CISPA
HELMHOLTZ CENTER FOR INFORMATION SECURITY

Formal Analysis of Real-World Security Protocols

Lecture 11: Relation to cryptography, scaling, and the future

Cas Cremers | 03.02.2025

This lecture

1. Relation to cryptography
2. Scaling and the future

1. Relation to cryptography

Historical context of symbolic protocol analysis

1970's

- Birth of provable security [Goldwasser, Micali, Yao]
- Show that breaking a cryptographic primitive implies breaking assumption

1980's

- Symbolic model [Dolev and Yao]
- **Higher-level "logical" reasoning** about security

1990's

- Model checking to explore finite cases
- Many tools (Casper/FDR, NRL, Prolog)
- Needham-Schroeder(-Lowe)

Historical context of symbolic protocol analysis

2000's

- Mature finite case exploration (AVISPA), more industrial examples
- Scaling barrier; switch to ProVerif, Scyther
- Symbolic soundness questions

2010's

- Proverif, **Tamarin**
- Increase expressiveness
  - equational theories
  - Loops & stateful protocols
  - Some hyperproperties
- Scaling towards real-world protocols
  - TLS 1.3
  - 5G
  - EMV

Reality, models, and gaps

Reality

Reality, models, and gaps

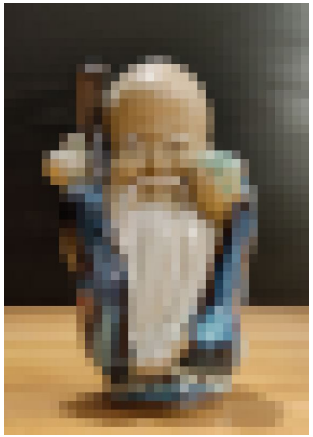
Reality

Computational

Reality, models, and gaps

Reality

Computational

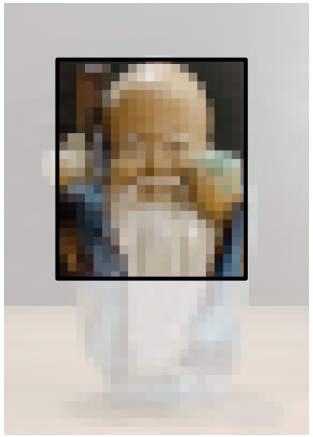


Symbolic

Reality, models, and gaps

Reality

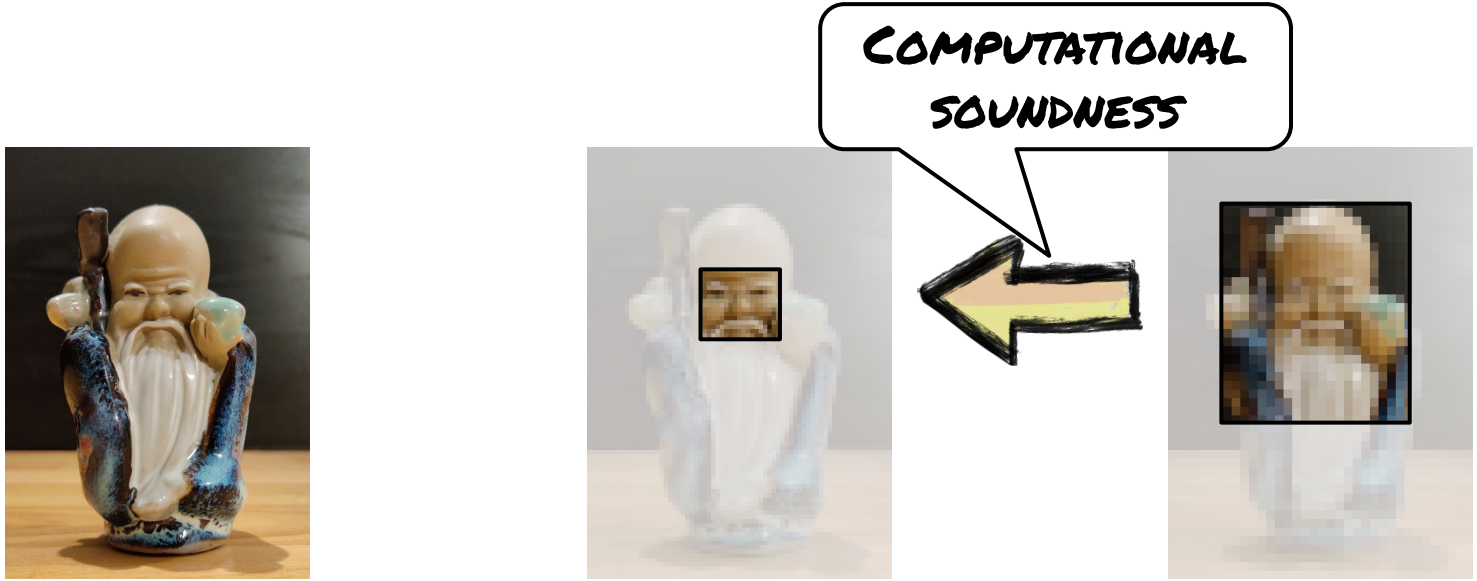
Computational



Symbolic

[BBBBCLP2021] IEEE S&P 2021: *SoK: Computer-Aided Cryptography*

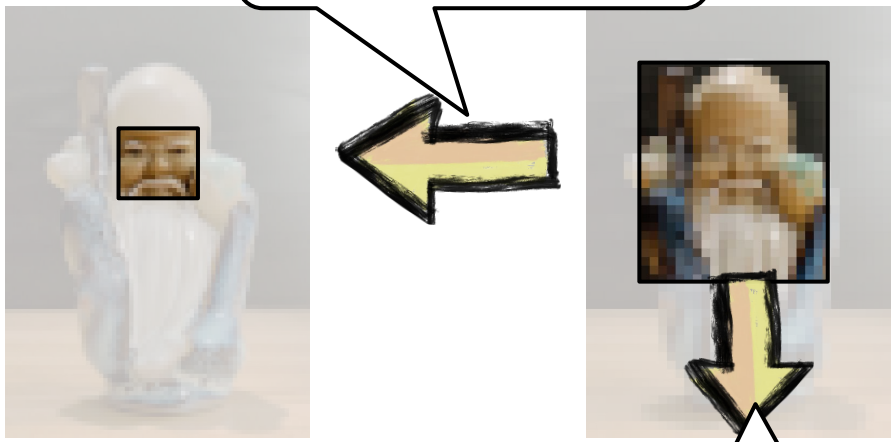
Reality, models, and gaps



Reality | Computational | Symbolic

[BBBBCLP2021] IEEE S&P 2021: *SoK: Computer-Aided Cryptography*

Reality, models, and gaps



Reality

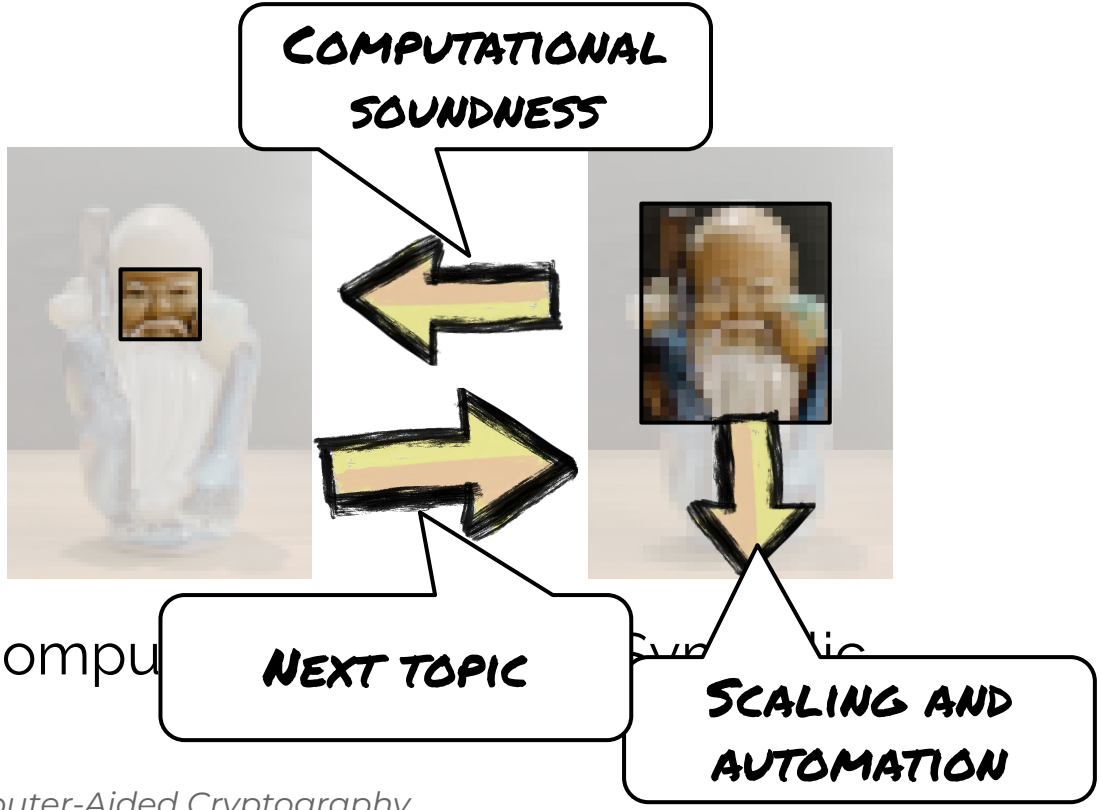
Computational

Sym

[BBBBCLP2021] IEEE S&P 2021: *SoK: Computer-Aided Cryptography*

Reality, models, and gaps

Reality



Compu

[BBBBCLP2021] IEEE S&P 2021: *SoK: Computer-Aided Cryptography*

Two things that stuck in the back of my head

Around 2006: 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)

Two things that stuck in the back of my head

Around 2006: 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)

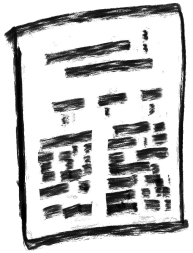
Around 2014: Small subgroups

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?*
Bharghavan et. al. make a basic model in ProVerif for channel bindings work.

2016

Let's write a paper!

"Better Dolev-Yao abstractions of cryptographic primitives"



Plan:

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

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

2017

Let's write a paper!

