CISPA
HELMHOLTZ CENTER FOR INFORMATION SECURITY

Research using the Tamarin Prover

Session 4 - Summer School on real-world crypto and privacy 2024

Research using the Tamarin Prover

The two main categories

Research using the Tamarin Prover

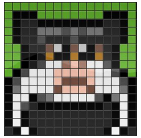
The two main categories

Improving the Tool / Expressibility

Security Protocol Analysis

Research using the Tamarin Prover

The two main categories



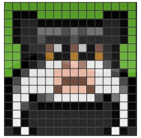
Improving the Tool / Expressibility

- Automatic Lemma Generation
- Natural Numbers
- Subterm Reasoning
- Custom Proof Tactics
- Better Models for Crypto Primitives

Security Protocol Analysis

Research using the Tamarin Prover

The two main categories



Improving the Tool / Expressibility

- Automatic Lemma Generation
- Natural Numbers
- Subterm Reasoning
- Custom Proof Tactics
- Better Models for Crypto Primitives

Security Protocol Analysis

100+ Models published, e.g.,

Research using the Tamarin Prover

The two main categories

Improving the Tool / Expressibility

- Automatic Lemma Generation
- Natural Numbers
- Subterm Reasoning
- Custom Proof Tactics
- Better Models for Crypto Primitives

Security Protocol Analysis

100+ Models published, e.g.,

- TLS 1.3
- 5G AKA
- WPA2
- Wireguard
- EMV

Research using the Tamarin Prover

The two main categories

Improving the Tool / Expressibility

- Automatic Lemma Generation
- Natural Numbers
- Subterm Reasoning
- Custom Proof Tactics
- **Better Models for Crypto Primitives**

I will talk about this now!

Security Protocol Analysis

100+ Models published, e.g.,

- TLS 1.3
- 5G AKA
- WPA2
- Wireguard
- EMV

Research using the Tamarin Prover

The two main categories

Improving the Tool / Expressibility

- Automatic Lemma Generation
- Natural Numbers
- Subterm Reasoning
- Custom Proof Tactics
- **Better Models for Crypto Primitives**

I will talk about this now!

Security Protocol Analysis

100+ Models published, e.g.,

- TLS 1.3
- 5G AKA
- WPA2
- Wireguard
- **EMV**

You will hear about EMV later again!

“Historic” Examples of Interesting Attacks

“Historic” Examples of Interesting Attacks

- **1999: Duplicate Signature Key Selection (DSKS) attacks**

  Given any (e.g. RSA) signature, you can create a second key pair whose verification key also verifies that same signature??

  (Related: unique ownership)

“Historic” Examples of Interesting Attacks

- **1999: Duplicate Signature Key Selection (DSKS) attacks**

  Given any (e.g. RSA) signature, you can create a second key pair whose verification key also verifies that same signature??

  (Related: unique ownership)

- **2009: Length Extension Attacks**

  One can use Hash(m1) and the length of m1 to calculate Hash(m1 ǁ m2) for some other message m2 without knowing m1

“Historic” Examples of Interesting Attacks

- **1999: Duplicate Signature Key Selection (DSKS) attacks**

  Given any (e.g. RSA) signature, you can create a second key pair whose verification key also verifies that same signature??
  (Related: unique ownership)

- **2009: Length Extension Attacks**

  One can use Hash(m1) and the length of m1 to calculate Hash(m1 ‖ m2) for some other message m2 without knowing m1

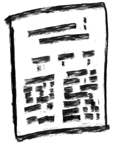
- **2014: Small subgroups**

  Diffie-Hellman protocols expect to receive an element of a prime order group, but often don't check this. *This is usually not a problem?*

First Idea

2016

Let's write a paper!

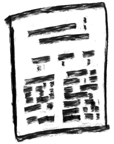


"Better Dolev-Yao abstractions of cryptographic primitives"

First Idea

2016

Let's write a paper!



"Better Dolev-Yao abstractions of cryptographic primitives"

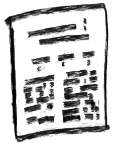
Plan:

- Revisit all Dolev-Yao primitives
  (signatures, exponentiation, encryption, hashes, etc.)
- Make better versions
- Submit
- ???
- Profit!!

First Idea

2016

Let's write a paper!



"Better Dolev-Yao abstractions of cryptographic primitives"

Plan:

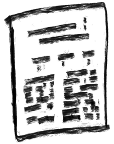
- ~~Revisit all Dolev-Yao primitives (signatures, exponentiation, encryption, hashes, etc.)~~
- ~~Make better versions~~
- ~~Submit~~
- ~~???~~
- ~~Profit!!~~

Too hard!

