Lapin
(an efficient authentication protocol based on Ring-LPN)

Stefan Heyse, Eike Kiltz, Vadim Lyubashevsky,
Christof Paar, Krzysztof Pietrzak
Authentication Protocols

Prover
shared AES key $K$

Verifier

$AES_K(c)$

c

HB-style authentication protocols based on LPN

suitable for *light-weight* authentication
Lightweight Authentication - Motivation

Lightweight authentication has many applications

• “We need security with less than 2000 gates for RFID tags” Sanjay Sarma (MIT AUTO-ID Labs) @ CHES 2002

• $3 trillion damage annually due to product piracy*
  → replacement parts and devices need authentication

*Source: www.bascap.com

• Remote keyless entry systems for buildings, cars...
Lightweight Authentication - Motivation

• Many embedded applications are very cost-sensitive → we need lightweight authentication

• Since ≈ 2006 a lot of research on lightweight ciphers (PRESENT and many other proposals)

• All previous lightweight ciphers...
  – are optimized for hardware complexity (gate count), even though the vast majority of embedded applications run in software / firmware → very small code attractive for many applications
  – are not based on hardness assumptions
Learning Parity with Noise (LPN)

We have access to an oracle who has a secret \( s \) in \( \mathbb{Z}_2^n \)

On every query, the oracle:
1. Picks \( r \leftarrow \mathbb{Z}_2^n \)
2. Picks a `noise' \( e \leftarrow \beta_{\frac{1}{4}} \) (i.e. \( e = 0 \) w.p. \( \frac{3}{4} \) and \( 1 \) w.p \( \frac{1}{4} \))
3. Outputs \((r, t=<r,s> + e)\)

The goal: Find \( s \)
Decision LPN

Thm [BFKL ‘93]: Decision-LPN is as hard as LPN

can’t distinguish from uniform
HB Protocol [HB ‘01]

Prover

common secret $s$ in $\mathbb{Z}_2^n$

For $1 \leq j \leq k$
- generate $e_j \leftarrow \beta_{\frac{1}{4}}$
- set $t_j = \langle r_j, s \rangle + e_j$

Verifier

Pick $r_1, \ldots, r_k \leftarrow \mathbb{Z}_2^n$

Accept iff for more than 60% of $j$, $t_j = \langle r_j, s \rangle$

As secure as LPN against a passive adversary

$kn \approx 2^{18}$ bits!!
HB Protocol [HB ‘01]

Prover

- common secrets $s_1, \ldots, s_k$ in $\mathbb{Z}_2^n$

For $1 \leq j \leq k$
- generate $e_j \leftarrow \beta_{\frac{1}{4}}$
- set $t_j = \langle r, s_j \rangle + e_j$

Verifier

- Pick $r \leftarrow \mathbb{Z}_2^n$

Accept iff for more than $60\%$ of $j$, $t_j = \langle r, s_j \rangle$

As secure as LPN against a passive adversary

$kn \approx 2^{18}$ bits!!
HB Protocol + Toeplitz Matrix [GRS ‘08]

**Prover**

common secrets $s_1,...,s_k$ in $\mathbb{Z}_2^n$

For $1 \leq j \leq k$

- generate $e_j \leftarrow \beta_\frac{1}{4}$
- set $t_j = \langle r, s_j \rangle + e_j$

**Verifier**

Pick $r \leftarrow \mathbb{Z}_2^n$

Accept iff for more than 60% of $j$, $t_j = \langle r, s_j \rangle$

As secure as “Toeplitz-LPN” against a passive adversary

$k+n-1 \approx 2^{10}$ bits
HB Protocol + Ring (field) $\mathbb{Z}_2[x]/\langle f(x) \rangle$

**Prover**

common secrets $s_1, \ldots, s_k$ in $\mathbb{Z}_2^n$

For $1 \leq j \leq k$

- generate $e_j \leftarrow \beta_{\nicefrac{1}{2}}$
- set $t_j = <r, s_j> + e_j$

**Verifier**

Pick $r \leftarrow \mathbb{Z}_2^n$

Accept iff for more than 60% of $j$, $t_j = <r, s_j>$

As secure as “Ring-LPN” against a passive adversary

$\approx 2^9$ bits
HB Protocol + Field $\mathbb{Z}_2[x]/\langle x^4+x+1 \rangle$
Ring-LPN Problem

\[ f(x) = \text{polynomial of degree } n \]

\[ R = \mathbb{Z}_2[x]/\langle f(x) \rangle \]

