Using Gröbner Basis Techniques to Construct Optimal Structure Preserving Signatures

Mehdi Tibouchi

NTT Secure Platform Laboratories

SIAM AG15, Minisymposium on Pairings in Cryptography, 2015–08–05
Outline

Structure-preserving signatures

Optimal Type I SPS

Optimal Type II SPS

Conclusion
Outline

Structure-preserving signatures

Optimal Type I SPS

Optimal Type II SPS

Conclusion
Structure-Preserving Signatures

- Fix as public parameter a non-degenerate bilinear pairing $e: \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ between cyclic groups of prime order $p$.

- A signature scheme $\text{Sig} = (\text{KeyGen}, \text{Sign}, \text{Verify})$ is called structure-preserving wrt those parameters when:
  - public keys, messages and signatures are all tuples of elements of $\mathbb{G}_1$ and $\mathbb{G}_2$;
  - signature verification is carried out by checking a number of pairing product equations (PPEs):
    $$\prod_{i,j} e(G_i, G_j) = Z.$$

- The security notion is EUF-CMA as usual.

- We can consider SPS both over Type I ($\mathbb{G}_1 = \mathbb{G}_2$), Type II (efficient isomorphism $\psi: \mathbb{G}_2 \rightarrow \mathbb{G}_1$ but not conversely) and Type III (no efficient isomorphism either way) bilinear groups.
Structure-Preserving Signatures

- Fix as public parameter a non-degenerate bilinear pairing
  \( e: \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T \) between cyclic groups of prime order \( p \).
- A signature scheme \( \text{Sig} = (\text{KeyGen}, \text{Sign}, \text{Verify}) \) is called structure-preserving wrt those parameters when:
  - public keys, messages and signatures are all tuples of elements of \( \mathbb{G}_1 \) and \( \mathbb{G}_2 \);
  - signature verification is carried out by checking a number of pairing product equations (PPEs):
    \[
    \prod_{i,j} e(G_i, G_j) = Z.
    \]
- The security notion is EUF-CMA as usual.
- We can consider SPS both over Type I \( (\mathbb{G}_1 = \mathbb{G}_2) \), Type II (efficient isomorphism \( \psi: \mathbb{G}_2 \rightarrow \mathbb{G}_1 \) but not conversely) and Type III (no efficient isomorphism either way) bilinear groups.
Structure-Preserving Signatures

- Fix as public parameter a non-degenerate bilinear pairing $e: G_1 \times G_2 \rightarrow G_T$ between cyclic groups of prime order $p$.
- A signature scheme $\text{Sig} = (\text{KeyGen}, \text{Sign}, \text{Verify})$ is called structure-preserving wrt those parameters when:
  - public keys, messages and signatures are all tuples of elements of $G_1$ and $G_2$;
  - signature verification is carried out by checking a number of pairing product equations (PPEs):

$$\prod_{i,j} e(G_i, G_j) = Z.$$

- The security notion is EUF-CMA as usual.
- We can consider SPS both over Type I ($G_1 = G_2$), Type II (efficient isomorphism $\psi: G_2 \rightarrow G_1$ but not conversely) and Type III (no efficient isomorphism either way) bilinear groups.
Structure-Preserving Signatures

- Fix as public parameter a non-degenerate bilinear pairing \( e: G_1 \times G_2 \rightarrow G_T \) between cyclic groups of prime order \( p \).
- A signature scheme \( \text{Sig} = (\text{KeyGen}, \text{Sign}, \text{Verify}) \) is called structure-preserving wrt those parameters when:
  - public keys, messages and signatures are all tuples of elements of \( G_1 \) and \( G_2 \);
  - signature verification is carried out by checking a number of pairing product equations (PPEs):
    \[
    \prod_{i,j} e(G_i, G_j) = Z.
    \]
- The security notion is EUF-CMA as usual.
- We can consider SPS both over Type I \((G_1 = G_2)\), Type II (efficient isomorphism \( \psi: G_2 \rightarrow G_1 \) but not conversely) and Type III (no efficient isomorphism either way) bilinear groups.
Why Structure-Preserving Signatures?

- Main motivation: essentially one efficient construction of NIZK proofs, the Groth-Sahai proof system, for pairing product equations. SPS can be combined with GS proofs conveniently.
  - Many resulting applications: round-optimal blind signatures, traceable signatures, group encryption, proxy signatures, delegatable credentials...
  - SPS are part of “structure-preserving cryptography” (with SP encryption, SP commitments, etc.) where all keys, messages, signatures, ciphertexts, etc. are tuples of group elements. Easy to compose such schemes.
  - SPS have a simple algebraic definition: nice target for mathematical reasoning.
Why Structure-Preserving Signatures?

