Walnut Digital Signature Algorithm: A lightweight, quantum-resistant signature scheme for use in passive, low-power, and IoT devices

Derek Atkins

CTO, SecureRF Corporation

NIST Lightweight Cryptography Workshop, October 18, 2016
Talk Outline

- Walnut Digital Signature Algorithm Overview
- Background Math
- WalnutDSA method description
- Security Analysis
- Performance Results

(joint work with Iris Anshel, Dorian Goldfeld, Paul Gunnells)
Introducing WalnutDSA

A group theoretic digital signature algorithm with the following features:

- **Very fast signature verification:**
  - Running time is linear in key/signature size.
  - Key/Signature size grows linearly with security level.

- **Security is based on the hard problems of solving a novel equation over the braid group and reversing a certain representation of the braid group.**

- **WalnutDSA appears resistant to known attacks in group theoretic cryptography.**

- **Quantum Resistant**
A braid on $N$ strands ($B_N$) is a collection of $N$ entangled strings. We can represent a braid by a *left-right crossing sequence* of signed nonzero integers $i_1 i_2 \cdots i_k$, ("Artin generators") each of which lies between $-N$ and $N$.

- A positive integer $i$ means “cross the $i$th strand under the $(i + 1)$st strand.”
- A negative integer $-i$ means “cross the $i$th strand over the $(i + 1)$st strand.”
Background Math

Colored Burau Representation of $B_N$

Each $b_i^{\pm 1}$ is associated with the ordered pair $(CB(b_i)^{\pm 1}, \sigma_i)$ where $\sigma_i$ is the transposition $(i, i + 1)$,

$$
CB(b_i) = \begin{pmatrix}
1 & & & & \\
 & \ddots & & & \\
& & t_i & -t_i & 1 \\
& & & \ddots & \\
& & & & 1
\end{pmatrix},
$$

$$
CB(b_i^{-1}) = \begin{pmatrix}
1 & & & & \\
 & \ddots & & & \\
& & 1 & -\frac{1}{t_i+1} & \frac{1}{t_i+1} \\
& & & \ddots & \\
& & & & 1
\end{pmatrix}.
$$

By letting permutations act on the left on the matrices $CB(b_i)$ (by permuting the variable entries), the ordered pairs $(CB(b_i)^{\pm 1}, \sigma_i)$ form a semi-direct product which satisfy the braid relations. This gives a representation of $B_N$.

Note the sparsity. This is why complexity scales linearly. It’s also why E-Multiplication can execute in one clock cycle in lightweight hardware.
E-Multiplication (denoted $\star$)

- $\mathbb{F}_q = \text{finite field of } q \text{ elements.}$
- $T = \{\tau_1, \ldots, \tau_N\} \subset (\mathbb{F}_q^\times)^N = \text{set of } T\text{-values.}$
- $m \in GL(N, \mathbb{F}_q), \ \sigma \in S_N.$

**E-multiplication by one Artin generator**

$$(m, \sigma) \star b_i = \begin{pmatrix} 1 & \cdots & \tau_{\sigma(i)} - \tau_{\sigma(i)} \cdots & 1 \\ m \cdot \tau_{\sigma(i)} \cdots & 1 \end{pmatrix}, \ \sigma \cdot (i, i + 1).$$

By iterating this computation we can compute the E-Multiplication of $(m, \sigma)$ with an arbitrary braid element (finite product of Artin generators and their inverses).
Laurent Polynomial entry: Short random word of length 10 in $B_4 \ltimes S_4$. What E-multiplication erases!
WalnutDSA Key Generation

WalnutDSA allows a signer with a fixed private/public key pair to create a digital signature associated to a given message which can be validated by anyone who knows the public key of the signer and the verification protocol.

**Public Information:**
- An integer $N \geq 8$ and associated braid group $B_N$.
- A rewriting algorithm $\mathcal{R} : B_N \rightarrow B_N$.
- A finite field $\mathbb{F}_q$ of $q \geq 32$ elements.
- $T$-values $= \{\tau_1, \tau_2, \ldots, \tau_N\}$ a set of invertible elements in $\mathbb{F}_q$.

