

Cryptographic Reverse Firewalls

Yevgeniy Dodis (NYU)

Ilya Mironov (Google)

Noah Stephens-Davidowitz (NYU)

Act I: Cryptography in Crisis

Classical Crypto

Classical Crypto



Classical Crypto



Classical Crypto

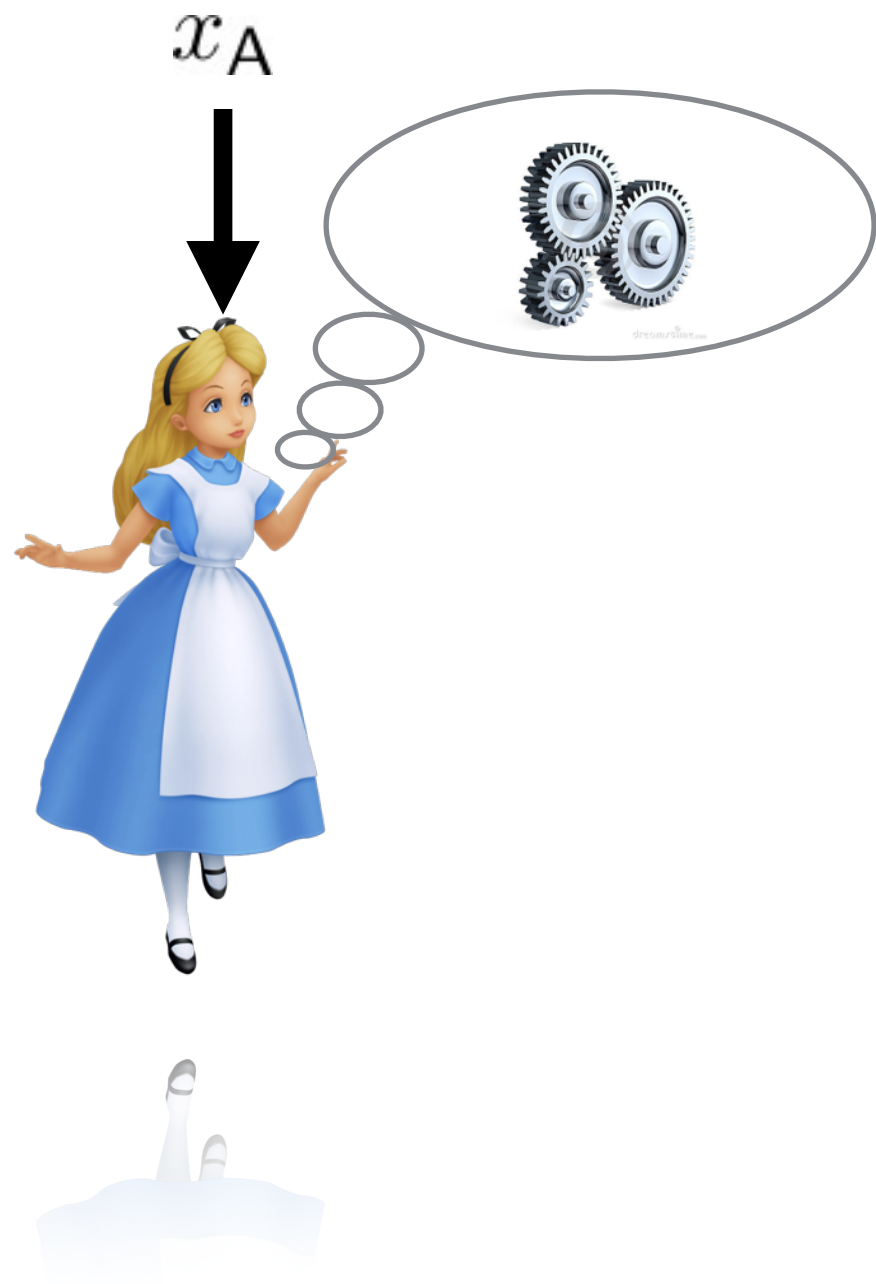
x_A



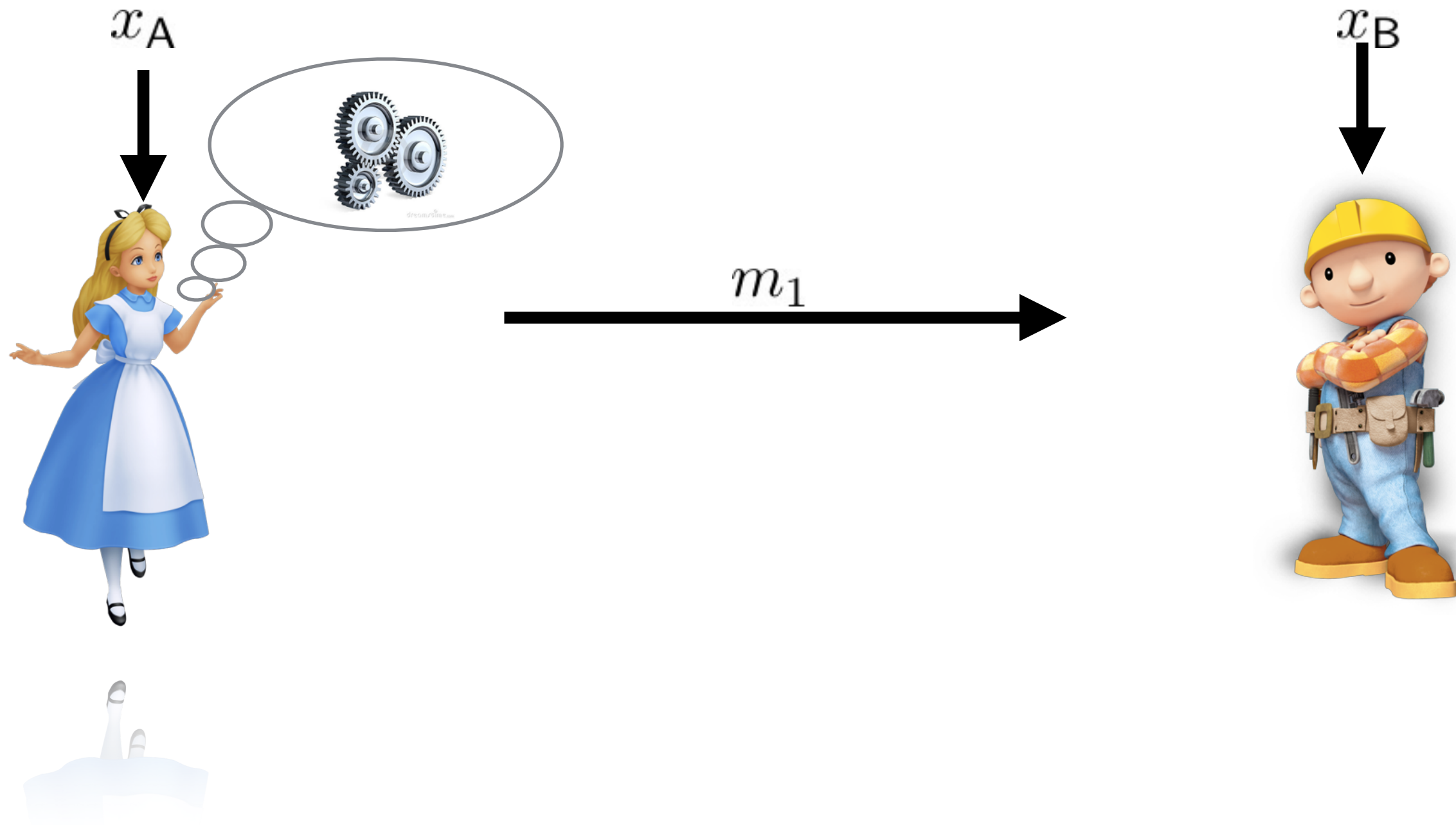
x_B



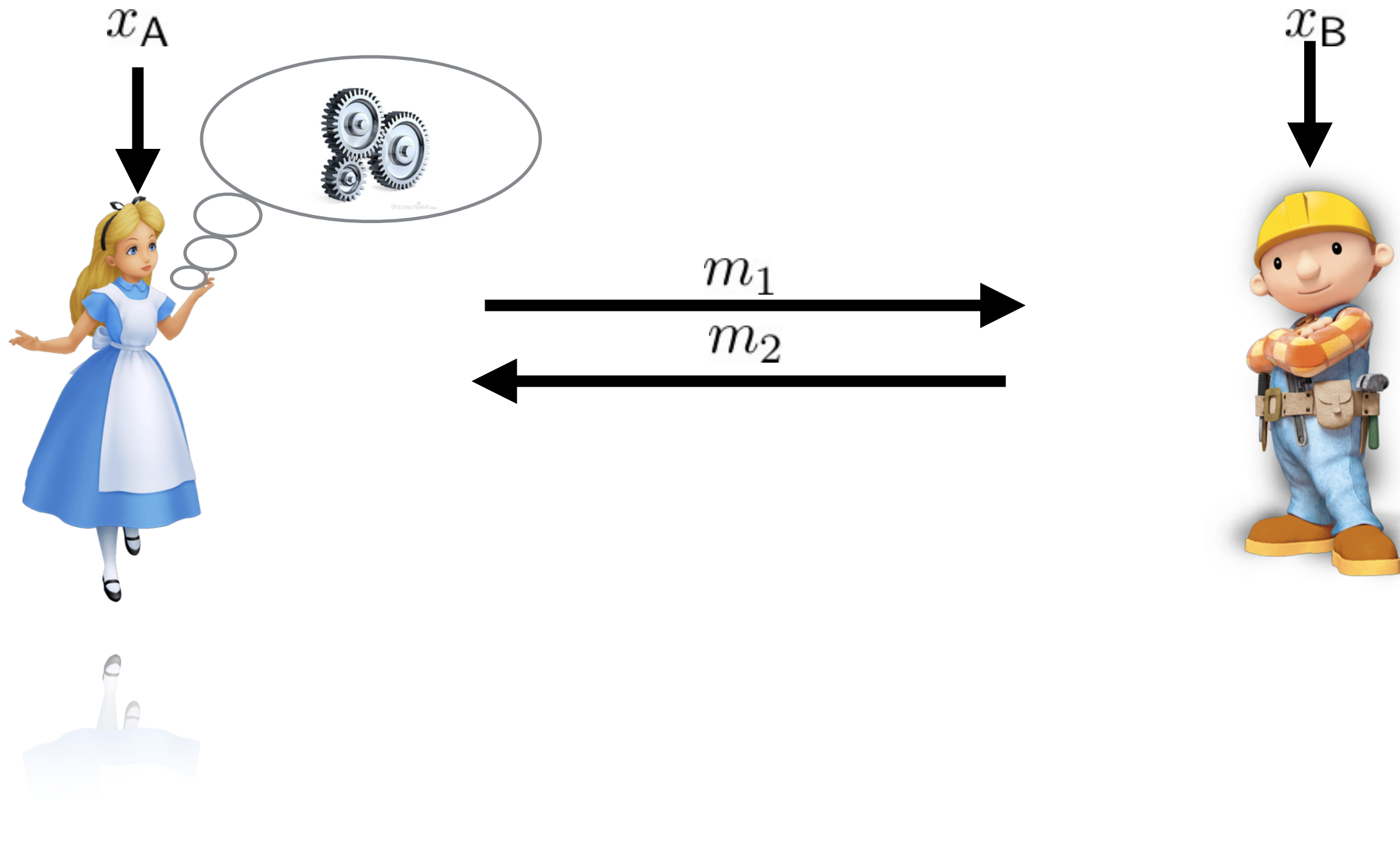
Classical Crypto



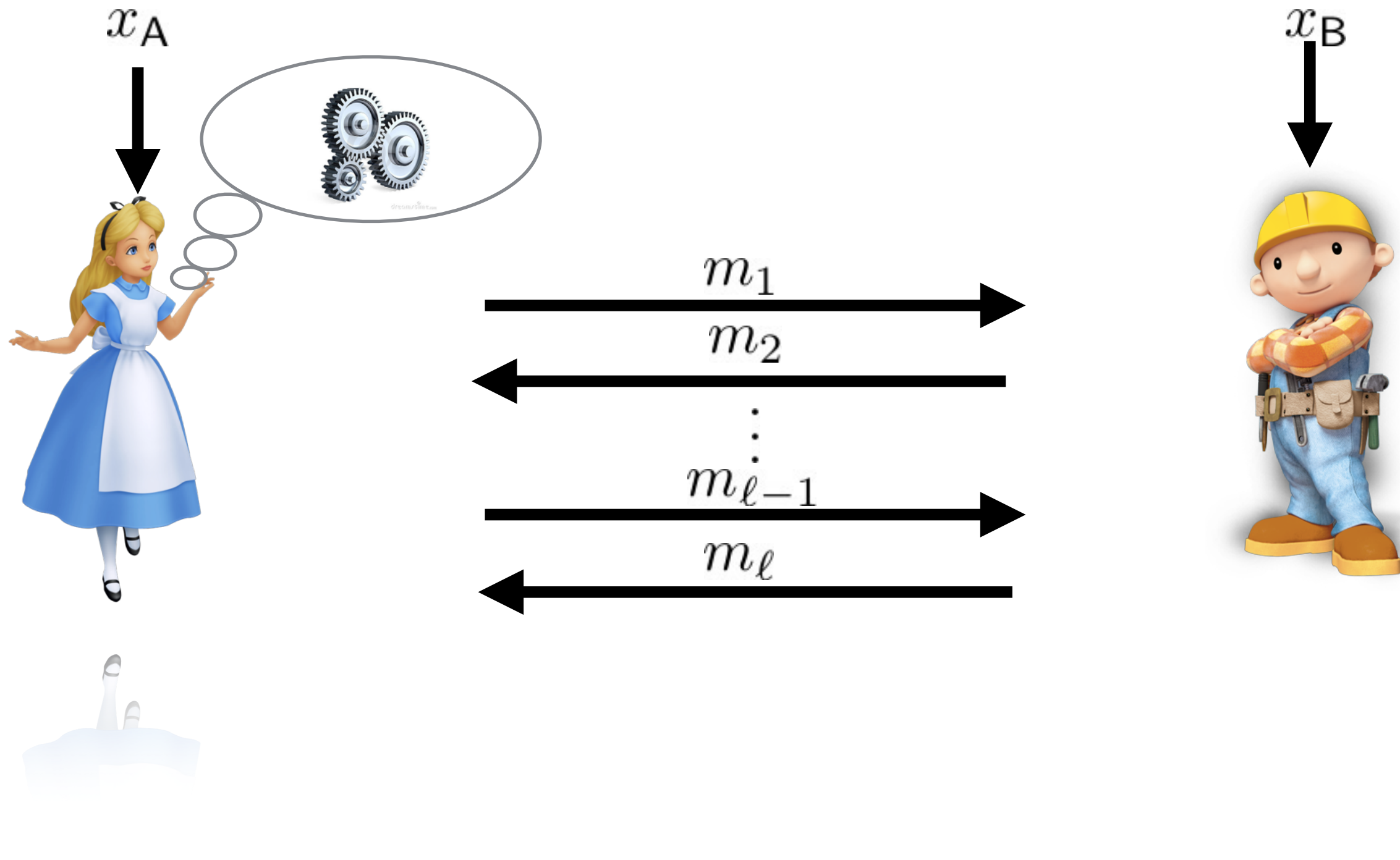
Classical Crypto



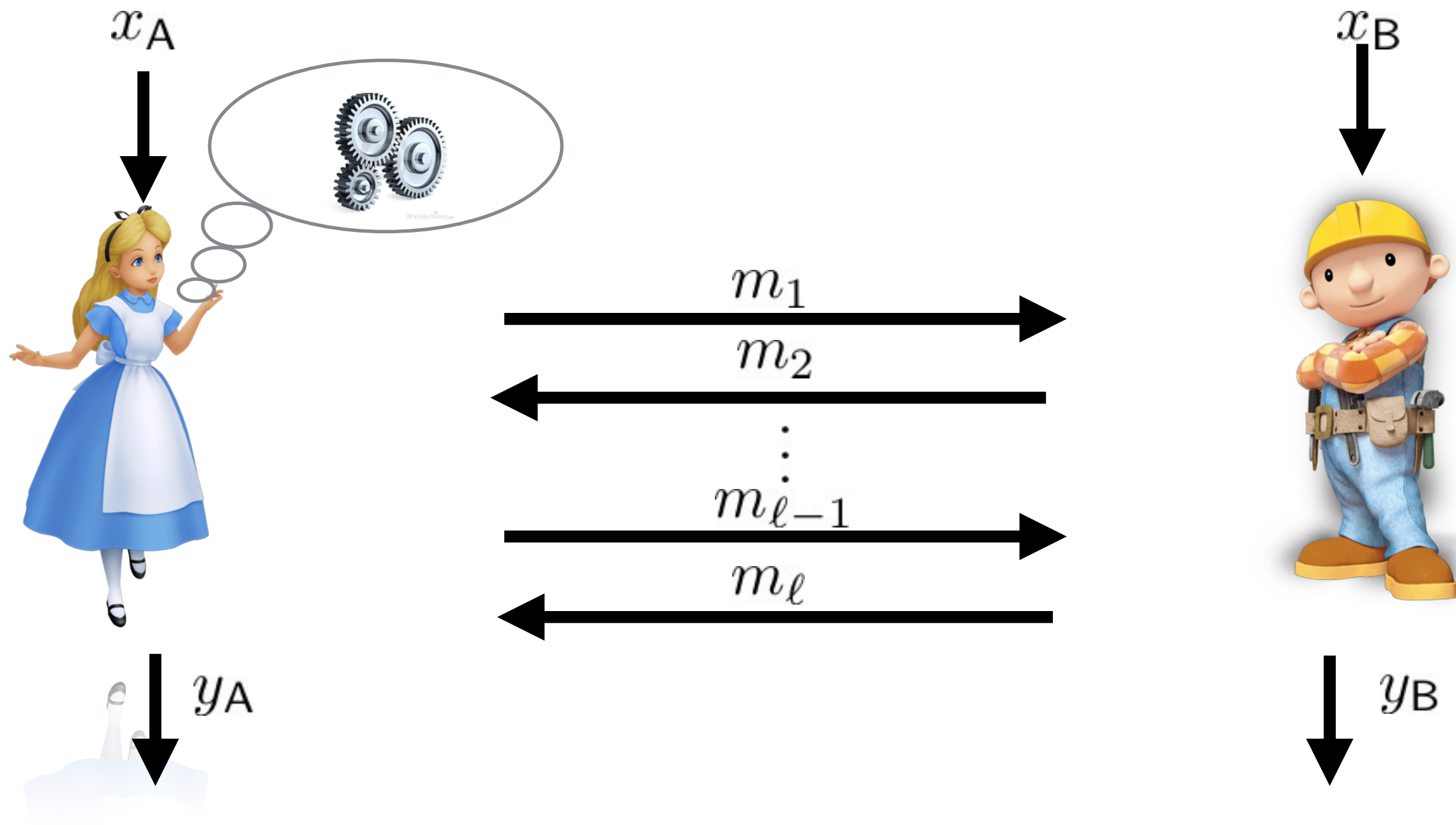
Classical Crypto



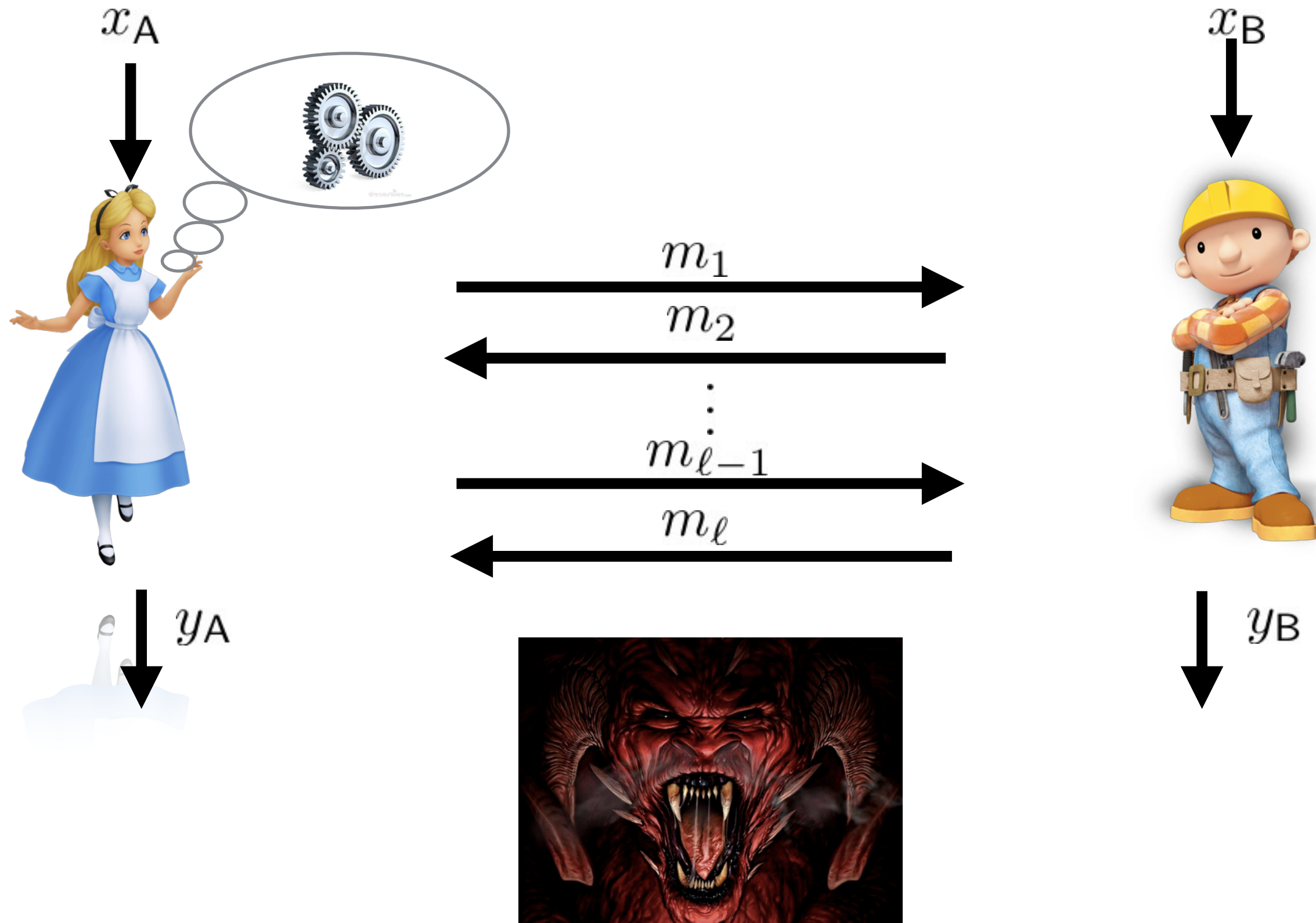
Classical Crypto



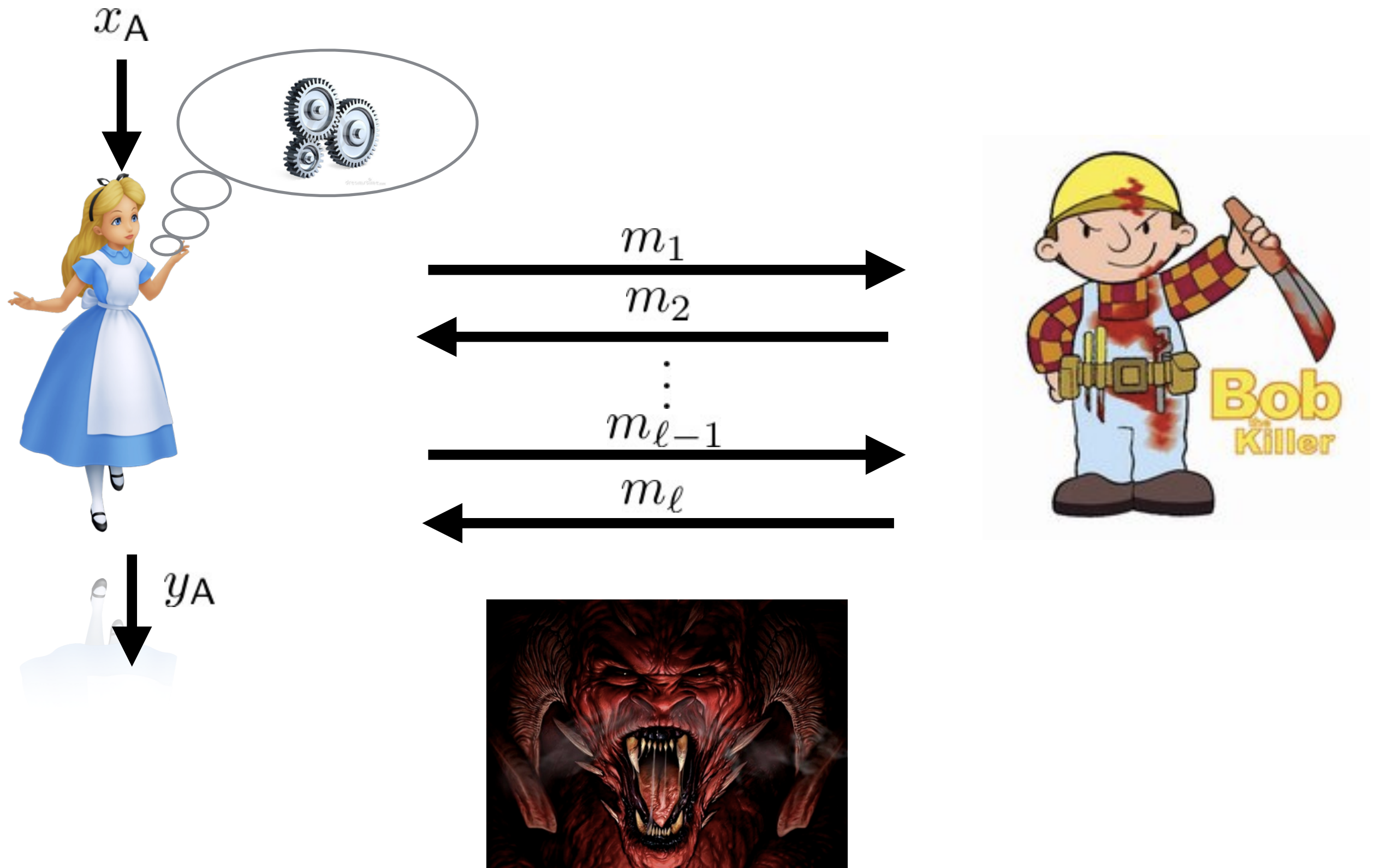
Classical Crypto



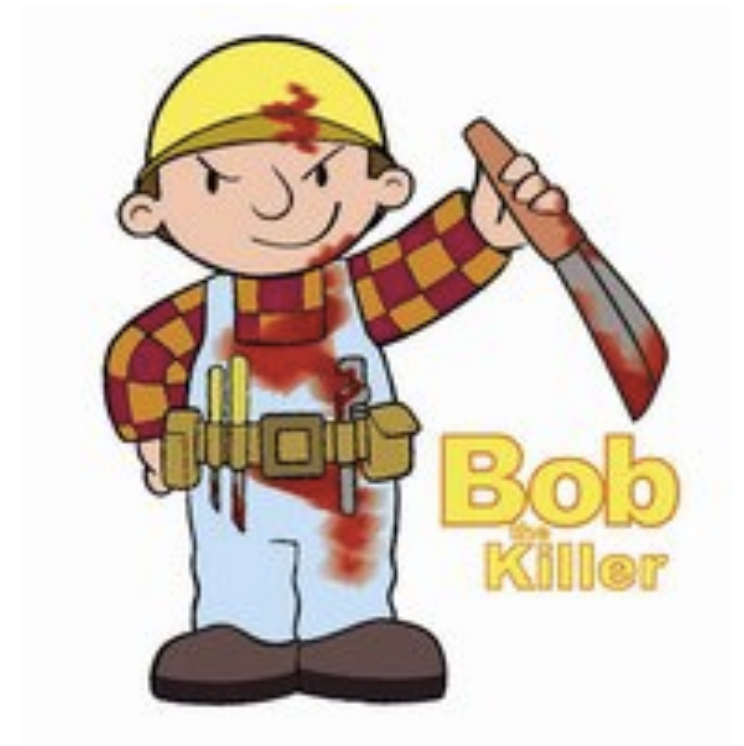
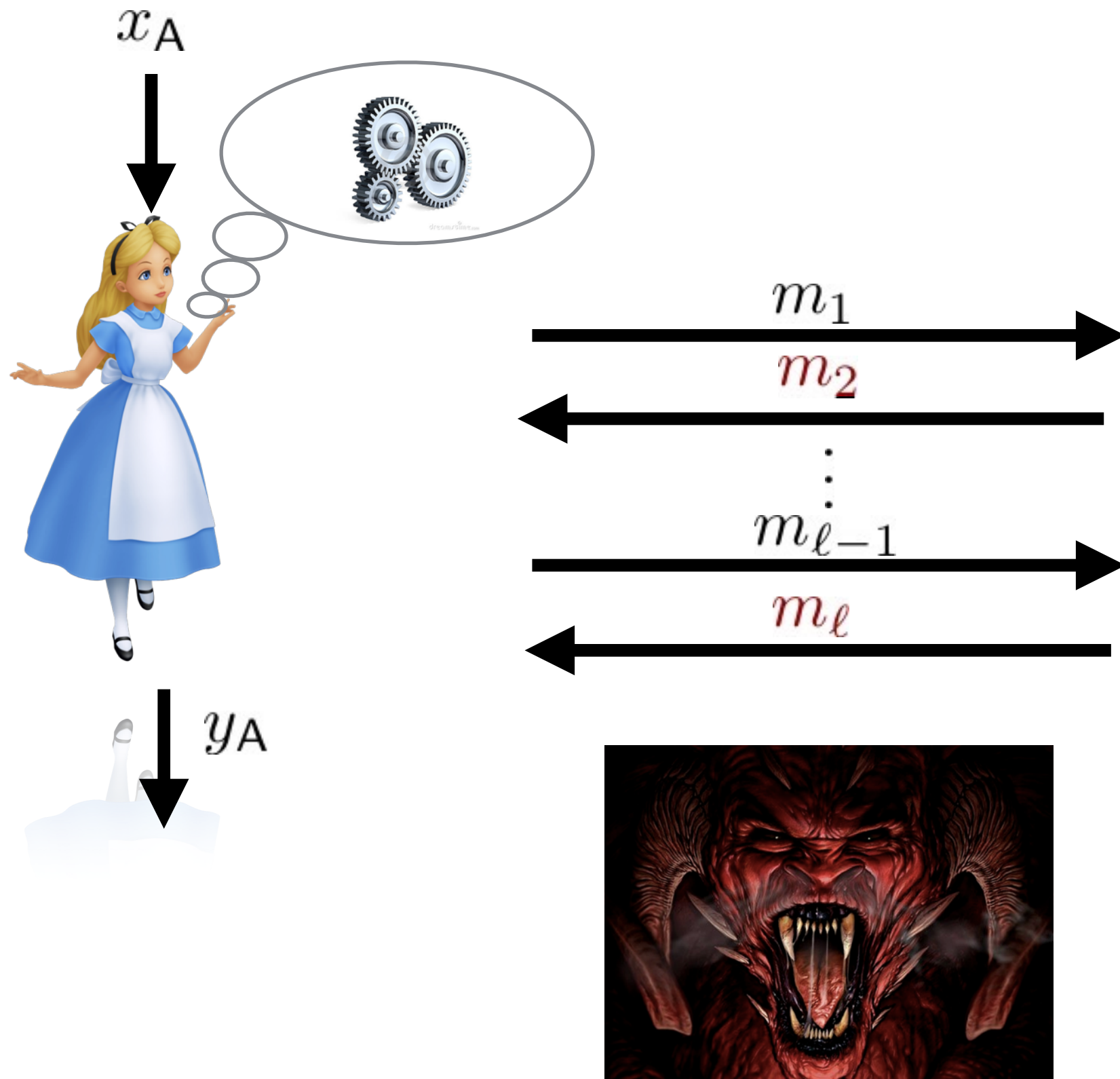
Classical Crypto



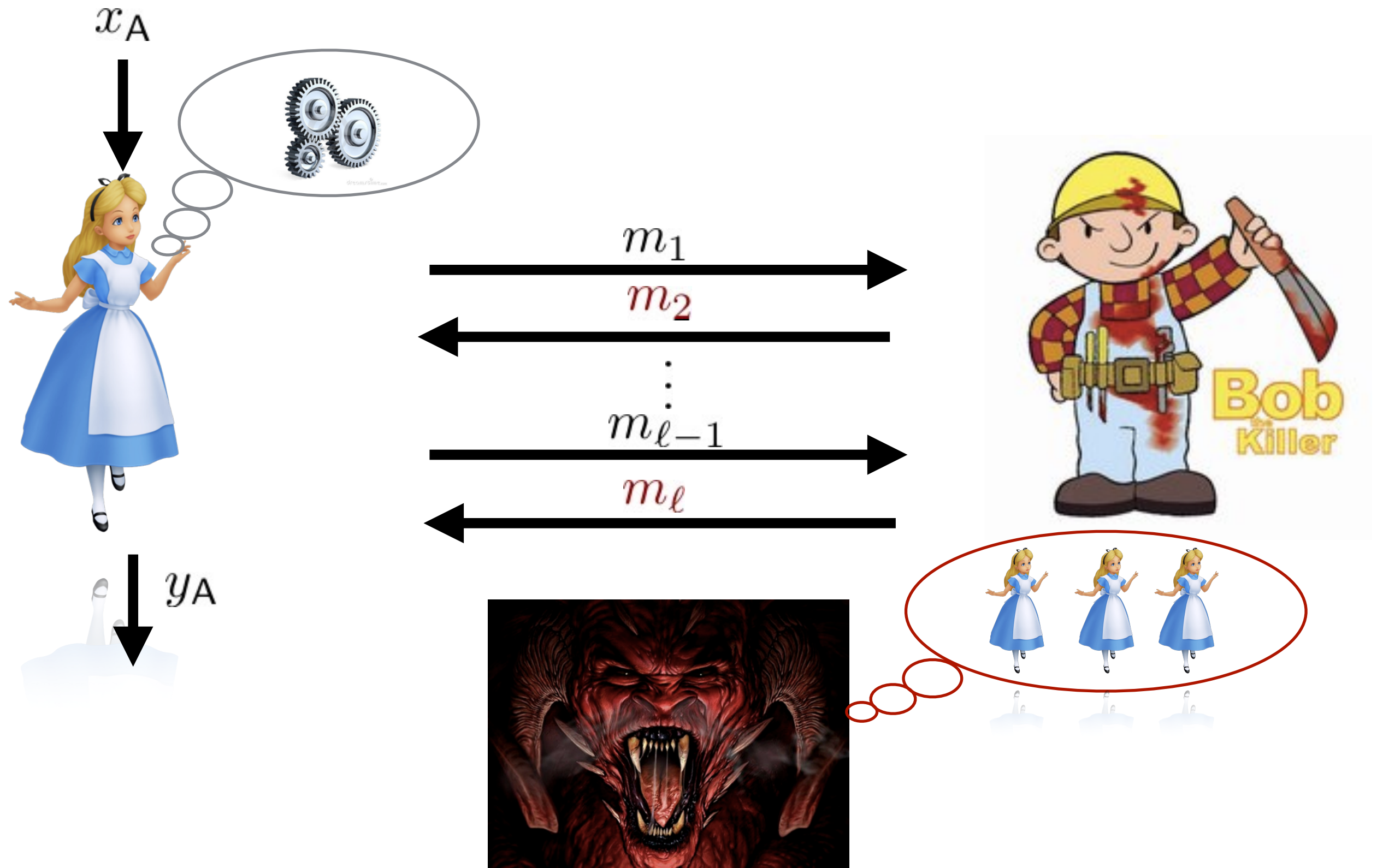
Classical Crypto



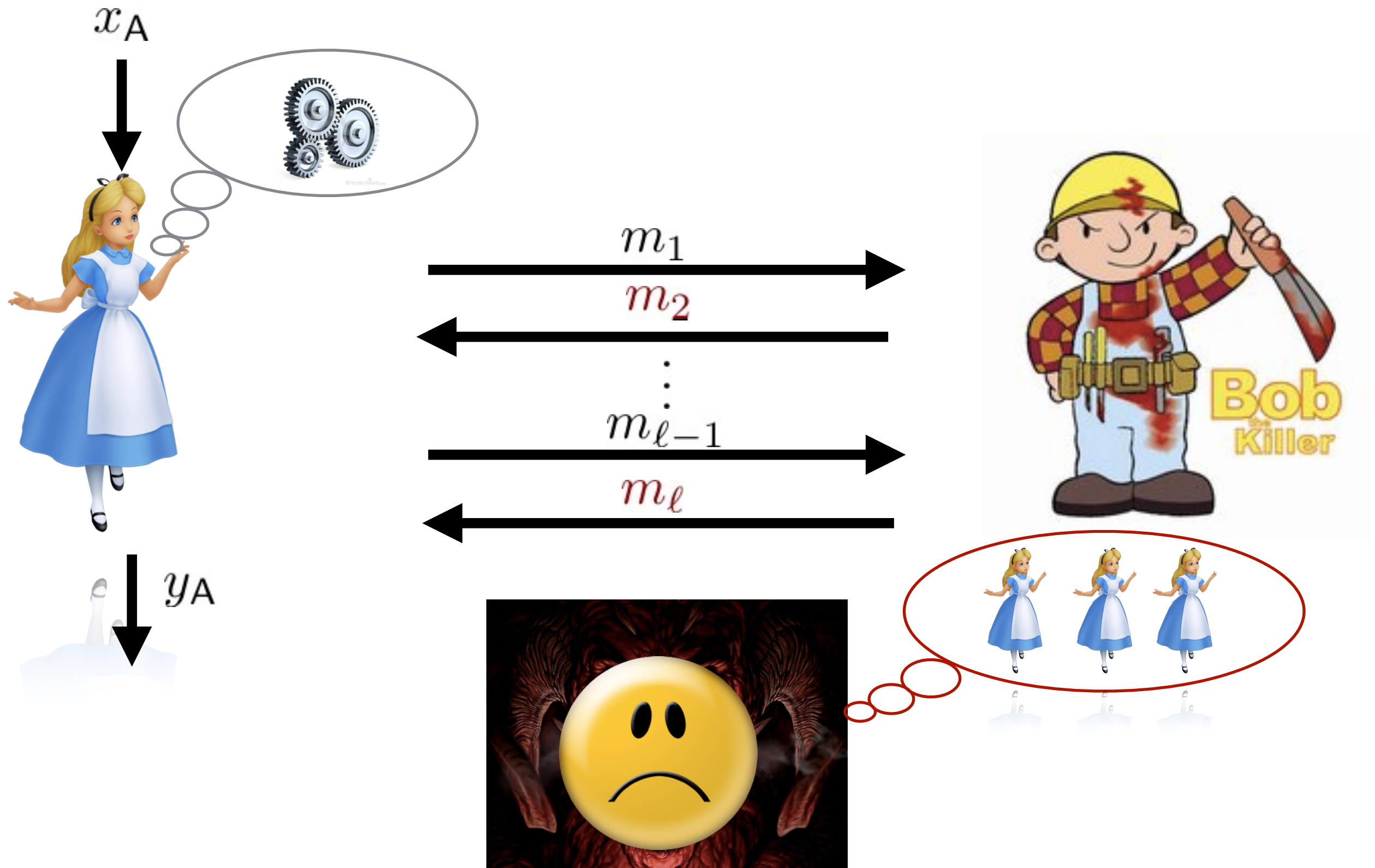
Classical Crypto



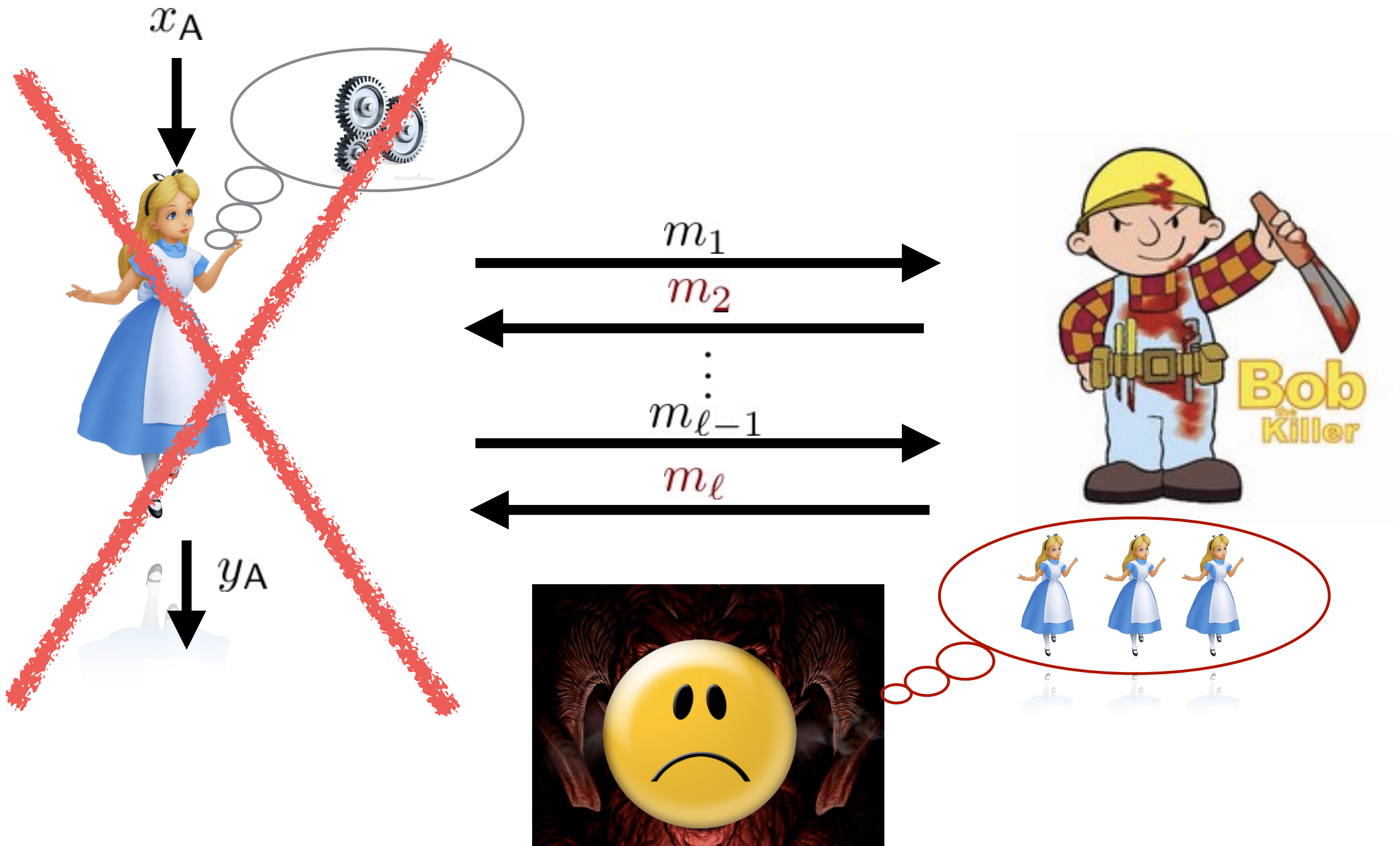
Classical Crypto



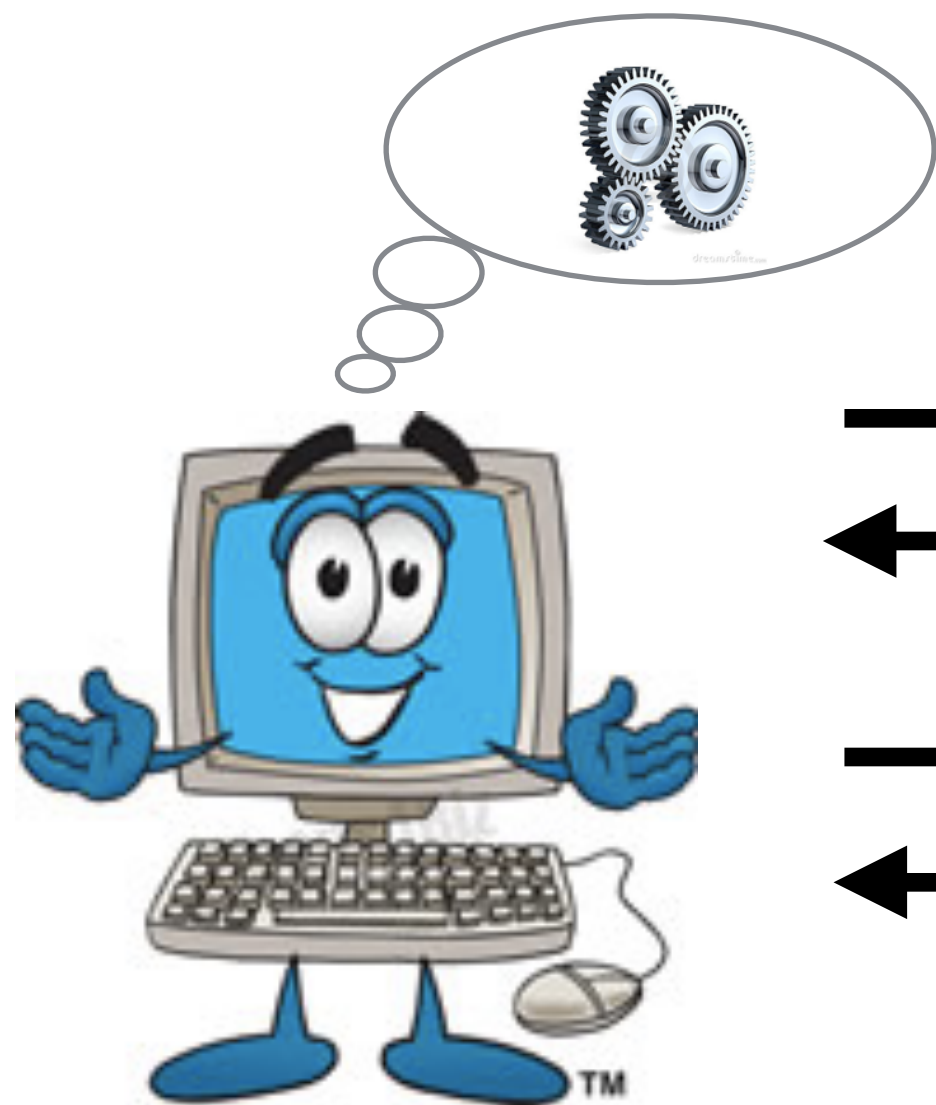
Classical Crypto



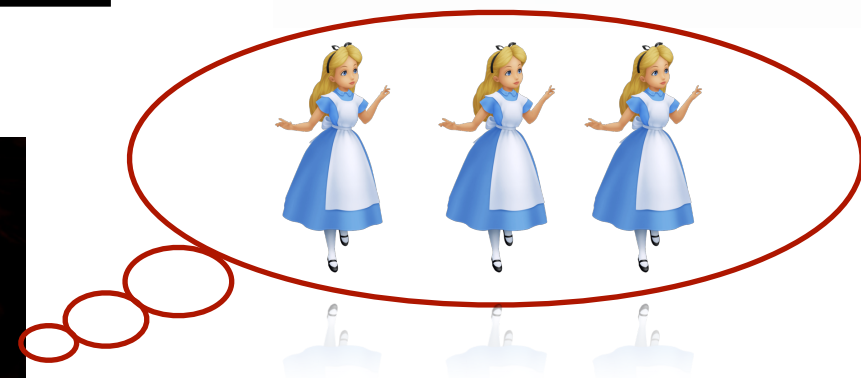
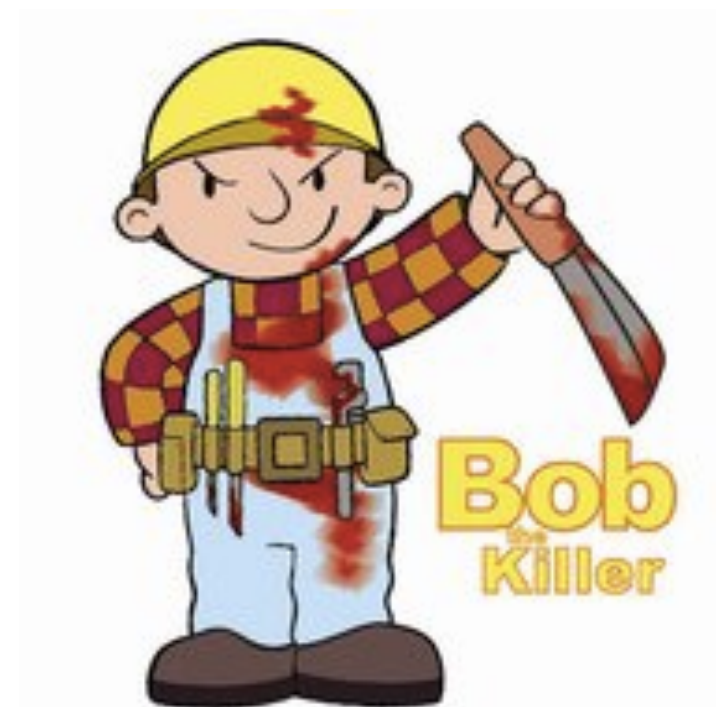
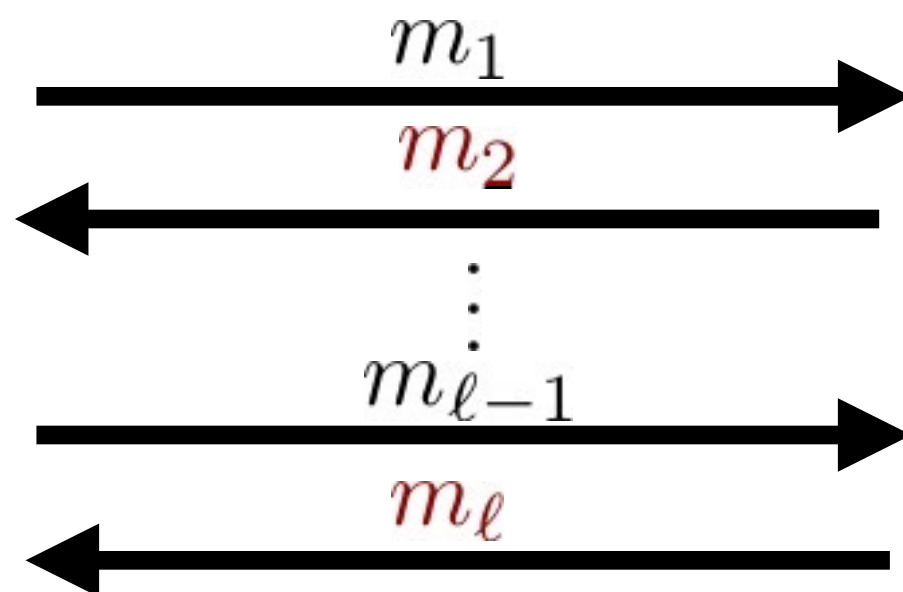
Classical Crypto



Classical Crypto



Alice's Trusty
Computer!