- Main motivation: essentially one efficient construction of NIZK proofs, the Groth-Sahai proof system, for pairing product equations. SPS can be combined with GS proofs conveniently.

- Many resulting applications: round-optimal blind signatures, traceable signatures, group encryption, proxy signatures, delegatable credentials...

- SPS are part of “structure-preserving cryptography” (with SP encryption, SP commitments, etc.) where all keys, messages, signatures, ciphertexts, etc. are tuples of group elements. Easy to compose such schemes.

- SPS have a simple algebraic definition: nice target for mathematical reasoning.
Why Structure-Preserving Signatures?

- Main motivation: essentially one efficient construction of NIZK proofs, the Groth-Sahai proof system, for pairing product equations. SPS can be combined with GS proofs conveniently.

- Many resulting applications: round-optimal blind signatures, traceable signatures, group encryption, proxy signatures, delegatable credentials...

- SPS are part of "structure-preserving cryptography" (with SP encryption, SP commitments, etc.) where all keys, messages, signatures, ciphertexts, etc. are tuples of group elements. Easy to compose such schemes.

- SPS have a simple algebraic definition: nice target for mathematical reasoning.
Why Structure-Preserving Signatures?

- Main motivation: essentially one efficient construction of NIZK proofs, the Groth-Sahai proof system, for pairing product equations. SPS can be combined with GS proofs conveniently.
- Many resulting applications: round-optimal blind signatures, traceable signatures, group encryption, proxy signatures, delegatable credentials...
- SPS are part of “structure-preserving cryptography” (with SP encryption, SP commitments, etc.) where all keys, messages, signatures, ciphertexts, etc. are tuples of group elements. Easy to compose such schemes.
- SPS have a simple algebraic definition: nice target for mathematical reasoning.
Bounds on SPS

- SPS are a low-level building block for more involved cryptographic protocols: important to understand their efficiency.
  - Many works devoted to obtain simpler, more efficient SPS, especially in terms of signature size and number of verification equations.
  - At the same time, a line of research has sought to establish lower bounds for the efficiency of SPS, showing that an SPS scheme cannot be secure (or cannot be proved secure) if it has fewer than a certain signature size/number of verification equations.
  - Ongoing program to obtain optimal schemes (matching lower bounds and constructions) for all pairing types.
Bounds on SPS

- SPS are a low-level building block for more involved cryptographic protocols: important to understand their efficiency.

- Many works devoted to obtain simpler, more efficient SPS, especially in terms of signature size and number of verification equations.

- At the same time, a line of research has sought to establish lower bounds for the efficiency of SPS, showing that an SPS scheme cannot be secure (or cannot be proved secure) if it has fewer than a certain signature size/number of verification equations.

- Ongoing program to obtain optimal schemes (matching lower bounds and constructions) for all pairing types.
Bounds on SPS

- SPS are a low-level building block for more involved cryptographic protocols: important to understand their efficiency.
- Many works devoted to obtain simpler, more efficient SPS, especially in terms of signature size and number of verification equations.
- At the same time, a line of research has sought to establish lower bounds for the efficiency of SPS, showing that an SPS scheme cannot be secure (or cannot be proved secure) if it has fewer than a certain signature size/number of verification equations.
- Ongoing program to obtain optimal schemes (matching lower bounds and constructions) for all pairing types.
• SPS are a low-level building block for more involved cryptographic protocols: important to understand their efficiency.

• Many works devoted to obtain simpler, more efficient SPS, especially in terms of signature size and number of verification equations.

• At the same time, a line of research has sought to establish lower bounds for the efficiency of SPS, showing that an SPS scheme cannot be secure (or cannot be proved secure) if it has fewer than a certain signature size/number of verification equations.

• Ongoing program to obtain optimal schemes (matching lower bounds and constructions) for all pairing types.
How I got involved

State of the art two years ago:

<table>
<thead>
<tr>
<th>Group type</th>
<th>Best lower bound</th>
<th>Best construction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asymmetric</td>
<td>$(#(\text{PPE}) \geq 2,</td>
<td>\Sigma</td>
</tr>
<tr>
<td>(Type III)</td>
<td>[AGHO11]</td>
<td>[AGHO11]</td>
</tr>
<tr>
<td>Symmetric</td>
<td>$(#(\text{PPE}) = 2,</td>
<td>\Sigma</td>
</tr>
<tr>
<td>(Type I)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$(\#(\text{PPE}))$: number of verification PPEs, 
$|\Sigma|$: signature size in group elements.

