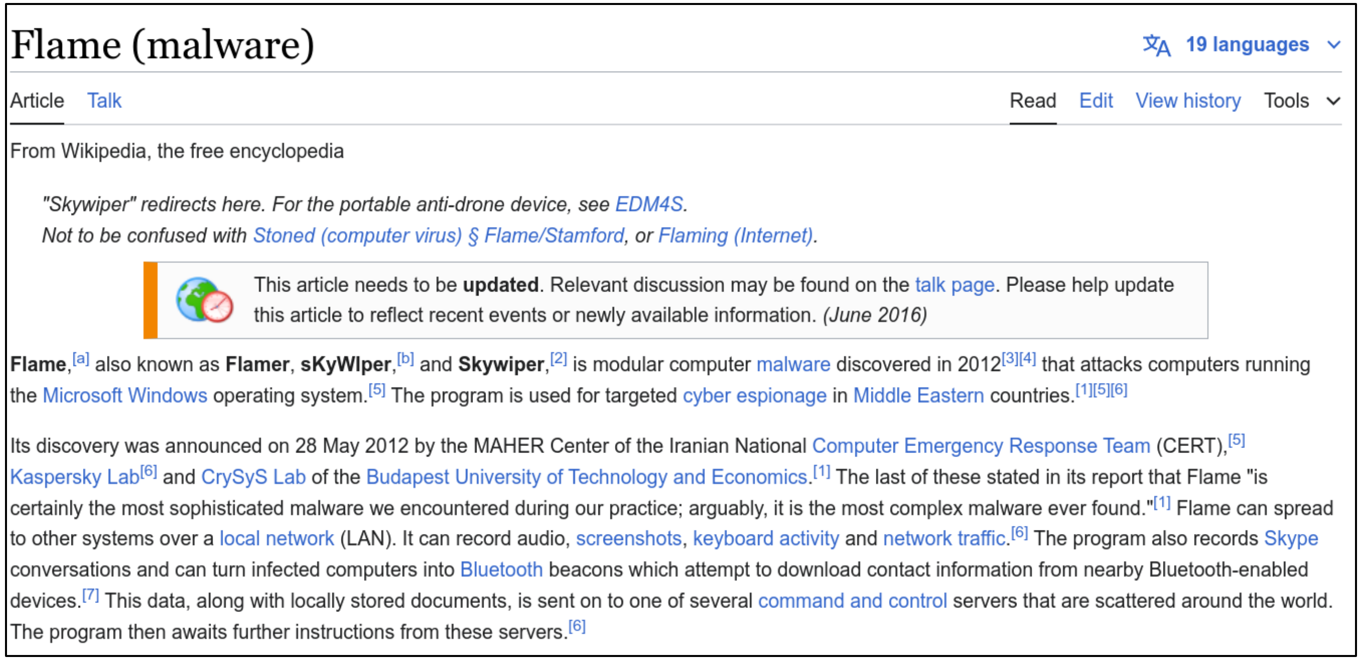
Post-Quantum Cryptography

Andreas Hülsing
TU Eindhoven & SandboxAQ

Why do cyber criminals focus on implementation bugs, phishing & co?

Because cryptography keeps them from simply taking over your communication!

And if crypto fails?



Flame (malware)

From Wikipedia, the free encyclopedia

"Skywiper" redirects here. For the portable anti-drone device, see EDM4S.
Not to be confused with Stoned (computer virus) § Flame/Stamford, or Flaming (Internet).

This article needs to be **updated**. Relevant discussion may be found on the talk page. Please help update this article to reflect recent events or newly available information. *(June 2016)*

Flame,[a] also known as **Flamer**, **sKyWIper**,[b] and **Skywiper**,[2] is modular computer malware discovered in 2012[3][4] that attacks computers running the Microsoft Windows operating system.[5] The program is used for targeted cyber espionage in Middle Eastern countries.[1][5][6]

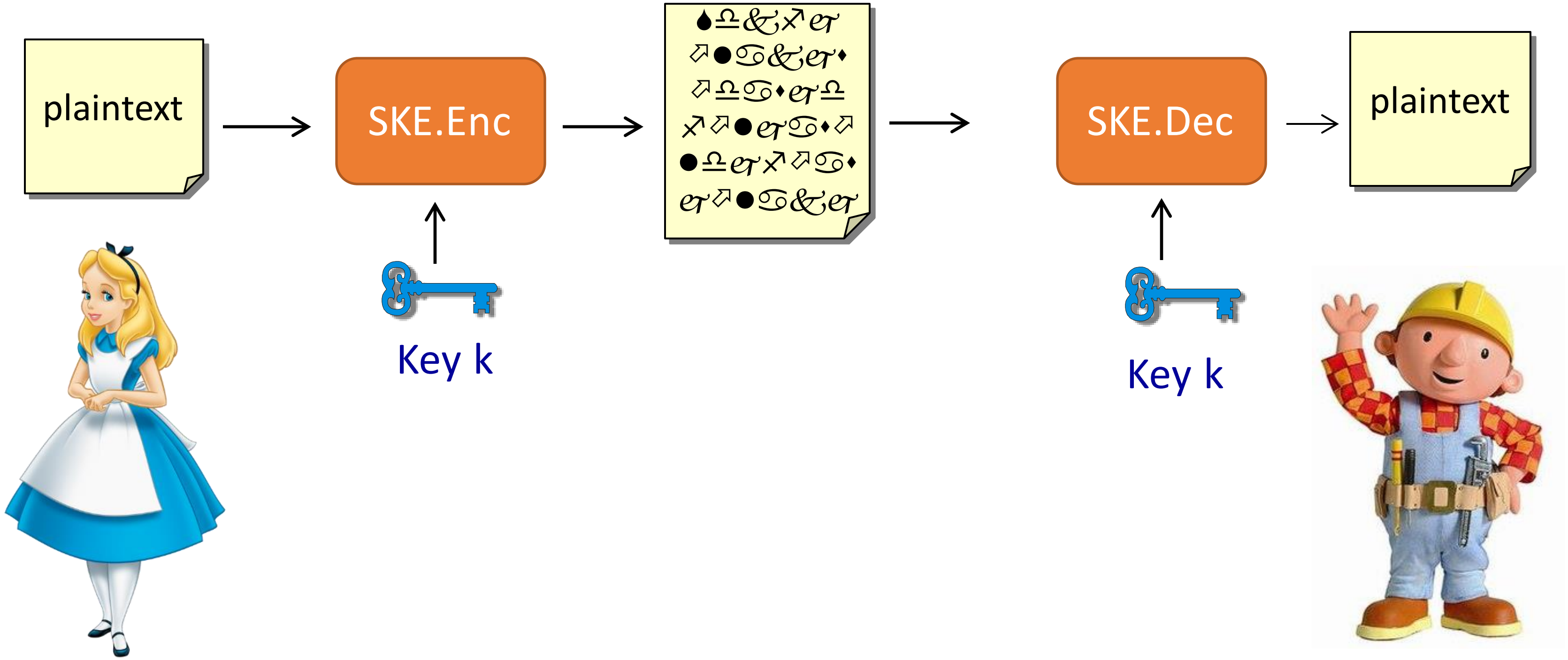
Its discovery was announced on 28 May 2012 by the MAHER Center of the Iranian National Computer Emergency Response Team (CERT),[5] Kaspersky Lab[6] and CrySyS Lab of the Budapest University of Technology and Economics.[1] The last of these stated in its report that Flame "is certainly the most sophisticated malware we encountered during our practice; arguably, it is the most complex malware ever found."[1] Flame can spread to other systems over a local network (LAN). It can record audio, screenshots, keyboard activity and network traffic.[6] The program also records Skype conversations and can turn infected computers into Bluetooth beacons which attempt to download contact information from nearby Bluetooth-enabled devices.[7] This data, along with locally stored documents, is sent on to one of several command and control servers that are scattered around the world. The program then awaits further instructions from these servers.[6]

Flame was signed with a fraudulent certificate purportedly from the Microsoft Enforced Licensing Intermediate PCA certificate authority.[19] The malware authors identified a Microsoft Terminal Server Licensing Service certificate that inadvertently was enabled for code signing and that still used the weak MD5 hashing algorithm, then produced a counterfeit copy of the certificate that they used to sign some components of the malware to make them appear to have originated from Microsoft.[19] A successful collision attack against a certificate was previously demonstrated in 2008,[20] but Flame implemented a new variation of the chosen-prefix collision attack.[21]

Property	Value
Compromised Microsoft certificate using the weak MD5 algorithm, and the unintended code-signing usage	

Background: Cryptography

Secret key encryption (SKE)



plaintext

SKE.Enc

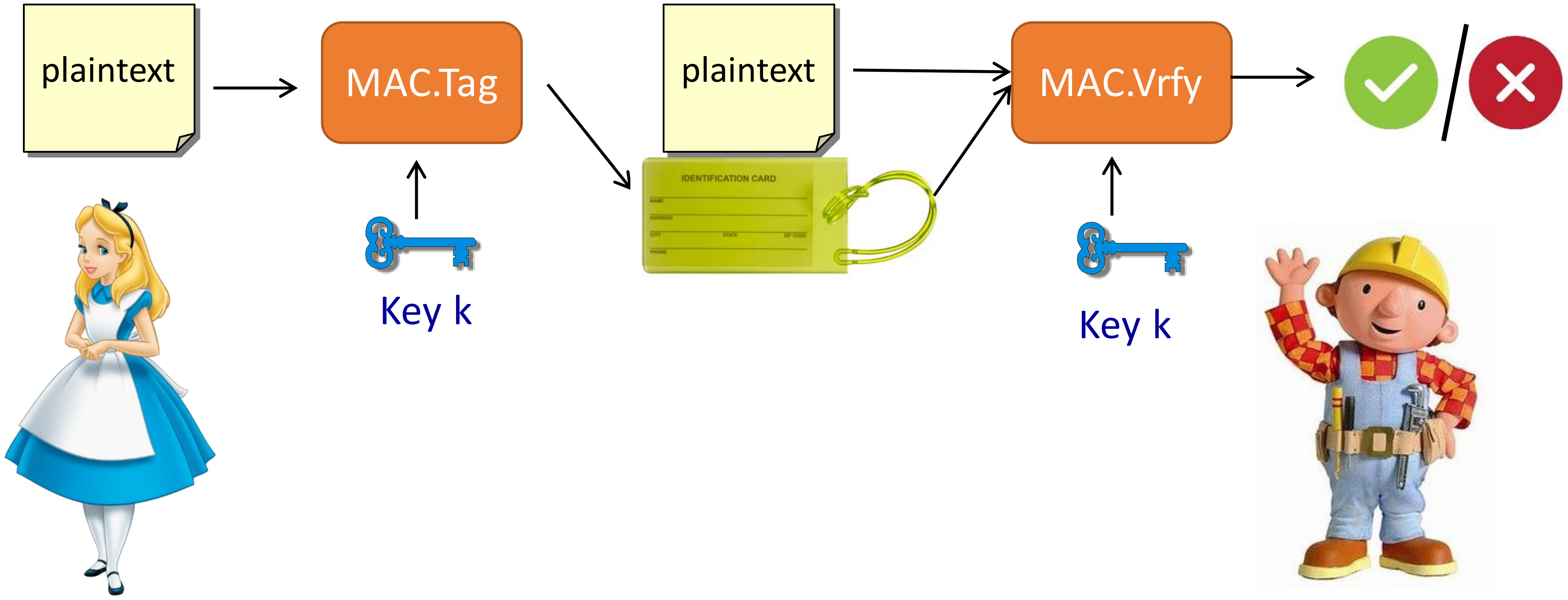
Key k

SKE.Dec

Key k

plaintext

Message authentication (MAC)



plaintext

MAC.Tag

Key k

plaintext

MAC.Vrfy

Key k

✓/✗

How to build secret key crypto?