Should Alice Trust Her Computer?

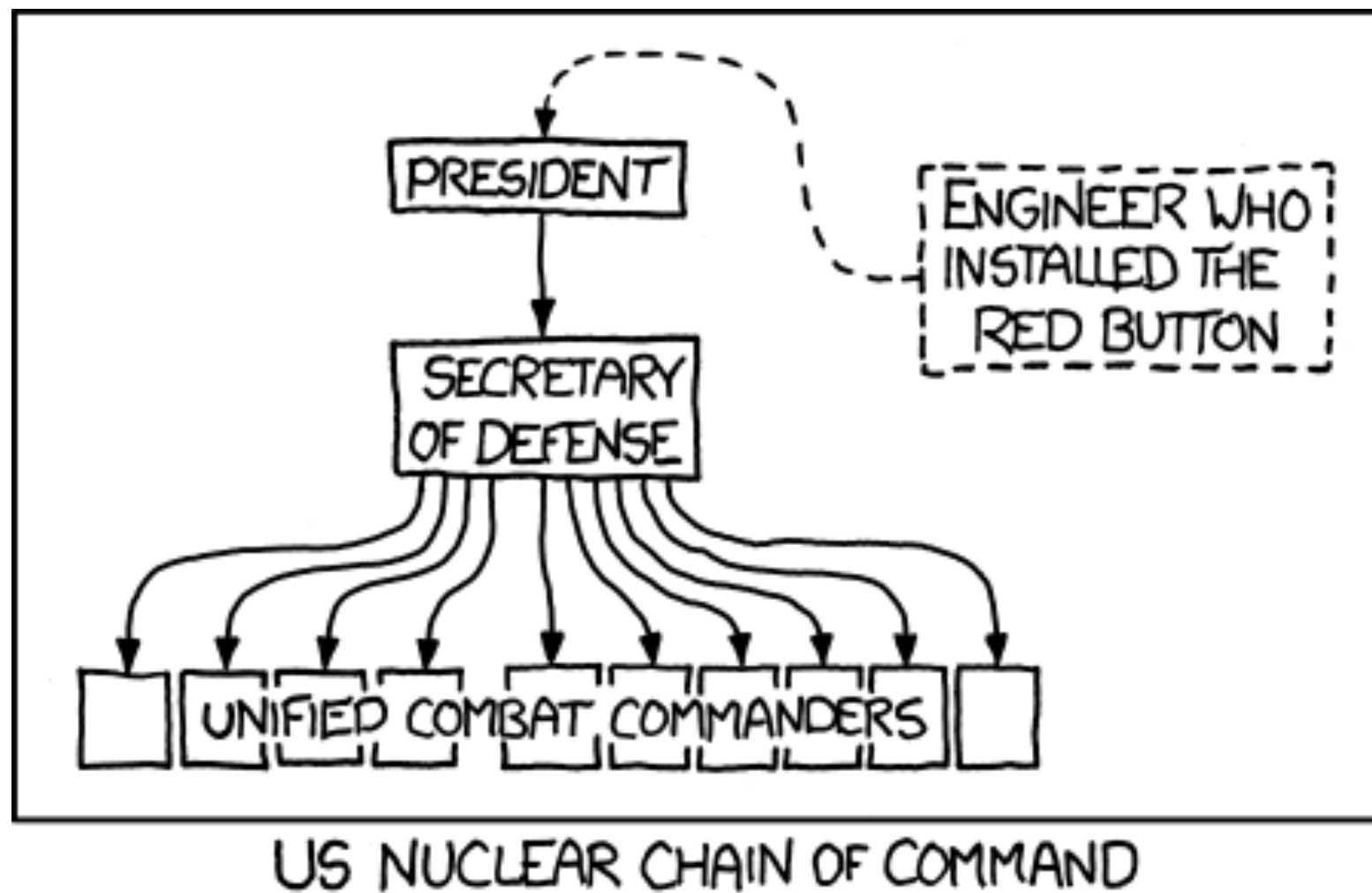
Should Alice Trust Her Computer?



Should Alice Trust Her Computer?



Should Alice Trust Her Computer?



(xkcd.com)

Widespread Deliberate Corruption of Hardware and Software



Widespread Deliberate Corruption of Hardware and Software

“The SIGINT Enabling Project [\$250M/year program] actively engages the US and foreign IT industries to covertly influence and/or overtly leverage their commercial products’ designs. These design changes make the systems in question exploitable ... with foreknowledge of the modification. To the consumer and other adversaries, however, the systems’ security remain intact.”

Excerpt from the N.S.A.’s 2013 budget request

The New York *Times*, September 5, 2013

Cryptographers have agreed not to accept this

The membership of the IACR repudiates mass surveillance and the undermining of cryptographic solutions and standards. Population-wide surveillance threatens democracy and human dignity. We call for expediting research and deployment of effective techniques to protect personal privacy against governmental and corporate overreach.

IACR Copenhagen Resolution
Eurocrypt 2014, Copenhagen

Widespread, (Apparently) Accidental Bugs in Cryptographic Software

Widespread, (Apparently) Accidental Bugs in Cryptographic Software



Widespread, (Apparently) Accidental Bugs in Cryptographic Software



Widespread, (Apparently) Accidental Bugs in Cryptographic Software

A screenshot of a code editor window titled 'sslKeyExchange.c'. The code is in C and shows several 'if' statements checking for errors. The line 'goto fail;' is highlighted in blue, indicating a bug or a specific point of interest in the code.

```
if ((err = ReadyHash(&SSLHashSHA1, &hashCtx))
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &cli
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &ser
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &sig
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &sig
    goto fail;
if ((err = SSLHashSHA1.final(&hashCtx, &hash
    goto fail;

err = sslRawVerify(ctx,
    ctx->peerPubKey,
```

Widespread, (Apparently) Accidental Bugs in Cryptographic Software



Debian Security Advisory

DSA-1571-1 openssl -- predictable random number generator

```
if ((err = ReadyHash(&SSLHashSHA1, &hashCtx))
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &cli
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &ser
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &sig
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &sig
    goto fail;
if ((err = SSLHashSHA1.final(&hashCtx, &hash
    goto fail;

err = sslRawVerify(ctx,
    ctx->peerPubKey,
```



Widespread, (Apparently) Accidental Bugs in Cryptographic Software



Debian Security Advisory

DSA-1571-1 openssl -- predictable random number generator

```
if ((err = ReadyHash(&SSLHashSHA1, &hashCtx))  
    goto fail;  
if ((err = SSLHashSHA1.update(&hashCtx, &cli  
    IA1.update(&hashCtx, &ser  
    IA1.update(&hashCtx, &sig  
    IA1.final(&hashCtx, &hash  
ify(ctx,  
ctx->peerPubKey,
```



Widespread, (Apparently) Accidental Bugs in Cryptographic Software



Debian Security Advisory

DSA-1571-1 openssl -- predictable random number generator

```
if ((err = ReadyHash(&SSLHashSHA1, &hashCtx))
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &cli
    SHA1.update(&hashCtx, &ser
    SHA1.update(&hashCtx, &sig
    SHA1.final(&hashCtx, &hash

if (ctx,
    ctx->peerPubKey,
```

Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices

Nadia Heninger^{†*}

Zakir Durumeric^{‡*}

Eric Wustrow[‡]

J. Alex Halderman[‡]

[†] *University of California, San Diego*
nadiiah@cs.ucsd.edu

[‡] *The University of Michigan*
{zakir, ewust, jhalderm}@umich.edu

Widespread, (Apparently) Accidental Bugs in Cryptographic Software

LOGJAM^{SSL} ATTACK

Warning! Your web browser is vulnerable to Logjam and can be tricked into using weak encryption.

OpenSSL Security Advisory

71-1 openssl -- predictable random number generator

```
if ((err = ReadyHash(&SSLHashSHA1, &hashCtx))  
    goto fail;  
if ((err = SSLHashSHA1.update(&hashCtx, &cli
```

```
SHA1.update(&hashCtx, &ser
```

```
SHA1.update(&hashCtx, &sig
```

```
SHA1.final(&hashCtx, &hash
```

```
ify(ctx,  
ctx->peerPubKey,
```

Authors: Detection of Widespread Weak Keys in Network Devices

Nadia Heninger^{†*}

Zakir Durumeric^{‡*}

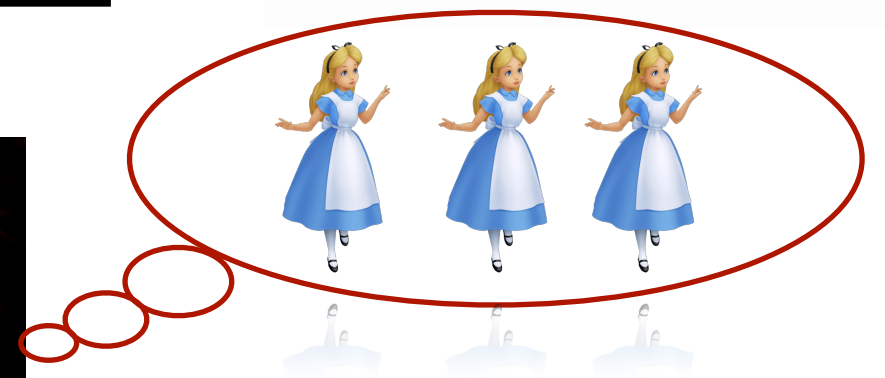
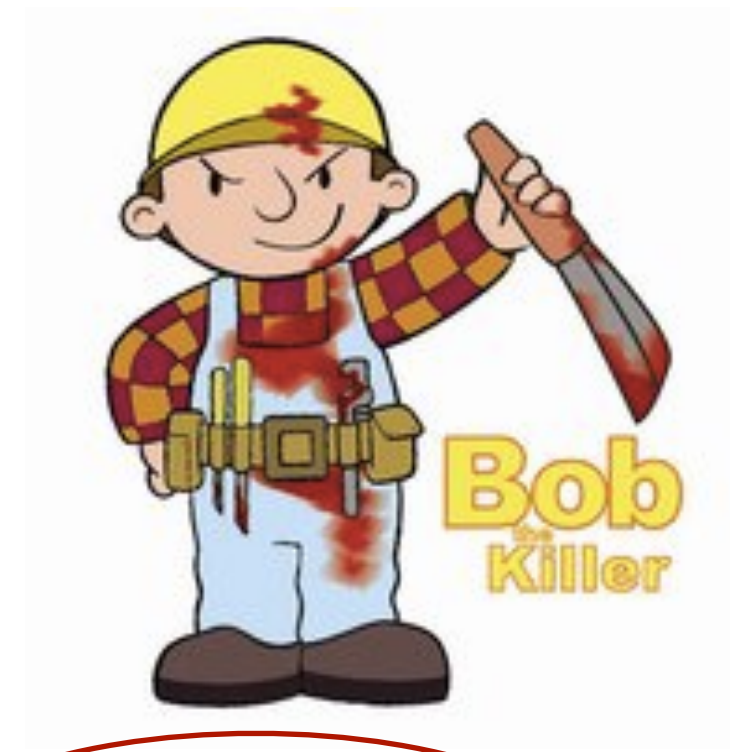
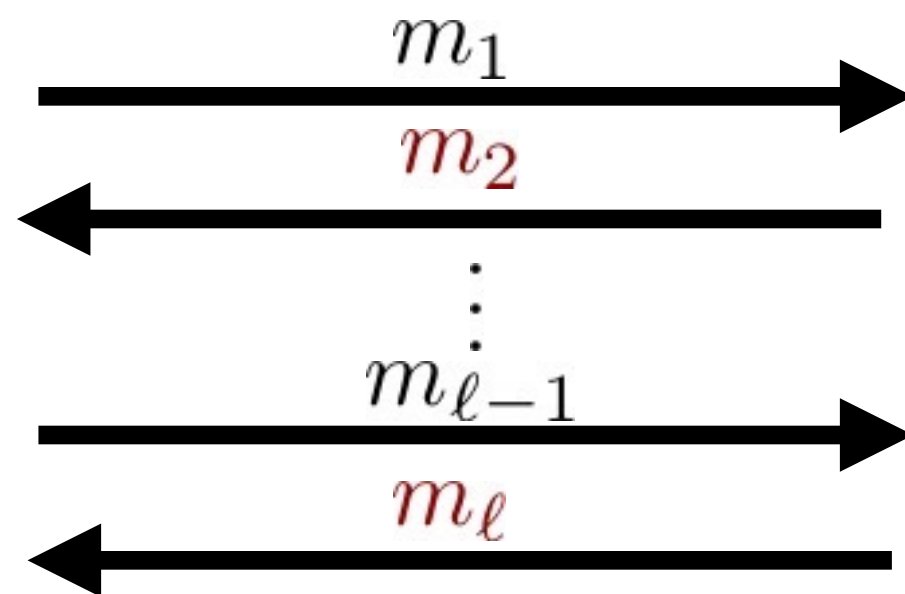
Eric Wustrow[‡]

J. Alex Halderman[‡]

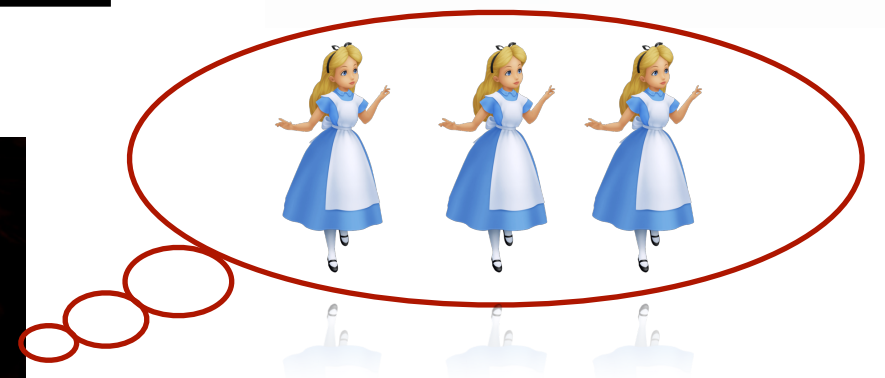
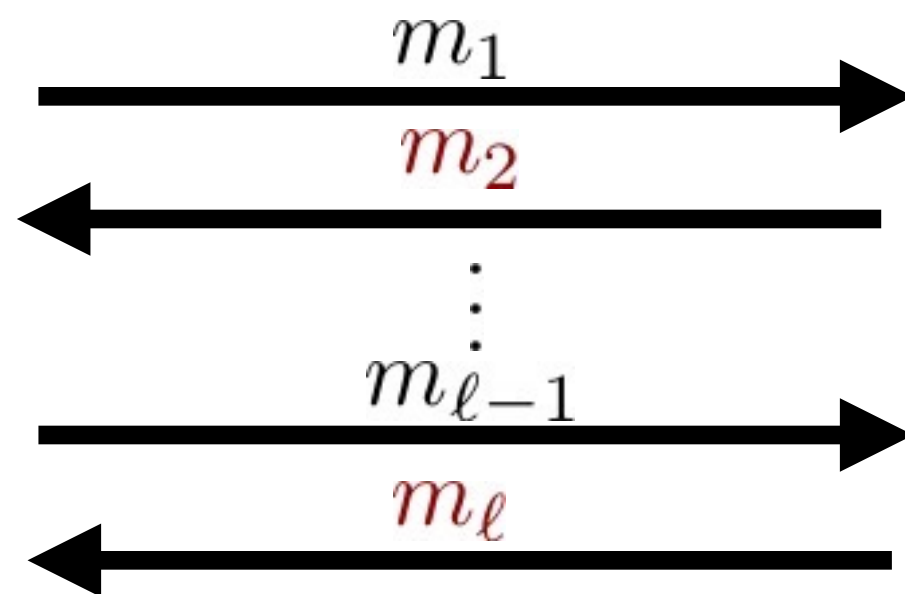
[†] *University of California, San Diego*
nadiah@cs.ucsd.edu

[‡] *The University of Michigan*
{zakir, ewust, jhalderm}@umich.edu

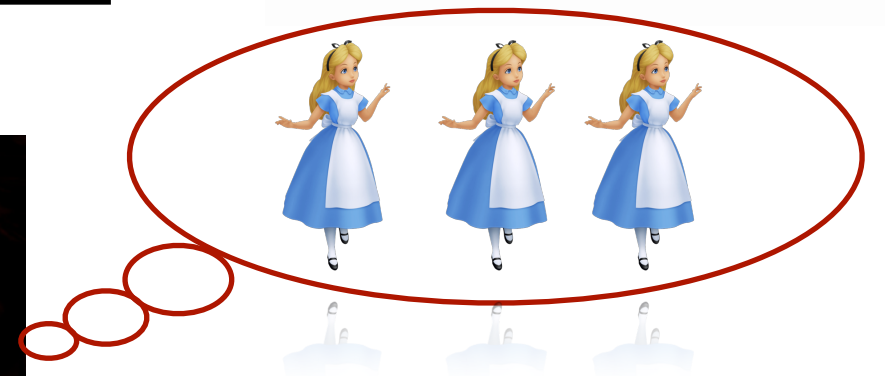
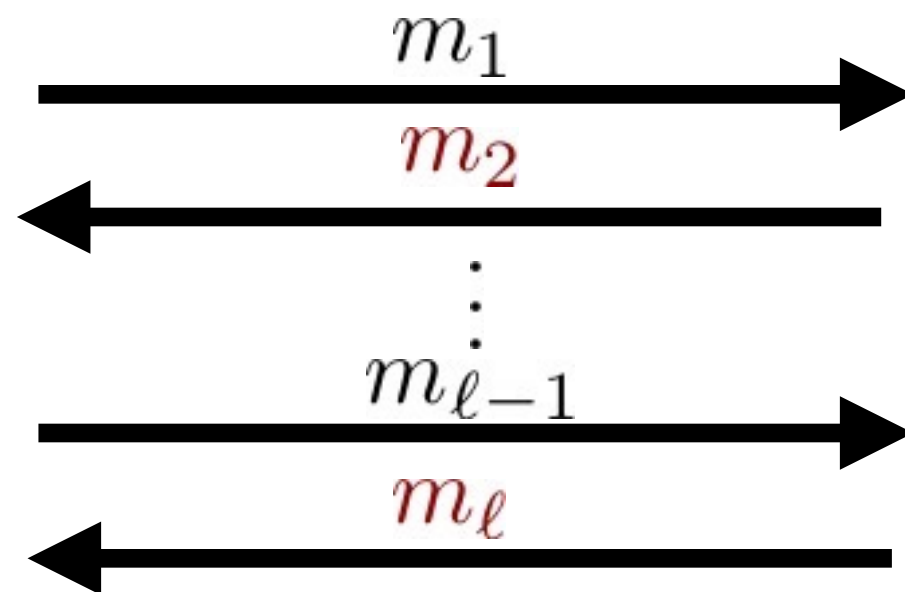
Crypto in the Real World?



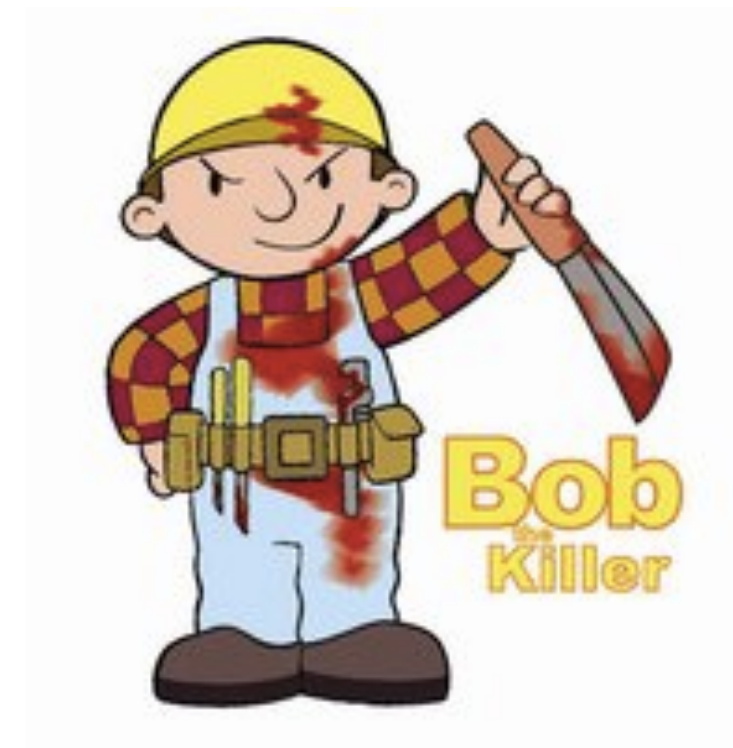
Crypto in the Real World?



Crypto in the Real World?



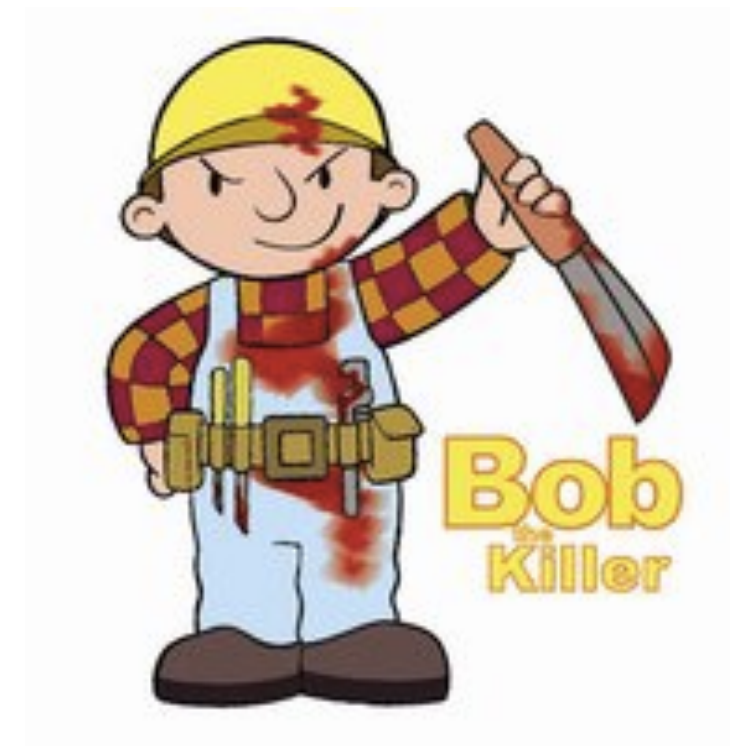
Crypto in the Real World?



Crypto in the Real World?



iHEARTdogs&CAT\$



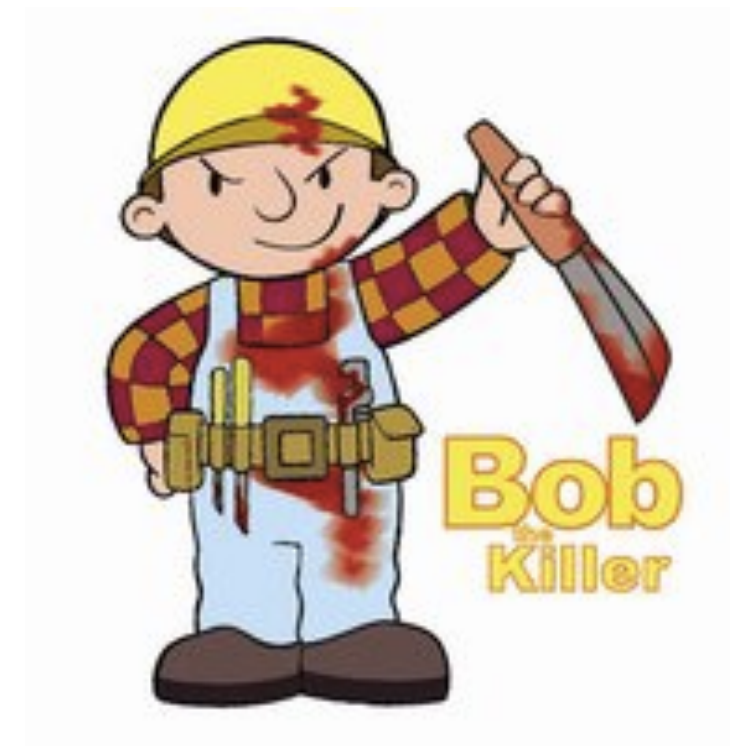
Crypto in the Real World?



iHEARTdogs&CAT\$



4117-8289-1856



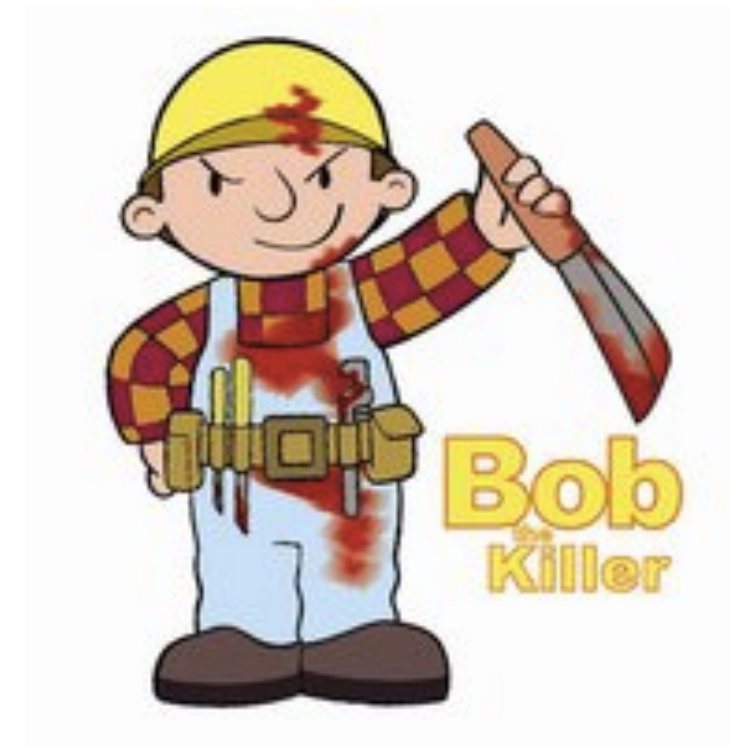
Crypto in the Real World?



iHEARTdogs&CAT\$



4117-8289-1856



Can we possibly do crypto
on a compromised machine?

Prior Work

Prior Work

- Subliminal channels

Prior Work

- Subliminal channels
 - Simmons 1984, ...

Prior Work

- Subliminal channels
 - Simmons 1984, ...
- Divertible protocols

Prior Work

- Subliminal channels
 - Simmons 1984, ...
- Divertible protocols
 - Blaze, Bleumer, Strauss 1998

Prior Work

- Subliminal channels
 - Simmons 1984, ...
- Divertible protocols
 - Blaze, Bleumer, Strauss 1998
 - Limited security against limited forms of corruption

Prior Work

- Subliminal channels
 - Simmons 1984, ...
- Divertible protocols
 - Blaze, Bleumer, Strauss 1998
 - Limited security against limited forms of corruption
 - Only synchronous protocols

Prior Work

- Subliminal channels
 - Simmons 1984, ...
- Divertible protocols
 - Blaze, Bleumer, Strauss 1998
 - Limited security against limited forms of corruption
 - Only synchronous protocols
- Kleptography and cryptovirology

Prior Work

- Subliminal channels
 - Simmons 1984, ...
- Divertible protocols
 - Blaze, Bleumer, Strauss 1998
 - Limited security against limited forms of corruption
 - Only synchronous protocols
- Kleptography and cryptovirology
 - Young and Yung 1996

Prior Work

- Subliminal channels
 - Simmons 1984, ...
- Divertible protocols
 - Blaze, Bleumer, Strauss 1998
 - Limited security against limited forms of corruption
 - Only synchronous protocols
- Kleptography and cryptovirology
 - Young and Yung 1996
- Algorithm Substitution Attacks

Prior Work

- Subliminal channels
 - Simmons 1984, ...
- Divertible protocols
 - Blaze, Bleumer, Strauss 1998
 - Limited security against limited forms of corruption
 - Only synchronous protocols
- Kleptography and cryptovirology
 - Young and Yung 1996
- Algorithm Substitution Attacks
 - Bellare, Paterson, Rogaway 2014

Prior Work

- Subliminal channels
 - Simmons 1984, ...
- Divertible protocols
 - Blaze, Bleumer, Strauss 1998
 - Limited security against limited forms of corruption
 - Only synchronous protocols
- Kleptography and cryptovirology
 - Young and Yung 1996
- Algorithm Substitution Attacks
 - Bellare, Paterson, Rogaway 2014
 - Symmetric-key encryption

Prior Work

- Subliminal channels
 - Simmons 1984, ...
- Divertible protocols
 - Blaze, Bleumer, Strauss 1998
 - Limited security against limited forms of corruption
 - Only synchronous protocols
- Kleptography and cryptovirology
 - Young and Yung 1996
- Algorithm Substitution Attacks
 - Bellare, Paterson, Rogaway 2014
 - Symmetric-key encryption
 - Limited forms of corruption

Prior Work

- Subliminal channels
 - Simmons 1984, ...
- Divertible protocols
 - Blaze, Bleumer, Strauss 1998
 - Limited security against limited forms of corruption
 - Only synchronous protocols
- Kleptography and cryptovirology
 - Young and Yung 1996
- Algorithm Substitution Attacks
 - Bellare, Paterson, Rogaway 2014
 - Symmetric-key encryption
 - Limited forms of corruption
 - Bellare and Hoang 2015

Prior Work

- Subliminal channels
 - Simmons 1984, ...
- Divertible protocols
 - Blaze, Bleumer, Strauss 1998
 - Limited security against limited forms of corruption
 - Only synchronous protocols
- Kleptography and cryptovirology
 - Young and Yung 1996
- Algorithm Substitution Attacks
 - Bellare, Paterson, Rogaway 2014
 - Symmetric-key encryption
 - Limited forms of corruption
 - Bellare and Hoang 2015
 - Deterministic PKE

Prior Work

- Subliminal channels
 - Simmons 1984, ...
- Divertible protocols
 - Blaze, Bleumer, Strauss 1998
 - Limited security against limited forms of corruption
 - Only synchronous protocols
- Kleptography and cryptovirology
 - Young and Yung 1996
- Algorithm Substitution Attacks
 - Bellare, Paterson, Rogaway 2014
 - Symmetric-key encryption
 - Limited forms of corruption
 - Bellare and Hoang 2015
 - Deterministic PKE
 - Limited forms of corruption

Prior Work

- Subliminal channels
 - Simmons 1984, ...
- Divertible protocols
 - Blaze, Bleumer, Strauss 1998
 - Limited security against limited forms of corruption
 - Only synchronous protocols
- Kleptography and cryptovirology
 - Young and Yung 1996
- Algorithm Substitution Attacks
 - Bellare, Paterson, Rogaway 2014
 - Symmetric-key encryption
 - Limited forms of corruption
 - Bellare and Hoang 2015
 - Deterministic PKE
 - Limited forms of corruption
- Backdoored PRGs

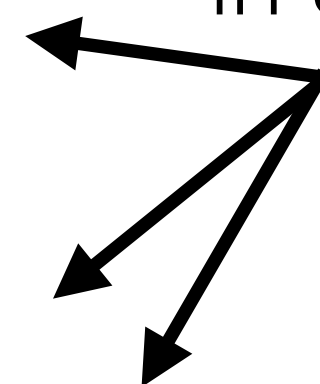
Prior Work

- Subliminal channels
 - Simmons 1984, ...
- Divertible protocols
 - Blaze, Bleumer, Strauss 1998
 - Limited security against limited forms of corruption
 - Only synchronous protocols
- Kleptography and cryptovirology
 - Young and Yung 1996
- Algorithm Substitution Attacks
 - Bellare, Paterson, Rogaway 2014
 - Symmetric-key encryption
 - Limited forms of corruption
 - Bellare and Hoang 2015
 - Deterministic PKE
 - Limited forms of corruption
- Backdoored PRGs
 - Dodis, Ganesh, Golovnev, Juels, and Ristenpart 2015

Prior Work

- Subliminal channels
 - Simmons 1984, ...
- Divertible protocols
 - Blaze, Bleumer, Strauss 1998
 - Limited security against limited forms of corruption
 - Only synchronous protocols
- Kleptography and cryptovirology
 - Young and Yung 1996
- Algorithm Substitution Attacks
 - Bellare, Paterson, Rogaway 2014
 - Symmetric-key encryption
 - Limited forms of corruption
 - Bellare and Hoang 2015
 - Deterministic PKE
 - Limited forms of corruption
- Backdoored PRGs
 - Dodis, Ganesh, Golovnev, Juels, and Ristenpart 2015

Impossibility results
in strongest models



Prior Work

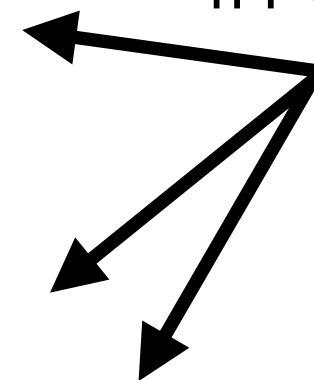
- Subliminal channels
 - Simmons 1984, ...
- Divertible protocols
 - Blaze, Bleumer, Strauss 1998
 - Limited security against limited forms of corruption

We generalize these models and show a way around the impossibility results.

- Algorithm Substitution Attacks
 - Bellare, Paterson, Rogaway 2014
 - Symmetric-key encryption
 - Limited forms of corruption
 - Bellare and Hoang 2015
 - Deterministic PKE
 - Limited forms of corruption

- Backdoored PRGs
 - Dodis, Ganesh, Golovnev, Juels, and Ristenpart 2015

Impossibility results
in strongest models



Act II...

Reverse Firewalls!

Reverse Firewalls!

- Firewall sits between Alice's computer and the outside world

Reverse Firewalls!

- Firewall sits between Alice's computer and the outside world



Reverse Firewalls!

- Firewall sits between Alice's computer and the outside world



Reverse Firewalls!

- Firewall sits between Alice's computer and the outside world



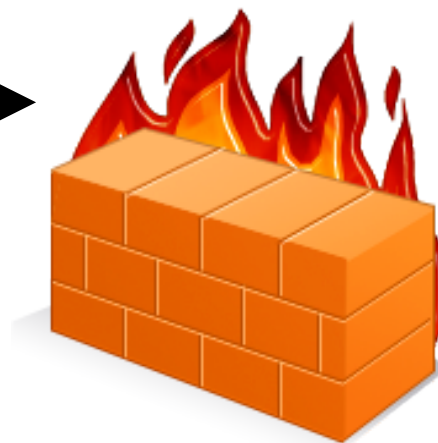
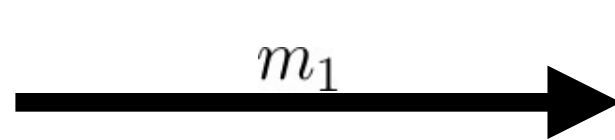
Reverse Firewalls!

- Firewall sits between Alice's computer and the outside world
- Modifies the messages that Alice sends and receives.



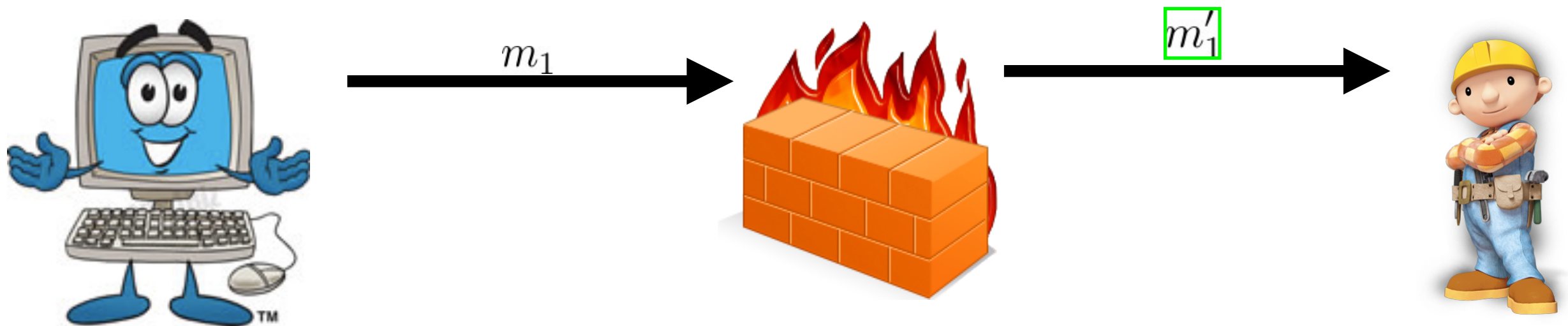
Reverse Firewalls!

- Firewall sits between Alice's computer and the outside world
- Modifies the messages that Alice sends and receives.



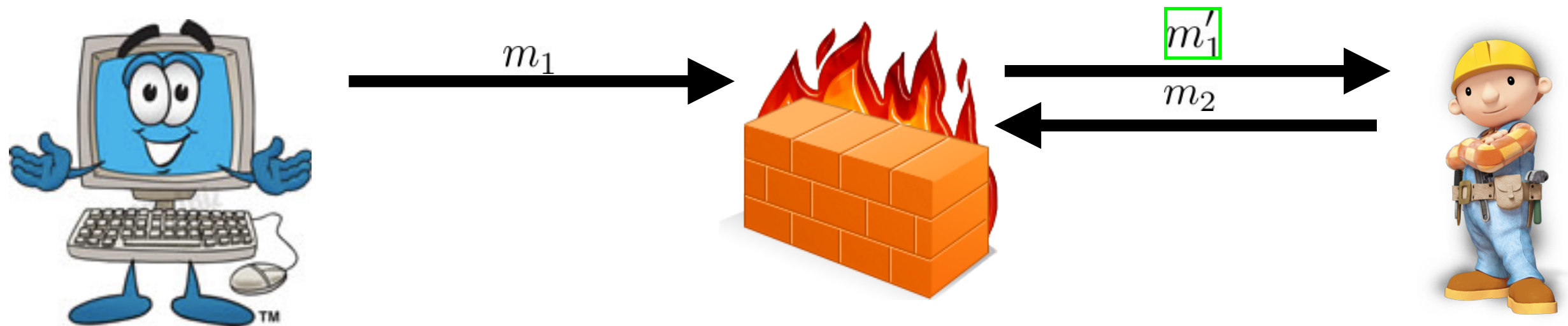
Reverse Firewalls!

- Firewall sits between Alice's computer and the outside world
- Modifies the messages that Alice sends and receives.



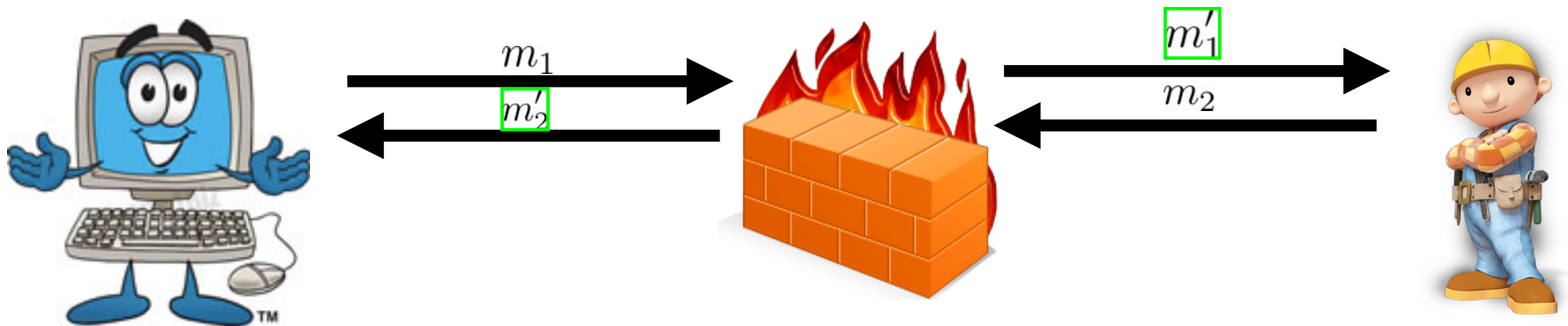
Reverse Firewalls!

- Firewall sits between Alice's computer and the outside world
- Modifies the messages that Alice sends and receives.



