KU LEUVEN

Banquet: Short and Fast Signatures from AES

NTNU NaCl meeting

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

Wed. 24 February 2021

1 Outline

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

1 Why you should buy our paper[1]

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

[1] Don't, it's free on ePrint.

1 Why you should buy our paper[1]

- Banquet signature scheme $=$ FS $\times$ (MPCitH $+$ 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).
    G. Zaverucha & D. Kales et al.
  - BBQ—Picnic with AES (6400 AND gates), attempt #1.
    E. Orsini & myself et al.

[1] Don't, it's free on ePrint.

1 Why you should buy our paper[1]

- Banquet signature scheme = FS × (MPCitH + ZKPoK).
- EUF-CMA security ≈ 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).
    G. Zaverucha & D. Kales et al.
  - BBQ—Picnic with AES (6400 AND gates), attempt #1.
    E. Orsini & myself et al.
- Improvements:
  1. Over Picnic: better assumption (AES instead of LowMC).
  2. Over BBQ: better performance (size and speed).

[1] Don't, it's free on ePrint.

1 Some numbers

Protocol	N	Sign (ms)	Verify (ms)	Size (bytes)
Picnic2	64	41.16	18.21	12 347
	16	10.42	5.00	13 831
Picnic3	16	5.33	4.03	12 466
AES Bin	64	-	-	51 876
BBQ	64	-	-	31 876
Banquet	16	7.03	5.76	19 776
	107	27.94	24.94	14 784

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

ePrint 2021/068; short version to appear at PKC'21.

2 Outline

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 $C(x, w)$.

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.

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.

Picnic uses plaintext x, key w, and circuit

$$C(x, w) = 1 \iff F_w(x) = y$$

for block cipher $F =$ LowMC written as binary over $\mathbb{F}_2$.

2 The BBQ signature scheme

$$\begin{aligned} \text{LowMC} &\longrightarrow \text{AES} \\ \text{AND gate} &\longrightarrow \textbf{INV gate (which is} \approx \textbf{S-box)} \\ \text{Binary circuit over } \mathbb{F}_2 &\longrightarrow \text{Arithmetic circuit over } \mathbb{F}_{2^8} \end{aligned}$$

2 The BBQ signature scheme

$$\begin{aligned} \text{LowMC} &\longrightarrow \text{AES} \\ \text{AND gate} &\longrightarrow \textbf{INV gate (which is} \approx \textbf{S-box)} \\ \text{Binary circuit over } \mathbb{F}_2 &\longrightarrow \text{Arithmetic circuit over } \mathbb{F}_{2^8} \end{aligned}$$

Masked inversion computation of input s and random r:

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

2 The BBQ signature scheme

$$\text{LowMC} \longrightarrow \text{AES}$$

$$\text{AND gate} \longrightarrow \textbf{INV gate (which is } \approx \textbf{ S-box)}$$

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

Masked inversion computation of input s and random r:

1: Compute $\langle s \cdot r \rangle$ with triple $(\langle a \rangle, \langle b \rangle, \langle c \rangle)$. ▷ $+2$ openings ($+1$ elt. for c)
2: Open$(s \cdot r)$. ▷ $+1$ 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 $\Rightarrow$ $4 + 1/|C|$ elts., no cut & choose.

$$0 \stackrel{?}{=} \langle v \rangle = \epsilon \langle z \rangle - \langle c \rangle + \alpha \langle b \rangle + \beta \langle a \rangle - \alpha \cdot \beta$$

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

3 Verifying inverses

Prover injects "suspicious" inverses $t = s^{-1}$ into MPCitH.
Parties have m 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).

$4 + 1/|C|$ elts. per gate

Can do better!

3 Polynomial-based verification I

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

$$
\begin{aligned}
S(1) &= s_1 & T(1) &= t_1 & P(1) &= s_1 \cdot t_1 = 1 \\
&\vdots & &\vdots & &\vdots \\
S(m) &= s_m & T(m) &= t_m & P(m) &= s_m \cdot t_m = 1
\end{aligned}
$$

3 Polynomial-based verification I

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

$$\begin{aligned} S(1) &= s_1 & T(1) &= t_1 & P(1) &= s_1 \cdot t_1 = 1 \\ &\vdots & &\vdots & &\vdots \\ S(m) &= s_m & T(m) &= t_m & P(m) &= s_m \cdot t_m = 1 \end{aligned}$$

Can check $P \stackrel{?}{=} S \cdot T$:

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

$$P(R) \stackrel{?}{=} S(R) \cdot T(R).$$

3 Polynomial-based verification II

Lemma (Schwartz–Zippel)

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

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

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

3 Polynomial-based verification II

Lemma (Schwartz–Zippel)

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

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

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

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

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

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

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: $2 + 4/|C|$ elts. per gate; no cut & choose, no triple.[2]

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

---

[2] Actually, one triple, but hidden!