- Random function sufficient (we need one-wayness)
- A
- H

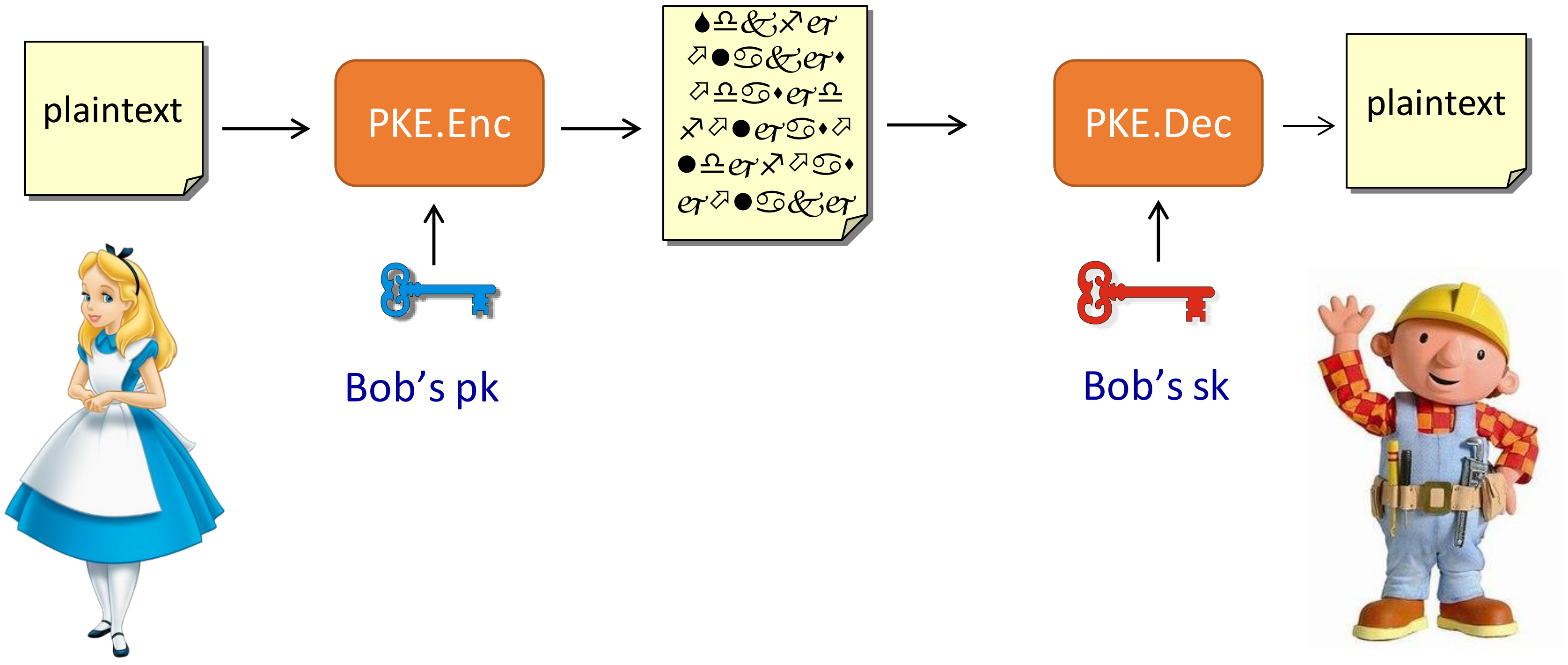
Spoiler:
Killed by quantum? Not that we know.
(but weakened)*

Engineering*

* Disclaimer: Massive simplification

How does Bob learn shared key k?

Public key encryption (PKE)

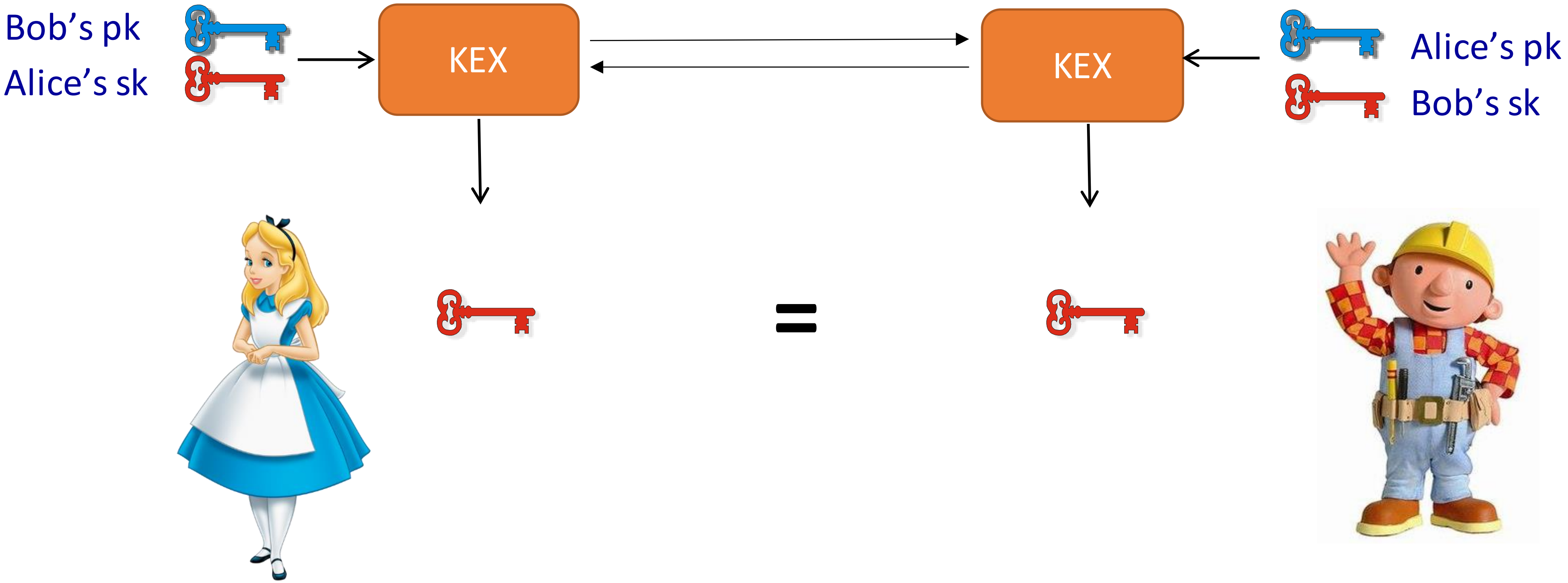


plaintext → PKE.Enc → → PKE.Dec → plaintext

Bob’s pk

Bob’s sk

Key Exchange (KEX)



Bob’s pk

Alice’s sk

KEX

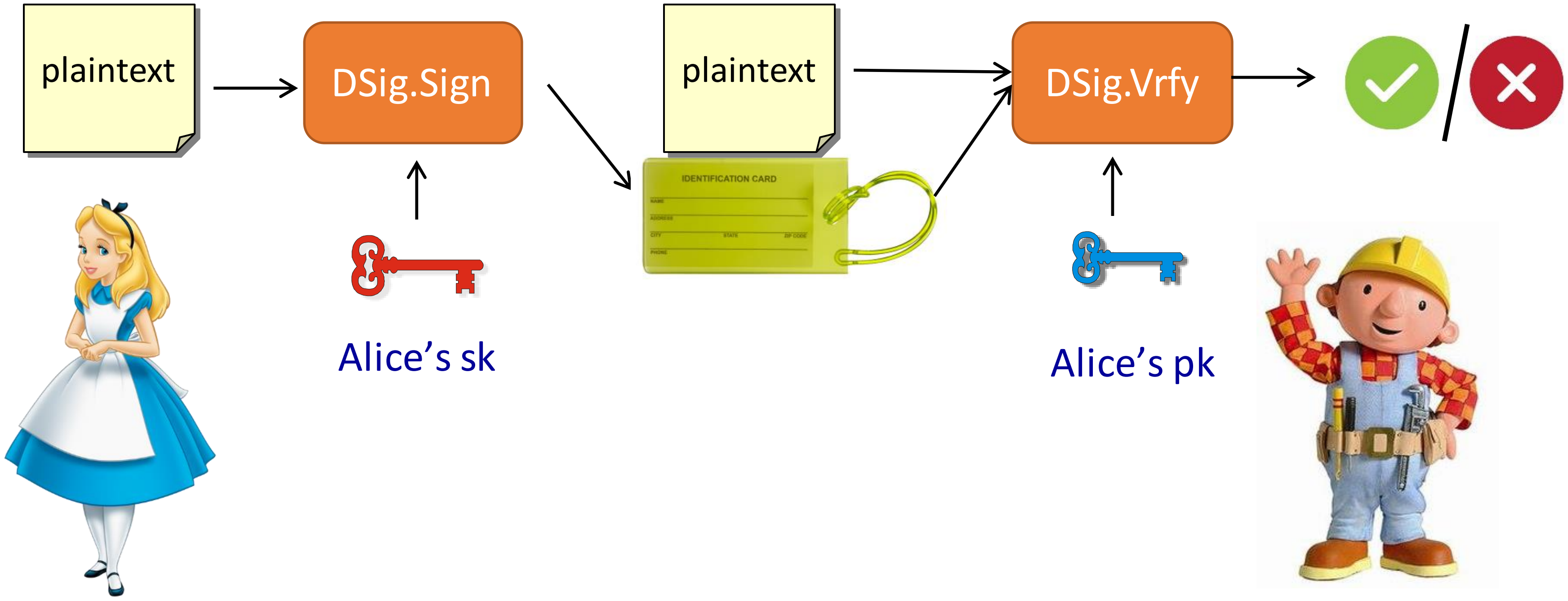
KEX

Alice’s pk

Bob’s sk

=

Digital Signature (DSig)



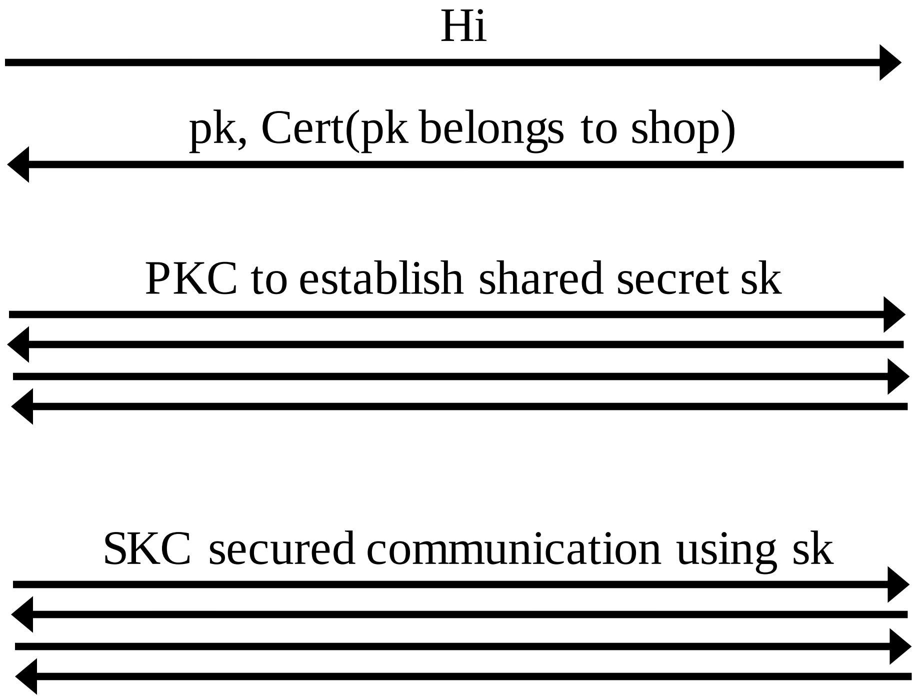
plaintext → DSig.Sign

Alice's sk

plaintext → DSig.Vrfy → ✓/✗

Alice's pk

Communication security (simplified)



