>
<td>2</td>
<td>proofs of possession</td>
<td>witness-indistinguishable keys</td>
</tr>
<tr>
<td>MWLD [MWLD10]</td>
<td>2</td>
<td>proofs of possession</td>
<td>(no security proof)</td>
</tr>
<tr>
<td>CoSi [STV+16]</td>
<td>2</td>
<td>proofs of possession</td>
<td>per-message HE commitments</td>
</tr>
<tr>
<td>MuSig 1 [MPSW18a]</td>
<td>2</td>
<td>MuSig key aggregation</td>
<td>DL oracle in one-more DL assumption</td>
</tr>
<tr>
<td>mBCJ [this work]</td>
<td>2</td>
<td>proofs of possession</td>
<td>per-message HE commitments</td>
</tr>
<tr>
<td>BDN-DL, MuSig-2 [BDN18, MPSW19]</td>
<td>3</td>
<td>MuSig key aggregation</td>
<td>preliminary round (H(t_i))</td>
</tr>
<tr>
<td>BDN-DLpop [BDN18]</td>
<td>3</td>
<td>proofs of possession</td>
<td>preliminary round (H(t_i))</td>
</tr>
<tr>
<td>BLS [Bol03,RY07]</td>
<td>1</td>
<td>proofs of possession</td>
<td>pairings</td>
</tr>
<tr>
<td>BDN-P [BDN18]</td>
<td>1</td>
<td>MuSig key aggregation</td>
<td>pairings</td>
</tr>
</tbody>
</table>
Attacks and non-provability
Wagner’s generalized birthday attack [W02]

$k$-sum problem in $\mathbb{Z}_q$:

Given $k$ lists of random elements in $\mathbb{Z}_q$

Find $(c_1, \ldots, c_k)$ in lists such that $c_1 + \ldots + c_k = 0 \mod q$

Subexponential solution: Solved for $k = 2^{\sqrt{n}}$ in time $O(2^{2\sqrt{n}})$ where $n = |q|$
Application to “plain” Schnorr and CoSi

- sk only appears in signature in $s = r + c * sk$, with $c = H(g^r, m)$
- If we have signatures with $c_1 + ... + c_{k-1} = H(t^*, m)$, we can forge a signature on $m^*$!

$$t_1 \leftarrow g^{r_1}$$

$$H(*, m)$$

$$H(*, m)$$

...  

$$t_{k-1} \leftarrow g^{r_{k-1}}$$

$$H(*, m)$$

$$H(*, m)$$

$$H(*, m)$$

$$t^* \leftarrow t_1 \cdot ... \cdot t_{k-1}$$

$$-H(t^*, m_1)$$

$$-H(t^*, m_2)$$

$$...$$

$$-H(t^*, m_L)$$

$$c_1 + ... + c_{k-1} = c^*$$
Attacks on two-round multi-signature schemes

• Attack applies to all previously* known two-round schemes
  • BCJ-1 and BCJ-2
  • MWLD
  • CoSi
  • MuSig-1

• Sub-exponential but practical
  (for 256-bit q)
  • 15 parallel signing queries: $2^{62}$ steps
  • 127 parallel signing queries: $2^{45}$ steps

• Prevented by increasing $|q|$
  ...any hope for provable (asymptotic) security?

* before first version of this paper
Non-provability of two-round schemes

**Theorem:** One-more discrete logarithm problem is hard

\[ \Downarrow \]

BCJ/MWLD/CoSi/MuSig-1 cannot be proved secure under one-more discrete logarithm

(through algebraic black-box reductions in random-oracle model)

Essentially excludes all known proof techniques (including rewinding) under likely assumptions.