~~"Better Dolev-Yao abstractions of cryptographic primitives"~~

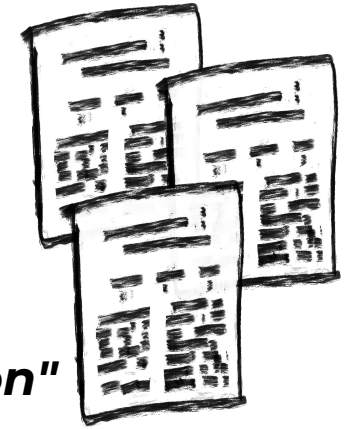
After months of work:

signatures alone are a paper

2017

*Let's write **three** papers!*

"Signatures"
"Diffie-Hellman"
"Authenticated Encryption"



Signatures

Definition: Signature Scheme

Three algorithms:

KeyGen() **-> (sk,pk)**

Sign(sk, msg) **-> sig**

Verify(sig, msg, pk) **-> {true,false}**

Correct if for all messages **m** and any **(sk,pk) <- KeyGen()**

- **Verify(Sign(m,sk),m,pk) = true** (almost always)

Definition: Unforgeability

Existential unforgeability under an adaptive chosen message attack

1. The referee uses **KeyGen** and outputs the public key

Definition: Unforgeability

Existential unforgeability under an adaptive chosen message attack

1. The referee uses **KeyGen** and outputs the public key

2. The adversary may (adaptively) ask the referee for a signature on a message of the adversary's choice.

Definition: Unforgeability

Existential unforgeability under an adaptive chosen message attack

1. The referee uses **KeyGen** and outputs the public key

2. The adversary may (adaptively) ask the referee for a signature on a message of the adversary's choice.

3. The adversary wins if they can produce a message and signature pair that passes **`Verify`**, but the adversary never submitted the message in step 2.

Definition: Unforgeability

Existential unforgeability under an adaptive chosen message attack

1. The referee uses **KeyGen** and outputs the public key

2. The adversary may (adaptively) ask the referee for a signature on a message of the adversary's choice.

3. The adversary wins if they can produce a message and signature pair that passes **`Verify`**, but the adversary never submitted the message in step 2.

Introduced[1] in **1988**, widely accepted as the standard definition.

[1] Goldwasser, S., Micali, S., & Rivest, R. L. (1988)

Definition: Unforgeability

Existential unforgeability under an adaptive chosen message attack

1. The referee uses **KeyGen** and outputs the public key

2. The adversary may (adaptively) ask the referee for a signature on a message of the adversary's choice.

3. The adversary wins if they can produce a message and signature pair that passes **`Verify`**, but the adversary never submitted the message in step 2

Introduced[1] in **1988**, widely accepted

Note:
Definition says nothing about what should hold for maliciously generated keys

[1] Goldwasser, S., Micali, S., & Rivest, R. L. (1988)

History of subtle signature properties

1999: Key Substitution [Blake-Wilson, Menezes]

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

History of subtle signature properties

1999: Key Substitution [Blake-Wilson, Menezes]

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

2000: Message-key Substitution [Baek, Kim]

Given **sig**,**pk**,**msg**, and **msg'**:

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

History of subtle signature properties

1999: Key Substitution [Blake-Wilson, Menezes]

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

2000: Message-key Substitution [Baek, Kim]

Given **sig**,**pk**,**msg**, and **msg'**:

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

Interesting observations, but no great examples on how to exploit this.
Largely ignored at the time.

2002: Colliding signatures [Stern, Pointcheval, Malone-Lee, Smart]

Prevalence

■ Proven Absent
▲ Unknown
● Present

Signature scheme	KS	MKS	Coll.
RSA-PKCSv1.5	● [64]	● [64]	▲
RSA-PSS	● [64]	● [64]	▲
DSA	● [64]	● [64]	● [69]
ECDSA-FreeBP	● [26]	● [26]	● [67]
ECDSA-FixedBP	■ [59]	■ [59]	● [67]
Ed25519	■ [47]	■ [47]	● [19]
Ed25519-IETF	■ [47]	■ [47]	● [19]

Simplified table from [JCCS2019] ACM CCS 2019: *Seems Legit: Automated Analysis of Subtle Attacks on Protocols that Use Signatures*

[64] Pornin, T., & Stern, J. P. (2005). [26] Blake-Wilson, S., & Menezes, A. (1999). [59] Menezes, A., & Smart, N. (2001).
[47] Günther, F., & Poettering, B. (2017). [69] Vaudenay, S. (2003). [67] Stern, Jacques, et al. (2002) [19] Bernstein, Daniel J., et al (2012).

Symbolic encoding of signatures

Before 2019:

- Only one signature definition in Tamarin, with very strong properties

Equational theory:

verify(sign(m,sk), m, pk(sk)) = true

[JCCS2019] ACM CCS 2019: *Seems Legit: Automated Analysis of Subtle Attacks on Protocols that Use Signatures*

New symbolic encoding of signatures

In 2019 we introduced several new symbolic signature models

(1) A **proof-oriented model** with minimal assumptions

- Consistency: Verification must be consistent
- NoForgery: Verification for honestly generated public keys cannot be forged
- Correctness: Honestly generated signatures verify correctly

(2) Several **attack-finding models**

- E.g., with the ability to compute a key pair for Key Substitution

Designed for Tamarin, but mostly reusable in ProVerif

[JCCS2019] ACM CCS 2019: *Seems Legit: Automated Analysis of Subtle Attacks on Protocols that Use Signatures*

Example of Tamarin encodings

No-CEO (Conservative Exclusive Ownership):

```
functions : CEOgen/1
equations : verify( sign (m, sk), m, pk( CEOgen( sign(m, sk )))) = true
```

Example of Tamarin encodings

No-CEO (Conservative Exclusive Ownership):

```
functions : CEOgen/1
equations : verify( sign (m, sk), m, pk( CEOgen( sign(m, sk )))) = true
```

Colliding signatures:

```
functions : weak/1
equations : verify( sign(m1, weak(x)), m2, pk( weak(x))) = true
```

Case studies

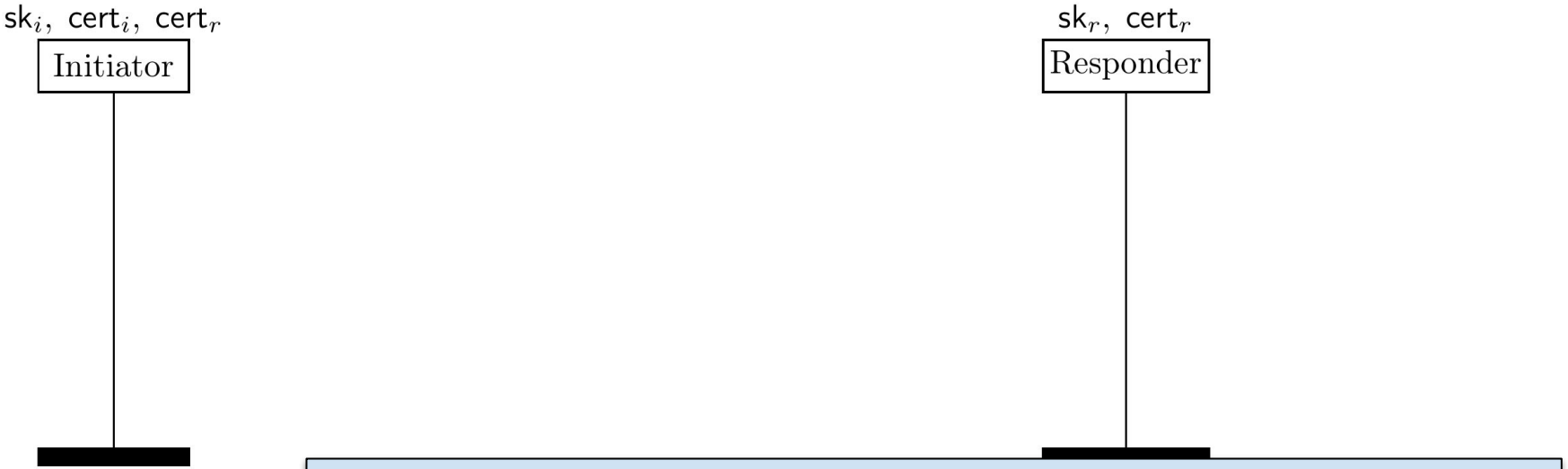
Protocol	Previous verification	
	Year	Methodology
X.509 Mutual Auth	2006	ProVerif
WS Request-Response	2008	F# → ProVerif
STS-MAC-fix1	2012	Tamarin
STS-MAC-fix2	2012	Tamarin
DRKey & OPT	2014	Coq
ACME Draft 4	2017	ProVerif

Case studies

Protocol	Previous verification		New Tamarin analysis [JCCS2019]		
	Year	Methodology	Property	Time (s)	Attack
X.509 Mutual Auth	2006	ProVerif	Correlation & Secrecy	5	**NEW ATTACK**
WS Request-Response	2008	F# → ProVerif			
STS-MAC-fix1	2012	Tamarin	Authentication	35	Rediscovered manual attack
STS-MAC-fix2	2012	Tamarin	Authentication	68	Rediscovered manual attack
DRKey & OPT	2014	Coq	Authentication	2640	**NEW ATTACK**
ACME Draft 4	2017	ProVerif	DNS Validation	53	Rediscovered manual attack

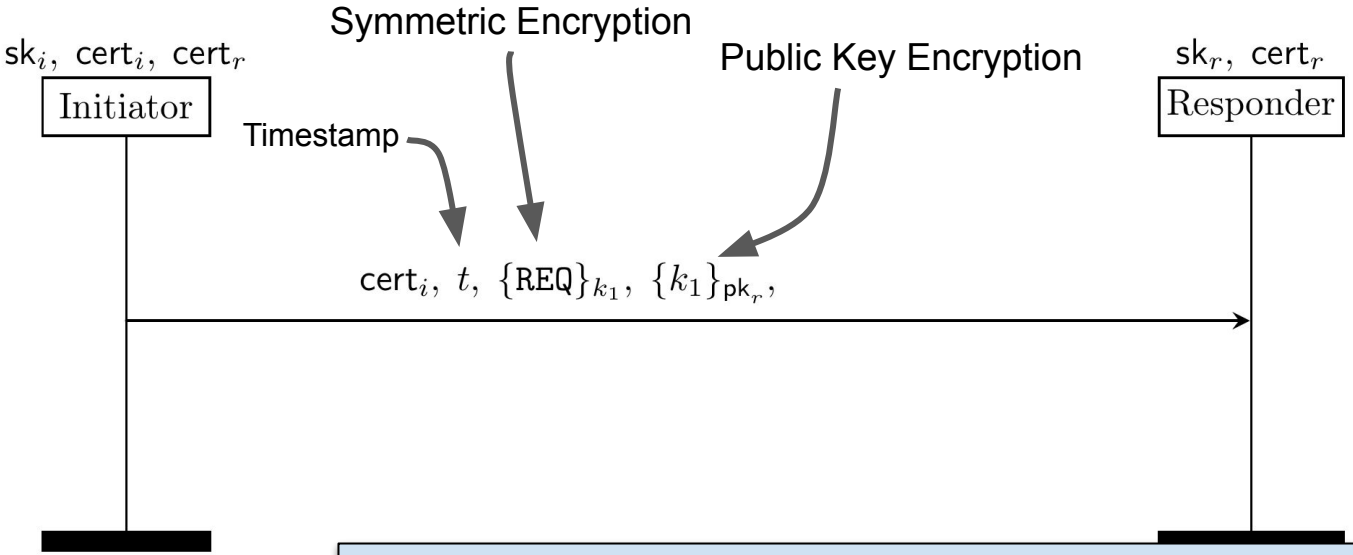
[JCCS2019] ACM CCS 2019: *Seems Legit: Automated Analysis of Subtle Attacks on Protocols that Use Signatures*