Hi

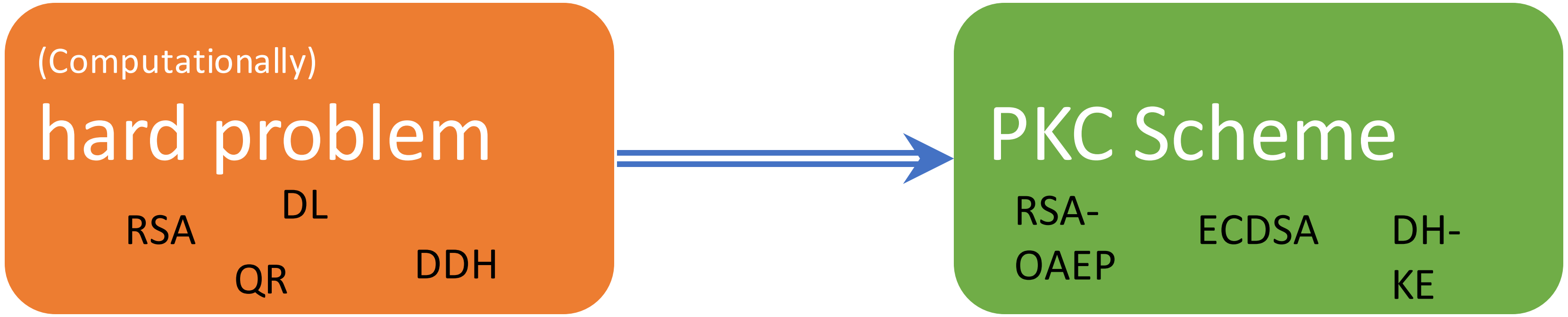
pk, Cert(pk belongs to shop)

PKC to establish shared secret sk

SKC secured communication using sk

The
TEA SHOP
FINE TEAS & TISANES
PURE & BLENDED

How to build PKC



(Computationally)

hard problem

RSA DL QR DDH

PKC Scheme

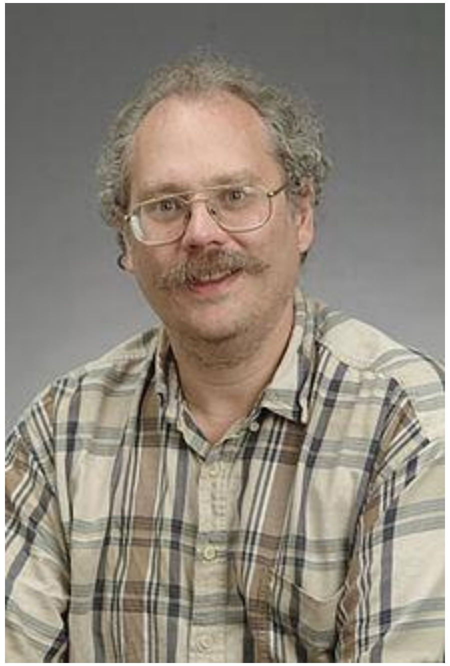
RSA-OAEP ECDSA DH-KE

The Quantum Threat

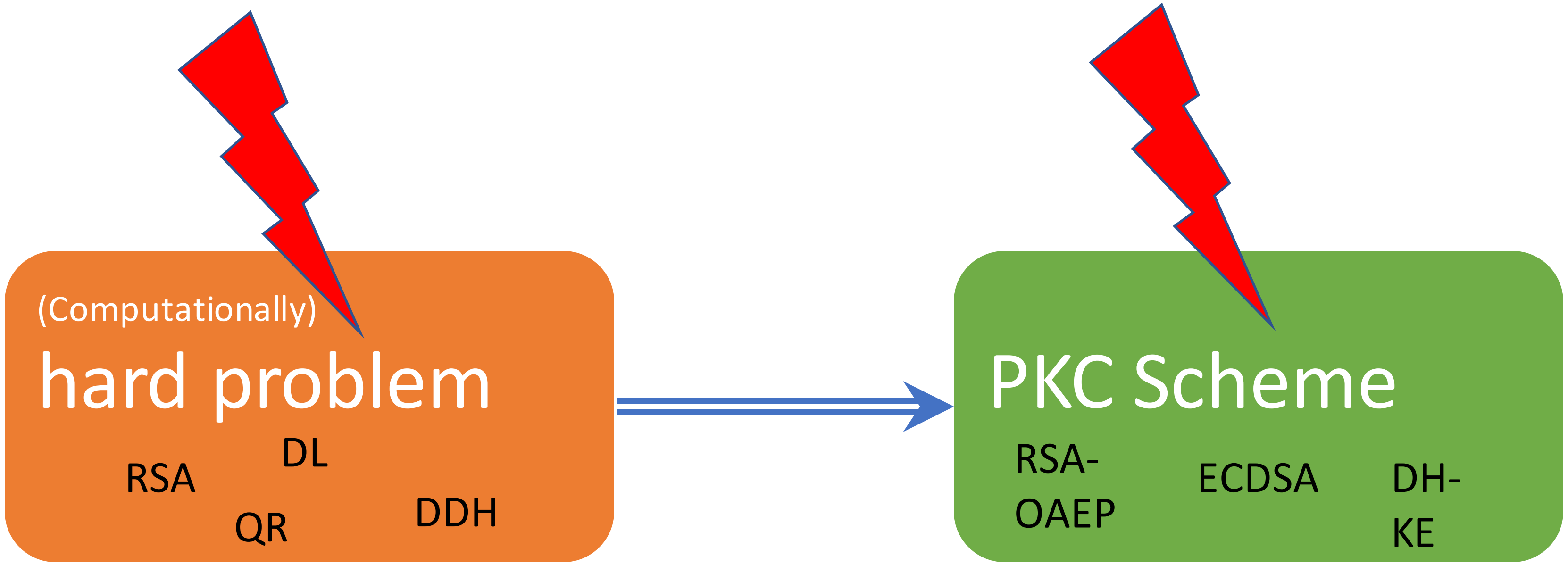
Quantum kills the Internet

Shor‘s algorithm (1994)

- Quantum computers can do FFT very efficiently
- Can be used to find period of a function
- This can be exploited to factor efficiently (RSA)
- Shor also shows how to solve discrete log efficiently (DSA, DH, ECDSA, ECDH)



How to build PKC



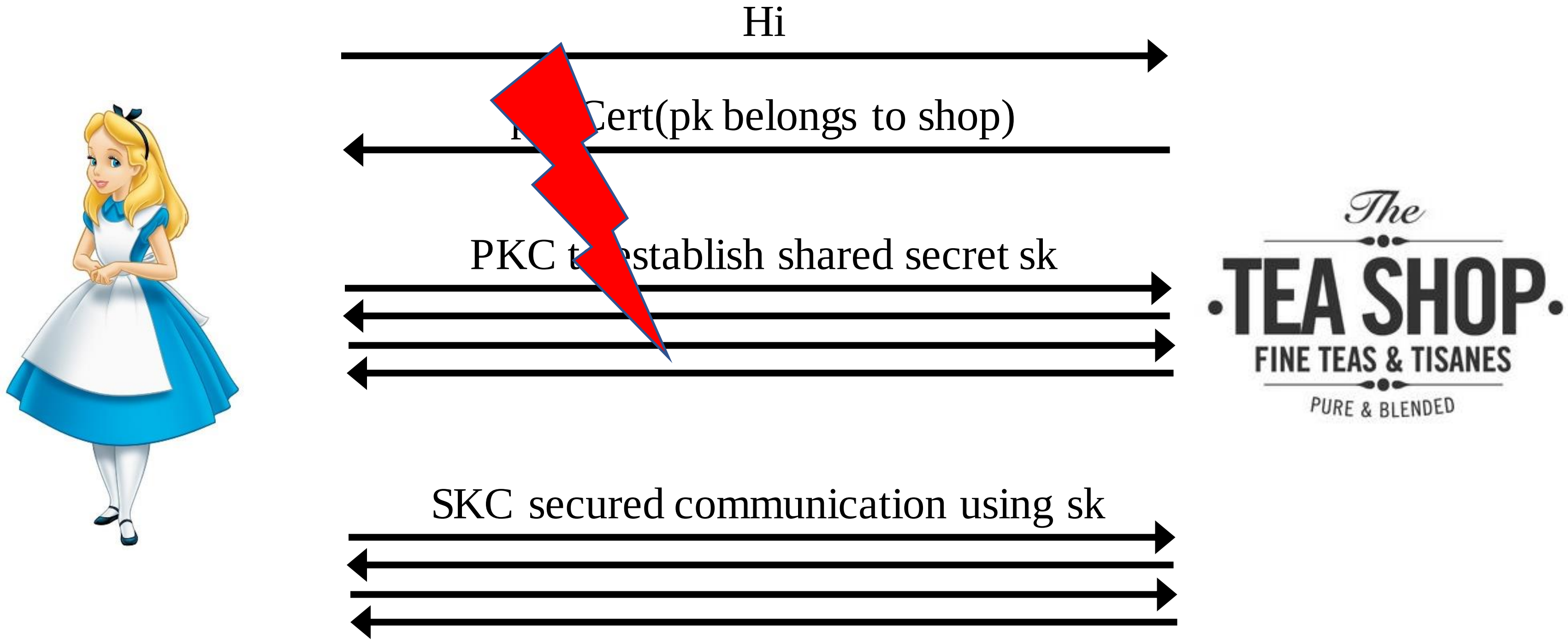
(Computationally)
hard problem

RSA DL QR DDH

PKC Scheme

RSA-OAEP ECDSA DH-KE

Communication security (simplified)



Hi

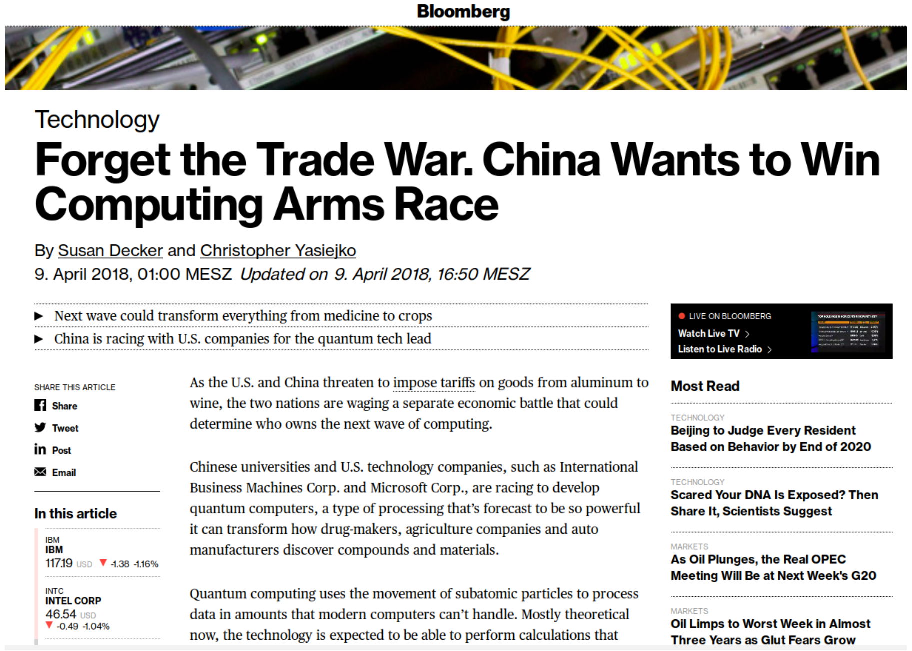
pk, Cert(pk belongs to shop)

PKC to establish shared secret sk

SKC secured communication using sk

Why care today

- **EU** launched a one billion Euro project on quantum technologies
- Similar range is spent in **China**
- **US** administration passed a bill on spending $1.275 billion US dollar on quantum computing research
- **Google**, **IBM**, **Microsoft**, **Alibaba**, and others run their own research programs.