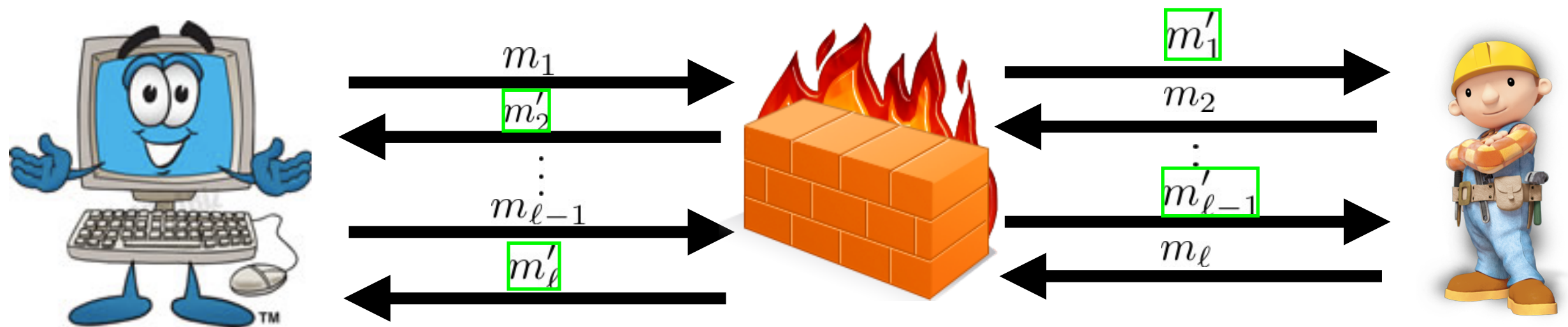
Reverse Firewalls!

- Firewall sits between Alice's computer and the outside world
- Modifies the messages that Alice sends and receives.



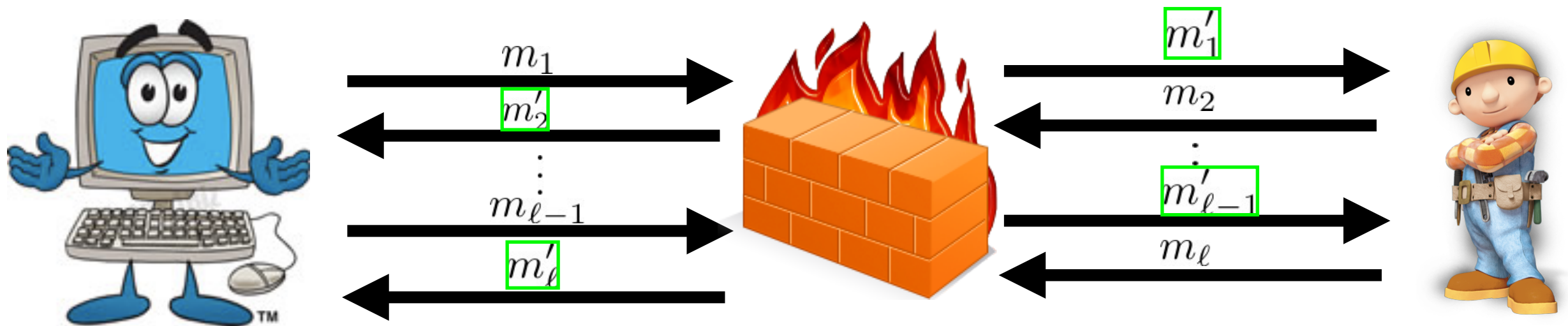
Reverse Firewalls!

- Firewall sits between Alice's computer and the outside world
- Modifies the messages that Alice sends and receives.



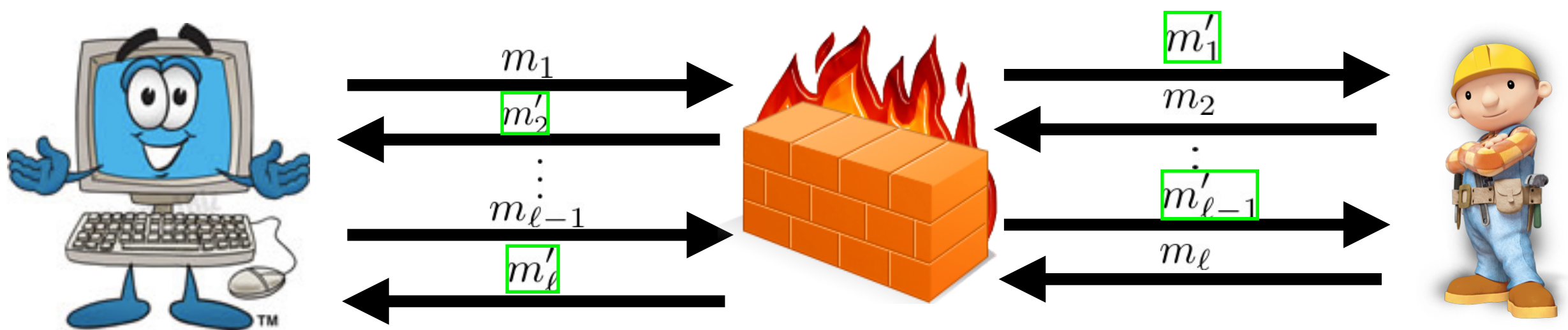
Reverse Firewalls!

- Firewall sits between Alice's computer and the outside world
- Modifies the messages that Alice sends and receives.
- Transparent to legitimate traffic.



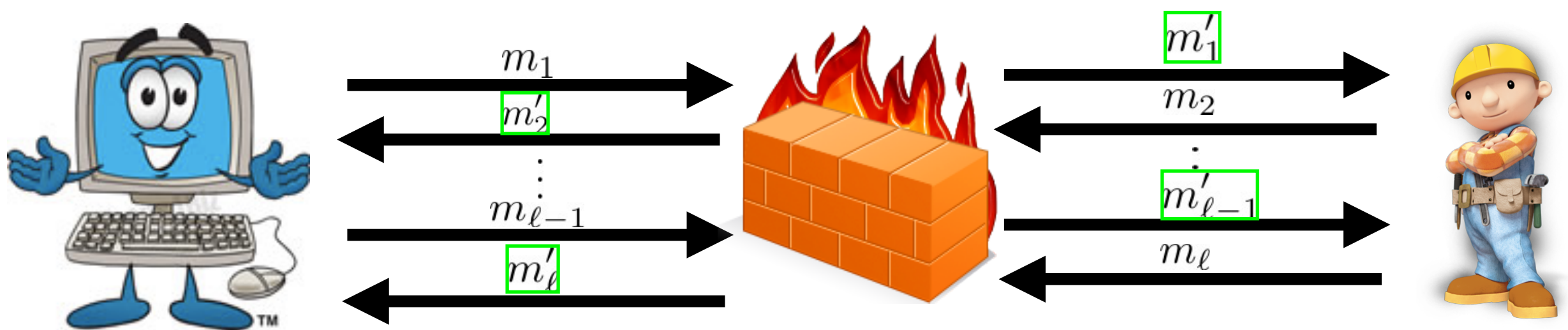
Reverse Firewalls!

- Firewall sits between Alice's computer and the outside world
- Modifies the messages that Alice sends and receives.
- Transparent to legitimate traffic.
 - Certainly doesn't break functionality.



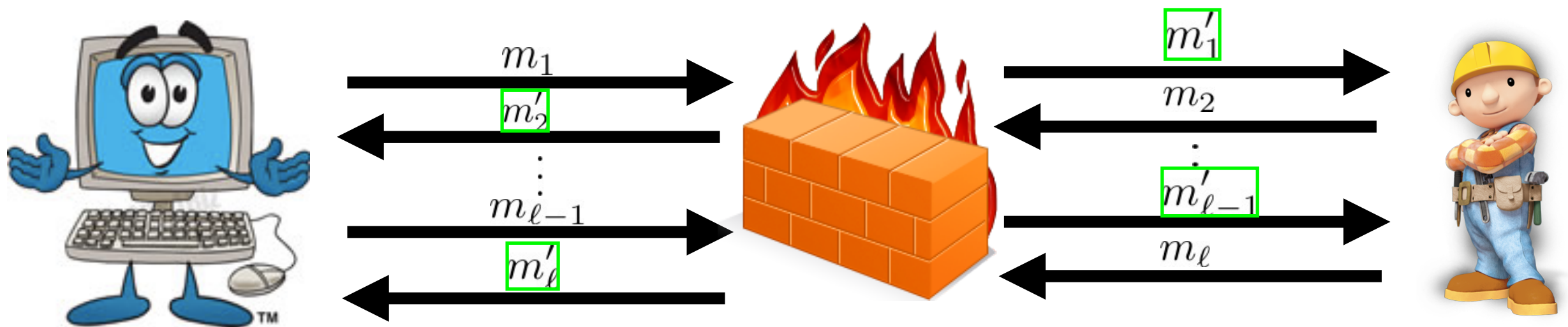
Reverse Firewalls!

- Firewall sits between Alice's computer and the outside world
- Modifies the messages that Alice sends and receives.
- Transparent to legitimate traffic.
 - Certainly doesn't break functionality.
- Shares no secrets with Alice.



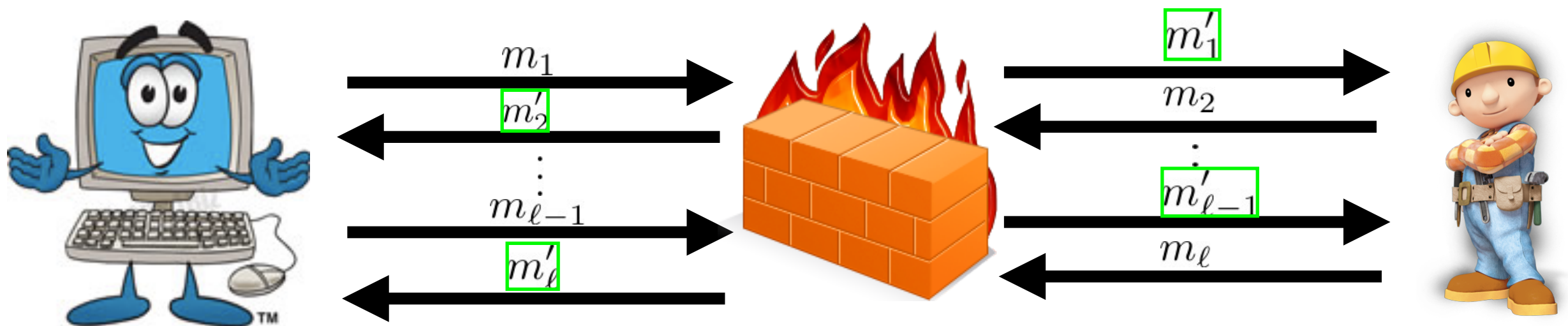
Reverse Firewalls!

- Firewall sits between Alice's computer and the outside world
- Modifies the messages that Alice sends and receives.
- Transparent to legitimate traffic.
 - Certainly doesn't break functionality.
- Shares no secrets with Alice.
 - **We don't trust the firewall.**



Reverse Firewalls!

- Firewall sits between Alice's computer and the outside world
- Modifies the messages that Alice sends and receives.
- Transparent to legitimate traffic.
 - Certainly doesn't break functionality.
- Shares no secrets with Alice.
 - **We don't trust the firewall.**
- "Improves security!"



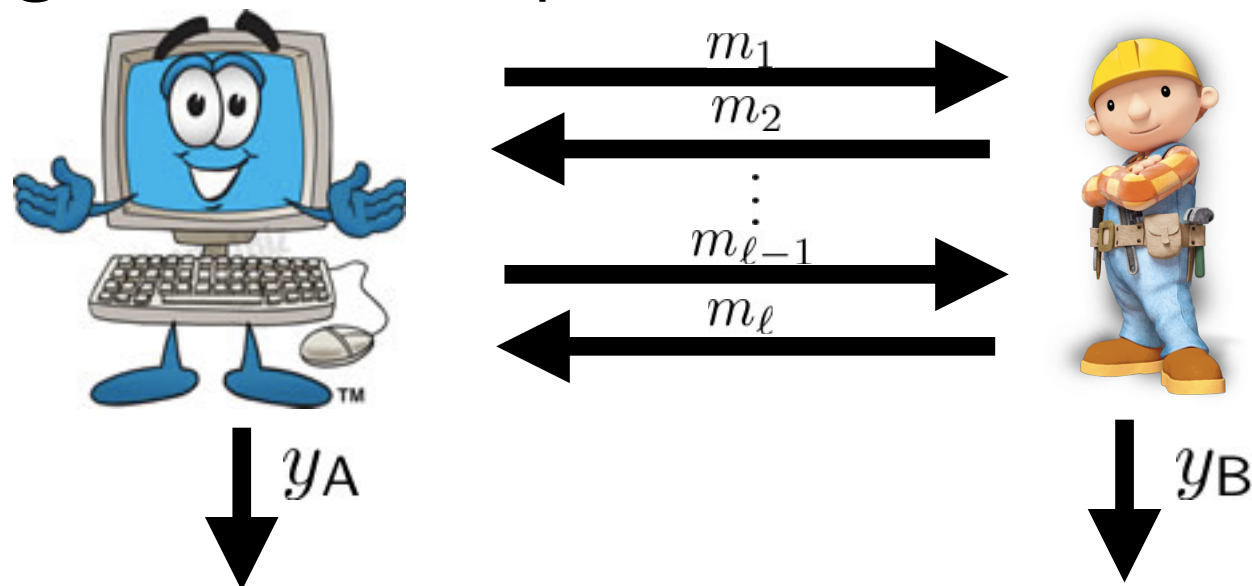
Reverse Firewall Functionality

Reverse Firewall Functionality

Underlying classical protocol has some functionality.

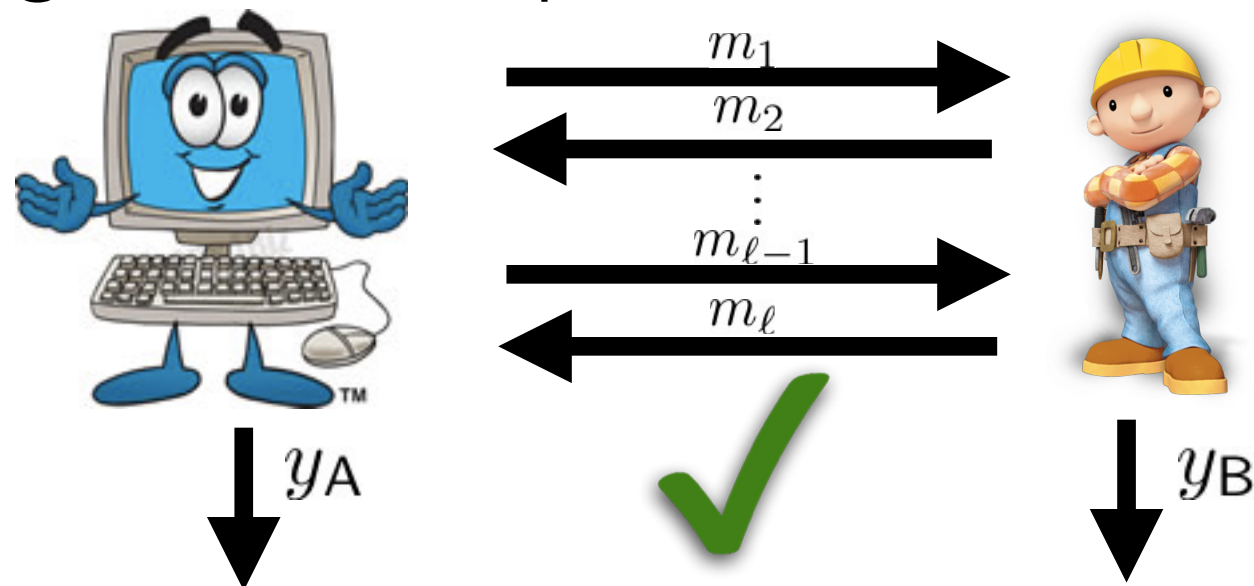
Reverse Firewall Functionality

Underlying classical protocol has some functionality.



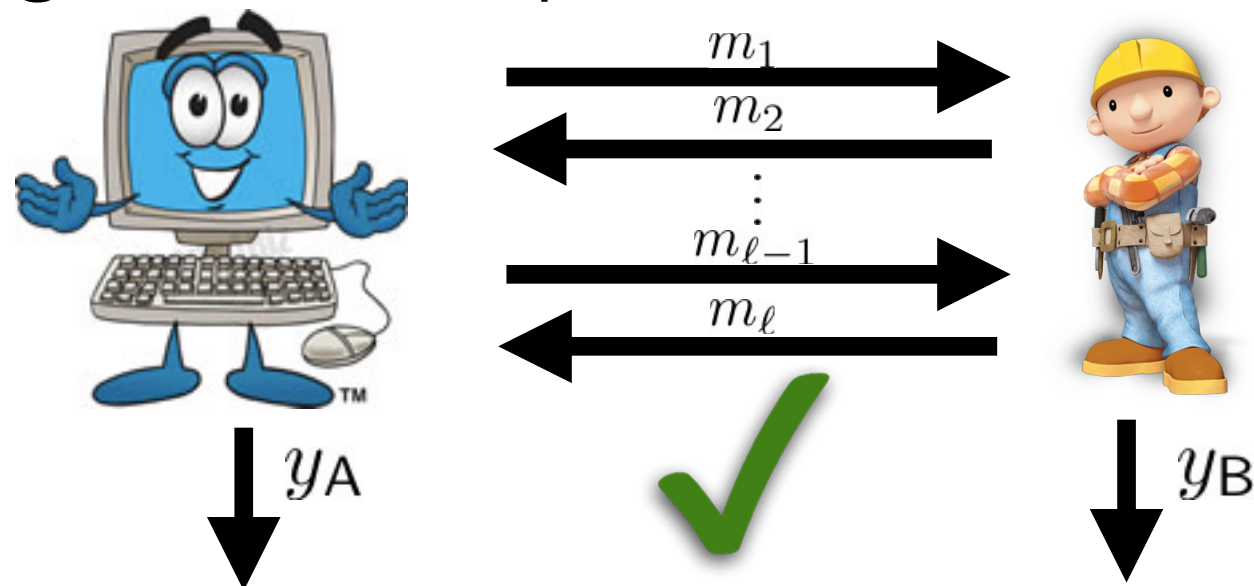
Reverse Firewall Functionality

Underlying classical protocol has some functionality.



Reverse Firewall Functionality

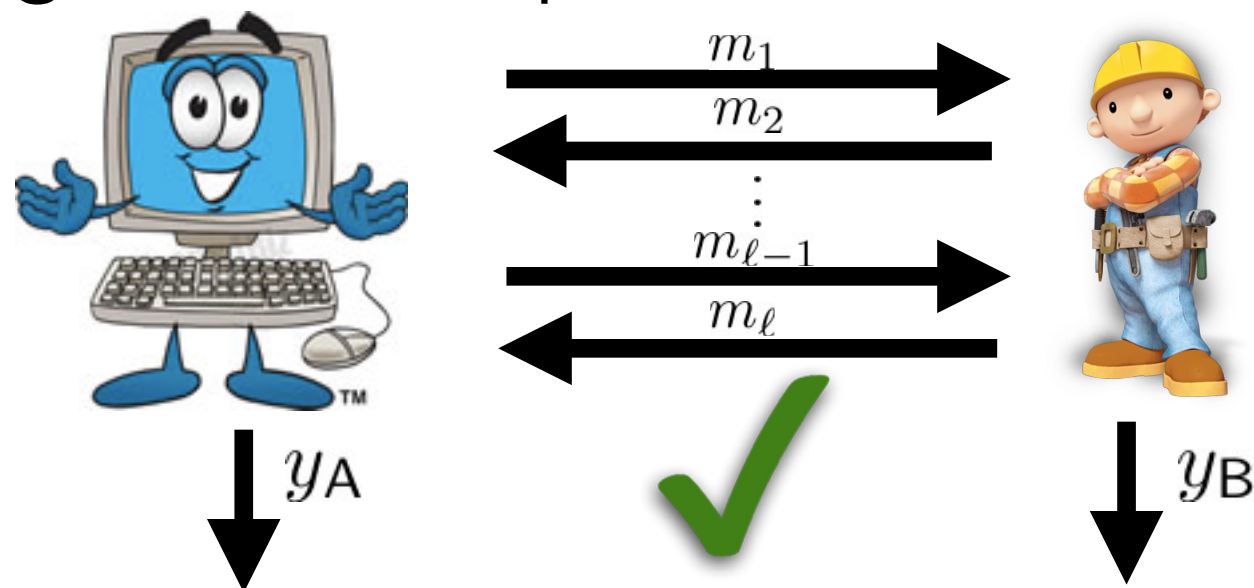
Underlying classical protocol has some functionality.



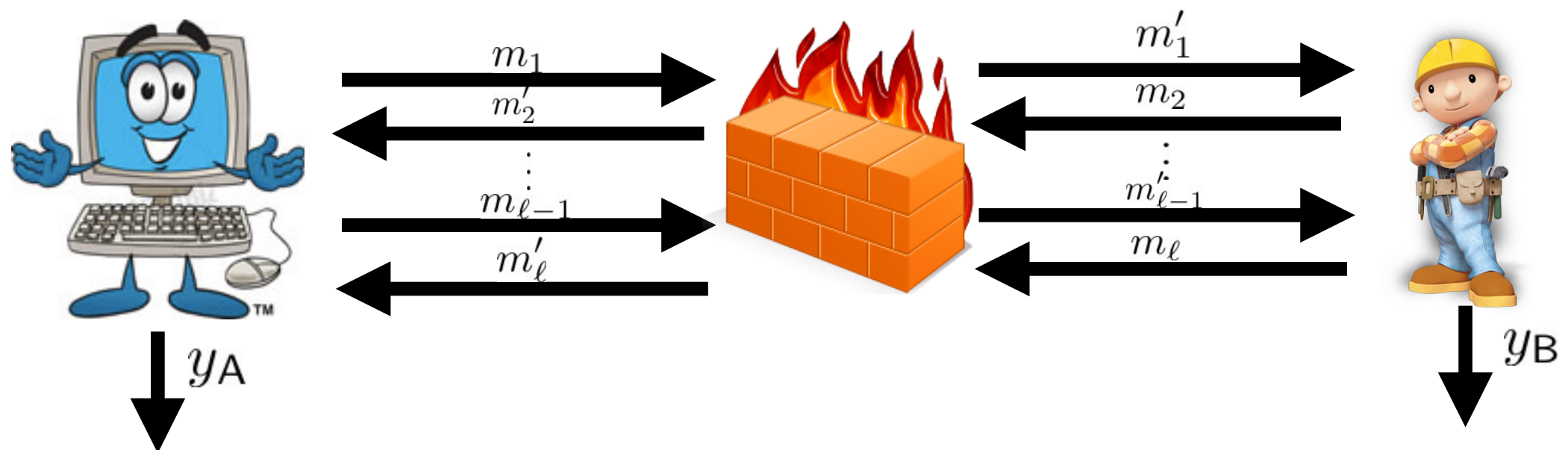
Protocol *with firewall* has same functionality.

Reverse Firewall Functionality

Underlying classical protocol has some functionality.

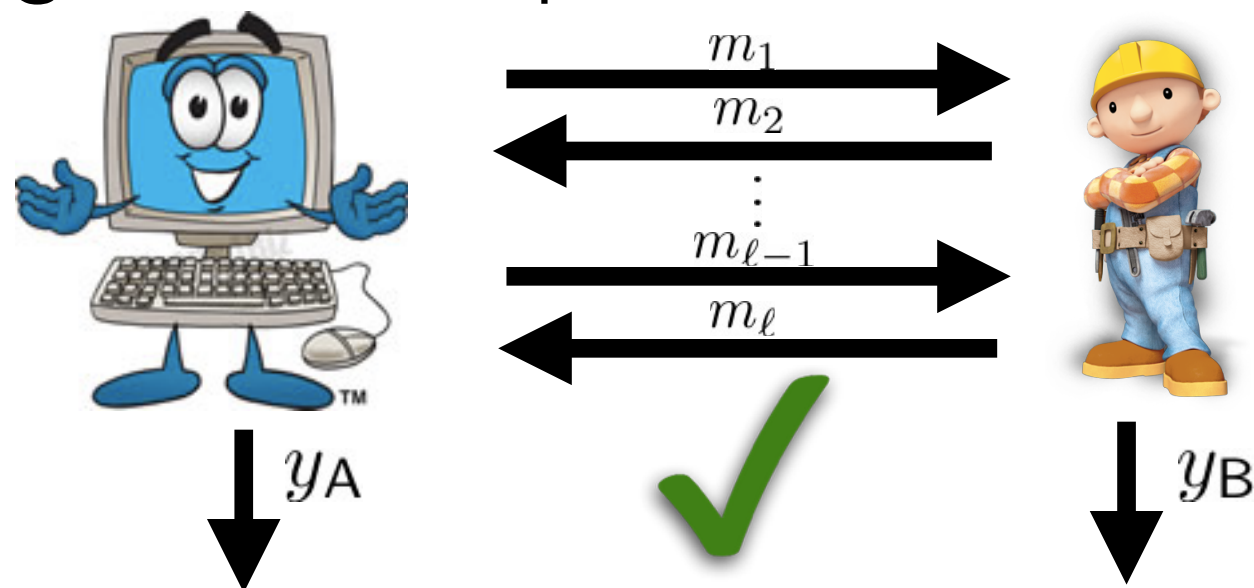


Protocol *with firewall* has same functionality.

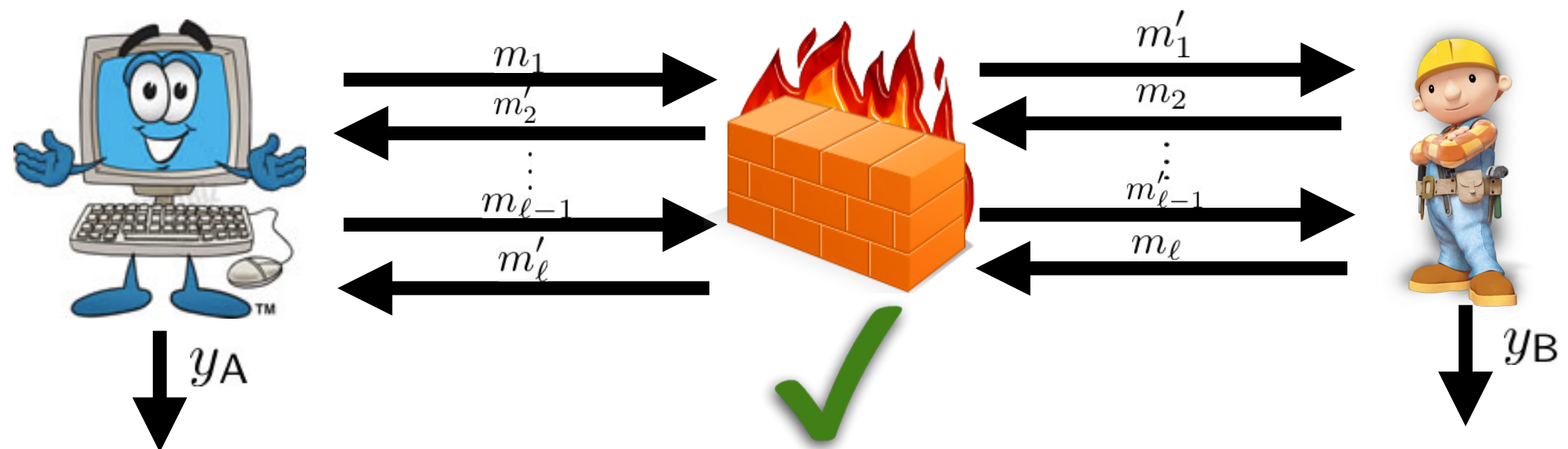


Reverse Firewall Functionality

Underlying classical protocol has some functionality.



Protocol *with firewall* has same functionality.



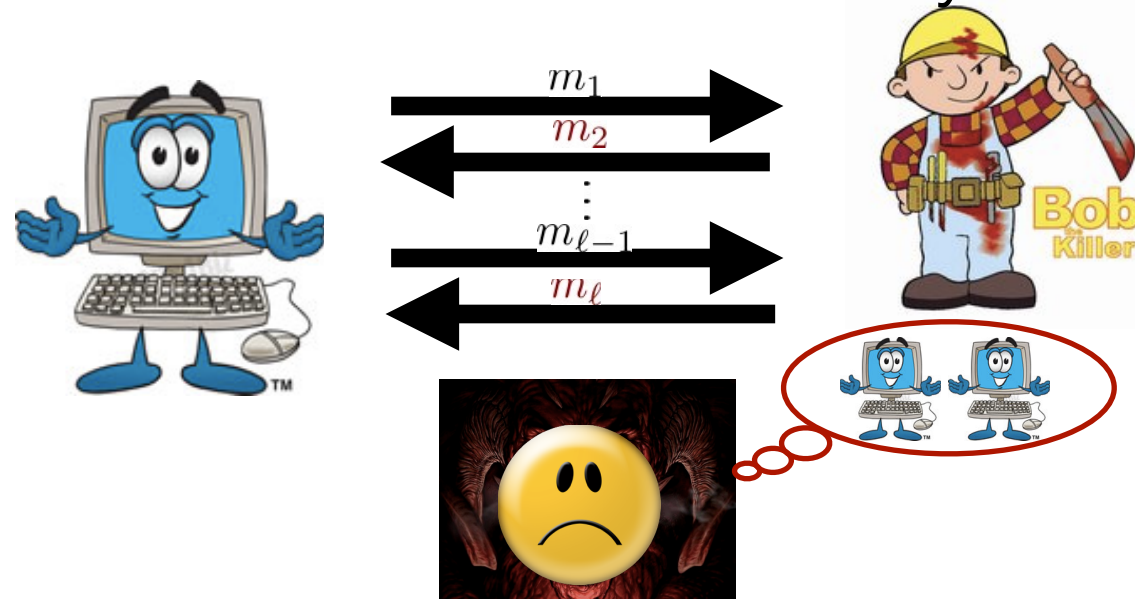
Reverse Firewall Security

Reverse Firewall Security

Underlying protocol satisfies some security notion.

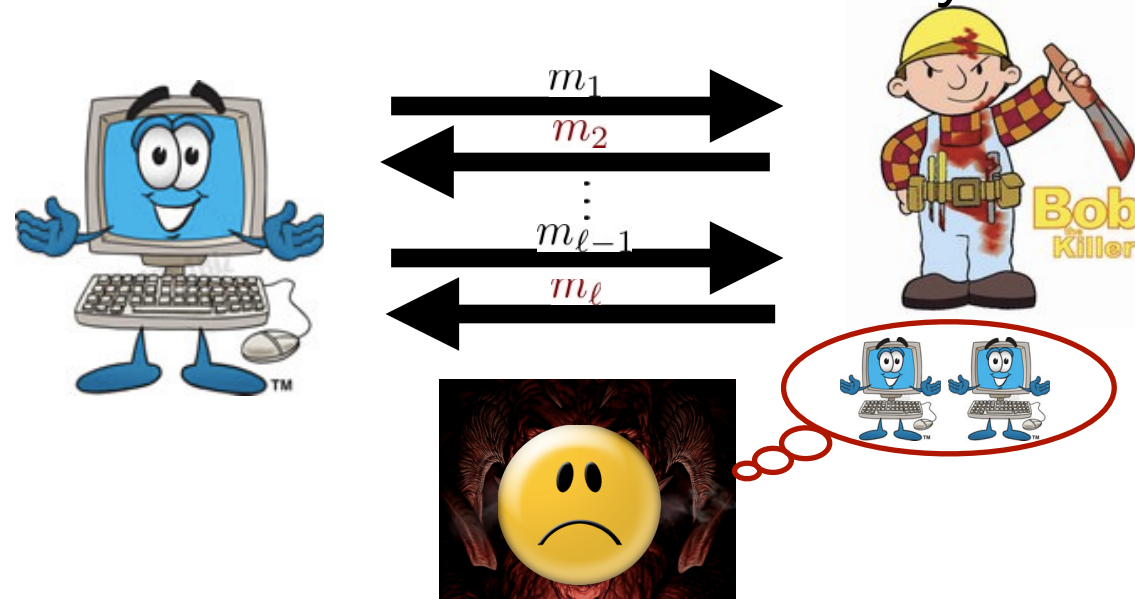
Reverse Firewall Security


Underlying protocol satisfies some security notion.



Reverse Firewall Security

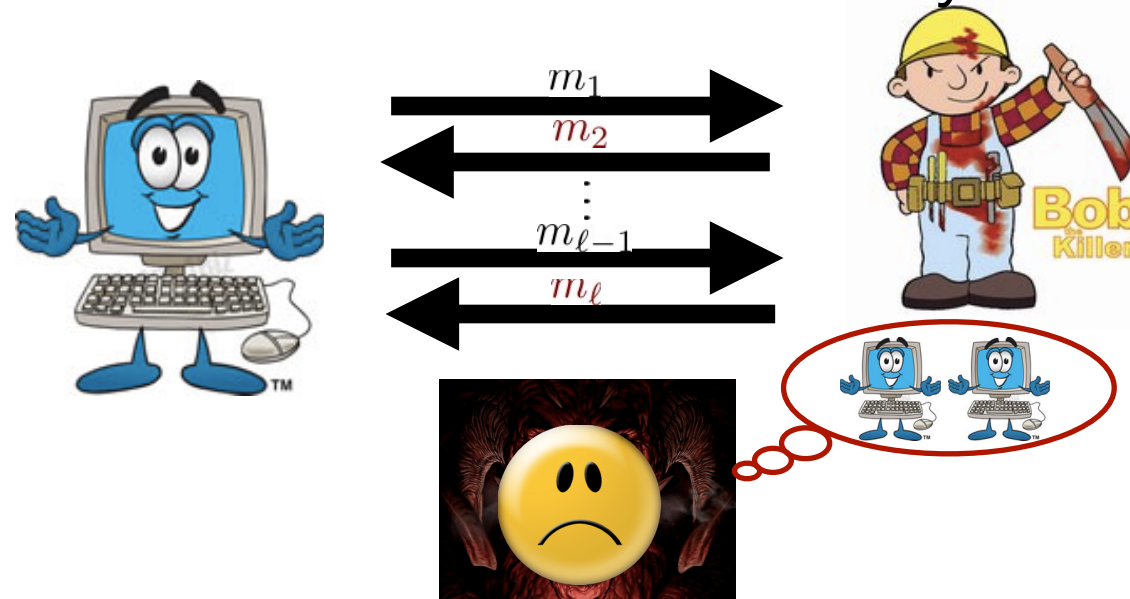
Underlying protocol satisfies some security notion.




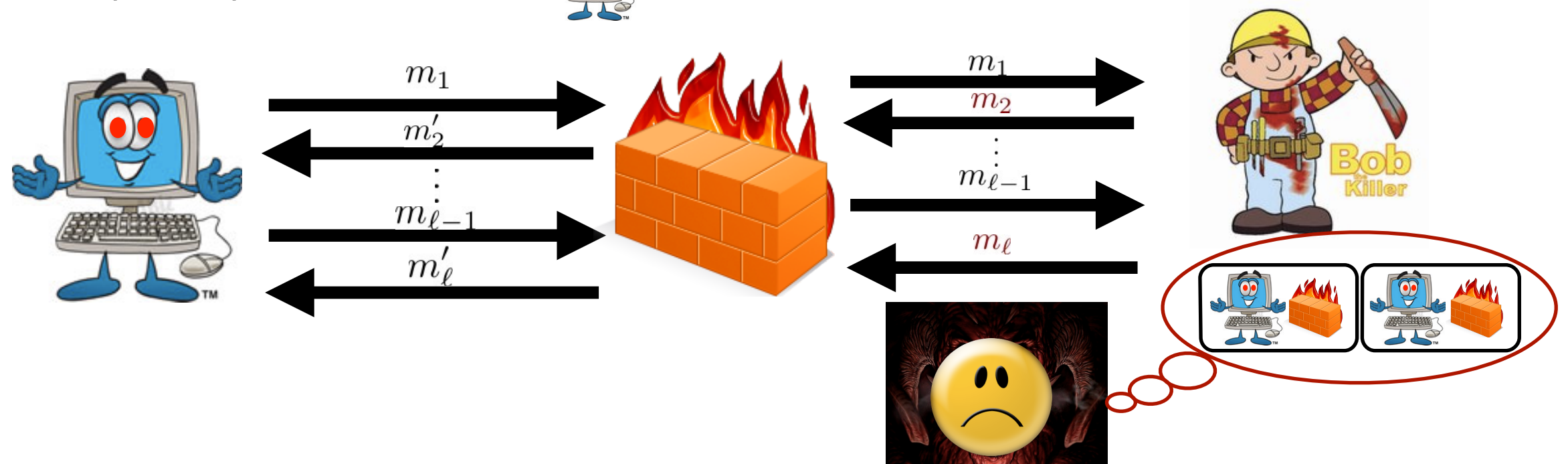
Protocol with firewall satisfies the same security notion *for any efficient corrupt implementation*  *of Alice.*

Reverse Firewall Security

Underlying protocol satisfies some security notion.

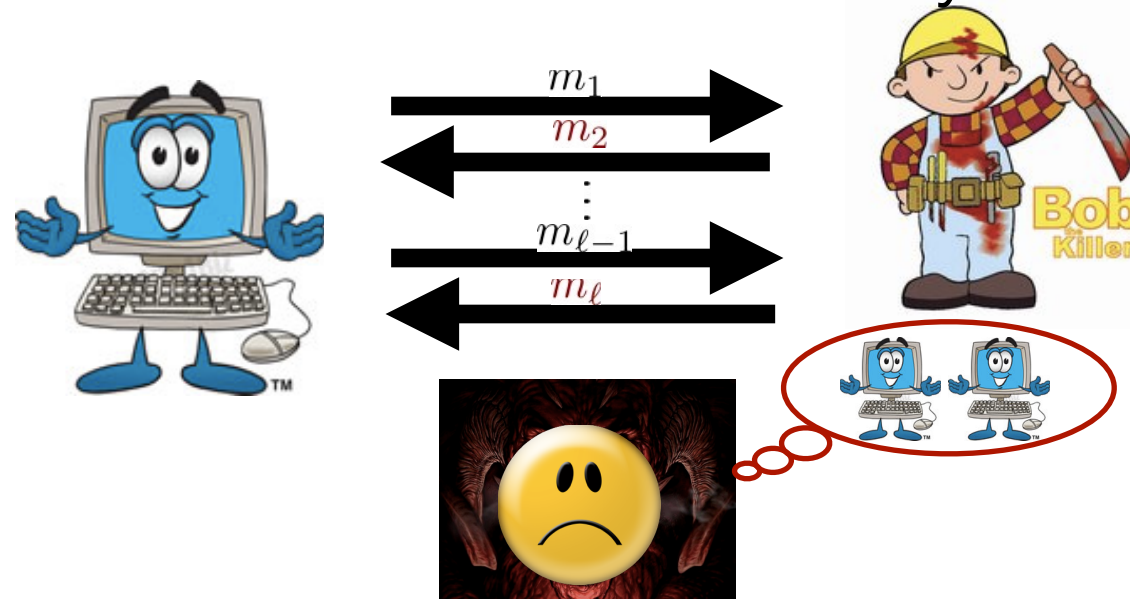



Protocol with firewall satisfies the same security notion *for any efficient corrupt implementation*  *of Alice.*

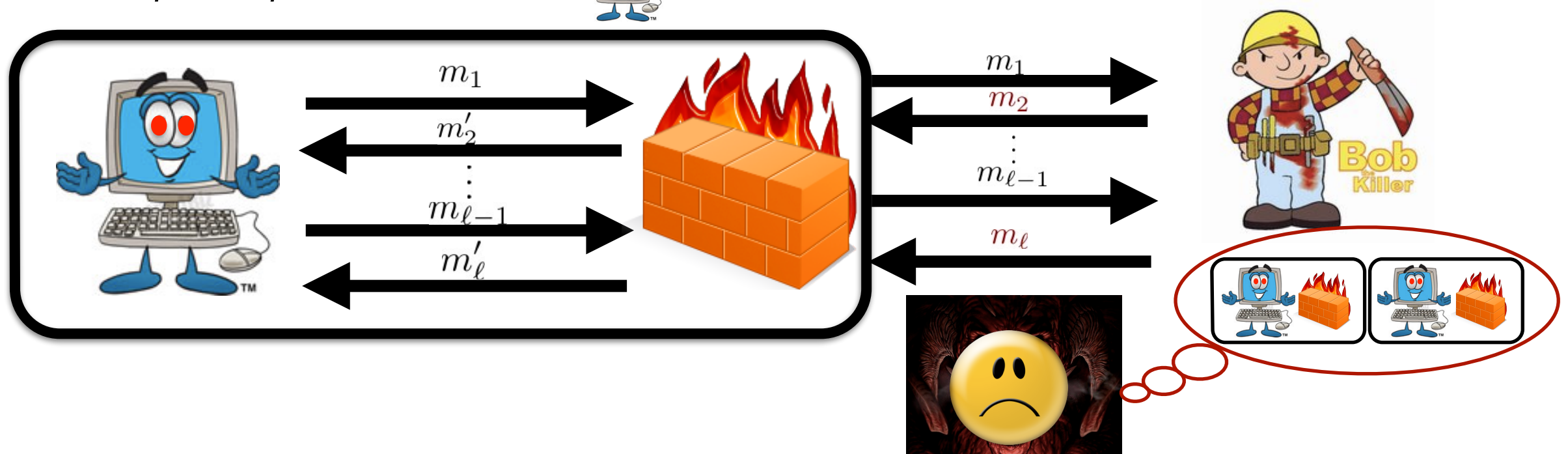


Reverse Firewall Security

Underlying protocol satisfies some security notion.

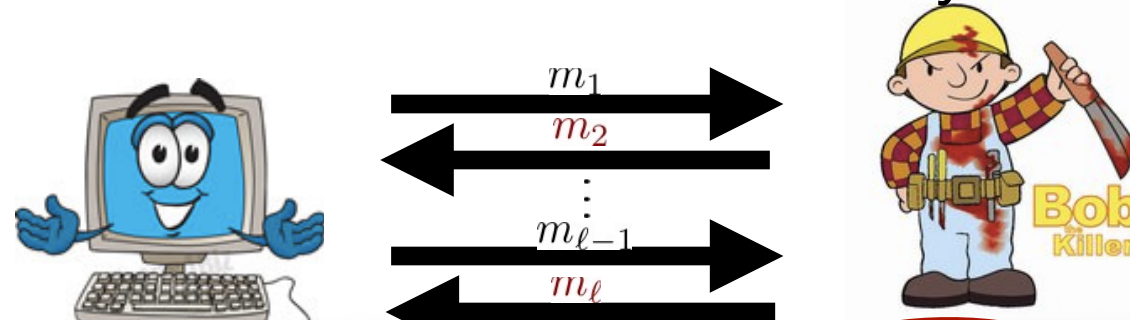


Protocol with firewall satisfies the same security notion *for any efficient corrupt implementation*  *of Alice.*



Reverse Firewall Security

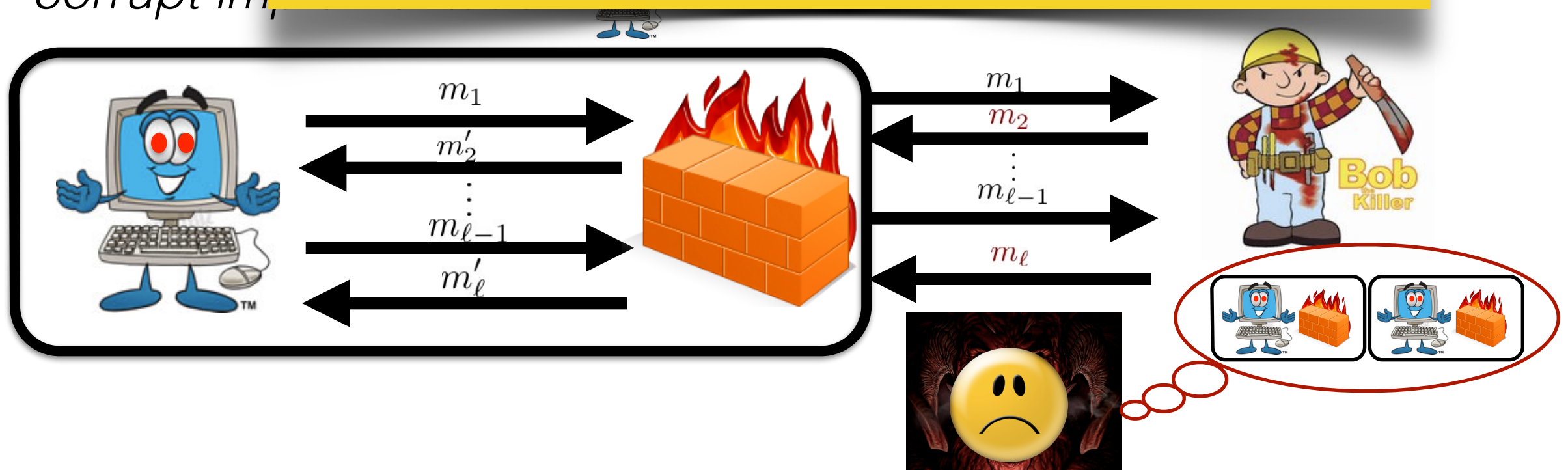
Underlying protocol satisfies some security notion.



Note: We require that the protocol is functional and secure *without the reverse firewall* when Alice's implementation is not corrupted.