WS Security X.509 Mutual Authentication



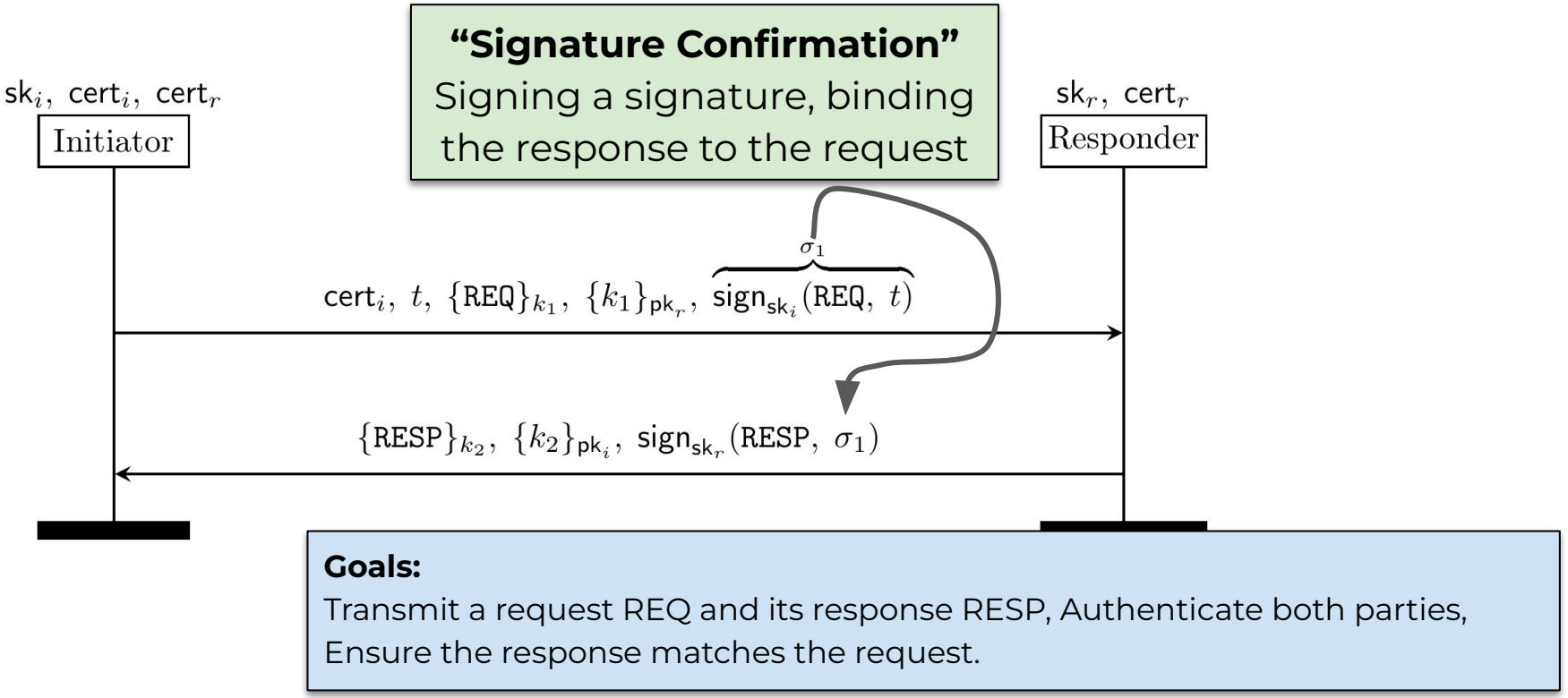
Goals:
Transmit a request REQ and its response RESP, Authenticate both parties, Ensure the response matches the request.

WS Security X.509 Mutual Authentication

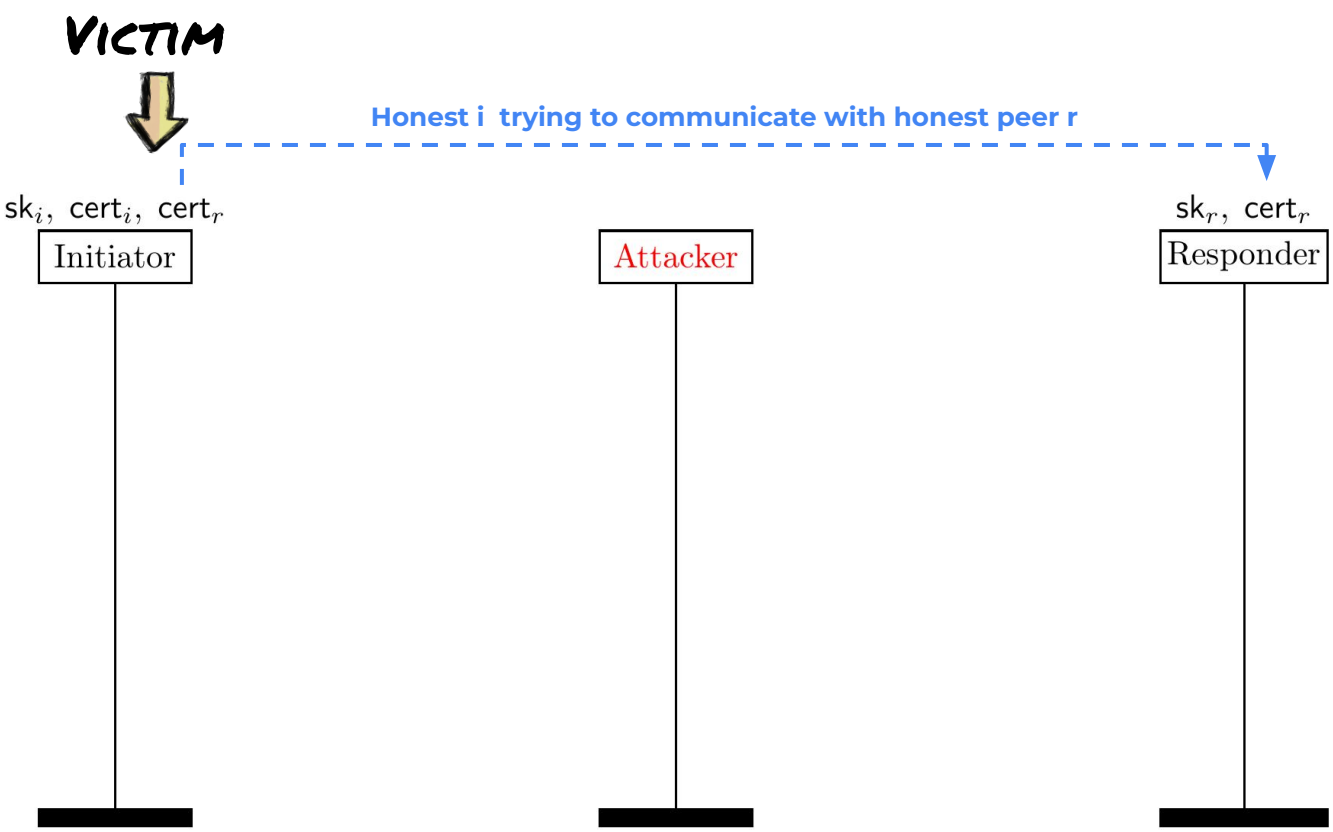


Goals:
Transmit a request REQ and its response RESP, Authenticate both parties, Ensure the response matches the request.