Technology

Forget the Trade War. China Wants to Win Computing Arms Race

By Susan Decker and Christopher Yasiejko

9. April 2018, 01:00 MESZ *Updated on 9. April 2018, 16:50 MESZ*

- Next wave could transform everything from medicine to crops
- China is racing with U.S. companies for the quantum tech lead

As the U.S. and China threaten to impose tariffs on goods from aluminum to wine, the two nations are waging a separate economic battle that could determine who owns the next wave of computing.

Chinese universities and U.S. technology companies, such as International Business Machines Corp. and Microsoft Corp., are racing to develop quantum computers, a type of processing that's forecast to be so powerful it can transform how drug-makers, agriculture companies and auto manufacturers discover compounds and materials.

Quantum computing uses the movement of subatomic particles to process data in amounts that modern computers can't handle. Mostly theoretical now, the technology is expected to be able to perform calculations that

In this article

IBM
IBM
117.19 USD ▼ -1.38 -1.16%

INTC
INTEL CORP
46.54 USD
▼ -0.49 -1.04%

Most Read

TECHNOLOGY
Beijing to Judge Every Resident Based on Behavior by End of 2020

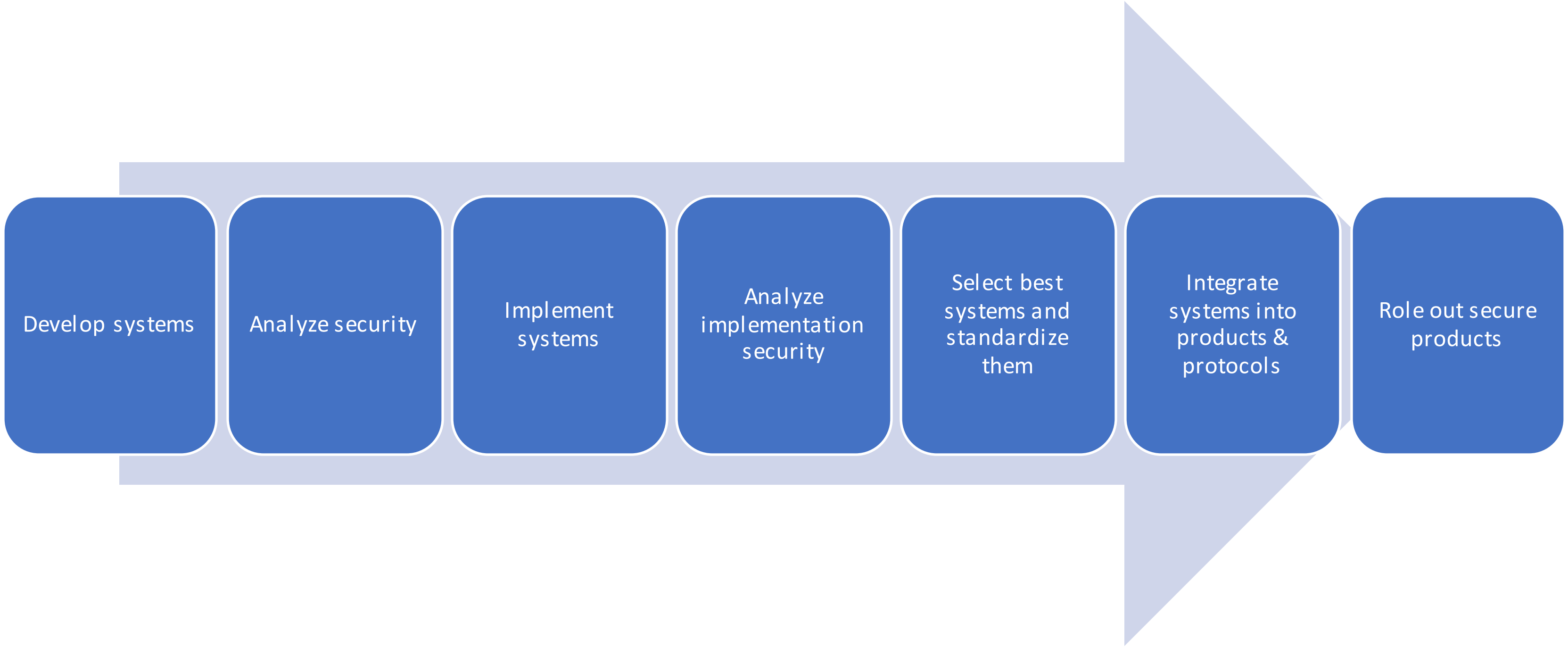
TECHNOLOGY
Scared Your DNA Is Exposed? Then Share It, Scientists Suggest

MARKETS
As Oil Plunges, the Real OPEC Meeting Will Be at Next Week's G20

MARKETS
Oil Limps to Worst Week in Almost Three Years as Glut Fears Grow

It‘s a question of risk assessment

Real world cryptography development



Develop systems

Analyze security

Implement systems

Analyze implementation security

Select best systems and standardize them

Integrate systems into products & protocols

Role out secure products

Who would store all encrypted data traffic?
That must be expensive!

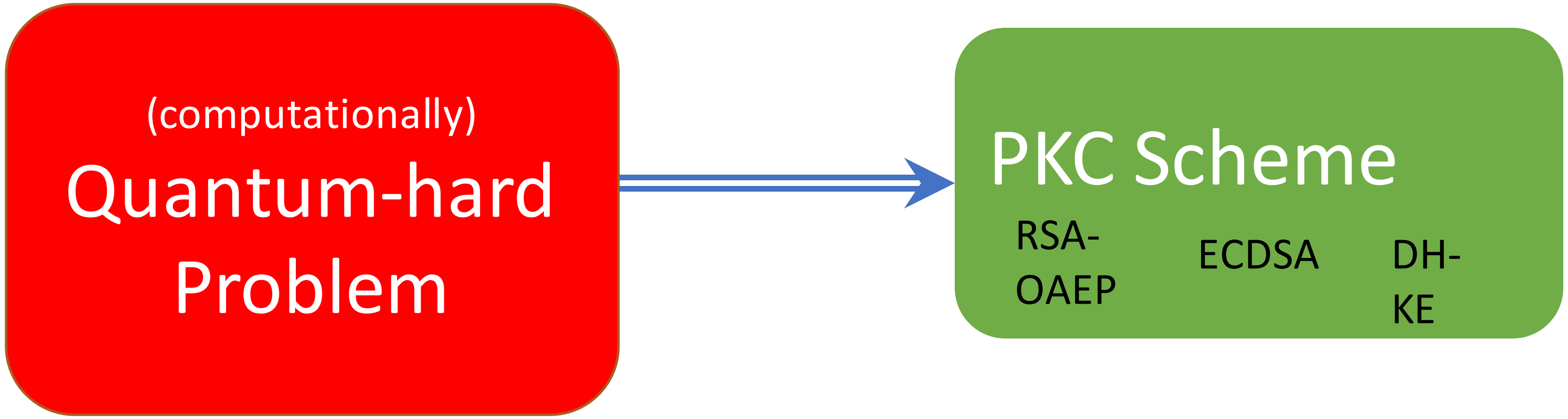
Long-lived systems

- Development time easily 10+ years
- Lifetime easily 10+ years

- At least make sure you got a secure update channel!

Solution to the problem caused by Shor?
Post-quantum cryptography

How to build PKC



(computationally)
Quantum-hard Problem

PKC Scheme

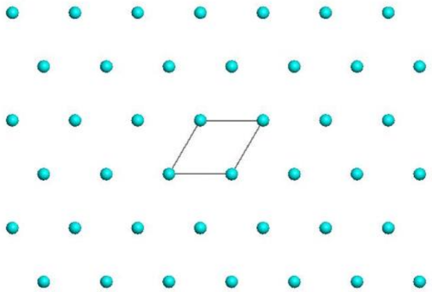
RSA-OAEP

ECDSA

DH-KE

Quantum-hard problems

Lattice-based: SVP / CVP



Hash-based: CR / SPR / ...

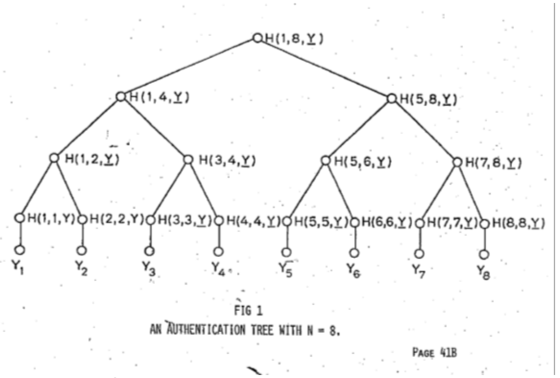
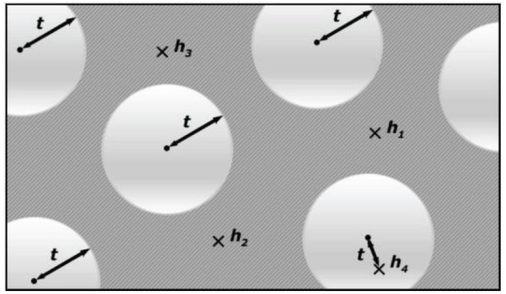


FIG 1
AN AUTHENTICATION TREE WITH N = 8.

Code-based: SD



Multivariate: MQ

$$y_1 = x_1^2 + x_1x_2 + x_1x_4 + x_3$$

$$y_2 = x_3^2 + x_2x_3 + x_2x_4 + x_1 + 1$$

$$y_3 = \dots$$

NIST Competition

CSRC HOME > GROUPS > CT > POST-QUANTUM CRYPTOGRAPHY PROJECT

POST-QUANTUM CRYPTO PROJECT

NEWS -- December 15, 2016: The National Institute of Standards and Technology (NIST) is now accepting submissions for quantum-resistant public-key cryptographic algorithms. The deadline for submission is ***November 30, 2017.*** Please see the Post-Quantum Cryptography Standardization menu at left for the complete submission requirements and evaluation criteria.

“We see our role as managing a process of achieving community consensus in a transparent and timely manner” NIST’s Dustin Moody 2018

Status of the competition

- Nov 2017: 82 submissions collected
- Dec 2017: 69 “complete & proper” proposals published
  - -> Starts round 1 (of 2 or 3 rounds)
- Jan 2019: 26 proposals selected for 2nd round.
  - 17 KEM, 9 Signature
- July 2020: 7 Finalists and 8 Alternate candidates selected for 3rd round
  - 4+5 KEM, 3+3 DSS
- July 2022 – End of 3rd round – Winners announced
- 2022-2023 – Release draft standards and call for public comments

Selected Algorithms

- KEM:
  - Crystals-Kyber (ML-KEM
- Sig:
  - Crystals-Dilithium (ML-D
  - Falcon (FN-DSA)
  - SPHINCS+ (SLH-DSA)

Dutch (Expads) Success!

- **Kyber** led by Peter Schwabe (then RU),
  with team member Leo Ducas (CWI)
- **Dilithium** with team members Schwabe (then RU) and Ducas (CWI)
- **SPHINCS+** led by Andreas Hülsing (TU/e)
  with team members Daniel J. Bernstein (then TU/e), Tanja Lange (TU/e), Ruben Niederhagen (then TU/e), Joost Rijneveld(then RU), Peter Schwabe (then RU), Bas Westerbaan (Cloudflare)

- SOLID SECURITY, BUT PERFORMANCE NOT AS GOOD IN COMPARISON TO DILITHIUM/FALCON


ARE WE THERE YET? ARE WE THERE YET?
HOW ABOUT NOW?
memegenerator.net

This is what we are actually interested in!