First Idea

2016

Let's write a paper!



"Better Dolev-Yao abstractions of cryptographic primitives"

Plan:

- ~~Revisit all Dolev-Yao primitives (signatures, exponentiation, encryption, hashes, etc.)~~
- ~~Make better versions~~
- ~~Submit~~
- ~~???~~
- ~~Profit!!~~

Too hard!

Let's start with the easiest thing, **signatures**

Let’s start with Signatures

Let's start with Signatures

Definition: Signature Scheme

A signature scheme **(gen,sign,verify)** is a triple of algorithms:

gen():
randomized alg. outputs a key pair **(pk, sk)**

sign(msg∈M, sk):
outputs signature sig

verify(sig, msg, pk):
outputs 'accept or 'reject'

Let’s start with Signatures

Definition: Signature Scheme

A signature scheme **(gen,sign,verify)** is a triple of algorithms:

gen():
randomized alg. outputs a key pair **(pk, sk)**

sign(msg∈M, sk):
outputs signature sig

verify(sig, msg, pk):
outputs ‘accept or ‘reject’

Correctness

for all **(pk, sk)** output by **gen()**
and for all **msg**∈M:

verify(sign(msg, sk), msg, pk) = ‘accept’

Let’s start with Signatures

Definition: Signature Scheme

A signature scheme **(gen,sign,verify)** is a triple of algorithms:

gen():
randomized alg. outputs a key pair **(pk, sk)**

sign(msg∈M, sk):
outputs signature sig

verify(sig, msg, pk):
outputs ‘accept or ‘reject’

Correctness

for all **(pk, sk)** output by **gen()**
and for all **msg**∈M:

verify(sign(msg, sk), msg, pk) = ‘accept’

Unforgeability

The adversary cannot generate a valid pair **(msg,sig)** that verifies using **pk** with **(pk, sk)** being the output by **gen()** and not knowing **sk**

How do we model Signatures?

How do we model Signatures?

```
functions:   verify/2, sign/2, pk/1

equations:   verify(sign(DATA,A),DATA,pk(A)) = true
```

How do we model Signatures?

functions: `verify/2, sign/2, pk/1`

equations:

$$\texttt{verify}(\overbrace{\texttt{sign}(\underbrace{\texttt{DATA,A}}_{\text{The Message}})}^{\text{The Signature}},\texttt{DATA},\overbrace{\texttt{pk(A)}}^{\text{The Signer}}) = \underbrace{\texttt{true}}_{\text{The Result}}$$

How do we model Signatures?

```
functions:   verify/2, sign/2, pk/1
```

$$\texttt{equations:}\quad \mathtt{verify}(\overbrace{\mathtt{sign}(\underbrace{\mathtt{DATA}}_{\textit{The Message}},\mathtt{A})}^{\textit{The Signature}},\mathtt{DATA},\overbrace{\mathtt{pk(A)}}^{\textit{The Signer}})\ =\ \underbrace{\mathtt{true}}_{\textit{The Result}}$$

First published in 2001, used by all contemporary tools

Thinking back: Key Substitution

1999: Key Substitution [Blake-Wilson, Menezes]

Given **sig**, **pk**, and **msg**:
Calculate (**sk'**,**pk'**) such that (**sig**,**msg**,**pk'**) verifies

Thinking back: Key Substitution

1999: Key Substitution [Blake-Wilson, Menezes]

Given **sig**, **pk**, and **msg**:
Calculate (**sk'**,**pk'**) such that (**sig**,**msg**,**pk'**) verifies

2005: Exclusive Ownership [Pornin, Stern]

A signature fails to provide **Conservative Exclusive Ownership** (CEO) if there is an efficient algorithm **`CEOgen(pk,(sig, msg)_i)`** outputs a new keypair (sk',pk'), s.t., **`verify(sig_j,msg_j,pk')=true`** for some j.