Protocol w
corrupt im

any efficient



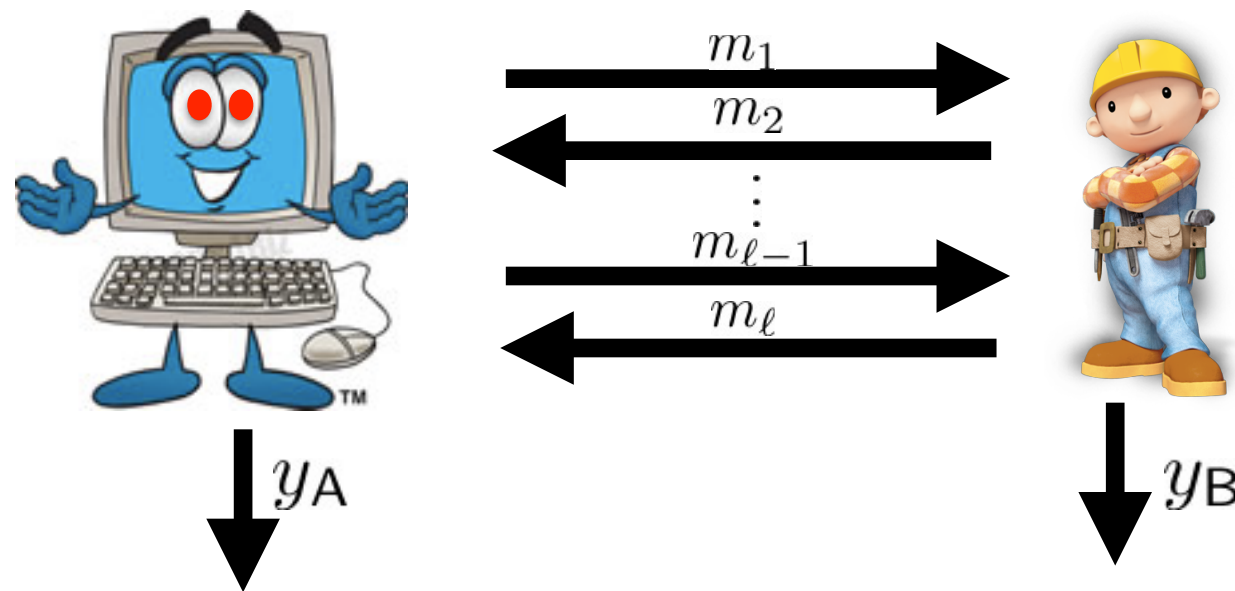
Functionality Maintaining Corruption

Functionality Maintaining Corruption

A corrupted implementation  of Alice is *functionality maintaining* if the protocol is still functional when Alice is replaced by .

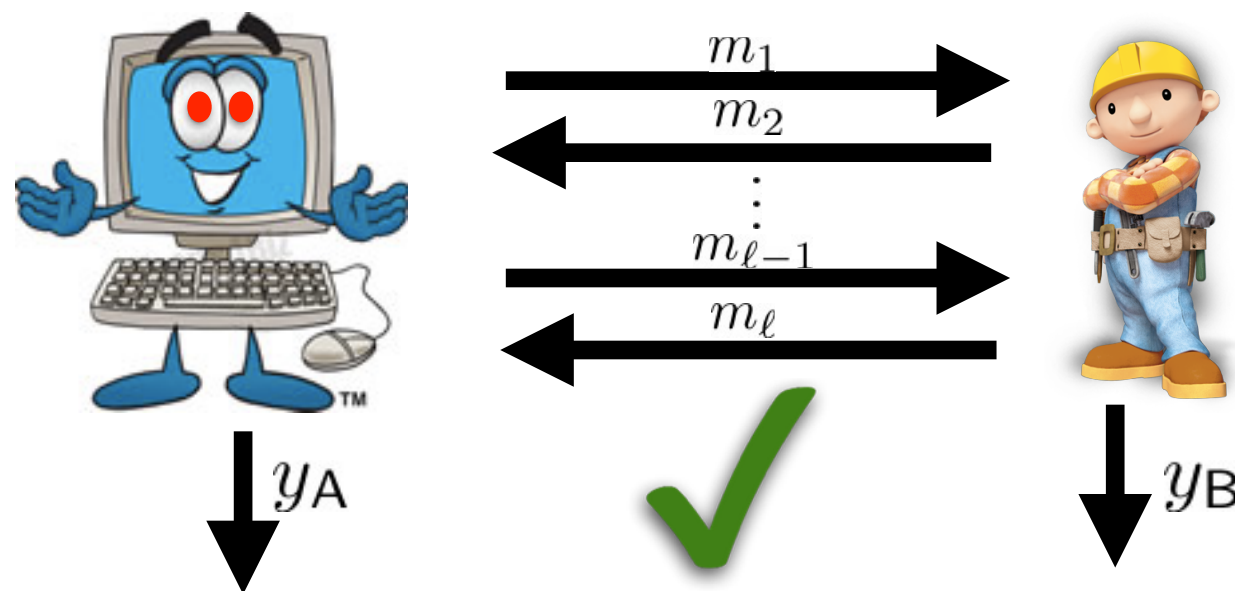
Functionality Maintaining Corruption

A corrupted implementation  of Alice is *functionality maintaining* if the protocol is still functional when Alice is replaced by .



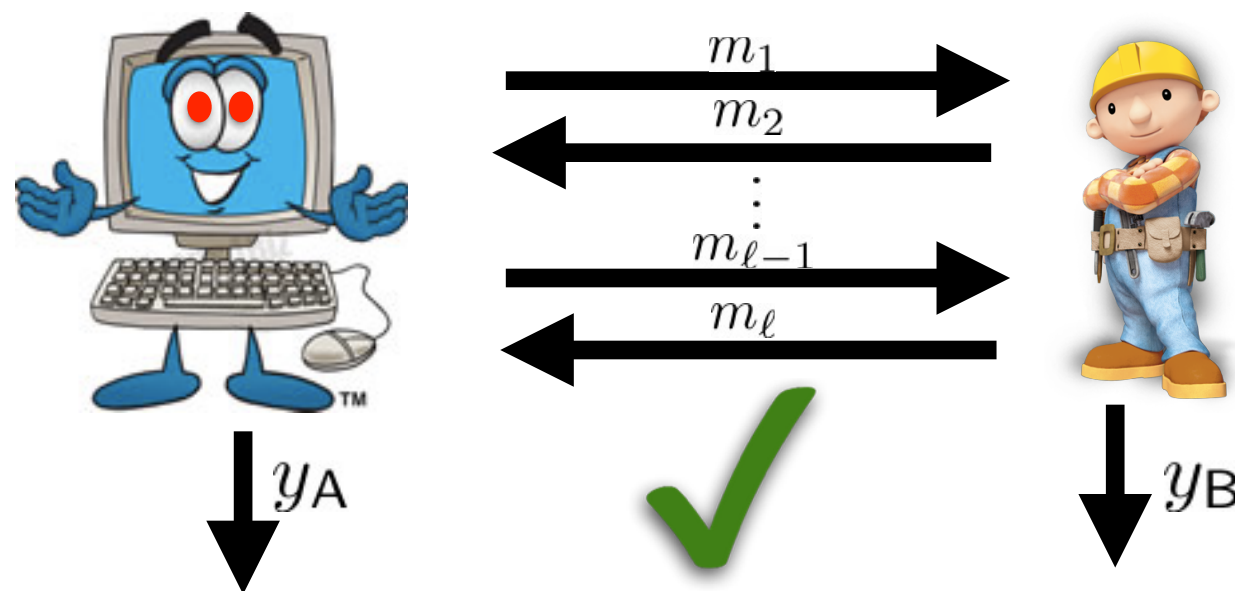
Functionality Maintaining Corruption

A corrupted implementation  of Alice is *functionality maintaining* if the protocol is still functional when Alice is replaced by .



Functionality Maintaining Corruption

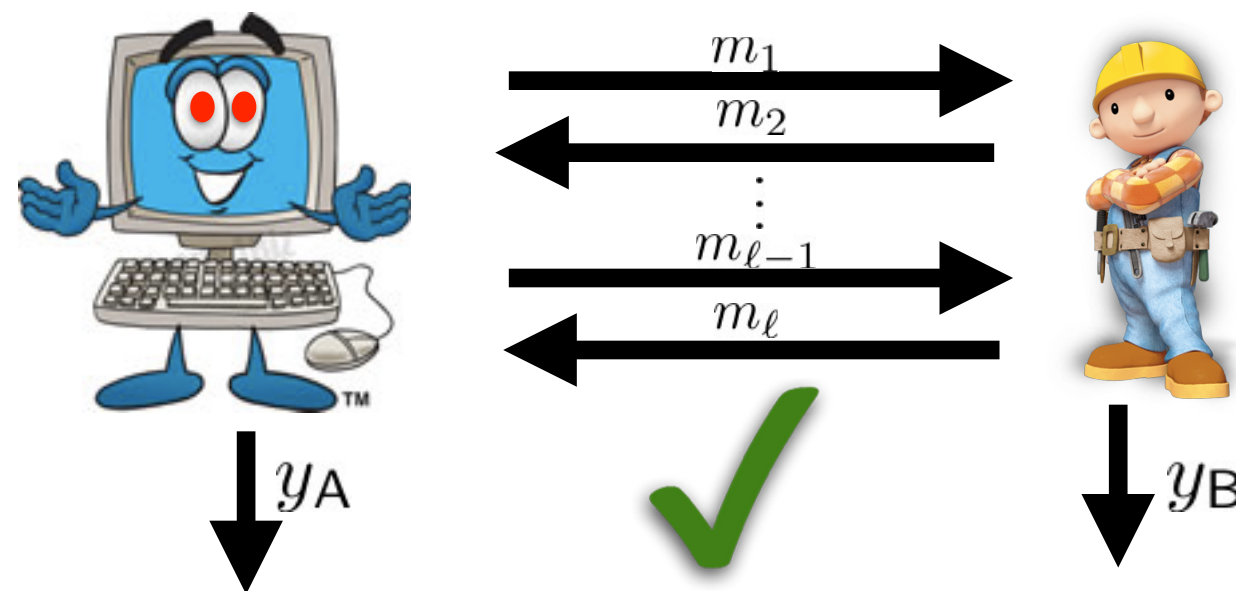
A corrupted implementation  of Alice is *functionality maintaining* if the protocol is still functional when Alice is replaced by .



(This is much more general than honest-but-curious adversaries, undetectable corruption, steganographic attacks, etc.)

Functionality Maintaining Corruption

A corrupted implementation  of Alice is *functionality maintaining* if the protocol is still functional when Alice is replaced by .



(This is much more general than honest-but-curious adversaries, undetectable corruption, steganographic attacks, etc.)

Simple Example: Semantically Secure PKE

Simple Example: Semantically Secure PKE

Let $(\text{Enc}, \text{Dec}, \text{Rerand})$ be a *rerandomizable* PKE scheme such that $\text{Dec}(\text{Rerand}(\text{Enc}(m))) = m$ and $\text{Rerand}(C) \approx_c \text{Enc}(0)$ for any C .

Simple Example: Semantically Secure PKE

Let $(\text{Enc}, \text{Dec}, \text{Rerand})$ be a *rerandomizable* PKE scheme such that $\text{Dec}(\text{Rerand}(\text{Enc}(m))) = m$ and $\text{Rerand}(C) \approx_c \text{Enc}(0)$ for any C .

Underlying protocol:

Simple Example: Semantically Secure PKE

Let $(\text{Enc}, \text{Dec}, \text{Rerand})$ be a *rerandomizable* PKE scheme such that $\text{Dec}(\text{Rerand}(\text{Enc}(m))) = m$ and $\text{Rerand}(C) \approx_c \text{Enc}(0)$ for any C .

Underlying protocol:



$\text{Enc}_{pk}(m)$



Simple Example: Semantically Secure PKE

Let $(\text{Enc}, \text{Dec}, \text{Rerand})$ be a *rerandomizable* PKE scheme such that $\text{Dec}(\text{Rerand}(\text{Enc}(m))) = m$ and $\text{Rerand}(C) \approx_c \text{Enc}(0)$ for any C .

Underlying protocol:



$\text{Enc}_{pk}(m)$

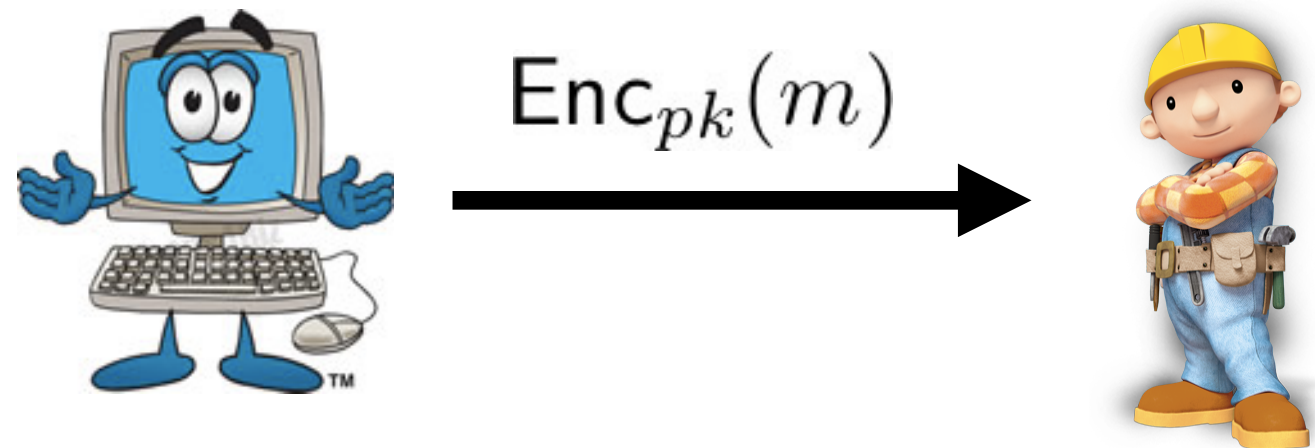


Firewall:

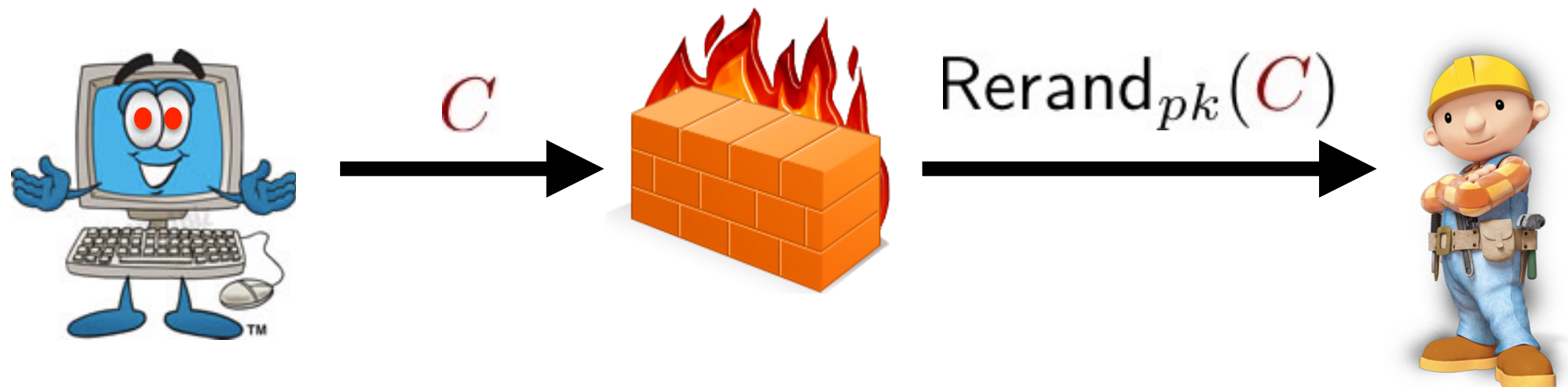
Simple Example: Semantically Secure PKE

Let $(\text{Enc}, \text{Dec}, \text{Rerand})$ be a *rerandomizable* PKE scheme such that $\text{Dec}(\text{Rerand}(\text{Enc}(m))) = m$ and $\text{Rerand}(C) \approx_c \text{Enc}(0)$ for any C .

Underlying protocol:



Firewall:



Simple Example: Semantically Secure PKE

Functionality:

Let (E, D) be a PKE scheme such that

$\text{Enc}(0) \approx_c \text{Enc}(1)$

Under

Firewall:



C



$\text{Rerand}_{pk}(C)$



Simple Example: Semantically Secure PKE

Functionality:

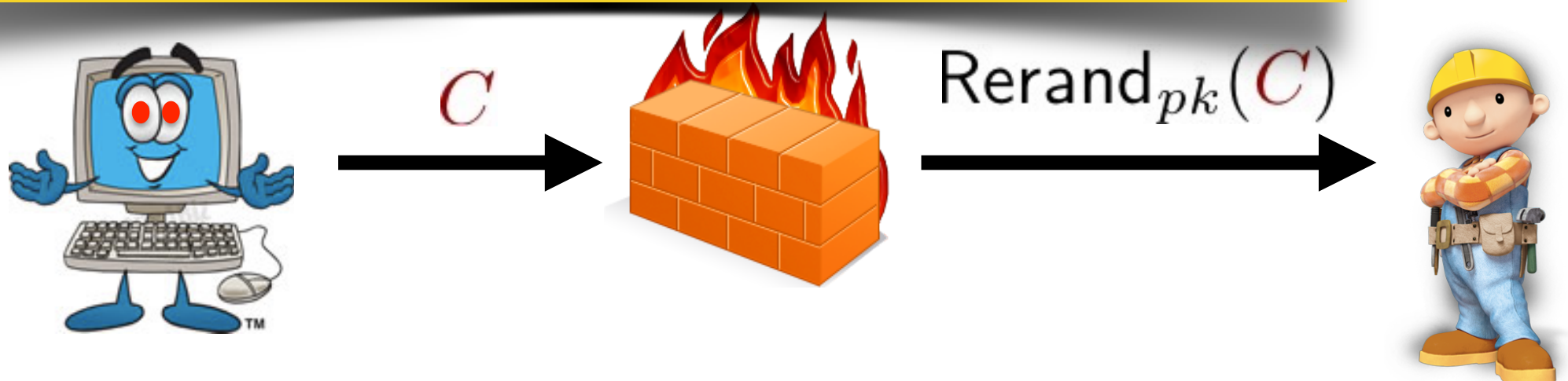
- Is the protocol functional without the firewall?

Let (E

scheme
 $\approx_c \text{Enc}(0)$

Under

Firewall:



Simple Example: Semantically Secure PKE

Functionality:

- Is the protocol functional without the firewall? ✓

Under

scheme
 $\approx_c \text{Enc}(0)$



Firewall:



C



$\text{Rerand}_{pk}(C)$



Simple Example: Semantically Secure PKE

Functionality:

- Is the protocol functional without the firewall? ✓
- Is the protocol functional with the firewall?

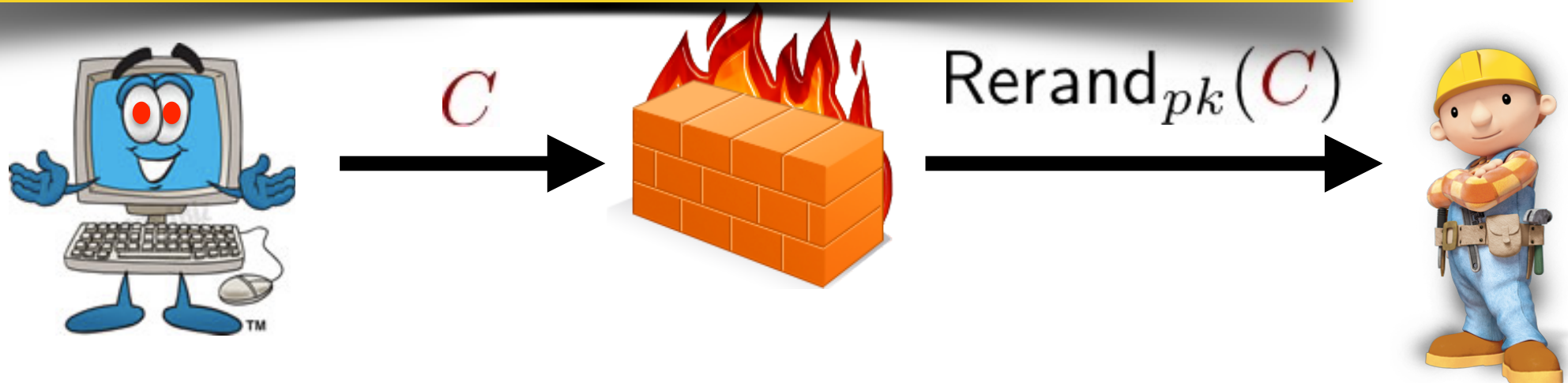
Let (E

scheme
 $\approx_c \text{Enc}(0)$

Under



Firewall:



Simple Example: Semantically Secure PKE

Functionality:

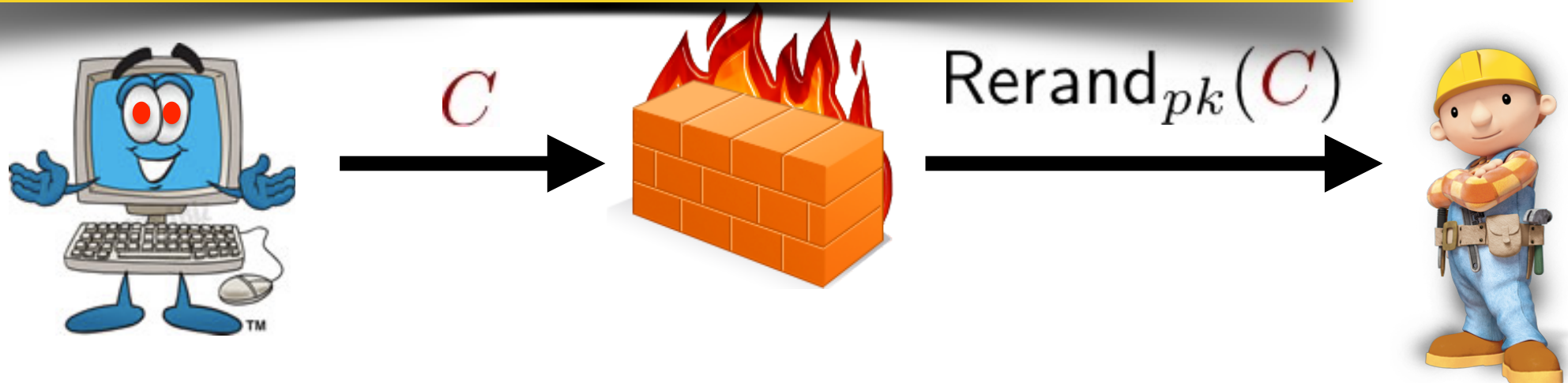
- Is the protocol functional without the firewall? ✓
- Is the protocol functional with the firewall? ✓

scheme
 $\approx_c \text{Enc}(0)$

Under



Firewall:



Simple Example: Semantically Secure PKE

Functionality:

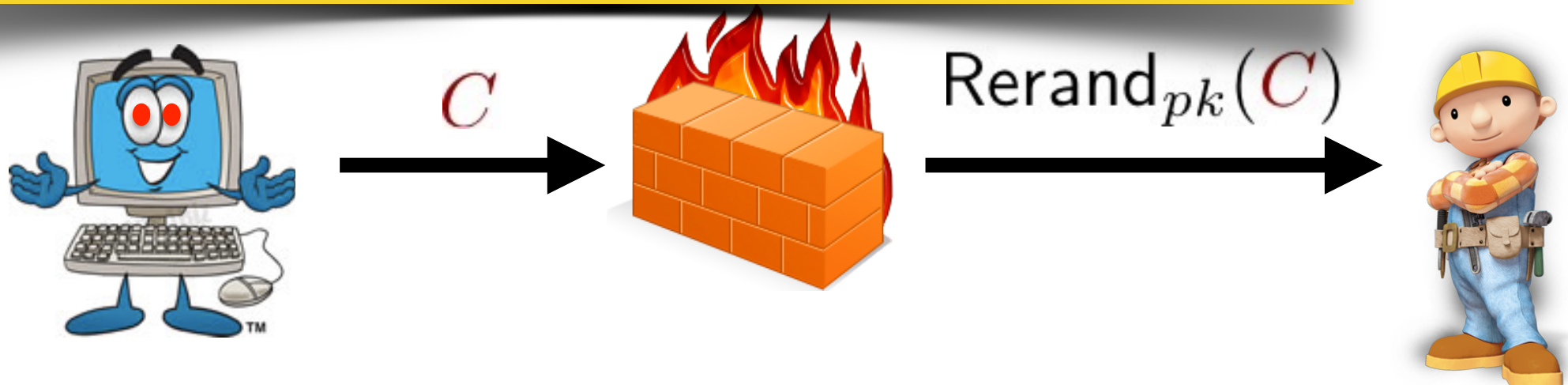
- Is the protocol functional without the firewall? ✓
- Is the protocol functional with the firewall? ✓

Security:

scheme
 $\approx_c \text{Enc}(0)$

Under

Firewall:



Simple Example: Semantically Secure PKE

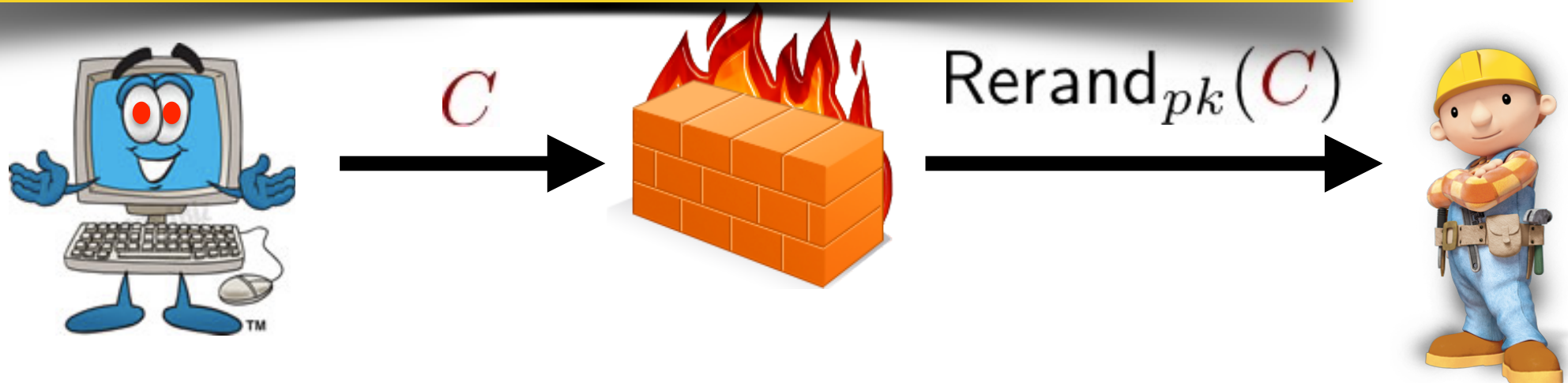
Functionality:

- Is the protocol functional without the firewall? ✓
- Is the protocol functional with the firewall? ✓

Security:

- Is the protocol semantically secure without firewall?

Firewall:



Simple Example: Semantically Secure PKE

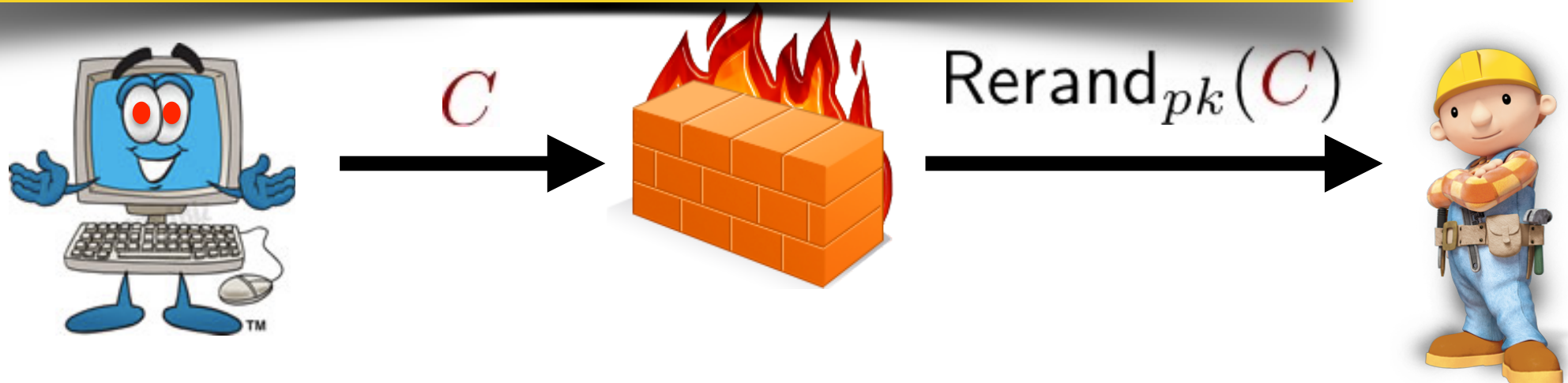
Functionality:

- Is the protocol functional without the firewall? ✓
- Is the protocol functional with the firewall? ✓

Security:

- Is the protocol semantically secure without firewall? ✓

Firewall:



Simple Example: Semantically Secure PKE

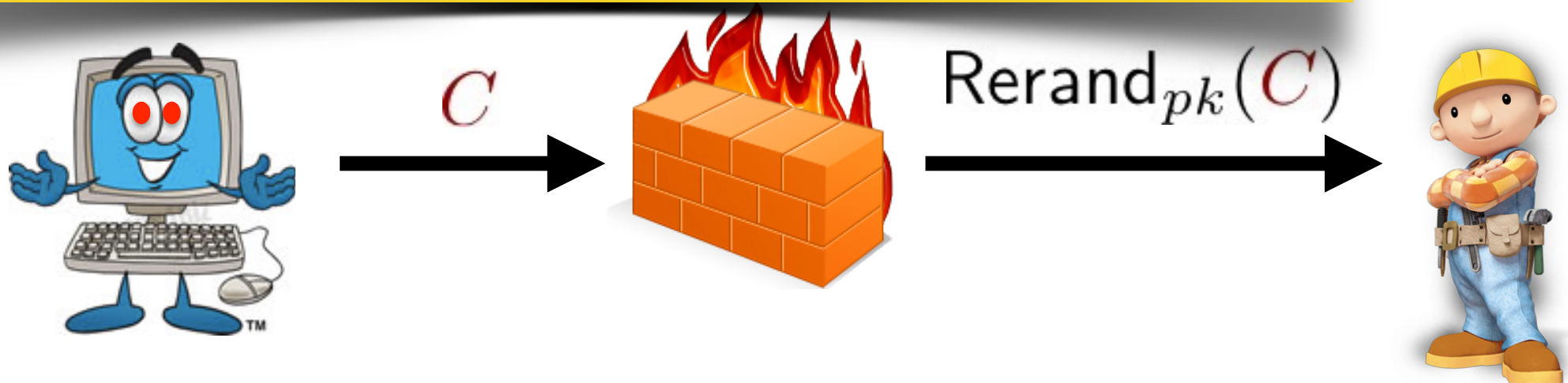
Functionality:

- Is the protocol functional without the firewall? ✓
- Is the protocol functional with the firewall? ✓

Security:

- Is the protocol semantically secure without firewall? ✓
- Is the protocol with the firewall semantically secure *regardless of how*  *behaves?*

Firewall:



Simple Example: Semantically Secure PKE

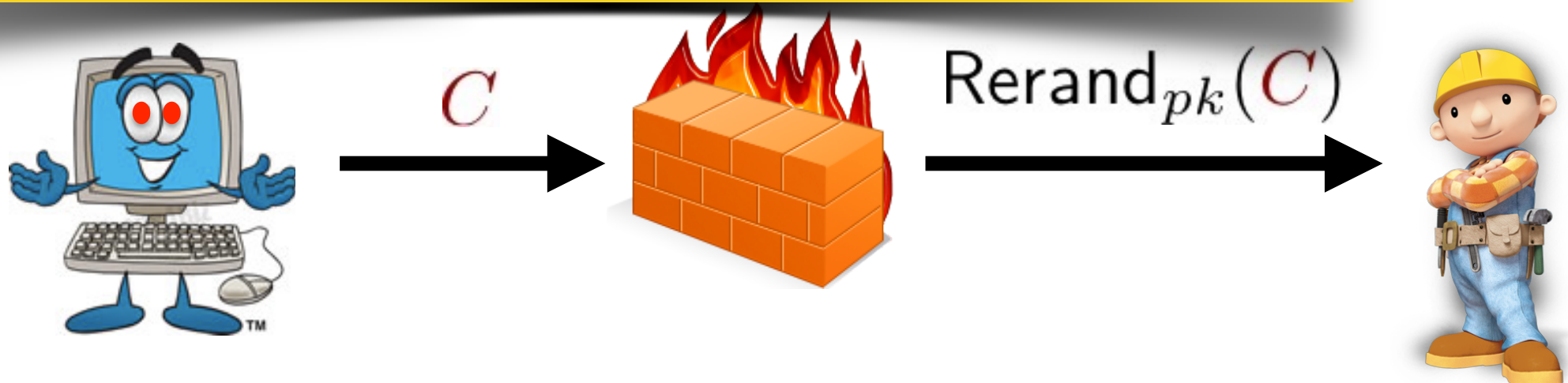
Functionality:

- Is the protocol functional without the firewall? ✓
- Is the protocol functional with the firewall? ✓

Security:

- Is the protocol semantically secure without firewall? ✓
- Is the protocol with the firewall semantically secure *regardless of how*  *behaves?* ✓

Firewall:



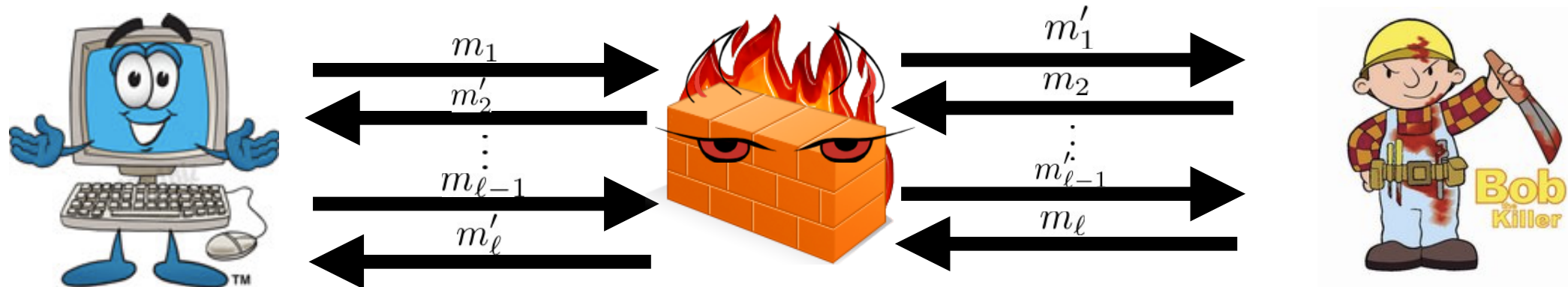
What if the firewall is corrupt?

What if the firewall is corrupt?

1) Honest Alice. Corrupt firewall.

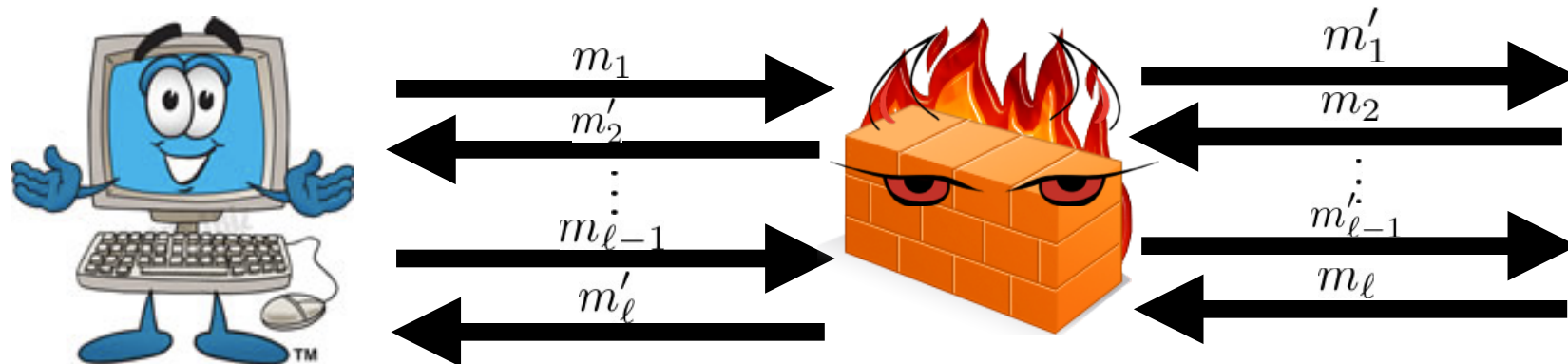
What if the firewall is corrupt?

1) **Honest** Alice. **Corrupt** firewall.



What if the firewall is corrupt?

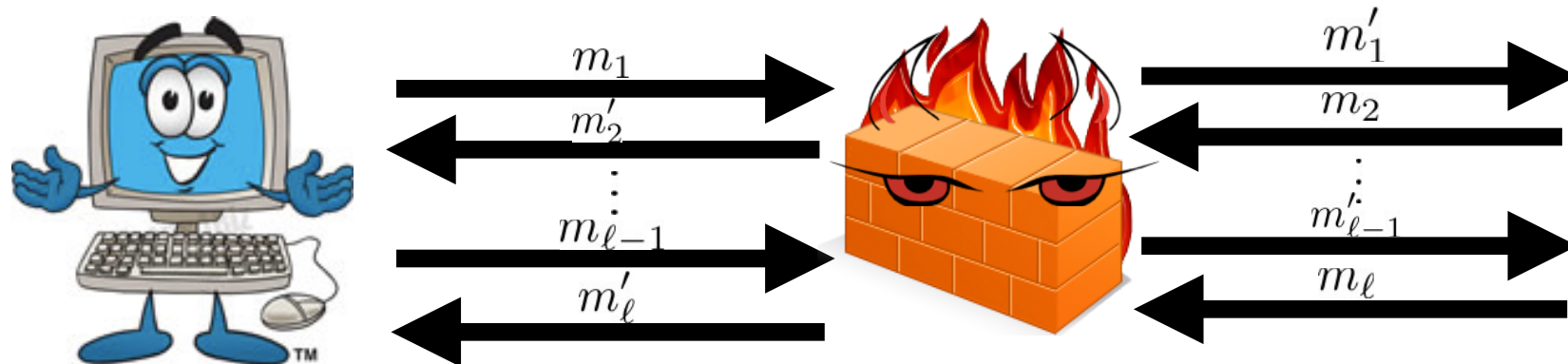
1) **Honest** Alice. **Corrupt** firewall.



(Security of the underlying protocol.)

What if the firewall is corrupt?

1) Honest Alice. Corrupt firewall.

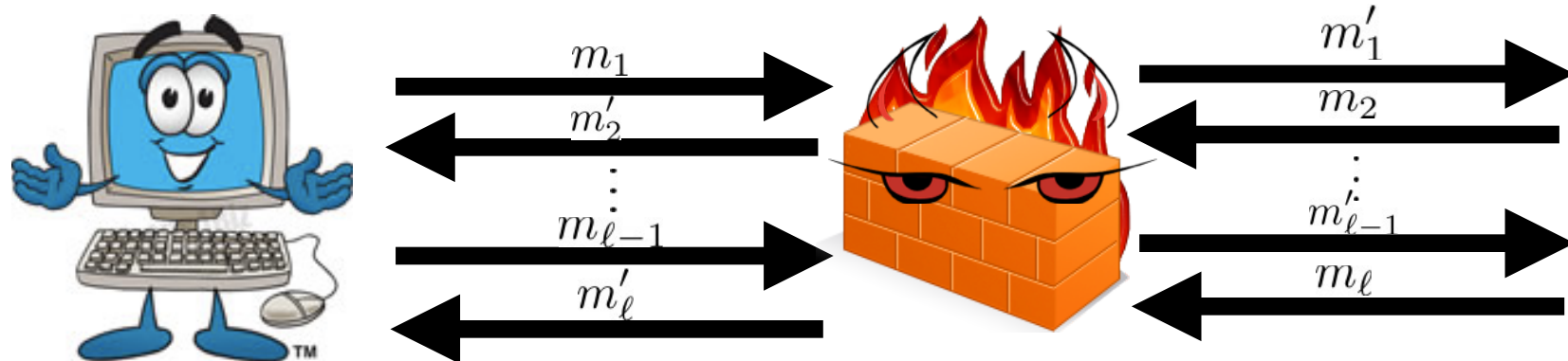


(Security of the underlying protocol.)

2) Corrupt Alice. Honest firewall.

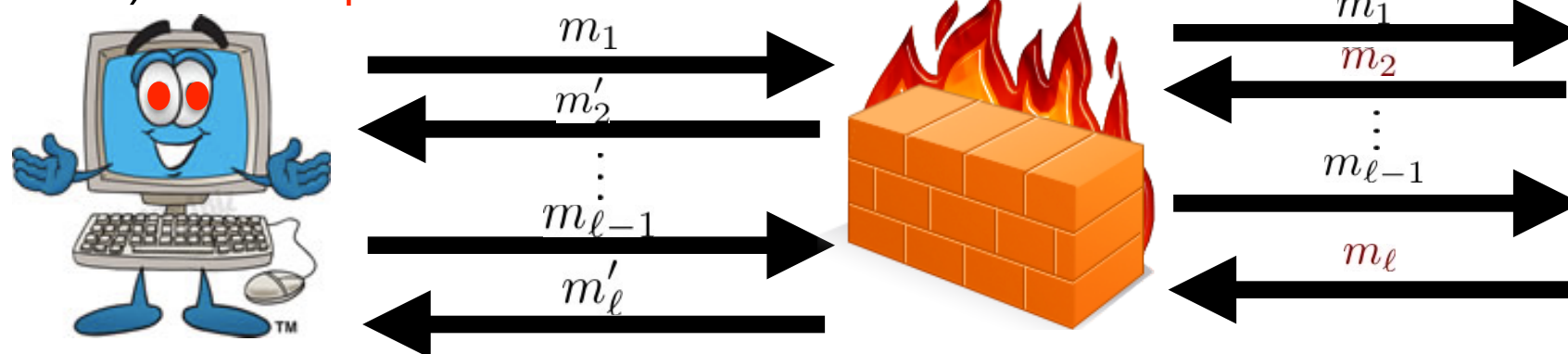
What if the firewall is corrupt?

1) Honest Alice. Corrupt firewall.



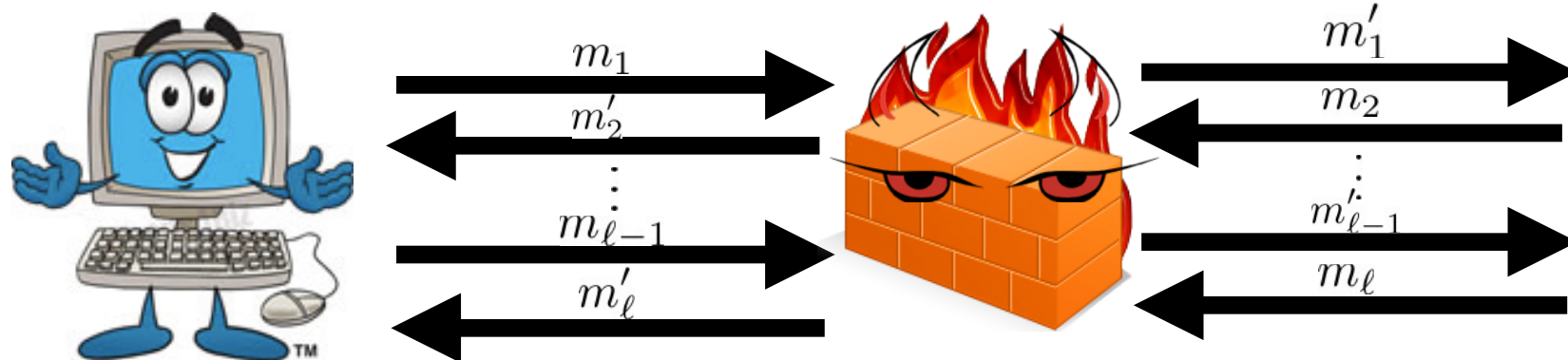
(Security of the underlying protocol.)

2) Corrupt Alice. Honest firewall.



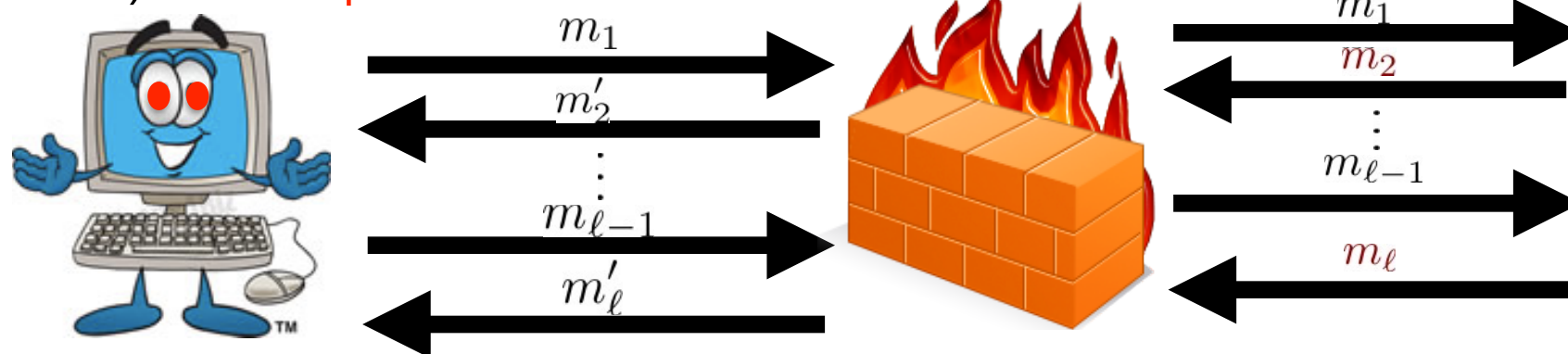
What if the firewall is corrupt?

1) Honest Alice. Corrupt firewall.



(Security of the underlying protocol.)