THE NEW SCHEMES
ARE NO
PLUG'N'PLAY REPLACEMENTS
imgflip.com

Challenges

(Along the example of PQWireGuard
[Hülsing, Ning, Schwabe, Weber, Zimmermann. S&P 2021])

Challenges

1. Size
2. Speed
3. Interface mismatch (KEM ≠ NIKE)
4. Security models
5. Standardizing the new protocols
6. Hybrids

Challenge 1: Size

- IPv6 Maximum Transmission Unit (MTU) = 1280 bytes = 1232 bytes + headers.
- Bigger packets risk fragmentation
  - complicates state-machine
  - can allow DoS

Sec Lvl	Kyber		Saber		NTRU		McEliece	
	PK	Ct	PK	Ct	PK	Ct	PK	Ct
I	800	768	672	736	699	699	261.120	128
III	1.184	1.088	992	1.088	930	930	524.160	188
V	1.568	1.568	1.312	1.472	1.230	1.230	1.044.992	240

Challenge 1: Size

- IPv6 Maximu) = 1280 bytes
  = 1232 bytes
- Bigger packe
  - complicate
  - can allow Do

PQWireGuard:
Some MACs + pk + ct /
Some MACs + 2 ct

Sec Lvl	Kyber		Saber		NTRU		McEliece	
	PK	Ct	PK	Ct	PK	Ct	PK	Ct
I	800	768	672	736	699	699	261.120	128
III	1.184	1.088	992	1.088	930	930	524.160	188
V	1.568	1.568	1.312	1.472	1.230	1.230	1.044.992	240

Challenge 1: Size

- IPv6 Maximu) =
  = 1232 bytes
- Bigger packe
  - complicate
  - can allow DoS

<u>PQWireGuard:</u>
Some MACs + pk + ct / ⚡
Some MACs + 2 ct

Solution:
McEliece +
passively secure Saber

Sec Lvl	Kyber		Saber		NTRU		McEliece	
	PK	Ct	PK	Ct	PK	Ct	PK	Ct
I	800	768	672	736	699	699	261.120	128
III	1.184	1.088	992	1.088	930	930	524.160	188
V	1.568	1.568	1.312	1.472	1.230	1.230	1.044.992	240

Challenge 1: Size

- IPv6 Maximu) =
  = 1232 bytes
- Bigger packe
  - complicate
  - can allow Do

<u>PQWireGuard:</u>
Some MACs + pk + ct /
S

Solution:
McEliece +
secure Saber

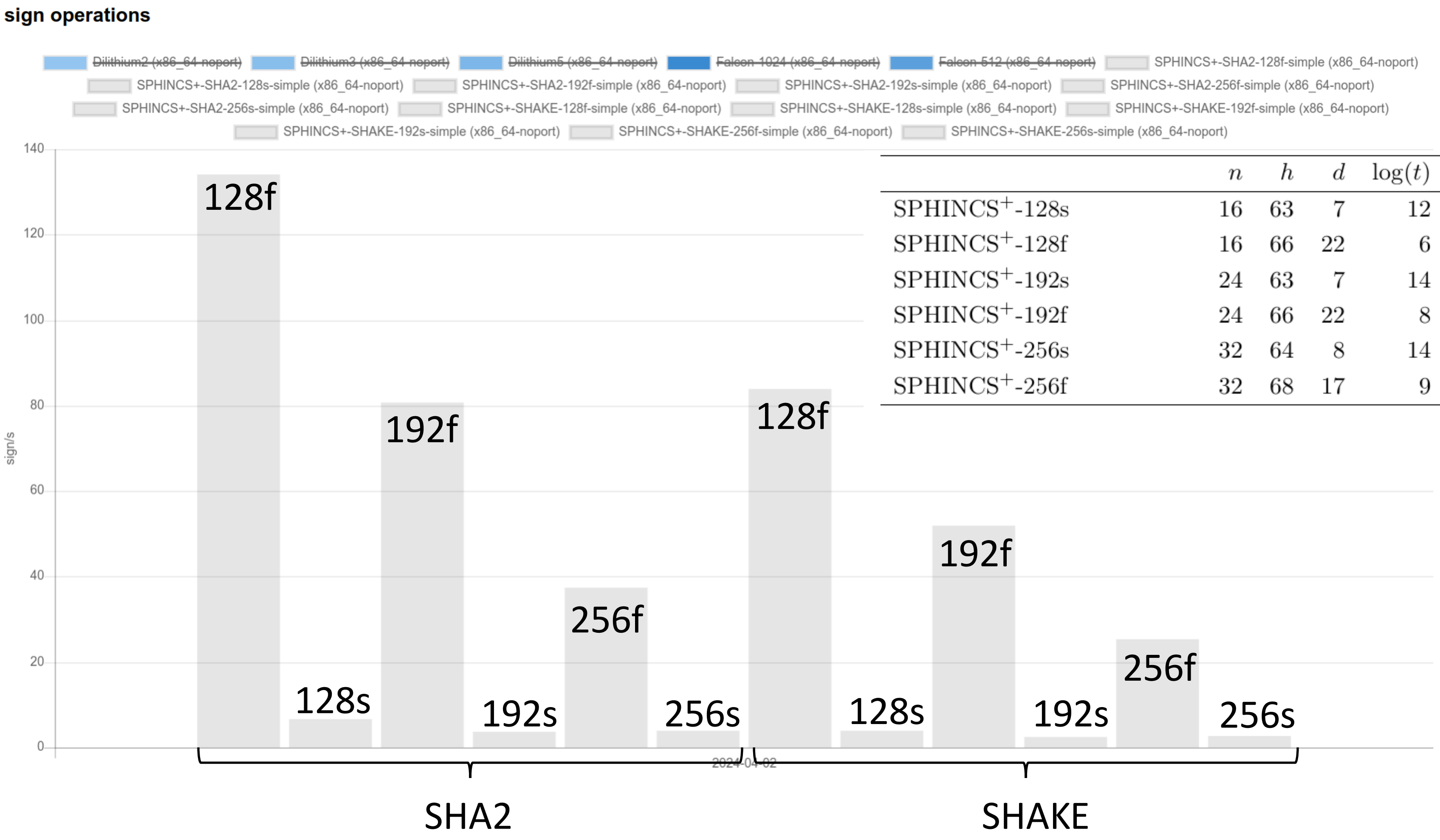
Similar situation for signatures!

Sec Lvl	Kyber						McEliece	
	PK						PK	Ct
I	800	768					261.120	128
III	1.184	1.08					524.160	188
V	1.568	1.568	1.312	1.472	1.230	1.230	1.044.992	240

Challenge 2: Speed

- Often we have trade-offs speed vs size.

Sign/s for SPHINCS+



	n	h	d	$\log(t)$	k	w	bitsec	sec level	sig bytes
SPHINCS$^+$-128s	16	63	7	12	14	16	133	**1**	7 856
SPHINCS$^+$-128f	16	66	22	6	33	16	128	**1**	17 088
SPHINCS$^+$-192s	24	63	7	14	17	16	193	**3**	16 224
SPHINCS$^+$-192f	24	66	22	8	33	16	194	**3**	35 664
SPHINCS$^+$-256s	32	64	8	14	22	16	255	**5**	29 792
SPHINCS$^+$-256f	32	68	17	9	35	16	255	**5**	49 856

Source: The open Quantum Safe Project, https://openquantumsafe.org/benchmarking/visualization/speed_sig.html

Challenge 1+2: Performance

- Performance penalty is noticeable
  - Only use PKC where really needed!
- Performance penalty is bigger for signatures
  - Only use signatures when needed

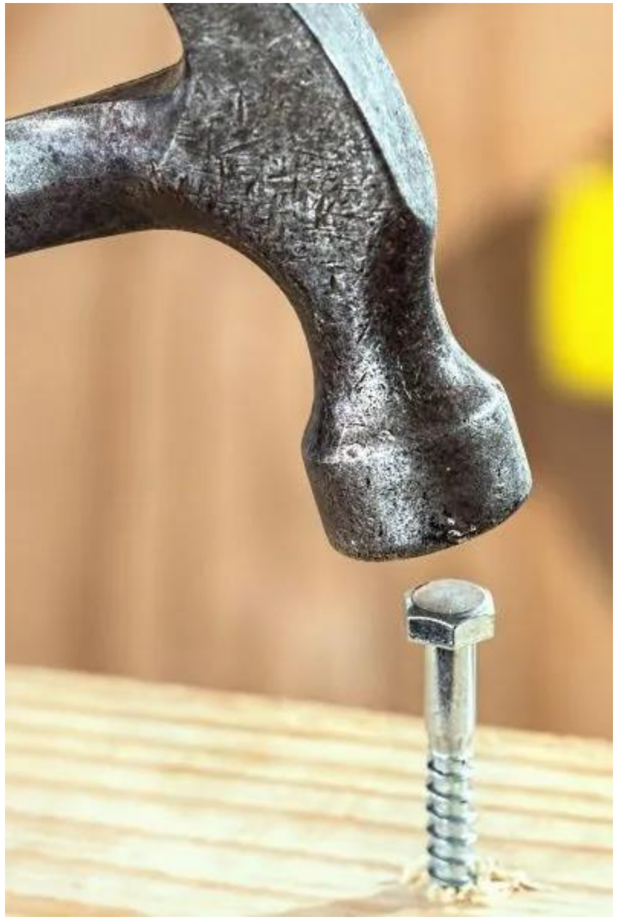
Challenge 1+2: Performance

- Performance penalty is noticeable
    - Only use PKC where really needed!
- Performance penalty is bigger for signatures
    - Only use signatures when needed

PQWireGuard:
Use KEM for
authentication

Challenge 3: KEM no NIKE (DH)

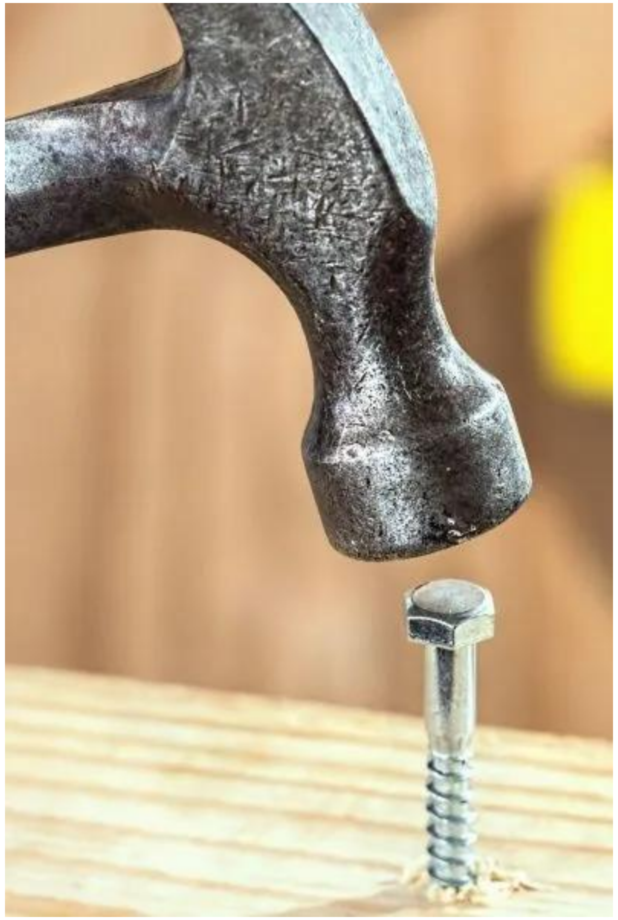
- Key transport in TLS 1.3: (EC)DH
- Key transport in WireGuard: ECDH
- Key transport in Noise: (EC)DH
- Key transport in Signal, WhatsApp, …: (EC)DH



Challenge 3: KEM no NIKE (DH)

- Key transport in TLS 1.3: (EC)DH
- Key transport in WireGuard: ECDH
- Key transport in Noise: (EC)DH
- Key transport in Signal, WhatsApp, …: (EC)DH

NIST will standardize: KEM



Key transport

NIKE

A(skA,pkA,pkB)

K = f(skA,pkB)
C = Enc(K,M)

$\xrightarrow{C}$

B(skB,pkB,pkA)

K = f(skB,pkA)
M = Dec(K,C)

KEM

A(skA,pkA,pkB)

K,ct = Encaps(pkB)
C = Enc(K,M)

$\xrightarrow{ct,C}$

B(skB,pkB,pkA)

K = Decaps(skB,ct)
M = Dec(K1,C)

Key transport

NIKE

<u>A(skA,pkA,pkB)</u> B(skB,pkB,pkA)

K = f(skA,pkB)
C = Enc(K,M)

$\xrightarrow{C}$

K = f(skB,pkA)
M = Dec(K,C)

Only A could have computed this K!