**Signer’s Private Key:** $\text{Priv}(S) \in B_N$.

**Signer’s Public key:** $\text{Pub}(S) = (\text{Id, Id}) \star \text{Priv}(S)$
WalnutDSA Signature Generation

• $\mathcal{M}$ is a hash of a message.

• $E(\mathcal{M})$ is an encoding of $\mathcal{M}$ into the pure braid subgroup of $B_N$.

**Step 1:** Generate *cloaking elements* $v_1$ and $v_2$ which cloak $(\text{Id, Id})$ and $\text{Pub}(S)$, respectively.

**Step 2:** Compute $\text{Sig} = R(\text{Priv}(S)^{-1} \cdot v_1 \cdot E(\mathcal{M}) \cdot \text{Priv}(S) \cdot v_2)$, a rewritten braid.

**Step 3:** The final signature for the message $\mathcal{M}$ is the ordered pair $(\text{Sig}, \mathcal{M})$. 
Cloaking Elements

Let $m \in GL(N, \mathbb{F}_q)$ and $\sigma \in S_N$. An element $v$ in the pure braid subgroup of $B_N$ is termed a cloaking element of $(m, \sigma)$ if

$$(m, \sigma) \star v = (m, \sigma).$$

The cloaking element is defined by the property that it essentially disappears when performing E-Multiplication.

**Remark:** When E-Multiplication is viewed as a right action of $B_N$ on $GL(N, \mathbb{F}_q) \times S_N$ then cloaking elements are stabilizers which form a subgroup of $B_N$. 
WalnutDSA Signature Verification

*The signature* \((\text{Sig}, \mathcal{M})\) *is verified as follows:*

**Step 1:** Generate the encoded message \(E(\mathcal{M})\).

**Step 2:** Evaluate \(\text{Pub}(E(\mathcal{M})) := (\text{Id}, \text{Id}) \star E(\mathcal{M})\).

**Step 3:** Evaluate \(\text{Pub}(\text{S}) \star \text{Sig}\).

**Step 4:** *Verify the equality*

\[
\text{MatrixPart}(\text{Pub}(\text{S}) \star \text{Sig}) = \text{MatrixPart}(\text{Pub}(E(\mathcal{M}))) \cdot \text{MatrixPart}(\text{Pub}(\text{S})),
\]

where the matrix multiplication on the right is performed over the finite field. *The signature is valid if this equality holds. If the results are not equal then the signature validation has failed.*
Linear Running Time of Signature Verification

- The bulk of Signature Verification only requires E-Multiplication computations.

- E-multiplication by one Artin generator can be performed in one clock cycle.

- Most of the running time of Signature Verification is taken up by computing $L$ successive E-Multiplications, each by one Artin generator, where $L$ is the number of Artin generators of the Signature (Sig).

- WalnutDSA Signature Verification is extremely fast and the running time is linear in the Signature length.
Security Discussion
Reversing E-Multiplication Is Hard

- Braid group $B_N$ and symmetric group $S_N$ with $N \geq 8$.
- Finite field $\mathbb{F}_q$ with $q \geq 32$.
- $\beta \in B_N$.
- $(m, \sigma) = (\text{Id}, \text{Id}) \star \beta \in \text{GL}(N, \mathbb{F}_q) \times S_N$.

**Conjecture:** It is infeasible to determine $\beta$ from $(m, \sigma)$ if the normal form of $\beta$ is sufficiently long.

**Quantum Resistance:** As the length of the word $\beta$ increases, the complexity of the Laurent polynomials occurring in the E-multiplication increases exponentially. It does not seem to be possible that E-Multiplication exhibits any type of simple periodicity, so it is very unlikely that inverting E-Multiplication can be achieved with a polynomial quantum algorithm.
Security Discussion
Cloaked Conjugacy Search Problem (CCSP)

- The braid group $B_N$ and symmetric group $S_N$ with $N \geq 8$.
- $Y, v_1, v_2 \in B_N$.
- Assume $v_1$ cloaks $(\text{Id}, \text{Id})$ and $v_2$ cloaks $(\text{Id}, \text{Id}) \ast Y$.