(Decision) Ring-LPN problem

\[ s \leftarrow R \]

\[
\begin{align*}
r & \leftarrow R \\
e & \leftarrow \beta^n \\
t & = rs + e
\end{align*}
\]

Output \((r, t)\)

\[
\begin{align*}
r & \leftarrow R \\
t & \leftarrow R
\end{align*}
\]

Output \((r, t)\)

Distinguish between the two distributions
Hardness of Ring-LPN

- Very little known
- For irreducible $f(x)$, seems as hard as general LPN
- For reducible $f(x)$ ... one needs to be careful
  - $f(x) = x^n + 1$ (where $n$ is a power of 2), there is a $2^{\sqrt{n}}$ algorithm
- No known connection between decision and search versions
HB Protocol + Ring (field) $\mathbb{Z}_2[x]/\langle f(x) \rangle$

**Prover**
- common secret $s$ in $\mathbb{Z}_2[x]/(f(x))$
- generate $e \leftarrow \beta^n$
- set $t = rs + e$

**Verifier**
- Pick $r \leftarrow \mathbb{Z}_2[x]/(f(x))$
- $t$
- Accept iff $t + rs$ is 0 for more than 60% of the coefficients

As secure as “Ring-LPN” against a passive adversary

What about active attacks?
Active Attack Model

Prover

Adversary Phase 1
Active Attack Model

Adversary Phase 2

Verifier

Accept!

Adversary wins
HB Protocol with Active Security
[JW ‘05, KS ’06, GRS ’08, ...]

Prover

Verifier

secret size doubled

3 Rounds

security proof uses rewinding (not tight):

adversary succeeding with probability $\delta$ lets us break LPN with probability $\delta^2$
Our Result

• 2 round *efficient* protocol based on Ring-LPN

• Uses ideas from [KPCJV ‘10]
  – [KPCJV ‘10] is a 2-round LPN-based protocol
  – It suffers from the same efficiency drawback as HB
  – Don’t know if it can be instantiated with a Toeplitz matrix
## New Authentication Protocol

<table>
<thead>
<tr>
<th><strong>Prover</strong></th>
<th><strong>Verifier</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>common secrets ( s, s' ) in ( R=\mathbb{Z}_2[x]/\langle f(x) \rangle )</td>
<td>Pick ( c \leftarrow D )</td>
</tr>
<tr>
<td>( R^* ) is the set of all invertible elements in ( R )</td>
<td>Accept iff ( r ) is in ( R^* ) and more than ( \frac{3}{4} ) of the entries of ( z + r(sc+s') ) are 0</td>
</tr>
<tr>
<td>( D ) is a subset of ( R ) such that for all ( c \neq c' ) in ( D ), ( c+c' ) is in ( R^* )</td>
<td></td>
</tr>
</tbody>
</table>

- generate \( r \leftarrow R^* \)
- generate \( e \leftarrow \beta_n^{\frac{1}{8}} \)
- set \( z = r(sc+s')+e \)
Security Proof

\( c^* \leftarrow D, \ a \leftarrow R, \ s' = c^* s + a \)

Phase 1

\[(r', t = r's + e) \]
\[r = r'(c + c^*)^{-1} \]
\[z = t + ra \]
\[= r(sc + s') + e \]

Phase 2

\[t = r's + e \]

if \( r \) is in \( R^* \) and more than \( \frac{3}{4} \) of the entries of \( z + r(sc^* + s') \) are 0.
else

\[(r, z) \]
Performance Comparisons

8-bit AVR ATmega163 smartcard implementations

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Online Time (cycles)</th>
<th>Offline Time (cycles)</th>
<th>Code Size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>f(x)=x^{621}+... (reducible)</td>
<td>30,000</td>
<td>82,500</td>
<td>1356</td>
</tr>
<tr>
<td>f(x)=x^{532}+x+1 (irreducible)</td>
<td>21,000</td>
<td>174,000</td>
<td>459</td>
</tr>
<tr>
<td>AES-Based</td>
<td>10,121</td>
<td>0</td>
<td>4644</td>
</tr>
</tbody>
</table>
Open Problems

• Man-in-the-middle security?
  – There is a $2^{k/2}$ time MIM attack against our protocol (requires $2^{k/2}$ observations)
  – Can we design a practical protocol provably secure against man-in-the-middle attacks?
    • Big step taken in [DKPW ‘12]
    • Is Lapin already secure against MIM attacks?

• How hard is the Ring-LPN problem?
  – Is there a search-decision reduction?

• A 2-round protocol with Toeplitz matrices?

Thank You!