3 Generalized polynomial-based checking I

Previous protocol verifies:

$$\begin{pmatrix} r_1 s_1 & \cdots & r_m s_m \end{pmatrix} \begin{pmatrix} t_1 \\ \vdots \\ t_m \end{pmatrix} \stackrel{?}{=} \sum_{\ell=1}^{m} r_\ell .$$

3 Generalized polynomial-based checking I

Previous protocol verifies:

$$\begin{pmatrix} r_1 s_1 & \cdots & r_m s_m \end{pmatrix} \begin{pmatrix} t_1 \\ \vdots \\ t_m \end{pmatrix} \stackrel{?}{=} \sum_{\ell=1}^{m} r_\ell .$$

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

$$\begin{pmatrix} r_1 s_{1,k} & \cdots & r_{m_1} s_{m_1,k} \end{pmatrix} \begin{pmatrix} t_{1,k} \\ \vdots \\ t_{m_1,k} \end{pmatrix} \stackrel{?}{=} \sum_{j=1}^{m_1} r_j , \qquad k \in \{0, \ldots, m_2 - 1\}.$$

($s_{j,k}$ and $t_{j,k}$ are rearranged from s_ℓ and t_ℓ.)

3 Generalized polynomial-based checking II

Define S_j and T_j as

$$\begin{aligned} S_j(k) &= r_j \cdot s_{j,k} & T_j(k) &= t_{j,k} & k \in \{0, \ldots, m_2 - 1\} \\ S_j(m_2) &= \bar{s}_j & T_j(m_2) &= \bar{t}_j; \end{aligned}$$

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

3 Generalized polynomial-based checking II

Define S_j and T_j as

$$S_j(k) = r_j \cdot s_{j,k} \qquad T_j(k) = t_{j,k} \qquad k \in \{0, \ldots, m_2 - 1\}$$
$$S_j(m_2) = \bar{s}_j \qquad T_j(m_2) = \bar{t}_j;$$

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. $= m + O(\sqrt{m})$, instead of $2m$.

4 Outline

4 The Banquet signature scheme I

Key generation

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

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

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

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, τ, λ.

- Prover simulates τ parallel MPC instances, each with N parties.
- Together with a sharing of k, the witness includes sharings of t_ℓ'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.

4 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* ϵ_{PRG}*-close to uniform, the seed tree construction is computationally hiding, the* (N, τ, m_2, λ) *parameters are appropriately chosen, and the key generation function* $f_x : k \mapsto y$ *is a one-way function.*

5 Outline

5 Implementation—Parameter selection

- Attacker can cheat by re-sampling challenges until they match its guess.
  Say guess τ_1 in 1st round, and τ_2 in 2nd round.
  $\Rightarrow$ must guess $\tau_3 = \tau - \tau_1 - \tau_2$ to win.

- Let $P_i = \Pr[\text{guess } \tau_i \text{ challenges}]$; depends on (N, τ, m_2, λ).
  Cost of attack is

  $$C = 1/P_1 + 1/P_2 + 1/P_3$$

  for a given strategy (τ_1, τ_2, τ_3). Need $C \geq 2^\kappa$ for all strategies.

- Choosing $m_1 \approx \sqrt{m}$ gives fast and short signatures.

5 Implementation—Performance variation

Scheme	N	λ	τ	Sign (ms)	Verify (ms)	Size (bytes)
AES-128	16	4	41	7.05	5.78	19776
	16	6	37	6.58	5.37	20964
	31	4	35	10.21	9.01	17456
	31	6	31	9.31	8.14	18076
	57	4	31	15.99	14.83	15968
	57	6	27	14.24	13.18	16188
	107	4	28	27.08	25.90	14880
	107	6	24	23.79	22.68	14784
	255	4	25	57.14	55.88	13696
	255	6	21	49.28	48.27	13284

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

Protocol	N	M	τ	Sign (ms)	Ver (ms)	Size (bytes)
Picnic2	64	343	27	41.16	18.21	12 347
	16	252	36	10.42	5.00	13 831
Picnic3	16	252	36	5.33	4.03	12 466
SPHINCS$^+$-fast	-	-	-	14.42	1.74	16 976
SPHINCS$^+$-small	-	-	-	239.34	0.73	8 080
Banquet	16	-	41	7.05	5.78	19 776
	107	-	24	23.79	22.68	14 784
	255	-	21	49.28	48.27	13 284

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

Thanks! Any questions?

ePrint/2021/068

cdsg@esat.kuleuven.be