Banquet: Short and Fast Signatures from AES

PKC 2021

C. Baum  C. dSG  D. Kales  E. Orsini  P. Scholl  G. Zaverucha
imec-COSIC, KU Leuven; Aarhus University; Graz UoT; Microsoft Research
Key Facts

Zero-Knowledge Proofs of Knowledge from MPC
  General idea
  Computing the circuit
  Verifying the circuit

Inverse Verification
  Naïve
  Polynomial-based
  Generalized poly-based

The Banquet signature scheme

Implementation
  Parameter selection
  Performance
  Optimizations
1 Outline

1 Key Facts

2 Zero-Knowledge Proofs of Knowledge from MPC

3 Inverse Verification

4 The Banquet signature scheme

5 Implementation
1 Paper highlights

- Banquet signature scheme $= \text{FS} \times (\text{MPCitH} + \text{ZKPoK})$.
- EUF-CMA security $\approx \text{OWF of AES (with modified key gen.) in RO}$. 
  No public-key assumptions.
1 Paper highlights

- Banquet signature scheme $= FS \times (\text{MPCitH} + \text{ZKPoK})$.
- EUF-CMA security $\approx$ OWF of AES (with modified key gen.) in RO. No public-key assumptions.
- Same line of work as:
  - Picnic (now Picnic 3, NIST round 3 alternate)—based on LowMC (600 AND gates).
  - BBQ—Picnic with AES (6400 AND gates), attempt #1.
1 Paper highlights

- Banquet signature scheme $= \text{FS} \times (\text{MPCitH} + \text{ZKPoK})$.
- EUF-CMA security $\approx$ OWF of AES (with modified key gen.) in RO.
  No public-key assumptions.
- Same line of work as:
  - Picnic (now Picnic 3, NIST round 3 alternate)—based on LowMC (600 AND gates).
  - BBQ—Picnic with AES (6400 AND gates), attempt #1.
- Improvements:
  1. Over Picnic: better assumption (AES instead of LowMC).
  2. Over BBQ: better performance (size and speed).
### Some numbers

<table>
<thead>
<tr>
<th>Protocol</th>
<th>(N)</th>
<th>Sign (ms)</th>
<th>Verify (ms)</th>
<th>Size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Picnic2</td>
<td>64</td>
<td>41.16</td>
<td>18.21</td>
<td>12347</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>10.42</td>
<td>5.00</td>
<td>13831</td>
</tr>
<tr>
<td>Picnic3</td>
<td>16</td>
<td>5.33</td>
<td>4.03</td>
<td>12466</td>
</tr>
<tr>
<td>AES bin</td>
<td>64</td>
<td>-</td>
<td>-</td>
<td>51876</td>
</tr>
<tr>
<td>BBQ</td>
<td>64</td>
<td>-</td>
<td>-</td>
<td>31876</td>
</tr>
<tr>
<td>Banquet</td>
<td>16</td>
<td>6.36</td>
<td>4.86</td>
<td>19776</td>
</tr>
<tr>
<td></td>
<td>107</td>
<td>21.13</td>
<td>18.96</td>
<td>14784</td>
</tr>
</tbody>
</table>

Table: Signature size and run times (if available) for Picnic2, Picnic3, AES binary, BBQ and Banquet for comparable MPCitH parameters and 128 bit security.

Full version available as ePrint 2021/068.
2 Outline

1 Key Facts

2 Zero-Knowledge Proofs of Knowledge from MPC
   General idea
   Computing the circuit
   Verifying the circuit

3 Inverse Verification

4 The Banquet signature scheme

5 Implementation
2 MPC-in-the-head: general idea

Zero-knowledge proof of knowledge from MPC:

▶ “I know $w$ such that $C(x, w) = 1$” for public circuit $C$ and input $x$.
▶ Proof: ability to simulate $N$-party MPC protocol computing $C'(x, w)$. 