For asymmetric groups: optimal construction!
How I got involved

State of the art two years ago:

<table>
<thead>
<tr>
<th>Group type</th>
<th>Best lower bound</th>
<th>Best construction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asymmetric</td>
<td>#(PPE) ≥ 2,</td>
<td>Σ</td>
</tr>
<tr>
<td>(Type III)</td>
<td>[AGHO11]</td>
<td>[AGHO11]</td>
</tr>
<tr>
<td>Symmetric</td>
<td>??</td>
<td>#(PPE) = 2,</td>
</tr>
<tr>
<td>(Type I)</td>
<td>![AFGHO10]</td>
<td>[AFGHO10]</td>
</tr>
</tbody>
</table>

#(PPE): number of verification PPEs,
|Σ|: signature size in group elements.

For symmetric groups: no lower bound yet. Strong suspicion that 2 PPEs were necessary, but computations looked scary.
Outline

Structure-preserving signatures

Optimal Type I SPS

Optimal Type II SPS

Conclusion
To prove that a Type I SPS must have at least 2 verification equations:

- suppose that a Type I SPS with single verification equation exists;
- give a simple, general expression for the verification PPE;
- exhibit an explicit forgery attack: any such forgery is a solution of a (relatively large) system of quadratic equations in the discrete logs of the group elements involved.
How to establish a lower bound

To prove that a Type I SPS must have at least 2 verification equations:

- suppose that a Type I SPS with single verification equation exists;
- give a simple, general expression for the verification PPE;
- exhibit an explicit forgery attack: any such forgery is a solution of a (relatively large) system of quadratic equations in the discrete logs of the group elements involved.
How to establish a lower bound

To prove that a Type I SPS must have at least 2 verification equations:

- suppose that a Type I SPS with single verification equation exists;
- give a simple, general expression for the verification PPE;
- exhibit an explicit forgery attack: any such forgery is a solution of a (relatively large) system of quadratic equations in the discrete logs of the group elements involved.
The verification equation for our SPS is a pairing product equation (PPE) involving the following elements of $G$:

- the message $M$;
- the components of the signature $\Sigma = (S_1, \ldots, S_n)$;
- public parameters and verification key elements.

It can thus be written in its most general form as:

$$e(M, M)^a \cdot e(M, U \prod_{i=1}^{n} S_i^{b_i}) \cdot \prod_{1 \leq i, j \leq n} e(S_i, S_j)^{c_{ij}} \cdot \prod_{i=1}^{n} e(S_i, V_i) = Z,$$

for public constants $U, V_i \in G$, $Z \in G_T$ and $a, b_i, c_{ij} \in \mathbb{Z}_p$.

A bit complicated: we have a whole matrix $(c_{ij})$ describing the quadratic form in the components of the signature. Simplify?
The verification PPE

- The verification equation for our SPS is a pairing product equation (PPE) involving the following elements of $G$:
  - the message $M$;
  - the components of the signature $\Sigma = (S_1, \ldots, S_n)$;
  - public parameters and verification key elements.

- It can thus be written in its most general form as:

$$e(M, M)^a \cdot e(M, U \prod_{i=1}^{n} S_i^{b_i}) \cdot \prod_{1 \leq i, j \leq n} e(S_i, S_j)^{c_{ij}} \cdot \prod_{i=1}^{n} e(S_i, V_i) = Z,$$

for public constants $U, V_i \in G, Z \in G_T$ and $a, b_i, c_{ij} \in \mathbb{Z}_p$.

- A bit complicated: we have a whole matrix $(c_{ij})$ describing the quadratic form in the components of the signature. Simplify?
The verification equation for our SPS is a pairing product equation (PPE) involving the following elements of $\mathbb{G}$:

- the message $M$;
- the components of the signature $\Sigma = (S_1, \ldots, S_n)$;
- public parameters and verification key elements.

It can thus be written in its most general form as:

$$e(M, M)^a \cdot e(M, U \prod_{i=1}^{n} S_i^{b_i}) \cdot \prod_{1 \leq i, j \leq n} e(S_i, S_j)^{c_{ij}} \cdot \prod_{i=1}^{n} e(S_i, V_i) = Z,$$

for public constants $U, V_i \in \mathbb{G}$, $Z \in \mathbb{G}_T$ and $a, b_i, c_{ij} \in \mathbb{Z}_p$.

A bit complicated: we have a whole matrix $(c_{ij})$ describing the quadratic form in the components of the signature. Simplify?
The verification equation for our SPS is a pairing product equation (PPE) involving the following elements of $\mathbb{G}$:

- the message $M$;
- the components of the signature $\Sigma = (S_1, \ldots, S_n)$;
- public parameters and verification key elements.