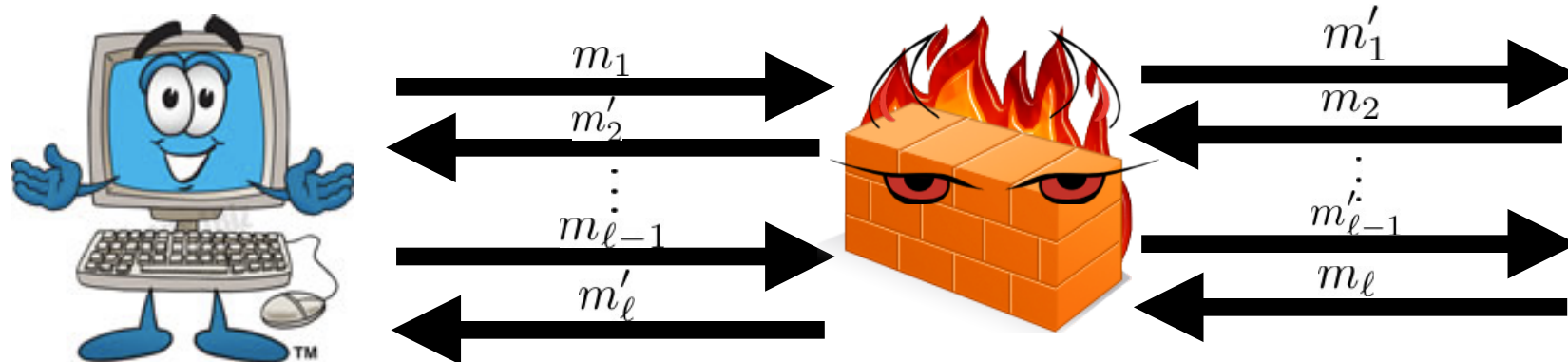
2) Corrupt Alice. Honest firewall.



(Security of firewall.)

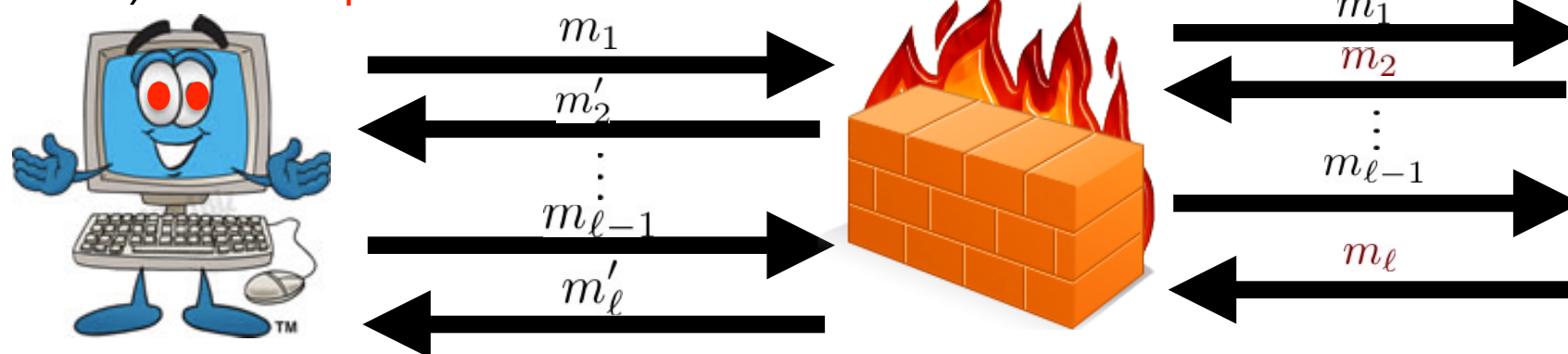
What if the firewall is corrupt?

1) Honest Alice. Corrupt firewall.



(Security of the underlying protocol.)

2) Corrupt Alice. Honest firewall.

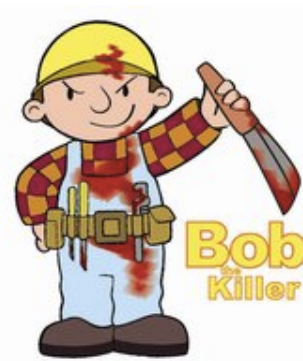
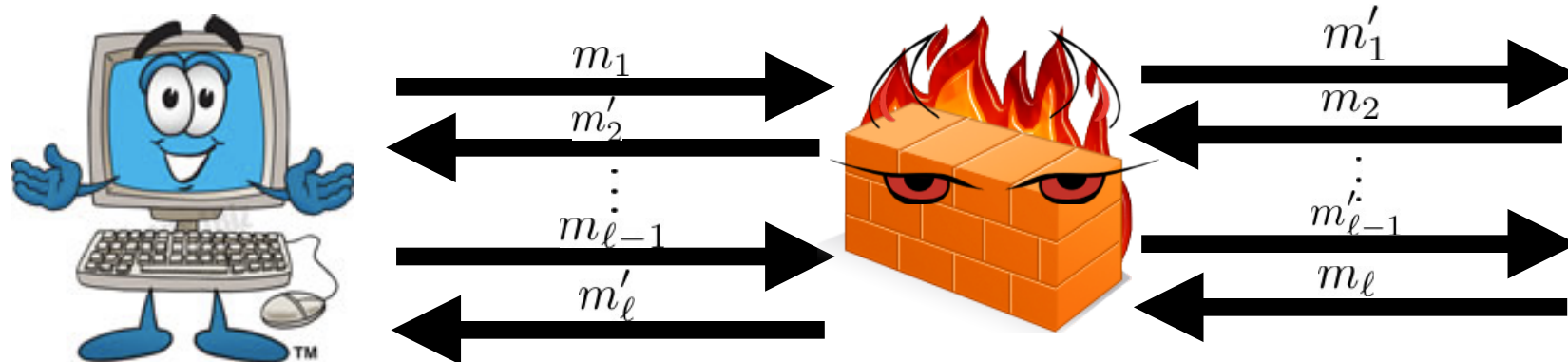


(Security of firewall.)

3) Corrupt Alice. Corrupt firewall.

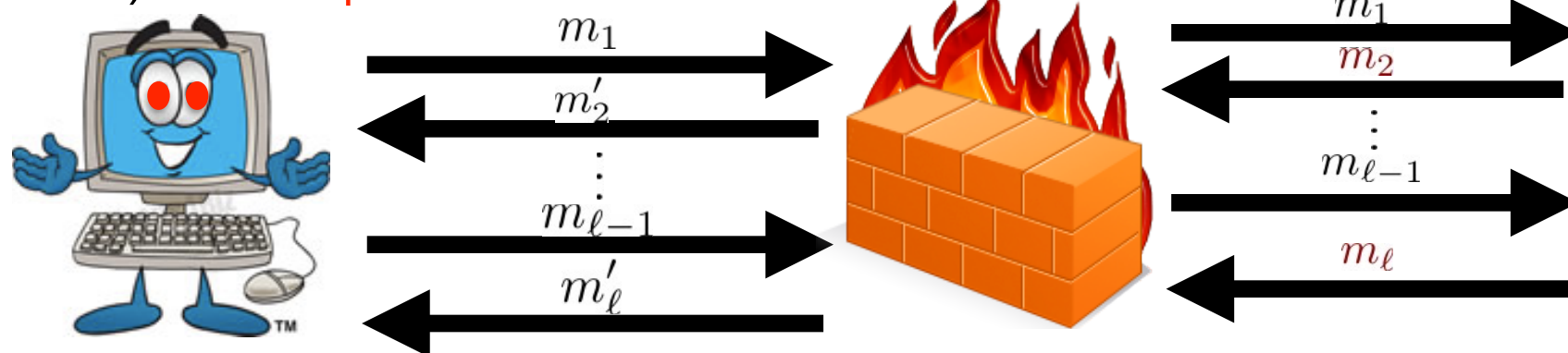
What if the firewall is corrupt?

1) Honest Alice. Corrupt firewall.



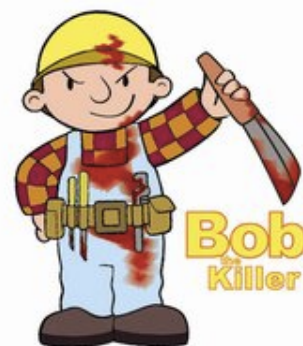
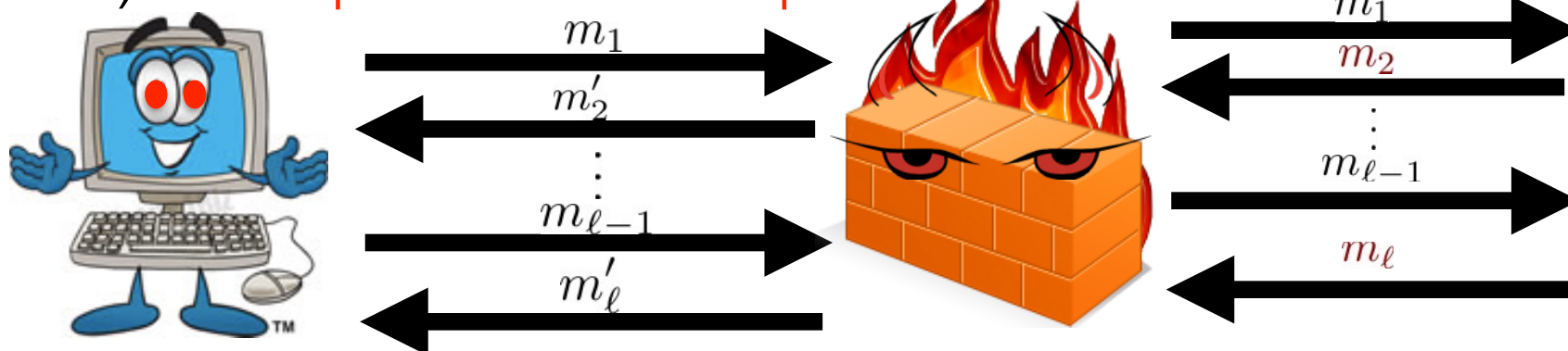
(Security of the underlying protocol.)

2) Corrupt Alice. Honest firewall.



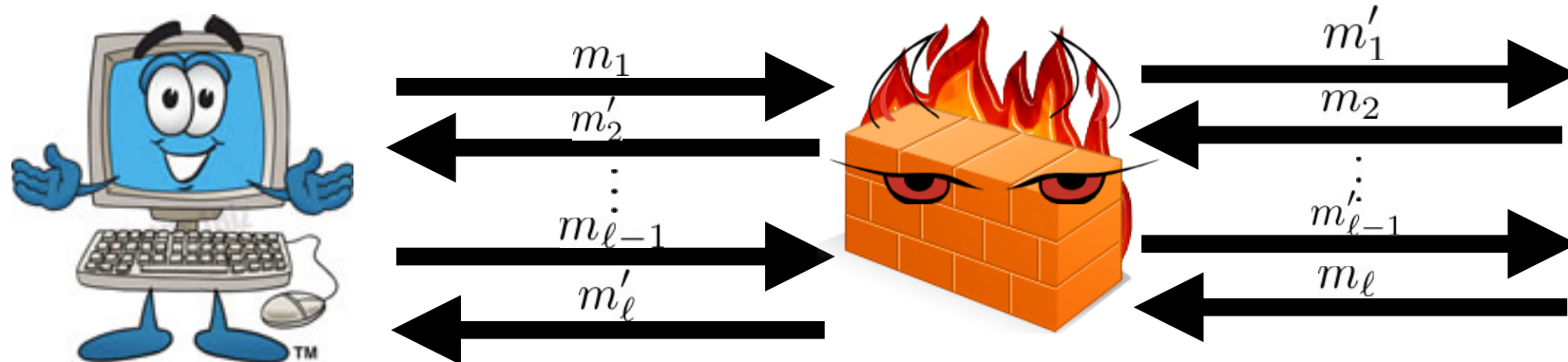
(Security of firewall.)

3) Corrupt Alice. Corrupt firewall.



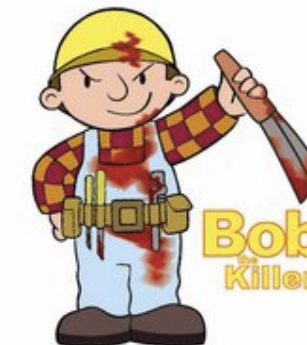
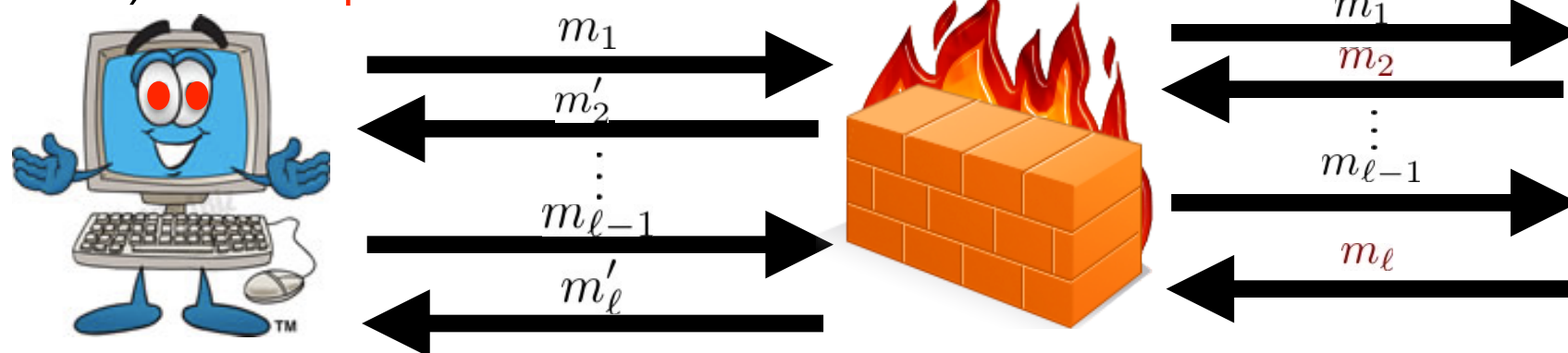
What if the firewall is corrupt?

1) Honest Alice. Corrupt firewall.



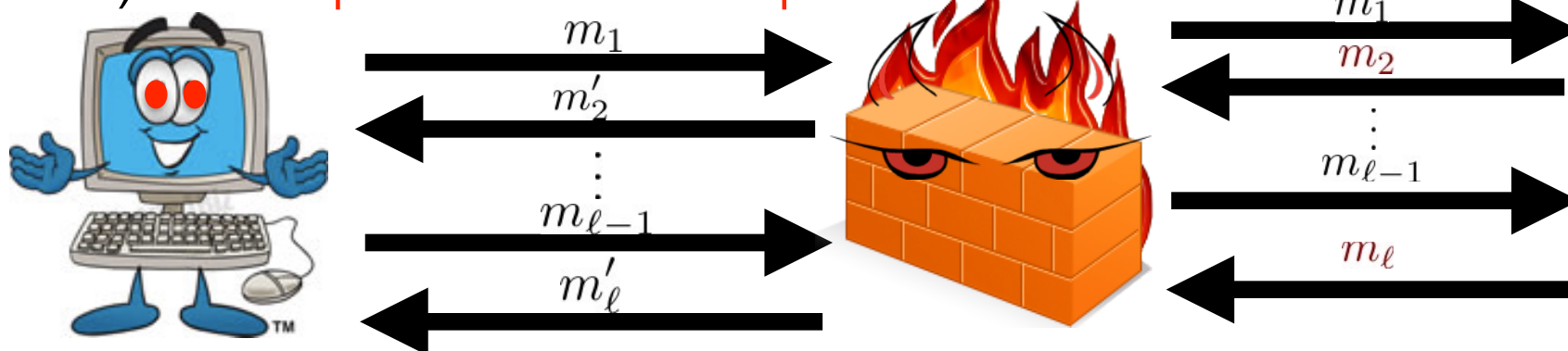
(Security of the underlying protocol.)

2) Corrupt Alice. Honest firewall.



(Security of firewall.)

3) Corrupt Alice. Corrupt firewall.



(The whole world is corrupt...)

What can we instantiate in
this crazy model?

Primitives with Reverse Firewalls

Primitives with Reverse Firewalls

- [Mironov, S '15]

Primitives with Reverse Firewalls

- [Mironov, S '15]
 - Oblivious transfer

Primitives with Reverse Firewalls

- [Mironov, S '15]
 - Oblivious transfer
 - Secure function evaluation

Primitives with Reverse Firewalls

- [Mironov, S '15]
 - Oblivious transfer
 - Secure function evaluation
 - “Exfiltration resistance” for arbitrary protocols

Primitives with Reverse Firewalls

- [Mironov, S '15]
 - Oblivious transfer
 - Secure function evaluation
 - “Exfiltration resistance” for arbitrary protocols
- [Dodis, Mironov, S '15]

Primitives with Reverse Firewalls

- [Mironov, S '15]
 - Oblivious transfer
 - Secure function evaluation
 - “Exfiltration resistance” for arbitrary protocols
- [Dodis, Mironov, S '15]
 - Message transmission in many different contexts

Primitives with Reverse Firewalls

- [Mironov, S '15]
 - Oblivious transfer
 - Secure function evaluation
 - “Exfiltration resistance” for arbitrary protocols
- [Dodis, Mironov, S '15]
 - Message transmission in many different contexts
 - Efficient CCA-secure scheme

Primitives with Reverse Firewalls

- [Mironov, S '15]
 - Oblivious transfer
 - Secure function evaluation
 - “Exfiltration resistance” for arbitrary protocols
- [Dodis, Mironov, S '15]
 - Message transmission in many different contexts
 - Efficient CCA-secure scheme
- [Ateniese, Magri, Venturi '15]

Primitives with Reverse Firewalls

- [Mironov, S '15]
 - Oblivious transfer
 - Secure function evaluation
 - “Exfiltration resistance” for arbitrary protocols
- [Dodis, Mironov, S '15]
 - Message transmission in many different contexts
 - Efficient CCA-secure scheme
- [Ateniese, Magri, Venturi '15]
 - Signatures

Act III: Message-Transmission Protocols

Message-Transmission Protocols

Message-Transmission Protocols



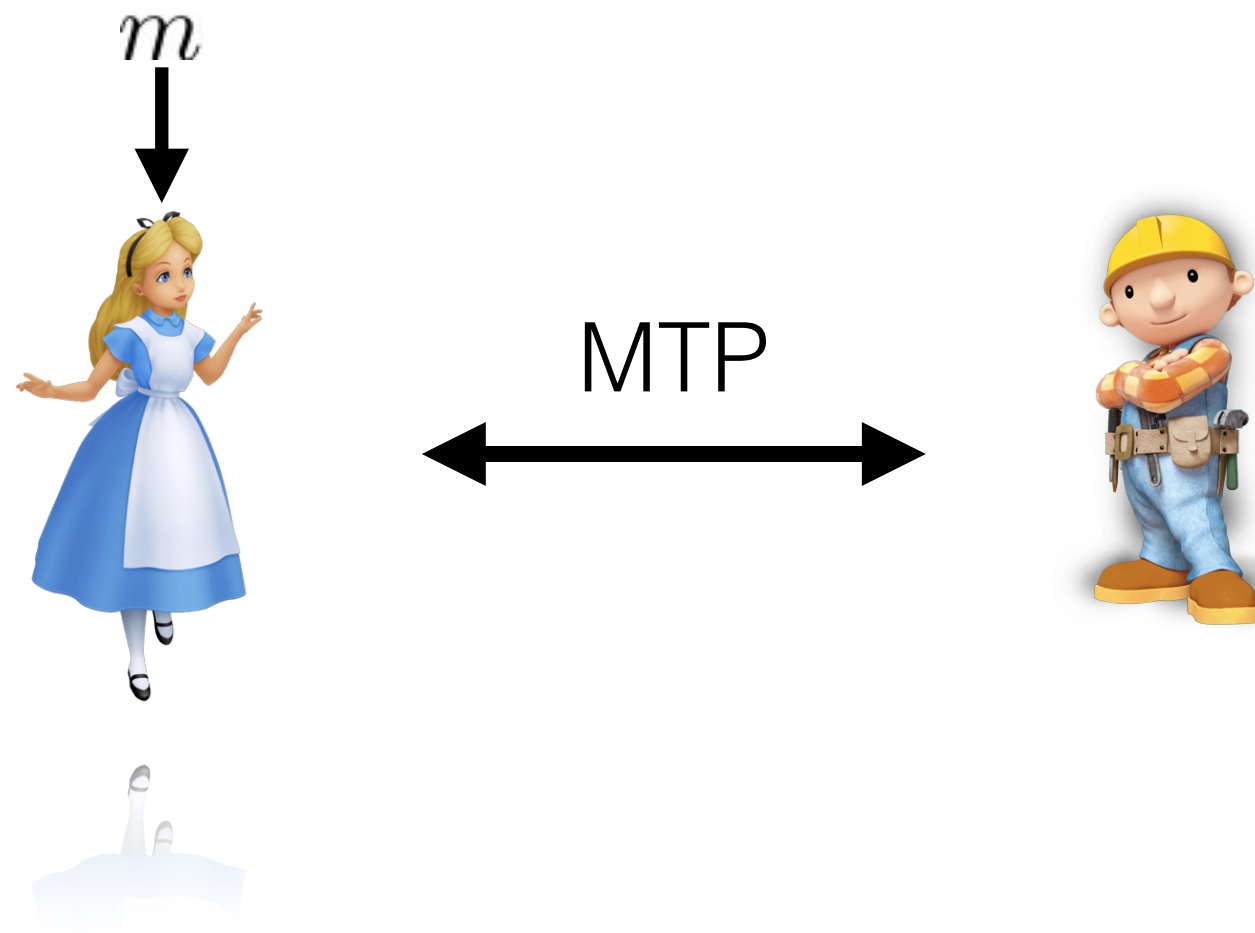
Message-Transmission Protocols



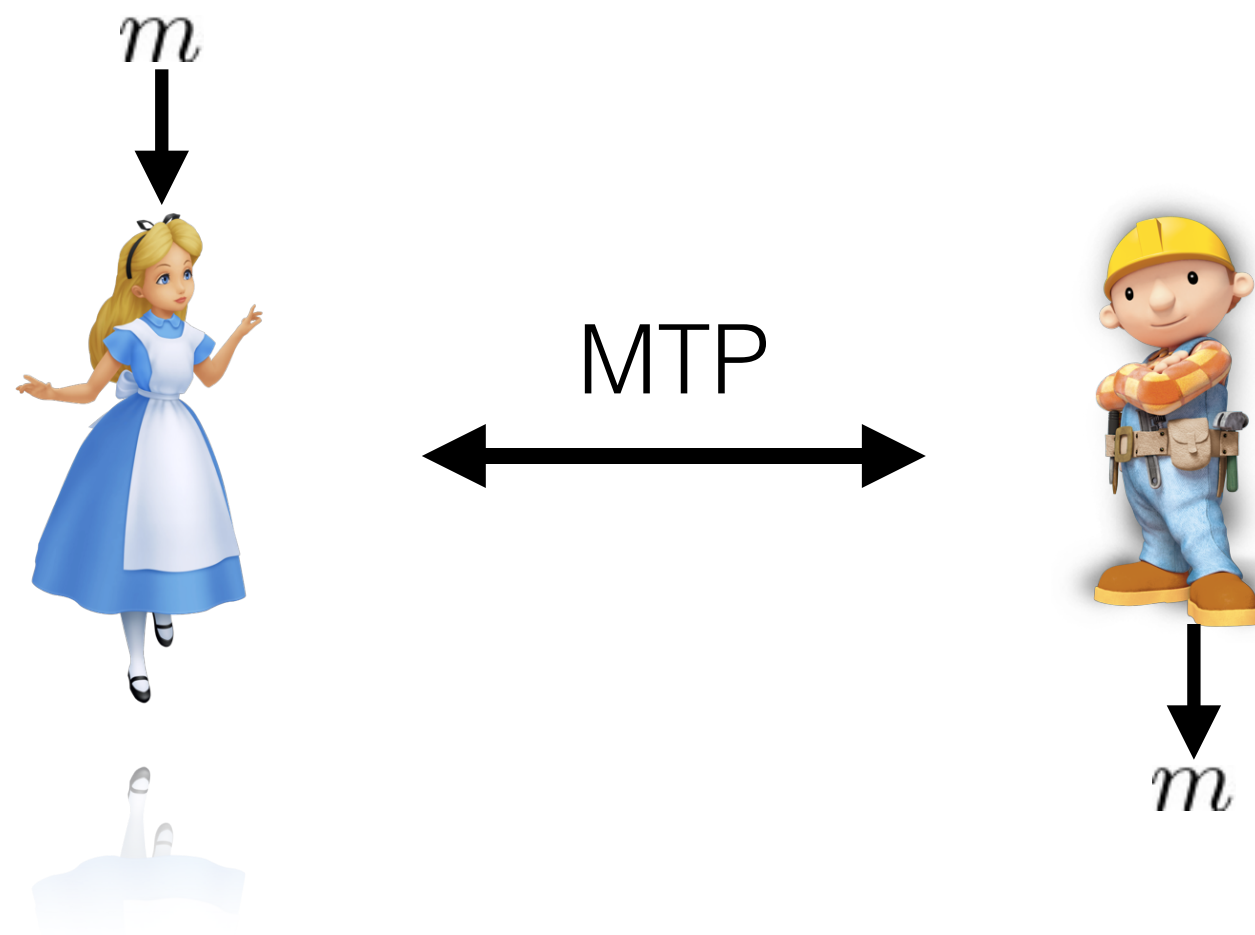
Message-Transmission Protocols



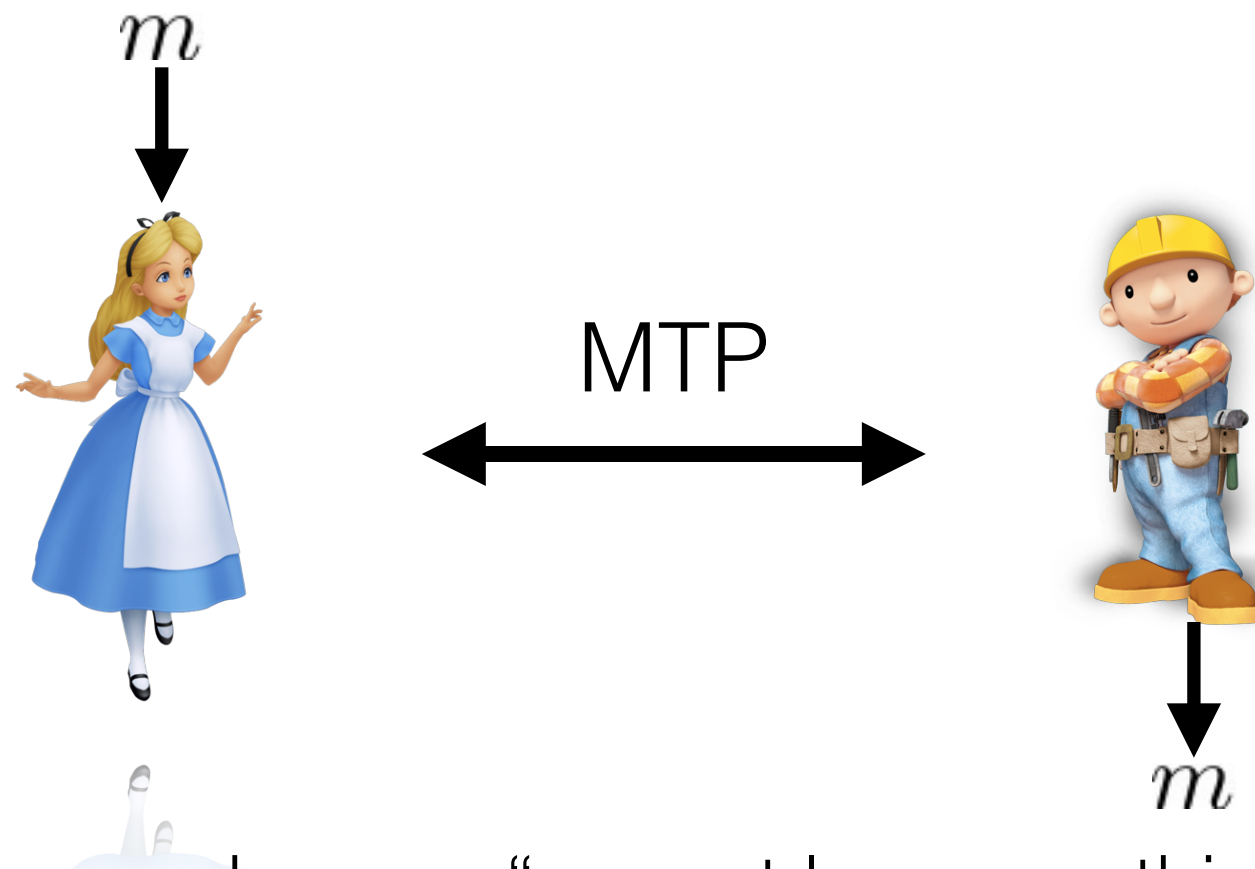
Message-Transmission Protocols



Message-Transmission Protocols

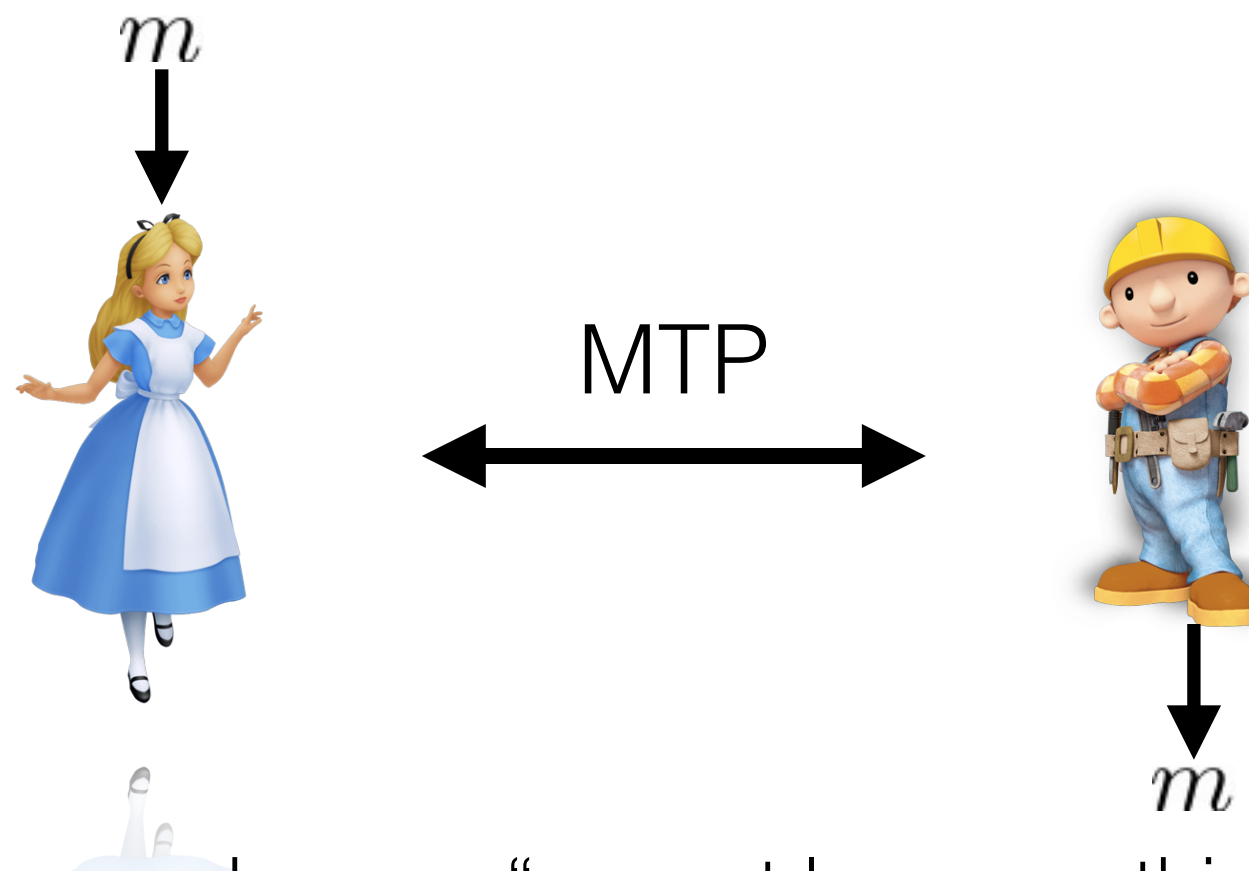


Message-Transmission Protocols



CPA Security: An eavesdropper “cannot learn anything about the plaintext m .”

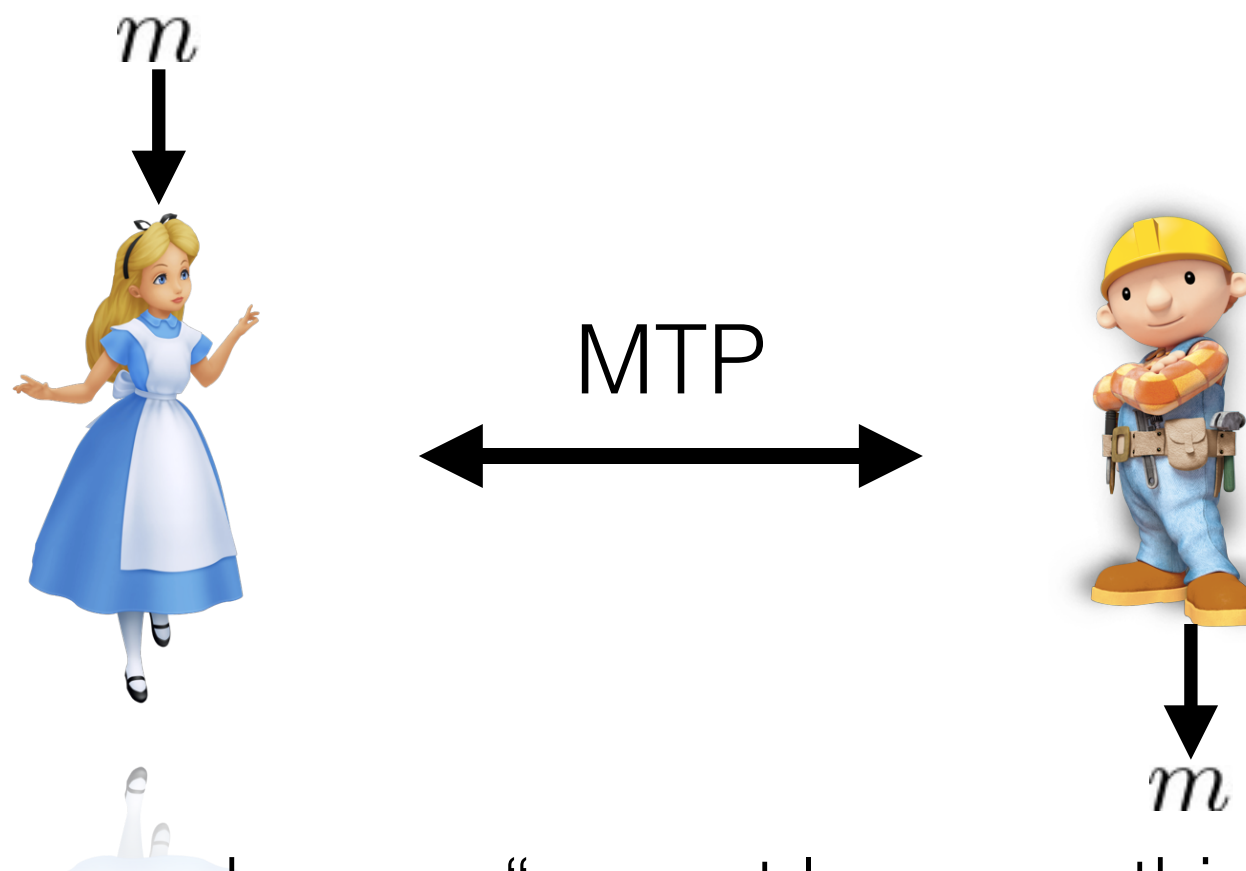
Message-Transmission Protocols



CPA Security: An eavesdropper “cannot learn anything about the plaintext m .”

CCA Security: An *active* adversary with access to a decryption oracle “cannot learn anything about the plaintext m .”

Message-Transmission Protocols



CPA Security: An eavesdropper “cannot learn anything about the plaintext m .”

CCA Security: An *active* adversary with access to a decryption oracle “cannot learn anything about the plaintext m .”

Forward Secrecy: Security holds even if “the adversary gets access to Alice and Bob’s secret keys later.”

Classical Message-Transmission Protocols

Classical Message-Transmission Protocols

	Key Infrastructure	Efficient?	CCA Secure?	Forward Secret?
--	-------------------------------	-------------------	------------------------	----------------------------

Classical Message-Transmission Protocols

	Key Infrastructure	Efficient?	CCA Secure?	Forward Secret?
Symmetric-key encryption	Shared Secret Key	Yes	Yes	No

Classical Message-Transmission Protocols

	Key Infrastructure	Efficient?	CCA Secure?	Forward Secret?
Symmetric-key encryption	Shared Secret Key	Yes	Yes	No
Public-key encryption	Public-key Infrastructure	No	Yes	No

Classical Message-Transmission Protocols

	Key Infrastructure	Efficient?	CCA Secure?	Forward Secret?
Symmetric-key encryption	Shared Secret Key	Yes	Yes	No
Public-key encryption	Public-key Infrastructure	No	Yes	No
Bob sends pk . Alice encrypts under pk .	None	No	No	Yes

Classical Message-Transmission Protocols

	Key Infrastructure	Efficient?	CCA Secure?	Forward Secret?
Symmetric-key encryption	Shared Secret Key	Yes	Yes	No
Public-key encryption	Public-key Infrastructure	No	Yes	No
Bob sends pk . Alice encrypts under pk .	None	No	No	Yes
KA + SKE	None	Yes	No	Yes

Classical Message-Transmission Protocols

	Key Infrastructure	Efficient?	CCA Secure?	Forward Secret?
Symmetric-key encryption	Shared Secret Key	Yes	Yes	No
Public-key encryption	Public-key Infrastructure	No	Yes	No
Bob sends pk . Alice encrypts under pk .	None	No	No	Yes
KA + SKE	None	Yes	No	Yes
AKA + SKE	Public-key Infrastructure	Yes	Yes	Yes

Classical Message-Transmission Protocols

	Key Infrastructure	Efficient?	CCA Secure?	Forward Secret?
Symmetric-key encryption	Shared Secret Key	Yes	Yes	No
Public-key encryption				No
Bob sends m to Alice Alice encrypts m under pk_B .				Yes
KA + SKE	None	Yes	No	Yes
AKA + SKE	Public-key Infrastructure	Yes	Yes	Yes

All of these results hold with reverse firewalls as well!

Composition Theorem (Informal)

$$\begin{array}{c} \text{KA w/ Firewall} \\ + \\ \text{SKE w/ Firewall} \\ = \\ \text{MTP w/ Firewall} \end{array}$$