In short:

▶ Prover generates and commits to views of $N$ parties.
▶ Verifier asks to see some of them, and checks they are consistent with each other and with $C(x, w) = 1$.
▶ Soundness: probability that verifier sees inconsistent views.
▶ Zero-knowledge: semi-honest security of the MPC protocol.
2 MPC-in-the-head: general idea

Zero-knowledge proof of knowledge from MPC:
- “I know \( w \) such that \( C(x, w) = 1 \)” for public circuit \( C \) and input \( x \).
- Proof: ability to simulate \( N \)-party MPC protocol computing \( C(x, w) \).

In short:
- Prover generates and commits to views of \( N \) parties.
- Verifier asks to see some of them, and checks they are consistent with each other and with \( C(x, w) = 1 \).
2 MPC-in-the-head: general idea

Zero-knowledge proof of knowledge from MPC:
- “I know $w$ such that $C(x, w) = 1$” for public circuit $C$ and input $x$.
- Proof: ability to simulate $N$-party MPC protocol computing $C(x, w)$.

In short:
- Prover generates and commits to views of $N$ parties.
- Verifier asks to see some of them, and checks they are consistent with each other and with $C(x, w) = 1$.
- **Soundness**: probability that verifier sees inconsistent views.
- **Zero-knowledge**: semi-honest security of the MPC protocol.
2 Picnic signature scheme

- KKW and Picnic technique: compute $C$ with an MPC protocol.

Other notes:
- Cut & choose $\Rightarrow$ verified correlated randomness (masks or triples) $\Rightarrow$ use communication-efficient MPC protocol.
- Drawback: 100's of cut & choose required for only 10's kept.

3-round proof: $C$ has to be wastefully executed each time.

Picnic3: 252 generated for 36 used.

For block cipher $F = \text{LowMC}$ written as binary over $F_2$, Picnic uses plaintext $x$, ciphertext $y$, key $w$, and circuit $C(x, w) = 1 \iff F(w)(x) = y$. 
2 Picnic signature scheme

- KKW and Picnic technique: compute $C$ with an MPC protocol.
- Cut & choose $\Rightarrow$ verified correlated randomness (masks or triples) $\Rightarrow$ use communication-efficient MPC protocol.
2 Picnic signature scheme

▶ KKW and Picnic technique: compute $C$ with an MPC protocol.
▶ Cut & choose $\Rightarrow$ verified correlated randomness (masks or triples) $\Rightarrow$ use communication-efficient MPC protocol.

▶ Drawback: 100’s of cut & choose required for only 10’s kept.
3-round proof: $C$ has to be wastefully executed each time.
Picnic3: 252 generated for 36 used.
2 Picnic signature scheme

- KKW and Picnic technique: compute $C$ with an MPC protocol.
- Cut & choose $\Rightarrow$ verified correlated randomness (masks or triples) $\Rightarrow$ use communication-efficient MPC protocol.
- Drawback: 100's of cut & choose required for only 10's kept.
- 3-round proof: $C$ has to be wastefully executed each time.
- Picnic3: 252 generated for 36 used.

For block cipher $F = \text{LowMC}$ written as binary over $\mathbb{F}_2$, Picnic uses plaintext $x$, ciphertext $y$, key $w$, and circuit

$$C(x, w) = 1 \iff F_w(x) = y.$$
2 The BBQ signature scheme

LowMC $\rightarrow$ AES
Binary circuit over $\mathbb{F}_2$ $\rightarrow$ Arithmetic circuit over $\mathbb{F}_{2^8}$
AND gate $\rightarrow$ INV gate (which is $\approx$ S-box)
2 The BBQ signature scheme

LowMC $\rightarrow$ AES

Binary circuit over $\mathbb{F}_2$ $\rightarrow$ Arithmetic circuit over $\mathbb{F}_{2^8}$

AND gate $\rightarrow$ **INV gate (which is $\approx$ S-box)**

Masked inversion computation of input $s$ and random $r$:

1. Compute $\langle s \cdot r \rangle$ with triple $(\langle a \rangle, \langle b \rangle, \langle c \rangle)$. $\triangleright$ +2 openings (+1 elt. for $c$)
2. Open$(s \cdot r)$. $\triangleright$ +1 opening
3. Compute $(s \cdot r)^{-1}$ locally.
4. Compute $\langle s^{-1} \rangle = (s^{-1} \cdot r^{-1}) \cdot \langle r \rangle$. Requires $r \neq 0$.
   Requires $s \neq 0$: choose AES key such that this doesn’t happen.
2 The BBQ signature scheme

LowMC $\rightarrow$ AES

Binary circuit over $\mathbb{F}_2$ $\rightarrow$ Arithmetic circuit over $\mathbb{F}_{2^8}$

AND gate $\rightarrow$ INV gate (which is $\approx$ S-box)

Masked inversion computation of input $s$ and random $r$:

1. Compute $\langle s \cdot r \rangle$ with triple $(\langle a \rangle, \langle b \rangle, \langle c \rangle)$. $\triangleright$ +2 openings (+1 elt. for $c$)
2. Open($s \cdot r$). $\triangleright$ +1 opening
3. Compute $(s \cdot r)^{-1}$ locally.
4. Compute $\langle s^{-1} \rangle = (s^{-1} \cdot r^{-1}) \cdot \langle r \rangle$.

Requires $r \neq 0$: restart if it is.
Requires $s \neq 0$: choose AES key such that this doesn’t happen.
2 Witness extension and verification

Idea from sacrificing techniques in MPC

- Prover “injects” the results of multiplications—no need to compute.
  - The witness is extended with the outputs of non-linear gates.
2 Witness extension and verification

Idea from sacrificing techniques in MPC

- Prover “injects” the results of multiplications—no need to compute.
  - The witness is extended with the outputs of non-linear gates.

- MPC parties execute a verification protocol—batching possibilities.
  - e.g. Sacrifice one “suspicious” triple to verify another.
2 Witness extension and verification

Idea from sacrificing techniques in MPC
- Prover “injects” the results of multiplications—no need to compute.
  - The witness is extended with the outputs of non-linear gates.
- MPC parties execute a verification protocol—batching possibilities.
  - e.g. Sacrifice one “suspicious” triple to verify another.

ZKPoK protocol sketch

MPC parties receive “suspicious” multiplication results and verify them by sacrificing “suspicious” random triples $\Rightarrow 4|C| + 1$ elts., no cut & choose.

Inherently $\geq$ 5-round protocol $\Rightarrow$ new analysis required for NI soundness.
3 Outline

1 Key Facts

2 Zero-Knowledge Proofs of Knowledge from MPC

3 Inverse Verification
   Naïve
   Polynomial-based
   Generalized poly-based

4 The Banquet signature scheme

5 Implementation
Verifying inverses

Prover injects “suspicious” inverses $t = s^{-1}$ into MPCitH. Parties have $m = |C|$ pairs $(s, t)$ which allegedly multiply to $s \cdot t = 1$.

Naïve verification protocol

For each $\ell \in [m]$:

1. Set multiplication tuple $(s_\ell, t_\ell, 1)$.
2. Sacrifice with triple $(a, b, c)$.

$4|C| + 1$ elts.

Can do better!
3 Polynomial-based verification I

Define $S, T$ and $P = S \cdot T$ as:

\[
\begin{align*}
S(1) &= s_1 & T(1) &= t_1 & P(1) &= s_1 \cdot t_1 = 1 \\
\vdots & & \vdots & & \vdots \\
S(m) &= s_m & T(m) &= t_m & P(m) &= s_m \cdot t_m = 1
\end{align*}
\]
3 Polynomial-based verification I

Define $S, T$ and $P = S \cdot T$ as:

$S(1) = s_1 \quad T(1) = t_1 \quad P(1) = s_1 \cdot t_1 = 1$

\[ \vdots \quad \vdots \quad \vdots \]

$S(m) = s_m \quad T(m) = t_m \quad P(m) = s_m \cdot t_m = 1$

Check $P \overset{?}{=} S \cdot T$:

1. Sample random $R \leftarrow \mathbb{F} \setminus \{1, \ldots, m\}$;
2. Open $P(R), S(R), T(R)$;
3. Check

$$P(R) \overset{?}{=} S(R) \cdot T(R).$$
3  Polynomial-based verification II

Lemma (Schwartz–Zippel)
Let \( Q \in \mathbb{F}[x] \) be non-zero of degree \( d \geq 0 \); for any \( S \subseteq \mathbb{F} \),
\[
\Pr_{R \leftarrow S}[Q(R) = 0] \leq \frac{d}{|S|}.
\]

▶ Here, \( Q = P - S \cdot T \); non-zero iff \( t_\ell \neq s_\ell^{-1} \) for some \( \ell \).
3 Polynomial-based verification II

Lemma (Schwartz–Zippel)

Let $Q \in \mathbb{F}[x]$ be non-zero of degree $d \geq 0$; for any $S \subseteq \mathbb{F}$,

$$\Pr_{R \leftarrow S}[Q(R) = 0] \leq \frac{d}{|S|}.$$

Here, $Q = P - S \cdot T$; non-zero iff $t_\ell \neq s_\ell^{-1}$ for some $\ell$.

Opening $S(R), T(R)$ leaks information $\Rightarrow$ add random points $S(0), T(0)$. 
3 Polynomial-based verification II

Lemma (Schwartz–Zippel)

Let $Q \in \mathbb{F}[x]$ be non-zero of degree $d \geq 0$; for any $S \subseteq \mathbb{F}$,

$$\Pr_{R \leftarrow S}[Q(R) = 0] \leq \frac{d}{|S|}.$$

- Here, $Q = P - S \cdot T$; non-zero iff $t_{\ell} \neq s_{\ell}^{-1}$ for some $\ell$.
- Opening $S(R), T(R)$ leaks information $\Rightarrow$ add random points $S(0), T(0)$.
- $P$ (and also $Q$) is of degree $d = 2m$ and $|S| = |\mathbb{F} - m|$, so

$$\Pr_{R \leftarrow S}[Q(R) = 0] \leq \frac{2m}{|\mathbb{F} - m|}.$$
3 Polynomial-based verification III

Improved protocol

1. Prover commits to $S$ (randomized) and $T$; $m$ elts. for $T$.
2. Prover commits to $P$; $(2m + 1) - m = m + 1$ elts. for $P$.
3. MPC parties open $Q(R) = P(R) - S(R) \cdot T(R)$, for random $R$; 3 elts.

In total: $2|C| + 4$ elts.; no cut & choose, no triple.\(^1\)

(Extra randomness in $S$ prevents correcting one wrong pair with another.)

\(^1\)Actually, one triple, but hidden!
3 Generalized polynomial-based checking

Previous protocol verifies:

\[
\begin{pmatrix}
    r_1 s_1 & \cdots & r_m s_m
\end{pmatrix}
\begin{pmatrix}
    t_1 \\
    \vdots \\
    t_m
\end{pmatrix}
\overset{?}{=}
\sum_{\ell=1}^{m} r_{\ell}.
\]
3 Generalized polynomial-based checking I

Previous protocol verifies:

\[
\begin{pmatrix}
    r_1s_1 & \cdots & r_ms_m
\end{pmatrix}
\begin{pmatrix}
    t_1 \\
    \vdots \\
    t_m
\end{pmatrix}
\approx m \sum_{\ell=1}^{m} r_{\ell}.
\]

Now, let \( m = m_1 \cdot m_2 \), and instead verify:

\[
\begin{pmatrix}
    r_1s_{1,k} & \cdots & r_ms_{m,k}
\end{pmatrix}
\begin{pmatrix}
    t_{1,k} \\
    \vdots \\
    t_{m_1,k}
\end{pmatrix}
\approx m_1 \sum_{j=1}^{m_1} r_j, \quad k \in \{0, \ldots, m_2 - 1\}.
\]