It can thus be written in its most general form as:

$$e(M, M)^a \cdot e(M, U \prod_{i=1}^{n} S_i^{b_i}) \cdot \prod_{i=1}^{n} e(S_i, S_i)^{c_i} \cdot \prod_{i=1}^{n} e(S_i, V_i) = Z,$$

for public constants $U, V_i \in \mathbb{G}$, $Z \in \mathbb{G}_T$ and $a, b_i, c_i \in \mathbb{Z}_p$.

Gaussian reduction of the quadratic form: we can assume that the matrix is diagonal (up to an explicit linear transformation on the signature vector).
A known-message attack (1)

- Suppose we are given a message $M$ and a valid signature $\Sigma = (S_1, \ldots, S_n)$.

- We try to find a message $M^*$ and a valid signature $\Sigma^*$ on it which is equal to $\Sigma$ except on one component $S^*$. It suffices to find $(M^*, S^*)$ such that:

$$e(M, M)^a \cdot e(M, US^bK) \cdot e(S, S)^c \cdot e(S, V) =
$$
$$e(M^*, M^*)^a \cdot e(M^*, U(S^*)^bK) \cdot e(S^*, S^*)^c \cdot e(S^*, V)$$

- We look for such a pair $(M^*, S^*)$ expressed explicitly in terms of the group elements we know, i.e. $U, V, M, S, K$:

$$M^* = U^{\mu_0} \cdot V^{\mu_1} \cdot M^{1+\mu_2} \cdot S^{\mu_3} \cdot K^{\mu_4}$$

$$S^* = U^{\sigma_0} \cdot V^{\sigma_1} \cdot M^{\sigma_2} \cdot S^{1+\sigma_3} \cdot K^{\sigma_4}.$$
A known-message attack (1)

- Suppose we are given a message $M$ and a valid signature $\Sigma = (S_1, \ldots, S_n)$.
- We try to find a message $M^*$ and a valid signature $\Sigma^*$ on it which is equal to $\Sigma$ except on one component $S^*$. It suffices to find $(M^*, S^*)$ such that:

$$e(M, M)^a \cdot e(M, US^bK) \cdot e(S, S)^c \cdot e(S, V) = e(M^*, M^*)^a \cdot e(M^*, U(S^*)^bK) \cdot e(S^*, S^*)^c \cdot e(S^*, V)$$

- We look for such a pair $(M^*, S^*)$ expressed explicitly in terms of the group elements we know, i.e. $U, V, M, S, K$:

$$M^* = U^{\mu_0} \cdot V^{\mu_1} \cdot M^{1+\mu_2} \cdot S^{\mu_3} \cdot K^{\mu_4}$$

$$S^* = U^{\sigma_0} \cdot V^{\sigma_1} \cdot M^{\sigma_2} \cdot S^{1+\sigma_3} \cdot K^{\sigma_4}.$$
Suppose we are given a message $M$ and a valid signature $\Sigma = (S_1, \ldots, S_n)$.

We try to find a message $M^*$ and a valid signature $\Sigma^*$ on it which is equal to $\Sigma$ except on one component $S^*$. It suffices to find $(M^*, S^*)$ such that:

$$e(M, M)^a \cdot e(M, US^bK) \cdot e(S, S)^c \cdot e(S, V) = e(M^*, M^*)^a \cdot e(M^*, U(S^*)^bK) \cdot e(S^*, S^*)^c \cdot e(S^*, V)$$

We look for such a pair $(M^*, S^*)$ expressed explicitly in terms of the group elements we know, i.e. $U, V, M, S, K$:

$$M^* = U^{\mu_0} \cdot V^{\mu_1} \cdot M^{1+\mu_2} \cdot S^{\mu_3} \cdot K^{\mu_4}$$

$$S^* = U^{\sigma_0} \cdot V^{\sigma_1} \cdot M^{\sigma_2} \cdot S^{1+\sigma_3} \cdot K^{\sigma_4}.$$
A known-message attack (2)