Applies to, e.g., RSA-PKCSv1.5, RSS-PSS, DSA, ECDSA with Free BP

Modelling: Key Substitution Attacks

Modelling: Key Substitution Attacks

```
functions: verify/2, sign/2, pk/1

equations: verify(sign(DATA,A),DATA,pk(A)) = true
```

Modelling: Key Substitution Attacks

```
functions: verify/2, sign/2, pk/1, CEOgen/1

equations: verify(sign(DATA,A),DATA,pk(A)) = true
```

Modelling: Key Substitution Attacks

```
functions: verify/2, sign/2, pk/1, CEOgen/1

equations: verify(sign(DATA,A),DATA,pk(A)) = true


equations: verify(sign(DATA,A),DATA,pk(CEOgen(sign(DATA,A)))) = true
```

Modelling: Key Substitution Attacks

```
functions: verify/2, sign/2, pk/1, CEOgen/1

equations: verify(sign(DATA,A),DATA,pk(A)) = true
```

$$\texttt{equations: verify(}\overbrace{\texttt{sign(DATA,A)}}^{\textit{The Signature}}\texttt{,}\underbrace{\texttt{DATA}}_{\textit{The Message}}\texttt{,pk(}\overbrace{\texttt{CEOgen(sign(DATA,A))}}^{\textit{Generating a new secret key from signature}}\texttt{)) = true}$$

Other Attacks on Signature Schemes

Destructive Exclusive Ownership (DEO) Attack

A CEO attack where the attacker can additionally choose/change the message

Colliding

The attacker can produce a **sig** and **pk**, s.t., **sig** verifies different messages using **pk**

Re-signing

Without knowing the message, the attacker can resign a given **sig** under different **sk**

Malleability

The attacker can create different signatures that verify under the same **(m,pk)**

Other Attacks on Signature Schemes

CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

Destructive Exclusive Ownership (DEO) Attack

```
verify(sign(m1,sk),m2,pk(DEOgen(sign(m1,sk),m2))) = true
```

Colliding

```
verify(sign(n,x),m,pk(weak(x))) = true
```

Re-signing

```
resign(sign(m,sk1),sk2) = sign(m,sk2)
```

Malleability

```
mutate(sign(m,r1,sk),r2)) = sign(m,r2,sk)
```

Was that it?

Was that it?

No! We only enumerated attacks…

We should find a more general approach!

Better Signature Model (using Restrictions)

Better Signature Model (using Restrictions)

1. We remove the equational theory (`verify`)

Better Signature Model (using Restrictions)

1. We remove the equational theory (**`verify`**) and introduce new *step labels*:

`verified(sig,m,pk,result)` **`result ∈ {true,false}`**

Better Signature Model (using Restrictions)

1. We remove the equational theory (**`verify`**)
   and introduce new *step labels*:

   `verified(sig,m,pk,result)`   **`result ∈ {true,false}`**

   `honest(pk)`

2. Any step where an honest party generates a public key, we label it with '**honest**'.

Better Signature Model (using Restrictions)

1. We remove the equational theory (**`verify`**) and introduce new *step labels*:

   `verified(sig,m,pk,result)`   **`result ∈ {true,false}`**

   `honest(pk)`

$$\text{State 1} \xrightarrow{\texttt{verified(sig,m,pk,true)}} \text{State 2}$$

2. Any step where an honest party generates a public key, we label it with ‘**honest**’.

Note:
We only get guarantees for keys from **gen()**!

Better Signature Model (using Restrictions)

1. We remove the equational theory (**`verify`**) and introduce new *step labels*:

   `verified(sig,m,pk,result)` **`result ∈ {true,false}`**

   `honest(pk)`

$$\text{State 1} \xrightarrow{\texttt{verified(sig,m,pk,true)}} \text{State 2}$$

Note:
We only get guarantees for keys from **gen()**!

2. Any step where an honest party generates a public key, we label it with ‘**honest**’.

3. Now we can use *restrictions* to control when the ‘**verified**’ event can occur.

The Restrictions!

The Restrictions!

Correctness

```
Honest(pk(a)) & Verified(sign(m,r,a),m,pk(a),False) => ⊥
```

The Restrictions!

Correctness

```
Honest(pk(a)) & Verified(sign(m,r,a),m,pk(a),False) => ⊥
```

Unforgeability

```
Honest(pk(a)) & Verified(s,m,pk(a),true) => s = sign(m,r,a)
```

The Restrictions!

