Securing Information for Encrypted Verification and Evaluation (SIEVE)

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

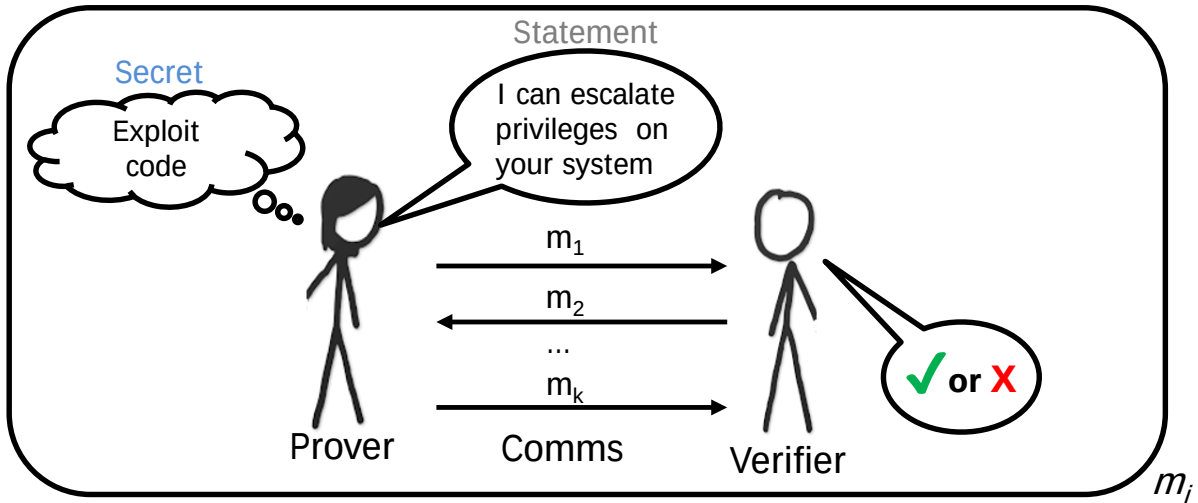
SIEVE Overview

- 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

Proving Verifiable Statements using Zero Knowledge

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



m_i = ith *message*

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 *without ever revealing the vulnerability/exploit*

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

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

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.*

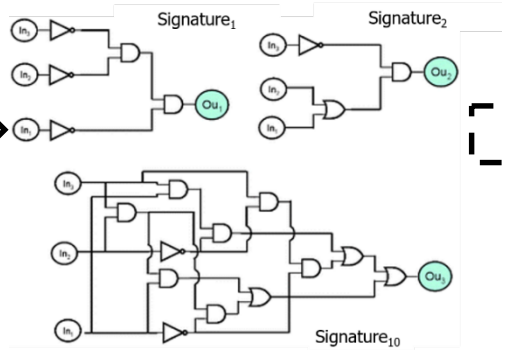
Problem Statement

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

"Verifier's code matches at least one of the encrypted malware signatures."

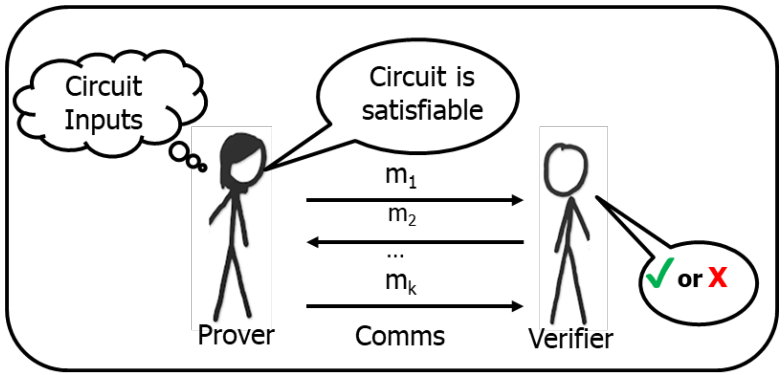
Encode

Intermediate Representation



Input

Verification



Boolean Circuit:
OR of each of the signature matching subcircuits

* 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.*

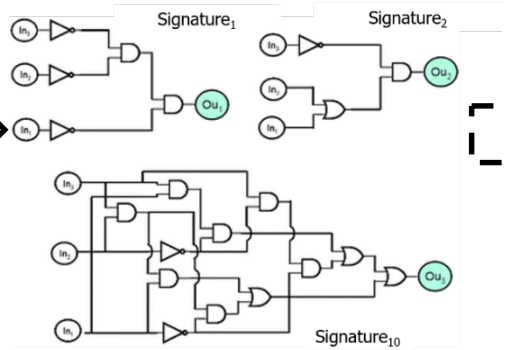
Problem Statement

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

"Verifier's code matches at least one of the encrypted malware signatures."