Act IV: Key Agreement

Key Agreement

Key Agreement



Key Agreement



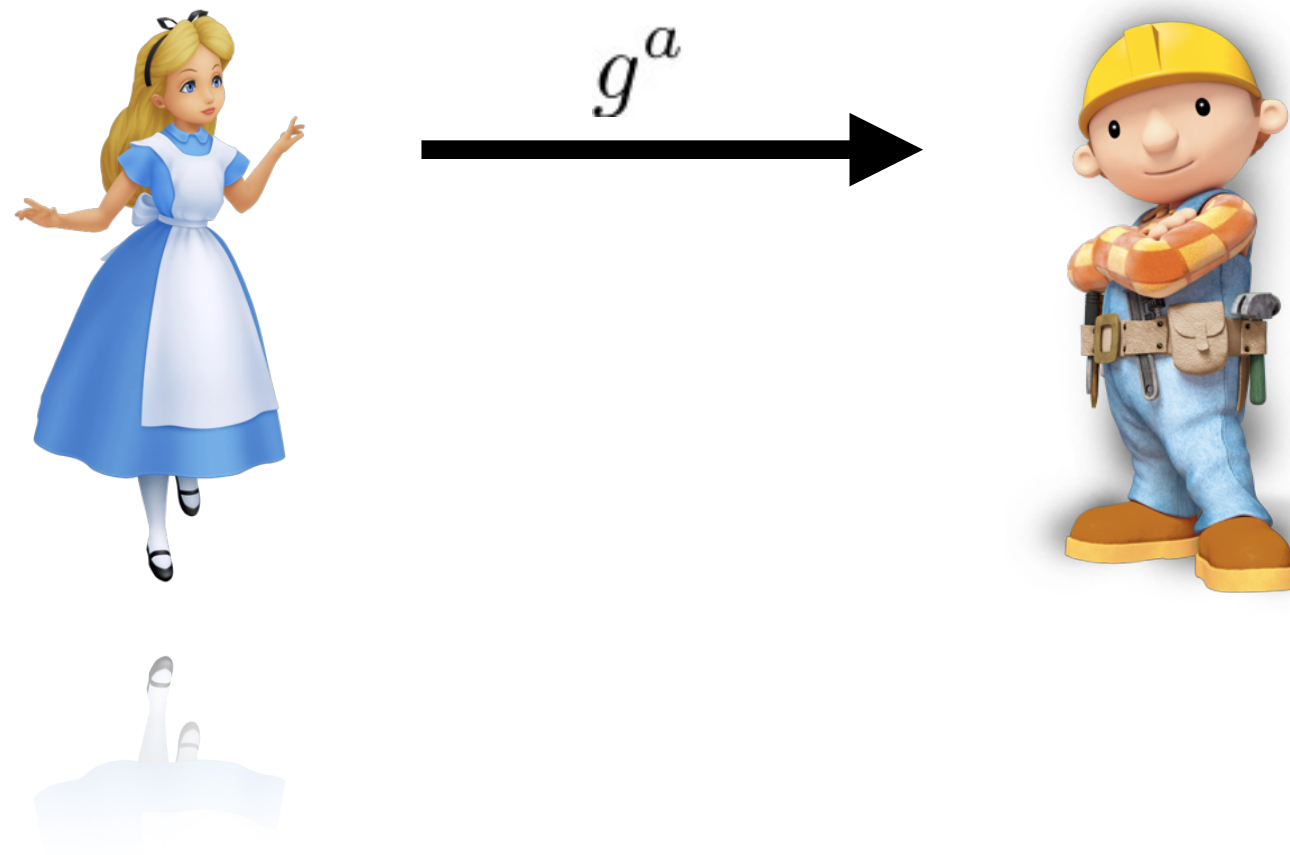
Key Agreement

$G =$ a group in which DDH is hard with generator g



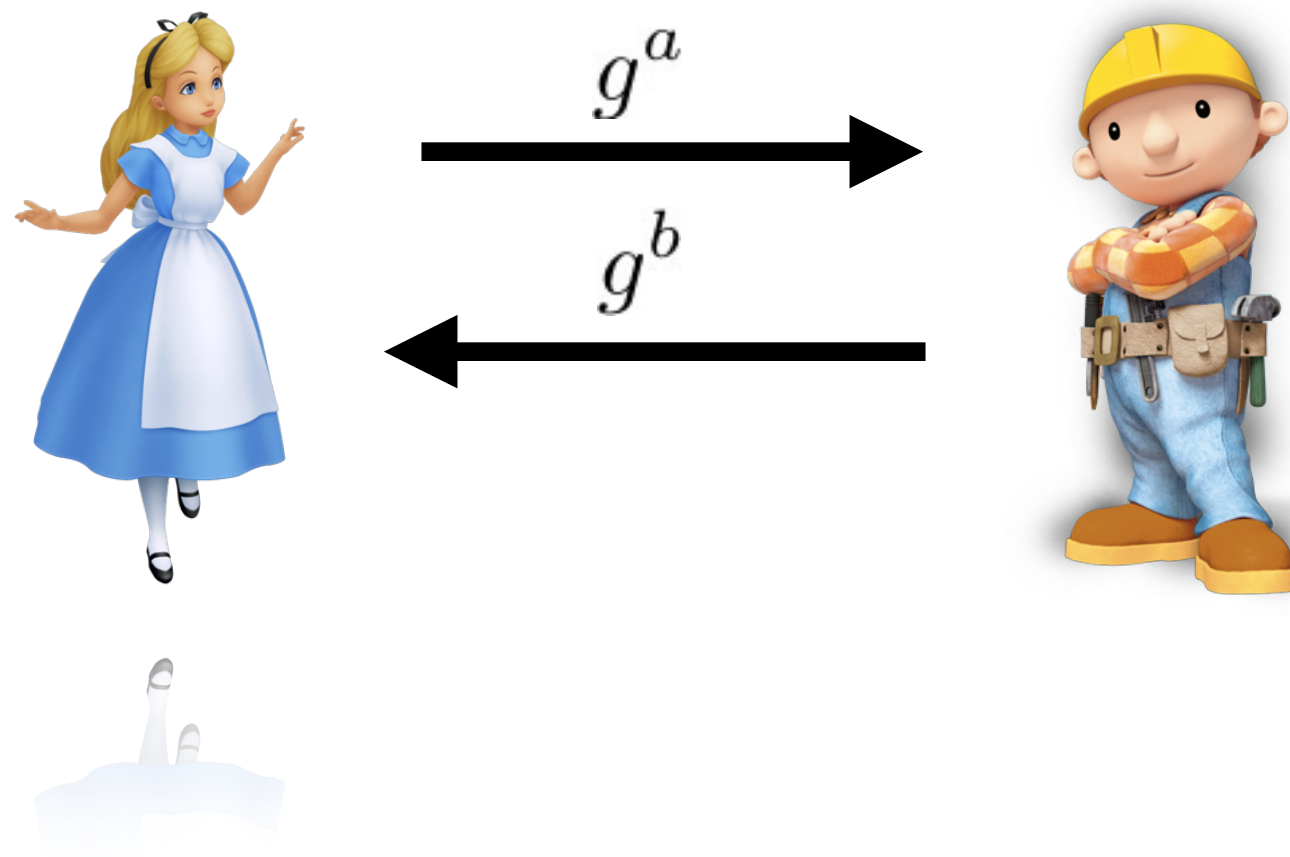
Key Agreement

$G =$ a group in which DDH is hard with generator g



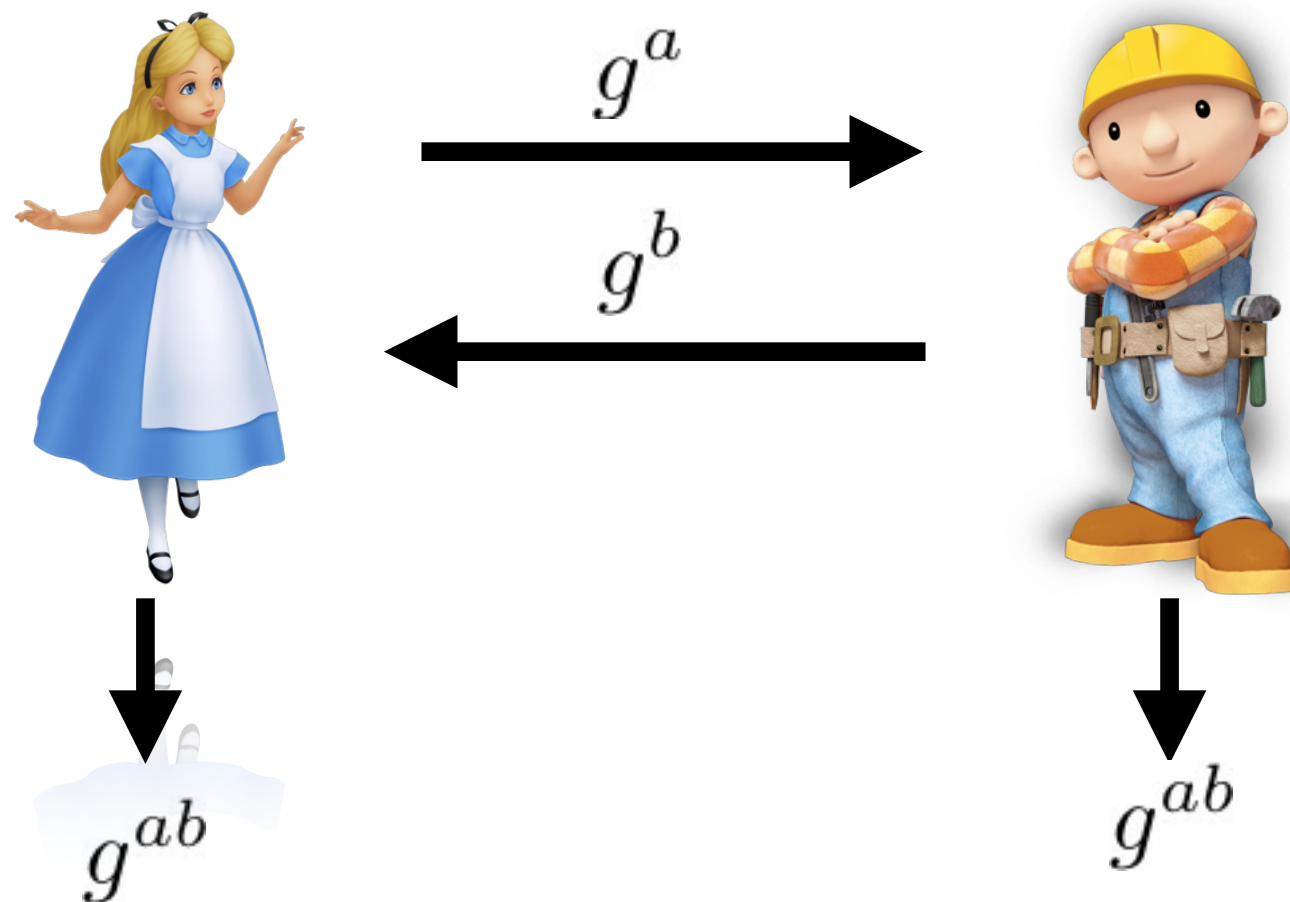
Key Agreement

$G =$ a group in which DDH is hard with generator g



Key Agreement

$G =$ a group in which DDH is hard with generator g



Key Agreement

$G =$ a group in which DDH is hard with generator g

Key Agreement

$G =$ a group in which DDH is hard with generator g



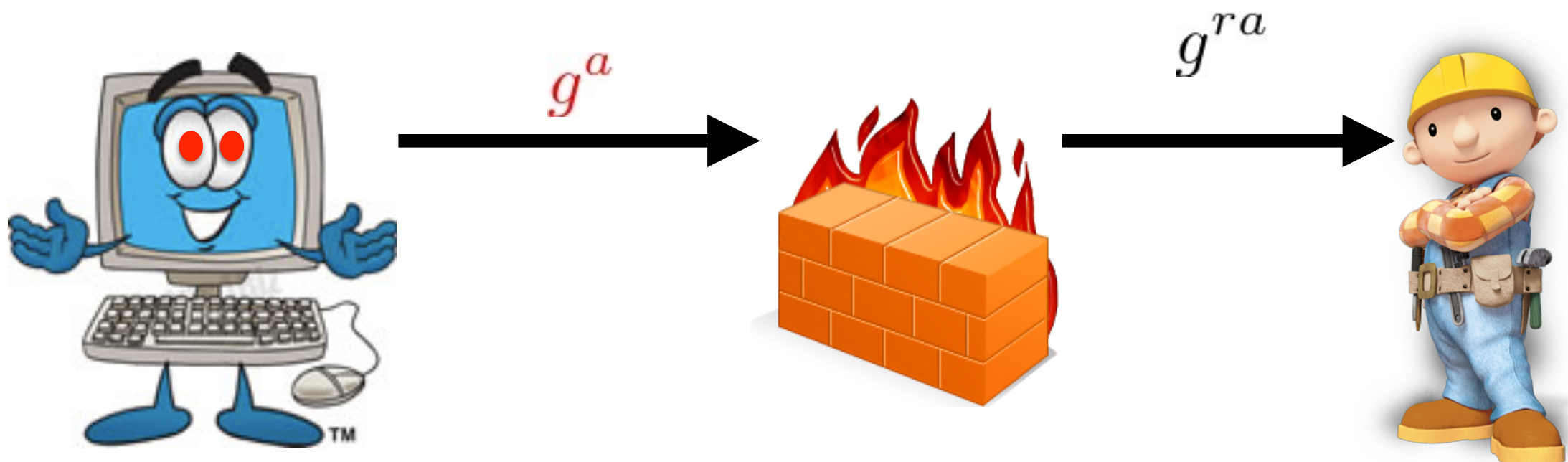
Key Agreement

$G =$ a group in which DDH is hard with generator g



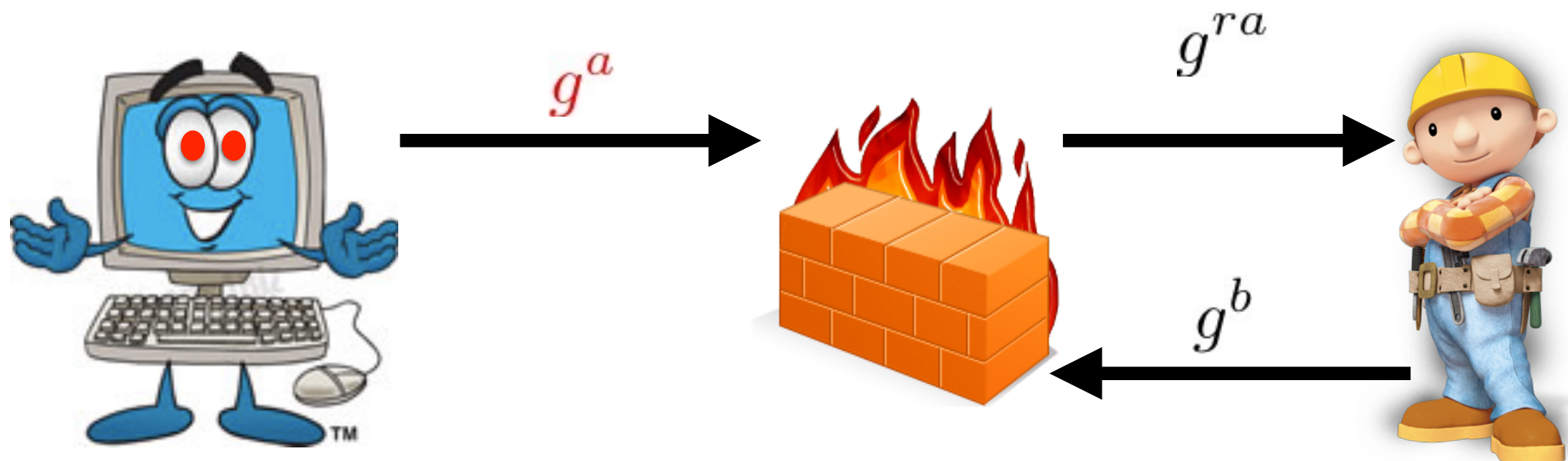
Key Agreement

$G =$ a group in which DDH is hard with generator g



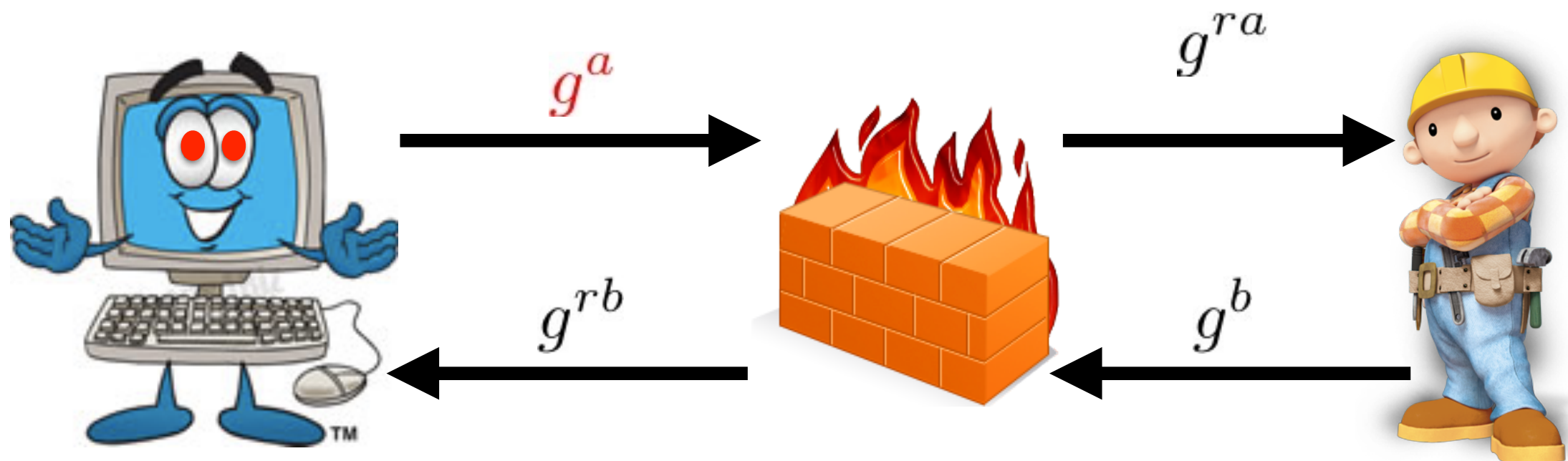
Key Agreement

$G =$ a group in which DDH is hard with generator g



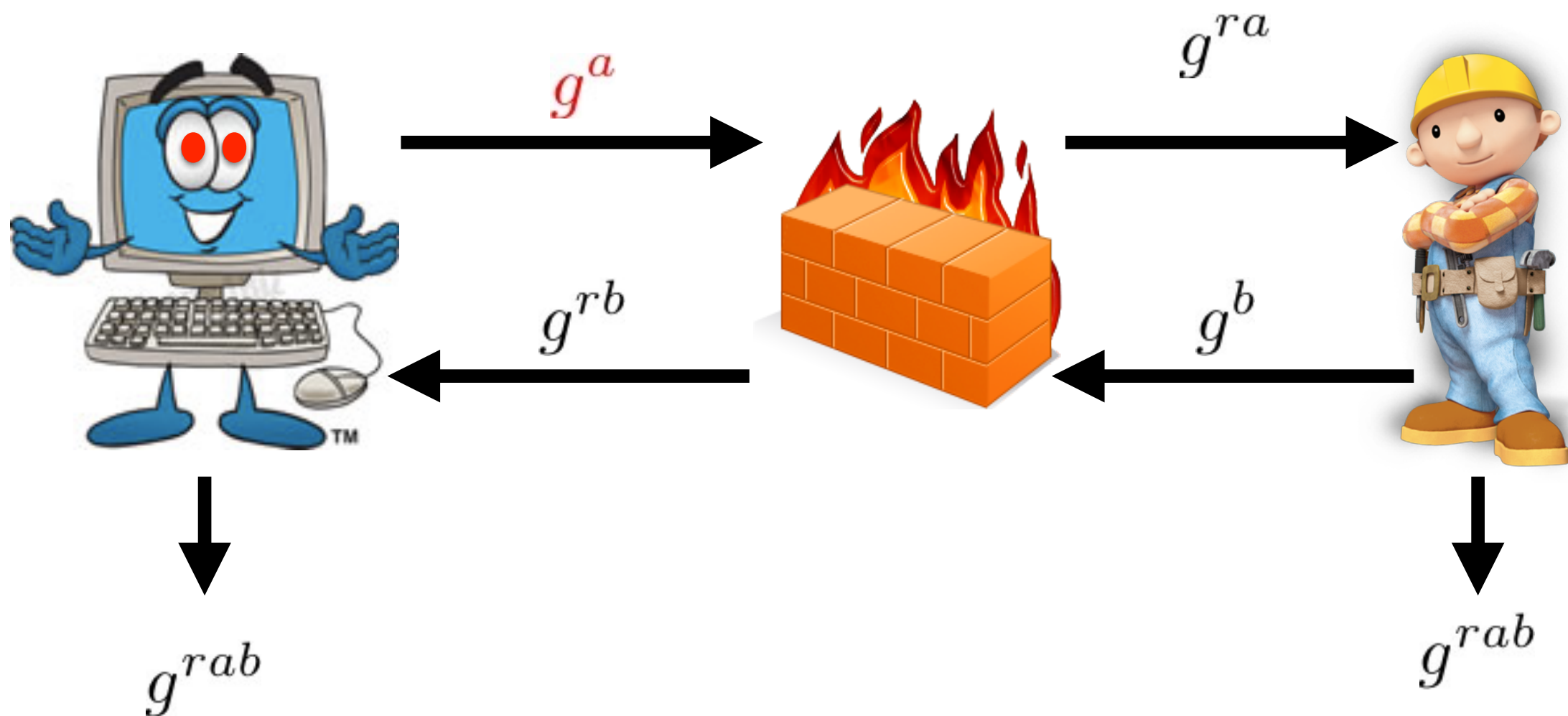
Key Agreement

$G =$ a group in which DDH is hard with generator g



Key Agreement

$G =$ a group in which DDH is hard with generator g



Authenticated Key Agreement

$G =$ a group in which DDH is hard with generator g

Authenticated Key Agreement

$G =$ a group in which DDH is hard with generator g



Authenticated Key Agreement

$G =$ a group in which DDH is hard with generator g



Authenticated Key Agreement

$G =$ a group in which DDH is hard with generator g

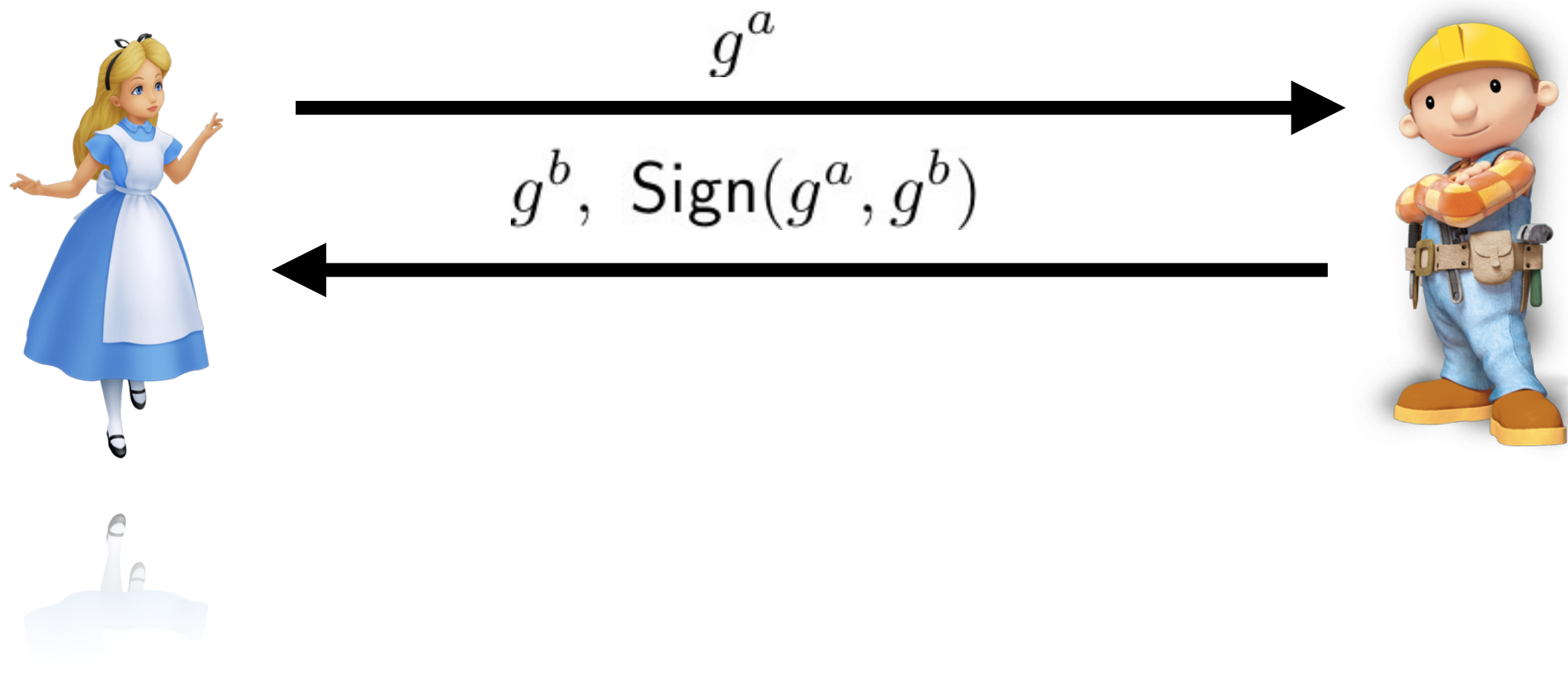


g^a



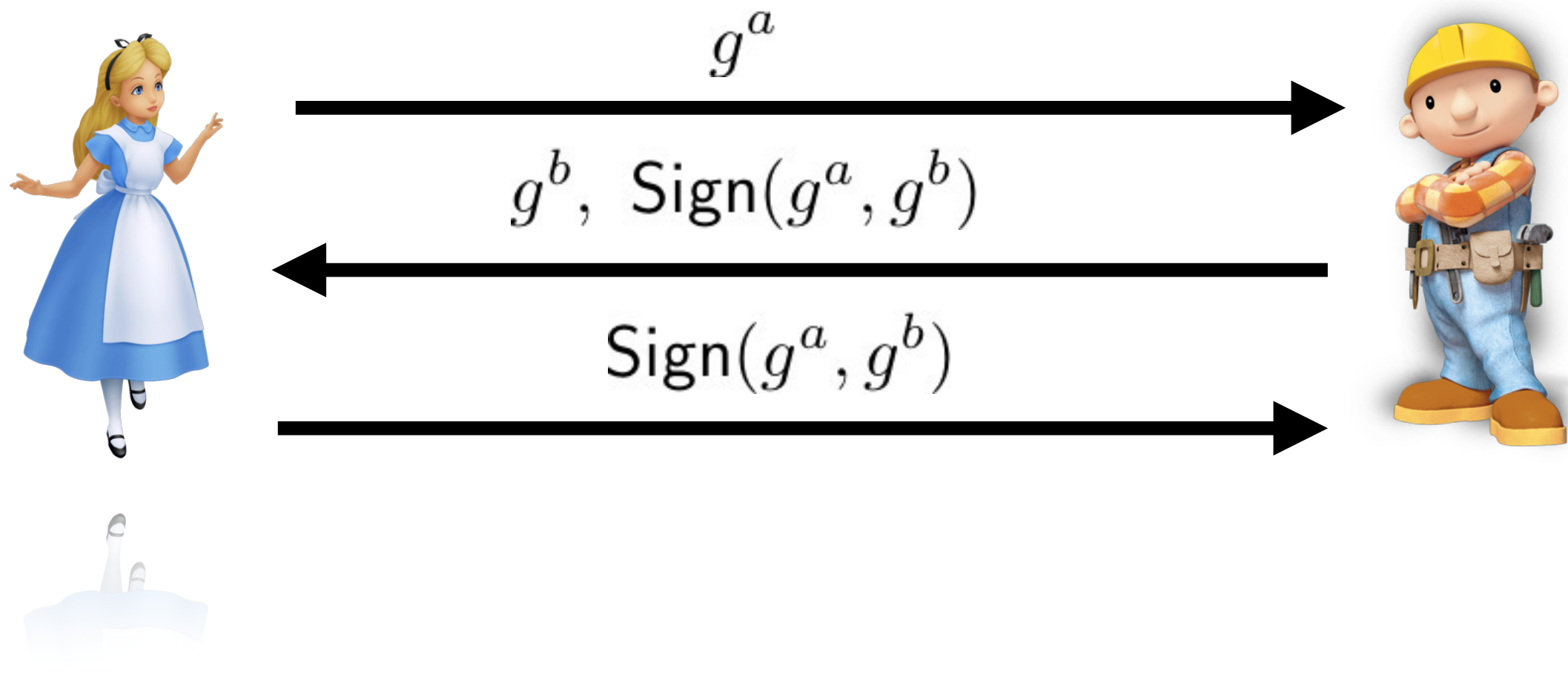
Authenticated Key Agreement

$G =$ a group in which DDH is hard with generator g



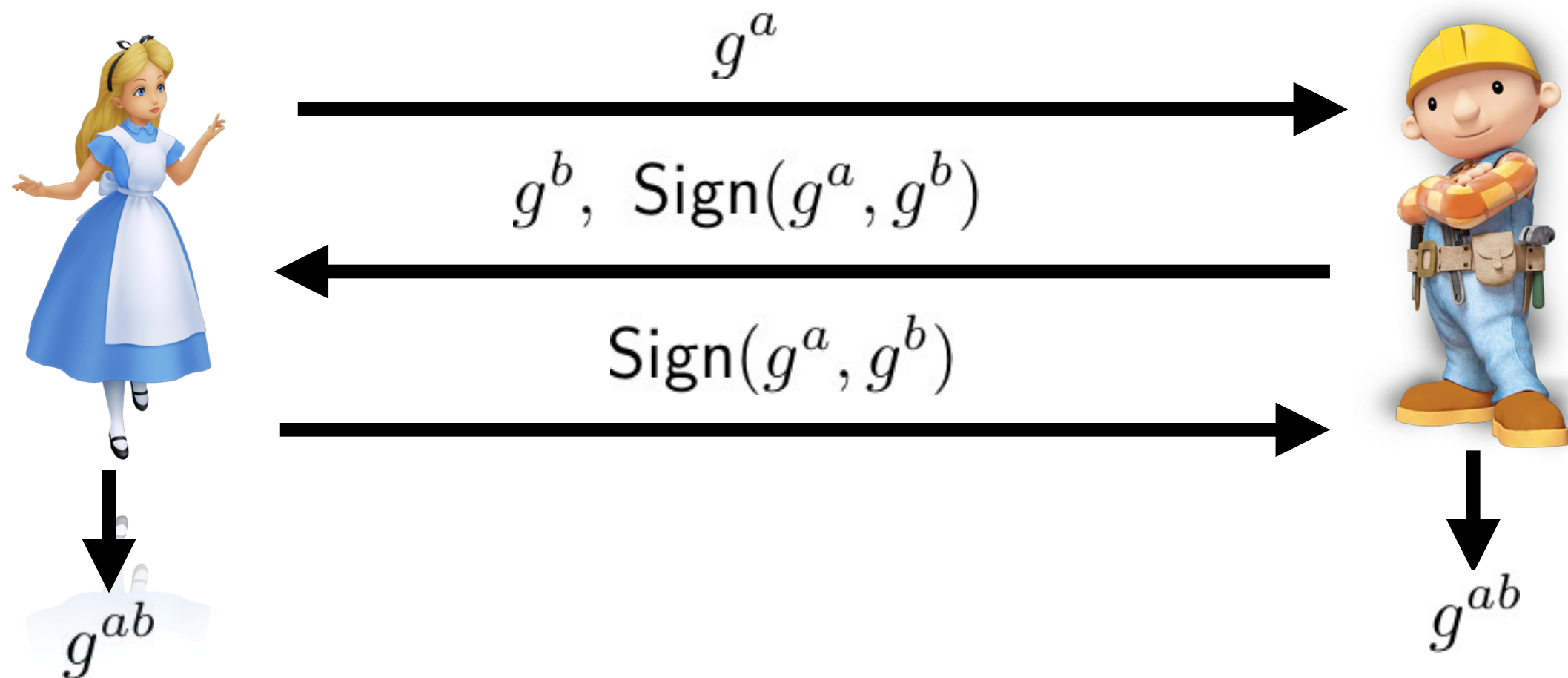
Authenticated Key Agreement

$G =$ a group in which DDH is hard with generator g



Authenticated Key Agreement

$G =$ a group in which DDH is hard with generator g



Authenticated Key Agreement

$G =$ a group in which DDH is hard with generator g

Authenticated Key Agreement

$G =$ a group in which DDH is hard with generator g



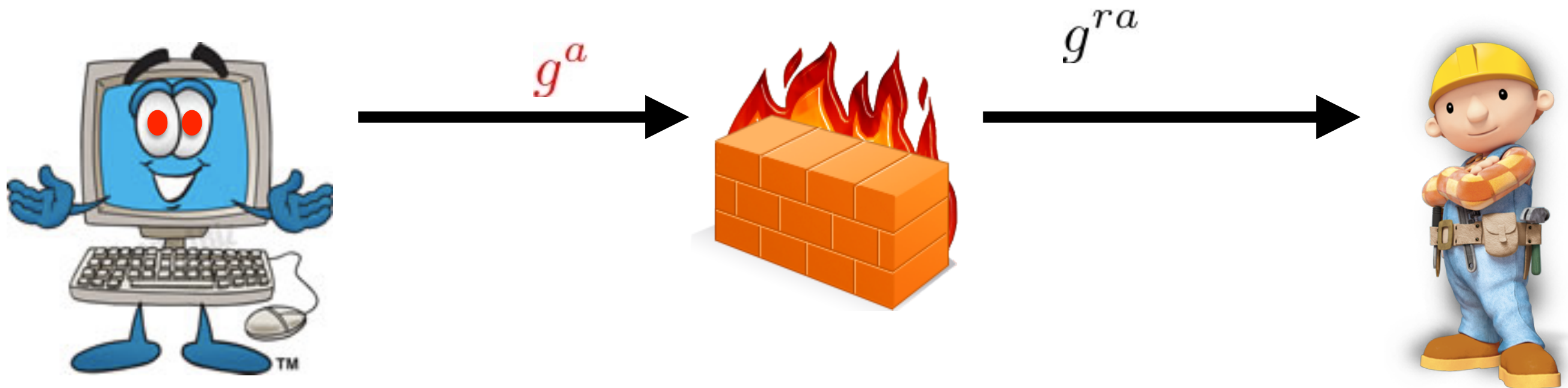
Authenticated Key Agreement

$G =$ a group in which DDH is hard with generator g



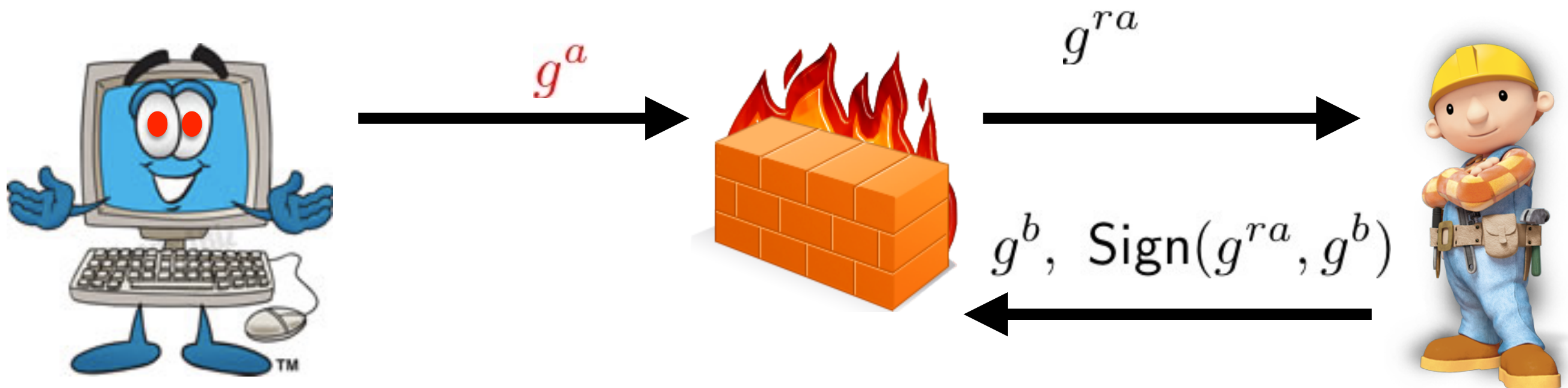
Authenticated Key Agreement

$G =$ a group in which DDH is hard with generator g



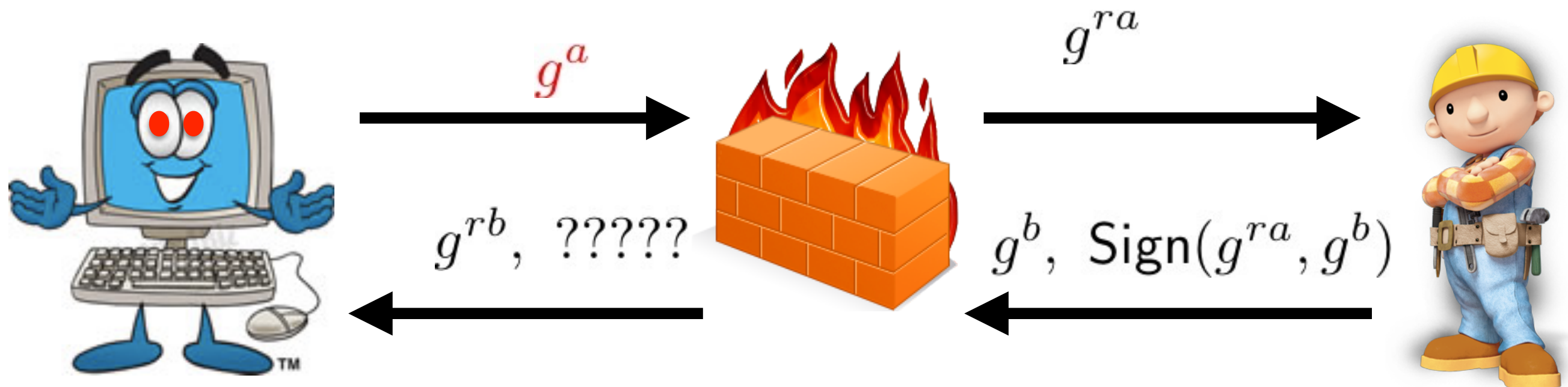
Authenticated Key Agreement

$G =$ a group in which DDH is hard with generator g



Authenticated Key Agreement

$G =$ a group in which DDH is hard with generator g



First Attempt

$G =$ a group in which DDH is hard with generator g

First Attempt

$G =$ a group in which DDH is hard with generator g



First Attempt

$G =$ a group in which DDH is hard with generator g



First Attempt

$G =$ a group in which DDH is hard with generator g

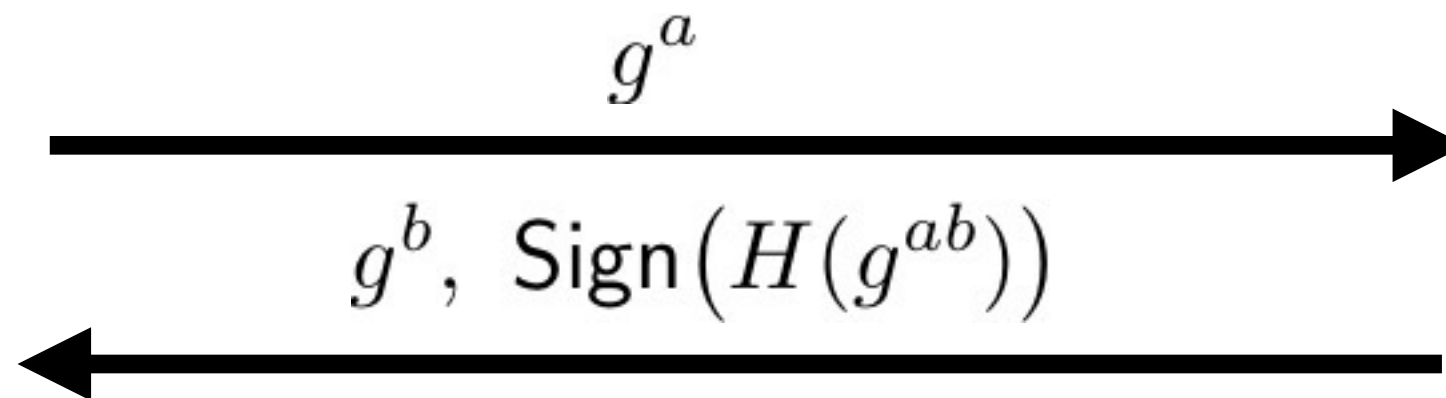


g^a



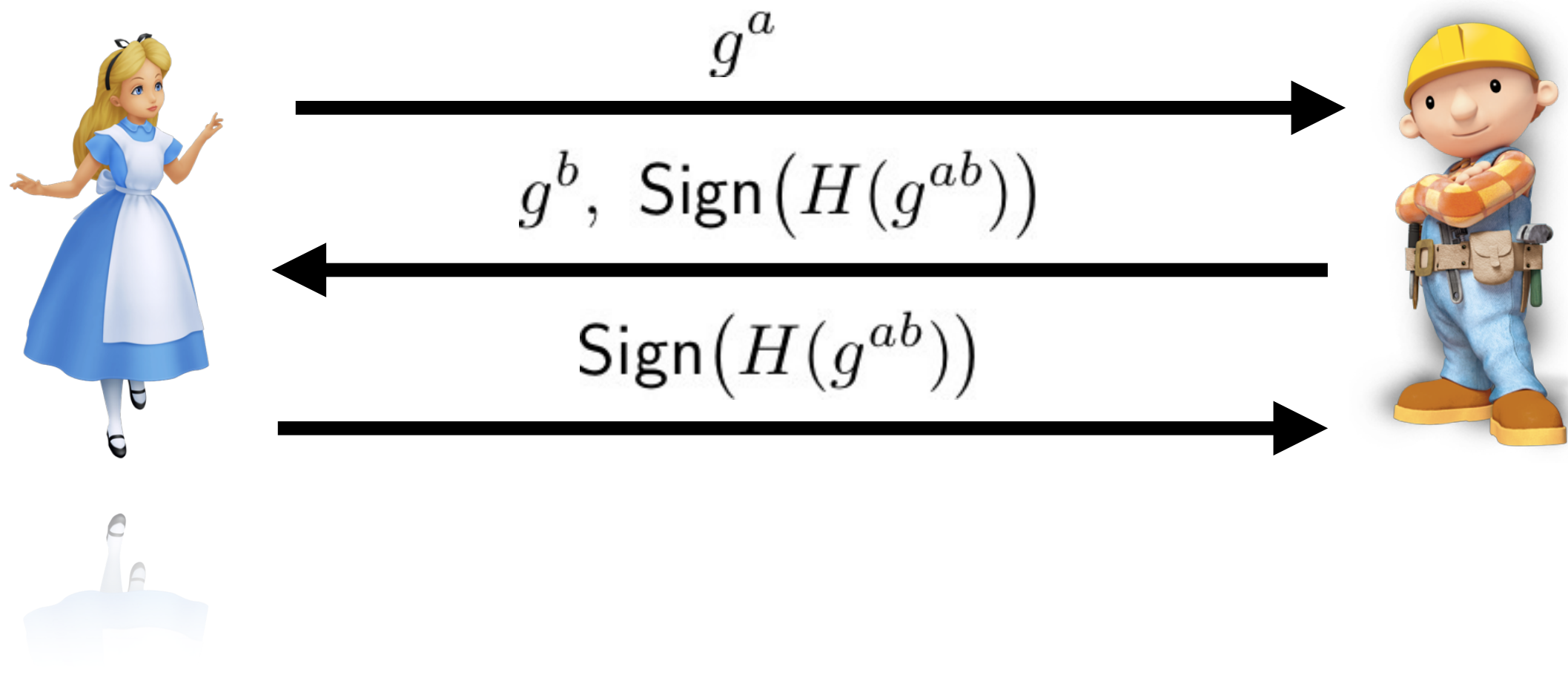
First Attempt

$G =$ a group in which DDH is hard with generator g



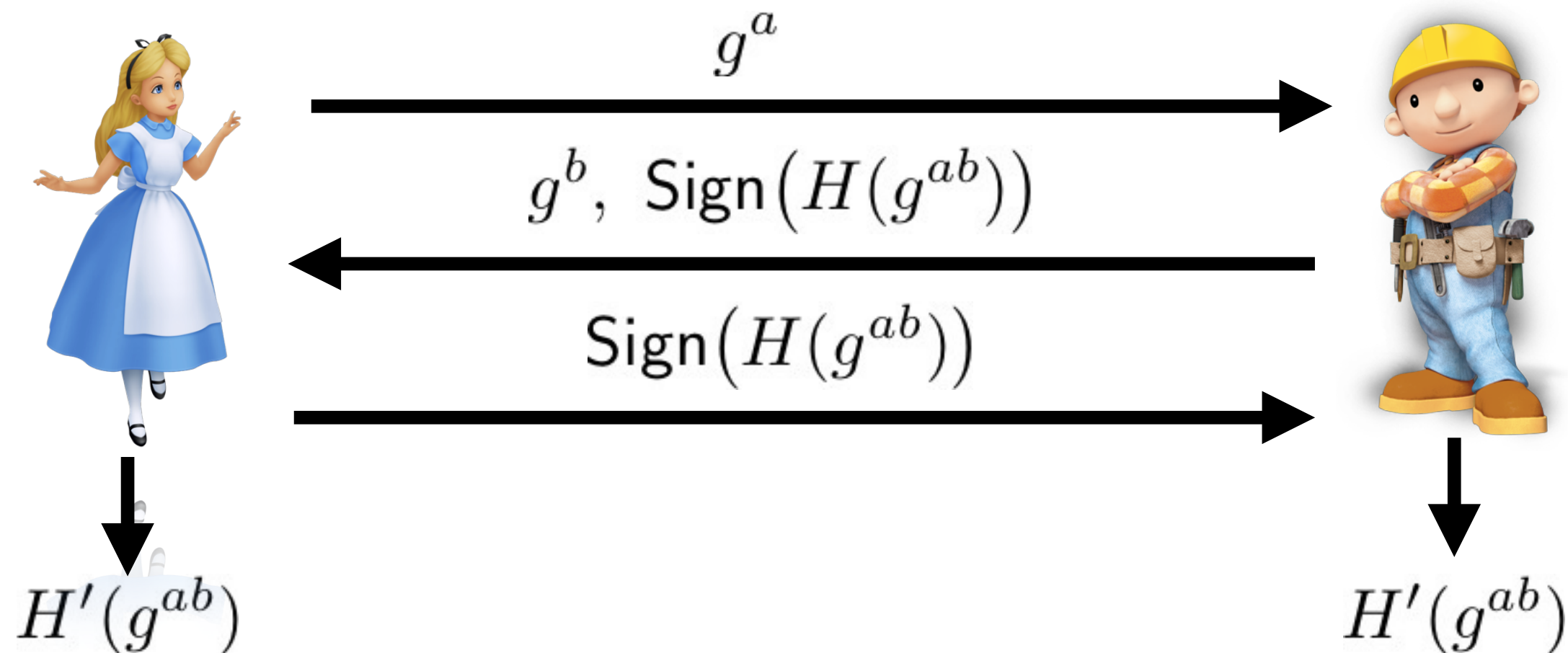
First Attempt

$G =$ a group in which DDH is hard with generator g



First Attempt

$G =$ a group in which DDH is hard with generator g



First Attempt

$G =$ a group in which DDH is hard with generator g

First Attempt

$G =$ a group in which DDH is hard with generator g



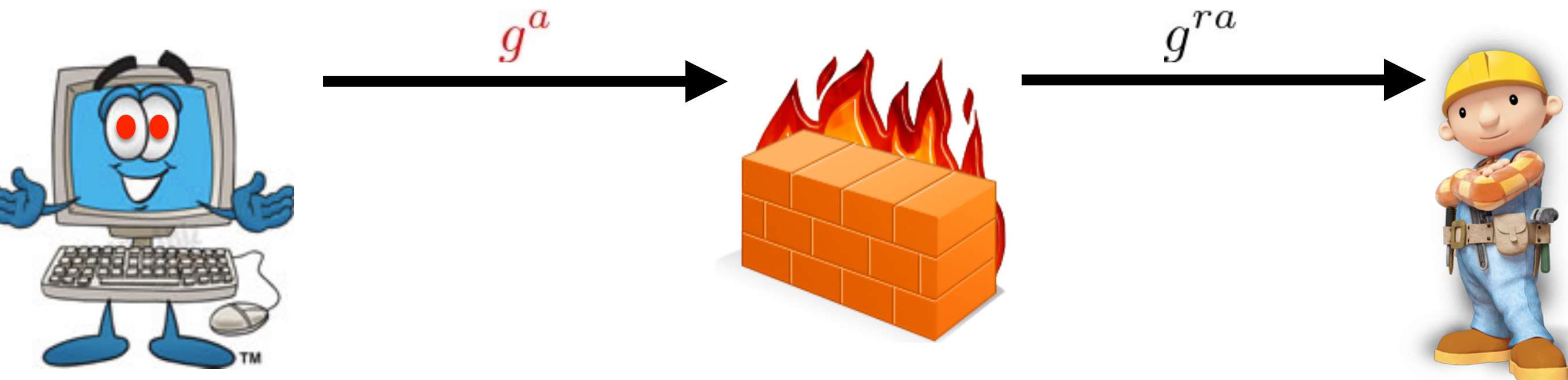
First Attempt

$G =$ a group in which DDH is hard with generator g



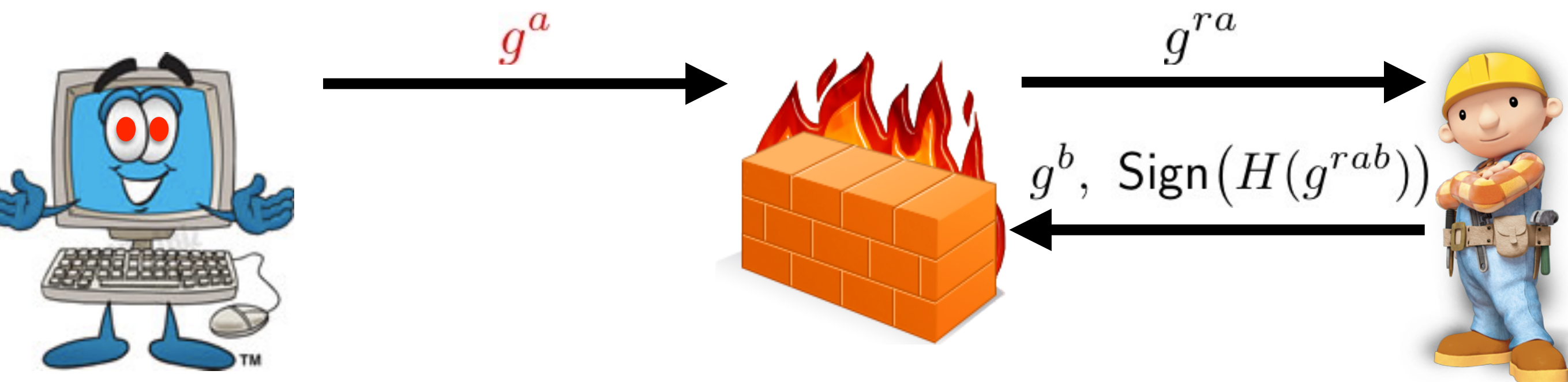
First Attempt

$G =$ a group in which DDH is hard with generator g



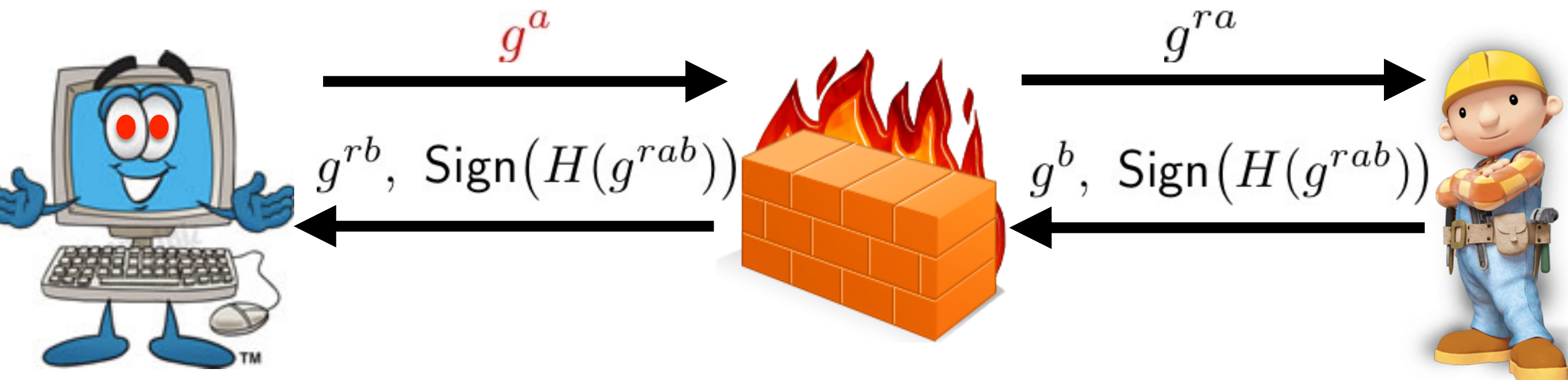
First Attempt

$G =$ a group in which DDH is hard with generator g



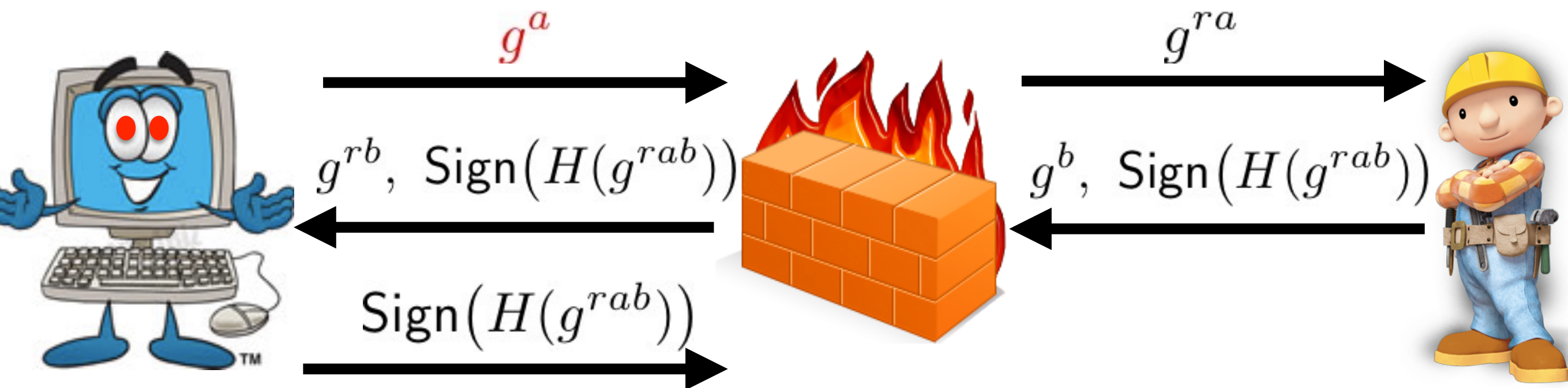
First Attempt

$G =$ a group in which DDH is hard with generator g



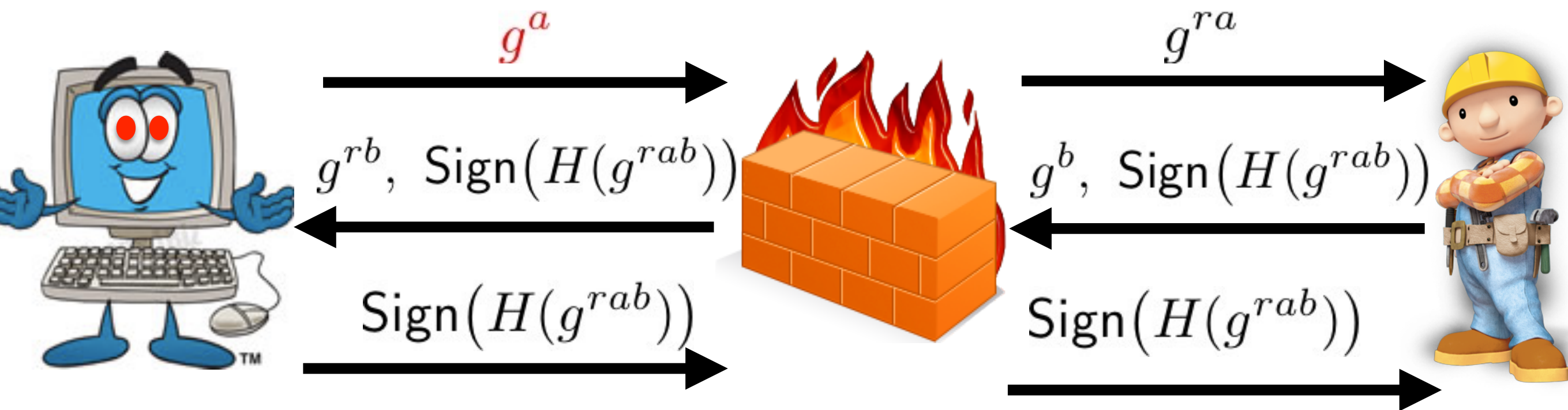
First Attempt

$G =$ a group in which DDH is hard with generator g



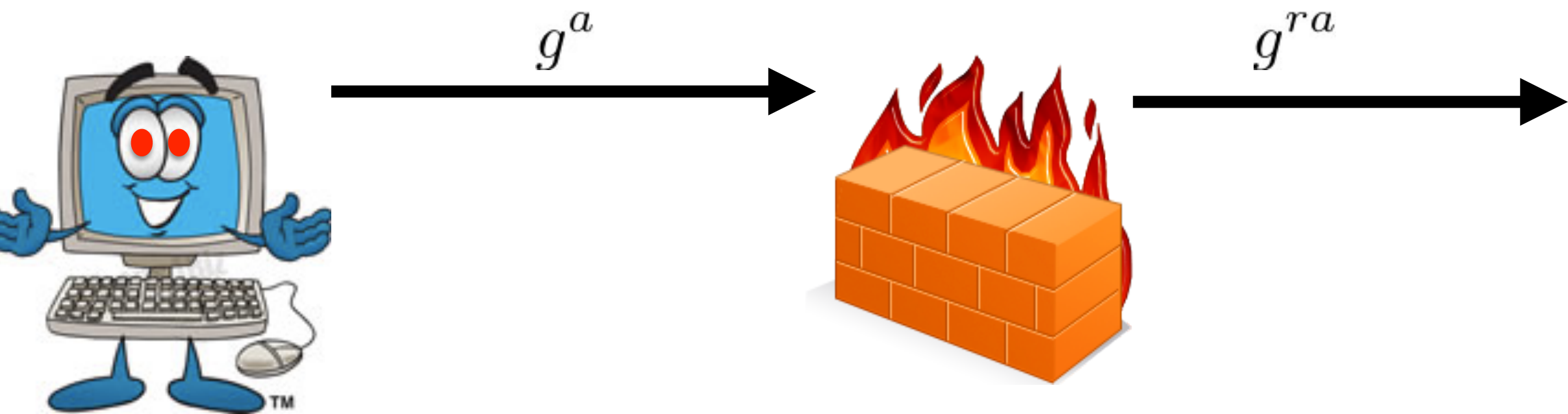
First Attempt

$G =$ a group in which DDH is hard with generator g



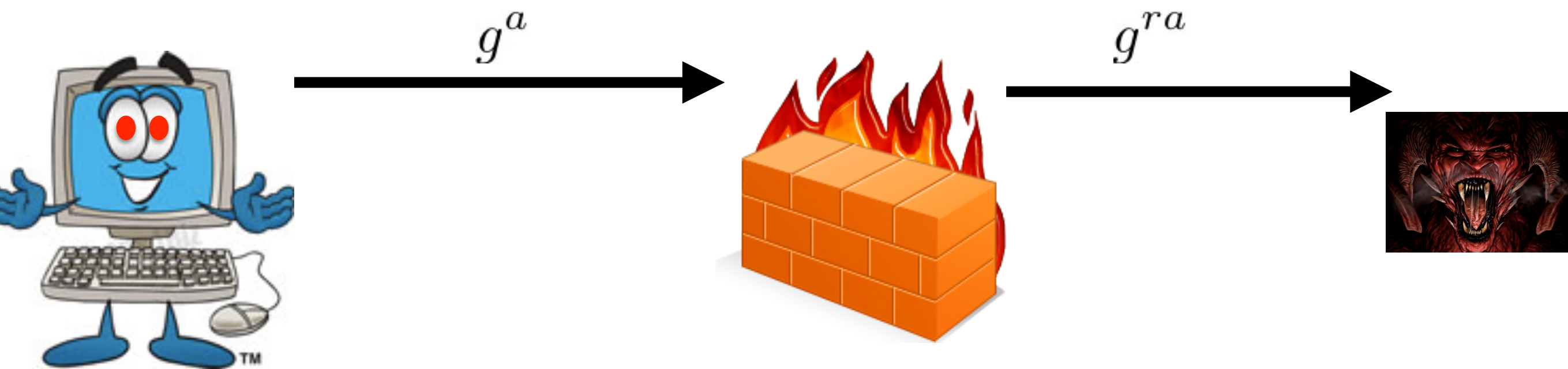