KEM

<u>A(skA,pkA,pkB)</u> B(skB,pkB,pkA)

K,ct = Encaps(pkB)
C = Enc(K,M)

$\xrightarrow{ct,C}$

K = Decaps(skB,ct)
M = Dec(K1,C)

<u>**Anyone**</u> could have computed this K!

Key transport

NIKE

A(skA,pkA,pkB)

B(skB,pkB,pkA)

K = f(skA,pkB)
C = Enc(K,M)

$\xrightarrow{C}$

K = f(skB,pkA)
M = Dec(K,C)

Only A could have computed this K!

I can even already mix in ephemeral keys!

KEM

A(skA,pkA,pkB)

B(skB,pkB,pkA)

K,ct = Encaps(pkB)
C = Enc(K,M)

$\xrightarrow{ct,C}$

K = Decaps(skB,ct)
M = Dec(K1,C)

Anyone could have computed this K!

Key transport

NIKE

<u>A(skA,pkA,pkB)</u>

K = f(skA,pkB)
C = Enc(K,M)

$\xrightarrow{C}$

<u>B(skB,pkB,pkA)</u>

K = f(skB,pkA)
M = Dec(K,C)

Only A could have computed this K!

I can even already mix in ephemeral keys!

KEM

<u>A(skA,pkA,pkB)</u>

K,ct = Encaps(pkB)
C = Enc(K,M)

$\xrightarrow{ct,C}$

<u>B(skB,pkB,pkA)</u>

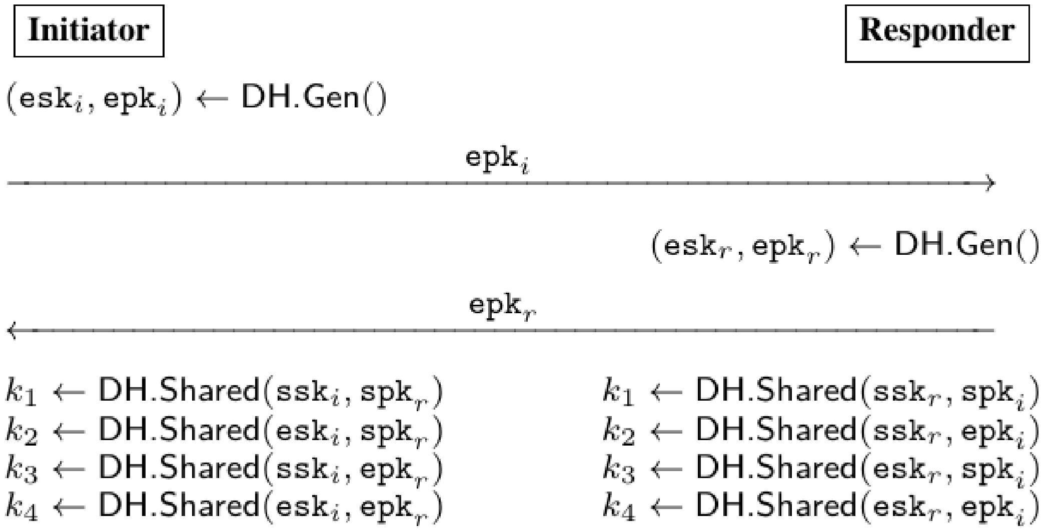
K = Decaps(skB,ct)
M = Dec(K1,C)

<u>Anyone</u> could have computed this K!

Can be rescued with one more message!

WireGuard vs PQWireGuard

WireGuard



Initiator

Responder

$(\mathsf{esk}_i, \mathsf{epk}_i) \leftarrow \mathsf{DH.Gen}()$

$\xrightarrow{\mathsf{epk}_i}$

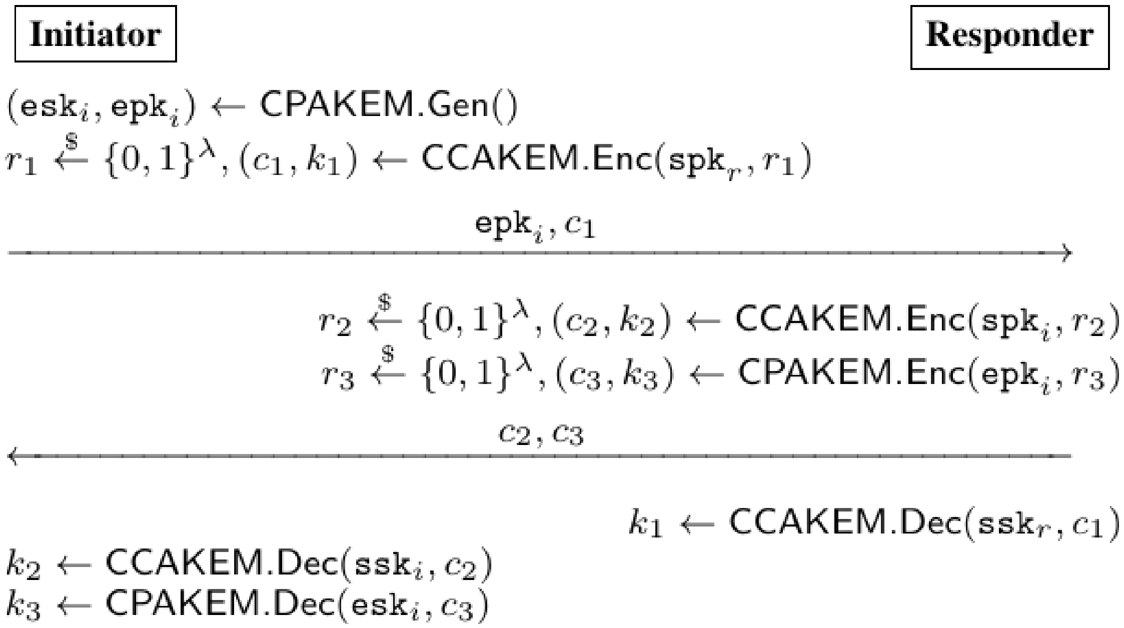
$(\mathsf{esk}_r, \mathsf{epk}_r) \leftarrow \mathsf{DH.Gen}()$

$\xleftarrow{\mathsf{epk}_r}$

$k_1 \leftarrow \mathsf{DH.Shared}(\mathsf{ssk}_i, \mathsf{spk}_r)$
$k_2 \leftarrow \mathsf{DH.Shared}(\mathsf{esk}_i, \mathsf{spk}_r)$
$k_3 \leftarrow \mathsf{DH.Shared}(\mathsf{ssk}_i, \mathsf{epk}_r)$
$k_4 \leftarrow \mathsf{DH.Shared}(\mathsf{esk}_i, \mathsf{epk}_r)$

$k_1 \leftarrow \mathsf{DH.Shared}(\mathsf{ssk}_r, \mathsf{spk}_i)$
$k_2 \leftarrow \mathsf{DH.Shared}(\mathsf{ssk}_r, \mathsf{epk}_i)$
$k_3 \leftarrow \mathsf{DH.Shared}(\mathsf{esk}_r, \mathsf{spk}_i)$
$k_4 \leftarrow \mathsf{DH.Shared}(\mathsf{esk}_r, \mathsf{epk}_i)$

PQWireGuard



Initiator

Responder

$(\mathsf{esk}_i, \mathsf{epk}_i) \leftarrow \mathsf{CPAKEM.Gen}()$
$r_1 \xleftarrow{\$} \{0,1\}^\lambda, (c_1, k_1) \leftarrow \mathsf{CCAKEM.Enc}(\mathsf{spk}_r, r_1)$

$\xrightarrow{\mathsf{epk}_i, c_1}$

$r_2 \xleftarrow{\$} \{0,1\}^\lambda, (c_2, k_2) \leftarrow \mathsf{CCAKEM.Enc}(\mathsf{spk}_i, r_2)$
$r_3 \xleftarrow{\$} \{0,1\}^\lambda, (c_3, k_3) \leftarrow \mathsf{CPAKEM.Enc}(\mathsf{epk}_i, r_3)$

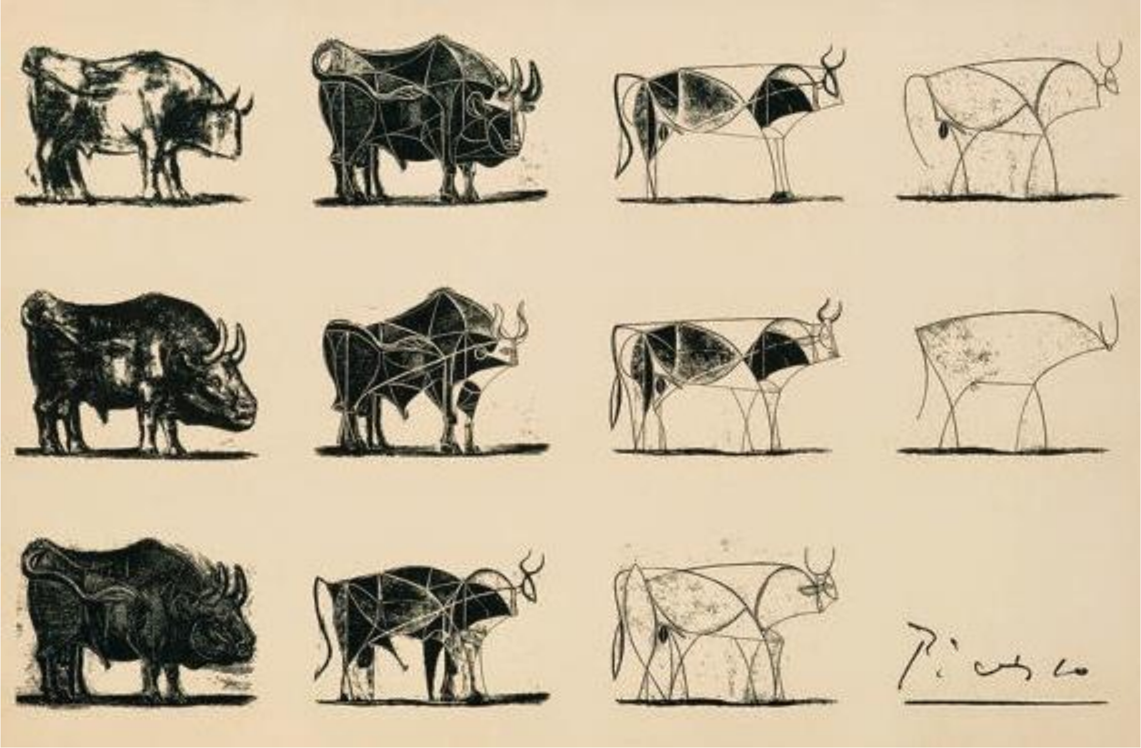
$\xleftarrow{c_2, c_3}$

$k_1 \leftarrow \mathsf{CCAKEM.Dec}(\mathsf{ssk}_r, c_1)$

$k_2 \leftarrow \mathsf{CCAKEM.Dec}(\mathsf{ssk}_i, c_2)$
$k_3 \leftarrow \mathsf{CPAKEM.Dec}(\mathsf{esk}_i, c_3)$