\((s_{j,k} \text{ and } t_{j,k} \text{ are rearranged from } s_{\ell} \text{ and } t_{\ell}).\)
Define $S_j$ and $T_j$ as

$$S_j(k) = r_j \cdot s_{j,k} \quad T_j(k) = t_{j,k} \quad k \in \{0, \ldots, m_2 - 1\}$$

$$S_j(m_2) = \bar{s}_j \quad T_j(m_2) = \bar{t}_j;$$

and let $P = \sum_{j=1}^{m_1} S_j \cdot T_j$. 
Define $S_j$ and $T_j$ as

$$S_j(k) = r_j \cdot s_{j,k} \quad \quad T_j(k) = t_{j,k} \quad \quad k \in \{0, \ldots, m_2 - 1\}$$

$$S_j(m_2) = \bar{s}_j \quad \quad T_j(m_2) = \bar{t}_j$$

and let $P = \sum_{j=1}^{m_1} S_j \cdot T_j$.

**Generalized verification protocol**

1. Prover commits to $S_j$ (randomized) and $T_j$; $m$ elts. for $T_j$’s.
2. Prover commits to $P$; $(2m_2 + 1) - m_2 = m_2 + 1$ elts. for $P$.
3. MPC parties open $Q(R) = P(R) - \sum_{j=1}^{m_1} S_j(R) \cdot T_j(R)$, for random $R$; $1 + 2m_1$ elts.

Total: $m$ (inherent) + $m_2 + 2m_1 + 2$ elts. = $|C| + O(\sqrt{|C|})$, instead of $2|C|$.
Outline

1. Key Facts
2. Zero-Knowledge Proofs of Knowledge from MPC
3. Inverse Verification
4. The Banquet signature scheme
5. Implementation
4 The Banquet signature scheme I

Key generation

Sample AES key $k$ and plaintext $x$ from $\{0, 1\}^\kappa$ such that

$$y \leftarrow \text{AES}_k(x)$$

presents no 0 input to S-boxes.
Set $pk = (x, y)$ and $sk = k$.

This sampling methods reduces security of the OWF assumption by $1 \sim 3$ bits.
4 The Banquet signature scheme II

Signature

Parameters: $m, m_1, N, \tau, \lambda$.

- Prover simulates $\tau$ parallel MPC instances, each with $N$ parties.
- Together with a sharing of $k$, the witness includes sharings of $t_\ell$'s.
- Random oracles are used to generate $r_j$'s, $R$'s and to select the views.

$\Rightarrow$ 7-round protocol

Verification (of signature)

Recompute executions, check hashes and output.
The Banquet signature scheme—security

Theorem

The Banquet signature scheme is EUF-CMA-secure, assuming that Commit, $H_1$, $H_2$ and $H_3$ are modelled as random oracles, Expand is a PRG with output computationally $\epsilon_{\text{PRG}}$-close to uniform, the seed tree construction is computationally hiding, the $(N, \tau, m_2, \lambda)$ parameters are appropriately chosen, and the key generation function $f_x : k \mapsto y$ is a one-way function.
5 Outline

1 Key Facts

2 Zero-Knowledge Proofs of Knowledge from MPC

3 Inverse Verification

4 The Banquet signature scheme

5 Implementation
   Parameter selection
   Performance
   Optimizations
Attacker can cheat by re-sampling challenges until they match its guess. Say guess $\tau_1$ in 1st round, and $\tau_2$ in 2nd round. ⇒ must guess $\tau_3 = \tau - \tau_1 - \tau_2$ to win.

Let $P_i = \Pr[\text{guess } \tau_i \text{ challenges}]$; depends on $(N, \tau, m_2, \lambda)$. Cost of attack is
\[ C = 1/P_1 + 1/P_2 + 1/P_3 \]
for a given strategy $(\tau_1, \tau_2, \tau_3)$. Need $C \geq 2^k$ for all strategies.