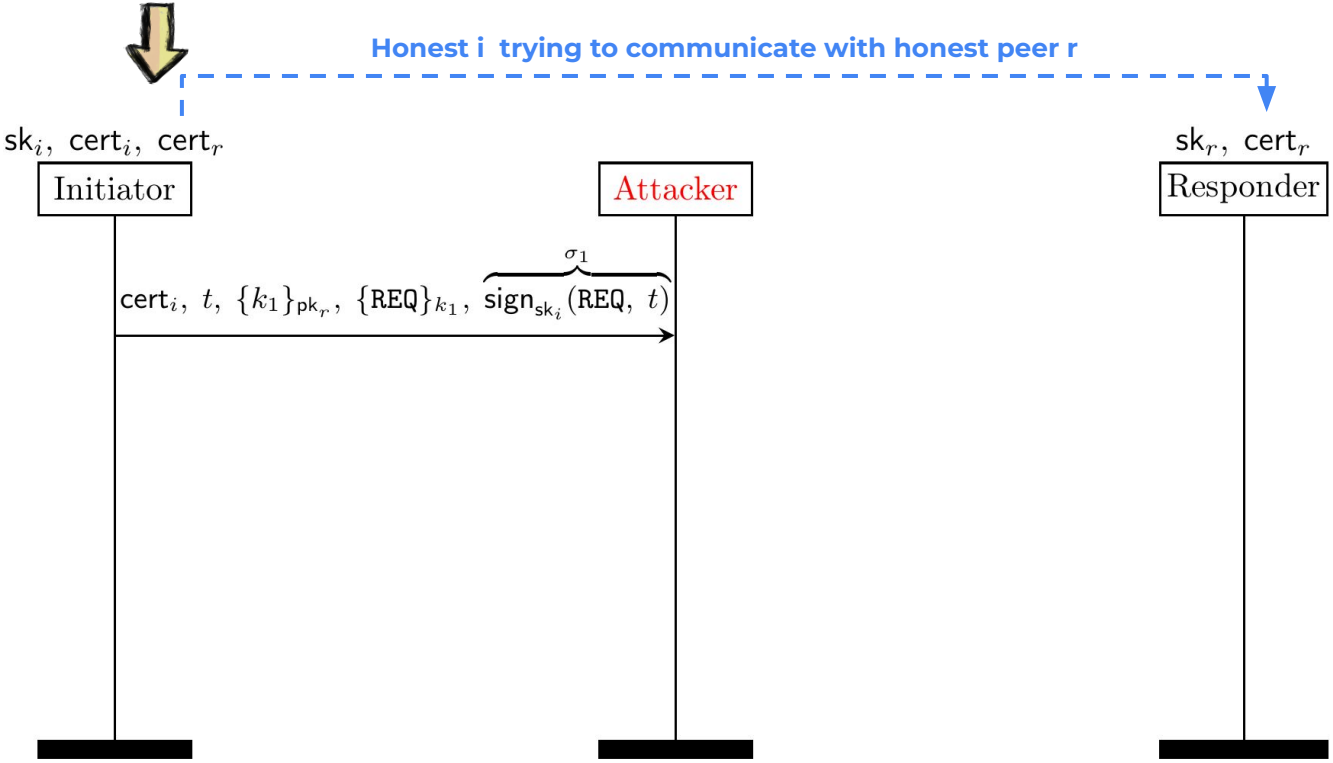
WS Security X.509 Mutual Authentication



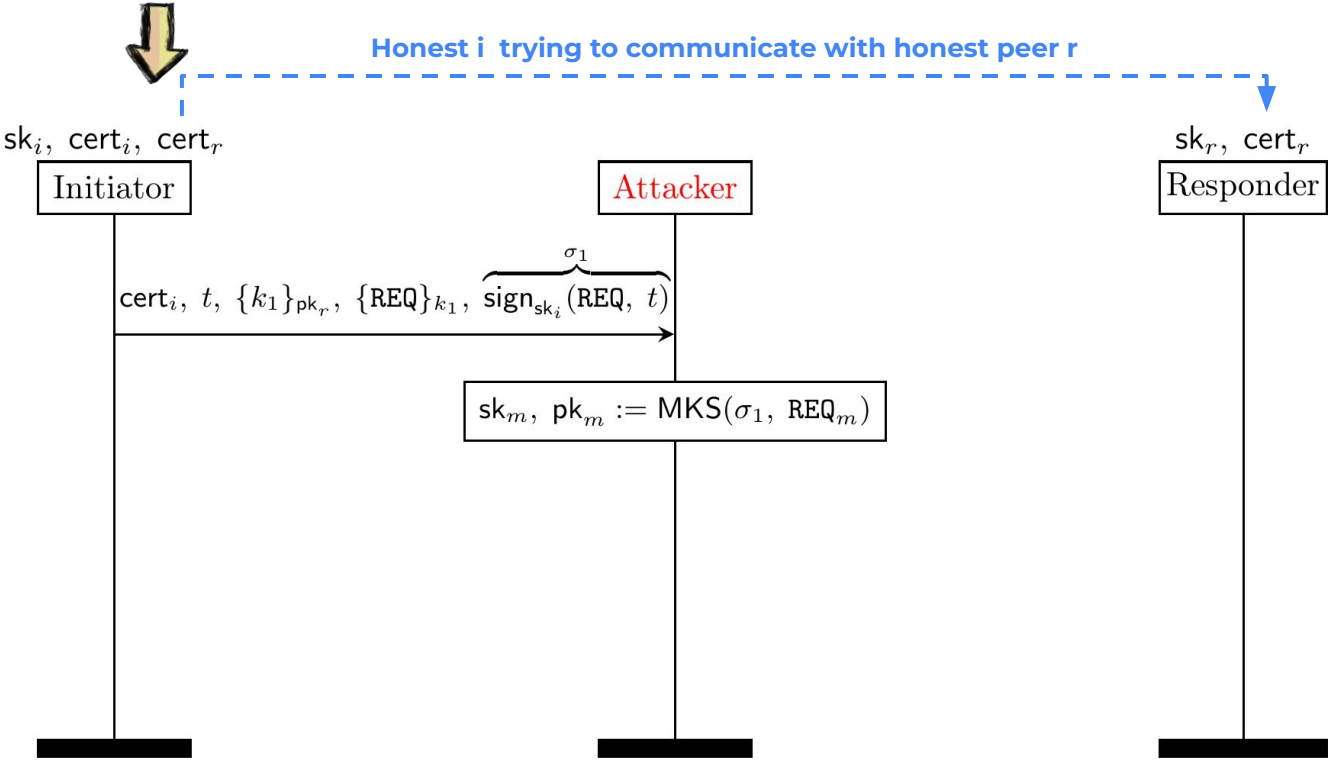
Goals:
Transmit a request REQ and its response RESP, Authenticate both parties, Ensure the response matches the request.


VICTIM
Honest i trying to communicate with honest peer r
$sk_i,\ cert_i,\ cert_r$
Initiator
Attacker
$sk_r,\ cert_r$
Responder

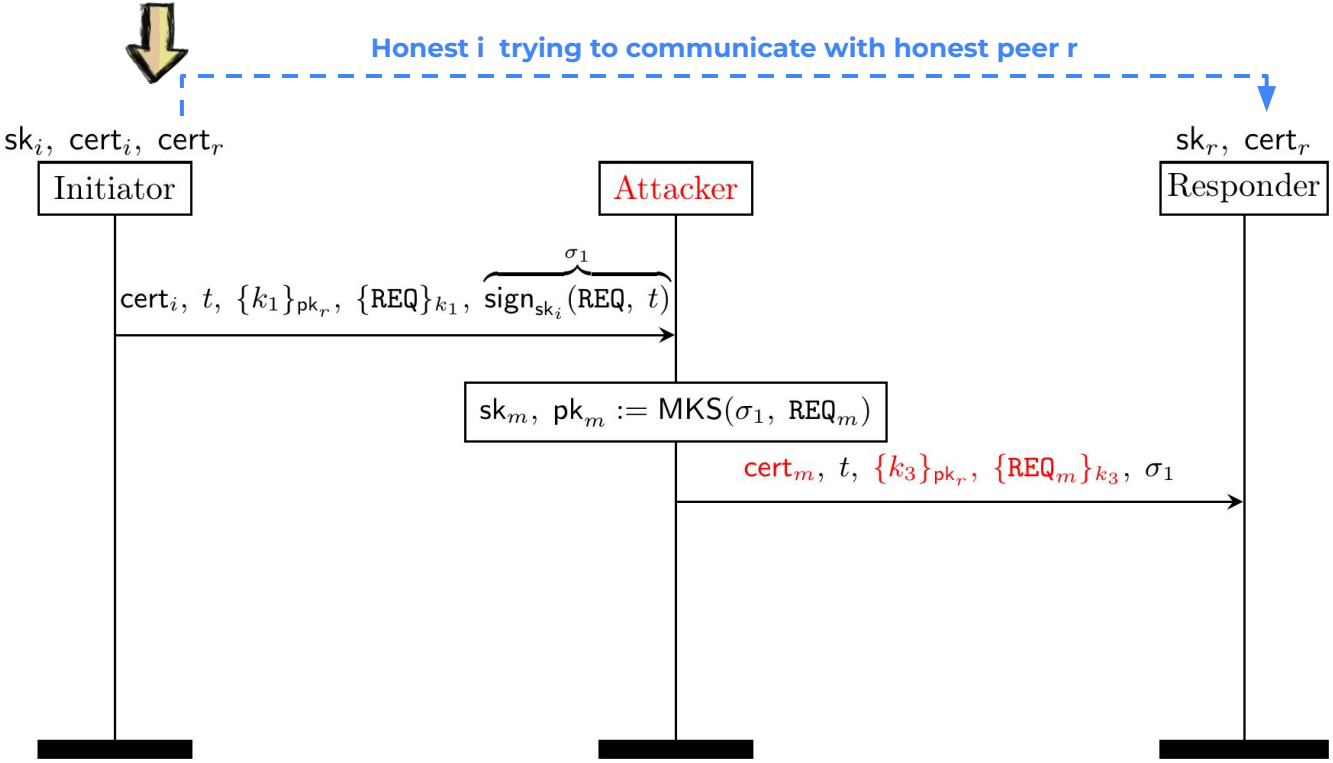
VICTIM


Honest i trying to communicate with honest peer r
sk$_i$, cert$_i$, cert$_r$
sk$_r$, cert$_r$
Initiator
Attacker
Responder
σ_1
cert$_i$, t, $\{k_1\}_{\mathsf{pk}_r}$, $\{\mathtt{REQ}\}_{k_1}$, $\mathsf{sign}_{\mathsf{sk}_i}(\mathtt{REQ}, t)$

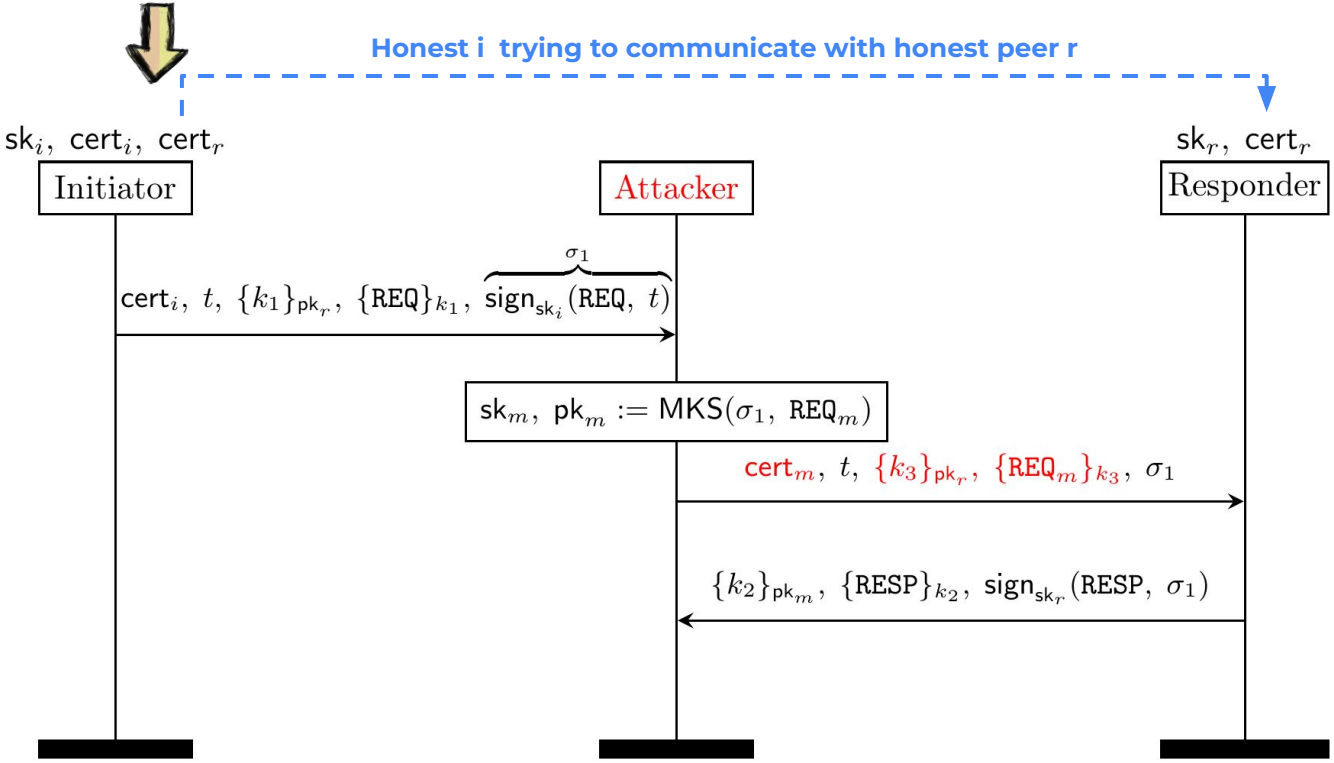
VICTIM


Honest i trying to communicate with honest peer r
sk$_i$, cert$_i$, cert$_r$
sk$_r$, cert$_r$
Initiator
Attacker
Responder
σ_1
cert$_i$, t, $\{k_1\}_{\mathsf{pk}_r}$, $\{\mathtt{REQ}\}_{k_1}$, $\mathsf{sign}_{\mathsf{sk}_i}(\mathtt{REQ}, t)$
$\mathsf{sk}_m, \mathsf{pk}_m := \mathsf{MKS}(\sigma_1, \mathtt{REQ}_m)$