Correctness

```
Honest(pk(a)) & Verified(sign(m,r,a),m,pk(a),False) => ⊥
```

Unforgeability

```
Honest(pk(a)) & Verified(s,m,pk(a),true) => s = sign(m,r,a)
```

Consistency

```
Verified(s,m,pk(a),r1) & Verified(s,m,pk(a),r2) => r1 = r2
```

Stepping Back

Initial Idea

Automation research

Reading About DSKs

Hearing about Small subgroups

Learning about Length Extension attacks

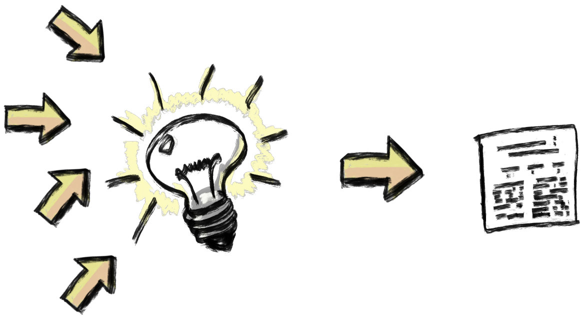
Initial Idea

Automation research

Reading about DSKs

Hearing about small subgroups

Learning about length extension attacks



Better Dolev-Yao

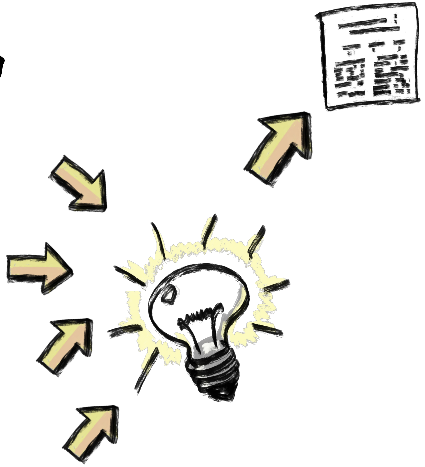
Initial Idea vs. Results

Automation research

Reading about DSKs

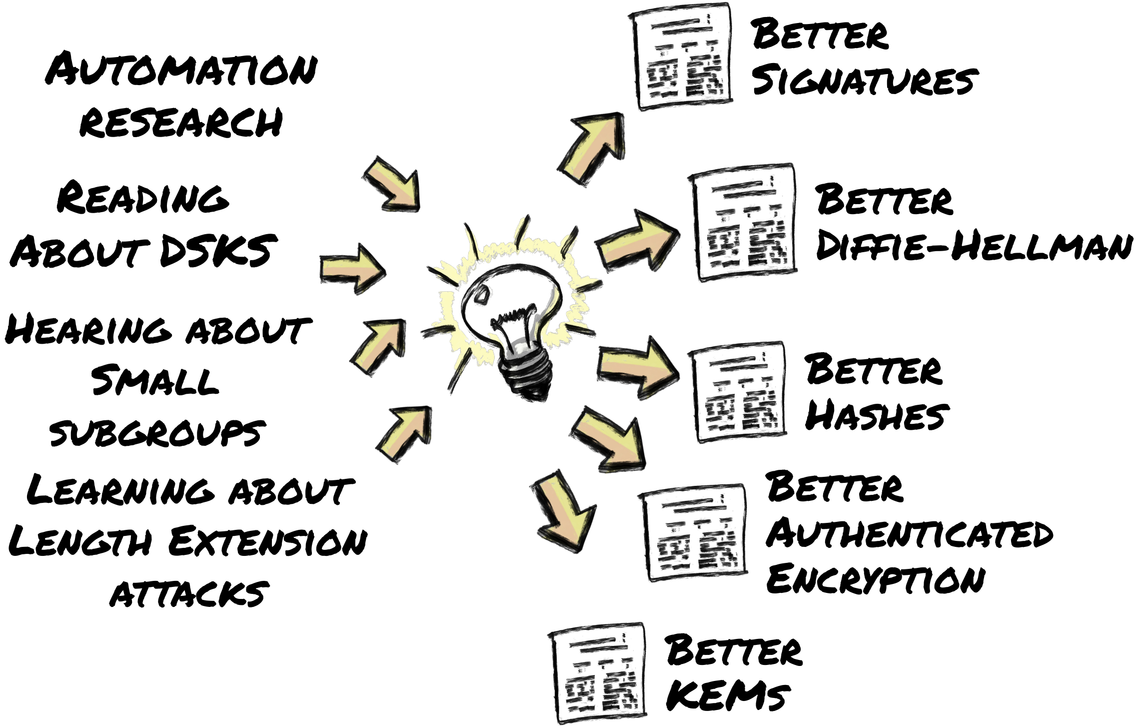
Hearing about small subgroups

Learning about Length Extension attacks

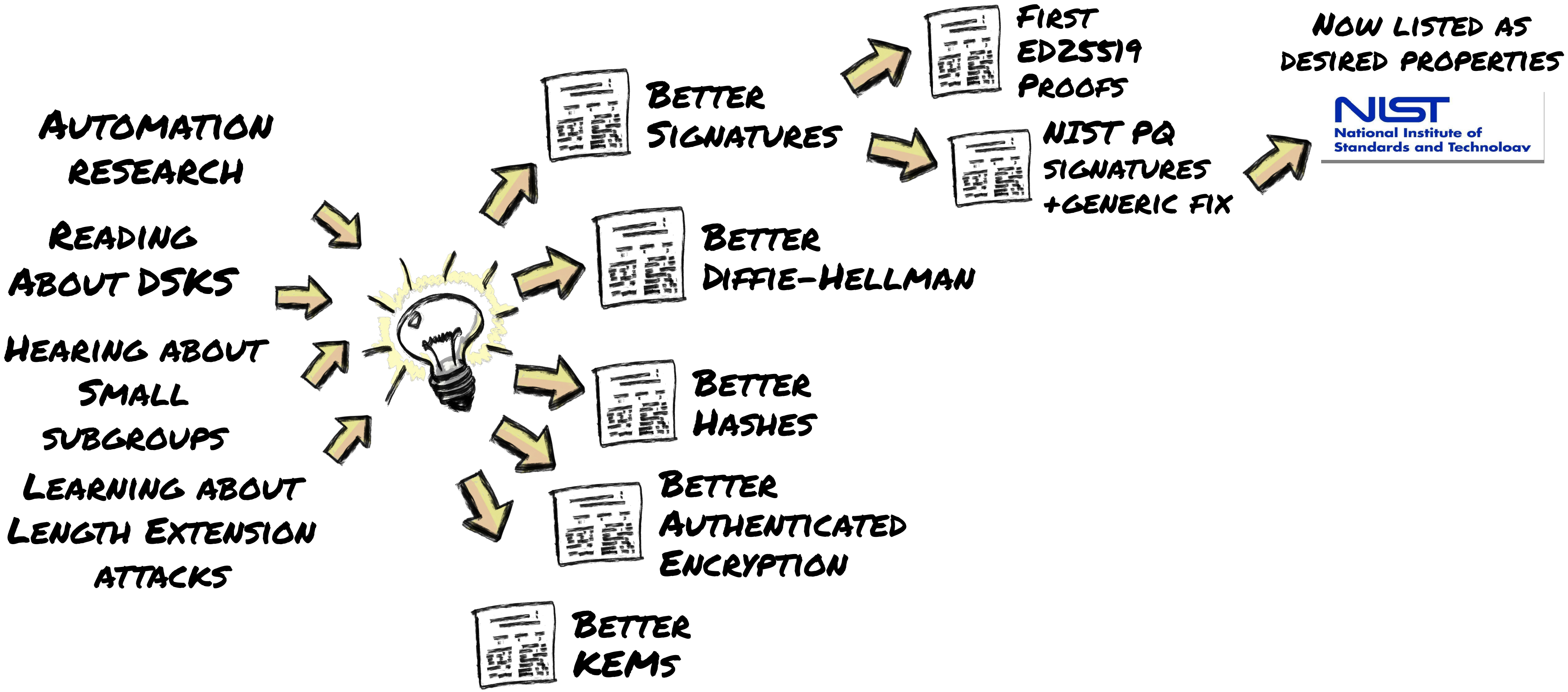