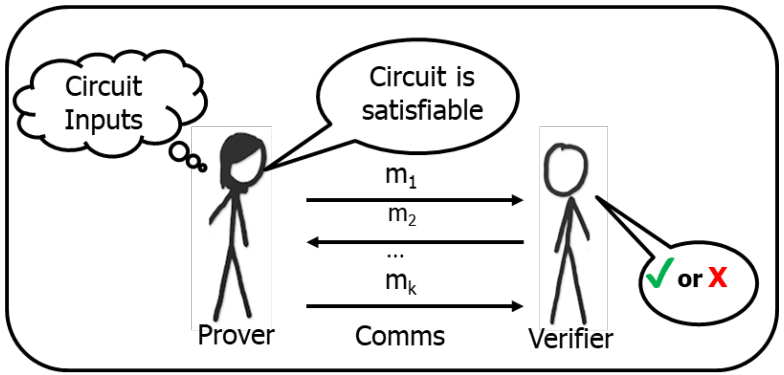
Encode

Intermediate Representation



Input

Verification



Proof takes ~1 month**

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

** Extrapolated from [NT16]

* Signatures are in the form of string matching with wildcards

SIEVE Program Vision

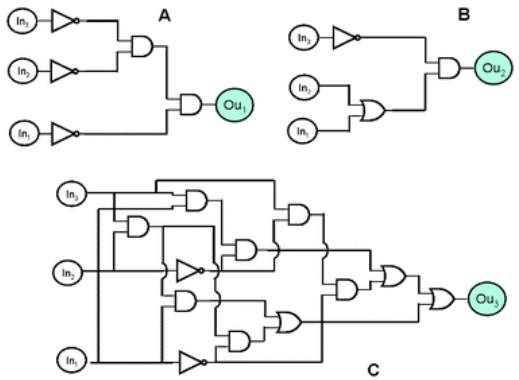
Problem Statement

```
#include "myMult.h"

void myMult(const double a[12], const double b[20], double c[15])
{
  int i0;
  int i1;
  int i2;
  for (i0 = 0; i0 < 3; i0++) {
    for (i1 = 0; i1 < 5; i1++) {
      c[i0 + 3 * i1] = 0.0;
      for (i2 = 0; i2 < 4; i2++) {
        c[i0 + 3 * i1] += a[i0 + 3 * i2] * b[i2 + (i1 << 2)];
      }
    }
  }
}
```

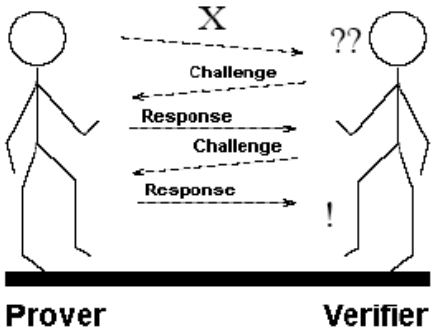
Encode

Intermediate Representation (IR)