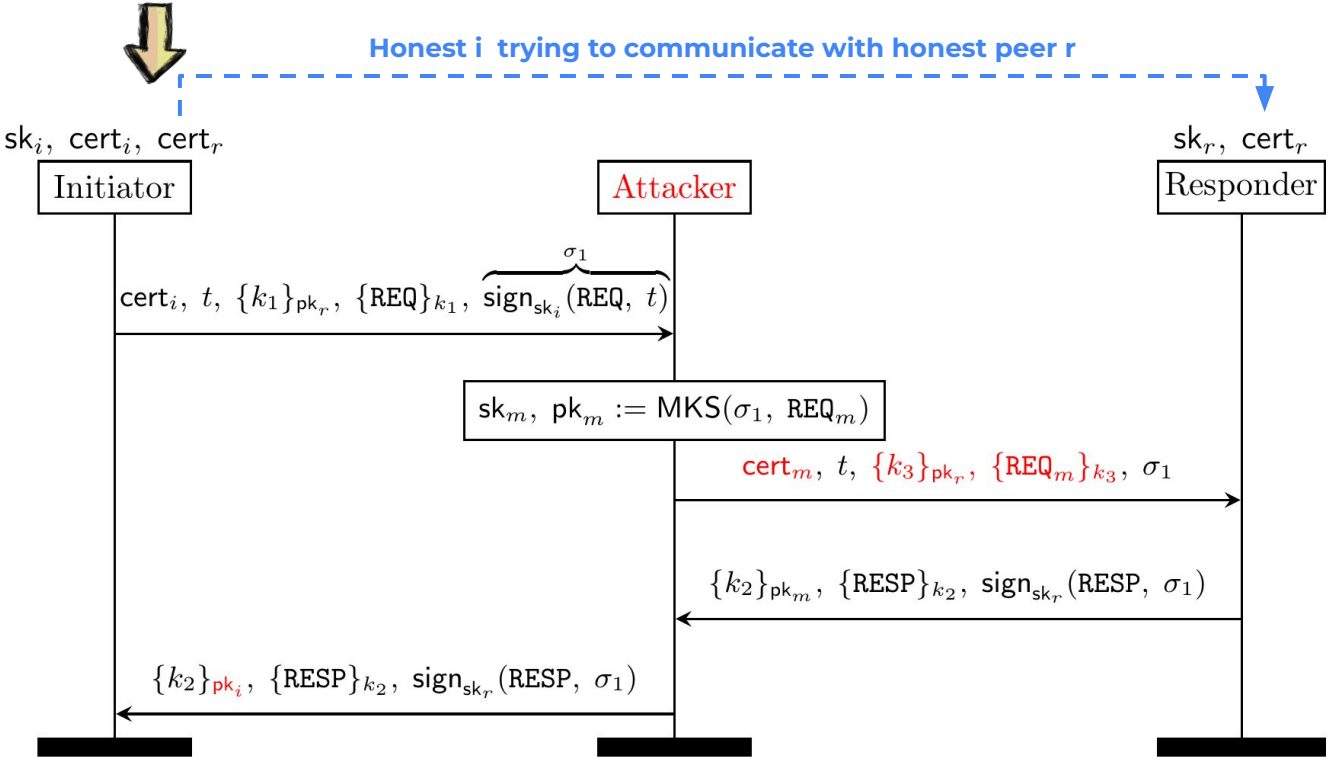
VICTIM

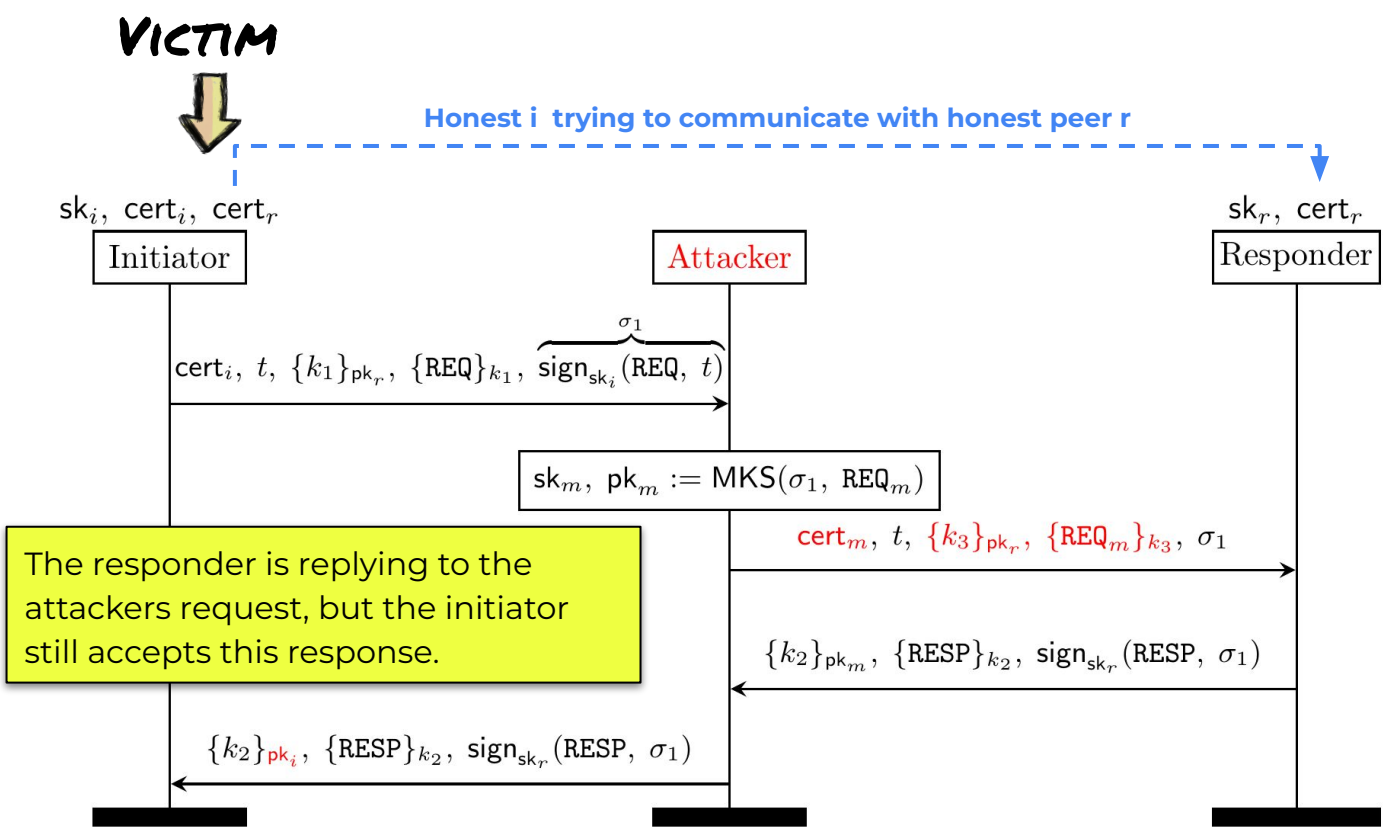

Honest i trying to communicate with honest peer r
sk$_i$, cert$_i$, cert$_r$
sk$_r$, cert$_r$
Initiator
Attacker
Responder
σ_1
cert$_i$, t, $\{k_1\}_{\mathsf{pk}_r}$, $\{\mathtt{REQ}\}_{k_1}$, $\mathsf{sign}_{\mathsf{sk}_i}(\mathtt{REQ}, t)$
sk$_m$, pk$_m$:= MKS(σ_1, REQ$_m$)
cert$_m$, t, $\{k_3\}_{\mathsf{pk}_r}$, $\{\mathtt{REQ}_m\}_{k_3}$, σ_1