- With $M^*$, $S^*$ in the form above, our PPE becomes an equation relating the discrete logs $u, v, m, s, k$ of $U, V, M, S, K$:

$$am^2 + m \cdot (u + bs + k) + cs^2 + sv = \tag{1}$$

$$a(m^*)^2 + m^* \cdot (u + bs + k) + c(s^*)^2 + s^* v$$

where:

$$m^* = \mu_0 u + \mu_1 v + (1 + \mu_2) m + \mu_3 s + \mu_4 k$$

$$s^* = \sigma_0 u + \sigma_1 v + \sigma_2 m + (1 + \sigma_3) s + \sigma_4 k.$$

- For the attack to work, we have to find $\mu_\ell, \sigma_\ell$ such that that the equation above is verified for any value of $u, v, m, s, k$.
- To do so, we view (1) as an identity between polynomials in the indeterminates $u, v, m, s, k$.
- This gives a quadratic system of 15 equations in the 10 unknowns $\mu_0, \ldots, \mu_4, \sigma_0, \ldots, \sigma_4$. 
With $M^*$, $S^*$ in the form above, our PPE becomes an equation relating the discrete logs $u, v, m, s, k$ of $U, V, M, S, K$:

$$am^2 + m \cdot (u + bs + k) + cs^2 + sv = a(m^*)^2 + m^* \cdot (u + bs + k) + c(s^*)^2 + s^* v$$

where:

$$m^* = \mu_0 u + \mu_1 v + (1 + \mu_2)m + \mu_3 s + \mu_4 k$$

$$s^* = \sigma_0 u + \sigma_1 v + \sigma_2 m + (1 + \sigma_3)s + \sigma_4 k.$$

For the attack to work, we have to find $\mu_\ell, \sigma_\ell$ such that that the equation above is verified for any value of $u, v, m, s, k$.

To do so, we view (1) as an identity between polynomials in the indeterminates $u, v, m, s, k$.

This gives a quadratic system of 15 equations in the 10 unknowns $\mu_0, \ldots, \mu_4, \sigma_0, \ldots, \sigma_4$. 
A known-message attack (2)

- With \( M^* \), \( S^* \) in the form above, our PPE becomes an equation relating the discrete logs \( u, v, m, s, k \) of \( U, V, M, S, K \):

\[
am^2 + m \cdot (u + bs + k) + cs^2 + sv = \]

\[
a(m^*)^2 + m^* \cdot (u + bs + k) + c(s^*)^2 + s^* v \quad (1)
\]

where:

\[
m^* = \mu_0 u + \mu_1 v + (1 + \mu_2)m + \mu_3 s + \mu_4 k
\]

\[
s^* = \sigma_0 u + \sigma_1 v + \sigma_2 m + (1 + \sigma_3)s + \sigma_4 k.
\]

- For the attack to work, we have to find \( \mu_\ell, \sigma_\ell \) such that that the equation above is verified for any value of \( u, v, m, s, k \).

- To do so, we view (1) as an identity between polynomials in the indeterminates \( u, v, m, s, k \).

- This gives a quadratic system of 15 equations in the 10 unknowns \( \mu_0, \ldots, \mu_4, \sigma_0, \ldots, \sigma_4 \).
A known-message attack (2)

- With $M^*$, $S^*$ in the form above, our PPE becomes an equation relating the discrete logs $u, v, m, s, k$ of $U, V, M, S, K$:

$$am^2 + m \cdot (u + bs + k) + cs^2 + sv = a(m^*)^2 + m^* \cdot (u + bs + k) + c(s^*)^2 + s^* v$$  \hspace{1cm} (1)

where:

$$m^* = \mu_0 u + \mu_1 v + (1 + \mu_2)m + \mu_3 s + \mu_4 k$$

$$s^* = \sigma_0 u + \sigma_1 v + \sigma_2 m + (1 + \sigma_3)s + \sigma_4 k.$$

- For the attack to work, we have to find $\mu_\ell, \sigma_\ell$ such that that the equation above is verified for any value of $u, v, m, s, k$.
- To do so, we view (1) as an identity between polynomials in the indeterminates $u, v, m, s, k$.
- This gives a quadratic system of 15 equations in the 10 unknowns $\mu_0, \ldots, \mu_4, \sigma_0, \ldots, \sigma_4$. 

Hand-solving?

Initial attempts to solve the corresponding system looked like this:
No, Gröbner bases!

Fortunately, we can do better. Solving systems of algebraic equations essentially amounts to computing Gröbner bases for a lexicographic monomial order. So your favorite computer algebra system does it for you.

\[ \text{Mst} = m_0 \cdot u + m_1 \cdot v + (1 + m_2) \cdot m + m_3 \cdot s + m_4 \cdot k \]
\[ \text{Sst} = s_0 \cdot u + s_1 \cdot v + s_2 \cdot m + (1 + s_3) \cdot s + s_4 \cdot k \]

\[ \text{Pl} = a \cdot m^2 + m \cdot (u + k + b \cdot s) + c \cdot s^2 + s \cdot v \]
\[ \text{Pr} = a \cdot \text{Mst}^2 + \text{Mst} \cdot (u + k + b \cdot \text{Sst}) + c \cdot \text{Sst}^2 + \text{Sst} \cdot v \]

\[ P = \text{Pl} - \text{Pr} \]
\[ I = \text{ideal}(P.\text{coefficients}()) \]
\[ \text{IB} = I.\text{groebner\_basis}() \]

The solution is a rational curve, which yields a one-parameter family of solutions, hence the desired attack!
No, Gröbner bases!