Input

Zero Knowledge Proof



SIEVE Program Vision

TA1: Constructing Useful Zero Knowledge Statements

TA2: Building Efficient Zero Knowledge Proof Generation Compilers

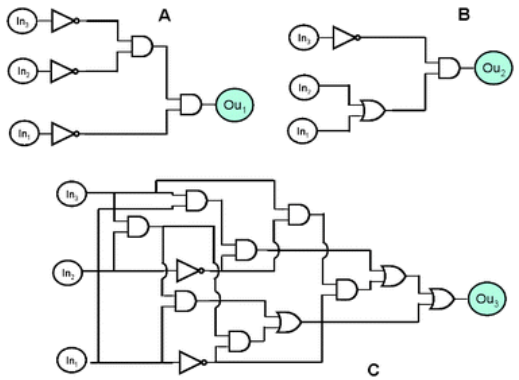
Problem Statement

```
#include "myMult.h"

void myMult(const double a[12], const double b[20], double c[15])
{
  int i0;
  int i1;
  int i2;
  for (i0 = 0; i0 < 3; i0++) {
    for (i1 = 0; i1 < 5; i1++) {
      c[i0 + 3 * i1] = 0.0;
      for (i2 = 0; i2 < 4; i2++) {
        c[i0 + 3 * i1] += a[i0 + 3 * i2] * b[i2 + (i1 << 2)];
      }
    }
  }
}
```

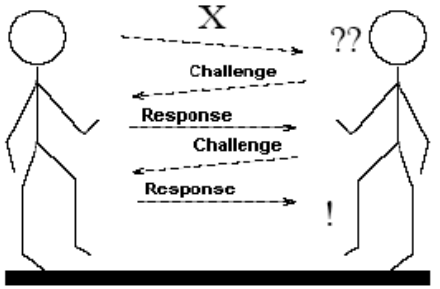
Encode

Intermediate Representation (IR)



Input

Zero Knowledge Proof



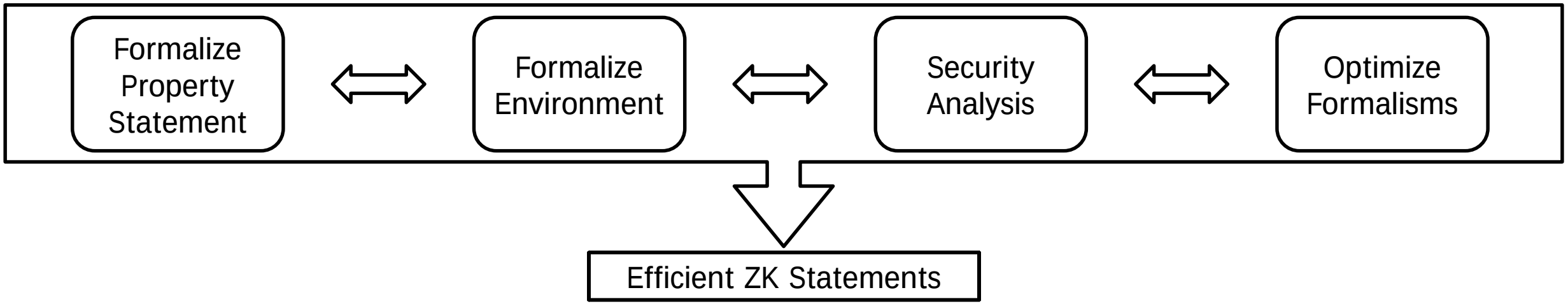
TA3: Post-Quantum Zero Knowledge

Challenge: Efficient ZK Proofs for Complex Problem Statements

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

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

Metric	Phase 1 (18 mo)	Phase 2 (18 mo)	Phase 3 (12 mo)
Problem Classes Addressed	1 class (e.g., exploit class)	3 classes	5 classes
Intermediate Representation (IR) Size	Viable representation in an IR	10x smaller than previous phase	10x smaller than previous phase
Statement complexity handled	Loops (do/for)	Conditionals (while)	Probabilistic events

Strong TA1 Proposals Will:

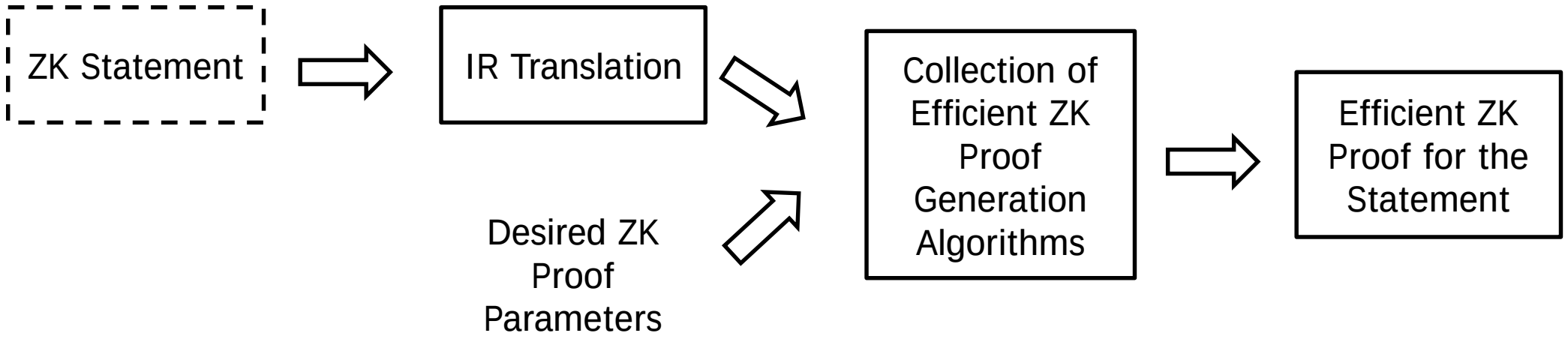
- Demonstrate why, for a particular class of problem statement, they possess both sufficient domain expertise and the ability to create encodings for that problem.
- Present a compelling argument grounded in the current state of the art for how their IRs will enable efficient ZK proofs for their statements. (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

TA2: Building Efficient Zero Knowledge Proof Generation Compilers

- **Objective:** Develop efficient zero knowledge proofs from existing proof statements



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

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

Metric	Phase 1 (18 mo)	Phase 2 (18 mo)	Phase 3 (12 mo)
ZK Total Asymptotic Complexity[#]	O(n*log(k)+k)	O(n+k)	100x smaller than Phase 2
Prover/Verifier Computation	<10µs/gate	<1µs/gate	<.1µs/gate
Communication Complexity[#]	<100*n*log(k) bits	<10*(n+k) bits	<10*(sqrt(n)+k) bits

[#] n = statement size in the IR, k = 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.

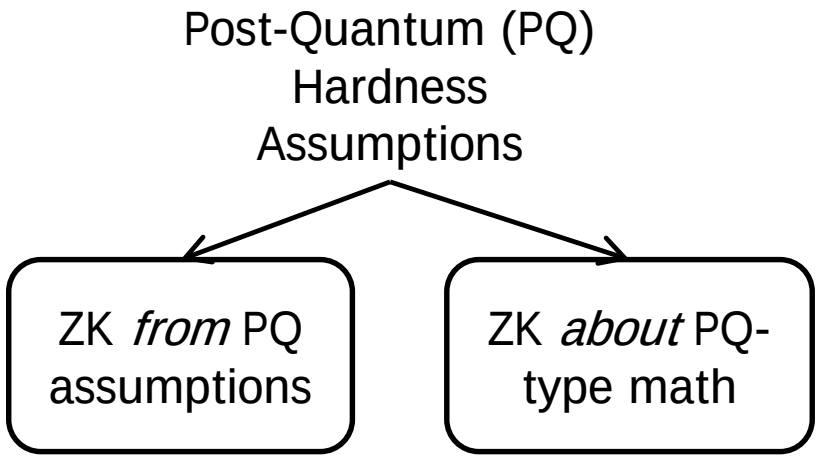
- Justify the concrete efficiency (e.g., bits of communication and seconds of computation) for the ZK proof classes that they wish to create. Proposals should specify the types of computers and networks that underlie such calculations, as well as how IR structure and/or size might have an impact.

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);

TA3: Post-Quantum Zero Knowledge

- **Objective:** Ensure viability of zero knowledge proofs if cryptography-relevant quantum computers become a reality



- **Challenge:** Creating asymptotically efficient zero knowledge proofs from or about PQ assumptions

TA3 Metrics

Metric	Phase 1 (18 mo)	Phase 2 (18 mo)	Phase 3 (12 mo)
TA3: LWE proof security[&], total complexity	$1/k^2$, O(k*polylog(k))	$1/k^c$, O(k*log(k))	1/exp(k) , O(k)
TA3: PQ NIZK complexity	$O(k^4)$	$O(k^2)$	O(k)

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

SIEVE 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.

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

SIEVE Metrics

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

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

www.darpa.mil