Subtle flaws in proofs of BCJ/MWLD/MuSig-1

(CoSi was never proved secure)
Secure schemes
Modified BCJ multi-signature

- 2 round, secure under discrete logarithm, same efficiency as BCJ
- Large scale deployment:
  - 16,384 signers generate signature within 2 seconds
  - 20% bandwidth, 75% computation increase compared to CoSi (plain schnorr)

**Fig. 4.** Comparing end-to-end latency of CoSi and mBCJ signing with varying amounts of signers.

**Fig. 5.** Bandwidth consumption (sent and received combined) of CoSi and mBCJ with varying amounts of signers.

**Fig. 6.** CPU time (User + System) of CoSi and mBCJ with varying amounts of signers.
Other secure schemes

• Three-round scheme [BDN18, MPSW19]
  • Secure under discrete-log assumption

• Non-interactive scheme from BLS [BLS01, Bol03, RY07, BDN18]
  • Smaller signatures
  • Non-interactive aggregation
  • Requires bilinear pairings
Lessons learned
Lessons learned

• Cryptographic schemes need security proofs
  • Don’t drop steps that look like they’re “just to make the proof work”

• Security proofs must be reviewed
  • Proofs can be subtle, especially with rewinding arguments
  • Tool support for checking proofs?

• Provable security is not perfect, but best tool we have
Thank you!

ia.cr/2018/417
References

[BN06] Bellare, Neven: Multi-signatures in the plain public-Key model and a general forking lemma. CCS 2006
[BCJ08] Bagherzandi, Cheon, Jarecki: Multisignatures secure under the discrete logarithm assumption and a generalized forking lemma. CCS 2008
[BDN18] Boneh, Drijvers, Neven: Compact Multi-signatures for Smaller Blockchains. ASIACRYPT 2018
Modified BCJ multi-signatures

\[ \text{pk}_i = g^{sk_i} + \text{PoP} \]

\[ (g_2, h_1, h_2) \leftarrow H'(m) \]
\[ r, \alpha_1, \alpha_2 \leftarrow R \mathbb{Z}_q \]
\[ t_{i,1} \leftarrow g_1^{\alpha_1} h_1^{\alpha_2} \]
\[ t_{i,2} \leftarrow g_2^{\alpha_1} h_2^{\alpha_2} g_1^{r} \]
\[ t_1 \leftarrow \Pi t_{i,1} ; t_2 \leftarrow \Pi t_{i,2} \]
\[ c \leftarrow H(t_1, t_2, \Pi \text{pk}_i, m) \]
\[ s_i \leftarrow r + c \cdot sk_i + \Sigma s_i \text{ mod q} \]
\[ s \leftarrow \Sigma s_i \text{ mod q} \]
\[ \alpha_1 \leftarrow \Sigma \alpha_{i,1} \text{ mod q} \]
\[ \alpha_2 \leftarrow \Sigma \alpha_{i,2} \text{ mod q} \]
\[ \sigma \leftarrow (t_1, t_2, s, \alpha_1, \alpha_2) \]

KAgg: Check PoPs, \( \text{apk} \leftarrow \Pi \text{pk}_i \)

Verify: \( c \leftarrow H(t_1, t_2, \text{apk}, m) \)

Check \( t_1 = g_1^{\alpha_1} h_1^{\alpha_2} \)

and \( t_2 = g_2^{\alpha_1} h_2^{\alpha_2} g_1^{s} \text{ apk}^{-c} \)

Efficiency

Sign: \(1 \text{ mexp}^2 + 1 \text{ mexp}^3\)
plain Schnorr: 1 exp

Verify: \(3 \text{ mexp}^2\)
plain Schnorr: 1 mexp²

Signature size: 160 B
plain Schnorr: 64 B
Application to “plain” Schnorr and CoSi

Query on $m_1$

\[
\begin{align*}
    r_1 &\leftarrow_R Z_q \\
    t_1 &\leftarrow g^{r_1} \\
    c_1 &\leftarrow H(t_1, m_1) \\
    s_1 &\leftarrow r_1 + c_1 \cdot sk \\
    \sigma_1 &\leftarrow (c_1, s_1)
\end{align*}
\]