Fortunately, we can do better. Solving systems of algebraic equations essentially amounts to computing Gröbner bases for a lexicographic monomial order. So your favorite computer algebra system does it for you.

\[
\begin{align*}
\text{Mst} &= m_0u + m_1v + (1+m_2)m + m_3s + m_4k \\
\text{Sst} &= s_0u + s_1v + s_2m + (1+s_3)s + s_4k \\
\text{Pl} &= a\cdot m^2 + m \cdot (u + k + b\cdot s) + c\cdot s^2 + s\cdot v \\
\text{Pr} &= a\cdot \text{Mst}^2 + \text{Mst} \cdot (u + k + b\cdot \text{Sst}) + c\cdot \text{Sst}^2 + \text{Sst} \cdot v \\
\text{P} &= \text{Pl} - \text{Pr} \\
\text{I} &= \text{ideal(P.coefficients())} \\
\text{IB} &= \text{I.groebner_basis()}
\end{align*}
\]

The solution is a rational curve, which yields a one-parameter family of solutions, hence the desired attack!
No, Gröbner bases!

Fortunately, we can do better. Solving systems of algebraic equations essentially amounts to computing Gröbner bases for a lexicographic monomial order. So your favorite computer algebra system does it for you.

\[
\begin{align*}
\text{Mst} &= m_0 u + m_1 v + (1+m_2)m + m_3 s + m_4 k \\
\text{Sst} &= s_0 u + s_1 v + s_2 m + (1+s_3)s + s_4 k \\
\text{Pl} &= a m^2 + m (u + k + b s ) + c s^2 + s v \\
\text{Pr} &= a Mst^2 + Mst (u + k + b Sst) + c Sst^2 + Sst v \\
\text{P} &= \text{Pl} - \text{Pr} \\
\text{I} &= \text{ideal}(\text{P}.coefficients()) \\
\text{IB} &= \text{I}.\text{groebner\_basis}()
\end{align*}
\]

The solution is a rational curve, which yields a one-parameter family of solutions, hence the desired attack!
## Updated state of the art

<table>
<thead>
<tr>
<th>Group type</th>
<th>Best lower bound</th>
<th>Best construction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type III</td>
<td>$(PPE) \geq 2,</td>
<td>\Sigma</td>
</tr>
<tr>
<td></td>
<td>[AGHO11]</td>
<td>[AGHO11]</td>
</tr>
<tr>
<td>Type I</td>
<td>$(PPE) \geq 2$</td>
<td>$(PPE) = 2,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[AFGHO10]</td>
</tr>
</tbody>
</table>

$(PPE)$: number of verification PPEs,

$|\Sigma|$: signature size in group elements.

Two PPEs required in Type I.
# Updated state of the art

<table>
<thead>
<tr>
<th>Group type</th>
<th>Best lower bound</th>
<th>Best construction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type III</td>
<td>( #(PPE) \geq 2,</td>
<td>\Sigma</td>
</tr>
<tr>
<td>Type I</td>
<td>( #(PPE) \geq 2,</td>
<td>\Sigma</td>
</tr>
</tbody>
</table>

\( \#(PPE) \): number of verification PPEs,
\( |\Sigma| \): signature size in group elements.

Since each message must have superpolynomially many different valid signatures, 3 sig. elements required.
## Updated state of the art

<table>
<thead>
<tr>
<th>Group type</th>
<th>Best lower bound</th>
<th>Best construction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type III</td>
<td>$(\text{PPE}) \geq 2,</td>
<td>\Sigma</td>
</tr>
<tr>
<td></td>
<td>[AGHO11]</td>
<td>[AGHO11]</td>
</tr>
<tr>
<td>Type I</td>
<td>$(\text{PPE}) \geq 2,</td>
<td>\Sigma</td>
</tr>
<tr>
<td></td>
<td>[AGOT14a]</td>
<td>[AGOT14a]</td>
</tr>
</tbody>
</table>

$(\text{PPE})$: number of verification PPEs,

$|\Sigma|$: signature size in group elements.