Better Signatures

Initial Idea vs. Results

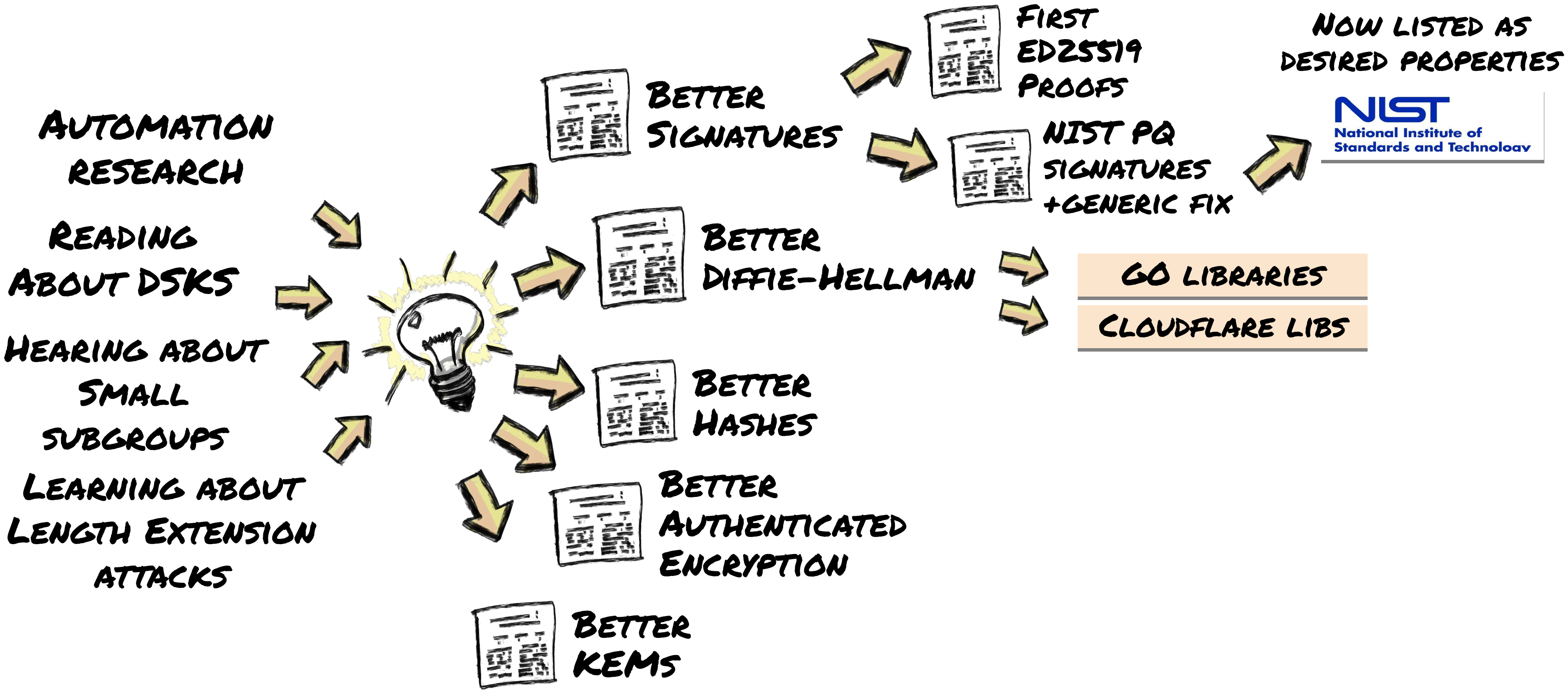


Initial Idea vs. Results and their Impact



CISPA
HELMHOLTZ CENTER FOR INFORMATION SECURITY

Initial Idea vs. Results and their Impact



Initial Idea vs. Results and their Impact

CISPA
HELMHOLTZ CENTER FOR INFORMATION SECURITY

AUTOMATION RESEARCH

READING ABOUT DSKS

HEARING ABOUT SMALL SUBGROUPS

LEARNING ABOUT LENGTH EXTENSION ATTACKS

BETTER SIGNATURES

FIRST ED25519 PROOFS

NIST PQ SIGNATURES +GENERIC FIX

NOW LISTED AS DESIRED PROPERTIES

NIST National Institute of Standards and Technology

BETTER DIFFIE-HELLMAN

GO LIBRARIES

CLOUDFLARE LIBS

BETTER HASHES

BETTER AUTHENTICATED ENCRYPTION

I E T F

DRAFT ON AEAD PROPERTIES

BETTER KEMS

MAY LEAD TO CHANGES IN ML-KEM AND IETF STANDARDS

I E T F

NIST National Institute of Standards and Technology

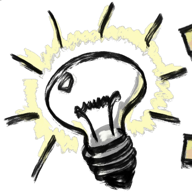
Initial Idea vs. Results and their Impact

Automation research

Reading about DSKs

Hearing about small subgroups

Learning about Length Extension attacks



Better Signatures

First Ed25519 proofs

NIST PQ signatures +generic fix

Now listed as desired properties

National Institute of Standards and Technology

Better Diffie-Hellman

Go libraries

Cloudflare libs

Better Hashes

Better Authenticated Encryption

IETF

Draft on AEAD properties

Better KEMs

May lead to changes in ML-KEM and IETF Standards

IETF

National Institute of Standards and Technology

??

Wider Research Questions

Is there still more work to do?

Wider Research Questions

Is there still more work to do?

Which attacks are covered by computational protocol proofs, but cannot be captured symbolically?