Choosing $m_1 \approx \sqrt{m}$ gives fast and short signatures.
## Implementation—Performance variation

<table>
<thead>
<tr>
<th>Scheme</th>
<th>$N$</th>
<th>$\lambda$</th>
<th>$\tau$</th>
<th>Sign (ms)</th>
<th>Verify (ms)</th>
<th>Size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AES-128</td>
<td>16</td>
<td>4</td>
<td>41</td>
<td>6.36</td>
<td>4.86</td>
<td>19776</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>6</td>
<td>37</td>
<td>5.91</td>
<td>4.51</td>
<td>20964</td>
</tr>
<tr>
<td></td>
<td>31</td>
<td>4</td>
<td>35</td>
<td>8.95</td>
<td>7.46</td>
<td>17456</td>
</tr>
<tr>
<td></td>
<td>31</td>
<td>6</td>
<td>31</td>
<td>8.19</td>
<td>6.76</td>
<td>18076</td>
</tr>
<tr>
<td></td>
<td>57</td>
<td>4</td>
<td>31</td>
<td>14.22</td>
<td>12.30</td>
<td>15968</td>
</tr>
<tr>
<td></td>
<td>57</td>
<td>6</td>
<td>27</td>
<td>12.45</td>
<td>10.75</td>
<td>16188</td>
</tr>
<tr>
<td></td>
<td>107</td>
<td>4</td>
<td>28</td>
<td>24.15</td>
<td>21.71</td>
<td>14880</td>
</tr>
<tr>
<td></td>
<td>107</td>
<td>6</td>
<td>24</td>
<td>21.13</td>
<td>18.96</td>
<td>14784</td>
</tr>
<tr>
<td></td>
<td>255</td>
<td>4</td>
<td>25</td>
<td>51.10</td>
<td>46.88</td>
<td>13696</td>
</tr>
<tr>
<td></td>
<td>255</td>
<td>6</td>
<td>21</td>
<td>43.81</td>
<td>40.11</td>
<td>13284</td>
</tr>
</tbody>
</table>

Table: Performance of different parameter sets; all instances $(m, m_1, m_2) = (200, 10, 20)$. 
5 Implementation—Optimizations

▶ All interpolation points have same $x$: pre-compute Lagrange coefficients.

▶ Interpolating shares of polynomials.
  (1) re-construct points, (2) interpolate polys. $1/N \times$ interpolations

▶ For $S$’s and $T$’s, $m_2$ points are the same across parallel repetitions.
  Last point only requires adding multiple of Lagrange poly.

▶ Reduces runtime by 30x to 100 ms, approx.
  Further improvements with dedicated field arithmetic and other tricks.
## 5 Implementation—Comparison

<table>
<thead>
<tr>
<th>Protocol</th>
<th>$N$</th>
<th>$M$</th>
<th>$\tau$</th>
<th>Sign (ms)</th>
<th>Ver (ms)</th>
<th>Size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Picnic2</td>
<td>64</td>
<td>343</td>
<td>27</td>
<td>41.16</td>
<td>18.21</td>
<td>12 347</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>252</td>
<td>36</td>
<td>10.42</td>
<td>5.00</td>
<td>13 831</td>
</tr>
<tr>
<td>Picnic3</td>
<td>16</td>
<td>252</td>
<td>36</td>
<td>5.33</td>
<td>4.03</td>
<td>12 466</td>
</tr>
<tr>
<td>SPHINCS$^+$-fast</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>14.42</td>
<td>1.74</td>
<td>16 976</td>
</tr>
<tr>
<td>SPHINCS$^+$-small</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>239.34</td>
<td>0.73</td>
<td>8 080</td>
</tr>
<tr>
<td>Banquet</td>
<td>16</td>
<td>-</td>
<td>41</td>
<td>6.36</td>
<td>4.86</td>
<td>19 776</td>
</tr>
<tr>
<td></td>
<td>107</td>
<td>-</td>
<td>24</td>
<td>21.13</td>
<td>18.96</td>
<td>14 784</td>
</tr>
<tr>
<td></td>
<td>255</td>
<td>-</td>
<td>21</td>
<td>43.81</td>
<td>40.11</td>
<td>13 284</td>
</tr>
</tbody>
</table>

Table: Comparison of signature sizes and run times for various MPCitH-based signature schemes and SPHINCS$^+$ (using “sha256simple” parameter sets).
Thanks!

ePrint/2021/068

cdsg@esat.kuleuven.be