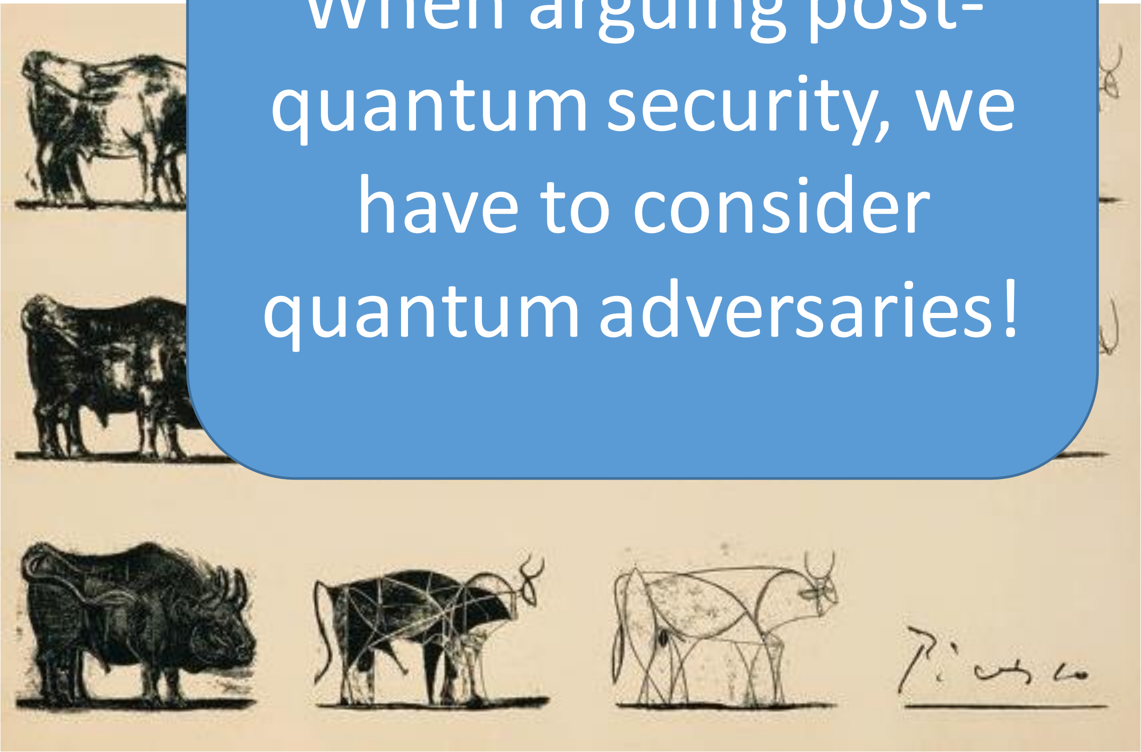
Challenge 4: Security models

- When arguing security,
  we have to simplify -> models
  - IND-CPA, IND-CCA, EUF-CMA, …
- Sometimes, we can only argue security when idealizing (some) building blocks -> idealized models
  - Random Oracle Model,
    UC-Framework



Challenge 4: Security models

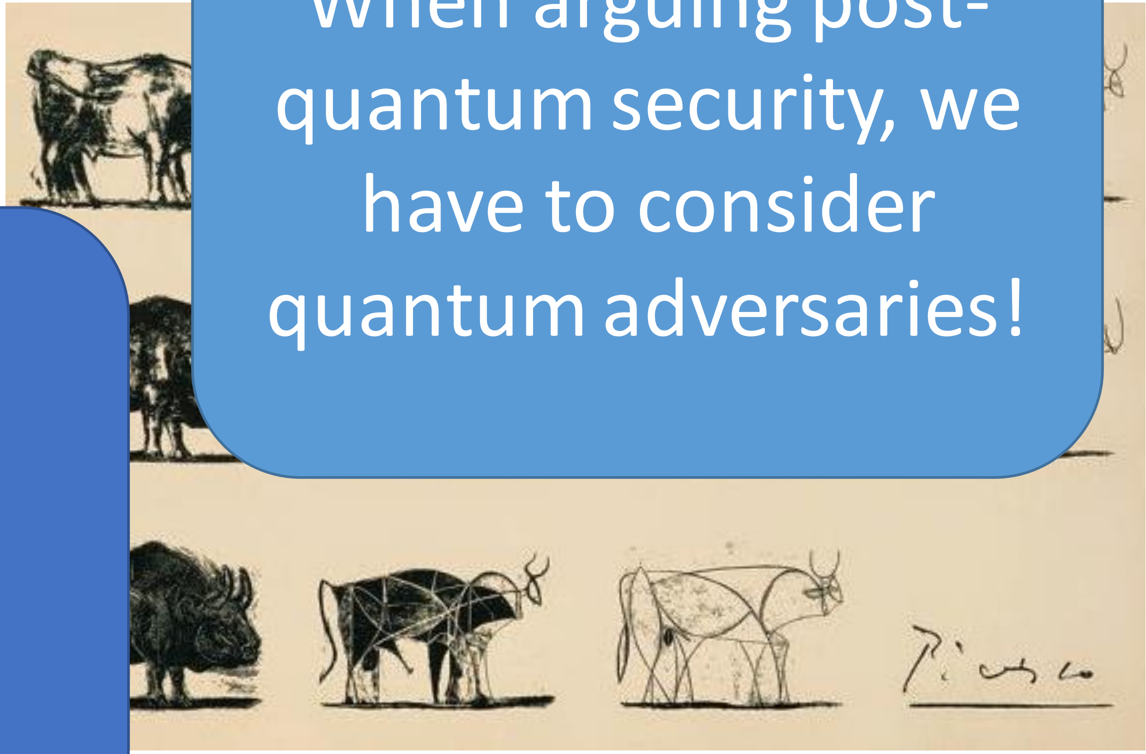
- When arguing security, we have to simplify -> models
  - IND-CPA, IND-CCA, EUF-CMA, …
- Sometimes, we can only argue security when idealizing (some) building blocks -> idealized models
  - Random Oracle Model, UC-Framework



When arguing post-quantum security, we have to consider quantum adversaries!

Challenge 4: Security models

- When arguing security, we have to simplify -> models
  - IND-CPA, IND-CCA, EUF-CMA, …
- Sometime
  security w
  building b
  - Random
    UC-Fran



Challenging for idealized models!

When arguing post-quantum security, we have to consider quantum adversaries!

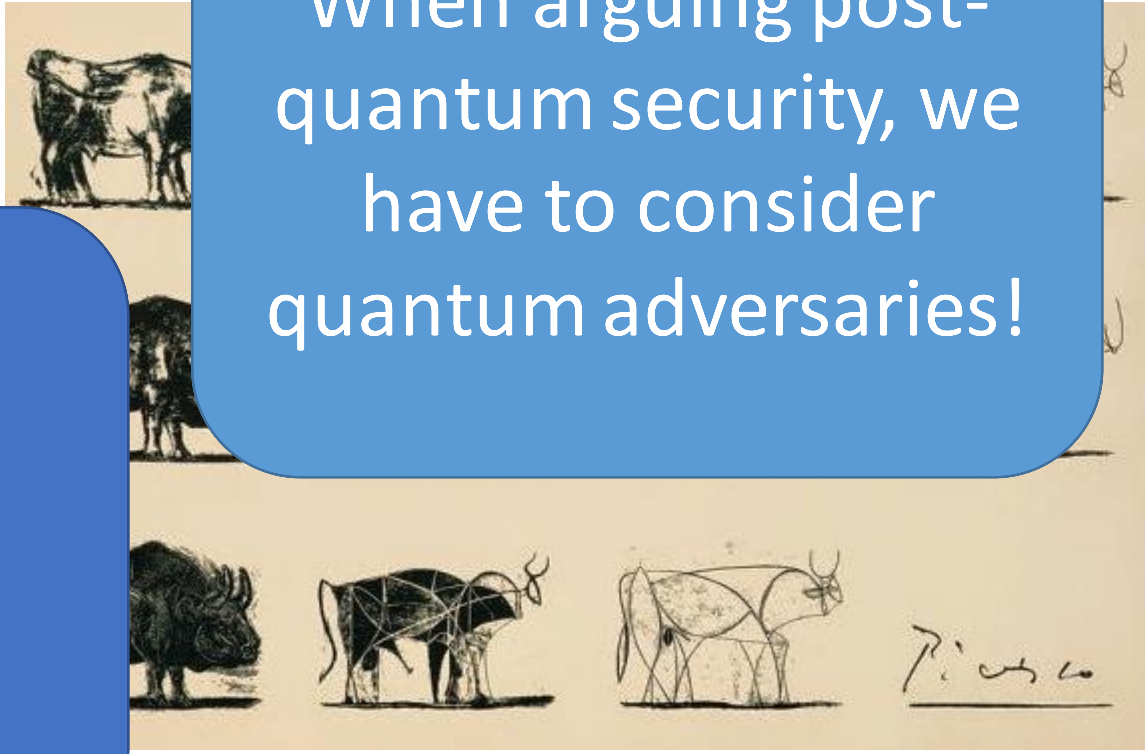
Challenge 4: Security models

PQWireGuard:
Standard Model
Adoption “easy”

odels
CMA, …

- Sometime
security w
building b
  - Random
UC-Fran

Challenging for idealized models!



When arguing post-quantum security, we have to consider quantum adversaries!

Challenge 5: Standardization

Super important!

(and **a lot** of work)

But not much different from before.

Challenge 6: Hybrids

Motivation:

- To achieve compliance
- When using non-conservative schemes

Solutions:

- KEM-Combiners / DSS-Combiners
- Exploiting protocol specifics

Challenge 6: Hybrids

Motivation:

- To achieve compliance
- When using non-conservative schemes

Solutions:

- KEM-Combiners / DSS-Combiners
- Exploiting protocol specifics

PQWireGuard:
Both options work.

Bonus challenge: Complicated proofs & implementations

Number theoretic schemes have a beautiful simplicity…

… PQC schemes don’t.

- Models get more complicated
- Proofs get more complicated
- Implementations get more complicated

Bonus challenge: Complicated proofs & implementations

Number theoretic schemes have a beautiful simplicity…

… PQC schemes don’t.

- Models get more complicated
- Proofs get more complicated
- Implementations get more complicated

How to prevent mistakes?

Bonus challenge: Complicated proofs & implementations

Number theoretic schemes have a beautiful simplicity…

…

d

Formal verification!

- Machine checked proofs
- Compiler with guaranteed security properties

(see e.g., https://formosa-crypto.org/)

FORMOSA CRYPTO

How to prevent mistakes?

Bonus challenge: Complicated proofs & implementations

Number theoretic schemes have a beautiful simplicity…

Formal verification!

- Machine checked proofs
- Compiler with guaranteed security properties

(see e.g., https://formosa-crypto.org/)

How to prevent mistakes?

<u>PQWireGuard:</u>
Machine-checked proof in symbolic model.

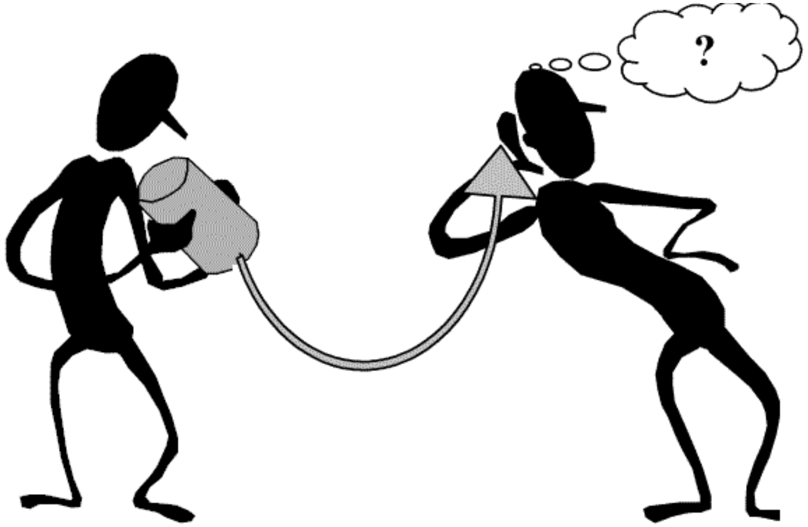
Conclusions

- We are not done with the end of the NIST competition
- We manage to handle the challenges well for “simple” protocols
  - We can even get close to previous performance if we design new protocols with challenges in mind!
- The challenges will get more problematic for advanced protocols
  - Ratcheting? (Signal, WhatsApp, OTR…)
  - Deniable authenticated key exchange? (OTR)
  - Tools involving ZKPs, e.g., group signatures, anonymous credentials?
  - …

Resources

- PQC Spring School (2024): https://pqc-spring-school.nl/
- PQ Summer School (2019): http://www.pqcschool.org/
- NIST PQC Standardization Project: https://csrc.nist.gov/Projects/Post-Quantum-Cryptography

Thank you!
Questions?