First Attempt

$G =$ a group in which DDH is hard with generator g



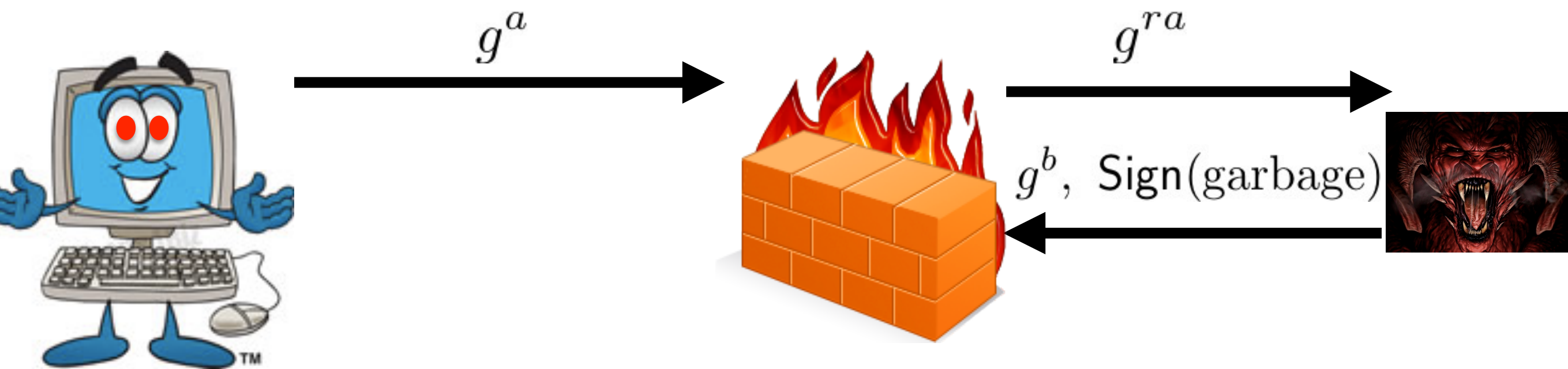
First Attempt

$G =$ a group in which DDH is hard with generator g



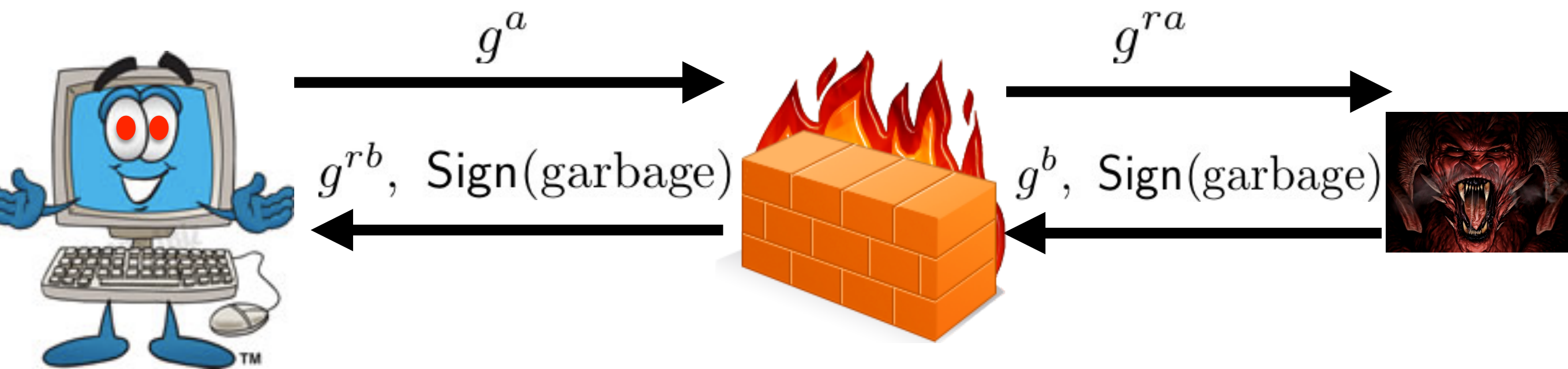
First Attempt

$G =$ a group in which DDH is hard with generator g



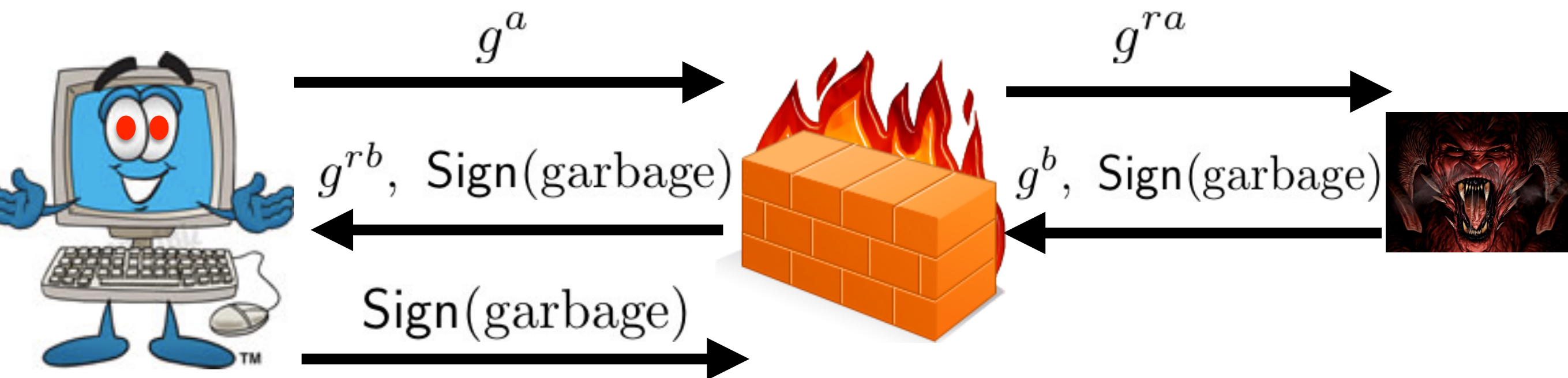
First Attempt

$G =$ a group in which DDH is hard with generator g



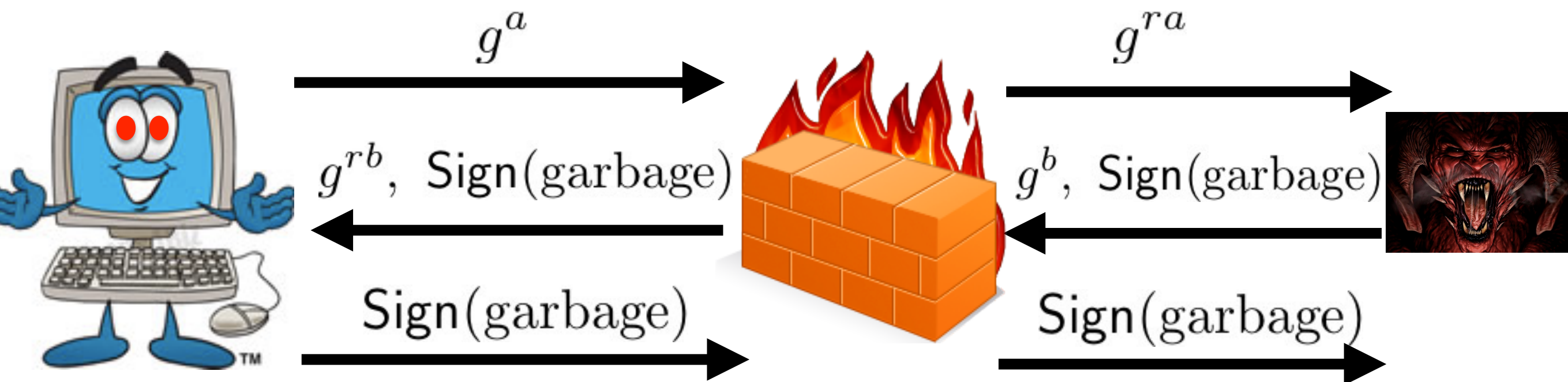
First Attempt

$G =$ a group in which DDH is hard with generator g



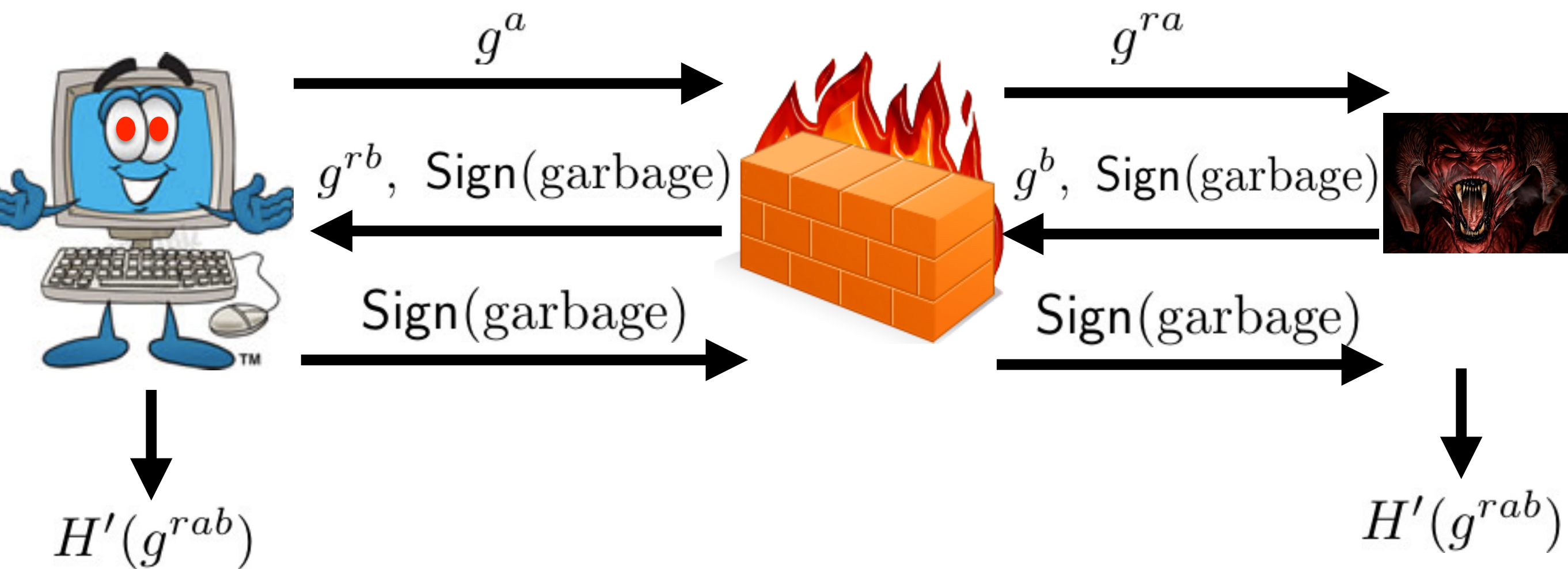
First Attempt

$G =$ a group in which DDH is hard with generator g



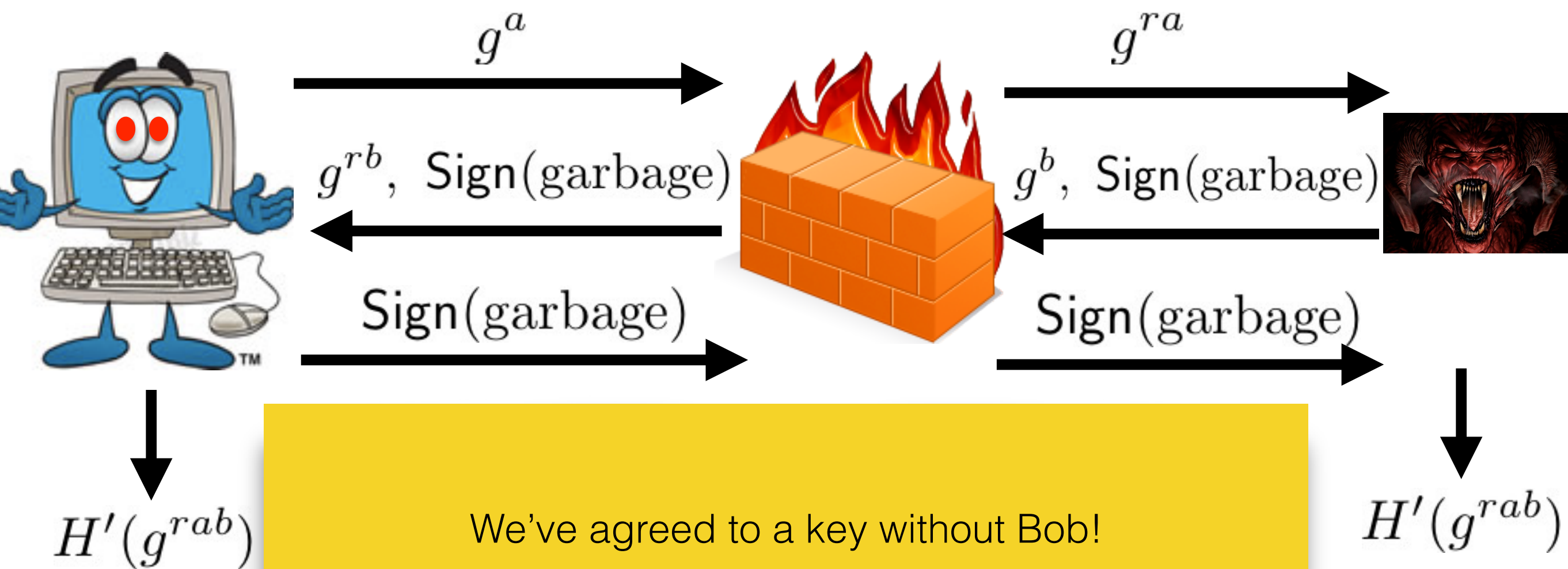
First Attempt

$G =$ a group in which DDH is hard with generator g



First Attempt

$G =$ a group in which DDH is hard with generator g



Smarter Solution

$G =$ a group in which DDH is hard with generator g

Smarter Solution

Smarter Solution

$G =$ a group in which CDH is hard with generator g

Smarter Solution

$G =$ a group in which CDH is hard with generator g
and bilinear map $e : G \times G \mapsto G_T$, such that $e(g^x, g^y) = e(g, g)^{xy}$

Smarter Solution

$G =$ a group in which CDH is hard with generator g
and bilinear map $e : G \times G \mapsto G_T$, such that $e(g^x, g^y) = e(g, g)^{xy}$



Smarter Solution

$G =$ a group in which CDH is hard with generator g
and bilinear map $e : G \times G \mapsto G_T$, such that $e(g^x, g^y) = e(g, g)^{xy}$



Smarter Solution

$G =$ a group in which CDH is hard with generator g and bilinear map $e : G \times G \mapsto G_T$, such that $e(g^x, g^y) = e(g, g)^{xy}$

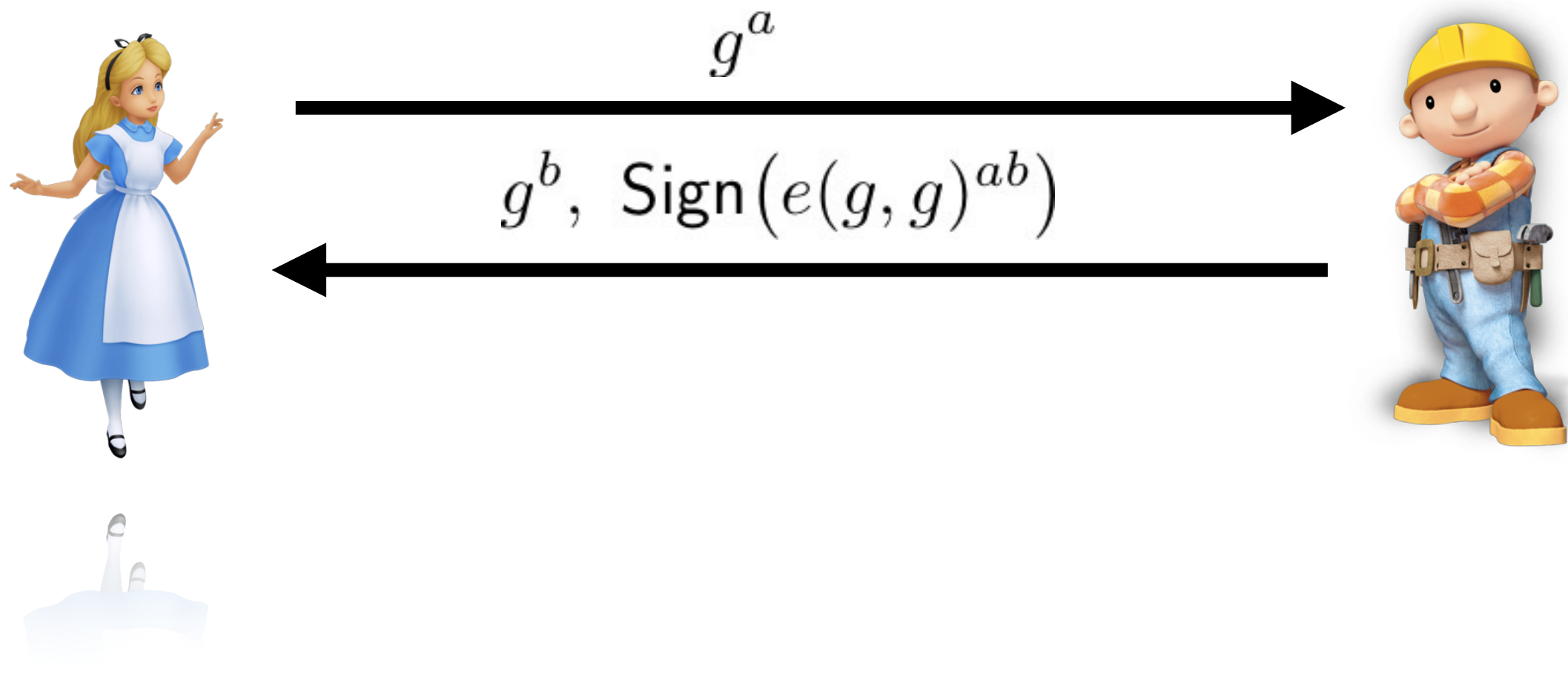


g^a



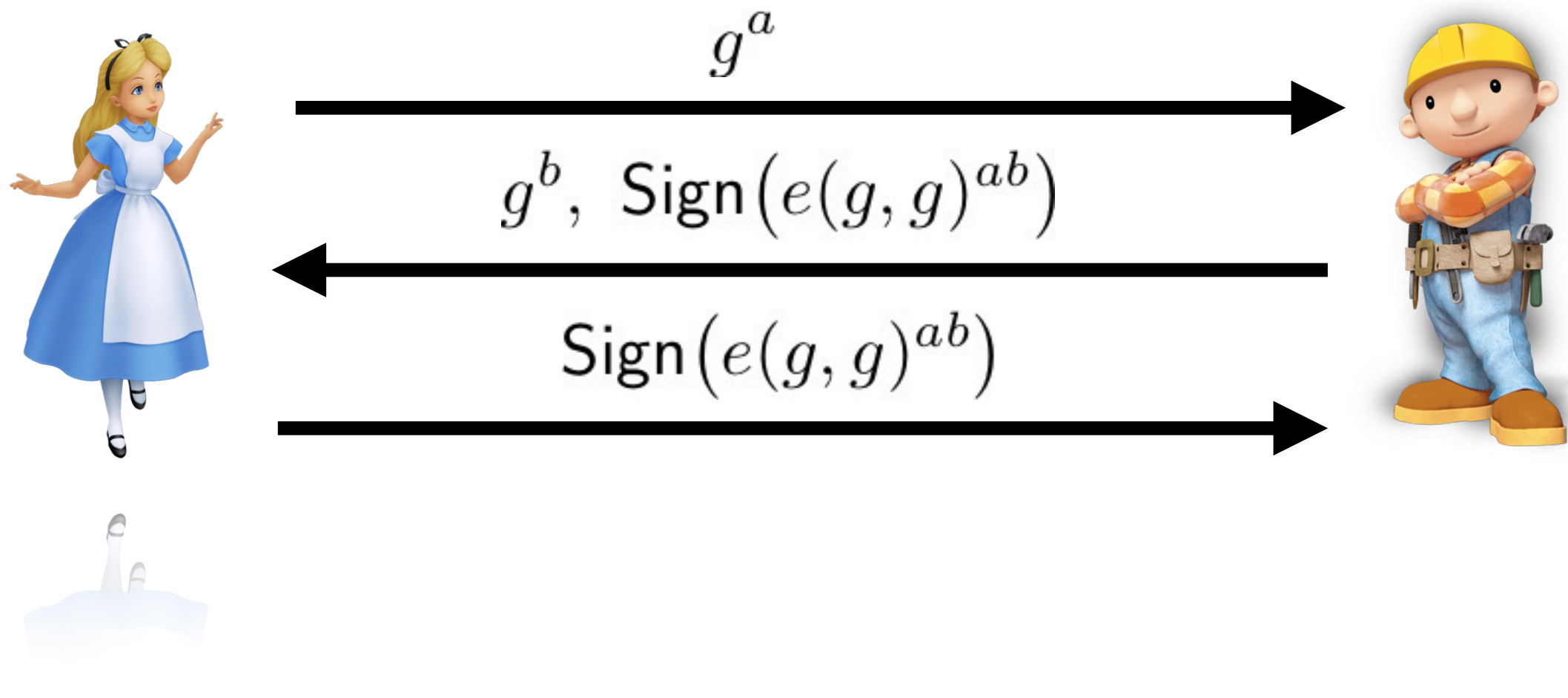
Smarter Solution

G = a group in which CDH is hard with generator g
and bilinear map $e : G \times G \mapsto G_T$, such that $e(g^x, g^y) = e(g, g)^{xy}$



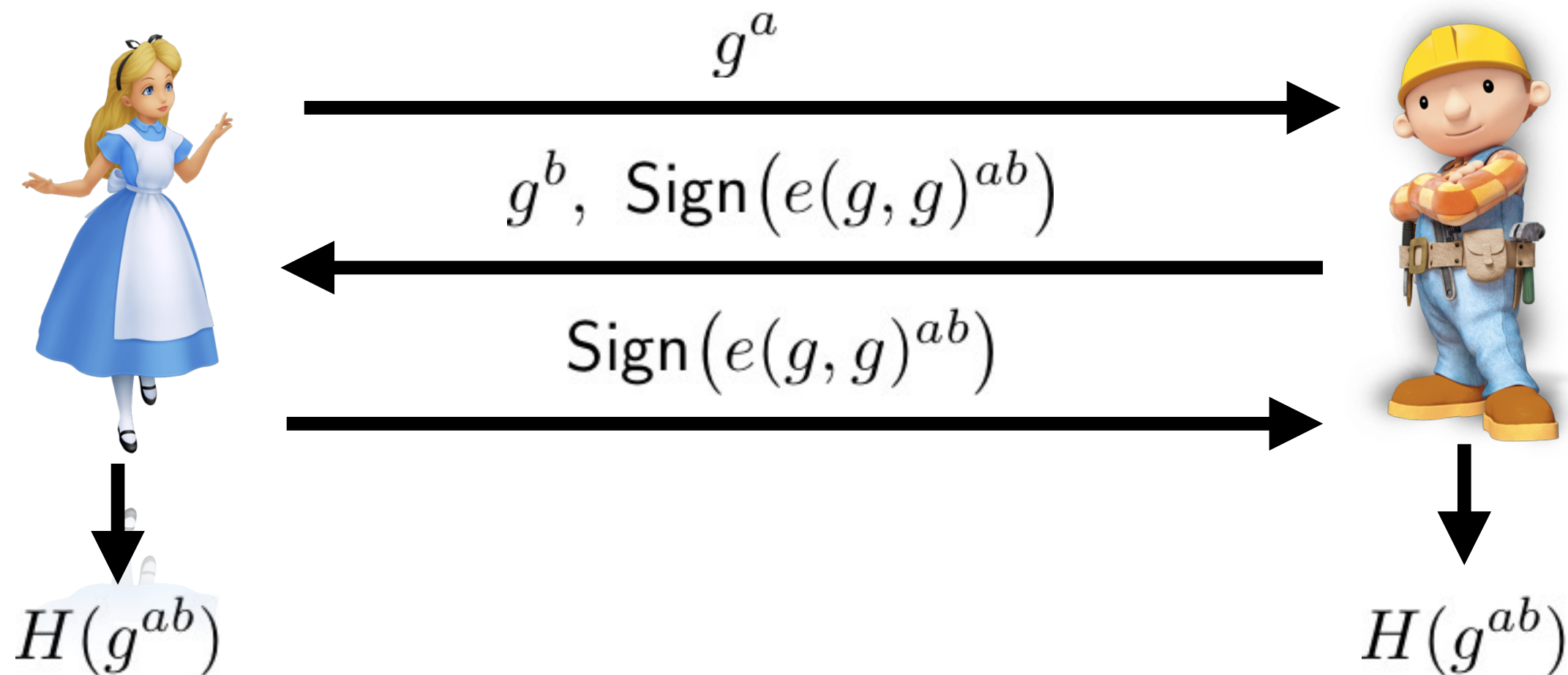
Smarter Solution

G = a group in which CDH is hard with generator g
and bilinear map $e : G \times G \mapsto G_T$, such that $e(g^x, g^y) = e(g, g)^{xy}$



Smarter Solution

G = a group in which CDH is hard with generator g and bilinear map $e : G \times G \mapsto G_T$, such that $e(g^x, g^y) = e(g, g)^{xy}$



Smarter Solution

$G =$ a group in which CDH is hard with generator g
and bilinear map $e : G \times G \mapsto G_T$, such that $e(g^x, g^y) = e(g, g)^{xy}$

Smarter Solution

$G =$ a group in which CDH is hard with generator g
and bilinear map $e : G \times G \mapsto G_T$, such that $e(g^x, g^y) = e(g, g)^{xy}$



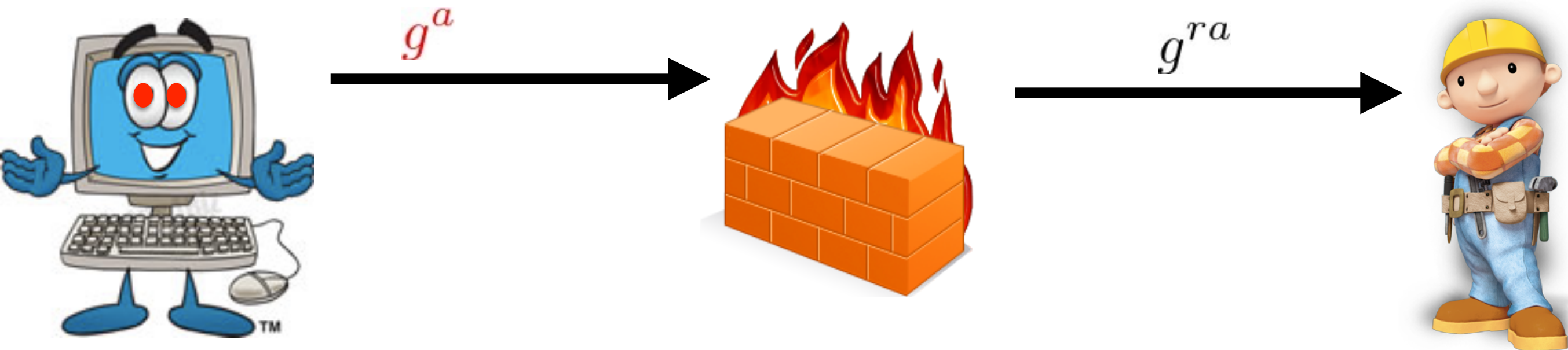
Smarter Solution

$G =$ a group in which CDH is hard with generator g and bilinear map $e : G \times G \mapsto G_T$, such that $e(g^x, g^y) = e(g, g)^{xy}$



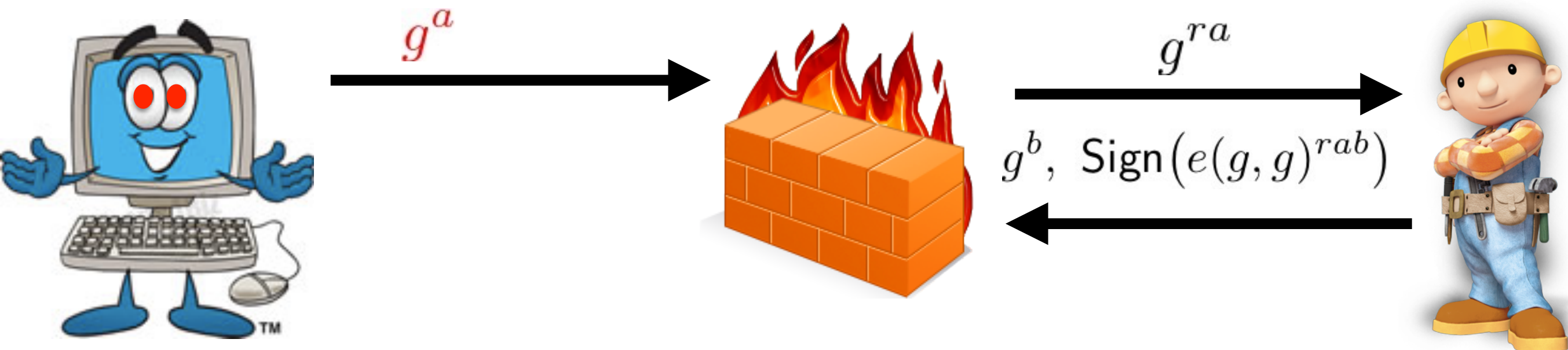
Smarter Solution

$G =$ a group in which CDH is hard with generator g and bilinear map $e : G \times G \mapsto G_T$, such that $e(g^x, g^y) = e(g, g)^{xy}$



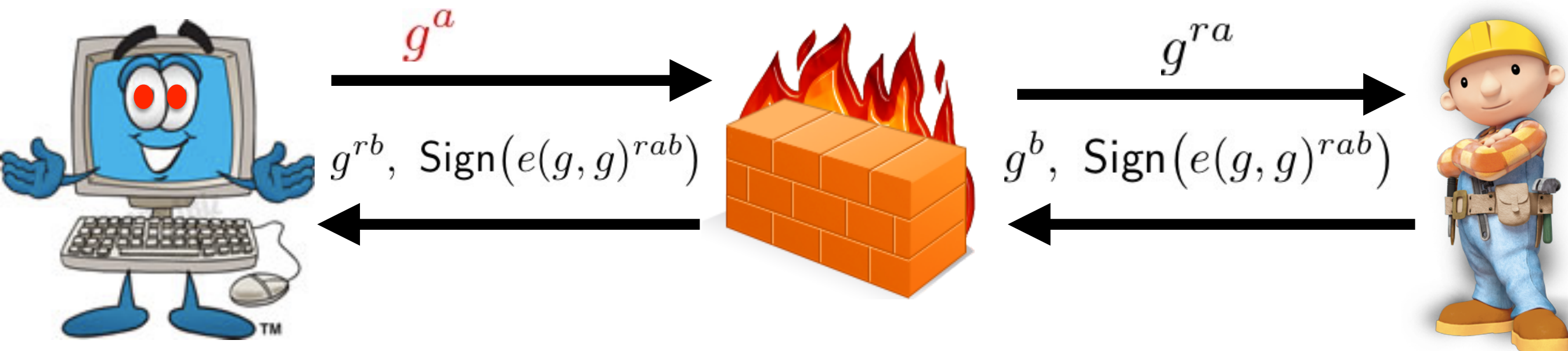
Smarter Solution

G = a group in which CDH is hard with generator g and bilinear map $e : G \times G \mapsto G_T$, such that $e(g^x, g^y) = e(g, g)^{xy}$



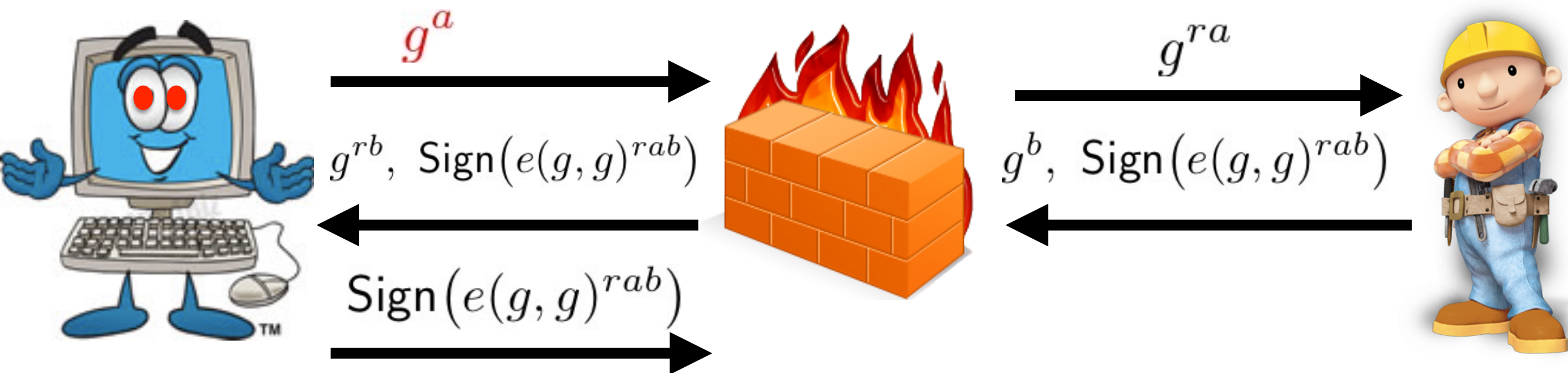
Smarter Solution

G = a group in which CDH is hard with generator g and bilinear map $e : G \times G \mapsto G_T$, such that $e(g^x, g^y) = e(g, g)^{xy}$



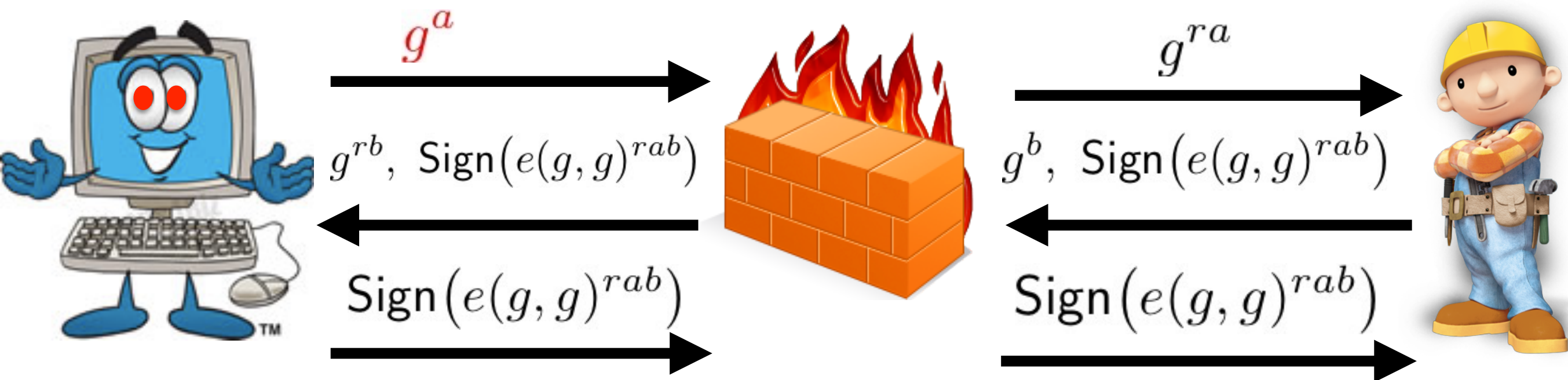
Smarter Solution

G = a group in which CDH is hard with generator g and bilinear map $e : G \times G \mapsto G_T$, such that $e(g^x, g^y) = e(g, g)^{xy}$



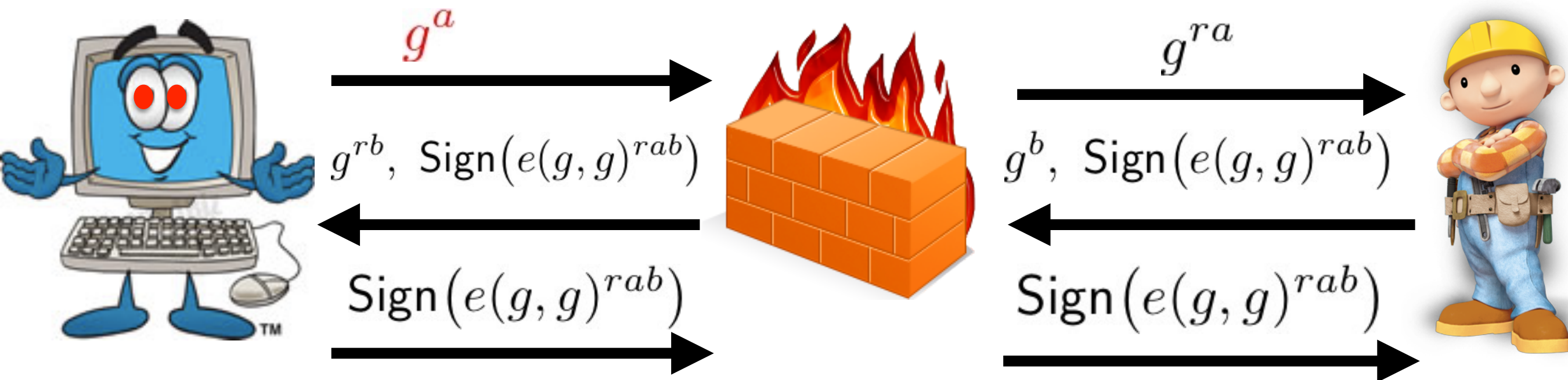
Smarter Solution

G = a group in which CDH is hard with generator g and bilinear map $e : G \times G \mapsto G_T$, such that $e(g^x, g^y) = e(g, g)^{xy}$



Smarter Solution



G = a group in which CDH is hard with generator g and bilinear map $e : G \times G \mapsto G_T$, such that $e(g^x, g^y) = e(g, g)^{xy}$





Note: Efficiency compares well with real-world protocols, such as TLS!