Matching construction!
### Updated state of the art

<table>
<thead>
<tr>
<th>Group type</th>
<th>Best lower bound</th>
<th>Best construction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type III</td>
<td>(#(PPE) \geq 2,</td>
<td>\Sigma</td>
</tr>
<tr>
<td></td>
<td>[AGHO11]</td>
<td>[AGHO11]</td>
</tr>
<tr>
<td>Type II</td>
<td>???</td>
<td>???</td>
</tr>
<tr>
<td>Type I</td>
<td>(#(PPE) \geq 2,</td>
<td>\Sigma</td>
</tr>
<tr>
<td></td>
<td>[AGOT14a]</td>
<td>[AGOT14a]</td>
</tr>
</tbody>
</table>

So what about Type II?
<table>
<thead>
<tr>
<th>Group type</th>
<th>Best lower bound</th>
<th>Best construction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type III</td>
<td>#(PPE) $\geq 2$, $</td>
<td>\Sigma</td>
</tr>
<tr>
<td></td>
<td>[AGHO11]</td>
<td>[AGHO11]</td>
</tr>
<tr>
<td>Type II</td>
<td>???</td>
<td>???</td>
</tr>
<tr>
<td>Type I</td>
<td>#(PPE) $\geq 2$, $</td>
<td>\Sigma</td>
</tr>
<tr>
<td></td>
<td>[AGOT14a]</td>
<td>[AGOT14a]</td>
</tr>
</tbody>
</table>

So what about Type II?
Computations similar to Type I; the same approach proves that 2 PPEs/3 sig. elements are needed for messages in $G_1$. But fails for messages in $G_2$! Maybe 1 PPE is enough?
Outline

Structure-preserving signatures

Optimal Type I SPS

Optimal Type II SPS

Conclusion
Clearly, an SPS has to have at least one verification PPE. And 2 signature elements is the minimum (to have superpolynomially many signatures per message). Such a scheme is not ruled out by previous lower bounds. Does it exist?
Type II SPS with single-line verification?

- Clearly, an SPS has to have at least one verification PPE.
- And 2 signature elements is the minimum (to have superpolynomially many signatures per message).
- Such a scheme is not ruled out by previous lower bounds. Does it exist?
Type II SPS with single-line verification?

- Clearly, an SPS has to have at least one verification PPE.
- And 2 signature elements is the minimum (to have superpolynomially many signatures per message).
- Such a scheme is not ruled out by previous lower bounds. Does it exist?
One such scheme [AGOT14b]

\textbf{Setup}(1^k): Return $PP = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \psi, G, H) \leftarrow \mathcal{G}(1^k)$.

\textbf{KeyGen}(PP): Choose $v, w \leftarrow \mathbb{Z}_p$ and compute $VK = (PP, V, W)$ and $SK = (PP, v, w)$ using

$$V \leftarrow G^v \quad \quad W \leftarrow G^w.$$ 

\textbf{Sign}_{SK}(M): On $M \in \mathbb{G}_2$ choose $t \leftarrow \mathbb{Z}_p^*$ and compute signature $\Sigma = (R, S)$ as

$$R \leftarrow H^{t-w} \quad \quad S \leftarrow M^v H^{1/t}.$$ 

\textbf{Verify}_{VK}(M, (R, S)): Accept if and only if $M, R, S \in \mathbb{G}_2$ and

$$e(W\psi(R), S) = e(V, M)e(G, H).$$
# Current state of the art

<table>
<thead>
<tr>
<th>Group type</th>
<th>Best lower bound</th>
<th>Best construction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type III</td>
<td>#(PPE) ≥ 2,</td>
<td>Σ</td>
</tr>
<tr>
<td></td>
<td>(AGHO11)</td>
<td></td>
</tr>
<tr>
<td>Type II</td>
<td>#(PPE) ≥ 1,</td>
<td>Σ</td>
</tr>
<tr>
<td></td>
<td>(AGOT14b)</td>
<td></td>
</tr>
<tr>
<td>Type I</td>
<td>#(PPE) ≥ 2,</td>
<td>Σ</td>
</tr>
<tr>
<td></td>
<td>(AGOT14a)</td>
<td></td>
</tr>
</tbody>
</table>

#(PPE): number of verification PPEs,
|Σ|: signature size in group elements.
The previous scheme is sEUF-CMA secure in the generic bilinear group model.

The proof is relatively straightforward.

Finding the scheme itself is not easy, however. This was done by hand.

Can we imagine a more systematic approach?
Security

- The previous scheme is sEUF-CMA secure in the generic bilinear group model.
- The proof is relatively straightforward.
  - Finding the scheme itself is not easy, however. This was done by hand.
- Can we imagine a more systematic approach?
The previous scheme is sEUF-CMA secure in the generic bilinear group model.

