Securing Information for Encrypted Verification and Evaluation (SI EVE)

Dr. Joshua Baron

Proposers Day

17 July 2019
SIEVE Goal

Develop computer science theory and software to create mathematically verifiable public statements derived from hidden, sensitive information in order to publically yet securely communicate about DoD capabilities

Enable verification while keeping secrets
SI EVE Overview

• 48 month effort with three phases (18 months/18 months/12 months)

• Three TAs to bid on: TA1, TA2 and TA3

• Multiple awards anticipated for each TA

• Abstract deadline: July 31, 12:00PM ET (HIGHLY recommended)

• Full Proposal deadline: September 20, 12:00PM ET

• One TA per abstract (can submit multiple abstracts to same or multiple TAs)

• TA1 and TA2 proposals can be combined, TA3 must be bid separately
  • Grants are allowable for TA3 ONLY

• Email: SIEVE@darpa.mil
A zero-knowledge proof is an interactive protocol that takes place between a prover and a verifier
• The Prover has a statement that they wish to have a Verifier accept
• The secret is hidden information that the Prover leverages to prove the statement

A zero knowledge proof must satisfy three security properties:
• Completeness: if the statement is true, the verifier will accept the proof
• Soundness: if the statement is false, the verifier will not accept the proof
• Zero knowledge: the verifier only learns whether the statement is true or not
  • The verifier doesn’t learn anything about the secret

Citation: [GMR85]
Verifiable Public Statements Derived From Hidden, Sensitive Information

• Statements about software
  • Prove that a system is vulnerable \textit{without ever revealing the vulnerability/exploit}

• Statements about computation
  • (Algorithm) Proving data provenance \textit{without ever revealing its source or how it was processed}
  • (Inputs) Prove that a machine learning classifier was created following appropriate regulations 
\textit{without ever revealing any of the training data}

• Statements about socio-technical interactions
  • Prove that the North Koreans have exploited a particular system \textit{without ever revealing USG intelligence on North Korean capabilities}

![USCYBERCOM Malware Alert](CNMF_VirusAlert Nov 9)
@CNMF_VirusAlert

\#CNMF has posted new malware to @VirusTotal

virus.com/en/file/dea3a9...

\[ 3 \quad \text{\#31} \quad \text{\#53} \]
Example: Finding Sensitive Malware

Prover wants to demonstrate if Verifier’s code sample matches one of her published, encrypted malware signatures without revealing anything about the signature.*

Problem Statement

Encrypt(Malware signature₁)
Encrypt(Malware signature₂)
... Encrypt(Malware signature₁₀)

“Verifier’s code matches at least one of the encrypted malware signatures.”

Intermediate Representation

Boolean Circuit:
OR of each of the signature matching subcircuits

Verification

* Signatures are in the form of string matching with wildcards
Example: Finding Sensitive Malware... Slowly

Prover wants to demonstrate if Verifier’s code sample matches one of her published, encrypted malware signatures without revealing anything about the signature.*

**Problem Statement**

Encrypt(Malware signature\(_1\))
Encrypt(Malware signature\(_2\))
... Encrypt(Malware signature\(_{10}\))

“Verifier’s code matches at least one of the encrypted malware signatures.”

**Intermediate Representation**

Boolean Circuit:
OR of each of the signature matching subcircuits
Total circuit size is \(\sim 10^{11}\) gates for a 100 kbit code sample

**Verification**

Proof takes \(~1\) month**

* Signatures are in the form of string matching with wildcards

** Extrapolated from [NT16]
SI EVE Program Vision

Problem Statement

```
#include "myutil.h"

void myfunc(const double a[12], const double b[12], double &c[12])
{
    int i;
    int j;
    int k;
    for (i=0; i<12; i++)
    {
        for (j=0; j<12; j++)
        {
            k = (i*12 + j)*a[12] + b[i*12 + j];
        }
    }
}
```

Intermediate Representation (IR)

Encode

Input

Zero Knowledge Proof

Prover

Verifier
TA1: Constructing Useful Zero Knowledge Statements

Problem Statement

```
#include "myfile.h"

void myfunc(const double x[12], const double y[20], double z[13])
{
    // Implementation
}
```