VICTIM

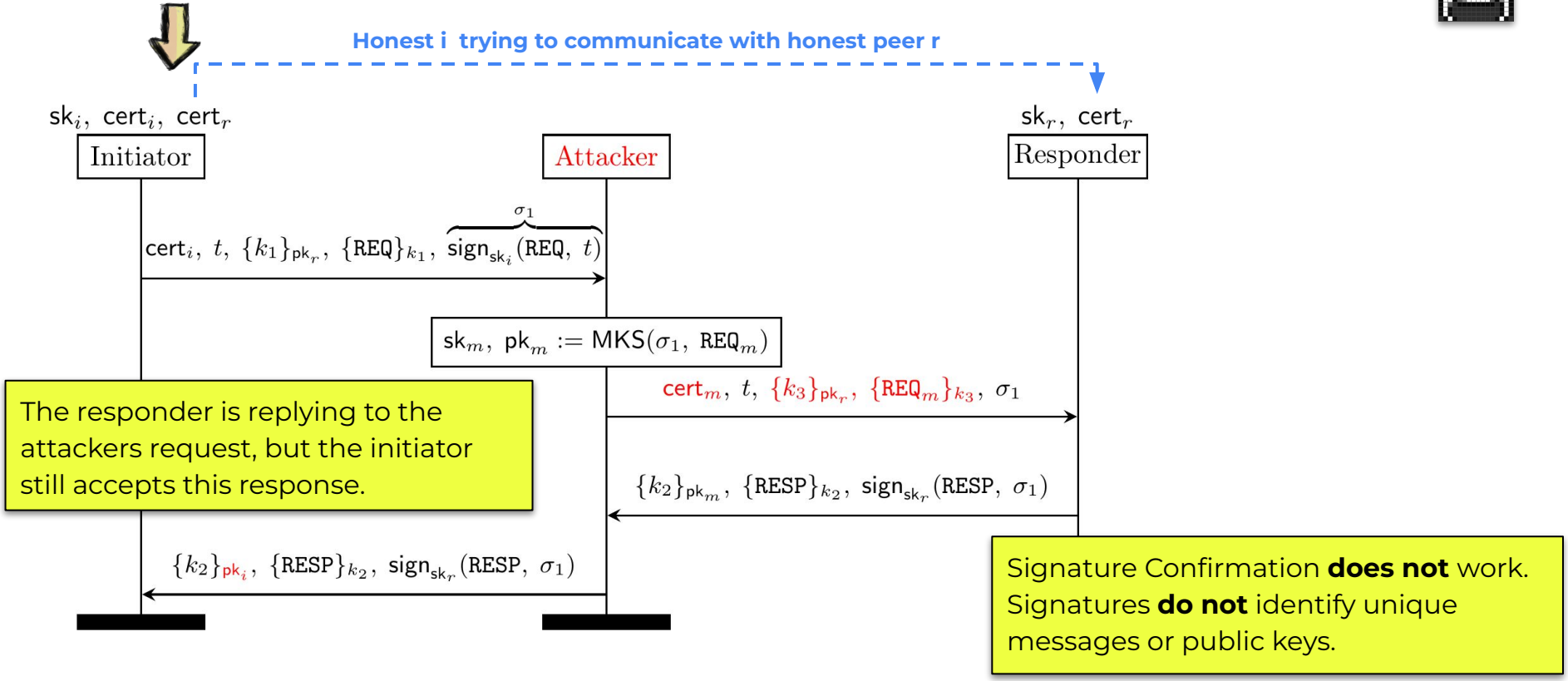

Honest i trying to communicate with honest peer r
sk$_i$, cert$_i$, cert$_r$
sk$_r$, cert$_r$
Initiator
Attacker
Responder
σ_1
cert$_i$, t, $\{k_1\}_{\mathsf{pk}_r}$, $\{\mathtt{REQ}\}_{k_1}$, $\mathsf{sign}_{\mathsf{sk}_i}(\mathtt{REQ}, t)$
sk$_m$, pk$_m$:= MKS(σ_1, REQ$_m$)
cert$_m$, t, $\{k_3\}_{\mathsf{pk}_r}$, $\{\mathtt{REQ}_m\}_{k_3}$, σ_1
$\{k_2\}_{\mathsf{pk}_m}$, $\{\mathtt{RESP}\}_{k_2}$, $\mathsf{sign}_{\mathsf{sk}_r}(\mathtt{RESP}, \sigma_1)$

VICTIM


Honest i trying to communicate with honest peer r
sk$_i$, cert$_i$, cert$_r$
sk$_r$, cert$_r$
Initiator
Attacker
Responder
σ_1
cert$_i$, t, $\{k_1\}_{\mathsf{pk}_r}$, $\{\mathtt{REQ}\}_{k_1}$, sign$_{\mathsf{sk}_i}$(REQ, t)
sk$_m$, pk$_m$:= MKS(σ_1, REQ$_m$)
cert$_m$, t, $\{k_3\}_{\mathsf{pk}_r}$, $\{\mathtt{REQ}_m\}_{k_3}$, σ_1
$\{k_2\}_{\mathsf{pk}_m}$, $\{\mathtt{RESP}\}_{k_2}$, sign$_{\mathsf{sk}_r}$(RESP, σ_1)
$\{k_2\}_{\mathsf{pk}_i}$, $\{\mathtt{RESP}\}_{k_2}$, sign$_{\mathsf{sk}_r}$(RESP, σ_1)

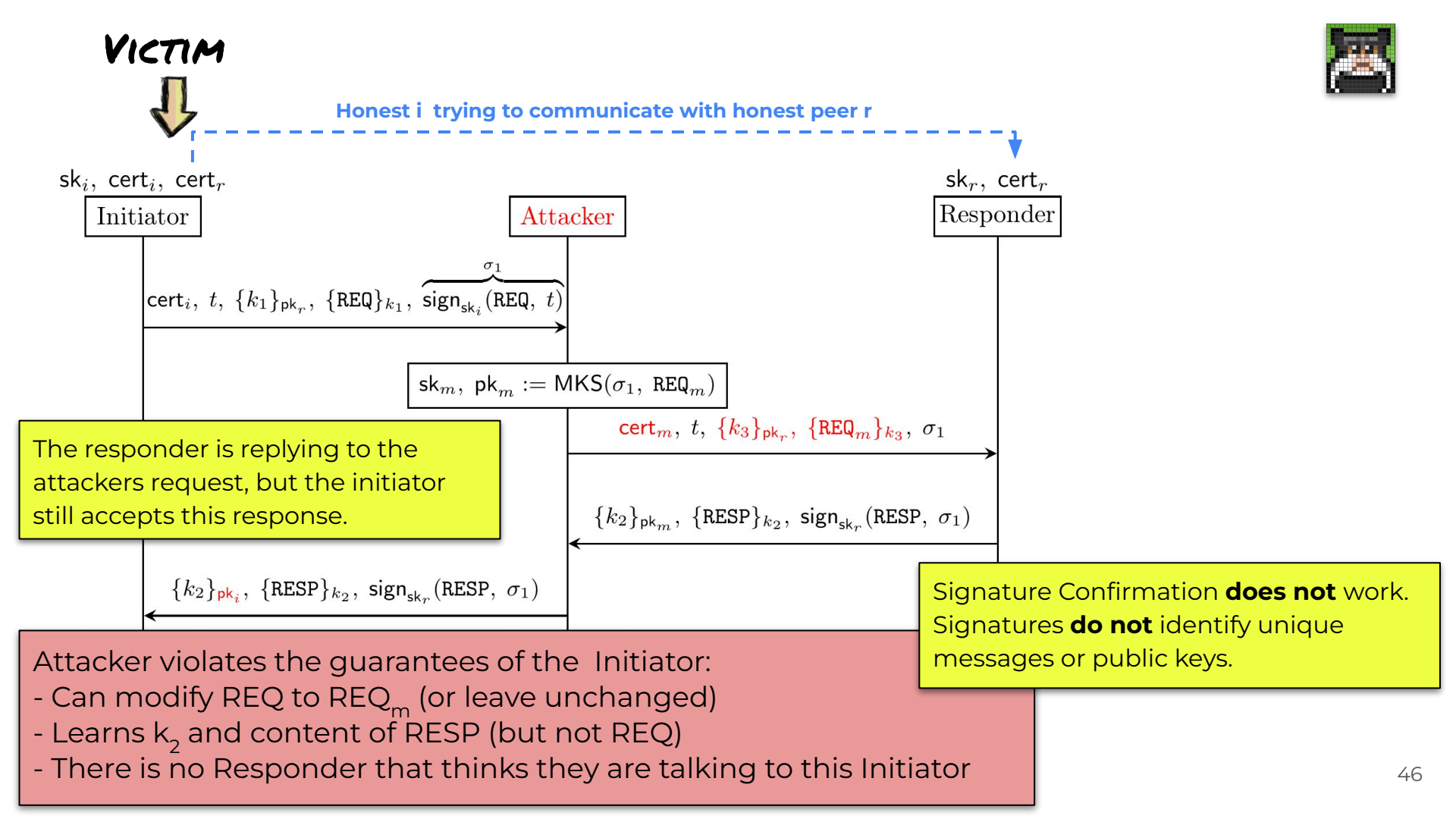

VICTIM
Honest i trying to communicate with honest peer r
$\mathsf{sk}_i,\ \mathsf{cert}_i,\ \mathsf{cert}_r$
$\mathsf{sk}_r,\ \mathsf{cert}_r$
Initiator
Attacker
Responder
σ_1
$\mathsf{cert}_i,\ t,\ \{k_1\}_{\mathsf{pk}_r},\ \{\mathtt{REQ}\}_{k_1},\ \mathsf{sign}_{\mathsf{sk}_i}(\mathtt{REQ},\ t)$
$\mathsf{sk}_m,\ \mathsf{pk}_m := \mathsf{MKS}(\sigma_1,\ \mathtt{REQ}_m)$
$\mathsf{cert}_m,\ t,\ \{k_3\}_{\mathsf{pk}_r},\ \{\mathtt{REQ}_m\}_{k_3},\ \sigma_1$
$\{k_2\}_{\mathsf{pk}_m},\ \{\mathtt{RESP}\}_{k_2},\ \mathsf{sign}_{\mathsf{sk}_r}(\mathtt{RESP},\ \sigma_1)$
$\{k_2\}_{\mathsf{pk}_i},\ \{\mathtt{RESP}\}_{k_2},\ \mathsf{sign}_{\mathsf{sk}_r}(\mathtt{RESP},\ \sigma_1)$
The responder is replying to the attackers request, but the initiator still accepts this response.

VICTIM



The responder is replying to the attackers request, but the initiator still accepts this response.

Signature Confirmation **does not** work. Signatures **do not** identify unique messages or public keys.