The proof is relatively straightforward.

Finding the scheme itself is not easy, however. This was done by hand.

Can we imagine a more systematic approach?
The previous scheme is sEUF-CMA secure in the generic bilinear group model.
The proof is relatively straightforward.
Finding the scheme itself is not easy, however. This was done by hand.
Can we imagine a more systematic approach?
Use formal methods!

- The existence of attacks against the $n$-time security of an SPS scheme is equivalent to the existence of solutions to a corresponding polynomial system.
  - Therefore, Gröbner basis computations let you eliminate insecure schemes among a large set of potential candidates.
  - The security of interesting remaining candidates can be checked manually afterwards.
- We carried out this program in a paper presented at PKC 2014, based on the Generic Group Analyzer tool of Barthe et al.
- Found new secure Type II SPS schemes, and a new lower bound on the number of pairings in the verification equation.
Use formal methods!

- The existence of attacks against the $n$-time security of an SPS scheme is equivalent to the existence of solutions to a corresponding polynomial system.
- Therefore, Gröbner basis computations let you eliminate insecure schemes among a large set of potential candidates.
- The security of interesting remaining candidates can be checked manually afterwards.
- We carried out this program in a paper presented at PKC 2014, based on the Generic Group Analyzer tool of Barthe et al.
- Found new secure Type II SPS schemes, and a new lower bound on the number of pairings in the verification equation.
The existence of attacks against the $n$-time security of an SPS scheme is equivalent to the existence of solutions to a corresponding polynomial system.

Therefore, Gröbner basis computations let you eliminate insecure schemes among a large set of potential candidates.

The security of interesting remaining candidates can be checked manually afterwards.

We carried out this program in a paper presented at PKC 2014, based on the Generic Group Analyzer tool of Barthe et al.

Found new secure Type II SPS schemes, and a new lower bound on the number of pairings in the verification equation.
Use formal methods!

- The existence of attacks against the $n$-time security of an SPS scheme is equivalent to the existence of solutions to a corresponding polynomial system.
- Therefore, Gröbner basis computations let you eliminate insecure schemes among a large set of potential candidates.
- The security of interesting remaining candidates can be checked manually afterwards.
- We carried out this program in a paper presented at PKC 2014, based on the Generic Group Analyzer tool of Barthe et al.
  - Found new secure Type II SPS schemes, and a new lower bound on the number of pairings in the verification equation.
Use formal methods!

- The existence of attacks against the $n$-time security of an SPS scheme is equivalent to the existence of solutions to a corresponding polynomial system.
- Therefore, Gröbner basis computations let you eliminate insecure schemes among a large set of potential candidates.
- The security of interesting remaining candidates can be checked manually afterwards.
- We carried out this program in a paper presented at PKC 2014, based on the Generic Group Analyzer tool of Barthe et al.
- Found new secure Type II SPS schemes, and a new lower bound on the number of pairings in the verification equation.
## Search results

<table>
<thead>
<tr>
<th>Template</th>
<th>Schemes</th>
<th>Results (for equiv. cl.)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>total</td>
<td>equiv. cl.</td>
</tr>
<tr>
<td>Template 1</td>
<td>212</td>
<td>57</td>
</tr>
<tr>
<td>Template 2</td>
<td>224</td>
<td>67</td>
</tr>
<tr>
<td>Template 3</td>
<td>1344</td>
<td>774</td>
</tr>
<tr>
<td>Template 4</td>
<td>224</td>
<td>126</td>
</tr>
<tr>
<td></td>
<td>2004</td>
<td>1024</td>
</tr>
</tbody>
</table>
Outline

Structure-preserving signatures

Optimal Type I SPS

Optimal Type II SPS

Conclusion
Final comments

- Gröbner basis techniques work nicely to analyze structure-preserving signatures:
  - Let you find attack if they exist
  - Help you construct new schemes using large-scale search
  - Make it easier to complete tedious proofs

- Future work:
  - More SPS synthesis: beyond Type II? Fully structure-preserving signatures?
  - Other structure-preserving primitives?
  - Counterpart in the multilinear setting?
Gröbner basis techniques work nicely to analyze structure-preserving signatures:
- Let you find attack if they exist
- Help you construct new schemes using large-scale search
- Make it easier to complete tedious proofs

Future work:
- More SPS synthesis: beyond Type II? Fully structure-preserving signatures?
- Other structure-preserving primitives?
- Counterpart in the multilinear setting?
Thank you!