Additional Issues



Additional Issues

- Can a corrupt implementation  use the signature to communicate with the adversary  ?



Additional Issues

- Can a corrupt implementation  use the signature to communicate with the adversary  ?
 - (Use unique or rerandomizable signatures as in [AMV15].)



Additional Issues

- Can a corrupt implementation  use the signature to communicate with the adversary  ?
 - (Use unique or rerandomizable signatures as in [AMV15].)
- Can Bob have a firewall too?



Additional Issues

- Can a corrupt implementation  use the signature to communicate with the adversary  ?
 - (Use unique or rerandomizable signatures as in [AMV15].)
- Can Bob have a firewall too?
 - Yes!



Additional Issues

- Can a corrupt implementation  use the signature to communicate with the adversary  ?
 - (Use unique or rerandomizable signatures as in [AMV15].)
- Can Bob have a firewall too?
 - Yes!
 - (All results in [MS15] and [DMS15] have firewalls for both parties.)



Additional Issues

- Can a corrupt implementation  use the signature to communicate with the adversary  ?
 - (Use unique or rerandomizable signatures as in [AMV15].)
- Can Bob have a firewall too?
 - Yes!
 - (All results in [MS15] and [DMS15] have firewalls for both parties.)
- Can Alice or Bob leak secrets through the protocols.

Additional Issues

- Can a corrupt implementation  use the signature to communicate with the adversary  ?
 - (Use unique or rerandomizable signatures as in [AMV15].)
- Can Bob have a firewall too?
 - Yes!
 - (All results in [MS15] and [DMS15] have firewalls for both parties.)
- Can Alice or Bob leak secrets through the protocols.
 - No!

Additional Issues

- Can a corrupt implementation  use the signature to communicate with the adversary  ?
 - (Use unique or rerandomizable signatures as in [AMV15].)
- Can Bob have a firewall too?
 - Yes!
 - (All results in [MS15] and [DMS15] have firewalls for both parties.)
- Can Alice or Bob leak secrets through the protocols.
 - No!
 - (We formalize this in the papers.)

Summary

Summary

- Lots of recent news suggests that we can no longer trust our computers when security is paramount.

Summary

- Lots of recent news suggests that we can no longer trust our computers when security is paramount.
- Reverse firewalls provide a framework to guarantee security of arbitrary cryptographic primitives *even on a compromised machine*.

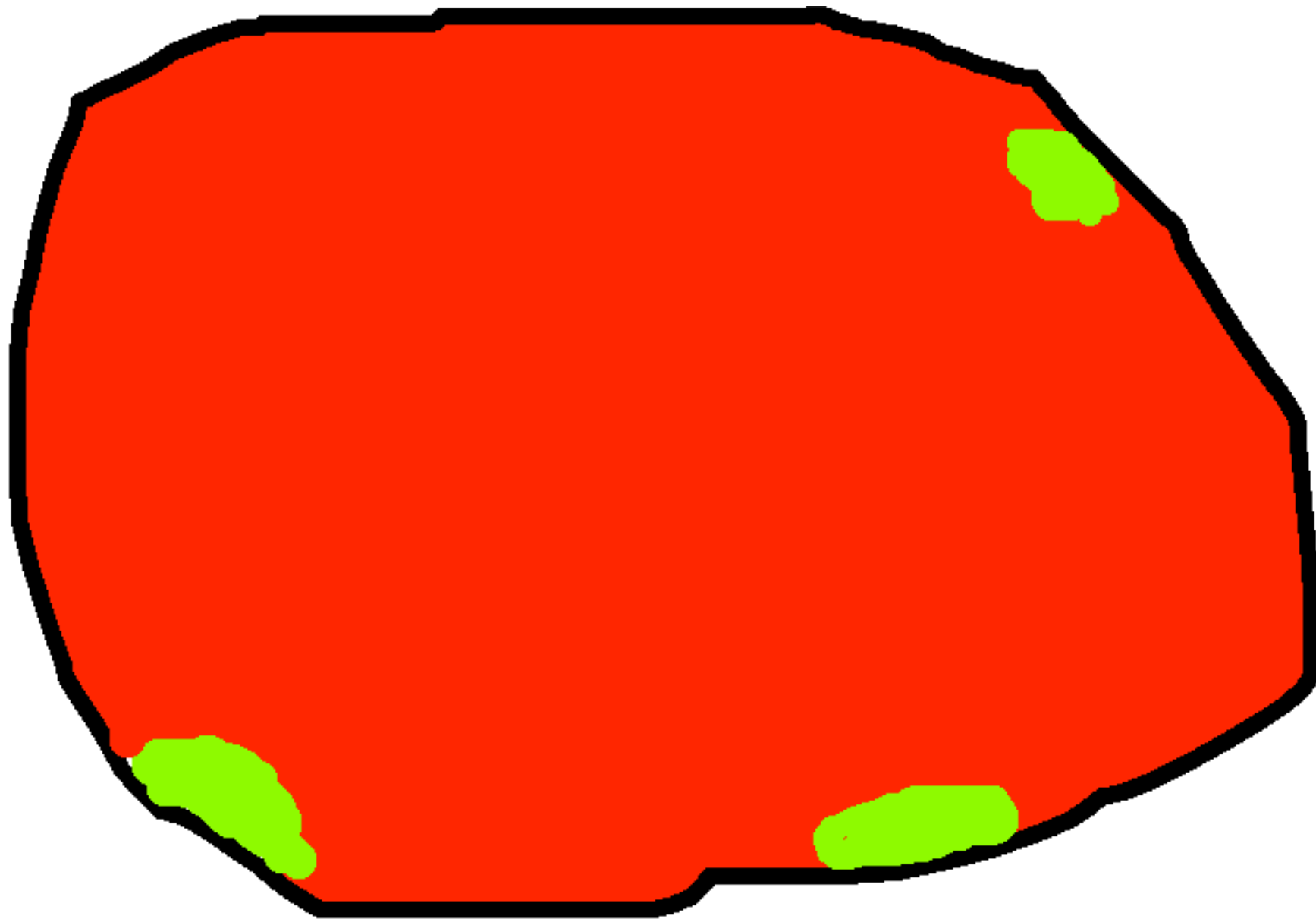
Summary

- Lots of recent news suggests that we can no longer trust our computers when security is paramount.
- Reverse firewalls provide a framework to guarantee security of arbitrary cryptographic primitives *even on a compromised machine*.
- Very strong cryptographic primitives can be instantiated in this model.

Summary

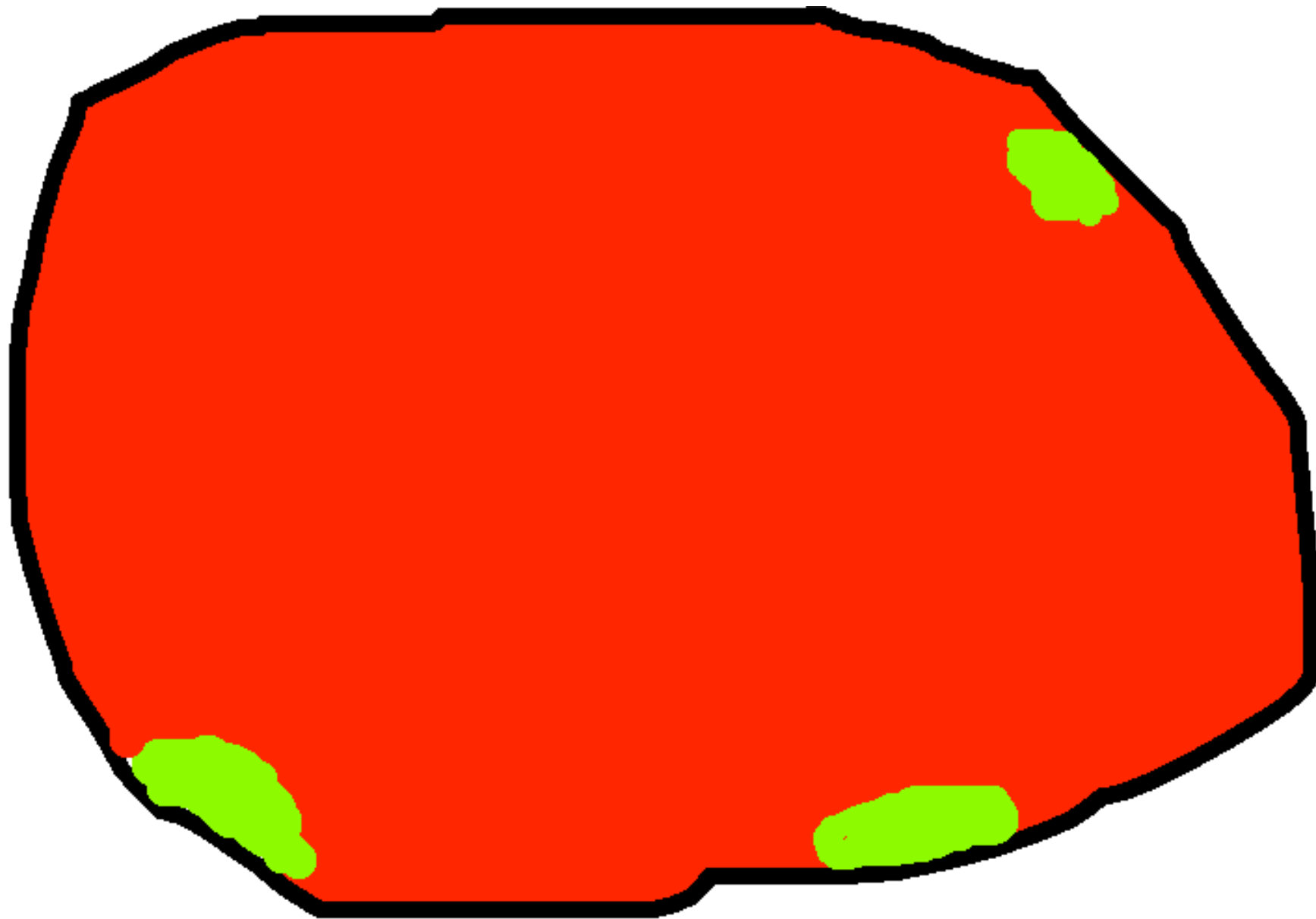
- Lots of recent news suggests that we can no longer trust our computers when security is paramount.
- Reverse firewalls provide a framework to guarantee security of arbitrary cryptographic primitives *even on a compromised machine*.
- Very strong cryptographic primitives can be instantiated in this model.
- The (arguably) most important cryptographic primitive, MTP, can be instantiated *efficiently* in this model.

A Lot of Work to Be Done!



A Lot of Work to Be Done!

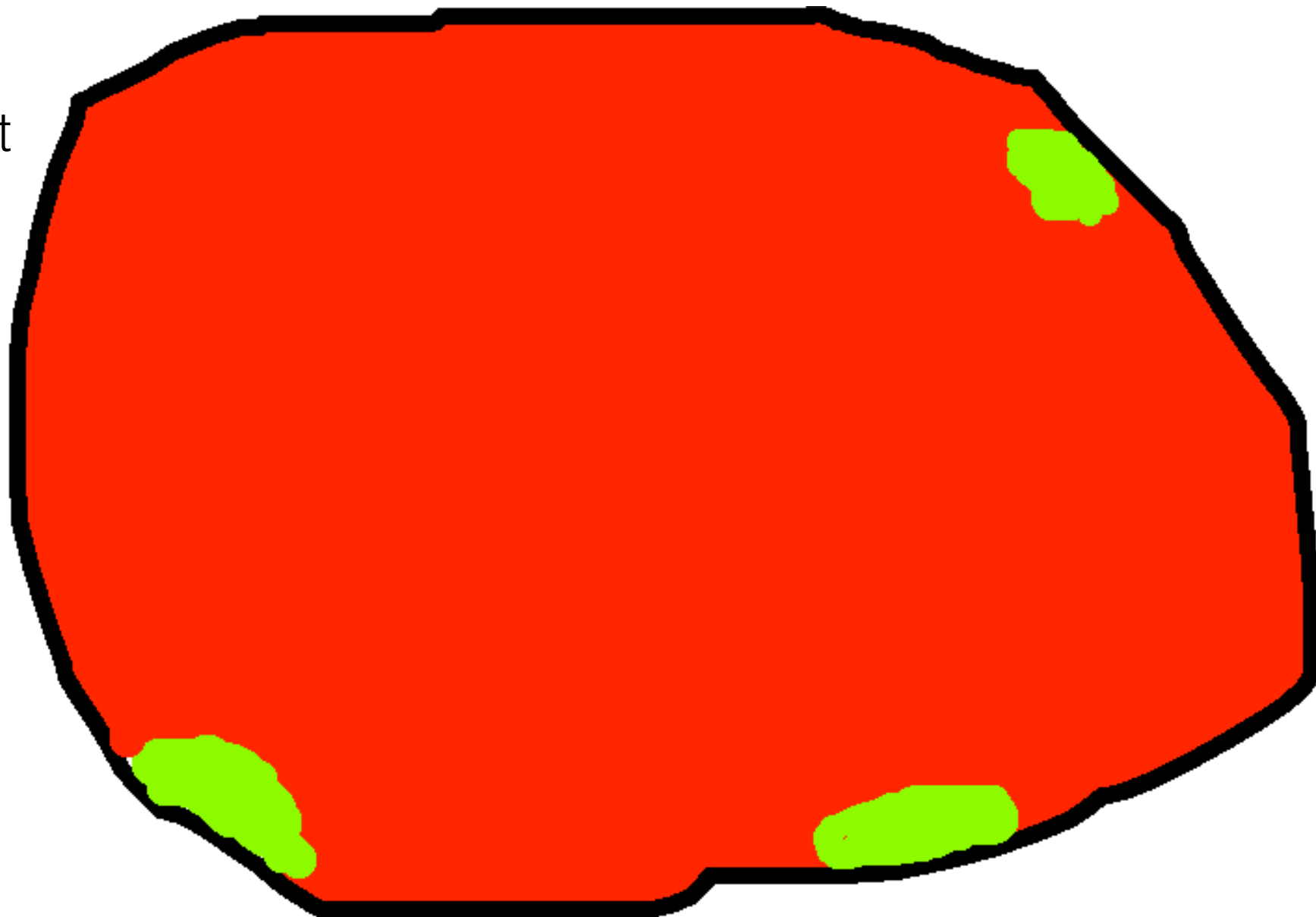
The space of all known cryptographic primitives



A Lot of Work to Be Done!



The space of all known cryptographic primitives

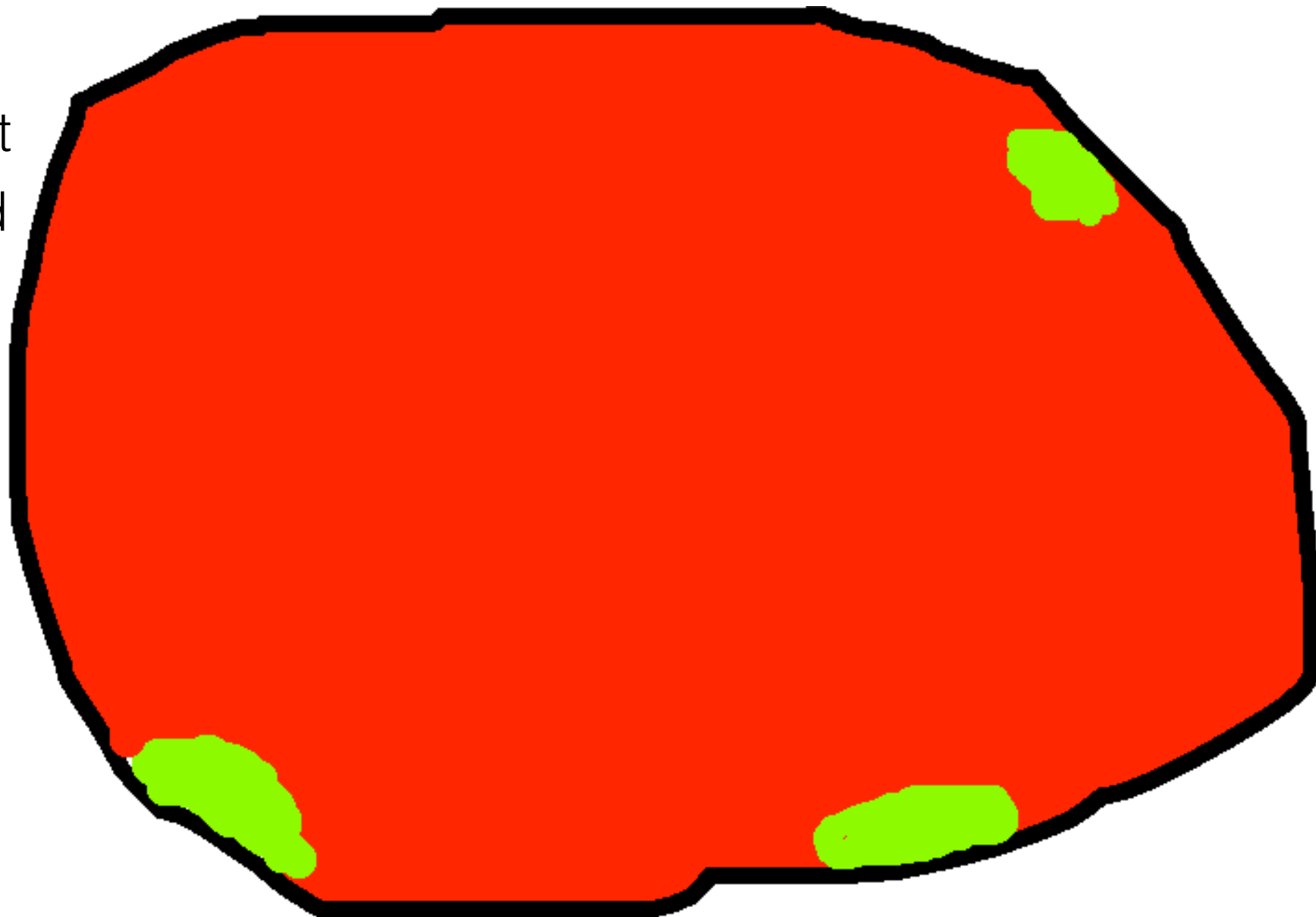
 Studied in
this context



A Lot of Work to Be Done!



The space of all known cryptographic primitives

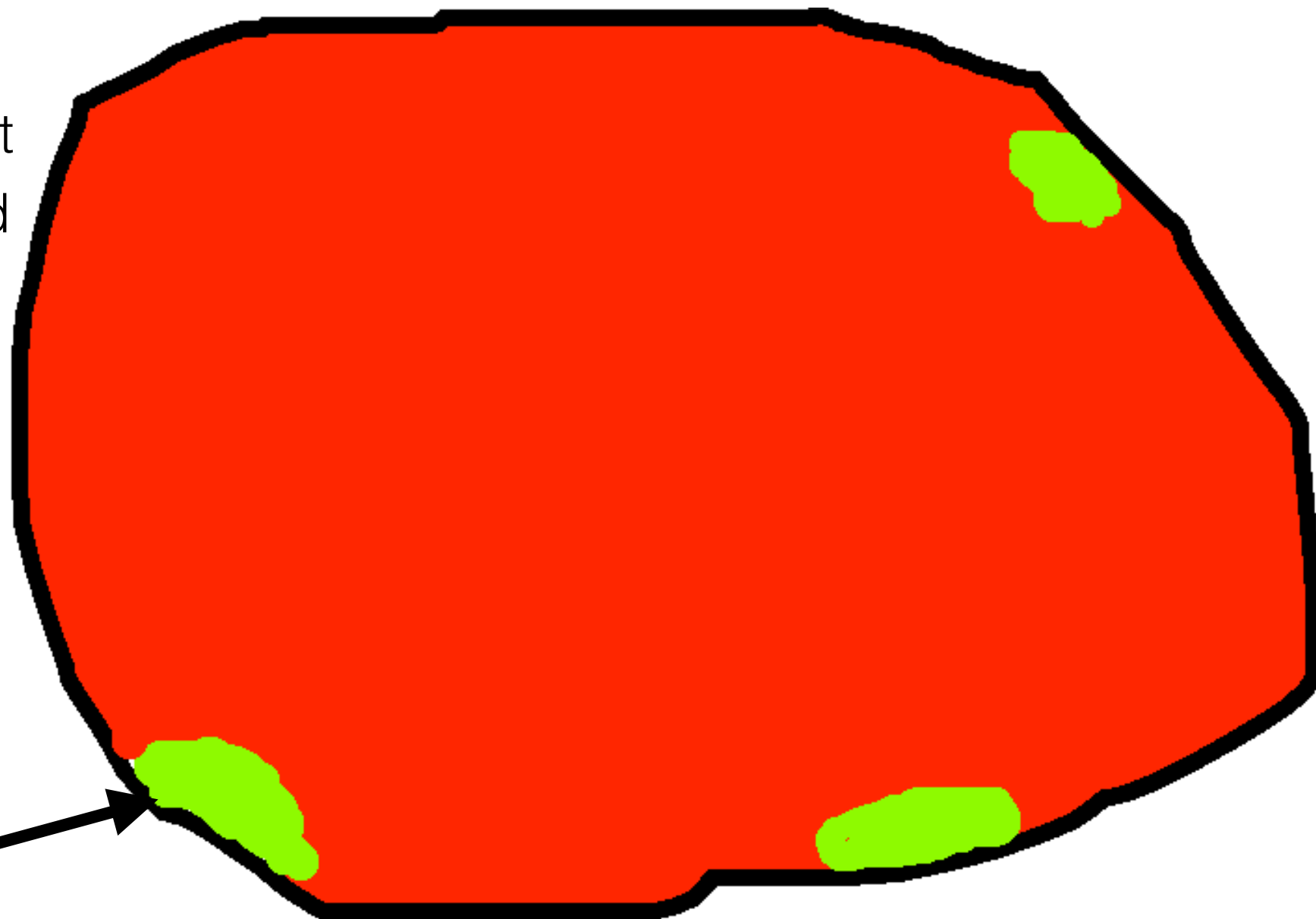
-  Studied in this context
-  Not studied



A Lot of Work to Be Done!

The space of all known cryptographic primitives



 Studied in this context
 Not studied

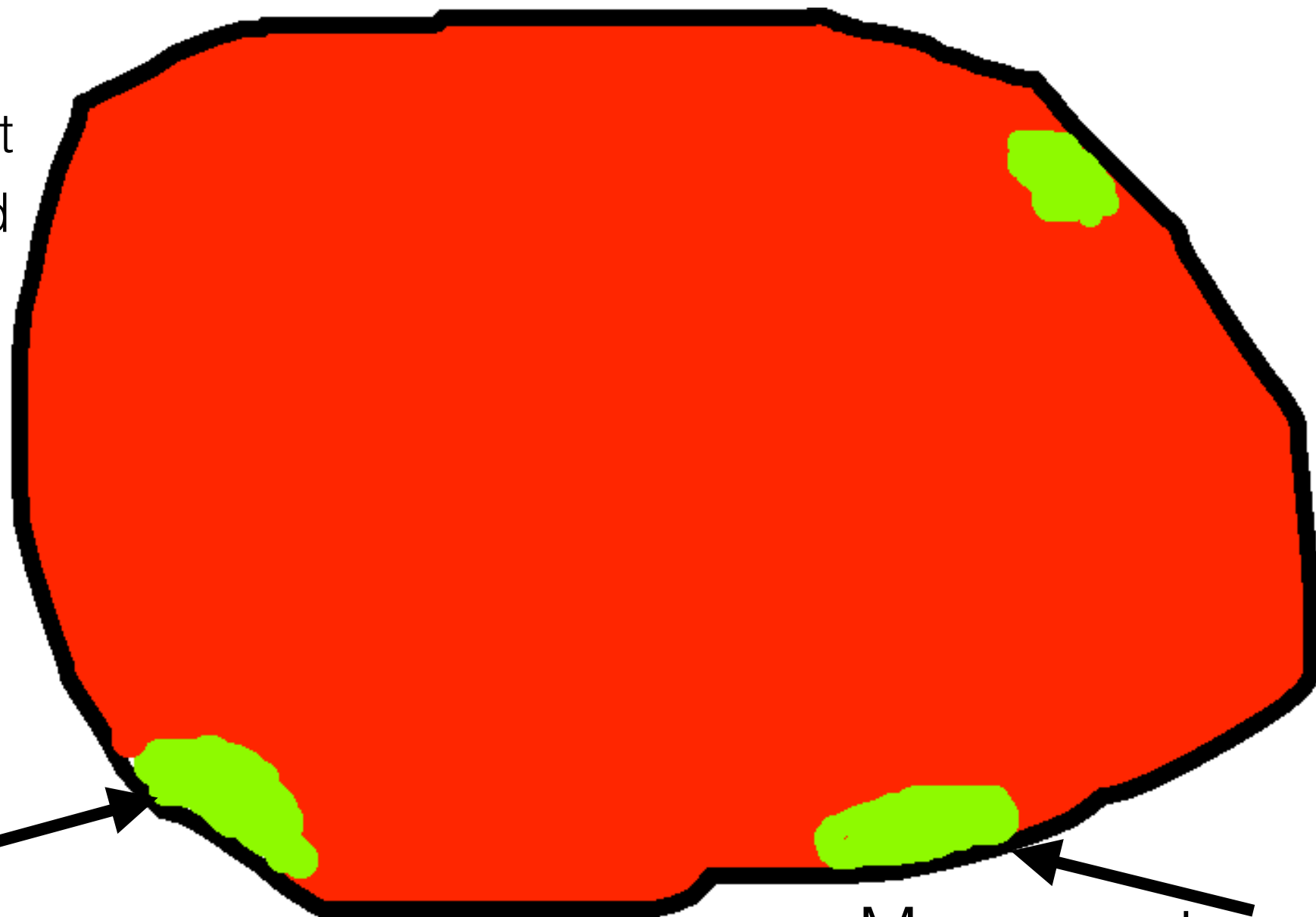


OT, SFE
[MS15]

A Lot of Work to Be Done!

The space of all known cryptographic primitives

 Studied in this context
 Not studied

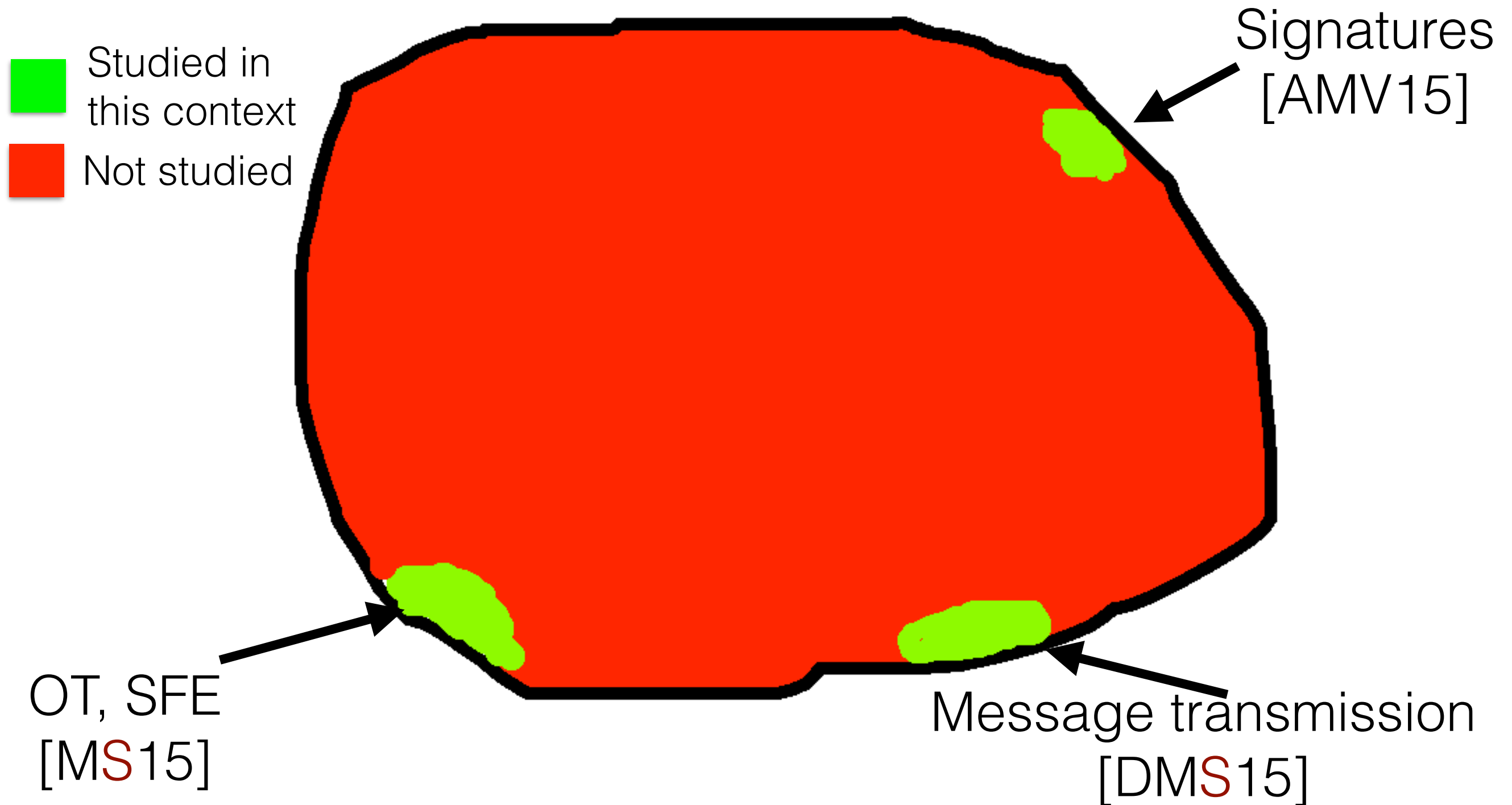


OT, SFE
[MS15]

Message transmission
[DMS15]

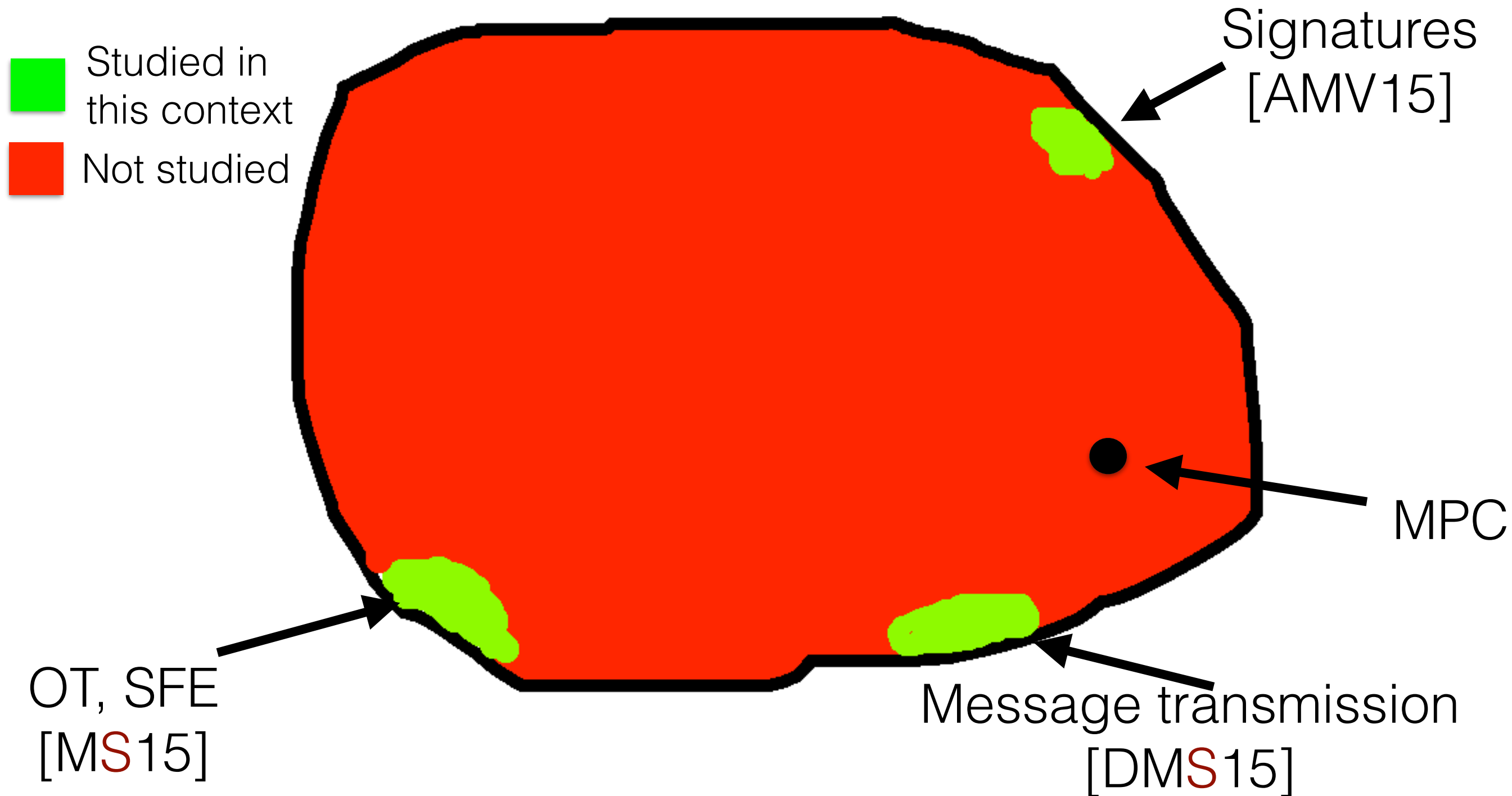
A Lot of Work to Be Done!

The space of all known cryptographic primitives



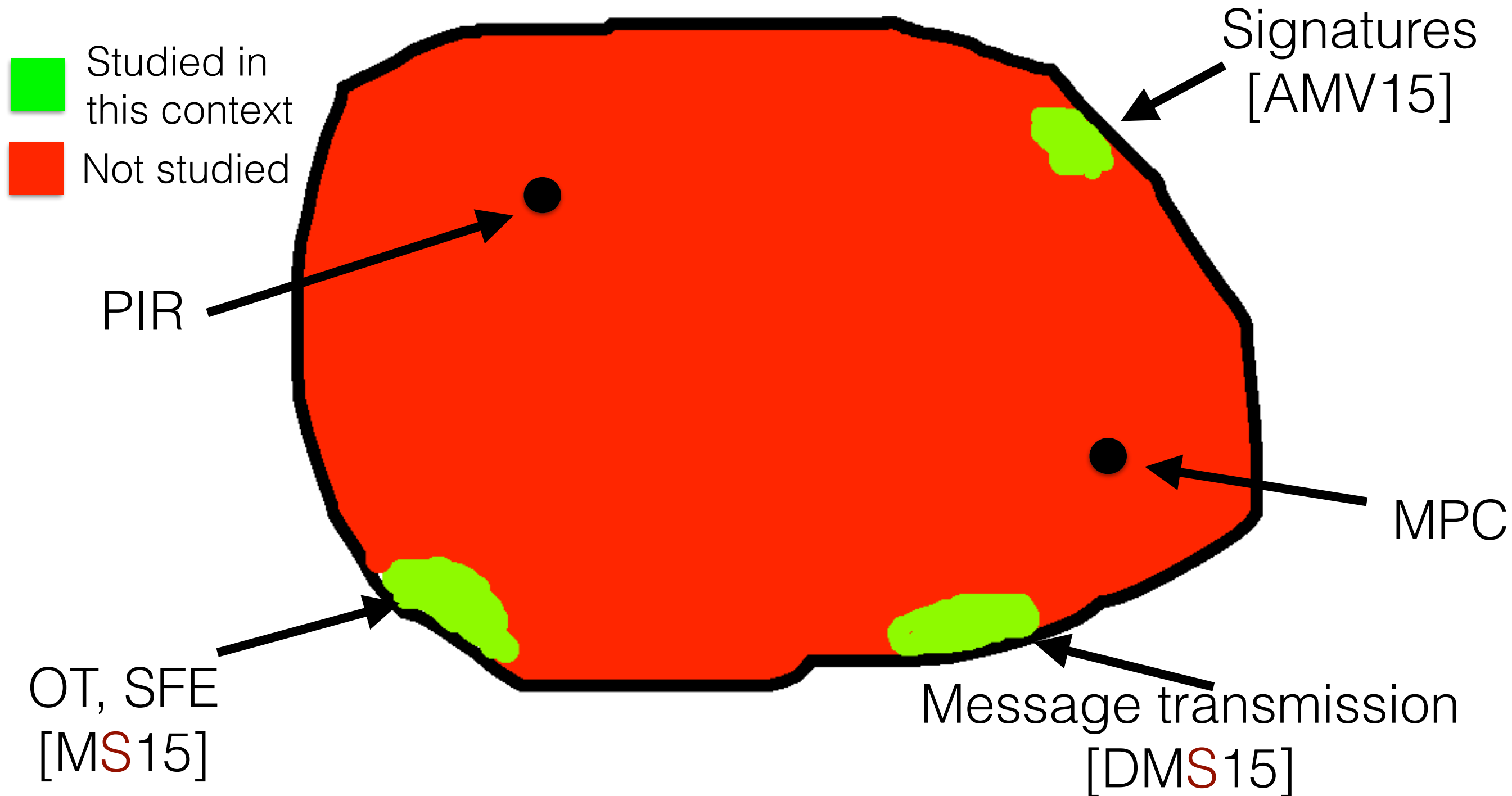
A Lot of Work to Be Done!

The space of all known cryptographic primitives



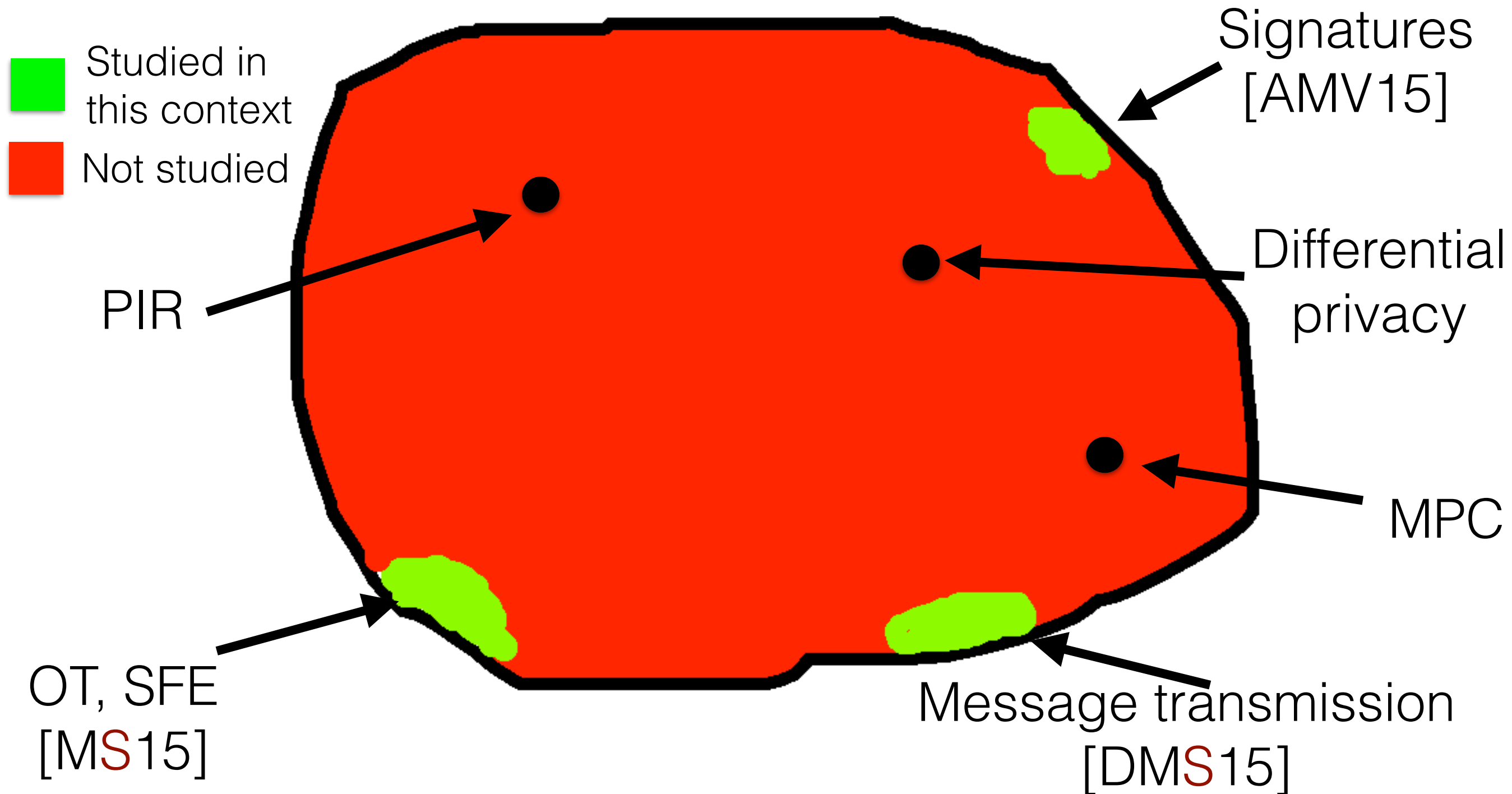
A Lot of Work to Be Done!

The space of all known cryptographic primitives



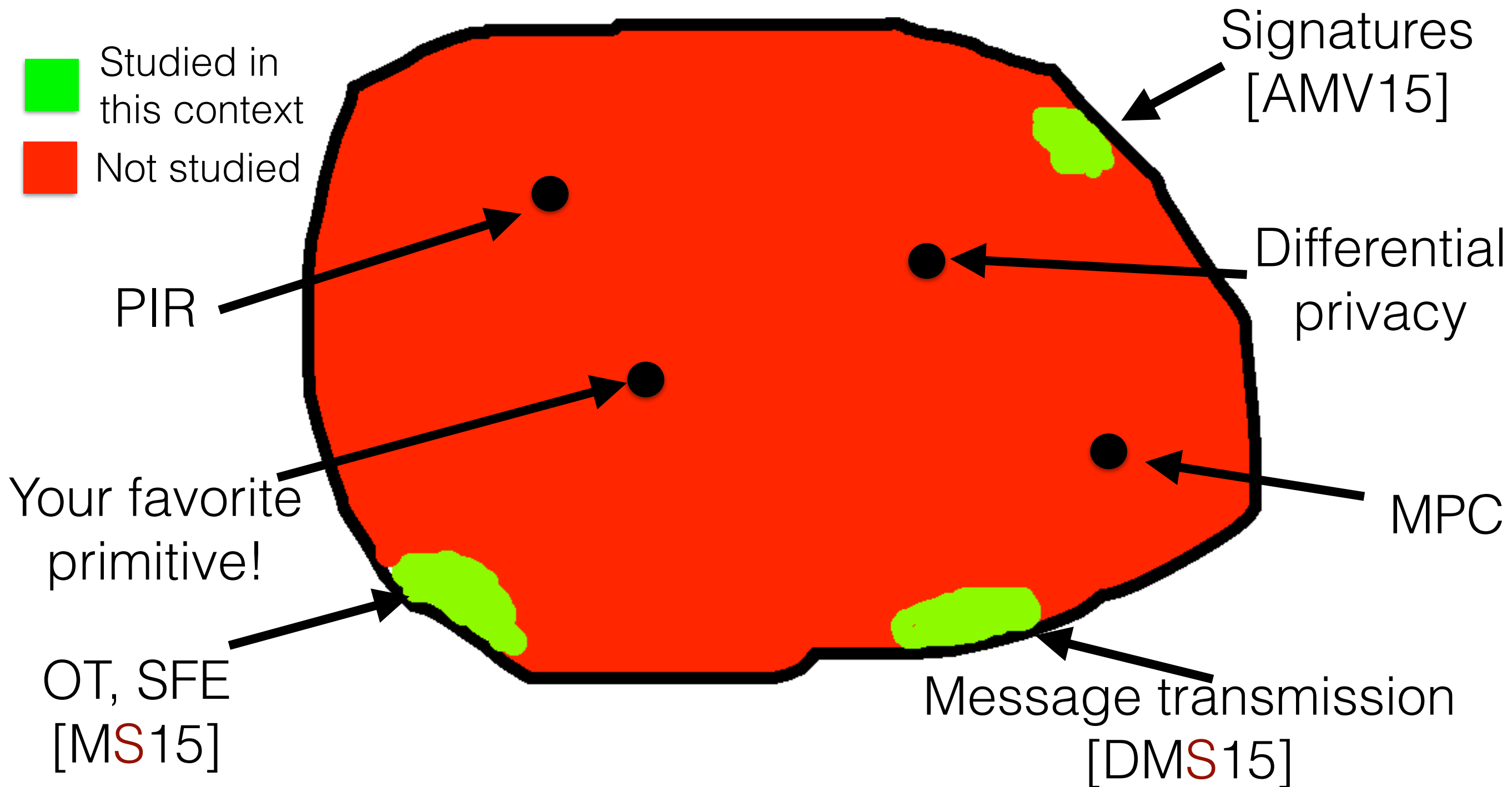
A Lot of Work to Be Done!

The space of all known cryptographic primitives



A Lot of Work to Be Done!

The space of all known cryptographic primitives



Thanks!