**Conjecture (CCSP):** Assume $A \in B_N$ and $Y^{-1} v_1 A Y v_2$ are known. Then it is infeasible to determine $Y$ if the normal forms of $Y, v_1, v_2$ are sufficiently long.
Hidden Subgroup Problem The Hidden Subgroup Problem asks to find an unknown subgroup $H \leq G$ using calls to a known function on $G$ which takes distinct constant values on distinct cosets of $G/H$.

- Shor’s quantum attack breaking RSA and other public key protocols such as ECC are essentially equivalent to the fact that there is a successful quantum attack on the Hidden Subgroup Problem for finite cyclic groups.

- Since the braid group does not contain any non-trivial finite subgroups at all, there does not seem to be any viable way to connect CCSP with HSP.
Security Discussion

Grover's Quantum Search Algorithm (GQSA)

- GQSA finds an element in an unordered $N$ element set in time $\mathcal{O}(N^{1/2})$.

- GQSA can find the private key in a cryptosystem with a square root speed-up in running time and cuts the security in half.

- GQSA can be defeated by increasing the key size.

- WalnutDSA Signature Verification runs in linear time in the signature length. GQSA can be defeated by doubling the signature length, which results in double the computation time.

- By comparison ECDSA runs in quadratic time. Defeating GQSA requires a four fold increase in computation time.
Security Discussion

• The recent attack of Ben-Zvi–Blackburn–Tsaban (BBT) ("A practical cryptanalysis of the Algebraic Eraser," CRYPTO 2016, see also “Defeating the Ben-Zvi, Blackburn, and Tsaban Attack on the Algebraic Eraser”) does not apply to WalnutDSA because the signature is a braid and there are no commuting subgroups involved. Hence, the linear algebraic attacks as proposed in BBT to solve CCSP or to forge a signature are not applicable.

• Length attacks of the type proposed by Myasnikov–Ushakov (2009) or Garber-Kaplan-Teicher-Tsaban-Vishnu (2005) do not appear to facilitate solving CCSP if the keys are sufficiently large. First of all, as pointed out by Gunnells (2011), the length attack only works effectively for short conjugates, which are not used in implementations. Secondly, the placement of the unknown cloaking elements \(v_1, v_2\) in the braid word \(Y^{-1} v_1 A Y v_2\) seems to completely thwart any type of length attack for the conjugacy problem.
• Note that if the cloaking elements $v_1, v_2$ are trivial then CCSP reduces to the ordinary conjugacy search problem (CSP). If an attacker can determine the cloaking elements $v_1, v_2$ then it is easy to see that CCSP again reduces to CSP and fast methods for solving for $Y$ were obtained in Gebhardt “A new approach to the conjugacy problem in Garside groups,” (2005) provided the super summit set of the conjugate $Y$ was not too large. In applications one should never use cloaking elements that are trivial or close to trivial.
## Performance of WalnutDSA Verification

\( B_8 F_{32}, 2^{128} \) Security level (equivalent to ECC P256)

<table>
<thead>
<tr>
<th>Platform</th>
<th>Clock MHz</th>
<th>WalnutDSA ROM1</th>
<th>WalnutDSA RAM1</th>
<th>ECDSA ROM1</th>
<th>ECDSA RAM1</th>
<th>ECDSA Time (ms)</th>
<th>Gain (Time)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSP430</td>
<td>8</td>
<td>3244</td>
<td>236</td>
<td>46</td>
<td>20-30K</td>
<td>2-5K</td>
<td>21-63x</td>
</tr>
<tr>
<td>8051</td>
<td>24.5</td>
<td>3370</td>
<td>312</td>
<td>35.3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ARM M3</td>
<td>48</td>
<td>2952</td>
<td>272</td>
<td>5.7</td>
<td>7168</td>
<td>540</td>
<td>40.8x</td>
</tr>
<tr>
<td>FPGA</td>
<td>50</td>
<td>0.05</td>
<td></td>
<td></td>
<td></td>
<td>2.08</td>
<td>41.6x</td>
</tr>
</tbody>
</table>

1. ROM/RAM in Bytes
2. Time is in milliseconds.
Thank You!

SecureRF Corporation
100 Beard Sawmill Rd, Suite 350
Shelton, CT 06484
(203) 227-3151

Derek Atkins (datkins@securerf.com)