Encode

Intermediate Representation (IR)

Zero Knowledge Proof

TA2: Building Efficient Zero Knowledge Proof Generation Compilers

TA3: Post-Quantum Zero Knowledge
## Challenge: Efficient ZK Proofs for Complex Problem Statements

<table>
<thead>
<tr>
<th>Property</th>
<th>Current state of the art</th>
<th>SI EVE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intermediate Representation (IR)</td>
<td>Classes of DoD-relevant statements expressible in an IR: Few and trivial</td>
<td>Broad and complex</td>
</tr>
<tr>
<td>Expressivity</td>
<td>Statements can be constructed from: Straight-line code</td>
<td>Probabilistic, indeterminate-branching conditions</td>
</tr>
<tr>
<td>Zero Knowledge (ZK) Proof Efficiency</td>
<td>ZK proof complexity, in terms of IR statement size: Superlinear</td>
<td>Linear</td>
</tr>
<tr>
<td></td>
<td>ZK prover efficiency is: .1 msec/gate</td>
<td>.1 µs/gate</td>
</tr>
<tr>
<td>Post-Quantum (PQ) ZK Capability</td>
<td>ZK in the post-quantum setting is: Theoretically possible</td>
<td>Asymptotically efficient</td>
</tr>
</tbody>
</table>
TA1: Constructing Useful Zero Knowledge Statements

- **Objective:** Create theory and algorithms to create ZK proof statements from complex, real-world problems
  - Statements about: Software, Computation, Sociotechnical
  - You do NOT have to cover all or any of the above areas or address the specific examples within those areas listed in the BAA
    - Abstracts HIGHLY recommended to help ensure your proposed problems to address are of interest to us

- **Challenge:** Creative means of efficiently and rigorously encapsulated the problem. Generality of statement vs complexity
## TA1 Metrics

<table>
<thead>
<tr>
<th>Metric</th>
<th>Phase 1 (18 mo)</th>
<th>Phase 2 (18 mo)</th>
<th>Phase 3 (12 mo)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem Classes Addressed</td>
<td>1 class (e.g., exploit class)</td>
<td>3 classes</td>
<td>5 classes</td>
</tr>
<tr>
<td>Intermediate Representation (IR) Size</td>
<td>Viable representation in an IR</td>
<td>10x smaller than previous phase</td>
<td>10x smaller than previous phase</td>
</tr>
<tr>
<td>Statement complexity handled</td>
<td>Loops (do/for)</td>
<td>Conditionals (while)</td>
<td>Probabilistic events</td>
</tr>
</tbody>
</table>
Strong TA1 Proposals Will:

• Demonstrate why, for a particular class of problem statement, they possess both sufficient domain expertise and the ability to create encodings for that problem.

• Present a compelling argument grounded in the current state of the art for how their IRs will enable efficient ZK proofs for their statements. (Make the encoding work with ZK)

• Present innovative IR encoding approaches that may reduce ZK computational complexity while meeting the needs of the program. (Create efficient representations that correspondingly require less ZK work)

• Describe the interplay of problem statement expressivity versus security leakage (e.g., “fact of” creating a statement for a particular class of problem narrows interest to that specific class).
Out of Scope for TA1

• Statements whose primary relevance is for cryptocurrency transactions are out of scope
**Objective:** Develop efficient zero knowledge proofs from existing proof statements

**Challenge:** Be able to construct 1) efficient proofs that 2) support multiple intermediate representations

---

**Diagram:**

1. ZK Statement
2. IR Translation
3. Collection of Efficient ZK Proof Generation Algorithms
4. Efficient ZK Proof for the Statement

**Diagram Notes:**
- Desired ZK Proof Parameters
TA2: Details about Effectiveness, Efficiency and Security

- TA2 performers will have to work with TA1 performers to ensure that they can create proofs for all TA1-encoded problem statements.
  - To that end, all TA2 performers will work together with TA1 and the T&E team to create a set of common standards for IRs (e.g., one each for Boolean circuits, arithmetic circuits, and rank 1 constraint systems), in order to ensure that all TA1-generated IRs can be easily used by all TA2 performers.