VICTIM
Honest i trying to communicate with honest peer r
$\mathsf{sk}_i,\ \mathsf{cert}_i,\ \mathsf{cert}_r$
$\mathsf{sk}_r,\ \mathsf{cert}_r$
Initiator
Attacker
Responder
σ_1
$\mathsf{cert}_i,\ t,\ \{k_1\}_{\mathsf{pk}_r},\ \{\mathtt{REQ}\}_{k_1},\ \mathsf{sign}_{\mathsf{sk}_i}(\mathtt{REQ},\ t)$
$\mathsf{sk}_m,\ \mathsf{pk}_m := \mathsf{MKS}(\sigma_1,\ \mathtt{REQ}_m)$
$\mathsf{cert}_m,\ t,\ \{k_3\}_{\mathsf{pk}_r},\ \{\mathtt{REQ}_m\}_{k_3},\ \sigma_1$
$\{k_2\}_{\mathsf{pk}_m},\ \{\mathtt{RESP}\}_{k_2},\ \mathsf{sign}_{\mathsf{sk}_r}(\mathtt{RESP},\ \sigma_1)$
$\{k_2\}_{\mathsf{pk}_i},\ \{\mathtt{RESP}\}_{k_2},\ \mathsf{sign}_{\mathsf{sk}_r}(\mathtt{RESP},\ \sigma_1)$
The responder is replying to the attackers request, but the initiator still accepts this response.
Signature Confirmation **does not** work. Signatures **do not** identify unique messages or public keys.
Attacker violates the guarantees of the Initiator:
- Can modify REQ to REQ_m (or leave unchanged)
- Learns k_2 and content of RESP (but not REQ)
- There is no Responder that thinks they are talking to this Initiator

Other primitives example: Diffie-Hellman

Diffie-Hellman

Investigation:

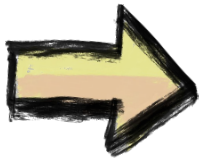
- Prime order groups / curves are encoded in various complex ways
- Lead to subtly different classes of behaviours
  - Prime order groups (= traditional DY model)
  - "Nearly-prime" order groups (small cogroup)
  - Composite groups
  - Single coordinate ladders (for EC)
  - General invalid curve points (for EC)

Diffie-Hellman

Investigation:

- Prime order groups / curves are encoded in various complex ways
- Lead to subtly different classes of behaviours
  - Prime order groups (= traditional DY model)
  - "Nearly-prime" order groups (small cogroup)
  - Composite groups
  - Single coordinate ladders (for EC)
  - General invalid curve points (for EC)

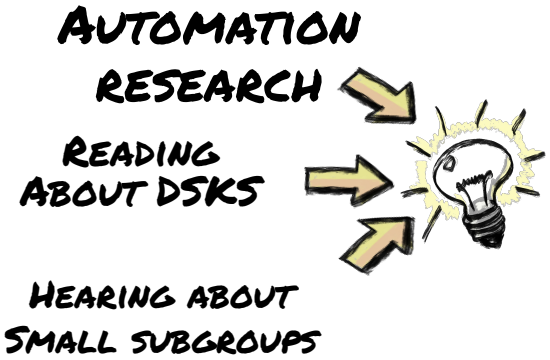
We give symbolic models for each, and for the implemented "checks"



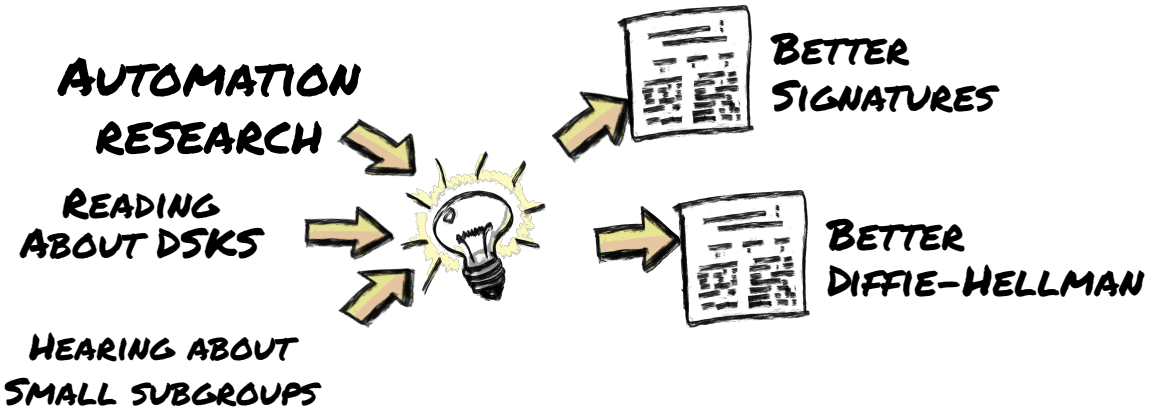
Tamarin finds new attacks automatically
Go's standard crypto library will get new API
Better checks in Cloudflare's standard libraries

Stepping back

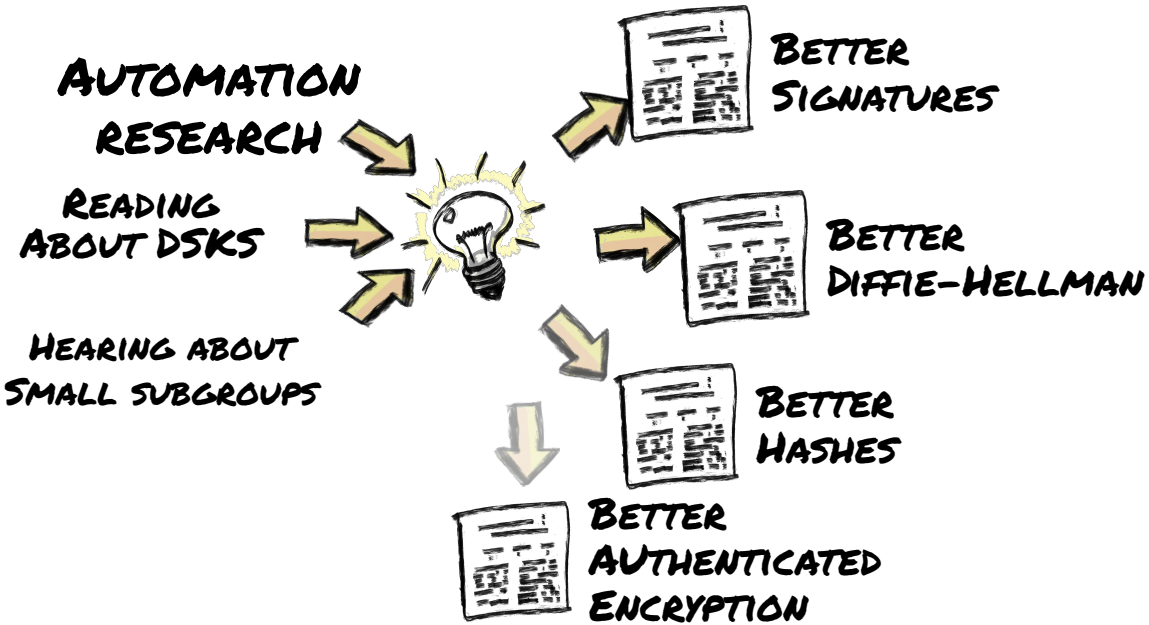
Sometimes ideas escalate!



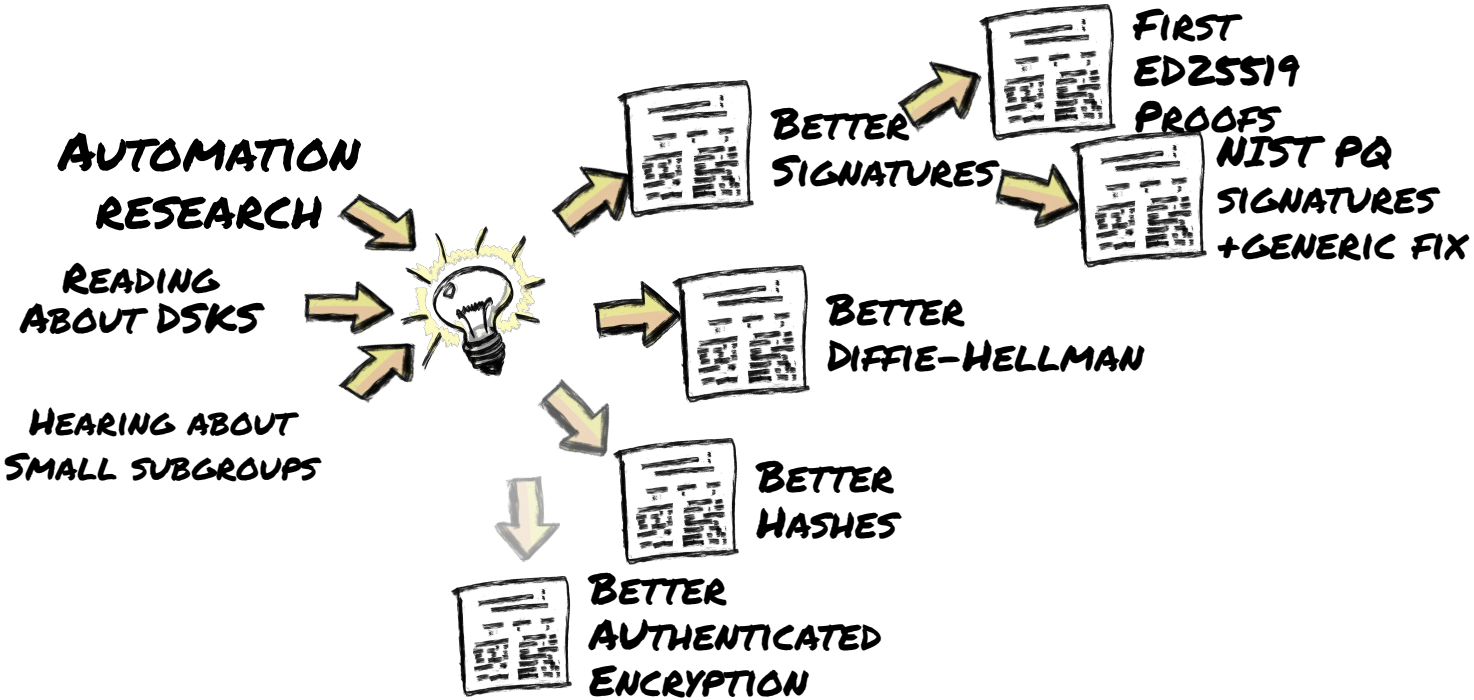
Sometimes ideas escalate!