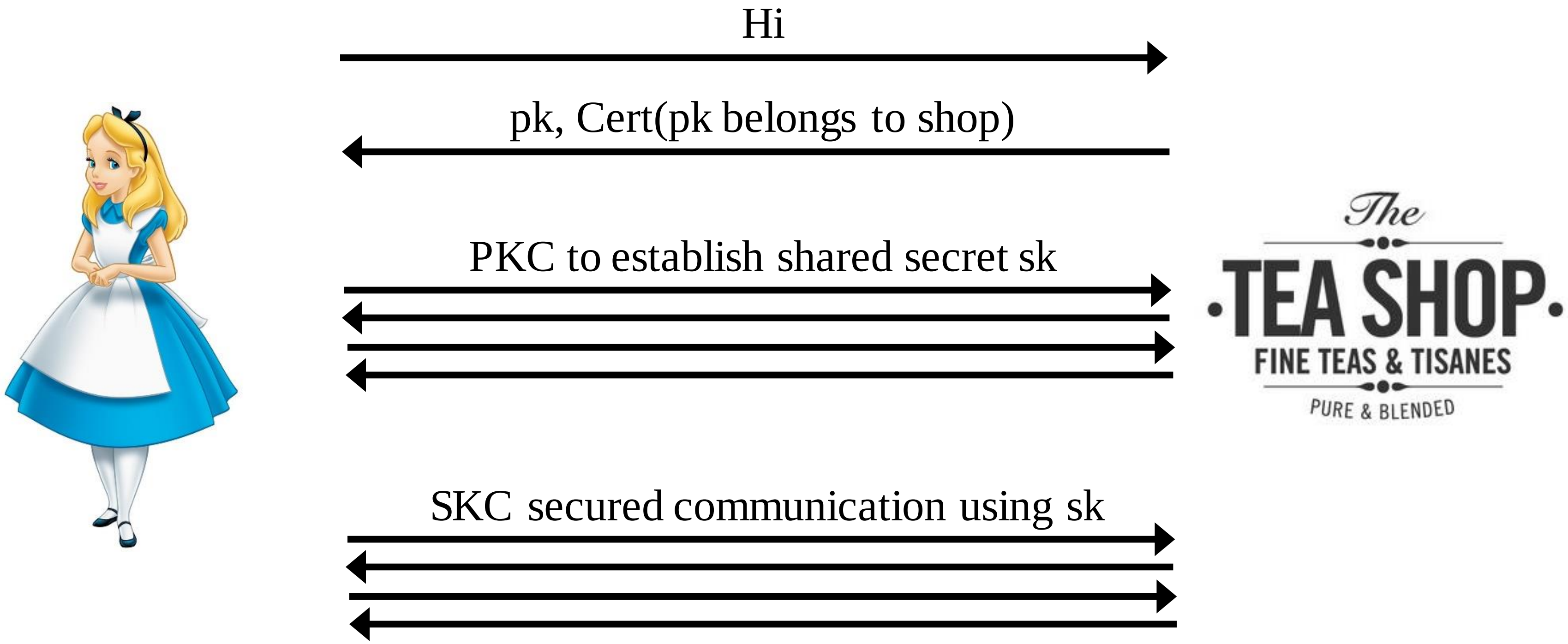


Grover‘s algorithm (1996)

- Quantum computers can search N entry DB in $\Theta(\sqrt{N})$
- Application to symmetric crypto
- Nice: Grover is provably optimal (For random function)
- Double security parameter.

What about QKD?

Recall:
Communication security (simplified)



Hi

pk, Cert(pk belongs to shop)

PKC to establish shared secret sk

SKC secured communication using sk

The problem solved by QKD

Given

- a shared classical secret,
- a physical chan
- compatible QK

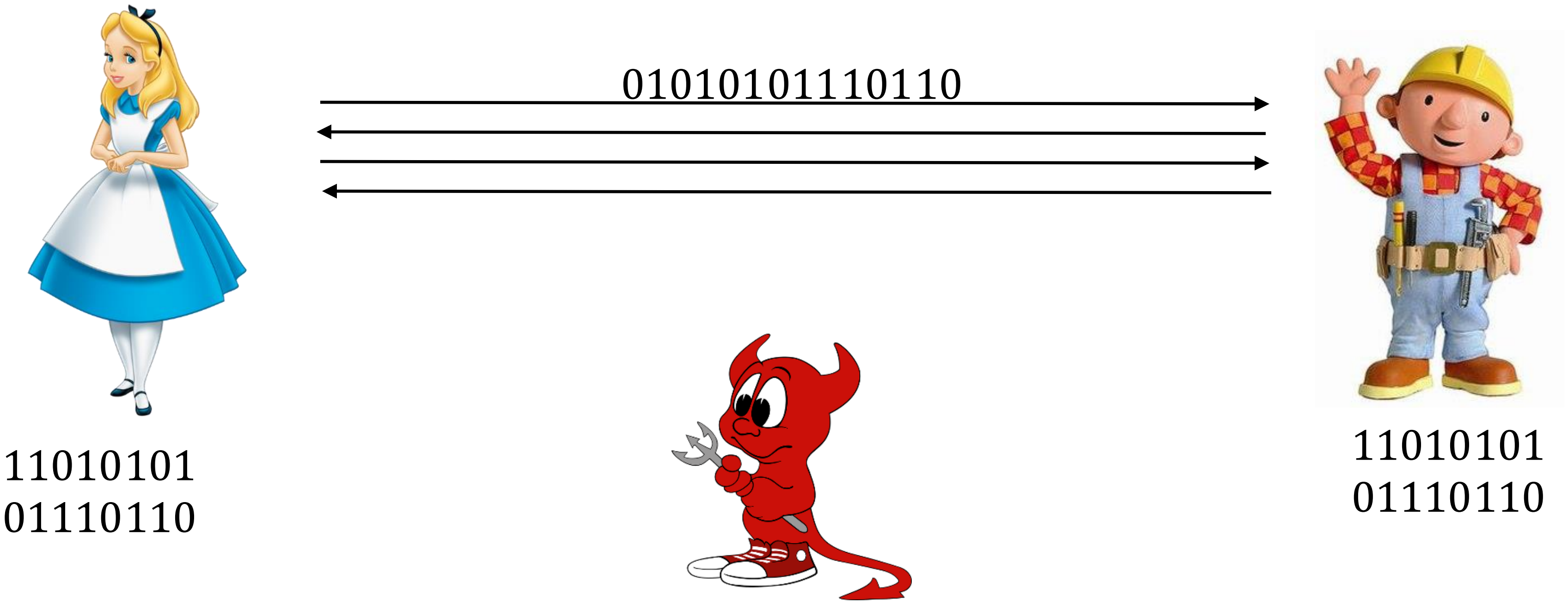
It is possible to

- generate a long

“Key growing”

(≠ “Key establishment“)

QS0: Classical security



0101010111010110

11010101
01110110

11010101
01110110

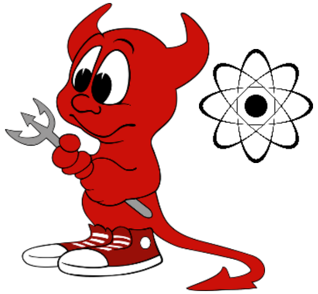
11000101
00010110

QS1: Post-quantum security

010101011110110

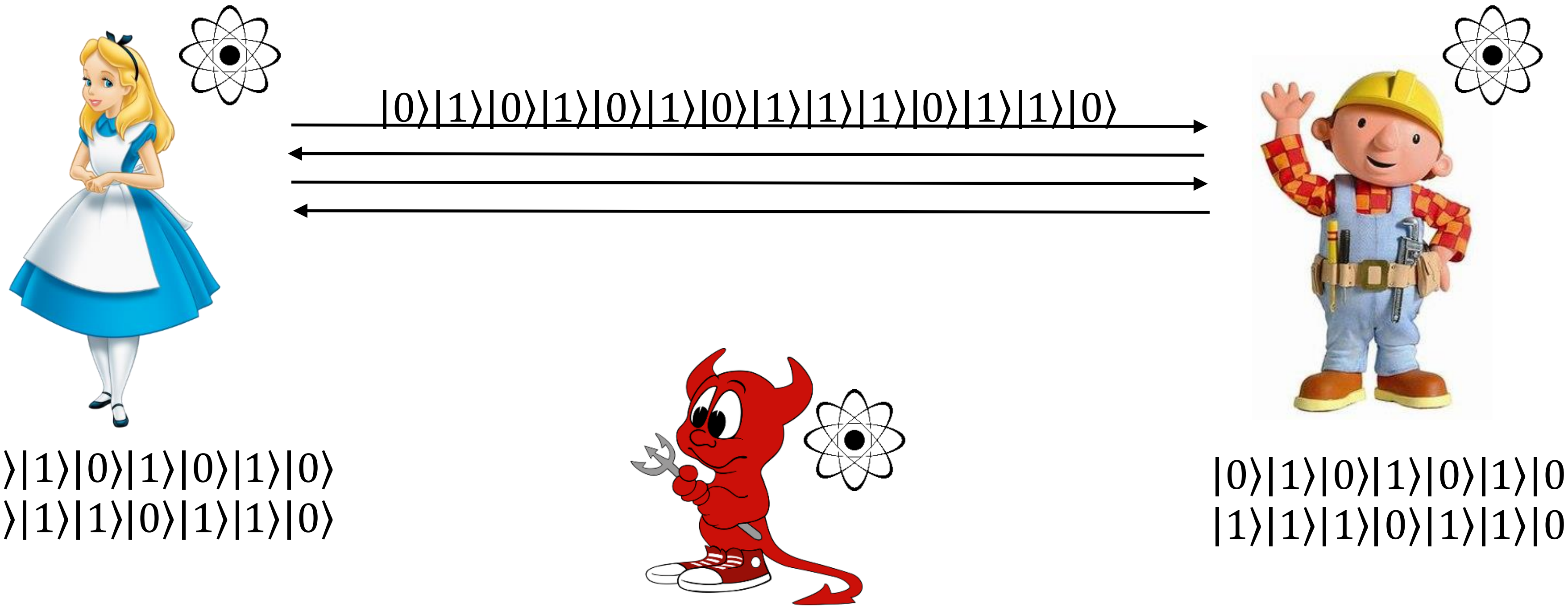
11010101
01110110

11010101
01110110



$|0\rangle|1\rangle|0\rangle|1\rangle|0\rangle|1\rangle|0\rangle$
$|1\rangle|1\rangle|1\rangle|0\rangle|1\rangle|1\rangle|0\rangle$

QS2: Quantum security



|0⟩|1⟩|0⟩|1⟩|0⟩|1⟩|0⟩|1⟩|1⟩|1⟩|0⟩|1⟩|1⟩|0⟩

|0⟩|1⟩|0⟩|1⟩|0⟩|1⟩|0⟩
|1⟩|1⟩|1⟩|0⟩|1⟩|1⟩|0⟩

|0⟩|1⟩|0⟩|1⟩|0⟩|1⟩|0⟩
|1⟩|1⟩|1⟩|0⟩|1⟩|1⟩|0⟩

|0⟩|1⟩|0⟩|1⟩|0⟩|1⟩|0⟩
|1⟩|1⟩|1⟩|0⟩|1⟩|1⟩|0⟩

For practical applications we care about QS1