- It is anticipated that ZK proofs that take a day over powerful computers and a high-bandwidth network may be realistic for some DoD-relevant applications, and completely inappropriate in others. Proposals should carefully discuss the degree to which their solutions are appropriate to different possible real-world settings.

- Proposals that minimize security assumptions are strongly desired. These assumptions include: setup (e.g., common random or reference string), non-standard hardness (e.g., some bi/multilinear group-based assumptions), non-falsifiable hardness (e.g., knowledge of exponent, (programmable) random oracle), and adversary restrictions (e.g., generic or algebraic group model).
**TA2 Metrics**

<table>
<thead>
<tr>
<th>Metric</th>
<th>Phase 1 (18 mo)</th>
<th>Phase 2 (18 mo)</th>
<th>Phase 3 (12 mo)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZK Total Asymptotic Complexity#</td>
<td>$O(n \cdot \log(k) + k)$</td>
<td>$O(n+k)$</td>
<td>100x smaller than Phase 2</td>
</tr>
<tr>
<td>Prover/Verifier Computation</td>
<td>$&lt;10\mu s/gate$</td>
<td>$&lt;1\mu s/gate$</td>
<td>$&lt;.1\mu s/gate$</td>
</tr>
<tr>
<td>Communication Complexity#</td>
<td>$&lt;100 \cdot n \cdot \log(k)$ bits</td>
<td>$&lt;10 \cdot (n+k)$ bits</td>
<td>$&lt;10 \cdot (\sqrt{n}+k)$ bits</td>
</tr>
</tbody>
</table>

# $n = \text{statement size in the IR}$, $k = \text{security parameter}$
Strong TA2 Proposals Will:

- For each class of ZK proof they propose to create (e.g., non-interactive vs constant round), justify a DoD-relevant employment scenario and carefully delineate any security assumptions they may require.

- Provide a basis to justify the efficiency of their proposed solutions, e.g., in terms of per-gate performance or other properties of an IR that make it more or less amenable to their approach.

- Justify the concrete efficiency (e.g., bits of communication and seconds of computation) for the ZK proof classes that they wish to create. Proposals should specify the types of computers and networks that underlie such calculations, as well as how IR structure and/or size might have an impact.
Out of Scope for TA2

- Topics of research that are specifically out of scope for TA2 include:
  - Reliance on specialized (e.g. FPGA or ASIC) or secure (e.g., Intel SGX) hardware;
  - Cryptanalysis (other than security proofs for provided systems);
• **Objective:** Ensure viability of zero knowledge proofs if cryptography-relevant quantum computers become a reality

- Post-Quantum (PQ) Hardness Assumptions
  - ZK from PQ assumptions
  - ZK about PQ-type math

• **Challenge:** Creating asymptotically efficient zero knowledge proofs from or about PQ assumptions
<table>
<thead>
<tr>
<th>Metric</th>
<th>Phase 1 (18 mo)</th>
<th>Phase 2 (18 mo)</th>
<th>Phase 3 (12 mo)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TA3: LWE proof security&amp;&lt;sup&gt;6&lt;/sup&gt;, total complexity</td>
<td>$1/k^2 \cdot O(k^{\text{polylog}(k)})$</td>
<td>$1/k^c \cdot O(k^{\log(k)})$</td>
<td>$1/\exp(k) \cdot O(k)$</td>
</tr>
<tr>
<td>TA3: PQ NIZK complexity</td>
<td>$O(k^4)$</td>
<td>$O(k^2)$</td>
<td>$O(k)$</td>
</tr>
</tbody>
</table>

$k =$ security parameter

\& LWE = Learning with errors, “proof” = Sigma protocol, “security” = soundness, all other security properties are at least computationally small.
**TA3 Scope**

- While the metrics outlines two specific directions of research in PQ ZK, other directions in efficient PQ ZK may be in scope;
  - Abstracts are HIGHLY recommended

- While the LWE hardness assumption is explicitly in scope, proposals that address ZK for other plausibly PQ assumptions are also in scope, provided that they enjoy widespread academic acceptance as plausibly PQ.
  - Hardness assumptions that have efficient reductions to standard hardness assumptions, such as the shortest independent vector problem, are preferable to less-standard PQ hardness assumptions (e.g., search-LWE).