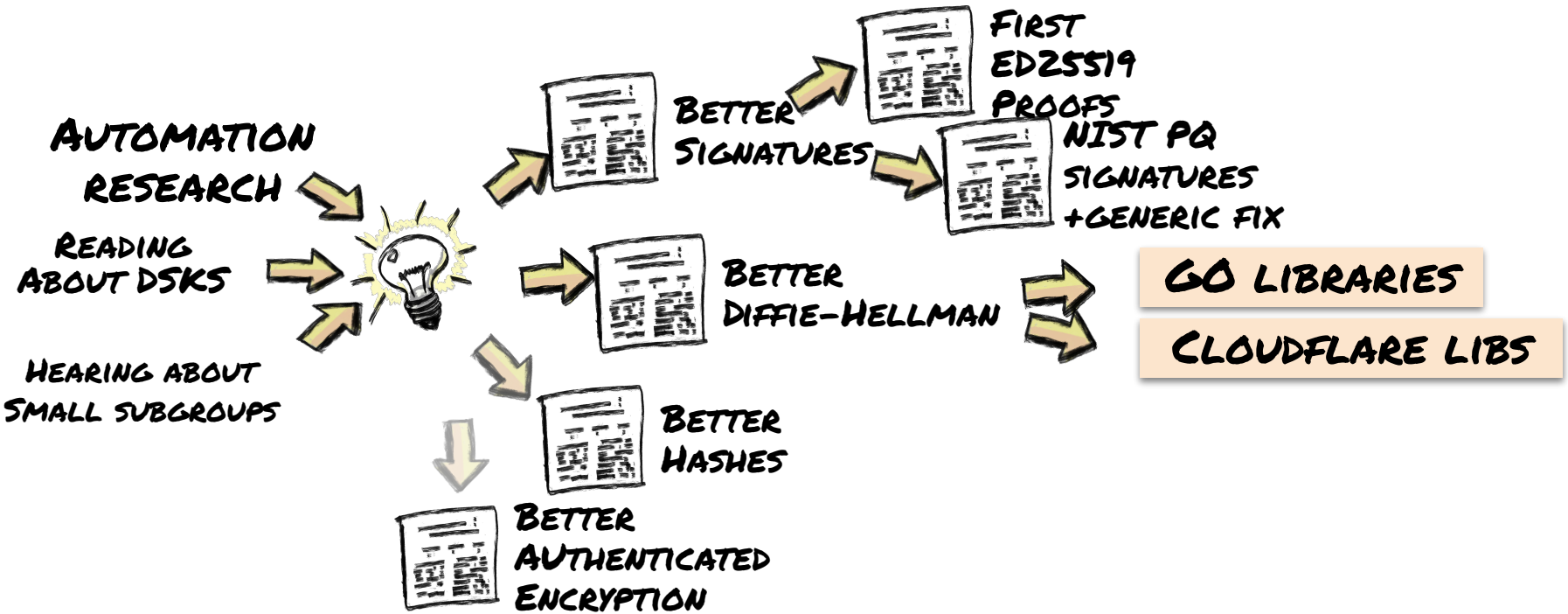
Sometimes ideas escalate!



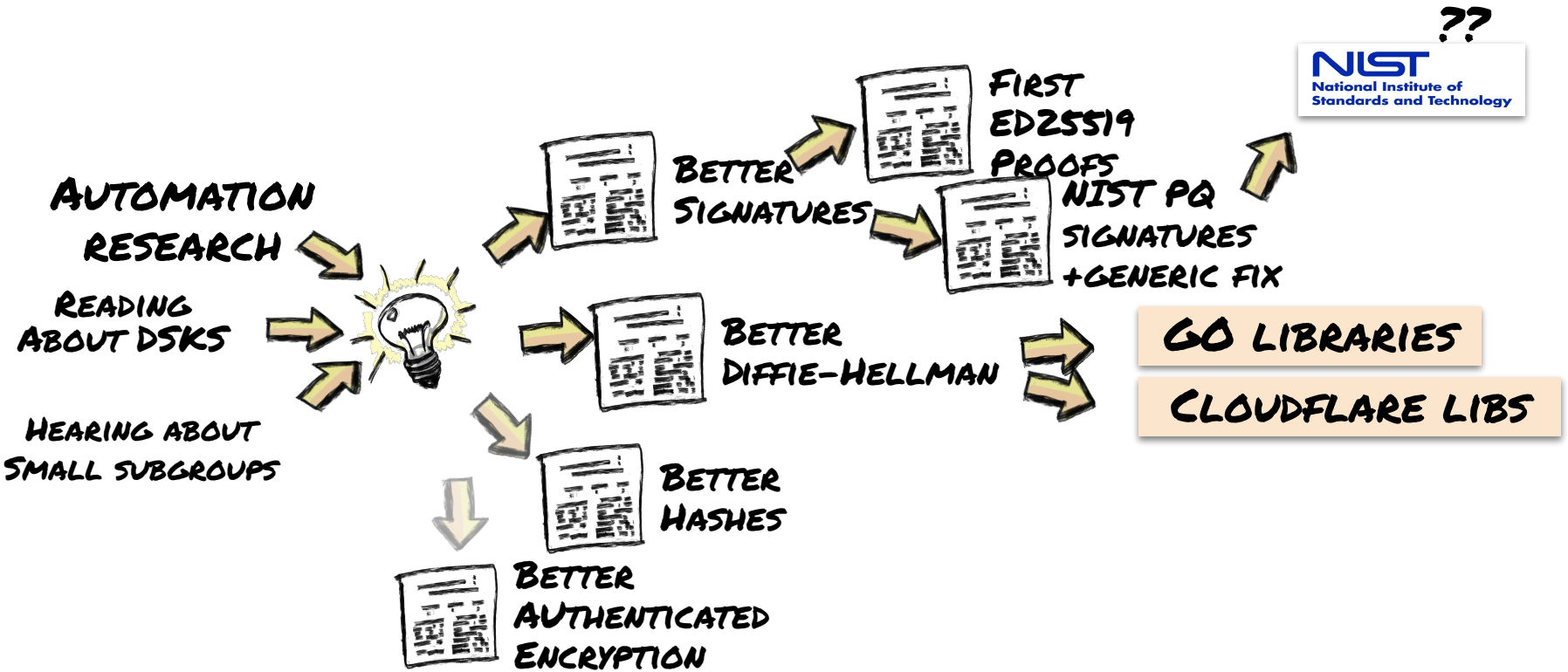
Sometimes ideas escalate!



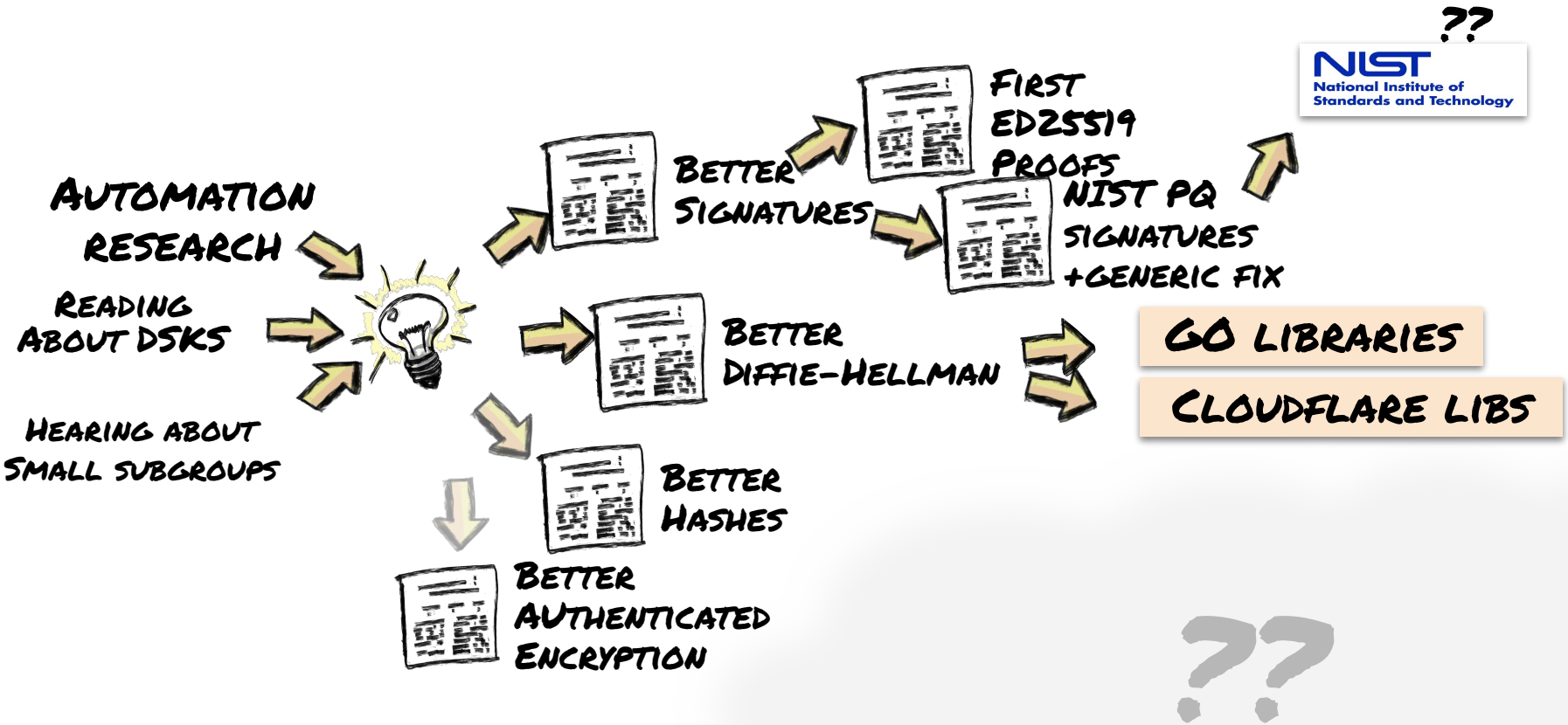
Sometimes ideas escalate!



Sometimes ideas escalate!



Sometimes ideas escalate!

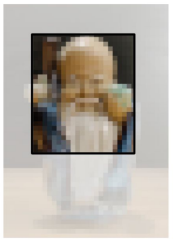


Summary: relation to cryptography

- Exact relation to computational proof unclear: in practice orthogonal
- Dolev-Yao crypto abstractions are 20+ years old
- Systematically reinvent all from scratch
  → analysis more meaningful and effective

Reality

Computational



Symbolic

- Not "just relevant for symbolic analysis"
  - Challenges computational proof methodology → cryptographers
  - Impacts implementations → security engineers

2. Scaling and the Future

Scaling protocol analysis?

Holy grail:
analyze complex real-world systems in full detail

Challenges:

- Modeling complexity for humans
  - Extracting from code and directly considering code unsuccessful so far
- Analysis complexity for algorithm at least exponential
  - Underlying problem undecidable

Wider question for future developments:

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

My original intuition:
Probably there are plenty of examples.

My current intuition:
Not so sure anymore there are many interesting ones!

My group is hiring!

If you liked this course and got great grades, maybe consider working with us!

- **Bachelor/Master thesis**

- **Hiwis**
  - Most urgent need: someone with affinity for *functional programming*
  - Frontend (real experience with angular/D3.js)

https://cispa.saarland/group/cremers/index.html

Good luck!

Thank you for attending and good luck with the project!

Hope you enjoyed the course and gained a lot of knowledge

cremers@cispa.de

Based on joint work with:

Tamarin:	David Basin, Simon Meier, Benedikt Schmidt, Ralf Sasse, Jannik Dreier, and many others
Signatures:	Dennis Jackson, Katriel Cohn-Gordon, Ralf Sasse, Lukas Schmidt
DH:	Dennis Jackson
Ed25519:	Jacqueline Brendel, Dennis Jackson, Mang Zhao
PQ sigs:	Samed Düzlü, Rune Fiedler, Marc Fischlin, Christian Janson
Hashes:	Vincent Cheval, Alexander Dax, Lucca Hirschi, Charlie Jacomme, Steve Kremer