Query on $m_2$

\[
\begin{align*}
    r_2 &\leftarrow_R Z_q \\
    t_2 &\leftarrow g^{r_2} \\
    c_2 &\leftarrow H(t_2, m_2) \\
    s_2 &\leftarrow r_2 + c_2 \cdot sk \\
    \sigma_2 &\leftarrow (c_2, s_2)
\end{align*}
\]

Forgery on $m_3$

\[
\begin{align*}
    t_3 &\leftarrow t_1 \cdot t_2 \\
    c_3 &\leftarrow H(t_3, m_3) \text{ such that } c_3 = c_1 + c_2 \\
    s_3 &\leftarrow s_1 + s_2 \\
    \sigma_3 &\leftarrow (c_3, s_3)
\end{align*}
\]
Lessons learned

• Provable security! 😐
• Review security proofs! 😐

• Proofs can be subtle, especially forking
• Tool support for checking proofs?
• Don’t drop steps that look like they’re “just to make the proof work”

• Provable security is not perfect, but best tool we have
Application to “plain” Schnorr and CoSi

\[ t_1 \leftarrow g^{r_1} \]
\[ \ldots \]
\[ t_{k-1} \leftarrow g^{r_{k-1}} \]
\[ t^* \leftarrow t_1 \cdot \ldots \cdot t_{k-1} \]

\[ S_1 \leftarrow r_1 + c_1 \cdot s^k \mod q \]
\[ S_{k-1} \leftarrow r_{k-1} + c_{k-1} \cdot s^k \mod q \]
\[ C_1 + \ldots + C_{k-1} = c^* \mod q \]
\[ S^* \leftarrow S_1 + \ldots + S_{k-1} \mod q \]

\[ pk^* = g^{sk^*} \]
\[ g^{s^*} = g^{\sum s_i} = g^{\Sigma r_i + \sum c_i \cdot s^k} = \prod t_i \cdot pk^* c^* \]
\[ t \cdot pk^* c^* \]
## Multi-signatures from discrete logarithms

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Rounds</th>
<th>Rogue Keys</th>
<th>Signature simulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>BN [BN06]</td>
<td>3</td>
<td>per-signer challenges</td>
<td>preliminary round $H(t_i)$</td>
</tr>
<tr>
<td>BCJ-1 [BCJ08]</td>
<td>2</td>
<td>per-signer challenges</td>
<td>homomorphic equivocable (HE) com.</td>
</tr>
<tr>
<td>BCJ-2 [BCJ08]</td>
<td>2</td>
<td>proofs of possession</td>
<td>homomorphic equivocable (HE) com.</td>
</tr>
<tr>
<td>MWLD [MWLD10]</td>
<td>2</td>
<td>per-signer challenges</td>
<td>witness indistinguishable keys</td>
</tr>
<tr>
<td>CoSi [STV+16]</td>
<td>2</td>
<td>proofs of possession</td>
<td>(no security proof)</td>
</tr>
<tr>
<td>MuSig1 [MPSW18]</td>
<td>2</td>
<td>MuSig key aggregation</td>
<td>DL oracle in one-more DL assumption</td>
</tr>
<tr>
<td>mBCJ (this work)</td>
<td>2</td>
<td>proofs of possession</td>
<td>per-message HE commitments</td>
</tr>
<tr>
<td>BDN-DL, MuSig2</td>
<td>3</td>
<td>MuSig key aggregation</td>
<td>preliminary round $H(t_i)$</td>
</tr>
<tr>
<td>[BDN18, MPSW19]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BDN-DLpop [BDN18]</td>
<td>3</td>
<td>proofs of possession</td>
<td>preliminary round $H(t_i)$</td>
</tr>
<tr>
<td>BLS-PoP [RY07]</td>
<td>1</td>
<td>proofs of possession</td>
<td>pairings</td>
</tr>
<tr>
<td>BDN-P [BDN18]</td>
<td>1</td>
<td>MuSig key aggregation</td>
<td>pairings</td>
</tr>
</tbody>
</table>