- Topics of research that are specifically out of scope for TA3 include:
  - Reliance on non-falsifiable hardness assumptions (e.g., random oracle model)
  - The development or security analysis of novel PQ hardness assumptions
  - The development or security analysis of PQ encryption schemes
SI EYE Test and Evaluation

• A test and evaluation (T&E) team will evaluate the correctness of TA1 IRs and the efficiency of TA2 ZK proofs. The T&E team will advise the IR standards document creation process.

• The T&E team will specify a reference architecture upon which TA2 solutions will be evaluated, including when they are used with TA1-generated IRs.

• The current intent is for the T&E team to host the specific system and network that comprises the reference architecture; TA2 teams will be required to enable T&E members to run their software, possibly with on-site assistance from TA2 team members.

• The evaluation of TA3 technologies will largely depend on the peer review process for publication in prominent conferences and journals.

<table>
<thead>
<tr>
<th>Testing emphasis</th>
<th>Phase 1 (18 mo.)</th>
<th>Phase 2 (18 mo.)</th>
<th>Phase 3 (12 mo.)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Investigate ability for TA2 to integrate TA1-generated IRs; IR(s) used to evaluate TA2 technologies may be T&amp;E-team generated or some reduced-complexity TA1-generated IRs</td>
<td>Test integrated solutions that use selected TA1-generated IRs with TA2-generated ZK proofs</td>
<td>Scale up testing of integrated solutions that use any TA1-generated IRs with TA2-generated ZK proofs</td>
</tr>
</tbody>
</table>
## SIEVE Metrics

<table>
<thead>
<tr>
<th>Metric</th>
<th>Phase 1 (18 mo)</th>
<th>Phase 2 (18 mo)</th>
<th>Phase 3 (12 mo)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TA1: Problem Classes Addressed</td>
<td>1 class (e.g., exploit class)</td>
<td>3 classes</td>
<td>5 classes</td>
</tr>
<tr>
<td>TA1: Intermediate Representation (IR) Size</td>
<td>Viable representation in an IR</td>
<td>10x smaller than previous phase</td>
<td>10x smaller than previous phase</td>
</tr>
<tr>
<td>TA1: Statement complexity handled</td>
<td>Loops (do/for)</td>
<td>Conditionals (while)</td>
<td>Probabilistic events</td>
</tr>
<tr>
<td>TA2: ZK Total Asymptotic Complexity#</td>
<td>$O(n \times \log(k) + k)$</td>
<td>$O(n+k)$</td>
<td>100x smaller than Phase 2</td>
</tr>
<tr>
<td>TA2: Prover/Verifier Computation</td>
<td>&lt;10µs/gate</td>
<td>&lt;1µs/gate</td>
<td>&lt;.1µs/gate</td>
</tr>
<tr>
<td>TA2: Communication Complexity#</td>
<td>&lt;100<em>n</em>\log(k) bits</td>
<td>&lt;10*(n+k) bits</td>
<td>&lt;10*(\sqrt{n}+k) bits</td>
</tr>
<tr>
<td>TA3: LWE proof security, total complexity</td>
<td>$1/k^2, O(k^*\text{polylog}(k))$</td>
<td>$1/k^c, O(k^*\log(k))$</td>
<td>$1/\exp(k), O(k)$</td>
</tr>
<tr>
<td>TA3: PQ NIZK complexity</td>
<td>$O(k^4)$</td>
<td>$O(k^2)$</td>
<td>$O(k)$</td>
</tr>
</tbody>
</table>

* $n$ = statement size in the IR, $k$ = security parameter

& LWE = Learning with errors, “proof” = Sigma protocol, “security” = soundness, all other security properties are at least computationally small